

AD-A178 681

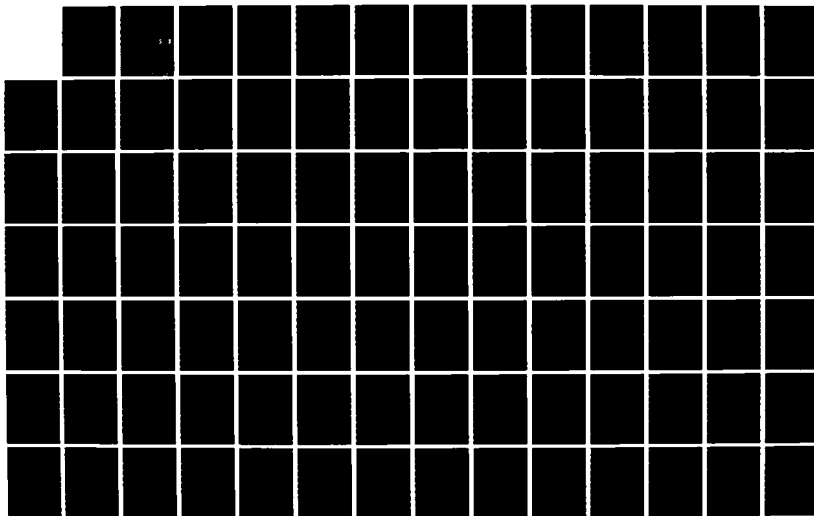
AGGREGATION OF NETWORK FLOW PROBLEMS(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH V E FRANCIS 1985
AFIT/CI/NR-86-800

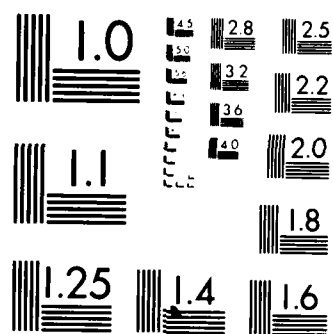
1/4

UNCLASSIFIED

F/G 12/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 86- 80D	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Aggregation of Network Flow Problems		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Vernon Edward Francis		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: University of California		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433-6583		12. REPORT DATE 1985
		13. NUMBER OF PAGES 295
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1		DTIC ELECTE AUG 12 1986 <i>Wolaver</i> LYNN E. WOLAVER 6 AUG 86 Dean for Research and Professional Development AFIT/NR
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED.		

AD-A170 681

DTIC FILE COPY

ABSTRACT OF THE DISSERTATION

Aggregation Of Network Flow Problems

by

Vernon Edward Francis

Doctor of Philosophy in Management

University of California, Los Angeles, 1985

Professor Arthur Geoffrion, Chair

Aggregation is a fundamental and widely used method of model simplification in management science, operations management, economics, database systems and related fields. It has also been used as a technical device in algorithm and software design for large scale optimization problems, particularly network flow problems. This class of problems is particularly amenable to aggregation owing to the special structural properties of networks.

In this research, we investigate aggregation of the capacitated transshipment problem. We establish a new framework for aggregation of this class of problems and provide an important characterization of arc types as a necessary preliminary to implementation. An aggregation procedure and a partition refinement process which generates

a sequence of successively restricted aggregate problems are developed within this framework.

A disaggregation methodology is defined that maps a feasible basic solution of the current aggregate problem to a basic solution of the next aggregate problem in the sequence. This map is incorporated into a complete algorithm which employs the aggregation-disaggregation concepts developed here.

Finally, the results of this research are verified and tested in a computer implementation.

UNIVERSITY OF CALIFORNIA

Los Angeles

Aggregation Of Network Flow Problems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Management

by

Vernon Edward Francis

1985

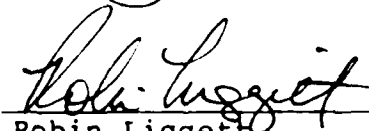


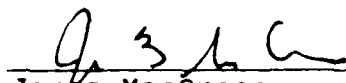
✓
A-1

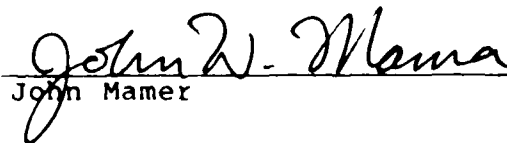
© Copyright by
Vernon Edward Francis
1986

The dissertation of Vernon Edward Francis is approved.


Stephen Jacobsen


Robin Liggett


James MacQueen


John Mamer


Arthur Geoffrion, Committee Chair

University of California, Los Angeles

1985

DEDICATION

This work is dedicated to my wife, Laura, for her constant encouragement, understanding, unwavering support, and unselfish sacrifices.

CONTENTS

	<u>Page</u>
LIST OF FIGURES AND TABLES	vi
ACKNOWLEDGEMENTS	vii
VITA	viii
ABSTRACT	ix
CHAPTER ONE: INTRODUCTION	1
I. Taxonomy For Aggregation Research	3
II. Literature Review	6
CHAPTER TWO: AGGREGATION OF A NETWORK FLOW PROBLEM. . .	43
I. Preliminary Notation.	43
II. Aggregation Of The Capacitated Transshipment Problem	46
III. A Measure Of The Scale Of Aggregation	61
IV. Use Of M^r In Aggregation Design	66
CHAPTER THREE: SEQUENCES OF AGGREGATE PROBLEMS.	71
I. Further Characterization Of Original Arcs	72
II. Procedure To Create $\overline{NP}^{r+1}$ From $\overline{NP}^r$	77
CHAPTER FOUR: THE GENERAL DISAGGREGATION MAP DM^G	93
I. Definitions, Notation And Terminology	94
II. The Logic Behind The Disaggregation Map DM^G	96
III. Definition Of DM^G	101
IV. DM^G Applied To The Partitioned Adjacency Matrix.	110

V. Some Characteristics Of DM^G	118
CHAPTER FIVE: PARTITION REFINEMENT PROCESS.	127
I. Rationale For The Partition Refinement Process	128
II. Finding A Simple Refinement That Yields A Small Subproblem.	140
CHAPTER SIX: THE COMPLETE ALGORITHM AND ITS IMPLEMENTATION	157
I. Basic Logic Of The Complete Algorithm	158
II. An Example Of The Complete Algorithm.	161
III. Implementation.	164
IV. Testing And Verification.	178
CHAPTER SEVEN: SUMMARY AND CONCLUSIONS.	180
APPENDIX A: GEOFFRION'S MODELING FRAMEWORK AND AGGREGATION THEORY.	183
APPENDIX B: LEE'S AGGREGATION FRAMEWORK AND NETWORK ALGORITHM	205
APPENDIX C: GNET DESIGN AND IMPLEMENTATION.	223
APPENDIX D: THE AGNET COMPUTER CODE	238
APPENDIX E: AGNET COMPUTER OUTPUT FOR EXAMPLE 2.1	266
BIBLIOGRAPHY	291

FIGURES AND TABLES

	<u>Page</u>
TABLE 1.1: SUMMARY AND CLASSIFICATION OF PREVIOUS STUDIES.	40
FIGURE 2.1: ELEMENT GRAPH FOR MODEL SCHEMA OF NP^0 . . .	49
TABLE 2.1: EXAMPLE PROBLEM FROM BRADLEY, ET. AL. [2] .	56
FIGURE 2.2: NP^0 AND $\overline{NP}^1$ FOR EXAMPLE 2.1	60
TABLE 2.2: ENUMERATION OF PARTITIONS FOR EXAMPLE 2.3 .	70
FIGURE 3.1: $\overline{NP}^1$ AND $\overline{NP}^2$ FOR EXAMPLE 3.2	87
FIGURE 4.1: REFINEMENT OF A GENERIC ADJACENCY MATRIX. .	97
FIGURE 4.2: DM^G APPLIED TO THE BASIS TREE OF EXAMPLE 4.2	111
FIGURE 4.3: DISAGGREGATED SOLUTION FOR EXAMPLE 4.4. . .	117
FIGURE 4.4: BASIS TREE FOR EXAMPLE 4.5.	119
FIGURE 6.1: LOGIC DIAGRAM FOR THE COMPLETE ALGORITHM. .	159
TABLE 6.1: SUMMARY OF EXAMPLE 6.1.	165
FIGURE 6.2: M^r VERSUS r FOR EXAMPLE 6.1.	166
FIGURE A-1: TRANSPORTATION MODEL EXAMPLE.	197
FIGURE A-2: MODEL SCHEMA FOR TRANSPORTATION MODEL . . .	198
FIGURE A-3: ELEMENT GRAPH FOR TRANSPORTATION MODEL. . .	199
FIGURE A-4: GENUS GRAPH FOR THE TRANSPORTATION MODEL. .	200
FIGURE A-5: ELEMENT GRAPH FOR AGGREGATE TRANSPORTATION MODEL	201
FIGURE A-6: AGGREGATE TRANSPORTATION MODEL.	203
FIGURE B-1: MODEL SCHEMA FOR (P) ADOPTED BY LEE	212
FIGURE C-1: PREDECESSOR GRAPH BEFORE AND AFTER PIVOT. .	234
TABLE C-1: ARRAYS BEFORE AND AFTER AN EXAMPLE PIVOT. .	236
FIGURE C-2: PIVOT SEGMENT LOGIC	237

ACKNOWLEDGEMENTS

I wish to thank my committee, especially Professor Arthur Geoffrion for their help and guidance in this research, my colleagues at the Department Of Management, United States Air Force Academy, especially Colonel Jim Woody, Lieutenant Colonel Bob Pizzi and Lieutenant Colonel Chuck Yoos, for supporting and encouraging me in this research effort, Professor Glen Graves for providing me with a copy of the GNET computer code and Mrs. Mary Lent and Mrs. Peggy Raymond for their typographical assistance.

VITA

October 26, 1949--Born, Baltimore, Maryland

1971--B.S. Civil Engineering, United States Air Force
Academy

1971-1976--ICBM Missile Crew Commander, USAF

1976--M.B.A., Univeristy Of Colorado, Boulder, Colorado

1977-1980--Instructor, Assistant Professor of Managment,
United States Air Force Academy

1983-1985--Assistant Professor of Managment, United States
Air Force Academy

1985--Deputy Department Head For Operations Research,
Department Of Management, United States Air Force Academy

CHAPTER ONE

INTRODUCTION

Aggregation is a fundamental and widely used method for simplifying models in management science, economics, data base systems, production and operations management and related fields. Aggregation is an important tool that:

1. simplifies data requirements for and solution of models
2. provides a method of condensing and simplifying models for presentation to higher levels of management
3. provides a technical device for algorithm and software design for large scale problems
4. provides a method for model simplification for diagnostic testing and yields insight into the structure of a problem and its solution.

Heretofore, research concerning model aggregation has proceeded in a rather piecemeal fashion and has been lacking in general principles, procedures and axiomatic foundation. Recently however, Geoffrion [19] has proposed a general modeling framework and a unifying theory of aggregation, hereafter referred to simply as "the modeling framework" and "the aggregation theory." This work and the discussion of previous research will be in terms of this modeling framework and aggregation theory.

This research will address aggregation as applied to a selected class of network flow problems. Specifically, the capacitated transshipment problem (CTP) will be considered. Methods of aggregation, disaggregation and algorithm development are of primary interest. Aggregation and disaggregation methods are developed for this problem class which map basic feasible (optimal) solutions to basic solutions. This important characteristic, which is an original contribution of this research, facilitates the development of aggregation-disaggregation algorithms that construct advanced-start basic solutions for successively refined surrogate problems. An algorithm employing these ideas is developed.

In this chapter we provide a taxonomy for aggregation research based upon Geoffrion's modeling framework and aggregation theory. We review and summarize the previous research bearing upon this work in terms of this taxonomy.

Chapter two rigorously defines the aggregation of a (CTP). An aggregation procedure is developed and illustrated using both the network diagram of the (CTP) and its associated adjacency matrix. This aggregation procedure requires only a partition of the node set to define and create the aggregate (CTP).

Chapter three describes a partition refinement process which creates a less aggregated (CTP) by refining the partition of the node set associated with the previous

aggregate model. This partition refinement process is fundamental to the development, description and implementation of the disaggregation methods which form the crux of the algorithm described in chapter six. A measure of the degree of aggregation is also proposed.

In chapter four, a general disaggregation method is developed that maps basic feasible (optimal) solutions of an aggregate (CTP) to basic solutions of a refined, less aggregated (CTP).

Since this disaggregation map requires solution of a network subproblem, factors which affect the size and solution difficulty of this subproblem are discussed in chapter five. Several refinement heuristics are developed which yield small, easily solved subproblems.

Chapter six proposes a complete algorithm for the (CTP) which employs the aggregation-disaggregation concepts developed in chapters two through five.

Chapter seven summarizes the conclusions of this research and proposes directions for further research.

I. Taxonomy For Aggregation Research

What we shall mean by aggregation is congruent with the intuitive notion of the collecting of similar objects or entities into a mass or whole. Locations are routinely aggregated into districts and districts are aggregated into regions. Operating agencies are aggregated into

departments, departments into companies and companies into industries. In a network, nodes are aggregated into aggregate nodes and arcs are aggregated into aggregate arcs. In general then, a smaller, simplified aggregate model is constructed from an original model. We hope to preserve in the aggregate model the basic properties and characteristics of the original model.

Given the elements of the original model and the relationships among these elements, we must clearly and unambiguously specify how the corresponding elements and relationships in the aggregate model are to be constructed. This depends in large part on how the aggregate model is to be used. These notions are formalized in Geoffrion's modeling framework and aggregation theory.

The modeling framework identifies five basic types of model elements, prescribes how the element types can be related and describes properties enjoyed by models falling within the purview of the framework. Model elements and relationships among these elements are portrayed as a directed, acyclic graph.

The modeling framework is an integral part of the aggregation theory. It provides a means of model organization and representation that facilitates the development of the general aggregation theory and an aggregation procedure.

Some of Geoffrion's results bearing directly upon this

research include:

1. The important aspects and components of aggregation are identified and terminology is defined.
2. Four common uses of aggregation are identified.
3. Measures of aggregation error are defined for each of the four common uses.
4. Aggregation design problems are formulated which are intended to guide the search for "good" aggregate models.

This research addresses aggregation as applied to a selected class of network flow problems within the context of the modeling framework and the aggregation theory. It is assumed hereafter that the reader has reviewed the Appendix A: Geoffrion's Modeling Framework And Aggregation Theory, and is familiar with the terminology, notation and concepts contained therein.

The aggregation theory provides a convenient taxonomy for aggregation research. Studies may be classified according to:

1. Purpose of the aggregate model.
2. Aggregation procedures employed.
 - a. Primitive entity genus aggregated and the aggregation set or partition.
 - b. Respecification maps used.
 1. Structural respecification maps.

2. Variable respecification maps.
3. Measures of aggregation error used.
4. Disaggregation map employed.

We begin by reviewing chronologically previous studies which bear directly upon this research. This literature is then summarized according to the above taxonomy.

II. Literature Review

Balas, 1963, [1]

Balas was the first to consider aggregation as an algorithmic device for the solution of network flow problems. He developed an algorithm for the transportation problem that proceeds basically as follows:

1. From an original transportation problem (OTP) create an aggregate transportation problem (ATP).
2. Solve (ATP).
3. Based upon the solution of (ATP), form a "partial transportation problem" (PTP) consisting of only the "relevant parts" of (OTP).
4. Solve (PTP).
5. Test to see if the solution of (PTP) is optimal for (OTP). If it is, stop: the optimal solution of (OTP) is at hand. If it is not, create a new, expanded (PTP) and return to step 4.

Balas provides an optimality condition and proves the

convergence of the algorithm. The optimality condition is derived from dual feasibility conditions relating (OTP) and (PTP).

Balas suggests that the (ATP) be formed by aggregating "neighboring" subsets of nodes. The aggregate problem (ATP) is formed as follows:

$$(B1) \quad \text{Min} \quad \sum_{k \in \bar{S}} \sum_{l \in \bar{T}} \bar{c}_{kl} \bar{x}_{kl}$$

S.T.

$$(B2) \quad \sum_{l \in \bar{T}} \bar{x}_{kl} = \bar{a}_k, \quad k \in \bar{S}$$

$$(B3) \quad \sum_{k \in \bar{S}} \bar{x}_{kl} = \bar{b}_l, \quad l \in \bar{T}$$

$$(B4) \quad \bar{x}_{kl} \geq 0, \quad k \in \bar{S}, l \in \bar{T}$$

where:

$$(B5) \quad \bar{a}_k = \sum_{s \in S_k} a_s$$

$$(B6) \quad \bar{b}_l = \sum_{t \in T_l} b_t$$

$$(B7) \quad \bar{c}_{kl} = \text{Min}_{\substack{s \in S_k \\ t \in T_l}} c_{st}$$

and:

S = set of sources

$\{S_k: k \in \bar{S}\}$ = partition of S

a_s = supply at original source s

$\bar{a}_k$ = supply at aggregate source k

T = set of destinations

$\{T_l: l \in \bar{T}\}$ = partition of T

b_t = demand at original destination t

$\bar{b}_l$ = demand at aggregate destination l

c_{st} = unit cost on original arc (s,t)

$\bar{c}_{kl}$ = unit cost on aggregate arc (k,l)

x_{st} = flow on original arc (s,t)

$\bar{x}_{kl}$ = flow on aggregate arc (k,l)

Relationships (B5), (B6) and (B7) are, in the terminology of Appendix A, structural respecification maps for supplies, demands and costs, respectively.

The scale of aggregation is limited by the arbitrary scalar α in the following expression which defines "neighboring sources and destinations:"

$$|\bar{c}_{kl} - c_{st}| \leq \alpha$$

The partial problem, based upon the solution of (ATP), is constructed as follows:

$$(B8) \quad \text{Min} \quad \sum_{(s,t) \in B} c_{st} x_{st}$$

$$(B9) \quad \sum_{t: (s,t) \in B} x_{st} = a_s, \quad s \in S$$

$$(B10) \quad \sum_{s: (s,t) \in B} x_{st} = b_t, \quad t \in T$$

$$(B11) \quad x_{st} \geq 0, \quad (s,t) \in B$$

where:

$$(B12) \quad \bar{B} = \{(k,l): \bar{x}_{kl} > 0\}$$

$$(B13) \quad B = \{(s,t): s \in S_k, t \in T_l, (k,l) \in \bar{B}\}$$

Balas' key results are summarized in Theorems (BT1), (BT2) and Proposition (BP1) below:

Theorem (BT1): For any feasible solution $\bar{x}^0 = (\bar{x}_{kl}^0)$ of (ATP) there exists a feasible solution $x^0 = (x_{st}^0)$ of (PTP) related to $\bar{x}^0$.

Proof: $x_{st}^0 = (a_s b_t / \bar{a}_k \bar{b}_l) \bar{x}_{kl}^0$, $s \in S_k$, $t \in T_l$, $(k, l) \in B$ yields a feasible solution x^0 of (PTP). Also, any feasible solution of (PTP) is obviously a solution of (OTP).

Proposition (BP1): An optimal solution x^* of (PTP) is also an optimal solution of (OTP) if and only if

$$c_{st} - u_s - v_t \geq 0$$

for $s \in S_k$, $t \in T_l$, $(k, l) \in A - (\bar{B} \cup V)$ where $A = \{(k, l)\}$ and

$$V = \{(k, l) : (k, l) \in A - \bar{B}, D_{kl} \geq 0\}$$

and where

$$D_{kl} = \bar{c}_{kl} - \max_{s \in S_k} u_s - \max_{t \in T_l} v_t$$

Theorem (BT2): In a finite number of iterations, the proposed algorithm yields an optimal solution to (OTP).

The proof of Theorem (BT1) is provided here because it is the first time the "fixed weight" disaggregation map

$$x_{st}^0 = (a_s b_t / \bar{a}_k \bar{b}_l) \bar{x}_{kl}^0, \quad s \in S_k, \quad t \in T_l$$

is mentioned in the literature although Balas' algorithm does not specifically use this or any other disaggregation map. As will be discussed subsequently, this is by far the predominant disaggregation map mentioned in the literature.

Balas concludes with a discussion of the potential

efficiency of the method and remarks concerning the scale of the aggregation. As an example, he indicates that an (OTP) with $|S| = 200$ origins and $|T| = 250$ destinations would require $|S| \cdot |T| = 50,000$ evaluations of $c_{st} - u_s - v_t$, whereas if the aggregate problem were formed such that $|S_k| = |T_1| = 5$, his algorithm would require only 4136 similar calculations. Balas also notes that the improvement in the computational effort required by his algorithm increases with (1) the size of the problem and (2) the concentration (unevenness of territorial distribution) of origins and destinations.

It should be noted at this point that Lee [39] tested Balas' algorithm on the 10×12 example provided in [1]. He found that Balas' algorithm arrived at the optimal solution of (OTP) in 0.0756 cpu seconds whereas direct solution of (OTP) required 0.1603 cpu seconds. No mention is made however, of which optimization code was used to solve (OTP), (ATP) and (PTP). Since Lee [39] predates the marked advance in the efficiency of primal network codes of the late 1970's, these statistics should be viewed skeptically.

Finally, Balas recommends that in order for the size of (ATP) to be approximately equal to that of (PTP), the (ATP) should be constructed such that:

$$|\bar{S}|^2 |\bar{T}|^2 / (|\bar{S}| + |\bar{T}| - 1) \approx |S| |T|$$

Lee, 1975, [39]

In his doctoral dissertation, Lee [39] addressed the use of aggregate linear programs as surrogates for an original linear program. He developed a theoretical framework for such surrogate problems, established equivalence classes for collections of surrogate problems, showed that they constitute a metric space and studied the connection between sequences of surrogate problems and descending chains through the lattice representing row/column index set partition refinement. Further, he proposed several "replacement rules" (structural respecification maps) and characterized these according to several criteria.

In the final chapter, Lee outlined a general algorithm for the capacitated transshipment problem that involves solving a sequence of surrogate problems. He proved convergence of the algorithm to the optimal solution of the original problem. Thus, as with Balas' algorithm, no error bounds were needed or developed. Lee's algorithm constructs an aggregate problem, solves it and tests for optimality. If the disaggregated solution is not optimal for the original problem, a single subset of the node set partition is refined into two subsets and a revised, less aggregated problem is formed and solved. This process continues until optimality is achieved. No guidance is given as to which subset should be refined at each iteration or how to

construct the refined subsets. Lee defined an aggregation map and disaggregation map as part of a "general disaggregation process."

The aggregate problems were formed using the same respecification maps as the ones employed by Balas with the following addition:

$$\bar{d}_{kl} = \sum_{s \in S_k} \sum_{t \in T_l} d_{st}$$

where: $\bar{d}_{kl}$ = capacity of the aggregate arc (k,l)

d_{st} = capacity of original arc (s,t)

Appendix B provides a detailed review of Lee's aggregation framework and network algorithm. It is included for two reasons: (1) Lee adopted an algorithmic perspective and his framework provides an interesting alternative to the aggregation framework developed by Geoffrion from a modeling perspective, and (2) this research is closely related to the work of Lee and extends some of his results.

Geoffrion, 1976, [14]

Geoffrion [14] was the first to address the question of error introduced by aggregation of network type models. Specifically, he developed a bound on the optimal value error for a particular distribution planning model. Geoffrion's error bound is an a priori bound meaning that it can be determined prior to solution of the aggregate model. The original model (OM) considered by Geoffrion is

summarized below:

$$(Ga1) \quad \text{Min}_{y,z} \quad \sum_{i,j} d_{ij} y_{ij} + \sum_i F_i \left(\sum_j q_j y_{ij}, z_i \right)$$

S.T.

$$(Ga2) \quad \sum_i y_{ij} = 1, \text{ all } j$$

$$(Ga3) \quad \underline{v}_i z_i \leq \sum_j q_j y_{ij} \leq \bar{v}_i z_i, \text{ all } i$$

$$(Ga4) \quad 0 \leq y_{ij} \leq 1, \text{ all } i, j$$

$$(Ga5) \quad z_i = 0, 1, \text{ all } i \text{ and } z_i \in Z$$

where:

- i indexes the possible facilities from which customers can be served
- j indexes customers
- y_{ij} a variable giving the fraction of the annual demand of customer j satisfied by facility i
- z_i a binary variable indicating whether facility i is selected for use
- d_{ij} annual variable costs incurred if the full demand of customer j is provided by facility i
- $F_i()$ other annual costs associated with facility i as a function of throughput
- q_j the nonnegative annual demand of customer j
- $\underline{v}_i$ the lower limit on annual throughput of facility i
- $\bar{v}_i$ the upper limit on annual throughput of facility i
- Z arbitrary constraint set on z

Pro rata demand (fixed weight) customer aggregation introduces the following two additional constraints:

(Ga6) for each i , the y_{ij} 's must be identical over $j \in J$

(Ga7) for each $j \in J$, the same set of (i,j) links exist

Letting the notation $v(\cdot)$ indicate the optimal value for the problem $(\cdot)$, and (AM) represent the aggregate model, i.e. (Ga1) through (Ga7), the main result of the paper is stated below:

Theorem GaT1: Let J be any subset of customers satisfying (Ga7) above. Then $v(OM) \leq v(AM) \leq v(OM) + \epsilon_J$ where:

$$\epsilon_J = \sum_{j \in J} \max \left[\frac{q_j \sum_{j' \in J} d_{ij'} - d_{ij}}{\sum_{j' \in J} q_{j'}} \right]$$

Moreover, if (y^*, z^*) is ϵ -optimal in (AM), then it is feasible and $\epsilon + \epsilon_J$ optimal in (OM).

This particular theorem laid the ground work for the development of other error bounds, both by Geoffrion and others.

Geoffrion, 1977, [17]

This paper presents the extension of the results of [14] to the aggregation of items, rather than customers, in

a more general model. The original distribution planning model defined in this paper is given below:

$$(Gb1) \quad \min_{x,y,z} \sum_{ijk} c_{ijk} x_{ijk} + F(y,z)$$

$$(Gb2) \quad \underline{s}_{ij} \leq \sum_k x_{ijk} \leq \bar{s}_{ij}, \text{ all } i,j$$

$$(Gb3) \quad \sum_j x_{ijk} = \sum_l D_{il} y_{kl}, \text{ all } i,k$$

$$(Gb4) \quad \sum_k y_{kl} = 1, \text{ all } l$$

$$(Gb5) \quad x_{ijk} \geq 0, \text{ all } i,j,k$$

$$(Gb6) \quad y_{kl} \geq 0, \text{ all } k,l \text{ and } (y,z) \in \Omega$$

where:

- i indexes procurement items
- j indexes geographical procurement zones
- k indexes the facilities being supplied
(from procurement zones)
- l indexes customers (to be supplied from
the facilities)
- x_{ijk} the annual amount of item i procured
from zone j for facility k
- y_{kl} the fraction of the annual demand of
customer l satisfied by facility k
- z vector of additional variables
- c_{ijk} unit cost of procurement plus transporta-
tion associated with the flow x_{ijk}
- $F(y,z)$ the total annual costs associated with
(y,z) exclusive of procurement and inbound

transportation costs

$\underline{S}_{ij}$ lower bound on the annual amount of item i
procured from zone j

$\bar{S}_{ij}$ upper bound on the annual amount of item i
procured from zone j

D_{il} the amount of item i required to satisfy
the total annual needs of customer l

Ω a constraint set that must be satisfied
by (y, z)

From (OM) an intermediate aggregate model $(\bar{AM})$ is
formed by aggregating the items of some subset, $i \in I$:

$(\bar{AM})$ is the same as model (OM) except (Gb2) is
replaced with:

$$(Gb7) \quad \underline{S}_{ij} \leq \sum_k x_{ijk} \leq \bar{S}_{ij}, \text{ all } i, j \text{ with } i \in I$$

$$(Gb8) \quad \underline{S}_{Ij} \leq \sum_{i \in I} \sum_k x_{ijk} \leq \bar{S}_{Ij}, \text{ all } j$$

where:

$$(Gb9) \quad \underline{S}_{Ij} = \sum_{i \in I} \underline{S}_{ij}$$

$$(Gb10) \quad \bar{S}_{Ij} = \sum_{i \in I} \bar{S}_{ij}$$

The final aggregate model (AM) formed by aggregating
items $i \in I$ and variables x_{ijk} for $i \in I$ is given below:

$$(Gb11) \quad \text{Min}_{x, y, z, \xi} \sum_{i \in I} \sum_{j, k} c_{ijk} x_{ijk} + \sum_{j, k} b_{jk} \xi_{jk} + F(y, z) \\ + L(y, b)$$

S.T.

$$(Gb12) \quad \underline{S}_{ij} \leq \sum_k x_{ijk} \leq \bar{S}_{ij}, \text{ all } i, j \text{ with } i \in I$$

$$(Gb13) \quad \underline{S}_{Ij} \leq \sum_k j_k \leq \bar{S}_{Ij}, \text{ all } j$$

$$(Gb14) \quad \sum_j \xi_{jk} = \sum_l D_{Il} Y_{kl}, \text{ all } k$$

$$(Gb15) \quad \sum_j x_{ijk} = \sum_l D_{il} Y_{kl}, \text{ all } i, k \text{ with } i \in I$$

$$(Gb16) \quad \sum_k Y_{kl} = 1, \text{ all } l$$

$$(Gb17) \quad x_{ijk} \geq 0, \text{ all } i, j, k \text{ with } i \in I$$

$$(Gb18) \quad \xi_{jk} \geq 0, \text{ all } j, k \text{ such that } i, j, k, \text{ exists} \\ \text{for } i \in I$$

$$(Gb19) \quad Y_{kl} \geq 0, \text{ all } k, l \text{ and } (y, z) \in \Omega$$

$$(Gb20) \quad b_{jk}'s \text{ are plausible surrogates for } c_{ijk}'s \\ \text{over } i \in I$$

$$(Gb21) \quad \xi_{jk} \text{ is a surrogate for } \sum_{i \in I} x_{ijk}$$

$$(GB22) \quad D_{Il} = \sum_{i \in I} D_{il}$$

$$(Gb23) \quad L(y, b) \text{ is some linear function of } y \\ \text{designed to compensate for aggrega-} \\ \text{tion error}$$

In the terminology of Appendix A, (Gb9) and (Gb10) are structural respecification maps for upper and lower annual procurement limits. The aggregate variable ξ_{jk} corresponding to the original variables x_{ijk} , $i \in I$ is determined by the aggregation map

$$\xi_{jk} = \sum_{i \in I} x_{ijk}$$

Relation (Gb22) is a structural respecification map for annual customer demand. Although (aggregate) costs require respecification, no specific structural respecification map is stated and the following result holds for any (general) cost respecification.

Finally, the a priori error bound arrived at by Geoffrion is given below:

Theorem (GbT1): Assume that the same j,k links exist for every item $i \in I$. Let b_{jk} be chosen arbitrarily for these links, and take the compensation function L to be:

$$L(y,b) = \sum_{kl} \left(\sum_{i \in I} D_{il} \min_j \{c_{ijk} - b_{jk}\} \right) y_{kl}$$

Then

$$v(AM) \leq v(\overline{AM}) \leq v(AM) + \epsilon_b$$

where:

$$\epsilon_b = \sum_l \max_k \left\{ \sum_{i \in I} D_{il} \max_j \{c_{ijk} - b_{jk}\} \right\}$$

Moreover, a complete ϵ_b -optimal solution of $(\overline{AM})$ can be obtained from any optimal solution (x^*, y^*, z^*, ξ^*) of (AM) by using (x^*, y^*, z^*) as is and supplementing it by values for the missing x_{ijk} for $i \in I$ according to the disaggregation formula: for all i, j, k with $i \in I$ put

$$x_{ijk}^* = \begin{cases} \left[\frac{\sum_{l=1}^I D_{il} y_{kl}^*}{\sum_{l=1}^I D_{Il} y_{kl}^*} \right] \xi_{jk}^* & \text{if } \xi_{jk}^* > 0 \\ 0 & \text{if } \xi_{jk}^* = 0 \end{cases}$$

The above "disaggregation formula," a generalization of the one defined in Balas [1], is of the fixed weight variety with weights determined according to pro rata demand.

It is noted by Geoffrion that the result is easily generalized for partitions of the index set of i . Guidance is also provided on how b should be selected. It is shown that the ϵ_b -minimizing choice of b can be obtained via linear programming and, under certain conditions, can be obtained analytically. A characterization of situations yielding $\epsilon_b = 0$ is also provided. Lastly, Geoffrion proposes how a cluster analysis approach, using the error bound and conditions for zero error, might guide the modeler in his selection of the subset I .

Zipkin, 1977, [54], 1980, [57]

Zipkin [54, 57] has produced a comprehensive piece of work concerning aggregation of minimum cost network flow problems. He extends the results of previous authors for the transportation problem by allowing aggregation of both sources and destinations and by developing a priori and a posteriori optimal value error bounds for this more general aggregation scheme. His results recover and unify those of

Geoffrion [14] and Cullen, et. al. [5]. Zipkin extends, under certain assumptions, these results to the transshipment and capacitated transshipment problems. Methods are proposed to simplify or modify the original network problem if the necessary assumptions are not met. He developed an optimal disaggregation map which requires solution of a suitably defined network subproblem. Finally, Zipkin outlined an algorithm which uses fixed weight aggregation and the a priori and/or a posteriori optimal value error bounds to provide stopping criteria and to guide partition refinement.

Zipkin initially considers the (original) transportation problem (OTP) and creates the aggregate transportation problem (ATP) using "fixed weight aggregation:"

$$\begin{aligned}
 (\text{OTP}) \quad & \text{Min } \sum_{\substack{s \in S \\ t \in T}} c_{st} x_{st} \\
 & \text{S.T.} \\
 & \sum_{t \in T} x_{st} = a_s, \text{ all } s \in S \\
 & \sum_{s \in S} x_{st} = b_t, \text{ all } t \in T \\
 & x_{st} \geq 0, \text{ all } s \in S, t \in T
 \end{aligned}$$

$$(\text{ATP}) \quad \text{Min } \sum_{\substack{k \in \bar{S} \\ l \in \bar{T}}} \bar{c}_{kl} \bar{x}_{kl}$$

S.T.

$$\sum_{l \in \bar{T}} \bar{x}_{kl} = \bar{a}_k, \text{ all } k \in \bar{S}$$

$$\sum_{k \in \bar{S}} \bar{x}_{kl} = \bar{b}_l, \text{ all } l \in \bar{T}$$

$$\bar{x}_{kl} \geq 0, \text{ all } k \in \bar{S}, l \in \bar{T}$$

where: S = the set of sources

T = the set of destinations

a_s = the positive supply at source $s \in S$

b_t = the positive demand at destination $t \in T$

c_{st} = unit cost for shipping one unit from
source $s \in S$ to destination $t \in T$

x_{st} = flow from source $s \in S$ to destination $t \in T$

$\{S_k: k \in \bar{S}\}$ = partition of the source set S

$\{T_l: l \in \bar{T}\}$ = partition of the destination set T

$\bar{a}_k = \sum_{s \in S_k} a_s$ = supply for aggregate node k

$\bar{b}_l = \sum_{t \in T_l} b_t$ = demand at aggregate node l

$$g_s^k = a_s / \bar{a}_k$$

$$g_t^l = b_t / \bar{b}_l$$

$$g_{st}^{kl} = g_s^k g_t^l$$

$$\bar{c}_{kl} = \sum_{\substack{s \in S_k \\ t \in T_l}} g_{st}^{kl} c_{st}$$

where sources in S_k are combined into a single aggregate source $k \in \bar{S}$ and destinations in T_l are combined into a single aggregate destination $l \in \bar{T}$. This aggregation scheme has been

commonly referred to in the literature as "weighted aggregation."

Defining $\bar{x} = (\bar{x}_{kl})$ as an optimal solution of (ATP) and $\bar{z}$ as the optimal objective function value of (ATP), Zipkin determines the "fixed weight solution" obtained from $\bar{x}$ by "fixed weight disaggregation" as

$$x_{st} = g_{st}^{kl} \bar{x}_{kl}, \text{ all } s \in S_k, t \in T_1$$

Proposition Z1: (a) If $\bar{x}$ is feasible for (ATP), and x is the fixed weight solution obtained from $\bar{x}$, then x is feasible for (OTP). (b) If $\bar{x}$ is optimal for (ATP), and x is the fixed weight solution obtained from $\bar{x}$, then

$$\sum_{\substack{s \in S \\ t \in T}} c_{st} x_{st} = \bar{z}$$

Corollary Z1: $\bar{z} \geq z^*$

Proposition Z2: $\bar{z} - z^* \leq \sum_t b_t \max \{(\bar{U}_{k(s)} + \bar{V}_1(t) - c_{st})\}$

and $\bar{z} - z^* \leq \sum_s a_s \max \{(\bar{U}_{k(s)} + \bar{V}_1(t) - c_{st})\}$

where $k(s) = \text{that } k \in \bar{S} \text{ such that } s \in S_k, s \in S$

$l(t) = \text{that } l \in \bar{T} \text{ such that } t \in T_1, t \in T$

$(\bar{U}, \bar{V}) = \text{the dual optimal solution of (ATP)}$

Proposition 2 gives an a posteriori optimal value error bound, whereas Corollary 2 below gives the analagous a priori optimal value error bound.

Corollary Z2: $\bar{z} - z^* \leq \sum_t b_t \max \{(\bar{c}_{k(s),l(t)} - c_{st})\}$

and $\bar{z} - z^* \leq \sum_s a_s \max \{(\bar{c}_{k(s),l(t)} - c_{st})\}$

Zipkin then generalizes the above results for the (transshipment) network problem (OTP) to include the following in the obvious way:

N = set of nodes

A = set of arcs

c_{ij} = unit cost of flow from node i to node j

I = set of intermediate nodes, i.e. no net supply or demand

$N_{to}^j = \{i \in N: (i,j) \in A\}, j \in N$

$N_{fr}^i = \{j \in N: (i,j) \in A\}, i \in N$

The aggregation of (OTP) is determined by a partition of the node set N satisfying the following assumptions:

- (A1) Each subset consists entirely of sources, destinations or intermediate nodes
- (A2) There are no arcs between nodes in a single subset in (OTP)
- (A3) All nodes in a subset J have identical connections to nodes outside the subset, i.e. if $i, j \in J$, then $N_{to}^i = N_{to}^j$ and $N_{fr}^i = N_{fr}^j$.

The partition of N consists of partitions of S , T , and I .

In addition to the notation S_k and T_1 defined previously, let:

$\{I_m: m \in \bar{I}\}$ be a partition of I .

$\bar{N} = \bar{S} \cup \bar{T} \cup \bar{I} = \text{nodes of (ATP)}$

$n, p = \text{general elements of } \bar{N}$

$J, J_n, J_p = \text{general subsets of the partition of } N$

$n(i) = \text{that } n \in \bar{N} \text{ such that } i \in J_n, i \in N$

In view of the assumptions (A1) through (A3), Zipkin defines the set of aggregate arcs to be

$$\bar{A} = \{(n, p): n, p \in \bar{N}, (i, j) \in A, i \in J_n, j \in J_p\}$$

To complete the aggregation of (OTP), Zipkin stipulates that $\bar{a}_k$ and $\bar{b}_1$ are obtained as previously explained and he defines $g_i^m, i \in J_n, m \in \bar{I}$ as arbitrary numbers such that

$$g_i^m \geq 0$$

$$\text{and } \sum_{i \in I_m} g_i^m = 1$$

Under weighted aggregation, then, the costs for the aggregate arcs are determined as:

$$g_{ij}^{np} = g_i^n g_j^p, i \in J_n, j \in J_p, (n, p) \in \bar{A}$$

$$\bar{c}_{np} = \sum_{i \in J_n} \sum_{j \in J_p} g_{ij}^{np} c_{ij}, (n, p) \in \bar{A}$$

Proposition Z3: Let r_t denote the length of the shortest path from any source to $t, t \in T$, using the costs

$[c_{ij} - (\bar{u}_p(j) - \bar{u}_n(i))]$. Then

$$\bar{z} - z^* \leq \sum_t b_t r_t$$

The final extension is to the capacitated transshipment problem. In addition to the notation used above let:

$$d_{ij} = \text{capacity of arc } (i,j) \in A$$

$$\bar{d}_{np} = \min [d_{ij}/g_{ij}^{np} : i \in J_n, j \in J_p, g_{ij}^{np} > 0], (n,p) \in \bar{A}$$

The rationale for the respecification of $\bar{d}_{np}$ is that it ensures that the fixed weight solution of (OTP) is feasible. Unfortunately, (ANP) may be infeasible under this respecification map.

Proposition 24:

$$\bar{z} - z^* \leq - \sum_{n,p} \bar{w}_{np} (\bar{d}_{np} - \sum_{J_n, J_p} d_{ij})$$

$$- \min_x \sum_{i,j} [c_{ij} - (\bar{u}_p(j) - \bar{u}_n(i) - \bar{w}_{n(i)p(j)})] x_{ij}$$

S.T. the constraints of (ONP)

where $\bar{w}$ are the optimal dual multipliers associated with I.

Zipkin discusses "optimal disaggregation" as a way to improve upon the "fixed weight solution" of (OTP). The "restriction of (OTP)," denoted (RONP) is defined to be the network problem which includes only the arcs

$$\{(i,j) : i \in J_n, j \in J_p, (n,p) \in \bar{A}, \bar{x}_{np} > 0\}$$

Solution of (RONP) is termed "the optimal disaggregation" of $\bar{x}$. This approach to disaggregation is generalized to the solution of one or several subproblems by the following

procedure:

1. Choose any subset $R \subseteq N$.
2. Form the subproblem using the nodes of R , arcs of (ONP) restricted to R on which the fixed weight solution has positive flow and with supplies and demands determined by net outflows and inflows of the fixed weight solution restricted to flows between the nodes of R .
3. Solve this subproblem.
4. Assign the fixed weight solution to arcs not included.

The resulting solution is feasible for (OTP) and is at least as good as the fixed weight solution. The above approach is easily extended to more than one subset.

Zipkin concludes with a heuristic method for choosing the partition of the sources and destinations of the transportation problem. It is analogous to Geoffrion's suggestion in [17] to using a priori optimal value error bounds in conjunction with cluster analysis. The heuristic is outlined below.

1. Initialize $i = 0$, $|\bar{S}^0| = |\bar{T}^0| = 1$ where $\{S_k^i: k \in \bar{S}^i\}$, $\{T_l^i: l \in \bar{T}^i\}$ is the i^{th} partition of S and T .
2. Compute $c_{st}^i = (\bar{c}_{kl}^i - c_{st})$, $s \in S_k^i$, $t \in T_l^i$, $k \in \bar{S}^i$, $l \in \bar{T}^i$, where $\bar{c}_{kl}^i$ is as defined previously and in terms of partition i .

3. If $\epsilon^{\max}$ is the stopping criterion, compute any of the a priori error bounds given previously using c_{st}^i as maximands, and set ϵ^i to the result.
4. If the stopping criterion is satisfied, stop.
5. Split each S_k^i into two subsets,
 $\{s \in S_k^i : \max_t \{c_{st}^i\} \leq \theta\}$ and
 $\{s \in S_k^i : \max_t \{c_{st}^i\} > \theta\}$.
 Similarly split each subset T_l^i .
 Set $i = i+1$ and relabel the nonempty subsets to form the new partition.
6. Return to step 2.

Proposed modifications to the above heuristic include

- (1) refining some, but not all, of the subsets at each iteration into any arbitrary number of subsets,
- (2) using alternative functions of c_{st}^i instead of the maxima,
- (3) using a posteriori bounds in conjunction with solution of (ATP).

Cullen, Kuhn, Frank, (1978), [5]

A portion of the work by Cullen, Kuhn and Frank [5] dealt with aggregation of destinations in the transportation problem. Fixed weight aggregation and the fixed weight disaggregation map were used. They developed an a

posteriori (requires solution of the aggregate model) error bound for optimal value error. It is based upon a disaggregation map that maps the optimal dual solution of the aggregate problem into a feasible dual solution of the original problem. Also proposed is a method of refining the destination node set partition based upon the dual optimal solution of the (current) aggregate problem. The refinement method requires "normalization of the cost vectors for each destination and projection of these vectors onto the plane defined by the average costs." Destinations projecting into the same point form a subset of the partition. The rationale for such a method is not clearly stated and no formal proofs are provided.

Keith, (1978), [37]

Keith [37] studied the transportation problem also. The primary purpose of her study was to investigate empirically the behavior of the a priori error bound developed by Geoffrion in [14], of a new a priori optimization value error bound and of the a posteriori error bound of Cullen, Kuhn and Frank [5]. She investigated the behavior of these bounds with respect to changes in the partition of the source or destination node set used to construct the aggregate problem. Keith also considered the effect on the disaggregated solution vector. She used fixed weight aggregation to construct the aggregate transportation

problems and the fixed weight disaggregation map.

Node set partitions were derived using subjective judgement and various clustering methods. Some of her conclusions were:

1. For alternative source or destination set partitions, the rank ordering according to a priori optimal value error bounds corresponded closely to the rank ordering induced by actual error.
2. A priori optimal value error bounds were good predictors of actual error.
3. There was little correlation among the alternative disaggregated solution vectors.

Burkett, (1978), [3]

The thesis by Burkett [3] also addressed the aggregation design problem of selecting a "good" destination node set partition for the transportation problem. His study was an empirical investigation of Geoffrion's a priori optimal value error bound [14] also. Fixed weight aggregation and the fixed weight disaggregation map were used. He determined partitions of the destination node set via cluster analysis, subjective judgement and a method based upon the initial solution of the original problem dictated by Vogel's Approximation Method (VAM).

The clustering methods used Geoffrion's a priori bound

as the clustering criterion. The VAM dictated partition grouped together in the same subset those destinations receiving (non-zero) flow from the same set of sources. Burkett's rationale for such a partitioning scheme was that actual optimal value error is equal to zero for the analagous partition obtained from the optimal solution of the original transportation problem. The VAM solution is easily obtained and is used to approximate the (unavailable) optimal solution.

Burkett's important conclusions include:

1. For alternative destination node set partitions, the rank ordering according to the a priori optimal value error bound showed no correlation to the rank ordering induced by actual optimal value error.
2. The VAM induced partitioning scheme clearly produced the best aggregate problems with respect to actual optimal value error.
3. There was no consistent relationship between a priori optimal value error bounds and actual optimal value error.

It is interesting to note that Burkett's conclusions 1. and 3. above are contradictory to Keith's conclusions 1. and 2. This can probably be attributed in part to the fact that Keith used a single symmetric test problem whereas Burkett used a battery of test problems of equivalent size.

Evans, 1979, [9]

In this paper, Evans develops an a posteriori error bound analagous to that presented by Zipkin in [57] for the generalized transportation problem. The generalized transportation problem of interest is the same as the classical transportation problem except that the demand constraints are replaced by:

$$\sum_s f_{st} x_{st} = b_t, \text{ all } t \in T$$

The extension of the weighted aggregation scheme for an aggregation set of sources $P = \{p, p+1, p+2, \dots, n\} \subset S$ is:

$$\bar{a}_k = a_s, \quad l=k < p$$

$$\bar{a}_p = \sum_{s \in P} a_s$$

$$\bar{c}_{kl} = c_{st}, \quad l=k < p$$

$$\bar{c}_{pl} = \sum_{s \in P} (a_s / \bar{a}_p) c_{st}, \text{ all } t \in T$$

$$\bar{f}_{kl} = f_{st}, \quad l=k < p$$

$$\bar{f}_{pl} = \sum_{s \in P} (a_s / \bar{a}_p) f_{st}, \text{ all } t \in T$$

Theorem ET1: Let $\bar{x}$ be a feasible solution of (ATP), with the fixed weight disaggregation defined as

$$x_{st} = \bar{x}_{st}, \quad l=s < p$$

$$x_{st} = (a_s / \bar{a}_p) \bar{x}_{pl}, \quad s \in P$$

Then x is feasible in (OTP) and

$$\sum_{s \in S} \sum_{t \in T} c_{st} x_{st} = \sum_{k=1}^p \sum_{l=1}^n \bar{c}_{kl} \bar{x}_{kl}$$

The a posteriori optimal value error bound derived by Evans is given in Theorem ET2 below.

Theorem ET2:

$$\begin{aligned} \bar{z} - z^* &\leq \sum_{s \in P} \max_t \{-\bar{u}_p + f_{st} \bar{v}_t - c_{st}\} \sum_t x_{st}^* \\ &= \sum_{s \in P} \max_t \{-\bar{u}_p + f_{st} \bar{v}_t - c_{st}\} a_s \\ &= \epsilon \end{aligned}$$

Evans notes that an a priori bound cannot be developed for this generalized transportation problem. He concludes with a discussion of how an analogous a posteriori bound can be developed when destinations are aggregated with an appropriate substitution of variables.

Geoffrion, 1982, [19]

This monograph, describing the modeling framework and aggregation theory, has been discussed previously in this chapter. A detailed summary is provided in Appendix A.

Zipkin and Raimer, 1983, [58]

This paper presents an extension of the "optimal disaggregation method" given in [57] that recovers an

all-integer solution of (OTP) when both sources and destinations are aggregated. The solution produced via this new method is at least as good as the solution produced by fixed weight disaggregation and the error bounds developed in [57] still apply.

The notation used in [57] applies to the discussion of this paper also. The aggregate transportation problem (ATP) is formed using fixed weight aggregation and is solved to obtain the optimal solution $\bar{x}^*$.

The disaggregation method proposed by Zipkin and Raimier requires solution of two sets of transportation subproblems. The first set requires solution of a transportation subproblem for each $k \in S$, where the k^{th} subproblem is defined as having:

sources: S_k

destinations: $\bar{T}'_k = \{l \in \bar{T} : \bar{x}_{kl} > 0\}$

supplies: $a_s, s \in S_k$

demands: $\bar{x}_{kl}, l \in \bar{T}'_k$

costs: $c'_{sl} = \sum_{t \in \bar{T}_1} g_t^l c_{st}, s \in S_k, l \in \bar{T}'_k$

Denote as $y' = (y'_{sl})$ the collected solutions of all of these subproblems and $y'_{sl} = 0$ when $s \in S_k$ and $l \notin \bar{T}'_k$.

The second set of subproblems includes a transportation problem for each $l \in \bar{T}$. The l^{th} subproblem has:

sources: $\bar{S}'_l = \{s \in S : y'_{sl} > 0\}$

destinations: T_l

supplies: $y'_{s1}, s \in \bar{S}'_1$

demands: $b_t, t \in T_1$

costs: $c_{st}, s \in \bar{S}'_1, t \in T_1$

Let $x' = (x'_{st})$ be the collected solutions of these subproblems and $x'_{st} = 0$ when $t \in T_1$ and $s \in \bar{S}'_1$.

Proposition ZR1: x' is feasible for (OTP) and

$$\sum_{s,t} c_{st} x'_{st} \leq \sum_{s,t} c_{st} x^0_{st}$$

where x^0 is the fixed weight disaggregated solution of (OTP).

The authors note that:

- (1) If (OTP) has integral supplies and demands, then so does each subproblem and hence the disaggregated solution is integral.
- (2) The roles of S and T can be reversed to yield an alternative disaggregation method with the same properties.
- (3) Proposition ZR1 remains valid even if (ATP) is not solved to optimality.
- (4) The same results hold if the respecification map for costs is taken to be the one proposed by Balas [1] and Lee [39], i.e.

$$\bar{c}_{k1} = \min \{c_{st} : s \in S_k, t \in T_1\}$$

provided that costs c'_{s1} are computed as

explained above. If instead these are computed as

$$c'_{s1} = \min \{c_{st} : t \in T_1\}$$

then x' is feasible for (OTP) and integral but the inequality in Proposition ZR1 may be violated

Lastly, the authors compare the computational effort required for direct solution of (OTP) versus solving (ATP) and disaggregating $\bar{x}^*$. For $|S| = n$, $|T| = m$, a commonly used empirical estimate for the computational complexity of the solution of (OTP) is $O[mn(m + n)]$. Define $|\bar{S}| = \bar{n}$, $|\bar{T}| = \bar{m}$, $|S_k| = n/\bar{n}$, $|T_1| = m/\bar{m}$ under the assumption that the subsets are of equivalent size. It is shown that the effort required to solve the first set of subproblems is

$$O[(n/\bar{n}) \max \{\bar{m}^2, (n/\bar{n})\bar{m}, n\}]$$

and for solution of the second set of subproblems is

$$O[(m/\bar{m}) \max \{\bar{n}^2, (m/\bar{m})\bar{n}, m\}]$$

and for solution of (ATP) is

$$O[\bar{m}\bar{n}(\bar{m} + \bar{n})]$$

It is noted that, although it is difficult to determine which of the above dominates, all three terms are considerably less than the effort required for (OTP).

Taylor, 1983, [48], Taylor and Sheety, 1984, [49]

Taylor and Sheety developed an algorithm for solution of the classical transportation problem that involves solution of a sequence of aggregate problems, each a restriction of the immediately preceeding one. Throughout the discussion of this paper, the notation of Zipkin [57] will be used.

Although derived from the perspective of aggregating variables/arcs rather than the more traditional aggregation of nodes, the authors develop, from a "general multiplicative aggregate model," the fixed weight aggregation of the original model (OTP) and the fixed weight aggregation map. From the general multiplicative aggregate model, conditions are derived for the multiplicative constants that guarantee that the resultant aggregate model will be a transportation model. These conditions are equivalent to the structural respecification maps of the fixed weight aggregation scheme.

Proposed is a "measure of closeness" of destinations with a view toward forming the initial aggregate model. This measure is based upon the "spatial closeness between pairs of dual constraints" as defined below:

$$h_t = \cos^{-1} \frac{c_t \cdot w}{\|c_t\| \|w\|}$$

$$c_t = (c_{st}) \quad \text{an } S \times 1 \text{ vector of costs associated with destination } t$$

$$w = (1/m)(c_1, c_2, \dots, c_m)$$

$$m = |T|$$

If there are to be $|\bar{T}| = \bar{m}$ aggregate destinations, define angles e_i such that

$$0 \leq e_1 \leq e_2 \leq \dots \leq e_{m-1} \leq 180$$

Original destinations are grouped into subsets as follows:

$$T_1 = \{t: 0 \leq h_t \leq e_1, t \in T\}$$

$$T_2 = \{t: e_1 < h_t \leq e_2, t \in T\}$$

.

.

.

$$T_m = \{t: e_{m-1} < h_t \leq 180, t \in T\}$$

The e_i 's are chosen to uniformly partition the interval $[0, 180]$, and the authors imposed the additional restriction that $|T_l| = 3$, all $l \in \bar{T}$, for the purposes of this paper.

Taylor and Sheety developed an a posteriori error bound that is an extension of the one given by Zipkin [57]. The authors state that the bound of Theorem TS1 below was, on average, 16.8% better than Zipkin's over forty observations.

Theorem TS1: Let x^* solve (OTP) with value z^* . Let (ATP) be the (fixed weight) aggregate problem of (OTP). Let $\bar{x}^*$ solve (ATP) with value $\bar{z}^*$, and with optimal dual multipliers $(\bar{u}^*, \bar{v}^*)$. Let $v_t^0 = v_l^*$ for each $t \in T_l$, for each l . Let the terms

$$[(v_t^0 + \theta d_t^0 + (u_s^* + \theta d_s) - c_{st})]$$

be ordered over i to form a nonincreasing sequence, where (d^0, d) is a general direction in the dual space and θ is a step size for the line search. Let

$$\begin{aligned} z(\theta) = & z^* + \theta \left(\sum_{t \in T} b_t d_t + \sum_{s \in S} a_s d_s^0 \right) \\ & - \sum_{s \in S} \sum_{t \in T} [(v_t^0 + \theta d_t^0) + (u_s^* + \theta d_s^0) - c_{st}] \end{aligned}$$

$$\max \{0, \min [a_s, b_t - \sum_{r=1}^{s-1} a_r]\}$$

Let $\theta^* = \operatorname{argmax}(\theta)$. Then $z(\theta^*) = z^*$.

For purposes of computational testing, the authors restricted attention to $\theta = 0$ also eliminating the need to choose the search direction. The authors developed the analagous bound, $z'(\theta)$, with the minimand in Theorem TS1 replaced by

$$[b_t, a_s - \sum_{r=1}^{t-1} b_r]$$

Both of the bounds $z(\theta)$ and $z'(\theta)$ are used in computation with

$$z^P = \max \{z(\theta), z'(\theta)\}$$

where $\theta = 0$ and $z^P \leq z^* \leq \bar{z}^*$. If this error tolerance proves adequate, the solution of (ATP) is disaggregated (using fixed weight disaggregation). Otherwise, a refined aggregate problem is formed and solved yielding a new bound. The new, refined aggregate problem is formed by disaggregating one original destination from each aggregate node. The original destination $t^0 \in T_1$ that is disaggregated is determined by the dual constraint that is "most violated," i.e.

$$t_1 \{t \in T_1: (\bar{u}_s + v_t^0 - c_{st}) \geq (\bar{u}_s - \bar{v}_t - c_{st}) \text{ for each } t \in T_1, \\ \text{and } (\bar{u}_s + \bar{v}_t - c_{st}) \geq 0\}$$

Letting, at each iteration i , (ATP^i) be the aggregate problem with optimal solution $\bar{x}_*^i$ and value $\bar{z}_*^i$, the authors prove:

Theorem TS2: $\bar{z}_*^i \geq \bar{z}_*^{i+1}$.

It is noted that the lower bounds are not necessarily monotonically increasing. At each iteration i the current percentage optimal value error is

$$\frac{z_u - z_r}{z_r}$$

where

$$\begin{aligned} z_r &= \max \{z_p^1, z_p^2, \dots, z_p^i\} \\ z_u &= \bar{z}_*^i \end{aligned}$$

Taylor and Sheety conclude with computational testing of their aggregation algorithm on randomly generated transportation problems of size 30×30 and 100×100 . The transportation code TPGO was used as the optimizer on a CYBER 170/730. In summary, the aggregation algorithm required slightly less cpu time to solve the test problems to within 10% of optimality than was required to solve (OTP) directly.

The above studies are categorized in Table 1 according to the taxonomy presented earlier in this chapter.

STUDY:	Balas [1]	Lee [39]
PURPOSE:	Opt. algo for (TP)	Aggn. framework, opt algo for (CTP)
AGGN. SET(S):	S,T	N
STRUCT. RES.:	RS(min)	RS(min)
VARIABLE RES.:	none	ϕ
DISAGG. MAP:	DM(f)	ψ (sum)
ERROR MEAS.:	none	none
STUDY:	Geoffrion [14]	Geoffrion [17]
PURPOSE:	Opt. val. surrogate, aggn. design, gen. distn. model	Opt. val. surrogate aggn. design, gen. distn. model
AGGN. SET(S):	subset of T	subset of C
STRUCT. RES.:	RS(fix)	RS(fix)
VARIABLE RES.:	none	AM(sum)
DISAGG. MAP:	DM(f)	DM(f)
ERROR MEAS.:	apriori opt. value error bound	apriori opt. value error bound
KEY:		
S: sources	(TP): transportation model	
T: destination	(CTP): capac. transship. model	
I: transshipment nodes		
C: commodities or items		
RS(min): as defined by Balas, see p. 7		
RS(fix): fixed wt. aggn., see p. 20		
AM(sum): as defined in Gb21, see p. 17		
DM(f): fixed wt. disaggn., see p. 9		
ψ (sum): general, Appendix B		
DM(opt): optimal disaggn., see p. 25		
DM(dual): feasible dual disaggn.		
ϕ : general, see Appendix B		

Table 1.1: Summary And Classification Of Previous Studies

STUDY:	Zipkin [54,57]	Cullen, et al [5]
PURPOSE:	Opt. val. surrogate, opt. surrogate, algo. for (TP),(CTP)	Empirical study of apriori, aposter. bounds for (TP)
AGGN. SET(S):	S,T S,I,T	T
STRUCT. RES.:	RS(fix)	RS(fix)
VARIABLE RES.:	none	none
DISAGG. MAP:	DM(f), DM(opt)	DM(fix), DM(dual)
ERROR MEAS.:	apriori,aposter. opt. val. bounds	aposteriori opt. value bound
STUDY:	Keith [37]	Burkett [3]
PURPOSE:	Empirical study of apriori, aposter. opt. val. bounds for (TP)	Empirical study of apriori opt. val. bounds for (TP)
AGGN. SET(S):	S or T	T
STRUCT. RES.:	RS(fix)	RS(fix)
VARIABLE RES.:	none	none
DISAGG. MAP:	DM(f), DM(opt)	DM(f)
ERROR MEAS.:	apriori, aposter. opt. val. bounds	apriori opt. value error bound
KEY:	S: sources (TP): transportation model T: destinations (CTP): capac. transship. I: transshipment nodes model C: commodities or items RS(min): as defined by Balas, see p. 7 RS(fix): fixed wt. aggn., see p. 20 AM(sum): as defined in Gb21, see p. 17 DM(f): fixed wt. disaggn., see p. 9 ψ (sum): general, see Appendix B DM(opt): optimal disaggn., see p. 25 DM(dual): feasible dual disaggn. ϕ : general, see Appendix B	

Table 1.1: Summary And Classification Of Previous Studies
(Continued)

STUDY:	Evans [9]	Geoffrion [19]
PURPOSE:	Opt. val.surrogate, opt. surrogate for (GTP)	Modeling framework, aggregation theory
AGGN. SET(S):	S	general
STRUCT. RES.:	RS(fix)	general
VARIABLE RES.:	none	general
DISAGG. MAP:	DM(f)	general
ERROR MEAS.:	aposter. opt. val. bound	general
STUDY:	Zipkin,et al [58]	Taylor [48,49]
PURPOSE:	Extension of opt. disaggn. for (TP)	Opt. val.surrogate for (TP)
AGGN. SET(S):	T, S	T
STRUCT. RES.:	RS(fix)	RS(fix)
VARIABLE RES.:	none	none
DISAGG. MAP:	DM(f)	DM(f)
ERROR MEAS.:	aposter. opt. val. error bound	aposter. opt. val. error bound
KEY:	S: sources T: destinations I: transshipment nodes C: commodities or items RS(min): as defined by Balas, see p. 7 RS(fix): fixed wt. aggn., see p. 20 AM(sum): as defined in Gb21, see p. 17 DM(f): fixed wt. disaggn., see p. 9 ψ (sum): general, see Appendix B DM(opt): optimal disaggn., see p. 25 DM(dual): feasible dual disaggn. Φ : general, see Appendix B	
	(TP): transportation model	(CTP): capac. transship. model

Table 1.1: Summary And Classification Of Previous Studies
(continued)

CHAPTER 2

AGGREGATION OF A NETWORK FLOW PROBLEM

In this chapter we describe the aggregation of a network flow problem, more specifically, of a capacitated transshipment problem (CTP). Other researchers have described this process and we have reviewed some of these methods in chapter one. The aggregation method proposed in this chapter is not fundamentally different from the aggregation schemes described previously. We develop and describe the aggregation of the (CTP) with a view toward implementation, however. The aggregation method to be posed here is actually the same as that suggested by Lee and, given the appropriate model schema, is compatible with the structural aggregation procedure described by Geoffrion. Given a general partition of the node set, we describe the aggregation of the (CTP) explicitly in terms of the adjacency matrix associated with the (CTP). In addition to being a necessary preliminary for efficient implementation, this perspective also makes for a simpler, more lucid and intuitive explanation of the aggregation process as applied to network flow problems.

I. Preliminary Notation

It is necessary to establish notational conventions that will be used for the remainder of this work.

The original, unaggregated capacitated transshipment problem NP^0 is given below:

$$\text{Min } \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$\text{S.T. } \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b_i, \quad \text{all } i \in N$$

$$t_{ij} \leq x_{ij} \leq u_{ij}, \quad \text{all } (i,j) \in A$$

where: c_{ij} = cost on arc (i,j)

t_{ij} = lower bound on arc (i,j)

u_{ij} = upper bound on arc (i,j)

b_i = supply at node i (negative supply = demand)

x_{ij} = flow from node i to node j

N = set of nodes of NP^0 , indexed by i and/or j ,

$$|N| = n$$

A = set of arcs of NP^0 , indexed by (i,j) ,

$$|A| = a$$

The aggregate capacitated transshipment problem can be determined by specifying a partition of the node set and structural respecification maps for costs, supplies, upper and lower bounds. For a given partition of the node set, N^r , the resultant aggregate model, $\overline{NP}^r$, is given below. Note that in general an overbar indicates reference the aggregate model while the absence of an overbar indicates reference to the original model.

$$\text{Min } \sum_{(m,q) \in \overline{A}^r} \overline{c}_{mq}^r \overline{x}_{mq}^r$$

S.T.

$$\sum_{q: (m,q) \in \bar{A}^r} \bar{x}_{mq}^r - \sum_{q: (q,m) \in \bar{A}^r} \bar{x}_{qm}^r = \bar{b}_m^r, \text{ all } m \in \bar{N}^r$$

$$\bar{t}_{mq}^r \leq \bar{x}_{mq}^r \leq \bar{u}_{mq}^r, \text{ all } (m,q) \in \bar{A}^r$$

where: $N^r = \{N_m^r: m \in \bar{N}^r\}$ is the r^{th} partition of N

$\bar{N}^r = \{1, 2, \dots, n^r\}$ is the set of aggregate nodes of $\bar{NP}^r$

$\bar{A}^r = \{(\bar{A}_{mq}^r), (m,q) \in \bar{N}^r \times \bar{N}^r\}$ is the set of aggregate arcs of $\bar{NP}^r$

We shall use the respecification maps first defined by Balas [1], and shall refer to them collectively as $RS^{\min}$. This respecification is selected because it requires no multiplication or division, a requisite of computationally efficient primal network codes. Also, $\bar{NP}^r$ is a relaxation of NP^0 under this respecification. $RS^{\min}$ is composed of the structural respecification maps for costs, supplies, and upper and lower bounds denoted $RS1^{\min}$, $RS2^{\min}$, $RS3^{\min}$ and $RS4^{\min}$ respectively.

$$RS1^{\min}: \quad \bar{c}_{mq}^r = \min_{\substack{i \in N_m^r \\ j \in N_q^r}} c_{ij}$$

$$RS2^{\min}: \quad \bar{b}_m^r = \sum_{i \in N_m^r} b_i$$

$$RS3^{\min}: \quad \bar{u}_{mq}^r = \sum_{\substack{i \in N_m^r \\ j \in N_q^r}} u_{ij}$$

$$\begin{aligned} \text{RS4}^{\text{min}}: \quad \bar{t}_{mq}^r &= \sum_{\substack{i \in N_m^r \\ j \in N_q^r}} t_{ij} \end{aligned}$$

Also let:

$$k^r(i) = m, \text{ iff } i \in N_m^r$$

II. Aggregation Of The Capacitated Transshipment Problem

The aggregation procedure described in this section is equivalent to the procedure described in Lee [39] and is a particular instance of the "Structural Aggregation Procedure" described in Geoffrion [19]. (See Appendices A and B.) This version however, is easily visualized and facilitates simpler computer implementation. Since it requires the graph theoretic notion of the condensation of a directed graph, some preliminary definitions are in order.

Definition 2.1: The condensation of a directed graph (N,A) with respect to a partition N^r is the directed graph whose nodes are the subsets N_m^r and whose arcs are determined by the following rule: there is an arc from node N_m^r to node N_q^r if and only if in (N,A) there is at least one arc from a node $i \in N_m^r$ to a node of N_q^r .

Associated with each network problem is the directed graph (N,A) obtained simply by ignoring cost, supplies and bounds. Thus, we can meaningfully speak of "the directed graph associated with (CTP)." This shall be what we mean

when we refer to "the directed graph" or "the digraph." We will also find it convenient to refer to "the condensation of NP^0 with respect to N^r ." It is understood that this refers more precisely to the condensation of the directed graph (N,A) associated with NP^0 with respect to the partition N^r .

Aggregation Procedure

1. For a specified partition, N^r , form the condensation of NP^0 :

- a. Each subset of original nodes, N_m^r , is replaced by a single aggregate node designated m , $m = 1, 2, \dots, n^r$
- b. All original arcs (i,j) such that $i \in N_m^r$ and $j \in N_m^r$ are eliminated.
- c. If there is an arc $(i,j) \in A$ such that

$$i \in N_m^r, j \in N_q^r, m \neq q$$

then there is an (aggregate) arc (m,q) in $\overline{NP}^r$.

2. For all aggregate arcs $(m,q) \in \overline{A}^r$ of $\overline{NP}^r$ determine costs, lower and upper bounds using $RS1^{\min}$, $RS3^{\min}$ and $RS4^{\min}$.

3. For all aggregate nodes $m \in \overline{N}^r$ of $\overline{NP}^r$ determine supplies using $RS2^{\min}$.

It is important to note that the structural respecification maps can be any respecification maps. $RS1^{\min}$, $RS2^{\min}$, $RS3^{\min}$ and $RS4^{\min}$ need not necessarily be

the ones chosen.

For the appropriately defined model schema, step 1 of the aggregation procedure is equivalent to (1) Geoffrion's structural aggregation procedure and to (2) Lee's $R_{\text{sum}} \times \min$ replacement rule.

Proposition 2.1: Consider the model schema for NP^0 whose element graph is shown in Figure 2.1. Step 1. of the aggregation procedure is equivalent to the "Structural Aggregation Procedure" given in [19].

Proof:

Given a partition N^r and the model schema in Figure 2.1, apply the structural aggregation procedure for a particular subset N_m^r . Assume, without loss of generality, that $N_m^r = \{1, 2, \dots, p\}$ to simplify notation. The resultant aggregate entities of concern are, using Geoffrion's notation (See Appendix A.):

$N_{1/2/\dots/p}$

$A_{1/2/\dots/p, k}$ $k: i, k$ is an index of ARC and $i \in N_m^r$

$A_{k, 1/2/\dots/p}$ $k: k, i$ is an index of ARC and $i \in N_m^r$

$A_{1/2/\dots/p, 1/2/\dots/p}$

$C_{1/2/\dots/p, k}$ $k: i, k$ is an index of ARC and $i \in N_m^r$

$C_{k, 1/2/\dots/p}$ $k: k, i$ is an index of ARC and $i \in N_m^r$

$C_{1/2/\dots/p, 1/2/\dots/p}$

Specifying the structural respecification map for costs to be:

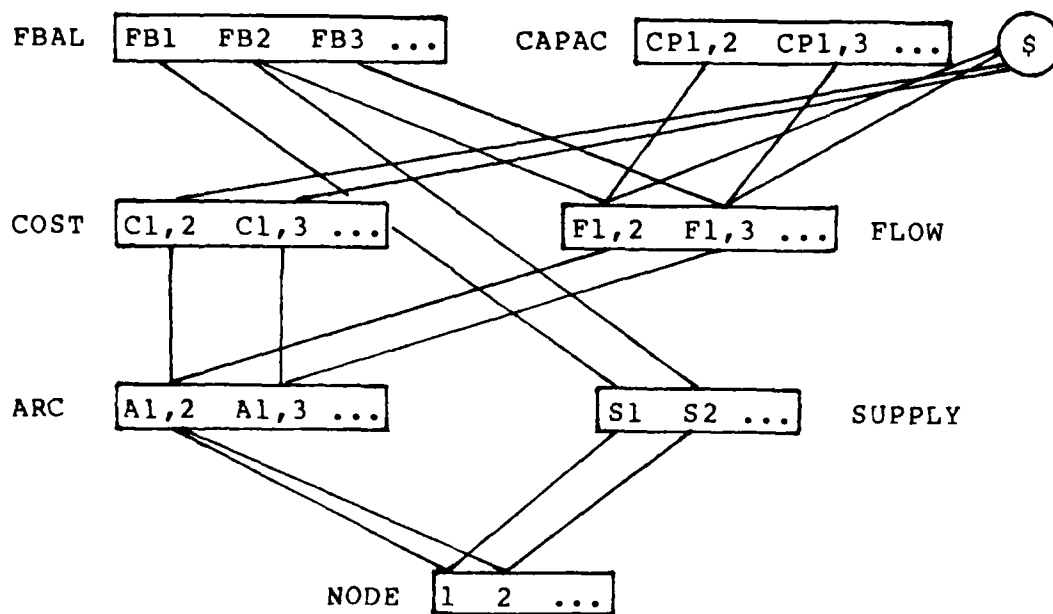


Figure 2.1: Element Graph For Model Schema Of NP^0

$$C_{kl} = \begin{cases} \min_{i \in N_m^r} C_{ki} & \text{for } l = 1/2/\dots/p \\ \min_{i \in N_m^r} C_{il} & \text{for } k = 1/2/\dots/p \\ +\infty & \text{for } k = l = 1/2/\dots/p \end{cases}$$

we obtain that:

1. the nodes $i \in N_m^r$ are replaced by the single aggregate node $1/2/\dots/p$
2. the arcs A_{ij} such that $i \in N_m^r, j \in N_m^r$ are replaced with the single aggregate arc $A_{1/2/\dots/p, 1/2/\dots/p}$ which has infinite cost.

Now, iteratively apply the structural aggregation procedure for each N_m^r in turn. According to 1. above each subset N_m^r is replaced by the aggregate node m . According to 2. above, all arcs with head and tail nodes in the same subset are effectively eliminated. What remains are arcs

$$A_{i_1/i_2/\dots/i_p, j_1/j_2/\dots/j_s}$$

such that $i_u \in N_m^r, u = 1, 2, \dots, p, j_v \in N_q^r, v = 1, 2, \dots, s, m = q$. Thus, the structural aggregation procedure iteratively applied to each subset N_m^r in conjunction with the structural respecification map for costs given above effectively produces the condensation of NP^0 with respect to N^r .

Proposition 2.2: For a specified partition N^r , the aggregation procedure, along with $RS1^{\min}, RS2^{\min}, RS3^{\min}$ and

$RS4^{\min}$, results in the same aggregate model as applying the " $R_{\text{sum} \times \text{min}}$ replacement rule" given by Lee [39].

Proof:

Using Lee's notation of Appendix B, $R_{\text{sum} \times \text{min}}$ has the following impact:

1. nodes $u \in t(q)$ are replaced by the single aggregate node q
2. arcs $(u,v) \in c(q,p)$ are replaced by the single aggregate arc (q,p) if $q \neq p$
3. arcs $(u,v) \in c(q,p)$ are eliminated if $q = p$
4. $l(q,p) = \sum_{(u,v) \in c(q,p)} l(u,v)$
5. $e(q,p) = \sum_{(u,v) \in c(q,p)} e(u,v)$
6. cost of $(q,p) = \text{Min cost of } (u,v)_{(u,v) \in c(q,p)}$

With the following correspondences:

$$(u,v) \Leftrightarrow (i,j)$$

$$(q,p) \Leftrightarrow (k,m)$$

$$t(q) \Leftrightarrow N_m^r$$

$$c(q,p) \Leftrightarrow A_{mq}^r$$

1. above $\Leftrightarrow$ Step 1a. of the aggregation procedure
2. above $\Leftrightarrow$ Step 1c. of the aggregation procedure
3. above $\Leftrightarrow$ Step 1b. of the aggregation procedure
4. above $\Leftrightarrow RS4^{\min}$
5. above $\Leftrightarrow RS3^{\min}$
6. above $\Leftrightarrow RS1^{\min}$

the result follows, i.e. $\overline{NP}^r = R_{\text{sum} \times \text{min}}(d_r, d_c)$ with $d_r = T$

and $d_c = C$.

Definition 2.2: The $n \times n$ adjacency matrix, γ^0 , associated with NP^0 is defined as

$$\gamma^0 = (\gamma_{ij}^0)$$

where:

$$\gamma_{ij}^0 = \begin{cases} 1 & \text{if } (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

Definition 2.3: Let $\tilde{N}^r = (\tilde{N}_i^r) = \{i_1, i_2, \dots, i_n\}$ be the unique ordered sequence associated with N^r such that

$$i_j \text{ preceeds } i_k \text{ if } i_j \in N_m^r, i_k \in N_q^r \\ \text{and } m < q$$

and

$$i_j \text{ preceeds } i_k \text{ if } i_j, i_k \in N_m^r \text{ and } i_j < i_k.$$

Definition 2.4: The partitioned adjacency matrix corresponding to N^r is defined as

$$\gamma^r = (\gamma_{ij}^r) \\ = I \gamma^0 I,$$

where

$I = (I_{ij})$ is an $n \times n$ permutation matrix with unit row vectors such that

$$I_{ij} = \begin{cases} 1 & \text{if } \tilde{N}_i^r = j \\ 0 & \text{otherwise} \end{cases}$$

and the ' indicates transposition. Also define the submatrices

$$\gamma_{mq}^r = (\gamma_{ij}^r) : (N_i^r, N_j^r) \in N_m^r \times N_q^r$$

Definition 2.5: The unique arc set partition corresponding to N^r is defined as

$$A^r = \{A_{mq}^r : m, q \in \bar{N}^r \times \bar{N}^r\}$$

where

$$A_{mq}^r = \{(i, j) : i \in N_m^r, j \in N_q^r\}$$

A^r is obviously the same arc set partition defined by Lee. With these new definitions however, A^r is easily obtained by direct matrix multiplication and interpretation of the partitioned adjacency matrix:

$$A_{mq}^r = \{(i, j) : I_{i.} \gamma^0 I_{.j} = 1, i \in N_m^r, j \in N_q^r\}$$

where $I_{i.}$ is the i^{th} row of I and $I_{.j}$ is the j^{th} column of I .

The aggregation procedure then can be viewed simply as:

1. replacing each subset N_m^r with an aggregate node m ,
2. replacing each nonempty subset A_{mq}^r ,
 $(m, q) \in \bar{N}^r \times \bar{N}^r, m \neq q$, with aggregate arc (m, q)
3. determining aggregate costs, supplies and bounds with $RS^{\min}$.

These ideas are illustrated in Example 2.1 below.

We shall find it not only convenient but in fact necessary to categorize the original arcs in order to:

1. have terminology available with which to refer to the different arc types for expository purposes,
2. carefully define the general disaggregation map of chapter four.
3. provide a notational convention for defining and generating sequences of aggregate problems.

The following definitions serve these purposes.

Definition 2.6: For a given partition N^r , an arc $(i,j) \in A$ shall be called an eliminated arc if $i,j \in N_m^r$

Definition 2.7: For a given partition N^r and for the adopted structural respecification map $RS1^{min}$, an arc (i^0, j^0) shall be called the representative arc of the nonempty subset A_{mq}^r , denoted R_{mq}^r , if

$$\min_{(i,j) \in A_{mq}^r} c_{ij} = c_{i^0, j^0}$$

The set of representative arcs shall be denoted R^r .

Definition 2.8: Any arc $(i,j) \in A_{mq}^r$, $m \neq q$, such that $(i,j) \notin R^r$ shall be called an absorbed arc.

The following example serves to clarify these definitions and concepts.

Example 2.1

Consider the capacitated transshipment problem taken

from [2] shown below in Table 2.1. This example shall be used throughout and will henceforth be referred to as the original network problem, NP^0 . For the particular partition:

$$N^1 = \{N_1^1, N_2^1, N_3^1, N_4^1, N_5^1\}$$

$$= \{\{2,3,4,6,11,12\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}\}$$

we have:

$$n^1 = 5$$

i:	1	2	3	4	5	6	7	8	9	10	11	12
$k^r(i)$:	2	1	1	1	2	1	2	3	4	5	1	1
N_i^r :	2	3	4	6	11	12	1	5	7	8	9	10

The adjacency matrix associated with NP^0 is

$$\gamma = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{matrix} & \left[\begin{array}{cccccccccccc} 1 & & & & 1 & & 1 & 1 & 1 & & & \\ & & 1 & & & 1 & & & 1 & & 1 & \\ & & & 1 & & & & & 1 & & 1 & \\ & & & & & & & 1 & & 1 & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & 1 & & & 1 \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \end{array} \right] \end{matrix}$$

The permutation matrix associated with N^1 is

Node	Supply	Arc		Cost	Bound	Bound
		i	j			
1	34	2	3	34	0	11
2	56	3	4	23	0	6
3	5	1	5	28	0	10
4	0	2	6	45	5	25
5	- 5	1	7	57	0	21
6	- 9	5	8	24	0	5
7	-18	1	8	56	0	7
8	-15	4	8	19	0	9
9	- 8	1	9	61	0	5
10	- 3	2	9	99	0	12
11	-21	6	9	48	0	3
12	-16	3	9	53	0	24
		3	10	26	0	8
		4	10	20	0	2
		2	11	14	10	23
		6	12	34	0	16

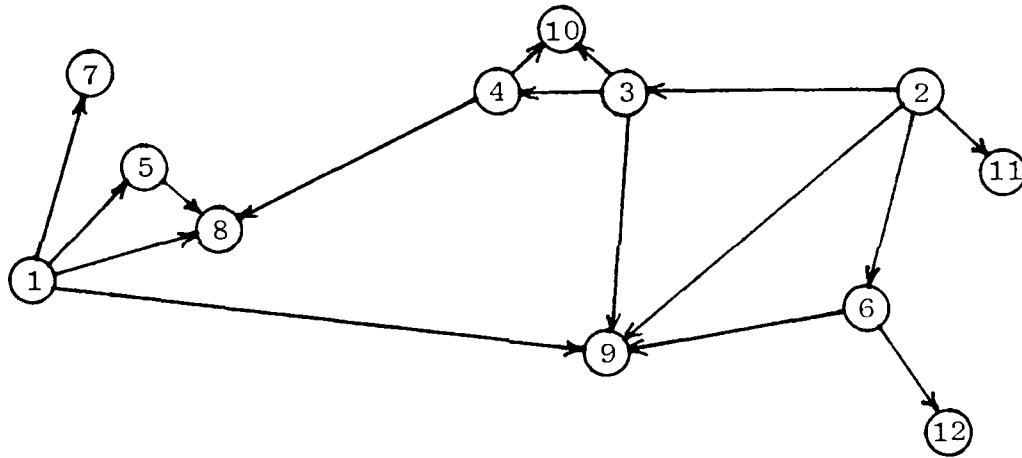


Table 2.1: Example Problem From Bradley, et. al. [2]

$$I =$$

	1	2	3	4	5	6	7	8	9	10	11	12
1		1										
2			1									
3				1								
4						1						
5											1	
6												1
7	1											
8					1							
9							1					
10								1				
11									1			
12										1		

Hence the partitioned adjacency matrix is

$$\gamma^1 =$$

i	$\tilde{N}_i^r$	2	3	4	6	11	12	1	5	7	8	9	10
1	2		1		1	1						1	
2	3			1								1	1
3	4										1		1
4	6						1					1	
5	11												
6	12												
7	1							1	1		1	1	
8	5										1		
9	7												
10	8												
11	9												
12	10												

The partition of the arc set is easily seen to correspond directly to the submatrices of the partitioned adjacency matrix:

$$\begin{aligned} \gamma_{11}^1 &\Leftrightarrow A_{11}^1 = \{(2,3), (2,6), (2,11), (3,4), (6,12)\} \\ \gamma_{13}^1 &\Leftrightarrow A_{13}^1 = \{(4,8)\} \\ \gamma_{14}^1 &\Leftrightarrow A_{14}^1 = \{(2,9), (3,9), (6,9)\} \\ \gamma_{15}^1 &\Leftrightarrow A_{15}^1 = \{(3,10), (4,10)\} \\ \gamma_{22}^1 &\Leftrightarrow A_{22}^1 = \{(1,5), (1,7)\} \\ \gamma_{23}^1 &\Leftrightarrow A_{23}^1 = \{(1,8), (5,8)\} \\ \gamma_{24}^1 &\Leftrightarrow A_{24}^1 = \{(1,9)\} \end{aligned}$$

all other $A_{mq}^1 = \{\emptyset\}$

Applying the aggregation procedure directly to the partitioned adjacency matrix yields the following:

1. The subsets $N_1^1, N_2^1, N_3^1, N_4^1, N_5^1$ are replaced by aggregate nodes 1, 2, 3, 4 and 5 respectively. Thus $\bar{N}^1 = \{1, 2, 3, 4, 5\}$.
2. The nonzero submatrices $\gamma_{mq}^1, m = q$, correspond directly to the nonempty subsets A_{mq}^1 which determine the aggregate arcs. Thus $A^1 = \{(1,3), (1,4), (1,5), (2,3), (2,4)\}$
3. Applying $RS^{\min}$ completes the structural respecification of the aggregate network, $\bar{NP}^1$, e.g.

$$\begin{aligned} RS1^{\min}: \bar{c}_{14}^1 &= \min_{\substack{i \in N_1^1 \\ j \in N_4^1}} c_{ij} \\ &= \min \{c_{29}, c_{39}, c_{69}\} \\ &= \min \{99, 53, 48\} = 48 \end{aligned}$$

$$\begin{aligned} RS2^{\min}: \bar{b}_2^1 &= \sum_{i \in N_2^1} b_i \\ &= b_1 + b_5 + b_7 = 34 - 5 - 18 = 11 \end{aligned}$$

$$\begin{aligned} RS3^{\min}: \bar{u}_{14}^1 &= \sum_{\substack{i \in N_1^1 \\ j \in N_4^1}} u_{ij} \\ &= u_{29} + u_{39} + u_{69} \\ &= 12 + 24 + 3 = 39 \end{aligned}$$

$$\begin{aligned} RS4^{\min}: \bar{t}_{14}^1 &= \sum_{\substack{i \in N_1^1 \\ j \in N_4^1}} t_{ij} \end{aligned}$$

$$\begin{aligned}
&= t_{29} + t_{39} + t_{69} \\
&= 0 + 0 + 0 = 0
\end{aligned}$$

The aggregate problem $\overline{NP}^1$ is shown in Figure 2.2.

As per definition 2.6, eliminated arcs are those arcs corresponding to the nonzero submatrices γ_{mm}^1 along the diagonal of the partitioned adjacency matrix, i.e.

$$\gamma_{11}^1 \Leftrightarrow A_{11}^1 = \{(2,3), (2,6), (2,11), (3,4), (6,12)\}$$

$$\gamma_{22}^1 \Leftrightarrow A_{22}^1 = \{(1,5), (1,7)\}$$

and the set of eliminated arcs is simply $A_{11}^1 \cup A_{22}^1$.

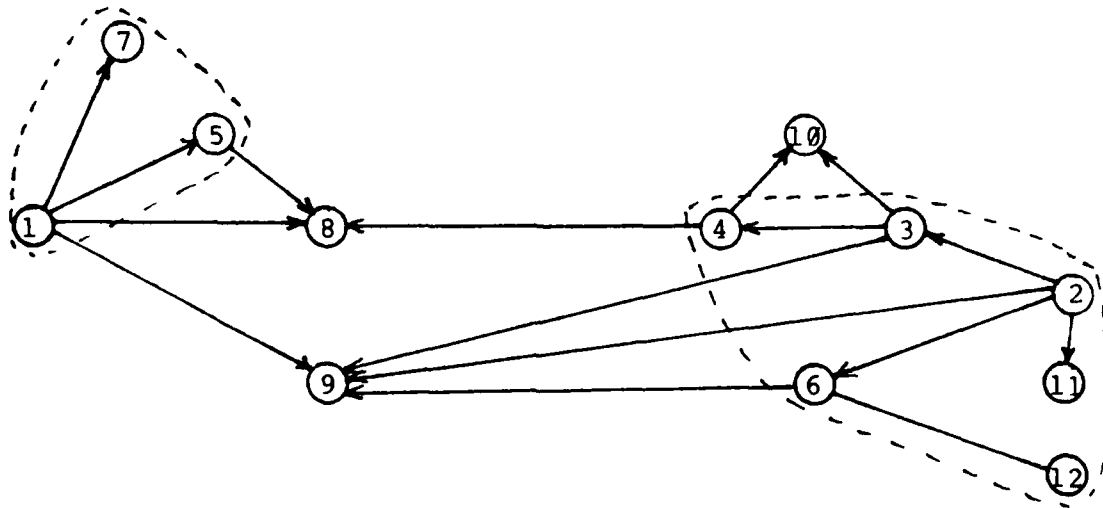
Intuitively then, we can view the aggregation process as:

1. collapsing the nodes of each subset N_m^r into a single aggregate node, m ,
2. removing all of the arcs from NP^0 except the representative arcs,
3. and "adjusting" (increasing) the bounds on the representative arcs according to $RS3^{\min}$, $RS4^{\min}$.

This conceptualization will prove useful in generating sequences of aggregate problems by node set partition refinement. For purposes of implementation, we will only have to keep track of the set of representative arcs and the associated head and tail nodes in order to define the digraph associated with $\overline{NP}^r$. The representative arc $(i,j) \in R^r$ can be thought of as "generating" aggregate arc $(k^r(i), k^r(j)) \in \bar{A}^r$. Each successive refinement of a node

NP^0 :

Partition N^1 indicated by - - -



$\overline{NP}^1$:

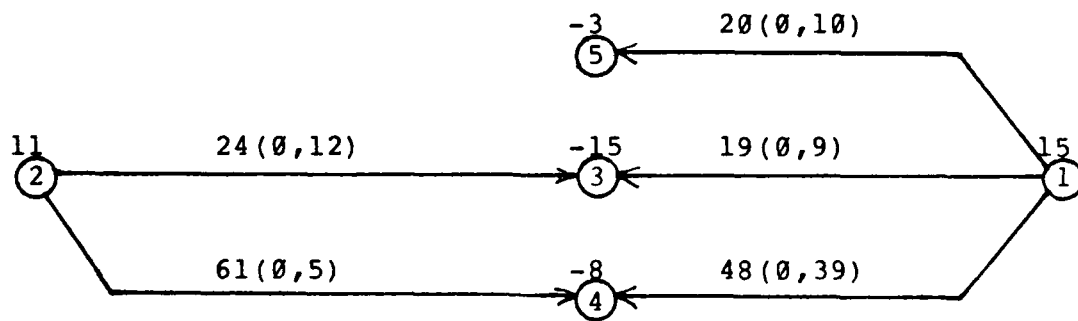


Figure 2.2: NP^0 And $\overline{NP}^1$ For Example 2.1

subset simply adds more arcs to the set R^I and an additional node to the set $\bar{N}^I$. These ideas will be formalized in the next chapter.

III. A Measure Of The Scale Of Aggregation

In the previous section we demonstrated that an aggregate network can be generated by specifying a partition of the node set along with the structural respecification maps for cost, supplies and bounds. One characteristic of an aggregate network is its size relative to that of the original network. In this section we propose a measure of the scale of aggregation and develop its matrix algebra formulation. In the next section, we show a potential use of this measure in aggregation design. First however, we investigate the combinatorics of node set partitions.

Definition 2.9: The Stirling Number Of The Second Kind is defined as the number of ways of partitioning n elements into exactly r (nonempty) subsets, and is denoted

$$\begin{bmatrix} n \\ r \end{bmatrix}$$

Stirling Numbers Of The Second Kind are generated by the recursive formula:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 1, \quad \begin{bmatrix} n \\ r \end{bmatrix} = \begin{bmatrix} n-1 \\ r \end{bmatrix} r + \begin{bmatrix} n-1 \\ r-1 \end{bmatrix}$$

Thus, it is easy to see that the number of possible

partitions of the node set is given by

$$\sum_{m=1}^n \begin{bmatrix} n \\ m \end{bmatrix}$$

Definition 2.10: Given NP^0 and N^r , the scale of aggregation shall be defined as

$$\begin{aligned} M^r &= M_1^r + M_2^r + M_3^r \\ &= \frac{(n - n^r)}{(n - 1)} + \frac{\sum_{m=1}^{n^r} |A_{mm}^r|}{a} + \frac{\sum_{(m,q) \in \bar{A}^r} (|A_{mq}^r| - 1)}{a} \end{aligned}$$

The interpretation of M^r is straightforward:

$$M_1^r = \frac{\text{number of nodes eliminated}}{\text{number of original nodes} - 1}$$

$$M_2^r = \frac{\text{number of eliminated arcs}}{\text{number of original arcs}}$$

$$M_3^r = \frac{\text{number of absorbed arcs}}{\text{number of original arcs}}$$

Note that for $N^r = N_1^r = \{1, 2, \dots, n\}$:

$$M_1^r = \frac{n - 1}{n - 1} = 1$$

$$M_2^r = \frac{a}{a} = 1$$

$$M_3^r = \frac{0}{a} = 0$$

$$\Rightarrow M^r = 2$$

and for $N^r = \{\{1\}, \{2\}, \dots, \{n^r\}\}$:

$$M_1^r = \frac{0}{n - 1} = 0$$

$$M_2^r = \frac{0}{a} = 0$$

$$M_3^r = \frac{0}{a} = 0$$

$$\Rightarrow M^r = 0.$$

Thus we have that $0 \leq M^r \leq 2$. The following definitions facilitate calculation of M^r by matrix algebra.

Definition 2.11: The $n^r \times n$ binary matrix N^r shall be called the node assignment matrix and is defined as

$$N^r = (N_{ij}^r)$$

where

$$N_{ij}^r = \begin{cases} 1 & \text{if node } j \in N_i^r \\ 0 & \text{otherwise} \end{cases}$$

Note that each column of N^r is a unit vector.

Definition 2.12: Define the $n^r \times n^r$ upper triangular binary matrix $O^r(\cdot) = (O_{ij}^r(\cdot))$ as follows:

$$O_{ij}^r = \begin{cases} 0 & \text{if } i \geq j \text{ or } (\cdot)_{ij} = 0 \\ 1 & \text{if } i < j \text{ and } (\cdot)_{ij} \neq 0 \end{cases}$$

where $(\cdot)_{ij}$ is the i, j^{th} component of $(\cdot)$.

Definition 2.13: Let the $n^r \times n^r$ matrix p^r be defined as:

$$p^r = N^r \gamma^0 N^{r'} - O(N^r \gamma^0 N^{r'})$$

It is easy to verify that:

1. $(N^r \gamma^0 N^{r'})_{ij} = |A_{ij}^r|$
2. p_{ii}^r is the number of eliminated arcs in A_{ii}^r

3. p_{ij}^r is the number of absorbed arcs in A_{ij}^r , $i \neq j$

4. $\underline{1} p^r \underline{1}' = (M_2^r + M_3^r)a$

where $\underline{1}$ is a $1 \times n^r$ vector of 1's

5. $M^r = (n - n^r)/(n - 1) + (\underline{1} p \underline{1}')/a$

Example 2.2

Consider the problem given in Example 2.1. Then:

$$N^1 = \begin{array}{c|cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline 1 & & 1 & 1 & 1 & & 1 & & & & & 1 & 1 \\ 2 & 1 & & & & 1 & & 1 & & & & & \\ 3 & & & & & & & & 1 & & & & \\ 4 & & & & & & & & & 1 & & & \\ 5 & & & & & & & & & & 1 & & \end{array}$$

$$N^1 \gamma^0 N^{1'} = \begin{array}{ccccc} 5 & 0 & 1 & 3 & 2 \\ 0 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

Note for example that $(N^1 \gamma^0 N^{1'})_{11} = 5 = |A_{11}^1|$

$$(N^1 \gamma^0 N^{1'})_{24} = 1 = |A_{24}^1|$$

$$O(N^1 \gamma^0 N^{1'}) = \begin{array}{ccccc} 0 & 0 & 1 & 1 & 1 \\ & 0 & 1 & 1 & 0 \\ & & 0 & 0 & 0 \\ & & & 0 & 0 \\ & & & & 0 \end{array}$$

$$p^1 = \begin{array}{ccccc} 5 & 0 & 0 & 2 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

Note for example that $p_{11}^1 = |A_{11}^1|$ = the number of arcs eliminated in A_{11}^1 and $p_{14}^1 = |A_{14}^1| - 1$ = the number of absorbed arcs in A_{14}^1 .

$$\begin{aligned}
M^1 &= (n - n^r)/(n - 1) + (\underline{1} \ P^1 \ \underline{1}')/a \\
&= (12 - 5)/(12 - 1) + 11/16 \\
&= 0.636 + 0.687 = 1.323
\end{aligned}$$

It is also interesting to compare the relative magnitudes of

1. M_1^r and $(M_2^r + M_3^r)$
2. M_2^r and M_3^r

For $(M_2^r + M_3^r)/(M_1^r) > 1$ the reduction in size from NP^0 to $\overline{NP}^r$ results predominately from the elimination of arcs.

When this same ratio is less than one the reduction is due predominately to the reduction in the number of nodes.

For $M_2^r = 0$, $M_3^r > 0$, the aggregation might be termed a "pure absorption type" aggregation since no arcs were eliminated but some were absorbed. Conversely, for $M_2^r > 0$ and $M_3^r = 0$, the aggregation might be called a "pure elimination type." When both of these terms are strictly positive, we have a "mixed type" aggregation with either absorption ($M_3^r > M_2^r$) or elimination ($M_2^r > M_3^r$) dominating.

For the above example we see that the reduction in size is due almost equally to reductions in the numbers of nodes and arcs. Since $M_2^1 = 7/16 > 4/16 = M_3^1$, this is a mixed aggregation with elimination dominating.

The measure M^r can also be used as a "gauge" in Lee's aggregation framework, thus establishing equivalence classes among the possible partitions of N .

IV. Use of M^r in Aggregation Design

In addition to providing descriptive information concerning the scale of aggregation, M^r can be used to rigorously define aggregation design problems. One such problem is:

Given that $\overline{NP}^r$ is to have n^r nodes, find the partition that results in the smallest aggregate problem, i.e. find the partition that maximizes M^r .

Alternatively stated, we seek the partition that 1) results in the specified number of nodes in $\overline{NP}^r$, and 2) results in the fewest number of arcs. Thus we wish to solve the aggregation design problem (ADP):

$$\max_{N^r \in \psi} M^r = \frac{n - n^r}{n - 1} + \frac{1}{a} p^r \underline{1}'$$

S.T.

$$|N^r| = n^r$$

where $\psi = \{\text{all possible partitions of } N\}$. Note that the number of partitions satisfying the constraint is $\begin{bmatrix} n \\ n^r \end{bmatrix}$. Since $(n - n^r)/(n - 1)$ and $|A|$ are positive constants, we may write (ADP) as:

$$\max_{N^r \in \psi} \underline{1}' p \underline{1}' = \underline{1}' [N^r \gamma^0 N^r - 0^r (N^r \gamma^0 N^r)] \underline{1}'$$

S.T.

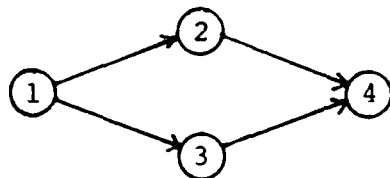
$$\sum_{i=1}^{n^r} N_{ij}^r = 1, \quad j = 1, 2, \dots, n$$

$$N_{ij}^r = 0, 1, \quad i = 1, 2, \dots, n^r \\ j = 1, 2, \dots, n$$

where the decision variables N_{ij}^r are the elements of N^r . The matrix $O(\cdot)$ can be handled implicitly by introducing additional binary variables as shown in the following example.

Example 2.3

Consider the simple network below (capacities, costs and supplies are not shown):



Suppose we are interested in creating the aggregate problem $\overline{NP}^1$ with $n^1 = 2$ aggregate nodes and with the fewest number of arcs. Then:

$$N^1 \gamma^0 N^1 = \begin{bmatrix} N_{11} & N_{12} & N_{13} & N_{14} \\ N_{21} & N_{22} & N_{23} & N_{24} \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} N_{11} & N_{21} \\ N_{12} & N_{22} \\ N_{13} & N_{23} \\ N_{14} & N_{24} \end{bmatrix}$$

By the definition of $O^1(\cdot)$ we have:

$$O_{12}^1 = \begin{cases} 1 & \text{if } (N^1 \gamma^0 N^1)_{12} > 0 \\ 0 & \text{if } (N^1 \gamma^0 N^1)_{12} = 0 \end{cases}$$

$$O_{21}^1 = \begin{cases} 1 & \text{if } (N^1 \gamma^0 N^1)_{21} > 0 \\ 0 & \text{if } (N^1 \gamma^0 N^1)_{21} = 0 \end{cases}$$

These conditions are easily handled by introducing two additional binary variables, y_{12} and y_{21} , as follows in problem (QP) below:

$$\begin{aligned} \text{Max } z = & [(N_{11}N_{12} + N_{13}(N_{11} + N_{12}) + N_{14}(N_{12} + N_{13})) \\ & + [(N_{21}N_{22} + N_{23}(N_{21} + N_{22}) + N_{24}(N_{22} + N_{23})) \\ & + [(N_{11}N_{22} + N_{23}(N_{11} + N_{12}) + N_{24}(N_{12} + N_{13}) - y_{12}] \\ & + [(N_{21}N_{12} + N_{13}(N_{21} + N_{22}) + N_{14}(N_{22} + N_{23}) - y_{21}] \end{aligned}$$

S.T.

$$N_{11} + N_{12} = 1$$

$$N_{12} + N_{22} = 1$$

$$N_{13} + N_{23} = 1$$

$$N_{14} + N_{24} = 1$$

$$0 \leq [N_{11}N_{22} + N_{23}(N_{11} + N_{12}) + N_{24}(N_{12} + N_{13})] \leq a \cdot y_{12}$$

$$0 \leq [N_{21}N_{12} + N_{13}(N_{21} + N_{22}) + N_{14}(N_{22} + N_{23})] \leq a \cdot y_{21}$$

$$\text{all } N_{ij}, y_{ij} = 0, 1$$

Although we have explicitly formulated the stated aggregation design problem as (QP), we observe that:

1. In general, (QP) has $n + 2[n^r n^r - n^r]$ constraints and $n^r(n^r + n - 1)$ variables.
2. The number of possible partitions of N with n^r subsets is $\begin{bmatrix} n \\ n^r \end{bmatrix}$.
3. The problem (QP) is a quadratic assignment problem and is thus NP-complete

To complete the example above, the table below

explicitly enumerates the seven possible partitions of N with two subsets.

<u>Partition</u>	<u>N</u>	<u>z</u>
{1}{2,3,4}	1 0 0 0 0 1 1 1	4
{2}{1,3,4}	0 1 0 0 1 0 1 1	3
{3}{1,2,4}	0 0 1 0 1 1 0 1	3
{4}{1,2,3}	0 0 0 1 0 1 1 1	4
{1,2}{3,4}	1 1 0 0 0 0 1 1	4
{1,3}{2,4}	1 0 1 0 0 1 0 1	3
{1,4}{2,3}	1 0 0 1 0 1 1 0	3

Table 2.2: Enumeration Of Partitions For Example 2.3

CHAPTER THREE

SEQUENCES OF AGGREGATE PROBLEMS

In this chapter we shall be concerned with generating sequences of aggregate problems. Specifically, we shall consider the sequence $\overline{NP}^r$, $r = 1, 2, \dots, q$ where $\overline{NP}^{r+1}$ is created from $\overline{NP}^r$ by refining a single subset of the partition N^r into two subsets.

We further characterize the arcs of the original network and describe a general process whereby $\overline{NP}^{r+1}$ is generated from $\overline{NP}^r$ as necessary prerequisites to the development of the general disaggregation map and complete algorithm to be described in subsequent chapters. Examples are included throughout.

We shall find it necessary henceforth to adopt a slight change in notation because of typographical considerations and to avoid confusion which might result from complicated, compound subscripts and superscripts. When such subscripts and superscripts are required and possibility of confusion exists, we shall instead use the parenthetical notation (a,b,c) where a is the superscript, b is the first position of the subscript and c is the second position of the subscript. We may also use the notation (a,b) and (a) when there is a one positional subscript and when there is no subscript, respectively. For example, A_{mq}^r is equivalent to $A(r,m,q)$, N_m^r is equivalent to $N(r,m)$ and N^r is equivalent to

$N(r)$. The parenthetical notation will be used interchangeably with the original notation but we shall endeavor to use subscripts and superscripts when there is no danger of confusion.

I. Further Characterization Of Original Arcs

This section presents definitions that complete the categorization of the original arcs that was begun in chapter two with Definitions 2.5, 2.6, 2.7 and 2.8. We shall consider relationships between aggregate problem $\overline{NP}^r$, generated from partition N^r , and aggregate problem $\overline{NP}^{r+1}$ generated from partition N^{r+1} , where N^{r+1} is obtained by refining a single subset of N^r into two subsets, i.e.

Before refinement:

$$N^r = \{N_1^r, N_2^r, \dots, N_m^r, \dots, N(r, n^r)\}$$

After refinement:

$$N^{r+1} = \{N_1^{r+1}, N_2^{r+1}, \dots, N(r+1, n^{r+1})\}$$

$$\text{where: } N_q^r = N_q^{r+1}, q \neq m$$

$$N_m^r = N_m^{r+1} \cup N(r+1, n^{r+1})$$

$$N_m^{r+1} \cap N(r+1, n^{r+1}) = \{\emptyset\}$$

We shall refer to such a refinement as a simple refinement (of N_m^r). More general refinements can be easily accommodated by iterative application of simple refinement.

We begin by stating, without comment, Definitions 3.1 through 3.6. An example and discussion follow.

Definition 3.1: An arc $(i,j) \in A$ shall be called a reinstated arc or an arc of type P if $(i,j) \notin R^r$ and $(i,j) \in R^{r+1}$. Denote the set of reinstated arcs as P^{r+1} .

Definition 3.2: An arc $(i,j) \in P^{r+1}$ shall be of type P-E if $(i,j) \in A_{mm}^r$ for some $m \in N^r$. Denote the set of P-E arcs as P_e^{r+1} .

Definition 3.3: An arc $(i,j) \in P^{r+1}$ shall be of type P-A if $(i,j) \in A_{mq}^r$, $m \neq q$, $|A_{mq}^r| > 1$. Denote the set of P-A arcs as P_a^{r+1} .

Definition 3.4: The representative arc $R_{mq}^r = (i,j)$ of subset A_{mq}^r is said to have been adjusted, or is an arc of type M, if under $RS3^{\min}$ and $RS4^{\min}$

- i. $\bar{t}(r+1, k^{r+1}(i), k^{r+1}(j)) < \bar{t}(r, k^r(i), k^r(j))$
- or ii. $\bar{u}(r+1, k^{r+1}(i), k^{r+1}(j)) < \bar{u}(r, k^r(i), k^r(j))$
- or iii. both i. and ii. hold. Let M^{r+1} be the set of adjusted arcs.

Definition 3.5: The pair of arcs $(g,h) \in M^{r+1}$ and $(i,j) \in P_a^{r+1}$ are called a corresponding adjusted-reinstated pair, a P-M pair or simply a corresponding pair if $(g,h) \in A_{mq}^r$ and $(i,j) \in A_{mq}^r$.

Definition 3.6: For the corresponding adjusted-reinstated

pair $(g,h) \in M^{r+1}$ and $(i,j) \in P_a^{r+1}$ define the point-to-point mapping $I(\cdot, \cdot)$ as $I(g,h) = (i,j)$, $I(i,j) = (g,h)$ and the set-to-set mapping $I(\cdot)$ as

$$I(M) = \{(i,j): I(g,h) = (i,j) \text{ for } (g,h) \in M, M \subseteq M^{r+1}\}$$

$$I(P) = \{(g,h): I(i,j) = (g,h) \text{ for } (i,j) \in P, P \subseteq P_a^{r+1}\}$$

$$I(\emptyset) = \{\emptyset\}$$

Remarks:

Remark 3.1: It should be noted that $P^{r+1} = R^{r+1} - R^r$.

Remark 3.2: We define $I(i,j) = \{\emptyset\}$ if (i,j) is not an arc of a corresponding pair. Also, the argument of I unambiguously determines whether we are referring to I as a point-to-point mapping or a set-to-set mapping.

Remark 3.3: It is apparent that $I(\bigcup_i M_i) = \bigcup_i I(M_i)$ for arbitrary subsets in the domain of I .

Example 3.1: Consider NP^0 and N^1 of Example 2.1. Let N^2 be obtained by refining N_1^1 into $N_1^2 = \{2,3,6,11\}$ and $N_6^2 = \{4,12\}$, i.e.

$$\begin{aligned} N^2 &= \{N_1^2, \dots, N_6^2\} \\ &= \{\{2,3,6,11\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}, \{4,12\}\} \end{aligned}$$

We have that:

1. $R^1 = \{(4,8), (6,9), (4,10), (5,8), (1,9)\}$
 $R^2 = \{R_{14}^2, R_{15}^2, R_{16}^2, R_{23}^2, R_{24}^2, R_{63}^2, R_{65}^2\}$
 $= \{(6,9), (3,10), (3,4), (5,8), (1,9), (4,8), (4,10)\}$
 $P^2 = R^2 - R^1 = \{(3,4), (3,10)\}$
2. $P_e^2 = \{(3,4)\}$ since $(3,4) \notin R^1$, $(3,4) \in R^2$

and $(3,4) \in A_{11}^1$.

3. $P_a^2 = \{(3,10)\}$ since $(3,10) \in A_{15}^1$, $|A_{15}^1| = 2$.
4. $M^2 = \{(4,10)\}$ since $(4,10) \in R^1$, $(4,10) \in R^2$
and $\bar{u}_{65}^2 = 2 < 10 = \bar{u}_{15}^1$.
5. $(3,10) \in P_a^2$ and $(4,10) \in M^2$ are a corresponding
adjusted-reinstated pair since $(3,10), (4,10) \in A_{15}^1$
6. $I((3,10)) = (4,10)$ and $I((4,10)) = (3,10)$.

As the name implies, reinstated arcs are those representative arcs that correspond to the "new" aggregate arcs in $\overline{NP}^{r+1}$. Such an arc could have been either an eliminated arc or an absorbed arc in $\overline{NP}^r$. If it was an eliminated arc in $\overline{NP}^r$, it is designated type P-E (for "put back in, previously eliminated"). If it was an absorbed arc, it is of type P-A (for "put back in, previously absorbed").

Consider the adjusted arc $(4,10)$ with the following characteristics:

$$\begin{aligned} (4,10) \in A_{15}^1, R_{15}^1 &= (4,10) \in R^1, (k^1(4), k^1(10)) \\ &= (1,5), \bar{u}_{15}^1 = 10 \\ (4,10) \in A_{65}^2, R_{65}^2 &= (4,10) \in R^2, (k^2(4), k^2(10)) \\ &= (6,5), \bar{u}_{65}^2 = 2 \end{aligned}$$

Original arc $(4,10)$ is used to "generate" aggregate arc $(k^1(4), k^1(10)) = (1,5)$ in $\overline{NP}^1$ and $(k^2(4), k^2(10)) = (6,5)$ in $\overline{NP}^2$. In essence, we view $(1,5)$ in $\overline{NP}^1$ as the "same" arc in $\overline{NP}^2$ but with a change in designation and with its bounds

"adjusted." The impact of refining N_1^1 into N_1^2 and N_6^2 was, in part, to

1. to refine A_{15}^1 into A_{15}^2 and A_{65}^2
2. to adjust (reduce) the bound(s) of the aggregate arc generated by representative arc (4,10)
3. to create a "new" arc for $\overline{NP}^2$ generated by the representative arc of the "new" subset A_{15}^2 , i.e. by $R_{15}^2 = (3,10)$.

The Definitions 3.1 through 3.6 are intended to capture these characteristics. We note the following:

1. If there is an adjusted arc, then there is necessarily a corresponding reinstated arc.
2. If there is a reinstated arc, then there may or may not be a corresponding adjusted arc. If the reinstated arc of $\overline{NP}^{r+1}$ was previously an eliminated arc, then there is no corresponding adjusted arc. If the reinstated arc was previously an absorbed arc, then there is necessarily a corresponding adjusted arc.

3. Some obvious relationships are:

$$P^{r+1} = P_a^{r+1} \cup P_e^{r+1}$$

$$I(M^{r+1}) = P_a^{r+1}$$

$$I(P_a^{r+1}) = M^{r+1}$$

$$I(P_e^{r+1}) = \{\emptyset\}$$

II. Procedure To Create $\overline{NP}^{r+1}$ From $\overline{NP}^r$

The previous section laid the ground work for a general procedure to create a sequence of aggregate network problems. Each successive aggregate network is created by a simple refinement of a subset of N^r into two subsets. By iteratively applying such a procedure, a sequence of aggregate network problems is created. In $n - n^r$ applications of the procedure NP^0 is obtained. This procedure, which we shall refer to as the " $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure," will be incorporated into the general disaggregation map and the complete algorithm described in subsequent chapters. We first describe the procedure and then illustrate its application to the preceding example.

We assume that NP^0 , $RS^{\min}$, N^r and N^{r+1} are given with N^{r+1} obtained from N^r by a simple refinement.

The $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ Procedure

The procedure proceeds basically as follows:

Step 0: Initialization.

Step 1: Add aggregate node n^{r+1} , make necessary adjustments to supplies and define $k^{r+1}(i)$.

Step 2: Add reinstated arcs that were previously eliminated and specify costs and bounds.

Step 3: Add reinstated arcs that were previously absorbed, adjust costs and bounds of reinstated-adjusted arc pairs leaving the "new" aggregate nodes 1 and n^{r+1} .

Step 4: Repeat Step 3 for reinstated-adjusted arc pairs entering the "new" aggregate nodes 1 and n^{r+1} .

The procedure is rigorously defined below and is followed by an example.

Procedure $NP^r \rightarrow NP^{r+1}$

For $N_m^r = N_m^{r+1} \cup N(r+1, n^{r+1})$:

Step 0: Set: $N^{r+1} = N^r \cup \{n^{r+1}\}$

$$A^{r+1} = A^r$$

$$R^{r+1} = R^r$$

$$M^{r+1} = p_a^{r+1} = p_e^{r+1} = \{\emptyset\}$$

Step 1: A:

$$k^{r+1}(i) = \begin{cases} k^r(i), & i \notin N(r+1, n^{r+1}) \\ n^{r+1}, & i \in N(r+1, n^{r+1}) \end{cases}$$

B:

$$\bar{b}_q^{r+1} = \begin{cases} \bar{b}_q^r, & q \neq m, q \neq n^{r+1} \\ \sum b_i, & q = n^{r+1} \\ & i \in N(r+1, n^{r+1}) \\ \sum b_i, & q = m \\ & i \in N(r+1, m) \end{cases}$$

Step 2: A: Refine subset A_{mm}^r into

1. $A(r+1, m, m) = \{(i, j) \in A: i, j \in N(r+1, m)\}$
2. $A(r+1, n^{r+1}, n^{r+1}) = \{(i, j) \in A: \\ i, j \in N(r+1, n^{r+1})\}$
3. $A(r+1, m, n^{r+1}) = \{(i, j) \in A: \\ i \in N(r+1, m), j \in N(r+1, n^{r+1})\}$

4. $A(r+1, n^r+1, m) = \{(i, j) \in A:$
 $i \in N(r+1, n^r+1), j \in N(r+1, m)\}$

B: For nonempty subset $A(r+1, m, n^r+1)$:

1. Find the representative arc
 $R(r+1, m, n^r+1)$
2. Set $R^{r+1} = R^{r+1} \cup R(r+1, m, n^r+1)$
 $p_e^{r+1} = p_e^{r+1} \cup R(r+1, m, n^r+1)$
 $A^{r+1} = A^{r+1} \cup \{(m, n^r+1)\}$
3. Determine respecified arc attributes
 $\bar{c}(r+1, m, n^r+1)$ using $RS1^{\min}$
 $\bar{u}(r+1, m, n^r+1)$ using $RS3^{\min}$
 $\bar{t}(r+1, m, n^r+1)$ using $RS4^{\min}$

C: For nonempty subsets $A(r+1, n^r+1, m)$,
 repeat Step 2B with indices reversed, i.e.
 replace m, n^r+1 with n^r+1, m

Step 3: Set $h = 1$.

Step 4: If $h = m$, go to Step 6, else go to Step 4A.

A: Refine subset $A(r, m, h)$ into

1. $A(r+1, m, h)$
2. $A(r+1, n^r+1, h)$
3. If $|A(r+1, m, h)| = |A(r+1, n^r+1, h)| = \emptyset$,
 GO TO Step 5.

B: Determine representative arcs

1. $R(r+1, m, h) = (g, h)$
2. $R(r+1, n^r+1, h) = (i, j)$

C: Calculate

1. $q_1 = |A(r, m, h)| - |A(r+1, m, h)|$
2. $q_2 = |A(r, m, h)| - |A(r+1, n^{r+1}, h)|$

D: Redesignate arcs if necessary

1. If $q_1 = 0$ then:
 If $(k^{r+1}(g), k^{r+1}(h)) \neq (k^r(g), k^r(h))$
 then:

$$\bar{A}^{r+1} = \bar{A}^{r+1} - \{(k^r(g), k^r(h))\}$$

$$\cup \{(k^{r+1}(g), k^{r+1}(h))\}$$

GO TO Step 5.

Else GO TO Step 4D2.

2. If $q_2 = 0$ then:
 If $(k^{r+1}(i), k^{r+1}(j)) \neq (k^r(i), k^r(j))$
 then:

$$\bar{A}^{r+1} = \bar{A}^{r+1} - \{(k^r(i), k^r(j))\}$$

$$\cup \{(k^{r+1}(i), k^{r+1}(j))\}$$

GO TO Step 5.

Else GO TO Step 4E.

- E:
1. If $0 < q_1 < |A(r, m, h)|$ and $(g, h) \in R^{r+1}$
 then:
 Set: $M^{r+1} = M^{r+1} \cup \{(g, h)\}$
 $\bar{A}^{r+1} = \bar{A}^{r+1} \cup \{(k^{r+1}(g), k^{r+1}(h))\}$
 GO TO Step 4E3
 - Else GO TO Step 4E2.
 2. If $0 < q_1 < |A(r, m, h)|$ and $(g, h) \notin R^{r+1}$
 then:
 Set: $R^{r+1} = R^{r+1} \cup \{(g, h)\}$

$$P_a^{r+1} = P_a^{r+1} \cup \{(g, h)\}$$

$$\bar{A}^{r+1} = \bar{A}^{r+1} \cup \{(k^{r+1}(g), k^{r+1}(h))\}$$

3. Determine respecified arc attributes

$$\bar{c}(r+1, m, h) \text{ using } RS1^{\min}$$

$$\bar{u}(r+1, m, h) \text{ using } RS3^{\min}$$

$$\bar{t}(r+1, m, h) \text{ using } RS4^{\min}$$

F: 1. If $\emptyset < q_2 < |A(r, n^{r+1}, h)|$ and $(i, j) \in R^{r+1}$
then:

$$\text{Set: } M^{r+1} = M^{r+1} \cup \{(i, j)\}$$

$$\bar{A}^{r+1} = \bar{A}^{r+1} \cup \{(k^{r+1}(i), k^{r+1}(j))\}$$

GO TO Step 4F3.

Else GO TO Step 4F2.

2. If $\emptyset < q_2 < |A(r, n^{r+1}, h)|$ and $(i, j) \notin R^{r+1}$
then:

$$\text{Set: } R^{r+1} = R^{r+1} \cup \{(i, j)\}$$

$$P_a^{r+1} = P_a^{r+1} \cup \{(i, j)\}$$

$$\bar{A}^{r+1} = \bar{A}^{r+1} \cup \{(k^{r+1}(i), k^{r+1}(j))\}$$

Step 5: Repeat Step 4 with indices reversed, i.e. replace m, h with h, m and n^{r+1}, h with h, n^{r+1} . Change Step 4A3 to read "GO TO Step 6" in place of "GO TO Step 5."

Step 6: If $h = n^r$ then STOP, else set $h = h+1$ and GO TO Step 4A.

Example 3.2

We now apply the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure to the previous example with:

$$N^1 = \{N_1^1, N_2^1, \dots, N_5^1\}$$

$$\begin{aligned}
&= \{\{2,3,4,6,11,12\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}\} \\
N^2 &= \{N_1^2, N_2^2, \dots, N_6^2\} \\
&= \{\{2,3,6,11\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}, \{4,12\}\}
\end{aligned}$$

The partitioned adjacency matrix is shown below with the elements of N_1^1 reordered to correspond to N_1^2 and N_6^2

	2	3	6	11	4	12	1	5	7	8	9	10
2		1	1	1							1	
3					1						1	1
6						1					1	
11												
4										1		1
12												
1							1	1		1	1	
5										1		
7												
8												
9												
10												

where (1) is a representative arc and (1) is a reinstated arc.

Step 0:

$$\begin{aligned}
\bar{N}^2 &= \bar{N}^1 \cup \{6\} = \{1,2,3,4,5,6\} \\
\bar{A}^2 &= \bar{A}^1 = \{(1,3), (1,4), (1,5), (2,3), (2,4)\} \\
R^2 &= R^1 = \{(4,8), (6,9), (4,10), (5,8), (1,9)\} \\
M^2 &= P_a^2 = P_e^2 = \{\emptyset\}
\end{aligned}$$

Step 1A:

i:	1	2	3	4	5	6	7	8	9	10	11	12
$k^2(i)$:	2	1	1	6	2	1	2	3	4	5	1	6

Step 1B:

$$\begin{aligned}
\bar{b}_m^2 &= \bar{b}_m^1, m = 2, 3, 4, 5 \\
\bar{b}_6^2 &= b_4 + b_{12} = -16 \\
\bar{b}_1^2 &= b_2 + b_3 + b_6 + b_{11} = 31
\end{aligned}$$

Step 2A:

- $A_{11}^2 = \{(2,3), (2,6), (2,11)\}$
- $A_{22}^2 = \{\emptyset\}$

$$3. A_{16}^2 = \{(3,4), (6,12)\}$$

$$4. A_{61}^2 = \{\emptyset\}$$

Step 2B:

$$\text{For } A_{16}^2 \neq \{\emptyset\}$$

$$1. R_{16}^2 = (3,4)$$

$$2. R^2 = R^2 \cup \{(3,4)\} \\ = \{(4,8), (6,9), (4,10), (5,8), (1,9), (3,4)\}$$

$$P_e^2 = P_e^2 \cup \{(3,4)\}$$

$$= \{(3,4)\}$$

$$\bar{A}^2 = \bar{A}^2 \cup \{(1,6)\}$$

$$= \{(1,3), (1,4), (1,5), (2,3), (2,4), (1,6)\}$$

$$3. \bar{c}_{16}^2 = c_{34} = 23$$

$$\bar{u}_{16}^2 = u_{34} + u_{6,12} = 22$$

$$\bar{t}_{16}^2 = t_{34} + t_{6,12} = 0$$

Step 2C:

$$A_{61}^2 = \{\emptyset\}$$

Step 3:

$h = 1.$ GO TO Step 6.

Step 6:

$h = 2.$ GO TO Step 4A.

Step 4A:

$$1. A_{12}^2 = \{\emptyset\}$$

$$2. A_{62}^2 = \{\emptyset\}$$

$$3. |A_{12}^2| = |A_{62}^2| = 0 \Rightarrow \text{GO TO } \underline{\text{Step 5.}}$$

Step 5A:

$$1. A_{21}^2 = \{\emptyset\}$$

$$2. A_{26}^2 = \{\emptyset\}$$

$$3. |A_{63}^2| = |A_{26}^2| = 0 \Rightarrow \text{GO TO } \underline{\text{Step 6.}}$$

Step 6:

$h = 3,$ GO TO Step 4A

Step 4A:

$$1. A_{13}^2 = \{\emptyset\}$$

$$2. A_{63}^2 = \{(4,8)\}$$

$$3. |A_{63}^2| = 1 \Rightarrow \text{GO TO } \underline{\text{Step 4B}}$$

Step 4B: 1. n/a
2. $R_{63}^2 = (4, 8)$

Step 4C: 1. $q_1 = 1 - 0 = 1$
2. $q_2 = 1 - 1 = 0$

Step 4D: 1. $q_1 = 1 \Rightarrow \text{GO TO Step 4D2}$
2. $q_2 = 0 \Rightarrow \text{continue}$
 $(k^1(4), k^1(8)) = (1, 3) \neq (6, 3)$
 $= (k^2(4), k^2(8))$
 $\bar{A}^2 = \bar{A}^2 - \{(1, 3)\} \cup \{(6, 3)\}$
 $= \{(1, 4), (1, 5), (2, 3), (2, 4), (1, 6), (6, 3)\}$
GO TO Step 5.

Step 5A: 1. $A_{31}^2 = \{\emptyset\}$
2. $A_{36}^2 = \{\emptyset\}$
3. $|A_{31}^2| = |A_{36}^2| = 0 \Rightarrow \text{GO TO Step 6.}$

Step 6: $h = 4, \text{GO TO Step 4A.}$

Step 4A: 1. $A_{14}^2 = \{(2, 9), (3, 9), (6, 9)\}$
2. $A_{64}^2 = \{\emptyset\}$
3. $|A_{14}^2| = 3 \Rightarrow \text{GO TO Step 4B.}$

Step 4B: 1. $R_{14}^2 = (6, 9)$
2. n/a

Step 4C: 1. $q_1 = 3 - 3 = 0$
2. $q_2 = 3 - 0 = 3$

Step 4D: 1. $q_1 = 0 \Rightarrow \text{continue}$
 $(k^1(6), k^1(9)) = (1, 4) = (1, 4) = (k^2(6), k^2(9)) \Rightarrow$
no designation change necessary, GO TO Step 5

Step 5A: 1. $A_{41}^2 = \{\emptyset\}$

AD-A170 681

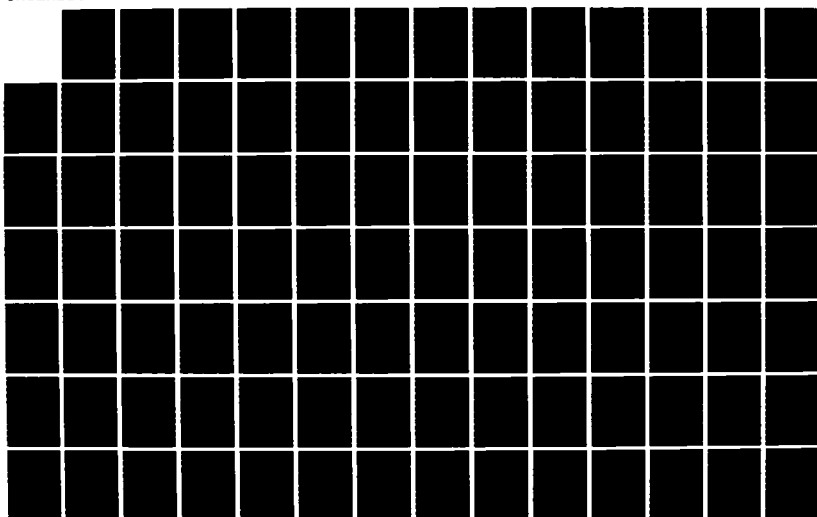
AGGREGATION OF NETWORK FLOW PROBLEMS(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH V E FRANCIS 1985
AFIT/CI/NR-86-800

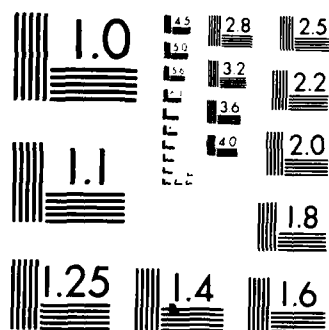
2/4

UNCLASSIFIED

F/G 12/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$2. A_{46}^2 = \{\emptyset\}$$

$$3. |A_{41}^2| = |A_{46}^2| = \emptyset \Rightarrow \text{GO TO Step 6.}$$

Step 6: $h = 5$, GO TO Step 4A.

Step 4A: 1. $A_{15}^2 = \{(3,10)\}$

$$2. A_{65}^2 = \{(4,10)\}$$

$$3. |A_{15}^2| = |A_{65}^2| = 1 \Rightarrow \text{GO TO Step 4B.}$$

Step 4B: 1. $R_{15}^2 = (3,10)$

$$2. R_{65}^2 = (4,10)$$

Step 4C: 1. $q_1 = 2 - 1 = 1$

$$2. q_2 = 2 - 1 = 1$$

Step 4D: 1. $q_1 = 1 \Rightarrow \text{GO TO Step 4D2}$

$$2. q_2 = 1 \Rightarrow \text{GO TO Step 4E}$$

Step 4E: 1. $\emptyset < q_1 < 2, (3,10) \notin R^2 \Rightarrow \text{GO TO Step 4E2}$

$$2. \emptyset < q_1 < 2, (3,10) \notin R^2 \Rightarrow \text{continue}$$

$$R^2 = R^2 \cup \{(3,10)\} =$$

$$\{(4,8), (6,9), (4,10), (5,8), (1,9), (3,4), (3,10)\}$$

$$P_a^2 = P_a^2 \cup \{(3,10)\}$$

$$= \{(3,10)\}$$

$$\bar{A}^2 = \bar{A}^2 \cup \{(k^2(3), k^2(10))\}$$

$$= \bar{A}^2 \cup \{(1,5)\}$$

$$= \{(1,4), (1,5), (2,3), (2,4), (1,6), (6,3)\}$$

$$3. \bar{c}_{15}^2 = c_{3,10} = 26$$

$$\bar{u}_{15}^2 = u_{3,10} = 8$$

$$\bar{t}_{15}^2 = t_{3,10} = \emptyset$$

Step 4F: 1. $\emptyset < q_2 < 2, (4,10) \in R^2 \Rightarrow \text{continue}$

$$M^2 = M^2 \cup \{(4,10)\}$$

$$\begin{aligned}
&= \{(4, 10)\} \\
\bar{A}^2 &= \bar{A}^2 \cup \{(k^2(4), k^2(10))\} \\
&= \bar{A}^2 \cup \{(6, 5)\} \\
&= \{(1, 4), (1, 5), (2, 3), (2, 4), (1, 6), \\
&\quad (6, 3), (6, 5)\}
\end{aligned}$$

$$\begin{aligned}
3. \quad \bar{c}_{65}^2 &= c_{4,10} = 20 \\
\bar{u}_{65}^2 &= u_{4,10} = 2 \\
\bar{t}_{65}^2 &= t_{4,10} = 0
\end{aligned}$$

Step 5A:

$$\begin{aligned}
1. \quad A_{51}^2 &= \{\emptyset\} \\
2. \quad A_{56}^2 &= \{\emptyset\} \\
3. \quad |A_{51}^2| &= |A_{56}^2| = 0 \Rightarrow \text{GO TO Step 6.}
\end{aligned}$$

Step 6: $h = 5 = n^1 \Rightarrow \text{STOP.}$

$\overline{NP}^1$ and $\overline{NP}^2$ are shown below in Figure 3.1 along with a summary of the associated parameters.

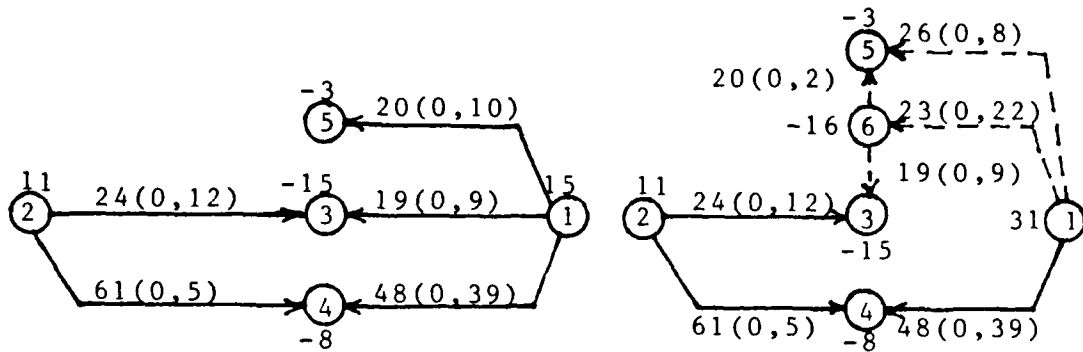
The $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure can be intuitively explained and more clearly visualized by direct reference to a "modified" partitioned adjacency matrix. We require the following definitions.

Defintion 3.7: Given N^r and N^{r+1} , where N^{r+1} is obtained by simple refinement of N_m^r , let the ordered sequence $\hat{N}^{r+1}$ be the $1 \times n$ vector defined as:

$$\hat{N}_i^{r+1} = \tilde{N}_i^{r+1} \text{ for } i \notin N_m^r$$

and for $i \in N_m^r$

if $i \in N_m^{r+1}$, $h \in N(r+1, n^{r+1})$ then i preceeds h



$\overline{NP}^1$

$$\bar{N}^1 = \{1, 2, 3, 4, 5\}$$

$$n^1 = 5$$

$$\bar{A}^1 = \{(1, 3), (1, 4), (1, 5), \\ (2, 3), (2, 4)\}$$

$$R^1 = \{(4, 8), (6, 9), (4, 10), \\ (5, 8), (1, 9)\}$$

$$M^1 = 1.324$$

$\overline{NP}^2$

$$\bar{N}^2 = \{1, 2, 3, 4, 5, 6\}$$

$$n^2 = 6$$

$$\bar{A}^2 = \{(1, 4), (1, 5), (1, 6), \\ (2, 3), (2, 4), (6, 3), \\ (6, 5)\}$$

$$R^2 = \{(6, 9), (3, 10), (3, 4), \\ (5, 8), (1, 9), (4, 8), \\ (4, 10)\}$$

$$M^2 = 1.107$$

$$P^2 = P_e^2 \cup P_a^2 \\ = \{(3, 4), (3, 10)\}$$

$$M^2 = \{(4, 10)\}$$

$$I(M^2) = I((4, 10)) = (3, 10)$$

Figure 3.1: $\overline{NP}^1$ And $\overline{NP}^2$ For Example 3.2

if $i, h \in N_q^{r+1}$, $q = m$, n^{r+1} , then i preceeds h
if $i < h$.

Definition 3.8: We shall refer to the adjacency matrix with rows and columns ordered according to $\hat{N}^{r+1}$ as the modified partitioned adjacency matrix (associated with N^r and N^{r+1}).

Defintion 3.9: Let N_m^r be refined into two subsets N_m^{r+1} and $N(r+1, n^{r+1})$. The submatrix γ_{mq}^r with rows and columns permuted to correspond to $\hat{N}^{r+1}$ is said to have been cut by the simple refinement if

i. for $m \neq q$: $\underline{1} \gamma_{mq}^r \tilde{\underline{1}} > \underline{1} \gamma(r+1, m, q) \tilde{\underline{1}}$
where $\underline{1}$ and $\tilde{\underline{1}}$ are conformable vectors all of whose entries are one

ii. for $m = q$: $\underline{1} \gamma(r+1, m, n^{r+1}) \tilde{\underline{1}} > 0$
or $\underline{1} \gamma(r+1, n^{r+1}, m) \tilde{\underline{1}} > 0$

Intuitively, Definition 3.9 simply states that the 1 entries of γ_{mq}^r have been "divided" among $\gamma(r+1, m, q)$ and $\gamma(r+1, n^{r+1}, q)$. In Example 3.1, γ_{15}^1 and γ_{11}^1 are cut by refining N_1^1 into N_1^2 and N_6^2 as shown below.

$$N_5^1 = N_5^2$$

$$\gamma_{15}^1 = \begin{matrix} 2 \\ 3 \\ 6 \\ 11 \\ 4 \\ 12 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{matrix} N_1^2 \\ N_6^2 \end{matrix} \left. \vphantom{\begin{matrix} N_1^2 \\ N_6^2 \end{matrix}} \right\} N_1^1$$

$$\gamma_{11}^1 = \begin{array}{c} 2 \quad 3 \quad 6 \quad 11 \quad 4 \quad 12 \\ \begin{array}{|c|c|c|c|c|c|} \hline 2 & & 1 & 1 & 1 & \\ \hline 3 & & & & & 1 \\ \hline 6 & & & & & \\ \hline 11 & & & & & 1 \\ \hline 4 & & & & & \\ \hline 12 & & & & & \\ \hline \end{array} \end{array} = \begin{array}{|c|c|} \hline \begin{array}{c} 2 \\ \gamma_{11}^2 \end{array} & \begin{array}{c} 2 \\ \gamma_{16}^2 \end{array} \\ \hline \begin{array}{c} 2 \\ \gamma_{61}^2 \end{array} & \begin{array}{c} 2 \\ \gamma_{66}^2 \end{array} \\ \hline \end{array}$$

We shall also speak of cutting the subsets A_{mq}^r . The analogous definition follows.

Definition 3.10: Let N_m^r be refined into the two subset N_m^{r+1} , $N(r+1, n^{r+1})$. The subset A_{mq}^r is said to have been cut by this refinement if

i. for $m \neq q$

$$|A_{mq}^r| > |A(r+1, m, q)|$$

ii. for $m = q$

$$|A(r+1, m, n^{r+1})| > 0$$

$$\text{or } |A(r+1, n^{r+1}, m)| > 0$$

We can intuitively explain the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure in terms of these new definitions.

Step 0: Initialization.

Step 1: Add new node n^{r+1} to $\overline{NP}^r$ and adjust supplies for nodes m and n^{r+1} .

Step 2: For the submatrix γ_{mm}^r : If γ_{mm}^r is cut by the refinement, an arc (perhaps two) that was previously eliminated is added to $\overline{NP}^r$; otherwise there is no change.

Step 3: Initialization of iteration counter.

Step 4: If submatrix $\gamma_{m,h}^r$ is cut: We have an

adjusted-reinstated arc pair. The reinstated arc (i, j) "generates" the new aggregate arc $(k^{r+1}(i), k^{r+1}(j))$ that is added to $\overline{NP}^r$. Arc attributes for the corresponding arc pair are adjusted accordingly. If submatrix $\gamma_{m,h}^r$ is not cut: We redesignate aggregate arc $(k^r(m), k^r(h))$ of $\overline{NP}^r$ as $(k^{r+1}(m), k^{r+1}(h))$ in $\overline{NP}^{r+1}$ if necessary.

Step 5: Performs the same function as Step 4 for the submatrix $\gamma_{h,m}^r$

Step 6: Increment iteration counter and check stopping criterion.

We conclude this chapter with the following remarks and propositions concerning the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure.

Remark 3.1: For a given NP^0 and an initial partition N^1 , iteratively applying the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure for $r = 1, 2, \dots, s \in n - n^1$ generates a sequence of aggregate network problems. The correspondence to Lee's sequence of surrogate problems is direct; we merely expound upon how to explicitly and efficiently generate each succeeding surrogate problem from the current one. For $r = n - n^1$ we have that $\overline{NP}^r = NP^0$. Such a sequence is finite since we assume that n is finite.

Remark 3.2: We have described the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure only for the simple refinement of a single subset of N^r .

Since refinements other than simple ones can be generated by iteratively applying simple refinement, the procedure is valid for the more general case.

Remark 3.3: We have made no mention up to this point of how N^r and N^{r+1} should be determined. We assume that N^1 is determined subjectively and logically for the particular problem context and managerial application at hand. We shall develop a criterion for determining N^{r+1} , given N^r , in chapter five. This criterion is based upon efficient disaggregation of the solution of $\overline{NP}^r$ into a basic solution of $\overline{NP}^{r+1}$.

Proposition 3.1: For the sequence $\overline{NP}^r$, $r = 1, 2, \dots, n - n^1$, the associated sequence M^r is monotonically decreasing and

$$\lim_{r \rightarrow n - n^1} M^r = 0$$

Proof:

Since N^{r+1} is a simple refinement of N^r , we have:

$$\begin{aligned} n^{r+1} &= n^r + 1 > n^r \\ |\bar{A}^{r+1}| &\geq |\bar{A}^r| \geq 0 \\ \Rightarrow M^{r+1} &= \frac{n - n^{r+1}}{n - 1} + \frac{a - |\bar{A}^{r+1}|}{a} \\ &< \frac{n - n^r}{n - 1} + \frac{a - |\bar{A}^r|}{a} = M^r \end{aligned}$$

For $r = n - n^1$ we have that $n^r = n$ and $|\bar{A}^r| = a$ since $\overline{NP}^r = \overline{NP}^0$.

Thus:

$$\begin{aligned}\lim_{r \rightarrow n-1} M^r &= \lim_{r \rightarrow n-1} \left[\frac{n - n^r}{n - 1} + \frac{a - A^r}{a} \right] \\ &= \frac{n - n}{n - 1} + \frac{a - a}{a} = 0\end{aligned}$$

CHAPTER FOUR

THE GENERAL DISAGGREGATION MAP DM^G

In chapter three we explained how a sequence of aggregate network models can be generated by successive refinement of the partition of the node set. We characterized important types of arcs which allowed us to efficiently create $\overline{NP}^{r+1}$ from $\overline{NP}^r$. From an algorithmic perspective, it would be beneficial to have a means to simultaneously create a solution of $\overline{NP}^{r+1}$ from the solution of $\overline{NP}^r$. Disaggregation methods have been studied by other researchers, however none of the proposed disaggregation methods map a basic solution of $\overline{NP}^r$ into a basic solution of $\overline{NP}^{r+1}$. The advantage of such a mapping is obvious: the disaggregated solution of $\overline{NP}^r$ would provide an advanced-start basic solution for $\overline{NP}^{r+1}$. The information available from the optimal solution of $\overline{NP}^r$ would thus be effectively used to reduce the computational effort required to solve $\overline{NP}^{r+1}$. A disaggregation map with this property is virtually essential for a viable, efficient network algorithm which uses aggregation-disaggregation concepts.

Research to date presumes that either $\overline{NP}^{r+1}$ would be "solved from scratch," or that the disaggregated solution of $\overline{NP}^r$ is to be used as a surrogate for the optimal solution of NP^0 . In this chapter, we develop a general disaggregation map, DM^G , that transforms a basic, feasible (optimal)

solution of $\overline{NP}^r$ into a basic solution of $\overline{NP}^{r+1}$. Moreover, $\overline{NP}^{r+1}$ along with a basic solution of $\overline{NP}^{r+1}$ are created simultaneously by a simple refinement of N^r . A criterion and methodology for selecting a refinement of N^r are the subjects of the next chapter.

We begin with definitions to establish necessary terminology and notational convention.

I. Definitions, Notation And Terminology

Definition 4.1: Define the following:

$\bar{x}^r$: an optimal solution of $\overline{NP}^r$

$\bar{x}^r(m,q)$: the $(m,q)^{th}$ component of $\bar{x}^r$

$B(\bar{x}^r)$: $\{(i,j) \in R^r : \bar{x}^r(k^r(i), k^r(j)) \text{ is a basic variable}\}$

$U(\bar{x}^r)$: $\{(i,j) \in R^r : \bar{x}^r(k^r(i), k^r(j)) = \bar{u}(r, k^r(i), k^r(j)) \text{ and } (i,j) \notin B(\bar{x}^r)\}$

$W(\bar{x}^r)$: $\{(i,j) \in R^r : (i,j) \notin B(\bar{x}^r), (i,j) \notin U(\bar{x}^r)\}$

Q^{r+1} : $R^r - \{M^{r+1} \cap \{B(\bar{x}^r) \cup U(\bar{x}^r)\}\}$

Remark 4.1: $B(\bar{x}^r)$ is simply the set of representative arcs which generate the basic aggregate arcs in the optimal solution of $\overline{NP}^r$. $U(\bar{x}^r)$ is the set of representative arcs which generate the aggregate arcs which are out of the basis at capacity. (We shall refer to "out of basis at capacity" as OOB.) $W(\bar{x}^r)$ is the set of representative arcs which

generate the nonbasic arcs of the optimal solution of $\overline{NP}^r$. Q^{r+1} is the set of representative arcs of $\overline{NP}^r$ excluding those which are adjusted and basic and those which are adjusted and OOB. Q^{r+1} is defined for notational convenience.

Example 4.1: The optimal solution of $\overline{NP}^1$ of Example 3.1 is given below.

$$\bar{x}^{1*} = \begin{bmatrix} \bar{x}^{1*}(1,3) \\ (1,4) \\ (1,5) \\ (2,3) \\ (2,4) \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 3 \\ 11 \\ 0 \end{bmatrix}$$

Noting that: $\underline{(i,j) \in R^1} \quad \underline{(k^1(i), k^1(j))}$

(4,8)	(1,3)
(6,9)	(1,4)
(4,10)	(1,5)
(5,8)	(2,3)
(1,9)	(2,4)

we have:

$$\begin{aligned} B(\bar{x}^{1*}) &= \{(4,8), (6,9), (4,10), (5,8)\} \\ U(\bar{x}^{1*}) &= \{\emptyset\} \\ W(\bar{x}^{1*}) &= \{(1,9)\} \\ Q^2 &= R^1 - \{M^2 \cap \{B(\bar{x}^{1*}) \cup U(\bar{x}^{1*})\}\} \\ &= \{(4,8), (6,9), (4,10), (5,8), (1,9)\} \\ &\quad - \{\{(4,10)\} \cap \{(4,8), (6,9), (4,10), (5,8)\} \cup \{\emptyset\}\} \end{aligned}$$

$$= \{(4,8), (6,9), (5,8), (1,9)\}$$

II. The Logic Behind The Disaggregation Map DM^G

Before defining the disaggregation map DM^G formally, we shall explain it intuitively in this section. The explanation uses the partitioned adjacency matrix to help clarify DM^G with its admittedly, albeit necessarily, cumbersome notation.

Consider the generic partitioned adjacency matrix in Figure 4.1. We assume that i) both the rows and columns have been ordered according to $\hat{N}^{r+1}$ and ii) N_1^r is simply refined into N_1^{r+1} and $N(r+1, n^{r+1})$.

We wish to assess the impact of the refinement on the solution x^* . Since, in a network problem, there is a one-to-one correspondence between variables and arcs, we shall use these terms interchangeably.

In general, the simple partition refinement potentially affects the nonzero submatrices:

$$\begin{aligned} \gamma_{1,m}^r, m \in \bar{N}^r \\ \gamma_{m,1}^r, m \in \bar{N}^r \end{aligned}$$

Furthermore, these nonzero submatrices are only affected if they are "cut" in the sense of Definition 3.8. Consider the submatrices γ_{1m}^r and γ_{m1}^r for $m \in \bar{N}^r$. We have several cases. For Cases 1 and 2 below, we discuss the impact of the partition refinement on submatrices $\gamma(r+1, 1, m_1)$ and $\gamma(r+1, n^{r+1}, m_1)$. Directly analagous results hold for the submatrices $\gamma(r+1, m_2, 1)$ and $\gamma(r+1, m_2, n^{r+1})$.

				$N_{m_1}^{r+1}$	
N_1^{r+1}	γ_{11}^{r+1}	$\gamma_{1,n^{r+1}}^{r+1}$		γ_{1,m_1}^{r+1}	
$N_{n^{r+1}}^{r+1}$	$\gamma_{n^{r+1},1}^{r+1}$	$\gamma_{n^{r+1},n^{r+1}}^{r+1}$		$\gamma_{n^{r+1},m_1}^{r+1}$	
$N_{m_2}^{r+1}$	$\gamma_{m_2,1}^{r+1}$	$\gamma_{m_2,n^{r+1}}^{r+1}$			

Figure 4.1: Refinement Of A Generic Adjacency Matrix

Case 1: For submatrices $\gamma(r+1, l, m_1)$, $\gamma(r+1, n^{r+1}, m_1)$, if either $\gamma(r+1, l, m_1) = \emptyset$ or $\gamma(r+1, n^{r+1}, m_1) = \emptyset$ then $\gamma(r, l, m_1)$ has not been cut by the specified refinement. This implies that aggregate arc (l, m_1) of $\overline{NP}^r$ has essentially been unaffected by the refinement, except possibly for a change in designation. A redesignation is necessary only if $\gamma(r+1, l, m_1) = \emptyset$ and $\gamma(r+1, n^{r+1}, m_1) \neq \emptyset$, in which case the arc (l, m_1) is redesignated (n^{r+1}, m_1) in $\overline{NP}^{r+1}$. For such arcs, it seems reasonable to leave the corresponding flows unchanged.

Case 2: For submatrices $\gamma(r+1, l, m_1)$, $\gamma(r+1, n^{r+1}, m_1)$ if $\gamma(r+1, l, m_1) \neq \emptyset$ and $\gamma(r+1, n^{r+1}, m_1) \neq \emptyset$, then $\gamma(r, l, m_1)$ has been cut. This implies that a reinstated arc must be added to the set of representative arcs, thus generating a new aggregate arc for $\overline{NP}^{r+1}$. Furthermore, the representative arcs $R(r+1, l, m_1)$ and $R(r+1, n^{r+1}, m_1)$ are a corresponding adjusted-reinstated pair. Since the reinstated arc was not a generator of an arc in $\overline{NP}^r$, the flow for this new arc is undefined in $\overline{NP}^{r+1}$. It is also possible that, since $\bar{u}(r+1, k^{r+1}(i), k^{r+1}(j)) < \bar{u}(r, k^r(i), k^r(j))$ for $R(r, l, m_1) = (i, j)$ and $(i, j) \in M^{r+1}$, the flow $\bar{x}^{r*}(l, m_1)$ is no longer feasible for arc $(k^{r+1}(i), k^{r+1}(j))$ in $\overline{NP}^{r+1}$ and hence must be recalculated. We consider separately the cases where $\bar{x}^{r*}(l, m_1)$ is A) nonbasic, B) OOB and C) basic.

Case 2A: $\bar{x}^*(1, m_1)$ nonbasic:

A simple and rational approach in this instance is to disaggregate $\bar{x}^*(1, m_1)$ into $\bar{x}_0^{r+1}(1, m_1)$ and $\bar{x}_0^{r+1}(n^{r+1}, m_1)$ as follows:

$$\bar{x}^*(1, m_1) = \emptyset \begin{cases} \rightarrow \bar{x}_0^{r+1}(1, m_1) = \emptyset \\ \rightarrow \bar{x}_0^{r+1}(n^{r+1}, m_1) = \emptyset \end{cases}$$

with both $\bar{x}_0^{r+1}(1, m_1)$ and $\bar{x}_0^{r+1}(n^{r+1}, m_1)$ designated as nonbasic.

Case 2B: $\bar{x}^*(1, m_1) = \bar{u}(r, 1, m_1)$ and OOB:

$\bar{x}^*(1, m_1)$ can easily and intuitively be disaggregated into $\bar{x}_0^{r+1}(1, m_1) = \bar{u}(r+1, 1, m_1)$ and $\bar{x}_0^{r+1}(n^{r+1}, m_1) = \bar{u}(r+1, n^{r+1}, m_1)$ with either or both OOB. Exactly which of the disaggregated variables is to be OOB is discussed later in Remark 4.5.

Case 2C: $\bar{x}^*(1, m_1)$ basic:

In this instance, there is no natural or intuitive way to disaggregate $\bar{x}^*(1, m_1)$. What we shall do here is to form a network subproblem consisting of i) aggregate arcs $(1, m_1)$ and (n^{r+1}, m_1) with bounds $\bar{f}(r+1, 1, m_1)$, $\bar{u}(r+1, 1, m_1)$, $\bar{f}(r+1, n^{r+1}, m_1)$ and $\bar{u}(r+1, n^{r+1}, m_1)$ and ii) aggregate nodes 1, m_1 and n^{r+1} with supplies adjusted for flows already fixed for arcs of $\bar{NP}^{r+1}$ fitting Cases 1, 2A and 2B.

Case 3: $\gamma(r+1, 1, n^{r+1}) \neq \emptyset$ and/or $\gamma(r+1, n^{r+1}, 1) \neq \emptyset$:

For this case, arcs $R(r+1,1,n^{r+1})$ and/or $R(r+1,n^{r+1},1)$ are added to R^r , depending upon which of the associated submatrices are nonzero. Each generates an aggregate arc in $\overline{NP}^{r+1}$. More specifically, $R(r+1,1,n^{r+1})$ and/or $R(r+1,n^{r+1},1)$ is/are of type P-E (reinstated arcs that were previously eliminated). Since these arcs were not elements of R^r , they did not generate aggregate arcs in $\overline{NP}^r$. Hence, flows for the corresponding aggregate arcs in $\overline{NP}^{r+1}$ are undefined. These new aggregate arcs, along with their associated tail and head nodes, are included in the network subproblem mentioned in Case 2C.

Carrying out this logic for all of the submatrices

$$\gamma(r+1,1,m), m \in \overline{N}^{r+1}$$

$$\gamma(r+1,n^{r+1},m), m \in \overline{N}^{r+1}$$

$$\gamma(r+1,m,1), m \in \overline{N}^{r+1}$$

$$\gamma(r+1,m,n^{r+1}), m \in \overline{N}^{r+1}$$

yields values for all variables of $\overline{NP}^{r+1}$ with:

1. flows fixed for all arcs falling within Cases 1, 2A and 2B
2. a network subproblem created for all arcs (and associated head and tail nodes) falling within Cases 2C and 3.

We shall term this disaggregation map "the general disaggregation map," denoted DM^G . Thus DM^G in essence determines the values for the variables of $\overline{NP}^{r+1}$ in two fundamental ways:

1. $\bar{x}_0^{r+1}(m,q)$ that are immediately determined for Cases 1, 2A and 2B,
2. $\bar{x}_0^{r+1}(m,q)$ that are determined via solution of the network subproblem.

We shall denote the network subproblem as SNP^{r+1} .

We delay presentation of an example until after the formal definition of DM^G in the next section.

III. Definition Of DM^G

Definition 4.2 provides a rigorous definition of DM^G and is followed by remarks and an example.

Definition 4.2: The general disaggregation map $DM^G(\bar{x}^{r*}) = \bar{x}_0^{r+1}$ is composed of five parts: DM^G_A , DM^G_B , DM^G_C , DM^G_D and DM^G_E defined as follows:

DM^G_A : For arcs $(i,j) \in Q^{r+1} \cup I(M^{r+1} \cap W(\bar{x}^{r*}))$ let

$$\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) = \bar{x}^{r*}(k^r(i), k^r(j))$$

with all such variables inheriting the status (basic, nonbasic, OOB) of the associated variable $\bar{x}^{r*}(k^r(i), k^r(j))$ unless otherwise specified in DM^G_D .

DM^G_B : For notational convenience let $Y^{r+1} = M^{r+1} \cap U(\bar{x}^{r*})$.

For arcs $(i,j) \in Y^{r+1} \cup I(Y^{r+1})$ let

$$\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) = \bar{u}(r+1, k^{r+1}(i), k^{r+1}(j))$$

with all such variables OOB unless otherwise specified in DM^G_D .

DM^G_C : For notational convenience let $D^{r+1} = M^{r+1} \cap B(\bar{x}^{r*})$.

For arcs $(i, j) \in D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}$ define the network subproblem $\overline{SNP}^{r+1}$ with arcs

$$(m, q) = (k^{r+1}(i), k^{r+1}(j))$$

having lower and upper bounds

$$\underline{E}(r+1, k^{r+1}(i), k^{r+1}(j)) \text{ and } \bar{U}(r+1, k^{r+1}(i), k^{r+1}(j))$$

and with corresponding head and tail nodes

$$k^{r+1}(i) \text{ and } k^{r+1}(j)$$

with supplies

$$\begin{aligned} \bar{b}_m^{r+1} &= \bar{b}_m^{r+1} - \sum_{q: (m, q) \in \bar{E}(r+1)} \bar{x}_0^{r+1}(m, q) \\ &\quad + \sum_{q: (q, m) \in \bar{E}(r+1)} \bar{x}_0^{r+1}(q, m) \end{aligned}$$

where

$$E(r+1) = Q^{r+1} \cup I(M^{r+1} \cap W(\bar{x}^{r*})) \cup Y^{r+1} \cup I(Y^{r+1})$$

$$\bar{E}(r+1) = \{(m, q) : m = k^{r+1}(i), q = k^{r+1}(j), (i, j) \in E\}$$

DM^G_D: If $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1} = \{\emptyset\}$ then designate as

basic either:

$$\begin{aligned} \text{i)} \quad \bar{x}_0^{r+1}(k^{r+1}(i_0), k^{r+1}(j_0)) &= \\ \bar{U}(r+1, k^{r+1}(i_0), k^{r+1}(j_0)) \end{aligned}$$

$$\text{for } (i_0, j_0) \in Y^{r+1} \cup I(Y^{r+1})$$

$$\text{or ii)} \quad \bar{x}_0^{r+1}(k^{r+1}(i_0), k^{r+1}(j_0)) = \emptyset$$

$$\text{for } (i_0, j_0) \in Q^{r+1} \cup I(M^{r+1} \cap W(\bar{x}^{r*}))$$

with

$$\text{i)} \quad i_0 \in H \text{ and } j_0 \in \bar{N}^{r+1} - H$$

$$\text{or ii)} \quad j_0 \in H \text{ and } i_0 \in \bar{N}^{r+1} - H$$

where

$H = \{m \in N^{r+1} : m \text{ is in the same tree as } n^{r+1}$
in the partially constructed basic solution
obtained from $DM^G_A), DM^G_B)$ and $DM^G_C)\}$

$DM^G_E)$: If

- i) $\{D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}\} = \{\emptyset\}$
- ii) $\hat{b}(r+1, n^{r+1}) = \bar{b}(r+1, n^{r+1}) - \sum_{q: (n^{r+1}, q) \in \bar{A}^{r+1}} \bar{x}_0^{r+1}(n^{r+1}, q)$
 $+ \sum_{q: (q, n^{r+1}) \in \bar{A}^{r+1}} \bar{x}_0^{r+1}(q, n^{r+1}) \neq 0$

then a pair of artificial arcs must be introduced to
conserve flow at nodes m_0 and n^{r+1} , where $N(r, m_0)$ is the
subset refined. Specifically,

- i) if $\hat{b}(r+1, n^{r+1}) > 0$ then artificial OOB arcs
 (n^{r+1}, root) and (root, m_0) are introduced
with flow $\hat{b}(r+1, n^{r+1})$
- ii) if $\hat{b}(r+1, n^{r+1}) < 0$ then artificial OOB arcs
 (m_0, root) and (root, n^{r+1}) are introduced
with flow $-\hat{b}(r+1, n^{r+1})$

Remark 4.2: $DM^G_A)$ pertains to the set of representative
arcs of NP^r minus the adjusted arcs that are also basic or
OOB and the set of reinstated arcs corresponding to nonbasic
adjusted arcs. These two sets combined yield the set of
"unaffected" representative arcs plus all nonbasic
adjusted-reinstated arc pairs. $DM^G_A)$ corresponds to Cases 1
and 2A of the previous section.

Remark 4.3: DM^G_B) pertains to adjusted, OOB arcs and their corresponding reinstated arcs. DM^G_B) corresponds to Case 2B of the previous section.

Remark 4.4: DM^G_C) pertains to basic adjusted-reinstated arc pairs and the reinstated arcs that were previously eliminated. These arcs generate the (aggregate) arcs of the network subproblem $\overline{SNP}^{r+1}$. DM^G_C) corresponds to Cases 2C and 3 of the previous section.

Remark 4.5: DM^G_D) handles the anomalous case where $DM^G(\bar{x}^{r*})$ would not yield a basic solution because $\overline{SNP}^{r+1}$ contains no nodes and no arcs. We shall call such a subproblem degenerate. In this special instance it is necessary to include a variable at capacity in the basis or change the status of a nonbasic variable to basic. This aspect of DM^G is discussed further in the proofs of Theorem 4.2 and its corollary later in the chapter. Example 4.5 is an instance where DM^G_D) is necessary.

Remark 4.6: DM^G_A) and DM^G_B) comprise the "natural or intuitive" part of DM^G alluded to earlier, whereas DM^G_C) defines the network subproblem to determine values for the arcs/variables with no "natural" disaggregation.

Remark 4.7: $E(r+1)$ in DM^G_C is simply the set of all arcs (i,j) such that the flows on the arcs $(k^{r+1}(i), k^{r+1}(j))$ have been fixed via DM^G_A or DM^G_B .

Remark 4.8: $DM^G(\bar{x}^{r*})$ does not, in general, result in a feasible solution of $\overline{NP}^{r+1}$ because $\overline{SNP}^{r+1}$ is not necessarily feasible. A basic solution for $\overline{NP}^{r+1}$ can always be obtained however, by augmenting with artificial arcs. As discussed in [2], an artificial basic solution for a capacitated transshipment problem is easily obtained by setting

$$x(i, \text{root}) = b_i \text{ for } b_i > 0$$

and

$$x(\text{root}, i) = -b_i \text{ for } b_i = 0$$

where "root" is the artificial root of the basis tree/predecessor graph. In instances where $\overline{SNP}^{r+1}$ is infeasible, artificial basic arcs remain in the solution. These arcs are maintained in $DM^G(\bar{x}^{r*})$ to complete the basis. Example 4.4 is such a case.

Remark 4.9: There are two instances when artificial arcs must be included in the disaggregated solution $DM^G(\bar{x}^{r*})$. Remark 4.8 discussed the case where $\overline{SNP}^{r+1}$ is infeasible. The other instance is where $\overline{SNP}^{r+1}$ is degenerate resulting in a violation of flow conservation at nodes n^{r+1} and m_0 . This situation is covered by DM^G_E . Example 4.5 is such a case.

Example 4.2

Consider example 3.1 with

$$N^1 = \{\{2,3,4,6,11,12\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}\}$$

$$N^2 = \{\{2,3,6,11\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}, \{4,12\}\}$$

$$\bar{x}^{1*} = \begin{bmatrix} \bar{x}^{1*}(1,3) \\ (1,4) \\ (1,5) \\ (2,3) \\ (2,4) \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 3 \\ 11 \\ 0 \end{bmatrix}$$

We note that:

$$i: \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12$$

$$k^1(i): \quad 2 \quad 1 \quad 1 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 1 \quad 1$$

$$k^2(i): \quad 2 \quad 1 \quad 1 \quad 6 \quad 2 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 1 \quad 6$$

$$R^1 = \{(4,8), (6,9), (4,10), (5,8), (1,9)\}$$

$$R^2 = \{(4,8), (6,9), (4,10), (5,8), (1,9), (3,4), (3,10)\}$$

$$P_e^2 = \{(3,4)\}$$

$$P_a^2 = \{(3,10)\}$$

$$M^2 = \{(4,10)\}$$

$$B(\bar{x}^{1*}) = \{(4,8), (6,9), (4,10), (5,8)\}$$

$$U(\bar{x}^{1*}) = \{\emptyset\}$$

$$W(\bar{x}^{1*}) = \{(1,9)\}$$

Now:

DM^{GA}): Arcs $(i,j) \in Q^1 \cup I(M^2 \cap W(\bar{x}^{1*}))$ are:

$$Q^2 = \{(4,8), (6,9), (4,10), (5,8), (1,9)\}$$

$$= \{(\{4,10\}) \cap \{(4,8), (6,9), (4,10), (5,8)\})\}$$

$$= \{(4,8), (6,9), (5,8), (1,9)\}$$

$$I(M^2 \cap W(\bar{x}^{1*})) = I(\{(\{4,10\}) \cap \{(1,9)\})\} = I(\{\emptyset\}) = \emptyset$$

$$Q^2 \cup I(M^2 \cap W(\bar{x}^{1*})) = \{(4,8), (6,9), (5,8), (1,9)\}$$

Applying DM^G_A we have

i, j	$\bar{x}^2_O(k^2(i), k^2(j))$	$=$	$\bar{x}^{1*}(k^1(i), k^1(j))$
4,8	$\bar{x}^2_O(6,3)$	$=$	$\bar{x}^{1*}(1,3) = 4$
6,9	$(1,4)$	$=$	$(1,4) = 8$
5,8	$(2,3)$	$=$	$(2,3) = 11$
1,9	$(2,4)$	$=$	$(2,4) = 0$

DM^G_B : Arcs $(i, j) \in Y^2 \cup I(Y^2)$ are:

$$Y^2 = M^2 \cap U(\bar{x}^{1*})$$

$$= \{(4,10)\} \cap \{\emptyset\} = \{\emptyset\}$$

Thus DM^G_B is n/a.

DM^G_C : Arcs $(i, j) \in D^2 \cup I(D^2) \cup P_e^2$ are:

$$D^2 = M^2 \cap B(\bar{x}^{1*})$$

$$= \{(4,10)\} \cap \{(4,8), (6,9), (4,10), (5,8)\}$$

$$= \{(4,10)\}$$

$$D^2 \cup I(D^2) \cup P_e^2 = \{(4,10)\} \cup I(\{(4,10)\}) \cup \{(3,4)\}$$

$$= \{(4,10), (3,10), (3,4)\}$$

The arcs of $\overline{SNP}^2$ are

$$(k^2(4), k^2(10)) = (6,5)$$

$$(k^2(3), k^2(10)) = (1,5)$$

$$(k^2(3), k^2(4)) = (1,6)$$

and the nodes of $\overline{SNP}^2$ are the associated head and tail nodes of these arcs, i.e. 1, 5 and 6.

$$E^2 = Q^2 \cup I(M^2 \cap W(\bar{x}^{1*})) \quad Y^2 \quad I(Y^2)$$

$$= \{(4,8), (6,9), (5,8), (1,9)\} \cup I(\{(4,10)\} \cap \{(1,9)\})$$

$$\cup \{\emptyset\} \cup \{\emptyset\}$$

$$= \{(4,8), (6,9), (5,8), (1,9)\}$$

$$\bar{E}^2 = \{(6,3), (1,4), (2,3), (2,4)\}$$

So:

$$\tilde{b}_5^2 = \bar{b}_5^2 - \sum_{q: (m,q) \in \bar{E}(2)} \bar{x}_0^2(m,q)$$

$$+ \sum_{q: (q,m) \in \bar{E}(2)} \bar{x}_0^2(q,m)$$

$$= -3 - 0 + 0 = -3$$

$$\tilde{b}_6^2 = \bar{b}_6^2 - \sum_{q: (m,q) \in \bar{E}(2)} \bar{x}_0^2(m,q)$$

$$+ \sum_{q: (q,m) \in \bar{E}(2)} \bar{x}_0^2(q,m)$$

$$= (b_4 + b_{12}) - x_0^2(6,3) = (0 - 16) - 4 = -20$$

$$\tilde{b}_1^2 = \bar{b}_1^2 - \bar{x}_0^2(1,4) = 31 - 8 = 23$$

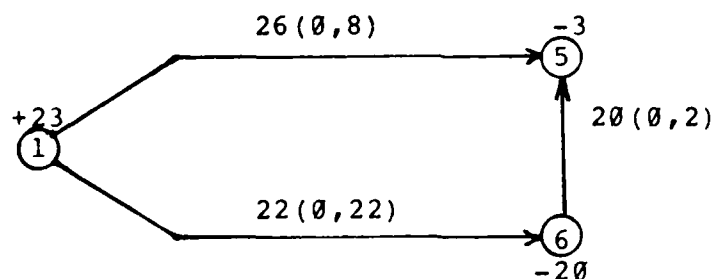
Also, from example 3.1 we have

$$\bar{c}_{15}^2 = 26 \quad \bar{c}_{16}^2 = 22 \quad \bar{c}_{65}^2 = 20$$

$$\bar{u}_{15}^2 = 8 \quad \bar{u}_{16}^2 = 22 \quad \bar{u}_{65}^2 = 2$$

$$\bar{t}_{15}^2 = 0 \quad \bar{t}_{16}^2 = 0 \quad \bar{t}_{65}^2 = 0$$

Thus we have the $\overline{SNP}^2$ depicted below.



The obvious solution of $\overline{SNP}^2$ is:

$$\begin{bmatrix} \bar{x}_0^2(1,5) \\ \bar{x}_0^2(1,6) \\ \bar{x}_0^2(6,5) \end{bmatrix} = \begin{bmatrix} 3 \\ 20 \\ 0 \end{bmatrix}$$

$\underline{DM}^G_D$): n/a

$\underline{DM}^G_E$): n/a

Thus we have the disaggregated solution as

$$DM^G(\bar{x}^{1*}) = \begin{bmatrix} x_o^2(1,4) \\ (1,5) \\ (1,6) \\ (2,3) \\ (6,3) \\ (6,5) \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 20 \\ 11 \\ 4 \\ 0 \end{bmatrix}$$

which is easily verified to be both basic and feasible for $\overline{NP}^2$.

Although the notation used to describe DM^G is necessarily quite cumbersome, the mapping itself is not complicated. It can be explained more clearly and is easily visualized by making direct reference to the basis tree associated with $\bar{x}^{1*}$. We explain DM^G informally below and illustrate it using basis trees in the next example.

Step 1: In the basis tree for $\bar{x}^{1*}$, "disaggregate" node m_o into the two nodes m_o and n^r+1 .

Step 2A: Draw arcs generated by adjusted basic arcs as dashed arcs ($---\rightarrow$).

Step 2B: Draw (add) in arcs generated by corresponding reinstated arcs as dashed arcs also.

Step 2C: Draw (add) in arcs generated by reinstated arcs that were previously eliminated as dashed arcs also.

Step 2D: Attached to dashed arcs the appropriately respecified costs and bounds using $RS1^{\min}$, $RS3^{\min}$ and $RS4^{\min}$.

Step 3: Flows on all:

- i. solid arcs in the basis tree remain unchanged

ii. arcs OOB = $\bar{u}$

iii. nonbasic arcs = $\emptyset$.

Step 4: For all nodes that are head or tail nodes of dashed arcs adjust supplies to account for flows fixed in Step 3.

Step 5: $\overline{SNP}^{r+1}$ is defined by dashed arcs with appropriately respecified arc attributes and associated head and tail nodes with adjusted supplies.

Example 4.3 DM^G is applied directly on the basis tree for the problem in Example 4.2 in Figure 4.2.

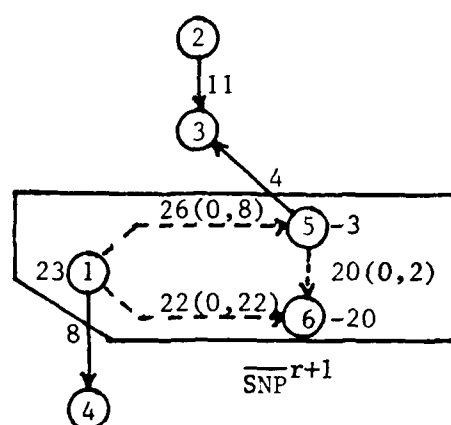
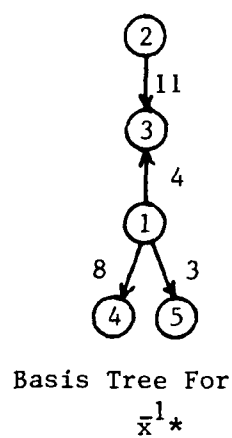
IV. DM^G Applied To The Partitioned Adjacency Matrix

As with the aggregation procedure of chapter two and the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure of the last chapter, we shall, in the interest of implementation, apply DM^G directly to the partitioned adjacency matrix associated with $\overline{NP}^r$.

We shall explain how this is accomplished by means of an example. We need the following definition before we can begin the example however.

Definition 4.3: A submatrix $\gamma_{m,q}^r$ of γ^r is said to be a basic submatrix if $\bar{x}^{r*}(m,q)$ is a basic variable in an optimal solution of $\overline{NP}^r$. A nonbasic submatrix and an OOB submatrix are defined analagously.

Example 4.4



"Basis Tree" After Application
Of DM^G A), B), C), Before
Solution Of $\overline{SNP}^{r+1}$

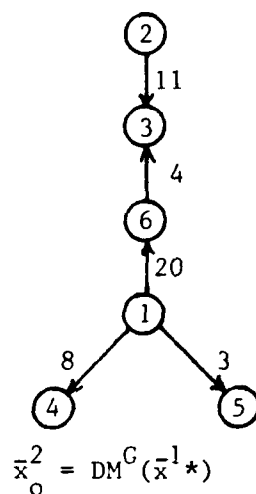


Figure 4.2: DM^G Applied To The Basis Tree Of Example 4.2

Consider NP^0 as given in example 2.1, and with

$$N^1 = \{N_1^1, N_2^1, N_3^1, N_4^1\}$$

$$= \{\{1,2\}, \{3,4\}, \{5,6,7,8\}, \{9,10,11,12\}\}$$

The optimal solution of $\overline{NP}^1$ is:

$$\begin{bmatrix} \bar{x}^{1*}(1,2) \\ (1,3) \\ (1,4) \\ (2,3) \\ (2,4) \\ (3,4) \end{bmatrix} = \begin{bmatrix} 3 \\ 47 \\ 40 \\ 0 \\ 8 \\ 0 \end{bmatrix}$$

$$B(\bar{x}^{1*}) = \{(2,3), (1,5), (4,10)\}$$

$$U(\bar{x}^{1*}) = \{(2,11)\}$$

$$W(\bar{x}^{1*}) = \{(4,8), (6,12)\}$$

$\gamma^1 =$

	1	2	3	4	5	6	7	8	9	10	11	12
1			①		①		1	1	①			
2			①			1			①			①
3				1					①	①		
4								①				
5								①				
6									①			①
7												
8												
9												
10												
11												
12												

Now consider refining N_4^1 into $N_4^2 = \{9\}$ and $N_5^2 = \{10,11,12\}$ as shown in γ^1 above, i.e.

$$N^2 = \{N_1^2, N_2^2, N_3^2, N_4^2, N_5^2\}$$

$$= \{\{1,2\}, \{3,4\}, \{5,6,7,8\}, \{9\}, \{10,11,12\}\}$$

This simple refinement is indicated by the dashed horizontal and vertical lines in γ^1 . Also, basic, nonbasic and OOB submatrices corresponding to the optimal solution $\bar{x}^{1*}$ are outlined in heavy lines, dotted lines and diagonal hash

marks respectively. Arcs of R^1 are indicated by ① and arcs of P^2 by ②. DM^G is easily explained and visualized by direct reference to the partitioned adjacency matrix.

DM^G_A): Pertains to arcs

$$(i,j) \in Q^{r+1} \cup I(M^{r+1} \cap W(\bar{x}^r*))$$

$$= \{R^r - \{M^{r+1} \cap \{B(\bar{x}^r*) \cup U(\bar{x}^r*)\}\}\} \cup I(M^{r+1} \cap W(\bar{x}^r*))$$

"adjusted arcs that are basic or OOB"	"reinstated arcs corresponding to adjusted arcs that are nonbasic"

"Adjusted arcs that are basic or OOB" are easily seen to be the arcs associated with the ①'s in the submatrices i) that have been cut by the refinement and ii) that are outlined with heavy lines or diagonal hash marks. These arcs are (2,11) and (4,10).

"Reinstated arcs corresponding to adjusted arcs that are nonbasic" are those arcs corresponding to the ②'s in the dotted submatrices that have been cut. Hence, $I(M^2 \cap W(\bar{x}^1*))$
 $= \{(6,9)\}.$

R^r is simply the set of arcs corresponding to the ①'s.

Thus

$$R^1 = \{(2,3), (1,5), (2,11), (4,8), (4,10), (6,12)\}$$

DM^G_A) then, pertains to the arcs:

$$\{(2,3), (1,5), (2,11), (4,8), (4,10), (6,12)\}$$

$$- \{(4,10), (2,11)\} \cup \{(6,9)\}$$

$$= \{(2,3), (1,5), (4,8), (6,12), (6,9)\}$$

and we have

i, j	$\bar{x}_0^2(k^2(i), k^2(j))$	=	$\bar{x}^{1*}(k^1(i), k^1(j))$
2,3	$\bar{x}_0^2(1,2)$	=	$\bar{x}^{1*}(1,2) = 3$
1,5	$\bar{x}_0^2(1,3)$	=	$\bar{x}^{1*}(1,3) = 47$
4,8	$\bar{x}_0^2(2,3)$	=	$\bar{x}^{1*}(2,3) = 0$
6,12	$\bar{x}_0^2(3,5)$	=	$\bar{x}^{1*}(3,4) = 0$
6,9	$\bar{x}_0^2(3,4)$	=	$\bar{x}^{1*}(3,4) = 0$

DM^G_B): Pertains to arcs

$$(i, j) \in Y^{r+1} \cup I(Y^{r+1})$$

$$= \{M^{r+1} \cap U(\bar{x}^{r*})\} \cup I(M^{r+1} \cap U(\bar{x}^{r*}))$$

"adjusted
OOB arcs"

"reinstated arcs corresponding
to adjusted OOB arcs"

"Adjusted OOB arcs" are the arcs corresponding to the ①'s of hashed submatrices that have been cut, i.e. (2,11)

"Reinstated arcs corresponding to adjusted OOB arcs" are the arcs corresponding to the ①'s of hashed submatrices that have been cut, i.e. (1,9).

DM^G_B is applied to arcs (2,11), (1,9) and we have

i, j	$\bar{x}_0^2(k^2(i), k^2(j))$	=	$\bar{u}(2, k^2(i), k^2(j))$
2,11	$\bar{x}_0^2(1,5)$	=	$\bar{u}(2, 1, 5) = 23$
1,9	$\bar{x}_0^2(1,4)$	=	$\bar{u}(2, 1, 4) = 17$

DM^G_C): Pertains to arcs

$$(i, j) \in D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}$$

$$= \{M^{r+1} \cap B(\bar{x}^{r*})\} \cup I(M^{r+1} \cap B(\bar{x}^{r*})) \cup P_e^{r+1}$$

"adjusted
basic
arcs"

"reinstated arcs
corresponding
to adjusted
basic arcs"

"reinstated arcs
that were
previously
eliminated"

"Adjusted basic arcs" are those corresponding to the ①'s of heavily outlined submatrices that have been cut: (4,10).

"Reinstated arcs corresponding to adjusted basic arcs" are

those corresponding to $\{1\}$'s of heavily outlined submatrices that have been cut: (3,9).

"Reinstated arcs that were previously eliminated" are those corresponding to the $\{1\}$'s of the submatrices along the diagonal of γ^1 that have been cut: $\{\emptyset\}$. Thus, DM^G_C applies to arcs $\{(4,10), (3,9)\}$ and the associated subproblem $\overline{SNP}^2$ is defined by:

$$\text{arcs: } (k^2(4), k^2(10)) = (2, 5)$$

$$(k^2(3), k^2(9)) = (2, 4)$$

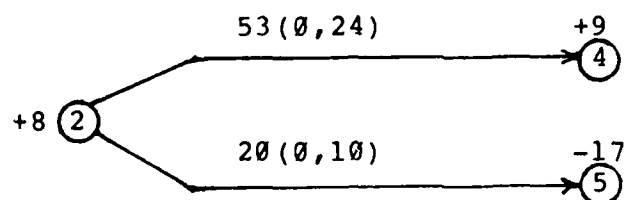
$$\text{nodes: } 2, 4, 5$$

$$\text{supplies: } \tilde{b}_2^2 = \bar{b}_2^2 + \bar{x}_0^2(1, 2) = 5 + 3 = 8$$

$$\tilde{b}_4^2 = \bar{b}_4^2 + \bar{x}_0^2(1, 4) = -8 + 17 = 9$$

$$\tilde{b}_5^2 = \bar{b}_5^2 + \bar{x}_0^2(1, 5) = -40 + 23 = -17$$

costs and bounds: as indicated on the below network diagram



The subproblem is easily seen to have no feasible solution. This is easily rectified by adding the artificial arcs (4,root) and (root,5) with cost BIG M and bounds $(0, \text{BIG } M)$. This yields the artificial solution of $\overline{SNP}^2$:

$$\begin{bmatrix} \bar{x}_0^2(2, 5) \\ \bar{x}_0^2(2, 4) \\ (4, \text{root}) \\ (\text{root}, 5) \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \\ 9 \\ 9 \end{bmatrix}$$

DM^G_D : This part of DM^G is only applicable if $D^{r+1} \cup I(D^{r+1})$

$UP_e^{r+1} = \{\emptyset\}$. For this example we have that $D^{r+1} \cup I(D^{r+1})$
 $UP_e^{r+1} = \{(4,10), (3,9)\}$ as obtained in DM^G_C . Thus DM^G_D is
not applicable.

DM^G_E : This part of DM^G is only applicable when

$$\{D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}\} = \{\emptyset\}$$

Thus, DM^G_E is not applicable for this example. The
complete disaggregated solution is shown in Figure 4.3.

Example 4.5 Consider example 4.4 but with

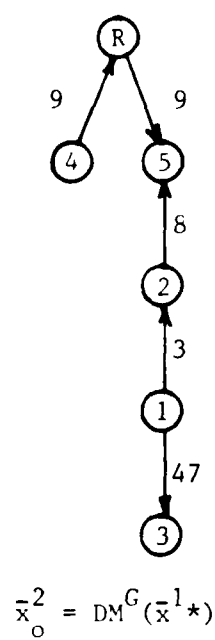
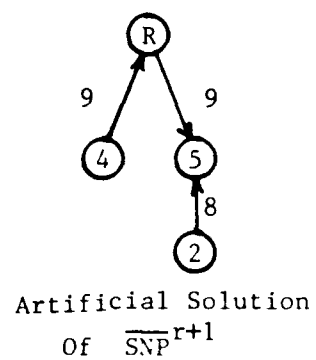
$$\begin{aligned} N^2 &= \{N_1^2, N_2^2, N_3^2, N_4^2, N_5^2\} \\ &= \{\{1,2\}, \{3,4\}, \{5,6,7,8\}, \{9,10,11\}, \{12\}\} \end{aligned}$$

The partitioned adjacency matrix is shown below:

	1	2	3	4	5	6	7	8	9	10	11	12
1					1		1	1	1	1	1	1
2			1			1			1	1	1	1
3				1					1			
4								1		1		
5								1				
6									1			1
7												
8												
9												
10												
11												
12												

Applying DM^G yields:

$$\begin{aligned} \underline{DM^G_A}: \quad Q^2 \cup I(M^2 \cap W(x^{1*})) \\ = \{(2,3), (1,5), (2,11), (4,8), (4,10), (6,12), (6,9)\} \end{aligned}$$



(R = Artificial Root Node)

Figure 4.3: Disaggregated Solution For Example 4.4

i, j	$\bar{x}_0^2(k^2(i), k^2(j))$	=	$\bar{x}^1(k^1(i), k^1(j))$
2,3	$\bar{x}_0^2(1,2)$	=	$\bar{x}^1(1,2) = 3$
1,5	$(1,3)$		$(1,3) = 47$
2,11	$(1,4)$		$(1,4) = 40$
4,8	$(2,3)$		$(2,3) = 0$
4,10	$(2,4)$		$(2,4) = 8$
6,12	$(3,5)$		$(3,4) = 0$
6,9	$(3,4)$		$(3,4) = 0$

DM^G_B): $Y^2 \cup I(Y^2) = \{\emptyset\} \Rightarrow \text{DM}^{\text{G}_B} \text{ is n/a.}$

DM^G_C): $D^2 \cup I(D^2) \cup P_e^2 = \{\emptyset\} \Rightarrow \text{DM}^{\text{G}_C} \text{ is n/a.}$

DM^G_D): $Y^2 \cup I(Y^2) = \{\emptyset\}, H = \{5\}, N^{r+1} - H = \{1,2,3,4\}.$
The only candidate for redesignation is arc
(3,5) which is redesignated from nonbasic
to basic.

Shown in Figure 4.4 below are the the basis "trees" for $\bar{x}^1$, $\bar{x}_0^2$ prior to application of DM^{G_D} and $\bar{x}_0^2$ after application of DM^{G_D} . Since DM^{G_D} is only applicable when $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}$ is empty, application of DM^{G_A} through DM^{G_C} results in a disconnected "basis tree." Applying DM^{G_D} serves to "fill out" the tree by designating an appropriate arc as basic.

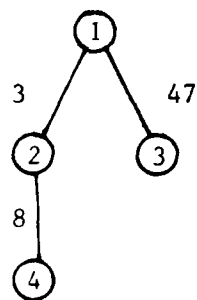
DM^G_E): Artificial OOB arcs (4,ROOT), (ROOT,5) are added with a flow of 16 to satisfy flow conservation constraints at nodes 4 and 5.

V. Some Characteristics Of DM^{G}

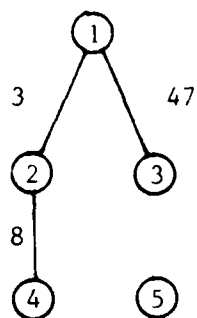
The following theorems describe some of the properties enjoyed by DM^{G} .

Theorem 4.1: DM^{G} specifies values for all variables of $\overline{N}^{r+1}$.

Proof:



Basis Tree
For $\bar{x}^1 \star$



"Basis Trees" After
 $DM^G(A), B), C)$

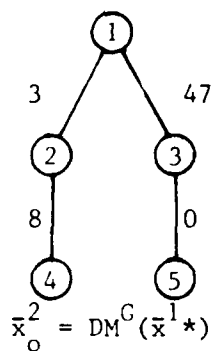


Figure 4.4: Basis Trees For Example 4.5

The arcs of $\overline{NP}^{r+1}$ have index set

$$\overline{A}^{r+1} = \{(k^{r+1}(i), k^{r+1}(j)) : (i, j) \in R^{r+1}\}$$

We must show that DM^G specifies values for all variables corresponding to the arcs of $\overline{A}^{r+1}$.

DM^G_A) specifies values for arcs generated by

$$(i, j) \in R^r - \{M^{r+1} \cap \{B(\overline{x}^{r*}) \cup U(\overline{x}^{r*})\}\} \cup I(M^{r+1} \cap W(\overline{x}^{r*}))\}$$

DM^G_B) specifies values for arcs generated by

$$(i, j) \in \{M^{r+1} \cap U(\overline{x}^{r*})\} \cup I(M^{r+1} \cap U(\overline{x}^{r*}))$$

DM^G_C) specifies values for arcs generated by

$$(i, j) \in \{M^{r+1} \cap B(\overline{x}^{r*})\} \cup I(M^{r+1} \cap B(\overline{x}^{r*})) \cup P_e^{r+1}$$

Now:

$$\begin{aligned} R^r &= \{M^{r+1} \cap \{B(\overline{x}^{r*}) \cup U(\overline{x}^{r*})\}\} \cup I(M^{r+1} \cap W(\overline{x}^{r*})) \\ &\cup \{M^{r+1} \cap U(\overline{x}^{r*})\} \cup I(M^{r+1} \cap U(\overline{x}^{r*})) \\ &\cup \{M^{r+1} \cap B(\overline{x}^{r*})\} \cup I(M^{r+1} \cap B(\overline{x}^{r*})) \cup P_e^{r+1} \\ &= R^r - \{M^{r+1} \cap B(\overline{x}^{r*})\} - \{M^{r+1} \cap U(\overline{x}^{r*})\} \cup I(M^{r+1} \cap W(\overline{x}^{r*})) \\ &\cup \{M^{r+1} \cap U(\overline{x}^{r*})\} \cup I(M^{r+1} \cap U(\overline{x}^{r*})) \\ &\cup \{M^{r+1} \cap B(\overline{x}^{r*})\} \cup I(M^{r+1} \cap B(\overline{x}^{r*})) \\ &\cup P_e^{r+1} \\ &= U R^r \cap I(M^{r+1} \cap \{W(\overline{x}^{r*}) \cup U(\overline{x}^{r*}) \cup B(\overline{x}^{r*})\}) \cup P_e^{r+1} \\ &= R^r \cup I(M^{r+1}) \cup P_e^{r+1} \\ &= R^r \cup P_a^{r+1} \cup P_e^{r+1} \\ &= R^r \cup P^{r+1} \\ &= R^{r+1} \end{aligned}$$

Thus, DM^G specifies values for all arcs $(k^{r+1}(i), k^{r+1}(j))$ for arcs $(i, j) \in R^{r+1}$.

Theorem 4.2: $DM^G(\bar{x}^r) = \bar{x}_0^{r+1}$ is a basic solution for $\overline{NP}^{r+1}$.

Proof:

We consider two cases: $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}$ empty and nonempty.

Case 1: $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1} \neq \{\emptyset\} \Rightarrow \overline{SNP}^{r+1}$ has at least one arc. We need only consider adjusted-reinstated arc pairs and arcs of type P-E, since all other original arcs of R^r generate aggregate arcs of $\overline{NP}^{r+1}$ which are either nonbasic, OOB or unaffected by refinement. Consider an adjusted-reinstated arc pair $(g,h) \in M^{r+1}$ and $(i,j) \in P_a^{r+1}$:

- i) If $(g,h) \in M^{r+1} \cap W(\bar{x}^r)$ then $\bar{x}_0^{r+1}(k^{r+1}(g), k^{r+1}(h))$ and $\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j))$ are both nonbasic in $\overline{NP}^{r+1}$.
- ii) If $(g,h) \in M^{r+1} \cap U(\bar{x}^r)$ then $\bar{x}_0^{r+1}(k^{r+1}(g), k^{r+1}(h))$ and $\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j))$ are both OOB.
- iii) If $(g,h) \in M^{r+1} \cap B(\bar{x}^r)$ then the values for $\bar{x}_0^{r+1}(k^{r+1}(g), k^{r+1}(h))$ and $\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j))$ are determined via solution of $\overline{SNP}^{r+1}$.
- iv) For arcs $(g,h) \in P_e^{r+1}$ values for $\bar{x}_0^{r+1}(k^{r+1}(g), k^{r+1}(h))$ are determined via solution of $\overline{SNP}^{r+1}$.

For definiteness and without loss of generality, assume that N_1^r was the subset simply refined into $N(r+1,1)$ and $N(r+1,n^{r+1})$. Consider now the basis tree associated with $\bar{x}^r$, call it $B(\bar{x}^r)$. The arcs of $B(\bar{x}^r)$ were generated by

either adjusted or unadjusted arcs, i.e.

$$B(\bar{x}^{r*}) = \{B(\bar{x}^{r*}) - M^{r+1}\} \cup \{B(\bar{x}^{r*}) \cap M^{r+1}\}$$

Let

$$|B(\bar{x}^{r*}) \cap M^{r+1}| = m_0$$

$$|P_e^{r+1}| = p_0$$

Remove from $B(\bar{x}^{r*})$:

- i) arcs $(k^r(i), k^r(j)), (i, j) \in B(\bar{x}^{r*}) \cap M^{r+1}$
- ii) the tail and head nodes of these arcs.

What has been removed from $B(\bar{x}^{r*})$ are basic arcs affected by partition refinement whose values in $\overline{NP}^{r+1}$ will be determined via solution of $\overline{SNP}^{r+1}$. Now, $\overline{SNP}^{r+1}$ consists of arcs

- i) $(k^{r+1}(i), k^{r+1}(j)), (i, j) \in B(\bar{x}^{r*}) \cap M^{r+1}$
- ii) $(k^{r+1}(i), k^{r+1}(j)), (i, j) \in I(B(\bar{x}^{r*}) \cap M^{r+1})$
- iii) $(k^{r+1}(i), k^{r+1}(j)), (i, j) \in P_e^{r+1}$

for a total of $m_0 + m_0 + p_0 = 2m_0 + p_0$ arcs, and nodes

- i) 1 and n^{r+1}
- ii) nodes m for all adjusted arcs $(m, l), (m, n^{r+1}), (l, m), (n^{r+1}, m)$

for a total of $m_0 + 2$ nodes. Solve $\overline{SNP}^{r+1}$. There are $m_0 + 2 - 1 = m_0 + 1$ basic arcs/variables for any solution of $\overline{SNP}^{r+1}$ since it is a minimum cost network flow problem.

Construct the following directed graph:

- i) Nodes 1, 2, ..., n^{r+1} for a total of n^{r+1} nodes.
- ii) Arcs (m, q) such that $k^{r+1}(i) = m, k^{r+1}(j) = q$, and $(i, j) \in B(\bar{x}^{r*}) - M^{r+1}$

There are $(n^r - 1) - m_0$ such arcs.

- iii) Arcs (m, q) such that $k^{r+1}(i) = m$, $k^{r+1}(j) = q$,
 $(i, j) \in \{B(\bar{x}^{r*}) \cap M^{r+1}\} \cup I(B(\bar{x}^{r*}) \cap M^{r+1}) \cup P_e^{r+1}$
and $\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j))$ basic in the
solution of $\overline{SNP}^{r+1}$. There are $m_0 + 1$ such arcs.

The directed graph has $n^r + 1$ nodes and $((n^r - 1) - m_0) + (m_0 + 1) = n^r$ arcs. The directed graph is connected. If it were not, this would imply that $B(\bar{x}^{r*})$ was not connected. Thus, since:

- i) the arcs included in this directed graph correspond precisely to the basic variables defined by DM^G
- ii) the directed graph is a tree since it has $n^r + 1$ nodes, n^r arcs and is connected
- iii) there is a one-to-one correspondence between basic solutions of a network problem and such trees
- iv) capacity and flow conservation constraints are satisfied, perhaps artificially, by the definition of DM^G and construction of $\overline{SNP}^{r+1}$

$DM^G(\bar{x}^{r*}) = \bar{x}_0^{r+1}$ is a basic solution of $\overline{NP}^{r+1}$

Case 2: $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1} = \{\emptyset\} \Rightarrow \overline{SNP}^{r+1}$ is degenerate and DM^G_D is applicable. Set

$H = \{m \in \bar{N}^{r+1} : m \text{ is in the same tree as } n^{r+1} \text{ in the partially constructed basic solution obtained from } DM^G_A, DM^G_B, DM^G_C\}$

After the application of DM^G_A , DM^G_B , and DM^G_C the arcs designated as basic result in two (disjoint) trees. One

consists of the nodes of H and adjacent basic arcs and the other consists of the nodes of $\bar{N}^{r+1} - H$ and adjacent basic arcs. DM^G_D) locates a nonbasic or OOB arc (i_0, j_0) such that:

$$i_0 \in H \text{ and } j_0 \in \bar{N}^{r+1} - H$$

or $i_0 \in \bar{N}^{r+1} - H, j_0 \in H$

This arc is designated as basic and connects the two trees into a single tree. Such an arc (i_0, j_0) must exist for if not, NP^0 would be disconnected. DM^G_E) is applied to ensure that flow conservation constraints are satisfied at nodes n^{r+1} and m_0 . Artificial arcs required by DM^G_E) are designated OOB. There are $n^r - 1$ basic arcs determined by DM^G_A) since no basic arcs were cut by the simple refinement of N^r . DM^G_D) adds one more basic arc. Since:

- i) there are n^r basic arcs and n^{r+1} nodes forming a connected, directed graph
- ii) there is a one-to-one correspondence between basic solutions and such connected directed graphs
- iii) flow conservation and capacity constraints are satisfied, perhaps artificially, by definition of DM^G

$$DM^G(\bar{x}^{r*}) = \bar{x}_0^{r+1} \text{ is a basic solution of } \bar{NP}^{r+1}.$$

Corollary 4.2: If $\bar{SNP}^{r+1}$ feasible, then $DM^G(\bar{x}^{r*}) = \bar{x}_0^{r+1}$ is a feasible basic solution of $\bar{NP}^{r+1}$.

Proof:

$$\text{Let: } R(r+1, A) = R^r - M^{r+1} \cap \{B(\bar{x}^r) \cup U(\bar{x}^r)\} \\ \cup I(M^{r+1} \cap U(\bar{x}^r))$$

$$R(r+1, B) = \{M^{r+1} \cap U(\bar{x}^r)\} \cup I(M^{r+1} \cap U(\bar{x}^r))$$

$$R(r+1, C) = \{M^{r+1} \cap B(\bar{x}^r)\} \cup I(M^{r+1} \cap B(\bar{x}^r)) \cup P_e^{r+1}$$

For the flow conservation constraints we have:

$$\begin{aligned} & \sum_{q: (m, q) \in \bar{A}(r+1)} \bar{x}^{r+1}(m, q) - \sum_{q: (q, m) \in \bar{A}(r+1)} \bar{x}^{r+1}(q, m) \\ &= \sum_{j: (i, j) \in R^{r+1}} \bar{x}^{r+1}(k^{r+1}(i), k^{r+1}(j)) - \sum_{j: (j, i) \in R^{r+1}} \bar{x}^{r+1}(k^{r+1}(j), k^{r+1}(i)) \\ &= \sum_{j: (i, j) \in R(r+1, A)} \bar{x}^{r+1}(k^{r+1}(i), k^{r+1}(j)) \\ &+ \sum_{j: (i, j) \in R(r+1, B)} \bar{x}^{r+1}(k^{r+1}(i), k^{r+1}(j)) \\ &- \sum_{j: (j, i) \in R(r+1, A)} \bar{x}^{r+1}(k^{r+1}(j), k^{r+1}(i)) \\ &- \sum_{j: (j, i) \in R(r+1, B)} \bar{x}^{r+1}(k^{r+1}(j), k^{r+1}(i)) \\ &+ \sum_{j: (i, j) \in R(r+1, C)} \bar{x}^{r+1}(k^{r+1}(i), k^{r+1}(j)) \\ &- \sum_{j: (j, i) \in R(r+1, C)} \bar{x}^{r+1}(k^{r+1}(j), k^{r+1}(i)) \\ &= \sum_{j: (i, j) \in R(r+1, E)} \bar{x}^{r+1}(k^{r+1}(i), k^{r+1}(j)) \\ &- \sum_{j: (j, i) \in R(r+1, E)} \bar{x}^{r+1}(k^{r+1}(j), k^{r+1}(i)) \\ &+ \tilde{b}(r+1, k^{r+1}(i)) \\ &= \tilde{b}(r+1, m) + \sum_{q: (m, q) \in \bar{E}} \bar{x}^{r+1}(m, q) - \sum_{q: (q, m) \in \bar{E}} \bar{x}^{r+1}(q, m) \\ &= \tilde{b}_m^{r+1} \end{aligned}$$

For capacity constraints, we have:

$$\begin{aligned} \underline{DM^G_A):} \quad & \bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) = \bar{x}^r(k^r(i), k^r(j)) \\ & \leq \bar{u}(r, k^r(i), k^r(j)) = \bar{u}(r+1, k^{r+1}(i), k^{r+1}(j)) \end{aligned}$$

and

$$\begin{aligned} & \bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) = \bar{x}^r(k^r(i), k^r(j)) \\ \Rightarrow & \bar{t}(r, k^r(i), k^r(j)) = \bar{t}(r+1, k^{r+1}(i), k^{r+1}(j)) \end{aligned}$$

for all $(i, j) \in R(r+1, A)$

$$\underline{DM^G_B):} \quad \bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) = \bar{u}(r+1, k^{r+1}(i), k^{r+1}(j))$$

for all $(i, j) \in R(r+1, B)$

$$\begin{aligned} \underline{DM^G_C):} \quad & \bar{t}(r+1, k^{r+1}(i), k^{r+1}(j)) \leq \bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j)) \\ & \leq \bar{u}(r+1, k^{r+1}(i), k^{r+1}(j)) \end{aligned}$$

for all $(i, j) \in R(r+1, C)$ since $\bar{x}_0^{r+1}(k^{r+1}(i), k^{r+1}(j))$ for $(i, j) \in R(r+1, C)$ is a feasible solution for SNP^{r+1} .

CHAPTER FIVE

THE PARTITION REFINEMENT PROCESS

In the previous two chapters we have shown how to create an aggregate problem $\overline{NP}^{r+1}$ and a corresponding basic solution $\bar{x}_0^{r+1}$ from $\overline{NP}^r$ and its optimal solution $\bar{x}^r$. No mention has been made however, about how the refinement N^{r+1} should be selected. That is the subject of this chapter.

Given N^r , $\overline{NP}^r$ and $\bar{x}^r$, we propose a criterion for selecting N^{r+1} that results in a relatively small subproblem $\overline{SNP}^{r+1}$ associated with the disaggregation map DM^G . If $\overline{SNP}^{r+1}$ is small and easily solved, then an advanced-start basic solution of $\overline{NP}^{r+1}$ is quickly obtained with little computational effort. Such a criterion would prove valuable to any network optimization code that employs aggregation-disaggregation concepts.

In addition to the criterion, a methodology is developed to specifically identify a simple refinement that results in an "acceptably small" subproblem. We define the size of the subproblem $\overline{SNP}^{r+1}$ to be its number of arcs and propose an optimal algorithm which locates a partition refinement resulting in a subproblem of minimum size. Quicker, more efficient heuristics yielding "small" $\overline{SNP}^{r+1}$'s are also provided along with an assessment of the degree of suboptimality between them and the optimal algorithm.

The partition refinement criterion and a heuristic for

generating small subproblems are incorporated into the complete algorithm of the next chapter.

We begin with a discussion of the logic underlying the partition refinement process of this chapter.

I. Rationale For The Partition Refinement Process

In the last chapter, we developed the general disaggregation map DM^G which provides a basic solution of $\overline{NP}^{r+1}$. Associated with DM^G is the network subproblem $\overline{SNP}^{r+1}$. If this subproblem is quickly and easily solved, then a complete basic solution of $\overline{NP}^{r+1}$ is obtained with minimal computational effort. Conversely, $\overline{SNP}^{r+1}$ could conceivably be as large as $\overline{NP}^{r+1}$ itself thus rendering DM^G inefficient from an algorithmic perspective. We hope to isolate and characterize those simple refinements of N^r that yield a small, tractable $\overline{SNP}^{r+1}$. We shall define the size of $\overline{SNP}^{r+1}$ to be the number of arcs that it contains.

Definition 5.1: By the size of $\overline{SNP}^{r+1}$, we shall mean:

$$|D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}|$$

Contributing to the size of $\overline{SNP}^{r+1}$ are i) basic adjusted-reinstated arc pairs and ii) arcs of type P-E. Each basic adjusted-reinstated arc pair contributes or generates two arcs in $\overline{SNP}^{r+1}$ and each P-E type arc contributes one. The following theorem describes the limits

of the size of $\overline{\text{SNP}}^{r+1}$.

Theorem 5.1: Given NP^0 , N^r , NP^r and $\bar{x}^{r*}$ for a simple refinement of N^r the resultant network subproblem $\overline{\text{SNP}}^{r+1}$ contains

$$\emptyset = |D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}| = 2n^r$$

arcs and between $\emptyset$ and n^r+1 nodes.

Proof:

When $D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1} = \{\emptyset\}$, $\overline{\text{SNP}}^{r+1}$ contains no arcs and no nodes (as in Example 4.5). For the particular instance where:

- i) all basic $\bar{x}^{r*}(m,q)$ are such that $m = m_0$ or $q = m_0$ for some $m_0 \in N^r$
- ii) a simple refinement of N^r cuts all basic submatrices $\gamma(r, m_0, q)$, $\gamma(r, q, m_0)$
- iii) $|P_e^{r+1}| = 2$

we have that

$$|D^{r+1} \cup I(D^{r+1}) \cup P_e^{r+1}| = 2(n^r-1) + 2 = 2n^r$$

and the number of nodes is of course n^r+1 .

When the simple refinement of N_m^r is viewed directly in the partitioned adjacency matrix, we observe that

- i) two arcs are contributed to $\overline{\text{SNP}}^{r+1}$ for each basic submatrix that is cut
- ii) one or two arcs are contributed to $\overline{\text{SNP}}^{r+1}$ if the submatrix γ_{mm}^r is cut.

Hence, in order to keep $\overline{SNP}^{r+1}$ small and tractable, we seek a simple refinement of N_m^r that cuts few basic submatrices. Definitions 5.1 and 5.2 develop and summarize necessary conditions for a submatrix to remain uncut by simple refinement.

Definition 5.2: For each basic submatrix γ_{mq}^r define:

$$R_{mq}^r = \{\tilde{N}^r(i) \in N_m^r: \gamma^r(i,j) = 1 \text{ for some } j: \tilde{N}^r(j) \in N_q^r\}$$

$$C_{mq}^r = \{\tilde{N}^r(j) \in N_q^r: \gamma^r(i,j) = 1 \text{ for some } i: \tilde{N}^r(i) \in N_m^r\}$$

where $\gamma^r(i,j)$ is the $(i,j)^{th}$ element of γ^r . The sets R_{mq}^r and C_{mq}^r shall be referred to as necessary conditions.

Example 5.1: The partitioned adjacency matrix of Example 2.1 is shown below with basic submatrices heavily outlined.

	j:	1	2	3	4	5	6	7	8	9	10	11	12
$\tilde{N}^1(j):$		2	3	4	6	11	12	1	5	7	8	9	10
1	2		1		1	1						1	
2	3			1								1	1
3	4									1			1
4	6						1					1	
5	11												
6	12												
7	1							1	1		1	1	
8	5									1			
9	7												
10	8												
11	9												
12	10												

The necessary conditions are:

<u>Basic $\gamma(1,m,q)$</u>	<u>$R(1,m,q)$</u>	<u>$C(1,m,q)$</u>
(1,1,3)	{4}	{8}
(1,1,4)	{2,3,6}	{9}
(1,1,5)	{3,4}	{10}
(1,2,3)	{1,5}	{8}

Consider, for example, the necessary condition $R(1,1,4) = \{2,3,6\}$. The interpretation is simply that it is necessary for original nodes 2, 3 and 6 to belong to the same subset of any simple refinement of N_1^1 if the basic submatrix $\gamma(1,1,4)$ is to remain uncut.

It is easy to see that there will be exactly $2(n^r-1)$ necessary conditions corresponding to the n^r-1 basic submatrices. Of these, many may apply to a single subset N_m^r whereas only a single necessary condition may apply to some other subset N_q^r . For a subset N_m^r , the necessary conditions that apply are R_{mq}^r , $q = 1, 2, \dots, n^r$ and C_{qm}^r , $q = 1, 2, \dots, n^r$. In Example 5.1, we have that R_{13}^1 , R_{14}^1 and R_{15}^1 all apply to N_1^1 whereas only C_{23}^1 applies to N_2^1 . We summarize the necessary conditions that are applicable to a particular subset in its necessary conditions matrix.

Definition 5.3: For each subset N_m^r , $m \in N^r$, define a matrix, T_m^r , called the necessary conditions matrix, with rows $i = 1, 2, \dots, |N_m^r|$ ordered according to $\tilde{N}^r(\tilde{i})$, where

$$\tilde{i} = \sum_{q=1}^{m-1} |N_q^r| + i$$

and whose columns correspond to the necessary conditions

$R_{mq}^r, C_{qm}^r, q = 1, 2, \dots, n^r$. Entries in the column R_{mq}^r are 1 for rows $i: \tilde{N}^r(i) \in R_{mq}^r$ and 0 otherwise. Entries in columns corresponding to C_{qm}^r are analogously defined.

Example 5.2: For Example 5.1, we have the necessary conditions matrices:

$$\begin{array}{cc} i & \tilde{N}^1(i) \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 6 \\ 5 & 11 \\ 6 & 12 \end{array} \quad \begin{array}{ccc} R_{13}^1 & R_{14}^1 & R_{15}^1 \\ \begin{bmatrix} & 1 & \\ & 1 & 1 \\ 1 & & 1 \\ & 1 & \\ & & \end{bmatrix} \end{array} = T_1^1$$

$$\begin{array}{cc} i & \tilde{N}^1(i) \\ 1 & 1 \\ 2 & 5 \\ 3 & 7 \end{array} \quad \begin{array}{c} R_{23}^1 \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{array} = T_2^1$$

$$\begin{array}{cc} i & \tilde{N}^1(i) \\ 1 & 8 \end{array} \quad \begin{array}{cc} C_{13}^1 & C_{23}^1 \\ \begin{bmatrix} 1 & 1 \end{bmatrix} \end{array} = T_3^1$$

$$\begin{array}{cc} i & \tilde{N}^1(i) \\ 1 & 9 \end{array} \quad \begin{array}{c} C_{14}^1 \\ \begin{bmatrix} 1 \end{bmatrix} \end{array} = T_4^1$$

$$\begin{array}{cc} i & \tilde{N}^1(i) \\ 1 & 10 \end{array} \quad \begin{array}{c} C_{15}^1 \\ \begin{bmatrix} 1 \end{bmatrix} \end{array} = T_5^1$$

From the necessary conditions matrix, it can be ascertained if a necessary condition will be violated by a proposed simple refinement. Such a violation would contribute two arcs to SNP^{r+1} .

Theorem 5.2: For any simple refinement of N_m^r : $N(r+1, m)$, $N(r+1, n^r+1)$, permute the rows of T_m^r to correspond to these two subsets to create the partitioned matrix:

$$T_m^r = \begin{array}{c} N(r+1, m) \\ \hline N(r+1, n^r+1) \end{array} \begin{bmatrix} T_{m1}^r \\ T_{m2}^r \end{bmatrix}$$

If there is a column k of the above matrix:

$$\begin{bmatrix} T_{m1}^r(\cdot, k) \\ T_{m2}^r(\cdot, k) \end{bmatrix} = \begin{array}{c} N(r+1, m) \\ \hline N(r+1, n^r+1) \end{array}$$

such that $T_{m1}^r(\cdot, k) \neq \emptyset$ and $T_{m2}^r(\cdot, k) \neq \emptyset$ then a necessary condition has been violated, a basic submatrix has been cut and two arcs are generated in $\overline{SNP}^{r+1}$.

Proof:

Column k corresponds to a particular necessary condition,

say R_{mq}^r . $T_{m1}^r(\cdot, k) \neq \emptyset$ and $T_{m2}^r(\cdot, k) \neq \emptyset$ implies that

i) $\gamma^r(i, j) = 1$ for some $\tilde{N}^r(\tilde{i}) \in N(r+1, m)$

and some $\tilde{N}^r(\tilde{j}) \in N(r+1, q)$

ii) $\gamma^r(i, j) = 1$ for some $\tilde{N}^r(\tilde{i}) \in N(r+1, n^r+1)$

and some $\tilde{N}^r(\tilde{j}) \in N(r+1, q)$

Thus γ_{mq}^r is cut by such a simple refinement. By definition,

the necessary condition has been violated. Two arcs $(g,h) \in M^{r+1}$, $(i,j) \in P_a^{r+1}$, (g,h) and $(i,j) \in D^{r+1} \cup I(D^{r+1})$ generate aggregate arcs in $\overline{SNP}^{r+1}$ according to DM^{GC} .

It is apparent that, for a given N_m^r , groups of necessary conditions are interrelated. The interrelated conditions of a group can be combined into a single, more stringent condition. More specifically, for two conditions, say $R_{m1,q1}^r$, $R_{m2,q2}^r$, if $R_{m1,q1}^r \cap R_{m2,q2}^r \neq \{\emptyset\}$ then the two conditions can be condensed into the single condition $R_{m1,q1}^r \cup R_{m2,q2}^r$.

Example 5.3: In Example 5.1, the necessary conditions for N_1^1 can be condensed to yield the single, more stringent condition:

$$R_{13}^1 \cup R_{14}^1 \cup R_{15}^1 = \{2,3,4,6\}$$

since $R_{13}^1 \cap R_{15}^1 = \{4\}$ and $R_{14}^1 \cap R_{15}^1 = \{3\}$.

This observation leads to the definition of the reduced necessary conditions matrix, $\tilde{T}_m^r$. This matrix reduces or condenses all of the necessary conditions associated with the subset N_m^r to a set of conditions that ensures no basic submatrix will be cut. Each column of $\tilde{T}_m^r$ represents a more stringent, implied necessary condition derived from the appropriate group of interrelated individual necessary conditions. The reduced necessary conditions matrix is most

easily defined by a procedure to create it, along with its interpretation.

Definition 5.4: For each subset N_m^r , $m \in \bar{N}^r$, the reduced necessary conditions matrix, $\tilde{T}_m^r$, is iteratively constructed as follows:

Step 0: Let T_m^r be the "current matrix."

Step 1: Add all of the entries in each row of the current matrix.

Step 2: If all row sums are ≤ 1 , STOP: $\tilde{T}_m^r$ is the current matrix, else GO TO Step 3.

Step 3: Pick the row with the greatest row sum. Ties are broken arbitrarily. Call this row i_0 .

Step 4: For each "1" entry in row i_0 , form the binary sum of the corresponding column vectors.

Step 5: Replace one of these columns with the binary sum vector and delete the others.

This is the new current matrix. GO TO Step 1.

At the conclusion of the procedure, each column of $\tilde{T}_m^r$ represents the condensed, more stringent condition implied by the corresponding group of individual interrelated conditions.

Example 5.4: The procedure of Definition 5.3 is applied to T_1^1 of Example 5.2.

Step 0: Current matrix is

$$\begin{array}{c}
 R_{13}^1 \quad R_{14}^1 \quad R_{15}^1 \\
 \begin{array}{c} 2 \\ 3 \\ 4 \\ 6 \\ 11 \\ 12 \end{array} \left[\begin{array}{ccc} & 1 & \\ & 1 & 1 \\ 1 & & 1 \\ & 1 & \\ & & \end{array} \right] = T_{11}^1
 \end{array}$$

<u>Step 1:</u>	<u>row i</u>	<u>$\tilde{N}^1(i)$</u>	<u>row sum</u>
	1	2	1
	2	3	2
	3	4	2
	4	6	1
	5	11	0
	6	12	0

Step 2: GO TO Step 3

Step 3: $i_0 = 2$

Step 4: "1" entries are in the second and third columns, so

$$R_{14}^1 \oplus R_{15}^1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Step 5: New current matrix:

$$\begin{array}{c}
 R_{13}^1 \quad (R_{14}^1 \oplus R_{15}^1) \\
 \begin{array}{c} 2 \\ 3 \\ 4 \\ 6 \\ 11 \\ 12 \end{array} \left[\begin{array}{ccc} & 1 & \\ & 1 & 1 \\ 1 & & 1 \\ & 1 & \\ & & \end{array} \right]
 \end{array}$$

<u>Step 1:</u>	<u>row i</u>	<u>$\tilde{N}^1(\tilde{i})$</u>	<u>row sum</u>
	1	2	1
	2	3	1
	3	4	2
	4	6	1
	5	11	0
	6	12	0

Step 2: GO TO Step 3.

Step 3: $i_0 = 3$

Step 4: Add columns 1 and 2 of current matrix:

$$R_{13}^1 + (R_{14}^1 + R_{15}^1)$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Step 5: New current matrix is:

$$R_{13}^1 + R_{14}^1 + R_{15}^1$$

$$\begin{matrix} 2 \\ 3 \\ 4 \\ 6 \\ 11 \\ 12 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

<u>Step 1:</u>	<u>row i</u>	<u>$\tilde{N}^1(\tilde{i})$</u>	<u>row sum</u>
	1	2	1
	2	3	1
	3	4	1
	4	6	1
	5	11	
	6	12	

Step 2: STOP: $\tilde{T}_1^1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

It is interesting to observe that the formation of the

reduced necessary conditions matrix can be viewed as an aggregation. If T_m^r is considered an adjacency matrix, we can rephrase the procedure of Definition 5.3 in terms of an iterative structural aggregation of the directed graph associated with T_m^r :

Step 0: The "current directed graph" is defined by viewing T_m^r as an adjacency matrix, and the "current partition" of the node set is taken to be all singleton subsets.

Step 1: Calculate the in-degree of each node corresponding to R_{mq}^r and C_{qm}^r .

Step 2: If the in-degree of each of the nodes equals 1,

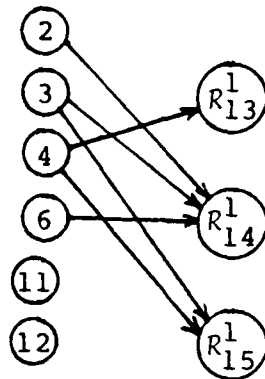
STOP: columns of $\tilde{T}_m^r$ are determined by the subsets of the current partition corresponding to nodes with out-degree ≥ 1 .

Step 3: Pick the node with greatest in-degree. Ties are broken arbitrarily. Call this node j_0 .

Step 4: Form the condensation (structural aggregation) of the current directed graph with respect to the partition formed by placing all nodes adjacent to j_0 in the same subset and leaving all others as singletons. GO TO Step 1.

Example 5.5: The above iterative aggregation procedure is performed to determine T_1^1 of Example 5.4.

Step 0:



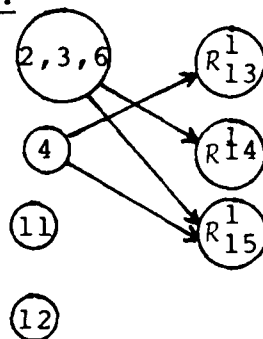
Step 1:

node	in-degree
R_{13}^1	1
R_{14}^1	3
R_{15}^1	2

Step 2: GO TO Step 3.

Step 3: $j_o = R_{14}^1$

Step 4:



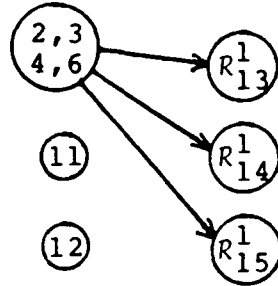
Step 1:

node	in-degree
R_{13}^1	1
R_{14}^1	1
R_{15}^1	2

Step 2: GO TO Step 3.

Step 3: $j_o = R_{15}^1$

Step 4:



Step 1:

node	in-degree
R_{13}^1	1
R_{14}^1	1
R_{15}^1	1

Step 2: STOP: $\tilde{T}_1^1$ determined by $\{2,3,4,6\}$.

II. Finding A Simple Refinement That Yields A Small $\overline{SNP}^{r+1}$

The reduced necessary conditions matrix yields necessary conditions that must be satisfied if no basic submatrix is to be cut by a simple refinement of N_m^r . These conditions are determined as follows:

- 1) If $\tilde{T}_m^r$ consists of a single column vector of 1's, denoted $\underline{1}$, then it is impossible to simply refine N_m^r without cutting at least one basic submatrix.
- 2) Nodes $\tilde{N}^r(i)$ and $\tilde{N}^r(j)$ must be included in the same subset of the simple refinement if row i of $\tilde{T}_m^r = \text{row } j \text{ of } \tilde{T}_m^r$.
- 3) Node $N^r(i)$ is left as a singleton subset if row i of $\tilde{T}_m^r = \underline{0}$.
- 4) After grouping nodes into subsets according to 2)

and 3) above, subsets are combined arbitrarily until exactly two remain to obtain a simple refinement of N_m^r .

We state these ideas formally in Theorem 5.3 subsequently.

Example 5.6: From $\tilde{T}_1^1$ of Example 5.4 we obtain that:

$$\{2,3,4,6\}, \{11\}, \{12\}$$

can be combined arbitrarily to obtain a simple refinement of N_1^1 that does not cut a basic submatrix, e.g.

$$N_1^2 = \{2,3,4,6\}, N_6^2 = \{11,12\}$$

is such a refinement.

Theorem 5.3: If A) $\tilde{T}_m^r = \underline{1}$ then it is not possible to refine N_m^r without cutting a basic submatrix, otherwise B) $\tilde{T}_m^r$ provides a simple refinement of N_m^r that does not cut any basic submatrix.

Proof:

A) Assume, to the contrary that $\tilde{T}_m^r = \underline{1}$ and that there exists a refinement, $N(r+1,m)$, $N(r+1,n^r+1)$, that does not cut a basic submatrix. The fact that no basic submatrix is cut by the refinement implies that each column of $\tilde{T}_m^r$, when partitioned according to $N(r+1,m)$, $N(r+1,n^r+1)$, has exactly one of the two resultant sub-vectors equal to the zero vector, i.e.

generic column of $\tilde{T}_m^r$:

$$\begin{array}{l} N(r+1, m) \\ N(r+1, n^r+1) \end{array} \begin{bmatrix} = \underline{0} \\ \neq \underline{0} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \neq \underline{0} \\ = \underline{0} \end{bmatrix}$$

If all such columns of $\tilde{T}_m^r$ satisfy this condition, then $\tilde{T}_m^r \neq \underline{1}$, a contradiction.

B) If $\tilde{T}_m^r \neq \underline{1}$, then we have two cases:

Case 1: $\tilde{T}_m^r$ is a single column vector with 1 and 0 entries.

Let $N(r+1, m) = \{\tilde{N}^r(\tilde{i}) : \text{row } i \text{ of } \tilde{T}_m^r = \underline{1}\}$ and $N(r+1, n^r+1) = \{\tilde{N}^r(\tilde{i}) : \text{row } i \text{ of } \tilde{T}_m^r = \underline{0}\}$. Such a refinement of N_m^r does not cut a basic submatrix since row i of $\tilde{T}_m^r = \underline{0}$ implies row i of $\tilde{T}_m^r = \underline{0}$, which implies $\tilde{N}^r(\tilde{i}) \notin R_{m,q}^r$, $\tilde{N}^r(\tilde{i}) \notin C_{q,m}^r$ for all q .

Therefore, there does not exist rows or columns in basic submatrices with 1 entries in row or column i . Hence, basic submatrices are not cut by the specified refinement of N_m^r .

Case 2: $\tilde{T}_m^r$ consists of more than one column and each column is, of course, a vector with entries of 1 and 0.

There cannot be a column vector $= \underline{1}$ since this would imply that $\tilde{T}_m^r$ is further reducible to a single column vector $= \underline{1}$. Further, there is no more than a single 1 entry in each row of $\tilde{T}_m^r$, for if this were not the case $\tilde{T}_m^r$ would be further reducible. Create a partition of N_m^r as follows:

- i) If row i of $\tilde{T}_m^r = \text{row } j$ of $\tilde{T}_m^r$, $\tilde{N}^r(\tilde{i})$ and $\tilde{N}^r(\tilde{j})$ are placed in the same subset.

ii) If row i of $\tilde{T}_m^r = \underline{0}$, $\tilde{N}^r(\tilde{i})$ is left as a singleton subset.

iii) Denote the subsets obtained by i) and ii) as $N_m^r(k)$, $k = 1, 2, \dots, k_0$.

Reorder the rows and columns of $\tilde{T}_m^r$ according to this partition to yield a matrix of the form

$$\begin{array}{l} N_m^r(k_1) \\ N_m^r(k_2) \\ \cdot \\ \cdot \\ \cdot \\ N_m^r(k_s) \\ N_m^r(k_t) \end{array} \left[\begin{array}{ccccc} \underline{1} & \underline{0} & \dots & \underline{0} \\ \underline{0} & \underline{1} & \underline{0} & \dots & \underline{0} \\ & & & & \\ & & & & \\ & & & & \\ \underline{0} & \underline{0} & \underline{0} & \underline{0} & \dots & \underline{1} \\ \underline{0} & \underline{0} & \underline{0} & \underline{0} & \dots & \underline{0} \end{array} \right]$$

where $\underline{0}$ and $\underline{1}$ are column vectors all of whose entries are 0 and 1 respectively. Any refinement of N_m^r of the form

$$N(r+1, m) = \bigcup_{k=1}^{\bar{k}} N_m^r(k), \quad \bar{k} < k_0$$

$$N(r+1, n^{r+1}) = \bigcup_{k=\bar{k}+1}^{k_0} N_m^r(k)$$

is a refinement that does not cut a basic submatrix. If it did, it would imply that there exists a column of $\tilde{T}_m^r$ such that there are 1 entries in some row i_0 : $\tilde{N}^r(\tilde{i}_0) \in N(r+1, m)$ and some row j_0 : $\tilde{N}^r(\tilde{j}_0) \in N(r+1, n^{r+1})$. This in turn would imply that there is a column of $\tilde{T}_m^r$ with 1 entries in the rows as i_0 and j_0 , $\tilde{N}^r(\tilde{i}_0)$ and $\tilde{N}^r(\tilde{j}_0)$ are in different subsets of the refinement. This is of course impossible as

can be seen in the permuted form of $\tilde{T}_m^r$ above and by the way the refinement is constructed.

In instances where the condition of Theorem 5.3 A) holds, Theorem 5.4 identifies a simple refinement that cuts the minimum number of basic submatrices.

Theorem 5.4: Suppose $\tilde{T}_m^r = \underline{1}$. A simple refinement of N_m^r that cuts the minimum number of basic submatrices is determined as follows:

Step 1: Partition N_m^r into subset $N_m^r(k)$,
 $k = 1, 2, \dots, k_0$ such that:

$$\begin{aligned} \tilde{N}^r(i) &\in N_m^r(k) \\ \text{and} \quad \tilde{N}^r(j) &\in N_m^r(k) \end{aligned}$$

where

$$\begin{aligned} \tilde{i} &= \sum_{q=1}^{m-1} |N_q^r| + i \\ \text{and} \quad \tilde{j} &= \sum_{q=1}^{m-1} |N_q^r| + j \end{aligned}$$

iff row i of $T_m^r =$ row j of T_m^r .

If $k_0 = 1$, STOP: any refinement of N_m^r cuts one basic submatrix.

Step 2: For each subset $N_m^r(k)$, $k = 1, 2, \dots, k_0$ define:

$$\begin{aligned} C(k) &= \{j: r_m(i,j) = 1, \text{ for some } i\} \\ &= \{\text{columns with a 1 entry in rows} \\ &\quad i: \tilde{N}^r(i) \in N_m^r(k)\} \end{aligned}$$

$$\begin{aligned}
Z(k) &= \{j \in C(k) : T_m^r(i, j) = \emptyset \text{ for} \\
&\quad \tilde{N}^r(\tilde{i}) \in N_m^r - N_m^r(k)\} \\
&= \{\text{columns } j \in C(k) \text{ such that entries in} \\
&\quad \text{rows } i: \tilde{N}^r(\tilde{i}) \notin N_m^r(k) \text{ are all } \emptyset\}
\end{aligned}$$

Step 3: The number of basic submatrices that are cut by the simple refinement $N(r+1, m) = N_m^r(k)$, $N(r+1, n^r+1) = N_m^r - N_m^r(k)$, for a given k is exactly $V_1(N_m^r(k)) = r(k) - |Z(k)|$, where $r(k)$ is the row sum of T_m^r for $i: \tilde{N}^r(\tilde{i}) \in N_m^r(k)$

Step 4: The minimum number of basic submatrices that must be cut any refinement of N_m^r is given by:

$$\begin{aligned}
V_1(N_m^r) &= \min_k V_1(N_m^r(k)) = V_1(N_m^r(\tilde{k})) \\
\text{For } k_0 > 1, \text{ the refinement } N(r+1, m) &= N_m^r(\tilde{k}), \\
N(r+1, n^r+1) &= N_m^r - N_m^r(\tilde{k}) \\
&\text{yields this minimum.}
\end{aligned}$$

Proof:

There are two cases.

Case 1: $T_m^r = T_m^r = \underline{1}$. There is only one subset $N_m^r(k)$, i.e. $k_0 = 1$. Thus $N_m^r = N_m^r(1)$ and $|C(1)| = 1$, $|Z(k)| = \emptyset$, $r(1) = 1$. Any refinement of N_m^r yields $V_1(N_m^r(1)) = r(1) - \emptyset = 1$.

Case 2: T_m^r consists of more than one column. For any subset $N_m^r(k)$ permute the rows and columns of T_m^r to obtain the following partitioned version of T_m^r :

$$\begin{aligned}
\frac{N_m^r - N_m^r(k)}{N_m^r(k)} &= \frac{\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & \underbrace{T_{22}}_{Z(k)} & T_{23} \end{bmatrix}}{\underbrace{\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \end{bmatrix}}_{C(k)}} \\
&= \begin{bmatrix} T_{11} & \emptyset & T_{13} \\ \hline \underline{1} & \underline{1} & \underline{0} \end{bmatrix}
\end{aligned}$$

Each column of T_{11} must have at least one 1 entry or else that column would be a column of T_{12} . Therefore, necessary conditions corresponding to columns $C(k) - Z(k)$ are violated. This implies, by Theorem 5.2, that basic submatrices corresponding to columns $C(\tilde{k}) - Z(\tilde{k})$ are cut. There are $V_1(N_m^r(\tilde{k}))$ such basic submatrices. Since $N_m^r(\tilde{k})$ is the subset providing the minimum $V_1(N_m^r(k))$, the refinement described is Step 5 cuts exactly $V_1(N_m^r(\tilde{k}))$ submatrices.

Example 5.6: The algorithm of Theorem 5.4 is applied to Example 5.1.

$$\begin{array}{ll}
\text{Step 1:} & 2 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
\tau_1^1 = & 3 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& 4 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& 6 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& 11 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& 12 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
\end{array}
\begin{array}{l}
N_1^1(1) = \{2, 6\} \\
N_1^1(2) = \{3\} \\
N_1^1(3) = \{4\} \\
N_1^1(4) = \{11, 12\}
\end{array}$$

$$\tau_2^1 = \begin{matrix} 1 \\ 5 \\ 7 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ \end{bmatrix} \quad \begin{matrix} N_2^1(1) = \{1, 5\} \\ N_2^1(2) = \{7\} \end{matrix}$$

Step 2: N_1^1 : $C(1) = \{2\}$ $Z(1) = \{\emptyset\}$
 $C(2) = \{2, 3\}$ $Z(2) = \{\emptyset\}$
 $C(3) = \{1, 3\}$ $Z(3) = \{1\}$
 $C(4) = \{\emptyset\}$ $Z(4) = \{\emptyset\}$

N_2^1 : $C(1) = \{1\}$ $Z(1) = \{1\}$
 $C(2) = \{\emptyset\}$ $Z(2) = \{\emptyset\}$

Step 3: N_1^1 : $V_1(N_1^1(1)) = r(1) - |Z(1)| = 1 - 0 = 1$
 $V_1(N_1^1(2)) = r(2) - |Z(2)| = 2 - 0 = 2$
 $V_1(N_1^1(3)) = r(3) - |Z(3)| = 2 - 1 = 1$
 $V_1(N_1^1(4)) = r(4) - |Z(4)| = 0 - 0 = 0$

N_2^1 : $V_1(N_2^1(1)) = r(1) - |Z(1)| = 1 - 1 = 0$
 $V_1(N_2^1(2)) = r(2) - |Z(2)| = 0 - 0 = 0$

Step 4: N_1^1 : $V_1(N_1^1) = \text{Min } V_1(N_1^1(k)) = 0$ and $\tilde{k} = 4$. So
 $N_1^2 = \{11, 12\}$, $N_6^2 = \{2, 3, 4, 6\}$.

N_2^1 : $V_1(N_2^1) = \text{Min } V_1(N_2^1(k)) = 0$ and $\tilde{k} = 1$ or 2 .
So $N_2^2 = \{1, 5\}$, $N_6^2 = \{7\}$.

Either of the above simple refinements given in Step 4 are easily verified to cut no basic submatrix.

Theorem 5.4 can be extended to an optimal algorithm which locates a particular subset and a particular refinement of that subset that results in an $\overline{\text{SNP}}^{r+1}$ of

minimum size. Such an algorithm is described in Theorem 5.5.

Theorem 5.5: The following algorithm will locate a subset $N(r, m_0)$ and a simple refinement of that subset which results in an $\overline{SNP}^{r+1}$ with minimum number of arcs.

Step 1: For all subsets $N_m^r, m \in \bar{N}^r$, find the subsets $N_m^r, m \in L \subset \bar{N}^r$ that cut the minimum number of basic submatrices. Call this minimum number of basic submatrices that are cut s_0 . This is determined using Theorems 5.3 and 5.4.

Step 2: For each subset $N_m^r, m \in L$, partition into $N_m^r(k), k \in \{1, 2, \dots, k_0\} = K$, as follows:

A) If $\tilde{T}_m^r = \underline{1}$: nodes $\tilde{N}^r(\tilde{i})$ and $\tilde{N}^r(\tilde{j})$ are in the same subset if row i of $\tilde{T}_m^r =$ row j of $\tilde{T}_m^r$. Node $\tilde{N}^r(\tilde{i})$ is left as a singleton if row i of $\tilde{T}_m^r = \underline{0}$.

B) If $\tau_m^r = \tilde{T}_m^r = \underline{1}$: $N_m^r(k)$ are all singleton subsets, each containing one $\tilde{N}^r(\tilde{i}) \in N_m^r$.

C) If $\tilde{T}_m^r = \underline{1} \neq \tau_m^r$: nodes $\tilde{N}^r(\tilde{i})$ and $\tilde{N}^r(\tilde{j})$ are in the same subset if row i of $\tau_m^r =$ row j of $\tau_m^r \neq \underline{0}$.

Node $\tilde{N}^r(\tilde{i})$ is left as a singleton if

row i of $T_m^r = \underline{0}$.

Step 3: For each subset N_m^r , $m \in L$, of the possible ways to combine the subsets $N_m^r(k)$ into two subsets, select the refinement $N(r+1, m)$, $N(r+1, n^{r+1})$ that minimizes the resultant $|P_e^{r+1}|$, and call this minimum p_m^{r+1} .

For this refinement, let

$$V_m^r = 2s_0 + p_m^{r+1}.$$

Step 4: Select $N(r, m_0)$, $m_0 \in L$, such that:

$$V(r, m_0) = \min_m \{V_m^r\}$$

The resultant $\overline{SNP}^{r+1}$ has $V(r, m_0)$ arcs.

Proof: Selecting $N(r, m_0)$, $m_0 \in L$, implies that $\overline{SNP}^{r+1}$ has $2s_0$ arcs generated from the s_0 adjusted-reinstated arc pairs in the s_0 basic submatrices that will be cut by refining $N(r, m_0)$ as specified. Step 2 determines the subsets $N(r, m_0)(k)$ such that any refinement of the form

$$N(r+1, m_0) = \bigcup_{k \in \bar{K} \subset K} N(r, m_0)(k)$$

$$N(r+1, n^{r+1}) = \bigcup_{k \in \bar{K} - K \subset K} N(r, m_0)(k)$$

cuts exactly s_0 basic submatrices. Step 3 determines a refinement, not necessarily unique, that i) cuts s_0 basic submatrices, and ii) adds the minimum number of P-E type arcs, $p(r+1, m_0)$. Thus, for this refinement of $N(r, m_0)$, $\overline{SNP}^{r+1}$ consists of $2s_0 + p(r+1, m_0)$ arcs. Any other refinement of $N(r, m_0)$ yields an $\overline{SNP}^{r+1}$ with $2s_0 + |P(r+1, m_0)| = 2s_0 + p(r+1, m_0)$ arcs. Refining any other

subset N_m^r , $m \in L$, $m \neq m_0$, results in an $\overline{SNP}^{r+1}$ with $2s_0 + p_m^r \geq 2s_0 + p(r+1, m_0)$ arcs. Refining any subset N_m^r , $m \notin L$, results in an $\overline{SNP}^{r+1}$ with at least $2(s_0 + 1) \geq 2s_0 + p(r+1, m_0)$ arcs. Thus $\overline{SNP}^{r+1}$ is of minimum size for the specified refinement of $N(r, m_0)$.

Example 5.7: The procedure of Theorem 5.5 is applied to

Example 5.1.

$$\begin{aligned} \text{Step 1: } N_1^1: & \begin{array}{c} 2 \\ 3 \\ 4 \\ 6 \\ 11 \\ 12 \end{array} \begin{bmatrix} & 1 & \\ & 1 & 1 \\ 1 & & 1 \\ & 1 & \\ & & \end{bmatrix} = \tau_1^1 \\ & \begin{array}{c} 2 \\ 3 \\ 4 \\ 6 \\ 11 \\ 12 \end{array} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \\ \end{bmatrix} = \tilde{\tau}_1^1 \\ N_1^2: & \begin{array}{c} 1 \\ 5 \\ 7 \end{array} \begin{bmatrix} 1 \\ 1 \\ \end{bmatrix} = \tau_2^1 = \tilde{\tau}_2^1 \end{aligned}$$

So $L = \{1, 2\}$ and $s_0 = \emptyset$. Thus it is possible to refine N_1^1 and N_2^1 without cutting a basic submatrix.

$$\begin{aligned} \text{Step 2: } A) \quad N_1^1(1) &= \{2, 3, 4, 6\}, \quad N_1^1(2) = \{11, 12\} \\ N_2^1(1) &= \{1, 5\}, \quad N_2^1(2) = \{7\} \end{aligned}$$

$$\begin{aligned} \text{Step 3: } N_1^1: \quad \text{For } N_1^2 &= \{2, 3, 4, 6\}, \quad N_6^2 = \{11, 12\} \\ |P_e^2| &= 1 = p_2^2, \\ v_1^1 &= 2s_0 + p_2^2 = 2(\emptyset) + 1 = 1 \\ N_2^1: \quad \text{For } N_2^2 &= \{1, 5\}, \quad N_6^2 = \{7\} \\ |P_e^2| &= 1 = p_2^2, \end{aligned}$$

$$v_2^1 = 2s_0 + p_2^2 = 2(\emptyset) + 1 = 1$$

Step 4: $\text{Min } \{v_1^1, v_2^1\} = \text{Min } \{1, 1\} = 1$

and $m_0 = 1$ or 2 . Selecting either:

$$N_1^2 = \{2, 3, 4, 6\}, N_6^2 = \{11, 12\}$$

$$\text{or } N_2^2 = \{1, 5\}, N_6^2 = \{7\}$$

results in an $\overline{\text{SNP}}^2$ with $v_1^1 = v_2^1 = 1$ arc.

The algorithm possibly requires explicit enumeration of the

$$\begin{bmatrix} k_0 \\ 2 \end{bmatrix}, k_0 = |N_m^r|$$

possibilities in Step 3, unless of course a refinement yielding $|p_e^{r+1}| = \emptyset$ is quickly found. Considering the fact that $|p_e^{r+1}| = 2$, we can eliminate the search in Step 3 and be assured that the resultant $\overline{\text{SNP}}^{r+1}$ contains no more than two arcs more than the minimum. We shall adopt this approach in the heuristics that follow. The columns $Z(k)$ described in Theorem 5.3 are also ignored to enhance the efficiency of the heuristics.

Additional features designed to balance the trade off between the size on $\overline{\text{SNP}}^{r+1}$ and the effort required by the heuristic can also be incorporated. One such feature orders the subsets according to their potential for cutting basic submatrices. Such a measure is proposed in the following definition.

Definition 5.4: Let $F^r = \{m: |N_m^r| > 1\}$. For each subset N_m^r ,

$m \in F^r$, define π_m^r , called its cutting potential, as

$$\pi_m^r = |\{q: \bar{x}^{r*}(m,q) \text{ is basic and } |\gamma_{mq}^r| > 1\}| \\ + |\{q: \bar{x}^{r*}(q,m) \text{ is basic and } |\gamma_{qm}^r| > 1\}|$$

π_m^r is equivalent to the number of "nonsingleton aggregate arcs" to which aggregate node m is adjacent in the basis tree of $\bar{x}^{r*}$. The heuristic will include a criterion to select subsets as candidates for refinement in increasing order of their cutting potential.

An "acceptable size" parameter, called $\tilde{V}^r$, is adjustable in the heuristic. This feature possibly precludes the checking of all n^r subsets and can be set based upon the characteristics of the problem at hand. The heuristic halts when a refinement has been found which results in an $\overline{SNP}^{r+1}$ of size $0 \leq |D^{r+1} \cup I(D^{r+1} \cup P_e^{r+1})| \leq \tilde{V}^r$. The selection of $\tilde{V}^r$ might be based, in part, upon $|F^r|$ and $\text{Min } \pi_m^r$. If both $|F^r|$ and $\text{Min } \pi_m^r$ are large, the size parameter might be adjusted upward. This potentially reduces the effort necessary to locate an acceptable refinement at the cost of potentially increasing the size of $\overline{SNP}^{r+1}$.

Heuristic 1 For Partition Refinement:

Step 1: Calculate π_m^r for $m \in F^r$.

Step 2: Form the ordered sequence $F = \{m_i\}$, $m_i \in F^r$, where m_i preceeds m_j if $\pi(r, m_i) < \pi(r, m_j)$.

Step 3: Set $i = 1$, $\bar{V} = \infty$, $L = \emptyset$ and specify $\tilde{V}^r$.

Step 4: If $2(\pi(r, F(i)) + 2 \leq \tilde{V}^r$: STOP: any refinement of $N(r, F(i))$ yields an $\overline{SNP}^{r+1}$ with no more than $2\pi(r, F(i)) + 2$ arcs. Otherwise, GO TO Step 5.

Step 5: Calculate $T(r, F(i))$. If $T(r, F(i)) = \underline{1}$, GO TO Step 8. Otherwise, GO TO Step 6.

Step 6: From $T(r, F(i))$, form $\tilde{T}(r, F(i))$. If $\tilde{T}(r, F(i)) = \underline{1}$, GO TO Step 8. Otherwise GO TO Step 7.

Step 7: (It is possible to refine $N(r, F(i))$ without cutting any basic submatrices.) Let:

$$N(r+1, F(i)) = \{j: \text{row } j \text{ of } \tilde{T}(r, F(i)) = \text{row } 1 \text{ of } \tilde{T}(r, F(i))\}$$

$$N(r+1, n^{r+1}) = N(r, F(i)) - N(r+1, F(i))$$

$$N(r+1, m) = N(r, m) \text{ for } m \neq F(i).$$

STOP: $\overline{SNP}^{r+1}$ consists of at most two arcs.

Step 8: (Any refinement of $N(r, F(i))$ cuts at least one basic submatrix.)

A) Compute the row sums of $T(r, F(i))$.

B) Denote the minimum row sum as s_0 .

C) Compute $V(r, F(i)) = 2s_0 + 2$.

If $V(r, F(i)) > \tilde{V}^r$, GO TO Step 9.

Otherwise, GO TO Step 8D).

D) If $T(r, F(i)) = \underline{1}$ or = a matrix of ones, GO TO Step 8F). Otherwise, GO TO 8E).

E) Let:

$$N(r+1, F(i)) = \{j: \text{row } j \text{ of } T(r, F(i)) = \text{row } j_0 \text{ of } T(r, F(i))\}$$

where j_0 is the row with row sum s_0

$$N(r+1, n^r+1) = N(r, F(i)) - N(r+1, F(i))$$

$$N(r+1, m) = N(r, m) \text{ for } m \neq F(i).$$

STOP: $\overline{SNP}^{r+1}$ has at most $V(r, F(i))$ arcs.

F) $N(r, F(i))$ can be arbitrarily refined into two subsets:

$N(r+1, F(i))$ chosen arbitrarily

$$N(r+1, n^r+1) = N(r, F(i)) - N(r+1, F(i))$$

$$N(r+1, m) = N(r, m) \text{ for } m \neq F(i).$$

STOP: $\overline{SNP}^{r+1}$ has at most $V(r, F(i))$ arcs.

Step 9: If $V(r, F(i)) < \bar{V}$, set $\bar{V} = V(r, F(i))$, $L = F(i)$.

Otherwise GO TO Step 10.

Step 10: Set $i = i + 1$. If $i > |F^r|$, GO TO Step 11.

Otherwise, GO TO Step 4.

Step 11: (All subsets have been checked and none can be refined to yield an $\overline{SNP}^{r+1}$ of acceptable size.)

A) If $T(r+1, L) = \underline{1}$ or = a matrix of ones, GO TO Step 11C). Otherwise GO TO Step 11B).

B) Let:

$$N(r+1, L) = \{j: \text{row } j \text{ of } T(r, L) = \text{row } j_0 \text{ of } T(r, L)\}$$

where j_0 is the row with row sum s_0

$$N(r+1, n^r+1) = N(r, L) - N(r+1, L)$$

$$N(r+1, m) = N(r, m) \text{ for } m \neq L.$$

STOP: $\overline{SNP}^{r+1}$ has at most V arcs.

C) $N(r, L)$ can be refined arbitrarily:

$N(r+1, L)$ chosen arbitrarily

$$N(r+1, n^r+1) = N(r, L) - N(r+1, L)$$

$$N(r+1, m) = N(r, m) \text{ for } m \neq L.$$

STOP: $\overline{SNP}^{r+1}$ has at most $\bar{V}$ arcs.

Example 5.8: The heuristic is applied to Example 5.1. $F^1 = \{1, 2\}$.

Step 1: $\pi_1^1 = |\{4, 5\}| + |\{\emptyset\}| = 3$

$$\pi_2^1 = |\{3\}| + |\{\emptyset\}| = 1$$

Step 2: $F = \{2, 1\}$

Step 3: $i = 1, L = \emptyset, \bar{V} = \infty, \tilde{V}^1 = 2$

Step 4: $2(\pi(1, F(1))) + 2 = 2(\pi_2^1) + 2$
 $= 2(1) + 2 = 4 > \tilde{V}^1$

Step 5: $T(1, F(1)) = T_2^1 = \frac{1}{5} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Step 6: $\tilde{T}(1, F(1)) = \tilde{T}_2^1 = \frac{1}{5} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$N(r+1, F(i)) = N_2^2 = \{1, 5\}$$

$$N(r+1, n^r+1) = N_6^2 = \{7\}$$

STOP: $\overline{SNP}^2$ has at most two arcs.

($\overline{SNP}^2$ for the specified refinement in fact has one arc.)

An even simpler, more direct heuristic is Heuristic 2 below. This method arbitrarily refines the subset with the minimum π_m^r . The resultant $\overline{SNP}^{r+1}$ will contain at most $2\pi_m^r + 2$ arcs, usually a relatively small number. The tradeoff for the potential increase in the size of $\overline{SNP}^{r+1}$ is the marked reduction in the effort required to locate a refinement

yielding a "small" $\overline{SNP}^{r+1}$.

Heuristic 2 For Partition Refinement:

Step 1: Calculate π_m^r for $m \in F^r$.

Step 2: Find m_0 such that $\min_m \pi_m^r = \pi(r, m_0)$.

Step 3: Refine subset $N(r, m_0)$ arbitrarily. The resultant $\overline{SNP}^{r+1}$ has no more than $2\pi(r, m_0) + 2$ arcs.

The worst case suboptimality in this heuristic is $2\pi(r, m_0) + 2$.

CHAPTER SIX

THE COMPLETE ALGORITHM AND ITS IMPLEMENTATION

Chapters two through five laid the necessary conceptual and theoretical ground work for a network optimization algorithm which employs aggregation and disaggregation methods. Such an algorithm is the subject of this chapter.

As with the design and implementation of any algorithm, there are many choices which must be made. The algorithm to be developed in this chapter will be a specific one which:

1. uses the $RS^{\min}$ respecification maps
2. uses the aggregation procedure of chapter two
3. uses the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ partition refinement procedure of chapter three
4. uses the DM^G disaggregation map developed in chapter four
5. uses Heuristic 2 for partition refinement given in chapter five
6. uses Lee's stopping criterion (See "Stopping Criterion" and "Theorem 13," Appendix B).

The $RS^{\min}$ respecification maps were chosen for two reasons:

1. $RS^{\min}$ requires only integer addition and subtraction operations and is thus compatible with highly efficient network codes.
2. The aggregate problem $\overline{NP}^r$ is a relaxation of NP^0 .

Lee's stopping criterion is used because it provides both an optimality test and an implicit disaggregation map that maps an optimal solution of $\overline{NP}^r$ into a optimal solution

of NP^0 .

I. Basic Logic Of The Complete Algorithm

Figure 6.1 gives the general flow diagram of the complete algorithm. It is assumed at Block 2 that N^1 , the initial node set partition, "makes sense" for the problem context at hand.

The Aggregation Procedure of Block 3 includes the structural respecification maps for supplies, costs, upper and lower bounds. We have selected the $RS^{\min}$ respecification maps for the reasons cited above. It should be noted however that other respecification maps, such as the popular "fixed weight" schemes, could be substituted for $RS^{\min}$ in steps 2 and 3 of the aggregation procedure. We will consider only aggregation schemes that result in aggregate problems that are in fact capacitated transshipment problems.

In Block 4, the aggregate problem is solved. We will assume that a primal network code is used as the solver. We discuss in particular the GNET code developed by Bradley, Brown and Graves [2]. This code is reviewed in detail in Appendix C.

Block 5 provides a stopping criterion for the algorithm. We shall use the stopping criterion first proposed by Lee. Other stopping criteria can easily be substituted for this choice, however. For instance, the a

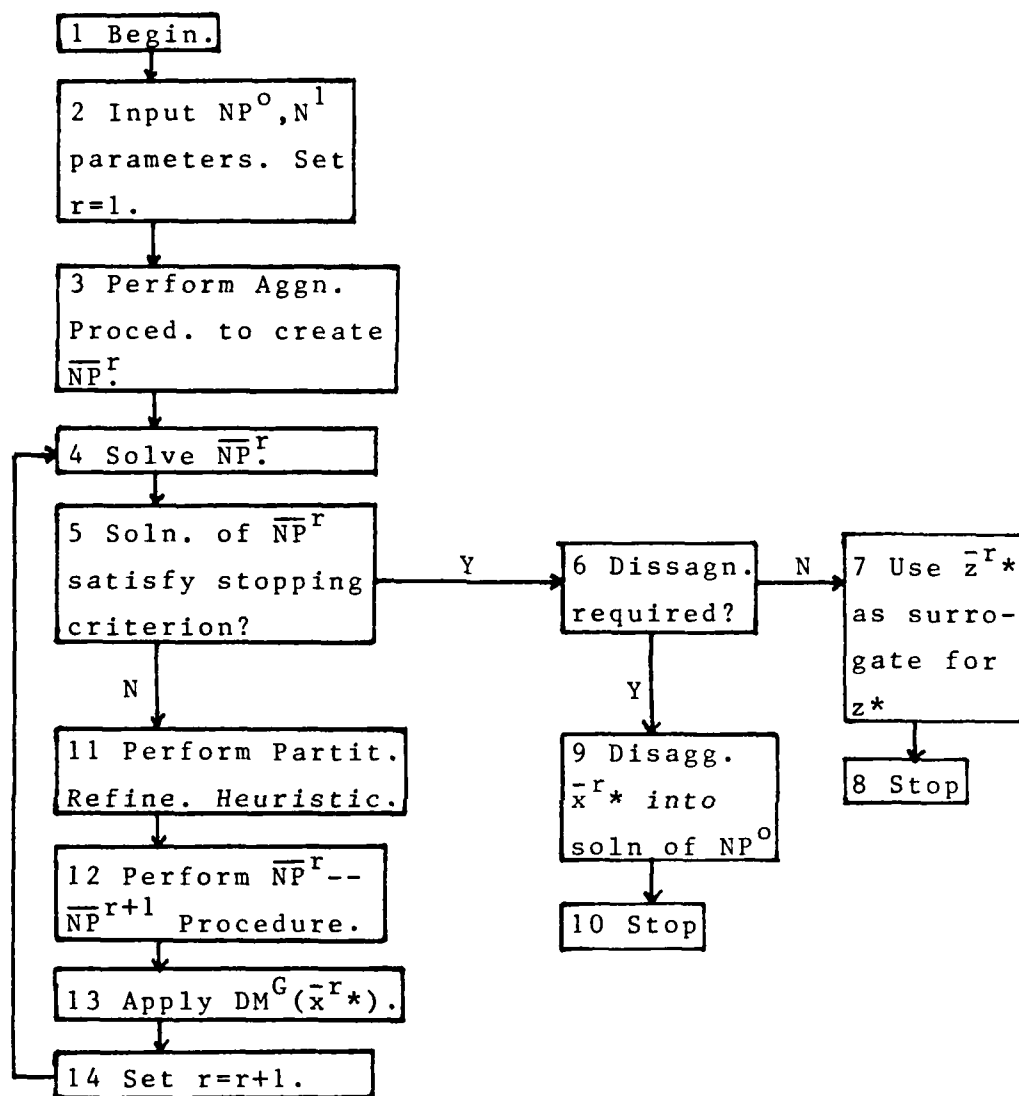


Figure 6.1: Logic Diagram For Complete Algorithm

posteriori error bounds developed by Zipkin in Proposition Z4 (see page 25) could be used to provide a stopping rule. This would of course necessitate using Zipkin's "fixed weight" respecification maps in Block 3 and the partition refinement procedure would have to be limited to the class of partitions satisfying Zipkin's assumptions A1 through A3 (see pages 23).

When $\overline{NP}^r$ is used to provide an optimal value surrogate for NP^0 , Block 7 is appropriate and the application of a disaggregation map is not necessary. However when $\overline{NP}^r$ is used as an optimization surrogate, a disaggregation map must be defined in Block 9 which maps an optimal solution of $\overline{NP}^r$ into a feasible solution of NP^0 . We shall use the disaggregation map implicit in Lee's Theorem 13. In Proposition 6.1 below we show that this disaggregation map is a special instance of the DM^G disaggregation map. Again, any viable disaggregation map could be used in place of this one in Block 9.

Proposition 6.1: The disaggregation map of Theorem 13 of [39] is a special case of DM^G .

Proof: $\bar{x}^r$ satisfies the "Stopping Criterion" of [39] if, for all (m,q) such that $(k(i),k(j)) = (m,q)$, $(i,j) \in B(\bar{x}^r) \cup U(\bar{x}^r)$ the condition $|A_{mq}^r| = 1$ holds. The disaggregation map of Theorem 13 of [39] requires that

$$1. \quad x^0(i,j) = \bar{x}^r(k^r(i),k^r(j)) \text{ for all}$$

$$(i,j) \in B(\bar{x}^r) \cup U(\bar{x}^r)$$

$$2. \ x^0(i,j) = 0 \text{ for all } (i,j) \in W(\bar{x}^r)$$

The arcs (i,j) of 1. are "unaffected arcs" in the current terminology. All arcs (i,j) in 2. generate either "unaffected nonbasic arcs" or nonbasic arcs which are members of "adjusted-reinstated pairs." All of these arcs fall under the purview of DM^G . Hence, when the "Stopping Criterion" is satisfied, the disaggregation map of Theorem 13 is equivalent to DM^G .

It is obvious that the complete algorithm presented in Figure 6.1 is finite. Since the primal simplex algorithm used as the solver in Block 4 is finite and since we can perform at most $n-n^1$ simple partition refinements in Block 11, the complete algorithm must be finite so long as n , the number of nodes in NP^0 , is finite.

II. An Example Of The Complete Algorithm

In this section we apply the complete algorithm to Example 2.1. We assume that $\overline{NP}^r$ is used as an optimization surrogate requiring the disaggregation of the solution of $\overline{NP}^r$ into a feasible solution of NP^0 , provided that $\overline{NP}^r$ is not equal to NP^0 .

The first iteration will be illustrated step by step with results shown graphically. Subsequent iterations are summarized in Table 6.1.

Example 6.1:

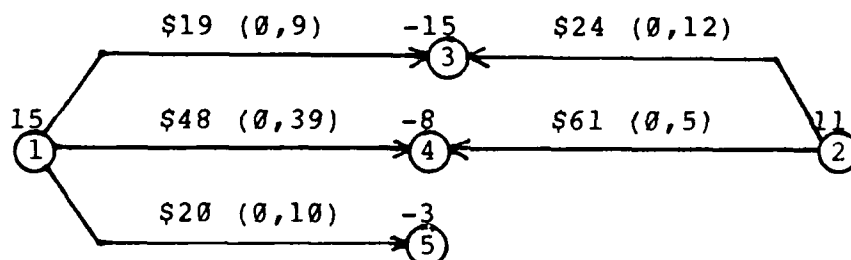
Block 1: Begin.

Block 2: Input NP^0 , N^1 . Set $r = 1$. The NP^0 network parameters are input as given in Table 2.1. The initial node set partition is:

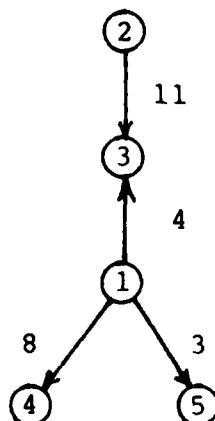
$$\begin{aligned} N^1 &= \{N_1^1, N_2^1, N_3^1, N_4^1, N_5^1\} \\ &= \{\{2,3,4,6,11,12\}, \{1,5,7\}, \{8\}, \{9\}, \{10\}\} \\ r &= 1 \end{aligned}$$

Block 3: Perform the Aggregation Procedure To Create $\overline{NP}^r$.

Using N^1 above the resultant aggregate problem $\overline{NP}^1$ is shown graphically below.



Block 4: Solve $\overline{NP}^r$. The solution of $\overline{NP}^1$ is shown graphically below.



Block 5: Solution Of $\overline{NP}^r$ Satisfy Stopping Criterion? No.

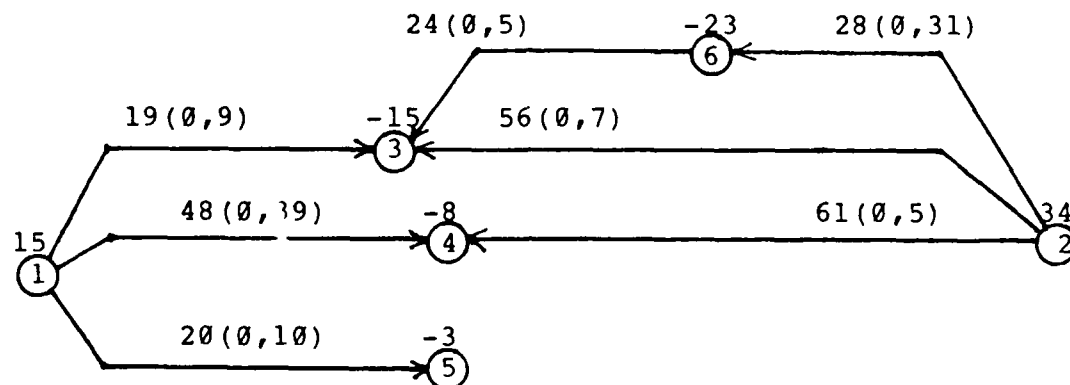
Go to Block 11.

Block 11: Perform Partition Refinement Heuristic.

i:	1	2	3	4	5
π_i^1 :	2	1	N/A	N/A	N/A

Thus N_2^1 is refined into $N_2^2 = \{1\}$, $N_6^2 = \{5,7\}$

Block 12: Perform $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ Procedure. This step yields the next aggregate problem $\overline{NP}^2$:

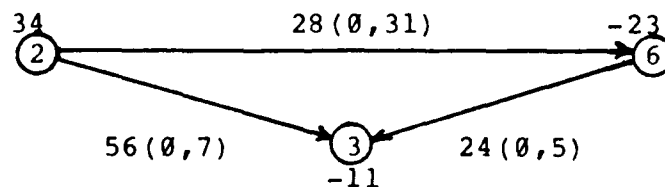


Block 13: Apply $DM^G(\bar{x}^{r*})$ To Obtain a Starting Solution Of $\overline{NP}^{r+1}$. Applying DM^G yields:

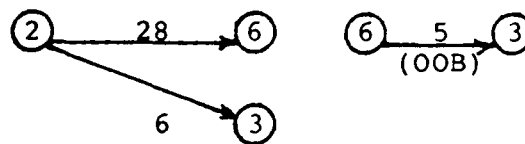
DM^G A): $\bar{x}_0^2(1,3) = \bar{x}^1(1,3) = 4$ (Basic)
 $\bar{x}_0^2(1,4) = \bar{x}^1(1,4) = 8$ (Basic)
 $\bar{x}_0^2(1,5) = \bar{x}^1(1,5) = 3$ (Basic)
 $\bar{x}_0^2(2,4) = \bar{x}^1(2,4) = 0$ (Nonbasic)

DM^G B): N/A

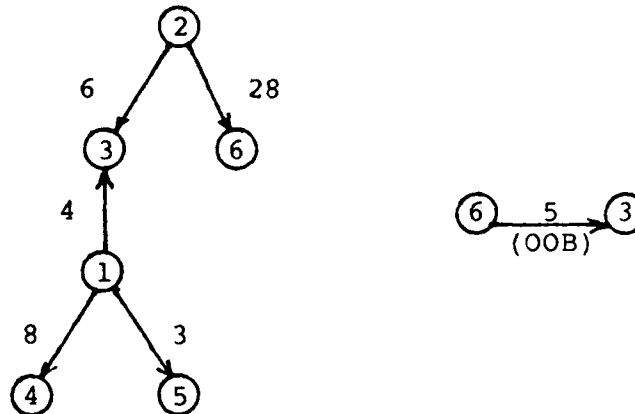
DM^G C): $\overline{SNP}^2$ is shown graphically below.



The solution of $\overline{SNP}^{r+1}$ is:



The complete basic solution of $\overline{NP}^2$ is then:



This and all subsequent iterations of the complete algorithm are summarized in Table 6.1 below. Figure 6.2 shows the behavior of the scale of aggregation, M^r , throughout the execution of the algorithm.

III. Implementation

A computer code called AGNET has been written with the expressed purpose of testing and validating the results of this research. AGNET is built around the GNET network optimization code of Bradley, Brown and Graves. GNET is described in detail in Appendix C. AGNET uses GNET as the solver for $\overline{NP}^r$ and $\overline{SNP}^{r+1}$ in the execution of the algorithm.

AGNET embodies the complete algorithm of Figure 6.1 with one exception. At Block 4, $\overline{NP}^r$ is solved "from scratch" using the initial artificial solution constructed by GNET rather than beginning with $\bar{x}_0^r = DM^G(\bar{x}^{r-1*})$. This

<u>r</u>	<u>N^r</u>	<u>$\overline{NP}^r$</u> <u>(1) (2)</u>		<u>M^r</u>	<u>$\overline{SNP}^{r+1}$</u> <u>(3) (4)</u>		<u>Notes:</u>
1	{2,3,4,6,11,12} {1,5,7}{8}{9}{10}	5	5	1.32	3	3	$\overline{SNP}^2$ feasible, $\bar{x}_0^2 = \bar{x}^{2*}$
2	{2,3,4,6,11,12} {1}{8}{9}{10} {5,7}	6	7	1.11	3	2	$\overline{SNP}^3$ feasible, $\bar{x}_0^3 = \bar{x}^{3*}$
3	{2,3,4,6,11,12} {8}{9}{10}{5} {7}	7	8	0.95	3	3	$\overline{SNP}^4$ feasible, $\bar{x}_0^4 = \bar{x}^{4*}$
4	{2,3,4}{1}{8}{9} {10}{5}{7} {6,11,12}	8	10	0.74	3	3	$\overline{SNP}^5$ feasible, $\bar{x}_0^5 = \bar{x}^{5*}$
5	{2,3,4}{1}{8}{9} {10}{5}{7}{6} {11,12}	9	12	0.52	0	0	$\overline{SNP}^6$ degenerate, DM ^G _D , DM ^G _E) invoked, $\bar{x}_0^6$ infeasible
6	{2,3,4}{1}{8}{9} {10}{5}{7}{6}{11} {12}	10	12	0.43	3	3	$\overline{SNP}^7$ feasible $\bar{x}_0^7 = \bar{x}^{7*}$
7	{2}{1}{8}{9} {10}{5}{7}{6} {11}{12}{3,4}	11	14	0.22	3	3	$\overline{SNP}^8$ infeasible, $\bar{x}_0^8$ infeasible
8	{2}{1}{8}{9}{10} {5}{7}{6}{11} {12}{3}{4}	12	16	0.00	N/A		$\overline{NP}^8 = NP^0$

KEY: (1) Number of nodes in $\overline{NP}^r$
(2) Number of arcs in $\overline{NP}^r$
(3) Number of nodes in $\overline{SNP}^{r+1}$
(4) Number of nodes in $\overline{SNP}^{r+1}$

Table 6.1: Summary Of Example 6.1

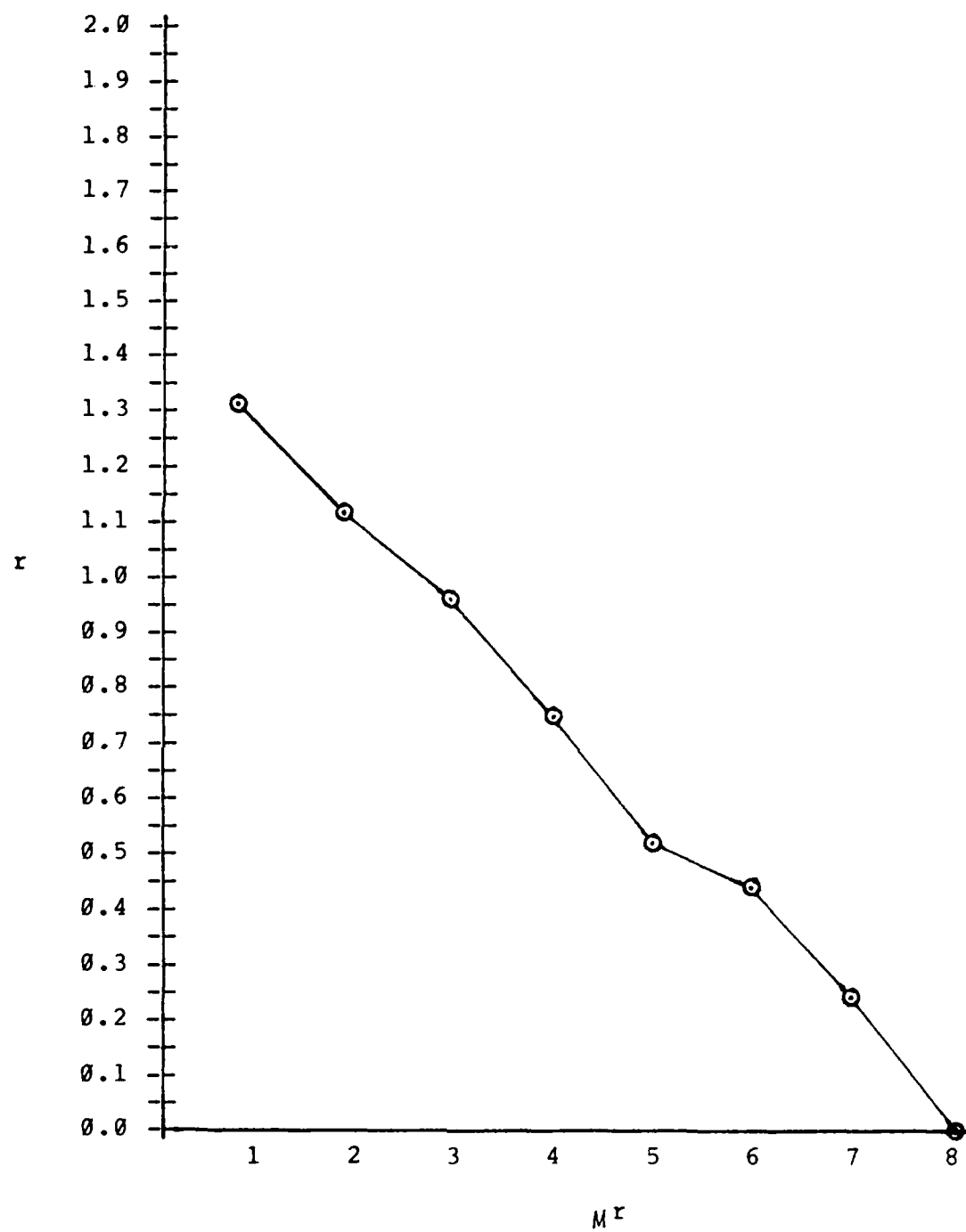


Figure 6.2: M^r Versus r For Example 6.1

aspect of the algorithm bears no direct relationship to the theoretical developments of this research and thus does not require testing and verification. Also, given the purpose of AGNET, it has been written with a view toward careful evaluation rather than efficiency.

In this section we describe the implementational logic of each of the blocks of Figure 6.1. Appendix D provides the actual AGNET Fortran code as well as definitions of all variables used in the program. A complete program listing is not included because GNET is a copyrighted code.

Block 2: Input NP^0 , N^1 . Set $r = 1$.

This portion of the AGNET code is straight forward. Original network parameters are entered in the same format as required by GNET (the popular SHARE format). For each arc the tail node, head node, cost, upper bound and lower bound are stored in the arrays $TO()$, $HO()$, $CO()$, $UBO()$ and $LBO()$ respectively. (The "O" suffix indicates "original.") Supplies are entered by specifying arcs from a "super source" to the original supply nodes with zero cost and with upper and lower bounds equal to supply. Demand is entered analogously using arcs from demand nodes to a "super sink." With MO equaling the number of original nodes in NP^0 , the super source and super sink are designated as nodes $MO+1$ and $MO+2$ respectively. The first node set partition is entered in a node length array $K()$, where $K(i) = j$ implies that node i is an element of N_j^r .

Whereas GNET does not store the information for arcs from the super source or to the super sink, AGNET permanently retains this information along the "real arc" parameters in the TO(), HO(), CO(), UBO() and LBO() arrays. This is necessary since supplies and demands for the aggregate nodes must be recomputed during the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ process and since supplies and demands must be adjusted when constructing $\overline{SNP}^{r+1}$.

Block 3: Perform The Aggregation Procedure To Create $\overline{NP}^r$.

The first thing that AGNET does in preparing to create $\overline{NP}^r$ is sort the original arcs according to arc subset. Arc subset A_{ij}^r preceeds A_{st}^r if $i \leq s$ and $j \leq t$. The procedure used to sort the arcs was taken from Horowitz [30], page 364. This sorting procedure is given in Appendix D-II.

Two linked lists, LINK() and ASETPT(), are used to maintain the sorted arc subsets. $ASETPT(i) = j$ implies that the first arc of arc subset i is the original arc stored in location j of the TO(), HO(), CO(), LBO() and UBO() arrays. To find all of the arcs of arc subset i , LINK() is iterated until the entry in LINK() equals $ASETPT(i+1)$, the first arc of arc subset $i+1$.

AALOC(,) is a two dimensional array that indicates which arc subset corresponds to a particular aggregate arc of $\overline{NP}^r$. $AALOC(i,j) = k$ implies that aggregate arc (i,j) is "composed of" the original arcs in arc subset k . $AALOC(i,j)$

$= \emptyset$ implies that $A_{ij}^r = \{\emptyset\}$. Once the arc sort is completed and ASETPT() has been determined, the AALOC(,) array is loaded by AGNET. During this phase of the program, the number of aggregate arcs of $\overline{NP}^r$ is also computed and stored in the variable NAGSET. Appendix D-III describes the loading of ASETPT() and AALOC(,) arrays.

Next AGNET fills the arrays NTIL(), NTILIN() and NSETPT(). NTIL() corresponds directly to the ordered sequence $\tilde{N}^r$ as defined in Definition 2.3. NTILIN() is the inverse of NTIL(), i.e. NTILIN(i) = j implies that NTIL(j) = i. The array NSETPT() merely indicates the locations of NTIL() where nodes of any particular subset reside. Specifically, the nodes of subset N_i^r can be found in locations NSETPT(i) through NSETPT(i+1)-1 of NTIL(). The portion of AGNET that accomplishes the filling of these arrays is in Appendix D-IV.

The aggregate problem $\overline{NP}^r$ is straight forwardly constructed by applying $RS1^{\min}$, $RS3^{\min}$ and $RS4^{\min}$ to each arc subset A_{ij}^r , $i \neq j$, and $RS4^{\min}$ to each node subset. The resultant aggregate problem is stored in the following arrays:

AT():	tail nodes of aggregate arcs of $\overline{NP}^r$
AH():	head nodes of aggregate arcs of $\overline{NP}^r$
CMIN():	costs of aggregate arcs of $\overline{NP}^r$
ACP():	capacities (upper bound) of aggregate arcs of $\overline{NP}^r$
ALB():	lower bound on aggregate arcs of $\overline{NP}^r$

ASUP(): supplies of aggregate nodes of $\overline{NP}^r$

REPARC(): representative arcs of $\overline{NP}^r$

NUMARC(): the number of original arcs in each arc subset

The supply and demand arcs from the super source and to the super sink are automatically constructed and stored in the AT(), AH(), CMIN(), ACP() and ALB() arrays.

When $r = 1$, $\overline{NP}^r$ is fed directly to GNET for solution. The arrays AT(), AH(), CMIN(), ACP() and ALB() are synchronously iterated as the parameters of each aggregate arc are provided to GNET. When $r \neq 1$, the node set partition is refined at Block 11, the $\overline{NP}^r \rightarrow \overline{NP}^{r+1}$ procedure is performed at Block 12 and DM^G is applied to the solution of $\overline{NP}^r$ at Block 13 to create an advanced-start basic solution for $\overline{NP}^{r+1}$. The original arcs are resorted according to N^{r+1} and all of the above mentioned arrays are updated. Appendix D-V gives the details of how the aggregate problem is constructed by AGNET.

Block 4: Solve $\overline{NP}^r$.

During this phase of execution, AGNET feeds the current aggregate problem to GNET for solution. GNET solves $\overline{NP}^r$ "from scratch" beginning with a totally artificial solution consisting of arcs with zero cost from supply nodes to the artificial root node and arcs with cost "Big M" from the root node to the demand nodes.

Block 5: Solution Of $\overline{NP}^r$ Satisfy Stopping Criterion?

When GNET has solved $\overline{NP}^r$, AGNET marks the aggregate arcs as basic, non-basic or out-of-basis at upper bound (OOB). Aggregate basic arcs are marked with a negative sign bit on NUMARC() and OOB aggregate arcs are marked with a negative sign bit on ACP(). Solution values are stored in the SOLN() array. Once this has been accomplished, it is an easy task to check the stopping criterion. Lee's stopping criterion is, in the terminology of AGNET:

if all nonzero basic arcs and OOB arcs are "composed of" only a single original arc, i.e. NUMARCS() = 1, then the stopping criterion is satisfied.

The Fortran code for the marking of basic and OOB arcs and for the stopping criterion is given in Appendix D-VI.

Block 6: Disaggregation Required?

If $\overline{NP}^r$ is used as an optimal value surrogate, disaggregation of the solution $\bar{x}^{r*}$ is not required. If $\overline{NP}^r$ is used as an optimization surrogate, a disaggregation map is necessary. As mentioned previously, the disaggregation map implicit in Lee's Theorem 13 will be used in this implementation. This disaggregation map is, in the current notation:

$$DM^L(\bar{x}^{r*}(m,q)) = \bar{x}^*(i,j) = \begin{cases} \bar{x}^{r*}(m,q) & \text{for } \bar{x}^{r*}(m,q) > 0, \\ & i \in N_m^r, j \in N_q^r, (i,j) \in A \\ 0 & \text{for } \bar{x}^{r*}(m,q) = 0, i \in N_m^r, \\ & j \in N_q^r, (i,j) \in A \end{cases}$$

Thus disaggregation under this mapping simply entails

1. translating arc/variable indices from $\overline{NP}^r$ designations to NP^0 designations
2. setting flows according to the above rule for "non-singleton" aggregate arcs with zero flow.

The current version of AGNET simply prints a message that the stopping criterion has been met and explains this simple disaggregation map.

Block 7: Use The Optimal Objective Function Value Of $\overline{NP}^r$ As An Optimal Value Surrogate For The Objective Function Value Of NP^0 .

Block 8: Stop.

Blocks 7 and 8 are self-explanatory and require no special AGNET coding.

Block 9: Disaggregate Solution Of $\overline{NP}^r$ Into Solution Of NP^0 :
 $DM^G(\bar{X}^r) = \bar{X}_0$. This step has been explained above under Block 6.

Block 10: Stop.

Block 11: Perform The Partition Refinement Heuristic.

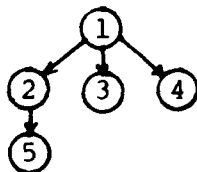
There are four main steps associated with this block:
1) transformation of the basis tree/predecessor graph into a binary tree, 2) determination of the node "cutting potentials," π_m^r , 3) selection of the node subset to be refined, and 4) refinement of the selected node subset.

The basis tree/predecessor graph is transformed by

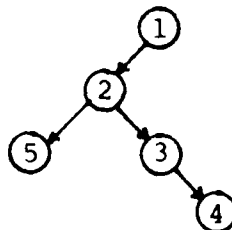
AGNET into a binary tree. A binary tree is a tree in which each node has either no children (successors), a single child designated either the left child or right child, or both a left child and a right child. An example of such a binary tree is given below. This transformation is accomplished via one pass through the basis tree using the IT() preorder traversal array of GNET. The purpose of this transformation is to facilitate direct computation of node cutting potentials and to simplify the search for nodes of the set H during DM^G_D). Operating on the binary tree eliminates the need to search through the complete basis subtree for each node to determine its node cutting potential. How AGNET uses the binary tree representation to determine H is discussed under Block 13 below.

The representation as a binary tree requires two additional arrays: LLINK() and RLINK(). LLINK(i)=j implies that node j is node i's left-most child in the basis tree/predecessor graph. RLINK(j)=k implies that node k is node j's right sibling. The following is an example of a binary tree representation of a basis tree.

Basis Tree:



Binary Tree



The LLINK() and RLINK() arrays for the above binary tree

representation are:

i:	1	2	3	4	5
LLINK(i):	2	5	0	0	0
RLINK(i):	0	3	4	0	0

An array LNDV() keeps track of the "Last Node of this Depth Visited" during the preorder IT() traversal of the basis tree/predecessor graph. This array, along with the GNET depth array, D(), affords easy determination of parent-child and sibling relationships necessary to effect the transformation.

Once the transformation to a binary tree representation is complete, it is a simple matter to compute node cutting potentials for each aggregate node. Each node, m , is first checked for the condition $|N_m^r| > 1$. If this condition is true, each arc adjacent to node i in the basis tree is checked for the condition that $|A_{mj}^r| > 1$ or $|A_{jm}^r| > 1$. Each time that this condition holds true, the m^{th} location of the node potential array, PI(), is incremented by one. The arcs adjacent to aggregate node m in the basis tree are easy to find using this binary tree representation since these arcs are also adjacent to:

- 1) the predecessor of m , $P(m)$,
- 2) m 's left child, j , given by $LLINK(m)=j$,
- 3) all of j 's siblings, which can be determined by iterating on $RLINK()$, beginning at $RLINK(j)$ until $RLINK()=0$.

Once the entries of the PI() array have been calculated, PI() is searched for its minimum value and the

location of $PI()$ containing this minimum value is designated as $ISTAR$. Thus, the subset to be refined is N_{ISTAR}^r .

Partition refinement is effected simply by redesignating $K(i)=n^r+1$ for $i \leq [s/2]$ where $s = |N_{ISTAR}^r|$ and $[]$ indicates only the integer portion of the enclosed.

The Fortran coding for these four steps is contained in Appendix D-VII.

Block 13: Apply $DM^G(\bar{x}^{r*}) = \bar{x}_0^{r+1}$ To Obtain A Starting Solution Of $\overline{NP}^{r+1}$.

The $\overline{NP}^r$ information is temporarily stored in the following arrays:

$AALOCO()$: stores $AALOC()$ for $\overline{NP}^r$
 $NUMAO()$: stores $NUMARC()$ for $\overline{NP}^r$
 $ACP()$: stores $ACP()$ for $\overline{NP}^r$
 $SOLNO()$: stores $SOLN()$ for $\overline{NP}^r$

This is accomplished prior to construction of $\overline{NP}^{r+1}$ of course.

$AALOC()$ (for $\overline{NP}^{r+1}$) holds information crucial for determining which part of DM^G is appropriate for each arc of $\overline{NP}^{r+1}$. The following observations are fundamental:

Observation 1: If $AALOC(ISTAR, n^r+1) \neq \emptyset$, then $(ISTAR, n^r+1)$ was generated by an arc $(i, j) \in P_e^{r+1}$. $AALOC(ISTAR, n^r+1)$ is marked negative indicating that arc $(ISTAR, n^r+1)$ is to be included in $\overline{SNP}^{r+1}$. $AALOC(n^r+1, ISTAR)$ is interpreted analogously.

Observation 2: If $AALOC(ISTAR, I) \neq \emptyset$ and $AALOC(n^r+1, I) \neq \emptyset$,

arc subset $A_{ISTAR,I}^r$ has been cut by the simple refinement of N_{ISTAR}^r . If $\bar{x}^r \star (ISTAR, I)$ is basic, i.e. $NUMAO(AALOCO(ISTAR, I)) < 0$, $AALOC(ISTAR, I)$ and $AALOC(n^{r+1}, I)$ are both marked negative to indicate that arcs $(ISTAR, I)$ and (n^{r+1}, I) are to be included in $\overline{SNP}^{r+1}$. If $\bar{x}^r \star (ISTAR, I)$ is OOB, i.e. $ACPO(AALOCO(ISTAR, I)) < 0$, DM_B^G is applicable and $\bar{x}_0^{r+1}(ISTAR, I)$ and $\bar{x}_0^{r+1}(n^{r+1}, I)$ are set equal to their upper bounds and marked as OOB. If $\bar{x}^r \star (ISTAR, I)$ is nonbasic, i.e. $NUMAO(AALOCO(ISTAR, I)) > 0$ and $ACPO(AALOCO(ISTAR, I)) > 0$, then $\bar{x}_0^{r+1}(ISTAR, I)$ and $\bar{x}_0^{r+1}(n^{r+1}, I)$ are set equal to zero and marked as nonbasic. Directly analagous results hold for the situation where $AALOC(I, ISTAR) \neq \emptyset$ and $AALOC(I, n^{r+1}) \neq \emptyset$.

Observation 3: If $AALOC(ISTAR, I) = \emptyset$ and $AALOC(n^{r+1}, I) \neq \emptyset$, then arc $(ISTAR, I)$ of $\overline{NP}^r$ is redesignated (n^{r+1}, I) in $\overline{NP}^{r+1}$. The same is true if $AALOC(I, ISTAR) = \emptyset$ and $AALOC(I, n^{r+1}) \neq \emptyset$.

The above observations along with the information in $AALOCO()$, $NUMAO()$ and $ACPO()$ is sufficient to identify which part of DM^G is applicable to each arc of $\overline{NP}^{r+1}$.

$\overline{SNP}^{r+1}$ is constructed by reference to the arrays $SNODE()$ and $SNP()$. $SNODE(I) = \emptyset$ implies that node I is a node of $\overline{SNP}^{r+1}$ and $SNP(K)$ gives the location (in $AT()$, $AH()$, etc.) of an aggregate arc of $\overline{SNP}^{r+1}$ for $K \leq SNAGSE$. The arrays $SAT()$, $SAH()$, $SCMIN()$, $SACP()$, $SALB()$ and $SSOLN()$ store information relevant to the aggregate arcs of $\overline{SNP}^{r+1}$. $ADSUP()$ holds the adjusted supplies for the nodes of $\overline{SNP}^{r+1}$.

Since GNET requires that nodes be numbered consecutively, the nodes of $\overline{SNP}^{r+1}$ must be renumbered before $\overline{SNP}^{r+1}$ is fed to GNET for solution. Upon solution, the nodes of $\overline{SNP}^{r+1}$ are translated back to their original $\overline{NP}^{r+1}$ designations. The arrays COR() and CORIN() are used for this purpose.

Supplies for the nodes of $\overline{SNP}^{r+1}$ are adjusted by searching the corresponding rows and columns of the AALOC() array for positive entries. For node J, if AALOC(I,J) is positive then the supply of node J is increased by the amount SOLN(AALOC(I,J)). (Note that SOLN() values for arcs corresponding to positive AALOC() entries have already been determined by DM^G_A and DM^G_B .) If AALOC(J,I) is positive, then the supply of node J is reduced by the amount SOLN(AALOC(J,I)).

Next, the (artificial) arcs from the super source and to the super sink are constructed using the same logic as in the construction of $\overline{NP}^r$. At this point, the variable SWITCH is set to 1 indicating to GNET that the problem to be read and solved is a subproblem and not the full $\overline{NP}^{r+1}$ problem. $\overline{SNP}^{r+1}$ is read and solved by GNET, node and arc designations are translated to $\overline{NP}^{r+1}$ designations and the solution is marked and stored in SOLN() completing the DM^G_C portion of DM^G .

If $\overline{SNP}^{r+1}$ is degenerate and DM^G_D is appropriate, AGNET uses the preorder traversal structure IT() and the

predecessor array $P()$ of GNET and the binary tree representation of the basis to determine the sets H and $N^{r+1} - H$. One pass through the binary tree marks $SNODE(I)$ either all negative or all positive for all $I \in H$. All nodes of the set $N^{r+1} - H$ are marked opposite in sign as those of H . With $SNODE()$ so marked, it is a simple matter to locate a nonbasic or OOB arc with $SNODE(\text{tail}) * SNODE(\text{head}) < 0$ to redesignate as basic.

DM^G_E is applicable when $\overline{SNP}^{r+1}$ is degenerate. Flow conservation is checked at node n^{r+1} . If flow conservation is violated, a pair of artificial OOB arcs are added with appropriate flow connecting nodes n^{r+1} and ISTAR to the root.

The AGNET coding for DM^G is given in Appendix D-VIII. Appendix E gives the computer output for AGNET applied to the example of Section II.

IV. Testing And Verification

Over fifty aggregate CTP's were constructed, solved and verified using AGNET. These problems were all constructed from the original problem NP^0 given in Example 2.1 using different initial node set partitions. This original problem, along with the 4,097,072 possible partitions of the node set, proved versatile and rich enough to test all aspects of the complete algorithm and the corresponding components of the AGNET code.

The computer output of each of the test problems was carefully verified. Two methods were employed in this verification. In the first method, the basis trees were constructed and all constraints were checked for consistency. The second method employed a direct and straight forward algebraic verification.

There were no discrepancies, inconsistencies or errors detected in any of the tests conducted.

CHAPTER SEVEN

SUMMARY AND CONCLUSIONS

This research has investigated aggregation as applied to the capacitated transshipment problem. While this topic has been studied by other researchers, we provide here:

1. a framework for aggregation of the capacitated transshipment problem that is motivated by implementational considerations
2. a categorization of important arc types as part of this framework and as a necessary prerequisite for a new disaggregation map and complete algorithm
3. an aggregation procedure along with a measure of the scale of aggregation and an illustration of the use of this measure in aggregation design
4. a methodology for partition refinement and for creation of a sequence of aggregate problems
5. a disaggregation method that maps a feasible basic solution of an aggregate problem to a basic solution of a refined aggregate problem
6. a criterion for partition refinement that results in a "small," easily solved subproblem associated with the disaggregation map
7. a complete algorithm that incorporates all of the above aggregation-disaggregation concepts
8. an implementation to test and verify the theoretical results of this work.

If aggregation is to be an effective computational tool, it is virtually necessary that the information provided by the solution of the current aggregate model be

AD-A170 681

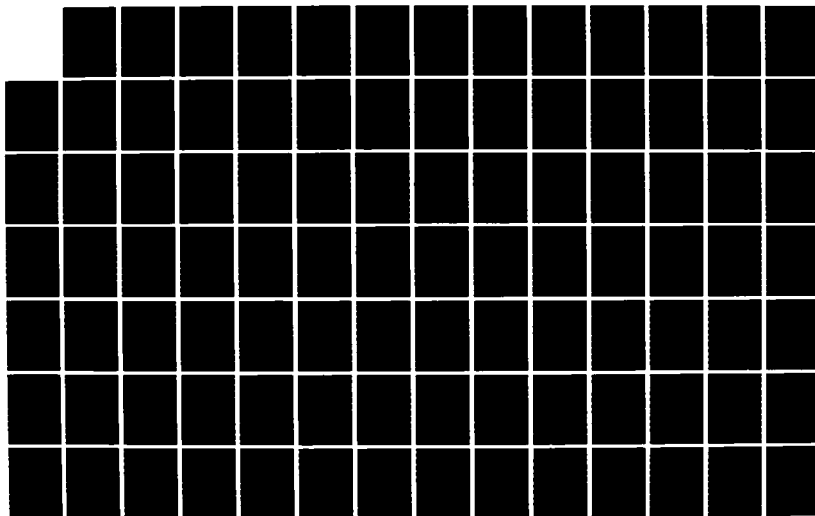
AGGREGATION OF NETWORK FLOW PROBLEMS(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH V E FRANCIS 1985
AFIT/CI/NR-86-800

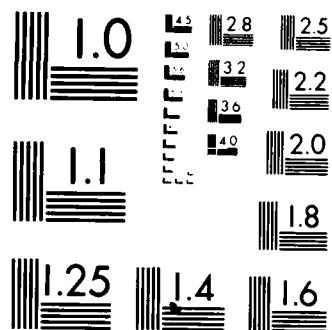
3/4

UNCLASSIFIED

F/G 12/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

used to reduce the effort required to solve the next aggregate problem in the sequence. This research has attempted to do that by constructing an advanced-start basic solution for each succeeding aggregate problem from the current one.

It should be noted that, although the complete algorithm and the AGNET implementation are specific ones, the concepts of chapters two through five are generally applicable. The framework, aggregation procedure, partition refinement process and the DM^G disaggregation map are valid under any structural respecification that results in a network aggregate problem.

The complete algorithm can be easily modified to accomodate other schemes such as the popular "fixed weight aggregation." Also, any of the various aposteriori error bounds can be incorporated.

While we have considered only a sequence of successively restricted aggregate problems, future work might investigate sequences of successively relaxed aggregate problem. Another possibility is to include both relaxations and restrictions to provide both upper and lower bounds for the optimal objective function value of the original problem.

It might also prove interesting to empirically investigate different partition refinement rules and their impact on the computational effort required by DM^G .

The AGNET code has been written as a testing and verification tool. The current version is three times the size of GNET and requires much more memory. Subsequent versions should be written with a greater emphasis on efficiency. It may prove advantageous to design a completely new aggregation-disaggregation type code rather than to build it around an existing network optimization code such as GNET. Other enhancements could be the inclusion of a variety of user selected respecification maps and error bounds.

Another area for future research is the specialization of this research to other network models. Specifically, it should be an easy task to tailor these results to such models as the transshipment model, the transportation model and the assignment model. Conversely, it may be possible to elaborate upon this work to encompass generalized networks.

APPENDIX A: Geoffrion's Modeling Framework And Aggregation Theory

This appendix summarizes the important terminology, notation and concepts of Geoffrion's modeling framework and aggregation theory as described in [19]. It should be noted at the outset that this appendix is based upon an early draft of Geoffrion's work in aggregation and structured modeling. Geoffrion has completed several revisions of [19] during the course of this research. Although Geoffrion's most recent work includes some new concepts and has modified some of the definitions and results contained herein, these additions and changes do not appreciably affect the results of this dissertation. It was necessary to use Geoffrion's earlier work because the two research efforts were being conducted concurrently.

Basic definitions, notation and terminology will be given first followed by an example, the classical transportation model. This example will be continued throughout this appendix to illustrate and clarify the ideas and concepts of the modeling framework and aggregation theory.

The Modeling Framework: Basic Definitions and Properties

Important characteristics of the modeling framework include the following:

1. It encompasses many of the models commonly occurring in management science and related fields.
2. It is pre-mathematical in nature in that it does not require mathematical notions other than set and graph theoretic concepts necessary for model representation.
3. Its primary focus, like other modeling techniques employing graphical representation, is on definitional dependence among model elements.
4. It explicitly recognizes natural groupings of related model elements.

Basic definitions and results are given below without comment for the sake of brevity. They will however, be illustrated in an example to follow.

We begin by defining the five basic types of model elements.

Definition 1: A primitive entity element is any distinctly identifiable thing or conception whose definition does not depend on any other element.

Definition 2: A compound entity is any distinctly identifiable thing or conception whose definition depends on a tuple of primitive elements and/or other compound entity elements.

Definition 3: An attribute element represents the association of a unique value in some range to a certain tuple of entities.

Definition 4: A function element represents a rule which associates a unique value in some range to a certain tuple of elements- more precisely, in the case of nonentity elements, to the value of these elements provided these fall within a prescribed domain.

Definition 5: A condition element is a function element with the two-valued range (TRUE, FALSE).

The following definitions and propositions describe how model elements can be related under the framework, characteristics of well-defined canonical models and how such models can be conveniently represented.

Definition 6: Every element other than a primitive entity has a calling sequence consisting of the tuple of elements to which its formal definition refers. For a compound entity it is the tuple of entities on which it is defined; for an attribute element it is the tuple of entities to which it associates a value; and for a function or condition element the calling sequence is the tuple of elements to

which it associates a value. We say that element A "calls" element B if B appears in A's calling sequence.

Definition 7: A collection of elements is acyclic if there is no sequence $(e_1, e_2, \dots, e_n = e_1)$ such that e_1 calls e_2 , e_2 calls e_3 , $\dots$, e_{n-1} calls e_n , where $n \geq 2$.

Definition 8: An elemental model is an acyclic collection of elements.

Definition 9: An elemental model is completely specified if all of its elements are formally defined in detail except, perhaps, that the function and condition element rules may not have been applied to determine the values of these elements. The process of determining these values is called evaluation. An A-partially specified elemental model is completely specified except for the values of certain attribute elements called variable attribute elements.

Definition 10: A well-defined model is one whose specification, if not complete, can be completed so that evaluation is a logically well-posed task.

Definition 11: A well-defined elemental model is feasible if its specification can be completed (if not already complete) so that evaluation is a well-posed task and

results in all condition elements being TRUE.

Definition 12: A generic structure of an elemental model is a collection of partitions, one for each of the five types of elements. The resulting mutually exclusive and exhaustive element sets are called genera (pl. of genus).

Definition 13: A generic structure satisfies the generic uniqueness property if no two elements of any genus have identical calling sequences.

Definition 14: A generic structure satisfies the generic similarity property if, given that some element of genus A calls some element of genus B, then every element of genus A calls some element of genus B.

Definition 15: A canonical model consists of a finite, acyclic, well-defined collection of elements together with a generic structure satisfying the generic uniqueness and similarity properties.

Definition 16: The element graph of a collection of elements is a directed graph with a node for every element, and an arc for every calling sequence reference with the head at the calling element and the tail at the called element.

Definition 17: The genus graph of an elemental model with given generic structure is a directed graph with a node for every genus and an arc from genus B to genus A if some element of genus A calls some element of genus B.

The following propositions describe properties possessed by canonical models and imply a convenient means of drawing or representing the element and genus graphs of such canonical models.

Proposition 1: If a finite, acyclic collection of elements has a generic structure satisfying the similarity property, then the corresponding genus graph is acyclic.

A well known and important property of acyclic graphs is that its nodes can be classified into ranks such that nodes of rank r ($r \geq 2$) have incoming arcs only from nodes of lower rank including at least one node of rank $r-1$.

Proposition 2: If an finite, acyclic collection of elements has a generic structure satisfying the similarity property, then no genus contains elements of different ranks.

Proposition 3: The nodes of an acyclic genus graph can be classified into ranks in such a manner that (i) rank one

genera call no other genera, and (ii) for $r \geq 2$ every genus of rank r calls at least one genus of rank $r-1$, possibly other genera of rank less than $r - 1$, but no genus of rank greater than $r-1$.

Modeling Framework: Notational Conventions

The notational conventions used in the modeling framework, intended:

1. to mnemonically index the elements of each genus for convenient reference
2. to facilitate checking acyclicity, generic uniqueness and generic similarity
3. to provide an abbreviated notation for working purposes as well as a full notation that captures all of the information needed to write down a complete model schema
4. to prepare the way for more conventional notation

is described below.

N1. Each genus has a distinct and mnemonic genus name written in upper case.

N2. Each genus has an indexed full generic name denoting a typical element of the genus, and an associated genus index

set called [GENUSNAME], where GENUSNAME is the name of the genus. For a primitive entity genus, the generic name is of the form

$$\text{GENUSNAME}_u$$

where the free index u is the argument of the index set.

For all other genera, the generic name is of the form

$$\text{GENUSNAME (generic calling sequence)}$$

where the generic calling sequence (N3) contains the free indices comprising arguments of the index set. Each genus is completely indexed, that is, its cardinality is identical to the cardinality of its index set. This may require introducing additional free indices beyond those already associated with the called genera. Indices are written in lower case.

N3. A generic calling sequence is positional and consists of the typical tuple of called elements designated by their short form generic names (N4). The following abbreviations may be employed:

- (i) use the genus name alone when all elements of a genus appear in the calling sequence
- (ii) use a dot in place of an index when all index values appear in the calling sequence
- (iii) use a function-style notation for indices that depend completely on other indices, with an asterisk denoting multi-valued functions (e.g.

$i(j)$ signifies that the value of index i depends uniquely upon the value of index j , and $i^*(j)$ signifies that several values of index i are associated with each value of index j).

N4. The short generic name corresponding to a full generic name is obtained by making all free indices subscripts of GENUSNAME, and dropping the generic calling sequence. For a primitive entity genus, which has no calling sequence, the short generic name is just the full generic name.

N5. The full (resp. short) elemental name of an element is the full (resp. short) generic name associated with its genus, with all free indices specified in detail. A full elemental name may still be somewhat implicit owing to abbreviations (see (i), (ii) and (iii) of N3). The explicit elemental name of an element avoids all such abbreviations and explicitly lists all called elements. Some full elemental names are also explicit elemental names.

N6. A genus with just one element is a degenerate case requiring no free index and no index set. The distinction between the full (resp. short) generic name and the full (resp. short) elemental name vanishes, and one speaks simply of "full name," "short name," and "explicit name."

The following abbreviations are also used:

PE Primitive entity
CE Compund entity
A Attribute
F Function
C Condition
VA Variable attribute

Aggregation Procedure

Armed with the above definitions, terminology and notation it is now possible to rigorously define an aggregation procedure applicable to any model falling within the purview of the modeling framework.

Given a canonical model (M) and an aggregation set S of elements within a primitive entity genus, the structural aggregation procedure of (M) with respect to S is generated by the following procedure. The result is a new canonical model wherein S is replaced by a single aggregate element.

Structural Aggregation Procedure

1. Classify all genera by rank. Let R be the number of ranks.
2. Set $k = 2$.
 - a. Merge all elements of S into a single aggregate entity element.
 - b. Identify all calling sequence references to elements

of S, and redirect them to the new element.

"Mark" the nonentity elements whose calls are redirected.

3. a. For all genus in rank k, merge all elements having identical calling sequences into a single aggregate element (to restore generic uniqueness).
- b. Identify all calling sequence references to any of the elements merged in step 3a, and redirect each one to the new aggregate element. "Mark" the non-entity elements whose calls are redirected.
4. If $k = R$, stop. Otherwise, increase k by 1 and return to step 3.

Note that the above procedure does not indicate how the values or ranges of the new aggregate attribute elements are to be specified, how the rules for the new aggregate function and condition elements are to be specified nor how modifications are to be made in the specification of the "marked" elements. The following discussion describes this respecification process.

Respecification pertains to a specific model instance and refers to the process of taking the structural aggregation to the same level of specification as the parent model (M) in such a way that the resulting model is well-defined. Such a rule for carrying out this respecification process is called a respecification map

(abbreviated RM). It maps pre-aggregation elements into post-aggregation elements.

It is convenient to decompose a respecification map into components: a structural respecification map (SRM), and a variable respecification map (VRM). The former pertains to everything except the values of variable attribute elements of the structural aggregation while the latter pertains only to those values. The extension of the VRM whose domain is all of the variable attribute elements (not just the new aggregate and marked variable attribute elements) is referred to as the aggregation map (AM) associated with the VRM.

The following proposition describes the result of the structural aggregation procedure.

Proposition 4: When applied to a canonical model, the structural aggregation procedure is a well posed task and yields a canonical model that is unique up to respecification and that has the same genus graph as before.

Uses Of Aggregate Models, Aggregation Design And Measures Of The Quality Of An Aggregation

Four typical uses for aggregate models are listed below.

1. Retrieval: answer queries about the model itself not requiring evaluation.

2. Evaluation: perform evaluation.
3. Satisfaction: find values for the variable attribute elements such that evaluation results in all TRUE condition elements; the resulting variable attribute element values are collectively called a feasible solution.
4. Optimization: find a feasible solution such that a distinguished real-valued function element achieves its maximum or minimum value; the resulting variable attribute element values are collectively called an optimal solution.

We shall be concerned in this research solely with using aggregate model $(\bar{M})$ as a surrogate for original model (M) in the context of optimization. This can be done in any of several ways. Before describing these alternatives, it is first necessary to define the concept of a disaggregation map (DM): any rule which, given values for all variable attribute elements of $(\bar{M})$, assigns values to all variable attribute elements of (M) in such a way as not to destroy the well-definedness of (M) . The ways in which $(\bar{M})$ can be used as an optimization surrogate for (M) , along with corresponding measures of the quality of $(\bar{M})$, follow.

Optimization surrogate: Given (M) possessing an optimal solution and given a feasible DM, find an optimal solution of $(\bar{M})$ and apply DM to this solution in order to obtain a solution of (M) . Notice that an AM is not needed. Call the amount of suboptimality in (M) resulting from using the

disaggregated solution of $(\bar{M})$ the optimization error.

Aggregate optimization surrogate: Given (M) and $(\bar{M})$ both possessing optimal solutions, find an optimal solution of $(\bar{M})$ and use it as a surrogate for the image of an optimal solution of (M) under an adopted aggregation map AM . Notice that a DM is not required. Call the amount of suboptimality in $(\bar{M})$ resulting from using the aggregated optimal solution of (M) the aggregate optimization error.

Optimal value surrogate: The optimal value of $(\bar{M})$ is used as a surrogate for the optimal value of (M) . Neither an AM nor DM is required. Call the discrepancy between the optimal values of (M) and $(\bar{M})$ the optimal value error.

Transportation Problem Example

We now illustrate these concepts using the simple transportation problem shown in Figure A-1. The model schema for this particular model is described below in Figure A-2 using the notational conventions of the modeling framework. Figures A-3 and A-4 show the element and genus graphs, respectively.

When the structural aggregation procedure is applied to the transportation model shown in Figure A-1 for $S = \{CUST_1, CUST_2\}$, the result is shown in Figure A-5. We shall refer to the original model of Figure 1 as (OTP) and to the

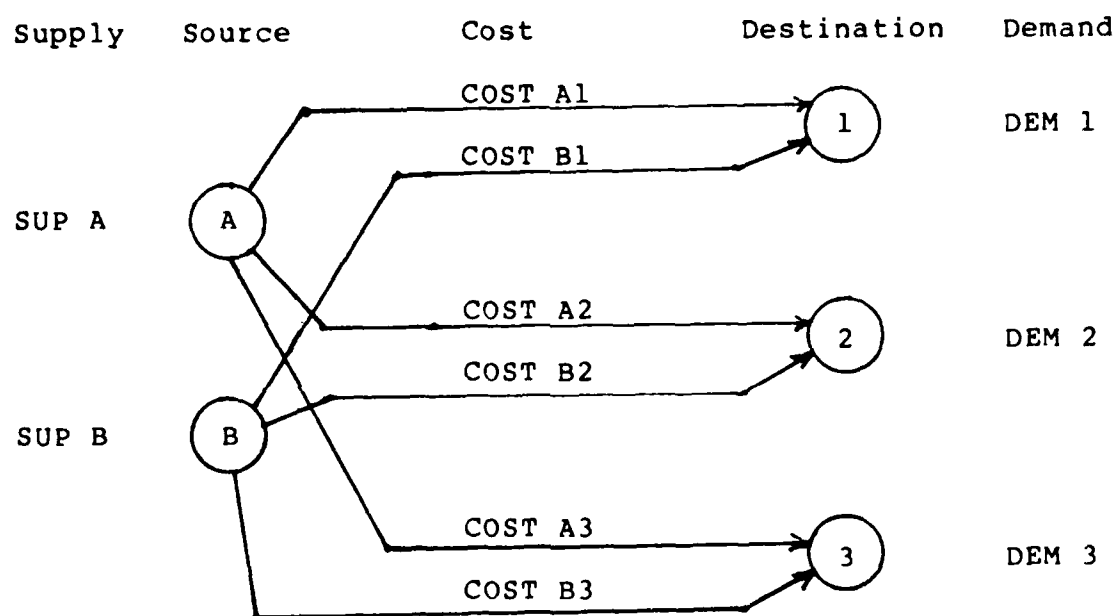


Figure A-1: Transportation Model Example

RANK	TYPE	GENERIC NAME (full)	INDEX SET	EXPLANATION
1	PE	$PLANT_i$	$i=A,B$	Plants
1	PE	$CUST_j$	$j=1,2,3$	Customers
2	CE	$LINK(PLANT_i, PLANT_j)$	$[PLANT] \times [CUST]$	Transp. links
2	A	$SUP(PLANT_i)$	$[PLANT]$	Supply of plant
2	A	$DEM(CUST_j)$	$[CUST]$	Demand of cust.
3	A	$COST(LINK_{ij})$	$[LINK]$	Transport. cost
3	VA	$FLOW(LINK_{ij})$	$[LINK]$	Flows on links
4	F	$\$(COST, FLOW)$		Obj. func. value
4	C	$SCON(FLOW_{i.}, SUP_i)$	$[PLANT]$	$SCON \text{ "True"} \Leftrightarrow$ supply constraint satisfied
4	C	$DCON(FLOW_{.j}, DEM_j)$	$[CUST]$	$DCON \text{ "True"} \Leftrightarrow$ demand constraint satisfied

Figure A-2: Model Schema For Transportation Model

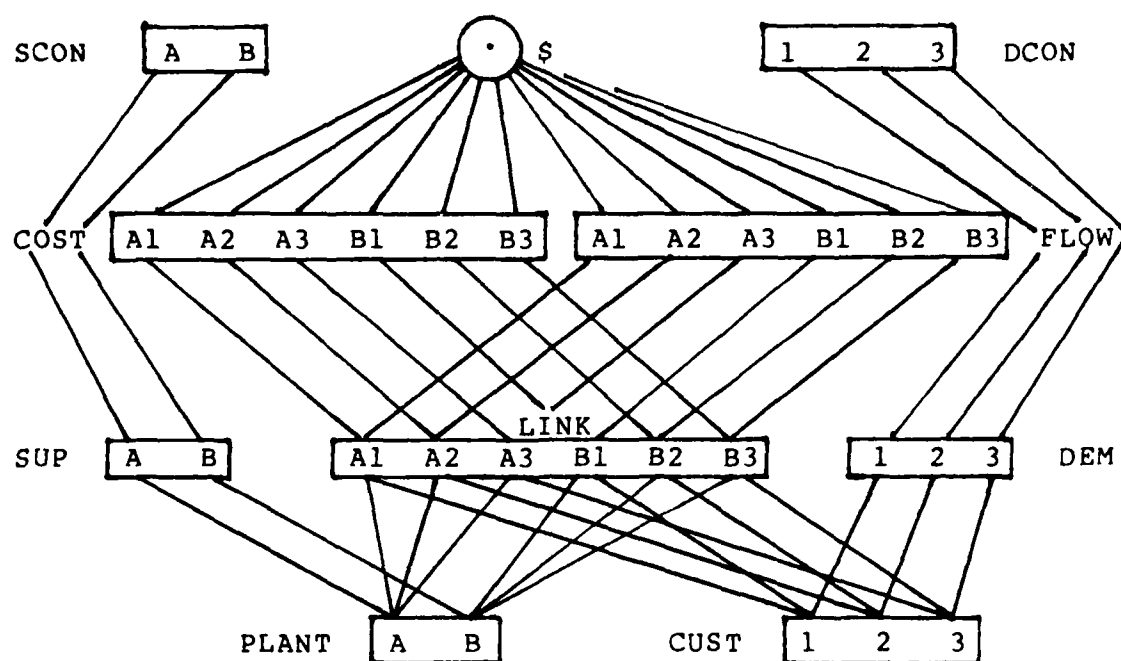


Figure A-3: Element Graph For Transportation Model

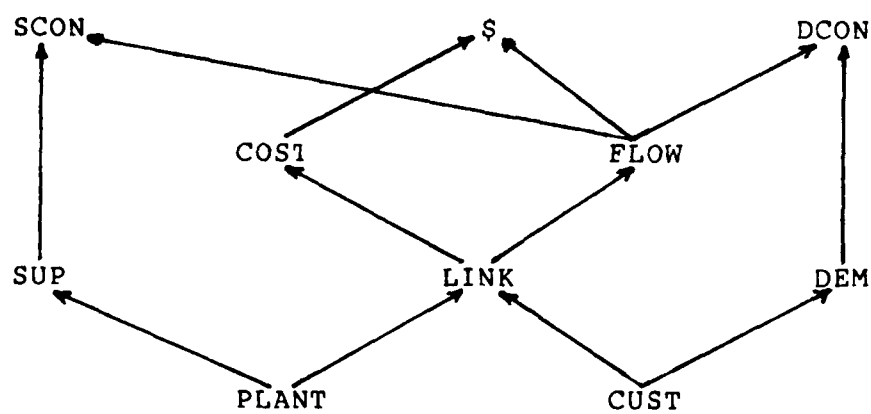


Figure A-4: Genus Graph For Transportation Model

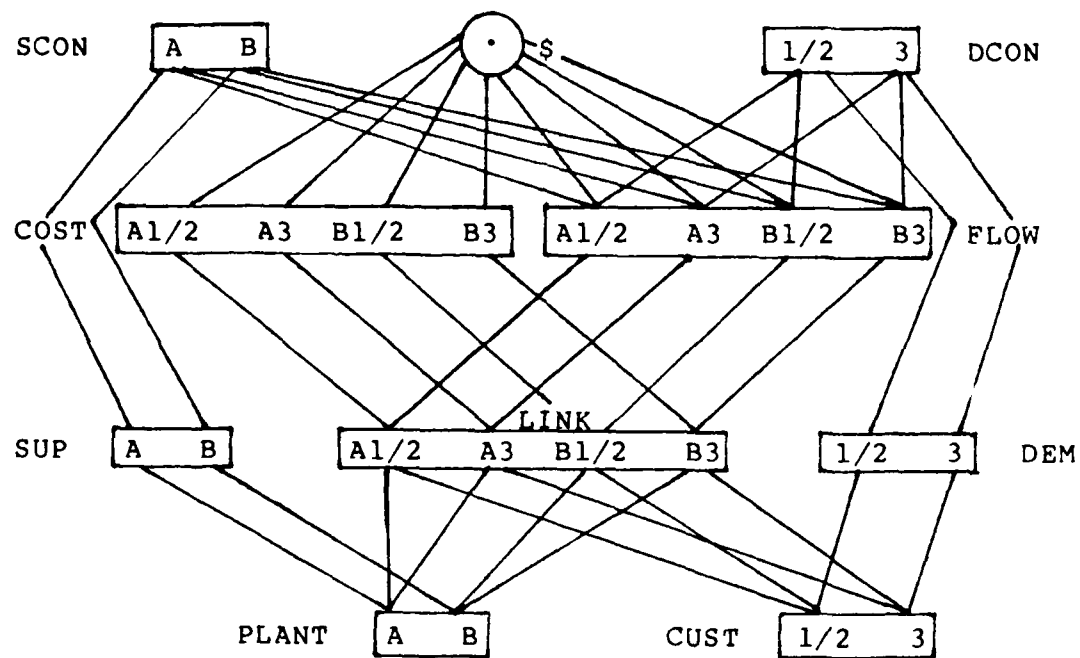


Figure A-5: Element Graph For Aggregate Transportation Model

resultant aggregate model as (ATP). The aggregate model (ATP) is also shown in the more traditional network diagram of Figure A-6.

To complete the specification of the (ATP), we must respecify the aggregate elements:

$$DEM_s$$

$$FLOW_s$$

$$COST_{i,s}$$

$$\$_s$$

$$SCON_s$$

$$DCON_s$$

where s is the index for the new aggregate element. One possible respecification map is described below. It, henceforth referred to as the "fixed weight aggregation" of the original network model, is the predominant one encountered in the literature of network aggregation.

$$DEM_s = \sum_{j \in S} DEM_j$$

$$FLOW_{i,s} = \sum_{j \in S} FLOW_{i,j} \quad \text{for } i \in [PLANT]$$

$$COST_{i,s} = \frac{\sum_{j \in S} DEM_j COST_{i,j}}{DEM_s} \quad \text{for } i \in [PLANT]$$

and the formulae for $\$_s$, $SCON_s$, for all s , and $DCON_s$ for $s=s$ are taken to be the same as in Figure A-2.

Note that the formula given for $FLOW_{i,s}$ is the variable respecification map and everything else constitutes the

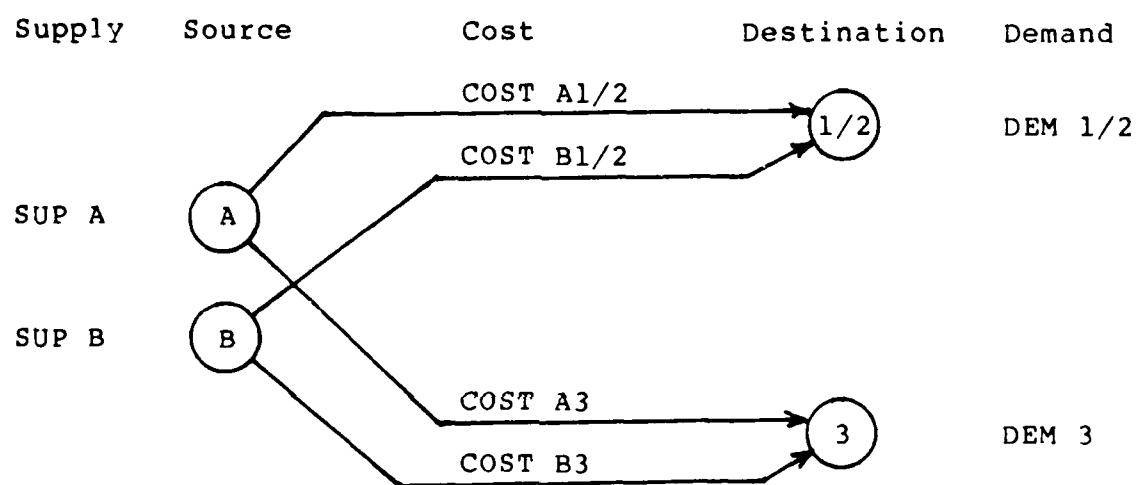


Figure A-6: Aggregate Transportation Model

structural respecification map. For completeness, the "fixed weight disaggregation map" is defined below:

$$FLOW_{i,j} = \frac{DEM_j}{DEM_s} FLOW_{i,s}, \text{ for } j \in S$$

It is easy to generalize the above aggregation procedure for a general partition of a single or multiple target primitive entity genera.

APPENDIX B: Lee's Aggregation Framework And
Network Algorithm

Of particular interest are Lee's aggregation framework for linear programming problems, developed from an algorithmic perspective, and his general algorithm for the capacitated transshipment problem as explained in [39]. We shall review these here in detail since they bear directly upon this research.

BI: Lee's Aggregation Framework

Define the linear programming problem (P) as:

$$\begin{array}{ll}\text{Min} & wy \\ \text{S.T.} & Ay = r \\ & y \geq 0\end{array}$$

where: A is an $(M \times N)$ matrix of constants
 y is an $(N \times 1)$ vector of variables
 r is an $(M \times 1)$ vector of constants
 w is an $(1 \times N)$ vector of constants

and letting:

$$\begin{array}{l}I = \{1, 2, \dots, M\} \\ \quad = \text{constraint index set} \\ J = \{1, 2, \dots, N\} \\ \quad = \text{variable index set}\end{array}$$

$$\begin{bmatrix} A & r \\ w & 0 \end{bmatrix} = \text{an } (M+1) \times (N+1) \text{ matrix called the} \\ \text{the } \underline{\text{fundamental array}}$$

Lee lists the following definitions.

Definition 1: A partition d of I is the collection of $K = \{1, 2, \dots, k\}$ subsets of I , $d = \{d(1), d(2), \dots, d(k)\}$ such that:

- (i) $\bigcup_{i \in K} d(i) = I$
- (ii) $d(i) \cap d(j) = \emptyset$, all $i, j \in K$, $i \neq j$
- (iii) $d(i) \neq \emptyset$, all $i \in K$

Definition 2: A refinement is a binary relation on the set D_I of all partitions of I , expressed $d_1 \prec d_2$, read " d_1 is a refinement of d_2 ," if for any $d_1(j)$ there exists $d_2(i)$ such that $d_1(j) \subset d_2(i)$.

Lee goes on to show that the set D_I of partitions of I is a partially ordered set with respect to refinement and that the set D_I is a lattice with universal bounds $0 = \{1, 2, \dots, M\}$ and I . Similar results are derived for the set D_J of partitions of J .

These definitions and results are necessary preliminaries for Lee's development of surrogate (aggregate) problems for (P) in terms of the partition pairs $(d_r, d_c) \in D_I \times D_J$. He gives the corollary that the set $L = (D_I, D_J)$ of partition pairs of (I, J) is a lattice with universal bounds $(0, 0)$ and (I, J) .

With these definitions, Lee proceeds to define "replacement rules" that, along with a partition pair $(d_r, d_c) \in D_I \times D_J$, facilitate construction of a surrogate problem of (P). For a given (d_r, d_c) , the fundamental array is partitioned into cells as shown below:

	$d_c(1)$	$d_c(2)$	$\dots$	$d_c(n)$	
$d_r(1)$	A_{11}	A_{12}	$\dots$	A_{1n}	r_1
$d_r(2)$	A_{21}	A_{22}	$\dots$	A_{2n}	r_2
$\vdots$					
$\vdots$					
$\vdots$					
$d_r(m)$	A_{m1}	A_{m2}	$\dots$	A_{mn}	r_m
	w_1	w_2	$\dots$	w_n	$\emptyset$

Let:

$H = \{h: 1 \leq h \leq m\}$ = the index set of cardinality d_r

$K = \{k: 1 \leq k \leq n\}$ = the index set of cardinality d_c

Then the cells or submatrices of the partitioned fundamental array are defined as:

$\bar{A}_{hk} = (a_{ij})$, such that $i \in d_h, j \in d_k$,
for all $h \in H, k \in K$

$\bar{r}_h = (r_i)$, such that $i \in d_h$, for all $h \in H$

$\bar{w}_k = (w_j)$, such that $j \in d_k$, for all $k \in K$

Definition 3: A replacement rule R is a rule which replaces each cell of a partitioned fundamental array by a unique element in the following correspondence:

$$\bar{A}_{hk} \leftarrow s_{hk}, \text{ all } h \in H, k \in K$$

$$\bar{r}_h \leftarrow t_h, \text{ all } h \in H$$

$$\bar{w}_k \leftarrow u_k, \text{ all } k \in K$$

Let $S=(s_{hk})$, $h \in H$, $k \in K$, be an $m \times n$ aggregate matrix, $t=(t_h)$, $h \in H$, be an $m \times 1$ aggregate vector and $u=(u_k)$, $k \in K$, be an $1 \times n$ aggregate vector.

Definition 4: A surrogate problem $R(d_r, d_c)$ of (P) having fundamental array:

$$\begin{bmatrix} S & t \\ u & \emptyset \end{bmatrix}$$

is uniquely defined by the replacement rule R and the partition pair (d_r, d_c) .

Definition 5: A row surrogate problem, denoted $R(d_r, 0_c)$, is a surrogate problem in which the column partition is the singleton set of the column index, i.e. $d_c = 0_c$.

Definition 6: A column surrogate problem, denoted $R(0_r, d_c)$, is a surrogate problem in which $d_r = 0_r$.

Definition 7: A uniform replacement rule is a replacement rule such that within sets S , t , u all elements are replaced with an identical rule.

Definition 8: Let $R_{\min}$ be a uniform replacement rule

defined as

$$R_{\min} = \begin{bmatrix} \text{minimum} & | & \text{maximum} \\ \hline & & \\ \text{minimum} & | & \emptyset \end{bmatrix}$$

such that $s_{hk} = \min_{\substack{i \in d_r(h) \\ j \in d_c(k)}} \{a_{ij}\}, \text{ all } h \in H, k \in K$

$$u_k = \min_{j \in d_c(k)} \{w_j\}, \text{ all } k \in K$$

$$t_h = \max_{i \in d_r(h)} \{r_i\}, \text{ all } h \in H$$

Definition 9: Let $R_{\max}$ be a uniform replacement rule defined as

$$R_{\max} = \begin{bmatrix} \text{maximum} & | & \text{minimum} \\ \hline & & \\ \text{maximum} & | & \emptyset \end{bmatrix}$$

such that $s_{hk} = \max_{\substack{i \in d_r(h) \\ j \in d_c(k)}} \{a_{ij}\}, \text{ all } h \in H, k \in K$

$$u_k = \max_{j \in d_c(k)} \{w_j\}, \text{ all } k \in K$$

$$t_h = \min_{i \in d_r(h)} \{r_i\}, \text{ all } h \in H$$

Definition 10: Let R_{cxc} be a uniform replacement rule defined as

$$R_{cxc} = \begin{bmatrix} \text{convex combination} & | & \text{convex combination} \\ \hline & & \\ \text{convex combination} & | & \emptyset \end{bmatrix}$$

such that $s_{hk} = \sum_{i \in d_r(h)} \alpha_i \sum_{j \in d_c(k)} \beta_j a_{ij}, \text{ all } h \in H, k \in K$

$$u_k = \sum_{j \in d_c(k)} \beta_j w_j, \text{ all } k \in K$$

$$t_h = \sum_{i \in d_r(h)} \alpha_i r_i, \text{ all } h \in H$$

where $\alpha_i \geq 0, i \in d_r(h), \sum_{i \in d_r(h)} \alpha_i = 1, \text{ all } h \in H$

and $\beta_j \geq 0, j \in d_c(k), \sum_{j \in d_c(k)} \beta_j = 1, \text{ all } k \in K$

Definition 11: A replacement rule R is a relaxation if

$$(i) \quad F(P) \neq \{\emptyset\} \Rightarrow F(R(d_r, d_c)) \neq \{\emptyset\}$$

$$(ii) \quad z^*(P) \geq z^*(R(d_r, d_c))$$

where $F(\cdot)$ is the feasible region of $(\cdot)$ and $z(\cdot)$ is the optimal objective function value of $(\cdot)$. It is a strict relaxation if (ii) holds as an inequality.

Definition 12: A replacement rule is a restriction if

$$(i) \quad F(R(d_r, d_c)) \neq \{\emptyset\} \Rightarrow F(P) \neq \{\emptyset\}$$

$$(ii) \quad z^*(P) \leq z^*(R(d_r, d_c))$$

It is a strict restriction if (ii) holds as an inequality.

Theorem 5: (i) Given (P) and a relaxed $R(d_r, d_c)$, if $R(d_r, d_c)$ is infeasible, so is (P) . If (P) is unbounded, so is $R(d_r, d_c)$. (ii) Given (P) and a restricted $R(d_r, d_c)$, if $R(d_r, d_c)$ is feasible or its optimal value is unbounded, the same is true for (P) . If (P) is infeasible, so is $R(d_r, d_c)$.

It is important to note the correspondence between these definitions and results and those of Appendix A. In the parlance of Appendix A, Lee has implicitly defined the primitive entity genera ROWS and COLUMNS, corresponding to the constraints and variables of (P) , with index sets I and

J respectively. Portrayed below in Figure B-1 is the model schema adopted by Lee.

Lee constructs the aggregate model, i.e. surrogate problem, by:

1. iteratively performing the structural aggregation of (P) with target genera ROWS and COLUMNS and with the individual subsets of the partitions (d_r, d_c) of the corresponding index sets.
2. defining the structural respecification maps as the "replacement rules" given in Definitions 8 through 10 above.

Note that Lee's replacement rules engender three respecification maps: one for the constraint coefficients, one for objective function coefficients, and one for constraint right-hand-sides. In essence then, Lee has defined, using a different terminology, the aggregation of the specific model instance (P). Additionally, Theorems 5 and 6 provide insight into the behavior of the aggregate model $R(d_r, d_c)$, constructed under certain respecification maps, when used as optimal value surrogates.

Next, Lee considers sequences of surrogate problems of (P) generated by successive refinement of the partition pair (d_r, d_c) and/or successively applying different replacement rules. He defines a general collection of surrogate problems as

MODEL SCHEMA

<u>RANK</u>	<u>TYPE</u>	<u>GENERIC NAME</u>	<u>INDEX SET</u>
1	PE	ROW_i	$i = 1, \dots, m$
1	PE	COL_j	$j = 1, \dots, n$
2	A	$A(ROW_i, COL_j)$	$[ROW] \times [COL]$
2	A	$B(ROW_i)$	$[ROW]$
2	A	$C(COL_j)$	$[COL]$
2	A	$X(COL_j)$	$[COL]$
3	F	$F(A_i, X)$	$[ROW]$
3	F	$S(C, X)$	singleton
4	C	$LEQ(F, B_i)$	$[ROW]$

GENUS GRAPH

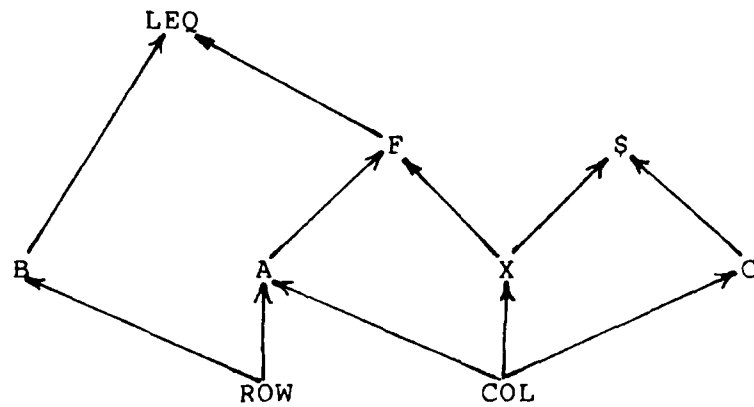


Figure B-1: Model Schema For (P) Adopted By Lee

$$P = \{R^i(l_j)\}, R^i \in R, l_j \in (d_I, d_J) = L$$

where $R = \{R^1, R^2, \dots\}$ is any class of operators generated by applying operators $R^1, R^2, \dots$ to the partition pairs $l_1, l_2, \dots$. Of particular interest are the two pure types of collections of surrogate problems:

1. $P_1 = \{R(l_j)\}$ where R is a single operator, $l_j \in L$
2. $P_2 = \{R^i(l)\}$ where l is a fixed partition, $R^i \in R$

Such collection of surrogate problems are not specifically discussed in Geoffrion's aggregation theory but can be easily accommodated.

For a particular mapping

$$\Psi: Y(o) \rightarrow E^N$$

where $Y(o) =$ the set of optimal solutions of $R(d_r, d_c)$

$E^N =$ N dimensional Euclidean space, the domain of (P) a gauge of the surrogate problems with respect to Ψ and (P) is defined as a mapping from $R(d_r, d_c)$ to the nonnegative real numbers:

$$G_\Psi : R(d_r, d_c) \rightarrow R^+$$

and such a gauge can be used to judge how "good" one surrogate problem is compared to another. Lee provides a viable "primal gauge" for surrogate problems of (P) as:

$$G(R(d_r, d_c)) = c(V) + (a_t y' - r_t) \theta^{-1}$$

where: $y' = \Psi(Y')$ for $Y' \in Y(o)$, $y' \in E^N$

$V =$ index set of constraints of (P) that are violated at y'

$c(V) =$ the cardinality of V

t = index of a "target constraint" in the set V
 θ = positive real number sufficiently large that

$$|a_t y' - r_t| \theta^{-1} = 1$$

Possible strategies for selecting the target constraint t include "first violated" and "most violated" rules. Also, Lee proposed that if $c(V) = \emptyset$, t is selected to be the objective function. It is apparent that any of the various a priori or a posteriori error bounds proposed in the literature can be considered equally viable gauges in Lee's aggregation framework.

The idea of a gauge leads to the concepts of equivalence classes of surrogate problems, class decomposition of a collection of surrogate problems, metric spaces associated with equivalence classes, convergent sequences in a metric space and descending chains through the lattice representation of partition refinement as summarized below.

Definition 13: Let $P = \{p_1, p_2, \dots\}$ be a collection of surrogate problems. Two surrogate problems $p_i, p_j \in P$ are equivalent ($\sim$) with respect to G if $|G(p_i) - G(p_j)| = \emptyset$

Definition 14: The equivalence class of $p_i \in P$, denoted $E(p_i)$, is the set of elements $p_j \in P$ such that $p_j \sim p_i$.

Definition 15: A metric $\rho(E(p_i), E(p_j))$ on $E(P)$ is defined

as:

$$\rho(E(p_i), E(p_j)) = |G(p_u) - G(p_t)|$$

for any $p_u \in E(p_i)$ and $p_t \in E(p_j)$.

Theorem 7: $E(P)$ is a metric space with respect to ρ .

Definition 16: Given a sequence $\{E(p_1), E(p_2), \dots\}$ of points in a metric space, a point $E(p)$ is said to be a limit of the sequence provided that, given $\xi > 0$, there exists an N such that $\rho(E(p), E(p_n)) < \xi$ for all $n > N$.

Definition 17: A subset C of (D_I, D_J) is simply ordered relative to the binary relation " $\prec$ " of refinement if, for $c_1, c_2, c_3 \in C$ the following axioms hold:

- (i) if $c_1 \neq c_2$, then $c_1 \prec c_2$ or $c_2 \prec c_1$
- (ii) if $c_1 \prec c_2$, then c_1 and c_2 are distinct
- (iii) if $c_1 \prec c_2$ and $c_2 \prec c_3$, then $c_1 \prec c_3$

Definition 18: A sequence of surrogate problems $\{R^i(l_j)\}$ is said to be a consistent sequence if the replacement rules R^i are all exclusively relaxations or restrictions.

To complete his aggregation framework, Lee considered the efficiency of descending chain processes and associated stopping rules. His ideas are summarized in the following definitions and theorems.

Definition 19: An efficient sequence of surrogate problems consists of problems belonging to distinct equivalence classes.

Definition 20: A replacement rule R is a strong relaxation if there exists

$$\phi_R: E^N \rightarrow E^N$$

such that: (i) $y \in F(P) \Rightarrow Y = \phi_R(Y) \in F(R(d_R, d_C))$
(ii) $z^*(P) \geq z^*(R(d_R, d_C))$.

Definition 21: A replacement rule is a strong restriction if there exists

$$\psi_R: E^N \rightarrow E^N$$

such that: (i) $Y \in F(R(d_R, d_C)) \Rightarrow y = \psi_R(Y) \in F(P)$
(ii) $z^*(P) \leq z^*(R(d_R, d_C))$.

Theorem 8: Let Y be an optimal solution of a surrogate problem $R(d_R, d_C)$ and $y = \psi(Y)$ where R is a strong relaxation replacement rule. If $y = \psi(Y) \in F(P)$ and $w \cdot y = z^*(R(d_R, d_C))$ then $E(R(d_R, d_C)) = E(P)$ and y is an optimal solution of the original problem.

Theorem 9: Given an ascending chain $\{a_i\}$ with $a_0 = (0_R, 0_C)$ and $a_i \leq a_{i+1}$, and R a strong relaxation operator. Let $R^A(a_i)$ be the surrogate problem generated by applying R

iteratively from a_0 to a_i . For any i there exists $\phi^i: E^{n_i}$
 -- $E^{n_{i+1}}$ such that

- (i) $Y^i \in F(R^A(a_i)) \Rightarrow Y^{i+1} = \phi^i(Y^i) \in F(R^A(a_{i+1}))$
- (ii) $z^*(R^A(a_i)) = z^*(R^A(a_{i+1}))$.

Finally, Lee illustrates his aggregation framework by developing the ground work for an algorithm for network optimization. He considers the minimal cost flow problem given below in circulation form:

$$\text{Min} \quad \sum_{(x,y) \in A} c(x,y) f(x,y)$$

$$\text{S.T.} \quad f(x,N) - f(N,x) = 0, \text{ all } x \in N$$

$$l(x,y) \leq f(x,y) \leq e(x,y), \text{ all } (x,y) \in A$$

where: f = flows

c = cost of corresponding flows

$l, (e)$ = lower (upper) bounds on flows

(x,y) = arc from node x to node y

A = set of all arcs

N = set of all nodes

Also let $C = \{(i,j): (i,j) \in N \times N, a_{ij} \in A\}$. Henceforth in this appendix, the above problem shall be referred to as the original problem (P). Thus, the row index set is $O_r = \{N, C\}$ where row $i \in N$ is the flow conservation constraint for node i and row $(j,k) \in C$ is the capacity constraint for the arc (j,k) . The column index set is $O_c = \{C\}$ where $(i,j) \in C$ corresponds to the arc $a_{ij} \in A$. The original network model

thus has partition pair (O_r, O_c) .

Lee shows that, given a partition d_r of the node set N , a corresponding partition d_c of the arc set A can be generated. Letting the index set of the flow conservation equations be T , for any partition

$$T = \bigcup_{q=1}^k t(q)$$

$$t(q) \cap t(p) = \emptyset, \text{ for all } p \neq q$$

$$t(q) \neq \emptyset, \text{ for all } q$$

define

$$c(q,p) = \{(u,v) : (u,v) \in t(q) \times t(p) \cap C\}$$

The following result holds directly:

Theorem 11: The collection $\bar{C} = \{c(q,p) : c(q,p) \neq \emptyset\}$ is a partition of C .

The surrogate problem $R_{\text{sum}} \times \min(d_r, d_c)$ obtained by applying

$$R_{\text{sum}} = \begin{bmatrix} \text{sum} & | & \text{sum} \\ \hline \text{---} & | & \text{---} \\ \hline \text{---} & | & \emptyset \end{bmatrix}$$

to the rows of the node-arc incidence matrix and

$$R_{\text{min}} = \begin{bmatrix} \text{min} & | & \text{---} \\ \hline \text{---} & | & \text{---} \\ \hline \text{min} & | & \emptyset \end{bmatrix}$$

to the columns is in fact an aggregate network problem with the following characteristic:

Theorem 12: $R_{\text{sum}} \times \min$ is a strong relaxation replacement

rule for problems with $y \geq 0$.

Thus a rigorous methodology is provided for creating an aggregate network problem from an original problem (P) and a partition d_r of the node set N. Furthermore, the optimal objective function value of $R_{\text{sum}} \times \min(d_r, d_c)$ provides a lower bound on the optimal objective function value of (P), i.e.

$$v(R_{\text{sum}} \times \min(d_r, d_c)) \leq v(P)$$

Lee suggests that an initial partition d_r of the node set N be selected based upon a "measure of compatability" between pairs of nodes.

Definition 22: The coefficient of commonality $\Pi(i,j)$ of two nodes $i, j \in N$ is defined as follows

$$\Pi(i,j) = \frac{c[(p(i) \cap p(j)) \cup (q(i) \cap q(j))]}{c[(p(i) \cup p(j)) \cup (q(i) \cup q(j))]}$$

where $p(i) = \{j \in N: a_{ji} \in A\}$
 $q(i) = \{j \in N: a_{ij} \in A\}$
 $c[\cdot] = \text{the cardinality of } [\cdot]$

Lee concludes with a description of a "general disaggregation process" and the development of a stopping criterion. The disaggregation process involves the refinement of a subset of the given partition. Given an aggregate problem P^u , an aggregate node, say the one associated with $t(q)$, is refined into two components. The cells of the node-arc incidence matrix of the aggregate node $t(q)$ are

(i) $t(q)$ by $t(q) \times t(p)$ for all $p \neq q$

(ii) $t(r)$ by $t(r) \times t(q)$ for all $r \neq q$

Partitioning $t(q)$ into $t_1(q)$, $t_2(q)$ the above cells are refined into

$t_1(q)$ by $t_1(q) \times t(p)$

$t_2(q)$ by $t_2(q) \times t(p)$

with at least one refinement column non-null for all $p \neq q$ and

$t(r)$ by $t(r) \times t_1(q)$

$t(r)$ by $t(r) \times t_2(q)$

with at least one refinement column non-null for all $r \neq q$.

For $p = q$, if aggregate column $t(q) \times t(q)$ exists, the cell $t(q)$ by $t(q) \times t(q)$ becomes four cells

$t_1(q)$ by $t_1(q) \times t_1(q)$

$t_1(q)$ by $t_2(q) \times t_1(q)$

$t_2(q)$ by $t_1(q) \times t_2(q)$

$t_2(q)$ by $t_2(q) \times t_2(q)$

A solution Y^V of the (refined) aggregate problem P^V can be constructed by mapping solution Y^U of aggregate problem P^U into the space of P^V . The solution map $\Psi: E^U \rightarrow E^V$ is a point to set map and, since P^U is a relaxation of P^V , this mapping may produce an infeasible solution. Defining $Y(\cdot)$ and $E(\cdot)$ to be the variable associated with and the capacity of the arc indicated in parenthesis, Lee defines the (disaggregation) solution map Ψ_{sum} as any solution to

$$Y^V(t_1(q) \times t(p)) \leq E(t_1(q) \times t(p)), \text{ all } p \neq q$$

$$Y^V(t_2(q) \times t(p)) \leq E(t_2(q) \times t(p)), \text{ all } p \neq q$$

$$Y^V(t(r) \times t_1(q)) \leq E(t(r) \times t_1(q)), \text{ all } r \neq q$$

$$Y^V(t(r) \times t_2(q)) \leq E(t(r) \times t_2(q)), \text{ all } r \neq q$$

$$Y^V(t_1(q) \times t_1(q)) = 0$$

$$Y^V(t_2(q) \times t_2(q)) = 0$$

$$Y^V(t_1(q) \times t_2(q)) \leq E(t_1(q) \times t_2(q))$$

$$Y^V(t_2(q) \times t_1(q)) \leq E(t_2(q) \times t_1(q))$$

$$Y^V(t_1(q) \times t(p)) + Y^V(t_2(q) \times t(p)) = Y^U(t(q) \times t(p)), \text{ all } p$$

$$Y^V(t(r) \times t_1(q)) + Y^V(t(r) \times t_2(q)) = Y^U(t(r) \times t(q)), \text{ all } r$$

$$Y^V(t(s) \times t(w)) = Y^U(t(s) \times t(w)) \text{ for } s \neq q, w \neq q.$$

Such a solution is obtained by splitting Y^U and can violate at most the two flow conservation constraints at the refined nodes. Addition of one artificial arc (either $t_1(q) \times t_2(q)$ or $t_2(q) \times t_1(q)$) will remove the infeasibility. Finally, a stopping criterion is provided and it is shown to produce an optimal solution of the original problem under the disaggregation map proposed in Theorem 13.

Stopping Criterion: Let Y^* be an optimal solution to a surrogate problem $P^* = R_{\text{sum}} \times \min(d_r, d_c)$. If all nonzero values of Y^* correspond to aggregate $t(q) \times t(p)$ generated from singleton sets $t(q) = \{u\}$ and $t(p) = \{v\}$, terminate the process. Otherwise refine an aggregate node $t(q)$ or $t(p)$ associated with an aggregate arc $t(q) \times t(p)$ with nonzero

flow.

Theorem 13: Let y^* be an optimal solution to a surrogate problem P^* satisfying the stopping criterion. With

y_{qp} as the flow on aggregate arc $t(q) \times t(p)$

and

y_{uv} as the flow on original arc $(u,v) \in t(q) \times t(p)$

when $y_{qp} > 0$,

take $y_{uv} = y_{qp}$ for sole arc $(u,v) \in t(q) \times t(p)$,

when $y_{qp} = 0$

take $y_{uv} = 0$ for all arcs $(u,v) \in t(q) \times t(p)$;

then y is an optimal solution to the original problem with value $z(P^*)$.

APPENDIX C: GNET Design And Implementation

The seminal article "Design And Implementation Of large Scale Primal Transshipment Algorithms" by Bradley, Brown and Graves [2] describes the design and implementation of a family of large scale, minimum cost network algorithms. In addition, the general capacitated minimum cost network problem and the bounded variable revised simplex algorithm are reviewed. Since this article provides the theoretical background and implementational details of the GNET computer code which is incorporated in the AGNET code developed in this work, we shall discuss it in detail in this Appendix.

C-I: The Primal Simplex Algorithm

This section reviews the bounded variable revised simplex method. Small letters denote column vectors and a prime indicates transpose. Capital letters indicate matrices, superscripts denote columns and subscripts denote rows.

The capacitated minimum cost network problem is defined as:

$$\begin{aligned} (1) \quad & \text{MIN} \quad c'x \\ & \text{S.T.} \quad Ax = b \\ & \quad \quad 1 \leq x \leq v \end{aligned}$$

Lower bounds on the variables can be eliminated by transformation to yield:

$$\begin{aligned}
 (2) \quad & \text{MIN} \quad c'x \\
 & \text{S.T.} \quad Ax = b \\
 & \quad \quad 0 \leq x \leq u
 \end{aligned}$$

Upper bounds are handled implicitly. When a variable reaches its upper bound, it is replaced ("reflected") by $u_k - x_k$; column k has its sign changed and the right-hand side is transformed to $b - A^k u_k$. A record is kept of which variables are "out of the basis at their upper bounds" facilitating easy recovery of the original problem's solution.

The matrix A of technological coefficients may be partitioned into $A = (B, N)$ with B an $(m \times m)$ matrix of linearly independent columns (the "basis"). It may be necessary to include artificial and/or slack variables in A to effect such a partition.

For a feasible basis, there always exists a unique solution $\hat{x}$ such that

$$(3) \quad B\hat{x} = b$$

and the unique "basic solution" is given by

$$x^0 = \begin{bmatrix} \hat{x} \\ 0 \end{bmatrix}$$

and, upon partitioning c in the same manner as A , we obtain

$$(4) \quad c'x^0 = (c'_B, c'_N) \begin{bmatrix} x \\ 0 \end{bmatrix} = c'_B \hat{x}$$

Any solution can be written in terms of the basis as

$$(5) \quad Ax = (B, N) \begin{bmatrix} x_B \\ x_N \end{bmatrix} = Bx_B + Nx_N = b$$

Since B is a basis, there exists a unique transformation Z

such that

$$(6) \quad BZ = N$$

Multiplying (6) by x_N and subtracting from (3) yields

$$(7) \quad B(\hat{x} - Zx_N) + Nx_N = b$$

Subtracting (7) from (5) gives

$$B(x_B - (\hat{x} - Zx_N)) = 0$$

and since the columns of B are linearly independent

$$x_B = \hat{x} - Zx_N$$

and the general solution becomes

$$(8) \quad x = \begin{matrix} x_B \\ x_N \end{matrix} = \begin{matrix} \hat{x} - Zx_N \\ x_N \end{matrix}$$

It is now easy to compare the value of x to any current solution x^0 and to identify an improved solution if one exists. The value of the generic solution, using (4) and (6) is

$$(9) \quad c'x = c'_B x + (c'_N - u'N)x_N$$

where u , often referred to as the "simplex multiplier," is obtained from

$$(10) \quad u'B = c'_B$$

From (9), a necessary condition for an improvement in the solution is that there must exist a column of N , N^k , such that

$$(11) \quad c_k - u'N^k < 0$$

With such a column, consider a specific vector obtained from

(8) by taking all components of x_N to be zero except x_k

$$(12) \quad x' = (\hat{x} - Z^k x_k, 0, \dots, x_k, \dots, 0)$$

For feasibility, the components $z_{ik} > 0$ require

$$(13) \quad x_k \leq \hat{x}_i / z_{ik}$$

and the components $z_{ik} < 0$ require

$$(14) \quad x^k = (\hat{x}_i - u_i) / -z_{ik}$$

and

$$x_k \leq u_k$$

If the least bound on x_k is u_k , then x_k stays out of the basis but goes to its upper bound and the new solution is

$$\bar{x} = \hat{x} - z^k u_k$$

with

$$B\bar{x} = b - N^k u_k$$

If (13) is the least bound, taking $x_k = \hat{x}_i / z_{ik}$ leads to the exchange of B^i and N^k in the partition of A and the new basic solution is

$$\bar{x}_r = \hat{x}_r - z_{rk}(\hat{x}_i / z_{ik}), \quad r \neq i$$

$$\bar{x}_i = \hat{x}_i / z_{ik}$$

If (14) is the least bound, taking $x_k = (u_i - \hat{x}_i) / -z_{ik}$ requires the basis exchange of B^i and N^k and yields

$$\bar{x}_r = \hat{x}_r - z_{rk}(u_i - \hat{x}_i) / -z_{ik}, \quad r \neq i$$

$$\bar{x}_i = (u_i - \hat{x}_i) / -z_{ik}$$

as the new basic solution with x_i at its upper bound.

Assume there is a current basis B , a current solution $\hat{x}$ to $B\hat{x} = b$ and a current solution u of $u'B = c'_B$. An iteration of the simplex method is summarized:

Step S1, Priceout. Select a candidate variable to enter the basis with

$$(c_k - u'N^k) < 0$$

Step S2, Ratio Test. Find the greatest bound such that
(with $BZ^k = N^k$):

- a) $x_k \leq u_k$,
- b) $x_k \leq x_r/z_{rk}$ for $z_{rk} > 0$,
- c) $x_k \leq (u_r - x_r)/-z_{rk}$ for $z_{rk} > 0$.

Step S3, Pivot. Update the primal solution:

$$\bar{x} = \hat{x} - x_k N^k.$$

If x_k is bounded by case a), reflect x_k and leave the basis and dual solution unchanged. Otherwise, change the basis by exchanging B^i and N^k , for case c) reflect x_i . For case b) or c), find the new dual solution to $\bar{u}'B = \bar{c}'_B$.

The following proposition shows that the $\bar{u}$ may be achieved directly from u by a simple direct transformation rather than explicitly inverting the basis matrix to obtain $Q = B^{-1}$.

Proposition. $\bar{u} = u + \lambda Q_i$, where Q_i is the i^{th} row of the inverse of B .

The proof of this proposition shows that $\lambda = (c_k - uN^k)/z_{ik}$.

The most elementary and explicit way to update Q after and exchange of B^i and N^k is to carry the following "tableau:"

$$\begin{bmatrix} QN & Qb \\ (c_N - u'N) \end{bmatrix}$$

The revised simplex method generates these elements as needed by access to columns of N , c and b and use of Q . Commonly, Q is expressed as the product of "eta" column

vectors, each representing the pivotal transformations necessary to generate Q from an initial basis.

C-II: Primal Network Specialization

The fundamental result that permits efficient basis representation and updating is the well-known result that any basis for a transshipment problem can be put in (upper) triangular form by a simple permutation. This triangularity allows for efficient solution of (6) by backward substitution and (10) by forward substitution. Also, an initial (artificial) basis can always be constructed by introducing for each row in (2) a unit vector with sign matching the right-hand side. The following theorem by Dantzig summarizes this result.

Theorem Any basis B extracted from A for a transshipment problem can be triangulated by rearranging rows and rearranging columns.

It is also well-known that a graph can be defined which represents the basis of a transshipment problem. Let

$$I = \{i_1, i_2, \dots, i_m\}$$

be a row ordering corresponding to a triangulated basis, B .

Associate with each node i_k the row number $p(i_k)$ of the offdiagonal element in the k^{th} column of the triangulated basis. If there is no nonzero offdiagonal element in the k^{th} column, set $p(i_k)$ to $m+1$. Define a graph with nodes $1, 2, \dots, m+1$ and let $(i_k, p(i_k))$, $k=1, 2, \dots, m$ be directed arcs from i_k to $p(i_k)$. The graph is called a rooted arborescence with root $m+1$ since there is a directed path from each node

to the root node $m+1$. If the directions of the arcs are ignored, it is a "tree" in the parlance of the graph theory literature.

The node $p(i)$ is called the predecessor of i and the resultant graph is called the predecessor graph. This predecessor graph is related to the "basis tree" as first defined by Koopmans, but the predecessor graph reverses the orientation of some of the original basis arcs in order to obtain the rooted arborescence with the artificial node $m+1$ serving as the root.

The function $p()$ can be iterated to obtain a unique backpath from any node to the root. The successors of node i are all the nodes whose backpaths include the node i . The immediate successors of i are all the nodes such that the first node on their backpaths is node i .

For a network linear program, each node is identified with an equality constraint and each arc k from node i to node j is identified with a variable with column N^k having a $+1$ in row i , a -1 in row j and zeros elsewhere. For the ensuing discussion it is assumed that the basis B is triangular and that row i of B corresponds to node i . Also, it is assumed that the diagonal entries of B are all $+1$; if B initially had a -1 on the diagonal, reflection $u^k = x^k$ transforms it to a $+1$. The simplex method can now be viewed in terms of the special structure provided by network problems.

Step S1, Priceout. Given the vector of simplex multipliers, u , the priceout step S1 of the simplex method becomes simply

$$c_k - u_i + u_j$$

involving only addition operations.

Step S2, Ratio Test. This step involves solution of the system of equations $BZ^k = N^k$ for Z^k . This calculation can be described by showing how to solve $BQ^j = e_j$ for Q^j , where e_j is the j th unit vector and Q^j is the j th column of the inverse of B . Since B is upper triangular, Q^j can be obtained by simple back substitution with the $m, m-1, \dots, j+1$ elements of Q^j equaling zero, the j th element is $+1$. Setting the j th element in the modified right-hand side equal to 0 puts a 1 in the row corresponding to the offdiagonal -1 in the j th column of B . This is the row corresponding to $p(j)$. The procedure is continued until a column of B is encountered with no nonzero offdiagonal elements. The resulting Q^j is all zeros except for ones in the rows corresponding to $j, p(j), p(p(j)),$ etc.

In terms of the predecessor graph, all arcs traversed on the backpath from j to the root have an element 1 in Q^j and zeros elsewhere. This important observation implies that the predecessor function $p()$, along with I , can be used directly to generate the basis inverse. The calculation of Z follows in a straight forward fashion according to

$$z^k = Q^i - Q^j.$$

Step S3, Pivot. In this step the primal solution is updated

using z^k . Also, as described in the Proposition, the simplex multipliers, u , can be simply updated rather than recalculated at each pivot. The update applies the l^{th} row of Q , Q_l , where the outgoing arc is the l^{th} column of B . Since, as described above, an element of Q^j is equal to 1 only if the corresponding basic arc is traversed on the backpath from node j to the root, it follows that Q_l is all zeros except for 1 in the l^{th} element and a 1 for any element j such that the outgoing arc is on the backpath from j to the root. In terms of the predecessor graph, Q_l is 1 for node l and each of its successors and 0 elsewhere. Since the nonzero element of z^k are either +1 or -1, the constant λ in the Proposition is either plus or minus

$$c_k - u_i + u_j$$

where k is the incoming arc. Since Q_l is all 0's and 1's, the update of u is accomplished by adding λ to u_l and to the u 's of the successors of node l .

C-III: Implementation

In this section, upper-case Roman letters indicate program variables and the addition of parentheses denotes an array.

The network is stored in three arc-length arrays: the tail nodes of each arc, $T()$; the costs of each arc, $C()$; and the upper bound capacities of each arc, $CP()$. Arcs are sorted according to head node with all arcs having the same head node being stored contiguously. One node-length array,

$H()$, is used to indicate the first location of $T()$ where arcs with a given head node are stored; i.e. arcs with head node i are stored in locations $H(i)$ to $H(i+1)-1$. Arcs which are out of the basis at their upper bounds are marked with a negative sign bit on the capacity, $-CP()$.

The predecessor graph is stored and recovered using a predecessor array, $P()$. A positive value in the predecessor array indicates that the corresponding basic arc has been reflected from its original orientation in the basis tree so that it is now directed toward the root in the predecessor graph. For example, if $P(i)$ is positive, the arc $(P(i), i)$ is basic but its orientation is reversed in the predecessor graph to $(i, P(i))$ so that it is oriented toward the root. The flow on arc $(i, P(i))$ is stored in the array $X()$ at location $X(i)$.

The array $CPX()$ is used to store capacity minus flow on basic arcs. This quantity is necessary at step S2 of the simplex method for basic arcs with $z_{ij} < 0$. The simplex multipliers are stored in a node length array $U()$.

With these data structures, it is easy to identify the outgoing arc at step S2 of the simplex method. Define the join to be the first node on the backpath from i to the root that is also on the backpath from j to the root. Thus, the join is the backpath node beyond which all terms of z^k are zero. Since the possible outgoing arcs correspond to the nonzero entries of z^k , these arcs are identified as those on

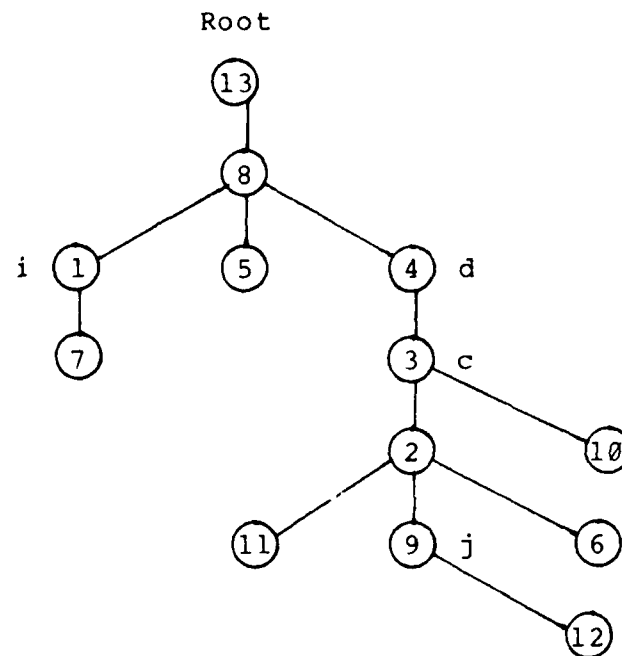
the backpath from i to the join and those on the backpath from j to the join for entering arc (i,j) .

To facilitate efficient identification of the backpaths from i to the join and from j to the join, an additional node-length array $D()$ is needed which indicates the depth of a particular node in the predecessor graph (basis tree). The depth of the root is 1; the depth of the nodes adjacent to the root is 2; etc. The depth array is used to indicate which backpath is "deeper" in the tree and should be iterated in the search for the join. If both backpath nodes have the same depth, they are compared for equality. If equality holds, the join has been found. If the two backpath nodes are different but of the same depth, both backpaths are iterated synchronously in the search for the join. As discussed in the original article, arcs on the backpath from i to the join are the $+1$ elements of z^k and those on the backpath from j to the join are the -1 elements. The ratio test at step S2 of the simplex method becomes simply:

$$\min \left\{ \begin{array}{ll} CP(k) & \text{the capacity of incoming arc } k \\ X() & \text{for arcs on the backpath from } i \\ & \text{to the join} \\ CPX() & \text{for arcs on the backpath from } j \\ & \text{to the join} \end{array} \right.$$

Consider the typical pivot with entering arc (i,j) and leaving arc (c,d) . Call the backpath from j to c the pivot stem. Figure C-1 below illustrates for Example 2.1 the

Before Pivot:



Entering arc $(i,j) = (1,9)$, leaving arc $(c,d) = 3,4$

After Pivot:

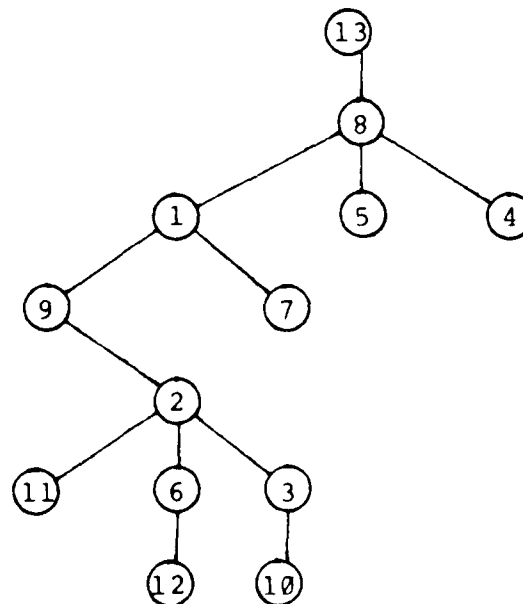


Figure C-1: Predecessor Graph Before And After Pivot

predecessor graphs before and after such a pivot with entering arc (1,9) and leaving arc (3,4). The subarborescence with root c is "rehung" from node i , and the nodes on the pivot stem have their predecessor relationships reversed. The flows $X()$ and $CPX()$ change only on the backpaths from i and j to the join. The simplex multipliers, $U()$, and the depth, $D()$, are recomputed for all nodes in the subarborescence with root c .

The update of the simplex multipliers by the addition of λ is accomplished efficiently by using an array $IT()$ which indicates the preorder traversal of the predecessor graph. $IT(i)$ is the next node to "visit" from node i in preorder. $IT()$ of the last node of the predecessor graph visited is the artificial node $m+1$ making $IT()$ a circular list. This preorder traversal will visit all nodes of a subarborescence contiguously, precisely as is required to update the simplex multipliers.

Table C-1 shows the array values for the example pivot of Figure C-1. Figure C-2 gives a flow diagram outlining the logic involved in the pivot segment.

The original article by Bradley, Brown and Graves goes on to discuss alternative data structures, pricing mechanisms, computational testing and results. Since these topics do not bear directly on this research, they will not be reviewed here.

Arrays Before Pivot:

Node:	i	1	2	3	4	5	6	7	8	9	10	11	12	13
Heads:	H	0	0	1	2	3	4	5	6	9	13	15	16	17
Predecessor:	P	-8	-3	-4	-8	-8	2	1	13	2	3	2	6	
Traversal:	IT	7	11	2	3	4	12	5	1	6	13	9	10	8
Flow:	X	6	2	4	4	5	0	3	96	4	5	2	0	
	CPX	1	9	2	5	0	20	18	0	8	3	11	16	
Depth:	D	2	4	3	2	2	5	3	1	5	4	5	6	0
Multiplier:	U	56	76	42	19	24	31	-1	0	-23	16	62	-3	

Arrays After Pivot:

Node:	i	1	2	3	4	5	6	7	8	9	10	11	12	13
Heads:	H	0	0	1	2	3	4	5	6	9	13	15	16	17
Predecessor:	P	-8	-9	2	-8	-8	2	1	13	1	3	2	6	
Traversal:	IT	9	11	10	13	4	12	5	1	2	7	6	3	8
Flow:	X	4	6	7	6	5	0	3	96	3	5	2	0	
	CPX	3	6	4	3	0	20	18	0	2	3	11	16	
Depth:	D	2	4	5	2	2	5	3	1	3	6	5	6	0
Multiplier:	U	56	94	60	19	24	49	-1	0	-5	86	80	15	

Table C-1: Arrays Before And After An Example Pivot

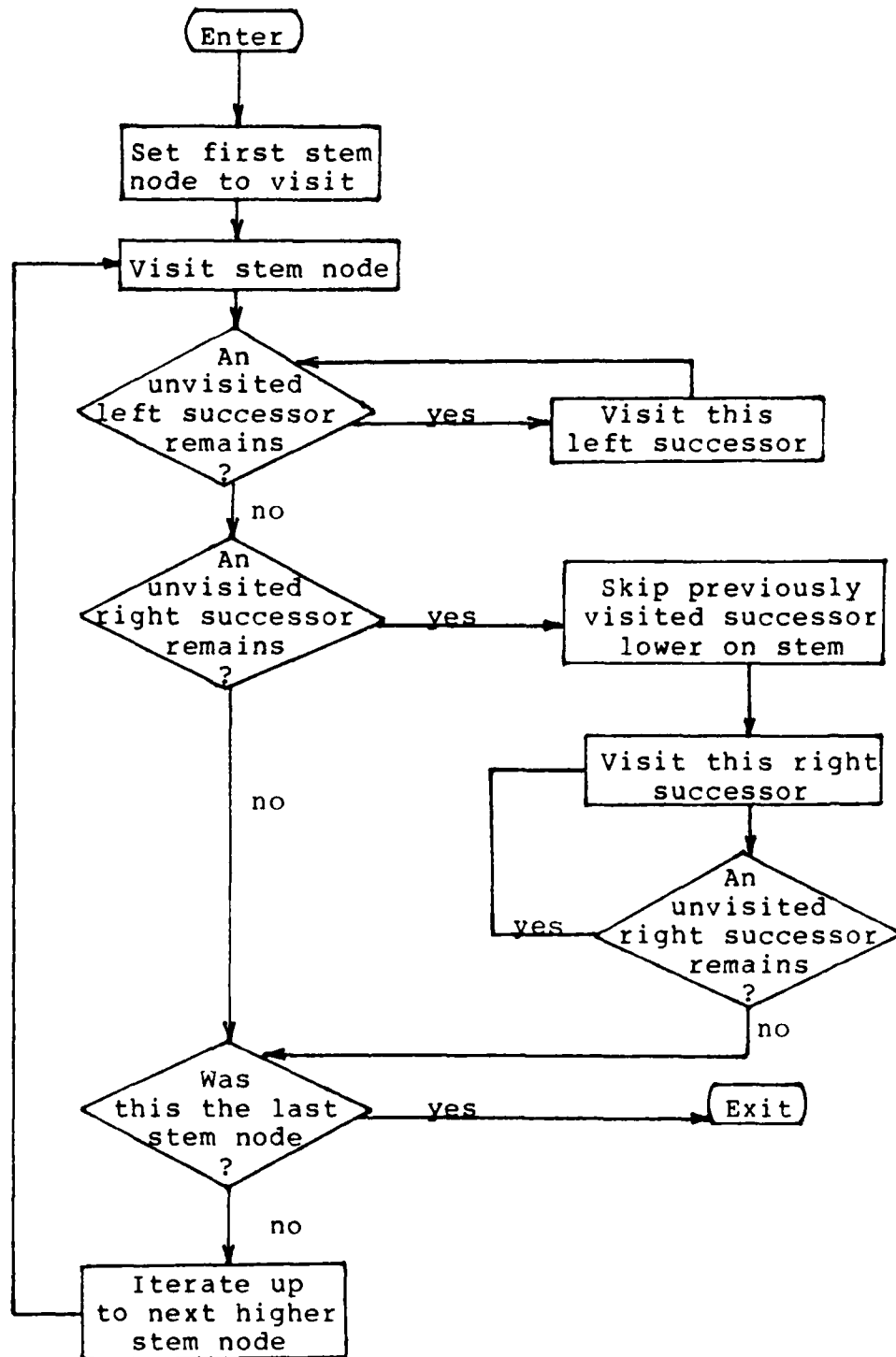


Figure C-2: Pivot Segment Logic

APPENDIX D-I: LIST OF VARIABLES ADDED TO GNET

AALOC(,): matrix whose (i,j) entry contains the arc set designation of aggregate arc (i,j), i.e. original arcs comprising aggregate arc (i,j) are in arc set AALOC(i,j) and the first original arc of this subset is located at ASETPT(AALOC(i,j))

AALOCO(,): "AALOC- Old" matrix to save AALOC(,) values from $\overline{NP}(r)$ necessary for the creation of $\overline{NP}(r+1)$ and execution of DMG

ABSORB: the number of absorbed arcs in the current aggregate problem, used in the calculation of the aggregation measure

ACP(): capacity (upper bound) of aggregate arcs of current aggregate problem

ACPO(): "ACP- Old" array to save values of ACP() from $\overline{NP}(r)$ necessary for creation of $\overline{NP}(r+1)$ and execution of DMG

ADSUP(): adjusted supplies for nodes of $\overline{SNP}(r+1)$

AH(): head node of aggregate arcs of the current aggregate problem, i.e. of problem $\overline{NP}(r)$

ALB(): lower bounds of aggregate arcs of current aggregate problem

ASETPT(): entries point to first original arc of an arc subset

ASUP(): supply of the corresponding aggregate node for the

current aggregate problem

AT(): tail node of aggregate arcs of the current aggregate problem, i.e. of problem $\overline{NP}(r)$

CMIN(): cost of aggregate arcs of current aggregate problem

CO(): cost of original arcs

COR(): indicates the correspondence between node designations in $\overline{SNP}(r+1)$ and the consecutive integer designation required by GNET. $COR(i) = j$ implies node i of $\overline{SNP}(r+1)$ is designated node j for solution by GNET

CORIN(): the inverse of $COR()$, i.e. $CORIN(i) = j$ implies node i in GNET designation is node j in $\overline{SNP}(r+1)$

E(): array used in sort of original arcs to determine arc subsets

ELIM: the number of arcs eliminated in the current aggregate problem, used in calculation of the aggregation measure

F(): array used in sort of original arcs to determine arc subsets

FIRST: variable used to indicate the location of the first node of a particular subset, used in calculation of $ASUP()$

HO(): head node of original arcs

INDEX(): index of arrays, i.e. integers 1,2,...

ISTAR: the index of the node subset to be refined to create the next aggregate problem

K(): indicates the subset to which each original node belongs in the current partition of the node set

LAST: variable to indicate location of last node in a particular node subset, used in computation of ASUP()

LBO(): lower bound on original arcs

LINK(): linked list that links original arcs according to arc subsets

LLINK(): linked list used in binary tree representation of basis tree, points to left-most child

LNDV(): "last node of this depth visited" during pre-order traversal of basis tree, used in construction of binary tree

MINPI: the minimum PI() value, used to determine the node subset to refine

MO: number of original nodes

MP01: number of original nodes + 1 (designation of super source)

MP02: number of original nodes + 2 (designation of super sink)

NAGSET: number of (non-empty) arc subsets for the current partition, includes arc subsets A(i,i)

NARCS: the number of original arcs

NSETPT(): entries point to location of NTIL() where the first node of a node subset is located

NR: the number of aggregate nodes in the current aggregate problem

NTIL(): original node numbers sorted according to node subset determined by K()

NUMAO(): "NUMARC- Old" array to store the values of

NUMARC() for the current problem necessary for the creation of $\overline{NP}(r+1)$ and execution of DMG

NUMARC(): entries are the number of original arcs in the corresponding arc subset

PI(): the P_i (node cutting potential) of each aggregate node of the current aggregate problem and its optimal solution

R: the R th aggregate problem, the r in the $N(r)$ and $\overline{NP}(r)$ notation

REPARC(): entries are the representative arcs of the corresponding arc subset

RLINK(): linked list used in binary tree representation of basis tree, entries point to next sibling to the right

RMEAS: the aggregation measure $M(r)$

RMEAS1: M_1 of the aggregation measure: # eliminated nodes / (# nodes - 1)

RMEAS2: M_2 of the aggregation measure: % of original arcs eliminated

RMEAS3: M_3 of the aggregation measure: % of original arcs absorbed

SET(): yields the set number or designation of the corresponding arc subset

SACP(): capacities of arcs of $\overline{SNP}(r+1)$

SALB(): lower bounds of arcs of $\overline{SNP}(r+1)$

SAT(): tail node of arcs of $\overline{SNP}(r+1)$

SAH(): head node of arcs of $\overline{SNP}(r+1)$

SCMIN(): cost of arcs of $\overline{SNP}(r+1)$
SM: the number of nodes of $\overline{SNP}(r+1)$
SNAGSE: the number of aggregate arcs to be included in $\overline{SNP}(r+1)$
SNODE(): listing of nodes to be included in $\overline{SNP}(r+1)$
SOLN(): array to store solution values for aggregate arcs of the current problem
SOLNO(): "SOLN- Old" saves values of SOLN() from $\overline{NP}(r)$ necessary for execution of DMG
SNP(): stores SET() designation of aggregate arcs to be included in $\overline{SNP}(r+1)$
SSOLN(): solution values of arcs of $\overline{SNP}(r+1)$
SUPO(): supply of original nodes
TEMP(): temporary array to hold LINK() entries during determination of NTIL(), NTILIN() and NSETPT()
TO(): tail node of original arcs
UBO(): upper bound on original arcs

APPENDIX D-II: ARC SET SORTING

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC  ARCSORT, REF. HOROWITZ P. 364      CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

69 R = R + 1
   IF(NODESR + 1 .GT. MO) STOP
   DO 68 I = 1, NAD
     TO(I) = IABS(TO(I))
     ASETPT(I) = 0
68 CONTINUE
   DO 770 I = 1, NOD
     NTIL(I) = 0
     NTILIN(I) = 0
     NSETPT(I) = 0
     DO 771 J = 1, NOD
       AALOC(I,J) = 0
771 CONTINUE
770 CONTINUE
   NODESR = 0
   DO 71 J = 1, MO
     IF(K(J) .GT. NODESR) NODESR = K(J)
71 CONTINUE
   DO 81 I = 1, NARCS - 1
     LINK(I) = I + 1
81 CONTINUE
   DO 82 I = NARCS, NAD
     LINK(I) = 0
82 CONTINUE
   PTR = 1
   DO 83 I = 1, 2
     DO 84 J = 1, MO
       E(J) = 0
       F(J) = 0
84 CONTINUE
92 IF(I .EQ. 2) GOTO 85
   II = K(TO(PTR))
   GOTO 86
85 II = K(HO(PTR))
86 IF(F(II) .NE. 0) GOTO 87
   F(II) = PTR
   GOTO 88
87 LINK(E(II)) = PTR
88 E(II) = PTR
   PTR = LINK(PTR)
   IF(PTR .NE. 0) GOTO 92
   DO 93 J = 1, NODESR
     IF(F(J) .NE. 0) GOTO 89
93 CONTINUE
89 PTR = F(J)
   JJ = E(J)

```

```
      DO 91 KK = J+1, NODESR
        IF(F(KK) .EQ. 0) GOTO 91
        LINK(JJ) = F(KK)
        JJ = E(KK)
91      CONTINUE
        LINK(JJ) = 0
83 CONTINUE
```

APPENDIX D-III: SETTING OF ARRAYS ASETPT() AND AALOC()

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC    SETTING AALOC, ASETPT                                CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      PTR = 1
      DO 101 J = 1, NODESR
        IF(F(J) .EQ. 0) GOTO 101
        I = F(J)
103      ASETPT(PTR) = I
        TT = K(TO(I))
        AALOC(TT,J) = PTR
        PTR = PTR + 1
        IF(I .EQ. E(J)) GOTO 101
102      I = LINK(I)
        TTNEXT = K(TO(I))
        IF(TTNEXT .NE. TT) GOTO 103
        IF(I .EQ. E(J)) GOTO 101
        GOTO 102
101 CONTINUE
      NAGSET = PTR - 1

```

APPENDIX D-IV: SETTING ARRAYS NTIL(), NTILIN() AND NSETPT()

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC  DETERMINING NTIL,NTILIN,NSETPT                                CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 169 I = 1, NAD
        TEMP(I) = LINK(I)
169  CONTINUE
      DO 72 J = 1,MO
        E(J) = 0
        F(J) = 0
72  CONTINUE
      DO 73 I = 1, MO
        II = K(I)
        IF(F(II) .NE. 0) GOTO 74
        F(II) = I
        GOTO 75
74  LINK(E(II)) = I
75  E(II) = I
73  CONTINUE
      PTR = 1
      DO 76 I = 1, NODESR
        J = F(I)
        NSETPT(I) = PTR
77  NTIL(PTR) = J
        NTILIN(J) = PTR
        PTR = PTR + 1
        IF(J .EQ. E(I)) GOTO 76
        J = LINK(J)
        GOTO 77
76  CONTINUE
      NSETPT(NODESR + 1) = MO + 1
C
      DO 108 I = 1, NAD
        LINK(I) = TEMP(I)
108  CONTINUE

```

APPENDIX D-V: CONSTRUCTION OF THE AGGREGATE PROBLEM

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC      CREATING THE AGGREGATE PROBLEM      CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 111 I = 1, NAD
        AH(I) = 0
        AT(I) = 0
        CMIN(I) = 1000000
        ACP(I) = 0
        ALB(I) = 0
        NUMARC(I) = 0
        REPARC(I) = 0
111 CONTINUE
      DO 126 I = 1, MO
        ASUP(I) = 0
126 CONTINUE
      DO 112 I = 1, NAGSET
        PTR = ASETPT(I)
        IF(K(TO(PTR)) .EQ. K(HO(PTR))) GOTO 112
        AT(I) = K(TO(PTR))
        AH(I) = K(HO(PTR))
113      IF(CO(PTR) .GE. CMIN(I)) GOTO 114
        CMIN(I) = CO(PTR)
        REPARC(I) = PTR
114      ACP(I) = ACP(I) + UBO(PTR)
        ALB(I) = ALB(I) + LBO(PTR)
        NUMARC(I) = NUMARC(I) + 1
        IF(LINK(PTR) .EQ. ASETPT(I+1)) GOTO 116
        IF(LINK(PTR) .EQ. 0) GOTO 116
        PTR = LINK(PTR)
        GOTO 113
116      TO(REPARC(I)) = -TO(REPARC(I))
112 CONTINUE
      DO 127 I = 1, NAGSET
        SET(I) = I
127 CONTINUE
C
      DO 141 I = 1, NODESR
        FIRST = NSETPT(I)
        LAST = NSETPT(I+1) - 1
        IF(I .EQ. NODESR) LAST = MO
        DO 142 J = FIRST, LAST
          ASUP(I) = ASUP(I) + SUPO(NTIL(J))
142      CONTINUE
141 CONTINUE
C
      DO 144 I = 1, NODESR
        IF(ASUP(I) .EQ. 0) GOTO 144

```

```

      IF(ASUP(I) .LT. 0) GOTO 146
      AT(NAGSET + I) = NODESR + 1
      AH(NAGSET + I) = I
      CMIN(NAGSET + I) = 0
      ACP(NAGSET + I) = ASUP(I)
      ALB(NAGSET + I) = ASUP(I)
      GOTO 144
146   AT(NAGSET + I) = I
      AH(NAGSET + I) = NODESR + 2
      CMIN(NAGSET + I) = 0
      ACP(NAGSET + I) = -ASUP(I)
      ALB(NAGSET + I) = -ASUP(I)
144  CONTINUE

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCC
C   CONSTRUCTING THE AGGREGATION MEASURE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCC

```

```

      IF(SWITCH .EQ. 1 .OR. NPSOLV .GT. 0) GOTO 214
      ELIM = 0
      ABSORB = 0
      DO 280 I = 1, NAGSET
        IF(NUMARC(I) .EQ. 0) THEN
          ELIM = ELIM + 1
          J = ASETPT(I)
281      J = LINK(J)
          IF(J .EQ. ASETPT(I+1)) GOTO 280
          ELIM = ELIM + 1
          GOTO 281
        ENDIF
        IF(NUMARC(I) .EQ. 1) GOTO 280
        ABSORB = ABSORB + NUMARC(I) - 1
280  CONTINUE
      RMO = MO
      RNODES = NODESR
      RELIM = ELIM
      RABSOR = ABSORB
      RNARCS = NARCS
      RMEAS1 = (RMO - RNODES)/(RMO - 1)
      RMEAS2 = RELIM/RNARCS
      RMEAS3 = RABSOR/RNARCS
      RMEAS = RMEAS1 + RMEAS2 + RMEAS3
      WRITE(6, 282) RMEAS1, RMEAS2, RMEAS3, RMEAS
282  FORMAT(1H0, 'AGGREGATION MEASURE...',/,1X,
C          'M1(R) = ',F4.2,2X,
C          'M2(R) = ',F4.2,2X, 'M3(R) = ',F4.2,2X,
C          'M(R) = ',F4.2)

```

APPENDIX D-VI: MARKING OF SOLUTION AND STOPPING CRITERION

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      INITIALIZE SOLN()
      IF(SWITCH .EQ. 1) GOTO 3009
      DO 3001 I = 1, NAGSET
          SOLN(I) = 0
          NUMARC(I) = IABS(NUMARC(I))
          ACP(I) = IABS(ACP(I))
3001 CONTINUE
C
3009 N=H(M+1)-1
      IF( IPRT .GT. 0 ) WRITE(OUTPUT,3010) N
3010 FORMAT( 1X, I6, 5H ARCS )
      WRITE(OUTPUT,3020) TIME,IPVT,NNE,NNS,IPG
3020 FORMAT( 6H0TIME=, I7, 17H SECONDS, PIVOTS=, I6, 6H,
NNE=, I4,
      X 6H, NNS=, I4, 6H, IPG=, I4 )
C
C
C...FIND OPTIMAL SOLUTION
C...DO BASIC ARCS FIRST, CHECK FOR ARTIFICIALS
      DO 3120 N1=1,M
          N2= P(N1)
          IF(N2.LT.0) GO TO 3070
C...TO GET OUTPUT OF BASIC ARCS AT ZERO FLOW,
C      EXCHANGE NEXT TWO STATEMENTS (GN-09620,09640)
          IF(N2 .NE. MP1) GO TO 3040
          IF(CPX(N1) .EQ. 0) GO TO 3120
          WRITE(OUTPUT,3030) N1, CPX(N1), U(N1), N2
3030 FORMAT( 22H ARTIFICIAL ARC,NODE= ,I10, 6H FLOW=
,I10, 6H U(I)=
      X ,I10, 4H N2= ,I10 )
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC      IF(SWITCH .EQ. 0) THEN
              WRITE(OUTPUT,3031)
3031      FORMAT(1H0,'INFEASIBLE NP(R)...SO, NP(0) IS
INFEASIBLE.')
              STOP
          ENDIF
          IF(ADSUP(N1) .GT. 0) THEN
              ART(ARPTR) = CORIN(N1)
              ARH(ARPTR) = NODESR + 1
          ELSE
              ART(ARPTR) = NODESR + 1
              ARH(ARPTR) = CORIN(N1)
          ENDIF
          ARCMIN(ARPTR) = 1000000
          ARLB(ARPTR) = 0

```

```

      ARCP(ARPTR) = 1000000
      ARSOLN(ARPTR) = CPX(N1)
      ARPTR = ARPTR + 1
C
      GO TO 3120
C...SEARCH FOR BASIC REPRESENTATIVE
3040 IXR = H(N1)
      GO TO 3060
3050 IXR = IXR + 1
3060 IF( T(IXR) .NE. N2 ) GO TO 3050
C...MULTIPLE ARC MATCH (COMMENT OUT IF NOT REQUIRED)
      IF( C(IXR) .NE. U(N2) - U(N1)
X      .OR. CP(IXR) .NE. X(N1) + CPX(N1) ) GO TO 3050
      COST = COST + CPX(N1)*C(IXR)
      GO TO 3110
3070 N2 = -N2
C...TO GET OUTPUT OF BASIC ARCS AT ZERO FLOW,
C      EXCHANGE NEXT TWO STATEMENTS (GN-10000,10020)
      IF(N2 .NE. MP1) GO TO 3080
      IF(X(N1) .EQ. 0) GO TO 3120
      WRITE(OUTPUT,3030) N1, X(N1), U(N1), N2
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC
      IF(SWITCH .EQ. 0) THEN
          WRITE(OUTPUT,3032)
3032      FORMAT(1H0,'NP(R) INFEASIBLE...SO, NP(0)
INFEASIBLE.')
          STOP
      ENDIF
      IF(ADSUP(N1) .GT. 0) THEN
          ART(ARPTR) = CORIN(N1)
          ARH(ARPTR) = NODESR + 1
      ELSE
          ART(ARPTR) = NODESR + 1
          ARH(ARPTR) = CORIN(N1)
      ENDIF
      ARCMIN(ARPTR) = 1000000
      ARLB(ARPTR) = 0
      ARCP(ARPTR) = 1000000
      ARSOLN(ARPTR) = X(N1)
      ARPTR = ARPTR + 1
C
      GO TO 3120
3080 IXR = H(N2)
      GO TO 3100
3090 IXR = IXR + 1
3100 IF( T(IXR) .NE. N1 ) GO TO 3090
C...MULTIPLE ARC MATCH (COMMENT OUT IF NOT REQUIRED)
      IF( C(IXR) .NE. U(N1) - U(N2)
X      .OR. CP(IXR) .NE. X(N1) + CPX(N1) ) GO TO 3090
      COST = COST + X(N1)*C(IXR)

```



```

C...MARK BASIC ARC IN ARC LIST
3110 IF(IPRT.GT.0) T(IXR) = -T(IXR)
3120 CONTINUE
C...DO ARCS OUT OF BASIS AT UPPER BOUNDS
    IF(IPRT.GT.0) GO TO 3140
    DO 3130 IXR=1,N
    IF( CP(IXR) .LT. 0 ) COST = COST - CP(IXR)*C(IXR)
3130 CONTINUE
    GO TO 3210
3140 WRITE(OUTPUT,3150)
3150 FORMAT(1H0,'FINAL SOLUTION...',/,1X,'FROM      TO
FLOW')
    DO 3200 N1=1,M
    IXR = H(N1) - 1
    IXSTOP = H(N1+1)
3160 IXR = IXR + 1
    IF( IXR .EQ. IXSTOP ) GO TO 3200
    IF( T(IXR) .GT. 0 ) GO TO 3180
    T(IXR) = -T(IXR)
    IF( IABS(P(N1)) .EQ. T(IXR) ) GO TO 3170
    NX = T(IXR)
    WRITE(OUTPUT,3190) T(IXR),N1,X(NX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MARKING BASIC ARCS AND LOADING SOLN()
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    IF(SWITCH .EQ. 0) THEN
        IF(NUMARC(IABS(AALOC(T(IXR),N1))) .GT. 0)
C      NUMARC(IABS(AALOC(T(IXR),N1))) =
C      -NUMARC(IABS(AALOC(T(IXR),N1)))
        SOLN(IABS(AALOC(T(IXR),N1))) = X(NX)
C      + ALB(IABS(AALOC(T(IXR),N1)))
    ELSE
        IF(NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
.GT. 0)
C      NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) =
C      -NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
        SOLN(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) = X(NX)
C      + ALB(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    GO TO 3160
3170 WRITE(OUTPUT,3190) T(IXR),N1,CPX(N1)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MARKING BASIC ARCS AND LOADING SOLN()
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    IF(SWITCH .EQ. 0) THEN
        IF(NUMARC(IABS(AALOC(T(IXR),N1))) .GT. 0)
C      NUMARC(IABS(AALOC(T(IXR),N1))) =
C      -NUMARC(IABS(AALOC(T(IXR),N1)))
        SOLN(IABS(AALOC(T(IXR),N1))) =

```

```

      C      CPX(N1) + ALB(IABS(AALOC(T(IXR),N1)))
      ELSE
        IF(NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
.GT. 0)
      C      NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) =
      C      -NUMARC(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
        SOLN(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) =
CPX(N1)
      C      + ALB(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
      ENDIF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      GO TO 3160
      3180 IF(CP(IXR).GT.0) GO TO 3160
      CPIXR = -CP(IXR)
      COST = COST + CPIXR*C(IXR)
      WRITE(OUTPUT,3190) T(IXR), N1, CPIXR
      3190 FORMAT(1X,I4,3X,I4,3X,I8)
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C      MARKING OOB ARCS AND LOADING SOLN()
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        IF(SWITCH.EQ. 0) THEN
          IF(ACP(IABS(AALOC(T(IXR),N1))) .GT. 0)
      C      ACP(IABS(AALOC(T(IXR),N1))) =
      C      -ACP(IABS(AALOC(T(IXR),N1)))
          SOLN(IABS(AALOC(T(IXR),N1))) = CPIXR
      C      + ALB(IABS(AALOC(T(IXR),N1)))
        ELSE
          IF(ACP(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) .GT.
0)
      C      ACP(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) =
      C      -ACP(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
          SOLN(IABS(AALOC(CORIN(T(IXR)),CORIN(N1)))) = CPIXR
      C      + ALB(IABS(AALOC(CORIN(T(IXR)),CORIN(N1))))
        ENDIF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      GOTO 3160
      3200 CONTINUE
      3210 WRITE(OUTPUT,3220) COST
      3220 FORMAT( 6H COST= ,I12 )
      IF(M.EQ. MO) STOP
      C
      CTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
      3207 IF(SWITCH.EQ. 1 .OR. SNAGSE.EQ. 0) THEN
        WRITE(OUTPUT,3299)
      3299 FORMAT(1H0,'DISAGGREGATION COMPLETE...')
      ENDIF
      IF(SWITCH.EQ. 0) THEN
        WRITE(OUTPUT,7014)
      7014 FORMAT(1H0,'MARKING OF SOLUTION OF NP(R)...')
      ENDIF
      WRITE(OUTPUT,3191) (NUMARC(I),I=1,NAGSET)

```

```

        WRITE(OUTPUT,3192) (ACP(I),I=1,NAGSET)
        WRITE(OUTPUT,3193) (SOLN(I),I=1,NAGSET)
3191  FORMAT(1H0,'NUMARC: ',30(I3,1X))
3192  FORMAT(1H , 'ACP:      ',30(I3,1X))
3193  FORMAT(1H , 'SOLN:     ',30(I3,1X))
        WRITE(OUTPUT,3298) (TYPE(I),I=1,NAGSET)
3298  FORMAT(1H , 'TYPE:      ',30(I3,1X))
        IF (ARPTR .GT. 1 .AND. NPSOLV .NE. 1) THEN
            WRITE(OUTPUT,3297)
3297  FORMAT(1H0,'ARTIFICIAL ARCS REQUIRED:')
            WRITE(OUTPUT,3194) (ART(I),I=1,ARPTR-1)
3194  FORMAT(1H0,'ART:      ',30(I3,1X))
            WRITE(OUTPUT,3196) (ARH(I),I=1,ARPTR-1)
3196  FORMAT(1H , 'ARH:      ',30(I3,1X))
            WRITE(OUTPUT,3197) (ARCMIN(I),I=1,ARPTR-1)
3197  FORMAT(1H , 'ARCMIN: ',30(I3,1X))
            WRITE(OUTPUT,3198) (ARLB(I),I=1,ARPTR-1)
3198  FORMAT(1H , 'ARLB:     ',30(I3,1X))
            WRITE(OUTPUT,3199) (ARCP(I),I=1,ARPTR-1)
3199  FORMAT(1H , 'ARCP:     ',30(I3,1X))
            WRITE(OUTPUT,3201) (ARSOLN(I),I=1,ARPTR-1)
3201  FORMAT(1H , 'ARSOLN: ',30(I3,1X))
        ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  STOPPING CRITERION
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        IF (SWITCH .EQ. 0) THEN
            DO 7011 I = 1,NAGSET
                IF (IABS(NUMARCS(I)) .GT. 1 .AND. SOLN(I) .GT. 0)
GOTO 3226
            7011  CONTINUE
                WRITE(OUTPUT,7012)
            7012  FORMAT(1H0,'STOPPING CRITERION SATISFIED...',//,1X
                C'ALL ARCS WITH NONZERO FLOW CORRESPOND TO SINGLETON
SUBSET.',//,1X,
                C'SET ALL OTHER ARCS TO ZERO FLOW FOR OPTIMAL
DISAGGREGATION.')
                STOP
            ENDIF

```

APPENDIX D-VII: BINARY TREE TRANSFORMATION
AND PARTITION REFINEMENT

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   UNMARK T() AFTER COMPLETION OF GNET
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
3226 DO 3223 I = 1, NARCS
      T(I) = IABS(T(I))
3223 CONTINUE
      IF(R .EQ. 1) GOTO 3228
      IF(SWITCH .EQ. 0) THEN
        DO 3241 I = 1, NARCS
          PO(I) = P(I)
3241   CONTINUE
          IF(NPSOLV .EQ. 0) THEN
            NPSOLV = 1
            GOTO 152
          ELSE
            NPSOLV = 0
            GOTO 3228
          ENDIF
        ENDIF
      IF(SWITCH .EQ. 1 .AND. NPSOLV .EQ. 0) THEN
        SWITCH = 0
        NPSOLV = 1
        GOTO 152
      ENDIF
      IF(SWITCH .EQ. 1 .AND. NPSOLV .EQ. 1) THEN
        WRITE(OUTPUT,*) 'UH-OH: SWITCH=NPSOLV=1'
        STOP
      ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   TRANSFORMATION OF BASIS TREE TO BINARY TREE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
3228 DO 3224 I = 1, MP1
      LLINK(I) = 0
      RLINK(I) = 0
      LNDV(I) = 0
3224 CONTINUE
      IF(NODESR .EQ. MO) GOTO 3995
      I = IT(MP1)
      LNDV(1) = I
3225 IF(D(IT(I)) .GT. D(I)) THEN
      LLINK(I) = IT(I)
      LNDV(D(IT(I))) = IT(I)
    ENDIF
      IF(D(IT(I)) .EQ. D(I)) THEN
      RLINK(I) = IT(I)
      LNDV(D(IT(I))) = IT(I)
    
```

```

ENDIF
IF(D(IT(I)) .LT. D(I)) THEN
    RLINK(LNDV(D(IT(I)))) = IT(I)
    LNDV(D(IT(I))) = IT(I)
ENDIF
I = IT(I)
IF(IT(I) .NE. MP1) GOTO 3225
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C    DETERMINING PI (REFINEMENT POTENTIAL) ARRAY
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DO 3240 I = 1,NOD
    PI(I) = 0
3240 CONTINUE
DO 3245 I = 1,NODESR
    IF(NSETPT(I+1) .LE. NSETPT(I) + 1) THEN
        PI(I) = 1000000
        GOTO 3245
    ENDIF
    IF(IABS(P(I)) .EQ. (NODESR+1)) GOTO 3250
    IF(P(I) .GT. 0) THEN
        IF(IABS(NUMARC(IABS(AALOC(P(I),I)))) .GT. 1)
C            PI(I)=PI(I)+1
    ELSE
        IF(IABS(NUMARC(IABS(AALOC(I,IABS(P(I))))) .GT.
1)
C            PI(I)=PI(I)+1
    ENDIF
3250    KK=LLINK(I)
3255    IF(KK .EQ. 0) GOTO 3245
    IF(P(KK) .GT. 0) THEN
        IF(IABS(NUMARC(IABS(AALOC(I,KK)))) .GT. 1)
C            PI(I)=PI(I)+1
    ELSE
        IF(IABS(NUMARC(IABS(AALOC(KK,I)))) .GT. 1)
C            PI(I)=PI(I)+1
    ENDIF
    KK=RLINK(KK)
    IF(KK .NE. 0) GOTO 3255
3245 CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C    DETERMINING SUBSET TO BE REFINED
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
3260 MINPI = 1000000
    ISTAR = 0
DO 3270 I = 1,NODESR
    IF(PI(I) .LT. MINPI) THEN
        MINPI = PI(I)
        ISTAR = I
    ENDIF
3270 CONTINUE
WRITE(OUTPUT,6001)

```

```

6001 FORMAT(1H0,'PARTITION REFINEMENT SEGMENT...')
      WRITE(OUTPUT,6002) (INDEX(I),I=1,NODESR)
6002 FORMAT(1H0,'NODE SUBSET:      ',30(I3,1X))
      WRITE(OUTPUT,6003) (PI(I), I=1,NODESR)
6003 FORMAT(1H ,'POTENTIAL (PI): ',30(I3,1X))
      WRITE(6,3275) MINPI,ISTAR
3275 FORMAT(1H ,'MIN. PI = ',I3,2X,'SUBSET ',I3,' WILL BE
REFINED.')
```

C

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  REFINES SUBSET ISTAR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      I1 = (NSETPT(ISTAR+1) - NSETPT(ISTAR))/2
      DO 3280 I = NSETPT(ISTAR)+I1,NSETPT(ISTAR+1)-1
          K(NTIL(I)) = NODESR+1
3280 CONTINUE
      WRITE(OUTPUT,6004)
6004 FORMAT(1H0,'NODES SET PARTITION N(R+1) AFTER
REFINEMENT...')
```

```

      WRITE(OUTPUT, 3281) (INDEX(I),I=1,MO)
3281 FORMAT(1H0,'NODE:      ',30(I3,1X))
      WRITE(6,3285) (K(I),I=1,MO)
3285 FORMAT(1H ,'SUBSET: ',30(I3,1X))
```

C

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  LOAD AALOCO(),NUMAO(),ACPO(),SOLNO()
      DO 3286 I=1,NODESR
          DO 3287 J=1,NODESR
              AALOCO(I,J) = IABS(AALOC(I,J))
              AALOC(I,J) = 0
3287 CONTINUE
3286 CONTINUE
      DO 3288 I = 1,NAGSET
          NUMAO(I) = NUMARC(I)
          NUMARC(I) = 0
          ACPO(I) = ACP(I)
          ACP(I) = 0
          SOLNO(I) = SOLN(I)
          SOLN(I) = 0
3288 CONTINUE
      NPSOLV = 0
      GOTO 69
3995 STOP
4000 WRITE(OUTPUT,4010)
4010 FORMAT( 25H CANDIDATE QUEUE OVERFLOW )
      STOP
      END
```

APPENDIX D-VIII: THE DISAGGREGATION MAP DM^G

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DISAGGREGATION MAP DMG
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF(R .EQ. 1) GOTO 421
      IF(SWITCH .EQ. 1 .OR. NPSOLV .GT. 0) GOTO 421
      DO 394 I = 1, NAD
        TYPE(I) = 0
394  CONTINUE
      PTR = 1
      ARPTR = 1
      DO 396 I = 1, NODESR
        SNODE(I) = I
396  CONTINUE
C
CCCCCCCCCCCCCCCCCCCC DMG C)
      IF(AALOC(ISTAR, NODESR) .GT. 0) THEN
        SNP(PTR) = AALOC(ISTAR, NODESR)
        AALOC(ISTAR, NODESR) = -AALOC(ISTAR, NODESR)
        SOLN(AALOC(ISTAR, NODESR)) = 0
        NUMARC(AALOC(ISTAR, NODESR)) =
C          IABS(NUMARC(AALOC(ISTAR, NODESR)))
        ACP(AALOC(ISTAR, NODESR)) =
C          IABS(ACP(AALOC(ISTAR, NODESR)))
        TYPE(IABS(AALOC(ISTAR, NODESR))) = 3
        IF(SNODE(ISTAR) .GT. 0) SNODE(ISTAR) =
-SNODE(ISTAR)
        IF(SNODE(NODESR) .GT. 0) SNODE(NODESR) =
-SNODE(NODESR)
        PTR = PTR + 1
      ENDIF
C
CCCCCCCCCCCCCCCCCCCC DMG C)
      IF(AALOC(NODESR, ISTAR) .GT. 0) THEN
        SNP(PTR) = AALOC(NODESR, ISTAR)
        AALOC(NODESR, ISTAR) = -AALOC(NODESR, ISTAR)
        SOLN(AALOC(NODESR, ISTAR)) = 0
        NUMARC(AALOC(NODESR, ISTAR)) =
C          IABS(NUMARC(AALOC(NODESR, ISTAR)))
        ACP(AALOC(NODESR, ISTAR)) =
C          IABS(ACP(AALOC(NODESR, ISTAR)))
        TYPE(IABS(AALOC(NODESR, ISTAR))) = 3
        IF(SNODE(NODESR) .GT. 0) SNODE(NODESR) =
-SNODE(NODESR)
        IF(SNODE(ISTAR) .GT. 0) SNODE(ISTAR) =
-SNODE(ISTAR)
        PTR = PTR + 1
      ENDIF
C
      DO 398 I = 1, NODESR-1

```

```

      IF(I .EQ. ISTAR) GOTO 398
C
      IF(AALOC(ISTAR,I) .GT. 0 .AND. AALOC(NODESR,I) .GT.
0) THEN
CCCCCCCCCCCCCCCCC DMG C)
      IF(NUMAO(AALOC(ISTAR,I)) .LT. 0) THEN
        SNP(PTR) = AALOC(ISTAR,I)
        AALOC(ISTAR,I) = - AALOC(ISTAR,I)
        SOLN(AALOC(ISTAR,I)) = 0
        NUMARC(AALOC(ISTAR,I)) =
C          IABS(NUMARC(AALOC(ISTAR,I)))
        ACP(AALOC(ISTAR,I)) =
C          IABS(NUMARC(AALOC(ISTAR,I)))
        TYPE(IABS(AALOC(ISTAR,I))) = 3
        PTR = PTR + 1
        SNP(PTR) = AALOC(NODESR,I)
        AALOC(NODESR,I) = -AALOC(NODESR,I)
        SOLN(AALOC(NODESR,I)) = 0
        NUMARC(AALOC(NODESR,I)) =
C          IABS(NUMARC(AALOC(NODESR,I)))
        ACP(AALOC(NODESR,I)) =
C          IABS(ACP(AALOC(NODESR,I)))
        TYPE(IABS(AALOC(NODESR,I))) = 3
        PTR = PTR + 1
        IF(SNODE(ISTAR) .GT. 0) SNODE(ISTAR) = -
SNODE(ISTAR)
        IF(SNODE(I) .GT. 0) SNODE(I) = -SNODE(I)
        IF(SNODE(NODESR) .GT. 0) SNODE(NODESR) =
-SNODE(NODESR)
      ENDIF
C
CCCCCCCCCCCCCCCCC DMG B)
      IF(ACPO(AALOC(ISTAR,I)) .LT. 0) THEN
        SOLN(AALOC(ISTAR,I)) = ACP(AALOC(ISTAR,I))
        ACP(AALOC(ISTAR,I)) =
-IABS(ACP(AALOC(ISTAR,I)))
        TYPE(AALOC(ISTAR,I)) = 2
        SOLN(AALOC(NODESR,I)) = ACP(AALOC(NODESR,I))
        ACP(AALOC(NODESR,I)) =
-IABS(ACP(AALOC(NODESR,I)))
        TYPE(AALOC(NODESR,I)) = 2
      ENDIF
C
CCCCCCCCCCCCCCCCC DMG A)
      IF(NUMAO(AALOC(ISTAR,I)) .GT. 0 .AND.
C      ACPO(AALOC(ISTAR,I)) .GT. 0) THEN
        SOLN(AALOC(ISTAR,I)) = 0
        NUMARC(AALOC(ISTAR,I)) =
IABS(NUMARC(AALOC(ISTAR,I)))
        ACP(AALOC(ISTAR,I)) =
IABS(ACP(AALOC(ISTAR,I)))
        TYPE(AALOC(ISTAR,I)) = 1

```



```

        SOLN(AALOC(NODESR,I)) = 0
        NUMARC(AALOC(NODESR,I)) =
IABS(NUMARC(AALOC(NODESR,I)))
        ACP(AALOC(NODESR,I)) =
IABS(ACP(AALOC(NODESR,I)))
        TYPE(AALOC(NODESR,I)) = 1
    ENDIF
ENDIF
C
CCCCCCCCCCCCCCCC UNAFFECTED ARC
    IF(AALOC(ISTAR,I) .GT. 0 .AND. AALOC(NODESR,I) .EQ. 0)
THEN
        SOLN(AALOC(ISTAR,I)) = SOLNO(AALOCO(ISTAR,I))
        IF(ACPO(AALOCO(ISTAR,I)) .LT. 0)
C            ACP(AALOC(ISTAR,I)) = -IABS(ACP(AALOC(ISTAR,I)))
            IF(NUMAO(AALOCO(ISTAR,I)) .LT. 0)
C                NUMARC(AALOC(ISTAR,I)) =
-IABS(NUMARC(AALOC(ISTAR,I)))
            ENDIF
C
CCCCCCCCCCCCCCCC REDESIGNATION OF UNAFFECTED ARC
    IF(AALOC(NODESR,I) .GT. 0 .AND. AALOC(ISTAR,I) .EQ. 0)
THEN
        SOLN(AALOC(NODESR,I)) = SOLNO(AALOCO(ISTAR,I))
        IF(ACPO(AALOCO(ISTAR,I)) .LT. 0) THEN
            ACP(AALOC(NODESR,I)) =
-IABS(ACP(AALOC(NODESR,I)))
        ENDIF
        IF(NUMAO(AALOCO(ISTAR,I)) .LT. 0) THEN
            NUMARC(AALOC(NODESR,I)) =
-IABS(NUMARC(AALOC(NODESR,I)))
            IF(IABS(PO(ISTAR)) .EQ. 1) DMGDI = 1
        ENDIF
    ENDIF
C
CCCCCCCCCCCCCCCC SWAP COLUMN TO ROW SEARCH
C
    IF(AALOC(I,ISTAR) .GT. 0 .AND. AALOC(I,NODESR) .GT.
0) THEN
CCCCCCCCCCCCCCCC DMG C)
        IF(NUMAO(AALOCO(I,ISTAR)) .LT. 0) THEN
            SNP(PTR) = AALOC(I,ISTAR)
            AALOC(I,ISTAR) = - AALOC(I,ISTAR)
            SOLN(AALOC(I,ISTAR)) = 0
            NUMARC(AALOC(I,ISTAR)) =
C                IABS(NUMARC(AALOC(I,ISTAR)))
            ACP(AALOC(I,ISTAR)) =
C                IABS(NUMARC(AALOC(I,ISTAR)))
            TYPE(IABS(AALOC(I,ISTAR))) = 3
            PTR = PTR + 1
            SNP(PTR) = AALOC(I,NODESR)
            AALOC(I,NODESR) = -AALOC(I,NODESR)

```

```

        SOLN(AALOC(I,NODESR)) = 0
        NUMARC(AALOC(I,NODESR)) =
C          IABS(NUMARC(AALOC(I,NODESR)))
        ACP(AALOC(I,NODESR)) =
C          IABS(ACP(AALOC(I,NODESR)))
        TYPE(IABS(AALOC(I,NODESR))) = 3
        PTR = PTR + 1
        IF(SNODE(I) .GT. 0) SNODE(I) = - SNODE(I)
        IF(SNODE(ISTAR) .GT. 0) SNODE(ISTAR) =
-SNODE(ISTAR)
        IF(SNODE(NODESR) .GT. 0) SNODE(NODESR) =
-SNODE(NODESR)
        ENDIF
C
CCCCCCCCCCCCCCCCCCCC DMG B)
        IF(ACPO(AALOC(I,ISTAR)) .LT. 0) THEN
            SOLN(AALOC(I,ISTAR)) = ACP(AALOC(I,ISTAR))
            ACP(AALOC(I,ISTAR)) =
-IABS(ACP(AALOC(I,ISTAR)))
            TYPE(AALOC(I,ISTAR)) = 2
            SOLN(AALOC(I,NODESR)) = ACP(AALOC(I,NODESR))
            ACP(AALOC(I,NODESR)) =
-IABS(ACP(AALOC(I,NODESR)))
            TYPE(AALOC(I,NODESR)) = 2
        ENDIF
C
CCCCCCCCCCCCCCCCCCCC DMG A)
        IF(NUMAO(AALOC(I,ISTAR)) .GT. 0 .AND.
C          ACPO(AALOC(I,ISTAR)) .GT. 0) THEN
            SOLN(AALOC(I,ISTAR)) = 0
            ACP(AALOC(I,ISTAR)) =
IABS(ACP(AALOC(ISTAR,I)))
            NUMARC(AALOC(I,ISTAR)) =
IABS(NUMARC(AALOC(I,ISTAR)))
            TYPE(AALOC(I,ISTAR)) = 1
            SOLN(AALOC(I,NODESR)) = 0
            ACP(AALOC(I,NODESR)) =
IABS(ACP(AALOC(I,NODESR)))
            NUMARC(AALOC(I,NODESR)) =
IABS(NUMARC(AALOC(I,NODESR)))
            TYPE(AALOC(I,NODESR)) = 1
        ENDIF
    ENDIF
C
CCCCCCCCCCCCCCCCCCCC UNAFECTED ARC
        IF(AALOC(I,ISTAR) .GT. 0 .AND. AALOC(I,NODESR) .EQ. 0)
THEN
            SOLN(AALOC(I,ISTAR)) = SOLNO(AALOC(I,ISTAR))
            IF(ACPO(AALOC(I,ISTAR)) .LT. 0)
C              ACP(AALOC(I,ISTAR)) =
-IABS(ACP(AALOC(I,ISTAR)))
            IF(NUMAO(AALOC(I,ISTAR)) .LT. 0)

```

```

      C          NUMARC(AALOC(I,ISTAR)) =
-IABS(NUMARC(AALOC(I,ISTAR)))
      ENDIF
CCCCCCCCCCCCCCCCCCCC REDESIGNATION OF UNAFFECTED ARC
      IF(AALOC(I,NODESR) .GT. 0 .AND. AALOC(I,ISTAR) .EQ. 0)
THEN
      SOLN(AALOC(I,NODESR)) = SOLNO(AALOC(I,ISTAR))
      IF(ACPO(AALOC(I,ISTAR)) .LT. 0) THEN
      ACP(AALOC(I,NODESR)) =
-IABS(ACP(AALOC(I,NODESR)))
      ENDIF
      IF(NUMAO(AALOC(I,ISTAR)) .LT. 0) THEN
      NUMARC(AALOC(I,NODESR)) =
-IABS(NUMARC(AALOC(I,NODESR)))
      IF(IABS(PO(ISTAR)) .EQ. I) DMGDI = I
      ENDIF
      ENDIF
C
CCCCCCCCCCCCCCCCCCCC OTHER ARCS NOT IN ISTAR OR NODESR
ROW/COLUMN
      DO 397 J=1,NODESR-1
      IF((J .EQ. ISTAR) .OR. (J .EQ. NODESR) .OR.
      C      (AALOC(I,J) .EQ. 0)) GOTO 397
      SOLN(AALOC(I,J)) = SOLNO(AALOC(I,J))
      IF(NUMAO(AALOC(I,J)) .LT. 0)
      C      NUMARC(AALOC(I,J)) =
-IABS(NUMARC(AALOC(I,J)))
      IF(ACPO(AALOC(I,J)) .LT. 0)
      C      ACP(AALOC(I,J)) = -IABS(ACP(AALOC(I,J)))
397      CONTINUE
398      CONTINUE
      SNAGSE = PTR - 1
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      DMG D): SNP IS DEGENERATE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF(SNAGSE .EQ. 0) THEN
      WRITE(OUTPUT,429) R
      429      FORMAT(1H0,'SNP(' ,I3,' ) DEGENERATE.  DMG D)
ACTIVATED.')
      IF(IABS(PO(ISTAR)) .EQ. DMGDI) J = ISTAR
      IF(IABS(PO(DMGDI)) .EQ. ISTAR) J = NODESR
      SNODE(J) = -SNODE(J)
C
      J1 = LLINK(ISTAR)
      J2 = RLINK(J1)
      441      IF(J1 .EQ. 0) GOTO 434
      DEPTH = D(J1)
      IF(PO(J1) .LT. 0 .AND. AALOC(J1,J) .EQ. 0) GOTO 443
      IF(PO(J1) .GT. 0 .AND. AALOC(J,J1) .EQ. 0) GOTO 443
      442      SNODE(J1) = - SNODE(J1)
      J1 = IT(J1)

```

```

                IF(D(J1) .GT. DEPTH .AND. J1 .NE. NODESR) GOTO
442
443      J1 = J2
          J2 = RLINK(J2)
          GOTO 441
C
434      DO 436 I = 1,NODESR
          DO 437 J = 1,NODESR
            IF(AALOC(I,J) .EQ. 0) GOTO 437
            IF(SNODE(I)*SNODE(J) .GE. 0) GOTO 437
            IF(NUMARC(AALOC(I,J)) .LE. 0) GOTO 437
            NUMARC(AALOC(I,J)) = -NUMARC(AALOC(I,J))
            TYPE(AALOC(I,J)) = 4
            IF(ACP(AALOC(I,J)) .LT. 0) THEN
              ACP(AALOC(I,J)) = -ACP(AALOC(I,J))
              WRITE(OUTPUT,438) I,J
438              FORMAT(1H0,'OOB ARC (' ,I3,1X,I3,
C                  ') REDESIGNATED AS BASIC.')
              GOTO 427
            ELSE
              WRITE(OUTPUT,439) I,J
439              FORMAT(1H0,'NONBASIC ARC (' ,I3,1X,I3,
C                  ') REDESIGNATED AS BASIC.')
              GOTO 427
            ENDIF
          CONTINUE
437      CONTINUE
436      CONTINUE
          ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC
427 WRITE(OUTPUT,401) (NUMARC(I), I= 1,NAGSET)
401 FORMAT(1H0,'BEGIN DISAGGREGATION...' ,/,1X,
C      'RESULTS AFTER DMG A), B), D) AND
REDESIGNATION;' ,/,1X,
C      'NUMARC: ' ,30(I3,1X))
      WRITE(OUTPUT,402) (ACP(I), I=1,NAGSET)
402 FORMAT(1H , 'ACP:      ' ,30(I3,1X))
      WRITE(OUTPUT,403) (SOLN(I), I=1,NAGSET)
403 FORMAT(1H , 'SOLN:      ' ,30(I3,1X))
      WRITE(OUTPUT,8818) (TYPE(I), I=1,NAGSET)
8818 FORMAT(1H , 'TYPE:      ' ,30(I3,1X))
      WRITE(OUTPUT,404) (AALOC(I,J), J=1,NODESR)
404 FORMAT(1H0,'AALOC = ' ,30(I3,1X))
      DO 405 I = 2,NODESR
        WRITE(OUTPUT,406) (AALOC(I,J), J=1,NODESR)
406      FORMAT(1H ,8X,30(I3,1X))
405      CONTINUE
        IF(SNAGSE .EQ. 0) GOTO 3202
        WRITE(OUTPUT,407)
407 FORMAT(1H0,'DMG C): CONSTRUCTION OF SUBPROBLEM
SNP(R+1)...' ,/,1X,

```

```

      C ' SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF
AALOC ABOVE.')
C
      IF(SNAGSE .EQ. 0) THEN
        GOTO 3202
      ENDIF

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   SET COR(), CORIN()
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 413 I = 1, NODESR
        COR(I) = 0
        CORIN(I) = 0
413  CONTINUE
      J = 1
      DO 408 I = 1, NODESR
        IF(SNODE(I) .GT. 0) GOTO 408
        COR(I) = J
        CORIN(J) = I
        J = J + 1
408  CONTINUE
      SM = J - 1

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   CREATE SNP AND STORE IN SAT(), SAH(), SCMIN(),
C       SACP(), SALB(), SSOLN()
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 414 I = 1, SNAGSE
        SAT(I) = COR(AT(SNP(I)))
        SAH(I) = COR(AH(SNP(I)))
        SCMIN(I) = CMIN(SNP(I))
        SACP(I) = IABS(ACP(SNP(I)))
        SALB(I) = ALB(SNP(I))
        SSOLN(I) = 0
414  CONTINUE

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   ADJUST SUPPLIES
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 416 I = 1, SM
        ADSUP(I) = ASUP(CORIN(I))
416  CONTINUE
      DO 417 I = 1, SM
        DO 418 J = 1, NODESR
          IF(CORIN(I) .EQ. J) GOTO 418
          IF(AALOC(CORIN(I), J) .GT. 0)
C             ADSUP(I) = ADSUP(I) - SOLN(AALOC(CORIN(I), J))
          IF(AALOC(J, CORIN(I)) .GT. 0)
C             ADSUP(I) = ADSUP(I) + SOLN(AALOC(J, CORIN(I)))
418  CONTINUE
417  CONTINUE
C

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   CONSTRUCTION OF SUPER SOURCE/SINK ARCS
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DO 419 I = 1, SM
    IF(ADSUP(I) .EQ. 0) GOTO 419
    IF(ADSUP(I) .GT. 0) THEN
        SAT(SNAGSE+I) = SM + 1
        SAH(SNAGSE+I) = I
        SCMIN(SNAGSE+I) = 0
        SACP(SNAGSE+I) = ADSUP(I)
        SALB(SNAGSE+I) = ADSUP(I)
    ELSE
        SAT(SNAGSE+I) = I
        SAH(SNAGSE+I) = SM+2
        SCMIN(SNAGSE+I) = 0
        SACP(SNAGSE+I) = -ADSUP(I)
        SALB(SNAGSE+I) = -ADSUP(I)
    ENDIF
419 CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   SET SWITCH: SWITCH = 1 SOLVE SNP
C   SWITCH = 0 SOLVE NPR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    SWITCH = 1
C
    GOTO 152
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   BALANCE DISAGGREGATED SOLUTION IF NECESSARY
C
3202 IF(SNAGSE .GT. 0) GOTO 3207
    SUPBAL = ASUP(NODESR)
    DO 3203 I = 1, NODESR-1
        IF(AALOC(I, NODESR) .NE. 0)
C            SUPBAL = SUPBAL + SOLN(AALOC(I, NODESR))
        IF(AALOC(NODESR, I) .NE. 0)
C            SUPBAL = SUPBAL - SOLN(AALOC(NODESR, I))
3203 CONTINUE
    IF(SUPBAL .EQ. 0) GOTO 3207
    IF(SUPBAL .GT. 0) THEN
        WRITE(OUTPUT, 3204) NODESR, ISTAR, SUPBAL
3204   FORMAT(1H0, 'SUPPLY NOT BALANCED AFTER
DISAGGREGATION.',
C       /, 1X, 'ARTIFICIAL OOB ARCS (' , I3, ', ', ROOT) AND (ROOT
, ', I3,
C       ') REQUIRED WITH FLOW = ', I5)
        ART(ARPTR) = NODESR
        ARH(ARPTR) = NODESR + 1
        ART(ARPTR+1) = NODESR + 1
        ARH(ARPTR+1) = ISTAR
    ELSE

```

```

        SUPBAL = IABS(SUPBAL)
        WRITE(OUTPUT,3206) ISTAR,NODESR,SUPBAL
3206    FORMAT(1H0,'SUPPLY NOT BALANCED AFTER
DISAGGREGATION.',
C      /,1X,'ARTIFICIAL OOB ARCS (' ,I3,' , ROOT) AND (ROOT
', ,I3,
C      ') REQUIRED WITH FLOW = ' ,I5)
        ART(ARPTR) = ISTAR
        ARH(ARPTR) = NODESR + 1
        ART(ARPTR+1) = NODESR + 1
        ARH(ARPTR+1) = NODESR
    ENDIF
    ARCMIN(ARPTR) = 1000000
    ARCMIN(ARPTR+1) = 1000000
    ARLB(ARPTR) = 0
    ARLB(ARPTR+1) = 0
    ARCP(ARPTR) = 1000000
    ARCP(ARPTR+1) = 1000000
    ARSOLN(ARPTR) = SUPBAL
    ARSOLN(ARPTR+1) = SUPBAL
    ARPTR = ARPTR + 2

```

APPENDIX E: AGNET COMPUTER OUTPUT
FOR EXAMPLE 6.1

```
*****
*****      AGNET      (JUNE 1985)      *****
*****
```

THIS PROGRAM INCORPORATES COPYRIGHTED MATERIAL AND
SHOULD NOT BE REPRODUCED IN WHOLE OR PART
WITHOUT PERMISSION OF THE AUTHOR.

NOTES...

1. THE SEQUENCE OF OPERATIONS ARE AS FOLLOWS:
 - A. READ ORIGINAL PROBLEM (NP(0)) INFORMATION
 - B. CREATE THE FIRST AGGREGATE PROBLEM NP(1)
 - C. SOLVE NP(1)
 - D. REFINES NODE SET PARTITION USING "NODE
POTENTIAL" HEURISTIC
 - E. CREATE THE NEXT AGGREGATE PROBLEM, NP(R+1)
 - F. DISAGGREGATE THE SOLUTION OF NP(R) TO
OBTAIN AN ADVANCED-START BASIC SOLUTION OF
NP(R+1)
 - G. SOLVE THE AGGREGATE PROBLEM NP(R+1)
2. REQUIRED INPUTS ARE THE STANDARD GNET INFORMATION
AND THE INITIAL NODE SET PARTITION.
3. THIS VERSION DOES NOT SOLVE NP(R+1)
BEGINNING WITH THE ADVANCED DISAGGREGATED SOLUTION.
THIS PROGRAM IS INTENDED ONLY TO TEST AND VERIFY
THEORETICAL RESULTS OF THE AUTHORS DISSERTATION.

```
*****
***** ORIGINAL PROBLEM NP(0) DATA *****
*****
```

PRINT FLAG SET AT 2 NUMBER OF ORIGINAL NODES = 12

TAIL	HEAD	COST	UB	LB
2	3	34	11	0
3	4	23	6	0
1	5	28	10	0
2	6	45	25	5
1	7	57	21	0
5	8	24	5	0
1	8	56	7	0
4	8	19	9	0
1	9	61	5	0
2	9	99	12	0
6	9	48	3	0
3	9	53	24	0
3	10	26	8	0

4	10	20	2	0
2	11	14	23	10
6	12	34	16	0

NODE:	1	2	3	4	5	6	7	8	9	10
11 12										
SUPPLY:	34	56	5	0	-5	-9	-18	-15	-8	-3
-21 -16										
SUBSET:	2	1	1	1	2	1	2	3	4	5
1 1										

 ***** AGGREGATE PROBLEM 1 *****

5 NODES

NODE: 1 2 3 4 5
 SUPPLY: 15 11 -15 -8 -3

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	19	9	0
0	2	3	24	12	0
0	1	4	48	39	0
0	2	4	61	5	0
0	1	5	20	10	0
TOTAL SUPPLIES =			26	TOTAL DEMANDS = 26	
INITIAL COST =			0		

AGGREGATION MEASURE...

M1(R) = .64 M2(R) = .44 M3(R) = .25 M(R) = 1.32

***** SOLUTION OF NP(R)) *****

5 ARCS

TIME= 0 SECONDS, PIVOTS= 7, NNE= 32, NNS= 3,
 IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	3	4
2	3	11
1	4	8
1	5	3

COST= 784

DISAGGREGATION COMPLETE...

MARKING OF SOLUTION OF NP(R)...

NUMARC:	0	0	-1	-2	-3	1	-2
ACP:	0	0	9	12	39	5	10
SOLN:	0	0	4	11	8	0	3
TYPE:	0	0	0	0	0	0	0

PARTITION REFINEMENT SEGMENT...

NODE SUBSET: 1 2 3 4 5
 POTENTIAL (PI): 2 1 *** *** ***
 MIN. PI = 1 SUBSET 2 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...
 NODE: 1 2 3 4 5 6 7 8 9 10 11 12
 SUBSET: 2 1 1 1 6 1 6 3 4 5 1 1

 ***** AGGREGATE PROBLEM 2 *****

6 NODES

NODE: 1 2 3 4 5 6
 SUPPLY: 15 34 -15 -8 -3 -23

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	19	9	0
0	2	3	56	7	0
0	6	3	24	5	0
0	1	4	48	39	0
0	2	4	61	5	0
0	1	5	20	10	0
0	2	6	28	31	0

TOTAL SUPPLIES = 49 TOTAL DEMANDS = 49 INITIAL
 COST = 0

AGGREGATION MEASURE...

M1(R) = .55 M2(R) = .31 M3(R) = .25 M(R) = 1.11

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC: 0 -1 1 1 -3 1 -2 2
 ACP: 0 9 7 5 39 5 10 31
 SOLN: 0 4 0 0 8 0 3 0
 TYPE: 0 0 3 3 0 0 0 3

AALOC = 1 0 2 5 7 0
 0 0 -3 6 0 -8
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 -4 0 0 0

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...

SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC
 ABOVE.

***** SUBPROBLEM SNP(R+1) *****

3 NODES

NODE: 1 2 3
 SUPPLY: 34 -11 -23

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	28	31	0
0	1	2	56	7	0
0	3	2	24	5	0

TOTAL SUPPLIES = 34 TOTAL DEMANDS = 34 INITIAL
 COST = 0

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE
 DESIGNATION-NP(R) DESIGNATION:
 GNET DESIGNATION: 1 2 3
 NP(R) DESIGNATION: 2 3 6

3 ARCS
 TIME= 0 SECONDS, PIVOTS= 4, NNE= 32, NNS= 2,
 IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	2	6
3	2	5
1	3	28

COST= 1240

DISAGGREGATION COMPLETE...

NUMARC:	0	-1	-1	1	-3	1	-2	-2
ACP:	0	9	7	-5	39	5	10	31
SOLN:	0	4	6	5	8	0	3	28
TYPE:	0	0	3	3	0	0	0	3

***** SOLUTION OF NP(R)) *****

7 ARCS

TIME= 0 SECONDS, PIVOTS= 8, NNE= 32, NNS= 4,
 IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	3	9
2	3	1
6	3	5
1	4	3
2	4	5
1	5	3
2	6	28

COST= 1640

MARKING OF SOLUTION OF NP(R)...

NUMARC:	0	-1	-1	1	-3	1	-2	-2
ACP:	0	9	7	-5	39	-5	10	31
SOLN:	0	9	1	5	3	5	3	28
TYPE:	0	0	3	3	0	0	0	3

PARTITION REFINEMENT SEGMENT...

NODE SUBSET:	1	2	3	4	5	6

POTENTIAL (PI):	2	***	***	***	***	1

MIN. PI = 1 SUBSET 6 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	1	1	6	1	7	3	4	5	1	1

 ***** AGGREGATE PROBLEM 3 *****

7 NODES

NODE: 1 2 3 4 5 6 7
 SUPPLY: 15 34 -15 -8 -3 -5 -18

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW	BND	
0	1	3	19	9	0	0	
0	2	3	56	7	0	0	
0	6	3	24	5	0	0	
0	1	4	48	39	0	0	
0	2	4	61	5	0	0	
0	1	5	20	10	0	0	
0	2	6	28	10	0	0	
0	2	7	57	21	0	0	
TOTAL SUPPLIES =			49	TOTAL DEMANDS =			49
INITIAL COST =			0				

AGGREGATION MEASURE...

M1(R) = .45 M2(R) = .31 M3(R) = .19 M(R) = .95

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC:	0	-1	-1	1	-3	1	-2	1	1
ACP:	0	9	7	-5	39	-5	10	10	21
SOLN:	0	9	1	5	3	5	3	0	0
TYPE:	0	0	0	0	0	0	0	3	3

AALOC =	1	0	2	5	7	0	0
	0	0	3	6	0	-8	-9
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	4	0	0	0	0
	0	0	0	0	0	0	0

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...

SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC ABOVE.

***** SUBPROBLEM SNP(R+1) *****

3 NODES

NODE: 1 2 3
 SUPPLY: 28 -10 -18

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	2	28	10	0
0	1	3	57	21	0

TOTAL SUPPLIES = 28 TOTAL DEMANDS = 28 INITIAL
COST = 0

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE

DESIGNATION-NP(R) DESIGNATION:

GNET DESIGNATION: 1 2 3

NP(R) DESIGNATION: 2 6 7

2 ARCS

TIME= 0 SECONDS, PIVOTS= 2, NNE= 32, NNS= 2,
IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	2	10
1	3	18

COST= 1306

DISAGGREGATION COMPLETE...

NUMARC:	0	-1	-1	1	-3	1	-2	1	-1
ACP:	0	9	7	-5	39	-5	10	-10	21
SOLN:	0	9	1	5	3	5	3	10	18
TYPE:	0	0	0	0	0	0	0	3	3

***** SOLUTION OF NP(R) *****

8 ARCS

TIME= 0 SECONDS, PIVOTS= 9, NNE= 32, NNS= 5,
IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	3	9
2	3	1
6	3	5
1	4	3
2	4	5
1	5	3
2	6	10
2	7	18

COST= 2162

MARKING OF SOLUTION OF NP(R)...

NUMARC:	0	-1	-1	1	-3	1	-2	-1	-1
ACP:	0	9	7	-5	39	-5	10	10	21
SOLN:	0	9	1	5	3	5	3	10	18
TYPE:	0	0	0	0	0	0	0	3	3

PARTITION REFINEMENT SEGMENT...

NODE SUBSET:	1	2	3	4	5	6	7

POTENTIAL (PI):	2	***	***	***	***	***	***

MIN. PI = 2 SUBSET 1 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	1	1	6	8	7	3	4	5	8	8

 ***** AGGREGATE PROBLEM 4 *****

8 NODES

NODE: 1 2 3 4 5 6 7 8
 SUPPLY: 61 34 -15 -8 -3 -5 -18 -46

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
Ø	1	3	19	9	Ø
Ø	2	3	56	7	Ø
Ø	6	3	24	5	Ø
Ø	1	4	53	36	Ø
Ø	2	4	61	5	Ø
Ø	8	4	48	3	Ø
Ø	1	5	2Ø	1Ø	Ø
Ø	2	6	28	1Ø	Ø
Ø	2	7	57	21	Ø
Ø	1	8	14	48	15

TOTAL SUPPLIES = 95 TOTAL DEMANDS = 95 INITIAL
 COST = 21Ø

AGGREGATION MEASURE...

M1(R) = .36 M2(R) = .19 M3(R) = .19 M(R) = .74

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC:	Ø	-1	-1	1	2	1	1	-2	-1	-1	2	Ø
ACP:	Ø	9	7	-5	36	-5	3	1Ø	1Ø	21	48	Ø
SOLN:	Ø	9	1	5	Ø	5	Ø	3	1Ø	18	Ø	Ø
TYPE:	Ø	Ø	Ø	Ø	3	Ø	3	Ø	Ø	Ø	3	Ø

AALOC =	1	Ø	2	-5	8	Ø	Ø	-11
	Ø	Ø	3	6	Ø	9	1Ø	Ø
	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	Ø	Ø	4	Ø	Ø	Ø	Ø	Ø
	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	Ø	Ø	Ø	-7	Ø	Ø	Ø	12

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...

SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC
 ABOVE.

***** SUBPROBLEM SNP(R+1) *****

AD-A170 681

AGGREGATION OF NETWORK FLOW PROBLEMS(U) AIR FORCE INST
OF TECH WRIGHT-PATTERSON AFB OH V E FRANCIS 1985
AFIT/CI/NR-86-800

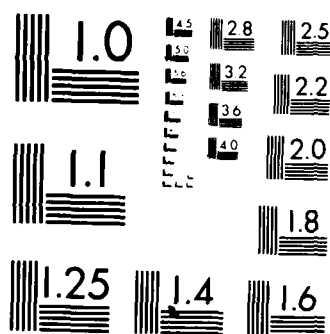
4/4

UNCLASSIFIED

F/G 12/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

3 NODES

NODE: 1 2 3
SUPPLY: 49 -3 -46

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	14	48	15
0	1	2	53	36	0
0	3	2	48	3	0

TOTAL SUPPLIES = 49 TOTAL DEMANDS = 49 INITIAL
COST = 210

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE
DESIGNATION-NP(R) DESIGNATION:

GNET DESIGNATION: 1 2 3
NP(R) DESIGNATION: 1 4 8
3 ARCS

TIME= 0 SECONDS, PIVOTS= 2, NNE= 32, NNS= 2,
IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	2	3
1	3	31

COST= 803

DISAGGREGATION COMPLETE...

NUMARC:	0	-1	-1	1	-2	1	1	-2	-1	-1	-2	0
ACP:	0	9	7	-5	36	-5	3	10	10	21	48	0
SOLN:	0	9	1	5	3	5	0	3	10	18	46	0
TYPE:	0	0	0	0	3	0	3	0	0	0	3	0

***** SOLUTION OF NP(R) *****

10 ARCS

TIME= 0 SECONDS, PIVOTS= 10, NNE= 32, NNS= 6,
IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	3	9
2	3	1
6	3	5
1	4	3

2	4	5
1	5	3
2	6	10
2	7	18
1	8	31

COST= 2821

MARKING OF SOLUTION OF NP(R)...

NUMARC:	0	-1	-1	1	-2	1	1	-2	-1	-1	-2	0
ACP:	0	9	7	-5	36	-5	3	10	10	21	48	0
SOLN:	0	9	1	5	3	5	0	3	10	18	46	0
TYPE:	0	0	0	0	3	0	3	0	0	0	3	0

PARTITION REFINEMENT SEGMENT...

NODE SUBSET:	1	2	3	4	5	6	7	8
POTENTIAL (PI):	3	***	***	***	***	***	***	1

MIN. PI = 1 SUBSET 8 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	1	1	6	8	7	3	4	5	9	9

 ***** AGGREGATE PROBLEM 5 *****

9 NODES

NODE: 1 2 3 4 5 6 7 8 9
 SUPPLY: 61 34 -15 -8 -3 -5 -18 -9 -37

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	19	9	0
0	2	3	56	7	0
0	6	3	24	5	0
0	1	4	53	36	0
0	2	4	61	5	0
0	8	4	48	3	0
0	1	5	20	10	0
0	2	6	28	10	0
0	2	7	57	21	0
0	1	8	45	25	5
0	1	9	14	23	10
0	8	9	34	16	0

TOTAL SUPPLIES = 95 TOTAL DEMANDS = 95 INITIAL
 COST = 365

AGGREGATION MEASURE...

M1(R) = .27 M2(R) = .13 M3(R) = .13 M(R) = .52

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC:	0	-1	-1	1	-2	1	1	-2	-1	-1	1	1	1
ACP:	0	9	7	-5	36	-5	3	10	10	21	25	23	16
SOLN:	0	9	1	5	3	5	0	3	10	18	0	0	0
TYPE:	0	0	0	0	0	0	0	0	0	0	3	3	3

AALOC =	1	0	2	5	8	0	0	-11	-12
0	0	3	6	0	9	10	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	7	0	0	0	0	-13	0
0	0	0	0	0	0	0	0	0	0

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...
 SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC
 ABOVE.

***** SUBPROBLEM SNP(R+1) *****

3 NODES

NODE: 1 2 3
 SUPPLY: 46 -9 -37

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
Ø	2	3	34	16	Ø
Ø	1	2	45	25	5
Ø	1	3	14	23	1Ø

TOTAL SUPPLIES = 46 TOTAL DEMANDS = 46 INITIAL
 COST = 365

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE
 DESIGNATION-NP(R) DESIGNATION:
 GNET DESIGNATION: 1 2 3
 NP(R) DESIGNATION: 1 8 9

3 ARCS

TIME= Ø SECONDS, PIVOTS= 3, NNE= 32, NNS= 2,
 IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	2	18
2	3	14
1	3	13

COST= 1833

DISAGGREGATION COMPLETE...

NUMAPC:	Ø	-1	-1	1	-2	1	1	-2	-1	-1	-1	1	-1
ACP:	Ø	9	7	-5	36	-5	3	1Ø	1Ø	21	25	-23	16
SOLN:	Ø	9	1	5	3	5	Ø	3	1Ø	18	23	23	14
TYPE:	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	3	3	3

***** SOLUTION OF NP(R)) *****

12 ARCS

TIME= 0 SECONDS, PIVOTS= 12, NNE= 32, NNS= 6,
IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
1	3	9
2	3	1
6	3	5
1	4	3
2	4	5
1	5	3
2	6	10
2	7	18
1	8	18
1	9	13
8	9	14

COST= 4010

MARKING OF SOLUTION OF NP(R)...

NUMARC:	0	-1	-1	1	-2	1	1	-2	-1	-1	-1	1	-1
ACP:	0	9	7	-5	36	-5	3	10	10	21	25	-23	16
SOLN:	0	9	1	5	3	5	0	3	10	18	23	23	14
TYPE:	0	0	0	0	0	0	0	0	0	0	3	3	3

PARTITION REFINEMENT SEGMENT...

NODE SUBSET:	1	2	3	4	5	6	7	8	9
POTENTIAL (PI):	2	***	***	***	***	***	***	***	0

MIN. PI = 0 SUBSET 9 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	1	1	6	8	7	3	4	5	9	10

 ***** AGGREGATE PROBLEM 6 *****

10 NODES

NODE: 1 2 3 4 5 6 7 8 9 10
 SUPPLY: 61 34 -15 -8 -3 -5 -18 -9 -21 -16

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW	BND
0	1	3	19	9		0
0	2	3	56	7		0
0	6	3	24	5		0
0	1	4	53	36		0
0	2	4	61	5		0
0	8	4	48	3		0
0	1	5	20	10		0
0	2	6	28	10		0
0	2	7	57	21		0
0	1	8	45	25		5
0	1	9	14	23		10
0	8	10	34	16		0

TOTAL SUPPLIES = 95 TOTAL DEMANDS = 95 INITIAL
 COST = 365

AGGREGATION MEASURE...

M1(R) = .18 M2(R) = .13 M3(R) = .13 M(R) = .43

SNP(6) DEGENERATE. DMG D) ACTIVATED.
 OOB ARC (1 9) REDESIGNATED AS BASIC.

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC: 0 -1 -1 1 -2 1 1 -2 -1 -1 -1 -1 -1
 ACP: 0 9 7 -5 36 -5 3 10 10 21 25 23 16
 SOLN: 0 9 1 5 3 5 0 3 10 18 23 23 14
 TYPE: 0 0 0 0 0 0 0 0 0 0 0 0 4 0

AALOC = 1 0 2 5 8 0 0 11 12 0
 0 0 3 6 0 9 10 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 4 0 0 0 0 0 0 0

0	0	0	0	0	0	0	0	0	0
0	0	0	7	0	0	0	0	0	13
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

SUPPLY NOT BALANCED AFTER DISAGGREGATION.
 ARTIFICIAL OOB ARCS (9, ROOT) AND (ROOT , 10) REQUIRED
 WITH FLOW = 2

DISAGGREGATION COMPLETE...

MARKING OF SOLUTION OF NP(R)...

NUMARC: 0 -1 -1 1 -2 1 1 -2 -1 -1 -1 -1 -1

ACP: 0 9 7 -5 36 -5 3 10 10 21 25 23 16

SOLN: 0 9 1 5 3 5 0 3 10 18 23 23 14

TYPE: 0 0 0 0 0 0 0 0 0 0 0 4 0

ARTIFICIAL ARCS REQUIRED:

ART: 9 11

ARH: 11 10

ARCMIN: *** **

ARLB: 0 0

ARCP: *** **

ARSOLN: 2 2

***** SOLUTION OF NP(R) *****

12 ARCS

0TIME= 0 SECONDS, PIVOTS= 12, NNE= 32, NNS= 7,

IPG= 2

0FINAL SOLUTION...

FROM	TO	FLOW
1	3	9
2	3	1
6	3	5
1	4	3
2	4	5
1	5	3
2	6	10
2	7	18
1	8	20
1	9	11
8	10	16

COST= 4140

DISAGGREGATION COMPLETE...

MARKING OF SOLUTION OF NP(R)...

NUMARC: 0 -1 -1 1 -2 1 1 -2 -1 -1 1 -1 -1

ACP:	0	9	7	-5	36	-5	3	10	10	21	-25	23	16
SOLN:	0	9	1	5	3	5	0	3	10	18	25	21	16
TYPE:	0	0	0	0	0	0	0	0	0	0	0	4	0

PARTITION REFINEMENT SEGMENT...

NODE SUBSET:	1	2	3	4	5	6	7	8	9	10
POTENTIAL (PI):	2	***	***	***	***	***	***	***	***	***
MIN. PI =	2	SUBSET 1 WILL BE REFINED.								

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	11	11	6	8	7	3	4	5	9	10

 ***** AGGREGATE PROBLEM 7 *****

11 NODES

NODE: 1 2 3 4 5 6 7 8 9 10 11
 SUPPLY: 56 34 -15 -8 -3 -5 -18 -9 -21 -16 5

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	2	3	56	7	0
0	6	3	24	5	0
0	11	3	19	9	0
0	1	4	99	12	0
0	2	4	61	5	0
0	8	4	48	3	0
0	11	4	53	24	0
0	11	5	20	10	0
0	2	6	28	10	0
0	2	7	57	21	0
0	1	8	45	25	5
0	1	9	14	23	10
0	8	10	34	16	0
0	1	11	34	11	0

TOTAL SUPPLIES = 95 TOTAL DEMANDS = 95 INITIAL
 COST = 365

AGGREGATION MEASURE...

M1(R) = .09 M2(R) = .06 M3(R) = .06 M(R) = .22

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC: -1 1 -1 1 1 1 1 -2 -1 -1 1 -1 -1
 1 0
 ACP: 7 -5 9 12 -5 3 24 10 10 21 -25 23 16
 11 0
 SOLN: 1 5 9 0 5 0 0 3 10 18 25 21 16
 0 0
 TYPE: 0 0 0 3 0 0 3 0 0 0 0 0 0
 3 0

AALOC = 0 0 0 -4 0 0 0 11 12 0 -14
 0 0 1 5 0 9 10 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
 0 0 2 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0

0	0	0	6	0	0	0	0	0	13	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	3	-7	8	0	0	0	0	0	15

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...

SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC ABOVE.

***** SUBPROBLEM SNP(R+1) *****

3 NODES

NODE: 1 2 3

SUPPLY: 10 -3 -7

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	1	3	34	11	0
0	1	2	99	12	0
0	3	2	53	24	0

TOTAL SUPPLIES = 10 TOTAL DEMANDS = 10 INITIAL COST = 0

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE DESIGNATION-NP(R) DESIGNATION:

GNET DESIGNATION: 1 2 3

NP(R) DESIGNATION: 1 4 11

3 ARCS

TIME= 0 SECONDS, PIVOTS= 3, NNE= 32, NNS= 2, IPG= 1

FINAL SOLUTION...

FROM	TO	FLOW
3	2	3
1	3	10

COST= 499

DISAGGREGATION COMPLETE...

NUMARC:	-1	1	-1	1	1	1	-1	-2	-1	-1	1	-1	-1
-1	0												
ACP:	7	-5	9	12	-5	3	24	10	10	21	-25	23	16
11	0												
SOLN:	1	5	9	0	5	0	3	3	10	18	25	21	16
10	0												
TYPE:	0	0	0	3	0	0	3	0	0	0	0	0	0
3	0												

***** SOLUTION OF NP(R) *****

14 ARCS

TIME= 0 SECONDS, PIVOTS= 13, NNE= 32, NNS= 8,
IPG= 2

FINAL SOLUTION...

FROM	TO	FLOW
2	3	1
6	3	5
11	3	9
2	4	5
11	4	3
11	5	3
2	6	10
2	7	18
1	8	20
1	9	11
8	10	16
1	11	10

COST= 4480

MARKING OF SOLUTION OF NP(R)...

NUMARC:	-1	1	-1	1	1	1	-1	-2	-1	-1	-1	-1	1
-1	0												
ACP:	7	-5	9	12	-5	3	24	10	10	21	25	23	-16
11	0												
SOLN:	1	5	9	0	5	0	3	3	10	18	25	21	16
10	0												
TYPE:	0	0	0	3	0	0	3	0	0	0	0	0	0
3	0												

PARTITION REFINEMENT SEGMENT...

NODE SUBSET: 1 2 3 4 5 6 7 8 9 10 11

POTENTIAL (PI): *** ** 1

MIN. PI = 1 SUBSET 11 WILL BE REFINED.

NODES SET PARTITION N(R+1) AFTER REFINEMENT...

NODE:	1	2	3	4	5	6	7	8	9	10	11	12
SUBSET:	2	1	11	12	6	8	7	3	4	5	9	10

 ***** AGGREGATE PROBLEM 8 *****

12 NODES

NODE: 1 2 3 4 5 6 7 8 9 10 11 12
 SUPPLY: 56 34 -15 -8 -3 -5 -18 -9 -21 -16 5 0

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	2	3	56	7	0
0	6	3	24	5	0
0	12	3	19	9	0
0	1	4	99	12	0
0	2	4	61	5	0
0	8	4	48	3	0
0	11	4	53	24	0
0	11	5	26	8	0
0	12	5	20	2	0
0	2	6	28	10	0
0	2	7	57	21	0
0	1	8	45	25	5
0	1	9	14	23	10
0	8	10	34	16	0
0	1	11	34	11	0
0	11	12	23	6	0

TOTAL SUPPLIES = 95 TOTAL DEMANDS = 95 INITIAL
 COST = 365

AGGREGATION MEASURE...

M1(R) = .00 M2(R) = .00 M3(R) = .00 M(R) = .00

BEGIN DISAGGREGATION...

RESULTS AFTER DMG A), B), D) AND REDESIGNATION;

NUMARC: -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 -1
 1 -1 1
 ACP: 7 -5 9 12 -5 3 24 8 2 10 21 25 23
 -16 11 6
 SOLN: 1 5 9 0 5 0 3 0 0 10 18 25 21
 16 10 0
 TYPE: 0 0 0 0 0 0 0 3 3 0 0 0 0
 0 0 3

AALOC = 0 0 0 4 0 0 0 12 13 0 15 0
 0 0 1 5 0 10 11 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0

0	0	2	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	6	0	0	0	0	0	14	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	7	-8	0	0	0	0	0	0	-16
0	0	3	0	-9	0	0	0	0	0	0	0

DMG C): CONSTRUCTION OF SUBPROBLEM SNP(R+1)...

SNP(R+1) ARCS CORRESPOND TO NEGATIVE ENTRIES OF AALOC ABOVE.

***** SUBPROBLEM SNP(R+1) *****

3 NODES

NODE: 1 2 3
SUPPLY: -3 12 -9

ARC LIST...

TITLE	FROM	TO	COST	CAPACITY	LOW BND
0	2	3	23	6	0
0	2	1	26	8	0
0	3	1	20	2	0

TOTAL SUPPLIES = 12 TOTAL DEMANDS = 12 INITIAL
COST = 0

CORRESPONDENCE ARRAY: TRANSLATES GNET NODE
DESIGNATION-NP(R) DESIGNATION:
GNET DESIGNATION: 1 2 3
NP(R) DESIGNATION: 5 11 12

3 ARCS

TIME= 0 SECONDS, PIVOTS= 2, NNE= 32, NNS= 2,
IPG= 1

ARTIFICIAL ARC,NODE= 2 FLOW= 3 U(I)=
0 N2= 4
ARTIFICIAL ARC,NODE= 3 FLOW= 3 U(I)=
-78 N2= 4

FINAL SOLUTION...

FROM	TO	FLOW
2	1	3
2	3	6

COST= 216

DISAGGREGATION COMPLETE...

```

NUMARC:  -1   1  -1   1   1   1  -1  -1   1  -1  -1  -1  -1
1  -1   1
ACP:      7  -5   9  12  -5   3  24   8   2  10  21  25  23
-16  11  -6
SOLN:     1   5   9   0   5   0   3   3   0  10  18  25  21
16  10   6
TYPE:     0   0   0   0   0   0   0   3   3   0   0   0   0
0   0   3

```

ARTIFICIAL ARCS REQUIRED:

```

ART:      11  13
ARH:      13  12
ARCMIN: *** ***
ARLB:     0   0
ARCP:     *** ***
ARSOLN:   3   3

```

***** SOLUTION OF NP(R)) *****

16 ARCS

```

TIME=      0 SECONDS, PIVOTS=    16, NNE=   32, NNS=    9,
IPG=      2

```

FINAL SOLUTION...

FROM	TO	FLOW
2	3	4
6	3	5
12	3	6
2	4	2
11	4	6
11	5	3
2	6	10
2	7	18
1	8	20
1	9	11
8	10	16
1	11	10
11	12	6
COST=	4723	

BIBLIOGRAPHY

1. Balas, E., "Solution Of Large-Scale Transportation Problems Through Aggregation," Operations Research, Vol. 14, No. 5, 1963.
2. Bradley, G. G., G. Brown and G. Graves, "Design And Implementation of Large Scale Primal Transshipment Algorithms," Management Science, Vol. 24, No. 1, 1977.
3. Burkett, D.L., An Application of Cluster Analysis Techniques To Customer Aggregation, M.S. Thesis, UCLA, 1978.
4. Corley, H.W. and S.D. Roberts, "A Partitioning Problem with Application in Regional Design," Operations Research, Vol. 20, No. 5, 1972.
5. Cullen, D., H. Kuhn and M. Frank, "Aggregation In Network Models For Transportation Planning," Report DOT-TSC-RSPD-78-7, Mathematica, Inc., February, 1978.
6. Day, R.H., "On Aggregating Linear Programming Models of Production," Journal of Farm Economics, Vol. 45, 1963.
7. Diehr, G., An Investigation of Computational Algorithms for Aggregation Problems, Ph.D. Dissertation, UCLA, 1969.
8. Dyer, M.E., "Calculating Surrogate Constraints," Mathematical Programming, Vol. 19, No. 3, 1980.
9. Evans, F., "Aggregation in Genaralized Transportation Problems," Computers And Operations Research, Vol. 6, 1979.
10. Fisher, F., "Approximate Aggregation and the Leontief Conditions," Econometrica, Vol. 37, No. 3, 1969.
11. Fisher, F., "On Grouping For Maximum Homogeneity," JASA, Vol. 53, 1958.
12. Fisher, F., "Simplification of Economic Models," Econometrica, Vol. 34, No. 3, 1966.
13. Flament, C., Application of Graph Theory To Group Structure, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1963.
14. Geoffrion, A.M., "Customer Aggregation In Distribution Modeling," Working Paper No. 259, Western Management Science Institute, UCLA, Los Angeles, CA, 1976.

15. Geoffrion, A.M., "Elements Of Large-Scale Mathematical Programming," Management Science, Vol. 16, No. 11, 1976.
16. Geoffrion, A.M., "Aggregation Theory And Its Application to Modeling in Mathematical Programming," Working Paper No. 278, Western Management Science Institute, UCLA, Los Angeles, CA, 1977.
17. Geoffrion, A.M., "A Priori Error Bounds for Procurement Commodity Aggregation in Logistics Planning Models," Naval Research Logistics Quarterly, Vol. 24, No. 2, June 1977.
18. Geoffrion, A.M., "Objective Function Approximations in Mathematical Programming," Mathematical Programming, Vol. 13, No. 1, 1977.
19. Geoffrion, A.M., Model Aggregation, preliminary draft, Western Management Science Institute, UCLA, Los Angeles, CA, 1982.
20. Glover, F. and D. Klingman, "Real World Applications of Network Related Problems and Breakthroughs In Solving Them Efficiently," Research Report CCS 159, University of Texas, Austin, 1974.
21. Glover, F. Karney and D. Klingman, "Implementation and Computational Study of Start Procedures and Basis Change Criteria For A Primal Network Code," Networks, Vol. 4, 1974.
22. Graves, G., Linear Programming, preliminary edition, Graduate School Of Management, UCLA, Los Angeles, CA, 1973.
23. Green, H.A.J., Aggregation In Economics, Princeton University Press, 1964.
24. Greenberg, H.J. and W.P. Pierskalla, "Surrogate Mathematical Programming," Operations Research, Vol. 18, No. 5, 1970.
25. Greenberg, H.J., J.R. Lundgren and J.S. Maybee, "The Structure of Linear Programs, A Foundation for Computer Analysis," Working Paper, Department of Energy, 1981.
26. Hax, A.C., "Aggregate Production Planning," in J.J. Moder and S.E. Elmaghraby (eds.), Handbook of Operations Research, Vol. 2, Van Norstad Reinhold, 1978.
27. Hearn, D. and H. Kuhn, "Network Aggregation in Transportation Planning- Final Report (draft)," Mathtech, Inc., Princeton, N.J., 1977.

28. Heudecker, H., "Aggregation in Input-Output Analysis: An Extension of Fisher's Method," Econometrica, Vol. 38, No. 6, 1970.
29. Hirose, H., "A Modified Aggregation Program for the PILOT Process Integrated Model," Technical Report, SOL 80-31, Systems Optimization Laboratory, Stanford University, 1980.
30. Horowitz, E. and S.Sahni,Fundamentals Of Data Structures, Potomac, MD: Computer Science Press, Inc.,1976.
31. Huberman, G., "Error Bounds for the Aggregated Convex Programming Problem," Mathematical Programming, Vol. 26, 1983.
32. Ijiri, Y., "Functional Analysis Of Aggregation," Technical Report No. 21, Department of Operations Research, Stanford University, 1965.
33. Ijiri, Y., The Linear Aggregation Coefficient as the Dual of the Linear Correlation Coefficient," Econometrica, Vol. 36, No. 2, 1968.
34. Ijiri, Y., "Fundamental Inquiries in Aggregation Theory," JASA, Vol. 66, No. 336, 1971.
35. Jensen, P.A., "Optimum Network Partitioning," Operations Research, Vol. 19, No. 4, 1971.
36. Kallio, M., "Computing Bounds for the Optimal Value in Linear Programming," Naval Research Logistics Quarterly, 1977.
37. Keith, N.K., Aggregation In Large-Scale Distribution Systems, Ph. D. Disseration, Krannert School of Management, Purdue University, 1978.
38. Landsdowne, Z.F., "Survey Of Research On Model Aggregation And Simplification," Technical Report SOL 79-26, Department Of Operations Research, Stanford University, 1979.
39. Lee, S., Surrogate Programming by Aggregation, Ph. D. Dissertation, UCLA, Los Angeles, CA, 1975.
40. May, K., "The Aggregation Problem for a One-Industry Model," Econometrica, Vol. 14, 1946.
41. McGinnis, L.F., Report 103, North Carolina State University, 1975.

42. Mendolssohn, R., "Improved Bounds for Aggregated Linear Programs," Operations Research, Vol. 28, No. 6, 1980.
43. Miller, T.A., "Sufficient Conditions for Exact Aggregation In Linear Programming Models," Agricultural Economics Research, Vol. 18, 1966.
44. Miranker, W.L. and V.Y. Pan, "Methods Of Aggregation," Linear Algebra Applications, Vol. 29, 1980.
45. Raine, P.S., R.B. Flavell and G.R. Salkin, "Determining Appropriate Levels Of Data Aggregation in a Linear Programming Model," European Journal Of Operational Research, Vol. 2, No. 1, 1978.
46. Ritzman, L.P., et. al., (eds.), Disaggregation, Martinus Nijhoff Publishing, Boston, 1979.
47. Srinivasan, V. and G. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," Journal Of The Association For Computing Machinery, Vol. 20, 1973.
48. Taylor, R.W., "Aggregate Programming in Large-Scale Linear Systems," Ph. D. Dissertation, Georgia Institute Of Technology, 1983.
49. Taylor, R.W. and C.M. Shetty, "Solving Transportation Problems Via Aggregation," Report Series No. J-84-6, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1984.
50. Thompson, G.L., F.M. Tonge and S. Zionts, "Techniques for Removing Non-Binding Constraints and Extraneous Variables From Linear Programming Problems," Management Science, Vol. 12, No. 7, 1966.
51. Vakhutinsky, I.Y., L.M. Dudkin and A.A. Ryvkin, "Iterative Aggregation-A New Approach To The Solution of Large-Scale Problems," Econometrica, Vol. 47, No. 4, 1979.
52. Wollmer, R., "Aggregation of Constraints And Variables in Linear Programs," Energy Project Memorandum 77-1, Systems Optimization Laboratory, Stanford University, 1977.
53. Zaremba, P., "A Note on Consistent Aggregation of Production Functions," Econometrica, Vol. 36, No. 2, 1969.
54. Zipkin, P., Aggregation In Linear Programming, Ph. D. Dissertation, Yale University, 1977.

55. Zipkin, P., "Bounds on the Effect of Aggregating Variables in Linear Programs," Operations Research, Vol. 28, No. 4, 1980.
56. Zipkin, P., "Bounds for Row-Aggregation in Linear Programming," Operations Research, Vol. 28, No. 4, 1980.
57. Zipkin, P., "Bounds For Aggregating Nodes in Network Problems," Mathematical Programming, Vol. 19, No. 2, 1980.
58. Zipkin, P. and K. Raimier, "An Improved Disaggregation Method For Transportation Problems," Mathematical Programming, Vol. 26, 1983.
59. Zoller, K., "Optimal Disaggregation of Aggregate Production Plans," Management Science, Vol. 17, No. 8, 1971.

END

DTIC

9-86