

T-1468

THE OPTIMAL LOCATION OF MINE SHAFTS:

An Out-of-Kilter Technique
for the Solution of the
Fixed-Charge Problem.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL OF MINES
GOLDEN, COLORADO

By

Oscar B. Nair

ARTHUR LAKES LIBRARY
COLORADO SCHOOL OF MINES
GOLDEN, COLORADO 80401

A Thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science.

Signed: Oscar B Nair
Oscar B. Nair

Golden, Colorado

Date: Apr. 17, 1972

Approved: Hunter Swanson
Dr. H. S. Swanson
Thesis Advisor

ARTHUR LAKES LIBRARY
COLORADO SCHOOL OF MINES
GOLDEN, COLORADO

JR Lee
Dr. J. R. Lee
Head of the Department

Golden, Colorado

Date: Apr. 17, 1972

ProQuest Number: 10781784

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10781784

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

ABSTRACT

The problem investigated here is that of determining the optimal number and location of hoisting shafts to service a known underground deposit at minimal total cost. The total cost is comprised of contributions from shaft sinking capital costs and from transportation costs that vary linearly with distance. The problem is formulated as a network. A modification of the out-of-kilter algorithm is used to produce a sequence of solutions which converge to a good (hopefully optimal) solution. It is believed that many fixed-charge problems with linear transportation costs may be solved with this approach.

TABLE OF CONTENTS

	<u>Page</u>
Abstract	iii
Acknowledgments	viii
Introduction	1
Statement of the Problem	4
Literature Survey	6
Preparation of the Model	11
Blocking	11
Rectilinear Distances	13
Block Weights	13
Surface Topography	14
Mathematical Formulation	15
The Shaft Location Model	17
Formulation as a Network	21
Elements of Network Theory	21
The Out-of-Kilter Algorithm	27
Formulating the Shaft Location Problem as a Network	41
Method of Solution	44

	<u>Page</u>
The Optimal Mine Shaft Location Algorithm	50
The Computer Program	53
Introduction	53
Description of Subroutines	53
Subroutine DIST	53
Subroutine INIT	55
Subroutine NETFLO	56
Subroutine SORT	56
Subroutine LABEL	57
Subroutine RECAP	57
Subroutine LABEL1	58
Subroutine PRICE	58
Subroutine TCOST	59
The Main Program	59
Computational Results	60
Sample Problems	60
Results	61
Discussion	66
Further Applications	67
Appendices	70
I: Program Listing	71

	<u>Page</u>
II: Sample Output	91
List of References	107

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1	Block Model of a Mineral Deposit	12
2	The Labeling Routine for the Out-of-Kilter Algorithm	36
3	The Flow-Augmentation Routine for the Out- of-Kilter Algorithm	37
4	The Pricing Routine for the Out-of-Kilter Algorithm	39
5	The Shaft Location Problem as a Network . . .	42
6	The Mine Shaft Location Algorithm	54
7	Plan of Deposit Used for Sample Problem . . .	62

TABLES

<u>Table</u>	<u>Title</u>	<u>Page</u>
1	Labeling Rules for Out-of-Kilter Algorithm	35
2	Input Data for Sample Problem	63

ACKNOWLEDGMENTS

The contribution of Dr. Thys B. Johnson, of the U. S. Bureau of Mines, is gratefully acknowledged. He suggested the possibility of formulating the problem as a network, and his constant guidance has helped in the evolution of the algorithm.

I am also indebted to my thesis advisor, Dr. H. S. Swanson, for his valuable comments and suggestions.

INTRODUCTION

Mining is essentially a business of extracting saleable material from the earth's crust for the least possible cost while complying with health, safety, and environmental constraints. In the process of designing a mine, one frequently has to make choices among several different alternatives. Historically, many of these choices were made intuitively by an engineer, using his experience as a guide. With the rapid advances made lately in Operations Research and with the availability of high speed computers with large core/disk storage, some (though not all) of these mining problems have become amenable to mathematical formulation and solution. One such problem is that which arises when a new underground mine is being planned: Where should hoisting shafts be located, how many should there be over the life of the mine, and to which shaft should material from each area underground be transported?

The purpose of this thesis is to present a heuristic algorithm that may be used to determine the number and location of hoisting shafts for a deposit for which the transportation network is on a single level. This algorithm does not guarantee optimality but computational results indicate that solutions are near optimal and are produced in reasonable computation times.

Present practice for determining the number and location of shafts is primarily based on the surface topography of the mine, the proximity of available or proposed surface rail transportation, and the expected subsurface geology, as related to the cost of shaft sinking. Such calculations, while using the best available methods, seldom determine shaft locations such that the expected total cost is a minimum. A common practice is to sink two shafts close to each other in an area over the deposit where the shaft sinking costs may be expected to be minimum. One of the shafts becomes a hoisting/intake shaft and the other becomes an exhaust shaft. The hoisting shaft is used as such until it becomes obvious that transportation costs underground have become exceedingly high. If there is sufficient reserve left underground at this time, a second hoisting shaft is sunk at another location so as to reduce underground transportation costs. There is nothing to recommend this approach other than its ease of application. If the optimum

number and location of hoisting shafts for a deposit were determined with respect to shaft sinking costs, underground transportation costs, and surface location of other facilities, it would be possible to determine the time period during which each shaft would be used as a hoisting shaft, the dates by which new shafts should be ready for use, and the daily capacity for which each shaft should be designed. Within the uncertainty connected with cost figures, expected future demand, and other such variables, prior knowledge of the kind described above will help the planning engineer to develop the best possible method of exploiting the deposit.

STATEMENT OF THE PROBLEM

The objective of this thesis is to develop a method to determine the location of hoisting shafts for a deposit of known configuration such that the sum of expected shaft capital costs and expected transportation costs is a minimum.

The information required for this problem is:

- a. The average cost of transporting one unit of material through a unit distance underground.
- b. The expected cost of sinking a shaft at any point over the deposit, or a list of possible shaft locations with the associated shaft sinking costs.

The assumptions made are:

- a. Transportation of material underground is along rectilinear paths on a single level.
- b. Transportation cost increases linearly with distance.

The required information described above is generally available to the planning engineer. The information is, of course, in the form of rough estimates, but these estimates are the best

available. For example, for a given shaft size and a pre-determined sub-surface geology, the cost of sinking a shaft may be estimated as being directly proportional to the depth from surface to the deposit. The average unit cost of underground transportation for a given mode of transportation may generally be estimated from manufacturer's information and from information gathered from operating mines in the vicinity. Often a list of possible shaft locations may be determined, a priori, through a process of elimination. Steep slopes, streams, surface structures, interference with the surrounding environment, and problems with surface rights may eliminate a large portion of the surface from being considered as possible shaft locations.

The assumption that the transportation network underground is rectilinear is explained in the next section. The assumption that the transportation network is on a single level permits this algorithm to be used not only for single seam mines like coal, but also for mines using block caving methods associated with single level transportation. By inference, this algorithm cannot be used for mines using multi-level simultaneous transportation. The assumption that transportation costs increase linearly with distance is a standard assumption. If transportation costs are divided into operating and capital costs, it is apparent that the operating costs increase linearly with the distance of

operation. The capital costs, when depreciated over the operating life of the piece of equipment will introduce a factor not necessarily proportional to the distance of transportation. It is assumed that this factor is insignificant in that the total transportation cost remains very nearly proportional to the distance of transportation.

It is appreciated that the capital cost of sinking a shaft is also proportional to the capacity (tonnage per day) for which the shaft is designed. Since this capacity is a function of the demand for the material mined, and since, on a cash flow basis, it is almost always economical to have only one hoisting shaft at any one time, it is assumed that the required shaft capacity is determined by the planning engineer. Hence, the cost of sinking a shaft on any of the potential shaft locations may be calculated as a function of only the depth of sinking and the subsurface geological and hydrological conditions.

Literature Survey

The current literature contains only one publication on the problem of the optimum location of mine shafts. Zambo (1968) wrote a book that describes the various problems of optimal location of mine facilities. He used a geometrical-calculus approach that, in general, does not take into consideration non-uniform variations of the minable tonnage in an orebody. This

and other simplifications of location problems severely limit the applicability of his approach.

Numerous articles have been written on the fixed-charge problem. Some of these are described in brief below:

Gray (1967) does an excellent job in describing a technique that can be used to solve the following classes of problems:

- a. A fixed-charge problem with linear variable costs in which a fixed charge is associated with each continuous variable that appears at a non-zero level.
- b. A warehouse location problem in which variable costs and constraints are of the transportation type. A fixed charge is associated with each warehouse opened, irrespective of the number of customers served.
- c. The fixed-charge transportation problem in which a charge is associated with each route rather than each warehouse.
- d. The 1 -knapsack problem with separable concave or convex costs rather than linear cost.

Gray decomposes each problem into a master problem and a series of linear programs while using a bound-and-scan algorithm to obtain 0-1 vectors that satisfy the fixed-charge part of the objective function. Computational results are given for

examples of each class of problems. An excellent literature survey and bibliography is also a part of the publication. The difficulty of the approach pursued by Gray is that the size of the problems that can be handled by the computer code written for the fixed-charge problem with transportation costs is too small to be used for a realistic shaft location problem. Moreover, the computation time is excessively large.

Benichou, et al (1971) present a branch and bound method for solving mixed integer linear programs. With proper formulation, some classes of fixed-charge problems may be solved with this method. The algorithm has been implemented as a computer program that is available in the IBM Extended Mathematical Programming System. No computational results on the fixed-charge problem with linear transportation costs are available for comparison.

Murty (1968) solves the fixed-charge problem by a procedure that ranks basic feasible solutions corresponding to a linear programming problem in increasing order of the linear objective function. The algorithm works efficiently when the range in values of the variable costs is large compared to the fixed charges. Here again, the difficulty lies in implementation. The size of the problems that can be solved are too small to be of use in solving the shaft location problem.

Steinberg (1970) presents an exact algorithm for the solution of fixed-charge problems. He uses a branch-and-bound approach to converge to an optimal solution. The computer program is capable of handling large problems and arrives at the optimal solution in realistic computation time. He also presents three heuristic algorithms that provide a "good" solution very rapidly. If there is a disadvantage with Steinberg's algorithms, it is that the formulation of the problem and preparation of the data would require a level of sophistication in Mathematical Programming not easily found in the mining industry.

The idea of using rectilinear distances came from a paper by Wesolowsky and Love (1971). Their technique of locating new facilities with respect to existing ones could, in particular, be used to locate new hoisting shafts with respect to existing hoisting shafts, and cleaning plants.

A paper that describes the use of the out-of-kilter algorithm to solve a rectilinear distance facility location problem was written by Cabot, et al (1970). This work is also of interest from the point of view of locating new shafts with respect to existing ones. The problem of minimizing total costs if a fixed charge is incurred with every new facility located, is mentioned in the paper as a subject of future research.

Other papers on the location of facilities include Francis (1964), and Humphreys, et al (1970).

Summarizing, a review of the literature on the solution of fixed-charge problems with linear transportation costs, shows that, except for Steinberg, none of the techniques presented can handle problems of the size posed by the shaft location problem. Further, computation times are often quite large.

The network approach of solving the shaft location problem that is described in this thesis, will permit any engineer, with hardly any difficulty, to formulate the problem, prepare the input and understand the answers. Further, the size of the problems that can be handled are realistic and the computation times are small. A 40 x 18 problem (40 blocks and 18 potential shaft locations) typically takes only 1.6 minutes on an IBM 360/65J. An equivalent mixed integer program will have about 778 variables and will run for hours before converging to a solution.

PREPARATION OF THE MODELBlocking

In order that the problem may be formulated mathematically, it should first be possible to describe the deposit in a numerical manner. This is done using the block concept (Johnson and Mickle, 1971). A two-dimensional vertical projection (plan) of the deposit is divided by a grid into blocks (Figure 1). The number of blocks (fineness of the grid) should be commensurate with the storage capacity of the computer.

The computer program given in Appendix I can handle a program of about 50 blocks and 20 possible shaft locations or any combination of m blocks and n shaft locations where $m \times n = 1000$. The storage capacity used for such a problem is about 30K words. By changing dimension and common statements the capacity could be increased or decreased to fit the core storage of the computer being used.

In case a more accurate determination of the location of shafts is desired, after the algorithm gives a solution to the problem defined by a grid with large blocks, those blocks that

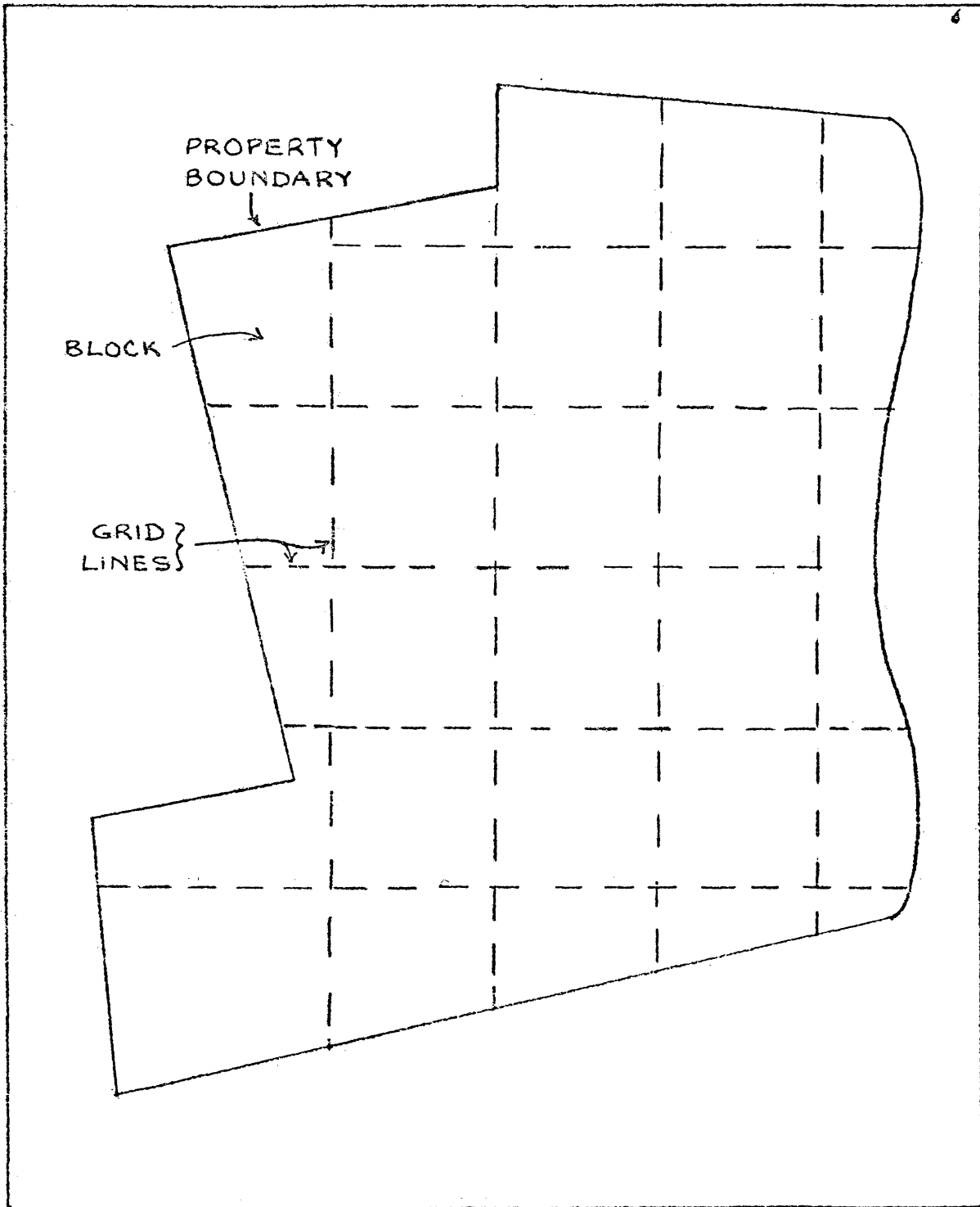


Figure 1 Block Model of a Mineral Deposit

have been eliminated from consideration as possible shaft locations may be merged into one. Blocks that have been selected as shaft locations may be subdivided into smaller blocks and the algorithm could be used again to give a better resolution of the shaft locations. Repeated subdivision of blocks in the manner described above could determine the shaft locations as accurately as the user desired.

Rectilinear Distances

Ideally, it would be desirable to determine the underground transportation network before determining the location of the shafts. However, due to the interaction between the two problems, an assumption must be made for one in order to determine the other. Thus, in this work, an assumption is made that the underground transportation network is rectilinear; i. e. the distance from point \underline{i} with coordinates (x_i, y_i) to point \underline{j} with coordinates $(x_j, y_j) =$

$$D_{ij} = |x_i - x_j| + |y_i - y_j|$$

In actual practice, transportation networks in coal, salt, trona, and similar seam mines are very nearly rectilinear in nature.

Block Weights

The rectangular grid divides the deposit into rectangular blocks. Since the approximate thickness of the deposit at all

points is known, it is possible to estimate the tonnage of minable material in each block. This tonnage may be defined as the block weight, W_i , for block i .

Surface Topography

Let the point (x_i, y_i) denote the coordinates of the center of gravity of each block i . By looking at a topographical map of the deposit and overlaying it with a surface topography map, it is possible to determine the depth of the deposit from the surface at the center of gravity of each block i . If capital costs of sinking shafts are assumed to be directly proportional to the depth of the deposit, these costs could then be calculated. Further, by looking at the surface topography map of the area, a large number of these blocks could be eliminated from consideration as potential shaft sites due to constraints posed by streams, steep and rugged slopes, existing structures, access roads, gas wells, old workings underground, and faults. In effect, one says that sinking a shaft in these blocks can only be done at an exorbitant cost. The remaining blocks are those considered as potential shaft locations and the capital cost of sinking shafts in these blocks is assumed to be known for each block.

MATHEMATICAL FORMULATION

The following is a listing and description of all the symbols used in this work.

C_{ij}	unit cost for arc (i, j). It is equivalent to the unit transportation cost from <u>i</u> to <u>j</u> .
\bar{C}_{ij}	effective cost coefficient = $C_{ij} - \pi_i + \pi_j$
D_{ij}	rectilinear distance from block <u>i</u> to potential shaft location <u>j</u>
f_{ij}	flow along arc (i, j) representing the tonnage transported from block i to shaft location j. The value of this variable will be determined by the algorithm.
i	a block made by the grid.
j	a potential shaft location block
k_{ij}	upper bound for arc (i, j).
l_{ij}	lower bound for arc (i, j).
M, m	total number of blocks.
N, n	total number of potential shaft locations.
P_j	capital cost of sinking a shaft at <u>j</u>
s	source node.
t	sink node.

t_{ij}	transportation cost to transport one unit of material from block \underline{i} to potential shaft location \underline{j}
T	unit transportation cost.
U_{jt}	upper bound for shaft arc (j, t) . This variable is unknown at the start and will be determined by the algorithm. The final value will represent the total tonnage passed through shaft j .
V	value of the objective function in the dual problem.
W_i	total minable tonnage from block \underline{i} .
w_{ij}	tonnage transported from block \underline{i} to potential shaft location \underline{j}
x_{ij}	quantity of commodity shipped from warehouse \underline{i} to customer j (Gray's warehouse location problem).
(x_i, y_i)	cartesian coordinates of block \underline{i} .
y_j	a variable that takes a value of 0 or 1.
Z	value of the objective function in the primal problem.
ϕ_{ij}	dual multiplier associated with the upper bound constraint.
ψ_{ij}	dual multiplier associated with the lower bound constraint
π_i	dual variable (price) for node i ; the dual variable associated with conservation of flow equations.

The Shaft Location Model

If T is the unit transportation cost, then the unit cost of transportation from block \underline{i} to block \underline{j} along rectilinear paths is

$$t_{ij} = T \left\{ |x_i - x_j| + |y_i - y_j| \right\} = T D_{ij} \quad (1)$$

The capital cost of sinking a shaft in block \underline{j} is P_j

(Assumption 2: P_j is known for each \underline{j}).

The shaft location problem may now be formulated as:

$$\text{Minimize } Z = \sum_{j=1}^N \sum_{i=1}^m t_{ij} w_{ij} + \sum_{j=1}^N P_j y_j$$

PROBLEM I

$$\text{Subject to } \sum_{j=1}^N w_{ij} = W_i ; i = 1, \dots, m$$

$$y_j = \begin{cases} 0, & \text{if } \sum_{i=1}^m w_{ij} = 0 \\ 1, & \text{otherwise} \end{cases}$$

$$w_{ij} \geq 0$$

where w_{ij} is the weight of material transported from block \underline{i} to potential shaft location \underline{j} and W_i is the total weight of material in block \underline{i} . This is in the form of a Fixed-Charge Problem (Hillier and Lieberman, 1967, p. 565).

Gray (1967) formulates the warehouse location problem as a fixed-charge model with transportation costs. A warehouse capacity is associated with each warehouse.

The warehouse location problem is formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_i \sum_j C_{ij} x_{ij} + \sum_i f_i y_i \\ \text{Subject to} \quad & \sum_i x_{ij} \geq D_j ; j=1, \dots, N \quad \text{PROBLEM II} \\ & M_i y_i - \sum_j x_{ij} \geq 0 ; i=1, \dots, m \\ & y_i \leq 1 \\ & y_i, x_{ij} \geq 0 \quad y_i \text{ integer} \end{aligned}$$

where D_j is the demand of customer \underline{j} and M_i is the capacity of warehouse \underline{i} .

The Shaft Location Problem is slightly different from the Warehouse Location Problem in that no "capacity" is associated with each shaft. By making M_i a very large number, it is possible to represent the shaft location problem as a fixed-charge problem with transportation costs:

$$\text{Minimize} \quad \sum_i \sum_j C_{ij} f_{ij} + \sum_j P_j \delta_j$$

$$\begin{aligned}
 \text{Subject to } \quad \sum_j f_{ij} &= W_i ; i=1, \dots, m && \text{PROBLEM III} \\
 M_j \delta_j - \sum_i f_{ij} &\geq 0 ; j=1, \dots, N \\
 \delta_j &\leq 1 \\
 \delta_j, f_{ij} &\geq 0, \delta_j \text{ integer}
 \end{aligned}$$

If the values of the M_j 's are not known, this problem cannot be solved by Gray's algorithm.

The capacity of the shaft in this context is not the maximum daily tonnage that can be hoisted in the shaft, but is the total tonnage that will pass through the shaft over the life of the shaft. Determination of daily tonnage capacity is a function of other considerations such as demand. If one were to specify that the total tonnage that passed through each shaft was not to exceed a certain value, then the M_j 's would be specified, and the problem might be solved by Gray's algorithm.

The algorithm described in this thesis to solve the shaft location problem, in effect keeps increasing the value of M_j for a subset of shafts until no capacity can be increased further without an increase in the total cost. The values of M_j for those shafts that have positive allocation, then, give the total volume of material that may be expected to pass through that shaft during

the life of the mine. In the context of the fixed-charge model, these values of M_j define the "capacity" of the shaft.

The present state-of-the-art of computer algorithms to solve mixed-integer problems such as Problems I, II, and III is such that only small problems can be solved and the computation time is generally large. Further, the formulation of the mixed-integer problem from the industrial problem generally needs some skill not often available in the industry. There is the need, therefore, for a method of solution that is computationally efficient, capable of handling problems of realistic size, and that can be formulated intuitively by an engineer (not necessarily a mathematician). The algorithm developed in this thesis for the solution of the shaft location problem fits the above criteria.

FORMULATION AS A NETWORK

Elements of Network Theory

In order to understand the representation of the shaft location problem as a network, a brief introduction into the theory of networks is given below. A more extensive treatment of the subject is given by Ford and Fulkerson (1962) and Elmaghraby (1971).

A graph is a collection of points (called nodes) and lines (called arcs) that connect the points. No quantitative characteristics need be associated with either the nodes or the arcs.

A network is a graph with quantitative characteristics associated with the nodes and/or the arcs. If a node is denoted by a lower case letter such as i , an arc connecting node i to node j may be denoted as arc (i, j) . Parameters (quantitative characteristics) associated with each arc may be denoted with the node numbers as subscripts. Thus, k_{ij} is the upper bound of flow per unit time in arc (i, j) , l_{ij} the lower bound, and C_{ij} the cost per unit flow. Flow along (i, j) is denoted by f_{ij} . For the out-of-kilter algorithm that will be used to solve the shaft location problem, all parameters of the network are assumed to be

rational numbers. The actual operation of the algorithm only permits integers. A network with non-integer rational numbers as characteristics may be formulated such that all characteristics are integers (Durbin and Kroenke, 1967).

An arc with its two end nodes is called a branch. If an arc has an orientation, it is a directed arc and the branch is a directed branch. A directed branch sequence from \underline{i} to \underline{j} in which some of the branches are traversed opposite to their orientation is called a path (Elmaghraby, 1970). A circuit, loop, or cycle is a path whose original node of the first arc and terminal node of the last arc coincide. A circulation in a network or along a loop is the existence of flows along each arc in the network or loop such that:

1. The principle of conservation of flow is maintained at each node, and
2. The flow along each arc does not violate the upper and lower bound constraints of the arc.

The principle of conservation of flow in a network states that the sum of the flows into a node must equal the sum of the flows out of the node.

A static network is one in which the parameters of the network remain the same regardless of the flow. A dynamic

network is one where some or all of the parameters are functions of the flow through the network.

Networks may be used to model or simulate various real life situations. Common applications of network theory are to problems connected with:

1. Transportation systems
2. Communication systems, and
3. Distribution systems.

In general, any system which can be formulated as a problem in which flow from one point to another is subject to a certain class of constraints, may be formulated as a network. Examples of real life problems that may be simulated by a network are the study of flow through a network of oil and gas pipelines, routing of vehicles through a highway system, airline routes, design of computer circuitry, and the assignment of men to machines.

Most network problems may be characterized as falling into one of the three following categories:

1. Find the maximum flow from a designated node to a different designated node subject to the capacity restrictions on flow in each arc and conservation of flow at each node (Max flow problem).

2. Find the maximum flow from a designated node to a different designated node which minimizes the cost incurred subject to the capacity restrictions on flow in each arc and conservation of flow at each node (Max flow at Min cost problem).
3. Find a minimum cost flow subject to capacity restrictions on flow in each arc and conservation of flow at each node. (Min cost circulation problem)

The minimum cost circulation problem is the most general of the three problems. The out-of-kilter algorithm (OKA) is an efficient algorithm to solve the minimum cost circulation problem.

Further useful definitions for understanding networks are as follows:

Let the set of all nodes in a network be denoted by N , and the set of all arcs be denoted by A .

Let X and Y be two subsets of nodes.

(X, Y) is defined as the set of all arcs (i, j) such that $\underline{i} \in X$ and $\underline{j} \in Y$.

Nodes \underline{i} and \underline{j} are defined as adjacent nodes if arc (i, j) belongs to A .

Let A_i be defined as the set of nodes to which arcs are directed from node \underline{i} and let B_i be defined as the set of nodes with arcs directed to node \underline{i} .

It is convenient to define source and sink nodes in a network.

A source is a node at which no flow terminates and a sink is a node from which no flow originates. In this work a source is denoted as \underline{s} and a sink as \underline{t} . A cut separating nodes \underline{p} and \underline{q} is a collection of arcs of the form (X, X^c) where $p \in X$ and $q \in X^c$. The cut will

have a capacity $c(X, X^c) = \sum_{i \in X} \sum_{j \in X^c} k_{ij}$

The Max flow-Min cut theorem (Ford and Fulkerson, 1962) which is a basic theorem used in network flow problems may be stated as follows: For any network, the maximal flow from \underline{p} to \underline{q} is equal to the minimal cut capacity of all cuts separating \underline{p} and \underline{q} .

The remainder of this section will be devoted to describing and formulating the minimum cost circulation problem. Given a network with a set of nodes, N , and a set of arcs, A , and given that each arc has upper and lower bounds on flow in the arc and has an associated cost per unit flow in the arc, the minimum cost circulation problem will be to find the flow, f_{ij} , along each arc (i, j) of the network which minimizes total cost subject to capacity and conservation of flow constraints.

In mathematical programming terms, the problem is

$$\text{Minimize } Z = \sum_{i,j} C_{ij} f_{ij} \quad (2)$$

$$\text{Subject to } f_{ij} \geq l_{ij} ; \forall (i,j) \in [A] \quad (3)$$

$$-f_{ij} \geq -k_{ij} ; \forall (i,j) \in [A] \quad (4)$$

$$-\sum_j f_{ji} + \sum_j f_{ij} = 0 ; \forall i \in [N] \quad (5)$$

$$f_{ij} \geq 0 ; \forall (i,j) \in [A] \quad (6)$$

The set of equations represented by equations 3 and 4 represent the capacity constraints on flow along each arc. Equation 5 represents the conservation of flow constraint for each node, \underline{i} .

Equations 2 through 6 is a linear program. The dual linear program may be written as

$$\text{Max } v = \sum_{i,j} \psi_{ij} l_{ij} - \sum_{i,j} \phi_{ij} k_{ij} \quad (7)$$

$$\text{Subject to } -\pi_i + \pi_j + \psi_{ij} - \phi_{ij} \leq C_{ij}; \forall (i,j) \in [A] \quad (8)$$

$$\psi, \phi \geq 0 \quad (9)$$

$$\pi \text{ unrestricted} \quad (10)$$

where ψ_{ij} is dual variable associated with the lower bound constraint for arc (i,j) , ϕ_{ij} is the dual variable associated with the upper bound constraint for arc (i,j) , and π_i is the dual variable associated with the conservation of flow constraint for node \underline{i} .

The out-of-kilter algorithm uses the primal-dual relationships (Ford and Fulkerson, 1962, p. 26) to re-route flows so as to find an optimum solution to the minimum cost circulation problem.

The Out-of-Kilter Algorithm

Introduction: The description of this algorithm relies heavily on that found in the paper by Durbin and Kroenke (1967).

The algorithm is initialized with a circulation flow of zero in each arc (conservation of flow is preserved at each node). Such a circulation might not satisfy capacity constraints. A number (the node "price") is assigned to each node. These numbers are actually the dual variables associated with the conservation of flow at each node.

Let π_i be the node "price" for node \underline{i} .

Let $\bar{C}_{ij} = C_{ij} + \pi_i - \pi_j$ define a net arc cost for arc (i, j) .

\bar{C}_{ij} represents the total cost to the system of transporting one unit of flow from node \underline{i} to node \underline{j} .

The costs, C_{ij} , and the node prices, π_i , π_j , determine the flow along arc (i, j) . The prices are related to the amount of flow demand at each node and are analogous to the price that a consumer must pay at that node for a unit of the commodity being circulated through the network. In moving a unit of flow from node \underline{i} to

node \underline{j} , if the sum of the price of the commodity at node \underline{i} , $\pi_{\underline{i}}$, and the cost of transportation, $C_{\underline{ij}}$, is greater than the price of the commodity at node \underline{j} , $\pi_{\underline{j}}$, then, it obviously does not pay to ship a unit from node \underline{i} to node \underline{j} . Thus, if $\bar{C}_{\underline{ij}}$ is positive, flow from \underline{i} to \underline{j} is not economically justified. Alternately, if $\bar{C}_{\underline{ij}}$ is negative, it pays to ship a unit from \underline{i} to \underline{j} . If $\bar{C}_{\underline{ij}}$ is zero, it neither pays nor costs to ship a unit from \underline{i} to \underline{j} and we are indifferent to any additional flow. Further if $\bar{C}_{\underline{ij}}$ is positive, flow from \underline{i} to \underline{j} may be made economically justifiable by increasing the price of the commodity at \underline{j} by at least an amount equal to $\bar{C}_{\underline{ij}}$.

In-Kilter Conditions: The above introduction is an intuitive description of the actions of the out-of-kilter algorithm (OKA). The OKA is used to solve the problem described by (2) - (6).

From the complementary slackness theorem (Dantzig, 1963; Elmaghraby, 1970) which defines the conditions for optimality for the primal and dual linear programs, the following conditions are obtained:

$$\psi_{ij} > 0 \implies f_{ij} = l_{ij} \quad \dots\dots\dots (11)$$

$$\phi_{ij} > 0 \implies f_{ij} = k_{ij} \quad \dots\dots\dots (12)$$

$$f_{ij} > 0 \implies -\pi_{\underline{i}} + \pi_{\underline{j}} + \psi_{ij} - \phi_{ij} = C_{\underline{ij}} \quad \dots\dots\dots (13)$$

Note that if we assign values for ψ_{ij} and ϕ_{ij} according to

$$\begin{aligned}\psi_{ij} &= \text{Max} [0, C_{ij} + \pi_i - \pi_j] \quad \text{and} \\ \phi_{ij} &= \text{Max} [0, -C_{ij} - \pi_i + \pi_j] \quad \dots\dots\dots (14)\end{aligned}$$

then the dual problem is always feasible and if one of the dual variables, ψ_{ij} or ϕ_{ij} , is positive, then the other is zero.

$$\text{If } \bar{C}_{ij} = C_{ij} + \pi_i - \pi_j \quad \dots\dots\dots (15)$$

$$\text{then } \bar{C}_{ij} > 0 \Rightarrow \psi_{ij} > 0 \Rightarrow f_{ij} = l_{ij} \quad \dots\dots\dots (16)$$

$$\text{and } \bar{C}_{ij} < 0 \Rightarrow \phi_{ij} > 0 \Rightarrow f_{ij} = k_{ij} \quad \dots\dots\dots (17)$$

The in-kilter or optimality conditions that should be satisfied by each arc in the network at optimality are therefore:

$$\bar{C}_{ij} < 0 \quad \text{and} \quad f_{ij} = k_{ij} \quad \dots\dots\dots \text{Condition A}$$

$$\bar{C}_{ij} = 0 \quad \text{and} \quad l_{ij} \leq f_{ij} \leq k_{ij} \quad \dots\dots\dots \text{Condition B}$$

$$\bar{C}_{ij} > 0 \quad \text{and} \quad f_{ij} = l_{ij} \quad \dots\dots\dots \text{Condition C}$$

Condition A states that when it is profitable to send a commodity from i to j then flow along arc (i, j) ought to be as large as possible. Condition B states that we are indifferent to the flow level if the net arc cost is zero so long as the flow lies within the bounds of the arc. Condition C states that since a loss is incurred by shipping from i to j, then flow along arc (i, j) should be kept at the minimum level possible - the lower bound.

Out-of-Kilter Conditions: Any arc that does not satisfy the in-kilter (optimality) conditions is "out-of-kilter". The OKA tries to bring each arc into kilter and thus to optimality. The out-of-kilter arcs may either have feasible or infeasible flows.

Out-of-kilter arcs with feasible flows must satisfy one of the following conditions:

$$\bar{C}_{ij} < 0 \text{ and } l_{ij} \leq f_{ij} < k_{ij} \quad \dots\dots\dots \text{Condition I}$$

$$\bar{C}_{ij} > 0 \text{ and } l_{ij} < f_{ij} \leq k_{ij} \quad \dots\dots\dots \text{Condition II}$$

Infeasible arcs which are automatically out-of-kilter must satisfy one of the following conditions:

$$\bar{C}_{ij} > 0 \text{ and } f_{ij} < l_{ij} \quad \dots\dots\dots \text{Condition III}$$

$$\bar{C}_{ij} = 0 \text{ and } f_{ij} < l_{ij} \quad \dots\dots\dots \text{Condition IV}$$

$$\bar{C}_{ij} = 0 \text{ and } f_{ij} > k_{ij} \quad \dots\dots\dots \text{Condition V}$$

$$\bar{C}_{ij} < 0 \text{ and } f_{ij} > k_{ij} \quad \dots\dots\dots \text{Condition VI}$$

Kilter Number: A measure of optimality, the kilter number, is associated with each arc. The kilter number associated with each out-of-kilter condition is defined below:

<u>Arc Condition</u>	<u>Kilter Number</u>
I	$\bar{C}_{ij} [f_{ij} - k_{ij}]$
II	$\bar{C}_{ij} [f_{ij} - l_{ij}]$
III, IV	$[l_{ij} - f_{ij}]$
V, VI	$[f_{ij} - k_{ij}]$

If the kilter number is zero, the arc is in kilter and satisfies the optimality conditions.

The OKA operates by selecting the first out-of-kilter arc it encounters and rearranging flows in the network in an effort to reduce the kilter number of that arc while not increasing the kilter number of any other arc. The algorithm terminates with an optimal solution when the kilter number of each arc is zero, which implies that one of the optimality conditions A, B or C is satisfied for each arc. A proof that the algorithm terminates in at most $N(N-1)/2$ steps (where N = number of arcs) is given by Fulkerson (1961) and Ford and Fulkerson (1962, pp. 162-169).

Summary of the OK Algorithm: The OKA starts with arbitrary flow conditions that satisfy conservation of flow requirements and with arbitrary prices, π_i . An out-of-kilter arc (p, q) is selected so additional flow from p to q is required to reduce the kilter number of the arc (if flow has to be reduced, nodes p and q may be reversed). Node q is then labeled. Nodes adjacent

to q are then examined and the nodes through which additional flow is possible without increasing the kilter number of any arc are labeled appropriately. The process is continued until either p is labeled (a circuit is formed or a breakthrough has occurred) or all nodes that can be labeled have been labeled and p does not belong to the set of labeled nodes (a non-breakthrough).

In the case of a breakthrough, flow is increased along the circuit by the maximum amount that is permitted by the labeling procedure. Any increase in the flow will reduce the kilter number of arc (p, q) . If the kilter number becomes zero, another out-of-kilter arc is selected and the procedure repeated until all the arcs are in kilter. If the kilter number of arc (p, q) is still positive, the labeling procedure is repeated and flow augmented if a labeled circuit is formed.

If for the out-of-kilter arc (p, q) , no labeled circuit is formed (a non-breakthrough), the problem may be infeasible in that no flow change can be made to satisfy

$$l_{pq} \leq f_{pq} \leq k_{pq} \quad (18)$$

or, it may be necessary to change node prices to permit flow augmentation along (p, q) . The pricing routine changes prices so that it guarantees that at least one node will move from the unlabeled set to the labeled set when the labeling routine is performed.

To summarize, the out-of-kilter algorithm proceeds as follows:

0. Start with any circulation and any set of node prices.
1. Find an out-of-kilter arc, (p, q) . If none, stop.
2. Determine if the flow in arc (p, q) is to be increased or decreased to bring (p, q) into kilter.

If the flow should be increased, go to step 3.

If the flow should be decreased, go to step 4.

3. Find a path from q to p along which the flow can be increased without causing any arc on the path to become "more" out-of-kilter. If a path is found, increase the flow along the path and also in (p, q) . If (p, q) is in-kilter, go to step 1. If (p, q) is still out-of-kilter, repeat step 3. If no path can be found, go to step 5.
4. Find a path from p to q along which the flow can be increased without causing any arc to become more out-of-kilter. If a path is found, increase the flow along the path and decrease the flow in (p, q) . If (p, q) is in-kilter, go to step 1. If (p, q) is still out-of-kilter, repeat step 4. If no path is found, go to step 5.

5. Change the node numbers. If (p, q) is in-kilter, go to step 1. If (p, q) is still out-of-kilter, repeat step 2. If the node numbers become infinite, stop. There is no feasible flow.

The out-of-kilter algorithm may thus be divided into two routines - the labeling and flow augmentation routine and the pricing routine. Each of these are described below.

The Labeling and Flow Augmentation Routine: A label given to a node \underline{j} , is of the form $[i^{\pm}, \epsilon(j)]$. The first component of the label indicates the previous node in the path and whether flow moves from \underline{i} to \underline{j} . ($i+$), or from \underline{j} to \underline{i} , ($i-$). The second component of the label is the amount by which flow on arc (i, j) may be changed without increasing kilter numbers of other arcs in the path.

In attempting to find a circuit connecting labeled nodes from \underline{p} to \underline{q} specific labeling rules are followed. The table below describes these rules for an arbitrary out-of-kilter arc (i, j) . The first four rules are for a forward arc and the last four are for a reverse arc.

Table 1: Labeling Rules for Out-of-Kilter Algorithm

Net Cost	Flow Condition	Label	
		i^+	$\epsilon(j)$
$\bar{C}_{ij} < 0$	$f_{ij} < k_{ij}$	i^+	$\text{Min}[\epsilon(i), k_{ij} - f_{ij}]$
$\bar{C}_{ij} < 0$	$f_{ij} \geq k_{ij}$		NO LABEL
$\bar{C}_{ij} > 0$	$f_{ij} \geq l_{ij}$		NO LABEL
$\bar{C}_{ij} \geq 0$	$f_{ij} < l_{ij}$	i^+	$\text{Min}[\epsilon(i), l_{ij} - f_{ij}]$
$\bar{C}_{ji} < 0$	$f_{ji} > k_{ji}$	i^-	$\text{Min}[\epsilon(i), f_{ji} - k_{ji}]$
$\bar{C}_{ji} < 0$	$f_{ji} \leq k_{ji}$		NO LABEL
$\bar{C}_{ji} \geq 0$	$f_{ji} > l_{ji}$	i^-	$\text{Min}[\epsilon(i), f_{ji} - l_{ji}]$
$\bar{C}_{ji} \geq 0$	$f_{ji} \leq l_{ji}$		NO LABEL

If a labeled circuit connecting nodes p and q has been formed as a result of the labeling rules, the maximum amount by which flow along this circuit may be changed without increasing kilter numbers along the circuit or violating conservation of flow, has been determined by ϵ . Flow along this circuit is therefore augmented by ϵ .

All labels are now erased and arc (p, q) examined again. If it is still out-of-kilter, the labeling procedure is repeated and

flow augmented if a labeled circuit from \underline{p} to \underline{q} is found. If no circuit is found and if the problem is feasible, it will be necessary to change the prices on all the unlabeled nodes. If arc (p, q) is in-kilter, the next out-of-kilter arc is searched out and the procedure repeated. If all the arcs are in-kilter, the existing flow pattern is optimal. Figure 2 is a flow chart of the labeling routine for the OKA and Figure 3 is a flow chart of the flow augmentation routine.

The Pricing Routine: If in attempting to find a labeled circuit for an out-of-kilter arc (a, b) , all the nodes adjacent to labeled nodes have been scanned and yet node \underline{a} has not been labeled, then a non-breakthrough has occurred.

Two possibilities give rise to such a situation. For one or more arcs (i, j) , either

1. The net arc costs, \bar{C}_{ij} , do not provide enough incentive to increase flow, i. e.,

$$\bar{C}_{ij} > 0 \text{ and } l_{ij} \leq f_{ij} < k_{ij} \quad (19)$$

- or
2. No flow augmentation is possible without violating a bound on the arc, i. e.,

$$\bar{C}_{ij} \leq 0 \text{ and } f_{ij} = k_{ij} \quad \text{or} \quad (20a)$$

$$\bar{C}_{ji} \leq 0 \text{ and } f_{ji} = l_{ji}. \quad (20b)$$

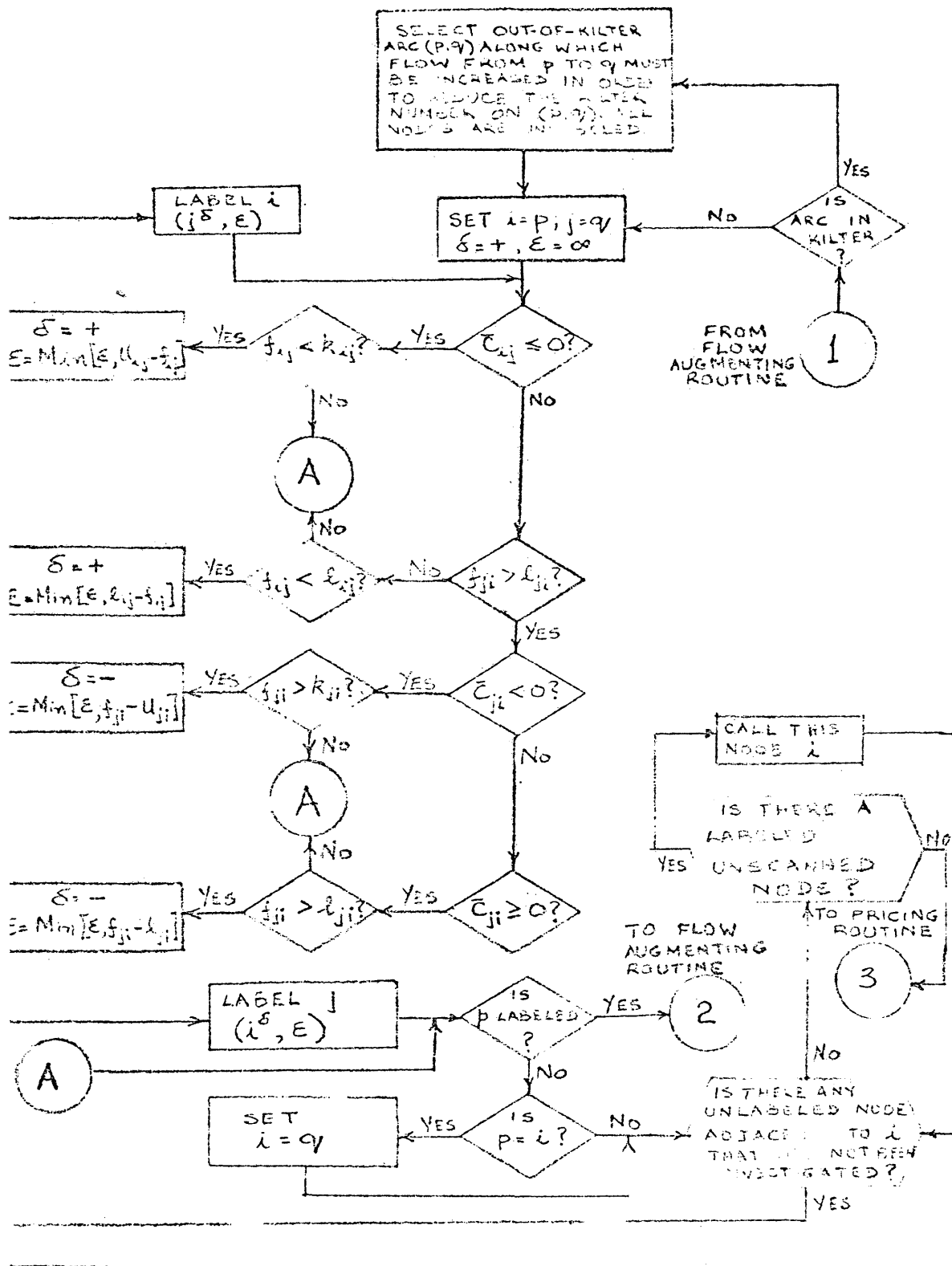


Figure 2. Labeling Routine of the Out-of-Kilter Algorithm

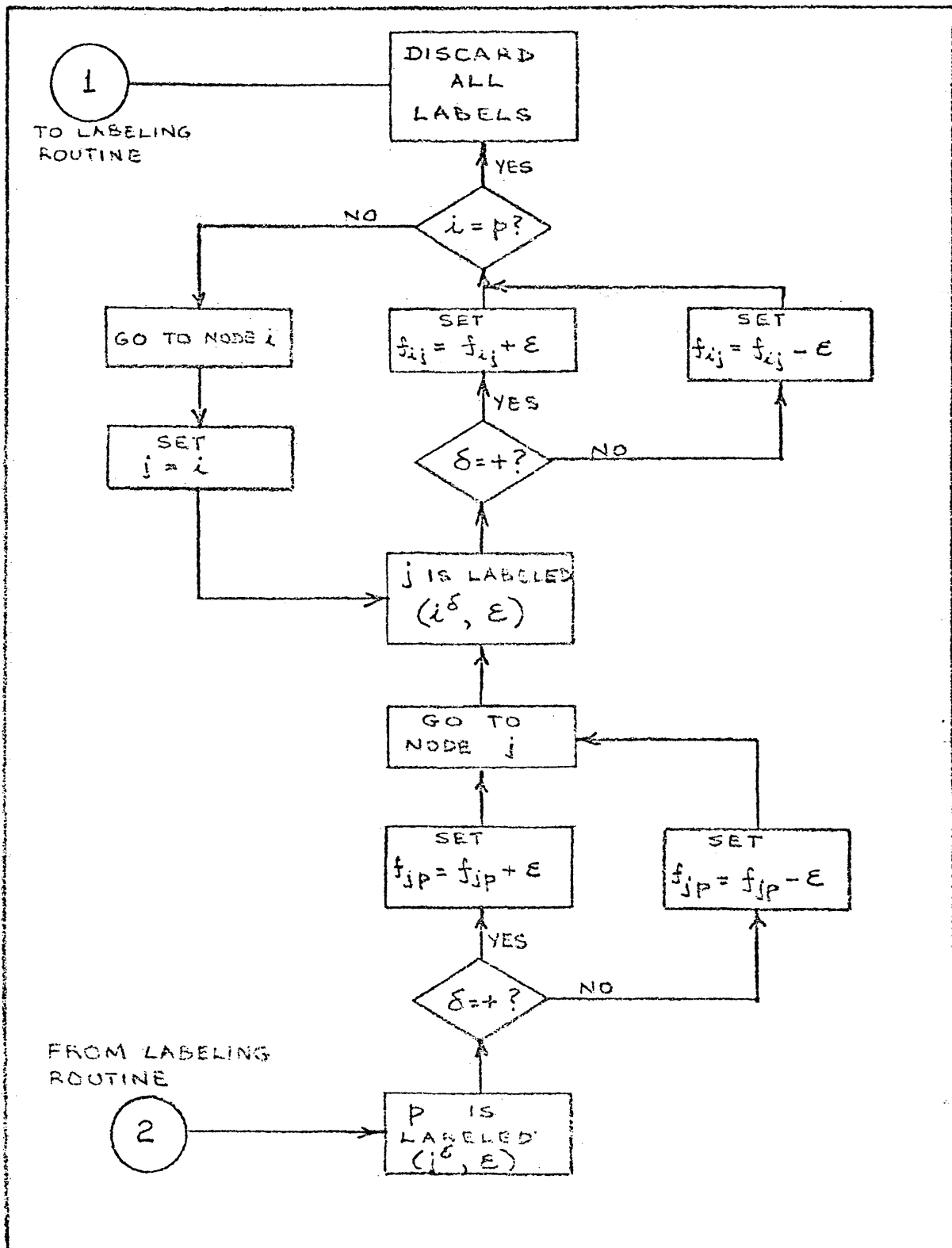


FIGURE 3: FLOW AUGMENTATION ROUTINE FOR THE OUT-OF-KILTER ALGORITHM.

If case 2 occurs for all arcs (i, j) such that $i \in [\text{labeled nodes}]$ and $j \in [\text{unlabeled nodes}]$, then the problem is infeasible. If case 1 occurs for any arc (i, j) , then the node prices on the unlabeled nodes are systematically changed to provide more incentive.

If the problem is feasible and calls for a change in node prices, suppose labeling stopped at some labeled node \underline{i} from which no more labels could be assigned. Let X be the set of all labeled nodes. Then X^c is the set of all unlabeled nodes. Define $A' = (X, X^c)$ as the cut separating the labeled from the unlabeled nodes. Define A_1 and A_2 as subsets of A' such that

$$\begin{aligned} A_1 &= [(i, j) \mid i \in X, j \in X^c, \bar{C}_{ij} > 0, f_{ij} \leq k_{ij}] \text{ and} \\ A_2 &= [(i, j) \mid i \in X, j \in X^c, \bar{C}_{ji} < 0, f_{ji} \geq l_{ji}] \end{aligned} \quad (21)$$

$$\text{If } \delta_1 = \min_{(i, j) \in A_1} [\bar{C}_{ij}] \text{ and}$$

$$\delta_2 = \min_{(i, j) \in A_2} [-\bar{C}_{ji}] \text{ then} \quad (22)$$

$\delta = \min(\delta_1, \delta_2)$ where $\delta_i = \infty$ if A_i is empty, is the minimum positive quantity which if added to π_j , $j \in X^c$ will guarantee that the labeled set X will increase by at least one node if the labeling routine is repeated for out-of-kilter arc (a, b) .

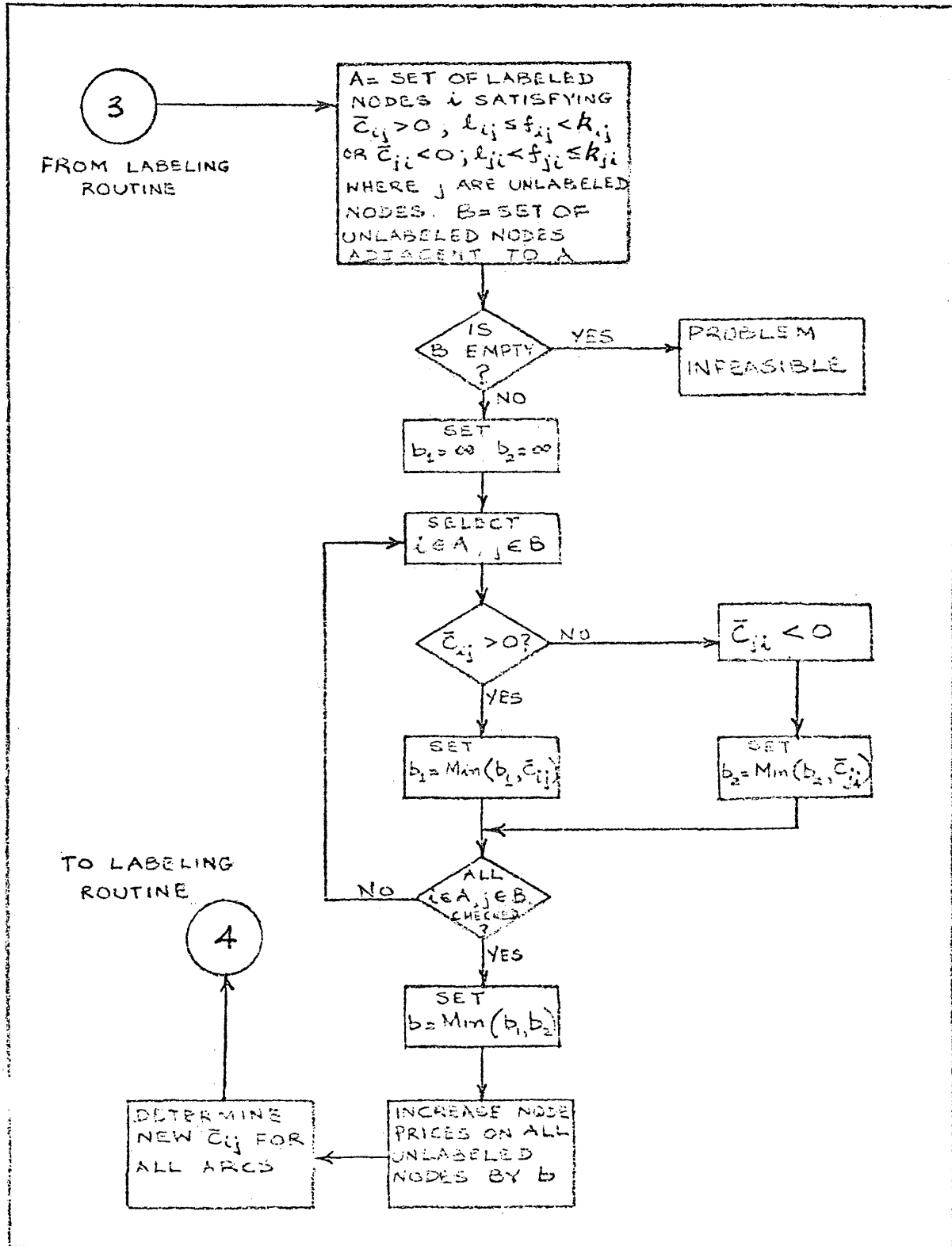


FIGURE 4: PRICING ROUTINE FOR THE OUT-OF-KILTER ALGORITHM.

This pricing procedure is analogous to finding the minimum increase in the price of a commodity at nodes j ($j \in X^C$) such that a commodity may be shipped to at least one more node after a non-breakthrough has occurred. The arc or set of arcs that yielded the minimum value of δ will, as a result of the node price increase, have their net arc costs, \bar{C}_{ij} , changed to zero. Hence, if the labeling procedure is repeated, for these arc(s) their nodes j will now be labeled where they were not before the pricing procedure. Figure 4 is a flow chart of the pricing routine.

Termination and Optimality: Repeated application of the Labeling, Flow Augmentation and Pricing Routines will ultimately bring an out-of-kilter arc into kilter in a finite number of iterations provided the problem is feasible. Each time a breakthrough occurs, the kilter number on each arc in the circuit is decreased as a result of augmenting the flow. Each time a non-breakthrough occurs, as a result of the pricing routine at least one more node is labeled and since there are only a finite number of nodes, we must eventually breakthrough if the problem is feasible. So, in a finite number of interactions, either all arcs will be brought into kilter which is the condition for optimality or the problem is infeasible. A formal proof of the finiteness of the OKA and of optimality is given by Ford and Fulkerson (1962, pp. 162-169).

FORMULATING THE SHAFT LOCATION PROBLEM
AS A NETWORK

The shaft location problem is solved by formulating it as a minimal cost network flow problem. A modification of the out-of-kilter algorithm (OKA) is used repeatedly to converge to a good (hopefully optimal) solution. Figure 5 shows how the shaft location problem is formulated as a network. The nodes on the left side represent the blocks that the deposit is divided into and the nodes on the right side represent the blocks that are potential shaft locations.

The arcs from the source node to each block node \underline{i} have lower bounds of 0, upper bounds of $W_{\underline{i}}$, and cost of 0. The transportation arcs from each block node \underline{i} to each potential shaft node \underline{j} have lower bounds of 0, upper bounds of $W_{\underline{i}}$ and costs of $t_{\underline{ij}}$.

The shaft arcs from each potential shaft node \underline{j} to the sink node, t , have lower bounds of 0, upper bounds of $\sum_{\underline{i}} W_{\underline{i}}$, and costs of $C_{\underline{jt}}$. The value of $C_{\underline{jt}}$ vary from iteration to iteration according to rules specified elsewhere. Essentially, $C_{\underline{jt}}$ is the

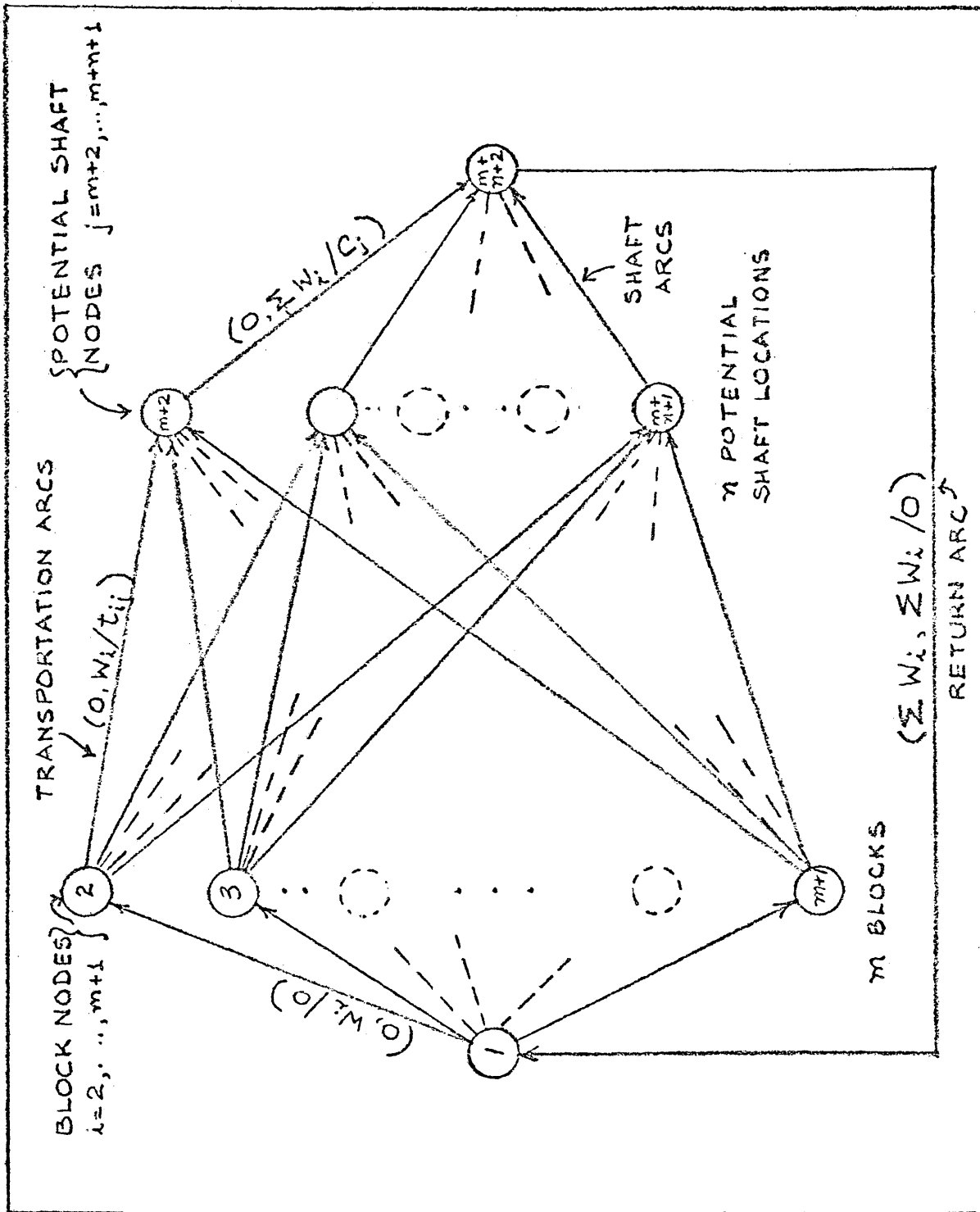


FIGURE 5: THE SHAFT LOCATION PROBLEM AS A NETWORK.

capital cost of shaft \underline{j} per unit of flow along arc (j, t) . Since the flow along (j, t) is unknown except at the end of the algorithm (it is one of the variables to be determined), the process of changing C_{jt} is an approximate one based on the flow along (j, t) in the previous iteration. The return arc (t, s) from the sink to the source has a lower bound and an upper bound of $\sum_i W_i$ with a cost of 0. This ensures that all the tonnage in the blocks is mined and sent to the surface. Flows f_{ij} , along transportation arcs (i, j) indicate (at optimality) that material from block \underline{i} will be transported and hoisted up shaft \underline{j} .

METHOD OF SOLUTION

The network problem in Fig. 5 may be formulated mathematically as the following primal linear program:

$$\text{Minimize } Z = \sum_{i=2}^{(m+1)} \sum_{j=m+2}^{(m+n+1)} t_{ij} f_{ij} + \sum_{j=m+2}^{m+n+1} C_{jt} f_{jt}$$

$$\text{Subject to } \sum_{j=1}^t (f_{ij} - f_{ji}) = 0; \quad i=1, \dots, t; \quad i \neq j$$

$$f_{ij} \geq l_{ij}; \quad \forall i, j \quad i \neq j$$

$$-f_{ij} \geq -U_{ij}; \quad \forall i, j \quad i \neq j$$

$$f_{ij} \geq 0$$

PROBLEM IV

where the variables are as defined earlier and the bounds are as given in Figure 5.

For the first iteration with the OKA,

$$C_{jt} = P_j / \left(\sum_{i=2}^{m+1} W_i \right) \quad (23)$$

The minimal cost solution obtained by solving Problem IV with the OKA has two possible forms.

Either

1. All flow goes through only one shaft arc (k, t).

$$\text{Then } f_{kt} = U_{kt} = \sum_i W_i \Rightarrow \bar{C}_{kt} \leq 0; \quad (24a)$$

$$0 = f_{jt} < U_{jt} = \sum_i W_i \Rightarrow \bar{C}_{jt} \geq 0; \quad \forall_j \neq k \quad (24b)$$

or

2. No shaft arc has flow at its upper bound.

$$\text{Then } 1_{jt} \leq f_{jt} < U_{jt} \Rightarrow \bar{C}_{jt} \geq 0; \quad j = m+2, \dots, m+n+1 \quad (25)$$

If the first type of solution occurs, the solution is optimal.

It is not profitable to increase flow along any of the other arcs with $f_{jt} = 0$ and $\bar{C}_{jt} = C_{jt} + \pi_j - \pi_t \geq 0$. Any increase in flow along those arcs will necessarily cause a decrease in flow along (k, t) and such an action does not contribute to a lessening of the total cost since $\bar{C}_{kt} \leq 0$. Note that $C_{kt} = P_k / \sum_i W_i$ is a correct estimate of the unit capital cost of current flow along (k, t).

If the second type of solution occurs, the capital costs per unit flow, $C_{jt} = P_j / \sum_i W_i$ are underestimated since $f_{jt} < \sum_i W_i = U_{jt}$. Let the arcs (j, t) be divided into two subsets, S_1 and S_2 such that $(j, t) \in S_1 \Leftrightarrow f_{jt} > 0$ and

$$(j, t) \in S_2 \Leftrightarrow f_{jt} = 0$$

The costs C_{jt} of the shaft arcs are corrected to reflect current flow:

$$C_{jt} = P_j / f_{jt} \quad \text{if } (j, t) \in S_1.$$

This will raise the values of C_{jt} and consequently of \bar{C}_{jt} .

$$\text{Hence } \bar{C}_{jt} > 0; (j, t) \in S_1$$

The fact that $(j, t) \in S_1 \Rightarrow f_{jt} > 0$ makes the arc out-of-kilter for each $(j, t) \in S_1$. This is corrected by finding $\text{MAX} = \text{Max} [\bar{C}_{jt} \mid (j, t) \in S_1]$ and changing the value of the price on the sink node, t , such that

$$\pi_t = \pi_t + \text{MAX}$$

Now, for $(j, t) \in S_1$, $f_{jt} > 0$ and the new value of

$$\bar{C}_{jt} = C_{jt} + \pi_j - \pi_t \leq 0$$

Most of all of these arcs are still out-of-kilter since

$\bar{C}_{jt} < 0 \Rightarrow f_{jt} = U_{jt}$ is the in-kilter condition. The original value of $U_{jt} = \sum_i W_i$ was just the most flexible estimate of the upper bound on flow in the shaft arcs. No information was available to make a better estimate. By now making $U_{jt} = f_{jt}$, all arcs $(j, t) \in S_1$ are brought into kilter. This new value of U_{jt} is now a lower bound on the future estimates of the upper bounds.

The arcs $(j, t) \in S_2$ are such that $f_{jt} = 0$ and $C_{jt} = P_j / \sum_i W_i$. The action of raising the value of π_t could make the values of $\bar{C}_{jt} < 0$. This out-of-kilter status is corrected by making the arc cost $C_{jt} = -\pi_j + \pi_t$ such that the net arc cost

$$\bar{C}_{jt} = C_{jt} + \pi_j - \pi_t = 0$$

This introduction of fictitious net arc costs keeps the arcs $(j, t) \in S_2$ in-kilter and also permits future flow along them if node j ever gets labeled. In this way, shaft arcs that do not have any flow are never saddled with a positive cost that will inhibit any chances of flow in the future.

If the network that was submitted to the OKA had been such that $[U_{jt}] \equiv [f_{jt}]$ (where f_{jt} is the flow along arc (j, t) as determined after the first iteration) and if

$$C_{jt} = \begin{cases} P_j / f_{jt} & ; (j, t) \in S_1 \\ -\pi_j + \pi_t & ; (j, t) \in S_2 \end{cases} \quad (26)$$

then the solution obtained would have been identical to that obtained at the end of the first iteration.

The solution obtained after the first iteration is a minimal cost solution for Problem IV. Changing the network by changing the costs according to (26) and solving the new network with the OKA in succeeding iterations give minimum cost solutions for the

changed network. These solutions need not be the optimal solution for the shaft location problem. These repeated iterations are essentially a repeated revision of the estimates of upper bounds on flow (U_{jt}) and of the capital cost coefficients (C_{jt}) along the shaft arcs.

The dual problem to the primal is as follows:

$$\text{Maximize } V = \psi_{ts} l_{ts} - \sum_{i,j} \phi_{ij} U_{ij} - \sum_j \phi_{jt} U_{jt}$$

$$\text{Subject to } -\pi_i + \pi_j + \psi_{ij} - \phi_{ij} \leq t_{ij};$$

$$\forall (i, j) \in [\text{transportation arcs}]$$

$$-\pi_j + \pi_t + \psi_{jt} - \phi_{jt} \leq C_{jt}; \forall j \in [S_1 \cup S_2]$$

$$\psi, \phi \geq 0$$

$$l_{ij} = 0; \forall (i, j) \neq (t, s)$$

π unrestricted

PROBLEM V

The objective function of the dual shows that the value of Max V may be decreased if U_{ij} or U_{jt} are increased if the corresponding dual variable ϕ is positive. The U_{ij} for the transportation arcs are fixed by the formulation of the problem since $U_{ij} = W_i$. The U_{jt} are essentially estimates and as explained above are repeatedly adjusted such that

$$[U_{jt}] \equiv [f_{jt}] \quad \forall (j, t) \in S_1 \quad (27)$$

The complementary slackness conditions satisfied by the solution are:

$$f_{ij} (t_{ij} + \pi_i - \pi_j + \psi_{ij} - \phi_{ij}) = 0 \quad (28)$$

$$\psi_{ij} (f_{ij} - l_{ij}) = 0 \quad (29)$$

$$\phi_{ij} (U_{ij} - f_{ij}) = 0 \quad (30)$$

$$\pi_i \left[\sum_i (f_{ij} - f_{ji}) \right] = 0 \quad (31)$$

$$i, j = 1, \dots, t; i \neq j$$

From (27), (29) and (28), for $(j, t) \in S_1$,

$$f_{jt} = U_{jt} \implies \psi_{jt} = 0 \text{ and}$$

$$\phi_{jt} = -C_{jt} + \pi_t - \pi_j = -\bar{C}_{jt} \geq 0 \quad (32)$$

For $(j, t) \in S_1$, $\phi_{jt} = -\bar{C}_{jt} \geq 0$ and from the objective function of the dual (Problem V), it would pay to increase U_{jt} and permit an increase in flow along (j, t) . The subsequent solution would have a lower value of Max V than that obtained from the previous iteration.

This procedure is repeated in each iteration and is in effect, a repeated upward adjustment of the upper bound of shaft arcs $(j, t) \in S_1$. During all this, the arcs $(j, t) \in S_2$ always have their arc costs adjusted such that $\bar{C}_{jt} = 0$, thus permitting flow along it at no penalty. If any flow is initiated along it, the arc then belongs to S_1 and follows the procedure described earlier.

If after any iteration, it is found that

$$\bar{C}_{jt} = 0 = \phi_{jt} ; \forall (j, t) \in S_1 \quad (33)$$

then there would be no change in the value of the objective function (Problem V) by increasing any upper bounds and hence the solution at that point would be optimal; i. e., the value of Max V cannot be reduced any further.

The Optimal Mine Shaft Location Algorithm

- Step 1: Read number of blocks (m), number of potential shaft locations (n), unit transportation cost, capital cost (P_j) associated with each shaft location j.
- Step 2: Set up network as in Figure 5. Set $U_{jt} = \sum_i W_i$ and $C_{jt} = P_j / \sum_i W_i$.
- Step 3: Solve network for minimum cost flow with out-of-kilter algorithm (NETFLØ).
- Step 4: For all shaft arcs (j, t); $j = m+1, \dots, n$, $t = 1, \dots, m$ calculate $C_{jt} = P_j / f_{jt} \mid f_{jt} > 0$ or $C_{jt} = C_{jt} \mid f_{jt} = 0$.
- Find MAX = Max [$\bar{C}_{jt} \mid f_{jt} > 0$]
- where $\bar{C}_{jt} = C_{jt} + \pi_j - \pi_t$

Step 5: Change sink node price: $\pi_t = \pi_t + \text{MAX}$

For all shaft arcs (j, t) ; $j = m+1, \dots, t-1$

$$C_{jt} = -\pi_j + \pi_t \mid f_{jt} = 0 \text{ or}$$

$$C_{jt} = C_{jt} \mid f_{jt} > 0$$

$$\text{Calculate } \bar{C}_{jt} = C_{jt} + \pi_j - \pi_t.$$

Step 6: SORT and renumber shaft arcs from low to high

$$\bar{C}_{jt} \ni \bar{C}_{j,t} \leq \bar{C}_{j+1,t}$$

Step 7: Augment flow along all possible labeled circuits (LABEL).

If there is any change in flow, check to see if any shaft

arc has $\bar{C}_{jt} > 0$ and $f_{jt} > l_{jt}$. (Note: \bar{C}_{jt} is

recalculated after every flow change according to rule

in Step 4). If such an arc exists, go to Step 8. If not

go to Step 4. If no change in flow, go to Step 8.

Step 8: If $\text{Min} [\bar{C}_{jt}] \geq 0$, go to Step 11. Name out-of-kilter arc,

AOK = (k, t) where $\bar{C}_{kt} = \text{Min} [\bar{C}_{jt}]$, $j = m+1, \dots, t-1$.

Step 9: Label nodes for out-of-kilter arc, AOK. Nodes are

divided into labeled and unlabeled sets. (LABEL1).

Step 10: Determine minimum increase in price of unlabeled nodes

so as to increase labeled set by at least one. Increase

price on the unlabeled nodes by this amount. Go to

Step 6.

Step 11: Check to see if any shaft arc has $\bar{C}_{jt} > 0$ and $f_{jt} > l_{jt}$.

If so, make out-of-kilter arc AOK = (j, t) $\left| \bar{C}_{jt} > 0, \right.$

$f_{jt} > l_{jt}$ Go to Step 9. If not, optimal solution has been reached. Print output and exit.

THE COMPUTER PROGRAM

Introduction

The shaft location algorithm has been programmed in FORTRAN IV and run on an IBM/ 360/ 65J computer. For ease in programming the program has been divided into a number of subroutines. Figure 6 is a flow chart of the computer program. The operation of each subroutine is described below.

Description of Subroutines

Subroutine DIST: Subroutine DIST is not an essential part of the algorithm. In this program, DIST sets up the network by defining arcs in terms of the nodes connected and allocating arc parameters such as cost (C_{ij}), upper bound (U_{ij}) and lower bound (l_{ij}). In order to calculate the transportation arc costs, it also calculates the rectilinear distances between block nodes, i , and the nodes, j , that represent the potential shaft locations, using coordinates that are read in. Alternately, DIST could be bypassed and the whole network with arc parameters read in as input.

Specifically, DIST does the following:

- 1 Read in block coordinates and block weight.
2. Read in block numbers for each potential shaft location.

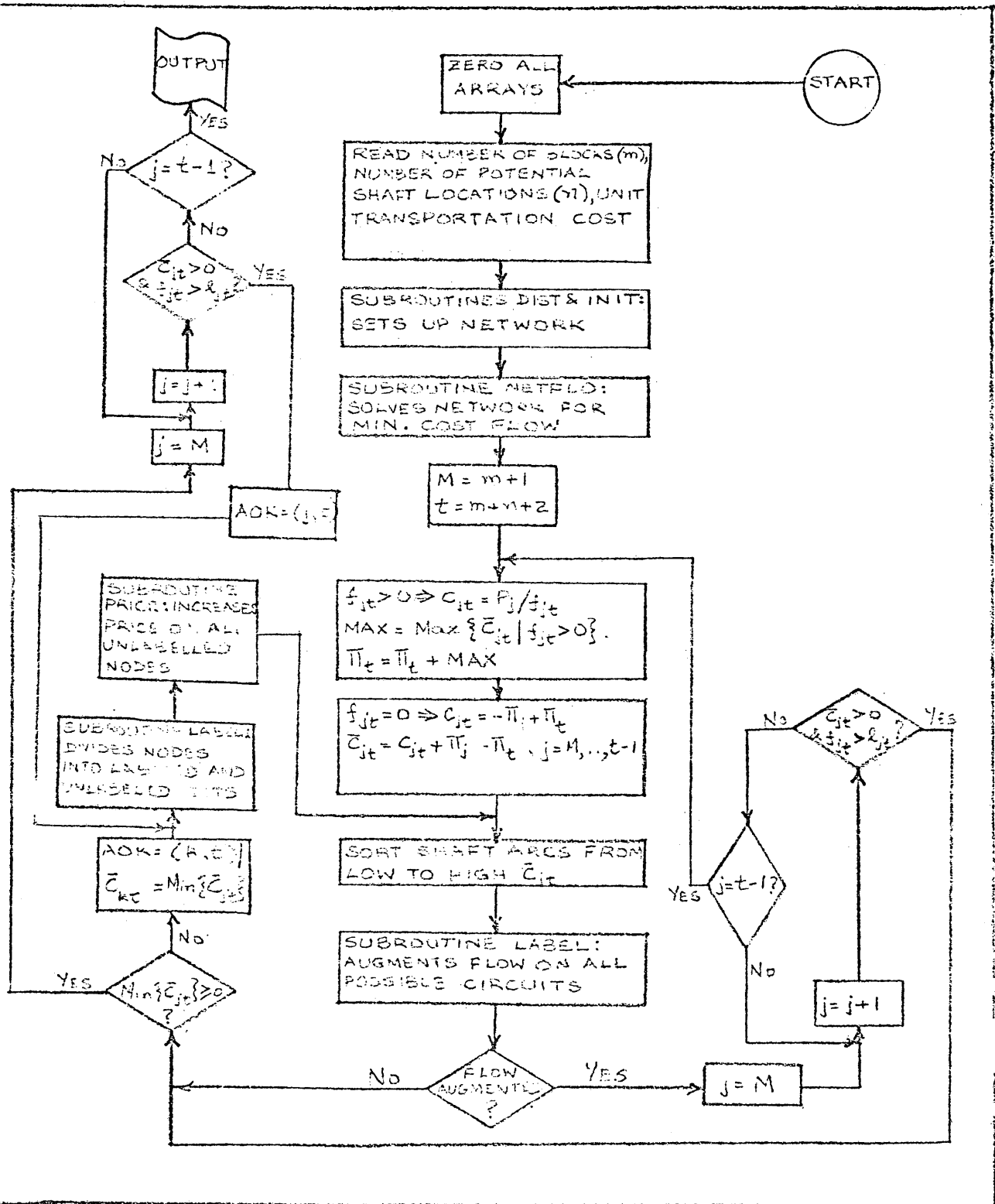


FIGURE 6: FLOWCHART OF THE MINE SHAFT LOCATION ALGORITHM.

3. Calculate rectilinear distances from each block to every block that is a potential shaft location.
4. Define all the arcs in the network in terms of the nodes that each arc connects.
5. Set the lower bound of all arcs to zero.
6. For the arcs from the source to each block \underline{i} (figure 5), set the upper bound = tonnage in block \underline{i} = $W_{\underline{i}}$.
7. For the transportation arcs from block \underline{i} to potential shaft location \underline{j} , set the upper bound = $W_{\underline{i}}$ and the cost = (transportation cost per unit distance) \times rectilinear distance from block \underline{i} to block \underline{j} .
8. For the shaft arcs from potential shaft location \underline{j} to sink \underline{t} , set upper bound = $\sum_i W_{\underline{i}}$, sum of the weights of all the blocks and read in the capital cost of sinking a shaft at block \underline{j} , $P_{\underline{j}}$.

Subroutine INIT: INIT prepares the network for submission to the shaft location algorithm. It sets flow along all the arcs = 0 and sets all node prices = 0. It then sets the upper bound for the shaft arcs, $U_{\underline{j}\underline{t}} = \sum_i W_{\underline{i}}$. The cost parameter for the shaft arcs, $C_{\underline{j}\underline{t}}$, is set such that $C_{\underline{j}\underline{t}} = P_{\underline{j}}/U_{\underline{j}\underline{t}}$. Finally the upper and lower bounds of the return arc is set equal to $\sum_i W_{\underline{i}}$. These actions

ensure that if the network is now submitted to the OKA, a minimum cost flow will be obtained and the flow along the return arc will equal $\sum_i W_i$.

Subroutine NETFLO: This is basically the out-of-kilter algorithm as described earlier. NETFLO essentially does the following:

1. Scan arc conditions and find an out-of-kilter arc. If there is no out-of-kilter arc, the existing flow pattern gives a minimal cost solution.
2. Start labeling nodes to find a labeled circuit that includes the out-of-kilter arc. If a labeled circuit is found, increment the maximum feasible flow along the circuit. If not, go to 3.
3. A non-breakthrough has occurred. Determine the minimum increment of node prices for the unlabeled nodes that will permit at least one more node to be labeled. Increase the prices on all the unlabeled nodes by this amount. Go to 1

Subroutine SØRT: SØRT first stores all shaft arc parameters in temporary storage and then goes to subroutine RSØRT. RSØRT is a modified binary sort routine that is used to sort the values of \bar{C}_{ij} for all the shaft arcs, producing an array that increases from most negative to most positive \bar{C}_{ij} . After

sorting, upon return to SØRT, the shaft arcs are renumbered so that they are in order of increasing \bar{C}_{ij}

Subroutine LABEL: LABEL is essentially the same as NETFLO without the pricing routine and with some modifications. The primary difference with NETFLO is that where NETFLO goes to the pricing routine whenever a non-breakthrough occurs, LABEL exhausts all possibilities of incrementing flow along all possible labeled circuits containing out-of-kilter arcs. Note that the only out-of-kilter arcs that LABEL encounters are the shaft arcs $(j, t) \ni \bar{C}_{jt} < 0$ and $f_{jt} < U_{jt}$. LABEL also calls RECAP after every increment of flow such that the cost coefficients C_{jt} are always the closest estimate of the prorated capital cost as is possible.

Subroutine RECAP: One of the keys to formulating the shaft location problem (the fixed-charge transportation problem) as a network is that when the final solution is reached, the arcs that contribute the fixed charge (a subset of the shaft arcs) will have their effective arc costs such that their fixed charge (capital cost of locating a shaft at block $j = P_j$) will be prorated according to the flow that passes along that arc. Thus, for the optimal solution to the shaft location problem, for those shaft arcs that are selected, $C_{jt}^0 = P_j / f_{jt}^0$ where the superscript 0 denotes the values at optimality. This is so that in the objective function of the primal problem:

$$\text{Min } Z = \sum_{\substack{\text{transp.} \\ \text{arcs}}} C_{ij} f_{ij} + \sum_{\substack{\text{shaft} \\ \text{arcs}}} C_{jt} f_{jt}$$

if C_{jt} is computed as above, the problem remains linear and may be solved by the OKA. RECAP therefore calculates the new values of C_{jt} after every change in flow in LABEL. It is not a continuous change. Summarizing, RECAP calculates $C_{jt} = P_j/f_{jt}$ for each shaft arc with $f_{jt} > 0$. If $f_{jt} = 0$, then $C_{jt} = -\pi_j + \pi_t$.

Subroutine LABEL1: LABEL1 is the labeling routine of NETFLO. There is no flow augmentation routine and no pricing routine. Given an out-of-kilter arc, LABEL1 will scan all nodes in an effort to find a labeled circuit containing the arc. As the program is set up however, LABEL1 will never find such a circuit as its input is always an out-of-kilter arc that will lead to a non-breakthrough. However, this labeling procedure is necessary as an input to subrouting PRICE.

Subroutine PRICE: PRICE determines the minimum increase in the node prices of unlabeled nodes that will permit at least one more node to be labeled. The out-of-kilter arc being considered is the same as that input to LABEL1. Having determined this increment in node price, δ , by the rules given by (21) and (22), PRICE then adds this δ to the old node prices π_i where i belongs to the set of unlabeled nodes.

Subroutine TCOST: TCOST calculates the total cost of the circulation in the network. The total cost equals the sum of the transportation and the capital costs. The transportation costs,

$$TC = \sum_{i,j} C_{ij} f_{ij}; i \in (\text{block nodes}), j \in (\text{shaft nodes})$$

The capital costs, $CC = \sum_j P_j y_j; j \in (\text{shaft nodes})$ and

$$y_j = \begin{cases} 0, & \text{if } f_{jt} = 0 \\ 1, & \text{if } f_{jt} > 0 \end{cases}$$

The total cost = TC + CC.

The Main Program

This is essentially a control program that makes sure that each subroutine is performed in the proper order. It carries out checks to determine the status of each arc and also to determine if the solution at the end of every iteration can be improved. If it can, the program initiates another iteration. If it cannot, (all $\bar{C}_{jt} = 0$), the solution is optimal and the program outputs the solution. The logic used in determining the sequence of subroutines is given in the flow chart of the algorithm (Figure 6) and is described earlier.

COMPUTATIONAL RESULTSSample Problems

Problem 1: The algorithm has been used to determine the optimal number and location of shafts for the fictional deposit described by Figure 7. The cost of shaft sinking was assumed to be directly related to only the depth of sinking and estimated capital costs for blocks 8-12, 14-19, 24-29, and 33 were thereby determined. Transportation was assumed to be rectilinear with a unit transportation cost of \$ 20/1000 tons transported over 1000 ft. All input data are given in Table 2.

Problem 2: The second problem is essentially similar to the first except that the capital cost for block 15 was changed from \$ 900,000 to \$ 400,000 and that of block 28 from \$ 510,000 to \$210,000.

Problem 3: The third problem is the same as problem 2 except that the capital cost of block 16 was changed from \$650,000 to \$ 50,000.

Results

Problem 1: The results are included as a sample output in Appendix II. The solution to problem 1 is:

1. The number of hoisting shafts = 2.
2. The shafts should be located in blocks 11 and 29.
3. The total tonnage that will pass through the shaft in block 11 = 29×10^6 tons; the corresponding "capacity" of the shaft in block 29 = 11×10^6 tons.
4. The expected total cost of transportation plus capital shaft sinking costs = \$ 3,110,000.
5. The minimal cost transportation network describing the shaft to which material from each block should be transported is described in the sample output given in Appendix II.

Problem 2:

1. The number of hoisting shafts = 2.
2. The location of the shafts are in blocks 15 and 28.
3. The "capacity" of each shaft for the life of the mine is 24×10^6 tons through the shaft in block 15 and 16×10^6 tons through the shaft in block 28.
4. The expected total cost = \$ 2,410,000.
5. The minimal cost transportation network is not included in this thesis.

Table 2: Input Data

Number of blocks = 40

Number of potential shaft locations = 18

Unit transportation cost = \$20/1000 tons transported 1000 ft.

Total tonnage = 40000×10^3 tons.

<u>Block</u>	<u>Coordinates, ft.</u>		<u>Capital Cost, \$</u>
	<u>X</u>	<u>Y</u>	
1	1500	6500	-
2	2500	"	-
3	3500	"	-
4	4500	"	-
5	5500	"	-
6	6500	"	-
7	1500	5500	-
8	2500	"	1,050,000
9	3500	"	800,000
10	4500	"	810,000
11	5500	"	430,000
12	6500	"	400,000
13	1500	4500	-
14	2500	"	800,000
15	3500	"	900,000
16	4500	"	650,000
17	5500	"	600,000
18	6500	"	520,000
19	7500	"	400,000
20	8500	"	-

Table 2: Input Data (Cont.)

<u>Block</u>	<u>Coordinates, ft.</u>		<u>Capital Cost, \$</u>
	<u>X</u>	<u>Y</u>	
21	9500	4500	-
22	1500	3500	-
23	2500	"	-
24	3500	"	850,000
25	4500	"	880,000
26	5500	"	720,000
27	6500	"	650,000
28	7500	"	510,000
29	8500	"	380,000
30	9500	"	-
31	1500	2500	-
32	2500	"	-
33	3500	"	700,000
34	4500	"	-
35	5500	"	-
36	6500	"	-
37	7500	"	-
38	8500	"	-
39	9500	"	-
40	1500	1500	-

Problem 3:

1. The number of shafts = 2.
2. The location of the shafts are in blocks 16 and 28.
3. The "capacity" of each shaft is 29×10^6 tons through the shaft in block 16 and 11×10^6 tons through the shaft in block 28.
4. The expected total cost = \$ 2,220,000.
5. The minimal cost transportation network shows that material from blocks 19, 20, 21, 27, 28, 29, 30, 36, 37, 38 and 39 are transported to the shaft in block 28 while the material from the other blocks goes to the shaft in block 16.

DISCUSSION

The algorithm to determine the optimal number and location of mine hoisting shafts presented in this thesis has not yet been proven to be optimal. At present, the best that can be done is to show that the value of the objective function decreases with every iteration. The difficulty in proving optimality is in showing that the best solution that is obtained from the sequence of minimal cost solutions of the network (Fig. 5) is actually the optimal solution to the fixed-charge shaft location problem. It is possible that Balas' duality theory (Balas, 1967) as applied to the fixed-charge problem could be used in proving the optimality of the shaft location algorithm. The results of test problems tried so far have all been optimal.

FURTHER APPLICATIONS

As mentioned earlier, the shaft location problem is a special case of the fixed-charge problem with linear transportation costs. The algorithm developed in this thesis can be used for any fixed-charge problem with linear transportation costs provided the transportation network is defined. This algorithm is unique in that a network approach is used to find a solution to a fixed-charge problem.

The simplicity of the algorithm will permit the user to determine optimal solutions to numerous problems that were hitherto only done by sophisticated mathematicians or done not at all. Examples of such problems are:

1. Optimal location of any facility where a transportation network is defined and a capital cost is associated with each facility; location of warehouses, distribution centers, survival shelters, railway stations, etc. With respect to the mining industry, the optimal location of shafts, cleaning plants, offices, shops, etc. may be determined. In combination with Sharp's algorithm (Sharp, 1972) for the determination of the

optimal transportation network in a coal mine, the shaft location algorithm may be used in the determination of the complete optimal layout (surface and underground) of an underground, shaft accessed coal mine.

2. With proper formulation of the problem, this algorithm could be used to solve the ventilation problem of determining the number and location of ventilation shafts in a mine and the fan capacity in each shaft.

Summarizing, the algorithm described in this thesis, may be used to solve any fixed-charge problem with linear transportation costs. Extensions of this work would be to develop a method of sensitivity analysis of the cost parameters. This is necessary since most of the cost parameters used as input in the fixed-charge problem are usually estimates subject to wide fluctuations.

Sensitivity analysis will permit best decisions to be made under the uncertainty of the cost parameters. With reference to the shaft location problem, sensitivity analysis could be used to determine the daily tonnage capacity for which the shaft should be designed. Since capital costs for sinking a shaft depend on the depth of sinking, the daily tonnage capacity and subsurface geological parameters, the capital costs of sinking a shaft is obviously a variable. Sensitivity analysis of the capital costs and repeated

application of the shaft location algorithm should permit the user to determine optimal shaft capacity in addition to the optimal number and location of the shafts.

APPENDIX I


```

0034 CAP(M)=0
0035 CONTINUE
0036 IF(DDI.EQ.C)GO TO 190
0037 CALL DIST(NBL,NLOC,UNIT,NODES,ARCS,SUMWT)
0038 GO TO 220
0039 190 READ(5,200)A,I(A),J(A),LO(A),HI(A),COST(A),CAPI(A)
0040 IF(A.LT.ARCS)GO TO 190
0041 200 FORMAT(3I3,4I10)
0042 220 ITER=0
0043 JTER=0
0044 MB=ARCS-NLOC
0045 MA=ARCS-1
0046 CALL INIT(NODES,ARCS,NLOC,SUMWT)
0047 1000 CALL NETFLO(NODES,ARCS,INFES,IBT,NBT,MB,MA,NLOC)
0048 GO TO 260
0049 50 CONTINUE
0050 230 CONTINUE
0051 ITER=ITER+1
0052 IF(.NOT.INFES)GO TO 240
0053 MAX=-1VF
0054 DO 330 A=MB,MA
0055 IA=I(A)
0056 JA=J(A)
0057 IF(FLOW(A).LE.0)GO TO 330
0058 COST(A)=CAP(A)/FLOW(A)
0059 C(A)=COST(A)+PI(IA)-PI(JA)
0060 IF(MAX.LT.C(A))MAX=C(A)
0061 330 CONTINUE
0062 PI(NODES)=PI(NODES)+MAX
0063 DO 700 A=MB,MA
0064 IA=I(A)
0065 JA=J(A)
0066 IF(FLOW(A).GT.0)GO TO 80
0067 COST(A)=-PI(IA)+PI(JA)
0068 80 C(A)=COST(A)+PI(IA)-PI(JA)
0069 IF(C(A).GT.0)GO TO 251
0070 70 CONTINUE
0071 71 CONTINUE
0072 CALL SORT(MB,MA,NLOC)
0073 IBT=IBT
0074 CALL LABEL(NODES,ARCS,IBT,NBT,ADK,CDK,MB,MA,NLOC)
0075 IF(IBT.GE.IBT)GO TO 275
0076 GO TO 270
0077 240 DO 250 A=MB,MA
0078 IF(C(A).GT.0.AND.FLOW(A).GT.0(A))GO TO 245
0079 250 CONTINUE
0080 251 CONTINUE
0081 JTER=1
0082 GO TO 260
0083 245 ADK=A
0084 GO TO 280
0085 270 DO 340 A=MB,MA

```

```

0086 IF(C(A).GT.0.AND.FLOW(A).GT.LO(A))GO TO.275
0087 340 CONTINUE
0088 GO TO 230
0089 275 IF(C(MB).GE.0)GO TO 240
0090 AUK=MB
0091 GO TO 200
0092 280 CONTINUE
0093 CALL LABELI(NODES,ARCS,IBT,NBT,AOK,COK,MB,MA)
0094 CALL PRICE(INFES,JIND,AOK,COK,NBT,ARCS,NODES)
0095 IF(JIND.EQ.0)GO TO 251
0096 GO TO 71
0097 260 IF(.NOT.INFES)PRINT 120
0098 120 FORMAT(2X,2CH2SOLUTION INFEASIBLE )
0099 106 FORMAT( 5X,23H THE NUMBER OF SHAFTS# ,15,5X,23H THE NUMBER OF RLOC
100 1KX# ,15,5X,41H THE NUMBER OF POTENTIAL SHAFT LOCATIONS# ,15,///)
0100 110 PRINT 115, NOOFS,ARCS
0101 115CFORMAT(//5X,17H NUMBER OF NODES#,15/5X,16H NUMBER OF ARCS#,15,///,1
19H ARC NODEI NODEJ,16X,4HCOST,18X,21H I,18X,2HLD,16X,4HFLOW,17X,
23HCAP,/)
0102 DO 97 M=1,ARCS
0103 PRINT 116, M,I(M),J(M),COST(M),HI(M),LO(M),FLOW(M),CAP(M)
0104 97 CONTINUE
0105 116 FORMAT(15,217,5(9X,111))
0106 PRINT 130, (M,PI(M),M=1,NODES)
0107 130 FORMAT(11HC,23X,24HMODE P12NODEC/(20X,14,10X,110) )
0108 PRINT 140, INT,NBT
0109 140CFORMAT(5X,30H THE NUMBER OF BREAKTHROUGHS IS,16//)
110H BREAKTHROUGHS IS,16//)
0110 CALL TCOST(ARCS,NLOC,NS)
0111 PRINT 106, NS, NBL,NLOC
0112 IF (ITER.LE.0) GO TO 50.
0113 IF(JTER.EQ.1) GO TO 1
0114 GO TO 270
0115 180 CONTINUE
0116 END

```

```

0001 SUBROUTINE INIT(NODES,ARCS,NLOC,SUMWT)
0002 COMMON /COST(I(1000)),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
      ICAP(1000),PI(100),X(100),Y(100),AA(100),AY(100),D(100,100),LOC(100
      2),WT(100)
0003 COMMON NR(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,AOK
0007 INTEGER XI,TEMP
0008 DO 10 A=1,ARCS
0009     FLOW(A)=0
0010     DO 20 A=1,NODES
0011         PI(A)=0
0012     MA=ARCS-1
0013     MB=ARCS-NLOC
0014     SUM=0.
0015     DO 30 A=MB,MA
0016         HI(A)=SUMWT
0017         B=CAP(A)/(HI(A))
0018         COST(A)=INT(B)
0019     CONTINUE
0020     LO(ARCS)=SUMWT
0021     HI(ARCS)=SUMWT
0022     RETURN
0023     END

```

T-1468

```

0001 SUBROUTINE DIST(NBL,NLOC,UNIT,NODES,ARCS,SUMWT)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
      ICAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
      2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,AOK
0007 INTEGER XI,TEMP
0008 INPUT COORDINATES AND WEIGHT OF EACH BLOCK
0009 INBL=NBL+1
0010 DO 10 M=2,INBL
0011 READ(5,20)X(M),Y(M),WT(M)
0012 10 CONTINUE
0013 20 FORMAT(2F10.3,I10)
0014 INPUT THE BLOCK NUMBERS FOR EACH POSSIBLE SHAFT LOCATION
0015 30 FORMAT(18I4)
0016 C**** INSERT A NUMBER EQUAL TO NLOC IN STMT.30 ABOVE BEFORE EACH RUN**
0017 DO 40 K=1,NLOC
0018 M=LOC(K)
0019 AX(K)=X(M)
0020 AY(K)=Y(M)
0021 JNBL=NBL+2
0022 KNBL=NBL+NLOC+1
0023 DO 50 M=2,INBL
0024 DO 50 N=JNBL,KNBL
0025 K=N-(NBL+1)
0026 D(M,N)=(ABS(X(M)-AX(K))+ABS(Y(M)-AY(K)))/1000.
0027 50 CONTINUE
0028 DO 60 A=1,NBL
0029 I(A)=1
0030 J(A)=A+1
0031 HI(A)=WT(A+1)
0032 LO(A)=0
0033 60 CONTINUE
0034 A=NBL+1
0035 DO 70 M=2,INBL
0036 DO 70 K=JNBL,KNBL
0037 I(A)=M
0038 J(A)=K
0039 HI(A)=WT(M)
0040 A=A+1
0041 70 CONTINUE
0042 A=NBL+NBL*NLOC+1
0043 DO 80 M=JNBL,KNBL
0044 HI(A)=SUMWT
0045 I(A)=M
0046 J(A)=NODES
0047 A=A+1
0048 80 CONTINUE
0049 IF(A.NE.ARCS)GO TO 100

```

```

0048 I(A)=NODES
0049 J(A)=1
0050 MNBL=ARCS-(NLOC+1)
0051 DO 110 A=INBL,MNBL
0052 IA=I(A)
0053 JA=J(A)
0054 110 COST(A)=INT(UNIT*D(IA,JA))
0055 MNBL=MNBL+1
0056 MARCS=ARCS-1
0057 READ(5,120)(CAP(A),A=MNBL,MARCS)
0058 120 FORMAT( 8I10)
      C*** INSERT NUMBER EQUAL TO NLOC IN STMT. 120 BEFORE EACH RUN *****
0059 RETURN
0060 100 PRINT 130
0061 130 FORMAT(36HERROR IN NUMBER OF ARCS. CHECK DIST.)
0062 CALL EXIT
0063 END

```

T-1468

```

0001 SUBROUTINE NETFLD(NODES,ARCS,INFES,IBT,NBT,MB,MA,NLDC)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
      ICAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
      2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,AOK
0007 INTEGER XI,TEMP
0008 LOGICAL INFES
0009 INFES=.TRUE.
0010 DO 10 A=1,ARCS
0011 IF (LO(A).GT.HI(A))GO TO 39
0012 10 CONTINUE
0013 INF=999999999
0014 ADX=0
0015 IBT=0
0016 NBT=0
      C ***** FIND OUT OF KILTER ARC *****
0017 20 DO 21 A=1,ARCS
0018 IA=I(A)
0019 JA=J(A)
0020 C(A)=COST(A)+PI(IA)-PI(JA)
0021 IF((FLOW(A).LT.LO(A)).OR.(C(A).LT.O.AND.FLOW(A).LT.HI(A))) GOT022
0022 IF((FLOW(A).GT.HI(A)).OR.(C(A).GT.O.AND.FLOW(A).GT.LO(A)))GOTO23
0023 21 CONTINUE
      C ***** NO REMAINING OUT OF KILTER ARCS *****
0024 GO TO 38
0025 22 SRC=J(A)
0026 SNK=I(A)
0027 E=+1
0028 GO TO 24
0029 23 SRC=I(A)
0030 SNK=J(A)
0031 E=-1
0032 GO TO 24
      C ***** ATTEMPT TO BRING OUT OF KILTER ARCS INTO KILTER *****
0033 24 IF((A.EQ.AOK).AND.(NA(SRC).NE.0)) GO TO 25
0034 ADX=A
0035 DO 26 Y=1,NODES
0036 NA(N)=0
0037 NB(N)=0
0038 NA(SRC)=IABS(SNK)*E
0039 NB(SRC)=IABS(AOK)*E
0040 26 CONTINUE
0041 25 COK=C(A)
0042 27 LAB=0
0043 DO 30 A=1,ARCS
0044 JA=J(A)
0045 IA=I(A)
0046 OIF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.(NA(IA).NE.0.AND.NA(JA).NE.0))
      1GO TO 30

```

```

0047      LIA)=CUST(A)+PI(I(A))-PI(J(A))
0048      IF(NA(I(A)).EQ.0)GO TO-28
0049      IF(FLOW(A).GE.HI(A)).OR.(FLOW(A).GE.LO(A)).AND.C(A).GT.0)GO TO 30
0050      NA(J(A))=I(A)
0051      NB(J(A))=A
0052      GO TO 29
0053      28 IF(FLOW(A).LE.LO(A)).OR.(FLOW(A).LE.HI(A)).AND.C(A).LT.0)GO TO 30
0054      IA=I(A)
0055      NA(IA)=-J(A)
0056      NB(IA)=-A
0057      29 LAB=I
0058      C ***** NODE LABELLED. TEST FOR BREAKTHRU. *****
0059      IF(NA(SNK).NE.0)GO TO 33
0060      30 CONTINUE
0061      C ***** NO BREAKTHRU *****
0062      IF(LAB.NE.0)GO TO 27
0063      NBT=NBT+1
0064      ***** DETERMINE INCREMENTAL CHANGE TO NODE PRICES *****
0065      DEL=INF
0066      DO 31 A=1,ARCS
0067      JA=J(A)
0068      IA=I(A)
0069      OIF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.(NA(IA).NE.0.AND.NA(JA).NE.0))
0070      IGO TO 31
0071      C(A)=COST(A)+PI(IA)-PI(JA)
0072      IF(NA(JA).EQ.0.AND.FLOW(A).LT.HI(A))DEL=MINO(DEL,C(A))
0073      IF(NA(JA).NE.0.AND.FLOW(A).GT.LO(A)) DEL=MINO(DEL,-C(A))
0074      31 CONTINUE
0075      OIF(DEL.EQ.INF.AND.(FLOW(AOK).EQ.HI(AOK)).OR.FLOW(AOK).EQ.LO(AOK)))
0076      IDEL=IABS(COK)
0077      IF(DEL.EQ.INF)GO TO 39
0078      C ***** NO FEASIBLE FLOW. EXIT. *****
0079      OO 32 N=1,NODES
0080      IF(PI(M).EQ.0)PI(N)=PI(N)+DEL
0081      32 CONTINUE
0082      GO TO 20
0083      C ***** BREAKTHRU. COMPUTE INCREMENTAL FLOW. *****
0084      33 EPS=INF
0085      IBT=IBT+1
0086      NI=SRC
0087      110 FORMAT(11X,1HA,10X,2HNI,10X,2HNJ,6X,6HPI%NI<,6X,6HPI%NJ<,6X,6HNB%N
0088      100 FORMAT(9I12)
0089      34 NJ=IABS(NA(NI))
0090      A=IABS(NB(NI))
0091      K=CUST(A)-ISIGN(IABS(PI(NI))-PI(NJ),NB(NI))
0092      IF(NB(NI).LT.0)GO TO 35
0093      IF(K.GT.0.AND.FLOW(A).LT.LO(A))EPS=MINO(EPS,LO(A)-FLOW(A))
0094      IF(K.LE.0.AND.FLOW(A).LT.HI(A))EPS=MINO(EPS,HI(A)-FLOW(A))
0095      GO TO 36
0096      35 IF(K.LT.0.AND.FLOW(A).GT.HI(A))EPS=MINO(EPS,FLOW(A)-HI(A))
0097      IF(K.GE.0.AND.FLOW(A).GT.LO(A))EPS=MINO(EPS,FLOW(A)-LO(A))

```

```

0091 36 NI=NIJ
0092 IF(NI.NE.SRC)GO TO 34
C ***** CHANGE FLOW BY EPS *****
0093 37 NIJ=ABS(NA(NIJ))
0094 A=ABS(NB(NIJ))
0095 FLOW(A)=FLOW(A)+ISIGN(EPS,NB(NIJ))
0096 NI=NIJ
0097 IF(NI.NE.SRC)GO TO 37
0098 AOK=0
0099 GO TO 20
0100 39 INFES=FALSE.
0101 38 CONTINUE
0102 RETURN
0103 END

```

T-1468

T-1468

```

0001 SUBROUTINE LABEL(NODES,ARCS,IBT,NBT,ADK,COK,MB,MA,NLLOC)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
1CAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMHT,COST,HI,LO,FLOW,PI,CAP,LDC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,ADK
0007 INTEGER XI,TEMP
0008
0009
0010 DO 10 A=1,ARCS
0011 IF (LO(A).GT.HI(A))GO TO 39
0012 10 CONTINUE
0013 C ***** FIND OUT OF KILTER ARC *****
15 A=MB-1
0014 KA=A
0015 20 KA=KA+1
0016 A=KA
0017 IF(A.GT.MA)RETURN
0018 IA=I(A)
0019 JA=J(A)
0020 C(A)=COST(A)+PI(IA)-PI(JA)
0021 IF((FLOW(A).LT.LO(A)).OR.(C(A).LT.O.AND.FLOW(A).LT.HI(A))) GOTD22
0022 IF((FLOW(A).GT.HI(A)).OR.(C(A).GT.O.AND.FLOW(A).GT.LO(A)))GOTD23
0023 21 IF(A.GE.MA)GO TO 38
0024 GO TO 20
0025 C ***** NO REMAINING OUT OF KILTER ARCS *****
22 SRC=J(A)
0026 SNK=I(A)
0027 E=1
0028 GO TO 24
0029 23 SRC=I(A)
0030 SNK=J(A)
0031 E=-1
0032 GO TO 24
0033 C ***** ATTEMPT TO BRING OUT OF KILTER ARCS INTO KILTER *****
24 CONTINUE
0034 ADK=A
0035 DO 26 N=1,NODES
0036 NA(N)=0
0037 NB(N)=0
0038 NA(SRC)=IABS(SNK)*E
0039 NB(SRC)=IABS(ADK)*E
0040 26 CONTINUE
0041 25 COK=C(A)
0042 27 LAB=0
0043 DO 30 A=1,ARCS
0044 JA=J(A)
0045 IA=I(A)
0046 OIF((NA(IA).EQ.O.AND.NA(JA).EQ.O).OR.(NA(IA).NE.O.AND.NA(JA).NE.O))
0047

```

80

```

0048      GO TO 30
0049      C(A)=COST(A)+PI(IA)-PI(JA)
0050      IF(MA(IA).EQ.0)GO TO 28
0051      IF(FLOW(A).GE.HI(A).OR.(FLOW(A).GE.LO(A).AND.C(A).GT.0))GO TO 30
0052      NA(JA)=I(A)
0053      NU(JA)=A
0054      GO TO 29
0055      28 IF(FLOW(A).LE.LO(A).OR.(FLOW(A).LE.HI(A).AND.C(A).LT.0))GO TO 30
0056      IA=I(A)
0057      NA(IA)=-J(A)
0058      NB(IA)=-A
0059      29 LAB=I
0060      C ***** NODE LABELLED. TEST FOR BREAKTHRU. *****
0061      IF(NA(SNK).NE.0)GO TO 33
0062      30 CONTINUE
0063      C ***** NO BREAKTHRU *****
0064      IF(LAB.VE.0)GO TO 27
0065      NBT=NBT+1
0066      GO TO 20
0067      C ***** BREAKTHRU. COMPUTE INCREMENTAL FLOW. *****
0068      33 EPS=INF
0069      IBT=IBT+1
0070      NI=SRC
0071      PRINT 110
0072      110 FORMAT(11X,1HA,10X,2HNI,10X,2HNJ,6X,6HPI%NI<,6X,6HPI%NJ<,6X,6HNB%N
0073      IK,5X,7HCOST%K<,11X,1HK,9X,3HEPS)
0074      34 NJ=IABS(NA(NI))
0075      A=IABS(NB(NI))
0076      K=COST(A)-ISIGN(IABS(PI(NI)-PI(NJ)),NB(NI))
0077      IF(NB(NI).LT.0)GO TO 35
0078      IF(K.GT.0.AND.FLOW(A).LT.LO(A))EPS=MINO(EPS,LO(A)-FLOW(A))
0079      IF(K.LE.0.AND.FLOW(A).LT.HI(A))EPS=MINO(EPS,HI(A)-FLOW(A))
0080      35 IF(K.LT.0.AND.FLOW(A).GT.HI(A))EPS=MINO(EPS,FLOW(A)-HI(A))
0081      IF(K.GE.0.AND.FLOW(A).GT.LO(A))EPS=MINO(EPS,FLOW(A)-LO(A))
0082      36 NI=NJ
0083      IF(NI.VE.SRC)GO TO 34
0084      C ***** CHARGE FLOW BY EPS *****
0085      37 NJ=IABS(NA(NI))
0086      A=IABS(NB(NI))
0087      FLOW(A)=FLOW(A)+ISIGN(EPS,NB(NI))
0088      NI=NJ
0089      IF(NI.NE.SRC)GO TO 37
0090      AOK=0
0091      CALL RECAP(MB,MA)
0092      CALL SORT(MB,MA,NLOC)
0093      GO TO 15
0094      39 PRINT 290,A
0095      290 FORMAT(9H LO GT HI ,13)

```

0094 38 CONTINUE
0095 RETURN
0096 END

T-1468

```
0001 SUBROUTINE RECAP(MB,MA)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
1CAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,CDK,C,DEL,EPS,ADK
0007 INTEGER XI,TEMP
0008
0009
0010 DO 10 A=MB,MA
0011 IA=I(A)
0012 JA=J(A)
0013 IF(FLOW(A).GT.0)GO TO 80
0014 COST(A)=-PI(IA)+PI(JA)
0015 GO TO 90
0016 80 COST(A)=CAP(A)/FLOW(A)
0017 90 C(A)=COST(A)+PI(IA)-PI(JA)
0018 10 CONTINUE
0019 RETURN
0020 END
```

T - 1468

83

```

0001 SUBROUTINE SORT(MB,MA,NLOC)
0002 COMMON CUST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
1CAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),DI(100,100),LOC(100
2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 DIMENSION IEMP1(50),IEMP2(50),IEMP3(50),IEMP4(50),IEMP5(50),IEMP6(
150),IEMP7(50)
0006 DIMENSION L(50),B(50)
0007 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0008 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,AOK,B
0009 DO 10 A=MB,MA
0010 JJ=A-MB+1
0011 IEMP1(JJ)=I(A)
0012 IEMP2(JJ)=J(A)
0013 IEMP3(JJ)=COST(A)
0014 IEMP4(JJ)=HI(A)
0015 IEMP5(JJ)=FLOW(A)
0016 IEMP6(JJ)=CAP(A)
0017 IEMP7(JJ)=C(A)
0018 L(JJ)=A
0019 B(JJ)=C(A)
0020 10 CONTINUE
0021 CALL RSORT(B,L,NLOC)
0022 DO 20 A=MB,MA
0023 JJ=L(A-MB+1)-MB+1
0024 I(A)=IEMP1(JJ)
0025 J(A)=IEMP2(JJ)
0026 COST(A)=IEMP3(JJ)
0027 HI(A)=IEMP4(JJ)
0028 FLOW(A)=IEMP5(JJ)
0029 CAP(A)=IEMP6(JJ)
0030 C(A)=IEMP7(JJ)
0031 20 CONTINUE
0032 RETURN
0033 END

```

-1468

0001 SUBROUTINE RSORT(RITEM,LA,N)
C CDC MODIFIED BINARY SORT & SUPER SORTC.
C WILL SORT ARRAY OF NOT GREATER THAN 2**15 # 23,768
DIMENSION RITEM(50),IU(50),IL(50),LA(50)
INTEGER RITEM

0002 M=1
0003 I=1
0004 J=N
0005
0006
0007 5 IF(I.GE.J)GOTO70
0008 10 K=I
0009 L=J
0010 IJ=(I+J)/2

0011 IF(RITEM(I).LE.RITEM(IJ))GOTO20
0012 XT RITEM(I)
0013 RITEM(I) = RITEM(IJ)
0014 RITEM(IJ) = XT
0015 NT=LA(I)
0016 LA(I)=LA(IJ)
0017 LA(IJ)=NT

0018 20 IF(RITEM(J).GE.RITEM(IJ))GOTO40
0019 XT = RITEM(J)
0020 RITEM(J) = RITEM(IJ)
0021 RITEM(IJ) = XT
0022 NT=LA(J)
0023 LA(J)=LA(IJ)
0024 LA(IJ)=NT

0025 IF(RITEM(IJ).GE.RITEM(I))GOTO40
0026 XT = RITEM(I)
0027 RITEM(I) = RITEM(IJ)
0028 RITEM(IJ) = XT
0029 NT=LA(I)
0030 LA(I)=LA(IJ)
0031 LA(IJ)=NT
0032 GOTO40

0033 30 XT = RITEM(L)
0034 RITEM(L) = RITEM(K)
0035 RITEM(K) = XT
0036 NT=LA(L)
0037 LA(L)=LA(K)
0038 LA(K)=NT
0039 IF(IJ.NE.L)GOTO35
0040 IJ=K
0041 GOTO40

0042 35 IF(IJ.NE.K)GOTO40
0043 IJ=L
0044 L=L-1
0045 IF(RITEM(IJ).LT.RITEM(L))GOTO40
0046 K=K+1
0047 IF(RITEM(K).LT.RITEM(IJ))GOTO50
0048 IF(K-L)30,51,52

0049 51 K=K+1
0050 L=L-1

```

0051 52 IF(L-I.LE.J-K)GOTO60
0052 IL(M)=I
0053 IU(M)=L
0054 I=K
0055 M=M+1
0056 GOTO80
0057 60 IL(M)=K
0058 IU(M)=J
0059 J=L
0060 M=M+1
0061 GOTO80
0062 70 M=M-1
0063 IF(M.EQ.0)RETURN
0064 I=IL(M)
0065 J=IU(M)
0066 80 IF(J-I.GE.11)GOTO10
0067 IF(I.EQ.1)GOTO5
0068 GOTO95
0069 90 I=I+1
0070 95 IF(I.EQ.J)GOTO70
0071 IF(RITEM(I).LE.RITEM(I+1))GOTO90
0072 K=I
0073 100 XT = RITEM(K)
0074 RITEM(K) = RITEM(K+1)
0075 RITEM(K+1) = XT
0076 NT=LA(K)
0077 LA(K)=LA(K+1)
0078 LA(K+1)=NT
0079 K=K-1
0080 IF(RITEM(K+1).LT.RITEM(K))GOTO100
0081 GOTO90
0082 END

```

T-1468

```

0001 SUBROUTINE PRICE(INFES,JIND,ADK,COK,NBT,ARCS,NODES)
0002 COMMON/ COST(1000),I(1000),J(1000),HI(1000),LD(1000),FLOW(1000),
ICAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LD,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,ADK
0007 INTEGER XI,TEMP
0008 LOGICAL INFES
C ***** DETERMINE INCREMENTAL CHANGE TO NODE PRICES *****
0009 INF=999999999
0010 JIND=0
0011 DEL=INF
0012 DO 31 A=1,ARCS
0013 JA=J(A)
0014 JA=I(A)
0015 OIF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.(NA(IA).NE.0.AND.NA(JA).NE.0))
IGO TO 31
0016 C(A)=COST(A)+PI(IA)-PI(JA)
0017 IF(NA(JA).EQ.0.AND.FLOW(A).LT.HI(A))DEL=MINO(DEL,C(A))
0018 IF(NA(JA).NE.0.AND.FLOW(A).GT.LO(A)) DEL=MINO(DEL,-C(A))
0019 31 CONTINUE
0020 OIF(DEL.EQ.INF.AND.(FLOW(ADK).EQ.HI(ADK).OR.FLOW(ADK).EQ.LO(ADK)))
IDEL=IABS(COK)
IF(DEL.EQ.INF)GO TO 39
C ***** NO FEASIBLE FLOW. EXIT. *****
0021
0022
0023
0024 DO 32 N=1,NODES
0025 IF(NA(N).EQ.0)PI(N)=PI(N)+DEL
0026
0027
0028 32 CONTINUE
0029 JIND=1
0030 RETURN
0031 39 INFES=.FALSE.
0032 RETURN
0033 END

```

T-146B

```

0001 SUBROUTINE LABEL1(NODES,ARCS,IBT,NBT,AOK,COK,MB,MA)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LO(1000),FLOW(1000),
CAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
2),WT(100)
0003 COMMON NB(100),NA(100),C(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LO,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,ADK
0007 INTEGER XI,TEMP
0008 INF=9999999
0009 IF(C(AOK).LT.0.AND.FLOW(AOK).LT.HI(AOK))GO TO 22
0010 IF(C(AOK).GT.0.AND.FLOW(AOK).GT.LO(AOK))GO TO 19
0011 23 PRINT 5,AOK,C(AOK),FLOW(AOK),HI(AOK)
0012 5 FORMAT (1H,4HAOK#,13,5H C#,18,6H FLOW#,18,6H HI#,18)
0013 DO 10 A=MB,MA
0014 IF(C(A).GT.0.AND.FLOW(A).GT.LO(A)) GO TO 15
0015 10 CONTINUE
0016 CALL EXIT
0017 15 AOK=A
0018 GO TO 19
0019 22 SRC=J(AOK)
0020 SNK=I(AOK)
0021 E=+1
0022 GO TO 24
0023 19 SRC=I(AOK)
0024 SNK=J(AOK)
0025 E=-1
0026 24 DO 26 N=1,NODES
0027 NA(N)=0
0028 NB(N)=0
0029 NA(SRC)=IABS(SNK)*E
0030 NB(SRC)=IABS(AOK)*E
0031 26 CONTINUE
0032 25 COK=C(AOK)
0033 27 LAB=0
0034 DO 30 A=1,ARCS
0035 JA=J(A)
0036 IA=I(A)
0037 IF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.(NA(IA).NE.0.AND.NA(JA).NE.0))
GO TO 30
C(A)=COST(A)+PI(IA)-PI(JA)
0038 IF(NA(IA).EQ.0)GO TO 28
0039 IF(FLOW(A).GE.HI(A).OR.(FLOW(A).GE.LO(A).AND.C(A).GT.0))GO TO 30
0040 NA(JA)=I(A)
0041 NB(JA)=A
0042 GO TO 29
0043 28 IF(FLOW(A).LE.LO(A).OR.(FLOW(A).LE.HI(A).AND.C(A).LT.0))GO TO 30
0044 NB(IA)=-A
0045 NA(IA)=-J(A)
0046 29 LAB=1
0047 IF(NA(SNK).NE.0)GO TO 33
0048 30 CONTINUE
0049

```

UNIFORM AT V. 12744 20

0050 IF(LAB.NE.0)GO TO 27
0051 RETURN
0052 33 PRINT 40,ACK
0053 40 FORMAT(20H BREAKTHRU IN LABEL1,13)
0054 CALL EXIT
0055 END

T-1468

89

```

0001 SUBROUTINE TCOST(ARCS,NLOC,NS)
0002 COMMON COST(1000),I(1000),J(1000),HI(1000),LD(1000),FLOW(1000),
1CAP(1000),PI(100),X(100),Y(100),AX(100),AY(100),D(100,100),LOC(100
2),WT(100)
0003 COMMON NR(100),NA(100),DUM(1000)
0004 COMMON TEMP(100)
0005 INTEGER A,ARCS,SUMWT,COST,HI,LD,FLOW,PI,CAP,LOC,WT,S
0006 INTEGER CA,FA,E,SRC,SNK,COK,C,DEL,EPS,ADK
0007 INTEGER XI,TEMP
0008 INTEGER TOTCOS
0009 NS=0
0010 MARCS=ARCS-NLOC-1
0011 TOTCOS=0
0012 DO 10 A=1,MARCS
0013 TOTCOS=FLOW(A)*COST(A)+TOTCOS
0014 10 CONTINUE
0015 IB=MARCS+1
0016 MB=ARCS-1
0017 DO 30 A=IB,MB
0018 IF(FLOW(A).LE.0)GO TO 30
0019 NS=NS+1
0020 TOTCOS=TOTCOS+CAP(A)
0021 30 CONTINUE
0022 PRINT 20, TOTCOS
0023 20 FORMAT(47H TOTAL COST OF TRANSPORTATION AND CAPITAL# ,I10)
0024 RETURN
0025 END

```

APPENDIX II

NUMBER OF NODES# 60
 NUMBER OF ARCS# 779

ARC	NODEI	NODEJ	COST	HI	LO	FLOW	CAP
1	1	2	0	1000	0	1000	0
2	1	3	0	1000	0	1000	0
3	1	4	0	1000	0	1000	0
4	1	5	0	1000	0	1000	0
5	1	6	0	1000	0	1000	0
6	1	7	0	1000	0	1000	0
7	1	8	0	1000	0	1000	0
8	1	9	0	1000	0	1000	0
9	1	10	0	1000	0	1000	0
10	1	11	0	1000	0	1000	0
11	1	12	0	1000	0	1000	0
12	1	13	0	1000	0	1000	0
13	1	14	0	1000	0	1000	0
14	1	15	0	1000	0	1000	0
15	1	16	0	1000	0	1000	0

752	41	51	160	1000	0	0	0	0	0
753	41	52	180	1000	0	0	0	0	0
754	41	53	80	1000	0	0	0	0	0
755	41	54	100	1000	0	0	0	0	0
756	41	55	120	1000	0	0	0	0	0
757	41	56	140	1000	0	0	0	0	0
758	41	57	160	1000	0	0	0	0	0
759	41	58	180	1000	0	0	0	0	0
760	41	59	60	1000	0	0	0	0	0
761	45	60	14	40000	0	29000	0	430000	0
762	46	60	37	40000	0	0	0	400000	0
763	55	60	57	40000	0	0	0	720000	0
764	51	60	57	40000	0	0	0	400000	0
765	52	60	74	40000	0	0	0	520000	0
766	56	60	74	40000	0	0	0	400000	0
767	58	60	34	40000	0	11000	0	650000	0
768	57	60	54	40000	0	0	0	380000	0
769	42	60	77	40000	0	0	0	510000	0
770	50	60	37	40000	0	0	0	1050000	0
771	53	60	97	40000	0	0	0	600000	0
772	43	60	57	40000	0	0	0	850000	0
773	44	60	37	40000	0	0	0	800000	0
774	54	60	77	40000	0	0	0	810000	0
775	49	60	57	40000	0	0	0	880000	0
776	47	60	97	40000	0	0	0	650000	0
777	48	60	77	40000	0	0	0	800000	0
778	59	60	117	40000	0	0	0	900000	0
779	60	1	0	40000	40000	40000	40000	700000	0

NODE	PIRNODEK
1	266
2	326
3	346
4	366
5	386
6	406
7	386
8	346
9	366
10	386
11	406
12	426
13	406
14	326
15	346
16	366
17	386
18	406
19	386
20	366
21	386
22	366
23	306
24	326
25	346
26	366
27	386

28	366
29	386
30	406
31	386
32	286
33	306
34	326
35	346
36	366
37	346
38	366
39	386
40	366
41	266
42	366
43	386
44	406
45	426
46	406
47	346
48	366
49	386
50	406
51	386
52	366
53	346
54	366
55	386
56	366
57	386
58	406
59	326
60	440

THE NUMBER OF BREAKTHROUGHS IS 95

THE NUMBER OF NONBREAKTHROUGHS IS 537

THE NUMBER OF SHAFTS# 16 THE NUMBER OF BLOCKS# 40 THE NUMBER OF POTENTIAL SHAFT LOCATIONS# 18

TOTAL COST OF TRANSPORTATION AND CAPITAL# 3110000

LIST OF REFERENCES

- Benichou, M., J. M. Gauthier, P. Girodel, G. Hentzes, G. Ribiere and O. Vincent, 1971, Experiments in mixed-integer linear programming; *Mathematical Programming*; v. 1, no. 1, pp. 76-94; Amsterdam, The North Holland Publishing Company.
- Balas, E., 1967, Duality in discrete programming; 6th International Symposium on Mathematical Programming, Princeton, N.J.
- Cabot, A.V. Francis and Stary, 1970, A network flow solution to a rectilinear distance facility location problem; *Trans. A.I.I.E.*; v. 11, no. 2, pp. 132-141
- Dantzig, G.B. 1963, *Linear Programming and Extensions*; Princeton, N.J.; Princeton University Press; 632 p.
- Durbin, E.P. and D. M. Kroenke, 1967, The out-of-kilter algorithm: a primer; Memorandum RM-5472-PR; Santa Monica, Calif.; The Rand Corporation; 22 p.
- El Maghraby, S.E. 1970, The theory of networks and management science. Part 1: *Management Science*; v. 17, no. 1; pp. 1-34; Part 2: *Management Science*; v. 17, no. 2; pp. 54-71; Providence, R.I.; Inst. of Management Science.
- _____, 1970 b, Some network models in management science; Lecture notes in O.R. and mathematical systems, No. 29; Berlin; Springer Verlag; 175 p.
- Francis, R.L. 1964, On the location of multiple new facilities with respect to existing facilities; *J. Ind. Eng.*; v. XV, no. 2; pp. 106-109.
- Fulkerson, D.R. 1961, An out-of-kilter method for minimal cost flow problems; *J. Soc. Ind. and Appl. Math.*; v. 9, no. 1; pp. 18-27.

- Gray, P., 1967, Mixed integer programming algorithms for site selection and other fixed charge problems having capacity constraints; SED-Special Report; Menlo Park, Calif.; Stanford Research Institute; 132 p.
- Hillier, F.S. and G. J. Liebermann, 1969, Introduction to operations research; San Francisco, Calif.; Holden-Day Inc.; 639 p.
- Humphreys, K.K. and J. W. Leonard, 1970, Optimum location of mining facilities for safety and economy; Rept. No. 57; Coal Research Bureau; Morgantown, W. Va.; West Virginia University.
- Jandy, G., 1967, Approximate algorithm for the fixed-charge capacitated site location problem; Tech. Rept. No. 67-3, Operations Research House; Calif.; Stanford University; 20 p.
- Johnson, T.B. and D.G. Mickle, 1971, Optimum design of an open-pit - an application in uranium; Decision-making in the mineral industry; C.I.M. Special Volume 12, pp. 331-338; Montreal, Canada; Canadian Inst. of Mining and Metall.
- Murthy, K.G. 1968, Solving the fixed-charge transportation problem by ranking the extreme points; Opns. Res. 16; pp. 268-279; Maryland; Opns. Res. Soc. of Amer.
- Sharp, W. R. 1972, Design of an ultimate underground mine layout; M.S. Thesis; Golden, Colo.; Colo. Sch. of Mines.
- Steinberg, D.I. 1970, The fixed-charge problem; Naval Research Logistics Quarterly; v. 17, no. 2; pp. 217-235.
- Wesolowsky, G.O. and R.F. Love, 1971, Location of facilities with rectangular distances among point and area destinations; Naval Res. Log. Quart.; v. 18, no. 1; pp. 83-90.
- Zambo, J., 1968, Optimum location of mining facilities; Budapest; Akademiai Kiado, Publishing House of the Hungarian Academy of Sciences and the Academy Press; 144 p.