RI 7828

Bureau of Mines Report of Investigations/1974

Design of an Underground Mine Layout

UNITED STATES DEPARTMENT OF THE INTERIOR

Report of Investigations 7828

Design of an Underground Mine Layout

By William R. Sharp
Mining Enforcement and Safety Administration, Denver, Colo.

UNITED STATES DEPARTMENT OF THE INTERIOR
Rogers C. B. Morton, Secretary

BUREAU OF MINES
John D. Morgan, Jr., Acting Director

This publication has been cataloged as follows:

Sharp, William R

Design of an underground mine layout. [Washington] U.S. Bureau of Mines [1974]

45 p. illus., table. (U.S. Bureau of Mines. Report of investigations 7828)

Includes bibliography.

1. Mining engineering—Computer programs. I. U.S. Bureau of Mines. II. Title. (Series)

TN23.U7 no. 7828 622.06173

U.S. Dept. of the Int. Library

CONTENTS

	Page
Abstract	1
Introduction	1
Acknowledgments	4
General underground mining problem	4
Literature review	4
Definitions and sample problem	5
Problem formulation	8
Solution algorithm	9
General algorithm	10
Least-cost path algorithm	11
Discussion of solution algorithm	12
Sample problem	12
Algorithm structure	14
Implementation	17
Application to a coal property	17
Computational experience	19
Conclusions and extensions	20
References	23
Appendix A.--Table of symbols	25
Appendix B.--Input and results of mining problem	27
Appendix C.--Computer program	30

ILLUSTRATIONS

1.	Graphical representation (B) of an orthogonal set of blocks (A)	5
2.	Block and graphic illustration for (A) general two-dimensional block configuration and (B) systematic two-dimensional hexagonal block	6
3.	Graphic (A) and geometric (B) illustration of the Steiner problem	7
4.	Graphic illustration for problem (A) and (B)	13
5.	Graphic illustration for problem (A) and (B)	14
6.	Illustration of proof of least-cost path algorithm for only negatively weighted nodes	16
7.	Two-dimensional application for a coal property	18

TABLE

1.	Computation results for four different size problems	19

DESIGN OF AN UNDERGROUND MINE LAYOUT

by

William R. Sharp [1]

ABSTRACT

This Bureau of Mines report details a technique for aiding the engineer with the long-range design of an underground mine. The method is based upon the assumption that sufficient data are available to assign numeric values to some finite number of mineralized blocks; assignment of numerical values is problem dependent. By assigning appropriate weighting schemes, the algorithm developed in this report may be used to solve the optimal end-of-life underground mine layout, design least-cost electrical layout networks and efficient material handling networks, find least risk escape routes in the event of an accident, etc.

The method is based upon the assumption that sufficient cost and geologic data is available to build a detailed mineral block inventory of the mineral deposit. It is further assumed that the cost assigned to any minable unit is independent of the neighboring blocks. The basic approach relies upon concepts and ideas from the theory of networks.

To make the approach user oriented, a computer program is included and discussed. The computer code is written for an orthogonal, three-dimensional set of blocks. Computational costs and experience indicate that the program could be of great benefit in evaluating large complex mineral properties from the viewpoint of predesigning an underground mine layout.

INTRODUCTION

The minerals industry in the United States is presently faced with the problem of meeting a rapidly increasing domestic and foreign demand for its mineral resources. This increased demand has added to the problems already plaguing the industry--increased land, labor, and material costs; increased foreign competition; more rigid Federal and State mining laws pertaining to health and safety; increased public awareness of air, stream, and land pollution; and increased public concern of mining practices and conditions within the mines. In addition, the industry must live with problems created by past

[1] Mining engineer.

generations, such as abandoned strip mines, acid mine water drainage, and abandoned unmapped underground workings.

The "need" for a healthy, viable minerals industry within the United States has been stated many times over. In addition, it is desirable, if not a necessity, from the viewpoint of the consumer and the national economy that the cost of making available a mineral resource be minimal. The development and utilization of scientific mine planning will be an invaluable aid to the accomplishment of the design and operation of profitable, safe, healthy, and environmental-preserving underground mining operations.

Mine planning is usually divided into three areas: long range, short range, and production scheduling. A long-range plan defines the economic limit of mining at the termination of the deposit's mining life and serves as a guide for short-range plans.

In the process of winning a mineral resource, short-sighted goals can lead to costly development and possible loss of some resource to future exploitation. It is for this reason that this report will be primarily concerned with long-range planning.

A trend within the mineral industry, not unusual in other industries, is toward the development of large, complex physical plants in an effort to reduce unit costs. The trend is toward fewer mines employing more men. For example, in Colorado in 1960, there were 607 noncoal operating mines with a yearly employment of 5,220 men, compared with only 353 mines employing 6,660 men in 1970 as given in Colorado's yearly report by the Deputy Commissioner of Mines. This trend seems likely to continue.

Another factor that has had a profound effect in reducing the number of small operations has been stricter mining laws. For example, as a result of the Coal Mine Health and Safety Act of 1969,[2] it is stated that "In any coal mine opened after March 30, 1970, the entries used as intake and return air courses shall be separated from belt haulage entries...." This regulation has sharply increased mining costs for the small coal mine operator using belt haulage by forcing any egress into a coal property with a minimum of three interconnected parallel entries. In the past, the mine operator was able to operate with two entries, one for intake air and the other for exhaust air. The economic impact of this law has been significantly less for the large mine operator, who now usually operates with an excess of five parallel main entries.

With the trend towards larger, more complex mine plants comes the increased responsibility of the mining company to comply with the needs of society as a whole, thus increasing the needs for good scientific mine design.

Today, coal companies are looking at properties for exploitation that may contain multiple seams within a very complex geologic environment covering

---

[2]Mandatory Safety Standards, Underground Coal Mines. Nov. 20, 1970, Federal Register, v. 35, No. 226, Sec. 75.326.

many square miles. Additional problems, such as land ownership, oil and/or gas wells intersecting the mining property, highways, surface buildings, powerlines, canals, water tables, and entrapped methane, present a mining company with many problems not encountered only a few years ago (3, 16).[3]

The need for sound mine planning within our complex sociological framework is well documented. The purpose of this report is to develop a tool for assisting the mine operator to efficiently evaluate alternative approaches to mine planning.

The mining engineer is faced with the need to respond quickly to modify a mine design to satisfy rapidly changing economical, ecological, and sociological constraints. Heretofore the mining engineer could be content with comparing only a few alternative mine designs. Today, he must explore many alternative plans constrained by factors unknown only a few years ago (16).

In terms of long-range planning, the design engineer must be alert to the changes in the overall mine plan resulting from changing land, material, and labor costs or the introduction of innovative mining concepts. Possibly the design engineer will be requested to make a quick evaluation as to whether or not a company should extend a lease on an unexploited piece of land. In terms of short-range planning, an overall mine plan is also extremely important to the engineer in charge of scheduling mine development and production to meet demand commitments. The end-of-life or ultimate mine layout is also of concern to the engineer in charge of designing efficient ventilation, haulage, drainage, and electrical networks.

The objective of this report will be to develop a method for defining an ultimate underground mine layout consistent with economic, safety, and conservation constraints. Knowing the ultimate mining configuration is vital to the efficient layout of the mine development, mine development is here defined as a network of primary accesses through which the bulk of the material is moved to the surface mine facilities from the mineral producing areas.

To achieve this objective, the following four criteria were used as guidelines in developing an algorithm for delineating the economic limits of a mineralized property:

(1) Mine layout should be everywhere accessible to the surface.

(2) Algorithm should be applicable to the general mineral-type deposit.

(3) Algorithm should be capable of solving realistic size problems.

(4) Technique should be user oriented.

Criteria 1, 3, and 4 above should be self-explanatory. What is intended in criterion 2 by the term "general mineral-type deposit" is that the

[3]Underlined numbers in parentheses refer to items in the list of references preceding the appendixes.

distribution of economic worth may be freely dispersed throughout some three-dimensional rock mass. In other words, the distribution of mineral context need not be rigorously prescribed by some simplistic mathematical function. The problem of forecasting the distribution of mineralization within the earth's crust is recognized. The problems of delineating a mineral deposit are referred to in Becker and Hazen (1).

ACKNOWLEDGMENTS

The work described in this report was sponsored by the U.S. Bureau of Mines in partial fulfillment of the requirements for the degree of Master of Science at the Colorado School of Mines. I am also indebted to the many people at the Bureau of Mines in Denver who have contributed much to the detail and thought contained in this work. I am especially grateful to Dr. Thys B. Johnson, operations research analyst, U.S. Bureau of Mines, for his help in clearly defining the general problem area.

I appreciate the help and encouragement I received from Dr. Walter W. Whitman, professor of mathematics, Colorado School of Mines, and Dr. Huntington S. Swanson, associate professor of mathematics, Colorado School of Mines, and their guidance with the many details contained in this study.

GENERAL UNDERGROUND MINING PROBLEM

In view of the increasing complexity of the general mining problem, mining companies are now beginning to rely upon mathematical techniques and tools to predesign the layout of a mining operation. The following sections contain a brief review of literature, detail a means for expressing the general ultimate underground mine layout as a network, and define a general formulation of the problem.

Literature Review

Perhaps one of the best single sources of operations research techniques applied to mining problems is a recent book published in Hungary (17). This book describes some techniques for such problems as the location of a shaft, determination of level spacing in a dipping deposit, and the "optimal" location of crosscuts. This book does give a wide variety of simple applications for a wide variety of facility location type problems. The term "optimal" used here and discussed later in this paper will be used in a strict mathematical sense.

In the area of ultimate underground mine layout there has been a notable lack of published papers. The problem of designing an ultimate open pit has generated much more interest in the past several years than has the subsurface problem. The ultimate open pit problem has been approached by Lerch and Grossmann (14) using a dynamic programing technique, Johnson (12) employing both a dynamic programing technique and a network-flow model, and Ganthier and Gray (9) using a heuristic approach. An interesting and computationally practical approach, based upon the dynamic programing of Lerch and Grossmann, is given by Johnson and Sharp (13). All of these approaches use what is referred

to as the "block concept," which is simply a method for subdividing a mineral deposit into discrete minable units formed by a set of more or less orthogonally intersecting planes. Discrete block units appear to be a reasonable basis for a subsurface excavation model for the following reasons:

(1) Block concept is accepted and used within the minerals industry for ore reserve and delineation studies.

(2) Blocks can be made to conform with most mining methods used within the industry today.

(3) Block inventory is easily stored in a digital computer.

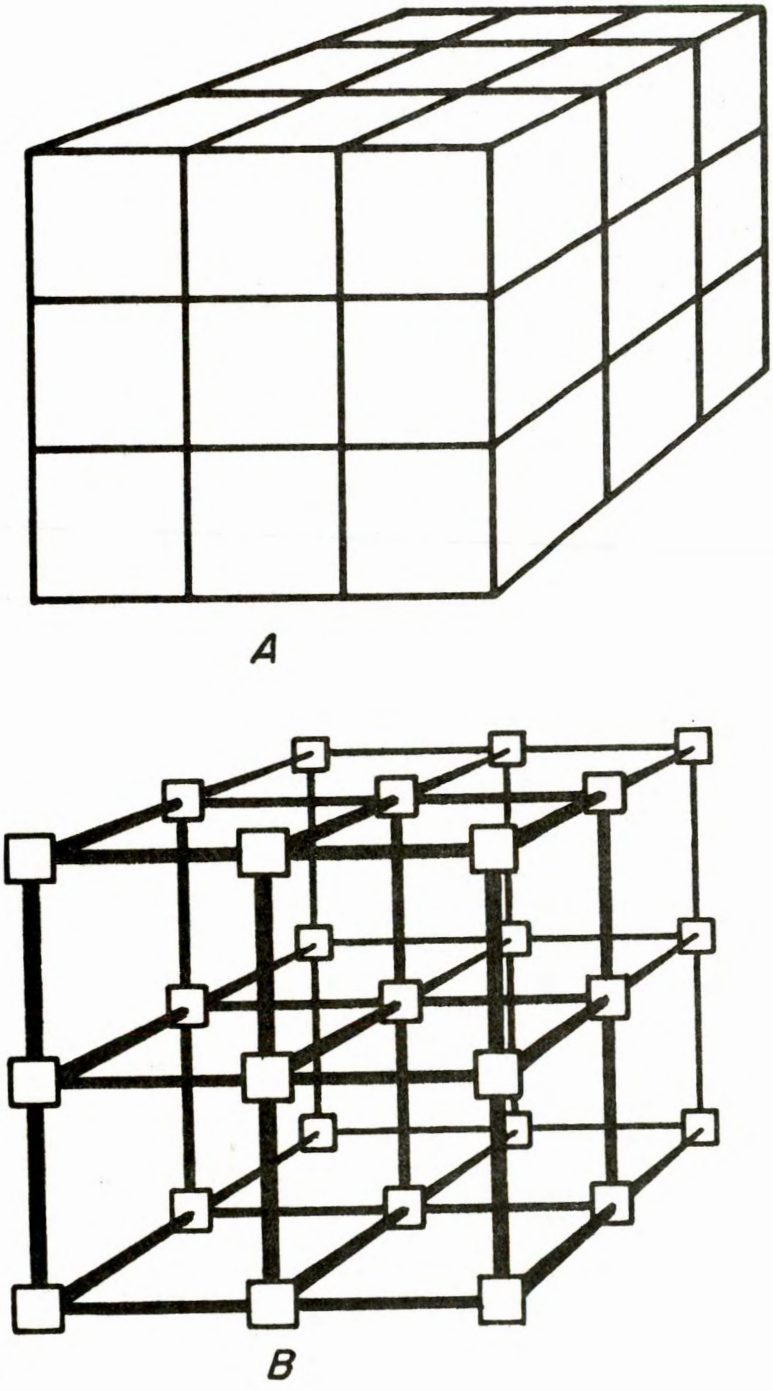


FIGURE 1. - Graphical representation *(B)* of an orthogonal set of blocks *(A)*.

With respect to the ultimate pit problem, the blocks themselves help to describe a pit wall. In comparison, the subsurface problem need only imply connectivity of blocks; that is, existence of a passage from any set of minable blocks to all others in the set to be mined together with a passage to the surface.

Definitions and Sample Problem

The ultimate underground problem becomes more amenable to analysis if a weight is assigned to an individual block equivalent to its worth without regard for its neighboring blocks. In terms of utility, the block worth might represent the least-cost or potential profit that a mining concern

would reap given the fact that any particular block is physically mined. Dependent upon the problem, all block weights might represent a true cost (negative weight), for example, a tunneling operation. In a tunneling problem the contractor or mining company is only concerned with finding a least-cost passage between two points. Therefore, in terms of the block model, it would be a connected sequence of blocks between two points in which the individual blocks would be equivalent to a small segment of the tunnel.

Figure 1-A gives a graphic illustration of how the block might be interpreted in three-dimensional space where the length, width, and height of each block would be determined by the user. Figure 1-B is a graphical illustration of the set of blocks in figure 1-A above. Each block has been pulled apart, figure 1-B, but remains connected to its adjacent neighbors by a line. For purposes of discussion, the lines will be referred to as arcs and the squares (dots or circles) as nodes. These terms are consistent with those used in network theory.

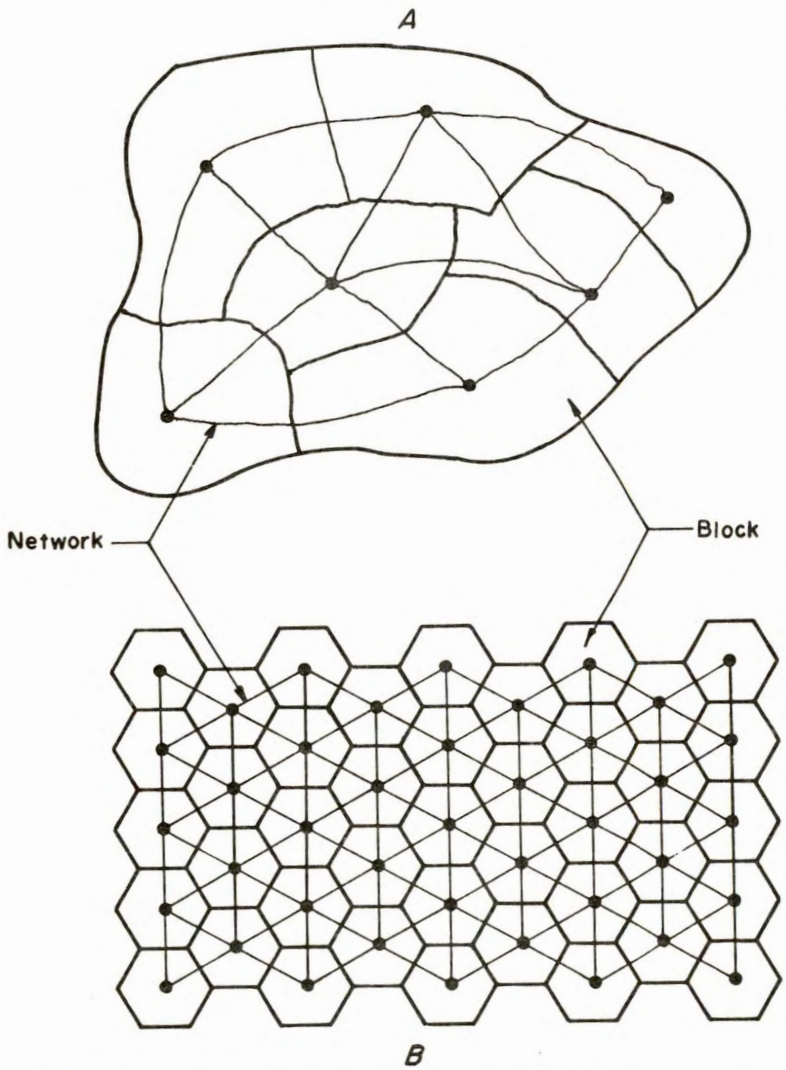


FIGURE 2. - Block and graphic illustration for *(A)* general two-dimensional block configuration and *(B)* systematic two-dimensional hexagonal block.

A block need not be defined as a cube. Figure 2 illustrates two different examples of a basic two-dimensional block configuration. Of interest is what may be represented by these block configurations again as a graph--similar to what was done in figure 1. The nodes represent the blocks and an arc indicates that two nodes are adjacent geometrically to one another.

Figure 3-A illustrates a two-dimensional graphic representation of a set of blocks. A value or weight which implies the worth or cost of mining that particular block has been assigned to each node (block). The path given in figure 3-A as a heavy line is the least-cost path connecting nodes A, B, and C. The problem of finding a connected subset of maximum total weight from a

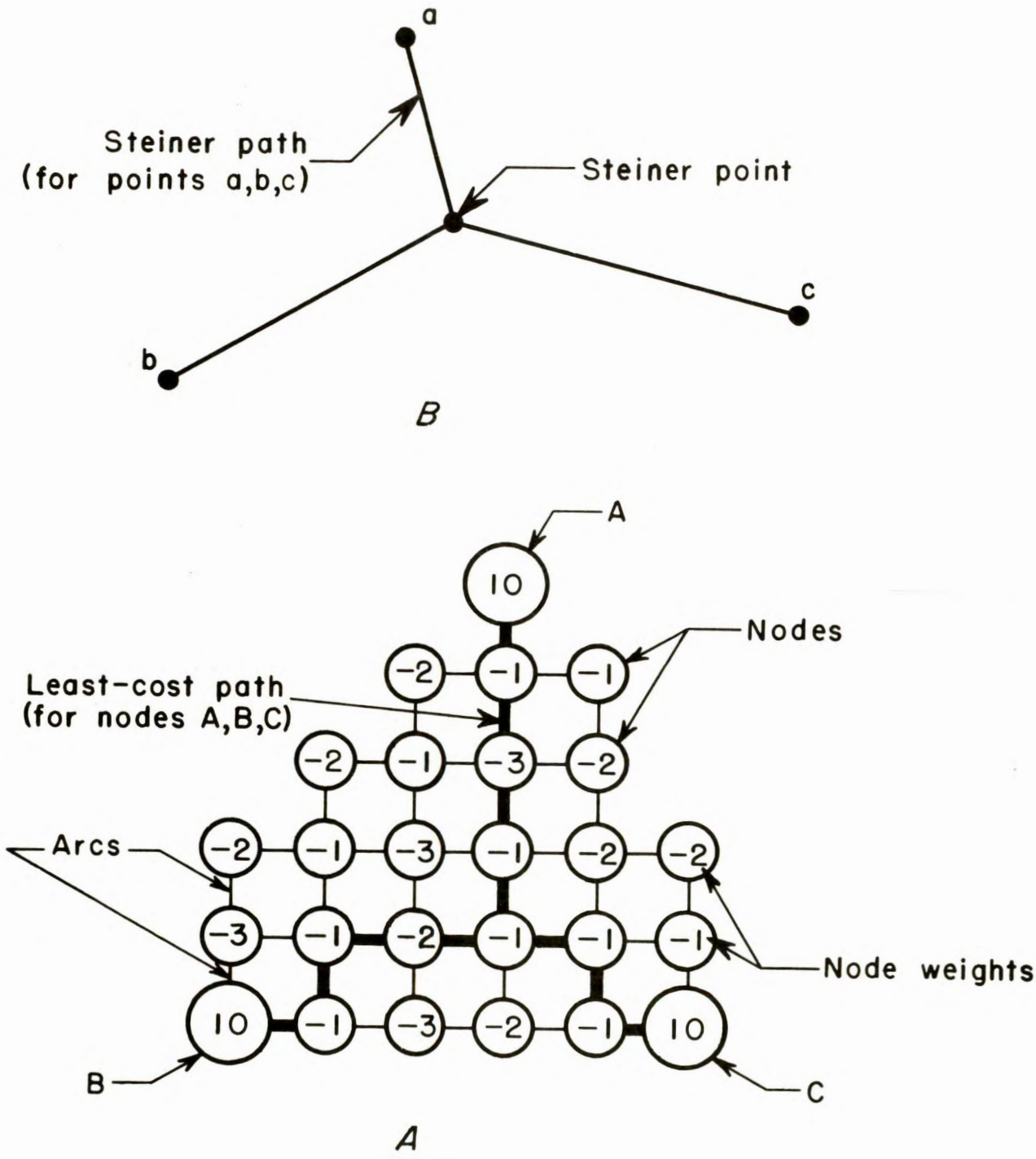


FIGURE 3. - Graphic *(A)* and geometric *(B)* illustration of the Steiner problem.

network is an often-pursued problem in network theory; however, there is no standard algorithm available to solve the problem. Approaching the mining problem in terms of a network gives flexibility of using any combination of

block configurations and ease of problem expression and manipulation with existing computational techniques.

Of particular interest in the mining problem is that there may exist a node belonging to the set of negatively weighted nodes, which, if forced into the solution set, will yield a better solution by acting as a common junction between three or more positively weighted node groups. Because this node is similar in many respects to a Steiner point defined in the classical Steiner problem (10), this type node will be referred to as a Steiner node. In brief, the Steiner problem may be stated as: Find the path of minimal geometric length connecting m points in a plane. Figure 3-B illustrates a least-length or Steiner path for three points in a plane. Figure 3-A is analogous to the general Steiner problem defined on a plane. The general Steiner problem was not pursued because (1) generally, an ore zone may be difficult and misleading to be represented as a geometric point, (2) generally, ore deposits do not occur in planes, and (3) Cockayne (4) states that available computer codes are limited to problems of 20 to 30 points.

Some additional terms are needed to clearly define the mining problem totally in terms of a network (7-8). A graph $G = G(V,\Gamma)$ may be simply defined as consisting of a set of nodes, $V = \{v_1, v_2 \ldots v_m\}$, and a set of arcs, $\Gamma = \{b_1, b_2 \ldots b_n\}$. In addition, it is sufficient to define a set of weights, $\{w(v_i)\}$, such that $w(v_1) \ldots w(v_g) \leq 0$ and $w(v_{g+1}) \ldots w(v_m) > 0$. An arc, b_i, denotes a line segment connecting some node, v_k, to another node, v_j, or more simply as (k,j). In terms of the mining problem, there is no need to describe a directed graph, therefore, $(i,j) = (j,i)$. Any two nodes will be called adjacent if they are joined by a single arc. A path, π_i^j, may then be defined as a sequence of ordered nodes and arcs as $(v_i, (i,s), v_s \ldots (t,j), v_j)$. A path as used in this paper does not contain any cycles; that is, for the sequence of nodes in the path, there is no repetition of any node. A connected graph is defined as a graph where at least one path exists from any arbitrary set of nodes to all others in the graph. Finally, a tree is defined as a connected graph having no cycles. The nearest neighbor to some node, i, as used in this report, will be a single node, j, from a set of possible nodes having the least-cost path from node i.

Problem Formulation

The general subsurface mining problem defined on a set of blocks may be simply stated as find

$$\max \sum_i w_i X_i ,$$

$$\text{such that } X_i - \sum_{k\,j} \pi X_j \leq 0 ,$$

where $\prod_j X_j$ represents the product of a sequence of geometrically adjacent blocks leading from block i to the surface and $\sum_k$ represents the sum of all possible sequences,

and $X_i = \begin{cases} 1 \text{ if block i is mined} \\ 0 \text{ if block i is not mined} \end{cases}$

and w_i = weight assigned to block i.

Mathematically this set of equations defines the general subsurface problem of concern in this paper; however, practically it would be extremely difficult to formulate or solve a problem in this form--even for small problems.

Considerable time and effort was taken to model this relationship by some standard mathematical programing techniques. Attempts to formulate the problem as both a linear and integer problem were not truly successful from the viewpoint of the general problem just defined. Difficulties arise in defining a set of linear constraints. It is a well-known fact that for practical-size problems with nonlinear constraints finding computationally efficient computer codes is a problem that must be considered. The need for an integer solution plus the need for comparing alternative block sequences make this problem extremely difficult to formulate in general terms. Computational efficiency of known integer codes decreases at a rate significantly faster than a comparable increase in the size of the problem.

Because no efficient mathematical programing technique was found, a network approach was pursued. In terms of a network, the ultimate underground mining plan may be simply stated as: Find a connected subgraph, call it G_o, such that the weights belonging to G_o is a maximum. It is also a generally recognized fact that there are many computationally efficient computer codes available to solve large network related problems; for example, the shortest route, the maximum spanning tree, and the maximum flow.

SOLUTION ALGORITHM

An algorithm for finding the ultimate subsurface mining configuration is given in this section and followed by a detailed description of the least-cost path algorithm which is a fundamental part of the solution algorithm. The least-cost path problem is comparable to the shortest route problem. Much work has been done in the development of algorithms for the shortest route problem. The algorithm for the mining problem is in two parts, phases 1 and 2.

Phase 1 of the solution algorithm begins by collecting all of the positively weighted nodes (blocks), which are adjacent into a single node having as its weight the sum of the weights of all its individual members. If no positive nodes exist or if only one is found, the algorithm terminates. If two or more positive nodes are found, phase 1 then proceeds by taking the maximum from the newly found disjoint positive nodes and finding its nearest positive neighbor using a least-cost path algorithm. If the weight of the two disjoint positively weighted nodes plus the least-cost path is strictly greater than the starting node weight, the algorithm proceeds by finding the

least-cost path to a positive neighbor from the combined subgraph consisting of the two positive nodes plus the least-cost path. The phase 1 algorithm continues until no additional nodes from the set of positive nodes can be brought into a connected set of nodes. Phase 1 will give a good solution; however, in general, it may not be optimal. The necessary conditions for a mathematically optimal solution will be discussed in detail under "Algorithm Structure." For phase 1, a computer code was developed and is discussed under "Computational Experience."

An upper bound on the value of the ultimate mine layout would be the sum of all the positive nodes and a lower bound would be zero, which implies the absence of any positive blocks. These facts could be useful in deciding when to terminate the algorithm. For example, if the upper bound is not sufficiently large to meet corporate objectives, the project could be abandoned immediately without further study.

Phase 2 describes a procedure by which a user may try to improve upon the solution resulting from phase 1 through the introduction of Steiner nodes. A Steiner node may be simply thought of as a node, which, if mined, would result in a negative return; however, if by mining that particular node it is possible to use this node as a common intersection of development entries (drifts) for three or more disjoint potential ore zones, then the overall mining costs would be reduced.

The solution algorithm assumes the problem to be defined as a general network as previously described. The mining problem was defined in terms of a general network so that a general algorithm could be found that will work for any number of different block configurations.

General Algorithm

In this section the algorithm used to solve the general ultimate subsurface mine problem is amplified. For definition of terms, see appendix A.

Phase 1:

Step 0 Set AMAX = 0, set ATEMP = 0, set S = 0.

Step 1 Collect all vertices from the set $\{(v_{g+1}) \ldots (v_m)\}$, which are adjacent, into P disjoint sets and form the set $\{(\overline{v}_j) \mid j = 1 \ldots P\}$. Set $T_o = \Omega$. If P = 0, stop; the set $\{(\overline{v}_j)$ for all j$\}$ empty. If P = 1, stop; the optimal solution is $w(\overline{v}_1)$. Otherwise continue. Set $L_j = 0$ for $j = 1 \ldots P$.

Step 2 Set TEMP = 0, set i = 1.

Step 3 If $L_j = \infty$, for all j, go to phase 2. Find q such that $w(\overline{v}_q) = \max \{w(\overline{v}_j) \mid L_j = 0, j = 1 \ldots P\} = TEMP$. Set $L_q = \infty$, $T_i = \overline{v}_q$ and $\overline{T} = \Omega$.

Step 4 Find least-cost path $l(\pi_s^j)$ from $T_i + \overline{T}$ to nearest positive disjoint neighbor k such that node s is adjacent to tree, $T_i + \overline{T}$, and node j is adjacent to node k. If $| l(\pi_s^j) | \geq$ TEMP, go to 7.

Step 5 If TEMP + $\ell(\pi_s^j)$ + $w(\overline{v}_k) >$ TEMP, set TEMP = TEMP + $\ell(\pi_s^j)$ + $w(\overline{v}_k)$ and set $L_k = \infty$ and go to step 6. Otherwise set $\overline{T} = \overline{T} + \pi_s^k$ and go back to step 4.

Step 6 Increment i; i = i + 1. Set $T_i = T_{i-1} + \pi_s^j + v_k$. If $L_j = \infty$ for all j, go to step 7. Otherwise set $\overline{T} = \Omega$ and go back to step 4.

Step 7 If TEMP $\geq$ ATEMP, set ATEMP = TEMP and set $T_o = T_i$. Go to step 2.

Phase 2 allows the user to force any node not present in the solution found in phase 1 into the solution set.

Phase 2:

Step 1 If S = 0, go to step 2. Otherwise set ATEMP = ATEMP - M * S. If ATEMP $\geq$ AMAX, set AMAX = ATEMP and T = T_o.

Step 2 Option to stop at this iteration.

Step 3 Out of the set of nodes, such that $w(v_t) \leq 0$, select a set of Steiner nodes t = 1 . . . S, S $\leq$ P - 2. Set $w(v_t) = w(v_t) + M$, t = 1 . . . S, and add v_t to the set of positive nodes. Go to phase 1, step 2.

Using good perceptual judgment at this point can speed convergence of a realistic problem by helping to reduce unnecessary computational time which could be introduced by an enumerative selection of Steiner nodes.

Least-Cost Path Algorithm

A detailed labeling algorithm is given for finding the least-cost path from any node, q, to every other node, t, in a network, given that all node weights are negative and weights on the arcs are all zero. The algorithm is essentially a dynamic programing approach very similar to that discussed by Dreyfus (6). The major difference is the fact that the weights are defined exclusively on the nodes as opposed to the arcs. This, in fact, simplifies the general approach suggested by Dreyfus. In terms of the general algorithm, a set of nodes {q} would represent a set of nodes in a tree.

Algorithm:

Step 1 Label a set of source nodes $\{q\}$ as L (-,0).

Step 2 Find an unlabeled node t and a labeled node s such that $l(\pi_q^t) = \min \{l(\pi_q^s) + w(v_t)$ for all nodes s and t such that node t is unlabeled and adjacent to a labeled node$\}$.

Step 3 Label node t as $L(s, l(\pi_q^t))$.

Step 4 Does an unlabeled node exist? If yes, go to step 2. Otherwise, stop.

NOTE: q and s are indexes for labeled nodes and t is for the unlabeled. Also, for the label, $L(s, l(\pi_q^t))$, s denotes the adjacent node from which node t is labeled and $l(\pi_q^t)$ denotes the best path from node t to node q.

DISCUSSION OF SOLUTION ALGORITHM

Sample Problem

The following problems were constructed to illustrate how the algorithm works and under what conditions the algorithm can lead to nonoptimal solutions. The illustrations are overly simplistic and in most cases the optimal solution is obvious. In a realistic problem, however, the optimal solution will not be obvious.

Figure 4-A gives a network equivalent to a mining problem having three disjoint ore pods (A,B,C) which would yield a payoff of 10 units each if they could be mined. In addition, node G is adjacent to node C and yields a value of one unit. For example, if node G were exposed to the surface, then nodes C and G could be mined at a combined payoff of 11 units because they are adjacent. In the winning of nodes A and/or B, it is possible to proceed from C by three different paths--all of which contain negatively weighted nodes. Obviously then, the user wishes to pick the path that yields the least-cost for the potential gain. For figure 4-A the optimal solution is, obviously, equivalent to the node blocks A through G having a total weight of 25 units.

For the problem in figure 4-A the algorithm would proceed as follows beginning at phase 1: Step 0, set AMAX, ATEMP, and S = 0 and move to step 1. For step 1 combine the node weights C and G and form a single node. Call it node C. Figure 4-B will then become equivalent to the network in figure 4-A where no two adjacent positively weighted nodes exist. P is equal to 3 or, in this problem, let j = A, B, and C. Set L_A, L_B, $L_C = 0$, i = 1, and TEMP = 0 in step 2. In step 3, q = C because the maximum of $\{w(v_A), w(v_B), w(v_C)\}$ is $w(v_C)$; therefore, set TEMP = 11. Now set $L_C = \infty$. In step 4 the least-cost path from node C to either node A or B would result in choosing node E, which yields a net of 18 units. The nearest neighbor, k, then is node B. Since $|\ell(\pi_\xi^\xi)| = 3$, which is less than 11. Go to step 5. In step 5 because 18 > TEMP (which was set = 11 in step 3), set $L_B = \infty$, and go to step 6. Increment i (i = 1 + 1 = 2) and set $T_2 = \{C, E, B,\}$ and go to step 4. The least-cost path from T_2 to the remaining disjoint positively weighted node

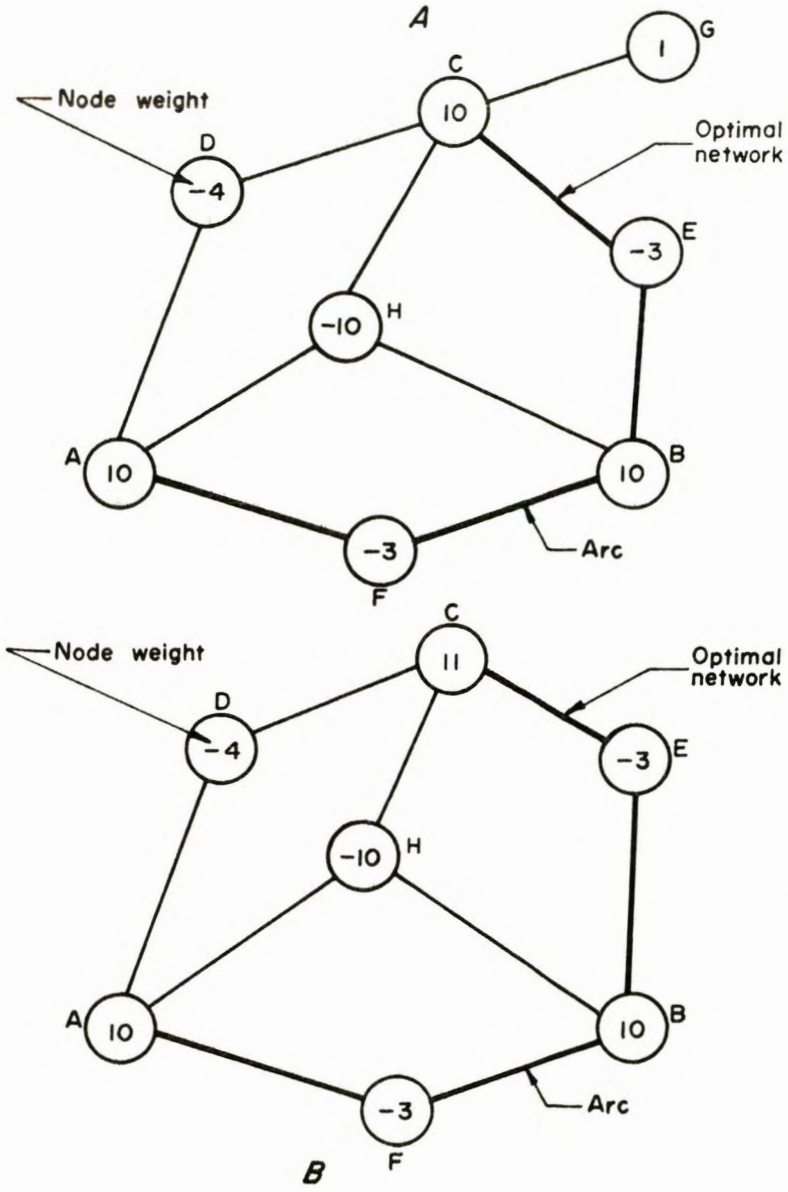


FIGURE 4. - Graphic illustration for problem *(A)* and *(B)*.

(k = A) would be the node {A}, which yields a net of 18 + (10 - 3) = 25, which is greater than TEMP (18). Therefore in step 5 let $L_A = \infty$ and proceed to step 6, to step 7, to step 2 and finally to step 3 where L_A, L_B, $L_C = \infty$, which terminates phase 1 part of the algorithm. If it were desirable to force the solution through the Steiner node H (phase 2), a large positive value, M, would be added to the node weight of H, which in this instance would be -10 + M. Returning to step 1, phase 1, and passing through the phase 1 algorithm again will yield a net of 21 + M. Subtracting M from 21 + M gives an answer of 21, which would be <25. Thus for this problem, figure 4-B, because there is only one possible Steiner node, the solution, {C, E, B, F, A}, is optimal.

To illustrate the two possible alternative kinds of things that can cause the phase 1 to yield a mathematically nonoptimal answer, refer to figure 5. Without the introduction of a Steiner node, the problem in figure 5-A would result in a solution, similar to that found in figure 4-B, yielding a weight of 25; however, for this problem, the network {H, A, B, C} yields 26 units, which is >25. By adding a large positive weight, M, to the node weight of H (-5 + M) the phase 1 algorithm would yield 26 + M, less M is 26, which is >25. Thus the optimal graph G_o = {H, A, B, C} has a total weight of 26. Figure 5-B illustrates a situation where the positive disjoint nodes are not of sufficient weight to allow the algorithm to yield an optimal answer without forcing the Steiner node into the solution. Of interest is that for the phase 1 algorithm not to yield an optimal solution, the optimal graph G_o must contain a Steiner node. Detection of Steiner nodes is left to the user in phase 2. However, the author feels that in a real problem it should not be

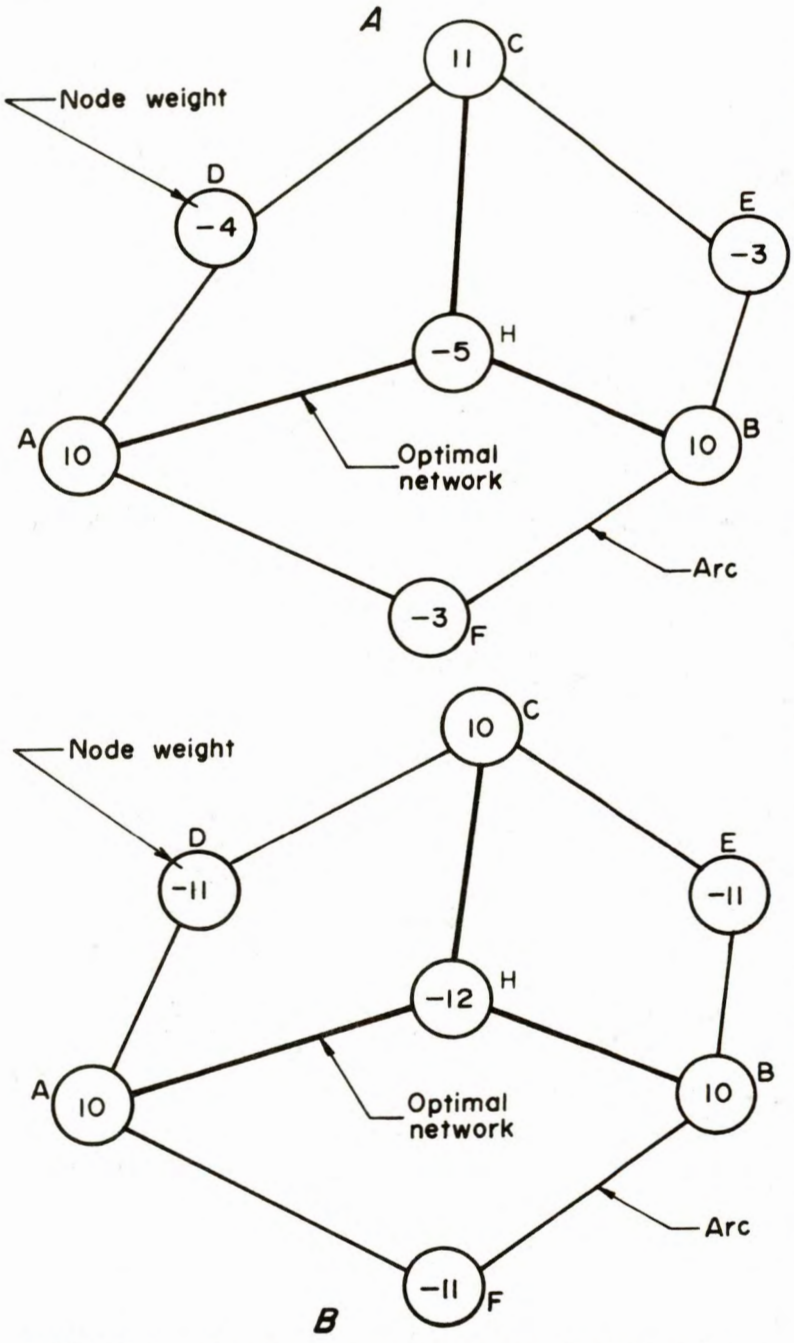


FIGURE 5. - Graphic illustration for problem *(A)* and *(B)*.

too difficult to locate the most obvious location of Steiner nodes. Mathematically, then, the described algorithm will yield an optimal given that no Steiner-type nodes need be introduced.

Algorithm Structure

Given that we have an optimal graph, G_o, to the ultimate mining layout previously stated on a network, there are several characteristics that the solution will have. They are as follows:

Statement 1--All adjacent positive nodes i and j will either be completely within the optimal solution or completely outside.

Statement 2--Considering only the positive disjoint nodes, $\overline{v}_i$, such that i=1 . . . P and an absence of Steiner nodes, the optimal path connecting these P disjoint nodes must be a maximum spanning tree.

Statement 3--There will be, at most, P-2 Steiner nodes for a problem with P positive disjoint nodes $\overline{v}_i$.

Statement 1 may easily be shown to be true. If we assume that two adjacent positive weighted nodes are not both in the optimal solution, the solution is obviously not optimal because a better solution could be found by simply including the adjacent node not presently in the solution set.

The second statement implies that the optimal solution, given no Steiner nodes, is a maximal spanning tree. First, the optimal must be a tree because

if it were not, at least one node (and its incident arcs) having a negative weight could be dropped without destroying the connectivity of the nodes in the set $\{\overline{v}_1\}$, thus a better solution is possible. A spanning tree is here defined as the tree consisting of only the negatively weighted nodes connecting all of the nodes belonging to the set $\{\overline{v}_1\}$. Secondly, given an optimal spanning tree, G_o, assume a path not found in the optimal tree between some pair of nodes in the set $\{\overline{v}_1\}$, which is better than some path in G_o. This assumed path together with some set of arcs and nodes in G_o will form a cycle. The length of the assumed path must be less negative than those paths connecting paired nodes in the set $\{\overline{v}_1\}$ if it is to lead to an improvement. Because this path is not in G_o, it must be strictly greater than the other possible paths. Therefore, by contradiction of the assumption, G_o must be a maximal spanning tree.

Repeated usage of the least-cost algorithm constructs a stage-wise spanning tree for the set of disjoint positively weighted nodes. However, it is possible that the general algorithm will construct a forest (more than one tree) in which case the maximum graph, G_o, will occur within some subset of $\overline{v}_1$-type nodes. In other words, the general algorithm terminates whenever the spanning tree for some subset results in a negative contribution. In this situation the algorithm simply picks the maximum from the subset of spanning trees.

In terms of the general problem, the algorithm itself may introduce Steiner nodes. Therefore, statement 2 is not a strong statement. The usefulness of this statement--given that all of the Steiner type nodes can be defined--is that the algorithm herein (phase 1) will yield an optimal solution.

The third statement can also be easily shown to be true because the number of arcs incident to Steiner nodes must be at least three. Any more than three will reduce the number of possible Steiner nodes for a fixed number of P disjoint positively weighted nodes. The total number of Steiner nodes S must be strictly less than S*, which will be here defined as the maximum number for a fixed number P of positively weighted nodes. Therefore, using the basic properties of a tree the total number of arcs may be given as $\frac{3S^* + P}{2}$, which is equivalent to S* + P-1 nodes. Setting these equal to each other and solving for S* in terms of P yields S*=P-2; therefore, in general the number of possible Steiner nodes S in any problem must be less than or equal P-2. This completes the proof.

The least-cost path algorithm, upon which the ultimate underground problem is based, is similar to the labeling algorithm first described by Dijkstra, Whiting, and Hillier as quoted by Dreyfus (6). The primary difference between what is written by Dreyfus and least-cost path algorithm described herein is how the weights are defined. For the mining problem all weights are defined on the nodes as opposed to the arcs for the standard shortest route algorithm. In addition, Dreyfus states the shortest route problem is a minimization technique compared with a maximization for the mining problem. He, also, defines positively weighted arcs (it is sufficient to have no negative cycles) as essentially a set of negatively weighted nodes in the mining problem.

The labeling algorithm, described by Dreyfus, works with two mutually exclusive and totally exhaustive sets of nodes, either permanently labeled or temporarily labeled nodes. At each iteration of the labeling algorithm, one node is transferred to the permanently labeled set and the algorithm terminates when there are no more nodes left in the temporary set. The algorithm thus constructs a maximal spanning tree.

Proof of the least-cost labeling algorithm, as used in this paper, may be made by induction. It is necessary to remember that the least-cost algorithm is defined completely on a set of negatively weighted nodes. At any stage in the least-cost algorithm the nodes may be classified into two sets--one containing the labeled set of nodes and another containing all the unlabeled sets

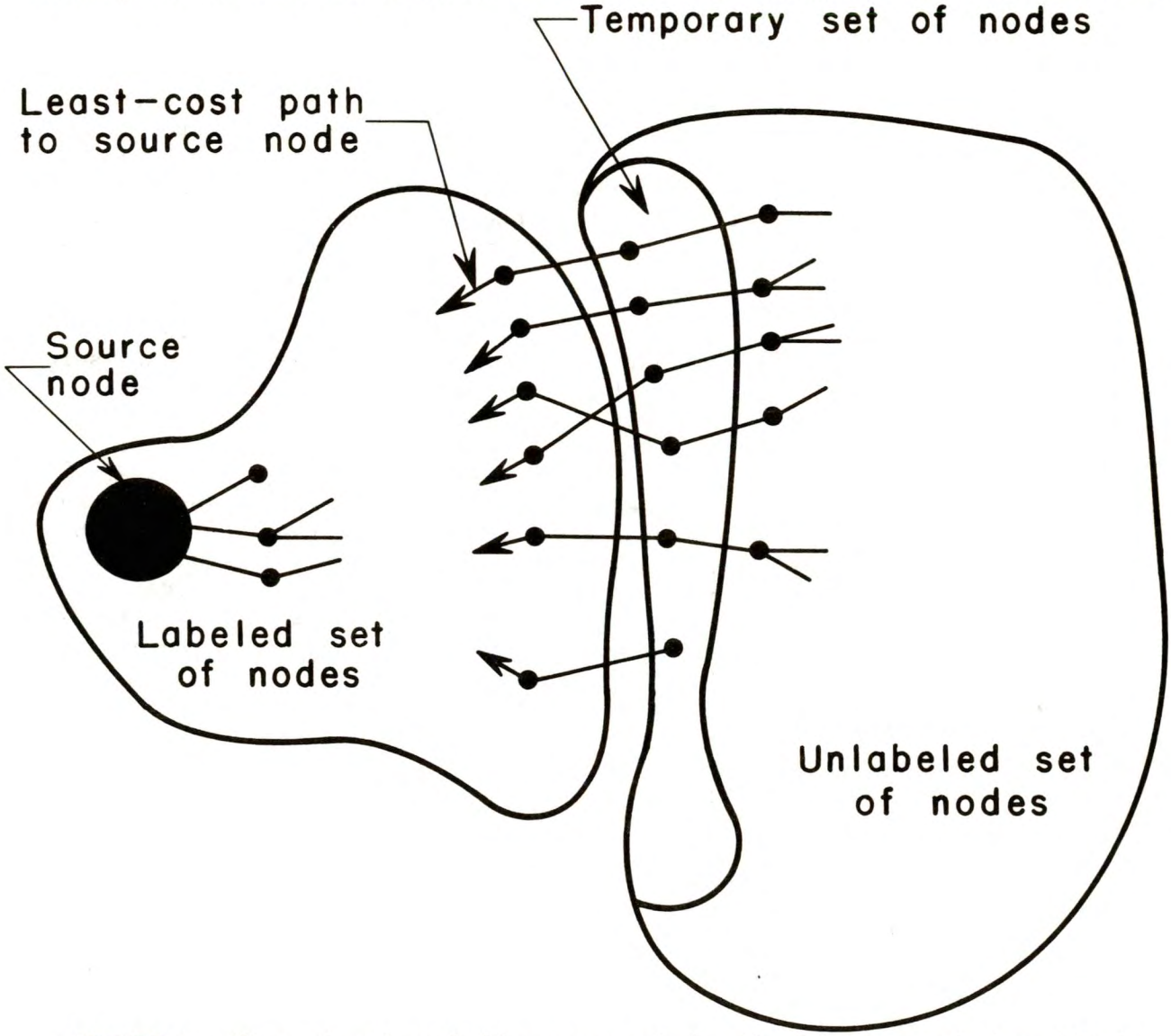


FIGURE 6. - Illustration of proof of least-cost path algorithm for only negatively weighted nodes.

of nodes (fig. 6). Labeled nodes are those for which a least-cost has been defined to a source node. For all those nodes in the unlabeled set adjacent to the labeled set, it is possible to find the least-cost distance to the source node such that all nodes in the least-cost path will be in the labeled set except one which will be in the unlabeled set. This distance is only temporary. Obviously, then the node in the unlabeled set which has the minimum temporary distance may be transferred to the labeled set. For if it is assumed that there exists a better less costly path, it would have to exist as a path in the unlabeled set. Because all of the nodes are negatively weighted, this path does not exist.

IMPLEMENTATION

Application to a Coal Property

Setting up a problem to be solved using this basic approach demands a certain amount of innovation on behalf of the user. For example, a user might wish to allocate to a block having length, width, and depth a portion of the unmined block for ground support. In terms of the network equivalent, this will be represented by a node weight. In the event of an uneconomic block (negative weight), the user might choose to consider only the cost of driving development entries through the block. On the other hand, a user might wish to define a block as a minable unit equivalent to net worth or a mine support unit and assign a large negative cost. The block weight need not be a cost; it could be the risk of roof fall in a particular block and the design engineer wishes to find the best development entry layout between two or more points having the least chance of a roof fall.

An illustrative problem is given in figure 7 for a hypothetical coal mine. For this illustration the problem is contained entirely within the coal seam for clarity of presentation; however, a three-dimensional model might be much more realistic. The square is a realistic block configuration, as a coal seam is normally mined consistent with a rectilinear movement. For a coal seam the coordinate directions would, more than likely, be structured so that they coincide with butts and cleats (planes of weakness) in the coal seam, but not necessarily. This would be dependent upon ground control problems. The block size has been taken equal to 800 by 800 feet with block weight equivalent to the profit or loss to the mining company. Block weights might be entirely different for another company. A block in which the net worth is less than zero for this problem indicates that as a minimum requirement, an access (set of entries) be driven through the block. This assumes that the cost of mining in either direction is the same, whereas, in reality this might not be precisely true.

If there does not exist a natural access to the surface, as is the situation for many mineral deposits, it would be necessary to assign some subset of surface blocks a large positive weight. This would force whatever material that could be economically mined to be connected to the surface. In addition, if it were desirable for some reason to force the mine through a particular set of blocks, this could be accomplished by adding an artificial block weight to those blocks, which would be subtracted later. To force access to the

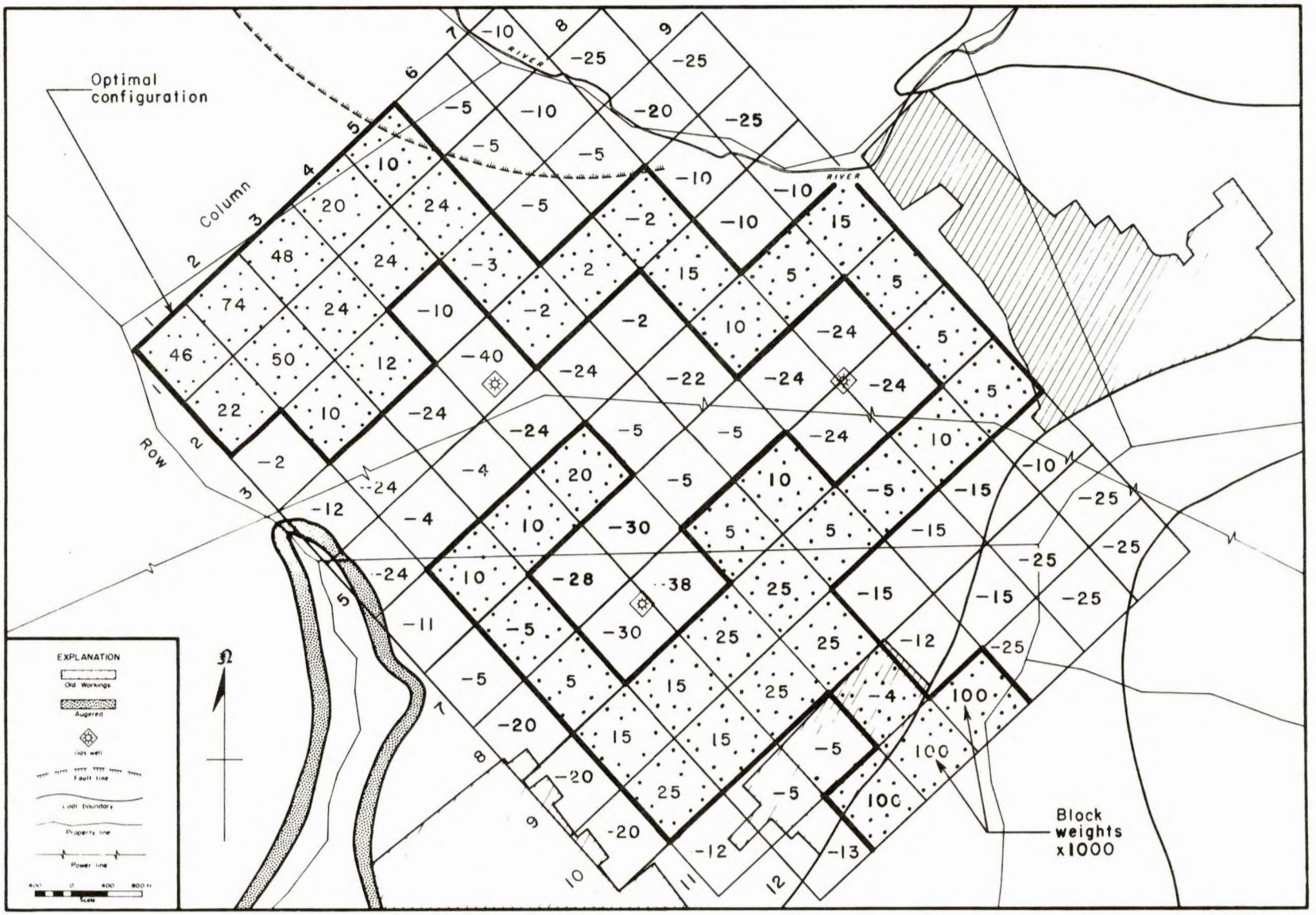


FIGURE 7. - Two-dimensional application for a coal property.

surface in this problem, blocks (columns 4 through 6 in row 12) were given an artificial weight equal to $100,000 each. From a practical viewpoint these blocks might be considered as potential sites of a mine plant or mine portal. For this problem three oil wells are located in the coal seam of interest. The cost of mining through these blocks considered such costs as plugging the wells or the uncertainty of the well location within the block. The Coal Mine Safety Act of 1969[4] specifies that a barrier of 300 feet must be maintained around an active or inactive oil or gas well. This fact is reflected in the weights of the blocks within the neighborhood of oil wells. The physical and chemical properties of the coal, ownership, nearness to powerlines, location of old mine workings, chemical and physical properties of the coal, etc., all had a bearing on the assigned block weight. Results of this particular problem are given in appendix B.

Of interest in this particular problem is that the results would not be consistent with one's intuitive concepts of mine layout. However, by strategically reassigning block weights and repetitive use of algorithm herein described, the user could study many alternative plans at a minimal cost. Varying the block size would allow the user to compare different methods of mining.

Computational Experience

A computer program was developed in Fortran IV for a Bureau of Mines Burroughs 5500 in Denver. A copy of the computer program is included in appendix C. The computer program is written to handle a problem with a maximum of 6,000 blocks. The upper bound on the problem size is only dependent upon the size of core available. The processing time was found to be primarily dependent upon the total number of blocks, and the number and distribution of disjoint positive blocks. Table 1 gives the results for four different size problems.

TABLE 1. - Computation results for four different size problems

Problem size			Number disjoint blocks	Computational times (sec)	
I	J	K		Processor	I. O.
6	6	1	3	3	14
12	9	1	5	7	14
5	20	3	17	38	16
10	10	6	2	33	14

Input requirements consist primarily of the block weights, problem parameter, problem title, and the output needs. Input must be prepared in the following sequence and contain the following information:

[4]Mandatory Safety Standards, Underground Coal Mines. Nov. 20, 1970, Federal Register, v. 35, No. 226, Sec. 75.1700.

Card 1--Option card:

Column	Variable name	Description
1 - 5	LIST	1 ⇒ List block weights
6 - 10	PLOT	1 ⇒ Plot disjoint positive blocks and negative path in solution

Card 2--Problem name:

Column	Variable name	Description
1 - 78	FMT	Problem title (13A6)

Card 3--Data format:

Format of data on cards (left justify and include brackets).

Card 4--Parameter card:

Column	Variable name	Description
1 - 5	IJK (1) - IMAX	Number rows
6 - 10	IJK (2) - JMAX	Number columns
11 - 15	IJK (3) - KMAX	Number sections

Card 4--Data . . .

Data [(I, J, K), I = 1, IMAX, J = 1, JMAX, K = 1, KMAX)].

Program stores data in a one-dimensional array, block (I).

Output is under program control to the extent that the block weights may be printed or suppressed as can the plot of the solution. Included in appendix B is a copy of the output for the sample problem given in figure 6.

CONCLUSIONS AND EXTENSIONS

The algorithm defined in this report offers the mining engineer a simple technique for long-range mine planning. In addition through the use of a computer program, the technique was demonstrated to be programable for solving an orthogonal, three-dimensional set of blocks and reasonably efficient for those problems tested. In view of the lack of comparable studies, it is exceedingly difficult to fairly evaluate this work; however, the author is extremely optimistic as to the potential of this study.

Such factors as time value of money can be taken into account by making allowance in the calculation of block weights. This forces the individual to design for the long term consistent with conservation of resources. In mining, development costs generally are higher than production costs, and, therefore, the faster a mine can be brought into production, the greater will be its present worth. However, forcing a mine into early production may have the effect of increasing future development costs. In fact, favoring production at the expense of development may lead to the loss of a potential resource.

The general procedure given in this paper will have application to a wide range of subsurface mining-type problems. Interpretation of block weights could be other than a dollar cost; for example, it could represent a risk or some estimate of hazard to be encountered in a particular block. Thereby, the algorithm could be used to study possible escape routes when a disaster has occurred at some place in the mine network.

This general algorithm is constrained in that no account is made for the direction in which a block is to be extracted. For example, it would be less costly to extract a block from the bottom as opposed to extracting it from the top or side. In fact, mining costs might be closely related to local fracture or joint patterns. The problem of mining cost might be best accomplished by assigning arc weights in addition to the node weights, allowing the arc weights to represent the mining cost of mining blocks j from i and the node weight, the block mineral worth.

A problem presented by the orthogonal network as developed herein for the computer program is that the alternatives for moving from block to block is constrained to an orthogonal movement (forwards and back, left or right, up or down). Without the loss of generality--specifically with respect to the computer program--the basic network could be modified to allow a user to move diagonally across blocks or possibly even skip particular blocks if the strategy for block jumping could be generalized. Use of other basic geometric configurations would also give the user more flexibility. Use of hexagons in a two-dimensional space, for example, allows the user six degrees of freedom to advance from a particular block compared with only four in a plane for which the computer program was written. Figure 2 and the book by Cole and King (5) indicate other possible geometries. The importance of using an orthogonal set of blocks lies in the fact that because of the highly structured resulting network, there is no need to store an elaborate arc incident matrix.

Much can be done in defining a strategy for designing a production scheduling sequence for a given ore zone or a safe pillar configuration for a given mine by modifying the basic orthogonal network presented in this paper.

The reader should be alerted that the mining problem should be investigated as a total system as the ultimate layout is but one small problem; however, due to the complexity of the underground problem, and the usual limited data base available on a property prior to mining, a piecemeal approach does seem appropriate dependent upon design requirements. For example, facility location and scheduling problems might or might not be of particular importance depending upon the particular property. This approach gives the

minerals industry a tool for studying alternatives to mine layout; it does not specifically answer to the facility location (15), or to the general problem of mine layout with respect to development and production faces, nor does it answer to the problem of scheduling a property to meet production demands and milling constraints (2).

REFERENCES

1. Becker, R. M., and Scott W. Hazen, Jr. Particle Statistics of Infinite Populations as Applied to Mine Sampling. BuMines RI 5669, 1961, 79 pp.

2. Blanks, C. Production Scheduling of a Room and Pillar Mining Method. Colo. School of Mines, Thesis No. 1437, 1973.

3. Canadian Institute of Mining and Metallurgy. Decision-Making in the Mineral Industry. Special v. 12, 1971, 507 pp.

4. Cockayne, E. J. On the Efficiency of the Algorithm for Steiner Minimal Trees. Siam J. Appl. Math., v. 18, No. 1, 1970, pp. 150-159.

5. Cole, John P., and Cuchlaine A. M. King. Quantitative Geography. John Wiley & Sons, Inc., New York, 1968, 644 pp.

6. Dreyfus, Stuart E., and Robert A. Wagner. The Steiner Problem in Graphs. Univ. of Calif., Berkeley, September 1970, 14 pp.

7. Ford, L. R., Jr., and D. R. Fulkerson. Flow in Network. Princeton Univ. Press, 1962, pp. 130-134.

8. Frank, Howard, and Ivan T. Frisch. Communication, Transmission, and Transportation Networks. Addison-Wesley Publishing Co., 1971, pp. 9-29, 192-198.

9. Ganthier, F. J., and R. G. Gray. Pit Design by Computer at Gaspe Copper Mines, Limited. Can. Min. and Met. Bull., November 1971, pp. 95-102.

10. Gilbert, E. M., and H. O. Pollack. Steiner Mineral Trees. Siam J. Appl. Math., v. 16, No. 1, 1968, 29 pp.

11. Hu, T. C. Integer Programming and Network Flows. Addison-Wesley Publishing Co., 1969, pp. 151-153.

12. Johnson, Thys B. Optimal Open Pit Mine Production Scheduling. Univ. of Calif., Berkeley, May 1968, 120 pp.

13. Johnson, T. B., and W. R. Sharp. A Three-Dimensional Dynamic Programing Method for Optimal Ultimate Open Pit Design. BuMines RI 7553, 1970, 25 pp.

14. Lerchs, Helmut, and I. F. Grossmann. Optimal Design of Open-Pit Mines. Can. Min. and Met. Bull., v. 633, January 1965, pp. 47-54.

15. Nair, Oscar B. Optimal Location of Mine Shafts. Colo. School of Mines, Thesis No. 1468, 1972.

16. Roberts, E. H. Design of New Mines To Comply With Modern Standards. Min. Cong. J., May 1971, pp. 29-32.

17. Zambo, J. Optimal Location of Mine Facilities. Muszaki Konynkiado (Budapest), 1966, 144 pp.

APPENDIX A.--TABLE OF SYMBOLS

Symbol	Description
A, B, C, D, E, F, G..	Nodes (used in illustrations)
AMAX.....	Maximum mine plan for a given problem
ATEMP.....	Temporary value (used in algorithm)
b_i.....	An arc i
G.....	A graph consisting of a set of nodes V and arcs
G_o.....	Optimal subgraph
i, k, q, s, t.....	Indexes 1, 2 . . ., etc.
$l(\overrightarrow{ij})$.....	Length of path from node i to node j
L(i).....	A label for disjoint positive blocks 1 . . . P
$L(s, l(\overrightarrow{tq}))$.....	A label for node t labeled from node s where $l(\overrightarrow{tq})$ is the least-cost path from node t to node q
M.....	Large positive number
m.....	Total number of vertices
n.....	Total number of arcs
P.....	Number of positive disjoint blocks (does not include Steiner nodes)
S.....	Number of Steiner nodes
T_o.....	Optimal tree (for a given set of node weights)
T_i.....	Tree i
$\overline{T}$.....	Temporary tree
TEMP.....	Temporary value (used in algorithm)
W_i.....	Weight of block i
$w(v_i)$.....	Weight assigned to node v_i
V.....	Set of nodes
v_i.....	Node i

Symbol	Description
$\bar{v}_i$	Node i such that v_j belongs to $\bar{v}_i$ for some set of j
X_i	Block i
Γ	Set of arcs
$\prod_j$	Product over all j
π_s^t	Path from node s to node t inclusive
$\sum_j$	Summation over all j
$\mid$	Such that
Ω	Empty set

APPENDIX B.--INPUT AND RESULTS OF MINING PROBLEM

B-1.--Card Format for the Problem

(SAMPLE)

PROJECT NO.________ **CODED BY**________ **DATE**________ GPO 859764

REMARKS (TYPE OF CARD, INTERPRETING, LISTING, ETC.)________

NOTIFY________ Name Address Ph.
WHEN CARDS ARE READY.

DATE REQUIRED______ **PAGE No.**____ **of**____ **Pages**

```
    1    1
CARBON COAL MINE NO I - HYPOTHETICAL PROBLEM I
(12F5.1)
   12    9    1
 46.0 22.0 -2.0-12.0-24.0-11.0 -5.0-20.0-20.0-20.0-25.0-25.0
 74.0 50.0 10.0-24.0 -4.0 10.0 -5.0  5.0 15.0 25.0-12.0-25.0
 48.0 24.0 12.0-24.0 -4.0 10.0-28.0-30.0 15.0 15.0 -5.0-13.0
 20.0 24.0-10.0-40.0-24.0 20.0-30.0-38.0 25.0 25.0 -5.0100.0
 10.0 24.0 -3.0 -2.0-24.0 -5.0 -5.0  5.0 25.0 25.0 -4.0100.0
 -5.0 -5.0 -5.0  2.0 -2.0-22.0 -5.0 10.0  5.0-15.0-12.0100.0
-10.0-10.0 -5.0 -2.0 15.0 10.0-24.0-24.0 -5.0-15.0-15.0-25.0
-25.0-25.0-20.0-10.0 10.0  5.0-24.0-24.0 10.0-15.0-25.0-25.0
-25.0-25.0-25.0-25.0-10.0 15.0  5.0  5.0  5.0-10.0-25.0-25.0
```

B-2.--Output Given by the Computer

CARBON COAL MINE NO 1 - HYPOTHETICAL PROBLEM 1

I = 12 J = 9 K = 1

INPUT

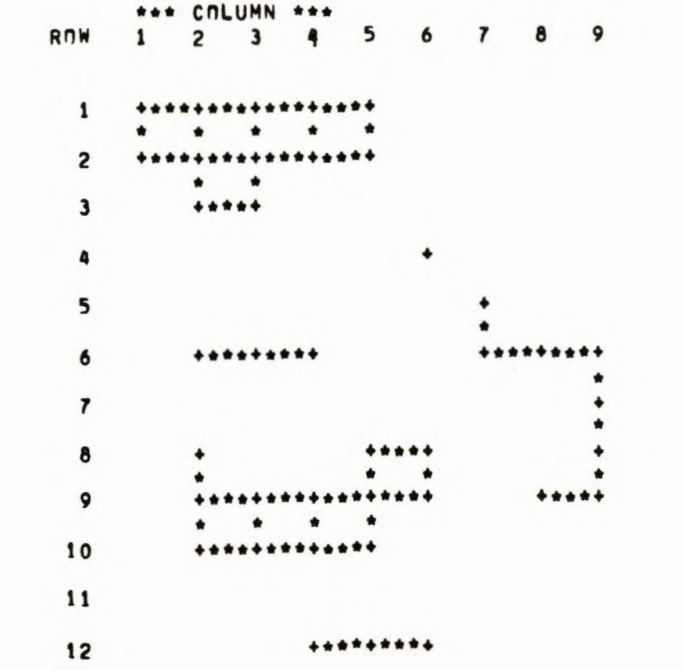
```
46.0 22.0 -2.0-12.0-24.0-11.0 -5.0-20.0-20.0-20.0-20.0-20.0
74.0 50.0 10.0-24.0 -4.0 10.0 -5.0  5.0 15.0 25.0-12.0-20.0
48.0 24.0 12.0-24.0 -4.0 10.0-28.0-30.0 15.0 15.0 -5.0-20.0
20.0 24.0-10.0-40.0-24.0 20.0-30.0-38.0 25.0 25.0 -5.0  1.0
10.0 24.0 -3.0 -2.0-24.0 -5.0 -5.0  5.0 25.0 25.0 -4.0  1.0
-5.0 -5.0 -5.0  2.0 -2.0-22.0 -5.0 10.0  5.0-15.0-12.0100.0
10.0-10.0 -5.0 -2.0 15.0 10.0-24.0-24.0 -5.0-15.0-15.0-20.0
20.0-25.0-20.0-10.0-10.0  5.0-24.0-24.0 10.0-15.0-20.0-20.0
20.0-25.0-25.0-25.0-10.0 15.0  5.0  5.0  5.0-10.0-20.0-20.0
```

PLOT OF POSITIVE NODES

TABLE OF SYMBOLS

SYMBOL	EXPLANATION
+	POSITIVE NODE
*	CONNECTIVITY IN I - J PLANE
U	CONNECTIVITY WITH K-1 PLANE
D	CONNECTIVITY WITH K+1 PLANE
B	CONNECTIVITY WITH ADJACENT PLANES

SECTION 1



POSITIVE DISJOINT TREES * INDEX = I + (J-1) * MAX(J) + (K-1) * MAX(K)

TREE	VALUE	INDEX	** TREE BOUNDARIES						** CENTER OF GRAVITY **			** CENTER OF UTILITY **		
			MIN I	MAX I	MIN J	MAX J	MIN K	MAX K	I	J	K	I	J	K
1	364.0	1	1	3	1	5	1	1	2	3	1	2	3	1
2	40.0	18	6	6	2	4	1	1	6	3	1	6	3	1
3	195.0	20	8	10	2	6	1	1	9	4	1	9	4	1
4	102.0	48	12	12	4	6	1	1	12	5	1	12	6	1
5	2.0	64	4	4	6	6	1	1	4	6	1	4	6	1
6	-70.0	77	5	9	7	9	1	1	7	8	1	7	8	1

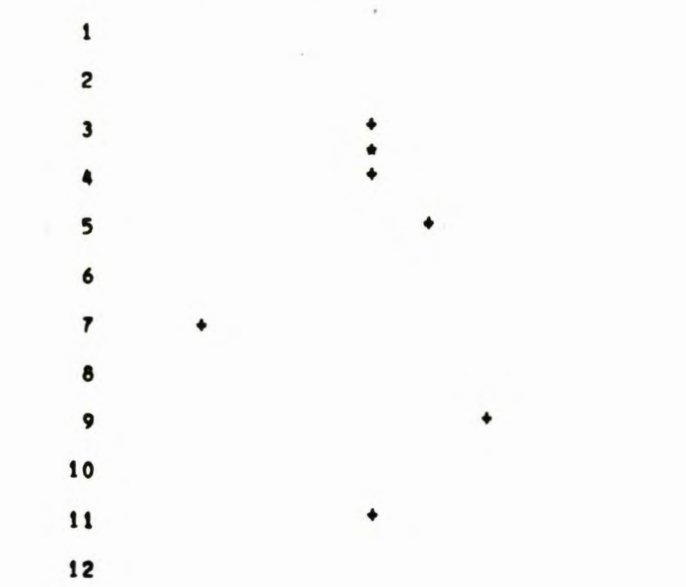
ITERATION	TREE	VALUE
1	1	364.0
1	6	429.0
1	3	619.0
1	4	717.0
1	2	752.0

PLOT OF POSITIVE NODES IN ULTIMATE PLAN

TABLE OF SYMBOLS

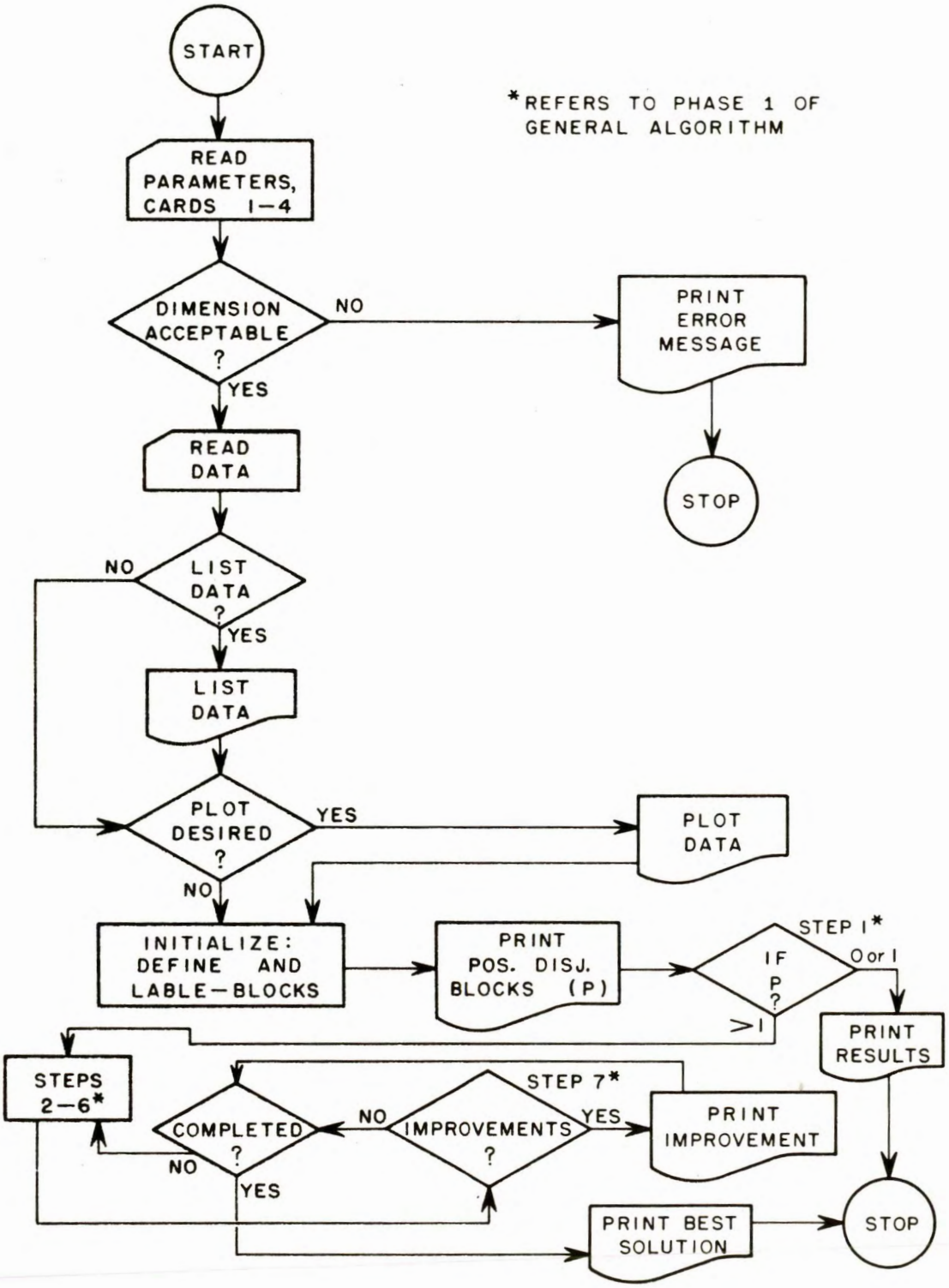
SYMBOL	EXPLANATION
+	POSITIVE NODE
*	CONNECTIVITY IN I = J PLANE
U	CONNECTIVITY WITH K-1 PLANE
D	CONNECTIVITY WITH K+1 PLANE
B	CONNECTIVITY WITH ADJACENT PLANES

SECTION 1



APPENDIX C.--COMPUTER PROGRAM

C-1.--Flow Chart of the Computer Code



C-2.--Output Given by the Computer

```
                    TUESDAY,  9/ 5/72, 14:05 HRS.

DENVER  INTERIOR  COMPUTER  CENTER  *** B-5500  F O R T R A N  COMPILER ***  XII. 3,

C      OPTIMAL TREE FOR A CONSTRAINED 3-DIMENSIONAL SYSTEM OF BLOCKS      00000200 R  0000
C      PROGRAMMER = W,R,SHARP                                             00000300 R  0000
C                                                                         00000400 R  0000
C      CARD 1 - OPTION CARD                                               00000500 R  0000
C      CARD 2 - HEADER CARD                                               00000600 R  0000
C      CARD 3 - DATA FORMAT                                               00000700 R  0000
C      CARD 4 - PARAMETER CARD                                            00000800 R  0000
C         IMAX = NO ROWS                                                  00000900 R  0000
C         JMAX = NO COLUMNS                                               00001000 R  0000
C         KMAX = NO SECTIONS                                              00001100 R  0000
C      DATA                                                               00001200 R  0000
C                                                                         00001300 R  0000
C      VARIABLE LIST                                                      00001400 R  0000
C         MAXJ = NO BLOCK IN SECTION                                      00001500 R  0000
C         MAXK = TOTAL NO BLOCKS                                          00001600 R  0000
C                                                                         00001700 R  0000
                                                    START OF SEGMENT **********    1
      COMMON BLOCK(6000),ITREE(6000),LBK(6000),SORT(1020),ITAG(1020),    00001800 R  0000
     1NPR(101),NPL(30),IDA(6),JDA(6),KDA(6),IJK(6),SUM(30),MAXJ,MAXK,    00001900 R  0000
     2ICON,NPOS(30),IT1(30),KTAG(1020),ADD(120),JTAG(120),ARRAY(1020),   00002000 R  0000
     3IZAP(60),NPATH(1020),BARRAY(1020),KKTAG(1020),NBEST(1020),         00002100 R  0000
     4LABEL(30)                                                          00002200 R  0000
      DIMENSION FMT(13)                                                  00002300 R  0000
C                                                                        00002400 R  0000
C     INITIALIZE                                                         00002500 R  0000
C                                                                        00002600 R  0000
      ICON = 6000                                                        00002700 R  0000
      ISEG = 30                                                          00002800 R  0003
      JSEG = ISEG - 1                                                    00002900 R  0004
      ZERO = 0,                                                          00003000 R  0005
      IDA(1) = 1                                                         00003100 R  0006
      DO 5 I = 2,6                                                       00003200 R  0008
    5 IDA(I) = IDA(I-1) * 10                                             00003300 R  0014
      KDA(1) = 1                                                         00003400 R  0026
      KDA(2) = 100                                                       00003500 R  0028
      KDA(3) = 10000                                                     00003600 R  0030
      KDA(4) = 10                                                        00003700 R  0034
      KDA(5) = 1000                                                      00003800 R  0036
      KDA(6) = 100000                                                    00003900 R  0038
C                                                                        00004000 R  0039
C     INPUT TO TREE                                                      00004100 R  0039
C                                                                        00004200 R  0039
      READ 1000 , LIST,IPLOT                                             00004300 R  0042
 1000 FORMAT(3I5,F10.0)                                                  00004400 R  0055
      READ 1001, FMT                                                     00004500 R  0055
 1001 FORMAT(13A6)                                                       00004600 R  0066
      PRINT 1002, FMT                                                    00004700 R  0066
      READ 1001, FMT                                                     00004800 R  0076
 1002 FORMAT(1H1/1X,13A6//)                                              00004900 R  0087
C                                                                        00005000 R  0087
      READ 1000, (IJK(I),I=1,3)                                          00005100 R  0087
      IMAX = IJK(1)                                                      00005200 R  0108
      JMAX = IJK(2)                                                      00005300 R  0109
      IF(JMAX .GT. 1) GO TO 7                                            00005400 R  0111
      PRINT 1005                                                         00005500 R  0113
```

```
 1005 FORMAT(1X,17HJ INDEX TOO SMALL )                                  00005600 R  0118
      GO TO 2000                                                        00005700 R  0118
    7 KMAX = IJK(3)                                                     00005800 R  0119
      DO 8 I = 4,6                                                      00005900 R  0120
    8 IJK(I) = 1                                                        00006000 R  0127
      PRINT 1004, (IJK(I),I=1,3)                                        00006100 R  0133
 1004 FORMAT(10X,5HI =  ,I5,3X,5HJ =  ,I5,3X,5HK =  I5)                 00006200 R  0155
C                                                                       00006300 R  0155
      MAXJ = IJK(1) * IJK(2)                                            00006400 R  0155
      MAXK = MAXJ * IJK(3)                                              00006500 R  0159
      IF(IJK(3) .EQ. 0) MAXK = MAXJ                                     00006600 R  0163
      IF(IJK(1) .LE. ICON) GO TO 10                                     00006700 R  0168
      PRINT 1003                                                        00006800 R  0173
 1003 FORMAT(1X,25HTOO MANY BLOCK IN I ARRAY)                           00006900 R  0177
      GO TO 2000                                                        00007000 R  0177
   10 INDEX = 0                                                         00007100 R  0178
      IF(LIST .NE. 0) PRINT 1010                                        00007200 R  0178
 1010 FORMAT(/ 6X, 5HINPUT/)                                            00007300 R  0184
      IF(MAXK .LT. ICON) GO TO 20                                       00007400 R  0184
      PRINT 1006                                                        00007500 R  0188
 1006 FORMAT(1X,15HPROBLEM TOO BIG )                                    00007600 R  0192
      GO TO 2000                                                        00007700 R  0192
   20 LMAX = MAXK                                                       00007800 R  0193
      READ FMT, (BLOCK(J), J=1,LMAX)                                    00007900 R  0194
      IF(LIST .EQ. 0) GO TO 40                                          00008000 R  0215
      PRINT FMT, (BLOCK(J),J=1,LMAX)                                    00008100 R  0217
   40 IF(IPLOT .EQ. 0) GO TO 100                                        00008200 R  0237
C                                                                       00008300 R  0238
C     FIND ALL POSITIVE TREES                                           00008400 R  0238
C                                                                       00008500 R  0238
      PRINT 1111                                                        00008510 R  0239
 1111 FORMAT(1H1,1X,23HPLOT OF POSITIVE NODES  //)                      00008520 R  0243
   50 CALL PLOT                                                         00008600 R  0243
C                                                                       00008700 R  0243
  100 CALL BEGIN(NSORT,MTREE,IMAX,JMAX,KMAX)                            00008800 R  0244
      IF(MTREE - 1) 61,65,70                                            00008900 R  0245
   61 PRINT 1012                                                        00009000 R  0251
 1012 FORMAT(1X,24HNO POSITIVE BLOCK EXISTS)                            00009100 R  0255
      GO TO 2000                                                        00009200 R  0255
   65 PRINT 1013, SUM(1)                                                00009300 R  0256
 1013 FORMAT(1X,45HONLY ONE POSITIVE SUBTREE EXISTS WHICH YIELDS,F12,2) 00009400 R  0267
      GO TO 2000                                                        00009500 R  0267
   70 CALL MAXTRE(MTREE,IMAX,JMAX,KMAX)                                 00009600 R  0268
 2000 CALL EXIT                                                         00009700 R  0270
      END                                                               00009800 R  0271
                                                                       SEGMENT    1 IS  282 LONG
```

```
                                                                    START OF SEGMENT **********   2
      SUBROUTINE BEGIN(NSORT,MTREE,IMAX,JMAX,KMAX)                                  00009900 R  0000
      COMMON BLOCK(6000),ITREE(6000),LBK(6000),SORT(1020),ITAG(1020),               00010000 R  0000
     1NPR(101),NPL(30),IDA(6),JDA(6),KDA(6),IJK(6),SUM(30),MAXJ,MAXK,               00010100 R  0000
     2ICON,NPOS(30),IT1(30),KTAG(1020),ADD(120),JTAG(120),ARRAY(1020),              00010200 R  0000
     3IZAP(60),NPATH(1020),BARRAY(1020),KKTAG(1020),NBEST(1020),                    00010300 R  0000
     4LABEL(30)                                                                     00010400 R  0000
C                                                                                   00010500 R  0000
      JDA(1) = 1                                                                    00010600 R  0000
      JDA(2) = IJK(1)                                                               00010700 R  0002
      JDA(3) = MAXJ                                                                 00010800 R  0005
      JDA(4) = -1                                                                   00010900 R  0008
      JDA(5) = -IJK(1)                                                              00011000 R  0010
      JDA(6) = -MAXJ                                                                00011100 R  0013
      DO 9 I= 1,30                                                                  00011200 R  0016
    9 SUM(I)= 0.                                                                    00011300 R  0023
      NTREE = 0                                                                     00011400 R  0029
      MTREE = 0                                                                     00011500 R  0030
      DO 10 I = 1,MAXK                                                              00011600 R  0030
   10 ITREF(I) = 0                                                                  00011700 R  0038
C                                                                                   00011800 R  0043
   20 PRINT 1002                                                                    00011900 R  0045
 1002 FORMAT(1H1/1X,69HPOSITIVE DISJOINT TREES * INDEX = I + (J-1) * MAX00012000 R  0049
     1(J) + (K-1) * MAX(K)//4X,4HTREE,5X,5HVALUE,4X,                                00012010 R  0049
     25HINDEX,3X,2H**,10X,18HTREE     BOUNDARIES,10X,2H**,1X,17HCENTER OF00012100 R  0049
     3 GRAVITY,1X,2H**,1X,17HCENTER OF UTILITY,1X,2H**/31X, 42H MIN I   M00012200 R  0049
     4AX I   MIN J   MAX J   MIN K   MAX K   /2(4X,1HI,6X,1HJ,6X,1HK,2X)/)          00012300 R  0049
      DO 30 INDEX = 1,MAXK                                                          00012400 R  0049
      IF(BLOCK(INDEX) .LE. 0.) GO TO 30                                             00012500 R  0055
      IF(ITREE(INDEX) .NE. 0) GO TO 30                                              00012600 R  0061
      GO TO 97                                                                      00012700 R  0069
   30 CONTINUE                                                                      00012800 R  0070
      GO TO 160                                                                     00012900 R  0070
C                                                                                   00013000 R  0070
   91 KK = 7 - I                                                                    00013100 R  0072
      INDEX = INDEX + JDA(KK)                                                       00013200 R  0073
      GO TO 100                                                                     00013300 R  0079
   97 NTREE = NTREE + 1000000                                                       00013400 R  0081
      MTREE = MTREE + 1                                                             00013500 R  0083
      ITEST = 0                                                                     00013600 R  0085
      SO = 0.                                                                       00013700 R  0085
      SI = 0.                                                                       00013800 R  0086
      SJ = 0.                                                                       00013900 R  0087
      SK = 0.                                                                       00014000 R  0088
      SWI = 0.                                                                      00014100 R  0088
      SWJ = 0.                                                                      00014200 R  0089
      SWK = 0.                                                                      00014300 R  0090
      MINX = IJK(1) + 1                                                             00014400 R  0091
      MAXX = 0                                                                      00014500 R  0093
      MINY = IJK(2) + 1                                                             00014600 R  0094
      MAXY = 0                                                                      00014700 R  0096
      MINZ = IJK(3) + 1                                                             00014800 R  0097
      MAXZ = 0                                                                      00014900 R  0099
      IBEG = INDEX                                                                  00015000 R  0100
      GO TO 100                                                                     00015100 R  0100
   99 IF(NDEX .EQ. IBEG) GO TO 130                                                  00015200 R  0102
      INDEX = NDEX                                                                  00015300 R  0104
```

```
100 DO 110 I = 1,6                                                    00015400 R  0105
    NDEX = INDEX + JDA(I)                                             00015500 R  0110
    GO TO (101,102,103,104,105,106) , I                               00015600 R  0116
101 IF(MOD(NDEX,IMAX)-1) 108,110,108                                  00015700 R  0126
102 IF(MOD(INDEX,MAXJ) .EQ. 0) GO TO 110                              00015800 R  0130
    IF(NDEX/MAXJ - INDEX/MAXJ) 110,108,110                            00015900 R  0134
103 IF(NDEX - MAXK) 108,108,110                                       00016000 R  0140
104 IF(MOD(NDEX,IMAX)) 108,110,108                                    00016100 R  0144
105 IF(NDEX .LE. 0) GO TO 110                                         00016200 R  0147
    IF(MOD(NDEX,MAXJ) .EQ. 0) GO TO 110                               00016300 R  0149
    IF(NDEX/MAXJ - INDEX/MAXJ) 110,108,110                            00016400 R  0153
106 IF(NDEX) 110,110,108                                              00016500 R  0159
108 IF(BLOCK(NDEX) .LE. 0.) GO TO 110                                 00016600 R  0162
    IF(ITREE(NDEX) .GT. 0) GO TO 110                                  00016700 R  0167
    ITREE(NDEX) = KDA(I) + NTREE                                      00016800 R  0175
109 SO = SO + 1                                                       00016900 R  0186
    IVAL = MOD(NDEX,IMAX)                                             00017000 R  0187
    IF(IVAL .EQ. 0) IVAL = IMAX                                       00017100 R  0188
    IF(IVAL .GT. MAXX) MAXX = IVAL                                    00017200 R  0191
    IF(IVAL .LT. MINX) MINX = IVAL                                    00017300 R  0193
    SI = SI + IVAL                                                    00017400 R  0195
    SWI = SWI + BLOCK(NDEX)*IVAL                                      00017500 R  0196
    KVAL = (NDEX + MAXJ - 1)/MAXJ                                     00017600 R  0201
    JVAL = (NDEX - MAXJ*(KVAL-1) + IMAX - 1)/IMAX                     00017700 R  0206
    IF(JVAL .GT. MAXY) MAXY = JVAL                                    00017800 R  0210
    IF(JVAL .LT. MINY) MINY = JVAL                                    00017900 R  0213
    SJ = SJ + JVAL                                                    00018000 R  0215
    SWJ = SWJ + BLOCK(NDEX)*JVAL                                      00018100 R  0216
    IF(KVAL .GT. MAXZ) MAXZ = KVAL                                    00018200 R  0221
    IF(KVAL .LT. MINZ) MINZ = KVAL                                    00018300 R  0223
    SK = SK + KVAL                                                    00018400 R  0226
    SWK = SWK + BLOCK(NDEX)*KVAL                                      00018500 R  0227
    SUM(MTREE) = SUM(MTREE) + BLOCK(NDEX)                             00018600 R  0232
    IF(ITEST .EQ. 1) GO TO 159                                        00018700 R  0247
    GO TO 99                                                          00018800 R  0249
110 CONTINUE                                                          00018900 R  0250
    IF(IBEG .EQ. INDEX) GO TO 151                                     00019000 R  0250
130 DO 150 I = 1,6                                                    00019100 R  0253
    IF(I .GT. 3) GO TO 131                                            00019200 R  0258
    K= 7 - 2*I                                                        00019300 R  0261
    GO TO 132                                                         00019400 R  0262
131 K = 14 - 2*I                                                      00019500 R  0264
132 ITEMP = ITREE(INDEX)/10**(K-1) - (ITREE(INDEX)/10**K)*10          00019600 R  0266
    IF(ITEMP .EQ. 0) GO TO 150                                        00019700 R  0282
    GO TO  91                                                         00019800 R  0284
150 CONTINUE                                                          00019900 R  0285
151 IF(SUM(MTREE) .GT. 0) GO TO 159                                   00020000 R  0286
    ITREE(INDEX) = NTREE                                              00020100 R  0293
    ITEST = 1                                                         00020200 R  0298
    NDEX = INDEX                                                      00020300 R  0299
    GO TO 109                                                         00020400 R  0300
159 ISI = SI/SO + 0.5                                                 00020500 R  0301
    ISJ = SJ/SO + 0.5                                                 00020600 R  0305
    ISK = SK/SO + 0.5                                                 00020700 R  0309
    ISWI = SWI/SUM(MTREE) + 0.5                                       00020800 R  0313
    ISWJ = SWJ/SUM(MTREE) + 0.5                                       00020900 R  0322
    ISWK = SWK/SUM(MTREE) + 0.5                                       00021000 R  0331
```

```
      PRINT 1001, MTREE,SUM(MTREE),INDEX,MINX,MAXX,MINY,MAXY,MINZ,MAXZ, 00021100 R  0340
     1ISI,ISJ,ISK,ISWI,ISWJ,ISWK                                        00021200 R  0368
 1001 FORMAT(4X,I3,2X,F11.1,2X,I4,6X,12(2X,I3,2X))                      00021300 R  0384
                                                                       SEGMENT    3 IS  126 LONG
      GO TO 30                                                         00021400 R  0384
C                                                                      00021500 R  0384
C     INITIALIZE SORT                                                  00021600 R  0384
C                                                                      00021700 R  0384
  160 IF(MTREE .LE. 1) GO TO 280                                       00021800 R  0385
      DO 161 I = 1,MAXK                                                00021900 R  0387
  161 IF(BLOCK(I) .EQ. 0.) BLOCK(I) = -1                               00022000 R  0394
  169 INDEX = 0                                                        00022100 R  0404
      N = 0                                                            00022200 R  0404
      DO 200 K = 1, KMAX                                               00022300 R  0405
      IT1(3) = K                                                       00022400 R  0410
      IT1(6) = K                                                       00022500 R  0412
      DO 200 J = 1, JMAX                                               00022600 R  0414
      IT1(2) = J                                                       00022700 R  0420
      IT1(5) = J                                                       00022800 R  0422
      DO 200 I = 1, IMAX                                               00022900 R  0424
      IT1(1) = I                                                       00023000 R  0430
      IT1(4) = I                                                       00023100 R  0432
      INDEX = INDEX + 1                                                00023200 R  0434
      IF(BLOCK(INDEX) .GT. 0) GO TO 200                                00023300 R  0436
      ISAVE = 0                                                        00023400 R  0441
  180 DO 189 II = 1,6                                                  00023500 R  0443
      IF(IT1(II) .EQ. IJK(II)) GO TO 189                               00023600 R  0448
      NDEX = INDEX + JDA(II)                                           00023700 R  0460
      IF(BLOCK(NDEX) .LE. 0.) GO TO 189                                00023800 R  0466
      NVAL = ITREE(NDEX)/1000000                                       00023900 R  0472
      GO TO 190                                                        00024000 R  0480
  189 CONTINUE                                                         00024100 R  0482
      GO TO 200                                                        00024200 R  0482
  190 IF(ISAVE .EQ. 0) GO TO 192                                       00024300 R  0484
      DO 191 KK = 1, ISAVE                                             00024400 R  0486
      IF(KTAG(KK) .EQ. NVAL) GO TO 189                                 00024500 R  0491
  191 CONTINUE                                                         00024600 R  0499
  192 ISAVE = ISAVE + 1                                                00024700 R  0500
      KTAG(ISAVE) = NVAL                                               00024800 R  0501
      N = N + 1                                                        00024900 R  0507
      SORT(N) = BLOCK(INDEX)                                           00025000 R  0509
      ITAG(N) = INDEX + NVAL * 1000000                                 00025100 R  0518
      GO TO 189                                                        00025200 R  0527
  200 CONTINUE                                                         00025300 R  0528
      NSORT = N                                                        00025400 R  0529
  205 K = 0                                                            00025500 R  0531
      DO 220 I = 2,N                                                   00025600 R  0531
      IF(SORT(I-1) .GE. SORT(I)) GO TO 220                             00025700 R  0537
  210 TEMP = SORT(I)                                                   00025800 R  0551
      ITEMP = ITAG(I)                                                  00025900 R  0556
      SORT(I) = SORT(I-1)                                              00026000 R  0562
      ITAG(I) = ITAG(I-1)                                              00026100 R  0574
      SORT(I-1) = TEMP                                                 00026200 R  0586
      ITAG(I-1) = ITEMP                                                00026300 R  0592
      K = 1                                                            00026400 R  0599
  220 CONTINUE                                                         00026500 R  0601
      N = N - 1                                                        00026600 R  0601
```

```
      IF(K .EQ. 1) GO TO 205                                            00026700 R  0602
      KOUNT = 0                                                         00026800 R  0604
      NPOS(1) = 0                                                       00026900 R  0605
      J = 1                                                             00027000 R  0607
      DO 270 I = 1, MTREE                                               00027100 R  0608
      DO 260 K = 1, NSORT                                               00027200 R  0613
      IF(ITAG(K)/1000000 .NE. I) GO TO 260                              00027300 R  0619
      KOUNT = KOUNT + 1                                                 00027400 R  0629
      ARRAY(KOUNT) = SORT(K)                                            00027500 R  0630
      KTAG(KOUNT) = ITAG(K) - I * 1000000                               00027600 R  0641
      ITAG(K) = KTAG(KOUNT)                                             00027700 R  0655
  260 CONTINUE                                                          00027800 R  0666
      J = J + 1                                                         00027900 R  0666
      NPOS(J) = KOUNT                                                   00028000 R  0667
  270 CONTINUE                                                          00028100 R  0674
      DO 275 I = 1,NSORT                                                00028200 R  0674
      ITAG(I) = KTAG(I)                                                 00028300 R  0679
  275 SORT(I) = ARRAY(I)                                                00028400 R  0691
 1113 FORMAT(1X,2I12)                                                   00028500 R  0702
  280 RETURN                                                            00028600 R  0702
      END                                                               00028700 R  0705
                                                                       SEGMENT    2 IS  725 LONG
```

```
                                                                    START OF SEGMENT **********    4
      SUBROUTINE MAXTRE(MTREE,IMAX,JMAX,KMAX)                       00028800 R  0000
      COMMON BLOCK(6000),ITREE(6000),LBK(6000),SORT(1020),ITAG(1020),  00028900 R  0000
     1NPR(101),NPL(30),IDA(6),JDA(6),KDA(6),IJK(6),SUM(30),MAXJ,MAXK,  00029000 R  0000
     2ICON,NPOS(30),IT1(30),KTAG(1020),ADD(120),JTAG(120),ARRAY(1020),  00029100 R  0000
     3IZAP(60),NPATH(1020),BARRAY(1020),KKTAG(1020),NBEST(1020),     00029200 R  0000
     4LABEL(30)                                                     00029300 R  0000
      KDA(1) = 1000                                                 00029400 R  0000
      KDA(2) = 10000                                                00029500 R  0002
      KDA(3) = 100000                                               00029600 R  0005
      KDA(4) = 1                                                    00029700 R  0009
      KDA(5) = 10                                                   00029800 R  0011
      KDA(6) = 100                                                  00029900 R  0013
      ISTOP = 0                                                     00030000 R  0015
      ITRE = 0                                                      00030100 R  0016
      TVAL = 0                                                      00030200 R  0017
      PRINT 5040                                                    00030300 R  0017
 5040 FORMAT(1H1,1X,16HITERATION  TREE ,15X,5HVALUE/)               00030400 R  0022
      DO 10 I = 1, MTREE                                            00030500 R  0022
   10 LABEL(I) = 0                                                  00030600 R  0028
C     HAVE ALL POSITIVE DISJOINT TREES BEEN ATEMPTED                00030700 R  0033
   20 IF(ITRE .EQ. 0) GO TO 15                                      00030800 R  0035
      DO 14 I = 1, MAXK                                             00030900 R  0037
      IF(ITREE(I) .GT. 0) GO TO 14                                  00031000 R  0043
      ITREE(I) = 0                                                  00031100 R  0051
   14 CONTINUE                                                      00031200 R  0057
      DO 8 I=1,MAXK                                                 00031300 R  0057
    8 PRINT1188,ITREE(I),BLOCK(I)                                   00031400 R  0064
 1188 FORMAT(10X,I12,F10.1)                                         00031500 R  0085
   15 ITRE = ITRE + 1                                               00031600 R  0085
      NODE = 0                                                      00031700 R  0086
      IZ = 0                                                        00031800 R  0087
      NP = 0                                                        00031900 R  0087
      AVAL = 0                                                      00032000 R  0088
      IT = 1                                                        00032100 R  0089
      ITP = IT                                                      00032200 R  0090
      DO 19 I = 1, MTREE                                            00032300 R  0090
      IF(LABEL(I) .NE. 0) GO TO 19                                  00032400 R  0096
      IF(SUM(I) .LE. AVAL) GO TO 19                                 00032500 R  0104
      NODE = I                                                      00032600 R  0111
      IT1(IT) = NODE                                                00032700 R  0111
      LABEL(NODE) = NODE                                            00032800 R  0117
      AVAL = SUM(I)                                                 00032900 R  0123
   19 CONTINUE                                                      00033000 R  0130
      DELTA = AVAL                                                  00033100 R  0130
      IF(NODE .EQ. 0) GO TO 1000                                    00033200 R  0131
C     INITIALIZE ARRAY                                              00033300 R  0132
      ILAST = NPOS(NODE + 1) - NPOS(NODE)                           00033400 R  0133
      IND = NPOS(NODE)                                              00033500 R  0144
      NSORT = ILAST                                                 00033600 R  0150
      NI = NSORT                                                    00033700 R  0151
      DO 40 I = 1, ILAST                                            00033800 R  0152
      IND = IND + 1                                                 00033900 R  0157
      ARRAY(I) = SORT(IND) + AVAL                                   00034000 R  0159
      KTAG(I) = ITAG(IND)                                           00034100 R  0170
      BARRAY(I) = ARRAY(IND)                                        00034200 R  0180
      KKTAG(I) = ITAG(IND)                                          00034300 R  0191
```

```
      ITREE(ITAG(IND)) = -99                                          00034400 R  0202
   40 CONTINUE                                                        00034500 R  0214
      PRINT 5050, ITRE,NODE,SUM(NODE)                                 00034600 R  0214
C                                                                     00034700 R  0228
C     DYNAMIC - SHORTEST PATH ALGORITHM                               00034800 R  0228
C                                                                     00034900 R  0228
   99 KK = 0                                                          00035000 R  0234
      DO 100 I = 1, 6                                                 00035100 R  0234
      INDEX = KTAG(1) + JDA(I)                                        00035200 R  0240
 1881 FORMAT(5X,3I12,F10.1)                                           00035300 R  0248
      GO TO (21,22,23,24,25,26),I                                     00035400 R  0248
   21 IF(MOD(INDEX,IMAX) - 1)28,100,28                                00035500 R  0257
   22 IF(MOD(KTAG(1),MAXJ) .EQ. 0) GO TO 100                          00035600 R  0261
      IF(INDEX/MAXJ - KTAG(1)/MAXJ) 100,28,100                        00035700 R  0266
   23 IF(INDEX - MAXK) 28,28,100                                      00035800 R  0273
   24 IF(MOD(INDEX,IMAX)) 28,100,28                                   00035900 R  0277
   25 IF(INDEX .LE. 0) GO TO 100                                      00036000 R  0280
      IF(MOD(INDEX,MAXJ) .EQ. 0) GO TO 100                            00036100 R  0282
      IF(INDEX/MAXJ - KTAG(1)/MAXJ) 100,28,100                        00036200 R  0286
   26 IF(INDEX) 100,100,28                                            00036300 R  0293
   28 IF(ITREE(INDEX) .LT. 0) GO TO 100                               00036400 R  0296
      IF(BLOCK(INDEX) .LE. 0) GO TO 50                                00036500 R  0303
      IV = ITREE(INDEX)/1000000                                       00036600 R  0308
      DO 31 IX = 1,IT                                                 00036700 R  0316
   31 IF(IV .EQ. IT1(IX)) GO TO 100                                   00036800 R  0323
C     THIS IS A BREAK THROUGH                                         00036900 R  0330
      MODE = IV                                                       00037000 R  0331
      GO TO 199                                                       00037100 R  0332
   50 IF(ITREE(INDEX) .EQ. -99) GO TO 100                             00037200 R  0333
      KK = KK + 1                                                     00037300 R  0340
      ADD(KK) = BLOCK(INDEX) + ARRAY(1)                               00037400 R  0341
      JTAG(KK) = INDEX                                                00037500 R  0352
      ITREE(INDEX) = - KDA(I)                                         00037600 R  0358
  100 CONTINUE                                                        00037700 R  0370
C     UPDATE ARRAY                                                    00037800 R  0370
 1882 FORMAT(1X,6I10)                                                 00037900 R  0371
      NSAVE = NSORT                                                   00038000 R  0371
      NSORT = NSORT + KK - 1                                          00038100 R  0371
      IF(NSORT .LE. 0) GO TO 20                                       00038200 R  0373
      N = KK                                                          00038300 R  0375
      NLAST = NSORT                                                   00038400 R  0376
      IF(KK - 1) 140,$79,105                                          00038500 R  0377
  105 K = 0                                                           00038600 R  0383
      DO 120 I = 2, N                                                 00038700 R  0383
      IF(ADD(I-1) .GE. ADD(I)) GO TO 120                              00038800 R  0389
      TEMP = ADD(I)                                                   00038900 R  0402
      ITEMP = JTAG(I)                                                 00039000 R  0408
      ADD(I) = ADD(I-1)                                               00039100 R  0414
      JTAG(I) = JTAG(I-1)                                             00039200 R  0426
      ADD(I-1) = TEMP                                                 00039300 R  0438
      JTAG(I-1) = ITEMP                                               00039400 R  0444
      K = 1                                                           00039500 R  0451
  120 CONTINUE                                                        00039600 R  0453
      N = N - 1                                                       00039700 R  0453
      IF(K .EQ. 1) GO TO 105                                          00039800 R  0454
  150 NLAST = NSORT + 1                                               00039900 R  0457
      DO 160 I = 1, NSORT                                             00040000 R  0458
```

```
      NLAST = NLAST - 1                                          00040100 R  0463
      IF(KK .EQ. 1) GO TO 179                                    00040200 R  0465
      IF(NSAVE .EQ. 0) GO TO 152                                 00040300 R  0467
      IF(ARRAY(NSAVE) .LT. ADD(KK)) GO TO 154                    00040400 R  0469
  152 ARRAY(NLAST) = ADD(KK)                                     00040500 R  0482
      KTAG(NLAST) = JTAG(KK)                                     00040600 R  0492
      KK = KK - 1                                                00040700 R  0503
      GO TO 160                                                  00040800 R  0504
  154 ARRAY(NLAST) = ARRAY(NSAVE)                                00040900 R  0506
      KTAG(NLAST) = KTAG(NSAVE)                                  00041000 R  0516
      NSAVE = NSAVE - 1                                          00041100 R  0527
  160 CONTINUE                                                   00041200 R  0529
      GO TO 99                                                   00041300 R  0529
  140 DO 145 I = 1, NSORT                                        00041400 R  0530
      II = I + 1                                                 00041500 R  0535
      ARRAY(I) = ARRAY(II)                                       00041600 R  0537
      KTAG(I) = KTAG(II)                                         00041700 R  0547
  145 CONTINUE                                                   00041800 R  0559
      GO TO 99                                                   00041900 R  0559
  179 NLAST = NLAST - 1                                          00042000 R  0560
      DO 180 I = 1, NLAST                                        00042100 R  0561
      IF(ARRAY(I+1) .LT. ADD(1)) GO TO 181                       00042200 R  0566
      ARRAY(I) = ARRAY(I+1)                                      00042300 R  0576
  180 KTAG(I) = KTAG(I+1)                                        00042400 R  0588
      ARRAY(NLAST+1) = ADD(1)                                    00042500 R  0600
      KTAG(NLAST + 1) = JTAG(1)                                  00042600 R  0607
      GO TO 99                                                   00042700 R  0615
  181 ARRAY(I) = ADD(1)                                          00042800 R  0617
      KTAG(I) = JTAG(1)                                          00042900 R  0623
 1401 FORMAT(1X, 5(F6.0,I5,I10))                                 00043000 R  0631
      GO TO 99                                                   00043100 R  0631
C     IS THIS BREAKTHROUGH AN IMPROVEMENT                        00043200 R  0631
  199 IZ = IZ + 1                                                00043300 R  0632
      IZAP(IZ) = MODE                                            00043400 R  0633
      IZ = IZ + 1                                                00043500 R  0639
      IZAP(IZ) = KTAG(1)                                         00043600 R  0641
      LABEL(IV) = IV                                             00043700 R  0647
      IT = IT + 1                                                00043800 R  0653
      IF(IT .EQ. MTREE) ISTOP = 1                                00043900 R  0655
      IT1(IT) = IV                                               00044000 R  0657
      IF(ARRAY(1) + SUM(IV) .GT. AVAL) GO TO 300                 00044100 R  0663
C     UPDATE ARRAY AND CONTINUE                                  00044200 R  0671
      IADD = NPOS(MODE + 1) - NPOS(MODE) - 2                     00044300 R  0672
      DO 220 I = 1, NSORT                                        00044400 R  0684
      II = NSORT - I + 1                                         00044500 R  0689
      IP = II + IADD                                             00044600 R  0691
      ARRAY(IP) = ARRAY(II)                                      00044700 R  0692
  220 KTAG(IP) = KTAG(II)                                        00044800 R  0704
      SAVE = ARRAY(1)                                            00044900 R  0715
      NSAVE = NSORT                                              00045000 R  0716
      NSORT = NSORT + IADD                                       00045100 R  0717
      KK = NPOS(MODE) + 1                                        00045200 R  0718
      MSAVE = TTAG(KK)                                           00045300 R  0725
      NTEST = NPOS(MODE + 1)                                     00045400 R  0730
      IADD=IADD + 1                                              00045500 R  0736
      ICT = 1                                                    00045600 R  0737
      TEMP = SORT(KK) + SUM(MODE) + SAVE                         00045700 R  0738
```

```
      L = 0                                                          00045800 R  0750
      DO 240 I = 1, NSORT                                            00045900 R  0751
      IF(KK .GT. NTEST) GO TO 239                                    00046000 R  0756
      IF(ICT .GT. NSAVE) GO TO 232                                   00046100 R  0759
      IF(ARRAY(I + IADD) .GE. TEMP) GO TO 239                        00046200 R  0761
  232 IF(ITAG(KK) .EQ. MSAVE) GO TO 236                              00046300 R  0769
      IL = L + 1                                                     00046400 R  0776
      IF(L .EQ. 0) GO TO 234                                         00046500 R  0777
      DO 233 IW = 1, L                                               00046600 R  0779
  233 IF(KTAG(IW) .EQ. ITAG(KK)) GO TO 236                           00046700 R  0785
  234 DO 235 IW = IL, NSAVE                                          00046800 R  0798
      IY = IW + IADD                                                 00046900 R  0803
  235 IF(KTAG(IY) .EQ. ITAG(KK)) GO TO 236                           00047000 R  0805
      L = L + 1                                                      00047100 R  0817
      ARRAY(L) = TEMP                                                00047200 R  0818
      KTAG(L) = ITAG(KK)                                             00047300 R  0824
      ITREE(KTAG(L)) = - 99 - MODE * 1000000                         00047400 R  0835
      IADD = IADD - 1                                                00047500 R  0850
  236 KK = KK + 1                                                    00047600 R  0852
      TEMP = SORT(KK) + SUM(MODE) + SAVE                             00047700 R  0853
      GO TO 240                                                      00047800 R  0865
  239 L = L + 1                                                      00047900 R  0866
      ARRAY(L) = ARRAY(L + IADD)                                     00048000 R  0867
      KTAG(L) = KTAG(L + IADD)                                       00048100 R  0879
      ICT = ICT + 1                                                  00048200 R  0891
  240 CONTINUE                                                       00048300 R  0893
      NSORT = NSORT - IADD                                           00048400 R  0893
      GO TO 99                                                       00048500 R  0894
C     FIND WAY BACK TO BEST TREE                                     00048600 R  0894
C     ADD ADDITIONAL NODE TO ARRAY                                   00048700 R  0894
  300 NJ = 0                                                         00048800 R  0896
      INDEX=KTAG(1)                                                  00048900 R  0896
      AVAL = ARRAY(1) + SUM(MODE)                                    00049000 R  0898
      IT = ITP + 1                                                   00049100 R  0905
      IT1(IT) = MODE                                                 00049200 R  0907
  400 LAST = NPOS(MODE + 1)                                          00049300 R  0913
      NST = NPOS(MODE) + 1                                           00049400 R  0919
      DO 420 I = NST, LAST                                           00049500 R  0926
      DO 410 J = 1, NI                                               00049600 R  0931
      IF(ITAG(I) .EQ. KKTAG(J)) GO TO 420                            00049700 R  0937
  410 CONTINUE                                                       00049800 R  0950
      NI = NI + 1                                                    00049900 R  0950
      KKTAG(NI) = ITAG(I)                                            00050000 R  0951
      NJ = NJ + 1                                                    00050100 R  0962
      ADD(NJ) = AVAL + SORT(I)                                       00050200 R  0963
      JTAG(NJ) = ITAG(I)                                             00050300 R  0975
  420 CONTINUE                                                       00050400 R  0987
      NSTART = NP + 1                                                00050500 R  0987
  301 IF(ITREE(INDEX) .LE. -1000000) GO TO 350                       00050600 R  0989
      NP = NP + 1                                                    00050700 R  0998
      NPATH(NP) = INDEX                                              00050800 R  0999
      IF(ITREE(INDEX) .EQ. -99) GO TO 316                            00050900 R  1005
      DO 309 II = 1, 6                                               00051000 R  1013
      IF(-ITREE(INDEX) .NE. KDA(II)) GO TO 309                       00051100 R  1018
                                                                   SEGMENT    4 IS 1023 LONG
                                                                 START OF SEGMENT **********   5
      NDX = INDEX                                                    00051200 R  0009
```

```
      IF(II .LE. 3) IY = II + 3                                          00051300 R  0010
      IF(II .GT. 3) IY = II - 3                                          00051400 R  0013
      INDEX = INDEX + JDA(IY)                                            00051500 R  0016
      ITREE(NDX) = -98                                                   00051600 R  0022
      GO TO 301                                                          00051700 R  0029
  309 CONTINUE                                                           00051800 R  0030
      PRINT 3490                                                         00051900 R  0031
 3490 FORMAT(1X,6HSTOP 1)                                                00052000 R  0035
      CALL EXIT                                                          00052100 R  0035
  316 DO 349 IDO = NSTART,NP                                             00052200 R  0037
  307 DO 340 I = 1, 6                                                    00052300 R  0043
      NDEX = NPATH(IDO) + JDA(I)                                         00052400 R  0048
      GO TO (321,322,323,324,325,326) , I                                00052500 R  0059
  321 IF(MOD(NDEX,IMAX) - 1) 328,340,328                                 00052600 R  0069
  322 IF(MOD(INDEX,MAXJ) .EQ. 0) GO TO 340                               00052700 R  0073
      IF(NDEX/MAXJ - INDEX/MAXJ) 340,328,340                             00052800 R  0077
  323 IF(NDEX - MAXK) 328,328,340                                        00052900 R  0083
  324 IF(MOD(NDEX,IMAX)) 328,340,328                                     00053000 R  0087
  325 IF(NDEX .LE. 0) GO TO 340                                          00053100 R  0090
      IF(MOD(NDEX,MAXJ) .EQ. 0) GO TO 340                                00053200 R  0092
      IF(NDEX/MAXJ - INDEX/MAXJ) 340,328,340                             00053300 R  0096
  326 IF(NDEX) 340,340,328                                               00053400 R  0102
  328 IF(BLOCK(NDEX) .GT. 0) GO TO 340                                   00053500 R  0105
      IF(ITREE(NDEX) .EQ. -98) GO TO 340                                 00053600 R  0110
      DO 330 J = 1, NI                                                   00053700 R  0118
      IF(NDEX .EQ. KKTAG(J)) GO TO 340                                   00053800 R  0123
  330 CONTINUE                                                           00053900 R  0131
      NI = NI + 1                                                        00054000 R  0131
      KKTAG(NI) = NDEX                                                   00054100 R  0132
      NJ = NJ + 1                                                        00054200 R  0138
      ADD(NJ) = AVAL + BLOCK(NDEX)                                       00054300 R  0140
      JTAG(NJ) = NDEX                                                    00054400 R  0149
  340 CONTINUE                                                           00054500 R  0156
  349 CONTINUE                                                           00054600 R  0157
      ITP = IT                                                           00054700 R  0157
      GO TO 359                                                          00054800 R  0158
C     BLOCK VALUE IS POSITIVE                                            00054900 R  0158
  350 NP = NP + 1                                                        00055000 R  0159
      NPATH(NP) = INDEX                                                  00055100 R  0160
      IV = - ITREE(INDEX)/1000000                                        00055200 R  0166
      IT = IT + 1                                                        00055300 R  0174
      IT1(IT) = IV                                                       00055400 R  0176
      IST = NPOS(IV) + 1                                                 00055500 R  0181
      ILT = NPOS(IV + 1)                                                 00055600 R  0188
      DO 354 I = IST, ILT                                                00055700 R  0194
      DO 353 J = 1, NI                                                   00055800 R  0199
      IF(ITAG(I) .EQ. KKTAG(J)) GO TO 354                                00055900 R  0205
  353 CONTINUE                                                           00056000 R  0218
      NI = NI + 1                                                        00056100 R  0218
      KKTAG(NI) = ITAG(I)                                                00056200 R  0219
      NJ = NJ + 1                                                        00056300 R  0230
      ADD(NJ) = AVAL + SORT(I)                                           00056400 R  0231
      JTAG(NJ) = ITAG(I)                                                 00056500 R  0243
  354 CONTINUE                                                           00056600 R  0255
      IND = -1                                                           00056700 R  0255
      DO 355 I = 1, IZ                                                   00056800 R  0256
      IND = IND + 2                                                      00056900 R  0261
```

```
      IF(IV .EQ. IZAP(IND)) GO TO 358                                    00057000 R  0263
  355 CONTINUE                                                           00057100 R  0271
  358 INDEX = IZAP(IND + 1)                                              00057200 R  0272
      GO TO 301                                                          00057300 R  0278
C     MERGE ADD AND BARRY TO ARRAY                                       00057400 R  0278
  359 DO 360 I = 1, NI                                                   00057500 R  0279
  360 BARRAY(I) = AVAL + BLOCK(KKTAG(I))                                 00057600 R  0285
      IF(NJ .EQ. 0) GO TO 371                                            00057700 R  0300
      N = NJ                                                             00057800 R  0302
  361 K = 0                                                              00057900 R  0304
      DO 369 I = 2, N                                                    00058000 R  0304
      IF(ADD(I-1) .GE. ADD(I)) GO TO 369                                 00058100 R  0310
      TEMP = ADD(I)                                                      00058200 R  0323
      ITEMP = JTAG(I)                                                    00058300 R  0329
      ADD(I) = ADD(I-1)                                                  00058400 R  0335
      JTAG(I) = JTAG(I-1)                                                00058500 R  0347
      ADD(I-1) = TEMP                                                    00058600 R  0359
      JTAG(I-1) = ITEMP                                                  00058700 R  0365
      K = 1                                                              00058800 R  0372
  369 CONTINUE                                                           00058900 R  0374
      N = N - 1                                                          00059000 R  0374
      IF(K .EQ. 1) GO TO 361                                             00059100 R  0375
  371 II = NI - NJ                                                       00059200 R  0378
      NLAST = NI + 1                                                     00059300 R  0379
      DO 380 I = 1, NI                                                   00059400 R  0380
      NLAST = NLAST - 1                                                  00059500 R  0385
      IF(II .EQ. 0) GO TO 372                                            00059600 R  0387
      IF(NJ .EQ. 0) GO TO 374                                            00059700 R  0389
      IF(BARRAY(II) .LT. ADD(NJ)) GO TO 374                              00059800 R  0391
  372 ARRAY(NLAST) = ADD(NJ)                                             00059900 R  0404
      KTAG(NLAST) = JTAG(NJ)                                             00060000 R  0414
      NJ = NJ - 1                                                        00060100 R  0425
      GO TO 380                                                          00060200 R  0426
  374 ARRAY(NLAST) = BARRAY(II)                                          00060300 R  0428
      KTAG(NLAST) = KKTAG(II)                                            00060400 R  0438
      II = II - 1                                                        00060500 R  0449
  380 CONTINUE                                                           00060600 R  0451
      NSORT = NI                                                         00060700 R  0451
      DO 390 I = 1,NI                                                    00060800 R  0452
      BARRAY(I) = ARRAY(I)                                               00060900 R  0457
      ITREE(KTAG(I)) = -99                                               00061000 R  0468
  390 KKTAG(I) = KTAG(I)                                                 00061100 R  0480
C     CLEAR ARRAY                                                        00061200 R  0485
      DO 449 I = 1,MAXK                                                  00061300 R  0491
      IF(BLOCK(I) .GT. 0) GO TO 449                                      00061400 R  0497
      IF(ITREE(I) .EQ. -98 .OR. ITREE(I) .EQ. -99) GO TO 449             00061500 R  0503
      ITREE(I) = 0                                                       00061600 R  0517
  449 CONTINUE                                                           00061700 R  0524
C     PRINT ARRAY                                                        00061800 R  0524
  500 IF(AVAL .LT. TVAL) GO TO 510                                       00061900 R  0525
      TVAL = AVAL                                                        00062000 R  0527
      IF(NP .EQ. 0) GO TO 506                                            00062100 R  0528
      DO 505 I = 1, NP                                                   00062200 R  0530
  505 NBEST(I) = NPATH(I)                                                00062300 R  0536
  506 NB = NP                                                            00062400 R  0547
  510 PRINT 5050, ITRE,MODE,AVAL                                         00062500 R  0548
 5050 FORMAT(1X,2(I5,5X),10X,F10.1)                                      00062600 R  0562
```

```
                                                                                SEGMENT     6 IS     7 LONG
      IF(ISTOP .EQ. 1) GO TO 1000                                                00062700 R  0562
      IZ = 0                                                                     00062800 R  0564
      GO TO 99                                                                   00062900 R  0565
 1000 IF(NB .EQ. 0) GO TO 2000                                                   00063000 R  0566
      DO 1100 I = 1, MAXK                                                        00063100 R  0568
 1100 BLOCK(I) = 0                                                               00063200 R  0575
      DO 1200 I = 1, NB                                                          00063300 R  0579
 1200 BLOCK(NBEST(I)) = 1                                                        00063400 R  0586
      PRINT 1111                                                                 00063410 R  0595
 1111 FORMAT(1H1,1X,40HPLOT OF POSITIVE NODES IN ULTIMATE PLAN //)               00063420 R  0599
      CALL PLOT                                                                  00063500 R  0599
 2000 RETURN                                                                     00063600 R  0600
      END                                                                        00063700 R  0603
                                                                                SEGMENT     5 IS   622 LONG
```

```
                                                                    START OF SEGMENT **********    7
      SUBROUTINE PLOT                                                                 00063800 R  0000
      COMMON BLOCK(6000),ITREE(6000),LBK(6000),SORT(1020),ITAG(1020),                 00063900 R  0000
     1NPR(101),NPL(30),IDA(6),JDA(6),KDA(6),IJK(6),SUM(30),MAXJ,MAXK,                 00064000 R  0000
     2ICON,NPOS(30),IT1(30),KTAG(1020),ADD(120),JTAG(120),ARRAY(1020),                00064100 R  0000
     3IZAP(60),NPATH(1020),BARRAY(1020),KKTAG(1020),NBEST(1020),                      00064200 R  0000
     4LABEL(30)                                                                       00064300 R  0000
C                                                                                     00064400 R  0000
      JCON = 24                                                                       00064500 R  0000
      ZERO = 0.                                                                       00064600 R  0000
      I4 = 4                                                                          00064700 R  0001
      I3 = -3                                                                         00064800 R  0002
      PRINT 1011                                                                      00065100 R  0003
 1011 FORMAT(10X,18HTABLE  OF  SYMBOLS/ 5X,6HSYMBOL, 7X, 11HEXPLANATION/00065200 R  0007
     1 7X, 1H+, 7X, 13HPOSITIVE NODE/ 7X, 1H*, 7X, 27HCONNECTIVITY IN I 00065300 R  0007
     2= J PLANE/ 7X, 1HU, 7X, 27HCONNECTIVITY WITH K-1 PLANE/ 7X, 1HD, 700065400 R  0007
     3X, 27HCONNECTIVITY WITH K+1 PLANE/ 7X, 1HB, 7X, 33HCONNECTIVITY WI00065500 R  0007
     4TH ADJACENT PLANES//)                                                           00065600 R  0007
C                                                                                     00065700 R  0007
      IVAL = - MAXJ                                                                   00065800 R  0007
      ITER = (IJK(2) + JCON - 1)/JCON                                                 00065900 R  0009
      DO 100 K =1,IJK(3)                                                              00066000 R  0012
      IEND = 0                                                                        00066100 R  0018
      IVAL = IVAL + MAXJ                                                              00066200 R  0019
      JVAL = 0                                                                        00066300 R  0021
      DO 100 JPLOT = 1, ITER                                                          00066400 R  0022
      PRINT 1001, K                                                                   00066500 R  0027
 1001 FORMAT(/1X, 9HSECTION  ,I5/)                                                    00066600 R  0038
      ISTART = IEND + 1                                                               00066700 R  0038
      IEND = ISTART + JCON                                                            00066800 R  0039
      IF(JPLOT .EQ. ITER) IEND = IJK(2)                                               00066900 R  0040
      PRINT 1002, (N,N=ISTART,IEND)                                                   00067000 R  0043
 1002 FORMAT(10X,14H*** COLUMN ***/ 4X,3HROW,25I4)                                    00067100 R  0060
      PRINT 1012                                                                      00067200 R  0060
 1012 FORMAT(/)                                                                       00067300 R  0064
      IE = (IEND - ISTART) + I4 + 1                                                   00067400 R  0064
      INT = IEND - ISTART + 1                                                         00067500 R  0066
      IZ = JVAL + IVAL - IJK(1)                                                       00067600 R  0068
      DO 49 I = 1,IJK(1)                                                              00067700 R  0070
      INDEX = IZ + I                                                                  00067800 R  0077
      NO = I3                                                                         00067900 R  0079
      DO 2 II = 1,IE                                                                  00068000 R  0079
    2 NPR(II) = 1H                                                                    00068100 R  0086
      DO 3 II = 1,INT                                                                 00068200 R  0094
    3 NPL(II) = 1H                                                                    00068300 R  0100
      DO 48 J = ISTART,IEND                                                           00068400 R  0108
      INDEX = INDEX + IJK(1)                                                          00068500 R  0113
      NO = NO + I4                                                                    00068600 R  0116
      IF(BLOCK(INDEX) .LE. ZERO) GO TO 48                                             00068700 R  0117
      NPR(NO) = 1H+                                                                   00068800 R  0122
    9 IF(J .EQ. IJK(2)) GO TO 15                                                      00068900 R  0131
      NDEX = INDEX + IJK(1)                                                           00069000 R  0134
      IF(BLOCK(NDEX) .LE. ZERO) GO TO 15                                              00069100 R  0136
      DO 13 KK = 1,3                                                                  00069200 R  0142
   13 NPR(KK+NO) = 1H*                                                                00069300 R  0148
   15 IK = 0                                                                          00069400 R  0157
      IF(K .EQ. 1) GO TO 20                                                           00069500 R  0157
```

```
      NDEX = INDEX + MAXJ
      IF(BLOCK(NDEX) .LE. ZERO) GO TO 20
      NPR(NO) = 1HU
      IK = 1
   20 IF(K .EQ. IJK(3)) GO TO 30
      NDEX = INDEX + MAXJ
      IF(BLOCK(NDEX) .LE. ZERO) GO TO 30
      NPR(NO) = 1HD
      IF(IK .EQ. 1) NPR(NO) = 1HB
   30 IF(I .EQ. IJK(1)) GO TO 48
      NDEX = INDEX + 1
      IF(BLOCK(NDEX) .LE. ZERO) GO TO 48
      NN = J - ISTART + 1
      NPL(NN) = 1H*
   48 CONTINUE
      JVAL = INDEX + IVAL
      PRINT 1003, I,(NPR(L),L=1,IE)
 1003 FORMAT(2X,I5,3X,101A1)

      IF(I .EQ. IJK(1)) GO TO 49
      PRINT 1004, (NPL(L),L=1,INT)
 1004 FORMAT(10X,24(A1,3X))
   49 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
 00069600 R  0160
 00069700 R  0162
 00069800 R  0167
 00069900 R  0175
 00070000 R  0177
 00070100 R  0180
 00070200 R  0182
 00070300 R  0188
 00070400 R  0195
 00070500 R  0205
 00070600 R  0208
 00070700 R  0209
 00070800 R  0215
 00070900 R  0216
 00071000 R  0225
 00071100 R  0225
 00071200 R  0226
 00071300 R  0250
SEGMENT     8 IS  125 LONG
 00071400 R  0250
 00071500 R  0253
 00071600 R  0275
 00071700 R  0275
 00071800 R  0276
 00071900 R  0277
 00072000 R  0280
SEGMENT     7 IS  290 LONG
```

U.S. GOVERNMENT PRINTING OFFICE: 1974—712-829:569