Helmut Lerchs*

Manager of Scientific Services,
Montreal Datacentre,
International Business Machines
Co. Ltd.,

Ingo F. Grossmann

Manager,
Management Science Applications,
International Business Machines
Co. Ltd.

Optimum Design of Open-Pit Mines

Joint C.O.R.S. and O.R.S.A. Conference,
Montreal, May 27-29, 1964

Transactions, C.I.M., Volume LXVIII, 1965, pp. 17-24

ABSTRACT

An open-pit mining operation can be viewed as a process by which the open surface of a mine is continuously deformed. The planning of a mining program involves the design of the final shape of this open surface. The approach developed in this paper is based on the following assumptions: 1. the type of material, its mine value and its extraction cost is given for each point; 2. restrictions on the geometry of the pit are specified (surface boundaries and maximum allowable wall slopes); 3. the objective is to maximize total profit — total mine value of material extracted minus total extraction cost.

Two numeric methods are proposed: A simple dynamic programming algorithm for the two-dimensional pit (or a single vertical section of a mine), and a more elaborate graph algorithm for the general three-dimensional pit.

Introduction

A SURFACE mining program is a complex operation that may extend over many years, and involve huge capital expenditures and risk. Before undertaking such an operation, it must be known what ore there is to be mined (types, grades, quantities and spatial distribution) and how much of the ore should be mined to make the operation profitable.

The reserves of ore and its spatial distribution are estimated by geological interpretation of the information obtained from drill cores. The object of pit design then is to determine the amount of ore to be mined.

Assuming that the concentration of ores and impurities is known at each point, the problem is to decide what the ultimate contour of the pit will be and in what stages this contour is to be reached. Let us note that if, with respect to the global objectives of a mining program, an optimum pit contour exists, and if the mining operation is to be optimized, then this contour must be known, if only to minimize the total cost of mining.

*Now Senior Research Mathematician, General Motors Research Laboratory, Warren, Mich.

Open-Pit Model

Besides pit design, planning may bear on questions such as:

— what market to select;
— what upgrading plants to install;
— what quantities to extract, as a function of time;
— what mining methods to use;
— what transportation facilities to provide.

There is an intimate relationship between all the above points, and it is meaningless to consider any one component of planning separately. A mathematical model taking into account all possible alternatives simultaneously would, however, be of formidable size and its solution would be beyond the means of present knowhow. The model proposed in this paper will serve to explore alternatives in pit design, given a real or a hypothetical economical environment (market situation, plant configuration, etc.). This environment is described by the mine value of all ores present and the extraction cost of ores and waste materials. The objective then is to design the contour of a pit so as to maximize the difference between the total mine value of ore extracted and the total extraction cost of ore and waste. The sole restrictions concern the geometry of the pit; the wall slopes of the pit must not exceed certain given angles that may vary with the depth of the pit or with the material.

Analytically, we can express the problem as follows: Let v, c and m be three density functions defined at each point of a three-dimensional space.

$v(x, y, z)$ = mine value of ore per unit volume
$c(x, y, z)$ = extraction cost per unit volume
$m(x, y, z) = v(x, y, z) - c(x, y, z)$ = profit per unit volume.

Let $\alpha\ (x, y, z)$ define an angle at each point and let S be the family of surfaces such that at no point does their slope, with respect to a fixed horizontal plane, exceed α.

Let V be the family of volumes corresponding to the family, S, of surfaces. The problem is to find, among all volumes, V, one that maximizes the integral

$$\int_V m(x, y, z)\, dx\, dy\, dz$$

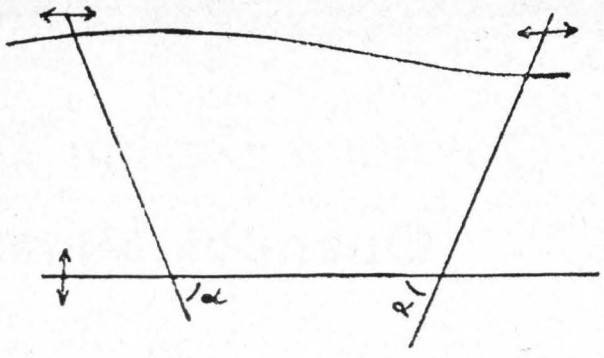

Figure 1.

Generally, there is no simple analytical representation for the functions v and c; consequently, numeric methods must be used. The traditional approach is to divide the whole pit into parallel vertical sections, and to consider each section as a two-dimensional pit. The technique used to determine the contour of a section consists in moving three straight lines, representing the bottom of the pit and two walls, at slopes α (see Figure 1), and in evaluating the ore and the extraction cost of materials limited by the three lines. The configuration of lines yielding the best results is then selected. (Here, α is taken to be constant over the entire pit).

The following dynamic programming technique is simpler, faster and more accurate.

Two-Dimensional Pit

Let us select the units u_i and u_j of a rectangular grid system such that

$$\frac{u_i}{u_j} = \tan \alpha$$

For each unit rectangle (i, j) determine the quantity $m_{ij} = v_{ij} - c_{ij}$. Construct a new tableau (Figure 2) with the quantities

$$M_{ij} = \sum_{k=i}^{i} m_{kj} .$$

M_{ij} represents the profit realized in extracting a single column with element (i, j) at its base.

In a final tableau, add a row zero and compute the following quantities:

$P_{oj} = O$ then, column by column starting with column 1: $P_{ij} = M_{ij} + \max_k (P_{i+k,\, j-1})$ with $k = -1, 0, 1$

Indicate the maximum by an arrow going from (i, j) to (i + k, j-1).

The interpretation of the P_{ij} is as follows:

P_{ij} is the maximum possible contribution of columns 1 to j to any feasible pit that contains the element (i, j) on its contour. It follows that if the element (i, j) is part of the optimum contour, then this contour, to the left of element (i, j), can be traced by following the arrows starting from element (i, j). Now, any feasible pit contour must contain at least one element of the first row. If the maximum value of P in the first row is positive, then the optimum contour is obtained by following the arrows from and to the left of this element. If all elements of the first row are negative, then there exists no contour with positive profit.

$\alpha = 45°$
$m_{ij} = v_{ij} - c_{ij}$

$$M_{ij} = \sum_{k=1}^{i} m_{kj}$$

$P_{oj} = O$

$$P_{ij} = M_{ij} + \max \begin{cases} P_{i-1,\, j-1} \\ P_{i,\, j-1} \\ P_{i+,\, j-1} \end{cases}$$

$$P_{max} = \max_k P_{1k}$$

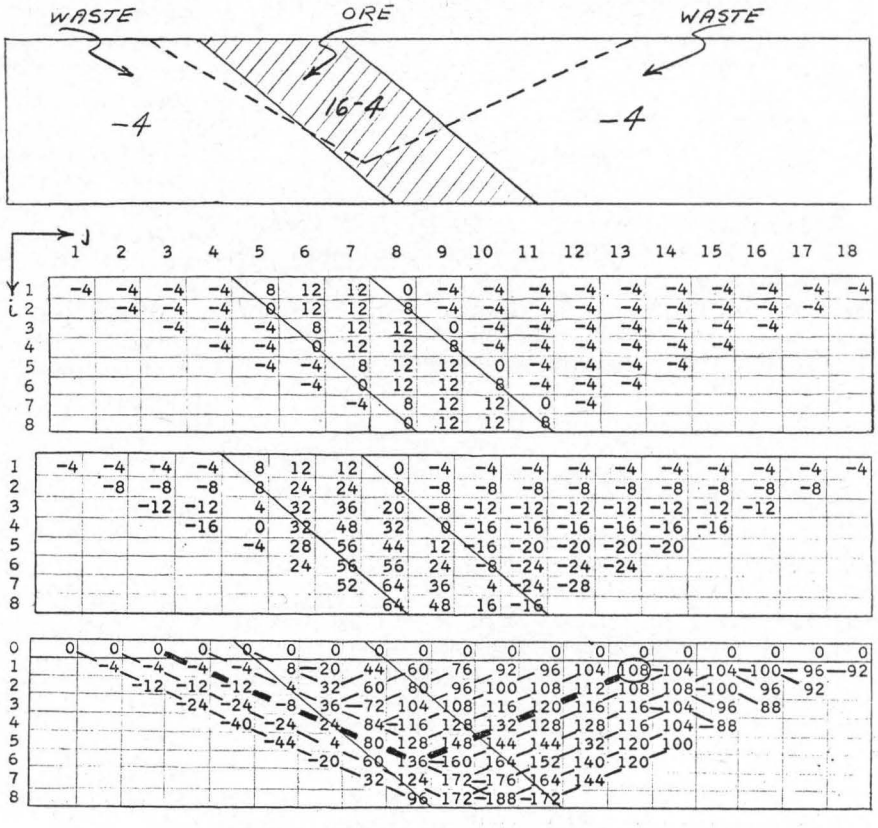

Figure 2.—Dynamic Programming.

Three-Dimensional Pit

When the optimum contours of all the vertical sections are assembled, it invariably turns out that they do not fit together because the wall slopes in a vertical section and at right angles with the sections that were optimized exceed the permissible angle α. The walls and the bottom of the pit are then "smoothed out." This takes a great amount of effort and the resulting pit contour may be far from optimum. Let us note that because the dynamic programming approach yields not only the optimum contour but also all alternate optima, if such exist, as well as next best solutions, it can be of help in "smoothing" the pit.

The dynamic programming approach becomes impractical in three dimensions. Instead, a graph algorithm can be applied. The model is derived as follows: Let the entire pit be divided into a set of volume elements V_i. This division can be quite arbitrary, but may also be obtained by taking for V_i the unit volumes defined by a three-dimensional grid. Associate to each volume element V_i a mass

$$m_i = v_i - c_i$$

where v_i and c_i are the mine value and the extraction cost of element V_i. Let each element V_i be represented by a vertex x_i of a graph. Draw an arc (x_i, x_j) if V_j is adjacent to V_i, that is, V_i and V_j have at least one point in common, and if the mining of volume V_i is not permissible unless volume V_j is also mined. We thus obtain a directed three-dimensional graph $G = (X, A)$ with a set of vertices X and a set of arcs A. Any feasible contour of the pit is represented by a *closure* of G, that is, a set of vertices Y such that if a vertex x_i belongs to Y and if the arc (x_i, x_j) exists in A then the vertex x_j must also belong to Y. If a mass m_i is associated to each vertex x_i, and if M_y is the total mass of a set of vertices Y, then the problem of optimum pit design comes to finding in a graph G a closure Y with maximum mass or, shortly, a maximum closure of G. (See Figure 3).

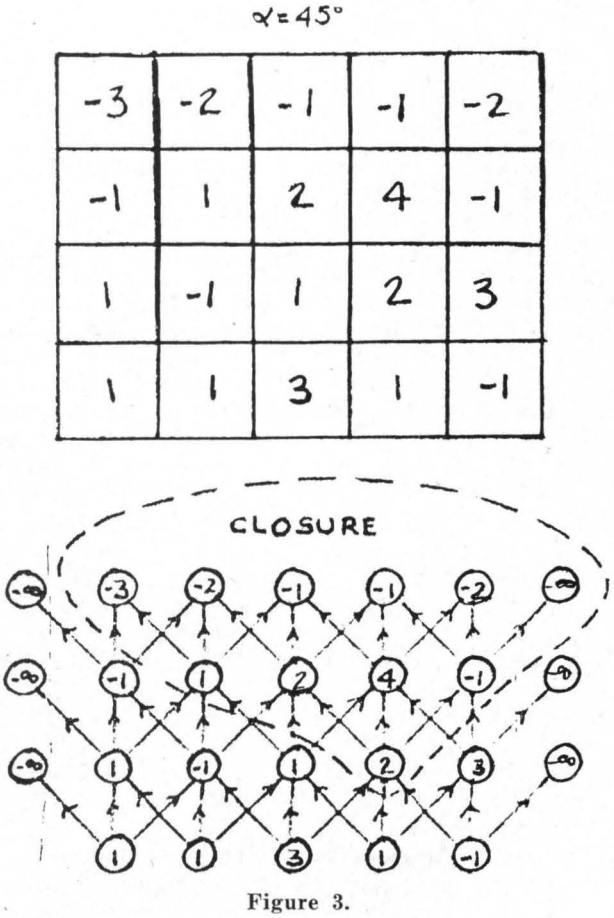

Figure 3.

This problem can be viewed as an extreme case of the time-cost optimization problem in project networks, to which several solutions have been proposed (2, 3, 4, 5). It can also be transformed into a network flow problem. However, there are obvious computational advantages to be gained from a direct approach; these advantages become important when the graphs considered contain a very large number of elements, as may be the case for an open-pit model.

An effective algorithm to find the maximum closure of a graph is developed in the Appendix. The procedure starts with the construction of a tree T^0 in G. T^0 is then transformed into successive trees $T^1, T^2, \ldots T^n$ following simple rules until no further transformation is possible. The maximum closure of G is then given by the vertices of a set of well-identified branches of the final tree.

The decomposition of the pit into elementary volumes V_i will depend on the structure of the pit itself and on the function α (x, y, z). When α is constant, as is the case in most instances, one of the grid systems shown in Figure 4 can be taken, with proper selection of units on the axis.

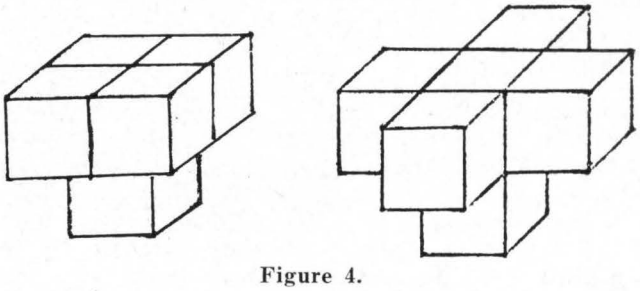

Figure 4.

The three-dimensional pit model can be illustrated by a physical analogue. In Figure 5, each block has a grid point at its center through which there is an upward force (the value of the ore in the block) and a downward force (the cost of removing the ore). The resulting force in a block is indicated with an arrow. If the system is left to move freely one unit along a vertical axis, some of the blocks will be lifted. The total work done in this movement is $F \times 1 = F$, where F is the resulting force of all blocks that participate in the movement. However, the movement of any free mechanical system is such as to maximize the work done. Hence, F is the maximum resulting force over any set of blocks that can freely move upward in this system, and, returning to our model, the blocks will separate along the optimum pit contour.

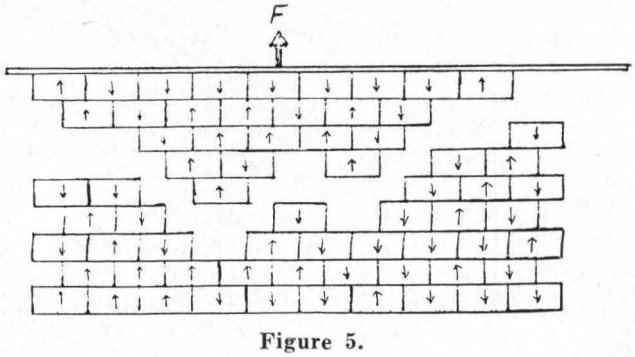

Figure 5.

Parametric Analysis

The established algorithm provides solutions to the final contour of a pit. There are, however, virtually unlimited numbers of ways of reaching a final contour, each way having a different cash flow pattern. Figure 6 illustrates some of the possible cash flows.

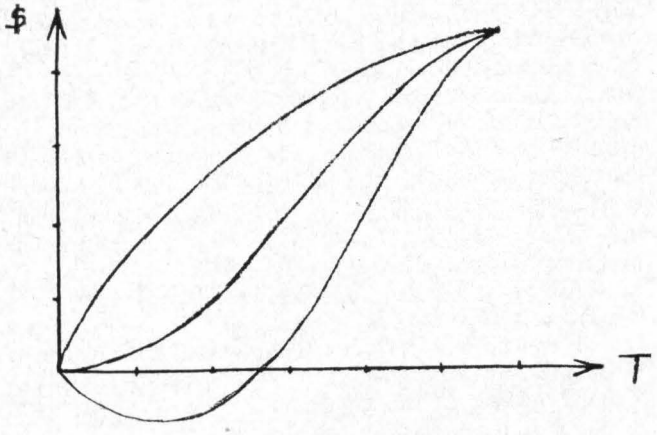

Figure 6.—Cash Flow Patterns

An optimum digging pattern might be one in which the integral of the cash flow curve is maximum. The problem of designing intermediate pit contours can become extremely complex. The following analysis will highlight some properties of the pit model, and the results may provide a basis for the selection of intermediate contours. Let us add a restriction to our pit model. Supposing that we want to maximize the profit in the first year of operations and that our mining capacity is limited to a total volume V. What is the optimum contour now? To answer this question we shall consider the function

$$P = M - \lambda V$$

where M is the mass of a closure, V the volume of the closure and λ a positive scalar. Instead of maximizing M as we did in our basic model, we now want to maximize P. This problem can be transformed into the basic problem by substituting each elementary mass by a new mass

$$m'_i = m_i - \lambda$$

For $\lambda = 0$ we obtain our old solution; when λ increases, P decreases, but for sufficiently small increments of λ the optimum contour and V will stay constant.

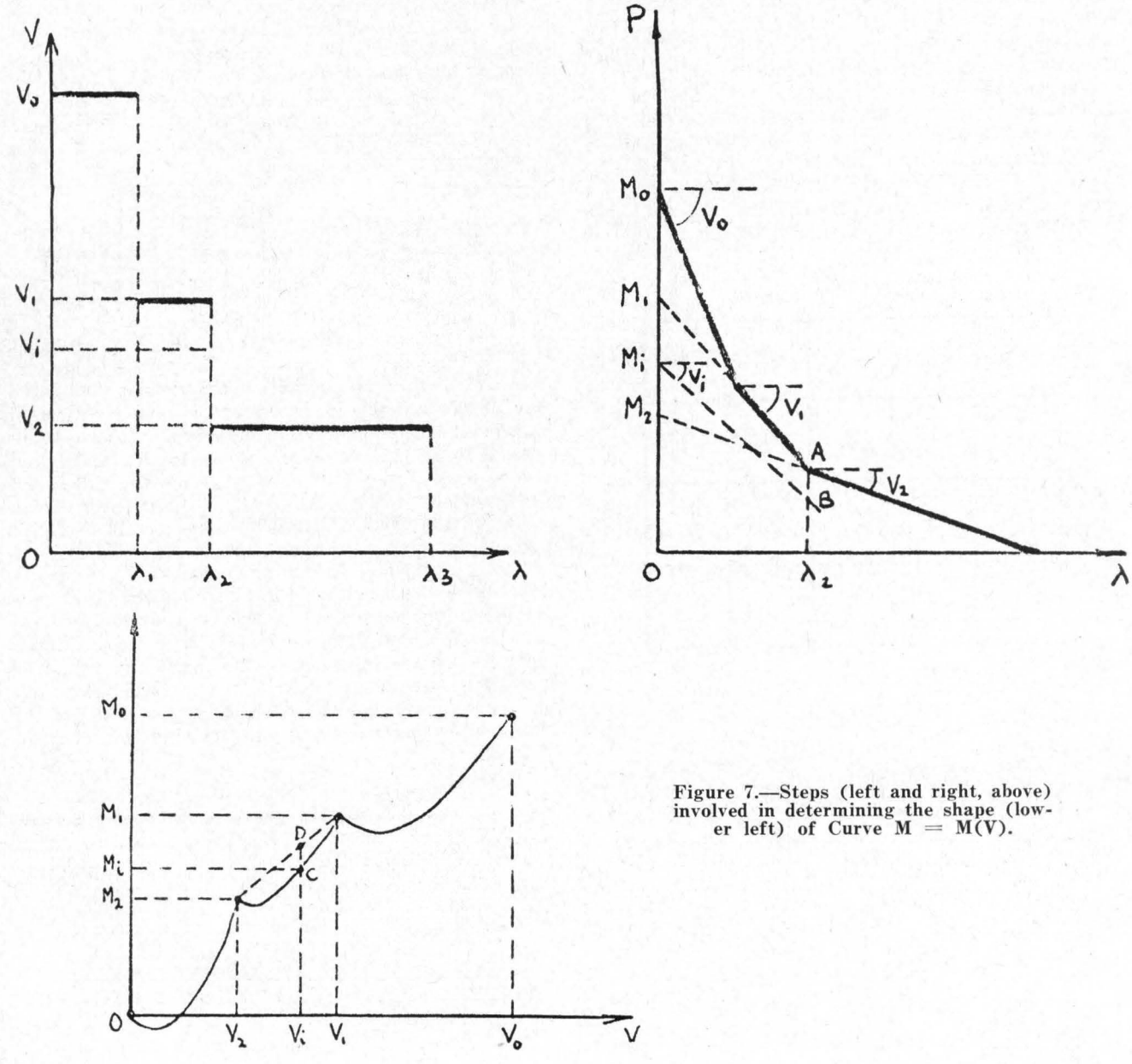

Figure 7.—Steps (left and right, above) involved in determining the shape (lower left) of Curve M = M(V).

With a sufficiently large λ, the contour will jump to a smaller volume. The function $V = V(\lambda)$ is a step function. $P = P(\lambda)$ is piecewise linear and convex; indeed, as long as V is constant, P is linear with λ and the slope of the line is V. As V jumps to a smaller value so does the slope.

$$M = P + \lambda V$$

Hence, the value of M corresponding to a volume V is given by the intersection of the segment of slope V and the axis OP. For each line segment of $P(\lambda)$, we can obtain a point of the curve $M = M(V)$. These points correspond to optimum contours for given volumes V. The total curve $M = M(V)$ cannot be generated by this process, but its shape is shown in Figure 7. Between any two of its characteristic points (M_2, V_2), (M_1, V_1), the curve $M(V)$ is convex. Indeed, if we go back to the curve $V(\lambda)$, the intermediate volume V_i defines the value λ_2. The point B on the surface P,λ, representing the optimum contour for V_i, must be situated below the curve $P(\lambda)$. To obtain M_i, we draw, from B, a segment of slope V_i and take its intersection with OP.

We can now write

$$M_i - \lambda_2 V_i < M_2 - \lambda_2 V_2 = M_1 - \lambda_2 V_1$$

From the equality

$$\lambda_2 = \frac{M_1 - M_2}{V_1 - V_2}$$

Substituting for λ_2

$$M_i < M_2 + (V_i - V_2)\ \frac{M_1 - M_2}{V_1 - V_2} \qquad (1)$$

But a point D on the segment (M_2, V_2), (M_1, V_1), has a value

$$M'_i = M_2 + (V_i - V_2)\ \frac{M_1 - M_2}{V_1 - V_2} \qquad (2)$$

From (1) and (2), it results that point C must indeed be situated below point D. The proposed graph algorithm can be easily extended to permit such parametric studies.

In summary, we have established the shape of the curve $M(V)$ and shown how its characteristic points can be obtained. To each point of this curve corresponds a contour that is optimum if the volume mined is exactly V. An interesting feature of the curve $M(V)$ is that given two optimum contours C_a and C_b corresponding to two volumes V_a and V_b then, for $V_a < V_b$, the contour C_b completely encloses the contour C_a, that is, any volume element contained in C_a is also contained in C_b.

It follows that if no other restrictions are imposed, the orebody can be depleted along the curve $M(V)$. This mining pattern will maximize the integral of cash flow with respect to total volume mined [$M(V)$ indeed is a cash flow].

APPENDIX

Maximum Closure of a Graph

Definitions

A directed graph $G = (X, A)$ is defined by a set of elements X called the *vertices* of G, together with a set A of ordered pairs of elements $a_i = (x, y)$, called the *arcs* of G. The graph G also defines a function Γ mapping X into X and such that

$$(x, y) \in A \rightleftarrows y \in \Gamma x.$$

A *path* is a sequence of arcs $(a_1, a_2, \ldots a_n)$ such that the terminal vertex of each arc corresponds to the initial vertex of the succeeding arc. A *circuit* is a path in which the initial vertex coincides with the terminal vertex. An *edge*, $e_i = [x, y]$ of G, is a set of two elements such that $(x, y) \in A$ or $(y, x) \in A$. This concept differs from that of an arc, which implies an orientation. A *chain* is a sequence of edges $[e_1, e_2, \ldots, e_n]$ in which each edge has one vertex in common with the succeeding edge. A *cycle* is a chain in which the initial and final vertices coincide.

A *subgraph* $G(Y)$ of G is a graph (Y, A_y) defined by a set of vertices of $Y \subset X$ and containing all the arcs that connect vertices of Y in G. A *partial graph* $G(B)$ of G is a graph (X, B) defined by a set of arcs $B \subset A$ and containing all the vertices of G. A *closure* of a directed graph $G = (X, A)$ is a set of vertices $Y \subset X$ such that $x \in Y \rightarrow \Gamma x \in Y$. If Y is a closure of G, then $G(Y)$ is a *closed subgraph* of G. By definition, the null set, $Y = \phi$, is also a closure of G.

A tree is a connected and directed graph $T = (X, C)$ containing no cycles. A *rooted tree* is a tree with one distinguished vertex, the *root*. The graph obtained by suppressing an arc a_i in a rooted tree T has two components. The component $T_i = X_i, A_i)$ which does not contain the root of the tree is called a *branch* of T. The root of the branch is the vertex of the branch that is adjacent to the arc a_i. A branch is a tree itself, and branches of a branch are called *twigs*.

The Problem

Given a directed graph $G = (X, A)$ and for each vertex x_i a numeric value $m_i \gtrless 0$, called the *mass* of x_i, find a closure Y of G with maximum mass. In other words, finds a set of elements $Y \subset X$ such that

$$x_i \in Y \longrightarrow \Gamma x_i \in Y$$

$$\text{and } M_Y = \sum_{x_i \in Y} m_i \text{ is maximum}$$

A closure with maximum mass is also called a *maximum closure.*

The Algorithm

The graph G is first augmented with a dummy node x_o and dummy arcs (x_o, x_i). The algorithm starts with the construction of a tree T^o in G. T^o is then transformed into successive trees $T^1, T^2, \ldots T^n$ following given rules, until no further transformation is possible. The maximum closure is then given by the vertices of a set of well identified branches of the final tree.

The trees constructed during the iterative process are characterized by a given number of properties. To highlight these properties and to avoid unnecessary repetitions we shall next develop some additional terminology.

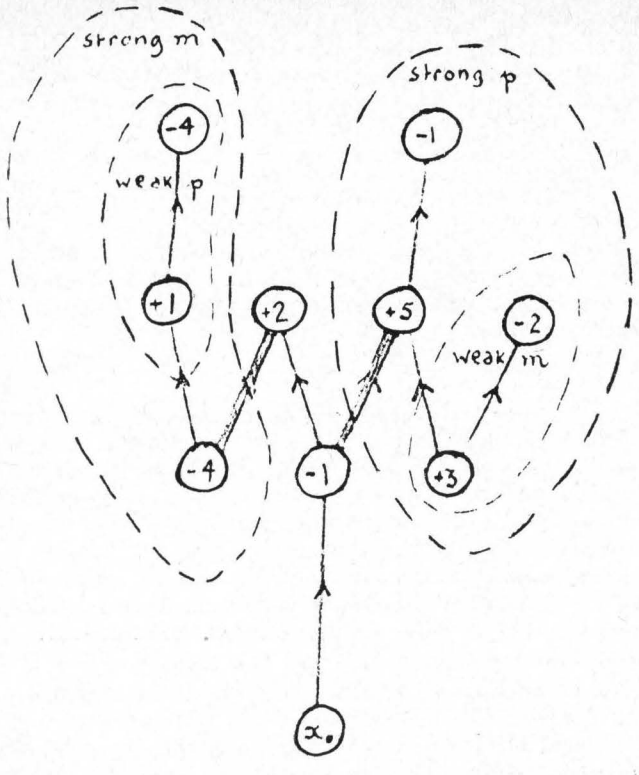

Figure 1.

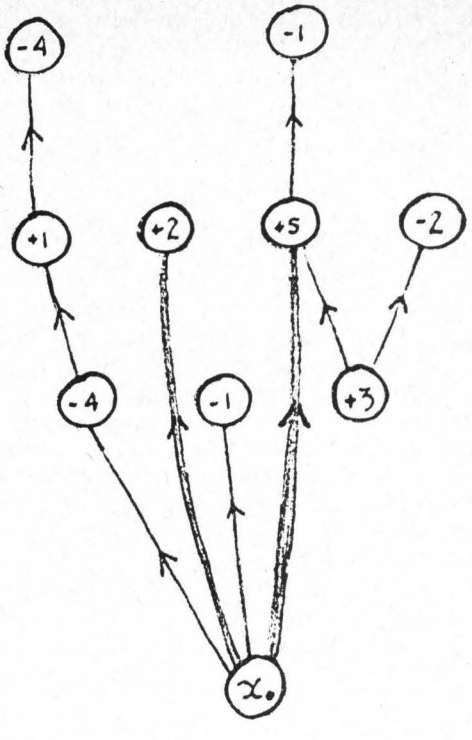

Figure 2.

Definitions

Each edge e_k (arc a_k) of a tree T defines a branch, noted as $T_k = (X_k, A_k)$. It is also convenient to write x_k for the root of the branch T_k. The edge e_k (arc a_k) is said to *support* the branch T_k. The mass M_k of a branch T_k is the sum of the masses of all vertices of T_k. This mass is associated with the edge e_k (arc a_k) and we say that the edge e_k (arc a_k) *supports* a mass M_k.

In a tree T with root x_o, an edge e_k (branch T_k) is characterized by the orientation of the arc a_k with respect to x_o; e_k is called a *p-edge* (plus-edge) if the arc a_k points toward the branch T_k, that is, if the terminal vertex of a_k is part of the branch T_k. T_k then is called a *p-branch*. If arc a_k points away from branch T_k, then e_k is called an m-edge (minus-edge) and T_k an *m-branch*. Similarly, all twigs of a branch can be divided into two classes: p-twigs and m-twigs. We shall also distinguish between *strong* and *weak* edges (branches). A p-edge (branch) is strong if it supports a mass that is strictly positive; an m-edge (branch) is strong if it supports a mass that is null or negative. Edges (branches) that are not strong are said to be weak. A vertex x_i is said to be strong if there exists at least one strong edge on the chain of T joining x_i to the root x_o. Vertices that are not strong are said to be weak. Finally, a tree is *normalized* if the root x_o is common to all strong edges. Any tree T of a graph G can be normalized by replacing the arc $(x_k\ x_l)$ of a strong p-edge with a dummy arc (x_o, x_l), the arc (x_q, x_r) of a strong m-edge with a dummy arc (x_o, x_q) and repeating the process until all strong edges have x_o as one of their extremities.

The tree in Figure 2 has been obtained by normalizing the tree in Figure 1. Note that as all dummy edges are p-edges, all strong edges of a normalized tree will also be p-edges.

The graph G considered in the sequel will be an augmented graph obtained by adding to the original graph a dummy vertex x_o with negative mass and dummy arcs (x_o, x_i), joining x_o to every vertex x_i. Because x_o cannot be part of any maximum closure of G, the introduction of dummy arcs (x_o, x_i) does not affect the problem. The vertex x_o will be the root of all trees considered.

We shall next establish properties of normalized trees. These properties will lead us to a basic theorem on maximum closures of a directed graph.

Property 1

If a vertex x_k belongs to the maximum closure Z of a normalized tree T, then all the vertices X_k of the branch T_k also belong to Z.

Proof:

We shall show that if a vertex, say x_r, of the branch T_k does not belong to Z, then Z is not a maximum closure.

Let (Figure 3) T(Z) and T(X-Z) be the subgraphs of T defined by the vertices of Z and X-Z, respectively, and assume

$$x_k \epsilon Z;\ x_r \epsilon X_k;\ x_r \epsilon X - Z$$

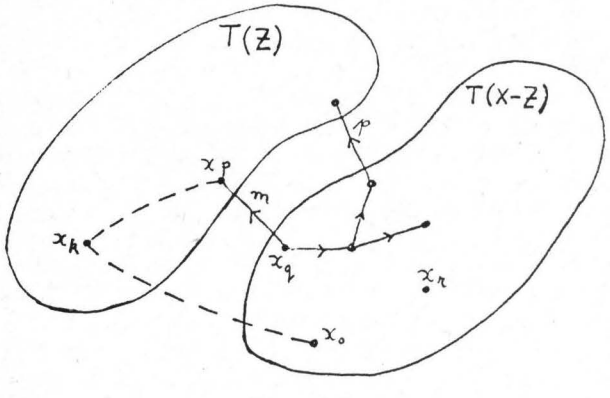

Figure 3.

All arcs A^* of T that join vertices of X-Z with vertices of Z have their terminal vertex in Z, as Z is a closure of T. At least one of the edges of A^* is an m-edge because the chain joining x_o to x_r must go over x_k and thus contain at least one of the edges of A^*. The first such edge between x_k and x_r is an m-edge. Let $e_q = [x_p, x_q]$ be this m-edge with $x_p \epsilon Z$ and $x_q \epsilon X\text{-}Z$ (possibly $x_p = x_k$ and/or $x_q = x_r$). T_q is an m-branch of T. Let T_q' be the component of T(X-Z) containing the vertex x_q. The edges of A^* connecting vertices of T_q' to vertices of Z, with the exception of $[x_q, x_p]$, are all p-edges in T, otherwise there would be a cycle in T. Hence T_q' is a branch obtained by removing p-twigs from the m-branch T_q of T. Because T is normalized, the mass of T_q is strictly positive; the mass of any p-twig of T_q is negative or null. Hence, the mass of T'_q is strictly positive and Z is not a maximum closure (the closure $Z + X_q'$ has larger mass). This completes the proof.

Property 2

The maximum closure of a normalized tree T is the set Z of its strong vertices.

If we note that any p-branch of T is a closed subgraph of T, but that an m-branch is not a closed subgraph of T, this property follows directly from property 1. If the tree has no strong vertex, that is, no strong edge, then the maximum closure is the empty set $Z = \phi$.

Theorem I

If, in a directed graph G, a normalized tree T can be constructed such that the set Y of strong vertices of T is a closure of G, then Y is a maximum closure of G.

Proof:

We shall use the following argument: If $S = (X, A^s)$ is a partial graph of G and if Z and Y are maximum closures of S and G, respectively, then obviously

$$M_Z \geq M_Y$$

If then we find a closure Y of G and a partial graph S of G for which Y is a maximum closure then, because of the above relation, Y must also be a maximum closure of G. Because T is a partial graph of G and because (property 2) Y is a maximum closure of T, the theorem follows immediately. If, in particular, the set of strong vertices of a normalized tree of G is empty, then the maximum closure of G is the empty set $Y = \phi$.

Steps of the Algorithm

Construct a normalized tree T^o in G and enter the iterative process. Iteration $t + 1$ transforms a normalized tree T^t into a new normalized tree T^{t+1}. Each tree $T^t = (X, A^t)$ is characterized by its set of arcs A^t and its set of strong vertices Y^t. The process terminates when Y is a closure of G. Iteration $t + 1$ contains the following steps:

1.—If there exists an arc (x_k, x_l) in G such that $Y_k \epsilon Y^t$, and $x_l \epsilon X\text{-}Y^t$, then go to step 2. Otherwise go to step 4.

2.—Determine x_m, the root of the strong branch containing x_k. Construct the tree T^s by replacing the arc (x_o, x_m) of T^t with the arc (x_k, x_l). Go to step 3.

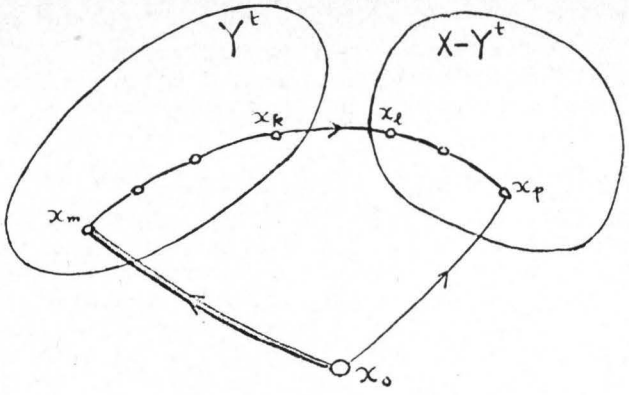

Figure 4.

3.—Normalize T^s. This yields T^{t+1}. Go to step 1.

4.—Terminate. Y^t is a maximum closure of G. (See Figure 4).

Construction of T°

T^o can be obtained by constructing an arbitrary tree in G and then normalizing this tree as outlined earlier. A much simpler procedure, however, is to construct the graph (X, A_D) where A_D is the set of all dummy arcs (x_o, x_i). This graph is a tree and it is, of course, normalized.

Transformations

The steps outlined above do not indicate the amount of calculation involved in each iteration nor do they establish that the process will terminate in a finite number of steps. To clarify these points we shall analyze, in more detail, the transformations taking place in steps 2 and 3 of the algorithm.

(a) *Construction of T^s*:

The tree T^s is obtained from T^t by replacing the arc (x_o, x_m) with the arc (x_k, x_l).

The arc (x_o, x_m) supports in T^t a branch $T_m{}^t$ with mass $M_m{}^t > O$. Let (Figure 4) $[x_m, \ldots, x_k, n_l, \ldots, x_p, x_o]$ be the chain of T^s linking x_m to x_o. Except for this chain, the status of an edge of T^s and the mass supported by the edge are unchanged by this transformation. On the chain $[x_m, \ldots, x_o]$ of T^s we have the following transformation of masses:

For an edge e_i on the chain $[x_m, \ldots, x_k]$

$$M_i{}^s = M_m{}^t - M_i{}^t \qquad (1)$$

For the edge $[x_k, x_l]$

$$M_k{}^s = M_m{}^t \qquad (2)$$

For an edge e_j on the chain $[x_l, \ldots, x_p, x_o]$

$$M_j{}^s = M_m{}^t + M_j{}^t \qquad (3)$$

In addition, all the edges e_i on the chain $[x_m, \ldots, x_k]$ have changed their status: a p-edge in T^t becomes an m-edge in T^s and vice versa. On the chains $[x_m, \ldots, x_k]$ and $[x_l, \ldots, x_p]$ in T^t, all p-edges support zero or negative masses and all m-edges support strictly positive masses as T^t is normalized. Hence, we obtain the following distribution of masses in T^s:

	m-edge	*p-edge*	
edge e_i on $[x_m, \ldots, x_k]$	$M_i{}^s \geq M_m{}^t$	$M_i{}^s < M_m{}^t$	(4)
edge $[x_k, x_l]$	$M_k{}^s = M_m{}^t$		(5)
edge e_j on $[x_l, \ldots, x_p, x_o]$	$M_j{}^s > M_m{}^t$	$M_j{}^s \leq M_m{}^t$	(6)

It results from these relations.

Property 3

If, in T^s, e_d is an m-edge on the chain $[x_m, \ldots, x_o]$ then the mass $M_d{}^s$ is strictly positive and larger than any mass suported by a p-edge that precedes e_d on the chain $[x_m, \ldots, x_o]$.

(b) *Normalization of* T^s:

As T^t was normalized, all strong edges must be on the chain $[x_m, \ldots, x_o]$. We remove strong edges one by one starting from the first strong edge encountered on the chain $[x_m, \ldots, x_o]$. This edge, say $e_a = [x_a, x_b]$, must be a p-edge (because of property 3, all m-edges are weak). We replace e_a with a strong dummy edge (x_o, x_a). Thus, we remove a p-twig from the branch $T_p{}^s$ and must subtract its mass from all the edges of the chain $[x_b, \ldots, x_o]$. Because of property 3, property 3 will remain valid on the chain $[x_b, \ldots, x_o]$. We now search for the next strong p-edge on the chain $[x_b, \ldots, x_o]$ and repeat the process until the last strong p-edge has been removed from the chain.

In practice, transformations (a) and (b) can be carried out simultaneously; we have analyzed them separately to establish the following:

Theorem II

In following the steps of the algorithm, a maximum closure of G is obtained in a finite number of steps.

Proof:

As the number of trees in a finite graph is finite, we only have to show that no tree can repeat itself in the sequence $T^o, T^1, \ldots, T^n$. Each normalized tree is characterized by its set Y of strong vertices and the mass M_y of this set. We shall show that either M_y decreases during an iteration or else M_y stays constant but the set Y increases, so that any two trees in the sequence $T^o, T^1, \ldots, T^n$ will differ either in their masses M_y or in their sets of strong vertices.

Indeed, let us see how M_y and Y transform during an iteration. Because, in the normalization process, we never generate a strong m-edge it is clear that the last p-edge removed from the chain $[x_m, \ldots, x_o]$ is the p-edge that supports the largest positive mass in T^s. Let e_w be this edge and $X_w{}^s$ the vertices of the branch $T_w{}^s$. As a result of steps (a) and (b), we now have

$$Y^{t+1} = Y - X_m{}^t + X_w{}^s \tag{7}$$

$$M_y{}^{t+1} = M_y{}^t - M_m{}^t + M_w{}^s \tag{8}$$

In any case, $M_w{}^s \leq M_m{}^t$ because of (4) and (6)

If $M_w{}^s < M_m{}^t$ then $M_Y{}^{t+1} < M_y{}^t$
If $M_w{}^s = M_m{}^t$ then $M_y{}^{t+1} = M_y{}^t$

The latter case can only occur if the equality applies in (6), and thus e_w must be situated on the chain $[x_l, \ldots, x_p, x_o]$ of T^s. Then, however, the set $X_w{}^s$ contains $X_m{}^t$ and the set Y^{t+1} is larger than the set Y^t.

This completes the proof.

References

(1) C. Berge, "The Theory of Graphs and its Applications," Wiley, 1962.

(2) Fulkerson, D. R., "A Network Flow Computation for Project Cost Curves," *Management Science*, Vol. 7, No. 2, January, 1961.

(3) Grossmann, I. F., and Lerchs, H., "An Algorithm for Directed Graphs with Application to the Project Cost Curve and In-Process Inventory, *Proceedings* of the Third Annual Conference of the Canadian Operational Research Society, Ottawa, May 4-5, 1961.

(4) Kelly, J. E., Jr., "Critical Path Planning and Scheduling: Mathematical Basis," *Operations Research* (U.S.), Vol. 9, (1961), No. 3, (May-June).

(5) Prager, William, "A Structural Method of Computing Project Cost Polygons," *Management Science*, April, 1963.