Operat. Res. Quart., Pergamon Press 1964. *Vol.* 15, *pp.* 261 *to* 270. *Printed in Great Britain*

On the Location of Supply Points to Minimize Transport Costs†

F. E. MARANZANA

IBM Italy

An algorithm applicable to the problem of locating supply points optimally with respect to transport costs is given.

Although the algorithm may fail to converge to an optimal solution, repeated application with judicious selections of alternative starting values will assure a good, if not optimal, solution.

The algorithm has been tested and some sample results are included.

1. INTRODUCTION

THE location of factories, warehouses—and supply points in general, to serve customers distributed over a network of cities—is often influenced by transport costs.

If transport costs are uniform and linear with respect to distance, the total transport cost is proportional to the sum of the distances from the supply points to the cities served, each weighted by the volume of shipments.

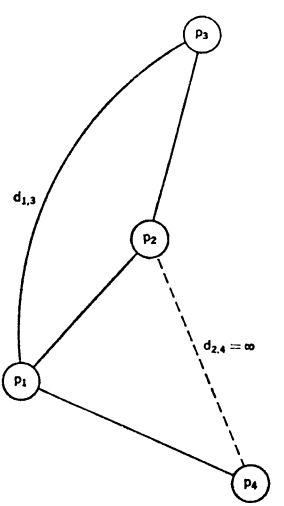

FIG. 1.

To illustrate notation and terminology, a network is shown in Figure 1. The cities in a network will be labelled with subscripted p's and referred to as

† Thanks are due to the Editor of the *IBM Systems Journal* for permission to publish this paper.

points or *nodes*. The set of p's will be denoted by P. The distance along a path from a node p_i to p_j not passing through intervening nodes will be denoted by $d_{i,j}$ and, of course $d_{i,j} = d_{j,i}$ and $d_{i,i} = 0$.

The shortest path from one node to another may pass through one or more intervening nodes—as in Figure 1 where the shortest path from p_1 to p_3 passes through p_2.

If a direct path exists between a pair of nodes, it will be assumed to be unique, but there may be no direct path between nodes, and in such cases we will put $d_{i,j} = \infty$ for computational purposes. However, it will be assumed that some path exists connecting every pair of nodes in the network.

The demand or volume of shipments to be made to node p_i will be denoted by an associated weight w_i. A node at which a supply point is located will be called a *source*; a node having a demand to be satisfied will be referred to as a *sink*. We may wish to consider locating a source at a city having no demand and it will be convenient to assign a weight of zero to such cities in order that they may be treated formally as sinks rather than on an exception basis.

Thus, in mathematical terms, we have the following problem. We *are given*: a set P of n points $p_1, \ldots, p_n$; a set of associated weights $w_1, \ldots, w_n$; a non-negative, n-dimensional, symmetric distance matrix $[d_{i,j}]$ and we *are required to find*: m sources, $p_{x_1}, \ldots, p_{x_m}$, an associated partition of P into m subsets of sinks, $P_{x_1}, \ldots, P_{x_m}$, served respectively by the m sources so that

$$\sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i,j} w_j \tag{1}$$

is a minimum where $D_{i,j}$ is the minimum path length from p_i to p_j. Transport cost is proportional to the summation (1). Solution by means of direct enumeration is obviously impractical for the typical problem.

In this paper, an iterative procedure is described which can be applied to an initial selection of m supply points to produce successively improved selections.

The final solution obtained in this manner is not necessarily optimal. However, with a computer it is feasible to carry out the procedure on a number of different initial selections so that one can be assured of arriving at a good solution even though it may not be optimal. The determination of the appropriate number of supply points can be based on a comparison of results obtained by repeating the procedure for different values of m.

The algorithm for the above procedure incorporates two items:

(1) An algorithm to find the shortest path between any two points of a network.

(2) A routine for determination of the "centre of gravity of a set of weighted nodes" (this notion is subsequently defined).

Both items (1) and (2) are detailed prior to presentation of the algorithm. Although the former is the well-known work of Bellman,[1] it is included for the convenience of the reader.

After statement of the algorithm, its monotonicity is shown and non-optimal convergence is examined. The paper is concluded with some sample results obtained by applying the algorithm with the aid of a digital computer.

2. BELLMAN'S ALGORITHM

Given a point p_s, to find $D_{r,s}$, the minimum path length from any point p_r to p_s, we use the recursive relations:

$$a_j^1 = d_{j,s} \quad \text{and} \quad a_j^i = \min_k \{d_{j,k} + a_k^{i-1}\}, \tag{2}$$

where $1 \leqslant j, k \leqslant n$ and $2 \leqslant i \leqslant n-1$.

Clearly, a_r^1 is the minimum distance of any path joining p_r and p_s and passing through at most two nodes including p_r and p_s, a_r^2 is the minimal length of any path passing through at most three nodes including p_r and p_s and a_r^i is the shortest path from p_r to p_s containing at most $i+1$ nodes including both p_r and p_s. It is also apparent that $a_r^1 \geqslant a_r^2 \geqslant \ldots \geqslant a_r^{n-1}$. Thus,

$$D_{r,s} = a_r^t, \tag{3}$$

where t is the smallest integer such that

$$\text{either} \quad a_r^t = a_r^{t+l} \quad \text{for all } l \quad \text{or} \quad t = n-1.$$

We may use (2) and (3) as the basis of an algorithm for the computation of $D_{r,s}$. We simply compute the vectors $a^i = a_1^i, a_2^i, \ldots, a_n^i$ (with ascending i) in accordance with (2) until (3) is satisfied.

Although the order of computation of the components of an individual vector is arbitrary, we note that the computation of a particular component of a particular vector, say a^i, requires prior computation of all components of all vectors earlier in the sequence (i.e. all components of the vectors a^j for $j < i$).

Improvement in the latter procedure is possible. For each vector a^i, we may agree to compute the components a_j^i, in order of ascending j, using (2) as before for the computation of the first component a_1^i, but substituting the following rule for the computation of the other components:

$$a_j^i = \min_k \{d_{k,j} + a_k^{i-l}\} \tag{4}$$

where $l = 0$ if $k < j$, otherwise $l = 1$.

In this recursive rule for the successive computation of the components of the vectors, the original procedure has been modified by using a_k^i in place of a_k^{i-1} in the computation of a_r^i whenever $k < r$. The amount of computation is precisely as before. The effect of the modification is that in computing the second component of the second vector, not only are all paths with at most three nodes

considered as before but, in addition, some paths with four nodes are included; in the case of the third component, some paths with five nodes; and so forth. The propagation of this effect through the computation of successive vectors will be apparent.

3. CENTRE OF GRAVITY

The physical concept of centre of gravity motivates definition of "the centre of gravity of a network". The *centre of gravity* is defined as follows: p_j is a centre of gravity of $Q \subseteq P$, if

$$\sum_{p_k \in Q} D_{j,k} w_k \leqslant \sum_{p_k \in Q} D_{i,k} w_k \quad \text{for all } i. \tag{5}$$

Observe that an optimal location of a single source is at the centre of gravity of the set. We will show later that the notion is not necessarily unique.

The sums appearing in (5) may be evaluated, a minimal one found by comparative examination which in turn determines j and hence a centre of gravity p_j. This is feasible since the Bellman algorithm for computing $D_{i,j}$ is rapid (often only two or three a^i-vectors need to be evaluated).

4. THE ALGORITHM

The algorithm starts with an arbitrary selection of sources and partitions the network into subsets to be served by these sources. This is accomplished by associating each point with its "nearest" source. Next, the centre of gravity of each set in the partition is determined and the original sources are replaced by these points. The process is repeated until the source points do not change. A formal statement of the algorithm follows.

Algorithm

Step 1. Arbitrarily select m distinct points in P and assign these points to the variable array, p_{x_i}, appearing in Step 2.

Step 2. Associated with the array of m points, $p_{x_1}, \ldots, p_{x_m}$, determine a corresponding partition of $P, P_{x_1}, \ldots, P_{x_m}$, by putting

$$P_{x_i} = \{p_k;\ D_{k,x_i} \leqslant D_{k,x_j} \text{ for all } j\}.$$

Step 3. Determine a centre of gravity, c_{x_i}, for each P_{x_i}.

Step 4. If $c_{x_i} = p_{x_i}$ for all i, computation is stopped and the current values of p_{x_i} and P_{x_i} constitute the desired solution. Otherwise, set $p_{x_i} = c_{x_i}$ and return to Step 2.

In Step 2, if a point is equidistant from more than one source a decision is required relative to placement of the point—we agree, arbitrarily, to put the point in the set associated with the source p_{x_i} having the smallest i. In Step 3, if the centre of gravity is non-unique, we will likewise make the arbitrary decision to select the point with smallest subscript.

5. MONOTONICITY OF THE ALGORITHM

The algorithm is demonstrated to be monotonic by showing that

$$\sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i,j}\, w_j$$

is monotonely non-increasing with respect to the successive selection of P_{x_i} according to Step 2 and the successive selection of values for x_i according to Step 4. This is accomplished, respectively, by the following lemmas:

Lemma 1. Given a set of m distinct points in $P, p_{x_1}, \ldots, p_{x_m}$; and an arbitrary partition of P into m subsets, $Q_{x_1}, \ldots, Q_{x_m}$, satisfying the condition $x_i \in Q_{x_i}$; if the partition $P_{x_1}, \ldots, P_{x_m}$ is constructed in accordance with Step 2, then

$$\sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i,j}\, w_j \leqslant \sum_{i=1}^{m} \sum_{p_j \in Q_{x_i}} D_{x_i,j}\, w_j.$$

Proof of Lemma 1. We introduce A_i, B_i, B_i' by putting $A_i = Q_{x_i} \cap P_{x_i}$, $B_i = Q_{x_i} - P_{x_i}$ and $B_i' = P_{x_i} - Q_{x_i}$. Thus we have

$$Q_{x_i} = A_i \cup B_i \quad \text{and} \quad P_{x_i} = A_i \cup B_i',$$

$A_i \cap B_i$ and $A_i \cap B_i'$ are null, and $\cup B_i = \cup B_i'$. Proceeding with the proof we have:

$$\sum_{i=1}^{m} \sum_{p_j \in Q_{x_i}} D_{x_i,j}\, w_j - \sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i,j}\, w_j = \sum_{i=1}^{m} \sum_{p_j \in B_i} D_{x_i,j}\, w_j - \sum_{i=1}^{m} \sum_{p_j \in B_i'} D_{x_i,j}\, w_j.$$

We now "re-index" each sum by means of k and k' as defined implicitly, respectively, in the expressions ($k = x_i$ if $p_j \in Q_{x_i}$) and ($k' = x_i$ if $p_j \in P_{x_i}$). The above difference in sums may now be written as

$$\sum_{p_j \in \cup B_i} D_{k,j}\, w_j - \sum_{p_j \in \cup B_i'} D_{k',j}\, w_j$$

and since $\cup B_i' = \cup B_i$ we may write the expression as

$$\sum_{p_j \in \cup B_i'} w_j (D_{k,j} - D_{k',j}).$$

If $p_j \in B_i'$ then $k' = x_i, k = x_l$ for some l, and $p_j \in P_{x_i}$. Thus

$$D_{k,j} - D_{k',j} = D_{x_l,j} - D_{x_i,j}$$

which by the symmetry of D and Step 2 is non-negative and the inequality stated in the lemma holds.

Lemma 2. Given a set of m points in P, $p_{x_1}, \ldots, p_{x_m}$; a partition of P into m subsets, $P_{x_1}, \ldots, P_{x_m}$ with $p_{x_i} \in P_{x_i}$; if the points, $p_{x_1'}, \ldots, p_{x_m'}$, are the respective centres of gravity of the sets in the partition, then

$$\sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i',j}\, w_j \leqslant \sum_{i=1}^{m} \sum_{p_j \in P_{x_i}} D_{x_i,j}\, w_j.$$

Proof of Lemma 2. Since $p_{x_{i'}}$ is the centre of gravity of P_{x_i} we have immediately from (5), the definition of the latter concept, that

$$\sum_{p_j \in P_{x_{i'}}} D_{x_{i'},j}\, w_j \leqslant \sum_{p_j \in P_{x_i}} D_{x_i,j}\, w_j$$

and the lemma follows.

6. NON-OPTIMAL CONVERGENCE

Certain conditions under which the algorithm will fail to converge to an optimal solution can be readily identified.

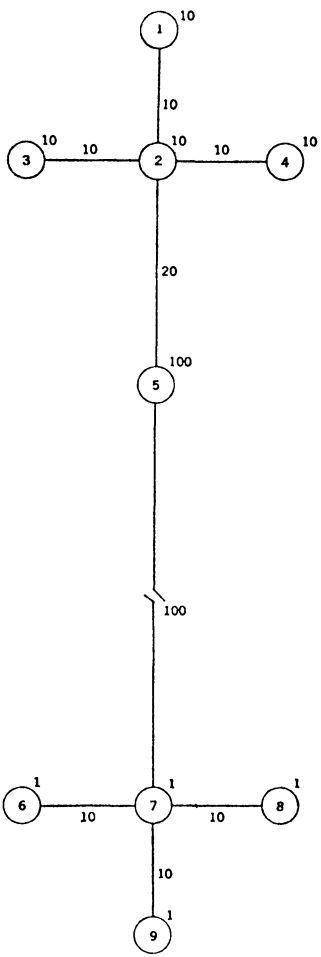

FIG. 2.

The example shown in Figure 2 has been constructed so that the optimal two-source solution, p_2 and p_5, is self-evident. Now if p_1, p_8 are chosen as the

initial values, the algorithm will produce the sequence of computations shown in Table 1 and converge to the non-optimal solution p_5, p_7. The difficulty is that the initial selection of sources generates a non-optimal partition $\{p_1, p_2, p_3, p_4, p_5\}$, $\{p_6, p_7, p_8, p_9\}$ which is "stable" with respect to the algorithm. However, p_5 and p_7 are optimal sources with respect to the partition.

TABLE 1

Sources	p_1, p_8	
Partition	$\{p_1, p_2, p_3, p_4, p_5\}$,	$\{p_6, p_7, p_8, p_9\}$
Centres of gravity	p_5, p_7	
Partition	$\{p_1, p_2, p_3, p_4, p_5\}$,	$\{p_6, p_7, p_8, p_9\}$
Centres of gravity	p_5, p_7	(convergence)

To avoid the difficulty arising in the previous paragraph, we might apply the algorithm to initial selection of both initial sources from one or another of the clusters. Suppose p_1, p_2 are selected. Action of the algorithm is displayed in Table 2 and this time is convergent to the optimal solution, p_2 and p_5 as sources serving, respectively, for the sets of sinks $\{p_1, p_2, p_3, p_4\}$ and $\{p_5, p_6, p_7, p_8, p_9\}$.

TABLE 2

Sources	p_1, p_2	
Partition	$\{p_1\}$,	$\{p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$
Centres of gravity	p_1, p_5	
Partition	$\{p_1, p_2, p_3, p_4\}$,	$\{p_5, p_6, p_7, p_8, p_9\}$
Centres of gravity	p_2, p_5	
Partition	$\{p_1, p_2, p_3, p_4\}$,	$\{p_5, p_6, p_7, p_8, p_9\}$
Centres of gravity	p_2, p_5	(convergence)

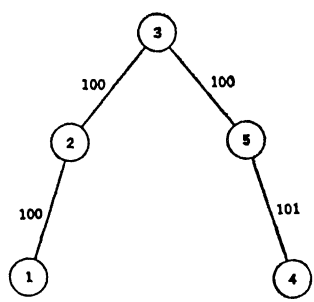

FIG. 3.

Another difficulty arises in cases where the centre of gravity is non-unique. The choice made in such instances may influence the solution attained by the algorithm. For the network shown in Figure 3 (weights assumed equal), Table 3 details the path traced by the algorithm with the decision rule as previously indicated.

If this rule were modified to select the centre of gravity with highest index, the algorithm would follow the path shown in Table 4. The former computation converges to p_2, p_4, an optimal solution, but the latter terminates with p_3, p_4 which is not optimal.

TABLE 3

Sources	p_4, p_5
Partition	$\{p_4\}, \quad \{p_1, p_2, p_3, p_5\}$
Centres of gravity	p_4, p_2
Partition	$\{p_4, p_5\}, \quad \{p_1, p_2, p_3\}$
Centres of gravity	p_4, p_2 (convergence)

TABLE 4

Sources	p_4, p_5
Partition	$\{p_4\}, \quad \{p_1, p_2, p_3, p_5\}$
Centres of gravity	p_4, p_3
Partition	$\{p_4\}, \quad \{p_1, p_2, p_3, p_5\}$
Centres of gravity	p_4, p_3 (convergence)

7. AN EXAMPLE

The work reported in this paper was undertaken in connexion with a two-source problem arising in Italy involving a network of 158 cities. The algorithm was programmed in Fortran and an IBM 704 used for computation. Processing time was consumed almost exclusively in the construction of the matrix of distances between nodes. In fact, a number of tests were performed with the same network and many initial solutions; the first test, during which the matrix was constructed, required 80 min, the following ones were carried out in less than 1 min. This time may seem excessive to the reader but it may reflect the minimal effort expended in coding rather than the efficiency of the algorithm.

The tests mentioned in the preceding paragraph led to interesting results. Regardless of the choice of initial solution, the algorithm converged constantly to the same final solution; as the two nodes, that represented this solution, seemed at the outset as sensible guesses, they often appeared in the initial solutions. In successive trials, the two nodes were chosen as the initial solution; each of them in turn was then coupled to a third node in the initial solution and nodes different from the first two were chosen as an initial solution. In all cases, convergence was attained when the sources were moved to Rome and Milan. The fact that the algorithm converged to the same solution regardless of the initial choice was thought to indicate that the solution obtained was indeed optimal.

Figure 4 shows the results of a computer run involving a forty-sink three-source problem used for programme checking. The map suggests the relative distances used in the problem. The weights, which are hypothetical, are listed.

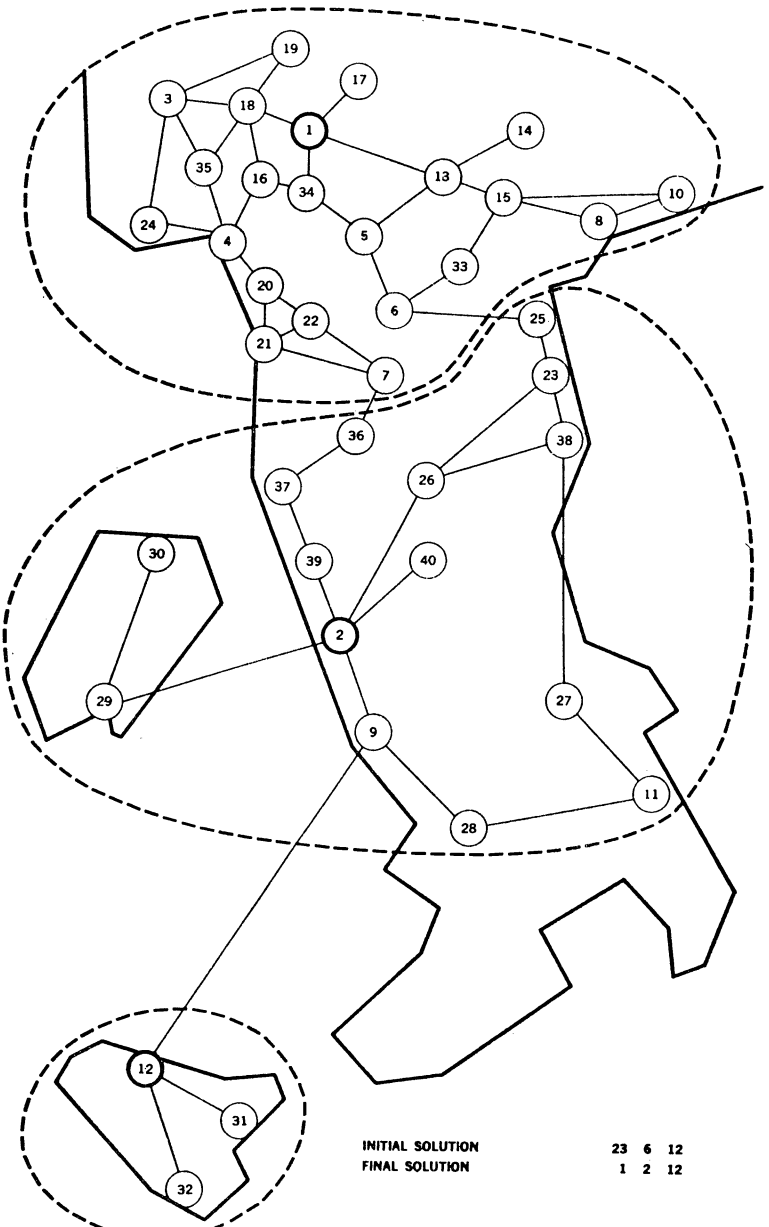

FIG. 4. *Example of forty-sink three-source problem.*

[For key see overleaf.]

Key to Figure 4

City	Index	Weight	City	Index	Weight
Milano	1	50,000	Pisa	21	100
Roma	2	50,000	Lucca	22	100
Torino	3	30,000	Fano	23	30
Genova	4	25,000	Savona	24	50
Parma	5	10,000	Pesaro	25	300
Bologna	6	15,000	Perugia	26	300
Firenza	7	15,000	Foggia	27	25
Venezia	8	10,000	Avellino	28	2
Napoli	9	15,000	Cagliari	29	5,000
Trieste	10	10,000	Sassari	30	15
Bari	11	5,000	Catania	31	8
Palermo	12	5,000	Siracusa	32	4
Verona	13	8,000	Ferrara	33	40
Trento	14	3,000	Piacenza	34	6
Padova	15	7,000	Alessandria	35	6
Pavia	16	500	Siena	36	30
Sondrio	17	300	Grosseto	37	15
Novara	18	500	Ancona	38	4,000
Biella	19	200	Viterbo	39	2
La Spezia	20	200	Rieti	40	2

The solution obtained, source points and partition, are indicated on the map. Processing time for the run was approximately 1 min.

Other recent work on the problem discussed in this paper, as well as related problems, is reported by Cooper in Refs. 2 and 3.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the assistance received from Michael Held—in particular for providing insight relative to cases in which the algorithm does not converge to an optimal solution.

He is also indebted to G. Sommi and G. Mori for their help in the preparation of the examples reported in this article and in programming part of the algorithm.

REFERENCES

[1] R. Bellman (1958) On a routing problem. *Quart. appl. Math.* **16**, 87.
[2] L. Cooper (1963) Location–allocation problems. *Operat. Res.* **2.**
[3] L. Cooper, Heuristic methods for location–allocation problems (unpublished).