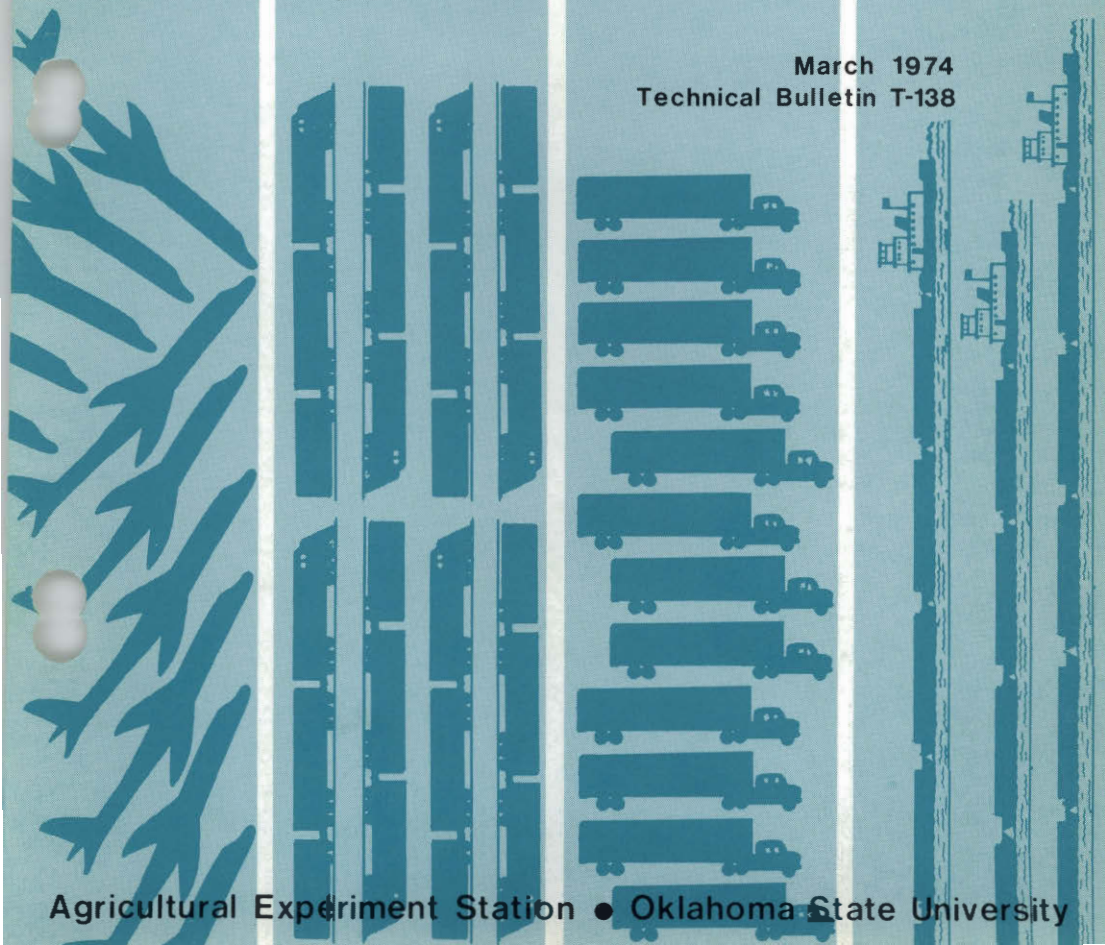
Algorithms for Developing Least-Cost Transportation Systems With Multiple Transportation Modes

March 1974
Technical Bulletin T-138



Agricultural Experiment Station • Oklahoma State University

CONTENTS

Preface

This paper describes two algorithms for determining least-cost transportation systems where more than one mode of transportation is considered.

The problem confronted here differs from the classical transportation problem in two respects. First, more than one activity may exist for transporting commodities between any two given shipment points, and second, non-negative transfer costs may be incurred whenever commodities are shifted between modes of transportation while in route.

Since many sophisticated linear programming packages are already available, (which can possibly be used to solve transportation problems) only the application of the other parts of the algorithms is discussed in detail.

Information pertaining to computer programs designed to carry out the dynamic programming phases of some of the algorithms is available from the author.

Research reported herein was conducted under Oklahoma Station Project No. 1024.

Algorithms for Developing Least-Cost Transportation Systems With Multiple Transportation Modes

Richard E. Just
Department of Agricultural Economics

INTRODUCTION

Transportation programming has found many applications in economic marketing research. However, the assumptions of the classical transportation model prevent the direct analysis of a transportation system with multiple modes of transportation where transfer costs are incurred when the mode of transportation is shifted. For example, the transportation model generally requires input data specifying the cost per unit of transporting goods from, say, any origination point A to any destination point B. However, the determination of the cost of transportation from A to B alone may involve a considerable problem in the selection of modes between various pairs of intermediate points. In addition, where transfer costs are involved, the possibility of applying the generalized transportation model (where origins are not computationally distinguished from destinations) is prevented since the transporting of goods from point A to some point C through point B may or may not involve transfer costs depending on the transportation modes selected.

In this paper, two general approaches are outlined for the solution of a generalized transportation problem with multiple modes and associated transfer costs. The first rather obvious approach involves conversion of the entire problem into a larger linear programming problem which can then be solved by any of the various existing simplex procedures. The second

approach involves splitting the problem into two steps. In the first step, a dynamic programming procedure is used to determine the least-cost mode-routes between all points of origination and all points of destination. Here, a mode-route specifies not only the route but also which modes of transportation are to be used on the various segments of the route. In the second step, the costs corresponding to the least-cost mode-routes are then used in a generalized transportation program to complete the solution of the problem. Again, any of the various existing simplex procedures for linear programming or transportation problems can be used for the second step of the procedure.

Although the simplex procedures involved in either of these approaches are not discussed in any detail here since many very efficient and highly adaptable computer programs in this area already exist, several dyanmic programming procedures which can be used for the second algorithm are presented in some detail.

The algorithms presented here should help permit study of a wide variety of economic marketing problems. The procedures are, of course, of obvious applicability in marketing problems where truck, rail, and possibly barge or air transportation must be considered and where transfer costs must be incurred when commodities are shifted from truck to rail, truck to barge, etc. Another important area of applicability, however, lies in areas such as determining an efficient pick-up or delivery system for a specific commodity where various sizes of trucks, etc. are available. For example, a least-cost system might involve service of many very small demand (supply) points with small trucks, but also the use of larger trucks to transport the commodity between the central firm or market and some point of

transfer to (from) the smaller trucks. Finally, a third area of application lies in problems of deciding where to locate processing plants. Defining the modes of transportation as (1) transport of raw goods, and (2) transport of processed goods, the unloading, processing, and loading cost can be considered as a transfer cost. If there are several independent steps involved in processing, then more than two modes of transportation could be considered. Perhaps then the least expensive transportation solution for some commodities might suggest transporting goods in some semi-processed state.

Solution by the Simplex Procedure

As described by Dantzig [3, pp. 299-322], the classical transportation problem involves determining the optimal schedule of shipments when

(i) fixed stockpiles exist at various supply points,

(ii) shipments are sent directly to various demand points which have fixed requirements,

(iii) total demand equals total supply,

and

(iv) costs satisfy a linear objective function.

This problem can be simply described in equational form as

$$\min_{x_{ij} \geq 0} \sum_{i,j} c_{ij} x_{ij}$$

subject to

$$\sum_{j} x_{ij} = a_i \qquad \forall_i$$

$$\sum_{i} x_{ij} = b_j \qquad \forall_j$$

where

x_{ij} = amount of shipment from point i to point j,

c_{ij} = cost per unit of shipment from point i to point j,

a_i = fixed supply at point i, and

b_j = fixed demand at point j.

For the problem at hand, however, it will be more convenient to relate our discussion to the generalized transhipment problem [3, pp. 335-342] which assumes more generally in place of (i) and (ii) that

(i') fixed production and/or consumption takes place at various shipment points, and

(ii') shipments can possibly pass through various intermediate points which might also be production and/or consumption points.

Since this generalized transportation model does not computationally distinguish origins from destinations, it may be simply represented as

$$(1) \quad \min_{x_{ij} \geq 0} \sum_{i,j} c_{ij} x_{ij}$$

subject to

$$(2) \quad \sum_{j \neq 1} x_{ij} - \sum_{k \neq i} x_{ki} = a_j^* - b_i^* \qquad \forall\, i$$

where

a_i^* = production at point i,

b_i^* = consumption at point i.

Consider then the variables and constraints which must be added to the problem in (1) and (2) to provide the appropriate generalizations for multiple modes. If transfer costs are not incurred, then multiple modes are simply included by adding variables corresponding to the various modes.

Let

x_{ijk} = amount of shipment from point i to point j by mode k,

c_{ijk} = cost per unit of shipment from point i to point j by mode k.

The problem then becomes

$$(3) \qquad \min_{x_{ijk} \geq 0} \sum_{i,j,k} c_{ijk} \, x_{ijk}$$

subject to

$$(4) \qquad \sum_{\substack{j \neq i \\ k}} x_{ijk} - \sum_{\substack{j \neq i \\ k}} x_{jik} = a_i^* - b_i^* \qquad \forall \, i.$$

Of course, in this simple case where no transfer costs are involved, many of the x_{ijk} can be eliminated from the problem by inspection. Clearly, all x_{ijk} such that $c_{ijk^*} < c_{ijk}$ for some k* will never be greater than zero in a minimum cost solution because another identical activity (corresponding to k*) exists with a smaller cost. For this reason, the problem in (3) and (4) can always be reduced to a problem of the same size as that in (1) and (2) and, furthermore, any standard transportation problem algorithm can be used to determine a least-cost transportation system in a rather obvious manner.

The problems created by the addition of transfer costs, however, cannot be so easily handled. For each shipment point we must include not only several variables representing the transfer of commodities from one mode to another, but also several constraints to insure that the transfer costs are correctly imposed in the model. Suppose additional variables are defined as follows:

x_{ijk}^T = amount of commodities transferred from mode j to mode k at shipment point i,

c_{ijk}^T = cost per unit of transferring commodities from mode j to mode k at shipment point i,

x_{ij}^{c} = excess consumption at point i inshipped by mode j,

x_{ij}^{p} = excess production at point i outshipped by mode j.

The objective function would then become[1/]

$$(5) \quad \min_{x_{ijk},\, x_{ijk}^{T} \geq 0} \sum_{i,j,k} c_{ijk}\, x_{ijk} + \sum_{i,j,k} c_{ijk}^{T}\, x_{ijk}^{T}.$$

$$x_{ij}^{p},\, x_{ij}^{c} \geq 0$$

The supply-demand constraints in (4) can be replaced by

$$(6) \quad \sum_{j} x_{ij}^{p} - \sum_{j} x_{ij}^{c} = a_i^{*} - b_i^{*} \quad \forall\ i.$$

The additional constraints needed to balance shipments by each mode at each shipment point are then

$$(7) \quad \sum_{j \neq 1} x_{ijk} - \sum_{j \neq 1} x_{jih} - \sum_{k} x_{ikh}^{T} + \sum_{k} x_{ihk}^{T} + x_{ih}^{c} - x_{ih}^{p} = 0$$

$$\forall\ i,\ h.$$

The additional constraints in (7) which must be imposed, basically force the transfers to mode h at point i less the transfers from mode h at point i to be equal to inshipments on mode h less outshipments on mode h aside from the production (consumption) shipped out (in) on mode h.

Where J is the total number of shipment points and K is the number of possible modes of transportation, the addition of transfer costs can then possibly lead to the addition of J x K! transfer variables (when all transfers are possible at all points), J x K production/consumption variables (if $a_i^{*} - b_j^{*} > 0$ then x_{ij}^{c} can be dropped for all j; if $a_i^{*} - b_i^{*} < 0$ then x_{ij}^{p} can be dropped for all j), and J x K constraints (assuming all J points are serviced by all K modes). Furthermore, the problem in (5) with constraints (6) and (7) can no longer be cast as a classical transportation model or generalized

[1/] Of course loading or unloading costs could also be attached to production or consumption, but for the purposes of this paper, such costs are ignored.

transhipment problem. Hence, transportation algorithms can no longer be used to solve the problem; computationally more involved linear programming procedures must be employed.[2/]

Solution by Dynamic Programming/Simplex Procedures

Obviously, the size of the linear programming problem in (5), (6), and (7) quickly get out of hand as J, the number of shipment points, or K, the number of modes, increase. A problem with 200 shipment points and only 3 modes requires consideration of over one-hundred thousand variables. In this section a dynamic programming procedure is outlined which allows us to determine a large number of activities in the problem which are clearly dominated (cost-wise) and can thus be discarded from the problem. After this procedure is carried out, the remaining problem can be simply solved as a classical transportation problem of at most, size M by N where M + N = J.[3/]

Several dynamic programming algotithms have been developed for the purpose of finding the shortest path (or least-cost) between specified

---

[2/] Actually it is possible, but not often practical, to solve this problem with a transportation algorithm directly if the loading docks for the various modes of transportation at each shipment point in the problem are considered as distinct shipment points with the transfer costs becoming the associated transportation costs from one loading dock to another. With this approach, however, it becomes necessary to consider, in effect, J x (K + 1) shipment points (when all points are serviced by all modes). This essentially amounts to solving a linear programming problem with J x (K + 1) constraints and $J^2 \times (K + 1)^2 - J \times (K + 1)$ variables by a compact and computationally efficient transportation algorithm. Alternatively solving the problem in (5), (6), and (7) by a standard linear programming algorithm involves J x (K + 1) constraints but only $J^2K + JK!$ variables. The problem in (5), (6), and (7) then involves considerably fewer variables when J is reasonably large and K is reasonably small (again assuming all points are serviced by all modes).

[3/] Note that this is even smaller than the corresponding generalized transhipment problem in (3) and (4) which does not include multiple modes and transfer costs.

pairs of nodes or shipment points (see, e.g., Dreyfus [5] for an appraisal of available methods). With a slight variation of these algorithms which account for transfer costs we can find least-cost mode-routes between all pairs of nodes which specify not only the route but also which modes are to be used between all pairs of intermediate points on the route. Since any flow of commodities between any two nodes in a least-cost transportation system must take place on a least-cost mode-route between those two nodes, no more than (J-1) x min (M,N) least-cost mode-routes between the M surplus (N deficit nodes) and all other nodes need be considered in solving the problem in (5), (6), and (7).[4/]

Consider the problem of finding the least-cost mode-route between given pairs of points where we know (as above) the (least-) cost c_{ijk} of transporting goods between all pairs of nodes (i, j) by a given mode k. Without loss of generality (since the problem is linear), we will assume only one unit of the commodity must be transported. The major dynamic programming procedures (see, e.g., Dreyfus [5]) proposed for solving this least-cost route problem when only one mode of transportation is possible are those of Dijkstra [4], Bellman [1] and Ford [7], Floyd [6], and Dantzig [2]. Of these, the Dijkstra and Bellman-Ford procedures solve the problem for a given pair of nodes whereas the Floyd and Dantzig procedures solve the problem for all pairs of nodes simutaneously (and more efficiently than the application of the former procedures to all pairs of nodes).

[4/] Actually, only the N x M mode-routes between surplus and deficit nodes must be used in the transportation procedure. However, all nodes must be considered as intermediate points and thus in the procedure proposed here, all nodes must be considered as destinations except in the last iteration.

A Generalized Bellman-Ford Procedure

Unfortunately, of the procedures dealing with a given pair of nodes, only the Bellman-Ford algorithm can seemingly be easily generalized for the K mode problem where transfer costs are involved. For the single-mode problem, the Bellman-Ford procedure involves defining

f_{ij}^{k} = least-cost route connecting nodes i and j with k or fewer intermediate nodes (k+1 or fewer arcs)

and solving the problem for routes between node m and all other nodes by computing in iterations k = 0, 1, 2 ...

$$f_{mj}^{k+1} = \min_{1 \leq i \leq J} (f_{mi}^{k} + c_{ij}) \qquad j = 1,\ldots, J;\ j \neq m$$

where initial or boundary conditions are given by

$$f_{mj}^{o} = c_{mj} \qquad j = 1,\ldots, J;\ j \neq m.$$

The procedure terminates when $f_{mj}^{k+1} = f_{mj}^{k}$ for j = 1,..., J (j ≠ m) which always occurs in J-2 or fewer iterations when negative cycles do not exist. The all-pairs problem can, of course, be solved by executing the procedure J times for m = 1,..., J. For a proof of the optimality of this algorithm, the reader is referred to Bellman [1].

Here a very simple generalization of this procedure for the K-mode case is proposed. Let

g_{ijpq}^{k} = least-cost mode-route connecting nodes i and j beginning on mode p and terminating on mode q with k or fewer transfers, and

f_{ij}^{k} = least-cost mode-route connecting nodes i and j.

The K-mode problem of least-cost routes between node m and all other nodes can then be solved by finding in iterations k = 0, 1, 2,...,

$$(8)\quad g_{mjpq}^{k+1} = \min_{\substack{1 \leq h \leq K \\ h \neq q \\ 1 \leq i \leq J \\ i \neq m}} (g_{miph}^{k} + c_{ijq} + c_{ihq}^{T}) \qquad \begin{array}{l} j = 1,\ldots, J;\ j \neq m \\ p = 1,\ldots, K \\ q = 1,\ldots, K \end{array}$$

where initial conditions are

$$g^{o}_{mjpp} = c_{mjp} \qquad \begin{matrix} j = 1,\ldots, J;\ j \neq m \\ p = 1,\ldots, K. \end{matrix}$$

$$g^{o}_{mjpq} = \infty \qquad \begin{matrix} j = 1,\ldots, J;\ j \neq m \\ p = 1,\ldots, K \\ q = 1,\ldots, K;\ q \neq p \end{matrix}$$

then

$$f^{k+1}_{mj} = \min_{\substack{1 \leq p \leq K \\ 1 \leq q \leq K}} g^{k+1}_{mjpq} \qquad j = 1,\ldots, J;\ j \neq m.$$

As in the single-mode case, this procedure terminates when $g^{k+1}_{mjpq} = g^{k}_{mjpq}$ for $j = 1,\ldots, J$; $p = 1,\ldots, K$; $q = 1,\ldots, K$; and $j \neq m$. If all c_{ijq} and c^{T}_{ihq} are nonnegative, the procedure will always terminate in J-2 or fewer iterations since all nodes could possibly be considered as intermediate nodes in that iteration. Again, the all-pairs problem is solved by repeating the algorithm J times for $m = 1,\ldots, J$.

That this algorithm results in optimality is obvious after a little thought and, since the proof is similar to that for the single-mode Bellman-Ford procedure, only a brief motivation will be given. That is, assuming optimal g^{k}_{mjpq} have been found for $j = 1,\ldots, J$ $(j \neq m)$; $p = 1,\ldots, K$; and $q = 1,\ldots, K$ the g^{k+1}_{mjpq} computed in (8) must be optimal when negative loops do not exist because (i) the optimal mode-route with k+1 transfers must reach the last transfer point in an optimal way with k or fewer transfers and (ii) all such possibilities originating at node m are considered in (8). Obviously, the algorithm can terminate in any iteration where all $g^{k+1}_{mjpq} = g^{k}_{mjpq}$ since only mode-routes with k or fewer transfers need be considered in attempting to add an additional transfer possibility in the k+2 (and succeeding) iteration(s); but this possibility is already considered in iteration k+1.

A Pseudo Bellman-Ford Procedure

A slight modification of the above procedure will also prove to be interesting for some of the particular kinds of problems this paper addresses. Suppose rather than solving all mode combinations simultaneously as in (8), we solve (8) within each iteration for each mode combination separately. That is, suppose we define

$$g^{k}_{ijp_1p_2\cdots p_{k+1}} = \text{least-cost mode-route connecting nodes i and j with } k \text{ transfers and using in order modes } p_1, p_2, \ldots, p_{k+1},$$

and compute in iterations k = 0, 1, 2,...,

$$(9)\quad g^{k+1}_{mjp_1p_2\cdots p_{k+1}q} = \min_{\substack{1 \le i \le J \\ i \neq m}} \left(g^{k}_{mip_1p_2\cdots p_{k+1}} + c_{ijq} + c^{T}_{ip_{k+2}q}\right)$$

$$\begin{aligned} j &= 1,\ldots, J;\ j \neq m \\ p_1 &= 1,\ldots, K, \\ p_2 &= 1,\ldots, K;\ p_2 \neq p_1,\ \text{5/} \\ &\vdots \\ p_k &= 1,\ldots, K;\ p_{k+1} \neq p_k \\ q &= 1,\ldots, K;\ q \neq p_{k+1}, \end{aligned}$$

where

$$g^{o}_{mjp} = c_{mjp} \qquad \begin{aligned} j &= 1,\ldots, m;\ j \neq m, \\ p &= 1,\ldots, K. \end{aligned}$$

Finally,

$$f^{k+1}_{mj} = \min_{p_1, p_2, \ldots, p_{k+2}} \left(g^{k+1}_{mjp_1p_2\cdots p_{k+2}}\right)$$

$$j = 1,\ldots, J;\ j \neq m.$$

5/ Here we explicitly assume that the input costs c_{ijp} represent the least-costs between all pairs of nodes i and j by mode p. Hence, we do not have to consider consecutive linking of two segments or areas of the same mode.

The optimality of this procedure is readily apparent since it theoretically involves evaluation of all the possibilities considered in (8) as well as others. Of course, the procedure in (8) will, for the same reason, be more efficient in the general case of the problem, but as indicated below, the procedure in (9) may be preferable in certain applications.

A Generalized Floyd Procedure

For the algorithms solving the all-pairs single-mode problem simultaneously, similar generalizations are made possible by considering all possibilities for originating and terminating modes of transportation in each iteration. In the single-mode problem, the Floyd procedure defines

f_{ij}^{k} = least-cost route from node i to node j when only nodes 1,..., k are considered as intermediate points

and solves the problem by computing in iterations k = 0,..., J-1

$$f_{ij}^{k+1} = \min (f_{ij}^{k}, f_{k,k+1}^{k} + f_{k+1,j}^{k}) \qquad \begin{matrix} i = 1,\ldots, J \\ j = 1,\ldots, J;\ j \neq i \end{matrix}$$

where initial conditions are

$$f_{ij}^{o} = c_{ij} \qquad \begin{matrix} i = 1,\ldots, J \\ j = 1,\ldots, J. \end{matrix}$$

We might generalize this procedure by defining

g_{ijpq}^{k} = least-cost mode-route from node i to node j originating on mode p and terminating on mode q when only nodes 1,..., k are considered as possible transfer points

and then computing in iterations k = 0,..., J-1

$$(10) \quad g_{ijpq}^{k+1} = \min \{g_{ijpq}^{k}, \min_{\substack{1 \leq h \leq K \\ 1 \leq n \leq K}} (g_{i,k+1,p,h}^{k} + g_{k+1,j,n,q}^{k} + c_{k+1,h,n}^{T})\}$$

$$\begin{matrix} i = 1,\ldots, J;\ i \neq k+1 \\ j = 1,\ldots, J;\ j \neq i;\ j \neq k+1 \\ p = 1,\ldots, K \\ q = 1,\ldots, K \end{matrix}$$

beginning with initial conditions

$$g^{o}_{ijpp} = c_{ijp} \qquad \begin{array}{l} i = 1,\ldots, J \\ j = 1,\ldots, J;\ j \neq i, \\ p = 1,\ldots, K. \end{array}$$

Finally, one could find

$$f^{k+1}_{ij} = \min_{\substack{1 \leq p \leq K \\ 1 \leq q \leq K}} g^{k+1}_{ijpq} \qquad \begin{array}{l} i = 1,\ldots, J \\ j = 1,\ldots, J;\ j \neq i. \end{array}$$

The optimality of the procedure in (10) is also not difficult to verify. Where we have optimal g^{k}_{ijpq} for $i = 1,\ldots, J$; $j = 1,\ldots, J$ $(j \neq i)$; $p = 1,\ldots, K$; and $q = 1,\ldots, K$, and negative loops do not exist, the optimality of g^{k+1}_{ijpq} is evident since (i) any optimal mode-route from node i to node j with nodes $1,\ldots, k+1$ as possible intermediate points must reach node k+1 from node i in an optimal way and move from node k+1 to node j in an optimal way using only nodes $1,\ldots, k$ as intermediate nodes in both cases, or must move from node i to node j in an optimal way using only nodes $1,\ldots, k$ (not through node k+1), and (ii) all such possibilities are considered in (10). Unfortunately, this algorithm will never terminate until all J-1 iterations are completed (all J nodes are considered as intermediate nodes). Variations in the Floyd algorithm similar to those in (9) are seemingly not possible.

A Generalized Dantzig Procedure

The single-mode Dantzig procedure defines

f^{i}_{ij} = least-cost route connecting nodes i and j where only nodes $1,\ldots, k$ are considered as initial, intermediate or terminal nodes

and computes in iterations $k = 1,\ldots, J-1$

$$f^{k+1}_{i,k+1} = \min_{1 \le j \le k} (f^{k}_{ij} + c_{j,k+1}) \qquad i = 1,\ldots, k$$

$$f^{k+1}_{k+1,i} = \min_{1 \le j \le k} (c_{k+1,j} + f^{k}_{ji}) \qquad i = 1,\ldots, k$$

$$f^{k+1}_{ij} = \min (f^{k}_{ij}, f^{k}_{i,k+1} + f^{k+1}_{k+1,i}) \qquad \begin{array}{l} i = 1,\ldots, k \\ j = 1,\ldots, k;\ j \neq i \end{array}$$

from initial condition

$$f^{1}_{11} = 0$$

This procedure might also be generalized by defining

g^{k}_{kjpq} = least-cost mode-route from node i to node j originating on mode p and terminating on mode q where only nodes 1,..., k are considered as initial, intermediate, or terminal nodes

and computing in iterations k = 1,..., J-1

(11)
$$g^{k+1}_{i,k+1,p,q} = \min_{\substack{1 \le j \le k \\ j \neq i \\ 1 \le h \le K \\ h \neq q}} (g^{k}_{ijph} + c_{j,k+1,q} + c^{T}_{jhq}) \qquad \begin{array}{l} i = 1,\ldots, k, \\ p = 1,\ldots, K, \\ q = 1,\ldots, K, \end{array}$$

$$g^{k+1}_{k+1,i,p,q} = \min_{\substack{1 \le j \le K \\ j \neq i \\ 1 \le h \le K \\ h \neq p}} (c_{k+1,j,p} + f^{k}_{jihq} + c^{T}_{jph}) \qquad \begin{array}{l} i = 1,\ldots, k, \\ p = 1,\ldots, K, \\ q = 1,\ldots, K, \end{array}$$

$$g^{k+1}_{ijpq} = \min \{g^{k}_{ijpq}, \min_{\substack{1 \le h \le K \\ 1 \le n \le K}} (g^{k+1}_{i,k+1,p,h} + g^{k+1}_{k+1,j,n,q} + c^{T}_{k+1,h,n})\} \qquad \begin{array}{l} i = 1,\ldots, k \\ j = 1,\ldots, k;\ j \neq i \\ p = 1,\ldots, K, \\ q = 1,\ldots, K, \end{array}$$

where initial conditions are

$$g^{1}_{11p} = 0 \qquad p = 1,\ldots, K.$$

Here,

$$f_{ij}^{k+1} = \min_{\substack{1 \le p \le K \\ 1 \le q \le K}} g_{ijpq}^{k+1} \qquad \begin{matrix} i = 1,\ldots, k \\ j = 1,\ldots, k;\ j \neq i. \end{matrix}$$

To verify the optimality of this procedure when negative loops do not exist, assume optimal g_{ijpq}^{k} have been found for i = 1,..., k; j = 1,..., k; p = 1,..., K; and q = 1,..., K. The optimality of g_{ijpq}^{k+1} must then follow because (i) any optimal mode-route from node k+1 to node j ($1 \le j \le k$) must move from the first transfer point to node j in an optimal way where only nodes 1,..., k are possible intermediate points, (iii) any optimal mode-route from node i ($1 \le i \le k$) to node j ($1 \le i \le k$) must move from node i to node k+1 in an optimal way and from node k+1 to node j in an optimal way, or must move from node i to node j in an optimal way where only nodes 1,..., k are possible intermediate nodes, and (iv) since all such possibilities are considered in (11). Like the Floyd algorithm, the Dantzig algorithm can never terminate short of J-1 iterations (until all nodes are considered as part of the system).

Comparison and Selection of Procedure

Consider the efficiency of the various algorithms indicated above. The maximum number of computations (additions and comparisons) required by the single-mode algorithms referenced here are discussed in Dreyfus [5] and are given in Table 1. We can similarly deduce the number of computations required for the K-mode algorithms in (8), (10), and (11) since in each step the number of comparisons required to find f_{ijpq}^{k+1} has simply been increased by a power of K and K-1 and the number of additions is additionally doubled by considering transfer costs. For each iteration (8) one can easily verify that $2(J-1)^2K^2(K-1)$ additions and $(J-1)^2K^2(K-1)$ comparisons

must be made in each iteration.[6/] Hence, if all J-2 iterations are required, the entries in Table 2 pertaining to the generalized Bellman-Ford procedure are obtained. For each iteration (10) we find $2(J-1)(J-2)K^4$ additions and $(J-1)(J-2)K^4$ comparisons must be made. Thus, since J iterations are always required, the information in Table 2 pertaining to the generalized Floyd procedure is obtained. For each iteration (11) we find the first and second equations require $3k(k-1)K^3$ computations (additions plus comparisons) each and the third requires $3k(k-1)K^4$ computations. Where k iterates from 1 to J-1 we then find

$$\sum_{k=1}^{J-1} 3k(k-1)(K^3 + K^3 + K^4) = J(J-1)(J-2)K^3(K+2)$$

computations are required by the generalized Dantzig procedure. For the procedure in (9) one can quickly verify that at most (if J-2 iterations are required)

$$2\sum_{k=0}^{J-2} (J-1)^2K(K-1)^{k+1} = 2(J-1)^2K(K-1)[(K-1)^{J-1} -1]/(K-2) \text{ (if } K > 2)$$

additions and

$$\sum_{k=0}^{J-2} (J-1)^2K(K-1)^{k+1} = (J-1)^2K(K-1)[(K-1)^{J-1} -1]/(K-2) \text{ (if } K > 2)$$

comparisons are required. Hence, the results in Table 2 are obtained.

[6/] Here we assume, as we will in the following cases, that all possible single-mode routes are used as data so that linking segments of the same mode need not be considered. Of course, finding all possible single-mode routes may involve a considerable problem in and of itself. The algorithms here could be used to solve both problems simultaneously if all K modes were always considered in each loop. If the algorithms presented here were so modified, then all K-1 terms in the formulas for number of computations should be replaced by K's.

Table 1. Maximum Number of Computations Involved in Single-Mode Procedures

Procedure		Dijkstra[7]	Bellman-Ford	Floyd	Dantzig
From node m to	Additions	$J(J-1)/2$	$(J-2)(J-1)^2$	--------	--------
all other nodes	Comparisons	$J(J-1)$	$(J-2)(J-1)^2$	--------	--------
For all pairs	Additions	$J^2(J-1)/2$	$J(J-2)(J-1)^2$	$J(J-1)(J-2)$	$J(J-1)(J-2)$
of nodes	Comparisons	$J^2(J-1)$	$J(J-1)(J-2)$	$J(J-1)(J-2)$	$J(J-1)(J-2)$

[7]As Dreyfus [5] indicates, computer application of the Dijkstra algorithm actually requires about $J^2/2$ additions and $2J^2$ comparisons in the node-m-to-all-other-nodes problem and, hence, $J^3/2$ additions and $2J^3$ comparisons for the all-pairs problem.

Table 2. Maximum Number of Computations Involved in K-Mode Procedures

Procedure		Generalized Bellman-Ford[8]	Pseudo Bellman-Ford	Generalized Floyd[9]	Generalized Dantzig
Recursion Equations		(8)	(9)	(10)	(11)
From node m to	Additions	$2(J-2)(J-1)^2K(K-1)^2$	$2(J-1)^2K(K-1)[(K-1)^{J-1}-1]/(K-2)$	--------	--------
all other nodes	Comparisons	$(J-2)(J-1)^2K(K-1)^2$	$(J-1)^2K(K-1)[(K-1)^{J-1}-1]/(K-2)$	--------	--------
For all pairs	Additions	$2J(J-2)(J-1)^2K(K-1)^2$	$2(J-1)^2K(K-1)[(K-1)^{J-1}-1]/(K-2)$	$2J(J-1)(J-2)K^4$	$2J(J-1)(J-2)K^3(K+2)/3$
of nodes	Comparisons	$J(J-2)(J-2)^2K(K-1)^2$	$J(J-1)^2K(K-1)[(K-1)^{J-1}-1]/(K-2)$	$J(J-1)(J-2)K^4$	$J(J-1)(J-2)K^3(K+2)/3$

[8]For problems requiring few iterations, it is worthy to note that $2(J-1)^2K(K-1)^2$ additions and $(J-1)^2K(K-1)^2$ comparisons can be avoided in the first iteration for each m since the incoming mode to the intermediate point must be the mode of origination. Also $2(J-1)^2K(K-1)$ additions and $(J-1)^2K(K-1)$ comparisons can be avoided in the second iteration since we need not examine the possibility where the incoming mode to the intermediate node is the same as the mode of origination. In the second iteration these two modes would be used consecutively. Hence, if they are the same, the two segments form a single-mode segment such as is considered already in the first iteration.

[9]Here also some calculations can be omitted in the first iteration because the incoming mode to the intermediate point must be the mode of origination but savings will always be negligible since the Floyd procedure can never stop short of J-1 iterations.

One can quickly note that efficiency of the various methods may not be the same in the K-mode case as in the single-mode case. Both the Floyd and Dantzig procedures involve a K^4 factor of computations so that the Bellman-Ford procedures become more efficient for large K and small J. Of course, the common case, however, would probably involve small K and large J. In this case, Table 2 may be somewhat misleading since both of the Bellman-Ford procedures can terminate after only a few iterations, while the Floyd and Dantzig procedures must complete all iterations. Indeed, when transfer costs are of significance, this would likely be the case. Another factor of considerable importance is that for the transportation problem considered here, we are not interested in least-cost mode-routes between all pairs of nodes--only those mode-routes between deficit and surplus nodes. Hence, where M is the number of surplus nodes and N is the number of deficit nodes, we need to carry out the calculations in the first two lines of Table 2 only M times rather than J times as indicated for the all-pairs problem. Of course, the Floyd and Dantzig procedures must always be performed for the entire all-pairs problem. Finally, by noting that the problem could be solved in reverse order, it is apparent that we could alternatively make the calculations in the first two lines of Table 2 only N times if $N < M$. Hence, since $M + N = J$, we must repeat the calculations in the first part of Table 2 at most $J/2$ times even if the algorithm does not terminate until the full J-2 iterations have been made.

To investigate the possibilities when accumulation of transfer costs prohibits many transfers in any mode-route, Table 3 has been constructed for several values of J and K assuming the Bellman-Ford procedures terminate in 2, 4, or 6 iterations. In each case, the number of calculations is

Table 3. Number of Computations (Additions plus Comparisons) Required to Develop a Transportation Matrix with Various K-Mode Dynamic Programming Procedures[10]

Number of Iterations[11]	Number of Nodes(J)	Number of Modes(K)	Generalized Bellman-Ford[12]	Pseudo Bellman-Ford	Generalized Floyd	Generalized Dantzig
				(100,000's)		
2	100	2	12*	12*	47	31
		3	53*	53*	236	131
		4	141*	141*	745	373
	1000	2	11,976*	11,976*	47,856	31,904
		3	53,892*	53,892*	242,271	134,595
		4	143,712*	143,712*	765,698	382,849
3	100	2	24*	24*	47	31
		3	105*	123	236	131
		4	283*	459	745	373
	1000	2	23,952*	23,952*	47,856	31,904
		3	107,784*	125,748	242,271	134,595
		4	287,424*	467,064	765,698	382,849
4	100	2	35	35	47	31*
		3	158	265	236	131*
		4	424	1,411	745	373*
	1000	2	35,928	35,928	47,856	31,904*
		3	161,676	269,460	242,271	134,595*
		4	431,136	1,437,121	765,698	382,849*

[10]Efficient algorithms are marked by an asterick (*) in each case.
[11]Here the number of iterations which applies only to the Bellman-Ford procedures, is the final value of k plus one.
[12]Entries for the Generalized Bellman-Ford procedure are computed assuming that all unnecessary computations in the first two iterations are not made.

determined by assuming M = N = J/2 (the situation which makes the Bellman-Ford procedures relatively as inefficient as possible).

As indicated in Table 3, any of the generalized algorithms except the generalized Floyd procedure can possibly become efficient depending on the number of nodes, modes, and iterations required by the Bellman-Ford procedures. Apparently, the generalized Dantzig procedure quickly becomes the more efficient procedure as the number of required iterations increases when the number of modes remains relatively small. When the number of iterations becomes small, then both the Bellman-Ford procedures tend to become efficient. As indicated above, this would also be the case when the number of modes increases.

Conclusions

Several other points may be of interest in the selection of a K-mode procedure. Perhaps the most important consideration may be the significance of economic information contained in the results. In many economic applications of the K-mode transportation problem, the major goal may be an evaluation of the use of a particular mode along a particular route segment. Such information would be of major interest in determining the feasibility of building a new highway or barge canal, or of closing a certain rail route. In this case, one might be concerned with not only determining least cost mode-routes between all pairs of points, but also with determining shaddow prices for non-optimal mode-routes which use the mode-segment of interest. One can easily see that the pseudo Bellman-Ford procedure is the only generalized procedure presented here that will automatically provide additional shaddow price information.

Another factor which may be important in the practical application of these procedures is the size of the problem relative to the size of the computer which is to be used to obtain the results. As the number of nodes becomes large, the size of the cost matrix connecting all pairs of nodes by even a single mode can quickly approach the upper limit for core storage available on most large computers. When core storage is exceeded and auxiliary storage must be used, the cost of obtaining computer results begins to depend greatly on the frequency of reference to auxiliary units. In that respect, the Bellman-Ford procedures appear to be somewhat easier to handle.

For example, with the pseudo Bellman-Ford procedure, each iteration can be broken into steps so that only $g^{k}_{mip_1p_2\cdots p_{k+1}}$, c_{ijq}, and $c^{T}_{ip_{k+1}q}$ for for $i = 1,\ldots, J$ must be held in core at one time. In each iteration, this material would be obtained only once from auxiliary storage (in a large problem situation) for each $j = 1,\ldots, J$; $p_1 = 1,\ldots, K;\ldots; p_k = 1,\ldots, K$; and $q = 1,\ldots, K$. The generalized Bellman-Ford procedure would apparently require K times as much core storage unless an additional vector were set up to save partial minimums. By saving minimums over $i = 1,\ldots, J$ for $h = 1,\ldots, K$ it is a simple matter to find the minimum specified in (8) by finding the minimum if the K values saved for $h = 1,\ldots, K$. This suggests a simple way in which a single computer program could be used for either the generalized or pseudo Bellman-Ford procedure. The minimization over h at each iteration is the only difference in the two procedures. If one desires shaddow prices on specific mode-segments then one need not minimize over h in each iteration, but if shaddow prices are not needed then considerable savings result (if more than a few iterations are required) when one minimizes over h in each iteration according to the generalized Bellman-Ford procedure.

Apparently, the Floyd and Dantzig procedures cannot be handled quite so easily in the large problem case. Either much more core storage must be used to store the entire matrix f^k_{ijpq} at each iteration or more auxiliary unit references must be made (due to the varying order of reference to f^k_{ijpq}).

References

[1] Bellman, R. E. "On a Routing Problem," Quart. Appl. Math. 16, 89-90 (1958).

[2] Dantzig, G. B. All Shortest Routes in a Graph, Operations Research House, Stanford University, Technical Report 66-3, November, 1966; also in Theorie des Graphes, Proceedings of the International Symposium, Rome, Italy, July, 1966, published by Dunod, Paris.

[3] _____. Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

[4] Dijkstra, E. W. "A Note on Two Problems in Connexion with Graphs," Numerishe Mathematik 1, 269-271 (1959).

[5] Dreyfus, S. E. "An Appraisal of Some Shortest-Path Algorithms," Operations Research 17, 395-412 (1969).

[6] Floyd, R. W. "Algorithm 97, Shortest Path," Comm. ACM 5, 345 (1962).

[7] Ford, L. R., Jr. Network Flow Theory, The Rand Corporation, P-923, August, 1956.


AGRICULTURAL
EXPERIMENT STATION
OKLAHOMA STATE UNIVERSITY

0274/1.4M