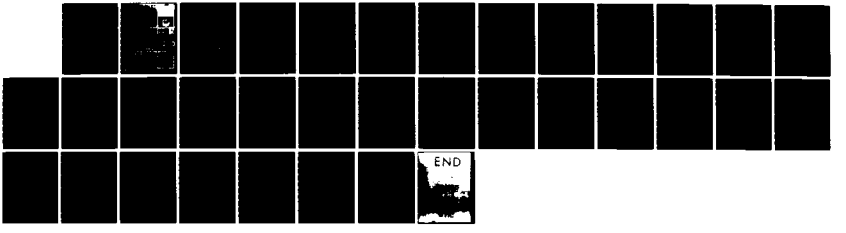
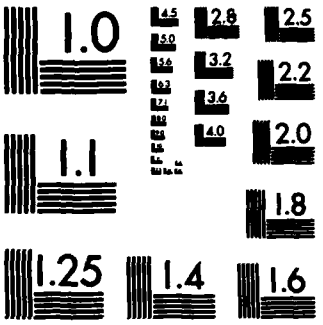
AD-A144 219 SOLVING TRANSPORTATION PROBLEMS VIA AGGREGATION(U) 1/1
GEORGIA INST OF TECH ATLANTA PRODUCTION AND
DISTRIBUTION RESEARCH CENTER R W TAYLOR ET AL. JUL 84
UNCLASSIFIED PDRC-84-10 N00014-83-K-0147 F/G 12/1 NL





SOLVING TRANSPORTATION PROBLEMS
VIA AGGREGATION

by

Richard W. Taylor†
C. M. Shetty††

PDRC 84-10

PRODUCTION and DISTRIBUTION RESEARCH CENTER

SCHOOL OF INDUSTRIAL
AND SYSTEMS
ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM
OF GEORGIA
ATLANTA, GEORGIA 30332

PDRC Report Series 84-10
July 1984

SOLVING TRANSPORTATION PROBLEMS VIA AGGREGATION

by

Richard W. Taylor†
C. M. Shetty††

PDRC 84-10

†Richard W. Taylor
School of Business Administration
University of Evansville
Evansville, Indiana 47702

††School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

This work was supported in part by the Office of Naval Research under Contract No. N00014-83-K-0147. Reproduction is permitted in whole or in part for any purpose of the U. S. Government.

Abstract

Solving Transportation Problems Via Aggregation

Richard W. Taylor
School of Business Administration
University of Evansville
Evansville, IN 47702

C. M. Shetty
The School of Industrial and Systems Engineering
The Georgia Institute of Technology
Atlanta, Georgia 30332

This paper deals with the solution of large scale transportation problems by aggregation as a special case of constraint and variable aggregation in linear programming. Using cluster analysis, an appropriate form of aggregation will be used. If the bounds developed at this point are not tight enough, a suitable unaggregation step will be specified. Computational results are presented.

Key Words: Aggregation methods, transportation problems, large scale problems

1. Introduction

This paper deals with a practical strategy for solving large transportation problems. Even with the present day computers, transportation problems can be encountered that are too large to solve practicably. Further, in many instances a more cost-effective standpoint might be to seek only approximate solutions to these problems. Currently strategies are available to solve large problems by taking advantage of the special structure of transportation problems. The proposed algorithm utilizes these strategies while also providing a means of reducing the total problem size. The application of aggregate programming to the transportation problem seems a natural one as it is intuitively appealing to aggregate sources or destinations that have other similar characteristics.

The main thrust of the proposed strategy is the aggregation of constraints and/or variables to form a related transportation problem. This problem is used as a starting point for a sequence of problems, each more restrictive than the preceding. Bounds on the optimal solution of the original problem at each stage are also computed. At each stage, we address the question of which constraints/variables we would like to aggregate/unaggregate in order to expedite moving towards an acceptable solution.

The general approach of aggregate programming has been widely applied in many areas of practical research, such as economics [7], input/output analysis [1,3,6,8], and production planning [4]. See [14] for an extended survey and references. Unfortunately, little theoretical insight is available in these areas.

2. Related Work

The classical transportation problem can be stated as

$$(TP) \qquad \text{Min} \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{N} x_{ij} = a_i \qquad i=1,2,\ldots,M \tag{2}$$

$$\sum_{i=1}^{M} x_{ij} = b_j \qquad j=1,2,\ldots,N \tag{3}$$

$$x_{ij} \geq 0 \qquad \text{for each } i,j$$

where without loss of generality, we assume that $\sum_{i=1}^{M} a_i = \sum_{j=1}^{N} b_j$. Let x^* be the optimal solution with z^* the optimal value at this point.

An important aspect to consider when solving transportation problems by aggregate programming is that the aggregate programs also be transportation problems so as to take advantage of the special structure in transportation problems.

We will first consider the case of aggregating destinations (see Figure 1). Later we will consider the consequences of concurrent aggregation of sources and destinations.

The general areas of study in aggregate programming as applied to the transportation problem can be separated into three sections:

1) Selection of the destinations to aggregate,

2) Derivation of bound information on x^* and z^* based on the

solution of the aggregate transportation problem, and finally

3) Reformulation of the aggregate program if 2) does not provide an adequate solution.

For brevity, we hereon refer to the above areas as aggregate development, model adequacy, and problem reformulation.

In the literature, there has been little work done on developing the initial aggregate program. Most of the research assumes some prior knowledge of how the problem is to be partitioned. They assume that the destinations to be combined are known. Furthermore, frequently the actual aggregations are made with equal weightings. Balas [2] proposes that aggregation may be applied to all origins (destinations) that are "neighbouring" based on the values of the c_{ij}'s. The definition of neighbouring is intentionally left vague as his emphasis is on the scale of aggregation. In this respect, he concludes with an intuitively argued best initial aggregate program size. Although his suggestion is more appropriate for his particular algorithm, his thoughts on certain tradeoffs is applicable to the level of aggregation in any scenario.

Zipkin [20] and Geoffrion [9] both suggest that the area of cluster analysis might be useful in developing the initial aggregate program. In this respect, we will adopt specific clustering techniques that are both efficient and adequate for our purposes.

Now suppose the aggregate program has been developed and solved to yield a solution point x^a with associated value z^a. Zipkin [20] and Geoffrion [9] have developed upper and lower bounds for the original problem solution value z^* using the aggregate program solution. These bounds will be slightly extended for use in our proposed procedure.

Further, a theoretical groundwork will be laid for further tightening of these bounds. This extension will utilize a method of marginal analysis similarly used by Mendelssohn [17] and Kallio [12].

Finally, a recent paper by Huberman [10] has captured some of the above bounds in a single cohesive format. Also adaptations of some of the earlier bounds have been made to the multicommodity network and generalized transportation problem by Evans [5].

In many instances, the above described bounds are not adequate and a new aggregate problem is formulated in an attempt to attain tighter bounds. Zipkin [20] suggests a possible method whereby aggregate terms are separated into two new aggregate terms in the subsequent problem. Our proposal will present a variation on this suggestion that has provided lower computation times in preliminary testing.

Finally, Balas [2] provides a method for aggregating transportation models and then successively solving these modified problems as a solution procedure. His emphasis is placed on achieving optimal solutions and as our procedure will suggest, a tradeoff exists between increased accuracy of the solution and computation time. No computation times are given by Balas. Lee [15] extended Balas' work to the general minimum cost network flow problem. These methods, as well as the general relaxation solution strategies, are used in part in this study.

3. Development of the Procedure

For the moment we will only consider the case of aggregation of destinations; later we will address the issue of concurrent source and destination aggregation. Recall that the proposed approach consists of

1) aggregate problem development; 2) model adequacy; and 3) problem reformulation. These are discussed in detail below.

3.1 Development of the Initial Aggregate Program

To develop an initial aggregate transportation problem we must decide first, which destinations to aggregate and secondly, with what weightages they should be aggregated. As in Figure 1, let us first consider a general aggregation of original destinations $j \in P_r$ into a single aggregate destination r. This corresponds to aggregating each of the demands b_j for $j \in P_r$ into a demand b_r. Additionally, the flows from each of the sources into these destinations can be aggregated for all $j \in P_r$. In particular, aggregation of the demands b_j corresponds to the aggregation of the constraints in Eqn. (3) for $j \in P_r$ with multipliers $t_j = 1, 2, \ldots, N$. Aggregation of the flows from each of the sources into the destinations $j \in P_r$ corresponds to the aggregation of M sets of variables x_{ij}, $j \in P_r$ with multipliers s_{ij}. These constraint and variable aggregations can be performed for all sets P_r, $r=1,2,\ldots,L$ to give a general aggregated transportation problem (TP-CV) as follows:

$$\text{(TP-CV)} \qquad \text{Min} \sum_{i=1}^{M} \sum_{r=1}^{L} \sum_{j \in P_r} s_{ij} c_{ij} x_{ir} \tag{4}$$

$$\text{s.t.} \qquad \sum_{r=1}^{L} \sum_{j \in P_r} s_{ij} x_{ir} = a_i \qquad i=1,2,\ldots,M \tag{5}$$

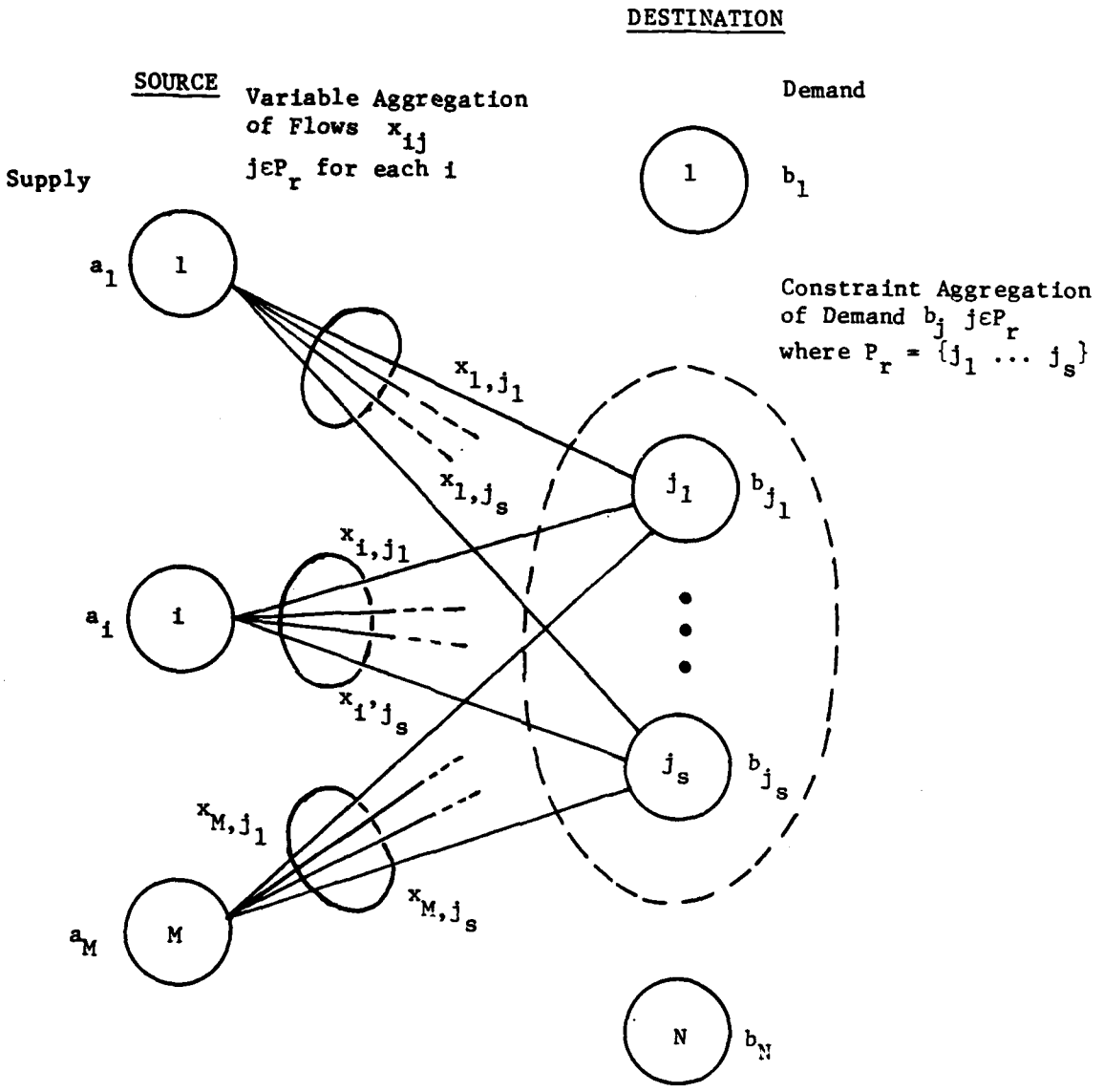


Figure 1

Aggregation of Destinations

$$\sum_{j \in P_r} t_j \sum_{i=1}^{M} s_{ij} x_{ir} = \sum_{j \in P_r} t_j b_j \quad r=1,2,\ldots,L \tag{6}$$

$$s_{ij}\, x_{ir} \geq 0 \quad \text{for each } i,j,r \tag{7}$$

As stated earlier, it is desirable that the aggregate program be a transportation problem. Let us view what conditions are required to put the program (TP-CV) in the form of an M-source, L-destination transportation problem as shown below:

$$\text{(ATP)} \qquad z^a = \text{Min} \sum_{i=1}^{M} \sum_{r=1}^{L} c_{ir} x_{ir} \tag{8}$$

$$\text{s.t.} \quad \sum_{r=1}^{L} x_{ir} = a_i \quad i=1,2,\ldots,M \tag{9}$$

$$\sum_{i=1}^{M} x_{ir} = b_r \quad r=1,2,\ldots,L \tag{10}$$

$$x_{ir} \geq 0 \quad \text{for each } i,r \tag{11}$$

Comparing Eqn. 5 with Eqn. 9 and Eqn. 7 with Eqn. 11, we must have

$$\sum_{j \in P_r} s_{ij} = 1 \quad \text{for each } i \; ; \quad s_{ij} \geq 0 \quad \text{for each } i,j \tag{12}$$

Also from Eqn. 6 and Eqn. 10, and setting the aggregate demand b_r equal to the sum of the demands, we have

$$\sum_{j \in P_r} t_j b_j = \sum_{j \in P_r} b_j = b_r \quad \text{for each } r \tag{13}$$

Again from Eqn. 6 and Eqn. 10 we will need

$$\sum_{j \in P_r} t_j s_{ij} = 1 \quad \text{for each } i,r \tag{14}$$

If the conditions of Eqns. 12, 13, and 14 are met then the program (TP-CV) will be a transportation problem. It is easy to see that there are an infinite number of multipliers t_j and s_{ij} that will satisfy these requirements. For example, let $t_j=1$ for each j and then any convex combination of the s_{ij}'s $j \in P_r$ will work. Later in this section we will specifically address the issue of the multipliers to be used in our algorithm. We first address which destinations should be clustered together.

It should be apparent that if we are to preserve the specialized structure, then aggregation of the demand constraints will also define an aggregation of the flows. Thus if we aggregate demand constraints for $j \in P_r$, we must similarly aggregate the variables x_{ij}, $j \in P_r$ for each i. Conversely, the partitioning of the flows will also determine the partitions for the demand constraints. In this light, we can discuss either the aggregation of the primal demand constraints or the dual flow constraints (corresponding to the primal variables).

To determine the constraints to aggregate, a previus study [18] suggested the angle between the normals as a primary measure of "closeness" of constraints, and the closeness of the right-hand-side values as a secondary measure. In the case of the transportation

problem, it is not difficult to show that in both the primal and the dual problems, the angles between all pairs of constraints are close. Hence, for a given pair of destinations q and k, we want to find the closeness of the vectors c_q and c_k which are the right-hand-sides of the dual constraints. However comparing this for all pairs can be computationally inefficient. Hence, we will compare each vector with an "average vector w defined by

$$w = \frac{1}{N} (c_1 + c_2 + \ldots + c_N)$$

by computing the angle between c_j and w given by:

$$h_j = \cos^{-1} \frac{c_j \cdot w}{\|c_j\| \|w\|}$$

Then if L partitions (clusters) are desired, it seems desirable to define angles $0 \leq e_1 \leq e_2 \leq , \ldots, \leq e_{L-1} \leq 180$, and let

$$\text{Cluster } P_1 = (j: \ 0 \leq h_j \leq e_1)$$

$$\text{Cluster } P_2 = (j: \ e_1 < h_j \leq e_2)$$

$$\vdots$$

$$\text{Cluster } P_L = (j: \ e_{L-1} < h_j \leq 180)$$

Of course in the actual partitioning procedure, one does not need to perform either the $\cos^{-1}(\cdot)$ operation or the common division by $\|w\|$, as long as the appropriate modifications are made to the e_i's. This method also has the advantage that it is computationally efficient.

A final topic in the partitioning of the destinations is the determination of the number of and the values of the e_i's. The e_i's are chosen to uniformly partition the interval [0,180]. As will be more evident in the reformulation process, it makes sense to require that a minimum number of destinations be contained in each cluster. For our purposes, the minimum number was chosen as three destinations (sources) per cluster which allows for at least two iterations of the reformulation process.

We developed above a procedure to partition the destinations into L clusters, P_r $r=1,2,\ldots,L$, based on variable aggregation of the flows. In order to retain the specialized structure of the transportation problem, the above variable aggregation forces a specific constraint aggregation of the demand constraints as discussed earlier. Thus the clusters $j\in P_r$ $r=1,2,\ldots,L$, will also be used to give L aggregate constraints of the form:

$$\sum_{i=1}^{M} x_{ir} = \sum_{j\in P_r} b_j \qquad r=1,2,\ldots,L$$

The previous section gives the procedure for the partitioning of the destinations. The next step in our algorithm is to develop the multipliers for combining the associated constraints and variables. Comparing Eqn. 4 with Eqn. 8, to make the total costs equal, we must have

$$c_{ir} = \sum_{j\in P_r} s_{ij}\, c_{ij}$$

Further if $t_j = 1$, then $x_{ir} = \sum_{k \in P_r} x_{ik}$. To make the costs associated with x_{ir} equal to the sum of the costs associated with the aggregated flows, we must have $c_{ir}x_{ir} = \sum_{j \in P_r} c_{ij}x_{ij}$ so that

$$c_{ir} = \frac{\sum_{j \in P_r} c_{ij}x_{ij}}{\sum_{k \in P_r} x_{ik}}$$

Comparing the expressions for c_{ir} we have

$$s_{ij} = \frac{x_{ij}}{\sum_{k \in P_r} x_{ik}}$$

Since the flows x_{ij} are not available, it seems reasonable to assume that the flows to the destinations aggregated are likely to be proportional to the demand so that a good approximation for s_{ij} will be

$$\hat{s}_{ij} = \frac{b_j}{\sum_{k \in P_r} b_k} \quad \text{for each } j \in P_r \text{ and for each } i,r \qquad (15)$$

Note that $t_j=1$ and $\hat{s}_{ij}$ defined above satisfy the conditions necessary for insuring that the aggregate program is a transportation problem. As we shall see in the section on bounds, use of these multipliers will also provide us with an immediate upper bound on z^*. Incidentally, these multipliers are precisely the pro-rata demand multipliers used by Zipkin [20] and Geoffrion [9].

3.2 Model Adequacy: Bounds

The previous section develops a procedure to form an initial aggregate transportation problem. In this section, we develop bounds on the original problem solution from the solution to the aggregate problem.

Consider the initial aggregate problem (ATP). The problem can be solved to obtain a solution x^a with solution value z^a and optimal dual multipliers (u^a, v^a). Zipkin shows that the solution value z^a forms an upper bound on z^*.

Theorem 1 [Zipkin]: Let x^* solve (TP) with value z^*. Let (ATP) be the aggregate program of (TP) formed by the aggregate multipliers s_{ij} in Eqn. 15 and $t_j=1$ for all j. Let x^a solve (ATP) with value z^a. Then

$$z^* \leq z^a$$

In order to obtain feasible points to (TP) it is necessary to disaggregate the solution x^a (size MLx1) to a vector of size MNx1. Two methods of disaggregation are discussed in [19]. The first and simplest to use is fixed weight disaggregation. We can obtain a feasible solution x^f to (TP) by disaggregating $x^a = [x^a_{ir}]$ as follows:

$$x^f_{ij} = \hat{s}_{ij} x^a_{ir} \quad \text{for each } j \in P_r, \text{ for each r and for each i}$$

where $\hat{s}_{ij}$ are the pro-rata demand multipliers from Eqn. 15. Hence an upper bound on z^* will be as follows:

$$z^* \leq z^f = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij}^f$$

An important issue to note is that a feasible solution x^f can be found by this method regardless of the multipliers s_{ij} used to form the initial aggregate program. A consequence of using $\hat{s}_{ij}$ to form the initial aggregate program is that $z^a = z^f$.

A second alternative is to use the method of optimal disaggregation as discussed by Balas [2] and Zipkin [19]. This method entails solving a transportation problem for each cluster P_r $r=1,2,\ldots,L$ in order to produce optimal flows within each cluster. Thus solving the following programs for each $r=1,2,\ldots,L$:

$$z^r = \text{Min} \sum_{i=1}^{M} \sum_{j \in P_r} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in P_r} x_{ij} = x_{ir}^a \quad i=1,2,\ldots,M$$

$$\sum_{i=1}^{M} x_{ij} = b_j \quad \text{for each } j \in P_r$$

$$x_{ij} \geq 0 \quad \text{for each } i, \text{ for each } j \in P_r$$

we will get a feasible solution to (TP) with solution value $z^f = \sum_{r=1}^{L} z^r$. Zipkin also shows that the upper bound formed by the optimal disaggregation procedure is at least as good as the fixed weight disaggregated solution value. The optimal disaggregation procedure does

however require considerably more computation time, as a transportation problem must be solved for each aggregate cluster. These two alternatives were computationally tested to determine the conditions which favours each method. Preliminary results indicate that the optimal disaggregation method is most effective at the later stages when the bound obtained is close to the desired accuracy. In these cases, the expense of the additional computation time of solving these sub-transportation problems is offset by achieving sufficient additional accuracy.

We now proceed to develop a lower bound on z^*. Let (u,v) be a set of dual variables with components

$$u_i = u_i^a \quad \text{and} \quad \hat{v}_j = v_r^a \quad \text{for } j \in P_r \text{ and } r = 1,2,\ldots,L.$$

when u_i^a and v_r^a are the optimal dual multipliers to (ATP). Let $(d,\hat{d})$ be an arbitrary direction. Then

$$z^* = \sum_{j=1}^{N} (v_j+\theta\hat{d}_j)b_j + \sum_{i=1}^{M} (u_i+\theta d_i)a_i - \sum_{j=1}^{N}\sum_{i=1}^{M} [(\hat{v}_j+\theta\hat{d}_j) + (u_i+\theta d_i) - c_i]x_{ij}^*$$

The first two terms are the dual objective function value at $(u,\hat{v}) + \theta(d,\hat{d})$ and the last term is a penalty for violating the dual constraints. Noting $b_r = \sum_{k \in P_r} b_k$, $z^a = \sum_{j=1}^{N} \hat{v}_j b_j + \sum_{i=1}^{M} u_i a_i$, and $\sum_{i=1}^{M} x_{ij}^* = b_j$, we have

$$z^* = z^a + \theta\left(\sum_{j=1}^{N} b_j\hat{d}_j + \sum_{i=1}^{M} a_i d_i\right) - \sum_{j=1}^{N}\sum_{i=1}^{M} [(\hat{v}_j + \theta\hat{d}_j)$$

$$+ (u_i^a + \theta d_i) - c_{ij}]x_{ij}^*$$

$$> z^a + \theta\left(\sum_{j=1}^{N} b_j\hat{d}_j + \sum_{i=1}^{M} a_i d_i\right) - \sum_{j=1}^{N} \max_i[(\hat{v}_j + \theta\hat{d}_j)$$

$$+ (u_i^a + \theta d_i) - c_{ij}]b_j$$

Zipkin [20] derives the same bounds using a relaxation procedure when $\theta = 0$.

However for each destination j, we are allocating the full demand $\sum_{i=1}^{M} x_{ij}^* = b_j$ to the particular term corresponding to $\max_i [(\hat{v}_j + \theta\hat{d}_j) + (u_i^a + \theta d_i) - c_{ij}]$. If however we reallocate the total amount b_j to specific indices i so that an allocation for any cell (i,j) will be no greater than min (a_i, b_j) then a better bound on z^* is available. Thus if $[(\hat{v}_j + \theta\hat{d}_j) + (u_i^a + \theta d_i) - c_{ij}]$ are in descending order, we have

$$z^* > z^a + \theta\left(\sum_{j=1}^{N} b_j\hat{d}_j + \sum_{i=1}^{M} a_i d_i\right) - \sum_{i=1}^{M}\sum_{j=1}^{N} [(\hat{v}_j + \theta\hat{d}_j)$$
$$+ (u_i^a + \theta d_i) - c_{ij}] \max\{0, \min[a_i, b_j - \sum_{s=1}^{i-1} a_s]\}$$

Hereinafter the right hand side of the above expression will be denoted as $z(\theta)$ which is stronger than the bound derived earlier. Both methods of bounding were incorporated into the full algorithm and it was found that both methods required less than 1% of the total time to solve

the problem. However in a preliminary comparison of the accuracy of the resulting bounds, it was found that out of 40 observations, the proposed lower bound was on the average 16.8% (standard deviation 4.3%) better than Zipkin's bounds.

The above discussion clarifies the lower bound to be used in our proposed algorithm is stated precisely in Theorem 2 below.

Theorem 2 Let x^* solve (TP) with value z^*. Let (ATP) be the aggregate program of (TP). Let x^a solve (ATP) with value z^a and optimal dual multipliers (u^a, v^a). Let $\hat{v} = v^a_r$ for each $j \in P_r$ for each r. Let the terms $[(\hat{v}_j + \theta\hat{d}_j) + (u^a_i + \theta d_i) - c_{ij}]$ be reordered over the index i to form a nonincreasing sequence. Let

$$z(\theta) = z^a + \theta\Big(\sum_{j=1}^{N} b_j\hat{d}_j + \sum_{i=1}^{M} a_i d_i\Big) - \sum_{i=1}^{M}\sum_{j=1}^{N} [(\hat{v}_j + \theta d_j) + (u^a_i + \theta d_i) - c_{ij}] \max\{0, \min[a_i, b_j - \sum_{s=1}^{i-1} a_s]\}$$

Let $\theta^* = \text{argmax } z(\theta)$. Then $z(\theta^*) \leq z^*$

Theorem 2 above gives a general lower bound. However, in order to use the bound it will be necessary to specify a direction of search and a line search method. First let us look at the function $z(\theta)$. As discussed in [17] and [18], the function $z(\theta)$ will be piecewise linear and convex with breakpoints θ_{ij} as defined below

$$\theta_{ij} = \frac{c_{ij} - u_i^a - \hat{v}_j}{d_i + \hat{d}_j} \quad \text{for each } i,j$$

Note that even for a relatively small transportation problem, there are a large number (MxN) of breakpoints. Clearly it is important that as few of these breakpoints be used as possible in finding θ^*. Certainly the bounds produced by the line search for bounding will be at least as good as when $\theta=0$. However additional computation time will be required for the line search. In order to provide preliminary testing of the algorithm, the computational runs will only use the value $\theta=0$. This assignment also eliminates determination of the specific directions d and $\hat{d}$ at this stage.

Recall that the bound in Theorem 2 was formed by allocating the demand $\sum_{i=1}^{M} x_{ij}^* = b_j$ to particular cells i, i=1,...,M. A similar bound could have been formed by allocating the supply $\sum_{j=1}^{N} x_{ij}^* = a_i$ to particular cells j j=1,...,N. This procedure would yield a lower bound $z'(\theta)$ of

$$z'(\theta) = z^a + \theta\left(\sum_{j=1}^{N} b_j\hat{d}_j + \sum_{i=1}^{M} a_i d_i\right) - \sum_{i=1}^{M}\sum_{j=1}^{N} [(\hat{v}_j + \theta\hat{d}_j)$$

$$+ (u_i^a + \theta d_i) - c_{ij}] \max\{0, \min[b_j, a_i - \sum_{s=1}^{j-1} b_s]\}$$

In the limited problems tested, both bounds were relatively close to each other. As discussed earlier, the time to compute these bounds is a small fraction of the entire solution time hence it seems reasonable to compute both of these bounds at each iteration. Then the maximum of these two bounds can be used as the best lower bound. Incidentally, Zipkin [20] also shows these parallel sets of bounds for his particular bound formulation. His testing also does not provide any insight into the conditions of when one approach provides a better bound than the alternative. We should note that these same bounds are also derived by Geoffrion [9] and Huberman [10], although by a different approach.

By the combination of Theorems 1 and 2 we have the following bounds on the optimal solution of the original problem

$$z^P \leq z^* \leq z^a$$

where

$$z^P = \max\{z(0),\ z'(0)\}$$

If the bounds above are adequate then we can stop. If not, then it is desirable to form a new aggregate problem that will provide tighter bounds than those described above. For this purpose we define in general the lower bound as z_r and the upper bound as z_u, where at this stage of the algorithm $z_r = z^P$ and $z_u = z^a$. It is convenient to measure the error at this stage by an error bound level defined by

$$\frac{z_u - z_r}{z_r}$$

Reformulation of the Aggregate Problem

In many instances the bounds provided by the aggregate program may be satisfactory. If not, a method of reformulation is adopted to tighten the available bounds. This process will be accomplished by unaggregating one of the destinations from each aggregate cluster.

Several reformulation methods are discussed in the literature. Zipkin [20] presents a rather broad algorithm that contains numerous possibilities for the regrouping of aggregate nodes. His main suggestion is to divide each constraint cluster into constraints that are violated at the current solution and those constraints that are not violated at the current solution. This alternative is eliminated since in preliminary testing, the size of the aggregate programs and hence the computational effort in solving the subsequent aggregate programs grows more rapidly than the accuracy of the bounds. Zipkin offers additional possibilities where some but not all of the clusters might be reformulated or that some other criteria other than the violation of the constraints might be used.

Balas [2] uses another reformulation algorithm, namely once a cluster has been identified that contains a violated constraint, then all of the destinations in that cluster are unaggregated in the subsequent program. This method has a similar drawback to Zipkin's method in that the size of the aggregate problem quickly aproaches the order of the original problem. However, Balas' method is focused towards solving the original problem to optimality without consideration of any resource constraints that may limit the size problem solvable. An additional point is that Balas' method uses different aggregate methods to obtain his initial aggregate program, hence his reformulation scheme

is partially based on this particular aggregation. We eliminate the strategy of constructing an entirely new initial aggregate program as requiring considerable effort without guaranteeing that the bounds at subsequent stages will prove to be any better than the ones already at hand.

The proposed method is to unaggregate only the "most violated destination" in each cluster, which seems to yield a favorable tradeoff between increase in the aggregate problem size and decrease in the error bound level. To do this, we unaggregate the destination j within each cluster P_r that has the most violated constraint, i.e., the unaggregated destination j_r in cluster P_r is defined as

$$\hat{j}_r \in \{k \in P_r: [u_i^a + \hat{v}_k - c_{ik}] > [u_i^a + \hat{v}_j - c_{ij}] \text{ for each } j \in P_r \text{ and}$$

$$[u_i^a + \hat{v}_j - c_{ij}] > 0) \qquad (16)$$

This method also has the advantage that the values $[u_i^a + \hat{v}_j - c_{ij}]$ are readily available as they represent the levels of violation in the error bound $z(\theta)$. Further, the aggregate multipliers s_{ij} can be easily updated in the subsequent program, where the multipliers in aggregate destination r will be $\hat{s}_{ij}(\frac{b_r}{b_r - b_{j_r}})$ for $j \in P_r$, $j \neq \hat{j}_r$. Thus given an aggregate program, we can construct a scheme that produces a reformulated aggregate program as follows:

1. For each cluster P_r, unaggregate destination $\hat{j}_r$ as defined in Eqn. 16 to form two new clusters $(\hat{j}_r)$ and $(j \in P_r, j \neq \hat{j}_r)$. Do for

all r=1,2,...,L.

2. Reaggregate using the clusters in step 1.

The above procedure describes the method of reformulating the initial aggregate program into a subsequent aggregate programs. The subsequent programs are based on the preceding programs, hence the updating from one program to the next is minimal. Further, at iteration k, we can construct bounds for the program $ATP^{(k)}$ using Theorem 1 to yield an upper bound $z_a^{(k)}$ and Theorem 2 to yield a lower bound $z_p^{(k)}$. Since the program $ATP^{(k+1)}$ is a restriction of the preceding program $ATP^{(k)}$, we have:

Theorem 3 Let $x_a^{(k)}$ solve $ATP^{(k)}$ with value $z_a^{(k)}$. Let $ATP^{(k+1)}$ be the reformulated program of $ATP^{(k)}$. Let $x_a^{(k+1)}$ solve $ATP^{(k+1)}$ with value $z_a^{(k+1)}$. Then $z_a^{(k)} \geq z_a^{(k+1)}$

Since every iteration yields a better upper bound than the previous, then at iteration k we can define the current best upper bound z_u, as $z_a^{(k)}$.

In a similar manner for every aggregate program $ATP^{(k)}$ we can derive a lower bound $z_p^{(k)}$. Unfortunately, the lower bounds are not necessarily monotonically increasing. Mowever, if we define the lower bound z_r as the maximum of all the lower bounds, i.e. as

$$z_r = \max\{z_p^{(1)}, z_p^{(2)}, \ldots, z_p^{(k)}\}$$

then the error bound level of

$$\frac{z_u - z_r}{z_r}$$

will be nonincreasing at every iteration. In practice, it is generally strictly decreasing, as the conditions yielding otherwise are rare.

In summary, given an initial aggregate program, it can be reformulated by unaggregating particular violated constraints in selected clusters to form a subsequent aggregate program that is still a transportation problem. The bounds formed from the subsequent programs are successively tighter and thus the algorithm can be continued until adequate bounds are attained.

Note that the above method of reformulating the aggregate problems will result in a sequence of aggregate solutions $z_a^{(k)}$, that will converge to the optimal solution value of (TP) in a finite number of iterations. This is so, since at each stage the number of destinations in $ATP^{(k)}$ increases by at least one, hence in at most N - L steps (where L was the number of aggregate destinations in $ATP^{(1)}$), the initial problem will be at hand.

Extension to Source Aggregation

For the most part, the above methods have been illustrated for the case of only destination aggregation. However the principles are identical for source aggregation. Joint source and destination aggregation is simply the combination of the separate source aggregation and destination aggregation. Again, the principles can be easily

extended. As we have seen, destination aggregation produces an aggregate problem that is a restriction of (TP). Source aggregation simply further restricts the problem. Thus the solution to a joint aggregated problem still yields an upper bound on z^*. Also, the proposed lower bounds can be easily generalized to this case.

In a recent paper, Zipkin and Raimer [21] present an alternative method of disaggregation which preserves integrality of the solution. No comparison is provided, however, with the bound obtained by the optimal disaggregation method.

4. Computational Results and Discussion

One of the main objectives of the research was to provide an algorithm that would initially form a good aggregate program, utilize the information contained therein to generate bounds on the original problem solution and, if need be, to reformulate the aggregate problem into a yet better aggregate problem with tighter bounds. Within these objectives lies the necessity to generate an algorithm that is efficient, or at least comparable in computation time to solving the original problem. Second, it is desirable that the algorithm should contain the capabilities to solve problems larger than the available transportation code can handle. Finally, the bounds developed on the optimal values should be reasonably tight.

Implementation of the Algorithm

The proposed algorithm was implemented in its entirety on the CDC CYBER 170/730 mainframe operating under the NOS 1.4 operating system. All control programs to run the various programs were written in a procedural machine language basic to the CYBER mainframe. The transportation code

TPGO used to solve both the original and aggregate problems is by McGinnis [16]. The code utilizes a tree representation for the basis and is reasonably efficient. The program is written in Fortran IV.

All algorithms to develop the initial aggregate programs and reformulate the subsequent aggregate programs were written in APL 2.1014. This allows for efficient handling of the problem's data structures.

Generation of Test Problems

The transportation problems were randomly generated from a uniform [0,1] distribution of values. Randomly generated demands and supplies were scaled to the range [5,50] with one dummy demand set up to balance supply and demand. Geoffrion [9] shows that a sufficient condition for an error bound level of zero (i.e. $z_u = z^*$) occurs when the cost c_{ij} can be written in a factored form, i.e. $c_{ij} = q_i b_j$ for each i,j. If the destinations are close and similar in the type of demand, then the q_i's may represent a cost/unit-mile. With this thought in mind, the costs were generated by $c_{ij} = q_i b_j + r_{ij}$ where $q_i \in [40,50]$ and the parameter $r_{ij} \in [-100,100]$ to produce some randomness.

Experimental Design

The problems tested were divided into two categories: medium size problems (30 sources, 30 destinations); and large problems (100 sources, 100 destinations). The size 100x100 was chosen as a reasonable extension over the medium size of 30x30. The initial cluster size was chosen as 3x3 for the 30x30 problems and 5x5 for the 100x100 problems. These sizes were chosen based on preliminary results experienced in the aggregation

of general linear programs (see [18]). Finally, all test problems were subjected to both destination and source aggregation.

Computational Results

The first set of problems tested were those problems in the range of 30x30. These problems were evaluated using the proposed methodology and the computation times compared with the time to solve the original problem. The problems were run on a CDC CYBER 760/140. For the five test problems generated, the results using both TPGO and the proposed method are given in Table 1. The table provides the error bound level and computation time (cpu-seconds) at each iteration for the five test problems. The average time to solve these problems to optimality by TPGO was 0.796 cpu-seconds. In this same time, the proposed method produced aproximate solutions with an average error bound level of 9.0%. Although the times to solve the problems using the aggregate method are slightly higher, the orders of magnitudes of the solution times are comparable.

The second set of problems tested were for those transportation probems with 100 sources and 100 destinations. These problems used an initial aggregate program size of five destinations and five sources. These programs were also run on a CDC CYBER 760/140. The results of these runs are shown in Table 2. The solution times for TPGO are again comparable to the times for using the proposed algorithm to an error bound level of 10%. It is clear that the algorithm can fully handle large type problems, as the largest aggregate transportation problems that was solved by the proposed method was a 32 source by 38 destintion transportation problem.

5. Conclusions

The main objective of the research was to provide a practical tool to solve large transportation problems through the enhancement of the computer's capabilities. The computational results show that aggregate programming can be used on problems larger than currently solvable on a particular system. Further, the proposed bounds produce tighter bounds than previously available. A procedure is developed to reformulate the aggregate problem and thus produce tighter bounds at each subsequent iteration. This strategy may provide a more cost effective approach to solving certain transportation problems.

Table 1

Computation Time for Proposed Algorithm and TPGO

Iteration	Problem Number #1	#2	#3	#4	#5
	a b	a b	a b	a b	a b
1	.136 20.4	.136 17.4	.135 29.2	.135 30.2	.138 34.2
2	.218 17.5	.219 17.1	.215 18.1	.213 24.9	.218 20.2
3	.338 13.1	.403* 14.2	.325 15.0	.301 16.7	.332 15.2
4	.494 10.9	.610* 13.0	.556* 12.3	.537* 9.6	.595* 13.9
5	.795* 9.6	.863* 12.2	.840* 10.0		.850* 7.4
6		1.001* 8.3			
Time to Solution by TPGO	.772	.922	.711	.763	.812

*with optimal disaggregation

a Computation time (cpus)
b Error bound level (%)

Remarks: All problems run on a CDC CYBER 760/140.

Table 2

Computation Times for Large Problems (100x100)

Iteration	Problem Number #1	#2	#3
	a b	a b	a b
1	.234 21.7	.221 54.7	.242 47.2
2	.359 21.1	.381 49.3	.371 28.6
3	.505 20.4	.537 37.4	.512 21.2
4	.682 18.7	.712 25.2	.699 19.3
5	1.051* 11.7	.924 18.3	1.062* 10.6
6	1.201* 9.9	1.257* 8.2	1.216* 10.2
Time to Solution by TPGO	1.194	1.186	1.203

*with optimal disaggregation

a Computation time (cpus)
b Error bound level (%)

Remarks: All problems run on a CDC CYBER 760/140

REFERENCES

1. Ara, K., "The Aggregation Problem in Input-Output Analysis," Econometrica, 27: 257-262 (1959).

2. Balas, E., "Solution of Large-Scale Transportation Problems through Aggregation," Operation Research, 13: 82-93 (1965).

3. Blin, J. M. and Cohen, C., "Technological Similarity and Aggregation in Input-Output Systems: A Cluster-Analytic Approach," The Review of Economics and Statistics, 59: 82-91 (1977).

4. Day, R., "On Aggregating Linear Programming Models of Production," Journal of Farm Economics, 45: 797-813 (1963).

5. Evans, F., "Aggregation in Generalized Transportation Problem," Computers and Operations Research, 6: 199-204 (1979).

6. Fei, J. C. H., "A Fundamental Theorem for the Aggregation Problem in Input-Output Analysis," Econometrica 24: 400-412 (1956).

7. Fisher, W. D., Clustering and Aggregation in Economics, Baltimore: Johns Hopkins University Press (1969).

8. Fisher, W. D., "Criteria for Aggregation in Input-Output Analysis," The Review of Economic and Statistics, 40: 250-260 (1958).

9. Geoffrion, A. M., "Customer Aggregation in Distribution Modeling," Working Paper 259, Management Science Study Center, UCLA (1976).

10. Huberman, G., "Error Bounds for the Aggregated Convex Programming Problem," Math Programming, 26: 100-108 (1983).

11. Ijiri, Y., "Fundamental Queries in Aggregation Theory," Journal of the American Statistical Association, 56: 766-782 (1971).

12. Kallio, M., "Computing Bounds for the Optimal Value in Linear Programming," NRLQ 24: 301-308 (1977).

13. Kuhn, H. W. and D. E. Cullen, "Aggregation in Network Models for Transportation Planning," Mathematica, Princeton, N. J. (prepared for U. S. Department of Transportation)(1975).

14. Lansdowne, Z. F., "Survey of Research on Model Aggregation and Simplification," Technical Report SOL 79-26, Department of Operations Research, Stanford University (1979).

15. Lee, G., Surrogate Programming Through Aggregation, Ph.D. Dissertation, UCLA (1975).

16. McGinnis, L. F., Report 103, North Carolina State University (1975).

17. Mendelssohn, R., "Improved Bounds for Aggregated Linear Programs," Operations Research, 28, 1450-1453 (1980).

18. Taylor, R. W., "Aggregate Programming in Large Scale Linear Systems," Ph.D. Dissertation, Georgia Institute of Technology (1983).

19. Zipkin, P. H., "Aggregation in Linear Programming," Ph.D. Dissertation, School of Organization and Management, Yale University (1977).

20. Zipkin, P., "Bounds for Aggregating Nodes in Network Problems," Mathematical Programming, 19: 155-177 (1980).

21. Zipkin, P. and Raimer, K., "An Improved Disaggregation Method for Transportation Problems," Mathematical Programming, 26: 238-242 (1983).

END

FILMED

9-83

DTIC