A MULTILEVEL COST-SPACE APPROACH TO SOLVING THE BALANCED LONG TRANSPORTATION PROBLEM*

Kevin J. Cavanaugh
U.S. Coast Guard
Research and Development Center
Groton, CT

Van Emden Henson
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943

SUMMARY

We develop a multilevel scheme for solving the balanced long transportation problem, that is, given a set $\{c_{kj}\}$ of shipping costs from a set of M supply nodes S_k to a set of N demand nodes D_j, we seek to find a set of flows, $\{x_{kj}\}$, that minimizes the total cost $\sum_{k=1}^{M}\sum_{j=1}^{N} x_{kj}c_{kj}$. We require that the problem be balanced, that is, the total demand must equal the total supply. Solution techniques for this problem are well known from optimization and linear programming. We examine this problem, however, in order to develop principles that can then be applied to more intractible problems of optimization.

We develop a multigrid scheme for solving the problem, defining the grids, relaxation, and intergrid operators. Numerical experimentation shows that this line of research may prove fruitful. Further research directions are suggested.

INTRODUCTION

The transportation problem is the simplest of network flow problems. It is posed on a bipartite graph, consisting of a set of M supply nodes, a set of N demand nodes, and a set of arcs connecting them. Each supply node S_i has a fixed amount s_i of a commodity which it can provide. Each demand node D_j has a fixed requirement d_j for that commodity, and for each arc (i, j) connecting supply node S_i to demand node D_j there is an associated cost per unit flow c_{ij}. When the total supply equals the total demand the problem is *balanced*. When $M << N$, the problem is referred to as a *long* transportation problem. Denoting the flow on arc (i, j) by x_{ij}, the transportation problem

*This work was supported in part by Naval Postgraduate School Research Council, Grant No. MA000-MA999/4476-4479

can be expressed

$$\text{Minimize} \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \quad \text{subject to:} \sum_{j=1}^{N} x_{ij} = s_i, \quad \sum_{i=1}^{M} x_{ij} = d_j, \quad x_{ij} \geq 0.$$

Let b denote an $(M+N)$-vector whose first M entries are the available supplies s_i at nodes S_1 through S_M, and whose last N entries are (negatives of) the required demands d_j at demand nodes D_1 through D_N. Let K be the number of arcs in the problem. Throughout this work we shall assume that every supply node is connected to every demand node, so that $K = MN$. Let the K-vector x be composed of the flow on the arcs from the M supply-nodes to the N demand nodes in some order, and the K-vector c be the cost of shipping on those arcs in the same order. We denote by A the incidence matrix of the graph , so that A has as many rows as there are nodes in the problem, $M+N$, and as many columns as there are arcs (MN). Each column of A is associated with one arc of the problem, and they are arranged in an order that matches the order of the vectors c and x. Each column has exactly two non-zero entries: a $+1$ in the row corresponding to the tail (supply) node S_i of the arc, and a -1 in the row corresponding to the head (demand) node D_j. Each row of A is associated with one of the constraints of the problem [1]. Then the problem may be written in matrix notation as

$$\begin{array}{rl} \text{Minimize:} & c^T x \\ \text{Subject to:} & Ax = b, \\ & x \geq 0. \end{array}$$

A simple example is presented in Figure 1. In the example, there are four supply nodes, having 12, 15, 10, and 7 units of the commodity to deliver. There are three demand nodes, requiring 13, 20, and 11 units of the commodity. We seek to find a flow vector

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} & x_{41} & x_{42} & x_{43} \end{pmatrix}^T$$

given that the vector of costs, written in corresponding order, is

$$\begin{pmatrix} 2 & 1 & 5 & 6 & 4 & 3 & 1 & 7 & 4 & 2 & 3 & 4 \end{pmatrix}^T.$$

The algebraic description of this problem is to find x such that $c^T x$ is minimized, subject to the system of constraints

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{41} \\ x_{42} \\ x_{43} \end{pmatrix} = \begin{pmatrix} 12 \\ 15 \\ 10 \\ 7 \\ -13 \\ -20 \\ -11 \end{pmatrix}.$$

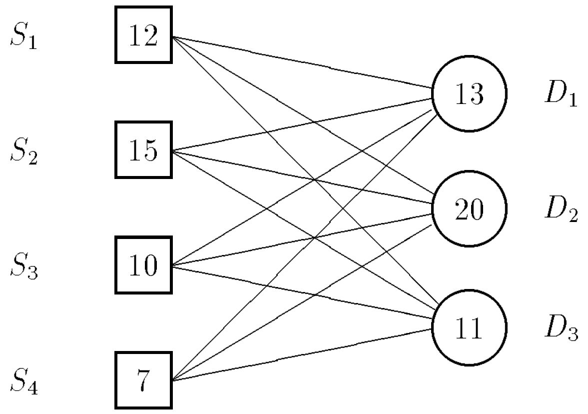

Figure 1: *A simple example of a transportation problem*

Very little work has been done on multigrid methods for discrete optimization problems. Significant studies to date are [2], [3], [4], and [5]. The traditional optimization algorithm which most closely resembles a multilevel algorithm is aggregation/disaggregation [6], [7], and [8], in which nodes are aggregated in a logical way in order to reduce the size of the problem, and the solution to the smaller problem is disaggregated to provide an initial estimate for the solution to the original problem. The most successful work to date, and the work that inspired this study, is that of Kaminsky [4].

COST-SPACE

In [4] it is required that the demand nodes occupy a physical location in space, and that a relationship exist between transportation costs and distances. This is done so that the coarsening step may be performed by aggregating together demand nodes that are physically near one another. For this to make sense, it in necessary that shipment to each of the aggregated demand nodes involve a similar cost, which naturally occurs if the shipping cost is a function of distance. For many applications this makes perfect sense; the cost of shipping a commodity is often directly linked to the distance the commodity must be shipped. This restriction is overly limiting for other types of problems, however. For example, the manpower assignment problem, in which a specified number of jobs must be assigned a given set of workers, can be formulated as a transportation problem. There is no distance involved in such a problem, and cost of assignment is related to other factors, such as the cost of training an individual for a specific task.

In order to address problems that have no geometrical dependence of cost on distance, we employ a change of coordinate systems from physical space to a space we describe as cost space. For the $M \times N$ problem, cost space is the M-dimensional space in which each of the coordinate axes is the cost of shipping from one of M supply nodes. Each of the N demand nodes is placed in cost space at the point whose coordinates are the unit costs of shipping from the supply nodes to it. For example, the three demand nodes in Figure 1 would be placed in a four-dimensional cost space, and would have the coordinates $D_1 = (2, 6, 1, 2)$, $D_2 = (1, 4, 7, 3)$, and $D_3 = (5, 3, 4, 4)$. This change of coordinate systems means shipping cost becomes the metric of the problem, so that two demand nodes are "near" each other only if the shipping costs are similar, and the aggregation of neighboring demand nodes automatically ensures the similarity of their costs.

Posed in cost space, the dimensionality of the problem equals the number of supply nodes. In traditional multigrid methods, one typically uses grids that are tensor products of one-dimensional grids, each having a cardinality of gridpoints that is a power of two. In the cost space approach this would lead to a very rapid growth in the size of the problem; for this reason the cost space approach can be applied only to problems with a relatively small number of supply nodes. This is one reason for restricting our attention to the long transportation problem.

Reduced dimension cost space

If at least one supply node is connected to all demand nodes (and in our work we assume this to be true of all supply nodes) then we can transform the $M \times N$ transportation to an equivalent $(M-1) \times N$ problem, which we call the *reduced dimension* problem. Since we are dealing with the long problem, the transformed problem is somewhat simpler and less expensive to solve. The transformation is accomplished as follows. Suppose that supply node S_l is connected to all demand nodes. Then for each demand node D_j, we subtract c_{lj}, the cost of shipping from supply node S_l to D_j, from all of the shipping costs into demand node D_j. That is, we form an auxilliary cost vector $\tilde{c}_{ij} = c_{ij} - c_{lj}$. The result is that, for supply node S_l, all the demand nodes map to the origin in cost space. Effectively S_l has been removed from the problem, leaving an $(M-1) \times N$ problem to be solved. For example, if we use the cost of shipping from S_2 on the example in Figure 1, the transformed cost vector becomes

$$\tilde{c} = \begin{pmatrix} -4 & -3 & 2 & 0 & 0 & 0 & -5 & 3 & 1 & -4 & -1 & 3 \end{pmatrix}^T.$$

We can show that while the objective function value is different for the new problem, a solution for one is equivalent to a solution for the other.

Theorem 1 *Let the $M \times N$ balanced long transportation problem be represented by a bipartite graph G, and suppose that supply node S_l is connected to all demand nodes. Let b be the $(M+N)$ length column vector whose first M entries are the supplies at the supply nodes and whose remaining N entries are the negative of the demands at the demand nodes. Let A be the adjacency matrix of the graph G; that is, for each arc (i,j) we have $A(i,(i-1)N+j) = 1$ and $A(M+j,(i-1)N+j) = -1$. Let c be the $(M+N)$ length vector whose $k = (i-1)N+j$ element is the cost c_{ij} of shipping from node S_i to node D_j along arc (i,j). Define $\tilde{c}$ to be the vector whose k^{th} entry is $\tilde{c}_k = c_{ij} - c_{lj}$. Then x^* is a solution to the problem*

$$\begin{array}{rl} \textit{Minimize:} & c^T x \\ \textit{Subject to:} & Ax = b, \\ & x \geq 0\ , \end{array}$$

if and only if it is a solution to the problem:

$$\begin{array}{rl} \textit{Minimize:} & \tilde{c}^T x \\ \textit{Subject to:} & Ax = b, \\ & x \geq 0\ . \end{array}$$

Proof:

$$\begin{aligned}
\tilde{c}^T x &= \sum_{k=1}^{M} \sum_{j=1}^{N} \tilde{c}_{kj} x_{kj} = \sum_{k=1}^{M} \sum_{j=1}^{N} (c_{kj} - c_{lj}) x_{kj} \\
&= \sum_{k=1}^{M} \sum_{j=1}^{N} (c_{kj} x_{kj} - c_{lj} x_{kj}) \\
&= \sum_{k=1}^{M} \sum_{j=1}^{N} c_{kj} x_{kj} - \sum_{k=1}^{M} \sum_{j=1}^{N} c_{lj} x_{kj} \\
&= c^T x - \sum_{j=1}^{N} c_{lj} \sum_{k=1}^{M} x_{kj} = c^T x - \sum_{j=1}^{N} c_{lj} d_j,
\end{aligned}$$

since $\sum_{k=1}^{M} x_{kj} = d_j$ in the balanced problem. But $\sum_{j=1}^{N} c_{lj} d_j$ does not depend on x, and therefore $\tilde{c}^T x$ achieves its extreme values precisely when $c^T x$ does. ∎

This transformation of the costs to reduced-dimension space maps all costs of shipping from S_l to the origin in cost space. As will be shown in the next section, our algorithm requires that the demand nodes be sorted once according to the cost of shipping. Since sorting is a fairly expensive operation, the savings generated by reducing the dimension of the problem are tangible. Once the transformation to reduced-dimension cost-space has been performed, the resulting problem may be solved with no further consideration of the transformation. Therefore, in the remainder of this work it is assumed that when an $M \times N$ problem is to be solved, it may be the reduced dimension version of a problem that was originally $(M+1) \times N$.

A MULTIGRID APPROACH TO THE TRANSPORTATION PROBLEM

Following traditional multigrid design approaches, we develop the necessary tools to devise a multigrid V-cycle, which we will combine with a nested iteration to create an FMG algorithm. In particular, it is necessary to devise restriction and prolongation methods, some form of local relaxation, and to weave them into an algorithm.

Restriction

To devise a restriction algorithm, it is first necessary to define a coarse grid. We use an approach in which each gridpoint on the coarse grid is a demand node for the coarse grid problem, and represents a pair of demand nodes on the fine grid. This is accomplished as follows. The demand nodes are first sorted by increasing cost of shipping from S_1, and divided into two groups about the median of the sorted cost. This procedure results in two groups of demand nodes, one with a lower cost of shipping from S_1, and one for which shipping from S_1 is more expensive. Each of these groups are then sorted according to increasing cost of shipping from S_2 and divided into two groups about the median cost. This results in four groups, one for which shipping is expensive from both supply nodes, one group for which shipping is inexpensive from both supply nodes, one group for which shipping is expensive from S_2 and inexpensive from S_1, and one group where shipping is expensive from S_1 and inexpensive from S_2.

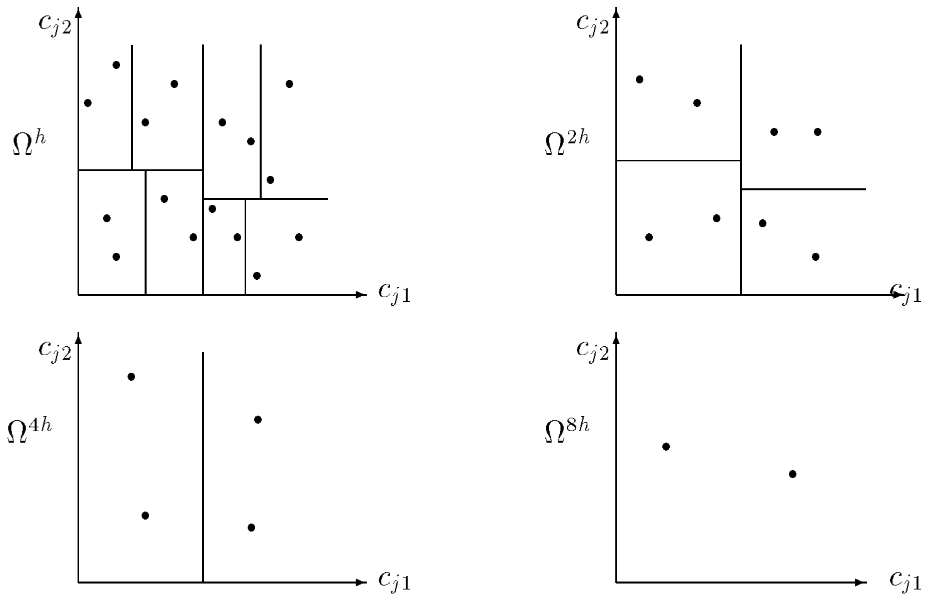

Figure 2: *A simple example of a coarsening process for the transportation problem.*

If there are more than two supply nodes in the problem, this process is continued. The four groups are each sorted by cost of shipping from supply node S_3, then divided into smaller groups if necessary and sorted again, according to cost from S_4, etc. If the groups contain more than two nodes after the nodes have been sorted according to cost from all supply nodes, the sorting process begins again with cost from S_1 on each of the groups. Eventually, the nodes will be sorted into pairs that have similar shipping costs from all supply nodes. Each of these pairs of demand nodes is then replaced with a single coarse grid demand node, the collection of which constitute the first coarse grid.

Further coarsening is accomplished by repeating the procedure described above on the coarse grids to produce still coarser grids. Figure 2 shows a simple example of the coarsening process. If the number of points on the original grid is a power of two, then in the limit a coarsest grid would consist of a single demand node. As in traditional multigrid methods, once the hierarchy of grids is established it is stored, so that the sorting process need never be repeated.

Three quantities must be restricted when aggregating a pair of fine grid demand nodes into a coarse grid demand node: the demands, flows, and costs. Let D_m^{2h} be the coarse grid node representing the fine grid nodes D_j^h and D_l^h. It seems natural that the demands can be restricted simply by summing the demands of the two fine grid nodes to produce the demand at the coarse grid node, $d_m^{2h} = I_h^{2h}[d_j^h, d_l^h] = d_j^h + d_l^h$. Similarly, the flow x_{km}^{2h} from any supply node S_k into the coarse demand node should be the sum of the flows from S_k to each of the fine grid demand nodes that make up the coarse grid node, $x_{km}^{2h} = I_h^{2h}[x_{kj}^h, x_{kl}^h] = x_{kj}^h + x_{kl}^h$.

Restricting the cost of shipment is more complicated, and no obvious "best" approach is apparent. However several methods can be considered. The simplest of these is to define the coarse cost c_{km}^{2h} to be the minimum of the fine costs, i.e., $c_{km}^{2h} = I_h^{2h}[c_{kj}^h, c_{kl}^h] = \min(c_{kj}^h, c_{kl}^h)$. Other simple schemes are readily devised, such as using the maximum of the fine costs, or a weighted average of the fine grid costs. We use a weighted average of the fine grid costs. Again, there are several possible weightings, each having valid arguments for and against it. Three schemes were tested in some depth, equal weighting, flow weighting, and demand weighting:

$$\begin{array}{lcccl}
\text{Equal weighting:} & c_{km}^{2h} & = & I_h^{2h}[c_{kj}^h, c_{kl}^h] & = \dfrac{c_{kj}^h + c_{kl}^h}{2}, \\
\text{Demand weighting:} & c_{km}^{2h} & = & I_h^{2h}[c_{kj}^h, c_{kl}^h] & = \dfrac{d_j^h c_{kj}^h + d_l^h c_{kl}^h}{d_j^h + d_l^h}, \\
\text{Flow weighting:} & c_{km}^{2h} & = & I_h^{2h}[c_{kj}^h, c_{kl}^h] & = \dfrac{x_{kj}^h c_{kj}^h + x_{kl}^h c_{kl}^h}{x_{kj}^h + x_{kl}^h}.
\end{array}$$

With flow weighting, provision must be made for the case where there is zero flow on both arcs. In such a case flow weighting can be replaced with either demand weighting or equal weighting. In general, we found that demand weighting most consistently gave the best results, and adopted it for our algorithm.

Prolongation, or Interpolation

Suppose that the problem has been solved on the coarse grid Ω^{2h}. We seek a method of prolongation, that is, a way in which the coarse grid solution can be interpolated onto the fine grid. In the coarse grid solution there is some quantity of flow x_{km}^{2h} giving the flow from each supply node S_k to each coarse grid demand node d_m^{2h}. Each such demand node on the coarse grid, however, represents the aggregation of two demand nodes on the fine grid, d_j^h and d_l^h. An interpolation of the coarse grid solution, therefore, can be constructed by treating the M flows $x_{1m}^{2h}, x_{2m}^{2h}, \ldots, x_{Mm}^{2h}$, into the coarse grid demand node d_m^{2h}, as supplies. Interpolation, then, consists of solving for each coarse grid demand node, the $M \times 2$ transportation problem with those supply values, the two demand nodes d_j^h and d_l^h, and the shipping costs c_{km}^{2h}, $k = 1, 2, \ldots, M$. (Figure 4 shows schematically how the interpolation process appears.)

Having defined the interpolation process as finding the solutions to many small transportation problems, we turn our attention to the mechanism for finding these solutions. A method for solving such $M \times 2$ problems is described below. The method is a special case of Vogel's approximation method.

Algorithm 1 *Solving the $M \times 2$ Balanced Transportation Problem*

1. *For each supply node S_k, find the difference in cost of shipping $\delta_k = |c_{kj}^h - c_{kl}^h|$ to the two fine grid demand nodes d_j^h and d_l^h.*

2. *Rank the M supply nodes in decreasing order of these cost differentials, so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_M$.*

3. *Repeat until all supply nodes are removed from the problem:*

   (a) *Denote the supply node at the top of the ordered list as the "current" supply node, and allocate flow to the demand node with the lower cost of shipment, that is, along the least expensive arc, thus determining a "current" demand node. (In the event that more than one node has the largest differential cost, select from among them the node with the*

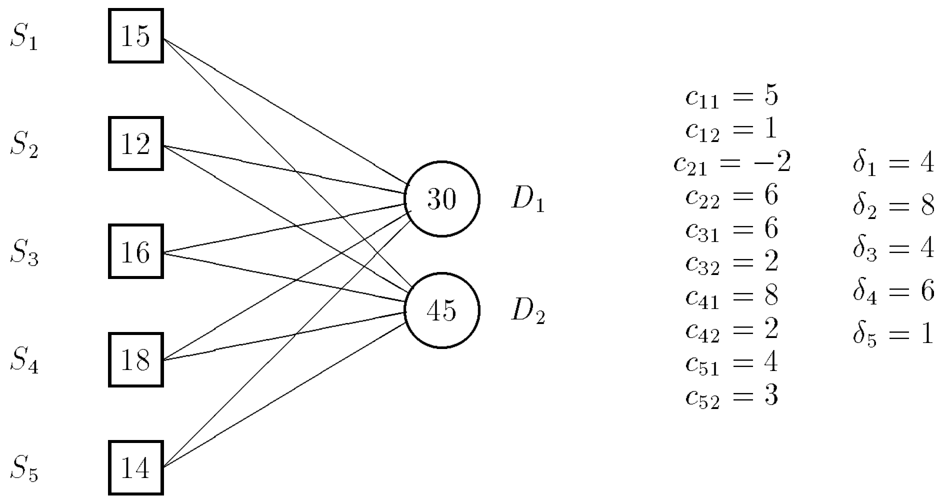

Figure 3: *Example problem illustrating the solution method for an $M \times 2$ problem.*

smallest cost along one of its two arcs). Allocate flow along this arc until either the demand at the current demand node is satisfied or until the supply at the current supply node is exhausted.

(b) If the supply at the current supply node is exhausted, remove that supply node from the problem.

(c) If the demand at the current demand node is satisfied, remove that demand node from the problem, allocate the remaining supply from the current supply node to the remaining demand node, and remove the current supply node from the problem.

4. Stop.

As an example of this procedure, consider the five by two problem shown in Figure 3. The five supply nodes $S_1, S_2, \ldots, S_5$ have, respectively, 15, 12, 16, 18 and 14 units of the commodity to deliver. The demands of the two demand nodes D_1 and D_2 are 30 and 45. Let $\delta = (4\ 8\ 4\ 6\ 1)^T$ be the vector whose i^{th} entry is the difference δ_i between shipping cost from supply node S_i to the two demand nodes (the costs themselves are given for each arc in the figure). Sorting from largest to smallest value of δ_i, the supply nodes are ordered $(S_2, S_4, S_1, S_3, S_5)$. Note that, while the differences for nodes S_1 and S_3 are the same, the cost c_{12} along the arc from node S_1 to node D_2 is less expensive than either of the arcs incident from node S_3. Starting with node S_2, then, as much flow as possible is sent along the least expensive arc. In this case, that is the arc to demand node D_2. Since this demand exceeds the available supply from node S_2, all of the flow from node S_2 goes along this arc. Similarly, node S_4 and then node S_1 send all of their supply to node D_2. When node S_3 has sent 12 units of flow along its least expensive arc, the demand at node D_1 is completely met. Thus node S_3 sends its remaining units to node D_2, as does node S_5. Although the arc from node S_5 to D_1 is less expensive, the demand at D_1 has been met from supply nodes where the difference in arc costs is greater.

We can show now that because of the special structure of the $M \times 2$ problem, i.e., the fact that there are only two demand nodes, this procedure produces an optimal solution.

Theorem 2 *Let x be the vector of flows assigned for the $M \times 2$ problem using the algorithm given above. Then x is an optimal solution to the $M \times 2$ problem.*

Proof: Suppose that x is not an optimal solution. Then there exists a flow $x^* \neq x$ such that $z^* = c^T x^*$ is optimal. We will show that if x is determined by the algorithm given above and $z = c^T x$, then $z^* \geq z$, contradicting the assumption that x is not an optimal solution. Letting $\delta_k = |c_{k1} - c_{k2}|$ for each $k = 1, 2, m \ldots, M$, assume the supply nodes have been ordered in decreasing order of δ_k so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_M$. Let i be the first supply node for which x^* differs from x, and without loss of generality, assume that $c_{i1} \leq c_{i2}$. Let $\Delta = x_{i1} - x^*_{i1}$. We first observe five useful facts:

1. Since the problem is balanced, total flow out of S_k equals the supply, so that $s_k = x_{k1} + x_{k2} = x^*_{k1} + x^*_{k2}$ for every k, implying $x_{k1} - x^*_{k1} = x^*_{k2} - x_{k2}$.
2. In particular, since $\Delta = x_{i1} - x^*_{i1}$ then $-\Delta = x^*_{i2} - x_{i2}$.
3. Since the problem is balanced, total flow into D_1 equals demand, so that $d_1 = \sum_{j=1}^{M} x_{j1}$ and $d_1 = \sum_{j=1}^{M} x^*_{k1}$. Subtracting these two relations, and noting that x and x^* do not differ for $j = 1, 2, \ldots, i-1$, we find that $0 = x_{i1} - x^*_{i1} + \sum_{j=i+1}^{M}(x_{j1} - x^*_{j1})$, implying that $\Delta = \sum_{j=i+1}^{M}(x^*_{j1} - x_{j1})$.
4. Using similar reasoning, we obtain $-\Delta = \sum_{j=i+1}^{M}(x^*_{j2} - x_{j2})$.
5. By construction, since $c_{i1} \leq c_{i2}$, then x_{i1} is as large as it can possibly be, so that if x and x^* differ for node i then $x_{i1} > x^*_{i1}$, implying that $\Delta > 0$.

Next, we observe that

$$\begin{aligned} z^* &= z + z^* - z \\ &= z + c^T x^* - c^T x \\ &= z + \sum_{j=i}^{M} \left(x^*_{j1} c_{j1} + x^*_{j2} c_{j2} \right) - \sum_{j=i}^{M} \left(x_{j1} c_{j1} + x_{j2} c_{j2} \right), \end{aligned}$$

where we have used the fact that x^* and x do not differ for $j < i$. Separating the flows from supply node i, we can write

$$\begin{aligned} z^* &= z + (x^*_{i1} - x_{i1})\, c_{i1} + (x^*_{i2} - x_{i2})\, c_{i2} + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) c_{j1} + \sum_{j=i+1}^{M} \left(x^*_{j2} - x_{j2} \right) c_{j2} \\ &= z - \Delta\, c_{i1} + \Delta\, c_{j2} + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) (c_{j1} - c_{j2}) \end{aligned}$$

where we have used the fact that $x_{k1} - x^*_{k1} = x^*_{k2} - x_{k2}$ for all k. Recalling that $\delta_k = |c_{k1} - c_{k2}|$, and that since $c_{i1} \leq c_{i2}$ then $\delta_i = c_{i1} - c_{i2}$, we observe that

$$z^* \geq z + \Delta\, \delta_i + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) (-\delta_j).$$

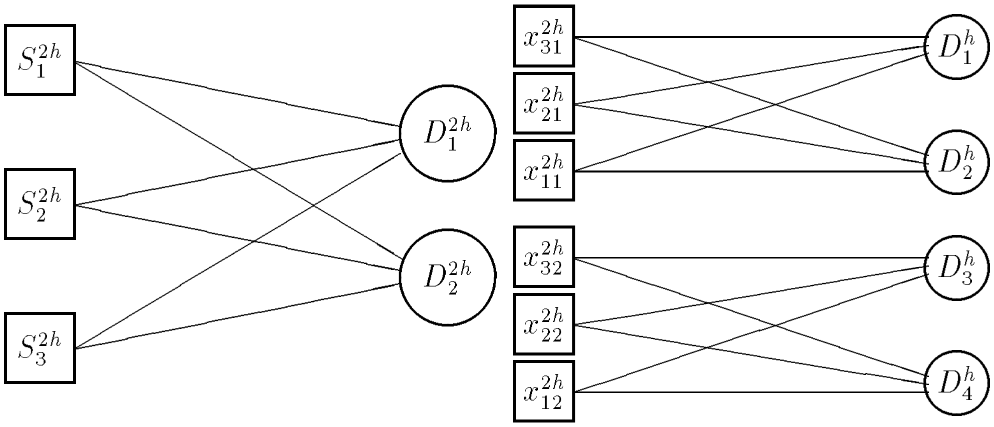

Figure 4:
An illustration of interpolation by local optimization. The flows in the 3×2 coarse grid problem are the supplies for the two 3×2 local problems. The combination of the flows solving those two local problems makes up the interpolated solution to the 3×4 fine grid problem.

Since the nodes are ordered in decreasing order of δ_k, we know that $-\delta_i \leq -\delta_j$ for all $j \geq i$, and therefore

$$z^* \;\geq\; z \,+\, \Delta\,\delta_i \,-\, \delta_i \sum_{j=i+1}^{M} \left(x_{j1}^* - x_{j1}\right).$$

Finally, recalling that $\Delta = \sum_{j=i+1}^{M}(x_{j1}^* - x_{j1})$, we obtain

$$z^* \;\geq\; z \,+\, \Delta\,\delta_i \,-\, \delta_i\,\Delta.$$

Therefore $z^* \geq z$, contradicting the assumption that x is not an optimal solution.

∎

Relaxation

Suppose that we have solved the coarse grid problem, and have interpolated that solution by solving an $M \times 2$ transportation problem for each coarse grid demand node in order to pass the local solution to the two fine grid demand nodes represented by each coarse grid node. The supplies for this local $M \times 2$ problem are the coarse grid flows. Figure 4 displays a schematic showing how the interpolation process appears graphically.

It is important to note that while each of the local $M \times 2$ problems has been solved optimally, there is no reason to expect that the total set of fine grid flows thus assigned will be optimal. For this reason, it is essential that we devise some kind of "relaxation" scheme, whose task is to smooth or correct errors left by the interpolation scheme.

When two $M \times 2$ subproblem solutions are viewed from a more global perspective, as a solution to an $M \times 4$ problem (or as a portion of a solution to a still larger problem), this combination of locally optimized solutions may be flawed, in that too many arcs may have flow on them. This is because the minimum value of the objective function for a balanced transportation problem can always be obtained with a flow regime having flow on at most $M + N - 1$ arcs. This simply reflects the fact from linear programming theory that an extreme point solution has flow on $M + N - 1$ arcs, if the solution is non-degenerate [9], and that an optimal solution can always be found at one of the extreme points (a degenerate solution is one in which distinct subsets of demand nodes are supplied by distinct subsets of supply nodes). If the solution is degenerate, there will be fewer arcs with flow on them. For example, if $N > M$, then in the extreme degenerate case each supply node provides flow to a disjoint subset of the demand nodes. This means that each demand node has exactly one arc with flow on to it, so that precisely N arcs have flow. If $N < M$, then the extreme degenerate case is when each supply node has exactly one arc with flow, giving M such arcs.

When interpolating from Ω^{2h} to Ω^{h}, each coarse demand node generates two fine grid demand nodes and the optimal solution to the $M \times 2$ subproblem has $M + 1$ arcs with flow, in the non-degenerate case. If the subproblem solution is degenerate, then M arcs have flow. If there are $N/2$ demand nodes on Ω^{2h}, then after the interpolation the collection of subproblem solutions (viewed as the initial feasible solution to the Ω^{h} problem) will have flow on at least $NM/2$ and at most $(NM + N)/2$ arcs, depending on how many subproblems are degenerate.

Thus, whenever $NM/2$ is greater than $M + N - 1$, (which is true for any long transportation problem where $M > 2$ and $N > 3$), the collection of local solutions has too many arcs with flow to be an extreme point solution to the fine grid problem, and is probably less than optimal. The local relaxation scheme developed here is designed to reduce the number of arcs with flow for the fine grid problem, which will generally have the effect of moving the global solution toward an optimal solution.

The mechanism by which we do this is cycle removal. Since there are $M + N - 1$ arcs in a spanning tree over $M + N$ nodes, and the addition of a single arc (or more) to a tree results in a graph with at least one cycle, then for most problems, the interpolation process will introduce cycles. We note that while this has been developed in the setting of the *entire* collection of local solutions, it is also true in a pairwise sense. That is, each of two $M \times 2$ local solutions will have either $M + 1$ or M arcs with flow. Viewing the pair as a solution to an $M \times 4$ problem, we observe that the combined solution will have at least $2M$ arcs with flow. If $M > 2$ this equals or exceeds the $M + 3$ arcs with flow that would be present in an extreme point solution.

To illustrate this, consider the possibilities when two 3×2 local solutions are combined into a 3×4 solution, as shown in Figure 5. In a), two non-degenerate solutions are combined. Numbering the demand nodes of the combined problem clockwise from the upper left and the supply nodes from top to bottom, we observe that there are three cycles in the combined solution $(S_1, D_3, S_2, D_1, S_1)$, $(S_1, D_3, S_3, D_4, S_2, D_1, S_1)$, and $(S_2, D_3, S_3, D_4, S_2)$. In b), a degenerate solution is combined with a non-degenerate solution, yielding a combined solution with one cycle. In c), two degenerate solutions are combined into a solution that has no cycles, while in d), two degenerate solutions are combined into a solution that has one cycle.

A reasonable candidate for a local relaxation process is to adjust the flow in the initial solution produced by the interpolation process, so that cycles are removed and the objective function is reduced. The effect of this procedure is to adjust the locally optimal flows which result from interpolation so that they are more nearly optimal in the global problem.

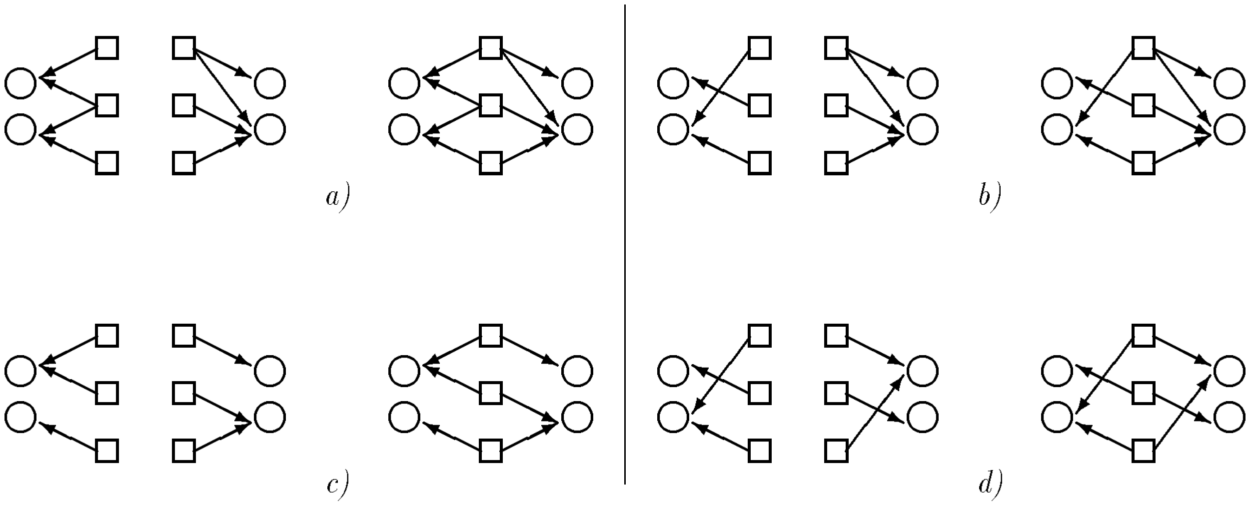

Figure 5: *Combining two 3×2 solutions into a 3×4 solution. Four typical cases are shown.*

Cycles are detected in the algorithm using a *depth first search (DFS)*. The DFS proceeds as follows:

Algorithm 2 *Depth First Search*

1. *Initialize all nodes with DFS number 0, to indicate they have not yet been visited.*
2. *Start at any node. Assign this node a DFS number of 1, and define node 0 to be the predecessor of this node.*
3. *If any node adjacent to the current node has been previously visited, and has a DFS number lower than the predecessor of the current node, then the path from that node through the current node and back is a cycle. Stop DFS and call the cycle removal routine.*
4. *If no adjacent nodes have lower DFS numbers, then look for any adjacent nodes which have not been visited. If there are any unvisited adjacent nodes, identify the current node as the predecessor of the unvisited node, make the unvisited node the current node, and assign the current node a DFS number equal to the DFS number of its predecessor plus 1.*
5. *If there are no unvisited nodes adjacent to the current node, make the predecessor of the current node the current node. If the current node is node 0, stop. Otherwise, return to step 3.*

Once a cycle is detected, a cycle removal algorithm is used to adjust the flows. The technique is illustrated in Figure 6. The effect of a unit increase in flow in the clockwise direction around the cycle is determined by adding together the costs of the arcs whose flow increases and subtracting the cost of the arcs whose flow decreases. The change in objective function value per unit change in flow in one direction will be the negative of the change in the opposite direction. An example is shown in Figure 6, with the initial flow regime on the left, and the flow after cycle removal on the right. The supplies and demands are shown in the boxes and circles, while the numbers in parentheses above each arc give the cost and flow for that arc. For example, the cost c_{34} is 5, while

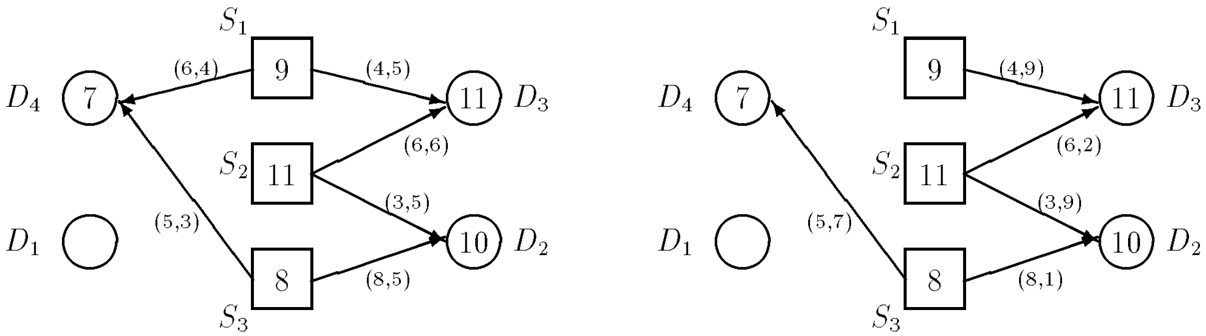

Figure 6: *An initial solution with a cycle (left), and the improved flow regime after cycle removal (right). The numbers in parentheses above each arc are* (c_{ij}, x_{ij}).

there is initially 3 units of flow on that arc. A unit increase in flow clockwise around the cycle (beginning at S_1) will cause a change in the objective function value of $4-6+3-8+5-6 = -4$, a net decrease. A unit increase in the counter-clockwise direction therefore yields 6, a net increase in the objective function. Clearly, increasing the clockwise flow is profitable, so flow is increased in this direction. Flow values will thus be increased for x_{13}, x_{22} and x_{34}, while flow is decreased for x_{23}, x_{32}, and x_{14}. That is, flow is increased on the arcs in the cycle which point in the profitable direction, and decreased on the the other arcs, until one of the decreasing arcs reaches zero flow. At this point, the cycle has been removed, and the value of the objective function has been decreased. In Figure 6, increasing the flow clockwise around the cycle by four units breaks the cycle by eliminating flow along x_{14}, and reduces the value of the objective function by 16 units. The improved flow regime is shown on the right side of the figure.

This technique is used as a local relaxation method by applying it to pairs of subproblems. Two subproblems which are adjacent in cost space are joined to form an $M \times 4$ problem, which is inspected for cycles. If any are found, they are removed and the problem is searched again.

Two different methods for applying this technique are investigated. The first, termed total relaxation, is to join adjacent pairs of $M \times 2$ problems, remove the cycles, then repeat the process by joining adjacent $M \times 4$ pairs, removing the cycles, then to join $M \times 8$ problems, and so on, until the global problem for the current level is inspected and certified to be cycle-free. This approach is, however, extremely expensive. The second approach only employs a local relaxation, and is therfore true to multigrid principles. In this case, only pairs of $M \times 2$ are checked for cycles. The gain in speed from using this second method is significant, while the decrease in accuracy is negligible (see Table 1 in the next section).

EXPERIMENTAL RESULTS AND CONCLUSIONS

The algorithm employed in this work is an FMG algorithm, using demand-weighting as the restriction method for computing costs, interpolation by local optimization, and local relaxation by cycle removal. The results are displayed in Table 1. While the multilevel algorithm performed well on problems with only two or three supply nodes, the results for the five supply node problem are unsatisfactory. The table clearly indicates that relaxation by cycle removal is as effective when applied over a local area as when applied globally, and the computational effort required for local relaxation is an order of magnitude smaller.

Problem Size	Relaxation Method	Run Time	% Above Optimality
2×1024	Total	1.210835	0.02 %
2×1024	Local	0.131437	0.02 %
3×1024	Total	1.15788	8.41 %
3×1024	Local	0.108765	8.41 %
5×1024	Total	1.18411	58.4 %
5×1024	Local	0.106392	59.7 %

We note also that this algorithm is not now competitive with the state of the art in network flow optimization methods. No numerical data are available as a careful comparison has not been made, however, some rough comparisons indicate that much remains to be done before a competitive algorithm could be obtained.

The most significant contribution of the current research is the removal of the requirement for a physical interpretation of the problem, and the dependence on a relationship between distance and shipping costs. By mapping the problem into cost-space, a multilevel approach can be applied to a much broader class of problems. Of course, there is a limit to the number of supply nodes which this approach can handle, due to the increasing dimensionality of the problem. However, for problems with few supply nodes, this approach can be helpful. We predict that further work will yield the result that problems which have either a very small number of supply nodes, or a geometrical interpretation, can be solved to within an acceptable degree of optimality using a multilevel approach. However, problems which do not meet either of these criteria probably cannot be solved with currently known multilevel methods.

Further Research

An algorithm analogous to the full approximation scheme (FAS) should be developed. In the current work, we were unable to find an effective method of extracting a *correction* from the solution on Ω^{2h} and applying it to the approximation on Ω^h, while still maintaining feasibility. Instead, we compute the solution on Ω^{2h} and use interpolation to replace the solution on Ω^h. Since a direct analog to the residual in a PDE is unknown for in an optimization problem, FAS is likely the method of choice, however, the difficulty mentioned above must be overcome.

Another possibility for improving this algorithm is to begin the procedure by overlaying the cost-space with a regular M-dimensional grid. The first step of the restriction process would then be to map the demand nodes from their natural irregularly spaced positions in cost-space to the regular grid points. Later, the final interpolation step would be to transfer from the regular grid back to the original demand points. This approach overcomes a shortcoming in the current algorithm, which aggregates demand nodes which are closest in relative distance in cost-space, regardless of the absolute distance between them. In using a regular grid, a demand node on Ω^{2h} would reflect only the demand at nodes a distance of $2h$ or less away from it. Another important potential advantage is that the work on each coarser level is reduced by 2^{-M}, instead of by one half as in the current research. If the regular grid approach proves worthwhile, then it could be extended to a fast adaptive composite (FAC) grid approach. In a network optimization setting, this might be done by overlaying a fine grid on those regions of cost-space where the density of demand nodes is high, and a coarser grid on the areas of low density. In this way, the flow to nodes which are most similar to their nearest neighbors in cost-space will receive the benefit of a finer grid spacing, while nodes which are naturally more distinct from their neighbors will only enter the problem on the coarser levels.

REFERENCES

[1] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network flows.* John Wiley and Sons, 1990.

[2] Dorit Ron. *Development of Fast Numerical Solvers for Problems in Optimization and Statistical Mechanics.* PhD thesis, Weizmann Institute of Science, 1987.

[3] Achi Brandt, Dorit Ron, and D. J. Amin. Multi-level approaches to discrete-state and stochastic problems. In W. Hackbusch and U. Trottenberg, editors, *Multigrid methods II (Proceedings, Cologne 1985)*, Lecture notes in mathematics 1228. Springer-Verlag, 1985.

[4] Ron Kaminsky. Multilevel solution of the long transportation problem. Master's thesis, Weizmann Institute of Science, 1986.

[5] B. Nilo. The transportation problem: a multi-level approach. Master's thesis, Weizmann Institute of Science, 1986.

[6] P. H. Zipkin. *Aggregation in linear programming.* PhD thesis, Yale University, 1977.

[7] P. H. Zipkin. Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(4):903–916, 1980.

[8] E. Balas. Solution of large-scale transportation problems through aggregation. *ORSA*, 1965.

[9] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* John Wiley and Sons, 1988.