The last three examples show that δ has no upper bound, although there is a lower bound, namely $\delta \geq 1/2\pi$, equality occurring if and only if the zone is a circle, the most compact shape. To see this, consider the following quadratic function of q:

$$\int_0^{2\pi} (q + R^2(\theta))^2 \, d\theta = 2\pi q^2 + 2q \int_0^{2\pi} R^2(\theta) \, d\theta + \int_0^{2\pi} R^4(\theta) \, d\theta .$$

As it is positive definite, the integrand on the left-hand side being a perfect square, the discriminant must be negative, so that

$$4\left[\int_0^{2\pi} R^2(\theta) \, d\theta\right]^2 - 4 \cdot 2\pi \int_0^{2\pi} R^4(\theta) \, d\theta \leq 0$$

from which the lower bound for δ follows immediately.

These examples show that it is not too difficult to calculate δ from the shape of the zone, but in practice it would probably be easier and sufficient to approximate it by a fixed value, say $\delta \approx 0.18$, which will be roughly true for all but the most extremely shaped zones and skewed population distributions. Putting together equations (2) and (4) and this last approximation means that a good approximation to use for the average distance between two zones is

$$\bar{d} \approx \sqrt{(\text{intercentroid distance})^2 + 0.18(\text{total area of the two zones})} .$$

In the special case when we are interested in the average intrazonal distance, the intercentroid distance is zero and we have

$$\bar{d} \approx \tfrac{3}{5}\sqrt{\text{area}} ,$$

where the 3/5 should be replaced by $\sqrt{2\delta}$ if the shape of the zone is known.

LITERATURE CITED

Bröcker, J. (1989). "How to Eliminate Certain Defects of the Potential Formula." *Environment and Planning A* 21, 817–30.

Rodriguez-Bachiller, A. (1983). "Errors in the Measurement of Spatial Distances between Discrete Regions." *Environment and Planning A* 15, 781–99.

The Hedetniemi Matrix Sum: An Algorithm for Shortest Path and Shortest Distance *by Sandra L. Arlinghaus, William C. Arlinghaus, and John D. Nystuen*

The geographic problem of finding the shortest distance between any two nodes in a real-world network of n nodes, given only distances between adjacent nodes (in the graph-theoretic sense [Harary 1969]), is traditionally resolved using Dijkstra's

Sandra L. Arlinghaus is director, Institute of Mathematical Geography (IMaGe). William C. Arlinghaus is associate professor and chairperson of mathematics and computer science at Lawrence Technological University. John D. Nystuen is professor of geography and urban planning, The University of Michigan.

Geographical Analysis, Vol. 22, No. 4 (October 1990)
Submitted 5/89. Revised version accepted 11/89.

algorithm (Dijkstra 1959). This algorithm in its simplest form finds the length of the shortest path between two nodes, and the complexity of this algorithm is $O(n^2)$ (Rosen 1988). Thus it would appear that finding the distances between each of the $n(n-1)/2$ pairs of nodes would be of complexity $O(n^4)$. However, with suitable modification, one use of Dijkstra's algorithm can be made to find not only the lengths of the paths from one node to all $(n-1)$ other nodes, but in such a way that the paths that yield these lengths can be reconstructed in $O(n)$ steps. The complete set of shortest distances can be obtained, therefore, in only n applications of Dijkstra's algorithm, reducing the complexity of the process to $O(n^3)$ (Hedetniemi 1977). Floyd's algorithm provides a strategy for determining the shortest distances between all pairs of points simultaneously, listing those distances in matrix form, and this algorithm is also of order $O(n^3)$ (Rosen 1988). Floyd's algorithm does not provide easy determination of the paths themselves. So Floyd provides easy display of lengths only, while Dijkstra is not designed for easy display of results but does make possible the determination of the actual position of shortest path. Both these algorithms require the same number of steps independent of the data. (Actually, Dijkstra's algorithm does not if only one pair of points is involved; but if the improvement to $O(n^3)$ for all pairs of points is desired, it does.)

The theoretical procedure presented below, based on the Hedetniemi matrix sum, displays in matrix form the set of shortest distances between all pairs of points in a network of n nodes. The process is of order $O(n^4)$. Like Floyd's algorithm, it provides a matrix showing simultaneously the shortest distances between all pairs of points in the network. Unlike Floyd's algorithm,

(1) the Hedetniemi Matrix Sum makes it possible to determine when to stop without examining all possibilities (in fact, the iteration-string is only as long as the number of links in the longest shortest path);
(2) this method does not require having the final matrix in hand to interpret the meaning of the succession of matrices leading to that final matrix; and
(3) the Hedetniemi method makes it clear how to use the final matrix and intervening matrices to determine the position of the shortest paths through the network.

In terms of complexity, the comparison between the Hedetniemi method and Floyd's method is parallel to that of the structural differences between "bubble" sort and "selection" sort. The former methods (Hedetniemi and bubble) require more work at a given stage than do the latter (Floyd and selection), but rate to have fewer stages than the latter to reach a final solution.

There are numerous applications in which a shortest path of some sort is desirable; readers wishing a review of the general literature might consider Rosen (1988), Hedetniemi (1977), or Lawler (1976); those wishing to see applications of theory within specific fields of study, as for example in urban geography, might consider references such as Shefli (1985). Often, shortest-path problems are NP-complete (Lawler 1976), as for example are both the traveling salesman problem and the Steiner problem, so that concern with how quickly an answer can be achieved is critical, especially in applications that need to reject large numbers of candidate shortest paths when there are large numbers of nodes. Different algorithms are well-suited to different applications.

As a theoretical source for geographic application, one might place the "pixel" in the role of the "point" or "node"; all are units fundamental to their environments —from electronic to Euclidean to graph-theoretic. Further, an electronic impulse from one location to another on a computer chip might play the role of "segment" or "edge" in corresponding environmental contexts. Thus, within a theoretical association that sees computer and graph-theoretic environments as parallel,

graph-theoretic shortest-path algorithms then have considerable potential for application to computerized technology. We suggest (to others) that one such technological geographic arena open to, and appropriate to, application of the theoretical material presented below is the so-called Geographic Information System: large numbers of pixels are colored and large numbers of decisions concerning these pixels, and the distances along the electronic paths linking them, are to be made (Burrough 1986). As a theoretical tool, the Hedetniemi operator might thus offer capability for geographical information storage comparable to that of the quadtree, insofar as both tools stop the process short once a solution has been reached, rather than exhausting all possibilities (Tobler and Chen 1986). The Hedetniemi operator does what Dijkstra and Floyd do, and it does so with the same theoretical advantage for information storage as the quadtree (which Dijkstra and Floyd do not). Investigation in this direction therefore appears desirable and is wide open.

HEDETNIEMI MATRIX SUMS

The idea for the following definition was proposed by Stephen Hedetniemi when he was a graduate student in mathematics at the University of Michigan (personal communication via Nystuen).

DEFINITION 1: *Hedetniemi Matrix Sum.* Consider two matrices

$$\mathbf{A} = \left[(a_{i,j}) | 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\right]$$

$$\mathbf{B} = \left[(b_{j,k}) | 1 \leqslant j \leqslant n, 1 \leqslant k \leqslant p\right]$$

where m, n, and p are all positive integers. Define the sum, denoted by $\mathbf{A} \dagger\!\dagger \mathbf{B}$ as:

$$\mathbf{A} \dagger\!\dagger \mathbf{B} = \begin{pmatrix} a_{11}a_{12}\ldots a_{1n} \\ a_{21}a_{22}\ldots a_{2n} \\ \ldots\ldots\ldots\ldots \\ a_{m1}a_{m2}\ldots a_{mn} \end{pmatrix} \dagger\!\dagger \begin{pmatrix} b_{11}b_{12}\ldots b_{1p} \\ b_{21}b_{22}\ldots b_{2p} \\ \ldots\ldots\ldots\ldots \\ b_{n1}b_{n2}\ldots b_{np} \end{pmatrix}$$

$$= \begin{pmatrix} c_{11}c_{12}\ldots c_{1p} \\ c_{21}c_{22}\ldots c_{2p} \\ \ldots\ldots\ldots\ldots \\ c_{m1}c_{m2}\ldots c_{mp} \end{pmatrix}$$

where

$$c_{ik} = \min(a_{i1} + b_{1k}, a_{i2} + b_{2k}, \ldots, a_{in} + b_{nk}).$$

This sum will be called the *Hedetniemi Matrix Sum* of the matrices **A** and **B**.

EXAMPLE. The Hedetniemi matrix sum of the matrices

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 4 \\ 5 & 6 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 & 3 & 4 \\ 5 & 0 & 6 \\ 1 & 2 & 0 \end{pmatrix}$$

is

$$\mathbf{A} \dagger\!\!\dagger \mathbf{B} = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 4 \\ 1 & 2 & 0 \end{pmatrix}.$$

This method of combining matrices is identical in global structure to the usual matrix multiplication; locally, however, like entries in combining rows and columns are added rather than multiplied, and the minimum of the resulting summands, rather than the sum of the resulting products, is taken.

Definition 2. Given two matrices **A** and **B** of dimensions $(m \times n)$ and $(n \times p)$, respectively. Let $a_{ij} \dagger\!\!\dagger b_{jk}$ denote the (i, k) element in the matrix $\mathbf{A} \dagger\!\!\dagger \mathbf{B}$ where $a_{ij} \dagger\!\!\dagger b_{jk} = \min(a_{i1} + b_{1k}, a_{i2} + b_{2k}, \ldots, a_{in} + b_{nk})$.

The Hedetniemi matrix sum is well-defined, at a most general level, for rectangular matrices of conformable dimensions; thus, it is well-defined for all square matrices, be they (nonsymmetric) matrices that represent distance-weighted adjacency patterns in a directed graph or symmetric matrices representing such patterns in an undirected graph. Because many applications concerning shortest paths assume a square distance-weighted adjacency matrix, we focus (here) only on the square matrices.

The Hedetniemi Matrix Sums Applied to Graphs of Networks

Given a graph representing a network of n nodes; in an $(n \times n)$ distance-weighted adjacency matrix $\mathbf{A} = [a_{ij}]$, label

a_{ij} = the distance from i to j if i is adjacent to j.

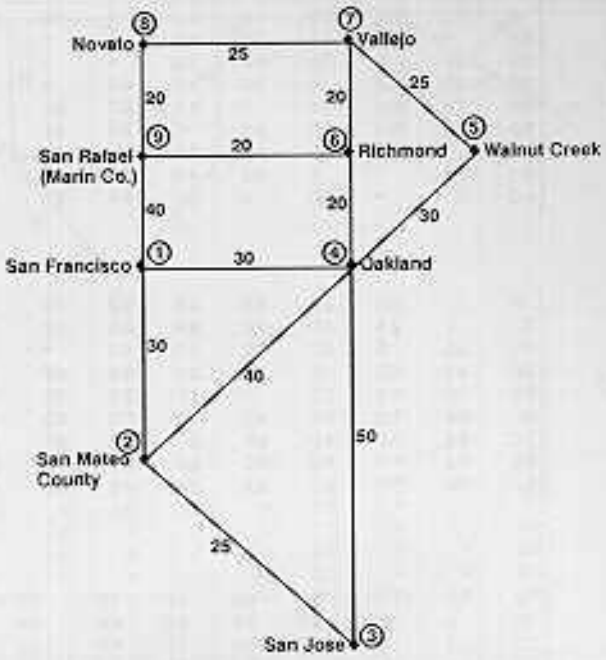
$a_{ii} = 0$

$a_{ij} = *$ if i is not adjacent to j (a "starred" entry).

We assign a distance of "0" from a location to itself. (In a computer program, a large number might be used in the data to represent "$*$".)

Example 1. *San Francisco Bay Area*. Consider a set of nine locations surrounding San Francisco Bay linked by major thoroughfares and bridges (Figure 1). The graph representing this network does not distinguish bridge-surface from road-surface; each is represented as an edge in the associated graph. Distances between the locations are measured in terms of minutes of time (Nystuen 1984). In the corresponding distance-weighted adjacency matrix, **A**, for this network (Figure 1), the a_{12} entry of 30 indicates that San Mateo County and San Francisco are adjacent and are thirty minutes apart, whereas the a_{13} entry of $*$ indicates that San Francisco and San Jose are not adjacent in the network originally given.

In the case of the San Francisco distance-weighted adjacency matrix A, when the Hedetniemi sum is applied successively, until all entries are "filled in," the first matrix in the iteration sequence, all of whose entries are identical to the matrix in the next stage, is the one that displays simultaneously all shortest distances between all possible origin-destination pairs in the network (Figure 2). Here, the fourth iteration provides the solution; fifth and higher iterations provide no new information. The length of the Hedetniemi iteration sequence is identical to the length of the longest shortest path in the example—in this case, four. In theory, as many as nine iterations might be required (the same as there are nodes in the graph representing the given network) (Arlinghaus and Nystuen 1989);



$$A = \begin{pmatrix} 0 & 30 & * & 30 & * & * & * & * & 40 \\ 30 & 0 & 25 & 40 & * & * & * & * & * \\ * & 25 & 0 & 50 & * & * & * & * & * \\ 30 & 40 & 50 & 0 & 30 & 20 & * & * & * \\ * & * & * & 30 & 0 & * & 25 & * & * \\ * & * & * & 20 & * & 0 & 20 & * & 20 \\ * & * & * & * & 25 & 20 & 0 & 25 & * \\ * & * & * & * & * & * & 25 & 0 & 20 \\ 40 & * & * & * & * & 20 & * & 20 & 0 \end{pmatrix}$$

FIG. 1. San Francisco Bay Area Graph of Time-Distances (in minutes)

applications of Floyd's algorithm always require the full set of iterations independent of the characteristics of the network under consideration (Rosen 1988).

The real-world bridges over the Bay do not stand out as graph-theoretic bridges; for this, and for other reasons, each newly filled-in entry in the Hedetniemi iteration sequence was a "best-possible" entry. There was no improvement in any of the non-"starred" values as one moved through the powering sequence. An abstract example will show that this need not be the case.

EXAMPLE 2. *A Case with a Graph-Theoretic Bridge.* Consider a graph of nine nodes and one graph-theoretic bridge linking two graph-theoretic components (Figure 3). The corresponding distance-weighted adjacency matrix reflects the presence of the bridge and the two components it hooks together: entries on or near the main diagonal hook a block of entries in the upper left-hand corner to a block of entries in the lower right-hand corner. The a_{13} entry in A is 70; when the Hedetniemi sum is applied to obtain A^2 from A, Definition 2 shows that the (1, 3) entry of A^2 is $\min(a_{31} + a_{11}, a_{32} + a_{21}, \ldots, a_{39} + a_{91}) = \min(70 + 0, 20 + 20, 0 + 70, * + *, 15 + *, * + *, * + *, * + *, * + *) = 40$ so that the (1, 3) entry

$$A^2 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & * & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & * & * & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & * & * & * \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & * & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & * \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ * & * & * & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & * & * & * & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & * & 40 & * & 20 & 40 & 20 & 0 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & 70 & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & 80 & 90 & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & 90 & * & 90 \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & 60 & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & 65 \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ 70 & 80 & 90 & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & 90 & * & 60 & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & 90 & 40 & 65 & 20 & 40 & 20 & 0 \end{pmatrix}$$

$$A^4 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & 70 & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & 80 & 90 & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & 90 & 110 & 90 \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & 60 & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & 65 \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ 70 & 80 & 90 & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & 90 & 110 & 60 & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & 90 & 40 & 65 & 20 & 40 & 20 & 0 \end{pmatrix}$$

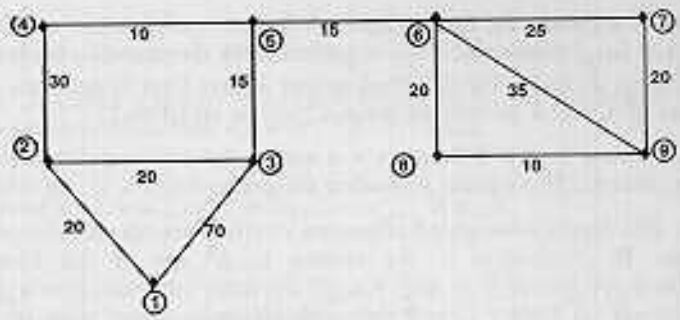
$$A^5 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & 70 & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & 80 & 90 & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & 90 & 110 & 90 \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & 60 & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & 65 \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ 70 & 80 & 90 & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & 90 & 110 & 60 & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & 90 & 40 & 65 & 20 & 40 & 20 & 0 \end{pmatrix}$$

$$A^4 = A^5 = \ldots = A^9$$

FIG. 2. Hedetniemi Iteration Sequence to Find the Set of All Possible Shortest Distances in the San Francisco Bay Area Network

"improves" from the non-starred value of 70 to a value of 40 (Figure 4). This raises the question as to whether or not the entries in the first matrix of the iteration sequence for which all entries are identical to those in the next stage are in fact lengths of shortest paths, or whether some further improvement might come in a later stage. We address such theoretical issues in material yet to come.

The authors have written a simple PASCAL program to implement this algorithm. In a simple program written independent of any system in which the algorithm is embedded, all powers A^i were stored in main memory. However, only three are needed at any one time (the initial matrix, the power just computed, and the next power to be computed). Such a program should compute powers until two in a row



$$A = \begin{pmatrix} 0 & 20 & 70 & * & * & * & * & * & * \\ 20 & 0 & 20 & 30 & * & * & * & * & * \\ 70 & 20 & 0 & * & 15 & * & * & * & * \\ * & 30 & * & 0 & 10 & * & * & * & * \\ * & * & 15 & 10 & 0 & 15 & * & * & * \\ * & * & * & * & 15 & 0 & 25 & 20 & 35 \\ * & * & * & * & * & 25 & 0 & * & 20 \\ * & * & * & * & * & 20 & * & 0 & 10 \\ * & * & * & * & * & 35 & 20 & 10 & 0 \end{pmatrix}$$

FIG. 3. Abstract Graph and its Associated Distance-weighted Adjacency Matrix A

$$A^2 = \begin{pmatrix} 0 & 20 & 40 & 50 & 85 & * & * & * & * \\ 20 & 0 & 20 & 30 & 35 & * & * & * & * \\ 40 & 20 & 0 & 25 & 15 & 30 & * & * & * \\ 50 & 30 & 25 & 0 & 10 & 25 & * & * & * \\ 85 & 35 & 15 & 10 & 0 & 15 & 40 & 35 & 50 \\ * & * & 30 & 25 & 15 & 0 & 25 & 20 & 30 \\ * & * & * & 50 & 40 & 25 & 0 & 30 & 20 \\ * & * & * & 45 & 35 & 20 & 30 & 0 & 10 \\ * & * & * & 55 & 45 & 30 & 20 & 10 & 0 \end{pmatrix}$$

$$A^7 = A^6 = \begin{pmatrix} 0 & 20 & 40 & 50 & 55 & 70 & 95 & 90 & 100 \\ 20 & 0 & 20 & 30 & 35 & 50 & 75 & 70 & 80 \\ 40 & 20 & 0 & 25 & 15 & 30 & 55 & 50 & 60 \\ 50 & 30 & 25 & 0 & 10 & 25 & 50 & 45 & 55 \\ 55 & 35 & 15 & 10 & 0 & 15 & 40 & 35 & 45 \\ 70 & 50 & 30 & 25 & 15 & 0 & 25 & 20 & 30 \\ 95 & 75 & 55 & 50 & 40 & 25 & 0 & 30 & 20 \\ 90 & 70 & 50 & 45 & 35 & 20 & 30 & 0 & 10 \\ 100 & 80 & 60 & 55 & 45 & 30 & 20 & 10 & 0 \end{pmatrix}$$

FIG. 4. Hedetniemi Matrix Iteration Sequence Showing Improvement in Nonstarred Entries

are identical, be able to display any power, and be able to display the length of the longest shortest path (that smallest t for which $A^t = A^{t+1}$).

THEORETICAL ISSUES ASSOCIATED WITH THE HEDETNIEMI MATRIX SUM

We now turn to a more formal statement of the ideas above that were set forth as motivation for the following theorems.

The set of theorems below expresses results based on the following hypotheses. In a simple (no multiple edges linking pairs of points) connected graph representing

a network of n points, $P_1, P_2, \ldots, P_n$, form the $(n \times n)$ distance-weighted adjacency matrix $A = [a_{ij}]$ where the entry a_{ij} represents the distance (weight assigned to an edge) from P_i to P_j. Use the Hedetniemi Matrix Sum to generate a sequence A^n of powers of A, for n an integer greater than or equal to 1.

THEOREM 1. *Let* A *be an* $n \times n$ *matrix representing distances in a network of* n *vertices. There exists a smallest integral value,* $t \leqslant n$, *for which* $A^t = A^{t+1}$.

Proof. The distance-weighted adjacency matrix A counts shortest paths traversing one edge. By Definition 2, the entries in A^2 are of the form $a_{ij} \# a_{jk} = \min(a_{i1} + a_{1k}, a_{i2} + a_{2k}, \ldots, a_{in} + a_{nk})$. An entry such as $a_{in} + a_{nk}$ represents a path of length $\leqslant 2$ from P_i to P_k through an intermediate point of P_n. The set in braces represents all possible paths of $\leqslant 2$ links from P_i to P_k (through all possible intermediate vertices) and because the minimum is chosen from this set, the shortest path of $\leqslant 2$ links is selected (although a shorter one with more links may exist). Thus, some of the starred entries of A are filled in by squaring A. Subsequent powering of A fills in more of these and may improve previously entered nonstarred values. In fact, the (i, j) entry of A^r gives the length of the shortest path of length $\leqslant r$ from i to j. Since no shortest path is of length greater than $n - 1$, $A^{n-1} = A^n = \ldots$. The smallest power of A that produces a matrix identical to that of the next highest power will be denoted as "t", the smallest value for which $A^t = A^{t+1}$. Q.E.D.

Theorem 1 is an existence theorem; it does not address the issue of whether any improvement can be found once $A^t = A^{t+1}$ (an issue suggested by Example 2 above).

THEOREM 2. *Let* A *be an* $n \times n$ *matrix representing distances in a network of* n *vertices. If* $A^t = A^{t+1}$, *then the* (i, j) *entry of* A^t *is the length of the shortest path from* i *to* j.

Proof. Since $A^t = A^{t+1}$, then $A^t = A^{t+1} = \ldots = A^n = \ldots$. But by Theorem 1, A^n clearly gives all shortest paths. Q.E.D.

Thus, A^t does in fact give all shortest paths and no improvement is available by subsequent powering in the iteration sequence.

COROLLARY 1. *The length of the path of shortest distance between* P_i *and* P_j *traverses no more than* t *edges of the network.*

This is clear from the discussion of how starred entries get replaced by numerical values through the Hedetniemi iteration process.

COROLLARY 2. *If any distance of* $a_{ij} = *$ *remains in the matrix* $A^t (= A^{t+1})$, *then the corresponding graph is disconnected and no paths link* P_i *to* P_j.

For disconnected graphs these theorems might be applied in each component and the components linked by graph-theoretic bridges (as in Example 2). For multigraphs, one approach might be to collapse the multigraphs to graphs or digraphs with the minimum weight selected for each edge.

One of the nice features of this algorithm is that at stage r, the shortest paths of length $\leqslant r$ are exhibited. In Floyd's algorithm, no intermediate calculations have immediate physical interpretation. Thus tracing the actual shortest path is possible in the Hedetniemi algorithm, as shown below.

FINDING THE INTERVENING POINTS IN A PATH OF SHORTEST DISTANCE

To find the actual route of the shortest path between vertices P_i and P_j, find the (i, j) entry in the matrix A^t. For example, the most distant node (in Figure 1) from San Francisco (node P_1) is Vallejo (node P_7). The travel time is seventy minutes.

To find the route of the shortest path between these two locations, proceed as follows. Find the (1, 7) entry in A^t according to Definition 3.

DEFINITION 3. Given t copies of an $(n \times n)$ matrix A. Let a^t_{ij} denote the (i, j) element of the matrix A^t so that $a^t_{ij} = a^{t-1}_{ik} \dagger a_{kj}$. Thus,

$$a^t_{ij} = \min\left(a^{t-1}_{i1} + a_{1j}, a^{t-1}_{i2} + a_{2j}, \ldots, a^{t-1}_{in} + a_{nj}\right).$$

In Example 1, referring to Figure 2, we see that in this case the length of the longest shortest path was $4(A^4 = A^5)$. So analysis of shortest paths from i to j begins with the value of a^4_{ij}. For example, $a^4_{17} = a^3_{1k} \dagger a_{k7}$. Now each of these summands represents a vector of the Hedetniemi matrix, and further, it is required that the sum of the components in this case be equal to 70. Here,

$a^3_{1k} = [0, 30, 55, 30, 60, 50, 70, 60, 40]$—first row vector of A^3;

$a_{k7} = [*, *, *, *, 25, 20, 0, 25, *]$—seventh column vector of A. When $k \neq 7$, the only value of k for which the sum of the components is 70 is $k = 6$, for which $50 + 20 = 70$. Thus, $P_1 \rightarrow \ldots \rightarrow P_6 \rightarrow P_7$.

To find the vertex to which P_6 is adjacent along the shortest path from P_1 to P_7, repeat the process using $a^4_{16} = a^3_{1k} \dagger a_{k6}$. Here $a^4_{16} = 50$, and

$a^3_{1k} = [0, 30, 55, 30, 60, 50, 70, 60, 40]$—first row vector of A^3;

$a_{k6} = [*, *, *, 20, *, 0, 20, *, 20]$—sixth column vector of A. The only value of $k(k \neq 6)$ for which the sum of the components is 50 is $k = 4$, for which $30 + 20 = 50$. Thus, $P_1 \rightarrow \ldots \rightarrow P_4 \rightarrow P_6 \rightarrow P_7$.

To find the vertex to which P_4 is adjacent along the shortest path from P_1 to P_7, repeat the process using $a^4_{14} = a^3_{1k} \dagger a_{k4}$. Here $a^4_{14} = 30$, and

$a^3_{1k} = [0, 30, 55, 30, 60, 50, 70, 60, 40]$—first row vector of A^3;

$a_{k4} = [30, 40, 50, 0, 30, 20, *, *, *]$—fourth column vector of A. The only value of $k(k \neq 4)$ for which the sum of the components is 30 is $k = 1$, for which $0 + 30 = 30$. Thus, the trace of the path is completed, back to P_1, and $P_1 \rightarrow P_4 \rightarrow P_6 \rightarrow P_7$. Only one value of k was determined at each iteration and therefore the shortest path from P_1 to P_7 is unique. If more than one value of k had been determined at any point, each would have been traced back to P_1 to determine the positions of multiple routes of shortest distance.

DIRECTIONS FOR FURTHER RESEARCH

1. Because shorter paths are found successively as one calculates the Hedetniemi powers, it may be possible to find the actual shortest path, by saving intermediate points, during the computation process (as is done in Dijkstra's algorithm). Such an algorithm would be useful.

2. The authors' program dealt only with integer entries. If real number entries were employed, one might expect measurement error to become a significant issue in real-world applications. A perfectly square urban street grid might be rendered as a wobbly network of nodes and edges in which error terms have undue influence on path specification. Thus, the size of acceptable error terms would need to be specified in order that the computer recognize actual alternate shortest paths (to some specified level of map resolution).

3. The main diagonal entries in the distance-weighted adjacency matrix were always regarded as zero. Assigning entries on the main diagonal a value of "*" would measure twice the distance of a vertex to its nearest neighbor. The modification of the program to tolerate nonzero main diagonal entries might serve, therefore, as a link between studies using the Hedetniemi algorithm and those employing nearest-neighbor analysis.

4. This algorithm, and also Dijkstra's, Floyd's, and numerous others, are substantially more efficient on smaller networks than on larger networks. Consider, for example, a set of five widely spread and sparsely connected major U.S. cities and five densely connected suburbs of each. To find the set of all distances between all possible pairs of them would have order of complexity $O((6 \cdot 5)^4) = O(30^4)$. When distances are measured, instead, only between each of the five major cities and between each major city and its own suburbs, a substantial reduction in complexity results: here the order of complexity is $O(5^4) + 5 \cdot O(6^4) \leq O(6^4) + 5 \cdot O(6^4) = 6 \cdot O(6^4) = O(6^5)$, which is more than one hundred times $((30^4/6^5) > 100)$ as fast as finding all distances.

Thus, if the geography of a given situation can be used to subdivide the data set into smaller data sets, to each of which the algorithm is to be applied, the computational task becomes much easier. In the five-cities case, geographical and computational issues have comparable scale problems associated with them: denseness and sparseness of geographic connectivity correspond to denseness and sparseness of matrix entries.

5. When large numbers of computations can clearly be eliminated by hand (the computer will calculate them unless a method is used to eliminate them) as in the multiplication of sparse matrices, the corresponding benefit in reduction of computational complexity is clear. Spatially, this reduction might be accomplished in a variety of ways. Suppose a path links nodes in an "origin" city to those in a "destination" city. The requirement that these cities be linked by a long-distance graph-theoretic "bridge" (such as an expressway with suitable exits in origin and destination cities) gives spatial expression to the notion of matrix "sparseness." The challenge therefore remains to determine geographic bases from which to select real-world criteria that eliminate appropriate subsets of data.

LITERATURE CITED

Arlinghaus, S., and J. Nystuen (1989). Poster. Association of American Geographers, National Meetings, Baltimore, March 22, 1989.

Burrough, P. A. (1986). *Principles of Geographical Information Systems for Land Resources Assessment*. Oxford Science Publications, Monographs on Soil and Resources Survey No. 12. Oxford: Clarendon Press.

Dijkstra, E. (1959). "Two Problems in Connexion with Graphs." *Numerische Mathematik* 1, 269–71.

Harary, F. (1969). *Graph Theory*. Reading: Addison-Wesley.

Hedetniemi, S. T. (1977). *Introduction to the Design and Analysis of Algorithms*. New York: McGraw-Hill.

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston.

Nystuen, J. (1984). Lecture Notes (mimeo), Urban Planning 504, The University of Michigan, College of Architecture and Urban Planning. Available on request from the author.

Rosen, K. H. (1988). *Discrete Mathematics and Its Applications*. New York: Random House.

Sheffi, Y. (1985). *Urban Transportation Networks*. Englewood Cliffs, N.J.: Prentice-Hall.

Tobler, W., and Chen, Z-t. (1986). "A Quadtree for Global Information Storage." *Geographical Analysis* 18, 360–71.