Intern. J. Computer Math., Vol. 34, pp. 171–176
Reprints available directly from the publisher
Photocopying permitted by license only

AN ALGORITHM FOR TREE-REALIZABILITY OF DISTANCE MATRICES*

VLADIMIR BATAGELJ and TOMAŽ PISANSKI†
University of Ljubljana, Yugoslavia

J. M. S. SIMÕES-PEREIRA
University of Coimbra, Portugal

(*Received 4 September 1989; in final form 7 January 1990*)

An algorithm for testing tree realizability of distance matrices is given. It is well-known that if a matrix D has a realization by a tree then such a realization is optimal and unique up to homeomorphism. Our algorithm produces a tree realization or a message that there is no such realization in time $O(n^2)$ where n is the number of points in a finite metric space with the distance matrix D. An $O(n^2)$ algorithm for computing distance matrix for a given tree is also given.

KEY WORDS Distance matrix, tree-realizability, complexity

C.R. CATEGORIES: E.1, F.2.2, G.2.2.

Math. Subj. Class. (1985): 05 C

1. INTRODUCTION

The aim of this note is to present a fast algorithm to check whether a distance matrix is tree-realizable and, if it is, to construct such a realization. Tree realizations of distance matrices, when they exist, are optimal and unique [3, 4].

Tree realizable distance matrices were characterized in [5] and, for the integer case, in [9]. From an algorithmic point of view, the time complexity of such a characterization is quadratic in n where n is the order of the matrix.

Graph realizations of distance matrices and, in particular, tree realizations, have spurred a lot of research; the topic has applications in many areas of pure and applied mathematics as the sample of papers we quote [3, 4, 5, 6, 7, 8, 9] clearly shows.

We recall that a distance matrix of order n, denoted D, is a nonnegative, symmetric, square matrix whose entries d_{ij} are such that, for $i,j,k \in \{1,2,\ldots,n\}$, $d_{ii}=0$, $d_{ij}\neq 0$ for $i\neq j$, and $d_{ij}\leqq d_{ik}+d_{kj}$. A graph $G=(W,E)$ with lengths assigned to its edges *realizes* D if there is a subset V of the vertex set W of G, with $|V|=n$ and

*This paper was presented at the SIAM Conference on Discrete Mathematics in San Francisco, May 1988.

†Supported in part by the Research Council of Slovenia, Yugoslavia.

such that, for $i,j \in V$ the length $d(i,j)$ of the shortest path between i and j equals d_{ij}.

An optimal graph realization is one with total length minimum among all realizations. Vertices in V are called *main*, those in $W \setminus V$, *auxiliary*.

When we subtract from the nondiagonal entries in the ith row and in the ith column of D the nonnegative, real number a, we obtain a new matrix, written $D_i(a)$, which is also a distance matrix when

$$a \leqq a_{\min}(i) = \min_{p,r \in \{1,2,\ldots,n\} \setminus \{i\}} \{\tfrac{1}{2}(d_{pi} + d_{ir} - d_{pr})\}$$

The operation which obtains $D_i := D_i(a_{\min}(i))$ from D is called a *compactification*. When, for some $k(\neq i)$, $a_{\min}(i) = d_{ik}$, then, by the minimality of $a_{\min}(i)$, we obtain, for every $q \neq i$,

$$d_{ik} + d_{iq} - d_{kq} \geqq d_{ik} + d_{ik} - d_{kk} = 2d_{ik}$$

hence $d_{iq} \geqq d_{kq} + d_{ik}$, hence $d_{iq} = d_{ik} + d_{kq}$; in such case we say that i is pendant from k in D. To avoid nondiagonal zero entries, we then remove duplicated parallel lines in D_i, thereby decreasing by one the order of the matrix. This operation is called *reduction*.

We know [6, 8] how to obtain an optimal realization G of D from an optimal realization G' of D_i:

If D_i is a compactification of D, the vertex i of G' becomes auxiliary, is renamed and linked by a new (pendant) edge of length $a_{\min}(i)$ to a new main vertex i;

If D_i is a reduction of D, the main vertex k of G' is linked by a new (pendant) edge of length $a_{\min}(i)$ to a new main vertex i. It follows immediately from the definitions that D is tree-realizable if and only if it yields the trivial matrix [0] of order 1 by successive compactifications and reductions.

2. THE ALGORITHM

2.1 Data Structures

D = Initially an n by n distance matrix for which we have to test tree-realizability. At the end it is expanded to an m by m distance matrix of a tree with $(m-n)$ auxiliary vertices if the test turns out positive.

Tr = An m array that we build gradually and in which we store the tree structure. For instance, vertex i is adjacent to vertex $Tr[i]$.

2.2 Comments

The basic step of the algorithm takes a vertex k which is not yet a part of the tree Tr and attaches the vertex to the partially built tree Tr. There are six possible outcomes of this step.

a) D turns out to be nontree-realizable.

b) k becomes a leaf in the tree and is attached to an existing vertex a of the tree.

c) Same as case (b) but the distance $D[k,a]$ turns out to be 0. We do not add vertex k.

d) k becomes a leaf in the tree and is attached to an edge, say from a to b. To this end the edge is subdivided and k is in fact attached to the newly constructed auxiliary vertex c. In this case D is expanded for another row and column that correspond to the vertex c.

e) Same as case (d) but it turns out that c already exists. It belongs to the part of original D that has not been examined yet.

f) Same as (d) but it turns out that $D[k,c]=0$. There is no need to introduce an auxiliary vertex.

Essentially we are performing inverse operations to compactifications and reductions.

2.3 Outline of the Algorithm

1. $O(1)$ Initialize. The tree Tr contains a single vertex 1. Let $Tr[1]:=1$. Set up a stack *link* of vertices in the tree. Let $m:=n$. The vertices will only be added to (and never removed from) the stack. The last entry in the stack is the *root*.
2. $O(n)$ **For** $k=2,3,\ldots,n$ **repeat**
2.1. $O(m)$ **For each** vertex a from the stack *link* **repeat**
2.1.1. $O(1)$ Let $b=Tr[a]$. Let $x:=(D[a,b]+D[b,k]-D[a,k])/2$. The value x tells us where on the edge from a to b (at the distance x from b) we have to attach our vertex k.
2.1.2. $O(1)$ **If** $x<0$ or $x>D[a,b]$ **then**
D is not a distance matrix. **Stop**.
2.1.3. $O(1)$ **If** $x\neq 0$ **Exit**
2.2. $O(1)$ **If** $0<x<D[a,b]$ **then**
2.2.1. $O(1)$ Let $p:=m+1$
2.2.2. $O(m)$ **For each** vertex $c:=1,\ldots,m$ **repeat**
2.2.2.1. $O(1)$ Let $D[p,c]:=(D[a,c]+D[c,b]-D[a,b]+$
$|D[a,k]+D[b,c]-D[k,b]-D[c,a]|/2$
2.2.2.2. $O(1)$ **If** $D[p,c]=0$ **then** Let $p:=c$; **Exit**
2.2.3. $O(1)$ **If** $p=m+1$ or $p=k$ **then**
we have to introduce a new vertex p that lies on the edge from a to b. Modify the tree Tr accordingly.
2.2.4. $O(1)$ **If** $p=m+1$ **then** Let $m:=m+1$
2.2.5. $O(1)$ Let $a:=p$.
2.3. $O(1)$ **If** $k\neq a$ **then** attach k to the vertex a. Modify the tree Tr.
3. Check whether augmented D is indeed the distance matrix for the tree Tr.

Note: If $x=0$ in the Step 2.1.3 it means that k has to be attached to the tree at or

above vertex b. The order in which the **For** loop is performed in Step 2.1. is important. Namely for each vertex $b(\neq root)$ of a tree the edge from b to $Tr[b]$ has to be examined only after all edges from a *to* $b=Tr[a]$ are examined.

3. THE TIME COMPLEXITY OF THE ALGORITHM

Let n be the size of original distance matrix and let m be the size of final distance matrix. This means that there are n main vertices and $m-n$ auxiliary vertices. It is easy to see that the time complexity of the algorithm is $O(n^2)$. We prove this as follows. We may assume that the matrix is tree realizable. If not the algorithm would terminate sooner and the worst case is not attained.

LEMMA *The number of auxiliary vertices is at most* $n-2$.

Proof Each auxiliary vertex is of degree at least 3. In a tree we have: $\#\text{edges}=m-1$. And we also have

$$2\ \#\text{edges}=\sum \text{degrees}\geqq 3(m-n)+n$$

Hence $2(m-1)\geqq 3(m-n)+n=3m-2n$ which is equivalent to $2n-2\geqq m$. Q.E.D.

This means that $O(n^k)=O(m^k)$ for each k, and in particular $O(m^2)=O(n^2)$. When we wrote the algorithm we already indicated time complexity for each partial step. Now we only have to combine them accordingly in order to determine the overall time complexity $Time=T(Step1)+T(Step2)+T(Step3)$.

$T(Step1)=O(1)$.
$T(Step2)=(n-1)(T(Step2.1)+T(Step2.2)+T(Step2.3))$
  $T(Step2.1)=O(m)(T(Step2.1.1)+T(Step2.1.2.)+T(Step2.1.3))$
    $T(Step2.1.1)=O(1)$
    $T(Step2.1.2.)=O(1)$
    $T(Step2.1.3)=O(1)$
$\Rightarrow$ $T(Step2.1)=O(m)$
  $T(Step2.2)=O(1)+$
    $T(Step2.2.1)+T(Step2.2.2)+T(Step2.2.3)+T(Step2.2.4)+T(Step2.2.5)$
    $T(Step2.2.1)=O(1)$
    $T(Step2.2.2)=O(m)$
    $T(Step2.2.3)=O(1)$
    $T(Step2.2.4)=O(1)$
    $T(Step2.2.5)=O(1)$
$\Rightarrow$ $T(Step2.2)=O(m)$
  $T(Step2.3)=O(1)$
$\Rightarrow T(Step2)=O(n)(O(m)+O(m)+O(1))=O(nm)=O(m^2)=O(n^2)$
  $T(Step3)=O(n^2)$

Therefore $Time=O(1)+O(n^2)+O(n^2)=O(n^2)$. This shows that our algorithm is indeed quadratic.

We have made experimental tests of our algorithm. First we randomly generated a tree, with a given number of vertices, and its matrix. Then we calculated its distance matrix. Afterwards we selected a principal submatrix that we used as input to our algorithm. Clearly each matrix was tree-realizable. The experimental results also show the quadratic behaviour of the algorithm.

For testing purposes we developed a special algorithm which computes the distance matrix of a given tree. This algorithm is also quadratic and therefore more efficient than standard algorithms for general graphs.

4. AN ALGORITHM FOR COMPUTING DISTANCE MATRIX FOR A GIVEN TREE

Given a tree Tr on n vertices with distances on edges already stored in matrix D we compute distance matrix D in time $O(n^2)$. The distance matrix D is symmetric ($D[i,j]=D(j,i]$).

It is assumed that $i=Tr[i]$ if and only if $i=root$. We are using an auxiliary *level* array $L[i]$ that represents the number of vertices on the path towards the root of the tree ($L[root]=1$). Initially all $L[i]$ are undefined. We need a recursive function $Lv(i)$ with side effect for calculating $L[i]$; and another recursive function $dist(i,j)$ ($=dist(j,i)$) for calculating $D[i,j]$:

$$Lv(i)=L[i]=\begin{array}{ll} L[i] & L[i] \text{ is defined} \\ 1 & i=root \\ 1+Lv(Tr[i]) & \text{otherwise} \end{array}$$

$$dist(i,j)=D[i,j]=\begin{array}{ll} D[i,j] & D[i,j] \text{ is defined} \\ D[i,Tr[i]]+dist(Tr[i],j) & Lv(i)>Lv(j) \\ dist(i,Tr[i])+D[Tr[i],j] & \text{otherwise} \end{array}$$

The algorithm is now simple

For $i=1,2,\ldots,n$ **repeat**
  For $j=i+1,2,\ldots,n$ **repeat** $D[i,j]:=dist(i,j)$

Note: Just before submitting this manuscript for publication we learned that J. C. Culberson and P. Rudnicki [2] obtained quite similar results. They also attribute the first $O(n^2)$ algorithm for tree-realizability to F. T. Boesch [1].

References

[1] F. T. Boesch, Properties of the distance matrix of a tree. *Q. Appl. Math.* **26** (1968), 607–609.

[2] J. C. Culberson and P. Rudnicki, A fast algorithm for constructing trees from distance matrices. *Information Processing Letters* **30** (1989), 215–220.

[3] S. L. Hakimi and S. S. Yau, Distance matrix of a graph and its realizability, *Quart. Appl. Math.* **22** (1964–65), 305–317.

[4] W. Imrich, J. M. S. Simões-Pereira and C. M. Zamfirescu, On optimal embeddings of metrics in graphs, *J. Combin. Theory Ser. B* **36** (1984), 1–15.

[5] J. M. S. Simões-Pereira, A. note on the tree realizability of a distance matrix, *J. Combin. Theory* **6** (1969), 303–310.
[6] J. M. S. Simões-Pereira, A note on optimal and suboptimal digraph realizations of quasidistance matrices, *SIAM J. Algebraic and Discrete Methods* **5** (1984), 117–132.
[7] J. M. S. Simões-Pereira, A note on distance matrices with unicyclic graph realizations, *Discrete Math.* **65** (1987), 277–287.
[8] J. M. S. Simões-Pereira and C. M. Zamfirescu, Submatrices of nontree-realizable distance matrices, *Linear Algebra Appl.* **44** (1982), 1–17.
[9] K. A. Zaretzkii, Constructing a tree on the basis of a set of distances between the hanging vertices, *Uspekhi Mat. Nauk.* **20** (1965), 90–92, in Russian.