To the Graduate Council:

I am submitting herewith a thesis written by Abderauf I. M. Taha entitled "Computational Algorithms for Evolutionary Trees and Related Applications." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Steven M. Serbin

Steven M. Serbin, Major Professor

We have read this thesis
and recommend its acceptance:

Linda H. Turner

Chung-Ping Hwang

Accepted for the Council:

Hilton A. Smith

Vice Chancellor
Graduate Studies and Research

COMPUTATIONAL ALGORITHMS FOR EVOLUTIONARY TREES AND RELATED APPLICATIONS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee

Abderauf I. M. Taha

August 1974

ACKNOWLEDGMENTS

The author wishes to express his gratefulness to Dr. Steven M. Serbin for his assistance by directing the preparation of this thesis.

The author also would like to thank Dr. Charles H. Nelson, Professor of Biology at The University of Tennessee at Chattanooga, who furnished the real data for his own study of the Genus Perlomya to apply to the algorithms used for computing the evolutionary tree.

A special appreciation to the Data Processing Staff of the Knoxville Utilities Board for providing the computer time to compile the programs.

1165753

ABSTRACT

The purpose of this study is to use the notion of trees of graph theory in some useful applications to solve some problems in evolution, mathematics, and the problem of minimal connector.

The Wagner method developed by Farris (1970) is used as a basis for the study. Computer programs written in FORTRAN by the author implement the method to construct the evolutionary tree for a group of organisms and show the relation between the members of the group.

Modification is made to the FORTRAN programs along with additional subroutines to apply to the tree of the Greatest Common Divisor and the Least Common Multiple of a set of numbers.

Another modification is made to solve the problem of minimal connector in general and the traveling salesman in particular.

A separate program is developed to implement Sollin's algorithm for the problem of minimal connector.

Final results are shown in the form of trees in the text, while instructions are given in the documentation to construct trees for these results from the programs' output.

TABLE OF CONTENTS

CHAPTER	PAGE
I. CONCEPTS FROM GRAPH THEORY	1
Introduction	1
Definitions	1
Trees	6
II. EVOLUTIONARY TREES	11
Introduction	11
Rates of Evolution	11
Trees	13
Methods for Computing Wagner Trees	15
Character analysis and representation	15
Representation of trees	16
Length of a Tree	18
Wagner Method for Constructing Evolutionary Trees: Algorithm and Procedure	20
Rootless Wagner Method: Algorithm and Procedure	25
Wiss Method for Constructing Evolutionary Trees	29
III. GENERAL APPLICATIONS	34
The Greatest Common Divisor (GCD)	34
Method	34
Explanation of the method	36
The Least Common Multiple (LCM)	37
The "Interval Difference" Meaning in Applications	40
Problem of Minimal Connector	42
Kruskal's algorithm	43
Sollin's algorithm	46
Programming the problem of the minimal connector for the computer	48
BIBLIOGRAPHY	52
APPENDIXES	54
Appendix A	55
Appendix B	61
Appendix C	67
Appendix D	73
Appendix E	74
Appendix F	80
VITA	82

LIST OF TABLES

TABLE		PAGE
1.	Data Matrix for Species and Characters for Wagner Method	20
2.	Constructing the Ancestor Z for Species C and E	22
3.	Constructing the Ancestor (Y) for Species D and C	23
4.	Data Matrix for Species and Characters	25
5.	Constructing the Ancestor Z for Species A and B	27
6.	Data Matrix for Species and Characters for the Wiss Method	29
7.	Constructing the Ancestor X for Species A and B	31
8.	Constructing the Ancestor Y for Species C and D	32
9.	Decomposition of a Set of Numbers to Its Prime Factors	35
10.	Decomposing the Numbers 36, 48, and 50	40
11.	Distance Matrix for the Cities A, B, C, D, E, and F	45

LIST OF FIGURES

FIGURE		PAGE
1.	Connected Graph	3
2.	Strongly Connected Graph	3
3.	Complete Graph	5
4.	Incomplete Graph	5
5.	Directed Tree	8
6.	Nondirected Tree	9
7.	The Relation Between the Dissimilarity and the Evolution Time	12
8.	Representation of Nodes and Ancestors in a Tree Graph	17
9.	Connecting a Node to Two Other Nodes Through an Ancestor	19
10.	Final Network of the Wagner Tree	24
11.	Final Network of the Rootless Wagner Tree	28
12.	Final Network of the Wiss Tree Showing the Advancement Index Measure	33
13.	The GCD Tree	36
14.	The LCM Tree for the Numbers 50, 48, and 36	39
15.	Final Network for the GCD Tree	42
16.	Distance Graph for the Cities A, B, C, D, E, and F	44
17.	The Shortest Distance Which Links the Cities A, B, C, D, E, and F	45
18.	Forming Subtrees	47
19.	The Final Optimal Tree	48

CHAPTER I

CONCEPTS FROM GRAPH THEORY

1. INTRODUCTION

The purpose of this study is to examine some applications of graph theory to applied science and to present certain algorithms and discuss their implementation for computers.

The object of the first chapter is to provide an intuitive background to the material to be presented in succeeding chapters. Definitions from graph theory are used in descriptive method, so the reader will not find them totally unfamiliar when he meets them later.

A simple framework of graph theory and some elementary concepts of trees will be covered in the first chapter to make it easy to deal with trees in later applications.

Evolutionary trees and their construction will be discussed in detail with an application to genetics. Some interesting different types of applications using the same methods which are used to construct evolutionary trees will be employed to solve different problems in Mathematics, Engineering, and Social Science. These methods will be programmed to be solved on computers.

2. DEFINITIONS

We shall begin by defining, following [2], a simple graph G to be a pair (V(G), E(G)), where V(G) is a non-empty finite set of

elements called vertices (or nodes, or points) and E(G) is a finite set of non-ordered pairs of distinct elements of V(G) called edges (or lines, or arcs). N(G) is sometimes called the vertex-set and E(G) the edge-set of G. For example, Figure 1 represents the simple graph of G whose vertex-set V(G) is the set {U,V,W,Z}, and whose edge-set E(G) consists of the pairs {U,V}, {V,W}, {U,W} and {W,Z}. The edge {V,W} is said to join the vertices V and W.

Before explaining various interesting applications of graph theory we will introduce some important concepts. To make our explanations sufficiently clear it will first be necessary to define a number of technical terms.

Path. A series of adjacent arcs giving access from one vertex to another. For example, in Figure 2 the edges AB, BE, EF form a path from A to F.

Circuit. A path in which the initial and final vertices coincide. In Figure 2 (B,E,F,B) is a circuit.

Length of path or circuit. The number of edges contained in the path or circuit. In Figure 2 the length of (B,E,F,B) is 3 and the length of (A,B,C) is 2.

Loop. A circuit of length 1. In Figure 2, H is a loop.

Symmetrical graph. If a vertex X is joined to a vertex Y, then Y should be joined to X. In other words it is a two-way street (path).

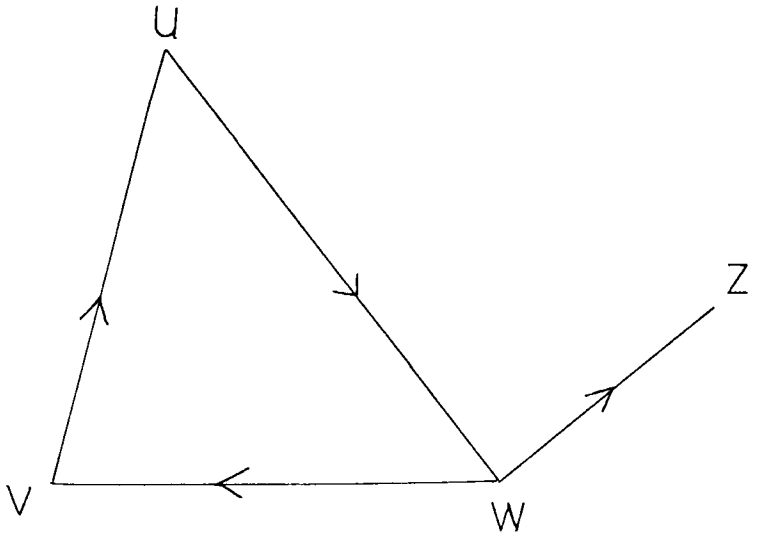


Figure 1. Connected graph.

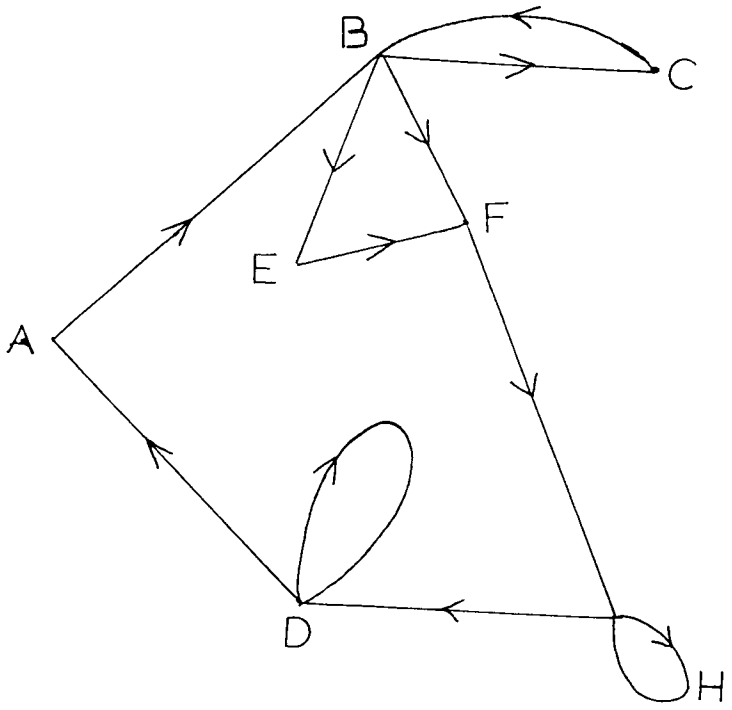


Figure 2. Strongly connected graph.

Directed graph. A graph in which all edges are one-way streets.

Degree of a vertex. The number of edges which have that vertex as an end-point.

Simple graph. A graph in which there is never more than one edge joining a given pair of vertices and which has no loops. For example, Figure 1 is a simple graph while Figure 2 is not.

Strongly connected graph. Whatever the position of the vertices X and Y which are being considered, there is a path between X and Y. For example, Figure 2 is strongly connected while Figure 1 is not, because there is no path from Z to W.

Link. A link exists between two vertices X and Y if there is an arc from X to Y and/or from Y to X.

Chain. A series of consecutive links.

Cycle. A closed chain.

Connected graph. Whatever the position of the vertices X and Y which are being considered, there is a chain between X and Y.

Complete graph. Whatever the position of X and Y which are being considered, there is a link between X and Y. For example, Figure 3 is a complete graph while Figure 4 is not, because there is no link between A and C.

Partial graph. If one or more arcs are omitted from a graph, a partial graph is formed (Figure 4).

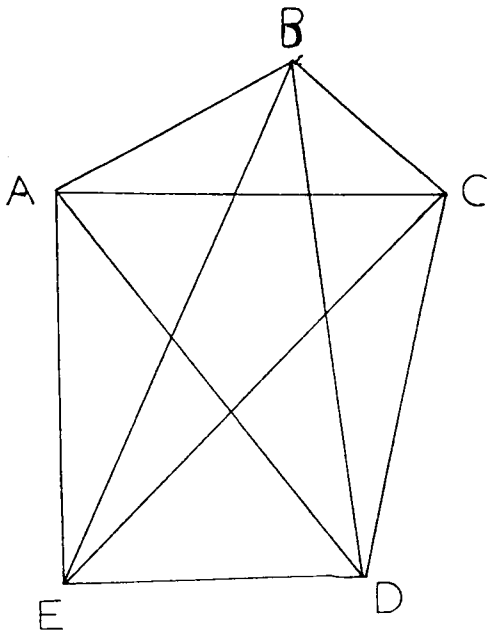


Figure 3. Complete graph.

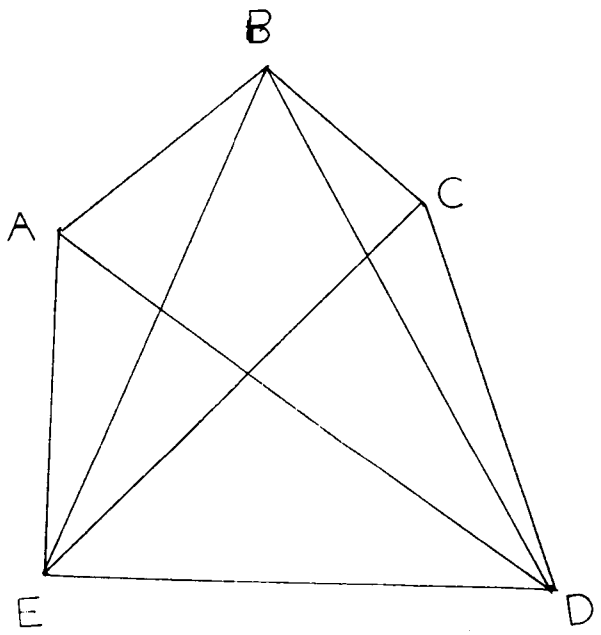


Figure 4. Incomplete graph.

Subgraph. If, in a graph, one or more vertices are omitted, together with the arcs to and from these points, the remaining portion of the graph is a subgraph.

Network. A connected graph without a loop and with more than one vertex.

Node. In a network, a vertex which is connected to the graph is called a node.

Steiner point. A node that is connected to two or more vertices [5].

Internode. The link between two nodes.

Associated matrix. Given a graph G = (X,U) with vertices X_1, X_2 ....X_n, let us assume

$M_{ij} = 0$ $\quad (X_i,X_j) \notin X$

$M_{ij} = 1$ $\quad (X_i,X_j) \in X$

The square matrix formed by M_{ij} terms is called the associate matrix of the graph G.

3. TREES

After all the definitions of the previous technical terms, we find ourselves in a position to define a tree and deal with its applications.

We say that a connected graph which does not contain any cycle is a tree. Given a graph G with at least two vertices, any one of

the following properties is sufficient to define a tree:

1. G is connected and without a cycle.

2. G is without a cycle and has n-1 links, n being the number of vertices.

3. G has no cycle, but by adding a link between two nonadjacent vertices we create one, and only one, cycle.

4. G is connected unless any link is suppressed.

5. Every pair of vertices is connected by one, and only one, chain [2].

Let us take property number 2 to show that it agrees with the definition of the tree. If we try to form a tree of two points we will have one link to make no cycle, for if we have more than one link to join two points we will have a cycle. If we take three points, we will need two links because three links will form cycles and one link will join only two points and leave the third free. For more than three points we will have the same situation. So, for less than n-1 links we will be short of links and the graph is not connected and for more than n-1 links we will have a cycle. It is easy to show that the rest of these properties agree with the tree definition.

A tree of a graph can be either directed, Figure 5, or undirected, Figure 6. If it is directed, a tree $T = (X,U)$ has a root $X_1 \in X$ if there is a path from X_1 to X_i for each $i \neq 1$ and the graph does not contain a circuit. If each $X_i \neq X_1$ is the terminal extremity of any arc, we call the tree <u>arborescence</u> of root X_1.

In a tree $T = (X,U)$, if a real value $L(u)$ is given to each link

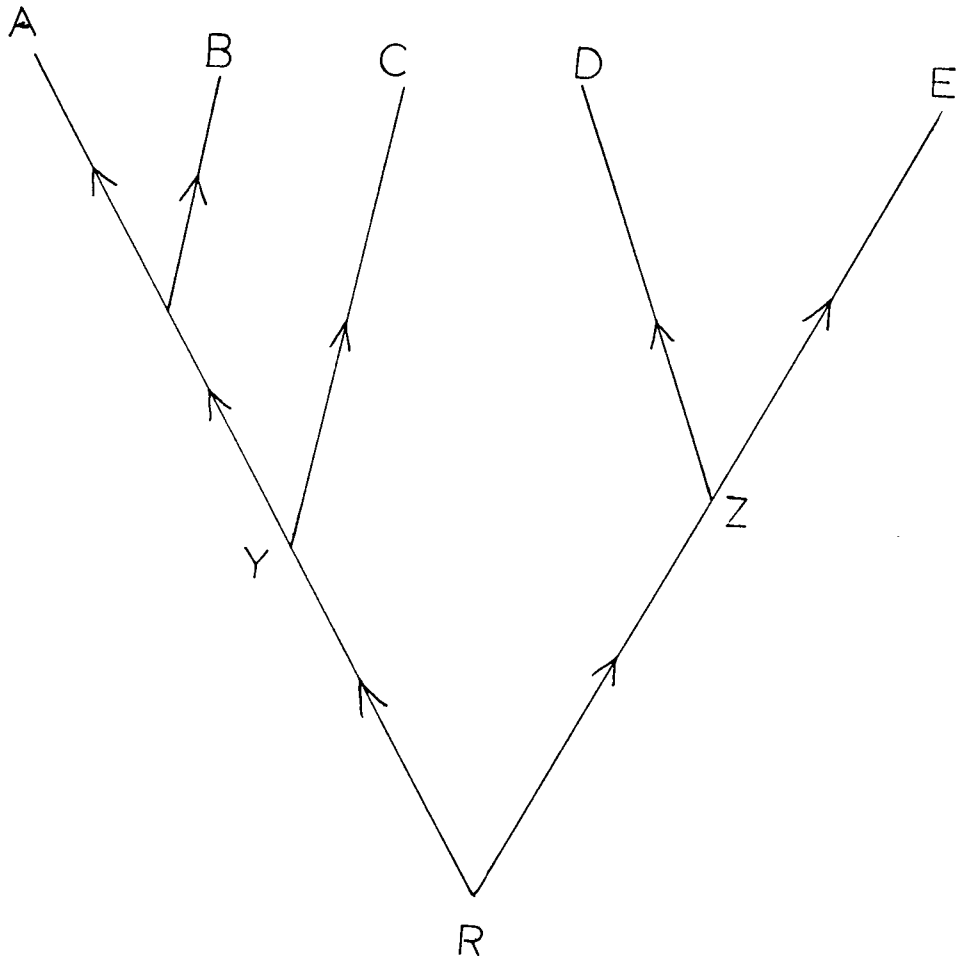


Figure 5. Directed tree.

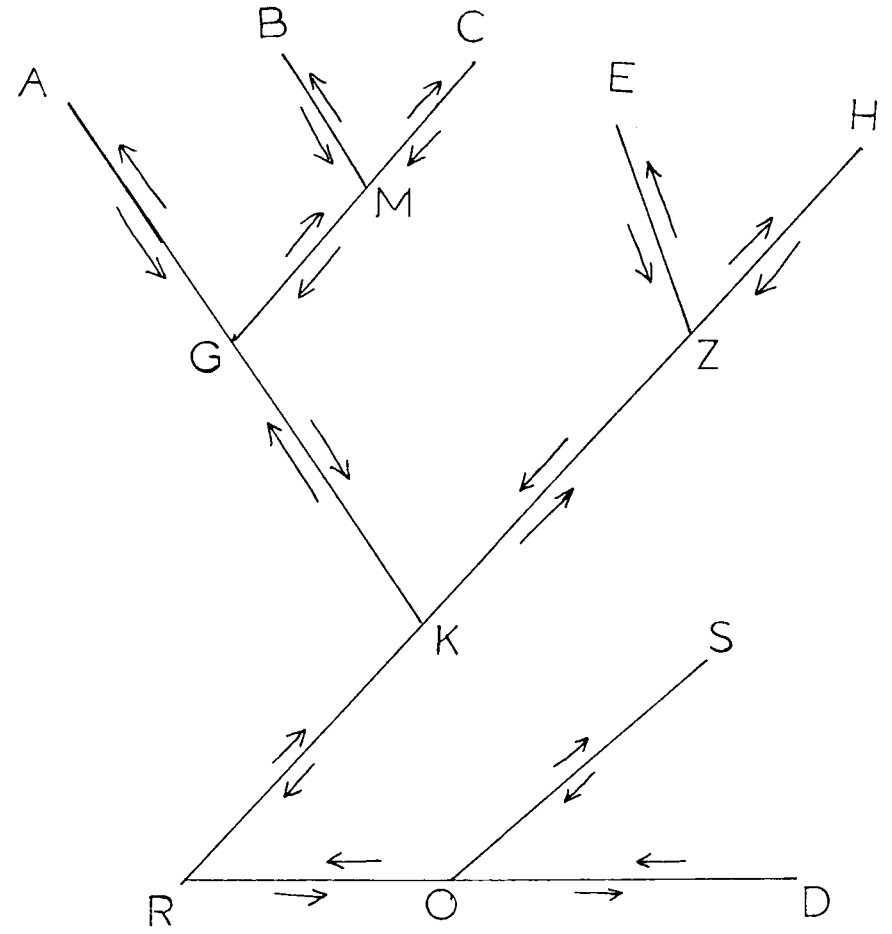


Figure 6. Nondirected tree.

$u \in U$, we give the term total value of the tree $T = (X,U)$ to the sum

$$S = \sum_{u \in U} L(u)$$

A graph of a set of trees which are not connected to each other is called a forest [4].

A tree is nondirected if at least one path from some X_i to X_j is a two-way street. If all the links permit paths both ways, we can consider any node as the root of the tree.

It is evident that the number of links in a connected tree is one less than the number of nodes. Among all the possible trees which we can form from a set of vertices there is one which is optimal, that is, of minimal length. The length of a tree is measured by the maximum sum of the internodes (chain) between the root and a terminal node, if the tree is directed.

An algorithm will be described later in the applications to find the minimum length for the undirected tree.

CHAPTER II

EVOLUTIONARY TREES

1. INTRODUCTION

Recently, many methods and procedures have been developed to find a representation for the relationship of a group of organisms.

Although this study will concentrate on finding the relationship between living species based on studies and observations of some of its characters in the present time, it is possible and of some interest to estimate a historical relationship between species from different time planes. This would be obtained from fossil records considering the evolutionary rates over different periods of past time. The relationship that we will seek will be in a form of a simple graph.

Doing this, we will be accepting the Darwin theory of evolution as a postulate. The notion of the Darwin theory will be used to go back to the "root" of the tree that represents some living as well as ancient organisms.

2. RATES OF EVOLUTION

When it is possible to study fossil material and obtain data from several known points of time, we will be able to estimate overall evolution rates. The dissimilarity between ancestral and descendant forms will be the measure of the overall evolution that has occurred in the intervening period, Figure 7.

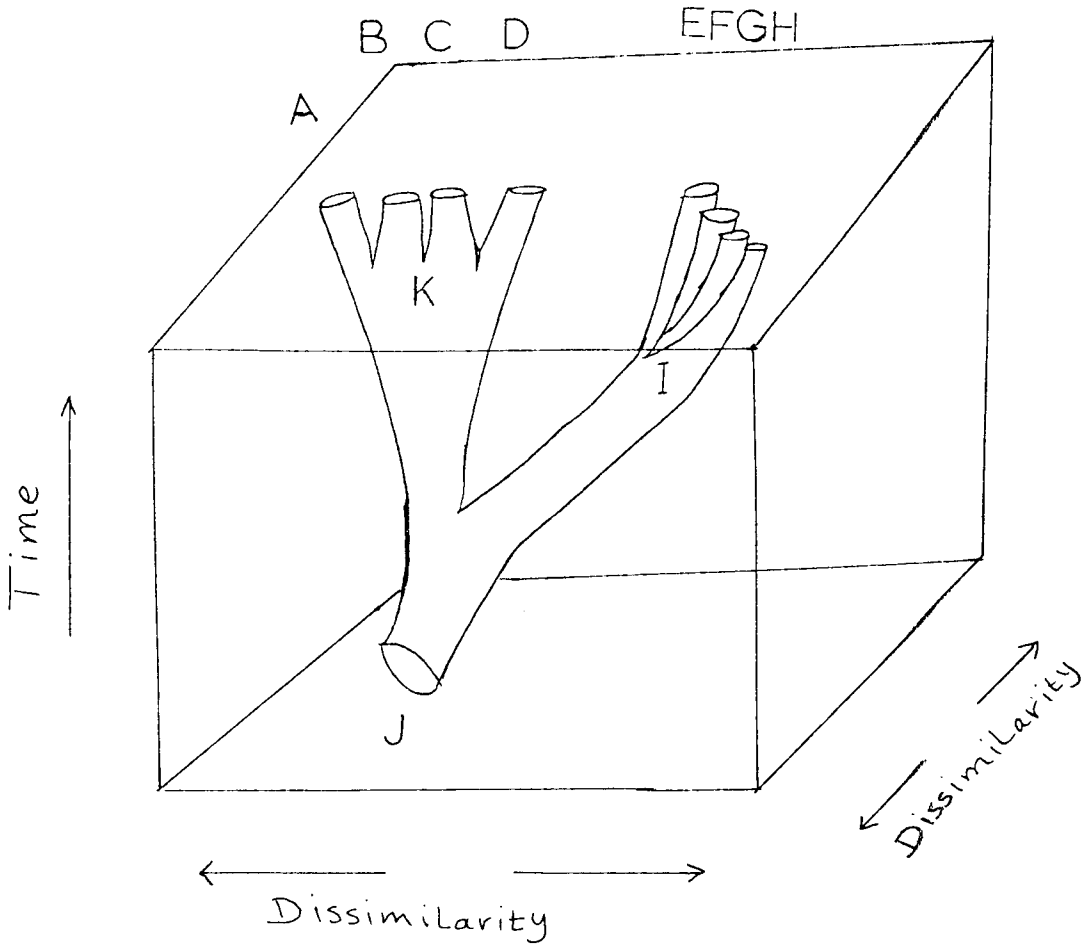


Figure 7. The relation between the dissimilarity and the evolution time.

The change in the state of a single character will be considered. The unit of the rate of change is the Darwin. The Darwin corresponds to a change by a factor of "e" in one million years; that is:

$$e^t = \frac{X_t}{X_o}$$

when the character has the value X_0 at time o, and X_t at time t, with t measured in crons (million years) [5].

All the evidence at hand indicates evolutionary rates in different periods of life history are not uniform. Different lines do evolve at different rates (Klug and Farris, 1969) [5]. However, various authors have estimated fairly constant rates over long periods of geological time, even periods of time back to the common ancestors of animals and plants.

The common ancestor most probably possessed character states close to the states in present-day organisms. Ancestors are unknown. So, even if some recent organisms are shown to be close to the common ancestors of a set of species, it is unlikely that the species living at present actually are descendants of these common ancestors. The caution must extend to fossil materials. It is rare that a known fossil can be assumed to be a direct ancestor of a later fossil, though it may be close to the ancestor.

The possible relationship between present living organisms and recent ancestors can be represented in a form of an evolutionary tree.

3. TREES

Trees showing the relationship between species and ancestors are either directed, with an assigned most primitive ancestor, or

nondirected graphs. By leaving the relationship as an undirected tree, one is able to assign experimentally one node or another as an ancestor and to make a reasonable judgement. The terminal nodes (species) in a network are called Ordinary Taxonomy Units (OTU's) while the other nodes within the network are called Hypothetical Taxonomy Units (HTU's). The term "taxonomy" could be defined as the laws of classification as applied to natural history. HTU's are called supplementary or Steiner points. They help to minimize the overall length of the graph. The additional points will rarely be living organisms and will normally be either fossils or Steiner points.

Taxonomy units are graphs which are known in graph theory as "Steiner Minimal Trees." They are minimal length trees for t vertices (OTU's, terminal nodes). In order to achieve minimum length they may contain other vertices (HTU's), Steiner points, in addition to the t vertices.

Although various authors have used minimum length trees to generate evolutionary hypotheses, the formal treatment of the subject was by Farris [5]. In addition to the t terminal nodes (OTU's) the Steiner minimal tree will have a variable number of t*-t additional nodes (HTU's), where $t^* \geq t$, yielding a total of t* nodes for the entire tree. He defines the length of the internode between two nodes j and k as the Manhattan distance:

$$d(j,k) = \sum_{i=1}^{n} |X_{ji} - X_{ki}|$$

where X represents the matrix of m species and n characters. We shall give some examples below to clarify this terminology.

A Wagner network as defined by Farris [5] is a Steiner minimal tree measured in Manhattan distance. The Wagner network permits reversibility of character states; that is, the tree is nondirected, where we can assign any node as the root for the entire tree.

4. METHODS FOR COMPUTING WAGNER TREES

The Wagner method for extimating evolutionary trees has been widely employed in botanical studies and has more recently been employed in zoological evolutionary taxonomy. The Wagner technique is of considerable interest because it is programmable and because the type of tree produced can tractably be extended to applications in a variety of fields as we will see later.

To be able to find a relation between a set of OTU's and its hypothetical ancestors, we have to know some information about these OTU's in the form of character states. We can collect this information by observing and studying the character states of present species or fossils from different periods of time.

Character Analysis and Representation

Given a group of organisms with a complete study of its characters, the problem is how to find a relationship between these organisms in the form of a tree.

A study was made by C. H. Nelson [3] for the Genus Perlomyia to find its position within the insect family of Leuctridae. Numbers were given to the different character states of these organisms according to its advancement to make it easy for the computer to

handle it. Thirty-one character states coded for 13 species were studied as follows:

Number of anal veins in hindwing:

6 = 0; 3 = 1; 2 = 2.

Stigma of the forewing:

Absent = 0; Present = 1.

Shape of each male sub-anal plate:

Rectangular shaped = 0; Triangular shaped = 1; Triangular shaped but more elongate = 2; Elongate narrow = 3; and so on.

We shall mention that we are not interested in the previous biological terms used in this study as much as we are interested in the method of representation of the character state to make it applied to applications on computers. Note that the more advanced the character state is the higher is the number assigned to it, while the most primitive state is given zero.

Representation of Trees

An evolutionary tree T, can be represented as an ordered pair, T(N,f), where N is a collection of nodes of the tree and f is a function that assigns to a member, $n \in N$, a unique node, f(n), such that n is directly derived from f(n) according to the tree, T; f(n) is called the immediate ancestor of n. Figure 8 shows a

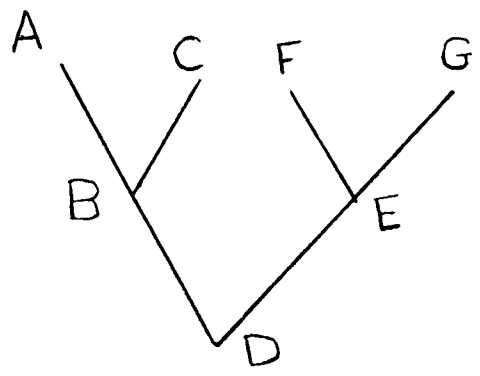


Figure 8. Representation of nodes and ancestors in a tree graph.

tree with nodes A, B, C, D, E, F, G, and ancestor functions:

$f(A) = B$ $\quad f(c) = B$

$f(B) = D$ $\quad f(E) = D$

$f(F) = E$ $\quad f(G) = E$

where f(D) is not defined. While D is the ancestor of the whole tree, it is not true, for example, that f(A) = D, because D is not the immediate ancestor of A.

We shall consider only pairs, T = (N,f) that have a unique node, P, in N such that if n is any element of N, then there exists a non-negative integer, K, for which $f^k(n) = P$. So, for the previous example: $f(f(A)) = D$ or $f^{(2)}(A) = D$.

D is called the unique "root" of the tree of which every element of N is connected to the tree, so that it is a descendant of the "root." We call this kind of trees "rooted" or "directed" trees, where the "root" possesses the most primitive character states [1].

5. LENGTH OF A TREE

In constructing an evolutionary tree it is necessary to choose among a large number of possible alternative trees. One way of making the choice is to select the tree that implies the minimum amount of evolutionary change between the OTU's. Camin and Sokal [1] took such an approach in introducing the idea of most parsimonious trees as trees with the smallest number of "steps" (change in integer-valued characters). The notion of length of a tree is a generalization of "number of steps."

A data matrix, X, assigns a state, X(a,i) to a node A (species connected to the network) for character i. This state given as an indication of the development of particular character of the species along some relative evolutionary scale. The difference between two nodes, A and B, is defined to be:

$$d(A,B) = \Sigma_i |X(A,i) - X(B,i)| \qquad (2.1)$$

which we call the Manhattan distance of dissimilarity measure. Two nodes, A and f(A) are the end points of a unique internode of a tree, T = (N,f), for A in N. The length of the internode between A and f(A) is defined to be d(A,f(A)). The tree T = (N,f) has just K-1 internodes if K is the number of nodes in N. The length of the tree is defined to be:

$$L(N,f) = \sum_{n \ddagger P} d(n, f(n))$$

where P is the root of the tree.

Some calculations for constructing evolutionary trees can be simplified by a relation of the differences between the species and its ancestors.

Suppose that a node B is connected to two other nodes C and D through an ancestor, A (Figure 9). For each character, the state of

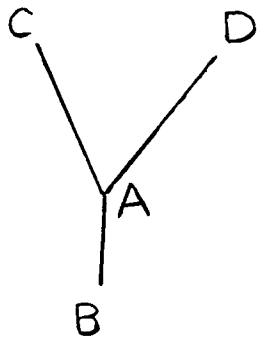


Figure 9. Connecting a node to two other nodes through an ancestor.

A is the median of states of B, C, and D. Then for any character, i,

$X(A,i)$ lies between $X(B,i)$ and $X(C,i)$,

$X(A,i)$ lies between $X(B,i)$ and $X(D,i)$,

$X(A,i)$ lies between $X(C,i)$ and $X(D,i)$,

consequently

$$|X(B,i) - X(C,i)| = |X(A,i) - X(B,i)| + |X(A,i) - X(C,i)|$$

$$|X(B,i) - X(D,i)| =$$

$$= |X(A,i) - X(B,i)| + |X(A,i) - X(D,i)|$$

$$|X(C,i) - X(D,i)| =$$

$$= |X(A,i) - X(C,i)| + |X(A,i) - X(D,i)| \qquad (2.2)$$

Summing over characters and using (2.1) and (2.2) yields:

$$d(B,C) = d(A,B) + d(A,C) \qquad (2.3)$$

$$d(B,C) = d(A,B) + d(A,D) \qquad (2.4)$$

$$d(C,D) = d(A,C) + d(A,D) \qquad (2.5)$$

Adding (2.3) and (2.4), we obtain

$$d(B,C) + d(B,D) =$$

$$= 2d(A,B) + d(A,C) + d(A,D) \qquad (2.6)$$

By (2.5), (2.6) becomes

$$d(B,C) + d(B,D) = ad(A,B) + d(C.D) \qquad (2.7)$$

Therefore,

$$d(A,B) = (d(B,C) + d(B,D) - d(C,D))/2 \qquad (2.8)$$

equation (2.8) gives the difference between a node, B, and an ancestor through which B may be connected to two other nodes, C and D. It will be useful to think of the connection between C and D as an object called an interval, INT(C,D), whose difference from a node, B, may be calculated as:

$$DINT(B,INT(C,D)) = (d(B,C) + d(B,D) - d(C,D))/2.$$

6. WAGNER METHOD FOR CONSTRUCTING EVOLUTIONARY TREES: ALGORITHM AND PROCEDURE

The steps of the procedure will be accompanied by a short example to explain the method.

1. Set up a data matrix X(I,J) of species and characters (Table 1).

Table 1. Data Matrix for Species and Characters for Wagner Method

	Characters (J)				
Species (I)	1	2	3	4	5
A(1)	0	0	2	0	2
B(2)	0	0	2	0	1
C(3)	0	1	1	0	0
D(4)	0	1	1	1	0
E(5)	1	0	0	0	0

2. Construct a "group ancestor" Q which has lowest value for all characters (can be set to "0" all the time).

	1	2	3	4	5
Q(6)	0	0	0	0	0

3. Compute the difference between each species and Q where (example):

$$d(A,Q) = \sum_{J=1}^{5} |X(1,J) - X(6,J))|$$

where:

d(A,Q) = 4 d(D,Q) = 3

d(B,Q) = 3 d(E,Q) = 1

d(C,Q) = 2

4. Find the species that differs least from Q (in above example d(E,Q) = 1). Connect it to Q to form a "network."

Network



5. Find the unplaced species that differs least from Q (in above example d(C,Q) = 2).

6. Find the "interval difference" between species C and other species placed in network excluding Q where:

$$DINT(C,INT(E)) = [d(C,E) + d(C,ANC(E)) - d(E,ANC(E)]/2$$

where ANC(E) = Q

$$DINT(C,INT(E)) = (3 + 2 - 1)/2 = 2.$$

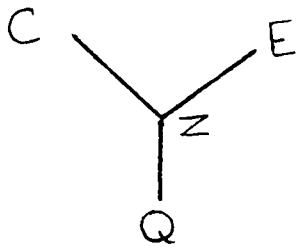
7. Select "interval difference" that is minimal and connect species C to the network at that point.

8. Construct a "hypothetical ancestor" (Z) of species C and E by taking lowest values of each character in C and E and giving it to Z (Table 2).

Table 2. Constructing the Ancestor Z for Species C and E

	Character				
Species	1	2	3	4	5
C(3)	0	1	1	0	0
E(5)	1	0	0	0	0
Z(7)	0	0	0	0	0

A network with species C and hypothetical ancestor Z will resemble



9. Find the unplaced species that differs least from Q (recall that was done under step 3) where now d(B,Q) = 3 and d(D,Q) = 3. Just arbitrarily select one of these, in this case d(D,Q) = 3.

10. Find the "interval difference" between D and other species and "hypothetical ancestors" placed in the network the same as step 6.

DINT(D,INT(E)) = (4 + 3 - 1)/2 = 3

DINT(D,INT(C)) = (1 + 3 - 2)/2 = 1

DINT(D,INT(Z)) = (3 + 3 - 0)/2 = 3.

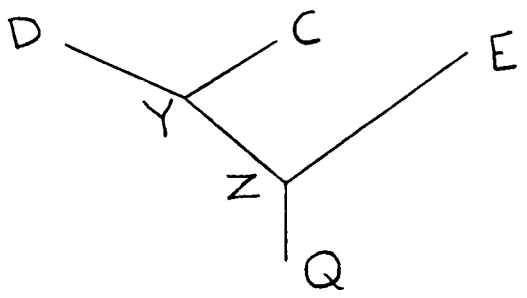
11. Select "interval difference" that is minimal and connect species D to the network at that point (the same as step 7).

12. Construct a hypothetical ancestor (Y) of species D and C by taking lowest values of each character in D and C and giving it to (Y) (Table 3).

Table 3. Constructing the Ancestor (Y) for Species D and C

	Characters				
Nodes	1	2	3	4	5
D(4)	0	1	1	1	0
C(3)	0	1	1	0	0
Y(8)	0	1	1	0	0

A "network with species D and hypothetical ancestor, Y, will resemble



13. If no more species stop.

14. Go to (9).

Figure 10 gives the final network which represents a tree

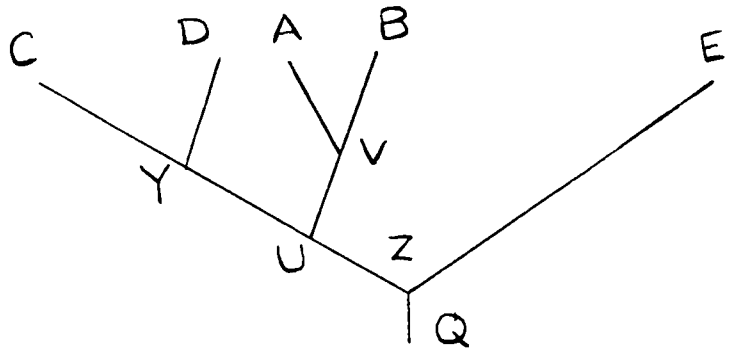


Figure 10. Final network of the Wagner tree.

relationship between species A, B, C, D, and E as terminal nodes. The tree can show, too, the relation between these species and their hypothetical ancestors Y, V, U, Z, and Q which was assigned as a primitive root of the whole tree. See Appendix A for program.

Another different method devised by Wagner will be described as the rootless Wagner method since it does not assign a root for the tree. This method will obtain a nondirected tree relationship between species which will allow us to consider any node as a root.

7. ROOTLESS WAGNER METHOD: ALGORITHM AND PROCEDURE

The steps again will be accompanied by the same example used in the Wagner method for explanation.

1. Set up the data matrix X(I,J) of OTU's (species) and characters (Table 4).

Table 4. Data Matrix for Species and Characters

	Characters				
Nodes	1	2	3	4	5
A	0	0	2	0	2
B	0	0	2	0	1
C	0	1	1	0	0
D	0	1	1	1	0
E	1	0	0	0	1

2. Compute the difference between each OTU and the rest where the difference is:

$d(A,B) = \Sigma_J \text{ABS}(X(A,J) - X(B,J))$

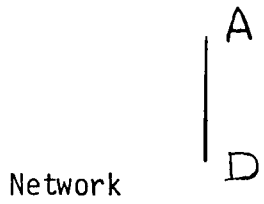
d(A,B) = 1 d(B,C) = 3 d(C,D) = 1 d(C,E) = 4

d(A,C) = 4 d(B,D) = 4 d(C,E) = 3

d(A,D) = 5

d(A,E) = 5

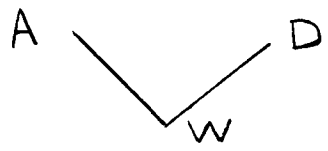
3. Select the largest difference (here (A,D) = 5) and link to form an interval.



Network

4. Construct a hypothetical ancestor W of A and D. This is done by comparing each character state of A,D and giving the minimal to W.

	1	2	3	4	5
W	1	0	0	0	0



Network

5. Compute the "advancement index" of each OTU as:

$d(I,INT(A,D)) = [d(I,A) + d(I,D) - d(A,D)]/2$

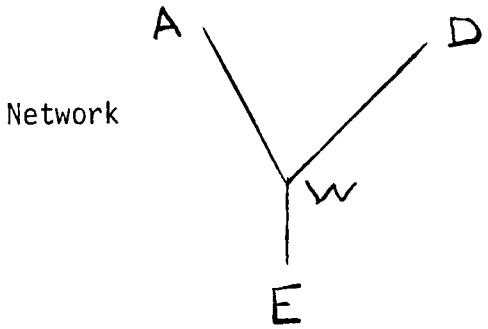
such that OTU INT(A,D):

$DINT(B,INT(A,D)) = (1 + 4 - 5)/2 = 0$

$DINT(C,INT(A,D)) = (4 + 1 - 5)/2 = 0$

$DINT(E,INT(A,D)) = (5 + 4 - 5)/2 = 2$ * Largest

6. Select the interval difference which is largest and link to form a network EINT(A,D).



Network

7. Recall step 5 to select the second largest "advancement index" (in this example either B or C).

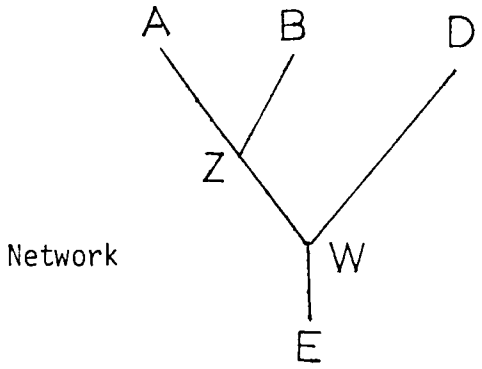
8. Find the interval on network which B is closest to:

DINT(B,INT(A,W) = (1 + 2 - 3)/2 = 0 * Minimal

DINT(B,INT(D,W) = (4 + 2 - 2)/2 = 2

DINT(B,INT(E,W) = (4 + 2 - 2)/2 = 2

9. Select interval difference which is minimal [DINT(B,INT(A,W))]. Attach B to network at this interval.



10. Construct a hypothetical ancestor to B and A, (Z) (Table 5).

Table 5. Constructing the Ancestor Z for Species A and B

	Characters				
Nodes	1	2	3	4	5
A	0	0	2	0	2
B	0	0	2	0	1
Z	0	0	2	0	1

11. If no more OTU's, stop.

12. To to setp 7.

The final network will be as shown in Figure 11 ((a), or (b), or (c)).

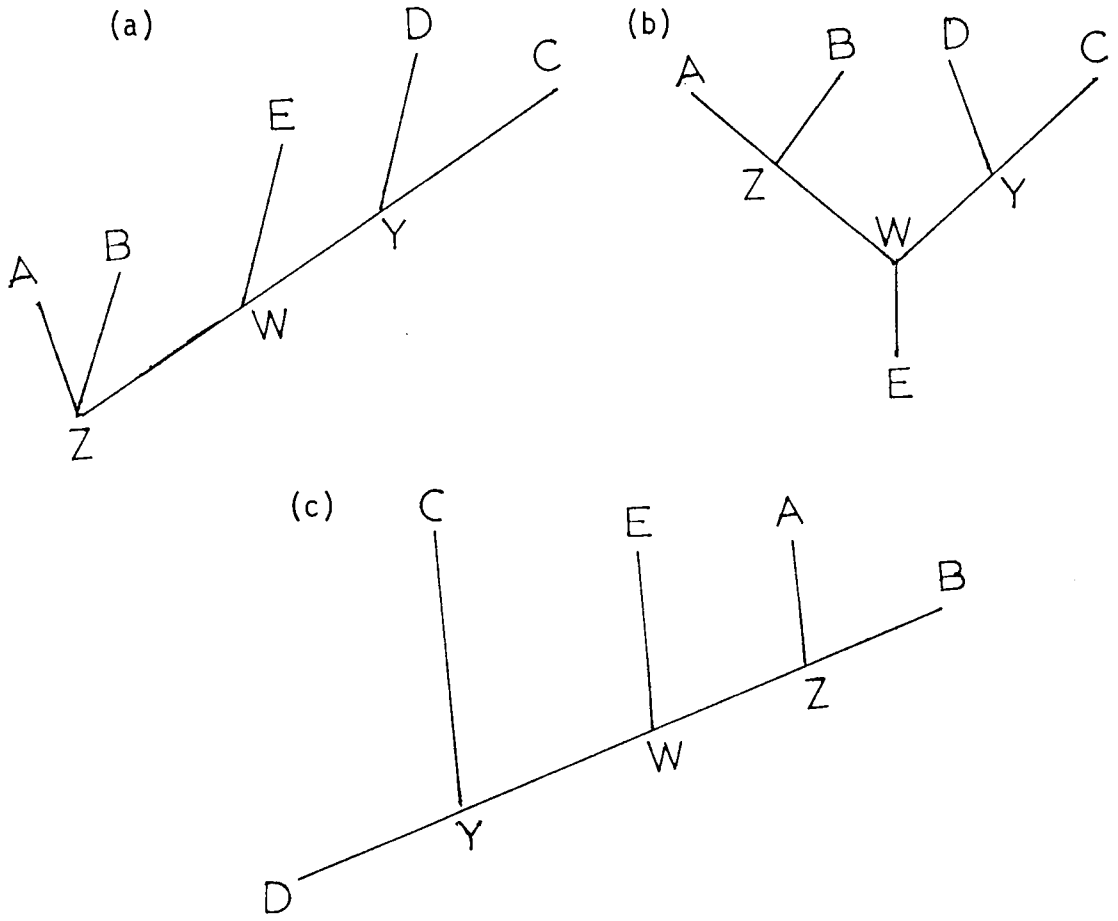


Figure 11. Final network of the rootless Wagner tree.

We can see that the rootless Wagner method permits us to assign any OTU or HTU as a root for the tree. All of the trees of the forest of Figure 18 show the same relationship between the species. See Appendix B for program.

A third method which was developed by Wiss [1] will be considered. It uses different approach to the problem, but it obtains the same result as the rooted Wagner. Wagner's method has an advantage over this method, for it uses fewer computations for a large number of species involved, and consequently uses less core in the computer. However, the Wiss method has the advantage of showing the advancement index as steps of advancement of a species along an evolutionary scale in the final network (tree).

8. WISS METHOD FOR CONSTRUCTING EVOLUTIONARY TREES

1. Set up the data matrix X(I,J) where I = OTU and J = character (Table 6).

Table 6. Data Matrix for Species and Characters for the Wiss Method

	Characters							
OTU's	1	2	3	4	5	6	7	8
A	1	0	1	1	0	1	0	1
B	1	0	1	0	1	0	1	1
C	0	1	0	1	1	1	1	0
D	0	1	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0

2. Select Q, an ancestor which has each character arbitrarily set in the most primitive state. Add it to the data matrix.

3. Compute the difference between each OTU and between each OTU and Q where difference:

d(A,B) = ABS(X(A,J) - X(B,J))

hence:

d(A,B) = 4 d(B,C) = 6 d(C,D) = 4 d(D,Q) = 1

d(A,C) = 6 d(B,D) = 6 d(C,Q) = 5

d(A,D) = 6 d(B,Q) = 5

d(A,Q) = 5

4. Compute the "advancement index" (interval difference) between each OTU and between each OTU and Q where "advancement index":

h(A,B) = (d(A,Q) + d(B,Q) - d(A,B)) * (1./2.)

hence:

h(A,B) = (5 + 5 - 4)/2 = 3

h(A,C) = (5 + 5 - 6)/2 = 2

h(A,D) = (5 + 1 - 6)/2 = 0

h(A,Q) = (5 + 0 - 5)/2 = 0

h(B,C) = (5 + 5 - 6)/2 = 2

h(B,D) = (5 + 1 - 6)/2 = 0

h(B,Q) = (5 + 0 -5)/2 = 0

h(C,D) = (5 + 1 - 4)/2 = 1

h(C,Q) = (5 + 0 - 5)/2 = 0

h(D,Q) = (1 + 0 - 1)/2 = 0

5. Select the pair of OTU's with the largest "advancement index" (in setp 3 this was h(A,B) = 3).

6. Contrast a hypothetical ancestor X of A and B, which has the lowest values of A and C. This is done by comparing each character state of A and B and assigning the lowest (primitive) state to X (Table 7).

Table 7. Constructing the Ancestor X for Species A and B

	Characters							
Nodes	1	2	3	4	5	6	7	8
A	1	0	1	1	0	1	0	1
B	1	0	1	0	1	0	1	1
X	1	0	1	0	0	0	0	1

7. Compute the difference between each unplaced OTU and between each unplaced OTU and X (following the procedure in step 2). Hence:

$d(C,D) = 4$ $d(D,Q) = 1$ $d(X,Q) = 3$

$d(C,Q) = 5$ $d(D,X) = 4$

$d(C,X) = 8$

8. Compute the "advancement index" between each unplaced OTU and between each unplaced OTU and X (following the procedure in step 3). Hence:

$h(C,D) = (5 + 1 - 4)/2 = 1$

$h(C,Q) = (5 + 0 - 5)/2 = 0$

$h(C,X) = (3 + 5 - 8)/2 = 0$

$h(D,Q) = (1 + 0 - 1)/2 = 0$

$h(D,X) = (3 + 1 - 4)/2 = 0$

$h(X,Q) = (3 + 0 - 3)/2 = 0$

9. Select the pair of species with the largest "advancement index" (in step 7 this was $h(C,D) = 1$).

10. Construct a hypothetical ancestor Y of C and D, which has the lowest values of C and D (following the procedure in step 5) (Table 8).

Table 8. Constructing the Ancestor Y for Species C and D

	Characters							
Nodes	1	2	3	4	5	6	7	8
C	0	1	0	1	1	1	1	0
D	0	1	0	0	0	0	0	0
Y	0	1	0	0	0	0	0	0

11. Compute the "advancement index" between each unplaced and ancestoral species and between each unplaced and ancestoral species and Y (following the procedure in step 3). Hence:

$h(X,Y) = (1 + 3 - 4)/2 = 0$

$h(X,Q) = (3 + 0 - 3)/2 = 0$

$h(Y,Q) = (1 + 0 - 1)/2 = 0$

12. Select the pair of species (excluding any pair with Q) with the largest "advancement index" (in step 10 this was $h(X,Y) = 0$).

13. If there are no further unplaced real or ancestoral OTU's stop.

14. Construct an evolutionary tree utilizing the "advancement indices" of real and ancestoral OTU's (Figure 12). See Appendix C for program.

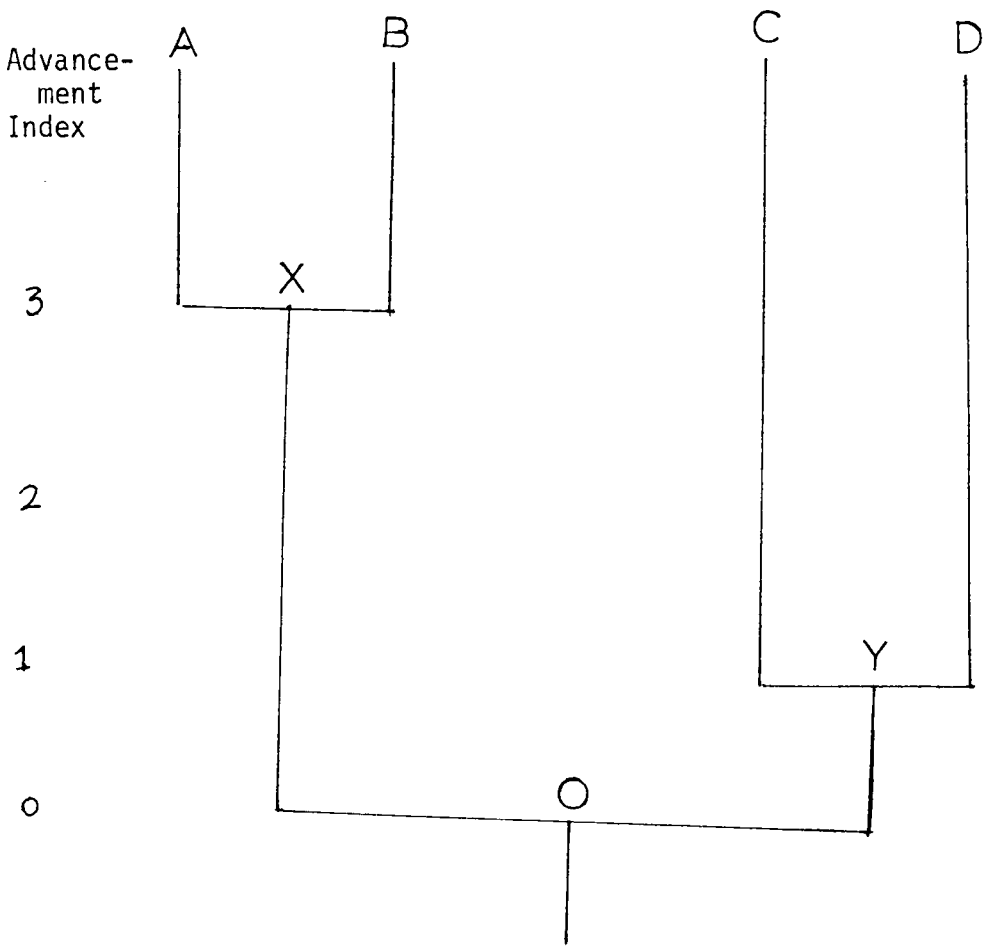


Figure 12. Final network of the Wiss tree showing the advancement index measure.

CHAPTER III

GENERAL APPLICATIONS

We now consider a class of problems for which the previous algorithms, with suitable modification, apply. These might be helpful in providing clear demonstrations of the steps of the algorithms.

1. THE GREATEST COMMON DIVISOR (GCD)

Usually, we find the greatest common divisor of a set of numbers by decomposing each member of the set into its prime factors. We look for the common factors among all these members and find their product to form the GCD for the group. That is an easy job when the set, under consideration, consists of just a few small members, but it will become a tedious job for large sets with more complicated members involving many different prime factors.

The previous computerized methods used for constructing evolutionary trees can be slightly modified to solve this problem. To find the GCD for a set of numbers using the computer, we use a subroutine FACTO to form the input data matrix described before in the Wiss and Wagner methods.

Method

The subroutine assigns the prime numbers in an increasing sequence up to a limit we select suitable to the largest number in the group. See Appendix E for this subroutine. The subroutine computes the first group of primes under 100. It, then, writes these primes as a heading

of Table 9, factors the members of the group and writes the matrix (Table 9).

Table 9. Decomposition of a Set of Numbers to Its Prime Factors

Numbers (Species)	Primes (Characters)				
	2	3	5	7	11
30	1	1	1	0	0
45	0	2	1	0	0
66	1	1	0	0	1
210	1	1	1	1	0
126	1	2	0	1	0
100	2	0	2	0	0

Thus, in this case, our "species" consist of numbers whose GCD is to be found, our characters consist of primes, and thus the character states indicate the occurrence of a power of a prime in the prime factors decomposition of the "species." For example, let us consider the set {30, 45, 66, 210, 126, 100}. This matrix will be the input to the program which constructs the tree. The terminal vertices will be the members of the group, while the ancestors will be the GCD's for these numbers (Figure 13).

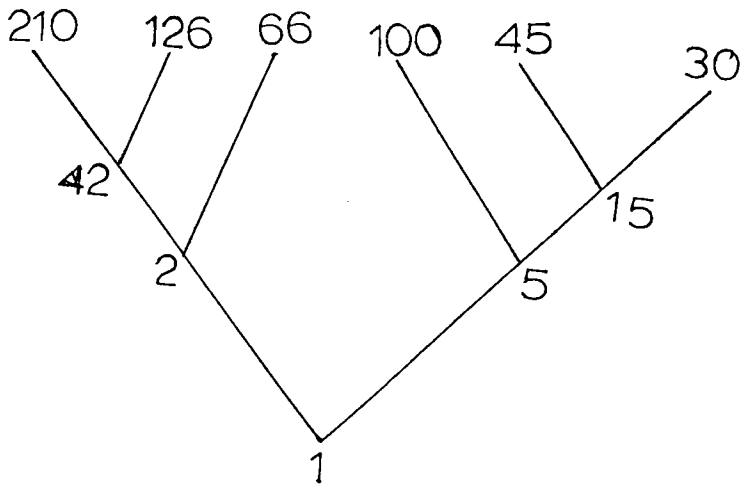


Figure 13. The GCD tree.

The tree shows the total GCD for the group as well as partial GCD's. The total GCD (that is, the GCD of the entire set) is the root of the tree. It is 1 in this example. We can see also that the GCD for 45 and 30 is 15 and for 100, 45, 30, and 15 is 5. So, the ancestor of any branch of the tree is the GCD of all the nodes of that branch.

Explanation of the Method

The procedures used in the three methods, Wiss, Wagner, and the rootless Wagner, computed the differences (Manhattan distance) between the different members to show which one is closest to the other. For example, 36 is closer to 48 than 50 is, because

	2	3	5	7
36 =	2	2	0	0
48 =	4	1	0	0
50 =	1	0	2	0

We can see that:

$d(36, 48) = 3$

$d(50, 48) = 6$

Thus, in this case, the Manhattan distance between two numbers is just the total number of prime factors which appear in one of the numbers but not the other.

So, the method will choose 36 to connect with 48 and find common ancestor for them. The character states of the ancestor will be the lowest states from both for each character, that is, the greatest power of a given prime which occurs in both numbers. The ancestor of 36 and 48 will have the states

2	3	5	7	
2	1	0	0	getting 12 by

multiplying $2^2 \cdot 3^1 \cdot 5^0 \cdot 7^0$. It is clear that 12 is the GCD for 36 and 48.

Using this terminology, we will have each ancestor of a set of numbers as the GCD of that group which consists of the terminal nodes of the branch. The ancestor of the ancestors is the GCD of these ancestors and that will result of being the GCD of the terminal nodes.

Our definition from graph theory states that a tree is a connected group of points without cycles. This is the case in the branch of a tree which allows us to conclude that the ancestor of a tree is the GCD of its nodes.

2. THE LEAST COMMON MULTIPLE (LCM)

The same programs mentioned before will compute the least common multiple in the same way in which it computed the GCD for a group

of numbers. The only difference is in stating the character states of the ancestor. Instead of choosing the lowest of the states in the comparison of the two numbers in the case of computing the GCD, we choose the highest state in the case of LCM. Just an interchange in statement numbers 80 and 60 in the subroutine CAT (Appendix E) will alter the process of computing the GCD to compute the LCM. To explain the method of computing the LCM, we will again consider the numbers 36, 48, and 50. Choosing an ancestor for 36 and 48, we select the highest state for each character in both numbers and multiplying, we get

2	3	5	7	
4	2	0	0	$\equiv 2^4 \cdot 3^2 = 144.$

The method used will select 36 to connect with 48 through an ancestor = 144 and then connect 50 with 144 in another ancestor = $2^4 \cdot 3^2 \cdot 5^2 = 3600$ which is the LCM for 36, 48, and 50 (Figure 14).

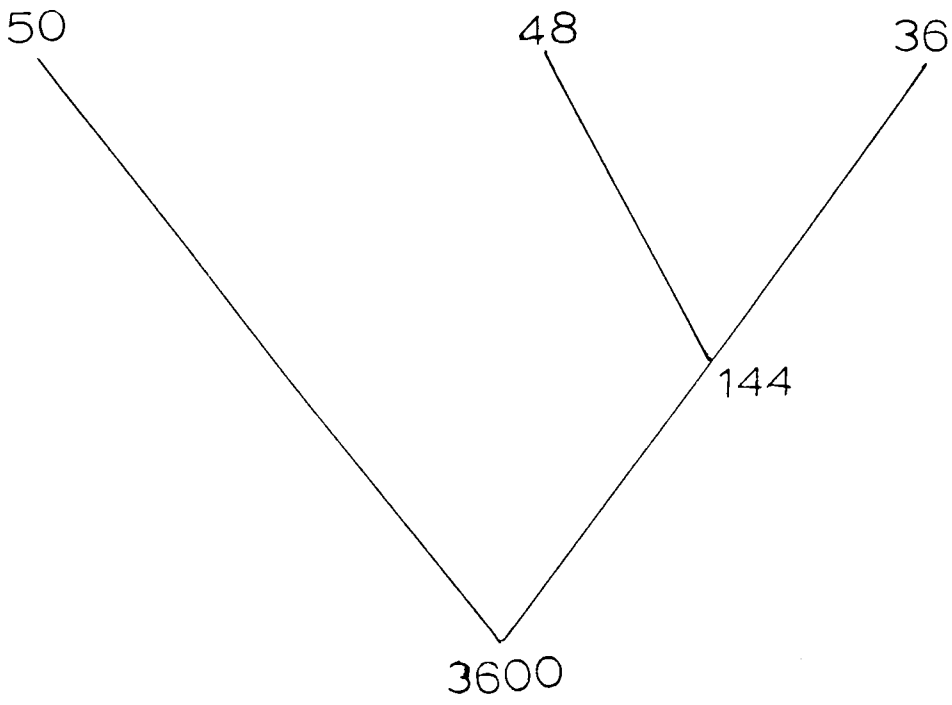


Figure 14. The LCM tree for the numbers 50, 48, and 36.

3. THE "INTERVAL DIFFERENCE" MEANING IN APPLICATIONS

We will show, step by step, how the Wagner method uses the interval difference notion to find the GCD for a set of numbers. Let us consider the same previous set 36, 48, and 50.

Steps 1, 2. Set up the data matrix with a root = 1 (Table 10).

Table 10. Decomposing the Numbers 36, 48, and 50

	Characters			
Nodes	2	3	5	7
36	2	2	0	0
48	4	1	0	0
50	1	0	2	0
1	0	0	0	0

Step 3. Compute differences between each species and the root.

$d(36,1) = 4$ $\qquad$ $d(50,1) = 3$

$d(48,1) = 5$

Step 4. Fifty differs least from 1, so it is connected to 1.



Step 5. The next least difference is 4 which is for 36 so we choose it.

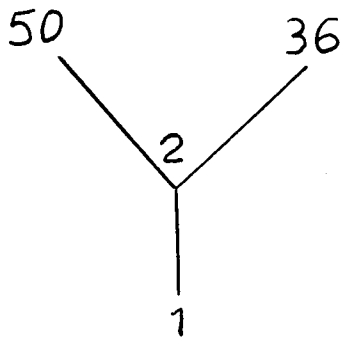
Steps 6, 7, 8. We find the interval difference between 36 and the nodes in the network:

DINT(36, INT(50)) = (d(36,50) + d(36,ANC(50))

- d(50,ANC(50))/2 = (5 + 4 - 3)/2 = 3

where ANC means the immediate ancestor. So, DINT(36,INT(50)) = 3.

We select 3 as minimal for it is the only one and connect 36 to 50 at this interval assigning an ancestor for them by taking lowest values of character states.



Step 9. Repeating step 4, we select 48.

Step 10. DINT(48,INT(50)) =

[d(48,50) + d(48,ANC(50)) - d(50,ANC(50))/2

= (6 + 4 - 2)/2 = 4

DINT(48,INT(36)) = 2

DINT(48,INT(2)) = 4

Steps 11, 12. Selecting the minimal, we select

DINT(48,INT(36)) = 2

construct an ancestor for 48 and 36 which is 12 and we notice that d(12,48) = 2, which DINT (48,INT(36)) and place it in the network.

Step 13. Are there more species which are not placed? The answer is no and Figure 15 is the final tree. If the answer for step

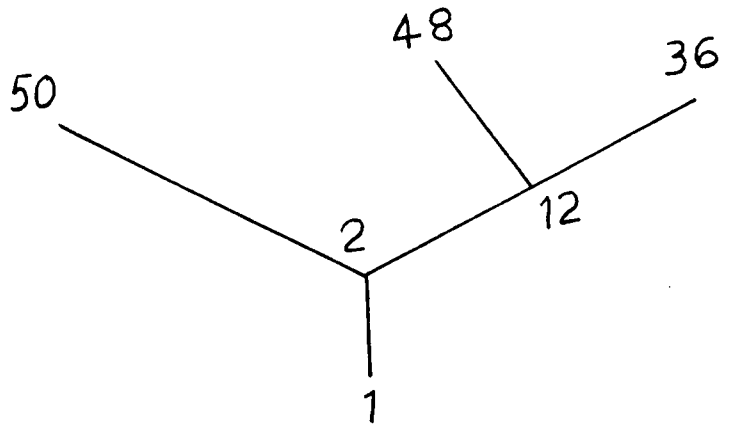


Figure 15. Final network for the GCD tree.

13 is yes, we repeat steps 4, 10, 11, and 12 and stop when the answer to step 13 is no.

Summarizing, INT(X) was used to denote the arc connecting X with its ancestor. DINT(Y,INT(X)) was used to indicate the Manhattan distance between the node Y and the common ancestor of Y and X which lies at a point on the arc X - ANC(X). The previous example gave us a numerical meaning for these terms used in the Wagner and Wiss methods.

4. PROBLEM OF MINIMAL CONNECTOR

We can state this problem in graph theoretic terms as follows: let G be a connected graph, and suppose that to each edge e of G is assigned a non-negative real number u(e), called its measure; then we

wish to find an algorithm for obtaining a spanning tree T whose measure-sum $L(T) = \Sigma\, u(e)$ is as small as possible, the sum being taken over the edges of T. This problem is known as the minimal connector problem. The following algorithm is often used to solve this problem.

Kruskal's Algorithm [7]

"Among the links which are not yet a part of the tree, choose the link which has the least value and which does not form a cycle with those already selected."

To prove it, we proceed by stages, each time choosing the lowest value which does not form a cycle with the links already chosen. By this process, we eventually manage to choose a set of links

$$U_{n-1} = \{u_1, u_2, u_3 \ldots u_{n-1}\}$$

of n-1 links and the graph (X, U_{n-1}) is a tree of minimal value. In fact, we can arrange for all the links to have different values, and if, for example,

$$L(u_1) = L(u_2) = L(u_3)$$

we make the change:

$$L(u_1) \rightarrow L(u'_1) = L(u_1) + \varepsilon$$

$$L(u_2) \rightarrow L(u'_2) = L(u_2) + 2\varepsilon$$

$$L(u_3) \rightarrow L(u'_3) = L(u_3) + 3\varepsilon$$

giving ε a fairly low value.

In the same way we can obtain a complete graph merely by adding links u'_k of sufficient values that

$$L(u'_k) > \sum_{u_i \in U} L(u_i)$$

In the new graph, there is one, and only one, tree $H = (X, U_{n-1})$ of minimal total value.

A problem which resembles the minimal connector problem is the well-known traveling salesman problem which asks for the solution of the following problem: a man wishes to visit n given cities; how can he visit them all at least once in such a way that the total distance traveled is as small as possible?

As a specific example of this problem, let us assume that n = 6 cities and the distances between these cities are as shown in Figure 16 and Table 11.

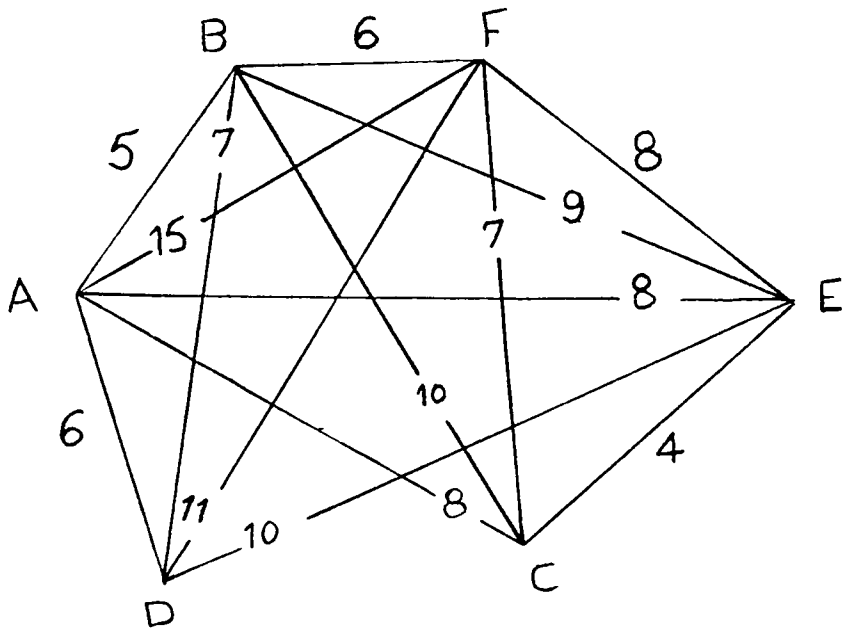


Figure 16. Distance graph for the cities A, B, C, D, E, and F.

Table 11. Distance Matrix for the Cities A, B, C, D, E, and F

	(1) A	(2) B	(3) C	(4) D	(5) E	(6) F
(1) A	100	5	8	6	8	15
(2) B	5	100	10	7	9	6
(3) C	8	10	100	∞	4	7
(4) D	6	7	∞	100	10	11
(5) E	8	9	4	10	100	8
(6) F	15	6	7	11	8	100

Let us make use of Kruskal's algorithm in this given example. In Table 11 we look for the link with the least value. By representing the value of link (i, j) as C_{ij}, we find $C_{35} = 4$ as the minimal value of the table, and we shall therefore start along this link which will be marked I on Figure 17. We next have to select the minimal value

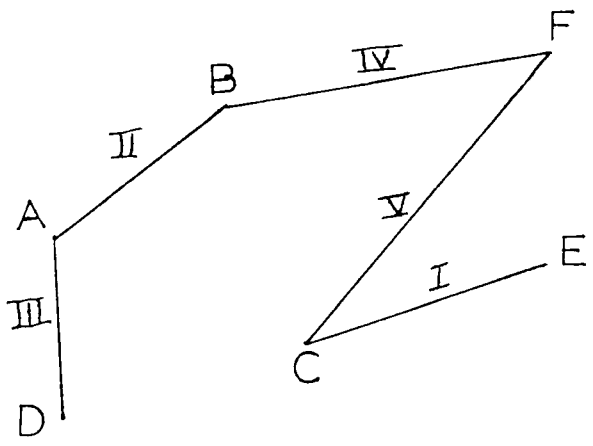


Figure 17. The shortest distance which links the cities A, B, C, D, E, and F.

between the remaining cities; so, we link A to B = 5 which will be marked II. We next have the choice of AD or BF each of which equal 6. If we mark both, no matter which one first, we will not form a cycle. Next we mark FC, but not BD, for that would form a cycle. None of the rest of the links will be marked, for they would form cycles. CD will not be considered because there is no connection between C and D in the graph; observe that CD was given a value of α in Table 11. Hence, the optimal tree is that shown in Figure 17 which gives the shortest path of 4 + 5 + 6 + 6 + 7 = 28.

Kruskal's algorithm could be used for such a purpose, but this algorithm contains a serious practical drawback due to the continual necessity of finding "the shortest link." In cases where a large number of points have to be connected, a great many links exist, of which only one (or at most a very few) is the shortest link. For this reason we will use an alternative method known as Sollin's algorithm.

Sollin's Algorithm

"We shall proceed in stages by joining any vertex to the one nearest to it, and by so doing form subtrees, none of these subtrees having a vertex in common with any other. We shall then treat these subgraphs as vertices, repeating the algorithm until the subtree becomes a tree of the given graph" [2].

The operation of joining any vertex or subtree to the one nearest to it is a simple one involving only a few comparisons. The number of such comparisons is, in practice, independent of the number of vertices to be connected.

Let us now consider the previous example to see how this algorithm can be used (the one shown in Figure 16, page 44, and Table 11, page 45, and reproduced in Figure 17).

Starting arbitrarily at B, we find that the "nearest" vertex is A. Connecting B with A we obtain the partial subtree BA. We now take any point other than B or A, let it be C, the nearest vertex to which is E; connect CE. Choosing a further point other than A, B, C, or E, we select D which has A as its nearest vertex, and connect DA. Finally, we take the last remaining point F, to which the nearest point is B, and connect FB. We have thus formed two subtrees X_1 and X_2 as shown in Figure 18.

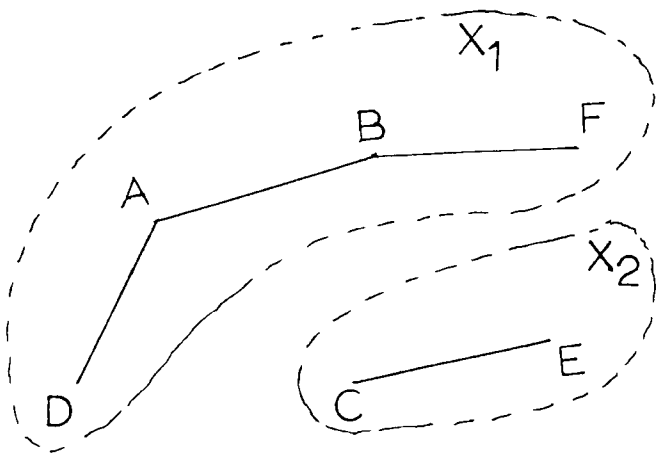


Figure 18. Forming subtrees.

An examination will show that the shortest link between X_1 and X_2 is CF, which is worth 7. We link CF to form the subtree X_3 (Figure 19), which is the same optimal tree obtained by Kruskal's method.

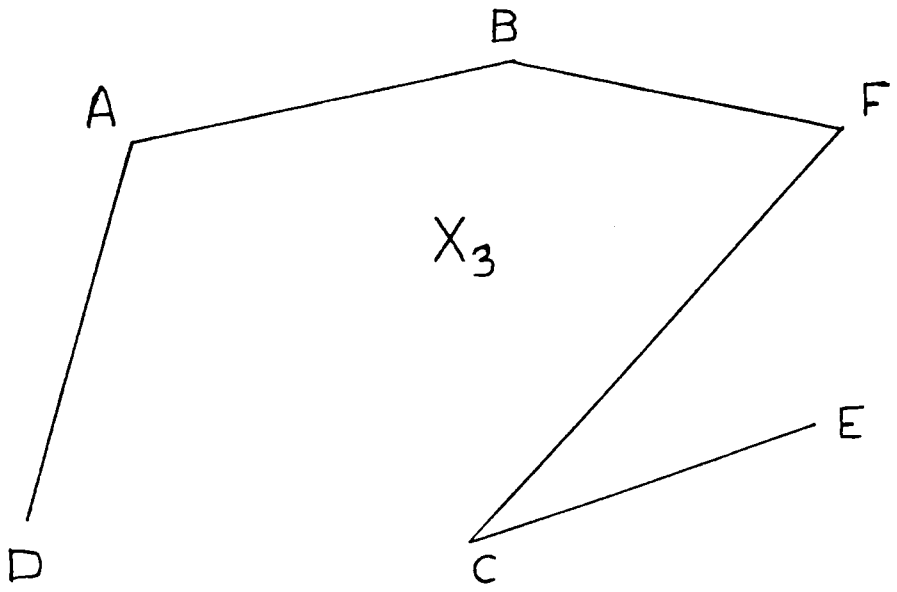


Figure 19. The final optimal tree.

Programming the Problem of the Minimal Connector for the Computer

The Wagner rootless method could apply to this problem by some modification of its computer program (Appendix B).

We feed to the program the distance matrix between the cities as the Manhattan difference matrix computed in the second step of the method. By choosing the largest difference between two points, we form an interval; then we connect the point with the largest interval difference to this interval through a hypothetical ancestor. By doing this we form a subtree with three terminal nodes and three intervals connecting these nodes to a hypothetical ancestor (Steiner point) which lies in between as a median. Then, we start connecting the rest of the points to these intervals selecting the

minimal interval difference each time until we construct the entire tree.

This method will work with the minimal connector problem in many realistic cases when the graph for these cities is complete, that is there is a direct road (link) between each city and every other city. It will fail to work if at least one link is missing. In this case we try to give big numbers to missing links (to fill in the table) to indicate the impossibility of linkage. For example, considering Table 11, page 45, again with a change of the link BC = 2, suppose we are trying to connect nodes with the interval DB (considering B as an ancestor of D). According to the rootless Wagner algorithm used, the computer will determine all interval differences and find that

$$DINT(E,INT(D)) =$$

$$= [d(E,D) + d(E,B) - d(D,B)]/2$$

$$= (10 + 9 - 7)/2 = 6$$

and

$$DINT(C,INT(D)) =$$

$$= [d(C,D) + d(C,B) - d(D,B)]/2$$

$$= (99 + 2 - 7)/2 = 47$$

where no link is possible between C and D, so that c(C,D) = 99, a number which is very much larger than any number in the table to indicate its impossibility.

When the computer chooses the minimal interval difference, it is obvious that it will not choose the point C to connect to the interval DB because of its large interval difference, even though C is very close to B, one of the end points of the interval DB.

So, the method then will fail to select a point with a missing link because of its large interval difference while it could be geographically the closest to the interval if the missing linkage exists. We conjecture that by redefining interval difference between a node X and an interval INT(Y) to be

DINT(X,INT(Y)) = min {d(X,Y), d(X,ANC(Y)}

that the rootless Wagner method applied to even problems with missing links.

So, Sollin's method was programmed on the computer in a separate program (Appendix D) to solve the problem in general whether or not missing links exist, and for comparison purposes. Appendix F shows a comparison between the output of the modified rootless Wagner and the output of Sollin's for the same example (with no missing links) of a traveling salesman. Both outputs came out with the same tree.

It is useful to mention that both programs apply to a wide class of problems of minimal connector in general. Considering the traveling salesman problem as a problem of this class, a restriction should be made when the final tree constructed has one or more nodes with degree three or more. The tree will be the right solution for the traveling salesman if and only if the degree of each node does not exceed two.

Sollin's program will be useful in some other wide applications to solve different problems such as the problem of constructing a network of television relays for a county or a

planned network of roads on a map. In all such cases this program will be able to choose the minimum network and reduce the cost.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Farris, J. S. "Methods for computing Wagner trees." New York: Biol. Sci., State University, Stony Brook, 1970.

2. Kaufmann, A. Graphs, Dynamic Programming and Finite Games. Translated by Henry C. Sneyd. New York, London. Academic Press, 1967.

3. Nelson, H. Charles and Hanson, F. John. "The Genus Perlomyia." Kansas: The Journal of the Kansas Entomological Society, Vol. 46, April 1973.

4. Simpson, G. G. The Major Features of Evolution. New York: Columbia University Press, 1953.

5. Sneath, P. H. A. and Sokal, R. R. Numerical Taxonomy: The Principles and Practice of Numerical Classification. San Francisco: W. H. Freeman and Co., 1973.

6. Wagner, W. H. Problems in the Classification of Ferns. Toronto: University of Toronto Press, 1961.

7. Wilson, Robin J. Introduction to Graph Theory. New York, London: Academic Press, 1972.

APPENDIXES

APPENDIX A

PROGRAM WAGNER

```
C   ************************************************************
C   *                                                          *
C   *  A PROGRAM FOR COMPUTING AN ESTIMATED EVOLUTIONARY       *
C   *     TREE IN BIOLOGY USING WAGNER METHOD                  *
C   *                                                          *
C   ************************************************************
C
C   ***  DIMENSIONS   ***
       DIMENSION X(49,49),
      PD(99),INT(99),IANC(99),
      PIY(99),JY(99)
       DATA IY(1),MIN,SMAL,MEN,M/49,2*99,2*00/,N,LOO/2*1/
C
C    ***  TO READ THE NUMBER OF CHARACTERS FOR EACH SPECIE **
       READ (1,40)NC
C
C    ***  TO READ THE CHARACTERS OF THE SPECIES IN THE ARRAY*
     2 READ (1,1,END=7) (X(N,J),J=1,NC)
       WRITE (3,4) (X(N,J),J=1,NC)
       N=N+1
       GO TO 2
     7 N=N-1
C    ***  TO NUMBER THE ANCESTOR OF THE TREE  ***
       K=N+2
C
C    ***  TO READ THE CHARACTERS OF THE ANCESTOR OF THE TREE
       READ (1,3) ( X(K,J),J=1,NC)
C    ***  INITIALIZATION   ***
       KI=N+1
       IANC(KI)=K
       NO=N+1
       NN=N
C    ***  COMPUTE THE DIFFERENCE BETWEEN EACH SPECIES AND K
C              SELECTING THE SMALEST DIFF.   ***
     9 DO 15 I=1,NN
       IF (M.EQ.NN) GO TO 90
       DO 8 KL=1,M
       IF(I.EQ.IY(KL)) GOTO 15
     8 CONTINUE
       D(I)=DIFF(X,I,K,NC)
       IF (D(I).GE.SMAL) GO TO 99
       I1=I
       SMAL=D(I)
    99 WRITE (3,22)I,D(I)
    15 CONTINUE
C    ***  CONNECT THE SMALLEST TO K TO FORM A NETWORK  ***
```

```
      WRITE(3,23)I1,SMAL
      N=N+1
      M=M+1
      IY(M)=I1
      SMAL=99.
      DO 31 J=1,NC
   31 X(N,J)=X(I1,J)
      IF (LOO.EQ.2) GOTO 44
      N=N+1
      LOO=2
      GO TO 9
C
C    ***  FIND THE INTERVAL DIFFERENCE BETWEEN THE SELECTED
C         SPECIES AND OTHER SPECIES PLACED IN THE NETWORK,
C         EXCLUDING K, SELECTING THE MINIMAL AND CONNECT IT
C         TO THE NETWORK    ***
   44 NO=N-1
      DO 66 J=KI,NO
      IF(J.EQ.K) GO TO 66
      INT(J)=(DIFF(X,N ,J,NC)+DIFF(X,N ,IANC(J),NC)-DIFF(X,J
     C,IANC(J),NC))/2.0
      IF (INT(J).GE.MIN) GO TO 55
      J1=J
      MIN=INT(J)
   55 WRITE (3,20)I1,J,INT(J)
   66 CONTINUE
      WRITE(3,21)J1,MIN
C
C    *** CONSTRUCTING A HYPOTHETICAL ANCESTOR FOR THE
C         SELECTED AND GIVING IT A NUMBER L    ***
C         GIVING IT A NUMBER L
      L=N+1
      DO 50 J=1,NC
      IF (X(I1,J).EQ.99.OR.X(J1,J).EQ.99) GO TO 56
      IF (X(N,J)-X( J1,J))60,60,80
   60 X(L,J)=X(N ,J)
      GO TO 50
   80 X(L,J)=X(J1,J)
      GO TO 50
   56 X(L,J)=99
   50 CONTINUE
C
C    ***  SELECT AN ANCESTOR FROM THE NETWORK FOR L  ***
      MEN=MEN+1
      IANC(L)=L-2*MEN
      IF (K.NE.IANC(J1)) IANC(L)=IANC(J1)
```

```
      IANC(J1)=L
      IANC(N )=L
   30 WRITE (3,5)L,  I1,J1,(X(L,J),J=1,NC)
      MIN=99
      SMAL=99
      N=N+1
C
C    ***  FORMATS  ***
    1 FORMAT(38F2.0)
    4 FORMAT ('0',38F3.0)
    3 FORMAT(40F2.0)
    5 FORMAT ('0',I2,' FOR',3X    ,I2,'_',I2,1X,38F3.0)
   20 FORMAT (2I4,'   INTERVAL',I4 )
   21 FORMAT ('0',I3,'   MINIMAL=',I4)
   22 FORMAT('0',I3,'DIFF=',F5.0)
   23 FORMAT('0',I3,'  SMALEST=',F5.0)
   40 FORMAT (I2)
C
C    ***  GO BACK TO REPEAT THE PROCESS UNTIL ALL SPECIES ARE
C         CONNECTED TO THE NETWORK   ***
      GO TO 9
   90 STOP
      END
```

DOCUMENTATION OF THE PROGRAM

Program Description

The sample problem used in this program consists of 24 species each of which has 38 characters to compare. The root of the tree considered as the most primitive with zero character states. The program consists of the main routine and a function subprogram named DIFF for computing the difference.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 44 observations (species).
2. Up to 38 characters for each specie.
3. Each character can be represented by an integer number between 0 and 98 according to its advancement.

Input Description

1. One leading card containing the number of characters to be read is read by the main program. This card contains the number punched in column 1-2.

2. Data cards follow the leading card with the character states occupying two columns each, starting in column 1-2, and using the format (40F2.0) to read it.

3. The root card is punched the same way as the data cards separated from them by a file end indication card.

Deck Setup

The deck setup is in the following order:

1. The main program.
2. The function DIFF.
3. The leading card.
4. The data cards.
5. File end indication card.
6. The root card.

Output Description

The output of the sample program for evolutionary tree includes:

1. Print of the original matrix of the species and their characters.
2. The two species number which have the largest advancement index.
3. The ancestor of the species in (2), giving it a rank starts from n+1 up, increasing by 1.
4. Repeating the process from (2)-(3) for all species and ancestors excluding what was already selected in step 2 until no further species are left.

Program Modification

Program capacity can be increased or decreased by making changes in dimension statements and in some initializations. More than one data card can be used for each species and more than one line can be printed to list the characters of the ancestor.

The following general rules are supplied in terms of the sample

problem:

1. The dimension of array X must be greater than or equal to the product of (2n)x(nc) where n is the number of the species in the sample and nc is the number of the characters for each.

2. The subscript of IY and JY must be greater than or equal to 2n.

3. The subscript of C and ADV must be greater than or equal to n(n-1)/2.

4. The ranks of the ancestors increasing by 1 start from n+1.

5. Changes of the input and output format must consider the capacity of the input cards and the printing paper.

APPENDIX B

PROGRAM ROOTLESS WAGNER

```
C
C   ************************************************************
C   *  A PROGRAM FOR COMPUTING AN ESTIMATED EVOLUTIONARY TREE
C        IN BIOLOGY USING WAGNER ROOTLESS METHOD
C   *
C   ************************************************************
C
C   ***  DIMENSIONS   ***
      DIMENSION X(49,49),NOW(10),
     PC(500),ADV(500),
     PIANC(99),DINT(99),
     PIY(99),JY(99),IX(99)
C
C    ***  SET A FUNCTION F WHICH COMPUTES THE ADVANCEMENT ***
C      *** INDEX AND CALLS A  FUNCTION DIFF                ***
      F(I,J,K)=ABS(
     P          (DIFF(X,I,J) +DIFF(X,I,K) -DIFF(X,J,K))/2.0)
C
C    ***  TO READ THE NUMBER OF CHARACTERS FOR EACH SPECIE **
      READ (1,40)NC
C
C    ***  TO READ THE CHARACTERS OF THE SPECIES IN THE ARRAY
      N=1
    2 READ (1,1,END=7) (X(N,J),J=1,NC)
      WRITE (3,4) (X(N,J),J=1,NC)
      N=N+1
      GO TO 2
    7 N=N-1
C
C    ***  INITIALIZATION   ***
      NN=1
      IY(1)=0
      JY(1)=0
      IX(1)=0
      IN=1
      BIG=-1
      LOOP=1
      N1=N-1
      M=0
C
C    ***  COMPUTE THE DIFFERENCES BETWEEN ALL SPECIES &
C         SELECTING THE LARGEST  ***
   14 DO 15 I=1,N1
      K=I+1
      DO 13 J=K,N
      M=M+1
```

```
      C(M)=DIFF(X,I,J)
      WRITE(3,20)I,J,C(M)
      IF  (C(M).LE.BIG) GO TO 13
      I1=I
      J1=J
      BIG=  C(M)
   13 CONTINUE
   15 CONTINUE
      IY(NN)=I1
      JY(NN)=J1
      WRITE (3,18)I1,J1,BIG
C
C    ***  COMPUTING THE ADVANCEMENT INDEX BY CALLING F FOR
          THE REMAINING SPECIES NOT IN THE NETWORK   ***
  100 ME=0
      SMAL=99
      BAG=-1
      DO 21 I=1,N
      DO 8 KL=1,N
      IF((I.EQ.IY(KL)).OR.(I.EQ.JY(KL))) GO TO 21
      IF(I.EQ.IX(KL)) GO TO 21
    8 CONTINUE
      ME=ME+1
      ADV(ME)=F(I,I1,J1)
      WRITE (3,22)I,I1,J1,ADV(ME)
      IF (ADV(ME).LE.BAG) GO TO 21
      BAG=ADV(ME)
      IE=I
   21 CONTINUE
      WRITE (3,23)IE,I1,J1,BAG
      IX(NN)=IE
      NN=NN+1
      IF (NN.GT.2) GO TO 200
C
C    ***  CALLING THE SUBROUTINE CAT THAT COPUTE THE ANCESTOR
C         FOR THE LARGEST SELECTED SPECIES   ***
      L=N
      L=L+1
      CALL CAT(X,L,IE,J1,NC,IANC)
      IANC(L)=I1
      GO TO 100
  200 CONTINUE
C
C    ***  EXCLUDING THE SPECIES WHICH IS NOT ON THE NETWORK *
      DO 300 I=1,L
      IF (I.GT.N) GO TO 3
      IF(I.EQ.IE    ) GO TO 300
      IF (           I.EQ.J1)GO TO 3
      NO=NN-1
      DO 11 J=1,NO
      IF (I.EQ.IX(J)) GO TO 3
```

```
   11 CONTINUE
      GO TO 300
    3 CONTINUE
C
C    *** COMPUTE THE INTERVAL ON NETWORK WHICH THE LAST
C          SELECTED IS CLOSEST TO   ***
      DINT(ME)=F(IE,I,IANC(I))
      WRITE(3,31)IE,I,IANC(I),DINT(ME)
      IF(DINT(ME).GE.SMAL)GO TO 300
      SMAL=DINT(ME)
      ID=IE
      I2=I
      J2=IANC(I)
  300 CONTINUE
      WRITE (3,32)ID,I2,J2,SMAL
C
C    ***  CALL SUBROUTINE CAT TO CONSTRUCT THE ANCESTOR    ***
      L=L+1
      CALL CAT(X,L,IE,I2,NC,IANC)
      IF (NN.EQ.N-1) GO TO 90
C
C    ***  FORMATS  ***
  201  FORMAT (10I4)
    1  FORMAT(38F2.0)
    4  FORMAT ('0',38F3.0)
   18  FORMAT('OLARGEST ',I2,'-',I2,'=',F4.0)
   20  FORMAT(/' DIFFERENCE',I3,'-',I2,'  IS',F4.0)
   22  FORMAT('0',I2,'INT',I2,I3,'=',F5.0)
   23  FORMAT('OLARGE',I2,I4,'-',I2,'=',F4.0)
   31  FORMAT('0',I3,' INT',2I3,' IS',F4.0)
   32  FORMAT('OSMAL ',I2,I4,'-',I2,'=',F4.0)
   40  FORMAT (I2)
C
C    ***  GO BACK TO REPEAT THE PROCESS UNTIL ALL SPECIES ARE
C          CONNECTED TO THE NETWORK    ***
      GO TO 100
   90 STOP
      END
```

DOCUMENTATION OF THE PROGRAM

Program Description

The evolutionary tree program consists of the main program and a function named DIFF to compute the differences.

Capacity

The capacity of the sample program and the format required for data have been set up as follows:

1. Up to 31 observations (species).
2. Up to 38 characters for each species.
3. Each character can be represented by any integer number between 0 and 98 according to its advancement.

Input Description

1. One leading card containing the number of characters to be read by the main program. This card contains the number punched in column 1-2.
2. Two data cards follow the leading card with the characters occupying two columns each, starting in column 1-2 in the first card; and the name of the species in columns 1-20 in the second card.
3. The root card punched the same was as the data cards separated from them by a file end indication card.

Deck Setup

The deck setup is the following order:

1. The main program.
2. The function DIFF.

3. The leading card.

4. The data cards.

5. End of file indication card.

6. The root card.

Output Description

The output of the sample program for evolutionary tree includes:

1. Print of the original matrix of the species and their characters.

2. List of the differences between the different species and K. (The rank of K is n+2.)

3. Selecting the species of the smallest difference and connecting it to K to form a network with a rank = n+1.

4. The same list of differences in (2) excluding what was selected before.

5. Selecting the species of the next lower difference from K which has not been selected before.

6. List of the interval difference between the species selected in (5) and other nodes in the network excluding k (node is any species or ancestor which previously connected to the network).

7. Selecting the node of the minimal interval difference in the network.

8. List of the rank and characters of the ancestor of (5) and (7).

9. Go back to (4) and repeat the printing.

A tree can be constructed by setting K as the root ancestor, then, whenever we have an ancestor we replace it for its descendents.

Program Modification

Program capacity can be increased or decreased by making changes in dimension statements and in some initializations. More than two data cards can be used for each species and more than one line can be printed to list the characters of the ancestor.

The following general rules are supplied in terms of the sample problem:

1. The dimension of array X must be greater than or equal to the product of (3n+1) x (nc) where n is the number of the species in the sample and nc is the number of the characters for each.

2. The subscript of IANC must be greater than 3n.

3. The subscript of D must be greater than or equal to n.

4. The subscripts of IY, JY must be greater than or equal to 2n.

5. The rank of the root ancestor is n+1.

6. The ranks of the other evolutionary ancestors in the tree increase by one starting from n+2.

7. Changes of the input and output format must consider the capacity of the input cards and the printing paper.

APPENDIX C

PROGRAM WISS

```
C   ************************************************************
C   *                                                          *
C   *  A PROGRAM FOR COMPUTING AN ESTIMATED EVOLUTIONARY TREE  *
C   *     IN BIOLOGY USING WISS METHOD                         *
C   *                                                          *
C   ************************************************************
C
C   ***  DIMENSIONS   ***
C
      DIMENSION X(49,59),
     PC(999),ADV(999),
     PIY(99),JY(99)
C
C    ***  TO READ THE NUMBER OF CHARACTERS FOR EACH SPECIE **
C
      READ (1,40)NC
C
C    ***  TO READ THE CHARACTERS OF THE SPECIES IN THE ARRAY
C
      N=1
    2 READ (1,1,END=7) (X(N,J),J=1,NC)
      WRITE (3,4) (X(N,J),J=1,NC)
      N=N+1
      GO TO 2
    7 N=N-1
C
C **      TO READ THE CHARACTERS OF THE ANCESTOR OF THE TREE
C
      READ (1,70)(X(49,J),J=1,NC)
C
C    ***  INITIALIZATION   ***
C
      NN=1
      IY(1)=0
      JY(1)=0
      IN=1
      BIG=-1
      LOOP=1
    9 N1=N-1
      M=0
C
C    ***  COMPUTE THE DIFFERENCE BETWEEN ALL SPECIES AND THE
C         ANCESTOR K EXCLUDING WHAT WAS ALREADY CONNECTED TO
C         THE NETWORK   ***
C
   14 DO 15 I=1,N1
```

```
      K=I+1
      DO 13 J=K,N
      DO 3 KL=1,99
      IF((I.EQ.IY(KL)).OR.(I.EQ.JY(KL))) GO TO 15
      IF((J.EQ.IY(KL)).OR.(J.EQ.JY(KL))) GO TO 13
    3 CONTINUE
      M=M+1
C
C    ***  CALL THE FUNCTION DIFF TO COMPUTE THE DIFFERENCES
C                  BETWEEN  EACH TWO SPECIES  ***
C
      C(M)=DIFF(X,I,J,NC)
C
C    ***  COMPUTE THE ADVANCEMENT INDEX, SELECT THE LARGEST
C
      ADV(M)=ABS(DIFF(X,I,49,NC)+DIFF(X,J,49,NC)-DIFF(X,I,J,
     CNC))/2.0
      IF(ADV(M).LE.BIG) GO TO 13
      I1=I
      J1=J
      BIG=ADV(M)
   13 CONTINUE
   15 CONTINUE
C
C    ***  STORE THE MEMBERS OF THE LARGEST ADVANCEMENT INDEX
C         IN ARRAYS   ***
C
      IY(NN)=I1
      JY(NN)=J1
      WRITE (3,18)I1,J1,BIG
C
C    ***  CONSTRUCT A HYPOTHETICAL ANCESTOR BY TAKING THE
C                        LOWEST VALUE OF EACH CHARACTER   ***
C
      DO 30 I=LOOP,IN,2
      L=N+1
      DO 50 J=1,NC
      IF (X(I1,J).EQ.99.OR.X(J1,J).EQ.99) GO TO 55
      IF (X(I1,J)-X(J1,J))60,60,80
   60 X(L,J)=X(I1,J)
      GO TO 50
   80 X(L,J)=X(J1,J)
      GO TO 50
   55 X(L,J)=99
   50 CONTINUE
   30 WRITE (3,5)L,I1,J1,(X(L,J),J=1,NC)
      N=N+1
      LOOP=LOOP+2
      BIG=-1
      NN=NN+1
      IN=IN+2
```

```
      IF (LOOP.GE.N) GO TO 90
C
C    ***  FORMATS  ***
C
     1 FORMAT(38F2.0)
     4 FORMAT ('0',38F3.0)
     5 FORMAT ('0',I2,' FOR',I2,'_',I2,1X,40F3.0)
    18 FORMAT('OLARGEST ',I2,'-',I2,'=',F4.0)
    40 FORMAT (I2)
    70 FORMAT (40F2.0)
C
C    ***  GO BACK TO REPEAT THE PROCESS UNTIL ALL SPECIES ARE
C         CONNECTED TO THE NETWORK
C
      GO TO 9
    90 STOP
      END
```

DOCUMENTATION OF THE PROGRAM

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 49 observations (species).
2. Up to 38 characters for each specie.
3. Each character can be represented by an integer number between 0 and 98 according to its advancement.

Input Description

1. One leading card containing the number of characters to be read is read by the main program. This card contains the number punched in column 1-2.
2. Data cards follow the leading card with the characters occupying two columns each, starting in column 1-2, and using the format (40F2.0) to read it.

Deck Setup

The deck setup is in the following order:

1. The main program with the statement function at the very beginning.
2. The function subprogram DIFF.
3. The subroutine CAT.
4. The leading card contains.
5. The data cards.

Output Description

The output of the sample program for the rootless evolutionary tree includes:

1. Print of the original matrix of the species and their characters.

2. List of the differences between each two species.

3. Selection of the two species with the largest difference to form the first interval.

4. List of the advancement index of each species and the interval of the two selected in (3).

5. Selecting the largest in (4) and connecting it to the interval of the two selected in (3) to form a network.

6. List of the rank and characters of the ancestor of (5) and the second of the two species of (3).

7. The same list of (4) excluding what was already connected to the network.

8. Repeating (5) through (7).

Program Modification

Program capacity can be increased or decreased by making changes in dimension statements and in some initializations. More than one card can be used for each specie and more than one line can be printed to list the characters of the ancestor.

The following general rules are supplied in terms of the sample problem:

1. The subscript of array X must be greater than or equal to the product of (2n) x (nc) where n is the number of the species in the sample and nc is the number of the characters for each.

2. The subscript of IANC must be greater than 2n.

3. The subscript of C must be greater than or equal to n(n-1)/2.

4. The subscript of ADV, IX must be greater than N.

5. The subscript of DINT must be greater than or equal to 2N-1.

6. The subscript of IY, JY must be greater than or equal to 2N.

7. The ranks of the ancestors increasing by 1 start from n+1.

8. Changes of the input and output format must consider the capacity of the input cards and the printing paper.

APPENDIX D

PROGRAM SOLLIN

```
C       THIS PROGRAM FINDS AND COMPUTS THE MINIMAL DISTANCE
C          BETWEEN A SET OF POINTS
        IMPLICIT INTEGER  (A-Z)
        DIMENSION S(20,20),T(20,20)
C       T IS THE ORIGINAL DISTANCE MATRIX
C       C IS THE CONNECTION MATRIX AT EACH STAGE
C       S IS A STORAGE MATRIX FOR DISTANCES BETWEEN
C          GENERALIZED NODES
        READ (1,1)N,F
    1   FORMAT (2I3)
C       N IS ORIGINAL NUMBER OF NODES
C       F IS A DISTANCE FACTOR LARGER THAN ANY ENTRY OF T
        WRITE (9,10)
   10   FORMAT ('1ORIGINAL DISTANCE MATRIX'/)
        DO 13 I =1,N
        READ (1,2) (T(I,J),J=1,N)
    2   FORMAT (20I2)
   13   WRITE (9,8)( T(I,J),J=1,N)
    8   FORMAT (20I4)
        CALL GRAP(T,N,F,S,K)
  200   IF (K.LE.2) GO TO 300
        N=K
        DO 3 I=1,K
        DO 3 J=1,K
    3   T(I,J)=S(I,J)
        WRITE (9,4)
    4   FORMAT(/' NEW GENERALIZED DISTANCE MATRIX'/)
        DO 6 I=1,K
    6   WRITE (9,8) (T(I,J),J=1,K)
        CALL GRAP (T,N,F,S,K)
        GO TO 200
  300   WRITE (9,7)
    7   FORMAT(/' FINAL CONNECTION MADE')
        STOP
        END
```

APPENDIX E

SUBROUTINES

FUNCTION DIFF

```
C    ***  A FUNCTION FOR COMPUTING THE DIFFERENCE BETWEEN ***
C      ***                  ANY TWO SPECIES                 ***
C
C      THIS FUNCTION WILL USE FOUR ARGUMENTS DIFF(X,L,M,NC)
C         WHEN IT IS CALLED BY WISS PROGRAM
       FUNCTION DIFF(X,L,M)
       DIMENSION X(49,49)
       DIFF=0
C
C
C    *** DO NOT COMPUTE THE UNKNOWN CHARACTER  ***
C     **      REPRESENTED BY 99     ***
       DO 10 J = 1,3
       IF (X(L,J).EQ.99.OR.X(M,J).EQ.99) GO TO 10
C
C    *** COMPUTE THE DIFFERENCES  ***
       DIFF=DIFF+ABS(X(L,J)-X(M,J))
    10 CONTINUE
       RETURN
       END
```

SUBROUTINE CAT

```
C    * SUBROUTINE TO CONSTRUCT AN ANCESTOR BY SELECTING THE
C                LOWEST CHARACTER IN BOTH SPECIES    ***
C    *
      SUBROUTINE CAT(X,L,I1,J1,NC,IANC)
      DIMENSION X(49,49),IANC(99)
      DO 50 J=1,NC
      IF (X(I1,J).EQ.99.OR.X(J1,J).EQ.99) GO TO 55
      IF (X(I1,J)-X(J1,J))60,60,80
   60 X(L,J)=X(I1,J)
      GO TO 50
   80 X(L,J)=X(J1,J)
      GO TO 50
   55 X(L,J)=99
   50 CONTINUE
   30 WRITE (3,5)L,I1,J1,(X(L,J),J=1,NC)
    5 FORMAT ('0',I2,' FOR',I2,'_',I2,1X,40F3.0)
      IANC(L)=IANC(J1)
      IANC(I1)=L
      IANC(J1)=L
      RETURN
      END
```

SUBROUTINE FACTO

```
C
C   *** THIS SUBROUTINE ASSIGNS THE PRIME NUMBERS UP TO A
C         LIMIT  WE SELECT AND DECOPOSES A SET OF NUMBERS TO
C         ITS PRIME FACTORS    ***
C
      SUBROUTINE FACTO (M)

      REAL M(49,49)
      DIMENSION IPR(100)
      DIMENSION NUM(100),          NO(100)
      IPR(1)=2
      L=1
      DO 16 K=3,100
      KL=K-1
      DO 10 ME=2,KL
      DO 19 LO=1,L
      IP=IPR(LO)
      IF (IP.EQ.ME)GO TO 15
 19   CONTINUE
      GO TO 10
 15   KD =K/ME
      IF (K.EQ.KD*ME)GO TO 16
 10   CONTINUE
      L=L+1
      IPR(L)=K
 16   CONTINUE
      WRITE (3,17) (IPR(I),I=1,10)
 17   FORMAT ('1',7X,10I4//)
      READ (1,7)(NUM(I),I=1,10)
      DO 5 J=1,10
      NO(J)=NUM(J)
      DO 5 I=1,10
      N=0
 3    MUM=NUM(J)/IPR(I)
      IF(NUM(J).NE.IPR(I)*MUM) GO TO 4
      NUM(J)=MUM
      N=N+1
      IF(NUM(J).EQ.1) GO TO 4
      GO TO 3
 4    M(J,I)=N
 5    CONTINUE
      DO 9 J=1,10
 6    FORMAT (1X,I4,3X,10F4.0)
 9    WRITE (3,6) NO(J),(M(J,I),I=1,10)
 7    FORMAT (10I4)
      RETURN
      END
```

SUBROUTINE GRAP

```
      SUBROUTINE GRAP (T,N,F,S,K)
C     THIS SUBROUTINE WILLSELECT THE MINIMAL DISTANCES
C        BETWEEN THE NODES , COONECT NODES TO FORM SUBNODES
C        AND TREAT THE SUBNODES AS NEW NODES AND REPEAT
C        UNTIL ALL NODES ARE CONNECTED
      IMPLICIT INTEGER  (A-Z)
      DIMENSION T(20,20),S(20,20),C(20,20),P(20),U(20),Z(20)
      DIMENSION L(20,20)
      DO 247 I=1,N
  247 U(I)=0
C     IF U(I)=0, NODE I IS NOT CONNECTED
      DO 500 I=1,N
      DO 500 J=1,N
  500 C(I,J)=0
      DO 660 I=1,N
      IF (U(I).EQ.1) GO TO 660
      U(I)=1
      D=F
C     D WILL BE USED TO COMPUTE MINIMAL DISTANCE FROM NODE
C     I TO ANOTHER NODE
      J1=I
      DO 580 J=1,N
      IF (J.EQ.I) GO TO 580
      IF (C(J,I).EQ.1) GO TO 580
      IF (T(I,J).GT.D) GO TO 580
      D= T(I,J)
      J1=J
  580 CONTINUE
      WRITE (9,21)I,J1,D
   21 FORMAT(/' CONNECT NODE ',I3,'TO NODE',I3,'DIST',I3)
      U(J1)=1
      C(I,J1)=1
      C(J1,I)=1
C     THIS UPDATES CONNECTIO MATRIX
      DO 650 K=1,N
      IF (K.EQ.I) GO TO 650
      IF (C(J1,K).EQ.0) GO TO 650
      C(I,K)=1
      C(K,I)=1
  650 CONTINUE
  660 CONTINUE
      WRITE(9,22)
   22 FORMAT(/' CURRENT CONNECTION MATRIX'/)
      DO 23 I=1,N
      WRITE (9,20) (C(I,J),J=1,N)
   20 FORMAT (20I4)
   23 CONTINUE
      DO 687 I =1,N
```

```
  687 C(I,I)=1
      K=1
      R=1
C     K INDICATES LIST NUMBER
C     R INDICATES FIRST UNLISTED NODE
      DO 698 I=1,N
  698 Z(I)=0
C     C(I)=0 MEANS NODE I IS UNLISTED
  700 IF (R.EQ.N+1) GO TO 850
      J=1
      DO 770 M= R,N
      IF (C(R,M).EQ.0) GO TO 770
      L(K,J)=M
      Z(M)=1
      J=J+1
  770 CONTINUE
      J=J-1
      WRITE (9,24)K
   24 FORMAT(/' CLASS NUMBER ',I4)
      WRITE(9,25) (L(K,I),I=1,J)
   25 FORMAT (20I4)
C     NEXT LOOP FINDS FIRST UNLISTED NODE
      DO 820 M=R,N
      IF (Z(M).EQ.1) GO TO 820
      R=M
      GO TO 835
  820 CONTINUE
      R=N+1
  835 P(K)=J
      WRITE (9,26) K,J
   26 FORMAT(/' NUMBER OF NODES IN CLASS ',I3,' IS',I3)
      K=K+1
      GO TO 700
  850 K=K-1
      WRITE(9,27)K
   27 FORMAT(/' NUMBER OF GENERALIZED NODES ',I3)
      IF (K.EQ.1) RETURN
      DO 875 I=1,K
      DO 875 J=1,K
      IF (I.EQ.J) S(I,J)=F
      IF (I.NE.J) S(I,J)=0
  875 CONTINUE
C     COMPUTE DISTANCES BETWEEN GENERALIZED NODES
      KM=K-1
      DO 1040 I=1,KM
      I3=I+1
      DO 1030 J=I3,K
      D=F
      W=P(I)
      DO 990 K1=1,W
      Y=P(J)
```

```
      DO 986 K2=1,Y
      E= T(L(I,K1),L(J,K2))
      IF (D.LE.E)GO TO 986
      D=E
      I1=L(I,K1)
      I2=L(J,K2)
  986 CONTINUE
  990 CONTINUE
      S(I,J)=D
      S(J,I)=D
      WRITE (9,28)I,J,S(I,J)
   28 FORMAT (' DISTANCE FROM GEN NODE ',I3,'TO NODE',I3
     C,' IS ',I3)
      WRITE (9,29) I1,I2
   29 FORMAT(' OBTAIN BY CONNECTING OLD NODES ',I4,',',I4)
 1030 CONTINUE
 1040 CONTINUE
      RETURN
      END
```

APPENDIX F

OUTPUT
FOR THE MINIMAL CONNECTOR PROBLEM

SOLLINS OUTPUT

```
ORIGINAL DATA MATRIX
    99     8     4     7     9     5
     8    99     9    11     8     3
     4     9    99     6    10     7
     7    11     6    99     5     8
     9     8    10     5    99     7
     5     3     7     8     7    99
CONNECT NODE 1 TO NODE 3 DIST 4
CONNECT NODE 2 TO NODE 6 DIST 3
CONNECT NODE 4 TO NODE 5 DIST 5
CLASS NUMBER 1
   1     3
NUMBER OF NODES IN CLASS 1 IS 2
CLASS NUMBER 2
   2     6
NUMBER OF NODES IN CLASS 2 IS 2
CLASS NUMBER 3
   4     5
NUMBER OF NODES IN CLASS 3 IS 2
NUMBER OF ORIGINALIZED NODES   3
DISTANCE FROM GEN NODE  1 TO NODE  2 IS  5
OBTAINED BY CONNECTING OLD NODES  1, 6
DISTANCE FROM GEN NODE  1 TO NODE  3 IS  6
OBTAINED BY CONNECTING OLD NODES  3, 4
DISTANCE FROM GEN NODE  2 TO NODE  3 IS  7
OBTAINED BY CONNECTING OLD NODES  6, 5
NEW GENERALIZED DISTANCE MATRIX
    99     5     6
     5    99     7
     6     7    99
CONNECT NODE 1 TO NODE 2 DIST 5
CONNECT NODE 3 TO NODE 1 DIST 6
```

ROOTLESS WAGNER OUTPUT

MANHATTAN DISTANCE MATRIX

99	8	4	7	9	5
8	99	9	11	8	3
4	9	99	6	10	7
7	11	6	99	5	8
9	8	10	5	99	7
5	3	7	8	7	99

7 FOR 1-3
8 FOR 3-4
9 FOR 4-5
10FOR 1-6
11FOR 6-2

VITA

Abderauf I. Taha was born in Surda, Palestine, on December 16, 1935. He was graduated from Teachers College-Amman in 1956. Following graduation, he worked as a teacher of mathematics in Ramallah High School.

He accepted a position to teach freshman mathematics in Bir-zeit College in 1960. He received a Bachelor of Science degree in mathematics in 1966 from Ain-Shams University in Cairo. After graduation he accepted a teaching position with the Ministry of Education of Kuwait where he wrote a textbook in technical mathematics for vocational schools which is still in use.

He came with a wife and two children to the United States in 1970. He worked as a programmer with The University of Tennessee at Chattanooga Computer Center after study of computer science in CSTI-Chattanooga. He enrolled as a graduate student at The University of Tennessee in September, 1972.