METHODS FOR COMPUTING WAGNER TREES

JAMES S. FARRIS

Abstract

Farris, J. S. (Biol. Sci., State Univ., Stony Brook, N.Y.) 1970. Methods for computing Wagner Trees. Syst. Zool., 19:83–92.—The article derives some properties of Wagner Trees and Networks and describes computational procedures for Prim Networks, the Wagner Method, Rootless Wagner Method and optimization of hypothetical intermediates (HTUs).

The Wagner Ground Plan Analysis method for estimating evolutionary trees has been widely employed in botanical studies (see references in Wagner, 1961) and has more recently been employed in zoological evolutionary taxonomy (Kluge, 1966; Kluge and Farris, 1969). Wagner Trees are one possible generalization of the most parsimonious trees of Camin and Sokal (1965). The Wagner technique is of considerable interest for quantitative evolutionary taxonomists because it is readily programmable and because the type of tree produced can tractably be extended to applications in a variety of novel quantitative phyletic techniques.

In this paper I shall formalize the concept of a Wagner Network and discuss a number of algorithms for calculating such networks. The rationale for the methods described will not be treated extensively here, as it is published elsewhere (Kluge and Farris, 1969).

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, of N a unique node, $f(n)$, such that n is directly derived from $f(n)$ according to the tree, T. We shall call $f(n)$ the *immediate ancestor* of n. Fig. 1(i) depicts a tree with nodes A, B, C, D, E, F, G, and *ancestor function*:

$$f(A) = B$$
$$f(C) = B$$
$$f(B) = D$$
$$f(F) = E$$
$$f(G) = E$$
$$f(E) = D$$

Note that $f(.)$ is not defined for D, the ancestor of the whole tree. 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 $f^K(n) = P$. The function $f^K(.)$ is the K-fold composition of f on itself; i.e. $f^4(x) = f(f(f(f(x))))$. These pairs are just the trees that have a unique "root" and on which every member of N is connected to the tree so that it is a descendent of the "root." The trees we consider are, in short, those that fit the usual taxonomic notion of "tree."

NETWORKS

Trees are directed entities in which the root is presumed to represent a point chronologically prior to any descendent point. We can think of a tree as being specified in two components, the first being the relative position of the nodes in the branching pattern, and the second being the location of the root. If the root is not specified, we have an "undirected tree," or a *network*. A network with a certain set of nodes may correspond to a wide class of trees with the same nodes, each tree differing from the others in the class only in the position of its root. Figs. 1(i) and 1(ii) show two trees that differ only in the location of their roots. Fig. 1(iii) depicts the generalization of both trees: a network with the "same" branching pattern, but with no root specified at all.

The central problem of evolutionary taxonomy is to construct an evolutionary tree.

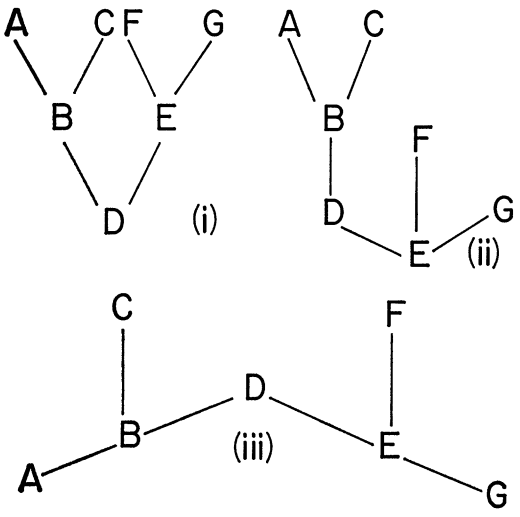

FIG. 1.—Three equivalent Wagner Networks, two of which, ((i) and (ii)) are also Wagner Trees.

One possible way to approach the problem is to first construct a network corresponding to the class of trees that contains the desired tree and then to select a tree from the class by specifying the root node. One possible advantage of this approach is that the form of the network may be used in inferring the proper position of the root.

There are many ways of specifying networks. The most convenient method for our purposes is to subvert the notation for trees. A network, W, will be defined as an ordered pair, W = (N, g), consisting of a class, N, of nodes, and a function, g, that assigns to a member, n, of N a unique node g(n) in N such that n is directly connected to g(n). The g function differs conceptually from the f function only in that no directionality of the linkage between g(n) and n is implied. We will treat only networks, W = (N, g), for which N contains a unique *base element*, Q, such that for any element, n, in N there is a non-negative integer, K, for which $g^K(n) = Q$. These are just the simply connected networks, and they are the only networks that can have the same form as a tree of the type we consider. If T = (N, f) is *any* member of the class of trees that have the same form as a network, W = (N, g), then W can be represented as (N, g*), where g* = f. We have listed above the ancestor function for Fig. 1(i). The ancestor function for Fig. 1(ii) is

$$f(A) = B$$
$$f(C) = B$$
$$f(B) = D$$
$$f(D) = E$$
$$f(F) = E$$
$$f(G) = E.$$

It should be clear that the ancestor function of either Fig. 1(i) or Fig. 1(ii) can serve as the *connection function* of Fig. 1(iii).

LENGTH

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 OTUs. Camin and Sokal (1965) took such an approach in introducing the idea of *most parsimonious* trees as trees with the smallest number of "steps" (changes 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 node A for character i. The *difference* between two nodes, A and B, is defined to be

$$d(A, B) = \sum_i |X(A, i) - X(B, i)|. \quad (1)$$

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 a tree is defined to be

$$L(N, f) = \sum_{n \neq P} d(n, f(n)), \quad (2)$$

where P is the ancestor of the tree. The length, L(N, f), of a tree, T = (N, f), is the sum of the lengths of the internodes of the tree.

If the data matrix, X, takes on only integer values, (1) clearly gives the "number of steps" between A and f(A). Length as defined here is thus related to the concept of parsimony as used by Camin and Sokal. Equation (1), however, is defined for any real-valued data matrix, X, so that we can speak of the length of a tree in terms of continuously coded characters.

The length of a network is defined analogously to the length of a tree. For a network, $W = (N, g)$, the length is

$$L(N, g) = \sum_{n \neq Q} d(n, g(n)), \qquad (3)$$

where Q is the base element of N.

WAGNER TREES AND NETWORKS

The most parsimonious trees of Camin and Sokal (1965) are trees with a minimum number of steps, provided they have no evolutionary reversals. Analogously, we will be interested in trees of minimum length. The assumption of irreversibility of evolution will not be made, however.

We define a *Wagner Tree* for a set, S, of OTUs as a tree, $T = (N, f)$ such that

1) S is a subset of N
2) if $T' = (N', f')$ is any other tree satisfying

(1), then $L(N, f) \leqslant L(N', f')$.

A *Wagner Network* for a set, S, is analogously defined as a network, $W = (N, g)$ such that:

1) S is a subset of N
2) if $W' = (N', g')$ satisfies (1), then $L(N', g') \geqslant L(N, g)$.

Any of the nodes of a network can be used as the root of a corresponding tree without altering its length (since (2) and (3) are the same equation). Since irreversibility of evolution is not assumed, any tree generated from a network in this way is a legitimate candidate as a Wagner Tree. Hence a Wagner Network on S can be converted into a Wagner Tree for S with no change in length. It is thus possible to find a Wagner Tree by finding a Wagner Network. This simplification of the task of finding a tree is possible only if evolutionary reversals are permitted.

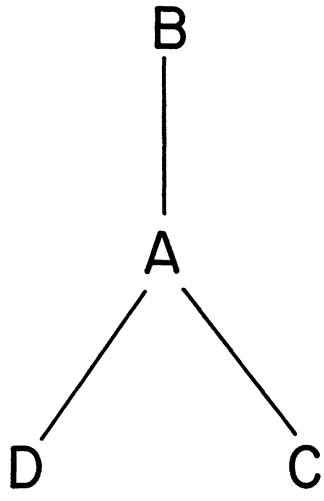

FIG. 2.—A local region of a Wagner Network.

HYPOTHETICAL TAXONOMIC UNITS

The nodes of a Wagner Tree are not required all to be OTUs, and in fact it is often necessary to use nodes not in S in order to achieve a shortest tree for S. The nodes of a Wagner Tree, $T = (N, f)$, that are not OTUs are purely hypothetical objects chosen simply to allow the length of the tree to be minimized. I shall refer to hypothetical nodes as Hypothetical Taxonomic Units (HTUs).

The definition of Wagner Trees implies a useful property of HTUs: the character state values of an HTU are related to the character state values of the OTUs and the other HTUs in a simple way. Consider the four-node network of Fig. 2 and suppose that it is a Wagner Network. Whether Fig. 2 is interpreted as a tree or a network is immaterial. Fig. 2 can be considered as a section of a more extensive network.

The length of the network of Fig. 2 is $d(A, D) + d(A, C) + d(A, B)$. Recalling (1), this can be rewritten as

$$\sum_i |X(A, i) - X(D, i)| +$$

$$\sum_i |X(A,i)-X(C,i)| + \sum_i |X(A,i)-X(B,i)| = \sum_i [|X(A,i)-X(D,i)| + |X(A,i)-X(C,i)| + |X(A,i)-X(B,i)|]. \quad (4)$$

Since (4) is completely additive over characters we can seek values X(A, i) to minimize (4) one character at a time. For character i, let

$a = X(A,i)$

p = the largest of X(B, i), X(C, i), X(D, i)

q = the median of X(B, i), X(C, i), X(D, i)

r = the smallest of X(B, i), X(C, i), X(D, i).

The ith component of (4) is equal to

$$|a-p| + |a-q| + |a-r|. \quad (5)$$

The optimal value of $a = X(A,i)$ minimizes (5). We can see that if a is chosen outside the interval (r, p), the value of (5) must exceed p − r. If a is chosen inside (r, p), the first and last terms of (5) sum to p − r, and if a = q, the total value of (5) is p − r. The optimal value of a is thus q, and in general, the optimal value of X(A, i) is the median of X(B, i), X(C, i), X(D, i) if A is connected just to the nodes B, C, D, of a Wagner Network.

The *median-state property* of HTUs is used in the algorithms described later to specify optimal HTUs by constructing them one character state at a time.

INTERVAL DIFFERENCES

Some calculations for constructing Wagner Trees can be simplified by a relation on the differences between HTUs and OTUs. The relation is a corollary of the median-state property of HTUs.

Suppose that a node, B, is connected to two other nodes, C and D, through an optimal HTU, A (Fig. 2). We know from the argument above that for each character the state of A is the median of the 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,

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

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

$$|X(C,i)-X(D,i)| = |X(A,i)-X(C,i)| + |X(A,i)-X(D,i)|. \quad (6)$$

Summing over characters and using (1), (6) yields

$$d(B,C) = d(A,B) + d(A,C); \quad (7)$$

$$d(B,D) = d(A,B) + d(A,D); \quad (8)$$

$$d(C,D) = d(A,C) + d(A,D). \quad (9)$$

Adding (7) and (8), we obtain

$$d(B,C) + d(B,D) = 2d(A,B) + d(A,C) + d(A,D). \quad (10)$$

By (9), (10) becomes

$$d(B,C) + d(B,D) = 2d(A,B) + d(C,D). \quad (11)$$

Therefore,

$$d(A,B) = (d(B,C) + d(B,D) - d(C,D))(\tfrac{1}{2}). \quad (12)$$

Equation (12) gives the difference between a node, B, and the optimal HTU 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 *interval*, INT(C, D), whose difference from a node, B, may be calculated as

$$d(B, INT(C,D)) = (d(B,C) + d(B,D) - d(C,D))(\tfrac{1}{2}). \quad (13)$$

PRIM NETWORKS

We now consider a number of algorithms for calculating approximations to Wagner Networks. It is important to realize that the current level of development of algo-

rithms for finding Wagner Networks is comparable to the state of methods of finding Camin-Sokal Trees prior to the work of Estabrook (1968): several reliable heuristic programs exist, but no existing algorithm is certain to produce a Wagner Network for an arbitrary set of data.

We shall first treat Prim Networks. A Prim Network, for our purposes, is a Wagner Network, subject to the constraint that the set of nodes, N, is identical to the set of OTUs, S. Thus, no HTUs are constructed. Prim Networks are so called because they were introduced into evolutionary taxonomy by Edwards and Cavalli-Sforza (1963), who named them with reference to the work of Prim (1957).

Prim Networks are quite crude approximations to Wagner Networks, but have the advantages that they can be computed exactly and very efficiently. Prim Networks may be most useful in evolutionary studies as tools for preliminary analyses of data, the more elaborate programs described below being reserved for refining final conclusions. A Prim Network can provide a fairly accurate picture of a Wagner Network. Fig. 3(i) depicts a Wagner Tree for the anuran data of Kluge and Farris (1969), while Fig. 3(ii) shows a tree produced from a Prim Network for the same data. In this instance, the Prim and Wagner Networks indicate virtually identical relationships and differ in length by only 1 unit.

A Prim Network may be computed as follows:

1) Pick an OTU, say Q, as a starting point. It does not matter which OTU is used. Go to 2.

2) Find the OTU in S that is closest to Q. Connect it to Q to form a network with one linkage. Go to 3.

3) Compute the difference between each unplaced OTU and the network. The difference between an OTU, A, and a network, W = (N, g), with nodes in N is defined to be $\min_{n \text{ in } N}(d(A, n))$. Go to 4.

4) Find the OTU that is closest to the network. Add it to the network by connecting it to the node from which it differs least. Go to 5.

5) If any OTUs remain unplaced, go to 3. Otherwise, stop.

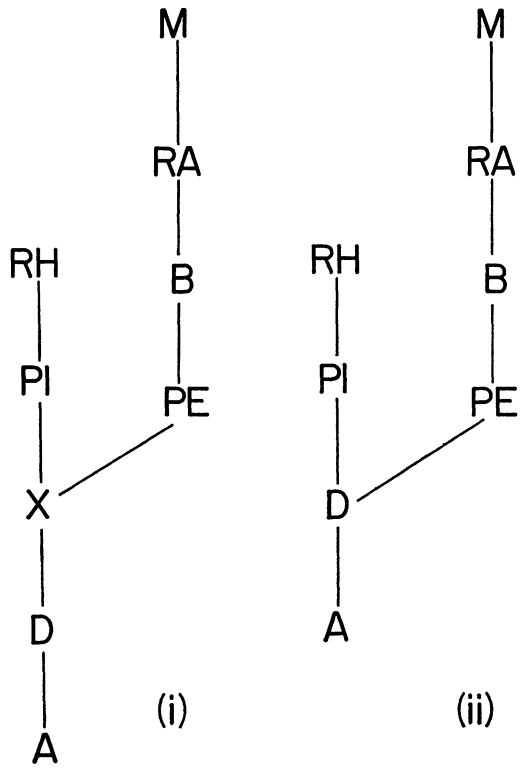

FIG. 3.—Evolutionary trees for families of anurans, produced by the Wagner Method (i), and from a Prim Network (ii). Legend: A: Ascaphidae, D: Discoglossidae, X: a hypothetical intermediate, PI: Pipidae, RH: Rhinophryhidae, PE: Pelobatidae, B: Bufonoid complex, RA: Ranoid complex, M: Microhylidae. (cf. Kluge and Farris, 1969).

A FORTRAN IV program to perform this algorithm is short enough to be profitably described here.

```
      DO 101 I = 2, IT
      LIN(I) = I
      JB(I) = 1
      DB(I) = 0.
      DO 101 J = 1, N
101   DB(I) = DB(I) + ABS(X(J, I) - X(J, 1))
      ITO = IT - 1
      DO 102 IP = 2, ITO
      D = DB(LIN(IP))
```

```
      IB = IP
      K = IP + 1
      DO 103 I = K, IT
      IF(DB(LIN(I)). GE. D) GOTO 103
      D = DB(LIN(I))
      IB = I
103   CONTINUE
      JEXT = LIN(IB)
      LIN(IB) = LIN(IP)
      LIN(IP) = JEXT
      DO 104 I = K, IT
      D = 0.
      L = LIN(I)
      DO 105 J = 1, N
105   D = D + ABS(X(J, L) - X(J, JEXT))
      IF(D. GE. DB(L)) GOTO 104
      DB(L) = D
      JB(L) = JEXT
104   CONTINUE
102   CONTINUE
```

The program operates as follows. N is the number of characters; IT is the number of OTUs; and X(J, I) is the data matrix. The *first* subscript of X indexes characters, the second, OTUs. OTU 1 is chosen as the starting point and the list, LIN, is loaded with the numbers of the unplaced OTUs (all the OTUs except 1). Each unplaced OTU is assigned a difference, DB(I), from the network, and DB(I) is initialized to the difference between OTU I and OTU 1. Each OTU, I, is assigned a closest OTU, JB(I), on the network. JB(I) is initialized to 1 for each OTU other than 1. These initializations are all performed by the 101 loop.

In the 103 loop the unplaced OTU that is closest to the network (has the smallest DB value) is located. The list LIN is then updated so that the numbers of the unplaced OTUs are stored in positions K, . . . , IT of LIN. The OTU selected in loop 103 is called JEXT, and the number of JEXT is saved in position IP of LIN, JEXT being the IPth OTU added to the network. The reason for saving the number of JEXT in LIN is that it is convenient to have the connection function printed out in the order in which the OTUs are placed on the network.

In the 104 loop the difference between each unplaced OTU, L, and JEXT is computed as D. If D exceeds DB(L) no action is taken. If D is smaller than DB(L), DB (L) and JB(L) are updated.

At each stage of the computation, for any unplaced OTU, I, DB(I) is the difference between I and the network, while JB(I) is the number of the network node closest to I. At the end of execution, JB(I) is the connection function of the Prim Network, and DB(I) is the length of the internode between I and JB(I).

The algorithm presented here has some advantages over other ways of computing Prim Networks. The program above computes the difference between OTUs I and J, $I < J$, just once and either saves the value in DB or discards it. An OTU-by-OTU matrix of differences is not stored, so that Prim Networks for quite large sets of data can be found on relatively small computers. Further, inspection of the program will reveal that it produces a network for IT OTUs by performing (IT-3) (IT-2) comparisons of differences. We may contrast this amount of work with that required by a Prim Network forming procedure described by Edwards and Cavalli-Sforza (1963). "[The Prim Network] can be found by listing all the distances between points in increasing order, and successively allocating segments to these distances, omitting any segment which completes a loop." The minimum number of comparisons needed to order the distances between IT OTUs is set by information theory at $(\frac{1}{2})(IT)(IT\text{-}1)(\log_2((\frac{1}{2})(IT)(IT\text{-}1)))$. The program presented here can thus be made more computationally efficient than other programs for computing Prim Networks, because it forms the networks by doing less work.

A working version of the FORTRAN IV Prim Network Program is available from the author, as is an IBM 360/67 Assembler Language routine using a similar algorithm.

THE WAGNER METHOD

The "Wagner Method" presented here is a computerized version of the original Wagner procedure for producing trees (Wagner, 1961, and references therein). This method

has been employed by Kluge and Farris (1969).

In the Wagner Method, the tree is formed by adding OTUs one at a time to a tree that initially consists of a single node—the ancestor. The ancestor may be "hypothetical" in that it is not an existing OTU. The "hypothetical" ancestor is, however, treated as an OTU rather than an HTU in that the character states of the ancestor are fixed, not computed by the algorithm.

The order in which OTUs are added to the tree is determined by the rank order of the *advancement index*. For OTU I, the advancement index is defined to be d(I, A), where A is the ancestor. OTUs with small advancement indices are added to the tree first. At each stage, the placement of the next OTU to be added is determined through the interval distance formula, and a new HTU to connect an OTU to the network is formed using the median-state property.

The algorithm of the Wagner Method is then:

1) Select an ancestor, A. Go to 2.

2) Compute the advancement index, AD (I) = d(I, A) for each OTU, I. Go to 3.

3) Find the OTU with smallest advancement index. Connect it to the ancestor to form a tree with one linkage (one interval). Go to 4.

4) Find the unplaced OTU, B, with smallest advancement index. Go to 5.

5) Find the interval (linkage), INT(C, f(C)), for C a node of the tree such that d(B, INT(C, f(C))) is minimal. Go to 6.

6) Construct an HTU, Y, as the median of B, C, and f(C). Go to 7.

7) Update the ancestor function:

$$f(Y) = f(C)$$
$$f(B) = Y$$
$$f(C) = Y.$$

Go to 8.

8) If any OTUs remain unplaced, go to 4. Otherwise stop.

The simple Wagner algorithm can be modified by changing the criterion for the order in which the OTUs are added to the tree. I have experimented with programs in which the addition sequence is given by

1) scanning the tree by the interval-distance formula at each step to determine which unplaced OTU is closest to the tree (as opposed to being closest to the ancestor),

2) adding OTUs to a Wagner Tree with ancestor A in the same order as they would be added to a Prim Network with base element A, and

3) adding OTUs with large advancement indices first.

None of these modifications has shown particular superiority to the original algorithm in forming ordinary Wagner Trees. Methods (2) and (3) have proved to be superior to the original algorithm in applications where Wagner programs are used in successive weighting procedures (see Farris, 1969).

The ancestor in the simple Wagner algorithm can be used to impart a direction to the tree, but it need not be interpreted in this way. If a real OTU is used as the "ancestor" in the Wagner algorithm, the output can be used as a Wagner Network.

ROOTLESS WAGNER METHODS

The simple Wagner algorithm does not impose irreversibility on the trees it produces, and from this standpoint, the choice of an ancestor may not be crucial. It has been found, however, that the form of the tree produced by a simple Wagner algorithm can be changed by altering the ancestor. The Rootless Wagner algorithms have been developed as an approach to reducing the dependency of the tree on the inferred ancestor.

A Rootless Wagner algorithm is produced from the simple algorithm described above by modifying the first 4 steps:

1) Select a pair of OTUs, A and D such that $d(A, D) \geqslant d(I, J)$ for any OTUs, I and J, in the study. Link them to form an interval. Go to 2.

2) Compute the "advancement index" of each OTU, I, as d(I, INT(A, D)). Go to 3.

3) Find the OTU, $C \neq A, D$, with the

largest "advancement index." Form an HTU, Y, as the median of A, B, and C. Construct an ancestor function:

$$f(A) = Y$$
$$f(D) = Y$$
$$f(C) = Y.$$

Go to 4.

4) Find the unplaced OTU, B, with the largest "advancement index." Go to 5.

As with rooted Wagner procedures, a number of Rootless Wagner algorithms can be generated by changing the criterion for the addition sequence of OTUs. Several addition criteria have been tried, but none seem to be appreciably more effective than the one given above. The criterion described can operate effectively in successive weighting applications.

The Rootless Wagner algorithm has the advantage that it can produce a Wagner Network with no reference to an ancestor at all. The algorithm can be used, however, to produce a directed Wagner Tree, simply by including a postulated ancestor among the OTUs to be analyzed. Hypothetical ancestor "OTUs" are given no special treatment by a Rootless Wagner procedure. It is still possible for the form of the tree to be influenced by the inclusion of an ancestor.

OPTIMIZATION OF TREES

A drawback of the algorithms described above is that the HTUs produced during the stepwise addition of OTUs to the tree may not be the optimal ones for the complete tree. The algorithms in which intervals are created by choosing the most disparate available OTUs are particularly subject to this difficulty. Suppose, for example, that the initial pair, A and D, of OTUs selected by the Rootless Wagner procedure have quite similar states, $X(A, i)$ and $X(D, i)$ for the ith character; i.e., $|X(A, i) - X(D, i)|$ is a relatively small number. There cannot, of course, be many such characters, since A and D were selected for the large value of $d(A, D)$ (cf. Eq. (1)). The presence of a few such characters is nonetheless possible. Now any HTU, Y, generated by the existing Rootless Wagner algorithms as a connector of some OTU to $INT(A, D)$ will have for character i a state, $X(Y, i)$, that lies numerically between $X(A, i)$ and $X(D, i)$, inclusive. It is quite probable that several OTUs, C, E, and F, say, will be independently connected to INT (A, D), say through HTUs, R, Y, Z, respectively. Then $X(R, i)$, $X(Y, i)$ and $X(Z, i)$ will all lie numerically between $X(A, i)$ and $X(D, i)$. Now suppose that C, E, and F are described by a state, $x = X(C, i)$, that does not lie between $X(A, i)$ and $X(D, i)$. Then the output of the Rootless Wagner procedure will effectively assert that state x is convergently present in OTUs C, E, and F, while the similarity of $X(A, i)$ and $X(D, i)$ is homologous. A more parsimonious interpretation is that state x is homologously present in C, E, and F, while the similarity of $X(A, i)$ and $X(D, i)$ is convergent. The latter interpretation can be imposed without changing the form of the Wagner Network. It is necessary only to alter the character state values of HTUs R, Y, Z.

A tree with a fixed branching form, that is, a fixed set of cladistic relationships, can be optimized for the parsimony criterion by computing for it an optimizing set of HTUs. I shall describe a procedure for doing so below. In the description, I shall utilize the concept of the *state set*, $S(Y, i)$ of an HTU, Y, for a character, i. The state set is a closed interval describing a range of character state values applicable to Y and i. The term *cladistic difference* is used in the sense of Farris (1967). In particular, two nodes are said to have *cladistic difference unity* if they share an immediate common ancestor.

The procedure is most conveniently described in sections. The first is the clustering procedure, as follows. Consider a set K of clusters to have initially as its elements just all the OTUs in the study, each OTU being considered as a "cluster" of one OTU. Then:

1. Select any two clusters, A and B, that have cladistic difference unity according to the tree being optimized.

2. Remove A and B from K.

3. Place Y = f(A) = f(B) in K.

4. For every character, i, compute S (Y, i) as described below.

5. If K has more than one element, return to (1).

The rules for computing new state sets (step (4)) are:

If clusters A and B in K are united to form Y = f(A) = f(B) then S(Y, i) is

R-1: the intersection of S(A, i) and S(B, i), *provided* that intersection is not empty; otherwise, S(Y, i) is

R-2: the smallest closed interval of one of the forms $[a_i, b_i]$ or $[b_i, a_i]$, where a_i is an element of S(A, i), and b_i is an element of S(B, i).

The state sets, S(Z, i), of HTUs, Z, will generally not be singleton, so that the character states, X(Z, i) of Z, will not generally be unique. The ambiguity of the states assigned to HTUs is reduced to a minimum by

R-3: if an HTU, F, with non-singleton state set, S(F, i), has f(F) = H, replace S(F, i) with the intersection of S(F, i) and S(H, i).

R-3 is most easily applied through a second pass through the tree after the clustering cycle that computes state sets of HTUs has been completed.

An example of the optimizing process may prove helpful. We use the hypothetical data:

	Character				
OTU	1	2	3	4	5
A	2	1	0	0	1
B	1	2	0	0	0
C	0	0	1	2	0
D	0	0	2	1	0
E	0	0	0	0	1
F	0	0	0	0	0

We shall optimize the tree of Fig. 4(i) for these data. For the example, I shall indicate state sets by expressions of the form [x, y], where x and y are the bounds of the range of state values in a set S(A, i).

The initial cluster set, K, contains A, B, C, D, and E. Clustering according to Fig. 4, we first unite A and B to form Y. By R-2

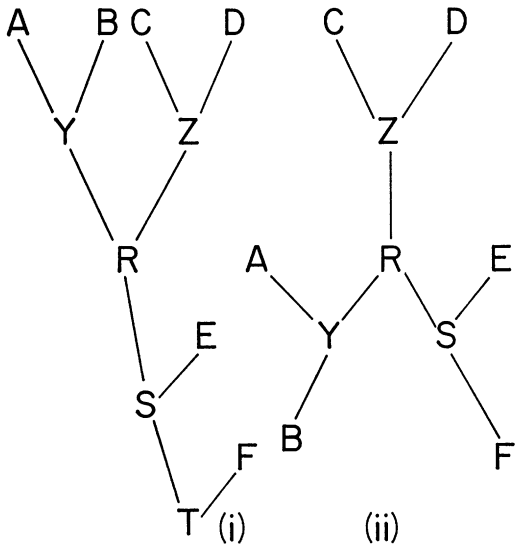

FIG. 4.—A tree (i) and a network (ii) to be optimized.

we assign to the state sets of Y the ranges of variation between A and B. Then Y has the state sets [1, 2], [1, 2], [0, 0], [0, 0], and [0, 1] for the five characters respectively. Similarly, Z has state sets [0, 0], [0, 0], [1, 2], [1, 2], and [0, 0]. For the first four characters, the state sets of Y and Z have empty intersections, so that the corresponding state sets of R are computed according to R-2. For character five, the state sets of Y and Z have a non-empty intersection, so that R-1 applies. R then has state sets [0, 1], [0, 1], [0, 1], [0, 1], [0, 0]. Similarly, the first four state sets of S can be computed as the intersections of the state sets of R and E, while the last state sets of S is the range between the "1" state of E and the "0" state of R. S has state sets [0, 0], [0, 0], [0, 0], [0, 0], and [0, 1]. The intersection of the state sets of S and F is non-empty for all characters. Thus, T has state sets [0, 0], [0, 0], [0, 0], [0, 0], and [0, 0].

Making the second pass through the tree, we replace the state sets of Y with their intersections with the state sets of R, obtaining new state sets [1, 1], [1, 1], [0, 0], [0, 0], and [0, 0] for Y, and proceed similarly for

the other HTUs. The character states of the HTUs are in the case uniquely determined and are

	Character				
HTU	1	2	3	4	5
Y	1	1	0	0	0
Z	0	0	1	1	0
R	0	0	0	0	0
S	0	0	0	0	0
T	0	0	0	0	0

Note that the optimized lengths of INT (R, S), INT(S, T), and INT(T, F) are zero.

The same procedure as described above can be used also to optimize the HTUs of approximated Wagner Networks simply by imposing an arbitrary direction on the connection function of the network. The tree example just performed is equivalent to optimizing the network of Fig. 4(ii).

DISCUSSION

The HTU optimizing procedure can be used to increase the parsimony of trees and networks generated by Wagner programs. It can also be used, however, to assign an optimal set of HTUs to the dendrogram generated by any sort of numerical taxonomic clustering procedure.

Since any dendrogram can now be assigned a parsimony-optimal set of HTUs, any dendrogram can also be assigned a length, and, consequently, a measure of parsimony. It is therefore practical to use parsimony as an optimality criterion on any sort of dendrogram, including strictly phenetic ones. The phenetic desirability of doing so is, of course, open to doubt and will have to be extensively investigated. It does seem possible, however, that parsimony will provide a general optimality criterion that lacks some of the less appealing qualities of the cophenetic correlation coefficient, now widely used as an optimality measure on phenograms (see Farris, 1969a).

The possibility of calculating an optimal set of HTUs for a tree of any branching form also allows us to compare the relative parsimony of intuitively derived and numerically derived hypotheses on phylogeny. The relative degree of the likelihood of alternative phylogenetic theories can thus be assessed, as can the relative efficiency of intuitive and numerical techniques.

Finally, since optimal HTUs can be calculated after a tree has been formed, we are freed of the necessity to compute trees by the sort of stepwise procedure used in the Wagner programs. We may be able to reduce the amount of computer time necessary to obtain taxonomic conclusions by using conventional clustering methods to obtain the form of the tree. HTUs could then be computed separately. The type of clustering scheme used will need to be carefully selected, however. Most phenetic clustering methods generate branching forms with an unacceptably low degree of parsimony. I am currently investigating clustering criteria to allow clustering programs to generate branching forms optimizable to acceptably small length.

REFERENCES

CAMIN, J. H., AND R. R. SOKAL. 1965. A method for deducing branching sequences in phylogeny. Evolution, 19:311–327.

EDWARDS, A. W. F., AND L. L. CAVALLI-SFORZA. 1963. Reconstruction of evolutionary trees. *In* Phenetic and phylogenetic classification. Systematics Assoc. Publ., 6:67–76.

ESTABROOK, G. F. 1968. A general solution in partial orders for the Camin-Sokal model in phylogeny. J. Theoret. Biol., 21:421–438.

FARRIS, J. S. 1969. A successive approximations approach to character weighting. Syst. Zool., 18:374–385.

FARRIS, J. S. 1969a. On the cophenetic correlation coefficient. Syst. Zool., 18:279–285.

KLUGE, A. G., AND J. S. FARRIS. 1969. Quantitative phyletics and the evolution of anurans. Syst. Zool., 18:1–32.

PRIM, R. C. 1957. Shortest connection networks and some generalizations. Bell Syst. Tech. J., 36:1389–1401.

WAGNER, W. H. 1961. Problems in the classification of ferns, p. 841–844. *In* Recent advances in botany. Univ. Toronto Press, Toronto.

Biological Sciences, State University of New York, Stony Brook, New York 11790.

Contribution No. 12 from the program in Ecology and Evolution, State University of New York at Stony Brook, Stony Brook, New York 11790.