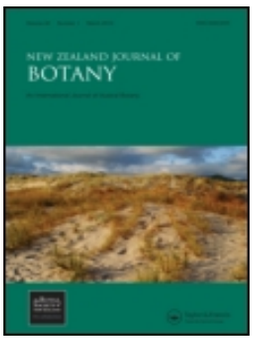

New Zealand Journal of Botany

ISSN: 0028-825X (Print) 1175-8643 (Online) Journal homepage: http://www.tandfonline.com/loi/tnzb20

Constructing phylogenetic trees efficiently using compatibility criteria

Tandy J. Warnow

To cite this article: Tandy J. Warnow (1993) Constructing phylogenetic trees efficiently using compatibility criteria, New Zealand Journal of Botany, 31:3, 239-247, DOI: 10.1080/0028825X.1993.10419501

To link to this article: http://dx.doi.org/10.1080/0028825X.1993.10419501

New Zealand Journal of Botany, 1993, Vol. 31: 239–248
0028–825X/93/3103–0239 $2.50/0

Constructing phylogenetic trees efficiently using compatibility criteria

TANDY J. WARNOW

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185, U.S.A.*

*Present address: Department of Computer and Information Sciences, University of Pennsylvania, 200 S 33rd Street, Moore 276, Philadelphia, PA 19104, U.S.A.

Abstract The Character Compatibility Problem is a classical problem in computational biology concerned with constructing phylogenetic trees of minimum possible evolution from qualitative character sets. This problem arose in the 1970s, and until recently the only cases for which efficient algorithms were found were for binary (i.e. two-state) characters and for two characters at a time, while the complexity of the general problem remained open. In this paper we will discuss the remarkable progress on this problem since 1990.

Keywords evolution; algorithms; phylogenetic tree;

INTRODUCTION

The Character Compatibility Problem is a classical problem in computational biology, which seeks to construct a phylogenetic tree of minimum possible evolution for an input species set, described by qualitative characters. Such a phylogeny is said to be perfect, and thus the problem of whether such a phylogeny exists is called both the Perfect Phylogeny Problem and the Character Compatibility Problem.

We begin with the section "Basic concepts from computer science". In this section we will define what we mean by the complexity of an algorithm or problem, so that the meaning of "efficient" is clear, and the consequences of saying that a problem is NP-hard is also understood. We continue by defining the Character Compatibilty Problem in the section "The Character Compatibility Problem", and describe the history of the problem in the section "History of the Character Compatibility Problem". We discuss a related graph-theoretic problem, which has formed the basis for many of the results on the Character Compatibility Problem, in the section "Triangulating coloured graphs". We then present various algorithms for the special cases of the Character Compatibility Problem in the section "Efficient algorithms". These algorithms will determine if a perfect phylogeny exists, and construct it if it does. A discussion of a new software package available for phylogenetic tree construction using one of the algorithms from this section is given in the section "Phylogeny construction from DNA sequences". We conclude in our final section, "Discussion and open problems", with some open problems and a discussion of the use of these algorithms in the design of heuristics for phylogenetic tree construction.

Basic concepts from computer science

In this section we will describe some basic concepts regarding the complexity of problems and algorithms, which are used many times in this paper. Readers who wish to obtain more than this cursory discussion of these concepts are directed to read Garey & Johnson's (1979) classic, "Computers and Intractability".

Many computational problems can be described as decision problems, where a yes-no question is asked. An algorithm to solve the problem must always return the correct answer. The measure of efficiency for the algorithm is defined by how long the algorithm takes for its worst case. This in turn is computed by counting the number of steps the algorithm uses on its input. Generally, each single type of computation has unit cost; thus, the number of comparisons, additions, subtractions, I/O (input-output) activities, and so on, are counted, and this

B93007
Received 19 January 1993; accepted 8 June 1993

total is the cost of the computation for the algorithm for its input.

An example may help make this clear. Suppose the problem is to determine whether a list of social security numbers includes a specific number. The input to the problem has therefore two parts: the list $\mathcal{L}$, and the number X that is being searched for. An algorithm for the problem might do the following: starting at the beginning of the list $\mathcal{L}$, compare each number in the list to X. Stop when a match is found or the end of the list is reached. The cost of the algorithm is computed as follows: each comparison of two social security numbers will cost us only one step (we could say it costs us nine steps if we like, since social security numbers are nine digits long, but this will not make an important difference to our final "cost" as we will see). The best case for the algorithm is that the very first number in the list is the number X we are searching for. The worst case occurs if the list does not contain the number X. In this case, we may have to compare X to every number in our list. If the list contains n social security numbers, then this algorithm will have to make n comparisons. Thus, the algorithm will have a worst case of cost n, or $9n$, depending on how we count the cost of comparing two social security numbers. We ignore the constant 9, and simply say that this algorithm has running time $O(n)$. Intuitively, this notation means that the algorithm's cost is asymptotically no worse than some constant times n, where n is the length of the input. The length of the input is the amount of space needed to efficiently represent the input.

An algorithm whose worst case running time is bounded by a polynomial in the length of the input is called either a polynomial-time algorithm, or simply said to be efficient. Some problems (such as the one above) can be solved efficiently. Some problems seemingly can not. That is, there are problems, such as the Travelling Salesman Problem, for which no efficient algorithms have ever been found. These problems belong to a class of problems which are called NP-Complete, which are each of them equally hard to solve efficiently. Thus, if an efficient (i.e., polynomial time) algorithm were found for any one of these problems, then all of them could be solved efficiently. It is generally believed in the computer science community that this is impossible, so that none of these problems will ever be solved efficiently (i.e., in polynomial time). The first question that a computer scientist asks when looking at a computational problem is whether it can be solved efficiently. Thus, the computer scientist either wishes to find an efficient (i.e., polynomial time algorithm) for the problem, or determine that the problem is NP-Complete.

An example of an NP-Complete problem is the famous Travelling Salesman Problem (see Garey & Johnson 1979). Here, the input to the problem is a map with intercity distances, and the objective is to find a minimum length tour for a salesman, which will permit him to visit each city exactly once, and return to his starting point. This problem is stated as a minimisation problem, but can be also stated as a decision problem in the following way. We can assume that a positive real number C can also be part of the input, with the yes-no question then being "Does there exist a tour of length at most C which visits every city exactly once and returns to the start position?". It is easy to see that if the yes-no question can be answered efficiently, then the minimisation problem can also be answered efficiently, by repeatedly asking (but only a polynomial number of times) for different choices of C, whether a tour of length at most C exists. When the minimum such C is found, then the minimisation problem is solved. To date, there are no efficient algorithms for solving this problem, and while a multitude of heuristics exist which may find pretty good solutions (and occasionally find the best solution), there are no algorithms that are both efficient (in that they always terminate in a polynomial number of steps) and always find the best solution.

Later in the paper we will state that the problem of finding optimal phylogenetic trees using compatibility criteria is in its general case NP-Complete. As a consequence, this will mean that efficient algorithms to solve the problem probably can not exist. On the other hand, by restricting the kinds of input to the problem, we may be able to solve it efficiently. Thus, the paper will show when the problem can be solved efficiently and when it probably can not.

The Character Compatibility Problem

Given a set S of n species, a phylogenetic tree or, more simply, a phylogeny or evolutionary tree, is a rooted tree with n leaves that describes the evolution of the species in S from a common ancestor. The internal nodes represent ancestral species, and the leaves represent the species in S.

One standard way to describe species is through the use of qualitative characters. A qualitative character (or character) is an equivalence relation on the set S of species, partitioning the species set into the distinct equivalence classes (which we will

call the character states). Since the number of possible states of any character is finite over any finite species set, we can number the states of each character. If there are n species and k characters, we can represent our data by a $n \times k$ integer matrix M, where the entry M_{ij} is the state of species s_i for the j^{th} character. The i^{th} row in this matrix is a vector over Z^k (where Z denotes the integers), which is identified with the species s_i.

In this model, in a phylogeny T for the species set S we replace the name of the species at the leaves by the vector for that species, and we also assign to each internal node of the tree T a vector in Z^k. Thus, we are hypothesising the character states of the ancestral species in our phylogeny. The desired property for this phylogeny is that, for each state of each character, the set of nodes in the phylogeny having that state should form a subtree. That is, the subgraph induced by those nodes should be a connected graph. Such a phylogeny is called a perfect phylogeny. This is also expressed by saying the phylogeny has the convexity property.

When a perfect phylogeny exists for a set C of characters defining a species set S, we say the character set is perfectly compatible or, simply, compatible, on the species set S. In the literature it is common to drop the reference to the species set S, and simply say the character set C is compatible. When there is no possibility of confusion about the species set, we shall use this convention.

We can now state the following:

Perfect Phylogeny Problem: Given a set of species defined by a set of characters, determine whether a perfect phylogeny exists.

This problem is also referred to in the literature as the Character Compatibility Problem. The convexity property is independent of rooting for the tree, and there is no information about distances in the tree, whether measured in time or in evolutionary change.

History of the Character Compatibility Problem

The character compatibility approach to phylogenetic tree construction is related to parsimony and compatibility criteria. These criteria give ways of measuring the fit of a phylogenetic tree to the data. The input to the problem is a set S of species, where each species in S is associated with a vector in Z^k (as described in the previous section). The output to the problem is a tree T in which each species in S is assigned to a leaf in T, and the internal nodes are also labelled by vectors in Z^k. The fit of this tree T to the data is then determined by some measure. There are various measures which are used, but the two most popular are called parsimony and compatibility criteria. The cost of the tree T under parsimony criteria is the number of character state changes indicated by the edges of the tree. That is, the endpoints of each edge are labelled by vectors in Z^k. The number of positions for which these two vectors disagree is the cost of the edge. If that cost is summed over all the edges in T, we obtain a parsimony score. The objective, using parsimony, is to find a tree of minimum cost. Certain variations of parsimony prohibit various types of changes, while others might permit those but forbid others. Compatibility criteria seek instead to keep as many characters as possible, while eliminating others, so that with respect to the characters that are kept, it is possible to construct a perfect phylogeny. Each approach has its proponents, but essentially they agree in their philosophy, which is that the tree of minimum evolution is the right tree.

This approach to phylogenetic tree construction was evident from the 1960s on (Edwards & Cavalli-Sforza (1963, 1964) are usually credited with introducing parsimony in their papers); later, many jumped in to contribute their variations on this theme (Kluge & Farris 1969; LeQuesne 1969, 1972; Camin & Sokal 1965, and others).

The concept of characters being perfectly compatible with a given tree arose naturally from these approaches, in that it represents the best possible tree (i.e., having minimum possible evolution on the tree). This objective was championed by LeQuesne (1969, 1972, 1974, 1977), and later given its firm mathematical foundation in the literature by a series of papers (Estabrook 1972; Estabrook et al. 1975).

Since construction of phylogenies using compatibility criteria was shown by Day & Sankoff (1986) to be an NP-hard problem, and many variations of parsimony were also shown to result in NP-hard problems (Foulds & Graham 1982; Day 1983; Gusfield 1984; Day et al. 1986), the Character Compatibility Problem was assumed to be NP-Complete as well. This was finally proved independently in 1991 by Steel (1992) and Bodlaender et al. (1992). These results thus imply that efficient algorithms to solve these problems probably can never be found.

Until 1990, the only progress on algorithms for the Character Compatibility Problem was to show that binary character compatibility (McMorris 1977) and compatibility of two characters at a time (Estabrook & Landrum 1975; Fitch 1975) could be

solved in polynomial time. An important theoretical breakthrough came in 1974, when Buneman (1974) proved a beautiful result showing that the problem reduced in polynomial time to the graph-theoretic Triangulating Coloured Graphs Problem. These problems were shown polynomially equivalent by Kannan & Warnow (1992) (i.e., a polynomial time algorithm for one problem will yield a polynomial time algorithm for the other as well). This equivalence was used by Bodlaender et al. (1992) to show that the Character Compatibility Problem was NP-Complete, so that no polynomial time algorithm for the Character Compatibility Problem will exist unless $P = NP$. These results are described in the section "Triangulating Coloured Graphs".

These results imply that it is unlikely that any polynomial time algorithm can exist that will handle all cases of the Character Compatibility Problem. The best that can be hoped for is to find special cases for which polynomial time algorithms can be found. Much of the work in the last few years has been to show that polynomial time solutions do exist for many cases of the problem. These results are described in the section "Efficient algorithms".

For further information about this and other models for computing phylogenetic trees, please see Felsenstein's (1982) review paper.

The Triangulating Coloured Graphs Problem

We begin by defining some basic graph-theoretic terminology. A graph $G = (V,E)$ consists of a vertex set V and an edge set E, where each edge is an unordered pair of distinct vertices. A map $f: V \rightarrow Z$ is said to be a vertex-colouring and if the vertex-colouring satisfies $f(v) \neq f(w)$ for all vertices v,w such that $(v,w) \in E$, then the vertex-colouring is said to be proper. A path in the graph is a sequence of vertices $v_1,v_2,\ldots,v_r$ where $(v_i,v_{i+1}) \in E$ for $i = 1,2,\ldots,r-1$, and a cycle is a path with $v_1 = v_r$ (i.e., it begins and ends in the same place). A graph is said to be acyclic if it has no cycles. A chord in a cycle is an edge that joins vertices that are not consecutive in the ordering given by the cycle. A graph G is said to be chordal if every cycle of size at least four contains a chord. Chordal graphs are also called triangulated graphs.

The Triangulating Coloured Graphs Problem is as follows:

Triangulating Coloured Graphs Problem:
Input: A graph $G = (V,E)$, and a vertex-colouring $c: V \rightarrow Z$.
Question: Does there exist a graph $G' = (V,E')$ with $E \subset E'$ such that G' is chordal, and G' is properly coloured by c?

In other words, is it possible to add edges between vertices in G, never adding an edge between vertices that have the same colour, so that the resultant graph is triangulated? If this is possible, the resultant graph is called a c-triangulation of G, and G is said to be c-triangulated.

In 1974, Buneman proved the following theorem:
Theorem 1: (Buneman 1974): An instance I to the Character Compatibility Problem has a perfect phylogeny if and only if the derived partition intersection graph G_I can be c-triangulated.

To understand the theorem, we need to define the partition intersection graph G_I. Let I be an instance to the Character Compatibility Problem (i.e., species defined by characters). Create a graph G_I as follows: each state of each character is made into a vertex. The colours of two vertices are identical if and only if they arise from the same character. Two vertices are made adjacent if and only if their corresponding character states have at least one species in common. The graph that is created in this transformation is called a partition intersection graph, and it is easy to see that the number of characters in the Character Compatibility Problem instance is equal to the number of colours in the derived Triangulating Coloured Graphs Problem. Also, since the characters define a partition on the species set, there is never an edge between vertices of the same colour; thus, the graph is properly coloured.

As a corollary to Buneman's theorem, we have the following:
Corollary 1: If the Triangulating Coloured Graphs Problem can be solved in polynomial time, then so can the Character Compatibility Problem.

The converse was then proved by Kannan & Warnow (1992):
Theorem 2: The Character Compatibility and the Triangulating Coloured Graphs Problems are polynomially equivalent, so that a polynomial time algorithm for one will yield a polynomial time algorithm for the other.

Since both problems were shown NP-Complete in Steel (1992) and Bodlaender et al. (1992), the best that can be hoped for are polynomial time algorithms for special cases.

EFFICIENT ALGORITHMS

In this section we consider the problem of determining whether a set of characters is perfectly compatible, and if it is, to construct a perfect phylogeny for the dataset. As we mentioned before, the property of being perfectly compatible is

independent of where the tree is rooted, and also provides no information about the distances on the edges. The output from any algorithm for the Character Compatibility Problem will therefore be an unrooted tree without indications of edge lengths. To root the tree, one would include in the dataset a sequence from a distantly related species, so that the obvious way to group the tree is on the edge between the added species and the remainder of the tree.

We are interested in deterministic polynomial time algorithms for this problem. By deterministic we mean algorithms whose operation is determined solely by the input, and do not include any randomisation procedure. These algorithms must always produce the correct answer on every input, and are therefore to be distinguished from heuristics, which have no such guarantee. We also require that the running time of the algorithm in its worst case input should grow no faster than a polynomial in the input size. This polynomial depends upon the algorithm, but not upon the input.

There are three natural parameters to the Character Compatibility Problem: n, the number of species (or sequences); k, the number of characters (or length of the sequences); and r, the maximum number of states per character. For this problem, we are interested in algorithms that have running times bounded by polynomials in n, r, and k. If we say that an algorithm is $O(p(n, r, k))$, we mean that there is a constant C (which we do not specify) such that the algorithm will complete its determination of the answer for any input I of n species, k characters, each having r states, in at most $Cp(n, r, k)$ steps.

In the last few years, there have been several breakthroughs, so that now it is known that if any one of these parameters is bounded, then the Character Compatibility Problem can be solved in polynomial time. It is not hard to see that if we bound the number of species by a constant c, then the number of binary phylogenetic trees on c species is also only a constant (equal to $1 \times 3 \times 5 \times \ldots \times(2c-5)$, and each candidate tree can be checked to see if the characters are perfectly compatible on it using Fitch's polynomial time algorithm (Fitch 1971). The other results are more surprising. We will describe algorithms for each of these cases in the sections below.

Algorithms when the number of characters is fixed

The algorithms that handle the case where the number of characters is fixed directly use Buneman's theorem. These algorithms first translate the input I (given as species defined by characters) into the partition intersection graph G_I, and then use graph-theory and Buneman's theorem to determine whether I admits a perfect phylogeny. These algorithms have running times that are only useful when k, the number of characters, is small, since the algorithms have complexity that grow exponentially in k. Still, the use of Buneman's theorem is extremely powerful, and makes seemingly complicated algorithms and theorems immediately accessible.

The case where the input has two characters is a good example of the use of Buneman's theorem. Since the data involve two characters, the partition intersection graph that we derive therefore has two colours, and it turns out to be easy to determine whether a two-coloured graph can be c-triangulated. Consider the following example, in which we have four species defined by two binary characters. The species are given as 2-tuples, so that the entry in the first position is the state under the first character α, and the entry in the second position is the state under the second character, β. Our input is the set of species A, B, C, D, given as follows: $A = (0,0)$, $B = (0,1)$, $C = (1,0)$, $D = (1,1)$.

To create the partition intersection graph G from this input, we have four vertices, one for each character state: $\alpha_0 = \{A, B\}$, $\alpha_1 = \{C, D\}$, $\beta_0 = \{A, C\}$, and $\beta_1 = \{B, D\}$. Thus, α_0 is the set of species having state 0 for α, and α_1, β_0, and β_1 are defined similarly. The vertices α_0 and α_1 are both coloured red, while the β_0 and β_1 vertices are coloured blue. The edges are added when the sets have non-empty intersection; thus, we add the edge (α_0,β_0) since both states include A, and similarly we add (α_0,β_1), (α_1,β_0), and (α_1,β_1). Thus, the graph G is a two-coloured four-cycle. Applying Buneman's theorem we determine that I has a perfect phylogeny if and only if G can be c-triangulated. Since G is a chordless cycle, in order to triangulate it we must add at least one edge; however, the two possible candidates involve adding an edge between two vertices of the same colour! Since this is forbidden, G cannot be c-triangulated, and hence I has no perfect phylogeny.

The following can be proved easily by induction.

Lemma 1: Two characters are compatible on a set of species if and only if the partition intersection graph derived is acyclic.

A consequence of this lemma is the following:

Lemma 2: Compatibility of two characters on n species can be determined in $O(n)$ time.

Proof: Given the input, construct the partition intersection graph $G = (V,E)$. Acyclicity of G can

then be tested in $O(|V| + |E|)$ time. Since $|V| = O(n)$ and $|E| = O(n)$, we can determine acyclicity of G in $O(n)$ time.

In the biological literature, the first algorithms presented to determine compatibility of two characters were given by Estabrook & Landrum (1975) and Fitch (1975), and only proved correct two years later by Estabrook & McMorris (1977). By contrast, we have presented an extremely simple algorithm with a very short proof, by applying Buneman's theorem!

Algorithms for three characters at a time were discovered first by Kannan & Warnow (1992), then later by Bodlaender & Kloks (1993) and Idury & Schaeffer (1993). Each of these algorithms uses linear time, but Kannan & Warnow (1992) required quadratic space, while the others managed to find linear space implementations. Each of these is actually an algorithm to determine whether a given three-coloured graph can be c-triangulated, and thus also exploits the Buneman construction.

Although compatibility of three characters can be tested in linear time, the best running time for four characters at a time has complexity $O(r^5 + n)$. The only algorithm for the Character Compatibility Problem which has polynomial time when the number of characters is bounded by a constant is the algorithm of McMorris et al. (in press), which achieves $O(r^{k+1} k^{k+1} + nk^2)$. Since this grows exponentially in the number of characters, it is primarily a theoretically interesting result rather than a practical result.

Efficient algorithms for fixed numbers of states

Characters derived from molecular data (such as DNA sequences) are associated with the columns of a multiple alignment, and thus have the restriction that the number of states achieved by any particular character is limited to a small constant. In the case of DNA or RNA sequences, this number is four, while for amino-acid sequences, the number is 20. As we will see, by restricting the number of states, we are able to obtain efficient algorithms for constructing perfect phylogenies (if they exist, of course!).

The first case of this type that was handled was for binary characters. Polynomial time algorithms for binary character compatibility were found by several authors with the best being the $O(nk)$ algorithm of Gusfield (1991), for testing compatibility of k characters on n species. This is optimal, since the input requires $O(nk)$ space, and each entry must be checked.

Algorithms for determining compatibility of three-state characters were found in 1990 by Dress & Steel (1992) and Kannan & Warnow (in press); these algorithms had complexity $O(nk^2)$ and $O(n^2 k)$, respectively, where n is the number of species and k the number of characters.

Kannan and Warnow then extended the techniques of their algorithm for three-state characters, and derived an $O(n^2 k)$ algorithm for four-state character compatibility (Kannan & Warnow in press). They conjectured that it is possible to determine compatibility of k r-state characters on n species in $O(f(r) n^2 k)$, where $f(r)$ is a function which depends only upon r and not upon n or k.

A recent result of Agarwala & Fernandez-Baca (in press) is an algorithm for character compatibility, which has running time $O(2^{3r}(nk^3 + k^4))$, so that the Character Compatibility Problem can be solved in polynomial time when the number of states per character is bounded. This result is impressive, but the algorithm is not as efficient as the Kannan and Warnow algorithm when $r = 4$, since $O(n^2 k)$ will generally be much better than $O(nk^3 + k^4)$, when data are derived from aligned DNA sequences.

PHYLOGENY CONSTRUCTION FROM DNA SEQUENCES

We discuss in this section a *C* (Kernighan & Ritchie 1988) program based upon the Kannan and Warnow algorithm (Kannan & Warnow in press) for constructing phylogenies from DNA sequences. This program has the distinction (amoungst the other software packages for phylogeny construction) in that it guarantees finding a perfect phylogeny, if one exists.

Assume a set S of species is given, and that each species is described by a DNA sequence. If the DNA sequences are all perfectly aligned (so that there are no gaps), then each position in the sequences represents a character. Thus we have the quaternary Character Compatibility Problem, since each position must take on one of only four possible values (A, C, T, or G). We can apply the algorithm to this case, and determine whether a perfect phylogeny exists for species defined by perfectly aligned DNA sequences. If we iteratively add characters to a set of compatible characters until no further characters can be added, then we can construct maximal sets of perfectly compatible

characters in $O(n^2 k^2)$ time. Thus, we can in $O(n^2 k^2)$ time construct a phylogeny for the species set S for which a maximal set of characters is perfectly compatible.

It is important to realise that "maximal" does not have the same meaning as maximum. By maximal we mean a set C' of characters, which can not be further enlarged by the addition of another character, without losing the ability to find a phylogeny that is perfect for these characters. Thus, if C' is a proper subset of C'', then no perfect phylogeny will exist withr respect to C''. By "maximum" we mean a set C^* of characters such that the number of characters in C^* is maximum—thus, any other set C^{**} that permits a perfect phylogeny has no more characters than C^* has. The difference between the problems of finding a maximal set of compatible characters and finding a maximum set is this: finding a maximum set of compatible characters is NP-Complete, but finding a maximal set can be done in polynomial time.

This $O(n^2 k^2)$ algorithm to construct phylogenetic trees compatible with a maximal set of characters was programmed in *C*, and was then applied to data studied by Brown et al. (1992), obtained from a set of seven perfectly aligned sequences of length 232. The data used by this program included only the nucleic acids in the third position of each codon from perfectly aligned mitochondrial DNA sequences of length 696. We ran our *C* program on this data once (i.e., we did not repeatedly search for larger sets of maximal characters, but found this on the first try) and obtained a tree on which 153 out of the 232 positions were perfectly compatible (Fig. 1). This tree has been arbitrarily rooted on the edge connecting the grouping mouse–bovine to the remainder of the species, gibbon, orang, human, chimp, and gorilla.

What is significant about this result is that the tree was found without any human direction, such as repeated searches, or through the use of any additional criterion.

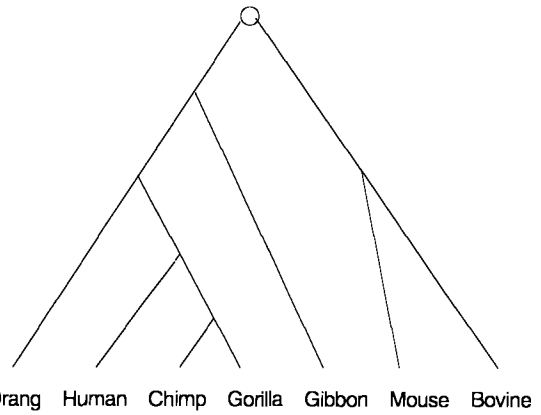

Fig. 1 Phylogenetic tree computed by Kannan-Warnow *C* Program on input of seven sequences of third positions from aligned mitochondrial DNA from Brown et al. (1992).

DISCUSSION AND OPEN PROBLEMS

Although it is now known that the Character Compatibility Problem is in *P* whenever either the number of states or the number of characters is fixed, the algorithms we have presented for these general cases are probably not optimal. Only the algorithms designed for special cases (compatibility of three characters, or compatibility of three or four state characters) are provably optimal, in that their running time is close to proven lower bounds. It would be interesting to see, for example, if we can define a function f and a polynomial p such that compatibility of k characters on n species can be tested in $O(f(k)\ p(n,k))$.

As stated before, the Character Compatibility Problem seeks to find a tree which is perfect, and therefore likely not to exist. Despite this, one of the tools that biologists frequently avail themselves of is to find large sets of characters which are pairwise compatible, in their search for optimal trees. Since pairwise compatibility does not ensure set-wise compatibility, the algorithms referred to in this paper could assist the biologists in the design of heuristics for tree constructions.

The main benefit of algorithms for the Character Compatibility Problem, besides their use in the development of heuristics for tree construction, is that they provably find the optimal tree when the data support a perfect phylogeny. Other approaches, such as the heuristics used in programs such as PHYLIP (Felsenstein 1993) and PAUP (Swofford 1993), have no performance guarantees. By a performance guarantee we mean either a promise of finding the optimal tree or of coming within some factor of the optimal. A common assumption seems to be that these heuristic programs will work well on "tree-like" data; this, however, remains an unproven assumption, and one worth investigating. It is easy to prove that a greedy heuristic, which adds species sequentially to the tree, will not necessarily find an optimal tree given tree-like data, and it can

probably also be shown that such a heuristic cannot even obtain a guaranteed error bound. More sophisticated heuristics may, however, be able to guarantee some error bound in their performance. Still, despite these limitations, PAUP and PHYLIP provide important tools to the practising biologist, who needs to find a reasonable phylogenetic tree now, instead of waiting until some point in the future when these more sophisticated tools are developed.

ACKNOWLEDGMENTS

The author was supported by NSF Grants CCR88–13632 and DMS9005833, and by the U.S. Department of Energy under Contract DE-AC04–76DP00789. Most of the author's work discussed in this paper was co-authored, and thanks are thus due to Sampath Kannan, Buck McMorris, and Tom Wimer. The referees for the paper (Mike Steel and an anonymous reviewer) are thanked for their thoughtful comments.

REFERENCES

Agarwala, R.; Fernandez-Baca, D. in press: A polynomial time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM journal on computing.*

Bodlaender, H.; Fellows, M.; Warnow, T. 1992: Two strikes against perfect phylogeny. Proceedings of the 19th International Congress on Automata Languages and Programming. Berlin, Springer-Verlag. *Lecture notes in computer science 623*: 273–283.

Bodlaender, H.; Kloks, T. 1993: A simple linear time algorithm for triangulating three-colored graphs. *Journal of algorithms 15*: 160–172.

Brown, W. M.; Prager, E. M.; Wang, A.; Wilson, A. C. 1992: Mitochondrial DNA sequences of primates: tempo and mode of evolution. *Journal of molecular evolution 18*: 225–239.

Buneman, P. 1974: A characterization of rigid circuit graphs. *Discrete mathematics 9*: 205–212.

Camin, J.; Sokal, R. 1965: A method for deducing branching sequences in phylogeny. *Evolution 19*: 311–326.

Day, W. H. E. 1983: Computationally difficult parsimony problems in phylogenetic systematics. *Journal of theoretical biology 103*: 429–438.

Day, W. H. E.; Sankoff, D. 1986: Computational complexity of inferring phylogenies by compatibility. *Systematic zoology 35 (2)*: 224–229.

Day, W. H. E.; Johnson, D. S.; Sankoff, D. 1986: The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical biosciences 81*: 33–42.

Dress, A.; Steel, M. A. 1992: Convex tree realizations of partitions. *Applied mathematics letters 5 (3)*: 3–6.

Edwards, A. W. F.; Cavalli–Sforza, L. L. 1963: The reconstruction of evolution. *Annals of human genetics 27*: 105.

Edwards, A. W. F.; Cavalli–Sforza, L. L. 1964: Reconstruction of evolutionary trees. *In*: Heywood, V. H.; McNeill, J. *ed.* Phenetic and phylogenetic classification. *Systematics Association publication 6*: 67–76.

Estabrook, G. F. 1972: Cladistic methodology: a discussion of the theoretical basis for the induction of evolutionary history. *Annual review of ecology and systematics 3*: 427–456.

Estabrook, G. F.; Landrum, L. 1975: A simple test for the possible simultaneous evolutionary divergence of two aminoacid positions. *Taxon 24*: 609–613.

Estabrook, G. F.; McMorris, F. R. 1977: When are two qualitative taxonomic characters compatible? *Journal of mathematical biology 4*: 195–200.

Estabrook, G. F.; Meacham, C. 1985: Compatibility methods in systematics. *Annual review of ecology and systematics 16*: 431–446.

Estabrook, G. F.; Johnson, C. S. Jun.; McMorris, F. R. 1975: An idealized concept of the true cladistic character. *Mathematical biosciences 23*: 263–272.

Felsenstein, J. 1982: Numerical methods for inferring evolutionary trees. *The quarterly review of biology 57 (4).*

Felsenstein, J. 1993: PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Seattle, Department of Genetics, University of Washington.

Fitch, W. M. 1971: Toward defining the course of evolution: minimum change for a specified tree topology. *Systematic zoology 20*: 406–416.

Fitch, W. M. 19.5: Toward finding the tree of maximum parsimony. *In*: Estabrook, G. F. *ed.* Proceedings of the Eighth International Conference on Numerical Taxonomy. San Francisco, W.H. Freeman. Pp. 189–230.

Foulds, L. R.; Graham, R. L. 1982: The steiner problem in phylogeny is NP-Complete. *Advances in applied mathematics 3*: 43–49.

Garey, M.; Johnson, D. S. 1979: Computers and intractability: a guide to the theory of NP-Completeness. New York, W. H. Freeman and Co.

Gusfield, D. 1984: The steiner tree problem in phylogeny. *Technical report 332*. Yale University, Department of Computer Science.

Gusfield, D. 1991: Efficient algorithms for inferring evolutionary trees. *Networks 21*: 19–28.

Idury, R.; Schaffer, A. 1993: Triangulating three-colored graphs in linear time and linear space. *SIAM journal on discrete mathematics 6*: 289–294.

Kannan, S.; Warnow, T. 1992: Triangulating three-colored graphs. *SIAM journal on discrete mathematics 5*: 249–258.

Kannan, S.; Warnow, T. in press: Inferring evolutionary history from DNA sequences. *SIAM journal on computing 23*.

Kernighan, B.; Ritchie, D. 1988: The *C* programming language. Englewood Cliffs, New Jersey, Prentice Hall.

Kluge, A. G.; Farris, J. S. 1969: Quantitative phyletics and the evolution of anurans. *Systematic zoology 18*: 1–32.

LeQuesne, W. J. 1969: A method of selection of characters in numerical taxonomy. *Systematic zoology 18*: 201–205.

LeQuesne, W. J. 1972: Further studies on the uniquely derived character concept. *Systematic zoology 21*: 281–288.

LeQuesne, W. J. 1974: The uniquely evolved character concept and its cladistic application. *Systematic zoology 23*: 513–517.

LeQuesne, W. J. 1977: The uniquely evolved character concept. *Systematic zoology 26*: 218–223.

McMorris, F. R. 1977: On the compatibility of binary qualitative taxonomic characters. *Bulletin of mathematical biology 39*: 133–138.

McMorris, F. R.; Warnow, T.; Wimer, T. in press: Triangulating vertex colored graphs. *SIAM journal on discrete mathematics 7*.

Steel, M. A. 1992: The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of classification 9*: 91–116.

Swofford, D. 1993: PAUP (Phylogenetic Analysis Using Parsimony). It can be ordered from the Center for Biodiversity, Illinois Natural History Survey, 607 East Peabody Drive, Champaign, Illinois 61820, U.S.A.