Faster reliable phylogenetic analysis

Vincent Berry* David Bryant†

December 19, 1998

Abstract

We present fast new algorithms for phylogenetic reconstruction from distance data or weighted quartets. The methods are *conservative*—they will only return edges that are well supported by the input data. This approach is not only philosophically attractive; the conservative tree estimate can be used as a basis for further tree refinement or divide and conquer algorithms. The capability to process quartet data allows these algorithms to be used in tandem with ordinal or qualitative phylogenetic analysis methods, such as maximum likelihood.

We provide algorithms for three standard conservative phylogenetic constructions: the Buneman tree, the Refined Buneman tree, and split decomposition. We introduce and exploit combinatorial formalisms involving trees, quartets, and splits, and make particular use of an attractive duality between unrooted trees, splits, and dissimilarities on one hand, and rooted trees, clusters, and similarity measures on the other. Using these techniques, we achieve $O(n)$ improvements in the time complexity of the best previously published algorithms (where n is the number of studied species). Our algorithms will be included in the next edition of the popular *SplitsTree* software package.

Keywords: computational biology, phylogeny reconstruction, combinatorially reliable edges, distance based methods, quartet based methods, single linkage tree, polynomial time algorithms.

1 Introduction

Inferring evolutionary trees, or *phylogenies*, is a well-known problem in computational biology, which aims at reconstructing the history of a set of sequences, genes or, more generally, species.

A fundamental drawback of many phylogenetic reconstruction methods, is that they will always return a fully resolved (binary) tree, even when the data set contains little phylogenetic information. While a fully resolved tree might seem more informative, in reality many of its internal edges could be artifacts of the method and of the particular data set analyzed, rather than issuing from the underlying tree [8, 9].

To avoid this problem, several studies have investigated distance based methods proposing evolutionary trees whose edges are supported by a significant number of combinatorial constraints [11, 14, 29]. These constraints are expressed on *quartets* of species, a basic structural unit to describe trees, which can be efficiently inferred from biological data [3, 7, 35, 36] and which have received much attention recently [10, 11, 13, 18, 25, 35].

One example is the Q^* method [3, 11] which has a strong connection with a tree construction introduced by Buneman in [16]. The Q^* method returns a tree containing only combinatorially safe edges, each supported by $\Omega(n^2)$ to $O(n^4)$ quartets. Unfortunately it is often the case that only a few edges in the historical phylogeny have a strong combinatorial support in the data.

Moulton and Steel [29] investigated a related method, called the *Refined Buneman tree*, that provides a *refinement* of the tree obtained by the Buneman construction (*i.e.*, a tree containing at least the edges inferred by the Buneman method). The edges of the Refined Buneman tree still satisfy between $\Omega(n)$ and $O(n^3)$ quartets inferred from the data.

The interest in these methods is twofold: first, they provide conservative but reliable estimates of the species history (*e.g.*, the Q^* method was experimentally shown to induce less than 1% incorrect edges [10, 11, 31]); more importantly, because of the important constraints they impose on inferred edges, the corresponding tree can be exactly computed in polynomial time, whereas most other methods use NP-hard reconstruction criteria, *i.e.*, requiring exponential time for an exact solution to be found.

This motivates a new and promising approach to phylogeny reconstruction [10, 11, 31, 32]: first compute a tree containing only safe edges through one of the above methods and then use this tree as a starting

*EURISE, Département de Mathématiques, Université de Saint-Etienne, 23, Rue du Docteur Paul Michelon, 42023 Saint-Etienne Cedex 2, France. E-mail vberry@univ-st-etienne.fr

†C.R.M. Université de Montréal, C.P. 6128, Succ. centre-ville, Montréal, (Québec) H3C 3J7. E-mail bryant@CRM.UMontreal.ca

point for more refined, and computationally intensive, reconstruction procedures [10, 31]. Alternatively, one can divide the data set into loosely grouped subsets of species and then apply these conservative tree methods to the resulting subproblems, an approach taken by the disc-cover method [24].

There are times when the data simply does not support a tree and forcing a tree structure onto the data can result in lost information. This occurs, for example, when there has been hybridization or horizontal gene transfer. With this problem in mind, Bandelt and Dress introduced split decomposition, which constructs a network instead of a simple phylogeny. We will see that split decomposition is an exact analogue of the Buneman tree construction, and this relationship proves useful when developing fast algorithms. Split decomposition has been implemented in the package *SplitsTree* [23] and has been successfully applied to the analysis of virus data [17].

In this paper, we present algorithms achieving an $O(n)$ improvement in the time complexity of the best previously published algorithms for computing the Buneman tree, the Refined Buneman tree and split-graph on an n species dataset. Our algorithms apply to dissilimilarity matrices or to sets of weighted quartets, as can be obtained by usual quartet inference methods [3, 20, 26]. The structure of the paper is as follows:

- In the next section we define the fundamental phylogenetic objects on which our algorithms are based, as well as introducing connections and dualities between them.

- In section 3 we exploit links between the Buneman tree, strong clusters and the single linkage tree to develop an $O(n^2)$ time algorithm for computing strong clusters and an $O(n^3)$ time algorithm for constructing the Buneman tree.

- In section 4 we employ a new path covering technique to give an $O(n^5)$ time algorithm for the Refined Buneman tree.

- In section 5 we use similar techniques to those developed in section 3 to give an $O(n^5)$ time algorithm for split decomposition.

2 Preliminaries

Rooted and unrooted trees

An **unrooted (phylogenetic) tree** is a connected acyclic graph with all vertices of degree less than three (and possibly some higher degree vertices) labeled injectively from some label set X (representing species or genes). An unrooted tree is **binary** if every internal vertex has degree three.

Given any three leaves a, b, c let $mid(a,b,c)$ denote the unique internal vertex located at the intersection of the three paths between a and b, b and c, a and c. Hence $mid(a,b,c) = mid(a,c,b) = mid(b,c,a)$.

The dual of an unrooted tree is a rooted tree, which is defined in the same way except that one internal vertex is distinguished and called the root, and this vertex can have degree two. A rooted tree is binary if all internal vertices except the root have degree three and the root has degree two. Rooted and unrooted trees are easily transformed into one another (*e.g.*, by using an outgroup).

2.1 Splits and clusters

A **split** of a finite set is a partition of a set into two nonempty parts. Removing an edge e from an unrooted phylogenetic tree T partitions the leaf set of the tree into two parts—this is the split of T associated with e. The complete set of splits obtained this way is denoted $splits(T)$. The splits $A|B$ where $|A| = 1$ or $|B| = 1$ are called trivial. Any unrooted tree T can be reconstructed from its set of splits in linear time [22, 28].

A set of splits $\mathcal{S}$ is **compatible** if $\mathcal{S} \subseteq splits(T)$ for some tree T. It is well-known that two splits $A|B$ and $C|D$ are compatible if and only if one of $A \cap C$, $A \cap D$, $B \cap C$, $B \cap D$ is empty and that a set of splits is compatible if and only if it is pairwise compatible [16].

A set of splits $\mathcal{S}$ is **weakly compatible** if for all $A_1|B_1, A_2|B_2, A_3|B_3$ at least one of $A_1 \cap A_2 \cap A_3$, $A_1 \cap B_2 \cap B_3$, $B_1 \cap A_2 \cap B_3$, or $B_1 \cap B_2 \cap A_3$ is empty. Every compatible set of splits is weakly compatible, so systems of weakly compatible splits can be seen as a generalization of unrooted trees. Sets of weakly compatible splits can be represented as a **splits graph** where every split corresponds to a set of parallel edges that form a cut set of the graph.

The rooted dual of a split is a cluster, which is simply a subset of a finite set X. If we fix a leaf x then the cluster corresponding to split $A|B$ with $x \in B$ is the subset $A \subset X$. Conversely the split corresponding to a cluster A of $X - \{x\}$ is the split $A|X - A$. Given any collection $\mathcal{C}$ of clusters we say that one cluster $A \in \mathcal{C}$ **covers** another cluster $B \subsetneq A$ if $\nexists C \in \mathcal{C}$ s.t. $B \subsetneq C \subsetneq A$.

Removing an edge e in a rooted tree partitions the tree into two parts. The set of leaves in the part not containing the root is called the **cluster corresponding to** v, where v is the endpoint of e furthest away from the root. A set of clusters obtained this way is said to be **compatible**. Compatible sets of clusters (also called **strong hierarchies** or just **hierarchies**) are characterized by the property that for any two clusters A and B in the set, if $A \cap B \neq \emptyset$ then $A \subseteq B$ or $B \subseteq A$.

A set of clusters $\mathcal{C}$ forms a **weak hierarchy** if for all $A, B, C \in \mathcal{C}$ at least one of $(A \cup B) - C$, $(A \cup C) - B$, $(B \cup C) - A$ is empty. Any strong hierarchy is also a weak hierarchy.

Given any collection of clusters $\mathcal{C}$ of a finite set X with $X \in \mathcal{C}$, and a subset $Y \subseteq X$, the closure of Y is given by

$$\langle Y \rangle = \bigcap_{A \in \mathcal{C}: Y \subseteq A} A$$

If $\mathcal{C}$ is closed under intersections then $\langle Y \rangle$ is contained in $\mathcal{C}$ for all $Y \subseteq X$. It is shown in [4] that if A is any cluster in a weak hierarchy then there is a, a' such that $\langle a, a' \rangle = A$. In strong hierarchies (rooted trees), $A = \langle a, a' \rangle$ if and only if A is the cluster corresponding to the least common ancestor of a and a'.

2.2 Quartets

To every set of four species $a, b, c, d \in X$ there are three ways to associate a leaf-labeled binary unrooted tree. The three possible resolutions are denoted $ab|cd$, $ac|bd$ and $ad|bc$, indicating how the central edge of the tree splits the four species. Each of these trees on four species is also called a *resolved quartet*, or simply a *quartet* [11, 14, 18, 29]. Species of quartets $ab|cd$ might not be necessary distinct, *i.e.*, we will also consider cases in which, say $a = b$ or $c = d$.

An unrooted tree T induces a quartet $ab|cd$ if we can remove leaves and contract edges of T to obtain $ab|cd$. In other words, T induces $ab|cd$ if the path from a to b in T does not intersect the path from c to d. Note that an unrooted tree T is uniquely characterized by the set $q(T)$ of quartets it induces, from which it can be reconstructed in $O(|q(T)| + n^2)$ time [3].

The set of quartets $q(A|B)$ corresponding to a split $A|B$ is defined by

$$q(A|B) = \{aa'|bb' : a, a' \in A, b, b' \in B\}.$$

Here a and a' need not be distinct; likewise for b and b'. It follows that $q(T) = \cup_{A|B \in splits(T)} q(A|B)$ and a set of splits $\mathcal{S}$ is compatible if and only if $q(\mathcal{S}) = \cup_{A|B \in \mathcal{S}} q(A|B)$ contains at most one quartet on each set of four leaves. Similarly, a set of splits $\mathcal{S}$ is weakly compatible if and only if $q(\mathcal{S})$ contains at most two quartets on each subset of four leaves [6].

Quartets have been shown to be a valuable intermediary step in phylogenetic tree reconstruction. They allow, *e.g.*, to apply some computationally expensive optimization criteria, like Maximum Likelihood [20], to data sets containing more than a moderate number of species, by restricting the analysis to four taxa at a time and then constructing trees by combining the inferred quartets according to a reconstruction principle [3, 10, 11, 15, 18, 25, 36, 38].

2.3 Dissimilarity measures and trees

Let T be any unrooted tree and suppose that the edges of T are weighted. The weighting induces a dissimilarity measure $d(T)$ on the leaf set of T: the distance between two leaves is taken to be the sum of the weights along the unique path connecting them. Dissimilarity measures arising in this way are called **additive** and can be easily characterized [16].

Given a split $A|B$ we define the **split metric** of $A|B$ by

$$\delta_{A|B}(a,b) = \begin{cases} 0 & if\ \{a,b\} \subseteq A \text{ or } \{a,b\} \subseteq B \\ 1 & \text{otherwise} \end{cases}$$

In a weighted unrooted tree T, if $\lambda_{A|B}$ is the weight of the edge associated with split $A|B$ then the dissimilarity measure $d(T)$ is given by

$$d(T) = \sum_{A|B \in splits(T)} \lambda_{A|B} . \delta_{A|B} .$$

In all of the three standard methods below the output is a compatible or weakly compatible set of splits together with their weights.

2.4 Conservative phylogenetic reconstruction methods

In this section we present the phylogeny reconstruction methods for which we provide improved algorithms. These methods are conservative in the sense that they output trees or networks containing only edges strongly supported by the data. Our algorithms can handle both distance data and weighted sets of quartets.

2.4.1 The Buneman Tree

Buneman showed how to construct a weighted unrooted tree from a dissimilarity measure d on X by considering quartets [16]. The *Buneman score* of a quartet $q = wx|yz$, $w, x, y, z \in X$, is defined as:

$$\beta_q = \beta_{wx|yz} := \tfrac{1}{2}(\min\{wy + xz, wz + xy\} - (wx + yz)),$$

where $d(x,y)$ is written xy for all $x, y \in X$. The *Buneman index* of a split $U|V$ of X is

$$\mu_{U|V} = \mu_{U|V}(d) = \min_{u,u' \in U, v,v' \in V} \beta_{uu'|vv'} .$$

Here u and u' need not be distinct; likewise for v and v'. Buneman showed that the set of splits $B(d) = \{U|V : \mu_{U|V}(d) > 0\}$ is compatible. We define the *Buneman tree* to be the weighted unrooted tree associated to $B(d)$, whose edges correspond to the splits $U|V \in B(d)$ and are weighted according to $\mu_{U|V}(d)$.

Figure 1 (i) shows the Buneman tree calculated for the mammalian data set of [30], containing 188 nucleotides, obtained from several genes (α- and β-hemoglobins, fibrinopeptides A and B, cytochrome c, myoglobins, α-chrystallin). There is enough structure in this data for the Buneman tree to infer several splits, corresponding to expected groups, *e.g.*, primates (ape, human, monkey), ungulates (cow, sheep, pig, horse). However, the Buneman method does not take any decision relative to the dog, kanga, rabbit and rodent species, whose position is still uncertain [30].

Buneman's method can be generalized to the case when quartets are not necessarily weighted by their Buneman scores [3]. Given any set of quartets Q and a weighting function w for Q such that at most one quartet has positive weight for each subset of four species, the set of splits

$$S(w) = \{U|V : \forall q \in q(U|V), w(q) > 0\}$$

is compatible, giving a tree which can be obtained in $O(n^4)$ time, through the Q^* method [11]. Moreover, weighting each split $U|V$ of $S(w)$ by $\min_{uu'|vv' \in q(U|V)} w(uu'|vv')$ produces a weighted unrooted tree that is an analogue to the Buneman tree.

2.4.2 Anchored Buneman tree

One relaxation of the condition that $\mu_{A|B} > 0$ is to only look at quartets containing a certain leaf. Fix $x \in X$. For each split $U|V$ with $x \in U$ define

$$\mu^x_{U|V} = \mu^x_{U|V}(d) := \min_{u \in U, v, v' \in V} \{\beta_{xu|vv'}\},$$

and put $B_x(d) = \{U|V \text{ s.t. } \mu^x_{U|V} > 0\}$. The set of splits $B_x(d)$ is compatible [14]. The weighted unrooted tree associated to $B_x(d)$, whose edges correspond to the splits $U|V \in B_x(d)$ and are weighted according to $\mu^x_{U|V}(d)$, is called the *Buneman tree anchored at* x.

Clearly $\mu^x_{U|V} \geq \mu_{U|V}$ for all splits $U|V$, so that $B(d) \subseteq B_x(d)$. Indeed $\mu_{U|V} = \min_{x \in X} \mu^x_{U|V}$ so $B(d) = \cap_{x \in X} B_x(d)$.

There is a straightforward generalization of the anchored Buneman tree for when the input is a quartet weight function. Given any set of quartets Q and a weighting function w for Q such that at most one quartet has positive weight for each subset of four species, define $q_x(U|V) = \{uu'|vv' \in q(U|V) : u' = x\}$ and

$$B_x(w) := \{U|V : q_x(U|V) \subseteq Q\}$$

where each split in $B_x(w)$ is weighted by

$$\mu^x_{U|V}(w) := \min_{q \in q_x(U|V)} w(q).$$

2.4.3 The Refined Buneman tree

Recently, Moulton and Steel [29] have shown that another special relaxation of the condition $\mu_{U|V} > 0$ also gives a set of compatible splits. Given two splits $U|V$ and $U'|V'$ put

$$conf l(U|V, U'|V') =$$
$$\{uu'|vv' \in q(U|V) : uv|u'v' \in q(U'|V') \text{ or } uv'|u'v \in q(U'|V')\}.$$

Then, $U|V$ and $U'|V'$ are compatible if and only if $conf l(U|V, U'|V') = \emptyset$. Hence if Q contains no two quartets on the same set of four species, and both $q(U|V) \subseteq Q$ and $q(U'|V') \subseteq Q$ then $U|V$ and $U'|V'$ must be compatible.

Moulton and Steel observed that, if $U|V$ and $U'|V'$ are incompatible then $conf l(U|V, U'|V') \geq n - 3$. They defined a split index $\bar{\mu}_{U|V}$ such that $conf l(U|V, U'|V') \geq n - 3$ implies $\bar{\mu}_{U|V} + \bar{\mu}_{U'|V'} \leq 0$. Hence the set of splits $\{U|V : \bar{\mu}_{U|V}(d) > 0\}$ is compatible.

Given a split $U|V$ put $m = |q(U|V)|$ and let $q_1, \ldots, q_m$ be an ordering of the elements in $q(U|V)$ such that for all $1 \leq i \leq j \leq m$ we have $\beta_{q_i} \leq \beta_{q_j}$. Put $q_{min}(U|V) = \{q_1, q_2, \ldots, q_{n-3}\}$. The *Refined Buneman index* of a split $U|V$ is defined as

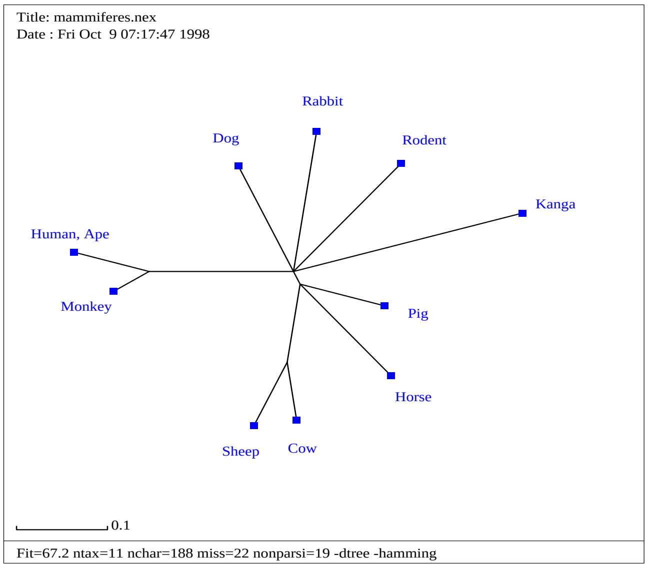
$$\begin{aligned} \bar{\mu}_{U|V} &= \bar{\mu}_{U|V}(d) \\ &:= \frac{1}{n-3} \cdot \sum_{i=1}^{n-3} \beta_{q_i} \\ &= \frac{1}{n-3} \cdot \sum_{q \in q_{min}(U|V)} \beta_{q_i} \end{aligned}$$

Our definition differs slightly from the definition in [29] since $q(U|V)$ includes quartets $uu'|vv'$ with $u = u'$ or $v = v'$. The change has no effect if the distance matrix satisfies the triangle inequality.

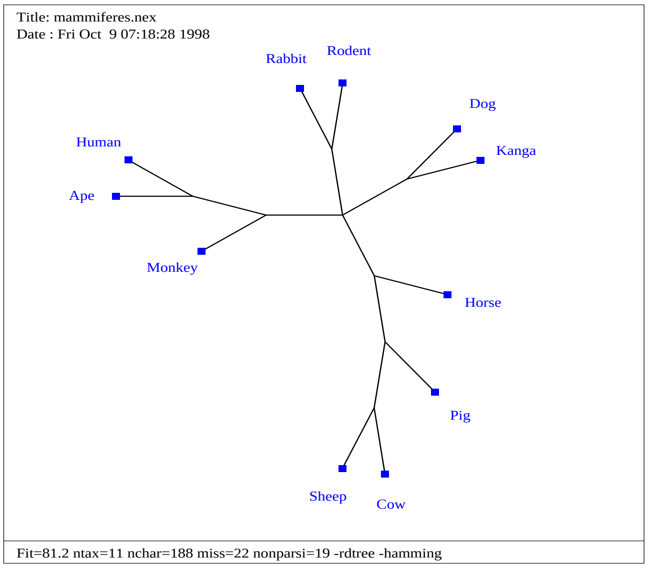
Put $RB(d) = \{U|V : \bar{\mu}_{U|V}(d) > 0\}$. The associated weighted unrooted-tree, whose edges correspond to the splits $U|V \in RB(d)$ and are weighted according to $\bar{\mu}_{U|V}(d)$, is called the *Refined Buneman tree*. It is clear that $B(d) \subseteq RB(d)$, and often $B(d)$ is strictly contained in $RB(d)$, in which case the Refined Buneman tree refines the Buneman tree.

The Refined Buneman tree (Figure 1 (ii)) adds three splits to the previous mammal tree. We draw the tree with equal length edges so that the additional splits are easy to identify (they have only small weight). The cluster (pig,sheep,cow) is expected; the cluster (dog,kanga) is also found by Neighbour Joining and Maximum Parsimony; the cluster (rabbit, rodent), also found by Maximum Parsimony and Neighbour Joining, is suspected to be wrong due to the long branch attract problem, but no firm conclusion as been reached concerning this group [30].

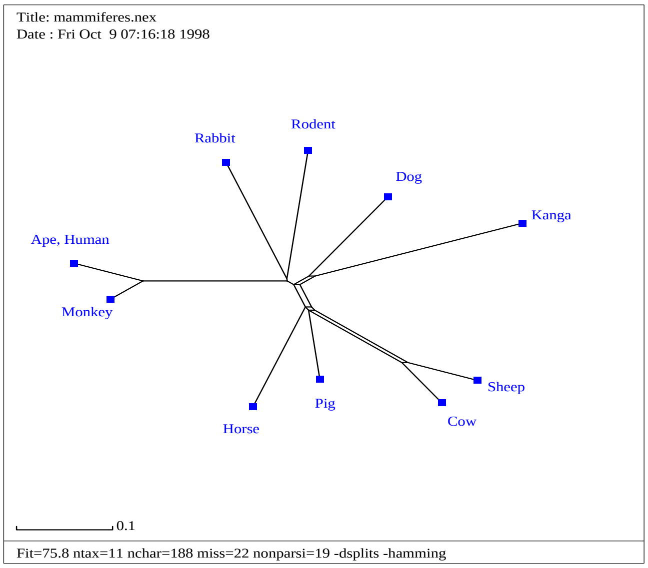
Like the Buneman and anchored Buneman trees, this method can be modified to handle more general quartet weights. Suppose that w is a weight on quartets such that $w(ab|cd) + w(ac|bd) \leq 0$ for any two quartets $ab|cd, ac|bd$ on the same set of species. Once again, let $q_{min}(U|V)$ be the set of $n - 3$ minimum score quartets in $q(U|V)$. A modification of the proof in [29] shows



(i) Buneman tree on the mammal data set [28]



(ii) Refined Buneman tree for the mammal data set [28]
(drawn with equal length edges)



(iii) Splits graph for the mammal data set [28]

Figure 1: Buneman tree, Refined Buneman tree, and splitsgraph for the mammal data set [30]

that if we define

$$\bar{\mu}_{U|V} = \bar{\mu}_{U|V}(w) := \frac{1}{n-3} \cdot \sum_{q_i \in q_{min}(U|V)} w(q_i).$$

then the set of splits $RB(w) = \{U|V : \bar{\mu}_{U|V} > 0\}$ is compatible.

2.4.4 The C-tree construction

We have recently discovered that the refined Buneman method can be further refined. It can be shown that

$$|confl(U|V, U'|V')| \geq (|U|-1)(|V|-1)$$

for any pair of incompatible splits $U|V$ and $U'|V'$. As before, put $m = |q(U|V)|$ and let $q_1, \ldots, q_m$ be an ordering of the elements in $q(U|V)$ such that for all $1 \leq i \leq j \leq m$ we have $\beta_{q_i} \leq \beta_{q_j}$. Define

$$\mu^*_{U|V}(d) = \frac{1}{(|U|-1)(|V|-1)} \cdot \sum_{i=1}^{(|U|-1)(|V|-1)} \beta_{q_i}.$$

By a similar argument to the refined Buneman tree case, the set of splits $CT(d) := \{U|V : \mu^*_{U|V}(d) > 0\}$ is shown to be compatible, and the associated unrooted tree with edge weights given by $\mu^*(d)$ is called the C-tree for d. As in the Buneman tree and refined Buneman tree cases, the C-tree can also be extended to handle sets of weighted quartets.

Clearly, $RB(d) \subseteq CT(d)$, and there are cases in which the C-tree strictly refines the refined Buneman tree. Extending the algorithm for $RB(d)$ gives an $O(n^5)$ time algorithm for the C-tree, though the C-tree algorithm is significantly more involved and will be left for an extended version of this paper.

There is a useful and intriguing link between the C-tree construction and the *Quartet Cleaning* method [12, 25]. Suppose that Q is a set of quartets containing at most one quartet for every set of four species. For consistency with Quartet Cleaning we assume Q contains all quartets $uu'|vv'$ such that $u = u'$ or $v = v'$ or both. If we put $w(ab|cd) = 1$ for all $ab|cd \in Q$ and $w(ab|cd) = -1$ for all $ab|cd \notin Q$ then the splits in $CT(w)$ are exactly those splits $U|V$ with $|q(U|V) - Q| < (|U|-1)(|V|-1)/2$ (the $\frac{1}{2}$ constant is due to the averaging of quartets with scores +/- 1).

2.4.5 Split decomposition and the splitsgraph

Just as weak compatibility of splits generalizes compatibility of splits, the d-split construction of Bandelt and Dress [5] generalizes the tree construction method of Buneman. We replace the Buneman quartet score β_q with the **weak quartet score** β^*_q for a quartet $q = wx|yz$, $w, x, y, z \in X$, defined as

$$\beta^*_q = \beta^*_{wx|yz} := \tfrac{1}{2}(\max\{wy + xz, wz + xy\} - (wx + yz)).$$

where, as before, $d(x, y)$ is denoted xy, all $x, y \in X$. The **(weak) isolation index** of a split $U|V$ is then defined

$$\alpha_{U|V} = \alpha_{U|V}(d) = \min_{u,u' \in U, v,v' \in V} \beta^*_{uu'|vv'}.$$

Here u and u' need not be distinct; likewise for v and v'. A split $U|V$ is called a d-split if $\alpha_{U|V} > 0$. Bandelt and Dress [5] showed that the set of d-splits of a distance function d is weakly compatible. They use this construction to describe a canonical decomposition of metrics into split metrics (and a residue).

The split decomposition approach can be applied to sets of weighted quartets. Given a set of quartets Q containing at most two quartets for each set of four leaves, the set of splits $S(Q) = \{U|V : q(U|V) \subseteq Q\}$ is weakly compatible. If w is a weighting function on quartets such that $w(ab|cd) + w(ac|bd) + w(ad|bc) \leq 0$ for all subsets of four leaves a, b, c, d then Q could be taken to be $\{ab|cd : w(ab|cd) > 0\}$. In this case the weight given to a split $U|V$ would be

$$w(U|V) = \min\{w(uu'|vv') : uu'|vv' \in q(U|V)\}.$$

Once the set of d-splits has been constructed, the set of weakly compatible splits can be used to construct a splitsgraph. A splitsgraph can be regarded as a generalized tree diagram, except that every split corresponds to a collection of parallel edges rather than a single edge. In many cases the splitsgraph looks like a normal tree with some additional boxes in regions of uncertainty. This is evident when we compute the splitsgraph for the mammal data (Figure 1 (iii)). The splitsgraph adds a split separating (rabbit,rodent,primates) from the other species, which was also proposed in [30].

3 An $O(n^3)$ time algorithm for the Buneman tree

The definition of the Buneman tree seems to imply that any algorithm for computing the tree and edge weights would take at least $\Omega(n^4)$ time—we need to calculate β_q for every possible quartet q. Here we provide an $O(n^3)$ algorithm. The gain in efficiency is achieved by converting the problem from one involving unrooted trees and distances to one involving rooted trees and similarities.

A **similarity measure** s on a finite set X is a symmetric function on $X \times X$. Intuitively, a high similarity between two objects indicates a high degree of relationship. The **strong isolation index** $i_s(A)$ of a cluster $A \subseteq X$ with respect to s is defined

$$i_s(A) = \min_{a,a' \in A, x \in X-A} \{s(a, a') - \max\{s(a, x), s(a', x)\}\}$$

which is equivalent to

$$i_s(A) = \min_{a,a' \in A, x \in X-A} \{s(a, a') - s(a, x)\}.$$

The clusters $\{A : i_s(A) > 0\}$ are called the **strong clusters** of s and form a (strong) hierarchy [4].

If we fix $x \in X$ then a distance d can be converted into a similarity s_x on $X - \{x\}$ using the **Farris transform**

$$s_x(a,b) = \tfrac{1}{2}(d(a,x) + d(b,x) - d(a,b))$$

for all $a, b \in X - \{x\}$. The inverse of the transform is given by

$$d(a,b) = s_x(a,a) + s_x(b,b) - 2s_x(a,b)$$

for all $a, b \in X - \{x\}$ with $d(a,x) = s_x(a,a)$ and $d(x,x) = 0$.

The connection with Buneman trees is provided by the following Lemma. It can be proved by expressing β_q in terms of s_x.

Lemma 1 *If d is a distance function on X with Farris transform s_x and $U|V$ is a split of X with $x \in U$ then*

$$\mu^x_{U|V} = i_{s_x}(V).$$

That is, strong clusters correspond to splits in the anchored Buneman tree.

Thus $A|B$ is a split in the Buneman tree if and only if A is a cluster of s_x for all $x \in B$ and B is a cluster of s_y for all $y \in A$. The question now becomes: how quickly can we construct the strong clusters?

3.1 Strong clusters and the single linkage tree

One of the most widely known tree constructions in classification is the single linkage clustering tree. It is the close cousin of popular phylogenetics algorithms neighbor joining [33] and UPGMA [34]. We use it to construct strong clusters.

Theorem 2 *If A is a strong cluster of s then A is a cluster in the single linkage tree for s.*

Proof
We use a characterization of single linkage trees described by [7] and rediscovered in [19] to solve a related problem. Given a similarity s on finite set X we construct the graph $G[k]$ with edge set $E[k] = \{\{a,b\} : s(a,b) \geq k\}$. A cluster is in the single linkage tree for s if and only if it is a component of $G[k]$ for some k.

Suppose that k is the maximum value such that A is connected in $G[k]$. Given any a in X and $x \in X - a$ if $\{a,x\} \in E[k]$ then $s(a,x) \geq k$ and so $s(a,a') > s(a,x) \geq k$ for all $a' \in A$, since A is a strong cluster. However this contradicts the maximality assumption for k. Hence A is a component of $G[k]$. □

This result has been proved independently in [2]. The single linkage tree for s can be constructed in $O(n^2)$ time using spanning tree based methods [7, 21, 27].

3.2 An $O(n^2)$ algorithm for strong clusters in a tree

We are able to create a superset of the collection of strong clusters in $O(n^2)$ time. The task that remains is to prune those clusters with zero, or negative, isolation index from this collection.

Let $\mathcal{C}$ be the collection of clusters returned by the single linkage algorithm. For each cluster $C_i \in \mathcal{C}$ and each $x \in C_i$ we calculate two values:

$$\begin{aligned} m(C_i,x) &:= min\{s(x,x') : x' \in C_i\} \\ M(C_i,x) &:= max\{s(x,x') : C_i = \langle x,x'\rangle\} \end{aligned}$$

Since $\mathcal{C}$ is compatible, $\langle x,x'\rangle = C_i$ if and only if C_i is the cluster corresponding to the least common ancestor of x and x'. Furthermore, if C_j is a cluster that is covered by C_i and contains x then

$$m(C_i,x) = \min\{m(C_j,x), \min\{s(x,x') : C_i = \langle x,x'\rangle\}\}.$$

Since each pair of leaves x, x' gives exactly one cluster $C_i = \langle x,x'\rangle$ the values m and M can be calculated for all leaves and clusters in $O(n^2)$ time, using a depth first search of the single linkage tree. Once these values are calculated the isolation indices can be computed:

Lemma 3 *If A is a cluster in $\mathcal{C} - \{X\}$ and B is a strong cluster that covers A in $\mathcal{C}$ then*

$$i_s(A) = \min_{a \in A}\{m(A,a) - M(B,a)\}.$$

Proof
The strong cluster B has positive isolation index so $s(a,y) > s(a,x)$ for all $a' \in A$, $y \in B - A$ and $x \in X - B$. Hence

$$\begin{aligned} i_s(A) &= \min_{a,a' \in A, y \in B-A}\{s(a,a') - s(a,y)\} \\ &= \min_{a \in A}\left\{ \min_{a' \in A}\{s(a,a')\} - \max_{y \in B-A}\{s(a,y)\} \right\} \\ &= \min_{a \in A}\{m(A,a) - M(B,a)\} \end{aligned}$$

as required. □

The values $i_s(A)$ can now be calculated for all clusters using a preorder traversal of the tree. Whenever a cluster is found with zero or negative isolation score, we remove the cluster from the tree.

Applying the pruning procedure to the single linkage tree gives us the strong clusters with their isolation index (algorithm 1 in the Appendix). We have now established

Theorem 4 *The strong clusters of a similarity measure can be recovered, together with their isolation indices, in $O(n^2)$ time.*

3.3 An $O(n^3)$ algorithm for the Buneman tree

We can now exploit the relationship between the anchored Buneman trees and the Buneman tree. We let s_i denote s_{x_i}, the Farris transform of d with respect to x_i. We construct the set of clusters from the single linkage tree of s_1 then successively prune off clusters that are not strong clusters for some other s_i (algorithm 2 in the Appendix). At the conclusion of the algorithm each split $A|B$ in $\mathcal{S}$ is weighted by $\min_{x \in X}\{\mu^x_{A|B}\} = \mu_{A|B}$, the Buneman score for $A|B$.

Note that a number of shortcuts can speed up execution, however they do not improve the order of time complexity and will be reserved for an extended version of this paper.

4 Computing Refined Buneman trees in $O(n^5)$ time

The algorithm of Bryant and Moulton [14] for computing the splits $RB(d)$ of the Refined Buneman tree is iterative. It assumes an arbitrary order on the species $X = \{x_1, \dots, x_n\}$, computes $RB(d)$ for a small subset of species then extends it by progressively incorporating the other species. The same iterative technique has been successfully used to solve other related problems [5, 11].

Put $X_k = \{x_1, \dots, x_k\}$ and let d_k be the dissimilarity d restricted to X_k. Each iteration step was based on the following Lemma:

Lemma 5 ([14]) *Suppose $|X| > 4$, and fix $x \in X$. If $\sigma = \{U, V\}$ is a split in $RB(d)$ with $x \in U$, and $|U| > 2$, then either $\{U, V\} \in B_x(d)$ or $\{U - \{x\}, V\} \in RB(d_{X-\{x\}})$ or both.*

Thus, the algorithm of [14] first computes $RB(d_4) = B(d_4)$, then at each step k (k ranging from 5 to n), the set $RB(d_k)$ is computed by considering the splits $U|V \in B_{x_k}(d_k)$ as well as $\{U \cup x_k, V\}$ and $\{U, V \cup x_k\}$ for each $\{U, V\} \in RB(d_{k-1})$. From these splits, only those having a positive Refined Buneman index are included in $RB(d_k)$.

The most time consuming step of the $O(n^6)$ time algorithm in [14] is computing the index of the splits considered for addition in the set $RB(d_k)$ at each step k. To compute the index of a split, we have to know the $k-3$ quartets of least Buneman score it induces. The set Q_X of all possible quartets with leaves in X is sorted at the beginning of the algorithm according to their Buneman score. Then, for each examined split $U|V$, the algorithm proceeds in ascending order through the sorted list Q_X, to find the $k-3$ quartets in $q(U|V)$ with smallest Buneman score. However, this can require up to $O(n^4)$ for each split, *i.e.*, $O(n^5)$ at each step ($O(k)$ splits are considered), hence the $O(n^6)$ complexity of the algorithm.

Our approach is to partition the quartet set into disjoint subsets, obtain the $n-3$ smallest quartet values for each subset, then calculate Refined Buneman scores using an analogue of merge sort.

4.1 Calculating Refined Buneman scores path by path

Let T be an unrooted tree and let $wx|yz$ be a quartet in $q(T)$. The path between w and y, and the path between x and z intersect along an internal path in the tree: the path connecting $mid(w, x, y)$ and $mid(w, y, z)$. Thus to every quartet there corresponds a unique internal path in the tree. Let $\mathcal{P}$ be the set of paths connecting internal vertices in T. For each $P \in \mathcal{P}$ let Q_P be the set of quartets corresponding to P. Thus $q(T)$ is the disjoint union of $\{Q_P : P \in \mathcal{P}\}$. Furthermore $wx|yz \in Q_P$ if and only if $wx|yz \in q(U|V)$ for exactly those splits $U|V$ corresponding to edges along P.

Given an internal edge e (with corresponding split $U|V$) let $\mathcal{P}(e)$ be the set of paths in $\mathcal{P}$ that traverse e. Then $|\mathcal{P}(e)| \le |\mathcal{P}| < n^2$ and $q(U|V)$ is the disjoint union of $\{Q_P : P \in \mathcal{P}(e)\}$. Hence the set $q_{min}(U|V)$ of $n-3$ quartets in $q(U|V)$ with minimum score can be constructed from the sets of $n-3$ minimal quartets contained in each Q_P such that $P \in \mathcal{P}(e)$. This idea forms the basis of algorithm 3 which calculates Buneman scores of all splits in a tree in just $O(n^4)$ time. We will assume, without loss of generality, that the tree is binary, and that the set of all $3\binom{n}{4}$ quartets on X has been sorted according to quartet score.

Note that locating all of the vertices $mid(a, b, x_0)$ takes a total of $O(n^2)$ time using a depth first search of the tree, and the map between edges e and path sets $\mathcal{P}(e)$ can be constructed in $O(n^3)$ time. After preprocessing we can determine the path P corresponding to a quartet $wx|yz$ in $O(1)$ time by examining the vertices $\{mid(a, b, x_0) : a, b \in \{w, x, y, z\}\}$. The quartets in $q(T)$ are already sorted, so each insertion in the list corresponding to a path takes $O(1)$ time. There are $O(n^2)$ paths in each $\mathcal{P}(e)$ so it takes $O(n^3)$ time to calculate the Refined Buneman score for each edge. We have now established

Lemma 6 *If we are given a sorted list of quartets in $q(T)$ then it takes at most $O(n^4)$ time to calculate the Refined Buneman scores for every edge of T.*

4.2 Constructing the Refined Buneman tree in $O(n^5)$ time

We now proceed from calculating Refined Buneman scores to constructing Refined Buneman trees. Algorithm 4 calculates the splits in $RB(d)$, as well as their weights, in $O(n^5)$ time. The algorithm requires $O(n^4)$ memory space. This amount of memory usage is quite acceptable when the input data is a set of weighted quartets, however it should be possible to improve space complexity when the input is an $n \times n$ distance matrix. In any case, sophisticated bitmap techniques enable quartet computation with 200 to 300 taxa (D. Swofford, personal communication).

5 Split decomposition

The definition of d-splits makes it clear that split decomposition is to weakly compatible splits what the Buneman tree and the Q^* tree are to compatible splits. That said, can we use an analogue of our fast Buneman tree algorithm to quickly perform split decomposition? The answer is yes, however we have to be careful. Systems of weakly compatible splits are a bit more complicated than trees.

Once again we convert the problem from distances to similarities. The **weak isolation index** of a cluster $A \subseteq X$ with respect to a similarity measure s on X is defined

$$i_s^*(A) = \min_{a,a' \in A, x \in X-A} \{s(a,a') - \min(s(a,x), s(a',x))\}$$

The set of clusters $\{A : i_s^*(A) > 0\}$ are called the **weak clusters** of s and form a weak hierarchy [4].

The connection back to d-splits is provided by

$$i_s^*(U) = \min\{\beta^*_{uu'|vx} : uu'|vx \in q(U|X-U)\}$$

where s is the Farris transform of d with respect to x. Thus $U|V$ is a d-split if and only if U is a weak cluster of s_x for all $x \in V$ and V is a weak cluster of s_y for all $y \in U$.

At the moment, the fastest algorithm for calculating the weak clusters of a similarity measure is still the original iterative $O(n^5)$ time algorithm of [4]. There appears to be no analogue of the single linkage tree connection in the strong cluster case.

Let $\mathcal{C}$ be the set of weak clusters with respect to s_x for a leaf x. We wish to prune out clusters of $\mathcal{C}$ that do not correspond to d-splits.

Fix a leaf x. For each cluster $A \in \mathcal{C}$ and leaf $b \in X - A$ we calculate

$$F_x(A,b) = \min\{\beta^*_{aa'|bx} : a, a' \in A\}.$$

The function F_x can be calculated recursively using

$$\begin{aligned} F_x(A,b) \quad = \quad & \min\big\{ \min\{F(B,b) : A \text{ covers } B\}, \\ & \qquad \min\{\beta^*_{aa'|bx} : \langle a,a' \rangle = A\}\big\}. \end{aligned}$$

There are at most $O(n^3)$ covering relations in a weak hierarchy such as $\mathcal{C}$ that is closed under intersections. Hence for each x the value of $F_x(A,b)$ for each $A \in \mathcal{C}$ and $b \in X$ can be calculated in $O(n^4)$ time. Thus $F_x(A) := \min\{\beta^*_{aa'|bx} : a, a' \in A, b \in X - A\}$ can be all calculated in $O(n^4)$ time. We store these values for each cluster then repeat for the remaining $x \in X$. The isolation index for $A|(X-A)$ is the minimum of $F_x(A)$ over all $x \in X$. In this way, the d-splits can be obtained in $O(n^5)$ time using only $O(n^3)$ memory.

We summarize the method in algorithm 5. Once again we let s_i denote the Farris transform of d with respect to x_i.

Acknowledgements

The authors would like to thank D. Penny for providing the mammal data set, B. Leclerc and B. Fichet for helpful discussions. This work was carried out while D. Bryant held a Bioinformatics Postdoctoral Fellowship from the Canadian Institute for Advanced Research, Evolutionary Biology Program. Research supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canadian Genome Analysis and Technology grants to D. Sankoff. V. Berry was supported during part of this work by ESPRIT LTR Project no. 20244 — ALCOM-IT.

Appendix:- algorithm code

Algorithm 1: Strong clusters.

input : A similarity s on a set X of taxa.
output: The tree containing the strong clusters $C_i \in \mathcal{C}(s)$ and their strong isolation index $i_s(C_i)$.

```
Construct the single linkage tree T for s using Prim's algorithm.
C := clusters of T.
foreach C_i ∈ C in a depth first traversal of T
    Compute m(C_i, x) for all x ∈ C_i.
    Compute M(C_i, x) for all x ∈ C_i.
foreach C_i ∈ C in a pre-order traversal of T
    Calculate i_s(C_i) using Lemma 3.
    if i_s(C_i) ≤ 0 then
        Remove C_i from C
        Let C_j ∈ C be the cluster covering C_i,
        foreach x ∈ C_i    M(C_j, x) = max{M(C_j, x), M(C_i, x)}
Output C with weights i_s.
```

Algorithm 2: Buneman Tree.

input : A dissimilarity d on a set $x_1 \dots x_n$ of taxa.
output: The set of splits $\mathcal{S}$ of the Buneman tree with weights w.

```
Construct the set C of strong clusters for s_1
foreach A ∈ C
    S := S ∪ {A|X − A}
    w(A|X − A) := i_{s_1}(A)
foreach i = 2, 3, ..., n
    Construct Farris transform s_i
    C := ∅
    foreach A|B ∈ S with x_i ∈ B
        C := C ∪ {A}
    Prune clusters from C that are not strong clusters of s_i.
    foreach A ∈ C
        S := S ∪ {A|X − A}
        w(A|X − A) := min{w(A|X − A), i_{s_1}(A)}
Output S with weights.
```

Algorithm 3: Refined Buneman scores in a tree

Fix a leaf x_0 and use a depth first search of the tree to determine $mid(a, b, x_0)$ for every pair of leaves a and b.

for $i = 1..|q(T)|$ **do**

- Determine the path P such that $q_i \in Q_P$.
- Insert the score of q_i into the sorted list corresponding to Q_P.

Construct $\mathcal{P}(e)$ for each edge e.

foreach *edge* e

- Merge the sorted lists of values for each P in $\mathcal{P}(e)$ together, halting the process when we obtain $n-3$ minimum scores.
- Calculate the Refined Buneman score for e from this list.

foreach *split* $U|V$ *in* T Output $U|V$, $\mu_{\bar{U|V}}$ and $q_{min}(U|V)$.

Algorithm 4: A merge sort algorithm on quartets to construct the Refined Buneman tree.

Construct the list Q_X of all possible quartets $wx|yz$ on X, sorted according to weight.

$\mathcal{S}_4 := B(d_4)$.

foreach k *from 5 to* n

- $S_k := \{x_k|X_k - x_k\}$.
- Construct $B_x(d_k)$ for d_k.
- Calculate $\bar{\mu}_{U|V}$ and $q_{min}(U|V)$ for each $U|V \in B_x(d_k)$ using algorithm 3.
- $S_k := S_k \cup \{U|V \in B_x(d_k) : \bar{\mu}_{U|V} > 0\}$.
- Construct the sorted list $Q_k = \{ab|cx_k : a, b, c \in X_{k-1}\}$.
- **foreach** *split* $U|V \in S_{k-1}$
  - Construct a list L of $n-3$ smallest quartets in $Q_k \cap q(U \cup \{x_k\}|V)$.
  - Construct $q_{min}(U \cup \{x_k\}|V)$ from L and $q_{min}(U|V)$.
  - Construct a list L' of $n-3$ smallest quartets in $Q_k \cap q(U|V \cup \{x_k\})$.
  - Construct $q_{min}(U|V \cup \{x_k\})$ from L' and $q_{min}(U|V)$.
  - Calculate $\bar{\mu}_{U\cup\{x_k\}|V}$ and $\bar{\mu}_{U|V\cup\{x_k\}}$.
  - **if** $\bar{\mu}_{U\cup\{x_k\}|V} > 0$ **then**
    - $S_k := S_k \cup \{U \cup \{x_k\}|V\}$
  - **if** $\bar{\mu}_{U|V\cup\{x_k\}} > 0$ **then**
    - $S_k := S_k \cup \{U|V \cup \{x_k\}\}$

Output $RB(d) = \mathcal{S}_n$ and $\{\bar{\mu}_{U|V} : U|V \in \mathcal{S}_n\}$.

Algorithm 5: Split decomposition

input : a dissimilarity measure d on $X = \{x_1, x_2, \ldots, x_n\}$.
output: d-splits of d together with their weights.

Construct s_1, the Farris transform of d with respect to x_1.
Construct the weak clusters of s_1 using the iterative algorithm of [4].
Let $\mathcal{C}$ be this set of clusters and put $w(A) = i^*_{s_1}(A)$ for all $A \in \mathcal{C}$.
Sort $\mathcal{C}$ by set inclusion and tabulate covering relations.
for *all* $x \in X - \{x_1\}$ **do**
  for *all clusters* $A \in \mathcal{C}$ **do**
    for *all* $b \in X - A$ **do**
      $F_x(A, b) := \min\left\{\min\{F_x(B, b) : A \text{ covers } B\}, \beta^*_{aa'|bx} : \langle a, a'\rangle = A\right\}$
    $w(A) := \min\{w(A), \min\{F_x(A, b) \text{ over all} b \in X - A\}\}$.
for *all* $A \in \mathcal{C}$ **do**
  if $w(A) > 0$ **then**
    $S := S \cup \{A|X - a\}$.
    $w(A|X - A) := w(A)$.
Output S with weights.

References

[1] K. Atteson. The performance of neighbor-joining algorithms of phylogeny reconstruction. In *Proc. of COCOON*, Computing and Combinatorics, pages 101–110. Springer, 1997.

[2] J. P. Benzecri (1967) and coll. *Description mathématique des classifications*, volume 1 : La Taxinomie of *L'analyse des données*, chapter TIB No 3 [D.M.Cl.], pages 146–148. Dunod, Paris, 1973.

[3] H-J. Bandelt and A.W. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Adv. in appl. math.*, 7:309–343, 1986.

[4] H.-J. Bandelt and A.W. Dress. Weak hierarchies associated with similarity measures - an additive clustering technique. *Bull. Math. Biol.*, 51:133–166, 1989.

[5] H.-J. Bandelt and A.W. Dress. A canonical decomposition theory for metrics on a finite set. *Advances Math*, 92:47–105, 1992.

[6] H.-J. Bandelt and A.W. Dress. A relational approach to split decomposition. Tech. rep. Univ. Bielefeld. 1994.

[7] J.P. Barthélemy and A. Guénoche. *Trees and proximities representations.* Wiley, 1991.

[8] V. Berry and O. Gascuel. On the interpretation of bootstrap trees: appropriate threshold of clade selection and induced gain. *Mol. Biol. Evol.*, 13(7):999–1011, 1996.

[9] V. Berry and O. Gascuel. Choosing the tree which actually best explains the data: another look at the bootstrap in phylogenetic reconstruction. Proc. of the *2nd World Conference of the Int. Assoc. for Stat. Comput. Computing Science and Statistics*, 29(2):227–232, 1997.

[10] V. Berry and O. Gascuel. Reconstructing phylogenies from resolved 4-trees. Tech. rep. 97076, LIRMM, 1997.

[11] V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial confidence. *Theoretical Computer Science*, to appear, 1998.

[12] V. Berry, T. Jiang, P. Kearney, M. Li and T. Wareham. Quartet cleaning: improved algorithms and simulations *Submitted*, 1999.

[13] D. Bryant and M. Steel. Extension operations on sets of leaf-labelled trees. *Advances in Appl. Math.*, 16:425–453, 1995.

[14] D. Bryant and V. Moulton. A polynomial time algorithm for constructing the refined buneman tree. *Appl. Math. Lett.*, in press, 1998.

[15] D. Bryant and M. Steel. Fast algorithms for constructing optimal trees from quartets. SODA'99. (in press).

[16] P. Buneman. *Mathematics in Archeological and Historical Sciences*, chapter The recovery of trees from measures of dissimilarity, pages 387–395. Edhinburgh University Press, 1971.

[17] J. Dopazo, A. Dress, and A. von Haeseler. Split decomposition: A technique to analyze viral evolution. *Proc. Natl. Acad. Sci. USA*, 90:10320–10324, 1993.

[18] P.L. Erdös, M.A. Steel, L.A. Székely, and T.J. Warnow. Constructing big trees from short sequences. In *24th International Colloquium on Automata Langages and Programming*, 1997.

[19] M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13:155–179, 1993.

[20] J. Felsenstein. Evolutionary Trees from gene frequencies and quantitative characteres : finding maximum likelihood estimates. *Evolution*, 35(6), 1229-1242, 1981.

[21] Gower and Ross. Minimum spanning tree and single linkage cluster analysis. *Appl. Stats.*, 18:54–64, 1969.

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

[23] D. Huson. SplitsTree - a program for analyzing and visualizing evolutionary data. *Bioinformatics* 14(1):68–73, 1998.

[24] Huson, D., S. Nettles, T. Parida, T. Warnow, and S. Yooseph. The disc-covering method for tree reconstruction. *Proceedings of ALEX, 1998, Trento, Italy*, 1998.

[25] T. Jiang, P. Kearney, and M. Li. Orchestrating quartets: approximation and data correction. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, to appear, 1998.

[26] P. Kearney. The ordinal quartet method. *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 125–134, 1998.

[27] B. Leclerc. Description combinatoire des ultramétriques. *Math. Sci. Hum.*, 73:5–37, 1981.

[28] C. Meacham. A manual method for character compatibility. *Taxon*, 30:591–600, 1981.

[29] V. Moulton and M. Steel. Retractions of finite distance functions onto tree metrics. *Discrete Applied Math.*, 1998. (To appear).

[30] D. Penny and M.D. Hendy and M.A. Steel Testing the theory of descent. In M.M. Miyamoto and J. Cracraft, editors, *Phylogenetic analysis of DNA sequences*, 155-183, Oxford University press, 1991.

[31] K. Rice, M. Steel, T. Warnow, and S. Yooseph. Hybrid tree construction methods. manuscript, 1997.

[32] K. Rice, M.A. Steel, T. Warnow, and S. Yooseph. Better methods for solving parsimony and compatiblity. In *Proc. of the 2nd Ann. Int. Conf. on Computational Molecular Biology (RECOMB)*. ACM, 1998.

[33] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstruction phylogenetic trees. *Mol. Biol. Evol.*, 4(4):406–425, 1987.

[34] P.H. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, San Fransisco, 1973.

[35] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. of Classification*, 9:91–116, 1992.

[36] K. Strimmer and A. von Haeseler. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, 13(7):964–969, 1996.

[37] D.L. Swofford, G.J. Olsen, P.J. Wadell, and D.M. Hillis. Phylogenetic Inference. In D.M. Hillis, C. Moritz, and B.K. Mable, editors. *Molecular systematics (2nd edition)* 407–514. Sunderland, USA, 1996.

[38] S. Willson. Measuring inconsistency in phylogenetic trees. *J. Theoret. Biol*, 190:15–36, 1998.