Fast algorithms for constructing optimal trees from quartets

David Bryant* and Mike Steel†

Abstract

The reconstruction of phylogenetic trees from small subtrees on overlapping leaf sets is an important contemporary problem in computational biology. Here we investigate the problem of constructing optimal phylogenetic trees from a weighted set of quartets (resolved trees on four leaves). This problem is faced by a general class of phylogenetic reconstruction techniques where small subsets of the taxon set are analysed first and these small phylogenies are reassembled into a complete phylogeny. The problem of constructing a phylogeny that agrees with the maximum number, or maximum weight set, of quartets is NP-hard.

We consider two constrained versions. In the first we specify that T has bounded degree and that the splits of T must come from some given set, for example from the set of characters in the sequence data. We give a polynomial time algorithm that determines an optimal weight tree, or shows that no such tree exists. In the second we assume that the selection procedure for the small phylogenies returns at most *two* of the three possible resolved quartets for every set of four leaves. We give an $O(n^5)$ algorithm that finds an optimal weight binary tree, if it exists, with quartets exclusively from the input set.

1 Introduction.

The reconstruction of large evolutionary (phylogenetic) trees from smaller subtrees is currently receiving considerable attention in the computational biology community [6, 7, 18, 22, 23, 27, 28].

Such methods are based on the principle that small is easier, and often more reliable. Small sets of taxa (species) allow for far more intensive analysis and the application of complex models to reconstruct trees from the corresponding sequences that these taxa are being compared by. Tree criteria like maximum likelihood, which are computationally horrendous on larger trees, can be solved exactly on four-taxa trees (quartets) and there are just four possible trees to consider.

There are also biological and statistical advantages to considering only small subsets of sequences at a time. In many cases the actual data limit the number of sequences that can be analysed at one time. The number of sites that can be aligned across four sequences is generally much more than the number of sites that can be aligned across the full set of n sequences, so aligning over the complete set of sequences can result in lost information. Secondly, a recognized source of error in standard tree building methods like neighbor joining is that distantly related sequences can distort the tree shape [22]. If only small sets of sequences are considered at one time then those sets containing distantly related sequences can be down-weighted (or even given a zero weighting, as in [18], [22]).

The main difficulty with quartet based methods is the question of how best to build large trees out of small ones. The general problem—determining a phylogenetic tree that agrees with the largest number of quartets, or maximum weight set of quartets—is NP-hard, by a simple reduction from QUARTET COMPATIBILITY [26]. Exhaustive search is generally infeasible: there are $\frac{(2n-5)!}{(n-3)!2^{n-3}}$ binary trees on n leaves to choose from. When the number of sequences is limited, and the computational time is not, the exact algorithm of [6] can be used: it runs in time $O(n^4 3^n)$ on n sequences and $O(n^4)$ quartets. We turn instead to imposing biologically reasonable constraints under which the reconstruction problem becomes tractable.

In particular we give exact, polynomial time algorithms for two constrained versions of the optimal tree construction from quartets problem. In both cases the degree bound of the output tree is constrained—we will only construct trees when there is sufficient phylogenetic signal to determine moderately well resolved trees. In the first problem we add the constraint that each edge in the tree is supported by at least one binary character in an input set. In the second problem we require that the tree is constructed only from quartets in a given set, and this set contains at most two out of three of the possible quartets on each subset of four leaves. The choice of two out of three quartets could be made according to some phylogenetic ranking criterion like maximum likelihood or parsimony.

It might be asked whether all of these constraints are necessary for polynomial complexity? We prove

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

†Biomathematics Research Centre, University of Canterbury, Private Bag 4800, Christchurch, NZ. E-mail: m.steel@math.canterbury.ac.nz

that removing one of these constraints in either problem leads to NP-hardness.

In a sense the two-out-of-three quartets problem can be seen as an extension of the Q^* technique of [7]. These authors present a fast exact algorithm for the case when there is at most one quartet in the input set for each four taxa, and the output tree is constrained to have only quartets from this set. The algorithm is exact because there is always exactly one maximal tree satisfying these conditions. The Q^* method is employed by Kearney [23] to construct trees from quartets selected by an ordinal quartet method.

The alternative to exact, constrained algorithms is heuristic tree construction algorithms, and these have been produced by a number of computer scientists, biologists and mathematicians. The heuristics of Sattath and Tversky [25], Fitch [21], Colonius and Schulze [12], and Bandelt and Dress [1] combine clustering procedures with a pairwise similarity, or neighbourliness, scores derived from the quartet sets. A novel variation on the scoring approach is described by Ben-Dor *et al.* [6]. Instead of constructing a similarity score then clustering, they embed the n taxa as points in $\Re^n$ using semi-definite programming, and then apply a nearest neighbour clustering method.

A related, but more sophisticated approach, is the disc-covering method of [22]. Trees are first constructed on overlapping sets of closely related sequences, and these trees are then combined using graph-theoretic techniques until an entire phylogeny has been constructed. This technique is an extension of an earlier 'short quartets' method [18] which sought to reconstruct trees from quartets of closely related sequences, but which sometimes would not return any tree. Both these methods were shown to reconstruct trees from short sequences when the sites evolve under simple Markov models.

The tree building, or 'puzzling', part of the Quartet Puzzling method of Strimmer and von Haeseler [27] works by ordering the taxon set arbitrarily, constructing a tree on the first four taxa, and then adding new taxa one at a time, selecting the edge to attach each taxa as the one that gives optimum quartet score. The same approach is used by Willson [28] to optimise according to a different, but related, criteria. These procedures can be seen as an analogues of the Wagner tree method [19].

1.1 Preliminaries. An **unrooted phylogenetic tree** is an acyclic connected graph with no vertices of degree two and all leaves (degree one vertices) labelled uniquely from some leaf set L. A phylogenetic tree is **binary** or **resolved** if all internal vertices have degree three.

A **rooted phylogenetic tree** is defined in the same way, except that one internal vertex, which may have degree two, is distinguished and called the **root**. Given any two vertices u, v in a rooted phylogenetic tree, if the path from u to the root passes through v then we say that u is a **descendent** of v. The descendents of a vertex v that are also adjacent to v are called the **children** of v.

Removing an internal edge e from an unrooted phylogenetic tree T divides the tree into two connected components and induces a **split** or **bipartition** of the leaf set of T. This split is called the split associated with e, and the set of all such splits in a tree is denoted $splits(T)$. A split with two blocks A and B is denoted $A|B$. If $|A| = 1$ or $|B| = 1$ then the split is **trivial**—trivial splits correspond to external edges in a tree. A set of splits $\mathcal{S}$ is **compatible** if $\mathcal{S} \subseteq splits(T)$ for some tree T.

The rooted analogue of a split is a **cluster**. Given a vertex v in a rooted tree the set of leaves that are descendents of v is called the cluster corresponding to v. The set of all clusters associated to vertices in a rooted tree T is denoted $clus(T)$.

A **quartet** is a resolved phylogenetic tree on four leaves. There are three possible quartets on a given set of four leaves $\{a, b, c, d\}$. We use $ab|cd$ to denote the quartet where a and b are separated from c and d by the internal edge. A phylogenetic tree T **agrees** with a quartet $ab|cd$ if a, b, c, d are all leaves of T and the path from a to b does not share any vertices with the path from c to d. Let $q(T)$ be the set of quartets that T agrees with.

The quartet set of a split $A|B$ is defined by

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

Conversely, if Q is a set of quartets on leaf set L put

$$S(Q) = \{A|B : q(A|B) \subseteq Q,\ A|B \text{ a split of } L\}.$$

Then

$$q(T) = \bigcup_{A|B \in splits(T)} q(A|B)$$

and $q(T) \subseteq Q$ if and only if $splits(T) \subseteq S(Q)$.

In the problems we consider, each quartet $ab|cd$ will be assigned a weight $w(ab|cd)$. The weight of a tree T is equal to the sum of the weights of its quartets; thus

$$w(T) = \sum_{ab|cd \in q(T)} w(ab|cd).$$

The general quartet to tree reconstruction problem is:

TREE WITH MAXIMUM QUARTET WEIGHT
INSTANCE: A weighting w for the quartets on a leaf set L.
PROBLEM: Determine a phylogenetic tree T for which $w(T)$ is maximized.

This problem is NP-hard, even when weights are all 0 or 1, by a straight-forward reduction from QUARTET COMPATIBILITY [26]. We impose two constraints on the problem: that the splits of the output tree come from a given set, and the maximum degree of the output tree is bounded. The first problem we address is:

HUNTING FOR TREES WITH WEIGHTED QUARTETS
INSTANCE: Weighting w for the quartets on a leaf set L. Set $\mathcal{S}$ of splits of L. Number λ.
PARAMETER: Degree bound d.
QUESTION: Is there a tree T with degree bounded by d and $splits(T) \subseteq \mathcal{S}$ such that $w(T) \geq \lambda$?

The time complexity of the algorithm we present to solve this problem is $O(nk + n^4K + n^2K^2 + ndK^{d-1})$, where k is the number of splits in $\mathcal{S}$, K is the number of distinct splits in $\mathcal{S}$ and $n = |L|$.

The problem is clearly NP-hard without the constraint that $splits(T) \subseteq \mathcal{S}$; we prove that it also NP-hard in the case when the constraint $splits(T) \subseteq \mathcal{S}$ remains and the degree bound restraint is removed (Theorem 3.6).

We also discuss a related problem, one that turns out to be a special case of HUNTING FOR TREES WITH WEIGHTED QUARTETS. In order to allow a degree of uncertainty in the quartet selection procedure we permit a quartet set Q to contain at most *two* of $ab|cd, ac|bd, ad|bc$ for every set of four leaves a, b, c, d. We then solve the following problem:

2/3 QUARTET PUZZLING
INSTANCE: Set Q of weighted quartets on a leaf set L such that for every four leaves a, b, c, d of L at most two of $ab|cd, ac|bd, ad|bc$ are in Q. Number λ.
QUESTION: Is there a binary tree T such that $q(T) \subset Q$ and $w(T) \geq \lambda$?

Our algorithm for this problem takes $O(n^5)$ time, which is quite respectable, given that the input size is $O(n^4)$. In the case that there is more than one optimal tree, or possibly exponentially many optimal trees, we can construct the strict consensus or majority rule consensus tree of the optimal trees, all in polynomial time.

2 Optimal bounded degree trees in splits.

Compatibility methods are perhaps the most widely understood, and least practiced, methods for constructing phylogenies from characters. Though conceptually simple—search for the largest set of compatible characters in the input set—they are computationally unattractive. Day and Sankoff [13] proved that the MAXIMUM COMPATIBLE SUBSET OF CHARACTERS is polynomially equivalent to MAX CLIQUE, and so inherits the depressing complexity attributes of MAX CLIQUE like W[1]-hardness [16] and non-approximability [5].

The equivalence with MAX CLIQUE might be seen to make the existence of a useful parameterization for MAXIMUM COMPATIBLE SUBSET highly unlikely. However it was shown in [10] that a bound on a natural parameter—the maximum degree of the output tree—leads to a useful and nontrivial restriction of MAXIMUM COMPATIBLE SUBSET. Formally,

HUNTING FOR TREES IN SPLITS
INSTANCE: Set $\mathcal{S}$ of weighted splits on L. Number λ
PARAMETER: Degree bound d.
QUESTION: Is there a tree T with degree bound d and splits in $\mathcal{S}$ such that the sum of the weights of splits in T exceeds λ?

This problem, together with the associated optimization problem, can be solved in $O(nk + ndK^{d-1})$ time, where n is the number of leaves, k is the number of splits in $\mathcal{S}$ and K is the number of distinct splits in $\mathcal{S}$. Unfortunately, determining the minimum d for which there exists a tree T with degree bound d and splits in $\mathcal{S}$ is NP-hard [11], and the HUNTING FOR TREES problem is itself W[1]-hard, due to a reduction by Fellows [20].

In the HUNTING FOR TREES WITH WEIGHTED QUARTETS problem the score of the tree is not determined by the sum of the weights of its splits, but by the sum of the weights of its quartets.

2.1 Decomposition table. The first step in either HUNTING FOR TREES method is to convert the problem from one involving unrooted trees and splits to one involving rooted trees and clusters [10], or in the holistic terminology of [17], from the projective case to the affine case. Fix a leaf $x \in L$. We define a map ψ_x from unrooted trees on L to rooted trees on $L - \{x\}$, and from splits of L to clusters of $L - \{x\}$. Given an unrooted tree T let $\psi_x(T)$ be the tree on leaf set $L - \{x\}$ obtained by rooting T at the internal vertex closest to x and then deleting the leaf x and its adjacent edge. Given a split $A|B$ of L let $\psi_x(A|B)$ be the cluster of

$L - \{x\}$ given by

$$\psi_x(A|B) = \begin{cases} A & \text{if } x \in B \\ B & \text{if } x \in A \end{cases}$$

We see that the function ψ_x is clearly invertible and that a set of splits S equals the splits of some unrooted tree T with degree bound d if and only if $\psi_x(S) = \{\psi_x(A|B) : A|B \in S\}$ equals the set of clusters of a rooted tree $\psi_x(T)$ with degree bound d and root with degree at most $d-1$.

The problem has now become

HUNTING FOR ROOTED TREES IN WEIGHTED QUARTETS.
INSTANCE: Set $\mathcal{C}$ of clusters of $L - \{x\}$. Weighting function w for the quartets on L. Number λ.
PARAMETER: Degree bound d.
QUESTION: Is there a rooted tree T with degree bound d, root with degree at most $d-1$, and $clust(T) \subseteq \mathcal{C}$ such that $w(\psi_x^{-1}(T)) \geq \lambda$.

The fundamental data structure, and basis of recursion, is a decomposition table D. First sort the clusters in $\mathcal{C}$ into lexicographic order and remove duplicates, obtaining a list of clusters $\{C_1, C_2, \ldots, C_K\}$ where $C_i \subset C_j$ implies $i < j$. For each $i = 1, \ldots, K$ we construct a list of tuples $D[i]$, with $\{p_1, p_2, \ldots, p_r\} \in D[i]$ if and only if $2 \leq r \leq d-1$ and C_i is the *disjoint* union of $C_{p_1}, C_{p_2}, \ldots, C_{p_r}$.

There are $O(K^{d-1})$ tuples with fewer than d clusters. Checking one tuple, and inserting it into the appropriate row of D takes $O(n + \log K)$ time, which is $O(n)$ time since $K < 2^n$. The sorting can be performed in $O(nk)$ time using radix sort, making a total of $O(nk + nK^{d-1})$ time for the construction of D.

2.2 Counting trees. Each row $D[i]$ in the decomposition table encodes a set of rooted trees with leaf set C_i, which we denote $\mathcal{T}(D, i)$. The set $\mathcal{T}(D, i)$ is defined recursively: if $C_i = \{a\}$ for some leaf a then $\mathcal{T}(D, i)$ contains the single vertex tree with leaf a; if $|C_i| \geq 2$ and $D[i] = \emptyset$ then $\mathcal{T}(D, i) = \emptyset$. Otherwise, $\mathcal{T}(D, i)$ is the set of all possible trees that can be formed by choosing a tuple $\{p_1, \ldots, p_r\}$ in $D[i]$, choosing subtrees $T_j \in \mathcal{T}(D, j)$ for each $j = 1, \ldots, r$, and attaching the roots of these subtrees to a new vertex that becomes the root of a rooted tree with leaf set C_i. Since there may be tuples $\{p_1, \ldots, p_r\}$ in $D[i]$ with $\mathcal{T}(D, p_j) = \emptyset$ for some j, we can have $D[i] \neq \emptyset$ but $\mathcal{T}(D, i) = \emptyset$.

It can be shown that for all i the set $\mathcal{T}(D, i)$ equals the set of trees with leaf set C_i, degree bound d, root with degree $d-1$, and clusters in $\mathcal{C}$ [10]. Hence $\mathcal{T}(D, i)$ can be exponential in size (see Theorem 3.2). It is still possible to calculate $|\mathcal{T}(D, i)|$ and extract indexed trees from $\mathcal{T}(D, i)$ in polynomial time.

Define $m[i], i = 1, \ldots, K$ recursively by $m[i] = 1$ if $|C_i| = 1$ and

$$m[i] = \sum_{\{p_1, p_2, \ldots, p_r\} \in D[i]} m[p_1] \times m[p_2] \times \cdots \times m[p_r]$$

when $|C_i| > 1$. Note that $D[i] = \emptyset$ implies $m[i] = 0$. Then $m[i] = |\mathcal{T}(D, i)|$ (see [10]). We remove clusters C_i such $m[i] = 0$ as well as tuples that refer to them. The values $m[i]$ can be calculated in $O(dK^{d-1})$ time.

To solve HUNTING FOR ROOTED TREES WITH WEIGHTED QUARTETS we need to optimize over the set $\mathcal{T}(D, K)$ and so we need only consider the clusters C_i such that $C_i \in clus(T)$ for some $T \in \mathcal{T}(D, K)$. We count the number of trees $T \in \mathcal{T}(D, K)$ containing each cluster C_i.

ALGORITHM 2.1. COUNTINCLUDETREE(D,m)

```
1. m_incl[i] ← 0 all i < K, m_incl[K] ← m[K]
2. For i from K down to 1 do
3.  For (p_1, p_2, ..., p_r) in D[i] do
4.   For j from 1 to r do
5.    Add m[p_1]m[p_2]...m[p_r]m_incl[i]/m[i] to
      m_incl[p_j].
6.   End(for)
7.  End(for)
8. End(for)
end.
```

Then $m_{incl}[i] = |\{T \in \mathcal{T}(D, K) : C_i \in clus(T)\}|$. We remove any clusters C_i such that $m_{incl}[i] = 0$, as well as any tuples containing them.

2.3 Recovering trees from a decomposition table. Now that all clusters C_i with $m[i] = 0$ have been removed, we extract arbitrary trees in $\mathcal{T}(D, K)$ using simple recursion. The procedure could be easily extended to index trees contained in D with an integer from 1 to $m[K]$ and extract the tree corresponding to a particular index.

We can extract consensus information from the set of trees contained in a decomposition table without having to extract the entire, possibly exponential size, set. The number of trees in the decomposition table is given by $m[K]$, and the number of trees containing a particular cluster C_i is given by $m_{incl}[i]$. Hence we can construct the strict consensus tree and majority rule tree [24] of the trees stored in a decomposition table in polynomial time. Maximum agreement subtrees of the trees in a decomposition table can also be constructed in polynomial time [11].

2.4 Quartet optimization. We need to extend the definition of $\psi_x^{-1}(T)$ to the case when T contains only a subset of the leaves in $L-\{x\}$. Suppose that T is a rooted tree with leaf set C_i. Let $\psi_x^{-1}(T,L)$ be the tree obtained from T by attaching a new vertex v to the root of T, attaching leaves $\{y : y \in L - C_i\}$ to the vertex v, and unrooting the tree. Note that if T has leaf set $L-\{x\}$ then $\psi_x^{-1}(T,L) = \psi_x^{-1}(T)$.

For each $T \in \mathcal{T}(D,i)$ define

$$Q_i(T) = \{ab|cd \in q(\psi_x^{-1}(T,L)) : |\{a,b,c,d\} \cap C_i| \geq 3\}.$$

Put

$$M[i] = \max\left\{ \sum_{ab|cd \in Q_i(T)} w(ab|cd) : T \in \mathcal{T}(D,i) \right\}.$$

Observe that if $C_K = L-\{x\}$ then $M[K] = \max\{w(T) : T \in \mathcal{T}(D,K)\}$. Note that there is not always a unique maximum tree giving weight $M[K]$.

The algorithm requires some pre-processing. Given subsets

$$A, B \in \mathcal{C} \cup \{L - X : X \in \mathcal{C}\}$$

and leaves $u, v \in L$ calculate $W(A;u,v) = \sum_{\{a,a'\} \subseteq A} w(aa'|uv)$ and then

$$W(A;B) = \sum_{a,a' \in A; b,b' \in B} w(aa'|bb') = \sum_{b,b' \in B} W(A;b,b').$$

These calculations take at most $O(Kn^4 + K^2n^2)$ time.

Given a tuple $\{p_1, p_2, \ldots, p_r\}$ in $D[i]$ define

$$W(\{p_1,p_2,\ldots,p_r\}) = \sum_{\alpha \in \{p_1,\ldots,p_r\}} W(C_\alpha; L - C_\alpha) - \sum_{\{\alpha,\beta\} \subseteq \{p_1,\ldots,p_r\}} W(C_\alpha, C_\beta)$$

Each evaluation of $W(\{p_1,p_2,\ldots,p_r\})$ takes $O(d^2)$ time, provided that the values $W(A;B)$ have been stored for all $A, B \in \mathcal{C} \cup \{L-X : X \in \mathcal{C}\}$. The function $W(\{p_1,p_2,\ldots,p_r\})$ enables the calculation of $Q_i(T)$ in terms of the maximal proper subtrees of T.

LEMMA 2.1. *Consider $C_i : |C_i| \geq 3$ and $T \in \mathcal{T}(D,i)$. Let $T_1, T_2, \ldots, T_r$ be the maximal proper subtrees of T. There is a tuple $\{p_1,p_2,\ldots,p_r\} \in D[i]$ such that subtree T_j has leaf set C_{p_j} for all $j = 1,\ldots,r$. Furthermore*

$$\sum_{q \in Q_i(T)} w(q) = \sum_{j=1}^{r} \sum_{q \in Q_{p_j}(T_j)} w(q) + W(\{p_1,p_2,\ldots,p_r\}).$$

Proof. The existence of the tuple $\{p_1,p_2,\ldots,p_r\} \in D[i]$ follows from $clus(T) \subseteq \mathcal{C}$.

We partition the set $Q_i(T)$ into three blocks:
(i) Those quartets $ab|cd \in Q_i(T)$ for which $|\{a,b,c,d\} \cap C_{p_j}| \geq 3$ for a unique $j \in 1,2,\ldots,r$ (in which case $ab|cd \in Q_j(T_j)$).
(ii) Those quartets $ab|cd \in Q_i(T)$ for which $|\{a,b,c,d\} \cap C_i| = 3$ and there is cluster C_{p_j} such that $\{a,b,c,d\} \cap C_{p_j}$ equals $\{a,b\}$ or $\{c,d\}$.
(iii) Those quartets $ab|cd \in Q_i(T)$ for which there are clusters C_{p_k}, C_{p_l} such that $a,b \in C_{p_k}$ and $c,d \in C_{p_l}$.
We see that $\sum_{j=1}^{r} \sum_{q \in Q_{p_j}(T_j)} w(q)$ is the sum of $w(q)$ of all quartets in the first block, while $W(\{p_1,p_2,\ldots,p_r\})$ is the sum of $w(q)$ over all quartets in the second and third blocks. □

We can now state the recursion on which our optimization algorithm is based. It is proved from Lemma 2.1 by induction [proof omitted].

THEOREM 2.1. *If $|C_i| \leq 2$ then $M[i] = 0$, otherwise $M[i]$ equals the maximum of*

$$\sum_{j=1}^{r} M[p_j] + W(\{p_1,\ldots,p_r\})$$

over all $\{p_1,p_2,\ldots,p_r\} \in D[i]$.

We are now ready for the main optimisation algorithm, MAXQWEIGHTTREE. The algorithm returns a new decomposition table D^* containing exactly all of the optimal weight trees.

ALGORITHM 2.2. MAXQWEIGHTTREE(C,D,m)

```
1. For i from 1 to K do
2.  If |C_i| ≤ 2 then
3.   M[i] ← 0
4.   D*[i] ← ∅
5.  else
6.   Let D*[i] be the set of tuples
     {p_1,...,p_r} ∈ D[i] that maximize

                  Σ_{j=1}^{r} M[p_j] + W({p_1,...,p_r})

     and let M[i] be this maximum value.
7.  End(If-else)
8. End(For)
End.
```

The time taken by the algorithm is dominated by the time taken to evaluate $W(\{p_1,p_2,\ldots,p_r\})$, that is, $O(d^2)$ time for every tuple in every row of $D[i]$.

This makes $O(d^2K^d + n^4K + n^2K^2)$ for the algorithm, once preprocessing is included. Note that, in practice, it would be more efficient to calculate and store the values $W(A;B)$ on the fly, since not every possible value is used. We separate the calculation out here to aid presentation and to encourage the development of a more efficient algorithm for computing these values.

THEOREM 2.2. *HUNTING FOR TREES WITH WEIGHTED QUARTETS can be solved in $O(nk + ndK^{d-1} + n^4K + n^2K^2)$ time, where k is the number of splits, K is the number of distinct splits, n is the number of taxa, and d is the degree bound (on unrooted trees).*

2.5 Extension to weighted fans. When the degree bound d in the HUNTING FOR TREES WITH WEIGHTED QUARTETS problem is greater than 3 we can obtain non-binary trees, and these trees can contain both resolved and unresolved four-taxa trees (called **fans**). So far we have only considered weighting quartets, with fans receiving an implied zero weight. The algorithms for HUNTING FOR TREES WITH WEIGHTED QUARTETS can, however, be extended to incorporate a weight on unresolved quartets, though the time complexity increases to $O(nk + nd^3K^{d-1} + n^4K^2 + n^2K^4)$.

3 2/3 Quartet Puzzling.

As we mentioned earlier, there are two main steps in quartet based phylogeny reconstruction. We first analyse sequences four at a time and construct quartet trees, and then assemble these quartets into a complete phylogenetic tree. One problem is that there can be a degree of uncertainty in the first step: with many methods there might be more than one optimal quartet tree, and specifying that at most one quartet is chosen will result in loss of information.

Strimmer and von Haessler [27] address this problem by randomly selecting between optimal quartets on the same leaf set. Others [1, 21] incorporate the uncertainty into their scoring schemes. Our approach is to allow the quartet selection procedure to choose not one, but two quartets for every set of four leaves, and we will use this larger set of quartets to construct the tree.

2/3 QUARTET PUZZLING
INSTANCE: Set Q of weighted quartets on leaf set L such that for every $a, b, c, d \in L$ at most two of $ab|cd, ac|bd, ad|bc$ are in Q. Number λ.
QUESTION: Is there a binary tree T with leaf set L such that $q(T) \subseteq Q$ and $w(T) \geq \lambda$.

We give an $O(n^5)$ time algorithm for constructing the optimal weight tree, if one exists.

3.1 Splits in quartet sets A set of splits $\mathcal{S}$ of L is *compatible* if $\mathcal{S} \subseteq splits(T)$ for some phylogenetic tree T. Compatible sets of splits can be characterized by the property that

$$q(S) = \bigcup_{A|B \in \mathcal{S}} q(A|B)$$

contains at most one of $ab|cd, ac|bd, ad|bc$ for every four leaves $a, b, c, d \in L$.

In their work on split decomposition of metrics, Bandelt and Dress [3] generalised the concept of compatible splits to weakly compatible splits. A set of splits is **weakly compatible** if for every three splits $A_1|B_1, A_2|B_2, A_3|B_3$, at least one of the intersections $A_1 \cap A_2 \cap A_3$, $A_1 \cap B_2 \cap B_3$, $B_1 \cap A_2 \cap B_3$, $B_1 \cap B_2 \cap A_3$ is empty. Whereas the size of a set of compatible splits is at most $O(n)$ the size of a set of weakly compatible splits is size at most $O(n^2)$. Furthermore

THEOREM 3.1. *([3, 4]). A set of splits $\mathcal{S}$ of L is weakly compatible if and only if $q(\mathcal{S})$ contains at most two of $ab|cd, ac|bd, ad|bc$ for every set of four leaves $a, b, c, d \in L$.*

Thus if Q is a set of quartets such that for every $a, b, c, d \in L$ at most two of $ab|cd, ac|bd, ad|bc$ are in Q then the set of splits $S(Q)$ is weakly compatible. An $O(n^5)$ time algorithm for constructing $S(Q)$ from Q is given in [8]. The number K of splits in $S(Q)$ is $O(n^2)$. By Theorem 2.2 we can find a binary tree with splits in $S(Q)$ and maximum quartet weight in $O(nk + nK^2 + n^4K + n^2K^2) = O(n^6)$ time (or show that no such tree exists). We will exploit the structure of $S(Q)$ and reduce this time complexity to $O(n^5)$.

Note that the number of binary trees with splits in a given set of weakly compatible splits can still be exponential in n. When $\mathcal{S}$ is a set of $\binom{n}{2}$ weakly compatible splits (the maximum number possible), the number of binary trees with splits in $\mathcal{S}$ can be calculated exactly.

THEOREM 3.2. *Let $\mathcal{S}$ be a weakly compatible set of $\binom{n}{2}$ splits of L, where $n = |L|$. The number of binary trees T with $splits(T) \subset \mathcal{S}$ equals $\frac{1}{n-1}\binom{2n-4}{n-2}$, a Catalan number.*

[Proof omitted].

3.2 Structural properties of weakly compatible splits. In order to improve the complexity of the

2/3 QUARTET PUZZLING algorithm we prove some structural properties of sets of weakly compatible splits. Once again we explore a rooted or 'affine' analogue to sets of splits: weak hierarchies. A **weak hierarchy** is a set of clusters such that for any three clusters A, B, C we have

$$A \cap B \cap C \in \{A \cap B, A \cap C, B \cap C\}.$$

A set of clusters $\mathcal{C}$ forms a **chain** if $A \subseteq B$ or $B \subseteq A$ for all $A, B \in \mathcal{C}$, and a set of splits $\mathcal{S}$ forms a chain if $\psi_x(\mathcal{S})$ is a chain for some x.

THEOREM 3.3. *Any set $\mathcal{S}$ of weakly compatible set splits L is the union of $n-1$ chains.*

Proof. Fix $w \in L$ and put $\mathcal{H} = \psi_w(\mathcal{S})$, which is a weak hierarchy [3]. For each pair of leaves a, b

$$< a, b >= \bigcap_{C \in \mathcal{H}: a,b \in A} C = \{c : ab|cw \notin q(\mathcal{S})\}.$$

Then $\mathcal{H} \subseteq \{< a, b >: a, b \in L - \{x\}\}$ with equality if and only if $\mathcal{H}$ is closed under intersections [2]. For each $A \in \mathcal{H}$ we choose one pair $\pi(A) = \{x, y\}$ such that $< x, y >= A$.

The set $\mathcal{H}$ can be partially ordered with respect to set inclusion. We say that a subset $\mathcal{A} \subseteq \mathcal{H}$ is an anti-chain of $\mathcal{H}$ $A_1, A_2 \in \mathcal{A}$ and $A_1 \neq A_2$ implies $A_1 \not\subseteq A_2$. Let $\mathcal{A}$ be a maximum size anti-chain of $\mathcal{H}$.

Given any $x \in L - \{w\}$ we can show that if there are distinct $A_1, A_2, A_3 \in \mathcal{A}$ such that $x \in \pi(A_1) \cap \pi(A_2) \cap \pi(A_3)$ then $A_1|(L - A_1)$, $A_2|(L - A_2)$ and $A_3|(L - A_3)$ are not weakly compatible. Hence for each x there are at most two clusters $A, A' \in \mathcal{A}$ such that $x \in \pi(A)$ and $x \in \pi(A')$. Furthermore if there is a cluster A such that $A = \{x\}$ then there is only one $A' \in \mathcal{A}$ such that $x \in \pi(A')$. It follows that $\mathcal{A}$ contains at most $|L - \{w\}| = n - 1$ members. By Dilworths theorem [15], $\mathcal{H}$ can be covered by $n - 1$ chains. □

Note that the number $n - 1$ of chains is tight since $\psi_x(\mathcal{S})$ of a maximum set of cyclic splits $\mathcal{S}$ [3] contains $n - 1$ clusters of size 2. The proof of Theorem 3.3 gives an efficient way to construct a chain decomposition into $O(n)$ chains.

LEMMA 3.1. *Given a set of splits $\mathcal{S}$ together with the set of quartets $Q_w = \{ab|cw : ab|cw \in q(\mathcal{S})$ for some $w \in L$ we decompose $\mathcal{S}$ into $O(n)$ chains in $O(n^3)$ time.*

Proof. We first construct and sort $\mathcal{H} = \psi_w(\mathcal{S})$ in $O(n^3)$ time using radix sort. For each pair of leaves $a, b \in L - \{w\}$ construct $< a, b >= \{c : ab|cw \notin Q_w\}$ and determine if $< a, b >$ is in $\mathcal{H}$. In this way, assign a unique pair $\pi(A)$ to each $A \in \mathcal{H}$.

For each $x \in L - \{w\}$ consider the set $\mathcal{H}_x = \{< x, y >: \{x, y\} = \pi(< x, y >)$. By the proof of Theorem 3.3 $\mathcal{H}_x$ contains no anti-chains of size three, so can be decomposed into two chains. This decomposition can be performed in $O(n^2)$ time by constructing an incomparability graph for $\mathcal{H}_x$ (noting that $< x, y > \subseteq < x, y' >$ if and only if $xy'|yw \notin Q_w$) and 2-colouring. Repeating this process for all $x \in L - \{w\}$ takes $O(n^3)$ time. □

The next efficiency gain results from determining a bound on the size of the decomposition table for $\mathcal{C}$. First we bound the size of a special kind of weak hierarchy.

LEMMA 3.2. *Suppose that $\mathcal{H}$ is a weak hierarchy on set L such that $A \in \mathcal{H}$ implies $(L - A) \in \mathcal{H}$. Then the number of clusters in $\mathcal{H}$ is at most $O(|L|)$.*

Proof. Fix $u \in L$ and let $\mathcal{H}_u$ be the set of clusters in $\mathcal{H}$ containing u. Let A, B, C be clusters in $\mathcal{H}_u$ that all have size k (where $1 \leq k < n$). By applying the weak hierarchies triplewise property to clusters $A, B, C, L - A, L - B, L - C$ three at a time we can infer that two out of A, B, C are equal. Therefore there can be at most two distinct clusters of size k in $\mathcal{H}_u$, so $\mathcal{H}_u$ contains at most $2n$ clusters. For any cluster $A \in \mathcal{H}$ either $A \in \mathcal{H}_u$ or $(L - A) \in \mathcal{H}_u$ so we get $|\mathcal{H}| \leq 2|H_u| \leq 4n$. □

THEOREM 3.4. *Given any cluster A in a weak hierarchy $\mathcal{H}$ there are at most $O(|A|)$ pairs of clusters B, C such that $B \cap C = \emptyset$ and $B \cup C = A$.*

Proof. Let $\mathcal{C}$ be the set of clusters $B \in \mathcal{H}$ such that $B \subset A$ and $(A - B) \in \mathcal{H}$. Then $\mathcal{C}$ is a weak hierarchy on ground set A with the property that $B \in \mathcal{C}$ implies $(A - B) \in \mathcal{C}$. Apply Lemma 3.2 to $\mathcal{C}$ to obtain $|\mathcal{C}| \in O(|A|)$. □

3.3 An $O(n^5)$ algorithm for 2/3 QUARTET PUZZLING. The $O(n^5)$ algorithm follows similar steps as the HUNTING FOR TREES WITH WEIGHTED QUARTETS algorithm. First we construct the set of splits, $S(Q)$, using the $O(n^5)$ algorithm of [8]. This algorithm can also remove quartets from Q that are not in $q(S(Q))$. set $S(Q)$ is weakly compatible, so contains $O(n^2)$ splits. Choose arbitrary $w \in L$ and construct the weak hierarchy $\mathcal{H} = \psi_w(S(Q))$. We then construct the decomposition table, and calculate the tables m and m_{incl}, as in §2.2. This all takes at most $O(n^5)$ time.

We now use Lemma 3.1 to construct $O(n)$ chains $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ with union containing $\mathcal{H}$. Order the clusters in each chain by inclusion. For each chain $\mathcal{C}_i$

and every pair of leaves u, v we can calculate all the values $\{W(C_j; u, v) : C_j \in \mathcal{C}_i\}$ in $O(n^2)$ time. Hence it takes $O(n^5)$ time to calculate $W(A; u, v)$ for all $A \in \mathcal{H}$ and $u, v \in L$. Once these values are calculated it takes a further $O(n^2)$ to evaluate each case of

$$W(\{p_1, p_2\}) = \sum_{u,v \in C_{p_2}} W(C_{p_1}; u, v) + \sum_{u \in C_{p_2}, v \in L - C_i} W(C_{p_1}; u, v) + \sum_{u \in C_{p_1}, v \in L - C_i} W(C_{p_2}; u, v).$$

By Theorem 3.4 the number of these tuples is $O(n|\mathcal{C}|) = O(n^3)$.

THEOREM 3.5. *2/3 QUARTET PUZZLING can be solved in $O(n^5)$ time.*

3.4 Dropping the degree bound Having found a fast algorithm for 2/3 QUARTET PUZZLING with a degree bound constraint, it might be asked whether a fast algorithm exists when we do not apply a degree constraint. The answer, in general, is no—assuming that $NP \neq P$. Determining the tree with splits in $\mathcal{S}$ and optimal quartet weight is NP-hard, even when $\mathcal{S}$ is weakly compatible.

THEOREM 3.6. *Determining the tree T with leaf set L and $q(T) \subset Q$ that maximizes $w(T)$ is NP-hard when Q contains at most two of $ab|cd$, $ac|bd$, $ad|bc$ for each $a, b, c, d \in L$.*

Proof. In [11] it was proven that determining the maximum compatible subset of a set of a weak compatible splits is NP-hard. Let $\mathcal{S}$ be an arbitrary set of weakly compatible splits and put $Q = \cup_{A|B \in \mathcal{S}} q(A|B)$. For every split $A_i|B_i \in \mathcal{S}$ there exists a quartet $a_i a_i'|b_i b_i' \in q(A_i|B_i)$ such that $a_i a_i'|b_i b_i' \notin q(A_j|B_j)$ for all other splits $A_j|B_j \in \mathcal{S}$ [3, 4]. Choose one of these quartets for each split and give it weight one. Give all other quartets weight zero. Then for any tree T with leaf set L and $q(T)$ we have

$$\sum_{ab|cd \in q(T)} w(ab|cd) = |splits(T)|. \quad \square$$

3.5 Probabilistic analysis. The 2/3 QUARTET PUZZLING method is clearly consistent: if it is given the set of quartets of some binary tree T then it will return T. The method has the capacity to handle more quartets than the Q^* method of Berry and Gascuel [7], so like the Q^* method the quartet hunting method has a polynomial convergence rate under simple models of site substitution. We show that, like the Q^* method, the 2/3 QUARTET PUZZLING method is highly unlikely to return a fully resolved tree when given random data.

Suppose for each set of four leaves in L two out of the three possible quartets are selected randomly and uniformly (i.e. with equal probability), and these selections are made independently between quartets. Let $P(n)$ be the probability that the resulting set Q of quartets contains $q(T)$ for some binary tree with n leaves.

THEOREM 3.7.

$$P(n) \leq \left(\frac{2}{3}\right)^{\binom{n}{4}} b(n)$$

where $b(n) = (2n-5)!! = (2n-5)!/(2^{n-3}(n-3)!)$. Consequently

$$\lim_{n \to \infty} P(n) = 0.$$

Proof. We calculate the probability that Q contains the quartets of a particular binary tree T on n leaves and then sum over all possible binary trees on n leaves to obtain the upper bound. $\square$

We note that $\left(\frac{2}{3}\right)^{\binom{n}{4}} b(n)$ goes to zero very quickly. When $n = 10$ we have $P(n) < 0.3 \times 10^{-30}$. When $n = 20$ we have $P(n) < 0.16 \times 10^{-832}$.

Acknowledgements

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. The second author thanks the New Zealand Marsden Fund. We also thank C. Semple for his careful proof reading.

References

[1] H-J. Bandelt and A. Dress, *Reconstructing the shape of a tree from observed dissimilarity data*, Adv. Appl. Math., 7 (1986), pp. 309–343.

[2] H-J. Bandelt and A. Dress, *Weak hierarchies associated with similarity measures–an additive clustering technique*, Bull. Math. Biol, 51(1) (1989), pp. 133–166.

[3] H-J. Bandelt and A. Dress, *A canonical decomposition theory for metrics on a finite set*, Adv. Math., 92 (1992), pp. 47–105.

[4] H-J. Bandelt and A. Dress, *A relational approach to split decomposition*, tech. report, Universität Bielefeld, 1994.

[5] M. Bellare, O. Goldreich, and M. Sudan, *Free bits, PCPs and non-approximability - towards tight results*, SIAM J. Comput. 27(3) (1998), pp. 804–915.

[6] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg, *From four-taxon trees to phylogenies: the case of mammalian evolution*, RECOMB (1998), pp. 9–19.

[7] V. Berry and O. Gascuel, *Inferring evolutionary trees with strong combinatorial evidence*, COCOON (1997), pp. 111–123.

[8] V. Berry and D. Bryant, *Faster reliable phylogenetic analysis*, Submitted to *RECOMB*, 1999.

[9] D. Bryant and M. Steel, *Extension operations on sets of leaf-labelled trees*, Adv. Appl. Math., 16 (1995), pp. 425–453.

[10] D. Bryant, *Hunting for trees in binary character sets: efficient algorithms for extraction, enumeration, and optimization*, J. Comput. Biol., 3(2) (1996), pp. 275–288.

[11] D. Bryant, *Building trees, hunting for trees and comparing trees*, Ph.D. thesis, University of Canterbury, NZ. 1997.

[12] N. Colonius and H. H. Schulze, *Tree structures for proximity data*, British J. Math. Statist. Psych., 34 (1981), pp. 167–180.

[13] W. H. E. Day and D. Sankoff, *Computational complexity of inferring phylogenies by compatibility*, Sys. Zool., 35(2) (1986), pp. 224–229.

[14] M. C. H. Dekker, *Reconstruction methods for derivation trees*, Master's thesis, Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, 1986.

[15] R. P. Dilworth, *A decomposition theorem for partially ordered sets*, Ann. Math., 29 (1950), pp. 1–10.

[16] R. Downey and M. Fellows, *Fixed-Parameter tractability and completeness II. On completeness for W*[1], Theoret. Comput. Sci., 141(1-2) (1996), pp. 109–131.

[17] A. Dress, *Towards a theory of holistic clustering*, in Mathematical hierarchies and biology, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 37 (1997), pp. 271–289.

[18] P. L. Erdös, M. Steel, L. Székeley, and T. Warnow, *Inferring big trees from short sequences*, Automata, Languages, and Programming (1997), pp. 827–837.

[19] J. Farris, *Estimating phylogenetic trees from distance matrices*, Amer. Naturalist, 106(951) (1972), pp. 645–667.

[20] M. Fellows, Private communication. 1996.

[21] W. M. Fitch, *A non-sequential method of constructing trees and hierarchical classifications*, J. Mol. Evol. 18 (1981), pp. 30–37.

[22] D. Huson, S. Nettles, T. Parida, T. Warnow, and S. Yooseph, *The disc-covering method for tree reconstruction*, ALEX (1998).

[23] P. Kearney, *The Ordinal Quartet Method*, RECOMB (1998), pp. 125–134.

[24] F. R. McMorris, D. B. Meronk, and D. A. Neumann, *A view of some consensus methods for trees*, in Numerical Taxonomy, J. Felsenstein (ed), Springer Verlag, pp. 122–125, 1983.

[25] S. Sattath and A. Tversky, *Additive similarity trees.* Psychometrika, 42(3) (1997), pp. 319–345.

[26] M. Steel, *The complexity of reconstructing trees from qualitative characters and subtrees*, J. Classif., 9 (1992), pp. 91–116.

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

[28] S. Willson, *Measuring inconsistency in phylogenetic trees*, J. Theoret. Biol, 190 (1998), pp. 15–36.