An improved polynomial time algorithm for computing the refined Buneman tree

Vincent Berry [*]
Dept. of Computer Science, University of Warwick
vincent@dcs.warwick.ac.uk, fax: +44 1203 525 714.

Abstract

We consider the problem of inferring a tree with positive weight edges on a set X from a dissimilarity measure on X. This problem, most commonly met in the classification and numerical taxonomy areas, also arises in evolutionary biology, where we try to estimate the evolutionary tree of a set of species.

Several studies recently put the emphasis on inferring trees containing reliable edges [7, 18]. Buneman [10] proposed a principle to build a tree whose edges all satisfy an important number of combinatorial constraints (called quartets) inferred from the data. The edges of this tree are in that sense *conbinarotially* reliable. However, due to the strict constraints imposed on its edges, this tree sometimes contains very few edges. In evolutionary biology, as in other areas aiming at the inference of a tree from a dissimilarity measure, this is a real drawback since such a tree has a low explanatory power.

Moulton and Steel [17] recently proposed a variant of the Buneman principle. This variant leads to a tree containing more edges (*i.e.*, of increased explanatory power) which still verify an important number of combinatorial constraints inferred from the data. Bryant and Moulton proposed an $O(n^6)$ polynomial time algorithm for solving this problem. We improve this result by providing an $O(n^5)$ algorithm which mainly relies on two ideas: *i)* a new combinatorial technique based on paths in a tree to factorize the computations of quartets commonly related to different edges; *ii)* an interesting merge sort procedure on quartets. These ideas are likely to apply to other problems related to quartets, currently the most popular paradigm used to build trees [7, 8, 11, 14].

Keywords: algorithms and data structures, combinatorial technique, computational biology, dissimilarity analysis, distance-based method, quartets, polynomial time algorithm.

1 Introduction

Phylogenetic reconstruction is a fundamental problem arising in computational biology. The problem is to retrieve the unknown evolutionary history of living species, modeled as a tree, called an *evolutionary tree* or a *phylogeny*. Phylogenies can be reconstructed from character or distance data [21]. Character data consists of homologous nucleotide sequences taken from the species' genome. Distance data, *i.e.*, estimates of the evolutive distance between pairs of species, can be obtained from the sequences by assuming a probabilist model of evolution or by DNA-hybridization. Recently, the emphasis has been put on distance-based methods for which we can prove a worst-case polynomial convergence rate under reasonable evolutive conditions [1, 7, 11].

Several studies have investigated distance-based methods proposing evolutionary trees whose edges are supported by an important number of combinatorial constraints [7, 8, 17]. These constraints are expressed on *quartets* of species, a basic structural unit to describe trees, which can be efficiently inferred from biological data [2, 4, 20] and which has received much attention recently to build phylogenies [6, 7, 9, 11, 14, 20]. *E.g.*, the Q^* method [2, 7] proposes a tree whose edges are supported by $\Omega(n^2)$ to $O(n^4)$ quartets. This method, which relies on a construction introduced by Buneman [10], leads to a tree only containing combinatorially safe edges, but sometimes containing

[*]This work was supported by ESPRIT LTR Project no. 20244 — ALCOM-IT.

few edges compared with the species' unknown phylogeny. At the same time, Moulton and Steel [17] investigated a related method, called the *Refined Buneman tree* since it 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 tree proposed by this new method 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 [6, 18]); more importantly, because of the important constraints they impose on inferred edges, the corresponding tree can be computed in polynomial time, whereas most other methods are NP-hard. This motivates a new and promising approach to phylogeny reconstruction [6, 7, 18, 19]: first compute a tree containing only safe edges through one of the above methods (see [17] for other similar constructions) or through other principles (*e.g.*, consensus methods [18]); then extends this good basis to estimate the species' phylogeny, by relying on an approximate algorithm for an usual reconstruction criterion [6, 18].

We are here interested in the first step of this scheme, and more precisely in the Refined Buneman tree method. This method was initially implemented in the well-known SplitsTree phylogenetic package [13] by an exponential algorithm. More recently, Bryant and Moulton have shown that this problem can be solved in polynomial time [8], but the $O(n^6)$ complexity of their algorithm impedes its application to large data sets.

Here we improve their result by providing an $O(n^5)$ algorithm. The interest in this algorithm also comes from the potentiality of the technique it uses. Indeed, a similar technique could be applied to solve related problems, *e.g.*, to quickly compute the split decomposition of a distance and the associated splitsgraph phylogenetic network [3, 13].

2 Preliminaries

Tree metrics and edge-weighted trees

Let X be a set of n species, and let $\mathcal{D}(X)$ denote the set of dissimilarity measures on X, that is, the set of symmetric functions $d : X^2 \to \mathbb{R}$ that are zero on the diagonal. An X*-tree* is a tree $T = (V, E)$ together with a labeling $L : X \to V$ such that all of the vertices in $V - L(X)$ have degree at least three [4]. In this study we will only consider *leaf-labelled* X-trees (*i.e.*, X-trees T whose leaves bijectively correspond with $L(X)$).

An X-tree $T = (V, E, L)$ together with an edge weighting $w : E \to \mathbb{R}_{>0}$ induces an associated distance function d_T on X: if we denote by $[L(x), L(y)]$ the unique path in T between species x and y, then $d_T(x, y) = \sum_{e \in [L(x),L(y)]} w(e)$. Any distance function arising in this way is called a *tree distance*, and we denote the set of all tree distances on X by $\mathcal{T}(X)$. In phylogenetic reconstruction, X represents a set of species (*e.g.*, animals) on which we know a dissimilarity $d \in \mathcal{D}(X)$. The problem consists in recovering the evolutionary tree of the species, modeled as an X-tree, whose internal nodes represent ancestral species and whose edges represent relationship hypotheses and are weighted by evolutive distances. In other words, we are concerned with maps $\phi : \mathcal{D}(X) \to \mathcal{T}(X)$. Several desirable properties of such maps maybe identified in the phylogenetic context, *e.g.*, continuity, homogeneity, equivariance, polynomial-time computable [17]. The maps that we consider below verify these properties.

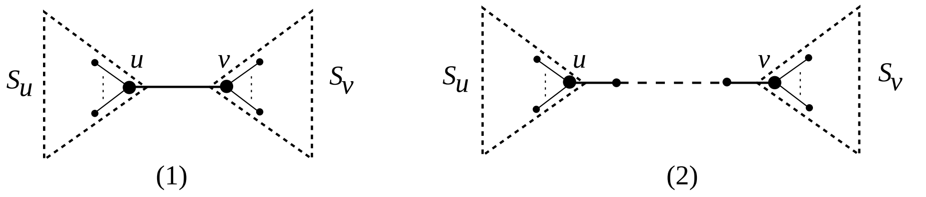

Figure 1: Subtrees S_u, S_v defined by: (1) an edge (u, v); (2) a path $[u, v]$.

Splits

Given an X-tree $T = (V, E, L)$, an edge $e \in E$ induces a *split* $\sigma = \{U, V\}$ of X (that is, a bipartition of X into two non-empty subsets) in a natural way: we say that $x, y \in X$ are in the same component of the split (U or V) if the unique path in T from $L(x)$ to $L(y)$ does not traverse e.

A set of splits is said to be *compatible*, if and only if there exists an X-tree $T = (V, E, L)$ s.t. all splits in the set corresponds to edges in E. Any X-tree T on n species is uniquely characterized by the set of splits corresponding to its k edges [10] and from which it can be reconstructed in $O(kn)$ [16, 12].

Subtrees

In this paper, the term *subtree* only denotes a connected component of an X-tree $T = (V, E, L)$, which can be obtained by deleting an edge $e \in E$. We denote by S_u and S_v the two subtrees associated with the end-points of $e = (u, v)$, *i.e.*, the subtrees containing the species U and the species V respectively (see Fig 1-1), where $\{U, V\}$ is the split induced by e. We refer to S_u and S_v as the subtrees *defined* by the edge $e = (u, v)$.

In the following we will also consider paths $[u, v]$ in a tree, and the subtrees S_u and S_v they *define*: S_u (resp. S_v) denotes the subtree defined by the edge of the path connected to u (resp. v) and not containing other nodes of the path (see Fig. 1-2).

We denote by X_{S_1} the species contained in a subtree S_1, *i.e.*, $X_{S_1} = \{x \in X \text{ s.t. } L(x) \in S_1\}$. Containment relations between subtrees are defined according to the species they contain: $S_1 \subset S_2$ if and only if $X_{S_1} \subset X_{S_2}$.

Quartets

To every set of four species $a, b, c, d \in X$ there are three ways to associate a leaf-labelled binary tree (see Fig 2). The three possible resolutions are denoted by $ab|cd$, $ac|bd$ and $ad|bc$, indicating how the central edge of the tree bipartitions the four species. Each of these trees on four species is also called a *resolved quartet*, or simply a *quartet* [7, 8, 11, 17].

Let $e = (u, v)$ be an internal edge of an X-tree T, we say that e *induces* a quartet $ab|cd$ if e bipartitions the four species in the same way as the central edge of the binary tree representing the quartet. We denote by $Q_e = \{ab|cd,$ s.t. $a, b \in X_{S_u}$, $a \neq b$, $c, d \in$

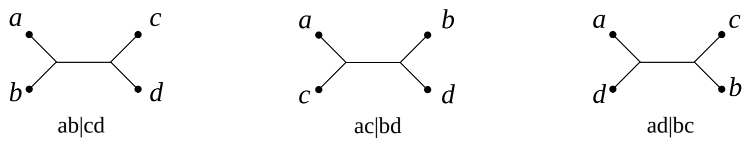

Figure 2: The three possible quartets on species a, b, c, d.

X_{S_v}, $c \neq d\}$ the set of quartets induced by e. If $\sigma = \{U, V\}$ is the split induced by e on X, then the set of quartets *induced* by σ is $Q_\sigma = Q_e$. The quartets *induced* by an X-tree $T = (V, E, L)$ are simply those induced by its edges: $Q_T = \cup_{e \in E} Q_e$. Note that an X-tree T is uniquely characterized by the set Q_T of quartets it induces [2] and from which it can be reconstructed in $O(|Q_T| + n^2)$ time [2].

Similar to the case of edges, the set of quartets *induced* by a path $P = [u, v]$ in an X-tree $T = (V, E, L)$ is defined as $Q_P = \{ab|cd \text{ s.t. } a, b \in X_{S_u}, a \neq b,\ c, d \in X_{S_v}, c \neq d\}$. It can be easily verified that $Q_P = \bigcap_{e \in E} Q_e$, *i.e.*, that Q_P denotes the set of quartets induced by all edges of the path P.

Note that external edges of a leaf-labelled X-tree (*i.e.*, connecting the species of X to the rest of the tree) are of no interest since they belong to all leaf-labelled X-trees. These edges (and the splits they induce) are called *trivial* and do not induce any quartet (they can be combined into a tree inducing either of the three possible resolutions for any set of four species). In the following, we will only consider non-trivial edges and paths containing only non-trivial edges.

The Buneman tree

Buneman first gave a useful map ϕ to obtain a tree from a dissimilarity measure d on X by considering quartets [10]: The *Buneman score* of a quartet $q = ab|cd$, $a, b, c, d \in X$, is defined as:

$$\beta_q = \beta_{ab|cd} := \frac{1}{2}(\min\{ac + bd, ad + bc\} - (ab + cd)),$$

where $xy = d(x, y)$ for $x, y \in X$, and the *Buneman index* of a split $\sigma = \{U, V\}$ of X is

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

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

The Refined Buneman tree

More recently, Moulton and Steel [17] shown that a special relaxation of the condition $\mu_\sigma > 0$ also gives a set of compatible splits: let $q_1, \ldots, q_{|Q_\sigma|}$ be an ordering of the elements in Q_σ such that for all $1 \leq i \leq j \leq |Q_\sigma|$ we have $\beta_{q_i} \leq \beta_{q_j}$. The *refined Buneman index* of a non-trivial split σ is defined as

$$\bar{\mu}_\sigma = \bar{\mu}_\sigma(d) := \frac{1}{n-3} \cdot \sum_{i=1}^{n-3} \beta_{q_i}.$$

Note that $\bar{\mu}_\sigma$ can also be defined for trivial splits σ [8, 17], but this definition implies that $\bar{\mu}_\sigma$ is always positive if d satisfies the triangle inequality (if this is not the case, then, without changing the index of the non-trivial splits, we can add a sufficiently high constant c to every entry of d such that d satisfies the triangle inequality). Thus, *w.l.o.g.*, we will always suppose that the trivial splits have a positive index. The set of splits $RB(d) = \{\sigma \text{ s.t. } \bar{\sigma}(d) > 0\}$ is shown to be compatible in [17] and the associated weighted X-tree, whose edges correspond to the splits $\sigma \in RB(d)$ and are weighted according to $\bar{\mu}_\sigma(d)$, is called the *refined Buneman tree* and is denoted by $T(d)$. 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 Anchored Buneman tree

Fix $x \in X$. Given a split $\sigma = \{U, V\}$ with $x \in U$ define

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

and put $B_x(d) = \{\sigma \text{ s.t. } \mu_\sigma^x > 0\}$. Clearly $\mu_\sigma^x \geq \mu_\sigma$ for all splits σ, so that $B(d) \subseteq B_x(d)$. The set of splits $B_x(d)$ is compatible [8]. The weighted X-tree associated to $B_x(d)$, whose edges correspond to the splits $\sigma \in B_x(d)$ and are weighted according to $\mu_\sigma^x(d)$, is called the *Buneman tree anchored at* x and is denoted by $T_x(d)$.

3 An improved algorithm for the refined Buneman tree

3.1 Mathematical basis of the algorithm

The algorithm designed by Bryant and Moulton [8] to compute the splits $RB(d)$ of the Refined Buneman tree is iterative, *i.e.*, it assumes an arbitrary order on the species $X = \{x_1, \ldots, x_n\}$, 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 [3, 7].

Let $X_k = \{x_1, \ldots, x_k\}$ and d_k be the dissimilarity d restricted to X_k, the correctness of the iterative approach for computing $RB(d)$ relies on the following lemma:

Lemma 1 ([8]) *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 [8] 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 $\sigma \in B_{x_k}(d_k)$ and $\{U \cup x_k, V\}, \{U, V \cup x_k\}$, where $\{U, V\} \in RB(d_{k-1})$. From these splits, only those having a positive refined Buneman index are included in $RB(d_k)$.

The most time consuming step of this $O(n^6)$ algorithm is to compute 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 existing quartets is sorted at the beginning of the algorithm according to their Buneman score. Then, for each examined split σ, the algorithm proceeds in ascending order through the sorted list Q_X, to find the $k-3$ quartets of lower Buneman score induced by σ. 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.

The main inefficiency of this algorithm is to consider the different splits separately. In doing so, it does not to take advantage of the structure of the set of splits considered at each step k, *i.e.*, does not use the fact that they originate from trees ($T_{x_k}(d_k)$ and $T(d_{k-1})$). In the following we propose a combinatorial technique based on this feature of the problem to reduce the number of steps required in the computation of $RB(d_k)$ at each step k.

3.2 Considering paths in a tree

We describe here a combinatorial technique based on the paths in a tree, which enables us to know in an efficient way the quartets of Q_X induced by any given split σ, thus avoiding to consider potentially irrelevant quartets. Moreover, this technique allows to factorize the processing of quartets commonly induced by different edges.

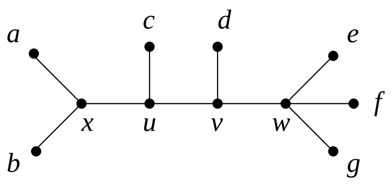

Figure 3: An X-tree with $X = \{a, b, c, d, e, f\}$.

In a given tree $T_{x_k}(d_k)$ (or similarly $T(d_{k-1})$), a quartet, let $ab|cd$, is usually induced by several edges. Define $[u,v]$ to be the path $[a,c] \cap [b,d] = [a,d] \cap [b,c]$. The set of edges of $[u,v]$ is exactly the set of all edges inducing $ab|cd$. Moreover, these edges do not only have $ab|cd$ in common: they also all induce the quartets $xy|zt$ where $x, y \in X_{S_u}$, $x \neq y$ and $z, t \in X_{S_v}$, $z \neq t$, where S_u and S_v are the two subtrees defined by the path $[u,v]$ (cf. Fig. 1-2). The fact that all these edges induce a non-negligible number of quartets in common (in the general case) gives the idea to consider the paths $P_1^e, \ldots, P_l^e$ to which an edge e belongs to describe the set Q_e of quartets induced by the edge as a function of the sets $Q_{P_i^e}$ of quartets induced by the paths P_i^e respectively. More precisely, we have:

$$Q_e = \bigcup_{1 \geq i \geq l} Q_{P_i^e} .$$

The correctness of this formula can be easily verified from the definitions. By sorting separately the lists $Q_{P_i^e}$ of quartets induced by the P_i^e paths, we can then know easily, by a merging procedure, the $k-3$ quartets of smallest Buneman score induced by any edge e. Since any edge e of an X-tree belongs to $O(n^2)$ paths, this merge sort procedure on quartets would allow us to compute the index of any split σ in $O(n^3)$, whereas this required $O(n^5)$ in the algorithm of [8]. Provided that all $Q_{P_i^e}$ lists can be obtained sorted by preprocessing in at most $O(n^4)$, this would allow us to process all the splits arising from $T_{x_k}(d_k)$ and $T(d_{k-1})$ in $O(n^4)$ rather than the $O(n^5)$ required in [8].

However, this analysis is valid only if each quartet is encountered only once in the $Q_{P_i^e}$ sets. This may not be the case since a given quartet may be induced by several P_i^e paths. To avoid this redundancy, we would like a quartet to be registered in only one path, but without losing any information when taking the union of the P_i^e's for *any* edge e. We do that by attaching each quartet $ab|cd$ only to the unique path $[u,v] = [a,c] \cap [b,d] = [a,d] \cap [b,c]$. Equivalently, we define the set $\mathcal{Q}_P$ of quartets which are *specifically induced* by a path $P = [u,v]$ as:

$$\begin{aligned} \mathcal{Q}_P = & \big\{ab|cd \text{ s.t. } a \in X_{S_a}, b \in X_{S_b}, c \in X_{S_c}, d \in X_{S_d}, \\ & S_a \subset S_u, S_b \subset S_u, S_a \neq S_b, S_c \subset S_v, S_d \subset S_v, S_c \neq S_d\big\}, \end{aligned}$$

where S_u and S_v are defined as in section 2. From this definition we have:

Lemma 2 *Let $T = (V, E, L)$ be an X-tree,*
(i) $\forall q = ab|cd \in Q_T$, $\exists P$ *unique s.t.* $q \in \mathcal{Q}_P$.
(ii) $\forall e \in E, Q_e = \bigcup_{1 \geq i \geq l} \mathcal{Q}_{P_i^e}$.

PROOF: *see appendix.*

For example, in Fig. 3, the edge $e = (u,v)$ belongs to paths $[u,v], [u,w], [x,v], [x,w]$ and we have $\mathcal{Q}_{[u,v]} = \{ac|de, ac|df, ac|dg, bc|de, bc|df, bc|dg\}$, $\mathcal{Q}_{[x,v]} = \{ab|de, ab|df, ab|dg\}$, $\mathcal{Q}_{[u,w]} = \{ac|ef, bc|ef, ac|eg, bc|eg, ac|fg, bc|fg\}$, $\mathcal{Q}_{[x,w]} = \{ab|ef, ab|eg, ab|fg\}$, and it can be easily verified that $Q_e = \mathcal{Q}_{[u,v]} \cup \mathcal{Q}_{[u,w]} \cup \mathcal{Q}_{[x,v]} \cup \mathcal{Q}_{[x,w]}$.

We now prove the important fact that the sorted lists $\mathcal{Q}_P$ for all paths P in an X-tree can be obtained in $O(n^4)$. The algorithm we propose relies on the fact that the end-points of (non-trivial) paths are internal nodes of the X-tree and on the following lemma:

Lemma 3 *If A is an internal node of an X-tree $T=(V,E,L)$ and $e_i=(A,B)$, $e_j=(A,C)$, $e_k=(A,D)$ are three edges connected to A (assuming also that e_i is an internal edge), then A is one of the two end-points of the unique path P s.t. $ab|cd \in \mathcal{Q}_P$, $\forall a,b \in X_{S_B}, \forall c \in X_{S_C}$ and $\forall d \in X_{S_D}$.*

Algorithm 1: Compute the sets $\mathcal{Q}_P$ for paths P in an X-tree $T=(V,E,L)$.

```
1 Traverse $T$ to compute for each edge $e = (A,B) \in E$ the lists $X_{S_A}$ and $X_{S_B}$ of
  species contained in the subtrees $S_A$ and $S_B$ it defines.
  /* Associate with each quartet $q$ the end-points of the path $P$ s.t. $q \in \mathcal{Q}_P$: */
2 Traverse $T$ and foreach internal node $A \in V$ do
      foreach $e_i = (A,B)$
          foreach $a,b \in X_{S_B}$
              foreach $e_j = (A,C), e_k = (A,D)$ $(e_i \neq e_j \neq e_k)$
                  foreach $c \in X_{S_C}, d \in X_{S_D}$
3                     Add $A$ to the entry of $Q_X$ which corresponds to $ab|cd$.
  /* Collect the elements of the $\mathcal{Q}_P$ lists from $Q_X$: */
  Let $M_P$ be a square matrix storing $\mathcal{Q}_P$ for all paths $P=[A,B]$ in $T$.
  Set all $\mathcal{Q}_P$ lists in $M_P$ to empty.
4 Traverse $Q_X$ in descending order and foreach quartet $q$ do
      Let $A,B$ be the two nodes associated to $q$ in $Q_X$ during loop 2,
      Add $q$ to $M_P[A,B]$.
```

The correctness of the algorithm follows from lemma 3. Moreover, we can prove:

Theorem 4 *If $T=(V,E,L)$ is an X-tree, then Algorithm 1 computes in $O(n^4)$ the sets $\mathcal{Q}_P$ of quartets specifically induced by the paths P in T, where $n=|X|$.*

PROOF:

- Line 1 is implemented by a simple post-ordered recursive search of the tree. For each edge $e=(A,B)\in E$ processed during this search, assuming *w.l.o.g.* that A is the father of B in the search of the tree, the list X_{S_B} of species contained in a subtree S_B is simply the union of the lists of species of the subtrees contained in S_B. Moreover, the list of species associated to the subtree S_A is simply $X - X_{S_B}$. The execution of line 1 thus requires no more than $O(n^2)$.

- The loop of line 2 requires $O(n^4)$ if we perform a global analysis: each quartet q is referenced only twice (respectively, when A and B are processed, where $P=[A,B]$ is the unique path P s.t. $q \in \mathcal{Q}_P$), and all referenced quartets are obtained in $O(1)$ through the X_{S_i} lists implemented as chained lists. Moreover line 3 can be performed in $O(1)$, by resorting to the usual matrix M_Q with four dimensions [7] (each entry $M_Q[a,b,c,d]$ has a sub-entry for each of the 3 possible resolutions

Algorithm 2: A merge sort algorithm on quartet to construct $RB(d)$.

```
5  Construct the list $Q_X$ of all possible quartets $ab|cd$ on $X$, sorted according to their
   Buneman score $\beta_{ab|cd}$.
6  Let $\mathcal{S}_4 := B(d_4)$.
7  foreach $k$ from 5 to $n$
8      Let $\mathcal{S}_k = \{\{x_k, X_k - x_k\}\}$.
9      Construct $B_{x_k}(d_k)$ and $T_{x_k}(d_k) = (V_{x_k}, E_{x_k}, L_{x_k})$.
10     Compute the sets $\mathcal{Q}_P$ of quartets induced by all paths $P$ in $T_{x_k}(d_k)$.
11     foreach edge $e \in E_{x_k}$  compute the lists of paths $P_i^e$ s.t. $e \subset P_i^e$.
12     foreach  each split $\sigma$ corresponding to an edge $e \in E_{x_k}$
13         Compute the list $L_\sigma$ of $k-3$ quartets of smallest Buneman score it induces.
14         Add $\sigma$ to $\mathcal{S}_k$ iff $\bar{\mu}_\sigma > 0$.
15     Construct $T(d_{k-1}) = (V_{d_{k-1}}, E_{d_{k-1}}, L_{d_{k-1}})$ from $\mathcal{S}_{k-1}$.
16     Compute the sets $\mathcal{Q}_P$ of quartets induced by all paths $P$ in $T(d_{k-1})$.
17     foreach edge $e \in E_{d_{k-1}}$  compute the lists of paths $P_i^e$ s.t. $e \subset P_i^e$.
18     Construct the list $L_{x_k}$ of all possible quartets to which $x_k$ belongs, sorted ac-
       cording to their Buneman score.
19     foreach split $\sigma = \{U, V\}$ corresponding to an edge $e \in E_{d_{k-1}}$
20         Compute the lists $L_{\sigma'}$ and $L_{\sigma''}$ of the $k-3$ quartets of smallest Buneman score
           induced respectively by $\sigma' = \{U \cup x_k, V\}$ and $\sigma'' = \{U, V \cup x_k\}$.
21         Add $\sigma'$ and $\sigma''$ to $\mathcal{S}_k$ whenever $\bar{\mu}_{\sigma'} > 0$, resp. $\bar{\mu}_{\sigma''} > 0$, or whenever they are trivial.
   Output $RB(d) = \mathcal{S}_n$ and $\{\bar{\mu}_\sigma$ s.t. $\sigma \in \mathcal{S}_n\}$.
```

of (a, b, c, d), which contains a pointer on the entry of Q_X corresponding to the resolution). This matrix is initialized in an $O(n^4)$ straight-forward way when sorting the quartets of Q_X, at the beginning of the main algorithm.

- The traversal of Q_X in line 4 requires $O(1)$ operations to process each entry of Q_X, since each one is associated with only three informations (the quartet q and the two end-points A and B of the path $P = [A, B]$ s.t. $q \in \mathcal{Q}_P$), which enables us to perform in $O(1)$ the addition of q to the head of the chained list $M_P[A, B] = \mathcal{Q}_P$. Since Q_X contains $O(n^4)$ entries, the execution of line 4 requires $O(n^4)$. □

3.3 The $O(n^5)$ algorithm to compute $BD(d)$

Thanks to the preprocessing of the paths described in the previous section, computing the refined Buneman index of all the splits arising from $T_{x_k}(d_k)$ and $T(d_{k-1})$ is performed as a series of merge sorts which only require $O(n^4)$. We detail the resulting procedure used to compute $RB(d)$ in Algorithm 2. Its correctness follows from lemmas 1, 2 and 3.

Theorem 5 *If d is a dissimilarity on X and $n := |X| \geq 4$, then Algorithm 2 constructs $RB(d)$ in time $O(n^5)$.*

PROOF:

- Line 5 requires $O(n^4 \log n)$ since $3\binom{n}{4}$ elements are sorted. Line 6 is in $O(1)$.
- The main loop, line 7, is performed $O(n)$ times. Line 8 requires $O(1)$ at each step of this loop, *i.e.*, is $O(n)$ on the overall execution.

- Line 9: the splits of $B_{x_k}(d_k)$ can be obtained in a natural way in $O(k^4)$, as indicated in [8], which leads to $O(n^5)$ on the overall execution of the algorithm. A more involved argument shows that $B_{x_k}(d_k)$ can actually be computed in $O(n^2)$ (Bryant, personal communication), but the $O(n^4)$ bound suffices to establish the result.

  Constructing the tree on n species whose edges correspond to the k splits of some set is only $O(kn)$ thanks to the linear algorithms of Meacham and Gusfield [16, 12]. Since $B_{x_k}(d_k)$ contains $O(n)$ splits, constructing $T_{x_k}(d_k)$ only requires $O(n^2)$ at each step, *i.e.*, $O(n^3)$ for the whole algorithm.

- Line 10 require $O(n^4)$ from theorem 4 and thus $O(n^5)$ on the whole algorithm.

- Line 11 is performed in a similar way as line 1 of Algorithm 1. The tree is recursively traversed and for each edge $e = (u, v)$ we establish the lists of internal nodes V_u^i and V_v^i contained in the two subtrees S_u and S_v defined by e, then we know that the lists of paths to which e belong is exactly $L_p[e] = \{[u, v] \text{ s.t. } u \in V_u^i, v \in V_v^i\}$. Since $L_p[e]$ contains up to $O(n^2)$ paths, each obtained in $O(1)$, line 11 requires $O(n^3)$ at each step k, *i.e.*, $O(n^4)$ for the whole algorithm.

- Lines 12 to 14 are performed like a merging procedure: we merge the $O(k^2)$ sorted lists $\mathcal{Q}_{P_i^e}$ into one sorted list, corresponding to the quartets induced by the edge. However, we stop the process when knowing the first $k - 3$ elements of the list, which suffice to compute $\bar{\mu}_\sigma$. This classical procedure can be performed in $O(k^3)$, for each split σ. Since at each step k we examine the $O(k)$ splits of the tree $T_{x_k}(d_k)$, this line requires $O(k^5)$, *i.e.*, $O(n^5)$ for the whole algorithm.

- Line 15, requires $O(n^2)$ at each step (as the second half of line 9), *i.e.*, $O(n^3)$ for the whole algorithm. Lines 16 and 17 (as steps 10 and 11) respectively require $O(n^5)$ and $O(n^4)$.

- Line 18 requires $O(n^3 \log n)$ at each of the k steps, *i.e.*, $O(n^4 \log n)$ on the whole.

- Lines 19 to 21: the loop of line 19 is performed $O(k)$ times at each step k, *i.e.*, $O(n^2)$ for the whole algorithm. The splits σ' and σ'' induce the same quartets as σ, plus some quartets of L_{x_k}. Thus, to obtain $L_{\sigma'}$ (equiv. $L_{\sigma''}$), line 20 proceeds by merging the $\mathcal{Q}_{P_i^e}$ lists, as line 13 (one more list being considered, namely L_{x_k}). The only difference with line 13 is that when the smallest quartet of L_{x_k} is considered, it is disregarded if not induced by σ' (resp σ''). Knowing if a quartet $q = ab|cx_k \in L_{x_k}$ is induced by σ' (resp σ'') can be checked in $O(1)$ if we store for each split $\sigma = \{U, V\}$ of $T(d_{k-1})$ the species contained in its two components in a binary array of size n (these arrays can be initialized in $O(k^3)$ during the traversal of line 1 of Algorithm 1, called at line 16 of the present algorithm). During the merging procedure, at most $O(k^3)$ quartets from L_{x_k} are considered, and at most $O(k)$ quartets from the $O(k^2)$ other lists, each quartet being processed in $O(1)$. As a result, line 20 requires $O(k^3)$ operations, *i.e.*, $O(n^5)$ for the whole algorithm since it is performed at each step of the loop of line 19. Computing $\bar{\mu}_{\sigma'}$ and $\bar{\mu}_{\sigma''}$ requires only $O(k)$, thus $O(n^5)$ for the whole algorithm. □

4 Discussion

In this paper we proposed a faster algorithm to compute the Refined Buneman tree on a set of species from a dissimilarity measure. The $O(n^5)$ complexity achieved by the new algorithm may still appear important, however, note that it is only cubic in the size of the input d. Moreover, the Refined Buneman method in itself is intrinsically tied to quartets of species (through the β_q's), whose number is in $O(n^4)$. This seems to indicate a lower bound on the best achievable complexity for this problem, as long as we want to compute the index of the splits we infer.

The space complexity of the algorithm is in $O(n^4)$ as also required by the algorithm of Bryant and Moulton [8]. Reducing this complexity is one of the next important steps.

The lower complexity achieved in this paper for computing the Refined Buneman Tree also makes the method more suitable for comparison with other methods of phylogeny reconstruction through intensive simulations, as is usual in the domain [7, 15].

Finally, note that besides the improvement in complexity it provides for the Refined Buneman problem, the quartet merge sort technique detailed in this paper seems promising for tackling other similar problems, *e.g.*, computing the splitsgraph phylogenetic network [3, 13].

Acknowledgments

I would like to thank D. Bryant and V. Moulton for discussing their results and for checking those of the current paper. Thanks are due to D. Bryant for a careful reading of this manuscript.

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

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

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

[5] V. Berry. Improving the bound for computing the refined buneman tree. manuscript, 1998.

[6] V. Berry and O. Gascuel. Reconstructing phylogenies from resolved 4-trees. Technical Report 97076, LIRMM, 1997.

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

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

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

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

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

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

[13] D. Huson. SplitsTree - a program for analyzing and visualizing evolutionary data. *CABIOS (to appear)*, 1997.

[14] T. Jiang, P. Kearney, and M. Li. Orchestrating quartets: approximation and data correction. submitted, 1998.

[15] M.K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.*, 11:459–468, 1994.

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

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

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

[19] 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.

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

[21] D.L. Swofford, G.J. Olsen, P.J. Wadell, and D.M. Hillis. *Molecular systematics (2nd edition)*, chapter Phylogenetic Inference, pages 407–514. Sunderland, USA, 1996.

Appendix - proof of lemma 2

Recall that P_i^e denotes the set of (non-trivial) paths in T to which the edge $e \in E$ belongs.

$\underline{(i)\ \forall q = ab|cd \in Q_T,\ \exists P \text{ unique s.t. } q \in \mathcal{Q}_P:}$

$$\begin{aligned} q \in \mathcal{Q}_{P_i^e} &\equiv \{\ [a,u]\cap[b,u] = u\ ,\ [c,v]\cap[d,v] = v\ ,\ a,b \in X_{S_u}, a \neq b\ ,\ c,d \in X_{S_v}, c \neq d\ \} \\ &\equiv [u,v] = [a,c]\cap[b,d] = [a,d]\cap[b,c]. \end{aligned}$$

However, there exists a unique path in T corresponding to $[a,c]\cap[b,d]$ (and to $[a,d]\cap[b,c]$), which means that $P_i^e = [u,v]$ is the unique path in T s.t. $q \in \mathcal{Q}_{P_i^e}$.

$\underline{(ii)\ \forall e \in E, Q_e = \bigcup_{1\geq i \geq l} \mathcal{Q}_{P_i^e}:}$

Let $e = (u,v)$ be any given edge of E.

a) $\underline{\forall q \in Q_e \Rightarrow \exists P_i^e \text{ s.t. } q \in \mathcal{Q}_{P_i^e}:}$

W.l.o.g. let $q = ab|cd$. $q \in Q_e \Rightarrow a,b \in X_{S_u}, a \neq b,\ c,d \in X_{S_v}, c \neq d \Rightarrow$

- the node A indicated by $[a,c]\cap[b,d]\cap[a,b]$ is in S_u.
- the node B indicated by $[a,c]\cap[b,d]\cap[c,d]$ is in S_v.

Since nodes A and B are the end-points of the unique path P s.t. $q \in \mathcal{Q}_P$, $A \in S_u$ and $B \in S_v$ imply that e belongs to P (otherwise there would be a cycle in T).

b) $\underline{\forall P_i^e, \forall q \in \mathcal{Q}_{P_i^e} \Rightarrow q \in Q_e:}$

W.l.o.g., let $P_i^e = [u',v']$ and $q = ab|cd$. $q \in \mathcal{Q}_{P_i^e} \Rightarrow a \neq b$ and $a,b \in X_{S_{u'}}$, $c \neq d$ and $c,d \in X_{S_{v'}}$. Since $e \in P_i^e$, we have $S_{u'} \subseteq S_u$ and $S_{v'} \subseteq S_v$, thus $a,b \in X_{S_u}$ and $c,d \in X_{S_v}$. This implies $ab|cd = q \in Q_e$ by definition of Q_e. □