Fishing for Minimum Evolution Trees with Neighbor-Nets

Sarah Bastkowski[a,*], Andreas Spillner[b], Vincent Moulton[a]

[a] *School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK*
[b] *Department of Mathematics and Computer Science, University of Greifswald, Germany*

Abstract

In evolutionary biology, biologists commonly use a phylogenetic tree to represent the evolutionary history of some set of species. A common approach taken to construct such a tree is to search through the space of all possible phylogenetic trees on the set so as to find one that optimizes some score function, such as the minimum evolution criterion. However, this is hampered by the fact that the space of phylogenetic trees is extremely large in general. Interestingly, an alternative approach, which has received somewhat less attention in the literature, is to instead search for trees within some set of bipartitions or splits of the set of species in question. Here we consider the problem of searching through a set of splits that is circular. Such sets can, for example, be generated by the NeighborNet algorithm for constructing phylogenetic networks. More specifically, we present an $O(n^4)$ time algorithm for finding an optimal minimum evolution tree in a circular set of splits on a set of species of size n. In addition, using simulations, we compare the performance of this algorithm when applied to NeighborNet output with that of FastME, a leading method for searching for minimum evolution trees in tree space. We find that, even though a circular set of splits represents just a tiny fraction of the total number of possible splits of a set, the trees obtained from circular sets compare quite favorably with those obtained with FastME, suggesting that the approach could warrant further investigation.

Keywords: Algorithms, Phylogenetics, Minimum Evolution, Dynamic Programming, Phylogenetic Network, Circular Split System

1. Introduction

A *phylogenetic tree* on a given set of species X is a connected, acyclic graph such that its leaf set is X and all its non-leaf vertices have degree at least three [24]. Such trees are used by biologists to represent the evolutionary history of the species in X. An important problem in phylogenetics is to construct such trees, and various methods have been developed for this purpose [18]. A common approach to tackling this problem is to search through the space of phylogenetic trees, trying to find a tree (or trees) that optimize some score such as the minimum evolution criterion [23]. However, a straight-forward exhaustive search is hampered by the fact that the space of phylogenetic trees on X grows exponentially in $n = |X|$. Moreover, it has been shown that finding an optimal tree is NP-hard for many of the popular optimization criteria (see e.g. [6, 8]).

Interestingly, there is an alternative approach to searching through tree space, which was studied quite early on in the development of phylogenetics (see e.g. [10, 21]), and more recently in [3], but that has received somewhat less attention in the literature. In particular, instead of searching through the set of all possible trees on the set X, we look for trees within a collection of bipartitions or *splits* of X. The rationale behind this approach is that any phylogenetic tree induces a set of splits of X in which every split corresponds to a branch of the tree, and that this set of splits uniquely determines the tree (cf. [24]). Intriguingly, in [4] a dynamic programming framework is developed to search for trees in a given collection of splits of X, also called a *split system*. Although still requiring exponential time in general, this approach has the advantage

*Corresponding author. Tel.: +44-7407385089
Email addresses: s.bastkowski@uea.ac.uk (Sarah Bastkowski), andreas.spillner@uni-greifswald.de (Andreas Spillner), vincent.moulton@cmp.uea.ac.uk (Vincent Moulton)

that it can yield polynomial time algorithms when restricted to split systems having size that is polynomial in $n = |X|$. It is therefore of interest to develop efficient algorithms to search for trees in special classes of split systems, as well as ways to generate split systems which capture salient information.

In this vein, here we develop an algorithm for searching for a tree that locally optimizes the minimum evolution criterion by searching in a *circular* split system. This is a special type of split system that can be generated, for example, by the NeighborNet algorithm [5] for constructing phylogenetic networks (see Figure 1 for an example). In particular, we show that for a circular split system there is an $O(n^4)$ time algorithm for computing an optimal minimum evolution tree, which improves on the run time of $O(n^7)$ for the more general minimum evolution algorithm presented by Bryant in [4, Section 5.5]. We also present some simulations which indicate that minimum evolution trees in circular split systems generated by NeighborNet can compare favorably with those obtained by searching through the whole of tree space.

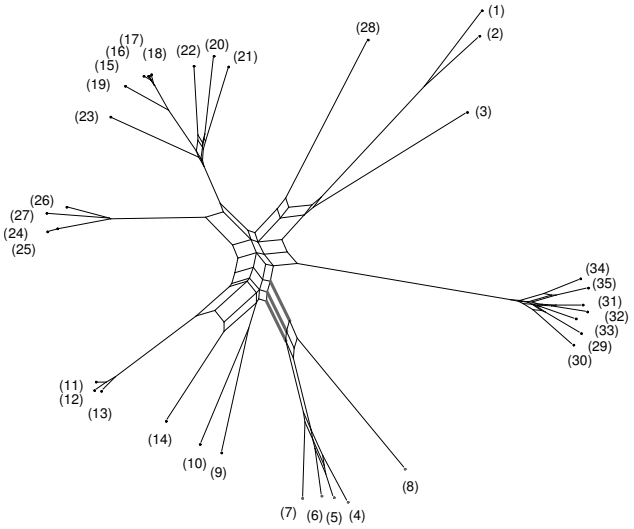

Figure 1: A phylogenetic network generated with the NeighborNet algorithm displaying a circular split system for 35 tomato-infecting begomoviruses [22]. The group of three bold gray branches, for example, represents the split of $X = \{1, 2, \ldots, 35\}$ into the subsets $A = \{4, 5, 6, 7, 8\}$ and $B = X - A$.

Before continuing, we note that, in view of the fact that split systems are often displayed by phylogenetic networks such as the one in Figure 1, it might appear that the problem of searching for trees in split systems is closely related to the problem of finding optimal subtrees in phylogenetic networks. While some recent results on this latter problem can be found in [16, 17], it is, in fact, quite different from the problem we study here since, for example, the minimum evolution tree in a circular split system generated by NeighborNet is not necessarily a subtree of the network used to display this split system.

The structure of the rest of this paper is as follows. After recalling some background material on the minimum evolution problem in the next section, in Section 3, we recall Bryant's dynamic programming algorithm for finding minimum evolution trees in a split system. We then describe our new algorithm in Section 4 and, in the following section, we present a short investigation into how the minimum evolution trees within split systems generated by NeighborNet and some related methods compare with those generated by FastME [11], one of the leading programs for finding minimum evolution trees by searching through tree space. We conclude with a discussion of some possible future directions in Section 6.

2. The Minimum Evolution Problem

We begin by recalling some relevant terminology and notation (cf. also [24]). Let X be a finite, non-empty set, usually corresponding to some set of species or taxa. A *phylogenetic tree* (on X) is a connected, acyclic graph $T = (V, E)$ with leaf set X. Any non-leaf vertex of T is called an *internal vertex* of T, a branch incident to a leaf is called an *external branch* of T and a branch whose endpoints are both internal vertices is called an *internal branch* of T. In this paper, we consider only *binary* phylogenetic trees, that is, trees in which every interior vertex is incident to precisely three branches. Often the branches $e \in E$ of a phylogenetic tree $T = (V, E)$ are assigned a real number $\omega(e)$ known as the branch's *length*. The sum of the lengths of all branches in a phylogenetic tree T is called the *length* of T and denoted by $\ell_\omega(T)$. In addition, we denote the total length of the branches on the path connecting any two leaves x and y of T by $\ell_\omega(x, y)$.

When constructing phylogenetic trees, biologists often begin by computing a distance matrix D on X (estimated from, for example, molecular sequences) [18, ch. 4], that is, a symmetric matrix D indexed by the set X which assigns the distance $D(x, y)$ between x and y to any pair x, y of elements in X and which is zero on the diagonal. Given such a matrix D and a phylogenetic tree $T = (V, E)$ on X, various ways have

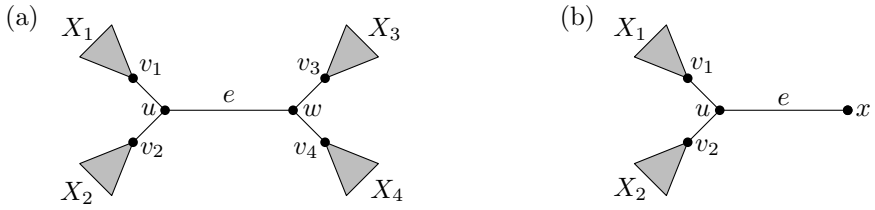

Figure 2: Schematic representation of the situation around (a) an internal branch e and (b) an external branch e of a phylogenetic tree referred to in the context of Formulae (1) and (2).

been proposed to assign branch lengths $\omega_D(e)$ to each $e \in E$ (see e.g. [6]). Here, we focus on the so-called Ordinary Least Squares (OLS) branch length estimates that result from minimizing the quantity

$$\sum_{x,y \in X, x \neq y} \big(\ell_{\omega_D}(x,y) - D(x,y)\big)^2$$

and that can be computed using the following formulae (cf. [4, p. 136], [23]). Specifically, for an internal branch $e = \{u, w\}$, letting v_i, $1 \leq i \leq 4$, denote the four vertices in $V - \{u, w\}$ that are adjacent to u or w (cf. Figure 2(a)), we define the subset $X_i \subseteq X$ consisting of those $x \in X$ for which the unique path from u to x in T contains vertex v_i. Then we have

$$\begin{aligned}\omega_D(e) =& \frac{1}{4(n_1+n_2)(n_3+n_4)}\Big((\frac{n}{n_4}+\frac{n}{n_3}+\frac{n}{n_2}+\frac{n}{n_1}-4)P_0 \\ &+ \frac{n_1+n_2}{n_1 n_2}((2n_2-n)P_1+(2n_1-n)P_2) \\ &+ \frac{n_3+n_4}{n_3 n_4}((2n_4-n)P_3+(2n_3-n)P_4)\Big) = \omega_D(X_1,X_2,X_3,X_4),\end{aligned} \tag{1}$$

where $P_0 = \sum_{x \in X_1 \cup X_2, y \in X-(X_1 \cup X_2)} D(x,y)$, $n_i = |X_i|$ and $P_i = \sum_{x \in X_i, y \in X - X_i} D(x,y)$, $1 \leq i \leq 4$. Similarly, for every external branch e (cf. Figure 2(b)) we have

$$\omega_D(e) = \frac{1}{4(n_1 n_2)}\Big((1+n_1+n_2)P_0 - (1+n_1-n_2)P_1 - (1-n_1+n_2)P_2\Big) = \omega_D(X_1,X_2). \tag{2}$$

Note that in both of these formulae $\omega_D(e)$ only depends on the subsets of X that form the leaf sets of the subtrees incident to the endpoints of e and not on the internal structure of these subtrees.

Now, given T and D as above, the *Minimum Evolution (ME) score* $\sigma_D(T)$ is defined to be $\ell_{\omega_D}(T)$. Note that, given T and D, it is possible to compute $\sigma_D(T)$ for given D and T in $O(n^2)$ time [4, p. 137] using the formulae above. The *Minimum Evolution problem* is, for a given distance matrix D, to find a phylogenetic tree T on X with smallest score $\sigma_D(T)$. Note that, even though we are not aware of an NP-hardness proof for precisely this problem in the literature, many related problems are known to be NP-hard (see e.g. [7, 13]).

3. Bryant's Algorithm

In the following, we denote the split of X into two non-empty subsets A and B by $A|B$ $(= B|A)$ and the set of all possible splits of X by $\Sigma(X)$. Any subset $\Sigma \subseteq \Sigma(X)$ is called a split system *on* X. As mentioned in the introduction, we are interested in the problem of searching for trees in split systems. More specifically, given a distance matrix D and a split system Σ on X, the *restricted ME-problem* requires us to find the minimum $\sigma_{(D,\Sigma)}$ of $\sigma_D(T)$ over all binary phylogenetic trees T on X with $\Sigma_T \subseteq \Sigma$, where Σ_T is the split system consisting of the splits associated to the branches of T. Any phylogenetic tree T on X with $\Sigma_T \subseteq \Sigma$ and $\sigma_D(T) = \sigma_{(D,\Sigma)}$ is called a *restricted ME-tree* for D and Σ. Note that the original ME-problem corresponds to the restricted version with $\Sigma = \Sigma(X)$. In [4] Bryant presented an algorithm for solving the restricted ME-problem with run time $O(n^2k + nk^3)$, where $n = |X|$ and $k = |\Sigma|$. In this section, we present

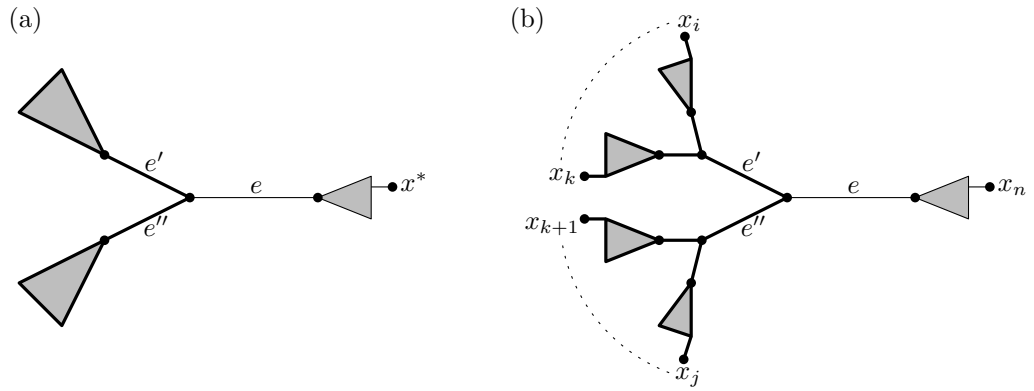

Figure 3: (a) The overall structure of a phylogenetic tree $T \in \mathcal{T}(S, S', S'')$ for some relevant triple $(S = S_e, S' = S_{e'}, S'' = S_{e''})$ of splits. (b) In case the split system Σ is circular, we can use the special structure of Σ to pin down the relevant triples: $S_e = S_{i,j}$, $S_{e'} = S_{i,k}$ and $S_{e''} = S_{k+1,j}$ for some $1 \leq i \leq k < j < n$.

a self-contained version of this algorithm, also describing it in such a way that it can be easily adapted to our specific needs in the next section.

As above, let D denote the given distance matrix on X and $\Sigma \subseteq \Sigma(X)$ denote the given split system. In addition, the split induced by a branch e in a phylogenetic tree is denoted by S_e. For any split $S = A|B$ of X and any $x \in X$, we denote by $S(x)$ that set, A or B, that contains x and by $\overline{S}(x)$ the other set. To be able to apply and evaluate the Formulae (1) and (2) in constant time in the course of the algorithm, we compute, as a preprocessing step, for every split $S = A|B \in \Sigma$, the cardinalities $|A|$ and $|B|$ as well as the value $P_S = \sum_{x \in A, y \in B} D(x, y)$. This preprocessing can clearly be done in $O(n^2 k)$ time. Moreover, we store the splits in Σ in a suitable data structure $\mathcal{D}$ (e.g. in a multidimensional dictionary [14]) that allows to check in $O(n)$ time whether a given split is contained in Σ.

Bryant's algorithm uses a dynamic programming scheme to compute $\sigma_{(D,\Sigma)}$ [9, ch. 15]. Each subproblem arising in this scheme corresponds to a particular triple $(S = S_e, S' = S_{e'}, S'' = S_{e''})$ of splits that correspond to edges e, e' and e'' in the resulting phylogenetic tree as indicated in Figure 3(a). To describe this more precisely, we introduce some further notation. Fix an arbitrary element $x^* \in X$. We call an ordered triple (S, S', S'') of splits $S, S', S'' \in \Sigma$ *relevant* if $\overline{S'}(x^*) \cup \overline{S''}(x^*) = \overline{S}(x^*)$ and $\overline{S'}(x^*) \cap \overline{S''}(x^*) = \emptyset$ hold, and denote the set of relevant triples of splits in Σ by $\mathrm{rel}(\Sigma)$. In addition, for any $(S, S', S'') \in \mathrm{rel}(\Sigma)$, we let $\mathcal{T}(S, S', S'')$ denote the set of binary phylogenetic trees T on X with $\{S, S', S''\} \subseteq \Sigma_T \subseteq \Sigma$ and define

$$\sigma_D(S, S', S'') = \min_{T=(V,E) \in \mathcal{T}(S,S',S'')} \left(\sum_{f \in E, \overline{S_f}(x^*) \subsetneq \overline{S}(x^*)} \omega_D(f) \right). \tag{3}$$

Note that, as indicated in Figure 3(a), only the lengths of the branches in the bold part of any tree $T \in \mathcal{T}(S, S', S'')$ are taken into account in the sum in Formula (3).

In view of the structure of the trees in $\mathcal{T}(S, S', S'')$ for any $(S, S', S'') \in \mathrm{rel}(\Sigma)$, it is not hard to derive the following recursive formula for $\sigma_D(S, S', S'')$:

$$\sigma_D(S, S', S'') = \alpha(S, S', S'') + \beta(S, S', S'') \quad \text{with}$$

$$\alpha(S, S', S'') = \begin{cases} \omega_D(S(x^*), \overline{S''}(x^*)) & \text{if } |\overline{S'}(x^*)| = 1 \\ \min\limits_{(S', S_1, S_2) \in \mathrm{rel}(\Sigma)} \left(\sigma_D(S', S_1, S_2) + \omega_D(\overline{S_1}(x^*), \overline{S_2}(x^*), \overline{S''}(x^*), S(x^*)) \right) & \text{otherwise,} \end{cases}$$

$$\beta(S, S', S'') = \begin{cases} \omega_D(S(x^*), \overline{S'}(x^*)) & \text{if } |\overline{S''}(x^*)| = 1 \\ \min\limits_{(S'', S_1, S_2) \in \mathrm{rel}(\Sigma)} \left(\sigma_D(S'', S_1, S_2) + \omega_D(\overline{S_1}(x^*), \overline{S_2}(x^*), \overline{S'}(x^*), S(x^*)) \right) & \text{otherwise,} \end{cases}$$

where, in case the minimum is taken over the empty set, we assume that the value $+\infty$ is obtained.

Note that the formulae for $\alpha(S, S', S'')$ and $\beta(S, S', S'')$ only involve (i) values $\omega_D(\cdot)$ which can be computed in constant time in view of the preprocessing mentioned above and (ii) values $\sigma_D(S', \cdot, \cdot)$ and

$\sigma_D(S'', \cdot, \cdot)$ for which, by definition, $|\overline{S'}(x^*)| < |\overline{S}(x^*)|$ and $|\overline{S''}(x^*)| < |\overline{S}(x^*)|$ hold. Also note that the number of relevant triples of the form (S', S_1, S_2) and (S'', S_1, S_2), respectively, is in $O(k)$ and that, given S' and S_1 (or, similarly, S'' and S_1), with the help of the data structure $\mathcal{D}$ we can check in $O(n)$ time whether there exists a suitable split $S_2 \in \Sigma$ to form a relevant triple (S', S_1, S_2) (or (S'', S_1, S_2)). Hence, within the dynamic programming scheme each value $\sigma_D(S, S', S'')$ can be computed in $O(nk)$ time. Since there are $O(k^2)$ triples in $\mathrm{rel}(\Sigma)$, the overall run time is therefore in $O(nk^3)$.

To conclude the description of the algorithm, note that, for any $(S, S', S'') \in \mathrm{rel}(\Sigma)$ and any $T \in \mathcal{T}(S, S', S'')$, the sum in Formula (3) does never include the length of the branch e^* of T that is incident to x^* and corresponds to the split $S^* = S_{e^*} = \{x^*\}|X - \{x^*\}$. Therefore, to obtain the minimum of the total length of the resulting tree, in the last step we compute

$$\sigma_{(D,\Sigma)} = \min_{(S^*,S_1,S_2)\in \mathrm{rel}(\Sigma)} \big(\sigma_D(S^*, S_1, S_2) + \omega_D(\overline{S_1}(x^*), \overline{S_2}(x^*))\big),$$

which can clearly be done in $O(nk)$ time. So, the overall run time is indeed in $O(nk^3)$ and, by tracing back the computation of $\sigma_{(D,\Sigma)}$ through the dynamic programming scheme, we can easily obtain a restricted ME-tree for D and Σ in case $\sigma_{(D,\Sigma)} \neq +\infty$. Otherwise there is no binary phylogenetic tree T on X with $\Sigma_T \subseteq \Sigma$.

4. Computing ME-trees in Circular Split Systems

We now focus on the restricted ME-problem for a circular split system. This is a special type of split system that can be generated, for example, from a distance matrix D using the NeighborNet algorithm [5]. More specifically, a split system $\Sigma \subseteq \Sigma(X)$ is *circular* [1] if there exists an ordering $x_1, x_2, \ldots, x_n$ of the elements in X such that, for every split $A|B \in \Sigma$, there exist $1 \leq i \leq j < n$ with $A = \{x_i, x_{i+1}, \ldots, x_j\}$ or $B = \{x_i, x_{i+1}, \ldots, x_j\}$. If such an ordering of X exists it can be computed in $O(nk)$ time, $k = |\Sigma|$, [12] and we will then also say that Σ *fits* on that ordering. Note that the maximum possible number of splits in a circular split system Σ on X is $\binom{n}{2}$ [2]. Thus, Bryant's algorithm runs in $O(n^7)$ time on a circular split system. We now show how this can be improved to $O(n^4)$.

Assume that Σ is a circular split system and let $x_1, x_2, \ldots, x_n$ be an ordering of X onto which Σ fits. We put $x^* = x_n$ and, for any $1 \leq i \leq j < n$, we define the split $S_{i,j} = \{x_i, x_{i+1}, \ldots, x_j\}|X - \{x_i, x_{i+1}, \ldots, x_j\}$. Note that $\Sigma \subseteq \{S_{i,j} : 1 \leq i \leq j < n\}$ holds, that is, every split in Σ corresponds to a unique pair of indices i and j. As an immediate consequence it follows that the preprocessing outlined above can be done in $O(n^2)$ time (for computing the values P_S see e.g. [20] for an $O(n^2)$ time algorithm in a more general context).

The key observation, however, is that every relevant triple $(S, S', S'') \in \mathrm{rel}(\Sigma)$ must be such that $S = S_{i,j}$ and either $S' = S_{i,k}$ and $S'' = S_{k+1,j}$ or $S' = S_{k+1,j}$ and $S'' = S_{i,k}$ for some $1 \leq i \leq k < j < n$ (cf. Figure 3(b)). Hence, for any splits $S', S'' \in \Sigma$, there are only $O(n)$ triples (S', S_1, S_2) and (S'', S_1, S_2) in $\mathrm{rel}(\Sigma)$. Moreover, for the data structure $\mathcal{D}$ we can simply use a two-dimensional array in which we mark the presence/absence of the split $S_{i,j}$ in Σ for each pair $1 \leq i \leq j < n$. Then, we can easily check whether a split is contained in Σ in constant time. As a consequence, within the dynamic programming scheme each value $\sigma_D(S, S', S'')$ can be computed much faster, namely in $O(n)$ time. Together with the fact that there are only $O(n^3)$ triples in $\mathrm{rel}(\Sigma)$, this implies that the overall run time is $O(n^4)$.

5. Simulations

To measure the computational performance of the algorithm we tested it on simulated data sets and compared it with FastME [11], one of the leading methods to construct an approximation of an ME tree. Note that FastME performs a local search in tree space using a neighborhood based on certain types of tree edit operations. In FastME we chose the NeighborJoining-tree option as the start topology for the local search together with nearest neighbor interchange (NNI) tree edit operations, and ordinary least squares (OLS) for searching the neighborhood of a tree.

In our experiments, we considered sets X with $n = 25$, 50, 100, 200, 400 and 800 taxa and generated 100 treelike as well as 100 random, non-treelike distance matrices on each of them. To simulate treelike matrices, we followed the procedure described in [15]. In particular, we evolved molecular sequences of length 1000 along a tree (with probability r of recombination set to 0), and computed a distance matrix

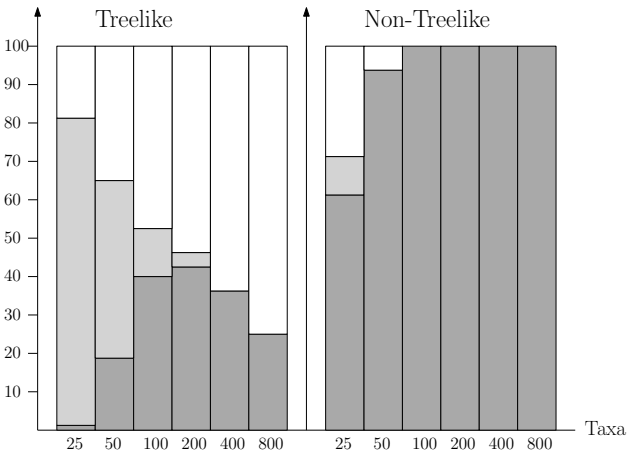

Figure 4: The bar chart shows the number of distance matrices (out of 100) for which the ME score of the phylogenetic tree produced for NNet was equal (light gray), smaller (dark gray) or larger (white) than the ME score of the tree produced by FastME.

Taxa	NNet	FastME	TSP	Random
	Treelike			
25	0.925	0.925	0.931	1.234
50	1.689	1.689	1.701	2.574
100	3.063	3.063	3.084	5.493
200	5.532	5.532	5.574	11.63
400	9.720	9.720	9.786	24.32
800	16.56	16.56	16.66	51.20
	Non-Treelike			
25	2.923	2.940	3.083	4.787
50	4.844	5.177	5.367	9.332
100	8.439	9.108	9.862	18.60
200	15.09	16.33	18.57	36.92
400	27.61	29.79	35.32	73.65
800	51.28	55.07	67.42	147.6

Figure 5: The average ME scores of the trees obtained with NNet, FastME, TSP and random orderings for treelike and non-treelike input data.

from the resulting alignment using the so-called Kimura 2 parameter model. To simulate random distance matrices we created symmetric matrices with zero's on the diagonal and random values between 0 and 1 in the remaining entries.

To generate circular split systems we tried various approaches. In particular, for each distance matrix we used NeighborNet (NNet) as well as its Traveling Salesman variant (TSP) [19] and, for comparison purposes, we also generated random orderings of X. Once an ordering of X was obtained, we computed a restricted ME-tree for the circular split system consisting of all possible splits that fit on the ordering, as described in Section 4.

In Figure 4 we present the results of our experiments using NNet to construct the ordering of X. Our results indicate that NNet seems to be capturing relevant splits for both treelike and for non-treelike data. For the tree-like data, FastME appears to perform somewhat better for larger numbers of taxa, but this trend is reversed for the non-treelike data. We also found that random orderings and orderings produced by TSP tended to be a lot worse than those produced by NNet, especially for larger numbers of taxa (not shown in Figure 4). The average ME scores of the trees generated using NNet, FastME, TSP and random orderings are shown in Figure 5. Interestingly, for treelike data, the average scores of trees generated using NNet and FastME coincide on the first 4 digits. In contrast, for non-treelike data, there is a noticeable difference in the average scores for these two methods.

6. Discussion

We have presented an efficient algorithm for finding a restricted ME tree in a circular split system which improves on the run time of a more general algorithm presented in [4]. We have also seen that the restricted ME trees obtained in split systems generated by NeighborNet compare favorably with the ones produced by FastME. This is of some interest since the split systems generated by NeighborNet only represent a tiny fraction of the total number of all possible splits ($\binom{n}{2}$ vs. $2^{n-1}-1$ on a set of size n). In addition, our computational experiments indicate that NeighborNet produces an ordering that is better at capturing splits relevant to building ME trees compared with other methods (such as the TSP ordering [19]). It could be of interest to better understand why this is the case and to see if there may be other ways to generate even better orderings.

In phylogenetics there are criteria other than ME that are commonly used to construct phylogenetic trees. For example, the Balanced Minimum Evolution (BME) criterion is also used for constructing trees from distances (cf. [6]). It would be interesting to know whether or not a restricted BME tree can be efficiently constructed for a circular split system, and whether a similar type of approach might work for other criteria such as likelihood [8]. In this regards, it might be useful to also consider searching in different

types of split systems (such as weakly compatible split systems [2]), and also to consider different ways to generate such split systems.

Finally, note that in this paper we have only considered unrooted trees, and so it could be of interest to develop restricted approaches for rooted trees. In this case it would be appropriate to consider special classes of cluster systems rather than split systems (cf. [3]).

Acknowledgments
We would like to thank the three reviewers for their helpful comments.

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

[2] H.-J. Bandelt and A. Dress. Split decomposition: A new and useful approach to phylogenetic analysis of distance data. *Molecular Phylogenetics and Evolution*, 1:242–252, 1992.

[3] D. Bryant. Hunting for trees in binary character sets: Efficient algorithms for extraction, enumeration and optimization. *Journal of Computational Biology*, 3:275–288, 1996.

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

[5] D. Bryant and V. Moulton. Neighbor-net: An agglomerative method for the construction of phylogenetic networks. *Molecular Biology and Evolution*, 21:255–265, 2004.

[6] D. Catanzaro. The minimum evolution problem: Overview and classification. *Networks*, 53:112–125, 2009.

[7] D. Catanzaro. Estimating phylogenies from molecular data. In R. Bruni (ed.), *Mathematical approaches to polymer sequence analysis and related problems*, Springer, New York, 149–176, 2011.

[8] B. Chor and T. Tuller. Finding a maximum likelihood tree is hard. *Journal of the ACM*, 53:722–744, 2006.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, Cambridge, MA, 2009.

[10] W. Day and D. Sankoff. Computational complexity of inferring phylogenies by compatibility. *Systematic Biology*, 35:224–229, 1986.

[11] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 9:687–705, 2002.

[12] A. Dress and D. Huson. Constructing split graphs. *IEEE Transactions on Computational Biology and Bioinformatics*, 1:109–115, 2004.

[13] M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13:155–179, 1995.

[14] T. Gonzalez. Simple algorithms for the on-line multidimensional dictionary and related problems. *Algorithmica*, 28:255–267, 2000.

[15] B. Holland, K. Huber, A. Dress, and V. Moulton. Delta plots: A tool for analyzing phylogenetic distance data. *Molecular Biology and Evolution*, 19:2051–2059, 2002.

[16] L. van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110:1037–1043, 2010.

[17] I. Kanj, L. Nakhleh, C. Than, and G. Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401:153–164, 2008.

[18] P. Lemey, M. Salemi, and A. Vandamme. *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing.* Cambridge University Press, Cambridge, UK, 2009.

[19] D. Levy and L. Pachter. The neighbor-net algorithm. *Advances in Applied Mathemathics*, 47:240–258, 2007.

[20] V. Moulton and A. Spillner. Optimal algorithms for computing edge weights in planar split networks. *Journal of Applied Mathematics and Computing*, 39:1–13, 2012.

[21] D. Penny and M. Hendy. TurboTree: A fast algorithm for minimal trees. *Computer Applications in the Biosciences*, 3:183–187, 1987.

[22] H. Prasanna and M. Rai. Detection and frequency of recombination in tomato-infecting begomoviruses of South and Southeast Asia. *Virology Journal*, 4:111, 2007.

[23] A. Rzhetsky and M. Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10:1073–1095, 1993.

[24] C. Semple and M. Steel. *Phylogenetics.* Oxford University Press, Oxford, UK, 2003.