A sub-cubic time algorithm for computing the quartet distance between two general trees

Thomas Mailund, Jesper Nielsen and Christian N.S. Pedersen
Bioinformatics Research Centre (BiRC), Aarhus University
C. F. Møllers Allé 8, DK-8000 Århus C, Denmark
Email: {mailund,jn,cstorm}@birc.au.dk

Abstract

We derive a sub-cubic time and space algorithm for computing the quartet distance between a pair of general trees, i.e. trees where inner nodes can have any degree ≥ 3. The time and space complexity of our algorithm is sub-cubic in the number of leaves and does not depend on the degree of the inner nodes. This makes it the fastest algorithm so far for computing the quartet distance between general trees independent of the degree of the inner nodes.

1. Introduction

The evolutionary relationship between a set of species is conveniently described as a tree, where the leaves represent the species and the inner nodes speciation events. Using different inference methods to infer such trees from biological data, or using different biological data from the same set of species, often yield slightly different trees. To study such differences in a systematic manner, one must be able to quantify differences between evolutionary trees using well-defined and efficient methods. One approach for this is to define a distance measure between trees and compare two trees by computing this distance. Several distance measures have been proposed, e.g. the symmetric difference metric [8], the nearest-neighbour interchange metric [12], the subtree transfer distance [1], the Robinson and Foulds distance [9], and the quartet distance [7]. Each distance measure has different properties and reflects different aspects of biology.

For an evolutionary tree, the *quartet topology* of four species is determined by the minimal topological subtree containing the four species. The four possible quartet topologies of four species are shown in Fig. 1. Given two evolutionary trees on the same set of n species, the *quartet distance* between them is the number of sets of four species for which the quartet topologies differ in the two trees.

Most previous work has focused on comparing *binary* trees and therefore avoided star quartets. Steel and Penny in [10] developed an algorithm for computing the quartet distance in time $O(n^3)$. Bryant *et al.* in [3] improved this result with an algorithm that computes the quartet distance in time $O(n^2)$. Brodal *et al.*, in [2], presented the currently best known algorithm that algorithm the computes the quartet distance in time $O(n \log n)$.

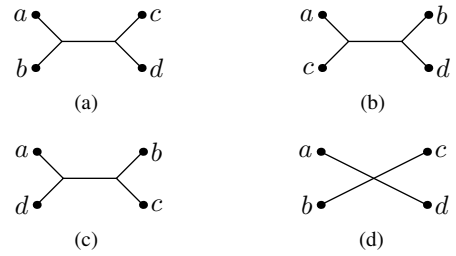


Figure 1. The four possible quartet topologies of species a, b, c, and d. Topologies (a): $ab|cd$, (b): $ac|bd$, and (c): $ad|bc$ are *butterfly* quartets, while topology (d): $\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}$, is a *star* quartet. For binary trees, only the butterfly quartets are possible.

Recently, we have developed algorithms for computing the quartet distance between two trees of *arbitrary* degrees, i.e. trees that can contain star quartets. In [4] we developed two algorithms: the first algorithm runs in time $O(n^3)$ and space $O(n^2)$—and is thus independent of the degree of the inner nodes—the second in time $O(n^2d^2)$ and space $O(n^2)$, where d is the maximal degree of inner nodes in the trees—and thus depend on the degree of the nodes. The $O(n^2d^2)$ was later improved to $O(n^2d)$ [5], and by taking an approach similar to the Brodal *et al.* [2] $O(n \log n)$ we developed a sub-quadratic algorithm in terms of n but at a significant cost in terms of d: $O(d^9 n \log n)$ [11].

In this paper we develop an $O(n^{2+\alpha})$, where $\alpha = \frac{\omega-1}{2}$ and $O(n^\omega)$ is the time it takes to multiply two $n \times n$ matrices. Using the Coppersmith-Winograd [6] algorithm, where $\omega = 2.376$, this yiels a running time of $O(n^{2.688})$. The running time is thus independent of the degrees of the inner nodes of the input trees, and this is the first sub-cubic time algorithm with this property.

2. Background

The quartet distance between two trees is the number of quartets where the quartet topology differ between the two trees, i.e. the number of quartets where one tree has the

star topology and the other a butterfly topology, plus the number of quartets where the trees have a different butterfly topology. As observed in [4], the former—where one tree has the star topology and the other a butterfly topology—can be expressed in terms of the total number of butterflies in the two trees, the number of shared butterflies and the number of different butterflies: For trees T and T', the number of different topologies due to one being a star and the other a quartet, $\text{diff}_S(T, T')$, is given by

$$\begin{aligned}\text{diff}_S(T, T') = B + B' \\ -2\left(\text{shared}_B(T, T') + \text{diff}_B(T, T')\right)\ ,\end{aligned} \tag{1}$$

where B is the number of butterflies in T, B' the number of butterflies in T', $\text{shared}_B(T, T')$ the number of quartets with the same butterfly topology in T and T' and $\text{diff}_B(T, T')$ the number of quartets with different butterfly topologies in T and T'. Thus the quartet distance between T and T' is given by the expression

$$\begin{aligned}\text{qdist}(T, T') = B + B' \\ -2\,\text{shared}_B(T, T') - \text{diff}_B(T, T')\ .\end{aligned} \tag{2}$$

Since, $B = \text{shared}_B(T, T)$ and $B' = \text{shared}_B(T', T')$, an algorithm for computing $\text{shared}_B(T, T')$ and $\text{diff}_B(T, T')$ gives an algorithm for computing the quartet distance between T and T'.

Our approach to counting the shared and different quartets is based on *directed quartets* and *claims* [2], [4]. An (undirected) butterfly quartet topology, $ab|cd$ induces two directed quartet topologies $ab \to cd$ and $ab \leftarrow cd$, by the orientation of the middle edge of the topology, as shown in Fig. 2. There are twice as many directed butterflies as undirected, and the number of shared (different) butterflies can be counted as

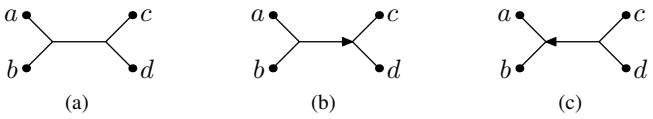


Figure 2. An undirected quartet topology, (a), and the two directed quartet topologies, (b) and (c), it induces.

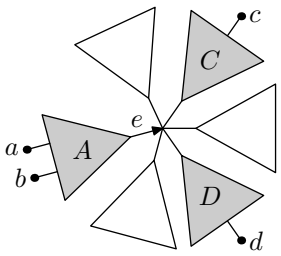


Figure 3. A *claim* $A \xrightarrow{e} (C, D)$. The claim $A \xrightarrow{e} (C, D)$ claims all ordered butterflies $ab \to cd$ where $a, b \in A$ and $c \in C$,$d \in D$ where C and D are two *different* subtrees in front of e.

half the number of shared (different) directed butterflies. To each directed quartet, $ab \to cd$, we can uniquely associate a directed edge, e so that a and b are leaves in the tree behind e, and c and d are leaves in *different* subtrees of the root of the tree in front of e, see Fig. 3. We call such a tree substructure, consisting of a directed edge e with a subtree, A behind e and two distinct subtrees, C and D, in front of e a *claim*, written $A \xrightarrow{e} (C, D)$, and say that the edge e *claims* the directed quartet $ab \to cd$, and we also say that an edge e claims an undirected quartet $ab|cd$ if it claims one of its directed quartets. Each (undirected) butterfly quartet defines exactly two directed butterfly quartets, and each directed quartet is claimed by exactly one directed edge; considering each claim and implicitly each directed butterfly claimed by the claim, we can examine each directed butterfly in a tree, or each undirected butterfly twice.

The crux of the algorithm is to consider each pair of claims, one from each tree, and for each such pair count the number of shared and different directed butterflies claimed in the two trees. Dividing these counts by two gives us $\text{shared}_B(T, T')$ and $\text{diff}_B(T, T')$.

3. A sub-cubic time and space algorithm

3.1. Preprocessing

Before counting shared and different butterflies, we calculate a number of values in two preprocessing steps. First, we calculate a matrix that for each pairs of subtrees $F \in T$ and $G \in T'$ stores the number of leaves in both trees, $|F \cap G|$. This can be achieved in time and space $O(n^2)$ [3].

Next, for each pair of inner nodes, $v \in T, v' \in T'$ with sub-trees $F_i, i = 1, \ldots, d_v$ and $G_j, j = 1, \ldots, d_{v'}$, respectively, we calculate a matrix, I, such that $I[i, j] = |F_i \cap G_j|$, and we calculate vectors of its row and column sums, and the total sum of its entries:

$$R[i] = \sum_{j=1}^{d_{v'}} I[i, j] \tag{3}$$

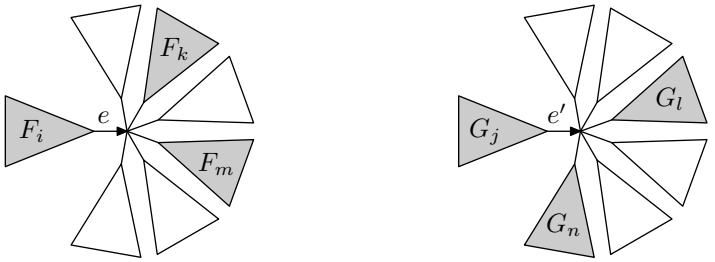
$$C[j] = \sum_{i=1}^{d_v} I[i, j] \tag{4}$$

$$M = \sum_{i=1}^{d_v} \sum_{j=1}^{d_{v'}} I[i, j] \tag{5}$$

Inspired by the sums (31) – (34) in the appendix, we calculate a matrix I', vectors of its row and column sums, the total sum of its entries, and some further values

$$I'[i, j] = I[i, j](M - R[i] - C[j] + I[i, j]) \tag{6}$$

$$R'[i] = \sum_{j=1}^{d_{v'}} I'[i, j] \tag{7}$$



(a) Edge e in T and the claim $F_i \xrightarrow{e} (F_k, F_m)$.

(b) Edge e' in T' and the claim $G_j \xrightarrow{e'} (G_l, G_n)$.

Figure 4. A pair of inner edges, $e \in T$, $e' \in T'$, where F_i (G_j) is the sub-tree behind e (e') and $F_k, k \neq i$ ($G_l, l \neq j$) the remaining subtrees of the node pointed to by e (e'). Highlighted are two claims, one from each tree.

$$C'[j] = \sum_{i=1}^{d_v} I'[i,j] \tag{8}$$

$$M' = \sum_{i=1}^{d_v} \sum_{j=1}^{d_{v'}} I'[i,j] \tag{9}$$

$$R''[i] = \sum_{j=1}^{d_{v'}} I[i,j](C[j] - I[i,j]) \tag{10}$$

$$C''[j] = \sum_{i=1}^{d_v} I[i,j](R[i] - I[i,j]) \tag{11}$$

$$R'''[i] = \sum_{j=1}^{d_{v'}} I[i,j]^2 \tag{12}$$

$$C'''[j] = \sum_{i=1}^{d_v} I[i,j]^2 \tag{13}$$

Calculating the values in Eq. (3) – (13) can be done in time $O(d_v d_{v'})$ for each pair of inner nodes, giving a total time of $O\big(\sum_{v\in T}\sum_{v'\in T'} d_v d_{v'}\big) = O\big((\sum_{v\in T} d_v)(\sum_{v'\in T'} d_{v'})\big) = O(n^2)$.

Finally, we need to calculate the following values:

$$I'''[i,j] = \sum_{k=1,k\neq i}^{d_v} \sum_{l=1,l\neq j}^{d_{v'}} I[i,l]I[k,j]I[k,l] \tag{14}$$

which takes time $O(d_v^2 d_{v'}^2)$ for each pair of inner nodes, giving a total time of $O(n^4)$, if done naively. However, as we show in the appendix, the values in Eq. (14) can be calculated faster by computing either $I_1'' = II^T$ and $I_1''' = I_1''I$, or $I_2'' = I^T I$ and $I_2''' = II_2''$, depending on which pair of matrices is fastest to compute, where I is the $d_v \times d_{v'}$ matrix defined above. We thus calculate either Eq. (15) and (16), or Eq. (17) and (18), depending on which pair is fastest to calculate.

$$I_1''[i,k] = \sum_{j=1}^{d_{v'}} I[i,j]I[k,j] \tag{15}$$

$$I_1'''[i,j] = \sum_{k=1}^{d_v} I[k,j]I_1''[i,k] \tag{16}$$

$$I_2''[j,l] = \sum_{i=1}^{d_v} I[i,j]I[i,l] \tag{17}$$

$$I_2'''[i,j] = \sum_{l=1}^{d_{v'}} I[i,l]I_2''[l,k] \tag{18}$$

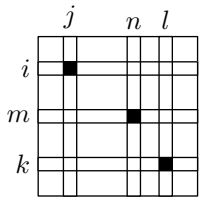
Calculating the values in Eq. (15) and (16) takes time $O(\max(d_v, d_{v'})^\omega)$ with $\omega = 2.376$ if padding the matrices to become square and using the Coppersmith-Winograd algorithm [6] for matrix multiplication, or time $O(d_v^2 d_{v'})$ if using naive matrix multiplication. Similarly, calculating the values in Eq. (17) and (18) takes time $O(\max(d_v, d_{v'})^\omega)$ or $O(d_v d_{v'}^2)$. Computing either I_1'' and I_1''', or I_2'' and I_2''', thus takes time $O(\min(\max(d_v, d_{v'})^\omega, d_v^2 d_{v'}, d_v d_{v'}^2))$.

3.2. Counting shared butterfly topologies

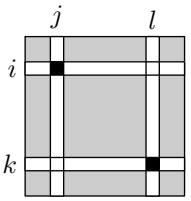
For each pair of inner edges, $e \in T$ and $e' \in T'$, see Fig. 4, we count the directed butterflies claimed by both e and e'. These are all on the form $ab \to cd$, where $a, b \in F_i \cap G_j$, $c \in F_k \cap G_l$ and $d \in F_m \cap G_n$ for some claims, $F_i \xrightarrow{e} (F_k, F_m)$ and $G_j \xrightarrow{e'} (G_l, G_n)$, of e and e'. The total number of directed butterflies common for both e and e' is therefore given by the expression

$$\frac{1}{4}\binom{|F_i \cap G_j|}{2} \sum_{k\neq i}\sum_{l\neq j} |F_k \cap G_l| \sum_{m\neq i,k}\sum_{n\neq j,l} |F_m \cap G_n| \tag{19}$$

or the sum of $\frac{1}{4}\binom{I[i,j]}{2} \cdot I[k,l] \cdot I[m,n]$ for all distinct entries in I but fixed (i,j), see Fig. 5(a). We divide by four since we count each quartet four times, due to symmetry between m and k and between n and l.



(a) The choices of entries in I summed over in eq. (19).



(b) Implicit representation of the inner sum in eq. (19).

Figure 5. Graphical illustration of the shared quartet expression, eq. (19). On the left, the matrix entries summed over are explicitly shown. On the right, the inner sum is implicitly shown. The sum of the greyed entries can be computed in constant time.

Notice, however, that the inner sum is simply the total sum of entries, M, except for the rows i and k and columns j and l, see Fig. 5(b). Using

$$\sum_{m\neq i,k}\sum_{n\neq j,l}|F_m\cap G_n| = \tag{20}$$
$$M-\sum_{q=i,k}R[q]-\sum_{r=j,l}C[r]+\sum_{q=i,k}\sum_{r=j,l}I[q,r]$$

and the precomputed values we can, as shown in the appendix, rewrite the expression in Eq. (19) to

$$\frac{1}{4}\binom{I[i,j]}{2}\Big(M'-R'[i]-C'[j]+I'[i,j]+ \tag{21}$$
$$(I[i,j]-R[i]-C[j])(M-R[i]-C[j]+I[i,j])+$$
$$R''[i]-I[i,j](C[j]-I[i,j])+$$
$$C''[j]-I[i,j](R[j]-I[i,j])\Big)$$

which can be computed in time $O(1)$, and thus we can compute all shared directed butterflies in total time $O(n^2)$. Dividing by two, we get the number of shared undirected butterflies.

3.3. Counting different butterfly topologies

Counting the number of different butterflies in the two trees is done similar to counting the number of shared butterflies. As before, we consider a pair of inner edges, $e\in T$ and $e'\in T'$. The quartets claimed by both e and e', but with different butterfly topology, are on the form $a\in F_i\cap G_j$, $b\in F_i\cap G_l$, $c\in F_k\cap G_j$ and $d\in F_m\cap G_n$ for some claims $F_i\xrightarrow{e}(F_k,F_m)$ and $G_j\xrightarrow{e'}(G_l,G_n)$. The number of butterflies claimed by both e and e' but with different topology is therefore given by

$$|F_i\cap G_j|\sum_{k\neq i}\sum_{l\neq j}|F_i\cap G_l||F_k\cap G_j|\sum_{m\neq i,k}\sum_{n\neq j,l}|F_m\cap G_n| \tag{22}$$

or the sum of $I[i,j]\cdot I[i,l]\cdot I[k,j]\cdot I[m,n]$ for all distinct entries in I but fixed (i,j), see Fig. 6(a). In this case there is no need to divide by any normalizing constant, since there are no symmetries between k and m or between l and n.

As before, the inner sum can be expressed as in Eq. (20), and using the precomputed values we can, as shown in the appendix, rewrite the expression in Eq. (22) as

$$I[i,j]\Big(\tag{23}$$
$$(M-R[i]-C[j]+I[i,j])(R[i]-I[i,j])(C[j]-I[i,j])+$$
$$(R[i]-I[i,j])(I[i,j](R[i]-I[i,j])-C''[j])+$$
$$(C[j]-I[i,j])(I[i,j](C[j]-I[i,j])-R''[i])+$$
$$I_1'''[i,j]-I[i,j]I_1''[i,i]-I[i,j](C'''[j]-I[i,j]^2)\Big)$$

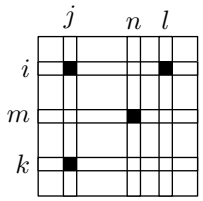
or

$$I[i,j]\Big(\tag{24}$$
$$(M-R[i]-C[j]+I[i,j])(R[i]-I[i,j])(C[j]-I[i,j])+$$
$$(R[i]-I[i,j])(I[i,j](R[i]-I[i,j])-C''[j])+$$
$$(C[j]-I[i,j])(I[i,j](C[j]-I[i,j])-R''[i])+$$
$$I_2'''[i,j]-I[i,j]I_2''[j,j]-I[i,j](R'''[i]-I[i,j]^2)\Big)$$

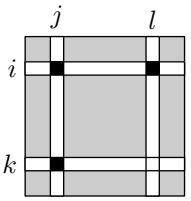
depending on whether we have precomputed I_1'' and I_1''', or I_2'' and I_2'''. We can thus compute Eq. (22) in time $O(1)$ for each pair of inner edges $e\in T$ and $e'\in T'$ giving a total time of $O(n^2)$ to compute different directed, and thus different undirected, butterfly topologies in the two trees.

4. Time analysis

The running time of the algorithm is dominated by the time $O(\min(\max(d_v,d_{v'})^{2.376},d_v^2d_{v'},d_vd_{v'}^2))$ it takes to compute either I_1'' and I_1''', or I_2'' and I_2''', for each pair of nodes $v\in T$ and $v'\in T'$. Let $O(n^\omega)$ be the time it takes to multiply two $n\times n$ matrices. The running of our algorithm is $O(n^{\alpha+2})$, where $\alpha=\frac{\omega-1}{2}$. Using the Coppersmith-Winograd algorithm [6] for matrix multiplication, where $\omega=2.376$, this yields a running time of $O(n^{2.688})$. To show this, we split the pairs of nodes $(v,v')\in T\times T'$ into four



(a) The choices of entries in I summed over in eq. (22).



(b) Implicit representation of the inner sum in eq. (22).

Figure 6. Graphical illustration of the different quartet expression, eq. (22). On the left, the matrix entries summed over are explicitly shown. On the right, the inner sum is implicitly shown. The sum of the greyed entries can be computed in constant time.

disjoint sets:

$$P_1 = \{(v, v') \in T \times T' | d_v \leq d_{v'} \wedge d_v \leq d_{v'}^\alpha\} \quad (25)$$
$$P_2 = \{(v, v') \in T \times T' | d_v \leq d_{v'} \wedge d_v > d_{v'}^\alpha\} \quad (26)$$
$$P_3 = \{(v, v') \in T \times T' | d_v > d_{v'} \wedge d_v^\alpha \geq d_{v'}\} \quad (27)$$
$$P_4 = \{(v, v') \in T \times T' | d_v > d_{v'} \wedge d_v^\alpha < d_{v'}\} \quad (28)$$

which can be analysed independently. Preprocessing for all pairs of nodes in P_1, where $d_v \leq d_{v'}^\alpha \leq n^\alpha$, takes time:

$$\begin{aligned}
&O\Big(\sum_{(v,v')\in P_1} \min(\max(d_v, d_{v'})^\omega, d_v^2 d_{v'}, d_v d_{v'}^2)\Big) \quad (29)\\
&= O\Big(\sum_{(v,v')\in P_1} d_v^2 d_{v'}\Big)\\
&\leq O\Big(\sum_{v\in T|d_v\leq n^\alpha} \sum_{v'\in T'} d_v^2 d_{v'}\Big)\\
&= O\Big(\sum_{v\in T|d_v\leq n^\alpha} d_v^2 \sum_{v'\in T'} d_{v'}\Big)\\
&= O\Big(\sum_{v\in T|d_v\leq n^\alpha} d_v^2 n\Big)\\
&= O\Big(\sum_{v\in T|d_v\leq n^\alpha} d_v d_v n\Big)\\
&\leq O\Big(\sum_{v\in T|d_v\leq n^\alpha} d_v n^\alpha n\Big)\\
&= O\big(n n^\alpha n\big)\\
&= O\big(n^{2+\alpha}\big)
\end{aligned}$$

Preprocessing for all pairs of nodes in P_2, where $d_{v'}^\alpha < d_v \leq n$, takes time:

$$\begin{aligned}
&O\Big(\sum_{(v,v')\in P_2} \min(\max(d_v, d_{v'})^\omega, d_v^2 d_{v'}, d_v d_{v'}^2)\Big) \quad (30)\\
&= O\Big(\sum_{(v,v')\in P_2} d_{v'}^\omega\Big)\\
&= O\Big(\sum_{v'\in T'} \sum_{v\in T|d_{v'}^\alpha < d_v \leq d'_v} d_{v'}^\omega\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'}^\omega \sum_{v\in T|d_{v'}^\alpha < d_v \leq d_{v'}} 1\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'}^\omega \sum_{v\in T|d_{v'}^\alpha < d_v \leq d_{v'}} \frac{d_v}{d_v}\Big)\\
&\leq O\Big(\sum_{v'\in T'} d_{v'}^\omega \sum_{v\in T|d_{v'}^\alpha < d_v \leq d_{v'}} \frac{d_v}{d_{v'}^\alpha}\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'}^{\omega-\alpha} \sum_{v\in T|d_{v'}^\alpha < d_v \leq d_{v'}} d_v\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'}^{\omega-\alpha} n\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'} d_{v'}^{\omega-\alpha-1} n\Big)\\
&\leq O\Big(\sum_{v'\in T'} d_{v'} n^{\omega-\alpha-1} n\Big)\\
&= O\Big(\sum_{v'\in T'} d_{v'} n^{\omega-\alpha}\Big)\\
&= O\big(n n^{\omega-\alpha}\big)\\
&= O\big(n^{1+\omega-\frac{\omega-1}{2}}\big)\\
&= O\big(n^{2+\frac{\omega-1}{2}}\big)\\
&= O\big(n^{2+\alpha}\big)
\end{aligned}$$

Since the cases P_3 and P_4 are symmetric to P_1 and P_2 respectively, the total running time is $O(n^{2+\alpha})$.

Acknowledgements

We are grateful to Chris Christiansen, Martin Randers and Martin S. Stissing for many fruitful discussions about the quartet distance.

References

[1] B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–13, 2001.

[2] G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen. Computing the quartet distance between evolutionary trees in time $O(n \log n)$. *Algorithmica*, 38:377–395, 2003.

[3] D. Bryant, J. Tsang, P. E. Kearney, and M. Li. Computing the quartet distance between evolutionary trees. In *Proceedings of the 11th Annual Symposium on Discrete Algorithms (SODA)*, pages 285–286, 2000.

[4] C. Christiansen, T. Mailund, C. N. S. Pedersen, and M. Randers. Algorithms for computing the quartet distance between trees of arbitrary degree. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI)*, volume 3692 of *Lecture Notes in Bioinformatics (LNBI)*, pages 77–88. Springer-Verlag, 2005.

[5] C. Christiansen, T. Mailund, C. N. S. Pedersen, M. Randers, and M. S. Stissing. Fast calculation of the quartet distance between trees of arbitrary degrees. *Algorithms for Molecular Biology*, 1, 2006.

[6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–281, 1990.

[7] G. Estabrook, F. McMorris, and C. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Zool.*, 34:193–200, 1985.

[8] D. F. Robinson and L. R. Foulds. Comparison of weighted labelled trees. In *Combinatorial mathematics, VI (Proc. 6th Austral. Conf)*, Lecture Notes in Mathematics, pages 119–126. Springer, 1979.

[9] D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[10] M. Steel and D. Penny. Distribution of tree comparison metrics–some new results. *Syst. Biol.*, 42(2):126–141, 1993.

[11] M. Stissing, C. N. S. Pedersen, T. Mailund, G. S. Brodal, and R. Fagerberg. Computing the quartet distance between evolutionary trees of bounded degree. In Sankoff, D and Wang, L and Chin, F, editor, *Proceedings of the 5th Asia-Pacific Bioinfomatics Conference 2007*, volume 5 of *Series on Advances in Bioinformatics and Computational Biology*, pages 101–110, 2007.

[12] M. S. Waterman and T. F. Smith. On the similarity of dendrograms. *Journal of Theoretical Biology*, 73:789–800, 1978.

Appendix

In this appendix, we give the details of the rewritings that yield Eq. (21), (23) and (24). Expanding Eq. (20) into Eq. (19) yields:

$$\begin{aligned}
&\frac{1}{4}\binom{|F_i \cap G_j|}{2} \sum_{k\neq i}\sum_{l\neq j} |F_k \cap G_l| \sum_{m\neq i,k}\sum_{n\neq j,l} |F_m \cap G_n| \\
&= \frac{1}{4}\binom{I[i,j]}{2} \Big(\\
&\big(\sum_{k\neq i}\sum_{l\neq j} I[k,l](M - R[k] - C[l] + I[k,l])\big) + && (31)\\
&\big(\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[i,j] - R[i] - C[j])\big) + && (32)\\
&\big(\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[i,l])\big) + && (33)\\
&\big(\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[k,j])\big)\Big) && (34)
\end{aligned}$$

Using the values introduced in Sec. 3.1, we can rewrite the sums (31) – (34) as:

$$\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[k,l](M - R[k] - C[l] + I[k,l]) \\
&= \sum_{k\neq i}\sum_{l\neq j} I'[k,l] \\
&= M' - R'[i] - C'[j] + I'[i,j] && (35)
\end{aligned}$$

$$\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[i,j] - R[i] - C[j]) \\
&= (I[i,j] - R[i] - C[j]) \sum_{k\neq i}\sum_{l\neq j} I[k,l] \\
&= (I[i,j] - R[i] - C[j])(M - R[i] - C[j] + I[i,j]) && (36)
\end{aligned}$$

$$\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[i,l]) \\
&= \sum_{l\neq j} I[i,l] \sum_{k\neq i} I[k,l] \\
&= \sum_{l\neq j} I[i,l](C[l] - I[i,l]) \\
&= R''[i] - I[i,j](C[j] - I[i,j]) && (37)
\end{aligned}$$

$$\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[k,l](I[k,j]) \\
&= \sum_{k\neq i} I[k,j] \sum_{l\neq j} I[k,l] \\
&= \sum_{k\neq i} I[k,j](R[k] - I[k,j]) \\
&= C''[j] - I[i,j](R[j] - I[i,j]) && (38)
\end{aligned}$$

which gives that Eq. (19) can be rewritten as Eq. (21).

Expanding Eq. (20) into Eq. (22) yields:

$$
\begin{aligned}
&|F_i \cap G_j| \sum_{k\neq i}\sum_{l\neq j} |F_i \cap G_l||F_k \cap G_j| \sum_{m\neq i,k}\sum_{n\neq j,l} |F_m \cap G_n| \\
&= I[i,j]\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j]\sum_{m\neq i,k}\sum_{n\neq j,l} I[m,n] \\
&= I[i,j]\Big(\\
&\quad \big(\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](M - R[i] - C[j] + I[i,j])\big) + \qquad (39)\\
&\quad \big(\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](I[k,j] - R[k])\big) + \qquad (40)\\
&\quad \big(\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](I[i,l] - C[l])\big) + \qquad (41)\\
&\quad \big(\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](I[k,l])\big)\Big) \qquad (42)
\end{aligned}
$$

Using the values introduced in Sec. 3.1, we can rewrite the sums (39) – (41) as:

$$
\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](M - R[i] - C[j] + I[i,j]) \\
&= (M - R[i] - C[j] + I[i,j])\cdot \\
&\qquad (R[i] - I[i,j])(C[j] - I[i,j]) \qquad (43)
\end{aligned}
$$

$$
\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](I[k,j] - R[k]) \\
&= \sum_{k\neq i} I[k,j](I[k,j] - R[k])\sum_{l\neq j} I[i,l] \\
&= \sum_{k\neq i} I[k,j](I[k,j] - R[k])(R[i] - I[i,j]) \\
&= (R[i] - I[i,j])\sum_{k\neq i} I[k,j](I[k,j] - R[k]) \\
&= (R[i] - I[i,j])(I[i,j](R[i] - I[i,j]) - C''[j]) \qquad (44)
\end{aligned}
$$

$$
\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j](I[i,l] - C[l]) \\
&= \sum_{l\neq j} I[i,l](I[i,l] - C[l])\sum_{k\neq i} I[k,j] \\
&= \sum_{l\neq j} I[i,l](I[i,l] - C[l])(C[j] - I[i,j]) \\
&= (C[j] - I[i,j])\sum_{l\neq j} I[i,l](I[i,l] - C[l]) \\
&= (C[j] - I[i,j])(I[i,j](C[j] - I[i,j]) - R''[i]) \qquad (45)
\end{aligned}
$$

The sum (42) is $I'''[i,j]$ introduced in Sec. 3.1. This value can be computed efficiently using either I_1'' and I_1''', or I_2'' and I_2''', also introduced in Sec. 3.1. If I_1'' and I_1''' are available, we can compute $I'''[i,j]$ as:

$$
\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j]I[k,l] \\
&= \sum_{k\neq i} I[k,j]\sum_{l\neq j} I[i,l]I[k,l] \\
&= \sum_{k\neq i} I[k,j](I_1''[i,k] - I[i,j]I[k,j]) \\
&= (\sum_{k\neq i} I[k,j]I_1''[i,k]) - (\sum_{k\neq i} I[k,j]^2 I[i,j]) \\
&= I_1'''[i,j] - I[i,j]I_1''[i,i] - (I[i,j]\sum_{k\neq i} I[k,j]^2) \\
&= I_1'''[i,j] - I[i,j]I_1''[i,i] - I[i,j](C'''[j] - I[i,j]^2) \qquad (46)
\end{aligned}
$$

Replacing the sums (39) – (42) with Eq. (43) – (46) shows that Eq. (22) can be rewritten as Eq. (23).

If I_2'' and I_2''' are available, we can compute $I'''[i,j]$ as:

$$
\begin{aligned}
&\sum_{k\neq i}\sum_{l\neq j} I[i,l]I[k,j]I[k,l] \\
&= \sum_{l\neq j} I[i,l]\sum_{k\neq i} I[k,j]I[k,l] \\
&= \sum_{l\neq j} I[i,l](I_2''[j,l] - I[i,j]I[i,l]) \\
&= (\sum_{l\neq j} I[i,l]I_2''[j,j]) - (\sum_{l\neq j} I[i,l]^2 I[i,j]) \\
&= I_2'''[i,j] - I[i,j]I_2''[j,j] - (I[i,j]\sum_{l\neq j} I[i,l]^2 \\
&= I_2'''[i,j] - I[i,j]I_2''[j,j] - I[i,j](R'''[i] - I[i,j]^2) \qquad (47)
\end{aligned}
$$

Replacing the sums (39) – (42) with Eq. (43) – (45) and (47) shows that Eq. (22) can be rewritten as Eq. (24).