

# Quartet Cleaning: Improved Algorithms and Simulations

Vincent Berry<sup>1\*</sup>, Tao Jiang<sup>2\*\*</sup>, Paul Kearney<sup>3\*\*\*</sup>, Ming Li<sup>3†</sup>, and Todd Wareham<sup>2‡</sup>

<sup>1</sup> Département de Mathématiques, Université de Saint-Etienne

<sup>2</sup> Department of Computing and Software, McMaster University

<sup>3</sup> Department of Computer Science, University of Waterloo

<sup>4</sup> Department of Computing and Software, McMaster University

**Abstract.** A critical step in all quartet methods for constructing evolutionary trees is the inference of the topology for each set of four sequences (i.e. *quartet*). It is a well-known fact that all quartet topology inference methods make mistakes that result in the incorrect inference of quartet topology. These mistakes are called *quartet errors*. In this paper, two efficient algorithms for correcting bounded numbers of quartet errors are presented. These “quartet cleaning” algorithms are shown to be optimal in that no algorithm can correct more quartet errors. An extensive simulation study reveals that sets of quartet topologies inferred by three popular methods (Neighbor Joining [15], Ordinal Quartet [14] and Maximum Parsimony [10]) almost always contain quartet errors and that a large portion of these quartet errors are corrected by the quartet cleaning algorithms.

## 1 Introduction

The explosion in the amount of DNA sequence data now available [3] has made it possible for biologists to address important large scale evolutionary questions [11,12,19]. In the analysis of this data an evolutionary tree  $T$  that describes the evolutionary history of the set  $S$  of sequences involved is produced.  $T$  is modeled by an edge-weighted rooted tree where the leaves are labeled bijectively by sequences in  $S$ . Due to the large data sets involved, standard approaches for constructing evolutionary trees, such as maximum likelihood [9] and maximum parsimony [18], that exhaustively search the entire tree space are not feasible.

In recent years *quartet methods* for constructing evolutionary trees have received much attention in the computational biology community [1,2,4,8,14,17]. Given a quartet

\* Supported in part by ESPRIT LTR Project no. 20244 — ALCOM-IT. [vberry@univ-st-etienne.fr](mailto:vberry@univ-st-etienne.fr)

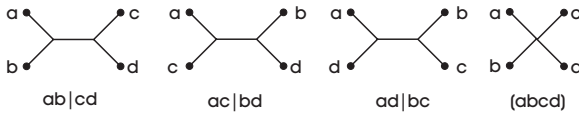
\*\* Supported in part by NSERC Research Grant OGP0046613 and a CGAT grant. [jiang@cas.mcmaster.ca](mailto:jiang@cas.mcmaster.ca)

\*\*\* Supported in part by NSERC Research Grant OGP217246-99 and a CITO grant. [pkearney@math.uwaterloo.ca](mailto:pkearney@math.uwaterloo.ca)

† Supported in part by NSERC Research Grant OGP0046506, CITO, a CGAT grant, and the Steacie Fellowship. [mli@math.uwaterloo.ca](mailto:mli@math.uwaterloo.ca)

‡ Supported in part by a CGAT grant. [harold@cas.mcmaster.ca](mailto:harold@cas.mcmaster.ca)

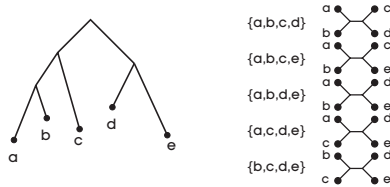
of sequences  $\{a, b, c, d\}$  and an evolutionary tree  $T$ , the *quartet topology* induced in  $T$  by  $\{a, b, c, d\}$  is the path structure connecting  $a, b, c$  and  $d$  in  $T$ . Given a quartet  $\{a, b, c, d\}$ , if the path in  $T$  connecting sequences  $a$  and  $b$  is disjoint from the path in  $T$  connecting sequences  $c$  and  $d$  then the quartet is said to be *resolved* and is denoted  $ab|cd$ . Otherwise, the quartet is said to be *unresolved* and is denoted  $(abcd)$ . The four possible quartet topologies that can be induced by a quartet are depicted in Fig. 1.



**Fig. 1.** The four quartet topologies for quartet  $\{a, b, c, d\}$ .

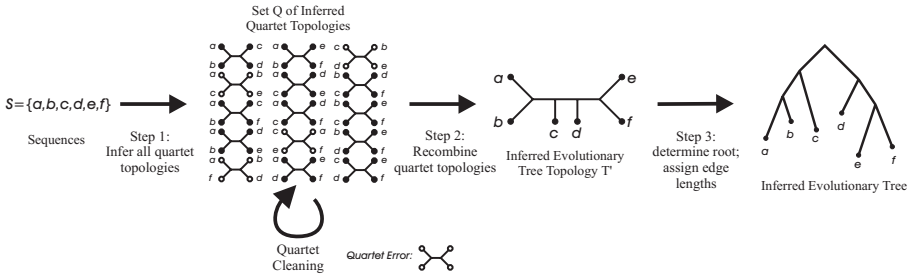
Quartet methods are based upon the fact that the topology of an evolutionary tree  $T$  (that is,  $T$  without its edge weights) is uniquely characterized by its set  $Q_T$  of induced quartet topologies [5] (see Fig. 2). This suggests the following three step process, referred to as the *quartet method paradigm*, for estimating an unknown evolutionary tree  $T$  for a set  $S$  of sequences:

1. For each quartet  $\{a, b, c, d\}$  of sequences in  $S$ , estimate the quartet topology induced by  $\{a, b, c, d\}$  in  $T$ . The procedure for producing this estimate is called a *quartet topology inference method*. Let  $Q$  be the set of  $\binom{n}{4}$  inferred quartet topologies.
2. The quartet topologies in  $Q$  are recombined to produce an estimate  $T'$  of  $T$ 's topology. The procedure for producing  $T'$  is called a *quartet recombination method*.
3.  $T'$  is rooted and edge weights determined.



**Fig. 2.** An evolutionary tree  $T$  and its set  $Q_T$  of induced quartet topologies.

The quartet method paradigm is illustrated in Fig. 3. There are many quartet topology inference methods including maximum parsimony [10], maximum likelihood [9], neighbor joining [15] and the ordinal quartet method [14]. The reader is directed to [18] for a good overview of these methods. Notice that computationally intensive methods such as maximum likelihood and maximum parsimony can be used to infer quartet topology but are infeasible when used to infer the entire tree topology. Existing quartet recombination methods include the  $Q^*$  method [2], short quartet method [8], and quartet puzzling [17] among others [1,13]. The third step in the quartet method is well-understood [4,18].



**Fig. 3.** The quartet method paradigm.

The algorithmic interest in the quartet method paradigm derives from the fact that quartet topology inference methods make mistakes, and so, the set  $Q$  of inferred quartet topologies contains *quartet errors*. The quartet  $\{a, b, c, d\}$  is a quartet error if  $ab|cd \in Q_T$  but  $ab|cd \notin Q$ . For example, in Fig. 3,  $\{a, b, c, e\}$  is a quartet error since  $ab|ce \in Q_T$  but  $ac|be \in Q$ . In this sense,  $Q$  is an estimate of  $Q_T$ . Consequently, the problem of recombining quartet topologies of  $Q$  to form an estimate  $T'$  of  $T$  is typically formulated as an optimization problem:

**Maximum Quartet Consistency (MQC)**

Instance: Set  $Q$  containing a quartet topology for each quartet of sequences in  $S$  and  $k \in \mathcal{N}$ .

Question: Is there an evolutionary tree  $T'$  labeled by  $S$  such that  $|Q_{T'} \cap Q| \geq k$ ?

Problem **MQC** is NP-hard if the input  $Q$  is a partial set of quartets, i.e., quartet topologies can be missing [16] (this proof also implies that this version of **MQC** is MAX-SNP hard). Though it was previously shown that **MQC** has a polynomial time approximation scheme [13], proving that **MQC** is NP-hard turns out to be more difficult since a complete set of quartets must be constructed<sup>1</sup>:

**Theorem 1.** *MQC is NP-hard.*

Clearly, the accuracy of the estimate  $T'$  depends almost entirely on the accuracy of  $Q$ . In fact, many quartet recombination methods are very sensitive to quartet errors in  $Q$ . For example, the  $Q^*$  method [2] and the Short Quartet Method [8] can fail to recover the true evolutionary tree even if there is only one quartet error in  $Q$ . Hence, although much effort has been directed towards the development of quartet recombination methods, of prior importance is the development of methods that improve the accuracy of inferred quartet topologies.

This paper presents research on the detection and correction of quartet errors in  $Q$ . Methods for detecting and correcting quartet errors are called “quartet cleaning” methods (see Fig. 3). The principal contributions of the research presented here are two quartet cleaning methods, namely *global edge cleaning* and *local vertex cleaning*, and

<sup>1</sup> Several proofs have been omitted due to space constraints. Please contact the authors for details.

an extensive simulation study that establishes both the need for these quartet cleaning algorithms and their applicability.

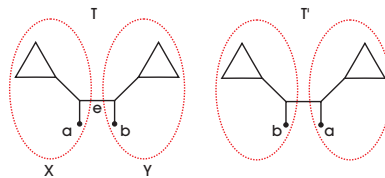
These results are described in more detail in Sect. 1.1 and contrasted with previous results in Sect. 1.2.

### 1.1 Terminology and Results

Let  $S$  be a set of sequences,  $Q$  be a set of quartet topologies and  $T$  the true evolutionary tree for  $S$ . In this paper it can be assumed that  $Q$  contains a quartet topology for each quartet taken from  $S$ . Several concepts must be introduced so that quartet cleaning can be discussed in detail.

An edge  $e$  in tree  $T$  induces the bipartition  $(A, B)$  if  $T - \{e\}$  consists of two trees where one is labeled by  $A$  and the other by  $B$ . This is denoted  $e = (A, B)$ . Let  $Q(A, B)$  denote the set of quartet topologies of the form  $aa'|bb'$  where  $a, a' \in A$  and  $b, b' \in B$ . An internal vertex  $v$  in tree  $T$  induces the tripartition  $(A, B, C)$  if  $T - \{v\}$  consists of three trees labeled by  $A, B$  and  $C$ , respectively. This is denoted  $v = (A, B, C)$ . Let  $Q(A, B, C)$  denote the union of  $Q(A, B)$ ,  $Q(A, C)$  and  $Q(B, C)$ . Two bipartitions  $(A, B)$  and  $(C, D)$  are *compatible* if they can be induced in the same tree, i.e., either  $A \subseteq C$  or  $C \subseteq A$ . Similarly, two tripartitions are compatible if they can be induced in the same tree.

In order to assess the performance of quartet cleaning algorithms an understanding of the distribution of quartet errors in  $T$  is needed. Let  $P_T(a, b)$  be the path between sequences  $a$  and  $b$  in tree  $T$ . If  $ab|cd$  is induced in  $T$  then  $P_T(a, c) \cap P_T(b, d)$  is called the *joining path* of  $\{a, b, c, d\}$  in  $T$ . Notice that the joining path is necessarily non-empty (see Fig. 1). Define the quartet error  $\{a, b, c, d\}$  to be *across* edge  $e$  if  $e$  is on the joining path of  $\{a, b, c, d\}$  in  $T$ . Similarly, define the quartet error  $\{a, b, c, d\}$  to be *across* vertex  $v$  if  $v$  is on the joining path of  $\{a, b, c, d\}$  in  $T$ . These definitions permit the assignment of quartet errors in  $Q$  to edges/vertices of  $T$ .



**Fig. 4.** Cleaning bounds.

Let  $e = (X, Y)$  be the bipartition induced in  $T$  as depicted in Fig. 4. Observe that  $Q_T$  and  $Q_{T'}$  differ by quartets of the form  $ax|by$  where  $x \in X$  and  $y \in Y$ . It follows that  $|Q_T - Q_{T'}| = (|X| - 1)(|Y| - 1)$ . If half of the quartets of the form  $\{a, b, x, y\}$  with  $x \in X$  and  $y \in Y$  have quartet topology  $ax|by$  in  $Q$  and the other half have quartet topology  $bx|ay$  in  $Q$  then no quartet cleaning algorithm can guarantee that all quartet errors across  $e$  can be corrected under the MQC principle of optimality. This implies that  $(|X| - 1)(|Y| - 1)/2$  is an upper bound on the number of quartet errors across an

edge of  $T$  that can be corrected. This example motivates the following formulations of quartet cleaning:

**Local Edge Cleaning** A local edge cleaning algorithm with *edge cleaning bound*  $b$  corrects all quartet errors across any edge with fewer than  $b$  quartet errors across it.

**Global Edge Cleaning** A global edge cleaning algorithm with *edge cleaning bound*  $b$  corrects all quartet errors in  $Q$  if each edge of  $T$  has fewer than  $b$  quartet errors across it.

Analogous definitions apply for local and global vertex cleaning algorithms. Local edge/vertex cleaning is more robust than global edge/vertex cleaning since it can be applied to an edge/vertex independently of the number of quartet errors across other edges/vertices. This is a significant feature especially when some edges/vertices have a high number of quartet errors across them. In contrast, global cleaning algorithms are applicable only if all edges/vertices satisfy the cleaning bound.

The example from Fig. 4 illustrates that cleaning bounds should not be constant but vary with bipartition sizes. Hence, an edge  $e = (X, Y)$  would have an edge cleaning bound that depends on  $|X|$  and  $|Y|$ . In particular, the example demonstrates that the optimal edge cleaning bound is  $(|X| - 1)(|Y| - 1)/2$ .

The first contribution of the paper is an  $O(n^4)$  time global edge cleaning algorithm with edge cleaning bound  $(|X| - 1)(|Y| - 1)/2$ . Following the above remarks, this algorithm is optimal in the number of quartet errors it can correct across an edge  $e = (X, Y)$ . The global edge cleaning algorithm is presented in Sect. 2.1.

The second contribution of the paper is an  $O(n^7)$  time local vertex cleaning algorithm with vertex cleaning bound  $(|X| - 1)(|Y| - 1)/4$ . Although this algorithm has a smaller cleaning bound than the global edge cleaning algorithm, it is more robust since it is local. Hence, there are situations where the local vertex cleaning algorithm is superior to the global edge cleaning algorithm and vice-versa. The local vertex cleaning algorithm is presented in Sect. 2.3.

The third contribution of the paper is an extensive simulation study that assesses the utility of the above quartet cleaning algorithms. This study establishes the following:

- Regardless of the quartet topology inference method used to obtain  $Q$ , quartet errors are prevalent in sets of inferred quartet topologies. To establish this three popular quartet topology inference methods are evaluated: maximum parsimony [10], neighbor joining [15] and the ordinal quartet method [14]. This establishes that there is a need for quartet cleaning algorithms.
- The global edge cleaning algorithm and the local vertex cleaning algorithm are very effective at correcting quartet errors. In particular, the local vertex cleaning algorithm is more effective (due to its robustness) and both algorithms dramatically increase the accuracy of the inferred quartet topology set  $Q$ .

The simulation study is presented in Sect. 3.

## 1.2 Previous Results

The idea of quartet cleaning was introduced in [13]. This paper presented a polynomial time global edge quartet cleaning algorithm with cleaning bound  $\alpha(|X| - 1)(|Y| - 1)/2$  where  $\alpha > 0$  is a fixed constant. Hence, this algorithm is suboptimal. Although this

algorithm has a polynomial time complexity, it is of very high degree, and so, of primarily theoretical interest. The algorithms presented here are more efficient, more effective (the global edge cleaning algorithm is optimal) and more robust.

## 2 Quartet Cleaning Algorithms

### 2.1 A Global Edge Quartet Cleaning Algorithm

Let  $S$  be a set of sequences that evolved on evolutionary tree  $T$  and let  $Q$  be a set of quartet topologies inferred from  $S$ . Assume that there are fewer than  $(|A| - 1)(|B| - 1)/2$  quartet errors across each edge  $e = (A, B)$  of  $T$ . In other words, assume that  $|Q(A, B) - Q| < (|A| - 1)(|B| - 1)/2$ . The following algorithm constructs  $T$  from  $Q$  by building  $T$  iteratively from the leaves up. Note that for the purposes of this algorithm, a leaf is in  $R$  is considered to be a rooted subtree.

**Algorithm** Global-Clean( $Q$ )

1. Let  $R := S$ .
2. For every pair of rooted subtrees  $T_1$  and  $T_2$  in  $R$
3.     Let  $A$  denote the leaf sequences of  $T_1$  and  $T_2$ .
4.     If  $|Q(A, S - A) - Q| < (|A| - 1)(|S - A| - 1)/2$  then
5.         Create a new tree  $T'$  that contains  $T_1$  and  $T_2$  as its subtrees.
6.         Let  $R := R - \{T_1, T_2\} \cup \{T'\}$ .
7. Repeat step 2 until  $|R| = 3$ .
8. Connect the three subtrees in  $R$  at a new vertex and output the resulting unrooted tree.

**Theorem 2.** *Algorithm Global-Clean outputs the tree  $T$  correctly.*

Note that if not all edges of  $T$  satisfy the global cleaning bound of  $(|A| - 1)(|B| - 1)/2$  then Global-Clean fails to return an evolutionary tree.

A straightforward implementation of Global-Clean results in a time complexity of  $O(n^2 \cdot n^4) = O(n^6)$ , since each quartet can be checked at most  $O(n^2)$  times. Section 2.2 presents a more careful implementation and analysis that yields a linear  $O(n^4)$  global edge cleaning algorithm.

### 2.2 An Efficient Implementation of Global-Clean

The idea is as follows. First, observe that the most time consuming step in Global-Clean is step 4, *i.e.*, the identification of discrepancies between the set of quartet topologies induced by a candidate bipartition and those in the given set  $Q$ . Our idea is to avoid considering the same quartet repeatedly as much as possible. In order to achieve this, for each correct bipartition  $(A, S - A)$ , we record the set  $Q(A, S - A) - Q$  using a linked list. Let  $A_1$  and  $A_2$  denote the leaf sequences of  $T_1$  and  $T_2$ , respectively, considered in step 2 of Global-Clean, and let  $A = A_1 \cup A_2$ , according to step 3. Then, in step 4, observe that  $Q(A, S - A) - Q = (Q(A_1, S - A) - Q) \cup (Q(A_2, S - A) - Q) \cup (Q'(A_1, A_2, S - A) - Q)$ , where  $Q'(A_1, A_2, S - A)$  denotes the set of all quartet topologies  $ab|cd$  with  $a \in A_1$ ,  $b \in A_2$ , and  $c, d \in S - A$ . Since  $Q(A_1, S - A) - Q \subseteq Q(A_1, S - A_1) - Q$ , we can compute  $Q(A_1, S - A) - Q$  from the list for  $Q(A_1, S - A_1)$  in at most  $|Q(A_1, S - A_1)| = O(n^2)$

time. Similarly,  $Q(A_2, S - A) - Q$  can be computed in  $O(n^2)$  time. Moreover, at each execution of step 2 (except for the first time), we need only check  $O(n)$  new possibilities of joining subtrees: joining the subtree resulting from the previous iteration, say  $T_1$ , with each of the  $O(n)$  other subtrees, say  $T_2$ . Since all the other possibilities have been examined in previous iterations, their outcomes can be easily retrieved. Because step 2 is repeated  $n - 2$  times, the overall time required for computing the sets  $Q(A_1, S - A) - Q$  and  $Q(A_2, S - A) - Q$  is  $O(n \cdot n \cdot n^2) = O(n^4)$ .

Next we show that computing the sets  $Q'(A_1, A_2, S - A) - Q$  also globally requires  $O(n^4)$  time, thanks to some delicate data structures. For each possible clustering  $\{T_1, T_2\}$  of two subtrees, we split the set  $Q'(A_1, A_2, S - A)$  into two lists,  $Q'_-(A_1, A_2)$  and  $Q'_+(A_1, A_2)$ , depending on whether a topology contradicts or corresponds to the one in the set  $Q$ . That is,  $Q'_-(A_1, A_2) = Q'(A_1, A_2, S - A) - Q$ . When subtrees  $T_1$  and  $T_2$  are joined, the linked list  $Q(A, S - A)$  associated to the new subtree  $T' = \{T_1, T_2\}$  is simply obtained by first adding  $Q(A_1, S - A) - Q$  and  $Q(A_2, S - A) - Q$ , which requires  $O(n^2)$  time, and then adding the list  $Q'_-(A_1, A_2)$ , which requires  $O(1)$  time if  $Q'_-(A_1, A_2)$  (and  $Q'_+(A_1, A_2)$ ) is represented as a doubly-linked list. Note that  $Q'_-(A_1, A_2)$  need not be scanned as its quartets are distinct from those already added to  $Q(A, S - A) - Q$ .

The  $Q'_-$  and  $Q'_+$  lists of the new possible clusterings involving  $T'$  are easy to obtain. If  $A_i$  denotes the leaf sequences of a subtree  $T_i \neq T_1, T_2$ , then  $Q'_-(A, A_i) = Q'_-(A_1, A_i) \cup Q'_-(A_2, A_i) - Q''(A_1, A_2, A_i)$  and  $Q'_+(A, A_i) = Q'_+(A_1, A_i) \cup Q'_+(A_2, A_i) - Q''(A_1, A_2, A_i)$  where  $Q''(A_1, A_2, A_i)$  denotes the members of  $Q'(A_1, A_2, S - A)$  involving a leaf sequence in  $A_i$ . We proceed by scanning first  $Q'_-(A_1, A_2)$  and  $Q'_+(A_1, A_2)$  to remove the occurrences of the elements of  $Q''(T_1, T_2, T_i)$  in the  $Q'_-$  and  $Q'_+$  lists of  $\{T_1, T_i\}$  and  $\{T_2, T_i\}$ . Then we merge these disjoint lists in  $O(1)$  time to obtain those of  $\{T', T_i\}$ . Linking together all the occurrences of each quartet in the  $Q'_-$  and  $Q'_+$  lists at the beginning of the algorithm enables us to remove each member of  $Q''(A_1, A_2, A_i)$  from other  $Q'_-$  and  $Q'_+$  lists in  $O(1)$  time.

We remark that at any time of the algorithm, each quartet appears only a constant number of times in all  $Q'_-$  and  $Q'_+$  lists. If  $q = a, b, c, d$  and  $T_1, T_2, T_3, T_4$  are the respective subtrees containing the leaf sequences  $a, b, c, d$ , then only the 6 clusterings  $\{T_1, T_2\}$ ,  $\{T_1, T_3\}$ ,  $\{T_1, T_4\}$ ,  $\{T_2, T_3\}$ ,  $\{T_2, T_4\}$ , and  $\{T_3, T_4\}$  resolve  $q$ , and hence  $q$  belongs only to the  $Q'_-$  and  $Q'_+$  lists of these 6 clusterings. As each occurrence of a quartet in the  $Q'_-$  and  $Q'_+$  lists is checked or removed only once during the algorithm, the maintenance of the  $Q'_-$  and  $Q'_+$  lists requires  $O(n^4)$  time totally. Now, these lists (and their sizes) enable us to compute  $|Q'(A_1, A_2, S - A) - Q| = |Q'_-(A_1, A_2)|$  in  $O(1)$  time and thus  $|Q(A, S - A) - Q|$  in  $O(n^2)$  time, when needed.

As mentioned above, Global-Clean requires  $n - 2$  clusterings. Before each clustering we have to examine  $O(n)$  new candidates, each requiring  $O(n^2)$  time. Hence this implementation requires  $O(n \cdot n \cdot n^2) = O(n^4)$  time.

### 2.3 A Local Vertex Quartet Cleaning Algorithm

Let  $S$  be a set of sequences that evolved on evolutionary tree  $T$  and let  $Q$  be a set of quartet topologies inferred from  $S$ . Define a bipartition  $(A, B)$  to be *good* if  $|Q(A, B) - Q| \leq (|A| - 1)(|B| - 1)/4$ . A tripartition  $(A, B, C)$  is good if each of its three induced

bipartitions is good, *i.e.*,  $|Q(A, B \cup C) - Q| \leq (|A| - 1)(|B \cup C| - 1)/4$ ,  $|Q(B, A \cup C) - Q| \leq (|B| - 1)(|A \cup C| - 1)/4$  and  $|Q(C, A \cup B) - Q| \leq (|C| - 1)(|A \cup B| - 1)/4$ .

The following algorithm corrects all quartets errors across vertices of  $T$  that induce good tripartitions.

**Algorithm** Tripartition( $Q$ )

1. Let  $Tri(Q) := \emptyset$ .
2. For every three sequences  $a, b$  and  $c$
3.   Let  $A = \{a\}$ ;  $B = \{b\}$ ;  $C = \{c\}$ .
4.   For each  $w \in S - \{a, b, c\}$ .
5.     If  $aw|bc \in Q$  then  $A = A \cup \{w\}$ .
6.     If  $bw|ac \in Q$  then  $B = B \cup \{w\}$ .
7.     If  $cw|ab \in Q$  then  $C = C \cup \{w\}$ .
8.   If  $(A, B, C)$  is good then let  $Tri(Q) := Tri(Q) \cup \{(A, B, C)\}$ .
9. Output  $Tri(Q)$ .

To show that the above algorithm is a local vertex cleaning algorithm with cleaning bound  $(|A| - 1)(|B| - 1)/4$ , it is first proven that  $Tri(Q)$  contains all vertex induced tripartitions of  $T$  that are good.

**Theorem 3.** *Let  $v$  be a vertex of  $T$  that induces tripartition  $(A, B, C)$ . If  $(A, B, C)$  is good then  $(A, B, C) \in Tri(Q)$ .*

Next we show that the tripartitions produced by the Tripartition algorithm are compatible. The following lemma will be useful.

**Lemma 4.** *Two tripartitions are compatible if and only if their induced bipartitions are all compatible with each other.*

**Theorem 5.** *If  $(A, B, C)$  and  $(X, Y, Z)$  are both in  $Tri(Q)$  then they are compatible.*

Theorem 3 informs us that  $Tri(Q)$  contains all good tripartitions of  $T$  and Theorem 5 informs us that  $Tri(Q)$  contains only tripartitions compatible with the good tripartitions of  $T$ . Let  $T'$  be the tree obtained by combining the tripartitions of  $Tri(Q)$ . Then  $Q_{T'}$  is a corrected version of  $Q$ . Note that unlike algorithm Global-Clean described in Sect. 2.1, algorithm Tripartition always produces a tree and this tree is either the same as  $T$  or is a contraction of  $T$ . A straightforward implementation of the Tripartition algorithm runs in  $O(n^3 \cdot n^4) = O(n^7)$  time. An  $O(n^5)$  implementation has been found recently by Della Vedova [7].

### 3 Simulation Study

In this section, a simulation study is presented that addresses the utility of the quartet cleaning algorithms. In particular, the simulation study addresses questions of need and applicability:

1. How common are quartet errors in sets of inferred quartet topologies?
2. How effective are the quartet cleaning algorithms at correcting quartet errors when they occur?

The approach of the simulation study was to simulate the evolution of sequences on an evolutionary tree  $T$ . From these sequences, a set of quartet topologies was inferred using a quartet inference method. This set of quartet topologies was then compared to the actual quartet topologies induced by  $T$ . Any quartet whose inferred and actual topologies did not match was considered a quartet error. By mapping these quartet errors back onto the vertices and edges of  $T$ , it was possible to establish the number of quartet errors across the vertices and edges of  $T$ . From this information it was determined which vertices and edges of  $T$  could be cleaned by the global edge and local vertex cleaning algorithms described in this paper.

The simulation study was extensive in that sequences were evolved under a broad range of parameters where the sequence length, evolutionary tree topology and evolutionary rates were varied. As well, three popular quartet inference methods, namely, Neighbor Joining [15], the Ordinal Quartet method [14] and Maximum Parsimony [10] were used to infer the sets of quartet topologies from the simulated sequences.

The details of the simulation study are as follows. For Neighbor Joining and the Ordinal Quartet method, distance matrices were generated from the sequences and corrected relative to the Kimura two-parameter (K2P) model of evolution [18, page 456]. For Maximum Parsimony, the sequences were corrected for transition/transversion bias by weighting the character-state transition matrix appropriately [18, pages 422–423]. Evolutionary trees and sequences were created in a manner similar to [6] using code adapted from program `listtree` in the PAML phylogeny analysis package [20]: Evolutionary tree topologies were generated by adding taxa at random, edge-lengths were assigned according to a specified mean edge-length, and sequences were evolved on these evolutionary trees using the K2P model with transition-transversion bias  $\kappa = 0.5$ . The simulation examined 1000 randomly-generated topologies on 10 sequences  $\times$  5 mean edge-lengths per topology  $\times$  100 edge-length sets per topology  $\times$  3 sequence lengths = 1,500,000 sequences datasets, where the set of mean edge-lengths considered was  $\{0.025, 0.1, 0.25, 0.5, 0.75\}$  and the set of sequence lengths considered was  $\{50, 200, 2000\}$ . The simulation code consists of a number of C-language and shell-script programs that were compiled and run under the UNIX system on several of the SUN workstations owned by the computational biology group at McMaster University. The simulations took 5 days to complete.

The results of the simulation study are summarized in Table 1. Consider how the results address the two questions posed at the beginning of this section.

Firstly, the results clearly establish that quartet errors are prevalent, independent of the quartet topology inference method used. The number of quartet errors decreases as sequence length increases and mean edge-length decreases but remains significant even for sequence length 2000 and mean edge-length 0.025. Hence, there is a definite need for quartet cleaning algorithms.

Secondly, a comparison of the parenthesized and unparenthesized values in this table establishes the effectiveness of these algorithms. Both algorithms decrease the number of quartet errors significantly under a wide variety of conditions. For example, under the Ordinal Quartet Method with mean edge-length 0.1 and sequence length 200, the increase in accuracy is approximately 25%. When the mean edge-length is large and/or the sequence length is small, the increased accuracy is dramatic.

**Table 1.** Performance of the Quartet Cleaning Algorithms Under Simulation. The unparenthesized number is the average percent of all evolutionary trees (vertices per evolutionary tree) that have no quartet errors (no quartet errors across them) after global edge cleaning (local vertex cleaning) has been applied. The parenthesized number is the average percent of all evolutionary trees (vertices per evolutionary tree) that have no quartet errors (no quartet errors across them) before quartet cleaning is applied.

Quartet Inference Method	Mean Edge Length	Global Edge (% of Trees Cleanable)			Local Vertex (Average % of Vertices per Tree that are Cleanable)		
		Sequence Length			Sequence Length		
		50	200	2000	50	200	2000
Maximum Parsimony (Corrected)	0.025	10.48% (0.45%)	62.85% (10.85%)	95.02% (56.67%)	38.98% (29.20%)	76.15% (59.82%)	94.80% (87.27%)
	0.1	38.62% (0.22%)	79.60% (6.17%)	92.34% (27.06%)	55.91% (28.82%)	80.76% (55.57%)	89.90% (74.01%)
	0.25	26.88% (0.01%)	64.44% (0.71%)	78.48% (6.83%)	45.18% (20.09%)	67.96% (37.73%)	77.03% (53.93%)
	0.5	3.25% (0.00%)	23.46% (0.03%)	56.33% (0.87%)	23.50% (10.59%)	43.47% (22.38%)	61.71% (38.42%)
	0.75	0.30% (0.00%)	3.81% (0.00%)	27.46% (0.08%)	13.08% (6.03%)	25.46% (14.11%)	46.57% (28.48%)
Neighbor Joining (Corrected)	0.025	16.36% (0.73%)	70.47% (13.49%)	95.67% (57.18%)	43.79% (32.53%)	79.85% (62.96%)	95.06% (87.46%)
	0.1	47.10% (0.34%)	81.36% (6.92%)	91.72% (24.57%)	60.86% (32.04%)	81.57% (56.65%)	88.97% (72.31%)
	0.25	30.85% (0.02%)	64.18% (0.84%)	76.48% (6.16%)	47.69% (23.19%)	67.50% (38.44%)	75.48% (52.55%)
	0.5	3.91% (0.00%)	22.27% (0.07%)	51.80% (0.79%)	26.91% (16.37%)	43.25% (26.27%)	58.88% (38.08%)
	0.75	0.41% (0.00%)	3.63% (0.00%)	21.47% (0.06%)	16.78% (11.33%)	27.56% (19.41%)	43.48% (30.49%)
Ordinal Quartet Method (Corrected)	0.025	20.49% (1.21%)	74.33% (17.11%)	96.12% (59.25%)	47.79% (37.36%)	82.19% (66.38%)	95.45% (88.19%)
	0.1	64.15% (1.21%)	86.91% (12.38%)	93.84% (34.53%)	71.92% (40.32%)	86.32% (63.45%)	91.50% (78.00%)
	0.25	63.64% (0.16%)	81.72% (2.42%)	87.31% (13.82%)	68.85% (30.29%)	81.13% (47.25%)	85.44% (65.10%)
	0.5	39.03% (0.04%)	67.26% (0.19%)	81.75% (1.82%)	52.41% (23.67%)	69.18% (32.06%)	79.47% (45.22%)
	0.75	12.32% (0.00%)	35.18% (0.01%)	63.60% (0.16%)	34.28% (18.56%)	49.52% (24.86%)	66.47% (33.28%)

**Acknowledgments:** The authors would like to acknowledge John Tsang and Diane Miller for early discussions about the structure of the simulation and John Nakamura and Matt Kelly for assistance in debugging the initial version of the simulation code. They would especially like to acknowledge Chris Trendall for his work on speeding up

and running the initial version of the simulation code and Mike Hu and Chris Trendall for their subsequent work on rewriting and further speeding up this code, which in large part made possible the results reported here.

## References

1. A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Peleg. From four-taxon trees to phylogenies: The case of mammalian evolution. *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 9–19, 1998.
2. V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Proceedings of the Third Annual International Computing and Combinatorics Conference*, pages 111–123, 1997.
3. M. Boguski. Bioinformatics – a new era. In S. Brenner, F. Lewitter, M. Patterson, and M. Handel, editors, *Trends Guide to Bioinformatics*, pages 1–3. Elsevier Science, 1998.
4. D. Bryant. *Structures in Biological Classification*. Ph.D. Thesis, Department of Mathematics and Statistics, University of Canterbury, 1997.
5. P. Buneman. The recovery of trees from measures of dissimilarity. In F.R. Hodson, D.G. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.
6. M. A. Charleston, M. D. Hendy, and D. Penny. The effects of sequence length, tree topology, and number of taxa on the performance of phylogenetic methods. *Journal of Computational Biology*, 1(2):133–152, 1994.
7. G. Della Vedova. An improved algorithm for local vertex cleaning. *Manuscript*, 1998.
8. P. Erdős, M. Steel, L. Székely, and T. Warnow. Constructing big trees from short sequences. *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming*, 1997.
9. J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
10. W. M. Fitch. Toward defining the course of evolution: Minimal change for a specific tree topology. *Systematic Zoology*, 20:406–411, 1971.
11. V. A. Funk and D. R. Brooks. *Phylogenetic Systematics as the Basis for Comparative Biology*. Smithsonian Institution Press, 1990.
12. R. Gupta and B. Golding. Evolution of hsp70 gene and its implications regarding relationships between archaeobacteria, eubacteria and eukaryotes. *Journal of Molecular Evolution*, 37:573–582, 1993.
13. T. Jiang, P. E. Kearney, and M. Li. Orchestrating quartets: Approximation and data correction. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 416–425, 1998.
14. P. E. Kearney. The ordinal quartet method. *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 125–134, 1998.
15. N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.
16. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.
17. K. Strimmer and A. von Haeseler. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13(7):964–969, 1996.
18. D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In D. M. Hillis, C. Moritz, and B. K. Mable, editors, *Molecular Systematics, 2nd Edition*, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.

19. L. Vigilant, M. Stoneking, H. Harpending, H. Hawkes, and A. C. Wilson. African populations and the evolution of human mitochondrial DNA. *Science*, 253:1503–1507, 1991.
20. Z. Yang. PAML: a program package for phylogenetic analysis by maximum likelihood. *CABIOS*, 15:555–556, 1997.