Journal of Computer and System Sciences, Vol. 67(4), pp. 723–741, 2003

A Fixed-Parameter Algorithm for Minimum Quartet Inconsistency[*]

Jens Gramm[†] Rolf Niedermeier
Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,
Sand 13, D-72076 Tübingen, Fed. Rep. of Germany,
Email: {gramm|niedermr}@informatik.uni-tuebingen.de.

Abstract

Given n taxa, exactly one topology for every subset of four taxa, and a positive integer k (the parameter), the Minimum Quartet Inconsistency (MQI) problem is the question whether we can find an evolutionary tree inducing a set of quartet topologies that differs from the given set in only k quartet topologies. The more general problem where we are not necessarily given a topology for every subset of four taxa appears to be fixed-parameter intractable. For MQI, however, which is also NP-complete, we can compute the required tree in time $O(4^k \cdot n + n^4)$. This means that the problem is fixed-parameter tractable and that in the case of a small number k of "errors" the tree reconstruction can be done efficiently. In particular, for minimal k, our algorithm can produce *all* solutions that resolve k errors. Additionally, we discuss significant heuristic improvements. Experiments underline the practical relevance of our solutions.

⟨Keywords: computational biology, minimum quartet inconsistency, parameterized complexity, phylogeny, quartet methods⟩

Proposed running title: A fixed-parameter algorithm for MQI.

1 Introduction

To determine the evolutionary relationship of a set of taxa, e.g., based on DNA or protein sequence data, is an important question in computational biology. A common model for this relationship is an *evolutionary tree*, a binary tree T in which the leaves are bijectively labeled by the taxa. In recent years, quartet methods for reconstructing evolutionary trees have received considerable attention [9, 19]. Here, a *quartet* is a size four subset $\{a, b, c, d\}$ of the set of taxa and the *quartet topology* for $\{a, b, c, d\}$ induced by T simply is the four leaves subtree of T for $\{a, b, c, d\}$. The three possible quartet topologies for $\{a, b, c, d\}$ are $[ab|cd]$, $[ac|bd]$, and $[ad|bc]$. They are shown in Figure 1.[1] The fundamental goal of quartet

[*]An extended abstract of this paper was presented at the 12th Annual Symposium on Combinatorial Pattern Matching (CPM 2001), Springer-Verlag, LNCS 2089, pages 241–256, held in Jerusalem, Israel, July 1–4, 2001.

[†]This work was supported by the Deutsche Forschungsgemeinschaft (DFG), research projects "KOMET," LA 618/3-3, and "OPAL" (optimal solutions for hard problems in computational biology), NI 369/2-1.

[1]A fourth possible topology is the star topology which is not considered here because it is not binary.

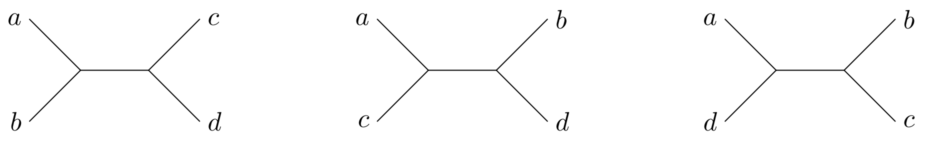


Figure 1: Possible quartet topologies for quartet $\{a, b, c, d\}$, which are (from left to right) $[ab|cd]$, $[ac|bd]$, and $[ad|bc]$.

methods is, given a set of quartet topologies, to reconstruct the corresponding evolutionary tree. Herein, the given set of quartet topologies can be incomplete, may contain errors or more than one topology for one quartet. Hence, to reconstruct (a good estimation of) the original evolutionary tree becomes an optimization problem, which generally turns out to be NP-hard.

In this paper, we focus on the Minimum Quartet Inconsistency (MQI) problem.

> Minimum Quartet Inconsistency (MQI)
> **Input:** A set S of n taxa and a set Q_S of $\binom{n}{4}$ quartet topologies such that there is exactly one topology for *every* quartet corresponding to S and a positive integer k.
> **Question:** Is there an evolutionary tree T where the leaves are bijectively labeled by the elements from S such that the set of quartet topologies induced by T differs from Q_S in at most k quartet topologies?

MQI is NP-complete [6, 20]. It is worth noting, as was pointed out by Steel [24], that the quartet cleaning algorithm by Berry *et al.* [6] finds the optimal solution for instances with $k < (n-3)/2$. Therefore, MQI is NP-hard only for $k \geq (n-3)/2$. It is known that MQI is polynomial time approximable with a factor n^2 [19], and it is an open question of [19] whether MQI can be approximated with a factor at most n or even with a constant factor. Heuristics for the problem include semidefinite programming [3] and the widely used *quartet puzzling* [25]. The parameterized complexity of MQI, however, so far, has apparently been neglected—we close this gap here. For the case that the number k of "wrong" quartet topologies is small in comparison with the total number of given quartet topologies, we show that MQI is *fixed-parameter tractable* and can be solved exactly in worst-case time $O(4^k n + n^4)$. Observe that the input size is $O(n^4)$. The more general variant of MQI where the set Q_S is not required to contain a topology for every quartet (subsequently referred to as Sparse MQI) is NP-complete even if $k = 0$ [23]. Hence, this excludes parameterized complexity studies and also implies inapproximability (with any factor).

To establish the correctness and the running time of our algorithm, we exhibit some nice combinatorial properties of MQI. For instance, loosely speaking, we point out that "global conflicts" due to erroneous quartet topologies in fact can be led back to "local conflicts." The basis for this was laid by Colonius and Schultze [11], and by Bandelt and Dress [2]. This property is fundamental for our algorithms. Moreover, for minimal k, our approach makes it possible to construct *all* evolutionary trees that can be (uniquely) obtained from the given input by changing k quartet topologies. This puts the user of the algorithm in the position to select (e.g., based on additional biological knowledge) the probably best, most reasonable solution. Our method also generalizes to weighted quartets. We consider further parameterizations of the base problems and fixed-parameter results for them, and we discuss some heuristic improvements to reduce the practical running time of our main algorithm significantly.

We performed several experiments on synthetic and real (fungi) data and, thereby, showed that our algorithm (due to intensive tuning) in practice runs much faster than its theoretical (worst-case) analysis predicts. For instance, with a small k (e.g., $k = 100$), we can solve relatively large ($n = 50$ taxa) instances optimally in around 8 minutes on a LINUX PC with a Pentium III 750 MHz processor and 192 MB main memory.

2 Preliminaries

Quartet methods infer the evolutionary tree only for four taxa, called a *quartet*, at a time. Once having determined the evolutionary tree for every quartet of taxa, they try to combine these evolutionary trees involving four taxa, called *quartet topologies*, in order to obtain a tree containing all taxa.

There are several reasons why quartet methods are widely used in practice. They are founded on the fact that an evolutionary tree is uniquely characterized by the quartet topologies for its size four sets of taxa [8]. From this set of topologies, we can efficiently compute the tree in polynomial time $O(n^4)$ [4]. Quartet methods clearly divide the tree construction process in two stages—we can use an arbitrary, even computationally expensive tree construction method for inferring the quartet topologies, while the recombination of topologies can be handled independently of the method chosen for inference. Another reason to use quartet methods is data disparity as discussed by Chor [9]: In practice, we often do not have the same amount of data for all considered taxa, e.g., not the same set of sequenced proteins. In general, tree construction methods cannot take advantage of information available only for a subset of taxa. Quartet methods, however, allow us to use the maximum amount of information available for the four taxa of a quartet when we compute its quartet topology.

The limitation of quartet methods in practice is caused by the process of quartet inference which can be erroneous. Therefore, we cannot be sure that there exists a tree inducing the inferred set of quartet topologies. Assuming that the number of errors is small compared to the number of correct topologies, we will try to overcome this problem by searching for a tree that matches the inferred topologies as "closely" as possible.

Recombination of quartet topologies without errors. Given exactly one quartet topology for every quartet of taxa, it is possible to decide in polynomial time whether there is a binary tree inducing all of the given quartet topologies, and, if so, to actually construct the tree [4]. Bryant and Steel [7] solve in time $O(n^5)$ the analogous problem in the case that we have one or two topologies for each quartet and the topologies are weighted. Both algorithms rely on the fact that the given set of topologies contains a topology for every quartet. In the more general situation in which we are not necessarily given a topology for every quartet, the problem of deciding whether there is a binary tree inducing all the given topologies is NP-complete [23].

Inferring strongly supported parts of the tree. There are situations in which there is no *binary* tree inducing the given set of topologies. In the following, we mention methods that infer bipartitions of taxa which are particularly supported by the given quartet topologies and which can be seen as edges of a tree; these methods yield trees that are not necessarily fully resolved, i.e., that are not binary.

The Q* method by Berry and Gascuel [4] infers, for a complete set of quartet topologies, only the *completely supported* edges, an approach proposed by Buneman [8]: Given a tree with its leaves bijectively labeled by the given set S of taxa, an edge e in the tree defines a bipartition of S into sets A_e and B_e, each of them containing the taxa in the subtree rooted at one end of e. We call edge e completely supported if, for every $a, a' \in A_e$ and $b, b' \in B_e$, the corresponding quartet topology is $[aa'|bb']$. The set of quartet topologies induced by the completely supported edges is called Q*. Berry and Gascuel [4] present an algorithm computing Q* in running time $O(n^4)$.

Another approach, called *quartet cleaning*, tries to correct obvious *quartet errors* if their number is bounded [5, 6, 12, 18, 20]. We distinguish *edge* and *vertex* quartet cleaning, as well as *global* and *local* algorithms. Edge quartet cleaning applies if the number of errors *across* one edge is smaller than some bound, in vertex quartet cleaning the errors across a vertex has to be bounded (for the definition of errors across an edge or a vertex, resp., see, e.g., [6]). Global quartet cleaning algorithms correct errors only if the errors are bounded for every edge or vertex. Local algorithms correct errors also when errors are bounded only for one edge or vertex. For an overview of quartet cleaning results refer to Della Vedova and Wareham [12]. The global edge quartet cleaning algorithm given by [6] computes the optimal tree if, for every edge e inducing a bipartition of taxa into sets A_e and B_e, we have fewer than $(|A_e|-1)(|B_e|-1)/2$ quartet errors across this edge. This value is minimal for $|A_e| = 2$ and $|B_e| = n-2$ and, therefore, quartet cleaning computes a guaranteed optimal solution if the total number of quartet errors is smaller than $(n-3)/2$.

Minimum quartet inconsistency. In order to find the "best" binary tree for a given set of quartet topologies, we can ask for a tree that violates a minimum number of topologies. If we are given exactly one quartet topology for every set of four taxa, this question is the MQI problem. If there is not necessarily a quartet topology for every set of four taxa, the more general question is referred to as Sparse MQI. Ben-Dor *et al.* [3] give an exact algorithm for the Sparse MQI problem, based on dynamic programming. For every subset of i taxa, it computes the optimal tree for these taxa based on the optimal trees for the subsets of $i-1$ taxa, with i running up to the total number n of species. The resulting running time is $O(3^n \cdot m)$, where n is the number of species and m is the number of given quartet topologies, and the memory requirement is $\Theta(2^n)$.

Regarding heuristics, Ben-Dor *et al.* [3] use semidefinite programming to obtain, in polynomial time, possibly non-optimal solutions for Sparse MQI. A widely used heuristic for MQI is *quartet puzzling* by Strimmer and von Haeseler [25]. Its main idea is to build the tree incrementally, starting with four taxa and adding one taxon at a time in a greedy way. To avoid local traps the algorithm repeats this process and, finally, constructs a (possibly non-binary) consensus of the single trees.

Not much is known about approximability of MQI. Jiang *et al.* [19] mention a factor n^2-approximation, at the same time asking for better approximation results. Note that the complement problem of MQI, where one tries to find a tree T that *maximizes* the number of given quartet topologies induced by T, possesses a polynomial time approximation scheme [18,

20] which, however, is not used in practice due to its high running time.
Some notation. Assume that we are given a set of n taxa S. A set of quartet topologies is *complete* if it contains exactly one topology for every quartet of S. A complete set of quartet topologies for the quartets over S we denote by Q_S. A set of quartet topologies Q is *tree-consistent* [2] if there exists a tree T such that, for the set Q_T of quartet topologies induced by T, we have $Q \subseteq Q_T$. Set Q is *tree-like* [2] if there exists a tree with $Q = Q_T$. Since an evolutionary tree is uniquely characterized by the topologies for all its quartets [8], a complete set of topologies is tree-consistent if and only if it is tree-like. Intuitively, a set of topologies has a "conflict" whenever it is not tree-consistent. We will call a conflict "global," when a complete set of topologies is not tree-consistent. In contrary, we call it "local," when a size three set of topologies, which necessarily is incomplete, is not tree-consistent.

We investigate MQI in the context of parameterized complexity [13], where a problem is called *fixed-parameter tractable* if it can be solved in deterministic time $f(k)n^{O(1)}$. Herein, f may be an arbitrary (usually exponential or worse) function only depending on a parameter k, but not depending on the input size n. Besides the monograph [13], some recent surveys on parameterized complexity are [1, 14, 15].

3 Global Conflicts are Local

The key to develop a fixed-parameter solution for MQI is as follows: It is sufficient to examine the size three sets of quartet topologies and to recursively branch on local conflicts. We use results by Bandelt and Dress [2] who introduced the *substitution property* to identify subsets of quartet topologies which cause conflicts: Given a set of taxa S and a complete set of quartet topologies Q_S over these taxa, a set $S_5 \subseteq S$ of five taxa satisfies the *substitution property* if, for every choice of distinct $a, b, c, d, e \in S_5$, $[ab|cd] \in Q_S$ implies $[ab|ce] \in Q_S$ or $[ae|cd] \in Q_S$.

Proposition 1. *(Proposition 6 in [2]) Given a set of taxa S and a complete set of quartet topologies Q_S and some taxon $f \in S$, then Q_S is tree-like iff every size five set of taxa that contains f satisfies the substitution property.*

The proof for Proposition 1 (given in [2]) relies on the "denseness" given in a complete set of quartet topologies (for short, we sometimes only write topologies).

In the following, we show that in Proposition 1, we can replace the substitution property with the more common term of tree-consistency.

Lemma 1. *Three topologies involving more than five taxa are tree-consistent.*

Proof. Assume we have topologies t_1, t_2, and t_3 involving more than five taxa. We distinguish two cases, in which Case (1) will apply if one of t_1, t_2, t_3 involves a taxon not occurring in the other two topologies. Case (2) will apply if, for each pair of topologies from t_1, t_2, t_3, there are exactly two taxa occurring in both topologies. Counting arguments make sure that either Case (1) or Case (2) applies. Assume that Case (1) does not apply: Then, we have three quartets, each of them containing four from the at least six given taxa, and every taxon has to occur in at least two of the three quartets. This is only possible for exactly six taxa—here Case (2) applies. With more than six taxa one would necessarily have a taxon occurring in only one of the topologies; this is handled in Case (1).

Case (1) A topology $t \in \{t_1, t_2, t_3\}$ contains a taxon occurring in none of the other topologies. Assume, w.l.o.g., that $t = t_1 = [ab|cd]$ and that a is the taxon occurring only in t_1 and not in t_2 or t_3. We can certainly find a tree T inducing t_2, and t_3, since the topologies for only two different quartets are always tree-consistent.

In the case that b does occur in t_2 or t_3, we replace in T the leaf b by an inner node having two leaves as its children, one labeled with a and the other with b. In the case that b does not occur in t_2 and t_3, we also create a new inner node with children a and b, and insert it at some arbitrary edge of T.

The modified tree induces t_1, t_2, t_3, hence they are tree-consistent.

Case (2) For each pair of topologies from t_1, t_2, t_3, there are exactly two taxa occurring in both topologies. W.l.o.g., we can assume that topology t_1 is given for quartet $\{a, b, c, d\}$, topology t_2 is given for quartet $\{a, b, e, f\}$, and topology t_3 is given for quartet $\{c, d, e, f\}$. Checking all possible combinations of topologies for t_1, t_2, t_3 (we omit the details here), we find that we always can find a tree inducing t_1, t_2, t_3. □

When searching for local conflicts, Lemma 1 makes it possible to focus on the case of three topologies involving only five taxa. If the substitution property is not satisfied for taxa $a, b, c, d, e \in S$, since $[ab|cd] \in Q_S$ but $[ab|ce] \notin Q_S$ and $[ae|cd] \notin Q_S$, then we say that the topologies for the quartets $\{a, b, c, d\}$, $\{a, b, c, e\}$, and $\{a, c, d, e\}$ *contradict* the substitution property.

Lemma 2. *For a given a set of taxa S, three topologies consisting of taxa from S are tree-consistent iff they do not contradict the substitution property.*

Proof. First, we note that with three topologies involving more than five taxa, on the one hand, we can build a tree inducing these taxa (according to Lemma 1) and, on the other hand, these taxa cannot contradict the substitution property (the substitution property is formulated over five taxa only). Therefore, we can in the following focus on the case of three topologies involving only five taxa.

($\Rightarrow$) As the three topologies are tree-consistent, we can find a tree inducing the topologies. The set of induced topologies is tree-like. With Proposition 1 the substitution property holds.

($\Leftarrow$) We are given three topologies which do not contradict the substitution property and which involve five taxa $\{a, b, c, d, e\}$.

First, we want to reduce the number of cases we have to consider. For three topologies over five taxa which do not contradict the substitution property, we show that it is, w.l.o.g., possible to assume that two of them are $[ab|cd]$ and $[ab|ce]$. This means that two of the topologies have to be equal on one side. Assuming that this is not true leads to a contradiction. To see this, we take two topologies $t_1 = [ab|cd]$ and $t_2 = [ac|de]$, and show that there is no topology t_3 with the properties (1) that t_1, t_2, t_3 do not contradict the substitution property and (2) that no side of t_3 equals a side of t_1 or t_2. Topology t_3 cannot be a topology for quartets $\{a, b, c, e\}$ or $\{a, b, d, e\}$. The reason is that, given topology $t_1 = [ab|cd]$, the substitution property would require either topology $[ae|cd]$ (and $[be|cd]$) or topology $[ab|ce]$ (and $[ab|de]$). Since $[ae|cd]$ would contradict t_2, we necessarily would have that the topology is $[ab|ce]$ or $[ab|de]$. These, however, would contradict property (2), because they equal t_1 in the "ab side." Analogously, t_3 cannot be a topology for quartet $\{b, c, d, e\}$—the substitution property would require that the topology is $[bc|de]$, which would contradict property (2), since

Topology t_1	Topology t_2	Topology t_3	Contradict the subst. prop.	Completion to tree-like set
$[ab\|cd]$	$[ab\|ce]$	$[ab\|de]$	no	$[ae\|cd]$, $[be\|cd]$
		$[ad\|be]$	yes	
		$[ae\|bd]$	yes	
		$[ac\|de]$	no	$[ab\|de]$, $[bc\|de]$
		$[ad\|ce]$	no	$[ab\|de]$, $[bd\|ce]$
		$[ae\|cd]$	no	$[ab\|de]$, $[be\|cd]$
		$[bc\|de]$	no	$[ab\|de]$, $[ac\|de]$
		$[bd\|ce]$	no	$[ab\|de]$, $[ad\|ce]$
		$[be\|cd]$	no	$[ab\|de]$, $[ae\|cd]$

Table 1: The quartet topologies considered in the proof of Lemma 2.

it equals t_2 in the "de side." Since there are no quartets over $\{a, b, c, d, e\}$ remaining, there are no choices left for t_3. Therefore, our assumption was wrong. Thus, for three topologies over five taxa, where the topologies do not contradict the substitution property, this justifies that two of the topologies have to be equal on one side.

With the preceding considerations, we can, w.l.o.g., assume that two of the given topologies are $t_1 = [ab|cd]$ and $t_2 = [ab|ce]$. We are given a third topology t_3. There remain three quartets over $\{a, b, c, d, e\}$ whose topology can take this place. These quartets are $\{a, b, d, e\}$, $\{a, c, d, e\}$, and $\{b, c, d, e\}$.

In Table 1, we list the three quartets and, for each of these three quartets, the three possible topologies it can take. In case the resulting triple of topologies does not contradict the substitution property, we complete them to a set of tree-like topologies, as shown in the last column of Table 1. For these choices of t_3 we, thereby, show that t_1, t_2, t_3 are tree-consistent. In two of the listed cases, we cannot complete the three topologies to a tree-like set. We find, however, that those triples of topologies contradict the substitution property. With the choice of $t_3 = [ad|be]$, the substitution property requires that we have either topology $[ad|bc]$ or topology $[ac|be]$, in contradiction to topologies t_1 and t_2. Analogously, the topologies contradict the substitution property with the choice of $t_3 = [ae|bd]$. □

Note that Lemma 2 involving a necessarily incomplete set of three topologies does not generalize from size three to an incomplete set of arbitrary size, as exhibited in the following example. For taxa $\{a, b, c, d, e, f\}$, consider the incomplete set of topologies $[ab|cd]$, $[bc|de]$, $[cd|ef]$, and $[af|de]$. Without going into the details, we only state here that these topologies are not tree-consistent, although there are no three topologies which contradict the substitution property.

With Lemma 2 we can now give another interpretation of Proposition 1.

Theorem 1. *Given a set of taxa S, a complete set of quartet topologies Q_S over S, and a taxon $f \in S$, Q_S is tree-like (and, thus, tree-consistent) iff every set of three topologies from Q_S which involves f is tree-consistent.*

Proof. By Lemma 2 we replace the substitution property in Proposition 1 with tree-consistency. □

Given a complete set of topologies Q_S for a set of taxa S, we do not necessarily know whether the set is tree-like or not. If it is not, we can, according to Theorem 1, choose an arbitrary taxon $f \in S$ and track down a subset of three topologies that involves f and that is not tree-consistent. Our goal will be to detect all these local conflicts involving f. This will be the preprocessing stage of the algorithm that will be described in Section 5; the subsequent stage of the algorithm will (try to) "repair" these conflicts. We can find all local conflicts involving f in time $O(n^4)$ as follows. Since, following Lemma 1, only three topologies involving five taxa can form a local conflict, it suffices to consider all size four sets of taxa $\{a, b, c, d\} \subseteq S$ together with the chosen $f \in S$. There are five quartets over this size five set of taxa, namely, $\{a, b, c, d\}$, $\{a, b, c, f\}$, $\{a, b, d, f\}$, $\{a, c, d, f\}$, and $\{b, c, d, f\}$. For the topologies of these quartets, we can test, in constant time, whether there are three among them that are not tree-consistent. Doing so for every choice of $\{a, b, c, d\} \subseteq S$, we will find *all* local conflicts involving f. We summarize these considerations in the following lemma.

Lemma 3. *If we are given a set S of taxa, some taxon $f \in S$, and a complete set Q_S of quartet topologies that is not tree-consistent, then Q_S has at least one local conflict involving f and all local conflicts involving f can be found in time $O(n^4)$.*

4 Combinatorial Characterization of Local Conflicts

Given three topologies, we need to decide whether they are tree-consistent or not. Directly using the definition of tree-consistency turns out to be a rather technical, troublesome task, since we have to reason whether or not a tree topology exists that induces the topologies. Similarly, it can be difficult to test, for the topologies, whether or not they contradict the substitution property. To make things less technical and easier to grasp, we subsequently give a combinatorial characterization of local conflicts. Note that in the following definition we distinguish two possible *orientations* of a quartet topology $[ab|cd]$, namely $[ab|cd]$ and $[cd|ab]$.

Definition 1. *Given a set of topologies where each of the topologies is assigned an orientation, let l be the number of different taxa occurring in the left hand sides of the topologies and let r be the number of different taxa occurring in the right hand sides of the topologies.*

The signature *of the set of topologies, then, is the pair (l, r) that, over all possible orientations for these topologies, minimizes l.*

Theorem 2. *Three quartet topologies are not tree-consistent iff they involve five taxa and their signature is $(3, 4)$ or $(4, 4)$.*

Proof. ($\Rightarrow$) We show that, given three topologies t_1, t_2, t_3 which are not tree-consistent, they involve five taxa and have signature $(3, 4)$ or $(4, 4)$. By Lemma 2, we know that three topologies are not tree-consistent iff they contradict the substitution property. Three topologies contradict the substitution property if for one of these topologies, w.l.o.g., $t_1 = [ab|cd]$, neither the topology t_2 for quartet $\{a, b, c, e\}$ is $[ab|ce]$ nor the topology t_3 for quartet $\{a, c, d, e\}$ is $[ae|cd]$. Therefore, the topology t_2 is either $[ac|be]$ or $[ae|bc]$, and the topology t_3 is either $[ac|de]$ or $[ad|ce]$. By exhaustively checking the possible combinations, we can find that the topologies involve five taxa and their signature is $(3, 4)$ (e.g., for $t_2 = [ac|be]$ and $t_3 = [ac|de]$) or $(4, 4)$ (e.g., for $t_2 = [ac|be]$ and $t_3 = [ad|ce]$).

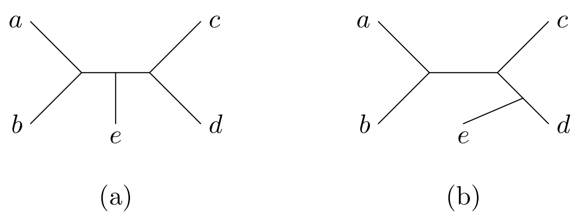


Figure 2: Trees inducing non-conflicting topologies in the proof of Theorem 2.

($\Leftarrow$) We are given three topologies t_1, t_2, t_3 involving five taxa and having signature $(3,4)$ or $(4,4)$. Assume that they are tree-consistent. Showing that this would imply signature $(2,3)$ or $(3,3)$, we prove that they cannot be tree-consistent. For tree-consistent t_1, t_2, t_3, we can find a tree inducing them. Given, w.l.o.g., taxa $\{a,b,c,d,e\}$ and $t_1 = [ab|cd]$, we essentially have two possibilities: we can attach the leaf e on the middle edge of topology t_1 as shown in Figure 2(a), or we can attach e on one of the four side branches of t_1 as exemplarily shown in Figure 2(b). Considering the set of quartet topologies induced by these trees, we find in each case that the set has signature $(3,3)$ or $(2,3)$. For instance, the topologies induced by the tree in Figure 2(a) are, besides t_1, $[ab|ce]$, $[ab|de]$, $[ae|cd]$, and $[be|cd]$. Three topologies selected from these, have signature $(3,3)$ (e.g., $[ab|cd]$, $[ab|ce]$, and $[ae|cd]$) or $(2,3)$ (e.g., $[ab|cd]$, $[ab|ce]$, and $[ab|de]$). $\square$

Using Theorem 2, we can determine whether three topologies are conflicting by simply counting the involved taxa and computing their signature.

5 Fixed-Parameter Algorithm for MQI

We describe a recursive algorithm for MQI in its four main parts: building the conflict list, the search tree, branching, and updating the conflict list.

Building the conflict list. We initially build the *conflict list* C of local conflicts, i.e., a list of three-sets of quartets whose topologies are not tree-consistent. By Lemma 3, it is sufficient to build a list of local conflicts containing some designated taxon, which can be chosen arbitrarily.

The search tree. The recursive procedure is outlined in Fig. 3. The procedure selects a local conflict from the conflict list and tries to resolve it by changing one of its topologies. After such a change it updates the conflict list (described below), and calls the recursive procedure with argument $k-1$ on the thereby created subcase. In the next paragraph, we will explain how to select and change the topologies in this branching and we will find that it is sufficient to branch into four subcases. The recursion stops if no conflicts are left in the conflict list (we have found a solution), or if $k = 0$ (with a non-empty conflict list we did not find a solution in this branch of the search tree). When a solution is found, the algorithm outputs the current, complete and tree-like set of topologies. From this, it is possible to derive the evolutionary tree in time $O(n^4)$ [4]. Scanning the whole search tree, we can find *all* solutions that can be obtained by altering k topologies when k is optimal.

Branching. By a careful selection of subcases to branch into, we can explore all ways to resolve an arbitrarily selected local conflict with only four recursive calls. Let t_1, t_2, t_3 be the three topologies which are not tree-consistent and which form the local conflict. By Lemma 2, t_1, t_2, t_3 contradict the substitution property. Recall that, given $[ab|cd]$, the substitution property (Proposition 1) requires topology $[ab|ce]$ or topology $[ae|cd]$. Therefore, we can assume the following setting for the three quartets contradicting the substitution property: Topology $t_1 = [ab|cd]$, topology t_2 is the topology for quartet $\{a, b, c, e\}$ different from $[ab|ce]$, and topology t_3 is a topology for quartet $\{a, c, d, e\}$ different from $[ae|cd]$. In order to change the three topologies such that they satisfy the substitution property, we have the following possibilities. We can change t_1; either (1) we change t_1 to $[ac|bd]$, or (2) we change t_1 to $[ad|bc]$. The cases in which t_1 is not changed remain to be covered. With unchanged t_1, we have to (3) change t_2 to $[ab|ce]$ or (4) change t_3 to $[ae|cd]$, because these are the only remaining possibilities to satisfy the substitution property.

Updating the conflict list. The main effort in every node of the search tree is to update the conflict list after changing a topology. In Fig. 3, this is done by instruction `update(`C`, `t'`)`, called with a conflict list C and a changed topology t' as arguments. We search, when changing t', the "neighborhood" of t', and update the conflict list: We (1) remove the three-sets of quartets in the list whose topologies are now tree-consistent, and (2) add the three-sets of quartets not in the list whose topologies now form a local conflict.

Correctness. To obtain a non-conflicting set of quartet topologies, we have to, following Theorem 1, resolve all local conflicts. Such a local conflict can be removed by altering (at least) one of the three involved quartet topologies. The recursive procedure has to try every possibility to resolve the local conflict in order to find every possible solution. If there is a solution, we will find it by the described branching strategy. If for none of the three topologies we can find a solution while altering the topology, the conflict cannot be removed.

Running time. Initially building the conflict list takes time $O(n^4)$ by Lemma 3. Using the conflict list, we can, in constant time, access a local conflict and determine the cases to branch into. Since it is sufficient to branch into four subcases for a local conflict, and, since, in every subcase, we decrease the parameter k by one, we obtain a search tree size at most 4^k.

With n species, updating the list of conflicting size three sets can be done in time $O(n)$: Following Lemma 1, local conflicts can only occur among three topologies consisting of no more than five taxa. Therefore, having changed the topology of one quartet $\{a, b, c, d\}$, we only have to examine the "neighborhood" of the quartet, i.e., those sets of five taxa containing all of a, b, c, d. For every such set of five taxa it can be examined in constant time whether for three topologies over the five taxa a new conflict emerged or whether an existing conflict has been resolved. Given taxa a, b, c, d, we have $n - 4$ choices for a fifth taxon. Thus, $O(n)$ is an upper bound for the update procedure.[2]

Altogether, we obtain:

Theorem 3. MQI *can be solved in time* $O(4^k \cdot n + n^4)$.

Our algorithm has only $O(n^4)$, i.e., optimal, memory requirement, where the input size

[2]In fact, as explained in Section 3, we only consider sets of five species containing a designated taxon f. Therefore, if we change the topology of a quartet $\{a, b, c, d\}$ which does not contain the designated taxon f, then we only have to consider *one* set of five topologies, namely $\{a, b, c, d, f\}$. In this special case, the update procedure can be done in time $O(1)$.

```
resolve(complete set of quartet topologies $Q$,
        integer $k$,
        conflict list $C$) {

(A0) if $C$ is empty then:
     We are done!  Output the current set of quartet topologies and stop;

(A1) if $(k <= 0)$ then return; /* more than $k$ recursions */

(A2) Take $c \in C$, with $c = \{t_1, t_2, t_3\}$ such that
       the substitution property states "$t_1 \in Q \Rightarrow (t'_2 \in Q$ or $t'_3 \in Q)$"
       where $t'_2$ and $t'_3$ are alternative topologies for $t_2$ and $t_3$, resp.

         for the two alternative topologies $t'_1$ of $t_1$ do
                 $C_{new}$ = Update($C$, $t'_1$);
                 resolve($Q$-$t_1$+$t'_1$, $k-1$, $C_{new}$);
         end do

         $C_{new}$ = Update($C$, $t'_2$);
         resolve($Q$-$t_2$+$t'_2$, $k-1$, $C_{new}$);

         $C_{new}$ = Update($C$, $t'_3$);
         resolve($Q$-$t_3$+$t'_3$, $k-1$, $C_{new}$);

         return; /* no success in current branch
                     -> step one level up in recursion */

}
```

Figure 3: Outline in pseudocode of a recursive procedure for eliminating conflicts by changing at most k quartet topologies (if possible). Further explanation is given in Section 5.

is already $O(n^4)$: Firstly, we have to store the topology for every of the $O(n^4)$ quartets. Secondly, we maintain the conflict list; as observed in Section 3, the size of the conflict list is $O(n^4)$. Finally, we have to keep track of the topologies changed on the path from the root of the search tree to the current search tree node; these are at most $k < n^4$ topologies.

6 Improving the Running Time in Practice

In Section 6.1, we collect some ideas for improvements still maintaining the algorithm's optimality. In Section 6.2, sacrificing guaranteed optimality, we propose to combine the algorithm with existing methods that infer strongly supported parts of a tree.

6.1 Enhancements Maintaining Optimality

Fixing topologies. Once a topology has been altered, we will, in subsequent stages of recursion, never alter it again. We call this *fixing* the topology. This will avoid redundant branchings in the search tree.
Forcing topologies to change. In contrary to the fixing of topologies, it might be possible to identify topologies which necessarily have to be altered in order to find a solution. We call this *forcing* a topology to change. The ideas described here are similar to those used in the so-called reduction to problem kernel of the 3-Hitting Set problem [22]. Here, however, they will not yield a reduction to problem kernel for our problem. Nevertheless, they are likely to result in a better performance of the algorithm since they allow recognizing situations in which we cannot find a solution and they also allow a better branching.

Lemma 4. *Consider an instance of* MQI *in which quartet q has topology t. If there are more than $3k$ distinct local conflicts which contain t, then in a solution for this instance the topology for q is different from t.*

Proof. We have shown in Section 3 that three topologies only can form a local conflict, if there are not more than five taxa occurring in them (Lemma 1). For five taxa, there are five quartets consisting of these taxa. Therefore, when we are given two quartet topologies t_1 and t_2 and if there are more than five taxa occurring in t_1 and t_2, they cannot form a conflict with a third topology. If there are exactly five taxa occurring in t_1 and t_2, then there are five quartets consisting of these five taxa, two of which are the quartets for t_1 and t_2. The remaining three topologies are the only possibilities for a topology t_3 that could form a conflict with t_1 and t_2.

Now, consider the situation in which, for a quartet topology t, we have more than $3k$ distinct local conflicts which contain t. We show by contradiction that we have to alter topology t to find a solution. Assume that we can find a solution while not altering t. By changing a topology t', we can cover at most three conflicts containing t since there are at most three local conflicts containing both t and t'. Therefore, by changing k topologies, we can resolve at most $3k$ local conflicts. This contradicts our assumption and shows that we have to alter t to find a solution. □

We call the topologies obtained from Lemma 4 "forced to change," and mark them appropriately in order to take them into consideration in the next branching situation.

Recognizing hopeless situations. In this paragraph, we describe situations in which, at some level in the search tree where we are allowed to alter at most k topologies, we cannot find a solution. This will allow us to avoid branching into further (useless) subcases. Having a local conflict consisting only of fixed topologies, we obviously cannot resolve this conflict while not changing one of the fixed topologies.

If, after identifying the topologies forced to change, there are more than k of them, it is obvious that a solution is not possible—already by changing these topologies we would change more topologies than we are allowed to.

The following two lemmas contain more involved observations. If a local conflict does not contain a topology which is forced to change, then we call it an *unforced* local conflict.

Lemma 5. *Let us have an instance of* MQI *in which we have identified p conflicts which are forced to change. If the number of unforced local conflicts is greater than $3(k-p)k$, then the instance has no solution.*

Proof. We have to change the p topologies that are forced to change. We, therefore, decrease the parameter by p and have the possibility to resolve all local conflicts containing such a topology. The conflicts which certainly remain to be resolved are the unforced conflicts. From the preceding paragraph we know that, by changing a topology, we can resolve at most $3k$ distinct local conflicts. Therefore, by altering $(k-p)$ topologies, we can resolve at most $3(k-p)k$ distinct local conflicts. □

Lemma 6. *An instance of* MQI *in which the number of local conflicts is greater than $6(n-4)k$ has no solution.*

Proof. By Lemma 1, local conflicts can only arise between three topologies that do not involve more than five taxa. Thus, given a quartet $q = \{a, b, c, d\}$ with topology t, a local conflict can arise with other quartets involving taxa from $\{a, b, c, d, e\}$ for some e. Since e has to be different from a, b, c, and d, there are $n-4$ choices for this taxon e. There are five quartets over $\{a, b, c, d, e\}$ and four of them excluding the given q. We have $\binom{4}{2} = 6$ ways to choose two from these four quartets in order to form size three sets containing q. Therefore, by altering one topology, we can resolve at most $6(n-4)$ distinct local conflicts, and by altering k topologies at most $6(n-4)k$ distinct local conflicts. □

6.2 Fixing Strongly Supported Edges in Advance

To improve the performance of exact fixed-parameter algorithms in practice, it is reasonable to combine them with known heuristics. For MQI, we propose to preprocess the quartet topologies using methods that infer strongly supported edges of the tree. Examples of those methods include the Q*-method [4], quartet cleaning [6, 12], or hypercleaning [5]. The advantage of the methods is that they are fast (e.g., hypercleaning runs in $O(n^5)$). Their disadvantage is that they resolve only the strongly supported part of the tree's edges; on unfavorable input they resolve no edges at all. We can take advantage of their output, however, by fixing the topologies induced by the inferred edges. On this modified input, we run our MQI algorithm in order to completely resolve the partly resolved tree. Even a small set of fixed edges can significantly prune our search space.

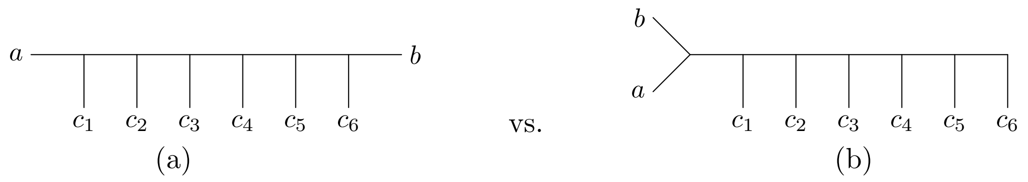


Figure 4: Example illustrating where the result from the Q* method leads to a suboptimal solution for MQI.

Preprocessing by the Q*-method. Here, we describe the use of the Q*-method by Berry and Gascuel [4] as a preprocessing for our algorithm. The Q*-method produces the maximum subset of the given quartet topologies that is tree-like. In the combined use with our algorithm, we fix these quartet topologies from the beginning. The tree we obtain will be a refinement of the tree reported by the Q*-method.

We cannot guarantee that the reported tree is the optimal solution for MQI as illustrated by the following example. Consider the tree in Fig. 4(a) on taxa $\{a, b, c_1, c_2, \dots, c_6\}$. Suppose we are given the complete set of 70 quartet topologies, where the 55 quartets containing at least three taxa from $\{c_1, c_2, \dots, c_6\}$ have the topology induced by the tree, but the remaining 15 quartets have the topology $[ab|c'c'']$ where $c'c'' \in \{c_1, c_2, \dots, c_6\}$ and $c' \neq c''$. On this input, the Q*-method infers the bipartition $ab|c'c''$ for the $c'c'' \in \{c_1, c_2, \dots, c_6\}$; then applying the MQI algorithm leads to a solution for $k = 20$, e.g., the tree shown in Fig. 4(b). The optimal solution of MQI, however, would be the tree depicted in Fig. 4(a) for $k = 15$.

The examples where the Q*-method yields misleading results are quite artificial. On real data, these mistakes are unlikely: before reporting misleading edges, i.e., edges not belonging to an optimal solution, the Q*-method will rather report no edge at all. Our experiments described in Section 7 support our conjecture that with the preprocessing by the Q*-method we find every solution that the MQI algorithm would find. Moreover, the experiments show that this enhancement allows us to process much larger instances than we could without using it.

7 Experimental Evaluation

To investigate the usefulness and practical relevance of our algorithm, we performed experiments on synthetic as well as on real data from fungi. The implementation of the algorithm was done using the programming language C. The algorithm contains the enhancements described in Section 6. The combined use with the Q*-method is, however, only applied when processing the fungi data, not when processing the synthetic data. The reported tests were done on a LINUX PC with a Pentium III 750 MHz processor and 192 MB main memory.[3]

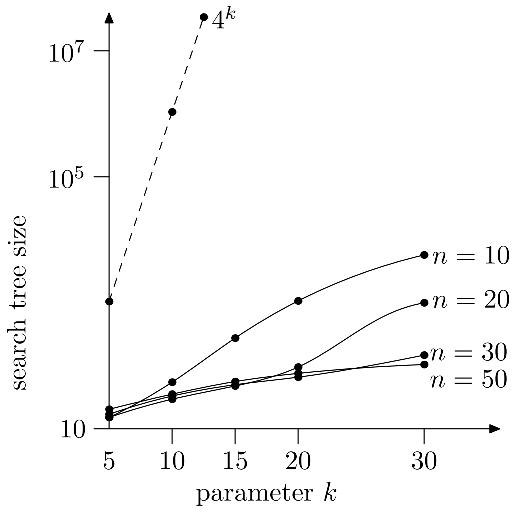
7.1 Synthetic Data

We performed experiments on artificially generated data in order to get some idea about the practical contexts in which our algorithm might be useful. For n given taxa n and

[3]Differences to the results of this paper's conference version are due to improvements of the implementation.

n	k	time	search tree size
10	10	0.02 sec	58
	20	0.29 sec	1070
	50	26 sec	104297
20	10	0.13 sec	30
	20	0.16 sec	94
	50	25 sec	34693
	100	50 min	4460508
30	20	0.72 sec	66
	50	9 sec	7007
	100	31 min	1519700
50	50	7 sec	186
	100	8 min	45514
	150	<5 h	6119560

(a)



(b)

Figure 5: Table (a) displays the results of our algorithm on MQI instances for different values of n and k. We give processing time and the size of the scanned search tree. Figure (b) displays, on a logarithmic scale, the difference of the theoretical 4^k bound (dashed) and the real search tree size (solid lines). Each solid line shows, for a fixed number of taxa n, how the search tree size increases for increasing values of k.

parameter k, we produce a data file as follows. We generate an evolutionary tree by recursively joining randomly selected subtrees. The subtrees are selected from a set which, initially, contains only the one-node subtrees corresponding to the taxa. When two subtrees are joined, we replace them in the set by the newly generated subtree. This procedure, finally, yields a tree for n taxa and we derive the quartet topologies from that tree. Then, we change k distinct, arbitrarily selected topologies in a randomly chosen way. For different pairs of values for n and k, ten different data sets were created for each pair. The reported results are the average for test runs on ten data sets.

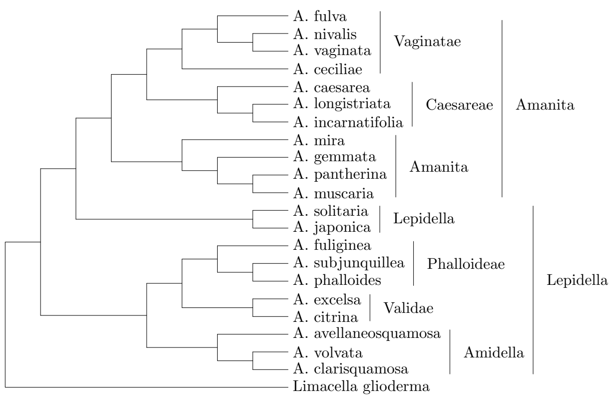
We experimented with different values of n and k. As a measure of performance, we use two values: We report the processing time and, since processing time is heavily influenced by system conditions, e.g., memory access time in case of cache faults, also the search tree size. The search tree size is the number of the search trees nodes.

Figure 5(a) gives a table of results for different values of n and k. We could process large instances of the problem, e.g., $n = 50$ and $k = 100$ in eight minutes. Regarding the search tree size, we compare in Figure 5(b), on a logarithmic scale, the theoretical upper bound of 4^k to the real size of the search tree. The search trees are, by far, smaller than the 4^k bound. This is mainly due to the practical improvements of the algorithm (see Section 6). We also note that, for equal value of k, a higher number of taxa n results in a smaller search tree, if the value of k exceeds some turning point.

	running time in sec.	
n	no Q*	with Q*
8	0.46	0.36 (21%)[a]
9	3.41	0.85 (32%)
10	35.96	2.68 (38%)
11	617.56	4.11 (41%)
12	7039.82	5.44 (43%)

[a]Percentage of quartet topologies fixed by Q*

(a)



(b)

Figure 6: (a) Speed-up when using Q* preprocessing. (b) Optimal tree found for a set of 21 *Amanita* species and one outgroup taxon; indicated is the grouping of *Amanita* species into 7 sections and 2 subgenera.

7.2 Real Data

Using our algorithm, we analyzed the evolutionary relationships of species from the mushroom genus *Amanita*, a group that includes well-known species like the Fly Agaric and the Death Cap. The underlying data are an alignment of nuclear DNA sequences coding for the D1/D2 region of the ribosomal large subunit (alignment length 576) from *Amanita* species and one outgroup taxon, as used by Weiß *et al.* [26, 27]. We inferred the quartet topologies by (1) using `dnadist` from the Phylip package [16] to compute pairwise distances with the maximum likelihood metric, and (2) using `distquart` from the Phyloquart package [4] to infer quartet topologies based on the distances.

The construction of the evolutionary tree was done by a preprocessing of the data using the Q*-method, which was also taken from the Phyloquart package. Experiments on small instances, e.g., 10 taxa, show that all solutions we find without using the Q*-method are also found when using it. Using the Q*-method, however, results in a significant speed-up of the processing. Figure 6(a) shows this impact for small numbers of *Amanita* species. Note, however, that the speed-up heavily depends on the data. In Figure 6(a) and in the following, we neglect the time needed for the preprocessing by the Q*-method, which is, e.g., 0.11 seconds for $n = 12$.

Our recursive algorithm processed a set of $n = 22$ taxa in 23 minutes (which includes 0.2 seconds needed by Q*). The resulting tree was rooted using the outgroup taxon *Limacella glioderma* and is displayed in Figure 6(b). The first k-value for the given 7315 quartet topologies for which we found a solution is $k = 978$. The Q*-method had fixed 41 percent of the quartet topologies in advance. Considering the tree, the grouping of taxa is consistent with the grouping into seven sections supported by Weiß *et al.* [27], who used the distance method neighbor joining, heuristic parsimony methods, and maximum likelihood estimations. Particularly, our grouping is nearly identical to the topology revealed by Weiß *et al.* using maximum likelihood estimation. This topology is well compatible with classification concepts based on morphological characters, e.g., the sister group relationship of sections *Vaginatae* and *Caesareae*, and the monophyly of subgenus *Amanita* which were not supported in the neighbor joining and heuristic parsimony analysis.

To contrast our results with another heuristic method, we ran quartet puzzling with 1000 puzzling steps on the same data set. The resulting tree is computed in 1.3 minutes and also reflects the mentioned grouping of taxa into the seven subsections. However, it cannot resolve the grouping of subsections and does, in particular, not indicate the division into two sections.

One might hope that quality of quartet inference techniques will improve in the future. This would lead to instances requiring smaller values of k.

8 Conclusion

We showed that Minimum Quartet Inconsistency can be solved in worst case time $O(4^k n + n^4)$, meaning that the problem is fixed-parameter tractable for parameter k denoting the number of faulty quartet topologies. Several ideas for tuning the algorithm show that its practical performance is much better than the theoretical bound given above (in particular, concerning the size of the search tree, 4^k). This is clearly expressed by our experimental results in Section 7. The combination of heuristic methods, inferring the strongly supported edges of a tree, and exact algorithms seems to result in trees of high quality and, therefore, deserves further attention. This combination is a good trade-off between the heuristics, which are fast but result in only partially resolved trees, and the exact algorithms, which result in completely resolved trees but have exponential running times.

Concerning future work on MQI, it remains to extend our experiments to weighted quartet topologies (to which our algorithm can be generalized to, see [17] for more details) and to other (non-fungi) taxa. From a parameterized complexity point of view, it remains open to find a so-called reduction to problem kernel, which is a kind of preprocessing that shrinks the input the search tree has to deal with in advance (see [13] for details). As suggested by Chor [10], we might consider other parameters that can also deal with the Sparse MQI problem, e.g., asking whether we can satisfy $(m/3) + k$ quartets for m given quartet topologies. It might also help to identify parameters inherent to the problem. This could possibly enable us to find efficient solutions, e.g., when facing a fixed portion $c \cdot m$, for some constant $0 < c < 1$, of quartet errors. Since MQI can be solved in polynomial time for $k < (n-3)/2$ [6], we can ask—in the "spirit of parameterizing above guaranteed values" [21]—whether it is fixed-parameter tractable to find a tree that violates at most $(n-3)/2 + k$, $k \geq 0$, quartet topologies.

Acknowledgement. We are grateful to Michael Weiß from the Biology Department (Systematic Botany and Mycology group), Universität Tübingen, for providing us with the *Amanita* data, for supporting us in the interpretation of our results, and for many constructive remarks improving the presentation significantly. We thank Mike Steel for pointing us to the guaranteed value for parameter k implied by quartet cleaning algorithms. Finally, we thank the anonymous referees of *Journal of Computer and Systems Sciences* for helpful comments improving the presentation.

References

[1] J. Alber, J. Gramm, and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. *Discrete Mathematics*, 229(1-3):3–27, 2001.

[2] H.-J. Bandelt and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Advances in Applied Mathematics*, 7:309–343, 1986.

[3] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg. Constructing phylogenies from quartets: elucidation of eutherian superordinal relationships. *Journal of Computational Biology*, 5:377–390, 1998.

[4] V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Theoretical Computer Science*, 240:271–298, 2000. Software available through `http://www.lirmm.fr/~vberry/PHYLOQUART/phyloquart.html`.

[5] V. Berry, D. Bryant, T. Jiang, P. Kearney, M. Li, T. Wareham and H. Zhang. A practical algorithm for recovering the best supported edges of an evolutionary tree. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 287–296, 2000.

[6] V. Berry, T. Jiang, P. Kearney, M. Li, and T. Wareham. Quartet cleaning: improved algorithms and simulations. In *Proceedings of the 7th European Symposium on Algorithms (ESA)*, number 1643 in Lecture Notes in Computer Science, pages 313–324, 1999. Springer-Verlag.

[7] D. Bryant and M. Steel. Constructing optimal trees from quartets. *Journal of Algorithms*, 38:237–259, 2001.

[8] P. Buneman. The recovery of trees from measures of dissimilarity. In Hodson *et al.*, editors, *Anglo-Romanian Conference on Mathematics in the Archaeological and Historical Sciences*, pages 387–395, 1971. Edinburgh University Press.

[9] B. Chor. From quartets to phylogenetic trees. In *Proceedings of the 25th Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, number 1521 in Lecture Notes in Computer Science, pages 36–53, 1998. Springer-Verlag.

[10] B. Chor. Personal communication. August 2001.

[11] H. Colonius, H. H. Schultze. Tree structure for proximity data. *British Journal of Mathematical and Statistical Psychology*, 34:167–180, 1981.

[12] G. Della Vedova and H. T. Wareham. Optimal algorithms for local vertex quartet cleaning. In *Proceedings of the 17th ACM Symposium on Applied Computing*, to appear, 2002.

[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.

[14] R. G. Downey and M. R. Fellows and U. Stege. Parameterized complexity: a framework for systematically confronting computational intractability. In Graham *et al.*, editors, *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 49:49–99, 1999. AMS Press.

[15] M. R. Fellows. Parameterized complexity: the main ideas and some research frontiers. In *Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC)*, number 2223 in Lecture Notes in Computer Science, pages 291–307, 2001. Springer-Verlag.

[16] J. Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Department of Genetics, University of Washington, Seattle. 1993. Available through `http://evolution.genetics.washington.edu/phylip`.

[17] J. Gramm and R. Niedermeier. Minimum Quartet Inconsistency is fixed parameter tractable. Technical Report WSI-2001-3, WSI für Informatik, Universität Tübingen, Fed. Rep. of Germany, January 2001. Report available through `http://www-fs.informatik.uni-tuebingen.de/~gramm/publications`.

[18] T. Jiang, P. Kearney, and M. Li. Orchestrating quartets: approximation and data correction. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 416–425, 1998.

[19] T. Jiang, P. Kearney, and M. Li. Some open problems in computational molecular biology. *Journal of Algorithms*, 34:194–201, 2000.

[20] T. Jiang, P. Kearney, and M. Li. A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM Journal on Computing*, 30(6):1942-1961, 2001.

[21] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31:335–354, 1999.

[22] R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 2(1):93–107, 2002.

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

[24] M. Steel. Personal communication. May 2001.

[25] K. Strimmer and A. von Haessler. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13(7):964–969, 1996.

[26] M. Weiß. Molecular investigations on phylogeny in the genus Amanita. Dissertation, Universität Tübingen, Fed. Rep. of Germany, 1999.

[27] M. Weiß, Z. Yang, and F. Oberwinkler. Molecular phylogenetic studies in the genus Amanita. *Canadian Journal of Botany*, 76:1170–1179, 1998.