Minimum Quartet Inconsistency is Fixed Parameter Tractable

Jens Gramm and Rolf Niedermeier

WSI-2001-3

Jens Gramm

Wilhelm-Schickard-Institut
für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

email:gramm@informatik.uni-tuebingen.de
Telefon: (07071) 29-77569
Telefax: (07071) 29-5061

Rolf Niedermeier

Wilhelm-Schickard-Institut
für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

email:niedermr@informatik.uni-tuebingen.de
Telefon: (07071) 29-77568
Telefax: (07071) 29-5061

ISSN 0946-3852

Minimum Quartet Inconsistency is Fixed Parameter Tractable

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

We study the parameterized complexity of the problem to reconstruct a binary (evolutionary) tree from a complete set of quartet topologies in the case of a limited number of errors. More precisely, we are given n taxa, exactly one topology for every subset of 4 taxa, and a positive integer k (the parameter). Then, 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 version of the problem where we are not necessarily given a topology for every subset of four taxa appears to be fixed parameter intractable. For the MQI problem, 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, our algorithm can produce *all* solutions that resolve at most k errors. To this end, we point out some nice combinatorial properties of the problem, e.g., that "global" conflicts can be always led back to "local" ones. Additionally, we discuss fixed parameter tractability of variations of the problem and significant heuristic improvements. Experiments underline the practical relevance of our solutions. E.g., they show that in practice a much smaller exponential growth can be achieved than the upper bound predicts.

1 Introduction

In recent years, quartet methods for reconstructing evolutionary trees have received considerable attention in the computational biology community [10, 17]. In comparison with other phylogenetic methods, an advantage of quartet methods is, e.g., that they can overcome the data disparity problem (see [10] for details). This approach is based on the fact that an evolutionary tree is uniquely characterized by its set of induced quartet topologies [9]. Herein, we

*Work supported by the DFG projects "KOMET," LA 618/3-3, and "OPAL" (optimal solutions for hard problems in computational biology), NI-369/2-1.

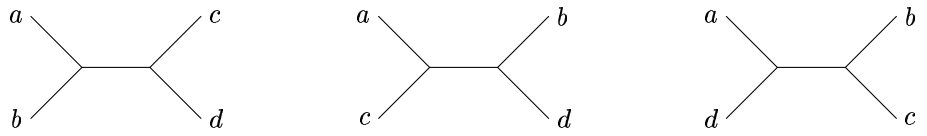


Figure 1: Possible quartet topologies for quartet $\{a, b, c, d\}$, which are (from left to right) $[ab|cd]$, $[ac|bd]$, and $[ad|bc]$.

consider an *evolutionary tree* to be a binary tree T in which the leaves are bijectively labeled by a set of taxa S. A quartet, then, is a size four subset $\{a, b, c, d\}$ of S and the topology for $\{a, b, c, d\}$ induced by T simply is the four leaves subtree of T induced by $\{a, b, c, d\}$. The three possible quartet topologies for $\{a, b, c, d\}$ are $[ab|cd]$, $[ac|bd]$, and $[ad|bc]$ and are shown in Figure 1.[1] The fundamental goal of quartet methods is, given a set of quartet topologies, to reconstruct the corresponding evolutionary tree. The computational interest in this paradigm derives from the fact that the given set of quartet topologies usually is incomplete, contains 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 following, perhaps most often studied optimization problem in the context of quartet methods. (In Section 2, we will survey other problems and results concerning the quartet paradigm.) The Minimum Quartet Inconsistency (MQI) problem is defined as follows.

> Minimum Quartet Inconsistency (MQI)
> **Input:** A set S of n taxa and a set Q_S of quartet topologies such that there is exactly one topology for *every* quartet set corresponding to S.[2]
> **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 [18]. Concerning the approximability of MQI, it is known that it is polynomial time approximable with a factor n^2 [6, 7, 17]. It is an open question of [17] whether MQI can be approximated with a factor at most n or even with a constant factor. The parameterized complexity [12] of MQI, however, so far, has apparently been neglected—we close this gap here. Assuming 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*

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

[2] Note that given n species, there are $\binom{n}{4} = O(n^4)$ corresponding quartet topologies.

parameter tractable; that is, MQI can be solved exactly in worst case time $O(4^k n + n^4)$. Observe that the input size is $O(n^4)$. It is worth noting here that the 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 Bandelt and Dress [3]. This forms the basic observation in order to show fixed parameter tractability of MQI. Moreover, our approach makes it possible to construct *all* evolutionary trees that can be (uniquely) obtained from the given input by changing at most 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 running time of the algorithm in practice significantly.

We performed several experiments on artificial and real (fungi) data and, thereby, showed that our algorithm (due to several tuning tricks) 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 40 minutes on a LINUX PC with a Pentium III 750 MHz processor and 192 MB main memory.

Our work is structured as follows. In Section 2, we provide some more motivation and background from computational biology and parameterized complexity theory. Afterwards, in Section 3 and 4, we develop some combinatorial properties of the MQI problem that lead to its fixed parameter algorithm in Section 5. Section 6 gives some practical improvements of the algorithm. In Section 7, we present extensions of our main result, dealing with generalizations of the problem. Section 8 describes our experiments. We conclude the paper with prospects for future research.

2 Preliminaries

In this section, we give more background on quartet methods (in particular from the viewpoint of computational biology) and parameterized complexity and we introduce notation that is used throughout the paper.

2.1 From quartets to evolutionary trees

For a given a set of n taxa, it is an important goal to determine the evolutionary relationship of the taxa, e.g., based on DNA or protein sequence data. This relationship is often displayed as a binary tree with a 1:1 labelling of the tree's leaves with the taxa. A variety of models and methods has been proposed for solving this question [10]. Most of the problems being considered to produce practically relevant results are, however, NP-hard and computationally expensive, allowing a solution only for a limited number of taxa.

The quartet method infers the evolutionary tree only for four taxa, called a *quartet*, at a time. Once having determined the evolutionary tree for every quartet of taxa, the quartet method tries to combine these evolutionary trees involving four taxa, called *quartet topologies*, in order to obtain a tree containing all taxa. Summarizing, we can distinguish two steps in quartet methods:

- Infer the quartet topologies for all quartets of given taxa.
- Recombine the quartet topologies to build an evolutionary tree for the whole set of taxa.

There are several reasons why the quartet method is widely used in practice. [3] Its motivation is the fact that an evolutionary tree is uniquely characterized by the quartet topologies for its size four sets of taxa [9]. From this set of topologies, we can efficiently compute the tree in polynomial time $O(n^4)$ [5], as will be explained in more detail in one of the following paragraphs. Besides this, the quartet method clearly divides 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 the quartet methods is data disparity as discussed by Chor [10]: 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. The quartet method, however, allows us to use the maximum amount of information available for the four taxa of a quartet when we compute its quartet topology.

Besides its advantages, the potential of the quartet method has its limitations, caused by the fact that it is not possible to build a tree from every set of topologies. Since the process of inferring the quartet topologies can be erroneous, 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, it is desirable to to find an optimal tree

[3] Note that St. John *et al.* [22] give a rather critical exposition of the practical performance of quartet methods (in particular, quartet puzzling) in comparison with the neighbor joining method. We believe, however, that the arguments given here show that the study of quartet methods still makes sense.

that matches the inferred topologies as "close" as possible. One approach is to detect and correct erroneous quartet topologies if their number is locally bounded, e.g., as it is done in *quartet cleaning* [7], explained in one of the following paragraphs. Another approach is to minimize the overall number of changes necessary to obtain a set of quartet topologies for which it is possible to construct a tree. The latter is exactly the question of minimum quartet inconsistency we address here.

Inferring quartet topologies. Quartet topologies can be computed directly from the sequence data or from a distance matrix given for the involved taxa. We can use every method proposed for building evolutionary trees, and even methods that are, in general, very time-consuming, can be practicable when processing only four taxa at a time. Methods especially proposed for inferring quartets include, e.g., using the four-point condition [3], the short quartet method [14], the ordinal quartet method [19], and the maximum likelihood approach used in quartet puzzling [24].

Recombination of quartet topologies. 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 [5]. There are, however, situations, in which there is no such *binary* tree. In the following, we mention three methods to handle these situations by producing trees that are not fully resolved, i.e., that are not binary. Non-binary branchings remain in the tree where a binary branching cannot be obtained due to conflicts in the given topologies. For these approaches, we deal with *completely supported* [5] edges: Given a tree with its leaves bijectively labeled by the given set S of taxa, an edge in the tree defines a bipartition of S into sets A and B, each of them containing the taxa in the subtree rooted at one end of the edge. We call the edge completely supported if, for every $a, a' \in A$ and $b, b' \in B$, the given quartet topology for a, a', b, b' separates a and a' from b and b'. An approach due to Buneman [9] is to construct a tree which contains every fully supported bipartition. Berry and Gascuel [5] present an algorithm doing this in running time $O(n^5)$ and an improved and more involved solution having running time $O(n^4)$ for n being the number of taxa. Bryant and Steel [8] consider the case that, for each quartet, we are given either one or two topologies. Their approach is even considering the topologies to be weighted. For a given weight λ, they are then looking for a tree inducing a subset of the given topologies such that the added weights of the induced topologies are at least λ. In time $O(n^5)$, their algorithm computes such a tree or tells us, when no such tree exists.

The mentioned algorithms by Berry and Gascuel [5] and by Bryant and Steel [8] rely on the fact that the given set of topologies is complete. In the more general situation that 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].

Quartet cleaning. To overcome errors made when inferring the quartet topologies, methods have been proposed to correct these *quartet errors* if their number is bounded [7, 16, 18]. This way is called *quartet cleaning* and Berry *et al.* [7] present two algorithms, one working in the situation that the number of errors *across an edge* is bounded, and another working in the case that the number of errors *across a vertex* is bounded. In the previous paragraph, we have shown how an edge e in the tree yields a bipartition of the involved taxa. Let us call the resulting sets A_e and B_e. Then a quartet whose topology is $[aa'|bb']$ is called to be across an edge e iff $a, a' \in A_e$ and $b, b' \in B_e$. It is called to be across a vertex v iff it is across an edge e and v is one of the endpoints of e. The first of the two proposed algorithms corrects in time $O(n^4)$ the erroneous quartet topologies if, for each edge e in the tree to be constructed, their number is bounded by $\frac{1}{2}(|A_e| - 1)(|B_e| - 1)$. For the algorithm to work, this condition has to apply for every edge, and, therefore, the algorithm is called *global*. The second algorithm is called *local* as it corrects errors where the condition applies, and leaves the tree unresolved where the condition does not apply. It corrects in time $O(n^5)$ the erroneous quartet topologies across a vertex in the tree to be constructed if their number is bounded by $\frac{1}{4}(|A_e| - 1)(|B_e| - 1)$.

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. In case we are given exactly one quartet topology for every set of four taxa, this question gives the MQI problem. If there is not a quartet topology for necessarily every set of four taxa, the question is referred to as SPARSE MQI. To solve the SPARSE MQI problem, Ben-Dor *et al.* [4] propose two solutions, namely, a heuristic approach and an exact algorithm. The heuristic solution is based on semidefinite programming, and does not guarantee the optimal solution, but has a polynomial running time. The exact algorithm uses dynamic programming for finding the optimal solution. 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 running time of this approach, however, even with some further optimizations, is exponential, namely, $O(m3^n)$, where n is the number of species and m is the number of given quartet topologies. Ben-Dor *et al.* ran all their experiments on MQI instances, i.e., there was exactly one quartet topology for every set of four taxa. In that case, we have $m = O(n^4)$. The memory requirement is $\Theta(2^n)$. According to Jiang *et al.* [17] there is 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 maximize $|Q_T \cap Q|$ (Q_T being the set of quartet topologies induced by a tree T), possesses a polynomial time approximation scheme [16, 18].

2.2 Some notation.

Assume that we are given a set of n taxa S. For a quartet $\{a, b, c, d\} \subseteq S$, we refer to its possible quartet topologies by $[ab|cd]$, $[ac|bd]$, and $[ad|bc]$. These are the only possible topologies up to isomorphism, since, e.g., a topology $[ab|cd]$ is (under isomorphism) the same as topologies $[ab|dc]$, $[ba|cd]$, $[ba|dc]$, $[cd|ab]$, $[cd|ba]$, $[dc|ab]$, and $[dc|ba]$. 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* [3] if the 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* [3] if there exists a tree with $Q = Q_T$. Since an evolutionary tree is uniquely characterized by the topologies for all its quartets [9], 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.

2.3 Parameterized complexity

The theory of parameterized complexity has been chiefly developed by Downey and Fellows and some of their co-authors [12]. The leitmotif of parameterized complexity is that "not all forms of computational intractability are created equal." That is, sometimes the combinatorial explosion seemingly inherent in solutions to complex (e.g., NP-hard) problems often can be restricted to a "small" part of the input, the parameter. Computational biology is considered to be one of the (future) core fields of problems that deserve intensive parameterized complexity studies [2, 12, 13].

As an example of a parameterized problem, consider the NP-complete DOMINATING SET problem[4] for planar graphs with n vertices. This problem can be solved in time $O(11^k n)$ [11, 12] or, alternatively, in time $O(c^{\sqrt{k}} n)$ for $c = 2^{36\sqrt{34}}$ [1]. Ongoing work is trying to improve both the constants in the bases of the exponential terms. To put it in more computational complexity-theoretic terms, consider the class of parameterized problems that can be solved in deterministic time $f(k)n^{O(1)}$, called FPT. Herein, f may be an arbitrary (usually exponential or worse) function only depending on k, but not depending on the input size n. The complexity class FPT is the set of *fixed parameter tractable* problems. Thus, in a sense, FPT comprises the "good" parameterized problems. As to the "bad" parameterized problems (as, for instance, CLIQUE) which do not allow for FPT-algorithms, we only mention in passing that there is an intricate completeness program with a whole hierarchy

[4]For a graph $G = (V, E)$ a set $S \subseteq V$ is called *dominating* if every vertex $v \in V - S$ has at least one neighbor in S. The DOMINATING SET problem is, given a graph G and a positive integer k, the question whether the graph has a dominating set of size $\leq k$.

of complexity classes (so-called W-hierarchy) classifying these problems according to their growing "degree of parameterized intractability." We refer to the monograph of Downey and Fellows [12] for any details.

It is strongly hoped that the concept of fixed parameter tractability opens a new, practical possibility how to deal with the computational intractability of hard problems in many applications. In particular, this approach is aiming at finding *optimal* solutions efficiently whenever the parameter k (i.e., "the size of the optimum") is relatively small.

3 Global conflicts are local

In this section, we show, intuitively speaking, that a "global" conflict in a set of quartet topologies can be led back to a "local" one. Recall that a *local conflict* is a set of three quartet topologies that is not tree-consistent. Given a complete set of quartet topologies which is not tree-consistent, the results of Bandelt and Dress [3] imply that there already is a subset of only three quartet topologies which is not tree-consistent. Proposition 1 and Theorem 1 following later in this section will make this precise. This is the key to develop a fixed parameter solution for the problem: It is sufficient to examine the size three sets of quartet topologies and to recursively branch on these local conflicts, as will be explained in Section 5.

Proposition 1. *(Proposition 2 in [3]) Given a set of taxa S and a complete set of quartet topologies Q_S over these taxa, Q_S is tree-like iff the following so-called* substitution property *property holds for every five distinct taxa $a, b, c, d, e \in S$:*

$$[ab|cd] \in Q_S \text{ implies } [ab|ce] \in Q_S \text{ or } [ae|cd] \in Q_S.$$

The proof for Proposition 1 (given in [3]) relies on the "denseness" given in a complete set of topologies.

In the following, we show that in Proposition 1, we can replace the substitution property introduced by Bandelt and Dress with the more common term of tree-consistency. This is because, for an incomplete set of only three topologies, the substitution property is tightly connected to the tree-consistency of the topologies. We will state this in the following technical Lemmas 1 and 2 and later use it to give, in Theorem 1, another interpretation of Proposition 1.

The substitution property is stated for three topologies involving exactly five taxa. To put it, in Lemma 2, in relation to tree consistency, we first show that three topologies involving more than five taxa are tree-consistent.

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:

Case (1) A topology $t \in t_1, t_2$, and 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, and 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 also occur in t_2 and t_3, we replace in T the leaf b by an inner node having two leaves as its children, one labeled as a and the other as 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, and t_3, hence showing that t_1, t_2, and t_3 are tree-consistent.

Case (2) will cover those t_1, t_2, t_3 such that for each pair of topologies, there are exactly two taxa occurring in both topologies. Counting arguments make sure that one of these two cases has to apply. Assume that Case (1) does not apply: Then, we choose three quartets, each time choosing four from the ≥ 6 given taxa, and every taxon has to occur in at least two of the three quartets. This is only possible for exactly six taxa, when Case (2) applies. With more than six taxa one would necessarily have a taxon occurring in only one of the topologies which is handled in Case (1).

Case (2) For each pair of topologies from t_1, t_2, and 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, and t_3 (we omit the details here), we find that we always can find a tree inducing t_1, t_2, and 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 as given in Proposition 1 is not satisfied, 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, i.e., there are no three topologies contradicting the substitution property.

($\Leftarrow$) 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 that (1) t_1, t_2, and 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 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 which 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, and 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 $t_3 = [ae|bd]$. □

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: For the proof of Lemma 2, we list the possible topologies t_1, t_2, and t_3 over five taxa $\{a, b, c, d, e\}$, when $t_1 = [ab|cd]$ and $t_2 = [ac|de]$. This table shows that whenever t_1, t_2, and t_3 do not contradict the substitution property they are tree consistent, i.e, we can complete them to a tree-like complete set of topologies.

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 $[ab|cd]$, $[ab|ce]$, $[bc|de]$, $[cd|ef]$, 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. This will make clearer that "global" tree-consistency of a complete set of topologies reflects in "local" tree-consistency of every three topologies taken from this set.

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

Proof. Due to Lemma 2 we may replace the substitution property in Proposition 1 with tree constistency. This gives the result. □

When we have 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, track down a subset of three topologies that is not tree-consistent. Our goal will be to detect all these local conflicts. This will be the preprocessing stage of the algorithm that will be described in Section 5, in order to (try to) "repair" the conflicts in a succeeding stage of the algorithm. We can find all these local conflicts in time $O(n^5)$ as follows. Since, following Lemma 1, only three topologies involving five taxa can form a local conflict, it suffices to

consider all size five sets of taxa $\{a, b, c, d, e\} \subseteq S$. There are five quartets over this size five set of taxa, namely, $\{a, b, c, d\}$, $\{a, b, c, e\}$, $\{a, b, d, e\}$, $\{a, c, d, e\}$, and $\{b, c, d, e\}$. 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 size five set, we will, if Q_S is not tree-consistent, certainly obtain a size three subsets of Q_S which is not tree-consistent. Moreover, from Lemma 2 we know that we find *all* these local conflicts, in, namely, time $O(n^5)$.

We can improve this time bound for the preprocessing stage of the algorithm to be described in Section 5 with the following observation by Bandelt and Dress [3]. They show that, for our purpose, it is sufficient to restrict our attention to the size five sets containing some arbitrarily fixed taxon f.

Proposition 2. *(Proposition 6 in [3]) Given a set of taxa S and a 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.*

This statement may not seem intuitive. It tells us that, in the search for possible conflicts between three topologies, we do not need to look at all size five sets of taxa, but only on those involving the arbitrarily chosen f. This is caused by the denseness in a complete set of topologies. Therefore, a local conflict between three arbitrary topologies reflects in the topologies involving the chosen f. If there is a local conflict in our set of topologies, then we also will necessarily find one by looking only at this restricted set of topologies.

Following Proposition 2, we can select some arbitrary $f \in S$ and examine only the size five sets involving f. Similar to our proceeding described above, we consider every such size five set containing f separately. Among the topologies over this size five set, we search the size three sets which are not tree-consistent. In case the set of quartet topologies Q_S is not tree-consistent, we will find a size three set of quartet topologies which is not tree-consistent. It will be sufficient to "repair" these conflicts when trying to make the set of quartet topologies tree-consistent. Therefore, finding the local conflicts involving f can be done 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 useful combinatorial characterization of local conflicts. To this end, we introduce the notion of a *signature* of a set of topologies, which is easy to compute and which will help us to recognize a local conflict. Note that in the following definition we distinguish two possible

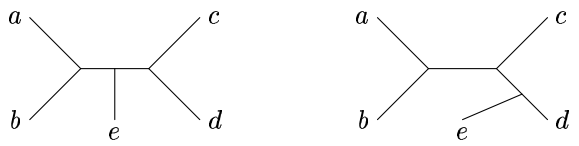


Figure 2: Trees inducing non-conflicting topologies in the proof of Theorem 2.

orientations of a quartet topology $[ab|cd]$, namely $[ab|cd]$, with a, b on its left hand side and c, d on its right hand side, and $[cd|ab]$, with the sides interchanged.

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.*

Using signatures, we now show a way how to characterize three topologies which are not tree-consistent.

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 $(4, 4)$. From Lemma 2 we know that three topologies are not tree-consistent iff they contradict the substitution property. To recall, 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 $t_3 = [ad|ce]$).

($\Leftarrow$) We are given three topologies t_1, t_2, and t_3 involving five taxa and having signature $(3, 4)$ or $(4, 4)$. Assume that they are tree-consistent. Showing that this implies signature $(2, 3)$ or $(3, 3)$, we prove that the assumption is wrong. For tree-consistent t_1, t_2, and t_3, we can find a tree inducing them. With, w.l.o.g., taxa $\{a, b, c, d, e\}$ and $t_1 = [ab|cd]$, we mainly 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 sets of quartet topologies induced by these trees, we find in

each case that the set has signature $(3,3)$. 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 $[ab|de]$). □

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 show that MQI is fixed parameter tractable by giving a simple version of a recursive procedure `resolve()` (to be improved later in this section) in Figure 3. Inputs are a complete set of quartet topologies Q and a positive integer k. Before calling the procedure for the first time, one has to build the list C of local conflicts. In fact, we only need to build a list of local conflicts containing some designated taxon, which can be chosen arbitrarily. The reason for this and the resulting preparation of this *conflict list* is explained in Section 3. Having done that, we call `resolve(`Q`, `k`, `C`)`. The subroutine called by `update(`C`, `t`)`, with a conflict list C and a topology t as arguments, searches, after t has been changed, the "neighborhood" of t, and updates the conflict list: It (1) removes the three-sets of quartets in the list whose topologies are now tree-consistent, and (2) adds the size three sets of quartets not in the list whose topologies now form a local conflict. The procedure `resolve()` will, if possible, output a complete set of quartet topologies that is tree-like and that can be obtained by altering at most k topologies in T. From this tree-like set of quartet topologies it is, then, possible to derive the evolutionary tree in time $O(n^4)$ [5].

For the biological application it is desirable to know not only one solution, but all evolutionary trees that we can obtain by altering at most k topologies in T. They can be taken as candidate solutions to be further evaluated, e.g., by human experts. To obtain all solutions, we can modify procedure `resolve()` by omitting the `stop` command in instruction (A0). Thus, the algorithm will browse the entire search tree and output all solutions requiring at most k alternations of quartet topologies.

Correctness. To obtain a non-conflicting set of quartet topologies, we have, following Theorem 1, to 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 tries every possibility to alter one of the three topologies. If there is a solution, we will, thereby, find it. If for none of the three topologies we can find a solution while altering the topology, the conflict cannot be removed, and there is no solution at all.

Running time. For each of the three quartet topologies there are two alternative topologies. Therefore, in the worst case, we branch into at most six subcases. In every subcase, we

```
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 ∈ C, with c = {t1, t2, t3}:
          for every alternative topology t1' of t1 do
                                C_new = Update(C, t1);
                                resolve(Q-t1+t1', k - 1, C_new);
          for every alternative topology t2' of t2 do
                                C_new = Update(C, t2);
                                resolve(Q-t2+t2', k - 1, C_new);
          for every alternative topology t3' of t3 do
                                C_new = Update(C, t3);
                                resolve(Q-t3+t3', k - 1, C_new);
          return; /* no success in current branch
                      -> step one level up in recursion */

}
```

Figure 3: Simple version in pseudocode of a recursive procedure for eliminating conflicts by changing at most k quartet topologies (if possible). Explanations in Section 5.

decrease the parameter k by one. This yields a search tree of height at most k in which each inner node has at most six children, meaning an upper bound of 6^k on the size of the search tree.

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 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.[5]

Initially building the conflict list, however, takes time $O(n^4)$, as explained in Section 3. These considerations are summarized in the following proposition.

Proposition 3. *With the simple version of procedure* `resolve()`*, we can solve the* MQI *problem in time* $O(6^k \cdot n + n^4)$.

Note that this running time is not only true for the algorithm reporting one solution, but also for its modified version reporting all binary trees satisfying the requirement. Our algorithm has only $O(kn^4)$ memory requirement, where the input size is already $O(n^4)$.

Decreasing the search tree size. We can lower the upper bound on the exponential growth of the search tree. Assume a triple of quartets whose topologies t_1, t_2, t_3 are conflicting. Then, an improvement can be achieved by not altering t_1, t_2, t_3 separately, but branching into every possibility to alter exactly one, exactly two, and exactly three topologies from t_1, t_2, t_3. Every quartet having three possible topologies and not considering the current topologies, we would have to branch into $3 \cdot 3 \cdot 3 - 1 = 26$ subcases. This, however, is not necessary, since not every of the proposed changes will eliminate the current conflict. Changing topologies t_1, t_2, t_3 into t_1', t_2', t_3', respectively, we can test whether t_1', t_2', t_3' are conflicting. Only if they are not conflicting, we *fix* topologies t_1', t_2', t_3' and branch into this subcase. By fixing the topologies we mean that we mark the three topologies, altered or not, to avoid that the topologies are unnecessarily altered in some following level of the recursion.

For the new analysis, we can take into account that parameter k will not always be decreased by only one. Changing exactly two topologies, we decrease k by two, changing exactly three topologies, we decrease k by three. For instance, Table 2 displays all possible

[5] 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 $O(1)$.

ways to alter topologies of the conflicting three topologies $[ab|cd]$, $[ac|be]$, and $[ac|de]$ (the last column in Table 2 will be considered later). We see that

- we have three ways to resolve the conflict by changing exactly one topology,
- we have five ways to resolve the conflict by changing exactly two topologies, and
- we have five ways to resolve the conflict by changing exactly three topologies.

This holds for all three topologies which are not tree-consistent. We can find this by checking all possible combinations of three topologies which are not tree-consistent.

Therefore, an bound on the search tree size is given by the recurrence

$$S_k \leq 1 + 3 \cdot S_{k-1} + 5 \cdot S_{k-2} + 5 \cdot S_{k-3}$$

. Clearly, $S_0, S_1, S_2 = O(1)$. The analysis of this recursion yields a bound of 4.397^k on the search tree size.

Proposition 4. *The* MQI *problem can be solved in time* $O(4.397^k \cdot n + n^4)$.

We can, however, improve the upper bound on the search tree size even further, and, thereby, arrive at an algorithm having an upper bound of 4^k on the search tree size. By a better selection of subcases to branch into we can find a way to make at most four recursive calls. In the following, we show that such a good branching can be obtained for every three topologies which are not tree-consistent. Let t_1, t_2, and t_3 be three topologies which are not tree-consistent, w.l.o.g., $t_1 = [ab|cd]$. Following Lemma 1, the topologies involve one additional taxon, say e. Following Lemma 2, t_1, t_2, t_3 contradict the substitution property. Given $t_1 = [ab|cd]$, the substitution property requires topology $[ab|ce]$ or topology $[ae|cd]$. Therefore, we can, w.l.o.g., assume the following setting for 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 to 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 $[ad|bc]$. Otherwise, we can assume that t_1 is not changed. Then, 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.

To clarify the improved branching, we revisit the example displayed in Table 2. We are given topologies $[ab|cd]$, $[ac|be]$, and $[ac|de]$, which are not tree-consistent and contradict the substitution property. Table 2 gives all possible ways to alter one, two, or three of the given topologies. In this example, the four subcases listed above are (1) Topology 1 set to $[ac|bd]$, (2) Topology 1 set to $[ad|bc]$, (3) Topology 1 fixed and Topology 2 set to $[ab|ce]$, and

Topology 1	Topology 2	Topology 3	Conflict?	Subcases
$[ab\|cd]$	$[ac\|be]$	$[ac\|de]$	conflicting	
$\star$	$\star$	$[ad\|ce]$	conflicting	
$\star$	$\star$	$[ae\|cd]$	ok	(4)
$\star$	$[ab\|ce]$	$\star$	ok	(3)
$\star$	$[ab\|ce]$	$[ad\|ce]$	ok	(3)
$\star$	$[ab\|ce]$	$[ae\|cd]$	ok	(3), (4)
$\star$	$[ae\|cb]$	$\star$	conflicting	
$\star$	$[ae\|cb]$	$[ad\|ce]$	conflicting	
$\star$	$[ae\|cb]$	$[ae\|cd]$	ok	(4)
$[ac\|bd]$	$\star$	$\star$	ok	(1)
$[ac\|bd]$	$\star$	$[ad\|ce]$	conflicting	(1)
$[ac\|bd]$	$\star$	$[ae\|cd]$	conflicting	(1)
$[ac\|bd]$	$[ab\|ce]$	$\star$	conflicting	(1)
$[ac\|bd]$	$[ab\|ce]$	$[ad\|ce]$	ok	(1)
$[ac\|bd]$	$[ab\|ce]$	$[ae\|cd]$	conflicting	(1)
$[ac\|bd]$	$[ae\|cb]$	$\star$	conflicting	(1)
$[ac\|bd]$	$[ae\|cb]$	$[ad\|ce]$	conflicting	(1)
$[ac\|bd]$	$[ae\|cb]$	$[ae\|cd]$	ok	(1)
$[ad\|bc]$	$\star$	$\star$	conflicting	(2)
$[ad\|bc]$	$\star$	$[ad\|ce]$	ok	(2)
$[ad\|bc]$	$\star$	$[ae\|cd]$	conflicting	(2)
$[ad\|bc]$	$[ab\|ce]$	$\star$	conflicting	(2)
$[ad\|bc]$	$[ab\|ce]$	$[ad\|ce]$	ok	(2)
$[ad\|bc]$	$[ab\|ce]$	$[ae\|cd]$	conflicting	(2)
$[ad\|bc]$	$[ae\|cb]$	$\star$	ok	(2)
$[ad\|bc]$	$[ae\|cb]$	$[ad\|ce]$	ok	(2)
$[ad\|bc]$	$[ae\|cb]$	$[ae\|cd]$	ok	(2)

Table 2: The set of three quartet topologies $[ab|cd]$, $[ac|be]$, and $[ac|de]$ is conflicting and has signature $(3, 4)$. Above, we list all possible ways to change one, two, or three of the topologies. We show which changes resolve the conflict between these three topologies. The first three columns give the new topologies, $\star$ denoting no change compared to the original topology given in the first line. The fourth column indicates whether the combination of topologies is conflicting or not ("ok"). The fifth column indicates which subcases of the last algorithm in this section (having a size 4^k search tree) cover the combination.

(4) Topology 1 fixed, Topology 3 set to $[ae|cd]$. In the last column in Table 2, we can see by which of these subcases the respective combination is covered. It is necessary to cover all prospective combinations which resolve the current conflict, the combinations not able to

resolve the conflict can be neglected. As we can read from the last column in Table 2, we cover all ways to resolve the conflict by branching into these four subcases. Considering the number of topologies altered in these subcases, we find that all subcases alter only one topology each, and decrease parameter k by one. The preceding considerations justify an upper bound of 4^k on the exponential growth and prove the following theorem, which summarizes our findings.

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

The reason that we also introduced the worse time bound given in Proposition 4 is the following. Given a local conflict, the underlying algorithm branches only in those subcases that resolve this conflict and only into those. In contrary to this, the algorithm underlying Proposition 3, however, contains some redundancy. In the above considerations, we resolve the conflict by altering t_2 to $[ab|ce]$ or by altering t_3 to $[ae|cd]$. We can, however, also resolve the conflict by altering both t_2 and t_3. We will encounter this case in both subcases (3) and (4).

6 Improving the running time in practice

Besides improving the worst case bounds on the algorithm's running time, we can also extend the algorithm in order to improve the running time in practice without affecting the upper bounds. In this section, we collect some ideas for such heuristic improvements.

Fixing topologies. It does not make sense to change a topology that, at some previous level of recursion, has been altered or for that we explicitly decided not to alter it. If we decide not to alter a topology in a later stage of recursion, 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, we show in this paragraph that 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 [20]. The observations described in this paragraph, however, will not yield a reduction to problem kernel (opposite to the case in [20]) 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, both of which we discuss in the succeeding paragraphs.

Lemma 3. *Consider an instance of the* MQI *problem 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, as was stated in Lemma 1. For five taxa, there are five quartets consisting of these taxa, e.g., for taxa $\{a, b, c, d, e\}$ the quartets are $\{a, b, c, d\}$, $\{a, b, c, e\}$, $\{a, b, d, e\}$, $\{a, c, d, e\}$, and $\{b, c, d, e\}$. Therefore, when we are given two quartet topologies t_1 and t_2, we make the following observations. 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. From the preceding discussion, we know that for any t', there are at most three topologies such that t and t' can form a conflict with it. Consequently, there must be more than k distinct topologies t' that occur in a local conflict with 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, 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. □

Lemma 3 can help us to identify topologies that have to be changed. We call these topologies "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. Thereby, we can "cut off," i.e., omit, complete subtrees of the search tree.

We discussed fixing topologies which are supposed not to be changed in the following levels of recursion. Having a local conflict consisting only of fixed topologies, we obviously cannot resolve this conflict while not changing one of the fixed topologies.

In the preceding paragraph, we discussed how to recognize topologies which are forced to change. We know that for a solution, we have to change these topologies. If, after identifying these 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 4. *Let us have an instance of the* MQI *problem 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 5. *An instance of the* MQI *problem in which the number of local conflicts is greater than* $6(n-4)k$ *has no solution.*

Proof. As described in Section 3, we have shown that 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 that can form a local conflict. Therefore, by altering k topologies, we can resolve at most $6(n-4)k$ distinct local conflicts. □

Clever branching. Applying the rules described above will also significantly improve our situation when branching. Having to select a conflict to branch on, we can take advantage from topologies which are forced to change and from topologies which are fixed. For the general branching situation on a local conflict, we have shown in Section 5 that it is sufficient to branch into four subcases. Regarding topologies forced to change, we can, however, reduce the number of subcases. When we have identified a topology t which is forced to change, it is sufficient to branch into two subcases: one for each alternative topology of t. Regarding fixed topologies, we can take advantage of local conflicts which contain fixed topologies. Having a local conflict with one or two fixed topologies, we omit the subcases which change a fixed topology. This will reduce the number of subcases to three, two, or even to one subcase. For instance, suppose topologies $t_1 = [ab|cd]$, $t_2 = [ac|be]$, and $t_3 = [ac|de]$ which are not tree-consistent. The general branching would branch into four subcases (1) $[ac|bd]$, (2) $[ad|bc]$, (3) $[ab|ce]$, and (4) $[ae|cd]$. In case topologies t_1 and t_2 are fixed, it only remains to consider subcase (4).

Preprocessing by the Q*-method. The algorithmic improvements described above do not sacrifice the guarantee to find the optimal solutions. Using these improvements, we will find every solution that we would find without them. This is not true for the following idea. We propose to use the Q*-method described by Berry and Gascuel [5] 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. Therefore, our algorithm will compute the minimum number of quartet topologies we have to change in order to obtain a tree-like set of topologies that contains the topologies fixed by the Q*-method. The tree we obtain will be a refinement of the tree reported by the Q*-method, which may contain unresolved branches. Thus, we cannot guarantee that the reported tree is the optimal solution for the MQI problem. On real data, however, it is the optimal tree with high certainty: Suppose it is not. Then there are three taxa a, b, c that are arranged in another way by the Q*-method as they would be arranged in the optimal solution for the MQI problem. As we are working on a complete set of topologies, this would imply that there are $n-3$ quartets, namely $\{a, b, c, d\}$ for all $d \in S - \{a, b, c\}$, that would make the same wrong prediction for the arrangement of a, b, c. On real data, this is very unlikely. Our experiments described in Section 8 support the 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 Related problems

We now come to some variants and generalizations of the basic MQI problem and their fixed parameter tractability. These variations arise in practice due to the fact that often quartet inference methods cannot non-ambiguously and with high certainty predict one topology for every quartet. Note, however, that the fixed parameter tractability of the problems heavily depends on the choice of parameters. Perhaps the most natural generalization of MQI is to consider weighted quartet topologies.

Weighted MQI**.** Weights arise since a quartet inference method can predict the topology for a quartet with more or less certainty. Therefore, we can assign weights to the quartet topologies reflecting the certainty they are predicted with. Given a complete set of weighted topologies Q_S and a positive integer k, we distinguish two different questions.

1. Assume that we are given a complete set of weighted topologies Q_S, with positive real weights, and a positive integer k. A binary tree is a candidate for a solution if the set of quartet topologies induced by this tree differs from Q_S in the topologies for at most k quartets. Can we, among all candidate trees satisfying this property, find the one such that the topologies in Q_S which are not induced by the tree have minimum total weight?

   The algorithm presented in Section 5 is capable to compute *all* solution trees. Therefore, we can, without sacrificing the given time bounds, find this tree among the solution

trees for which the "wrong" quartet topologies have minimal total weight.

2. Assume that we are given a complete set of weighted topologies Q_S, each topology having a real weight ≥ 1, and a positive real K. Is there a binary tree such that the quartet topologies induced by the tree differ from the given topologies only for topologies having total weight less than K?

   Again, we can use the algorithm presented in Section 5. When branching into different subcases, the time analysis of the algorithm relied on the fact that in each subcase at least one quartet topology is changed, i.e., added to the "wrong" topologies. In the current situation of weighted topologies with weights ≥ 1, each subcase changes quartet topologies having total weight at least 1. The time analysis of our algorithm is, therefore, still valid and the time bounds remain the same.

When allowing in Question 2 arbitrarily small weights, then the problem cannot be fixed parameter tractable, unless P = NP. To see this, take an instance of unweighted MQI with parameter k. We can turn this instance into an instance of weighted MQI by assigning all topologies weight $1/k$ and setting the parameter to 1. A fixed parameter algorithm for the problem with arbitrary weights > 0 would thus give a polynomial time solution for MQI, which contradicts the NP-completeness of MQI unless P = NP. Having, however, weights $> \epsilon$ for some positive real ϵ, the problem is fixed parameter tractable as we described here for the special case that $\epsilon = 1$. These observations were analogously made for Weighted Vertex Cover [21].

Underspecified MQI. Due to lack of information or due to ambiguous results, a quartet inference method may not be able to compute a topology for every quartet, such that there can be quartets for which no topology is given. Let us assume that the number of quartets which do not have a topology is bounded: Given a set S of taxa, a set of topologies Q_S, and a positive integer k. The set Q_S contains quartet topologies for all quartets over S except for k' many for which no topology is given. Is there a binary tree such that the quartet topologies induced by the tree differ from the given topologies only for k topologies?

The set of topologies is "underspecified" by k' topologies. Having three possible topologies for each quartet, we can, for a quartet having no given topology, branch into three subcases, one for each of its three possible topologies. Having selected a topology for each such quartet, we run the algorithm from Section 5. The resulting algorithm has time complexity $O(3^{k'} \cdot 4^k \cdot n + n^5)$ and shows that the problem is fixed parameter tractable for parameters k and k'.

Note that for unbounded k' this problem is the Sparse MQI problem and, therefore, is not fixed parameter tractable, as mentioned in Section 2.

Overspecified MQI. This case arises when the quartet inference method cannot definitely resolve a quartet topology and proposes two different topologies which it considers to be

equally good. Note that we do not consider the case that we are given three different topologies for a quartet as this would be as much information as giving no topology at all. Hence, let us assume that the number of quartets for which we are given two topologies is bounded: Given a set S of taxa, a set of topologies Q_S, and a positive integer k. The set Q_S contains a quartet topology for every quartet over S and contains two different topologies for k'' many. Is there a binary tree such that the quartet topologies induced by the tree differ from the given topologies only for k topologies?

The set of topologies is "overspecified" by k'' topologies. If we are given two topologies for a quartet, this means that we can choose one topology from these. Having two topologies for a quartet, we can branch into two subcases, each choosing one of the two given topologies. When we have selected a topology for each such quartet, we run the algorithm from Section 5. The resulting algorithm has time complexity $O(2^{k''} \cdot 4^k \cdot n + n^5)$ and shows that the problem is fixed parameter tractable for parameters k and k''.

It is straightforward that we can combine the case of underspecified and overspecified MQI. This leads to a fixed parameter tractable solution for MQI with parameters k', k'' and k when k' is the number of underspecified and k'' is the number of overspecified quartets.

8 Experimental evaluation

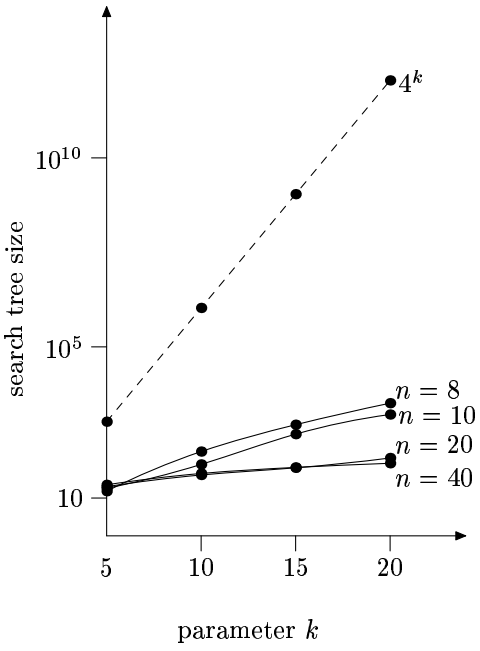
To investigate the usefulness and practical relevance of the algorithm, we performed experiments on artificial as well as on real data from fungi. The implementation of the algorithm was done using the programming language C. The algorithm contains the enhancements described in Section 6. The combined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

8.1 Artificial data

We performed experiments on artificially generated data in order to find out which kind of data sets our algorithm can be especially useful for. For a given number of taxa n and parameter k, we produce a data file as follows. We generate a 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. This results in a MQI instance that certainly can be solved with parameter k. For each pair of values for n and k, ten different data sets were created. The reported results are the average for test runs on ten

n	k	time in sec	search tree size
8	5	0.02	15
	10	0.05	167
	20	0.72	3216
10	5	0.03	16
	10	0.05	93
	20	0.61	1792
20	5	0.19	20
	10	0.20	41
	20	0.25	115
	40	16.83	20375
50	5	9.21	20
	10	9.01	46
	20	8.63	92
	40	23.25	178
	100	2409.07	653571

(a)



(b)

Figure 4: 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 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.

data sets.

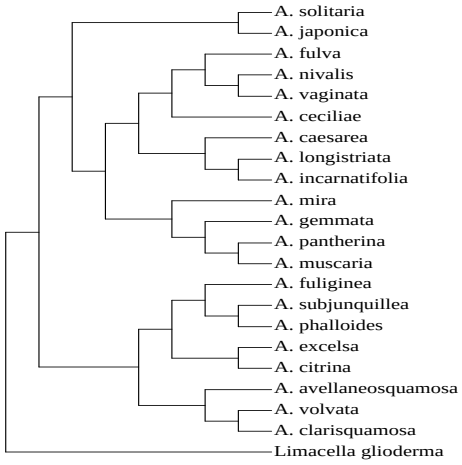
We experimented with different values of n and k. As a measure of performance, we use two values: We report the processing time and, since processing time is heavily influenced by system conditions, e.g., memory access time in case of cache faults, also the search tree size. The search tree size is the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

Figure 4(a) gives a table of results for different values of n and k. Regarding the processing time, we note, on the one hand, the increasing time for fixed a n and growing k. On the other hand, we observe that for moderate values of k, we can process large instances of the problem, e.g., $n = 50$ and $k = 100$ in 40 minutes. Ben-Dor *et al.* [4] do only report on processing up to 20 taxa. Regarding the search tree size, we compare in Figure 4(b), on a logarithmic scale, the theoretical upper bound of 4^k to the real size of the search tree. For each fixed number of taxa n, we give a graph displaying the growth of search tree size for increasing k. We note that the search trees are, by far, smaller than the 4^k bound. This is

n taxa	running time in sec. without 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 quartets topologies fixed by the Q*-method

(a)



(b)

Figure 5: (a) Speed-up when using the Q* method as preprocessing. (b) Optimal tree found for a set of 21 Amanita species and one outgroup taxon.

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 often results in a smaller search tree.

8.2 Real data

Using our algorithm, we analyzed the evolutionary relationship of the Amanita species. The underlying data are DNA sequences of length 576 from Amanita species and one outgroup taxon, as they are used by Weiß *et al.* [25, 26]. We inferred the quartet topologies by using `dnadist` taken from the Phylip package [15], and `distquart` taken from the Phyloquart package [5].

The analysis was done by a preprocessing of the data using the Q*-method, 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 5(a) shows this impact for small numbers of Amanita taxa. Note, however, that the speed-up heavily depends on the data. In Figure 5(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 $n = 12$.

To give an example for the algorithm's performance, we processed a set of $n = 22$ taxa in 35 minutes. The resulting tree was rooted using the outgroup taxon and is displayed in Figure 5(b). We found the best solution for $k = 979$ for the given 7315 quartet topologies. 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 shown in the results by

Weiß *et al.* [26], who used, e.g., heuristic maximum parsimony methods. The relations among the sections and within the sections differ in single cases from the results obtained by Weiß *et al.* Thus, we can suggest our tree as a new hypothesis on the evolutionary relationship. Another reason for these differences may, however, be that these relations cannot be resolved on the basis of the used data. For comparison of the performance of the algorithm, consider the results reported by Ben-Dor *et al.* [4], who solve a MQI instance giving a guaranteed optimal result. They list a running time of 128 hours for a set of 20 taxa (on a SUN Ultra-4 with 300 MHz).

Note that our experiments were performed on closely related taxa. We suppose that this is a reason for the high value of k in this case and that quartet inference can be performed with higher quality for more divergent taxa which have a less close relationship. One might also expect that quality of quartet inference techniques will improve in the future. This would lead to instances requiring a smaller value of k.

9 Conclusion

In this paper, we showed that the Minimum Quartet Inconsistency problem can be solved in worst case time $O(4^k n + n^4)$, meaning that the problem is fixed parameter tractable when parameter k is the number of faulty quartet topologies. Several ideas for tuning the algorithm show that the practical performance of the algorithm is much better that 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 8.

Concerning future work, we want to extend our experiments to weighted quartet topologies and to other (non-fungi) taxa, in particular more divergent data which might enable better solutions. Also, the fact that we can obtain all optimal and near-optimal solutions and the usefulness of this deserves further investigation. 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 [12] for details). In addition, the further reduction of the tree size concerning theoretical, as well as experimental bounds, is a worthwhile future challenge.

Acknowledgement. We thank Michael Weiß from the Biology Department (Special Botany and Mycology group) for providing us with the Amanita data and supporting us in the interpretation of our results.

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed parameter algorithms for Planar Dominating Set and related problems. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, number 1851 in Lecture Notes in Computer Science, pages 97–110, 2000. Springer-Verlag.

[2] J. Alber, J. Gramm, and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. To appear in *Discrete Mathematics*, 2001.

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

[4] 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.

[5] 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`.

[6] 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*, pages 287–296, 2000.

[7] 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*, number 1643 in Lecture Notes in Computer Science, pages 313–324, 1999. Springer-Verlag.

[8] D. Bryant and M. Steel. Fast algorithms for constructing optimal trees from quartets. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 147–155, 1999. SIAM Press.

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

[10] 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.

[11] R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In Clote and Remmel, editors, *Feasible Mathematics II*, pages 219–244, Birkhäuser, Boston, 1995.

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

[13] R. G. Downey and M. R. Fellows and U. Stege. Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability. In Graham *et al.* (eds.) *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.

[14] P. Erdős, M. Steel, L. A. Székely, and T. Warnow. Constructing big trees from short sequences. In *Proceedings of the 24th International Conference on Automata, Languages, and Programming*, number 1256 in Lecture Notes in Computer Science, pages 827–837, 1997. Springer-Verlag.

[15] 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`.

[16] 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*, pages 416–425, 1998.

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

[18] T. Jiang, P. Kearney, and M. Li. A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. To appear in *SIAM Journal on Computing*, 2001.

[19] P. Kearney. The ordinal quartet method. In *Proceedings of the 2nd ACM Annual International Conference on Computational Molecular Biology*, pages 125–134, 1998.

[20] R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-Hitting Set. Technical Report WSI-99-18, WSI für Informatik, Universität Tübingen, October 1999. Revised version to appear in *Journal of Discrete Algorithms*.

[21] R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for Weighted Vertex Cover. In *Proceedings of the 11th International Symposium on Algorithms and Computation*, number 1969 in Lecture Notes in Computer Science, pages 180–191, 2000. Springer-Verlag.

[22] K. St. John, T. Warnow, B.M.E. Moret, and L. Vawter. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, pages 196–205, 2001. SIAM Press.

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

[24] 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.

[25] M. Weiß. Molecular investigations on phylogeny in the genus Amanita. Dissertation, Universität Tübingen, 1999.

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