Algorithms for Phylogenetic Supertree Construction

Caleb Koch

December 2019

Abstract

The authors in [JMT19] introduce the q-MAXRTC problem for phylogenetic supertree construction and prove that it is NP-hard for every $q \geq 2$. We generalize their result to showing that q-MAXRTC is in fact fixed parameter tractable (**FPT**) and there is a fixed-parameter tractable reduction from MAX q-CUT to q-MAXRTC. This result shows in particular that any **FPT** lower bound for MAX q-CUT also translates into an **FPT** lower bound for q-MAXRTC. We then consider an approximation algorithm for q-MAXRTC which is discussed in [JMT19] but not fully presented. We give a formal treatment and presentation of the generalized q-MAXRTC algorithm from their paper. Additionally, we implement and run this approximation algorithm on various phylogenies of bacterial species and demonstrate that the algorithm is quite accurate in practice.

Contents

1 Preliminaries **2**
- 1.1 Purpose and Motivation 2
- 1.2 Biology 2
- 1.3 Parametrized Complexity 2

2 Complexity of q-MAXRTC **4**

3 Generalized algorithm **6**
- 3.1 Randomized q-MAXRTC 7
- 3.2 Deterministic approximation algorithms for q-MAXRTC 8

4 Experimentation and Results **9**

5 Discussion **11**

6 Future research directions **11**

7 References **12**

1 Preliminaries

1.1 Purpose and Motivation

We explore various algorithms for *phylogenetic supertree construction*. Our goal, roughly, is to understand this problem from a computational perspective so as to better understand the feasibility of performing supertree construction on a computer. In order to do so, we first need to formalize the problem in a way that is amenable to algorithmic analysis. Here, we primarily consider the q-MAXRTC formalization of the problem from the recent study [JMT19]. As is typical in bioinformatics, our algorithmic analysis proceeds in two steps. First we seek to understand the *computational intractability* of the problem by developing conditional lower bounds, which, generally, take form *if conjecture C holds then no efficient algorithm can be designed for the problem* where *efficient algorithm* has a precise technical meaning. Since proving lower bounds are notoriously difficult, conditional lower bounds are often the best one can hope for (e.g. an NP-hardness result is an example of a conditional lower bound and is often the best one can accomplish, and often one cannot do much better in terms of lower bounds). The second step of our algorithmic analysis involves developing *upper bounds* for the problem at hand. In our case, these upper bounds take the form of explicit algorithms solving the problem. Thus, by studying upper and lower bounds for the problem we can achieve a better understanding of the tractability of the problem at hand. Given that phylogenetic trees represent evolutionary relationships between species, and are thus in practice quite large, understanding the tractability of supertree construction is well-motivated and can help direct future research efforts by suggesting avenues which may lead to more efficient algorithms.

1.2 Biology

Often biological studies focus on a few individual taxa and so deducing evolutionary relationships between taxa which are not often studied together can be a difficult task [JNSS05]. Algorithms for computing supertrees are thus important insofar as they allow one to merge disparate phylogenetic trees together in a way that minimizes the loss of branching information. Hence, supertrees can be used to gain insight into the evolutionary relationships between species studied separately. In fact, the authors in [BE04a] remark that supertrees provide the "only phylogenetic method that can build complete phylogenies of very large clades." Hence, it should be no surprise that phylogenetic supertrees are studied quite a bit in the bioinformatics community, see e.g. [BE04b, CM04, BE05, War18].

1.3 Parametrized Complexity

Parametrized complexity appears in various places in the bioinformatics literature including in [BM11, HRB$^+$09, BDF$^+$95, MZ08]. Nonetheless, we present briefly the preliminary notions from parametrized complexity we use in understanding the computational intractability of supertree construction. We follow primarily the presentation in [CFK$^+$15].

Definition 1.1. A parametrized problem is a set $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ is a finite alphabet, Σ^* is the set of finite length strings using symbols in Σ and $\mathbb{N} = \{0, 1, ...\}$ is the set of natural numbers. For $(x, k) \in L$, we say that k is the parameter.

Often problems are already parametrized. For example, the input to k-CLIQUE, the problem of deciding whether a graph has a k-clique or not, is (G, k) where G is a graph and k

is a natural number. Thus the language $\{(G,k) \in \{0,1\}^* \times \mathbb{N} : G$ encodes a graph with a k-clique$\}$ is a parametrized problem. Likewise the input to the *motif finding problem* is of the form $((x_1, \ldots, x_t), k) \in (\{A, C, T, G\}^n)^t \times \mathbb{N}$ and hence can be viewed as a parametrized problem where the goal is to find a motif of length k in the t sequences $x_1, \ldots, x_t$, each of length n. There is a standard notion for an *efficient* algorithm for a parametrized problem which is called a polynomial-time *fixed-parameter* algorithm.

Definition 1.2. A parametrized problem is *fixed-parameter tractable* (FPT) if there is an algorithm A and an arbitrary computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial p such that A decides if $(x,k) \in L$ in time at most $f(k) \cdot p(|(x,k)|)$. We write **FPT** to denote the class of problems L which are FPT.

Continuing with the motif finding problem, the brute force algorithm takes time $O(4^k(n-k)tk)$ on input $((x_1, \ldots, x_t), k) \in (\{A, C, T, G\}^n)^t \times \mathbb{N}$. Hence setting $f(k) = 4^k$ and $p(x) = x$ we see that the brute force algorithm runs in time $O(f(k)p(|((x_1, \ldots, x_t), k)|))$ and therefore the motif finding problem is fixed-parameter tractable. Note that any parametrized problem can be viewed as a standard decision problem where the task is: given some input (x,k), decide whether $(x,k) \in L$. In this case, the size of the input is $|(x,k)| = |x| + |k|$ where $|x|, |k|$ are the number of bits required to encode x, k respectively. Thus, although a problem may be FPT, the general decision version of the problem may have no efficient (polynomial-time) algorithm.

We can relate the difficulty of various parametrized problems via *parametrized reductions*.

Definition 1.3 (Parametrized reduction)**.** Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two parametrized problems. We say that A reduces to B, written $L \leq L'$, if there is a parametrized reduction from A to B, which is a map $F : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that $(x,k) \in L \Leftrightarrow F(x,k) \in L'$ and if $(x', k') = F(x,k)$ then $k' \leq g(k)$ for some computable g and F is computable in time $f(k)p(|x|)$ for an arbitrary computable f and polynomial p.

Essentially, $L \leq L'$ if there is an efficient algorithm for translating inputs in L into inputs for L'. Equivalently, if $L \leq L'$ and if one were to devise an FPT algorithm for L' then this would immediately yield an FPT algorithm for L: to decide if $(x,k) \in L$ we apply our reduction F to the input and then use our algorithm for L' to decide membership. The analogous problem in the standard decision problem world is that of *polynomial-time reductions*.

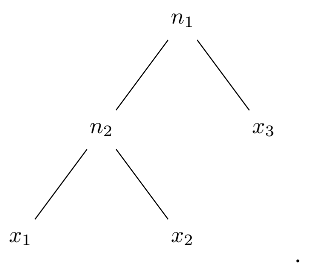
Reductions allow us to define classes of problems which are roughly the same difficulty as each other. In particular, the class $W[i]$ consists of the parametrized problems L such that every input (x,k) can be represented as a Boolean circuit with *weft* at most i such that $(x,k) \in L$ if and only if the circuit has a satisfying assignment setting at most k of the input bits to 1 (the rest must be 0). The precise details are unimportant for our purposes, but suffice it to say that a problem B is $W[i]$-hard if for every other L there is an FPT reduction from L to B: $L \leq B$. Likewise, a problem is $W[i]$-complete if it is in $W[i]$ and is $W[i]$-hard.

The class $W[0]$ is precisely **FPT**. Moreover, k-CLIQUE and k-INDSET[1] are both canonical W[1]-complete problems. Detailed proofs of these facts can be found in [CFK+15].

[1]We write k-INDSET to denote the problem of determining whether a graph G has an independent set of size k. An independent set is set of vertices no two of which share an edge.

2 Complexity of q-MAXRTC

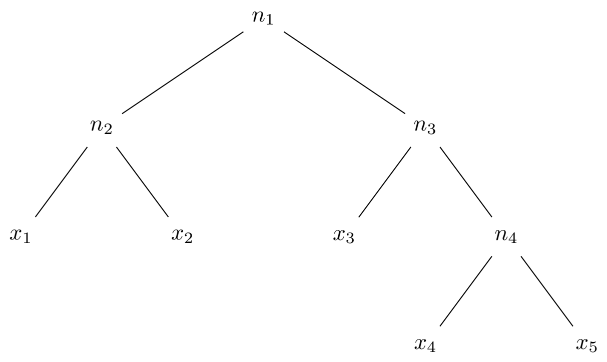
We introduce the q-MAXRTC problem and show it is in **FPT**. A *resolved triplet* is a binary tree with three labeled leaves:



We write such a tree as $x_1x_2|x_3$ where x_3 indicates the leaf closer to the root. The phylogenetic trees we consider are rooted and have leaves labeled with elements fomr a common leaf set L. Given such a tree T, there is an associated set of resolved triplets. In particular, the triplet $xy|z$ is *induced* by T if

$$\text{lca}(x,z) = \text{lca}(y,z) \qquad \text{lca}(x,y) \neq \text{lca}(x,z)$$

where lca denotes the *least common ancestor*. For example in the tree



the triplet $x_4x_5|x_1$ is induced by the tree because $\text{lca}(x_4,x_1) = n_1 = \text{lca}(x_5,x_1)$ however $\text{lca}(x_4,x_5) = n_4 \neq n_1 = \text{lca}(x_4,x_1)$. In contrast, $x_4x_3|x_5$ is *not* induced because $\text{lca}(x_4,x_5) = n_4 \neq n_3 = \text{lca}(x_3,x_5)$. Given a tree T we write $\text{rt}(T)$ to denote the set of resolved triplets of T. We write $|T|$ to denote the number of *internal* nodes of T. In the example above, the internal nodes are n_1, n_2, n_3 and so $|T| = 3$. We are now ready to formally define q-MAXRTC.

q-MAXRTC

Input: (L,$\mathcal{R}$,q), a leaf set L and a set of resolved triplets $\mathcal{R}$ and parameter q
Output: a tree T with leafs labeled from L such that $|T| = q$ and $|\mathcal{R} \cap \text{rt}(T)|$ is maximized

As a technical point, the problem presented is not a decision problem, nor is it a parametric problem. This issue can be easily fixed by introducing a parameter k and modifying the requirement that $|\mathcal{R} \cap \text{rt}(T)|$ is maximized to being that $|\mathcal{R} \cap \text{rt}(T)| \geq k$. The input will then simply be $(\mathcal{R}, (q, k)) \in \{0,1\}^* \times \mathbb{N}^2$ (we can infer L from $\mathcal{R}$ and hence it need not be explicitly part of the input). We can without loss of generality view this as a parametrized problem since there is a bijection $\lambda : \mathbb{N}^2 \to \mathbb{N}$ such that $\lambda(x, y) \leq p(x, y)$ where p is a quadratic polynomial in x, y (hence the transformation doesn't lead to a large blow up in the size of the parameter).[2]

Relation of q-MAXRTC to phylogenetic supertree construction. *A priori* it may not be obvious the connection of the q-MAXRTC to phylogenetic supertree construction. The main idea is that if we are given a set $T_1, \ldots, T_m$ of phylogenetic trees consisting of species S (i.e. S is the leaf label set) and we want to build a phylogenetic supertree from this set, we form the set

$$\mathcal{R} = \bigcup_{i=1}^{m} \text{rt}(T_i)$$

of induced triplets and use this along with some parameter q as input into q-MAXRTC. Intuitively the induced set of triplets for a phylogenetic tree T_i synthesizes the important branching information of T_i into a smaller, more manageable unit of information. The importance of being able to fine-tune our parameter q is that we can adjust the tradeoff between the simplicity of the tree T returned by q-MAXRTC and its accuracy. The larger q is the more likely we will be able to recover all of the induced triplets; however we may also accidentally introduce *spurious novel clades* which are false groupings of species [BE04a]. On the other hand, a smaller q enforces less extrapolations to be made about the interrelationships of the phylogenetic trees and thus yields a simpler model that may be inaccurate. Thus, having q as a separate parameter allows biologists to fine-tune the algorithm to their needs.

In [JMT19] the authors prove that computing q-MAXRTC is NP-hard for $q \geq 2$. We strengthen this result to showing that q-MAXRTC is fixed parameter tractable when viewed as a parametrized problem.

We first remark on W[1]-completeness.

Theorem 2.1. *The problem k-INDSET is W[1]-complete.*

The proof of Theorem 2.1 is well known and can be found in standard references on parametrized complexity, e.g. [CFK$^+$15, Theorem 13.18]. Thus we omit the proof here.

The problem MAX-CUT is the following: given a graph $G = (V, E)$ and a parameter k, determine if there is a partition of V into sets A and B such that the number of edges connecting a vertex in A with a vertex in B is k. The problem MAX q-CUT is the analogous problem of finding a partition of the vertices into sets $(A_1, \ldots, A_q)$ such that the size of the cut across all $(A_1, \ldots, A_q)$ is k for some given k (note that an edge (v, v') is in this cut if there is some $i \neq j$ with $v \in A_i$ and $v' \in A_j$). We show that there is a parametric reduction from MAX q-CUT to q-MAXRTC. Note that this result is essentially implicit in [JMT19, Theorem 1].

Proposition 2.2. *The problem MAX q-CUT satisfies*

$$\textit{MAX } q\textit{-CUT} \leq q\textit{-MAXRTC}.$$

[2]The polynomial that achieves this result is called the Cantor pairing function and is given by $\lambda(x, y) = 1/2((x + y)^2 + 3x + y)$ [Lis07].

Proof. Let (G, q, k) for $G = (V, E)$ be an instance of MAX q-CUT. We form an instance of q-MAXRTC in the following way. Let z be a new variable not in V and define a leaf set $L = V \cup \{z\}$ and triplet set $\mathcal{R} = \{xz|y, yz|x : \{x, y\} \in E\}$. Then G has a cut $(A_1, \ldots, A_q)$ of size k if and only if $(L, \mathcal{R}, q, k)$ is a valid instance of q-MAXRTC. The proof of this fact is contained in [JMT19, Theorem 1] so we need only verify now that this reduction is in FPT.

Indeed the sets L and $\mathcal{R}$ can be constructed in time polynomial in $|G|$ since we need only inspect every edge of G. Moreover, the new parameter for q-MAXRTC is the same as (and hence upper bounded by) the parameter for MAX q-CUT. Therefore, this reduction is a valid parametrized reduction. □

A corollary of Proposition 2.2 is that if MAX q-CUT is W[1] hard then so is q-MAXRTC. It may be tempting to show then that MAX q-CUT is W[1]-hard via another reduction from, e.g. k-INDSET since by Theorem 2.1 we know that k-INDSET is W[1]-hard. However, it is unlikely that q-MAXRTC is W[1]-hard as the following proposition reveals.

Proposition 2.3. *The problem q-MAXRTC is in* ***FPT****.*

Proof. Let $(L, \mathcal{R}, q, k)$ be an instance of q-MAXRTC. We can iterate through every binary tree of size q. In particular there are $f(q) = \binom{2q}{q}$ such trees (the qth Catalan number) and so we can simply check every single one. For a given tree T and triplet $xy|z \in \mathcal{R}$, we need to determine if there is a way to assign x, y, z to leaves such that $xy|z$ is induced by T. We can do this by trying out every possible assignment of x, y, z to leaves in T. There are at most $O(q^3)$ such possible assignments. We do this for at most $|L|^3$ triplets. If at any point we find a tree that induces the desired number of triplets, we halt and output the tree. Otherwise there isn't one. Overall we take time

$$O(g(q)|L|^3)$$

where $g(q) = \binom{2q}{q}q^3$. Thus since q is a parameter this problem is fixed-parameter tractable. □

Note that as a corollary of Proposition 2.3, we get that restriction of q-MAXRTC when $k = |\mathcal{R}|$ is also in **FPT**. This result thus, partially gets at an open question posed by [JMT19, Section 6].

3 Generalized algorithm

The authors in [JMT19] give an explicit characterization of the approximation algorithm for q-MAXRTC when $q = 2$. We generalize their algorithm to work for any q. To our knowledge, there is no known published paper or book that contains a complete description of this fully generalized version. It is mentioned briefly in [JMT19, Section 3]; however, not much more detail is given beyond stating that it generalizes the version where $q = 2$ in a natural way.

Before jumping into the fully generalized algorithm, we present a simple randomized algorithm for q-MAXRTC when $q = 2$. This algorithm is described by Algorithm 1.

Algorithm 1 Randomized 2-MAXRTC procedure

```
Require: $L$ the leaf set.
  Construct the tree $T = a - b$, the tree consisting of two nodes $a$ and $b$ with $a$ the root and
  $b$ the child.
  for all $\ell \in L$ do
    With probability $2/3$ add $\ell$ to $T$ as a child of $b$ and with probability $1/3$ add $\ell$ to $T$ a
    child of $a$.
  end for
  return $T$
```

3.1 Randomized q-MAXRTC

Algorithm 1 yields a $4/27$ approximation of q-MAXRTC. In particular, we can view T and hence $\mathrm{rt}(T)$ as a random variable for a fixed input $\mathcal{R}$ and can verify that

$$\mathbb{E}[|\mathcal{R} \cap \mathrm{rt}(T)|] = \frac{4}{27}|\mathcal{R}|.$$

This expectation follows from the observation that given a fixed $xy|z \in \mathcal{R}$, the probability that $xy|z$ is also in $\mathrm{rt}(T)$ is $1/3 \cdot 2/3 \cdot 2/3 = 4/27$ since this event only occurs if z is assigned to a and both x and y are assigned to b. There is an analogous randomized algorithm for q-MAXRTC where q is any value.

Algorithm 2 Randomized q-MAXRTC procedure

```
Require: $L$ the leaf set. Assume $q = 2k + 1$ is odd.
  Construct a minimal depth binary tree $T$ consisting of $q$ nodes.
  Let $N$ denote the number of leaves of $T$. Since $q = 2k + 1$ we have $N = k + 1$.
  $T' \leftarrow \mathrm{copy}(T)$
  for all $\ell \in L$ do
    Assign $\ell$ to leaf $\ell'$ in $T'$ with probability $1/N$ if $\ell'$ corresponds to a leaf in $T$ and with
    probability $0$ otherwise.
  end for
  return $T'$
```

In the above algorithm we require that q is odd so that our binary tree can be rooted. If q is not odd we can just run the algorithm on $q - 1$ and add an extra internal node to the resulting tree T'. With regard the above algorithm we get the following result which is analogous to our analysis of Algorithm 1.

Theorem 3.1. *Let $\mathcal{R}$ be an input into Algorithm 2 and let $rt(T')$ be the random variable which is the set of induced triplets of T' returned by the algorithm. Then*

$$\mathbb{E}[|\mathcal{R} \cap rt(T)|] = \left(\frac{1}{3} - \frac{4}{3(q+1)^2}\right)|\mathcal{R}|.$$

Note that this statement is precisely that of [JMT19, Theorem 9].

3.2 Deterministic approximation algorithms for q-MAXRTC

To achieve a deterministic approximation algorithm for q-MAXRTC, we derandomize Algorithm 2 using the *method of conditional expectations*. This algorithm is given as Algorithm 3.

Algorithm 3 Derandomized q-MAXRTC procedure

```
Require: $L$ the leaf set. Assume $q = 2k + 1$ is odd.
  Construct a minimal depth binary tree $T$ consisting of $q$ nodes.
  Let $\mathcal{L}$ denote the set of leaves of $T$.
  Initialize $\mathcal{A} : L \to \mathcal{L}$, an assignment of input leaves to leaves of $T$. Initially $\mathcal{A}[\ell] =$ null
  for all $\ell \in L$.
  $p \leftarrow (1/3 - 4/3(q + 1)^{-2}$
  for all $\ell \in L$ do
    Initialize $V[l] = p$ for all $l \in \mathcal{L}$.
    for all $xy|z \in \mathcal{R}$ such that $\ell \in \{x, y, z\}$ do
      Let $\text{PR}(xy|z, \mathcal{A})$ denote $\mathbb{P}[xy|z \in \text{rt}(T) \mid \mathcal{A}]$
      $\mathcal{A}[\ell] \leftarrow$ null
      $V[l] \leftarrow V[l] - \text{PR}(xy|z, \mathcal{A})$ for all $l \in \mathcal{L}$
      for all $l \in \mathcal{L}$ do
        $\mathcal{A}[\ell] \leftarrow l$
        $V[l] \leftarrow V[l] + \text{PR}(xy|z, \mathcal{A})$
      end for
    end for
    $\mathcal{A}[\ell] \leftarrow \text{argmax}_{l \in \mathcal{L}} V[l]$
    $p \leftarrow \max_{l \in \mathcal{L}} V[l]$
  end for
  For every $\ell \in L$, make $\ell$ a child of $\mathcal{A}[\ell]$ in $T$
  return $T$
```

The main idea of Algorithm 3 is the following. Let W be the random variable $|\mathcal{R} \cap \text{rt}(T)|$. For every leaf $\ell \in L$, we want to assign ℓ to a leaf in our full binary tree T in a way that maximizes $\mathbb{E}[W]$. To do this, at each step we maximize $\mathbb{E}[W \mid \mathcal{A}, \ell \leftarrow l]$ over all $l \in \mathcal{L}$ where $\mathcal{A}$ is all of the assignments of leaves thus far, $\mathcal{L}$ is the set of leaves in our original tree T (which serve as potential parents for leaves in the input set L) and $\ell \leftarrow l$ denotes an assignment of ℓ as a child of l. We achieve this computation by using the fact that for any assignment $\mathcal{A}$ we have

$$\mathbb{E}[W \mid \mathcal{A}] = \sum_{xy|z \in \mathcal{R}} \mathbb{P}[xy|z \in \text{rt}(T) \mid \mathcal{A}]$$

since in particular $W \mid \mathcal{A}$ is a random variable expressible as the sum of $|\mathcal{R}|$ indicator variables (indicating whether the ith triplet is in $\text{rt}(T)$ or not) each of which takes on a value of 1 with probability $\mathbb{P}[xy|z \in \text{rt}(T) \mid \mathcal{A}]$. We do not include the computation of $\mathbb{P}[xy|z \in \text{rt}(T) \mid \mathcal{A}]$ in the pseudocode as there is a separate procedure for this. In particular, one must consider every possibility of whether x, y, z is assigned by $\mathcal{A}$ or not (so there are 8 possibilities to consider). In the case where they have all been assigned the computation simply involves checking whether $xy|z$ is induced by the tree T with the given assignments.

Otherwise if for example, x, y are assigned but z is not, then the probability is expressible as

$$1/N \cdot (N - |\text{leafs}(\text{lca}(x, y))|)$$

where N is the number of possible leaves z can be assigned to, $|\text{leafs}(\text{lca}(x, y))|$ denotes the number of leaves in the subtree rooted at the least common ancestor of x, y. This probability is derived from the fact that if x, y are assigned then in order for $xy|z$ to be induced by T, z must be assigned to any other leaf whose least common ancestor with x and y lies closer to the root than lca(x, y), and z is assigned to any particular leaf with probability $1/N$. The computation in the other 6 cases proceeds analogously.

Let $\mathcal{A}$ be an assignment of variables at the beginning of an iteration of the outermost loop in Algorithm 3. Let $\mathcal{A}'$ denote the assignment at the end of that iteration which is derived from $\mathcal{A}$ by assigning the leaf ℓ to some leaf node of T. Then by maximizing the expectation over all possible assignments we ensure that $\mathbb{E}[W \mid \mathcal{A}'] \geq \mathbb{E}[W \mid \mathcal{A}]$. Since we start with $\mathcal{A} = []$ and $\mathbb{E}[W]$, we guarantee that in the end our algorithm satisfies

$$\mathbb{E}[W \mid \mathcal{A}] \geq \mathbb{E}[W] = \left(\frac{1}{3} - \frac{4}{3(q+1)^2}\right) |\mathcal{R}|$$

and is therefore, a deterministic approximation algorithm.

4 Experimentation and Results

We test our implementation on the bacterial phylogenetic trees studied in [HBA$^+$16]. Their data has been made publicly available and was tested on q-MAXRTC in [JMT19]. Specifically, we run q-MAXRTC on the trees in supplementary datasets $1, 2, 3, 4, 5$ which we denote `nms1`, `nms2`, `nms3`, `nms4`, `nms5` following the notation in [JMT19]. The phylogenetic trees themselves were too large given our computational resources (an initial run of q-MAXRTC for $q = 2$ on `nms1` failed to terminate after running for a full day). Thus, we extracted smaller subtrees from the raw data sets and worked with those. Information for each dataset is presented in Table 1.

Data set	$\|T\|$	$\|\text{rt}(T)\|$
`nms1`	221	221815
`nms2`	173	105995
`nms3`	183	125580
`nms4`	185	129766
`nms5`	163	88560

Table 1: Static data associated with each data set, where $|T|$ denotes the number of nodes in the phylogenetic tree and $|\text{rt}(T)|$ denotes the number of induced triplets from that tree.

We summarize the raw data of the results in Table 2. For our experiments we use the ratio between the size of the induced triplet set for the input tree versus the size of the intersection between the induced triplet set of the resultant tree and that of the original tree. That is, if we write $\mathcal{R}$ for the set of input triplets, and T for the tree we construction from our experiment, then $|\mathcal{R} \cap \text{rt}(T)|/|\mathcal{R}|$ is our measure of accuracy. This is the approximation ratio for our algorithm, so in the case of $q = 2$, we know that we are guaranteed this value is at least $4/27$. Hence, our goal is to make this ratio as large as possible.

Our experiments are run on a PC running Microsoft Windows 10 Pro, with an Intel i7-6820HQ quad core processor clocking in at 2.70 GHz. Our experiments took approximately 6 hours to complete. The raw data from the experiments, in addition to the code for the experiements and the implementation can be found on the public github repo at `github.com/cakoch10/Phylogenetic_supertrees/`.

Data set \ q	3	7	15	31
`nms1`	0.42256	0.65794	0.74867	0.79375
`nms2`	0.39713	0.67703	0.79400	0.83638
`nms3`	0.65384	0.78664	0.90330	0.93316
`nms4`	0.44361	0.64395	0.74815	0.76804
`nms5`	0.56308	0.68065	0.74353	0.75299

Table 2: Raw data from our experiments running q-MAXRTC on the various datasets. The table entries denote the ratio between the intersection of the tree construction and the size of the input triplet set, i.e. $|\mathcal{R} \cap \text{rt}(T)|/|\mathcal{R}|$.

The plot in Figure 1 is a graphical depiction of the raw data in Table 2.

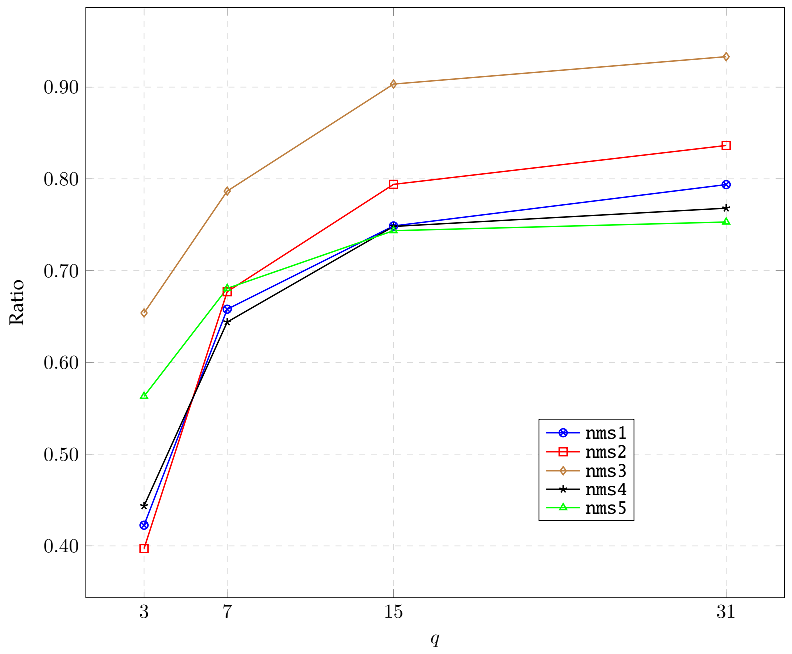


Figure 1: A plot capturing the results of q-MAXRTC on the five data sets.

5 Discussion

There are a few key differences between our experiments running q-MAXRTC and that of [JMT19]. First, our experiments are run on a strict subset of the data samples used in [JMT19]. In particular, the authors in [JMT19] use the full phylogenetic trees in the `nms2` and `nms4` data sets whereas we have to truncate the trees due to limited computational resources.

Also, we run our algorithm for $q = 3, 7, 15, 31$ while the authors in [JMT19] use $q = 3, 5, 7, 9$. We felt that it made more sense to enforce the resulting tree to be a complete binary tree consisting of q nodes which means q must take the form $2^n - 1$ for some n. Also the authors in [JMT19] randomly sample a fixed number of triplets from the induced triplet set for the phylogenetic trees. This random sampling makes their experiment impossible to repeat deterministically. So we opted for the more replicable approach of just using the entire induced triplet set. Note that even for relatively small trees, consisting of 200 nodes, the induced triplet set is very large, on the order of $150,000$. Thus, our choice to run the algorithm on the entire triplet set inevitably incurred a significant computational cost.

Our results match that of [JMT19] insofar as the accuracy of the reconstructed supertrees increases as q increases. Moreover, this increase seems to be more significant for smaller values of q. In general, our accuracy is quite high. For $q = 15$, the algorithm has an approximation ratio

$$\frac{1}{3} - \frac{4}{3(15+1)^2} = \frac{21}{60} \approx 0.328$$

however in practice we achieve an approximation ratio of $\approx 0.75 - 0.9$. This could be reflective of the fact that phylogenetic trees constructed from real data are generally more amenable to supertree reconstruction. Intuitively, this observation is unsurprising since phylogenetic trees capture informative relationships between induced triplets.

6 Future research directions

Our analysis was limited in several ways which can be subsequently improved upon. For example, we studied primarily phylogenetic trees of bacterial species. Presumably, supertree reconstruction across more general biological taxa (e.g. kingdoms) would yield less accurate results. It would be interesting to see just how inaccurate the resulting construction would be. Another interesting extension of the work here is to include comparisons with other q-MAXRTC alternatives such as `One-Leaf-Split` or Wu's algorithm as considered in [JMT19].

There are also several extensions of this work with respect to the study of the computational hardness of q-MAXRTC. A natural variant of q-MAXRTC is the problem where the constructed tree T is *enforced* to satisfy $|\mathcal{R} \cap \mathrm{rt}(T)| = |\mathcal{R}|$. Presumably, this problem ought to be computationally harder than q-MAXRTC since the algorithm is essentially required to construct the best possible tree. Moreover, this problem is potentially W[1] complete as the reliance on the parameter k is no longer necessary and hence our initial attempts of establishing hardness do not break down in the same manner. The authors in [JMT19] also mention this problem as worthy of future investigation. There are also several other natural variants of q-MAXRTC to consider including the case where triplets are weighted and the goal is to maximize the weight of $\mathcal{R} \cap \mathrm{rt}(T)$. Yet another variant is one where instead of parameterizing the problem by q, the number of internal nodes in the constructed supertree, we parametrize the problem by ℓ, the number of leaves of the constructed supertree.

7 References

[BDF+95] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, Michael T. Hallett, and Harold T. Wareham. Parameterized complexity analysis in computational biology. *Bioinformatics*, 11(1):49–57, 1995.

[BE04a] Olaf RP Bininda-Emonds. The evolution of supertrees. *Trends in ecology & evolution*, 19(6):315–322, 2004.

[BE04b] Olaf RP Bininda-Emonds. *Phylogenetic supertrees: combining information to reveal the tree of life*, volume 4. Springer Science & Business Media, 2004.

[BE05] Olaf RP Bininda-Emonds. Supertree construction in the genomic age. In *Methods in Enzymology*, volume 395, pages 745–757. Elsevier, 2005.

[BM11] Christina Boucher and Bin Ma. Closest string with outliers. *BMC bioinformatics*, 12(1):S55, 2011.

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[CM04] CJ Creevey and James O McInerney. Clann: investigating phylogenetic information through supertree analyses. *Bioinformatics*, 21(3):390–392, 2004.

[HBA+16] Laura A Hug, Brett J Baker, Karthik Anantharaman, Christopher T Brown, Alexander J Probst, Cindy J Castelle, Cristina N Butterfield, Alex W Hernsdorf, Yuki Amano, Kotaro Ise, et al. A new view of the tree of life. *Nature microbiology*, 1(5):16048, 2016.

[HRB+09] Daniel H Huson, Regula Rupp, Vincent Berry, Philippe Gambette, and Christophe Paul. Computing galled networks from real data. *Bioinformatics*, 25(12):i85–i93, 2009.

[JMT19] Jesper Jansson, Konstantinos Mampentzidis, and Sandhya TP. Building a small and informative phylogenetic supertree. In *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[JNSS05] Jesper Jansson, Joseph H-K Ng, Kunihiko Sadakane, and Wing-Kin Sung. Rooted maximum agreement supertrees. *Algorithmica*, 43(4):293–307, 2005.

[Lis07] Meri Lisi. Some remarks on the cantor pairing function. *Le Matematiche*, 62(1):55–65, 2007.

[MZ08] Ion Mandoiu and Alexander Zelikovsky. *Bioinformatics algorithms: techniques and applications*, volume 3. John Wiley & Sons, 2008.

[War18] Tandy Warnow. Supertree construction: Opportunities and challenges. *arXiv preprint arXiv:1805.03530*, 2018.