# A Collapsing Method for Efficient Recovery of Optimal Edges in Phylogenetic Trees

by

**Michael Hu**

A thesis
presented to the University of Waterloo
in fulfillment of the thesis requirement
for the degree of Masters of Mathematics
in Computer Science

Waterloo, Ontario, Canada, 2002

## Abstract

In this thesis we present a novel algorithm, *HyperCleaning*$^*$ for effectively inferring phylogenetic trees. The method is based on the quartet method paradigm and is guaranteed to recover the best supported edges of the underlying phylogeny based on the witness quartet set. This is performed efficiently using a collapsing mechanism that employs memory/time tradeoff to ensure no loss of information. This enables *HyperCleaning*$^*$ to solve the relaxed version of the Maximum-Quartet-Consistency problem feasibly, thus providing a valuable tool for inferring phylogenies using quartet based analysis.

**Acknowledgments**

Many people gave me constructive advice and encouragement during my stay at Waterloo. In particular I would especially like to thank my supervisor: Paul Kearney for his guidance, my parents for their support and understanding, my readers: Therese Biedl, Ming Li, and Dale Schuurmans, for their patience, and the folks in the bioinformatics lab for the sometimes constructive:), always geeky discussions, of which some were actually relevant to this thesis.

# Contents

# 1 Introduction

The study of phylogenetic inference is fundamental to the understanding of biological evolution at the molecular level. Phylogenetics not only poses an interesting problem in its own right but also plays a key role for solving many other problems in computational biology. To date, the problem has proven to be extremely hard from both a biological and computational standpoint. The biological difficulty stems from the lack of understanding of the complex evolutionary process. The computational difficulty arrives from the implicit tradeoff between the effectiveness/sophistication of an algorithm and its computational resource requirements. Sophisticated methods such as Maximum Likelihood (ML) [17], and Maximum Parsimony (MP) [16], are NP-hard optimization problems, making them computationally infeasible.

The Quartet Based paradigm (see 2.5) and numerous algorithms produced under it, have been designed to exploit some of the features of 'ML-like' algorithms but at the same time keep the computational requirements tractable. The main idea behind the Quartet Paradigm is to first break the original problem set into smaller subproblems of input size four, called a quartet. Expensive methods such as ML can then be employed to solve these quartets. Then a quartet method will analyze and assemble a final solution to the overall problem based on the solutions to the quartets in a consistent manner. However, under the quartet paradigm, any attempt to assemble a solution optimally from the solutions of the subproblems (known as the Maximum-Quartet-Consistency (MQC) problem) is itself NP-hard [49]. Many quartet based methods such as Quartet Puzzling (QP) [51], WO method [41], Short Quartet Method [15] solves the MQC problem in a heuristic sense. One method called HyperCleaning [3] will solve a slight relaxed version of the MQC problem exactly [1] under the assump-

[1]In a nutshell, the solution to the MQC problem yields a tree topology that optimally agrees with the solutions on the quartet set. HyperCleaning returns the set of tree edges that

tion that certain percentage of all the quartets (smaller subproblems) are solved correctly. Moreover both the running time and accuracy of HyperCleaning is highly coupled to the quality of the solutions on the quartet set. Thus HyperCleaning provides a valuable tool for solving exactly the relaxed version of the MQC problem for inferring phylogenies.

There are however two shortcomings of HyperCleaning which are the focus of this thesis : 1) HyperCleaning treats the solution to all the quartet sub-problems with equal weighting, which is not biologically sound since some sub-problems are much harder to tackle than others; and 2) HyperCleaning runs in parametric polynomial time (parameterized on both the problem input size, and the quality of the inferred quartet set). But under realistic/interesting biological scenarios, the solutions provided to the quartet set contain enough inaccuracies to force a high input parameter for HyperCleaning to solve the MQC problem. This in turn induces HyperCleaning to run in high order polynomial time. The main contribution of this thesis is then to present a time/space tradeoff collapsing mechanism that allows us to keep the accuracy guarantees on the solutions produced by HyperCleaning, but also reduce the running time to render it computationally efficient at the expense of more memory. This is achieved by collapsing the input space such that the problem can be solved efficiently, but with the same accuracy guarantees as provided by HyperCleaning. Moreover, we enable a general weighting scheme to be applied to the quartets.

In the rest of this chapter, we introduce in more depth the problem domain of phylogenetics and formally define the problem under study, our motivation for studying this problem and a brief outline of the remaining chapters.

optimally agrees instead. Note that the set of edges returned might not be consistent, in that they cannot all exist in the same tree.

## 1.1 Genes, Trees and Evolution

Phylogenetics is the study of evolutionary relationships among a set of objects. [2] In biology the objects in question are usually 1) organisms, 2) genomes, or 3) gene sequences. The fundamental assumption of evolution on a set of objects is that these objects are related by descent from a common ancestor. As such the objects through evolutionary pressure and mutation undergo a speciation process, whereby the object becomes two distinct objects. As such, the evolutionary history of a family of related objects (genes, organisms, etc.) can be represented by an evolutionary tree.

**Definition 1** *An evolutionary tree $T$ on a set $S$ of objects, is either a rooted or unrooted tree where the leaves of $T$ are labelled by elements of $S$. $T$ is called* ***weighted*** *if its edges have associated lengths (i.e. evolutionary distances, etc.).*

Tree $T$ succinctly describes the evolutionary relationship on the objects in $S$, where the internal nodes of $T$ are of degree 3 or higher, except its root may have degree 2. A root node denotes the common ancestor to all the leaves, and an internal node of degree 3 denotes an ancestral speciation event, where a parent object gives rise to two multiple children objects. This continues until the terminal leaves are reached, denoting the current observable objects (*extant objects*). An edge $uv \in T$ denotes the evolutionary time/distance on some object. In this sense, given an internal edge $uv \in T$, $uv$ denotes a particular object under the evolutionary process, and without loss of generality one endpoint $u$ denotes its conception from a speciation event from its parent object and another endpoint $v$ denotes its split into two new species. A tree where all internal (with exception of the root node) nodes have degree 3 is called binary and describes a **resolved**

[2]The objects need not be biological in nature. For example phylogenetic methods have been applied in determining the evolutionary relationship on a set of hand-written medieval manuscripts of Chaucer's Canterbury Tales; or to the evolutionary relationship/development of various linguistic forms.

evolutionary relationship of its leaves.

Evolutionary trees are also called *phylogenetic trees* and we will use the two terms interchangeably.

Given a set of objects, the tree that describes its true evolutionary history is called its *phylogeny*. Note that a phylogeny might not be resolved (i.e. binary), since to date the notion of binary descent for evolution is only a theory. In reality, we are given a set of objects to which its true evolutionary relationship, is not known. The problem is then to infer its phylogeny.

**Definition 2** - *Phylogenetic Inference Problem. Given a set of $n$ objects, $S$, infer the phylogeny on these extant objects.*

The inferred tree $T$ can have the option of having numeric labels on the edges, denoting the estimated evolutionary distance (time, amount of mutation, etc.) on that edge. We call these *labelled phylogenies.*

Figure (1) shows an outline of the phylogenetic inference process.

The evolutionary relationship on a set of objects can be represented by either a rooted or unrooted phylogenetic tree. A rooted tree provides more information in that the *root* object is the common ancestor to all extant leaf objects and moreover induces an evolutionary history between all pairs of nodes (e.g. we can answer questions such as: given nodes $u$, $v$, is $u$ an ancestor or descendant of $v$ of are they evolving in parallel?) With an unrooted tree however, we can only talk about the relative evolutionary relationships between the nodes (e.g. given 3 leaves $u, v, w$, which two are evolutionarily closer to each other than to the third ?). Figure (2) illustrates this difference.

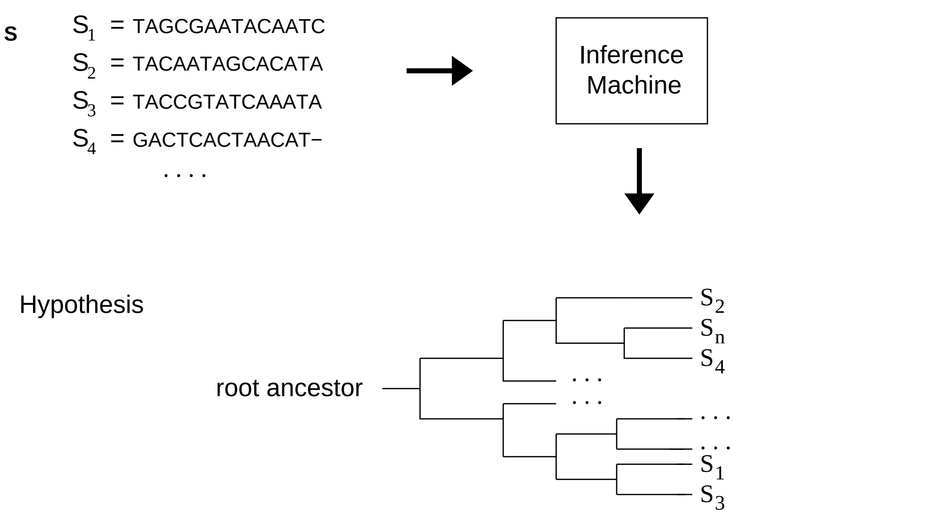


Figure 1: The phylogeny inference problem, where the input $S$ is a set of extant organisms. These can be represented by many types of data. In this case, the organisms are represented by an alignment of their gene sequences. The input $S$ is fed into an inference machine which then presents a hypothesis (e.g. phylogenetic tree) which attempts to describe the evolutionary relationship between the organisms of $S$ based on its input data.

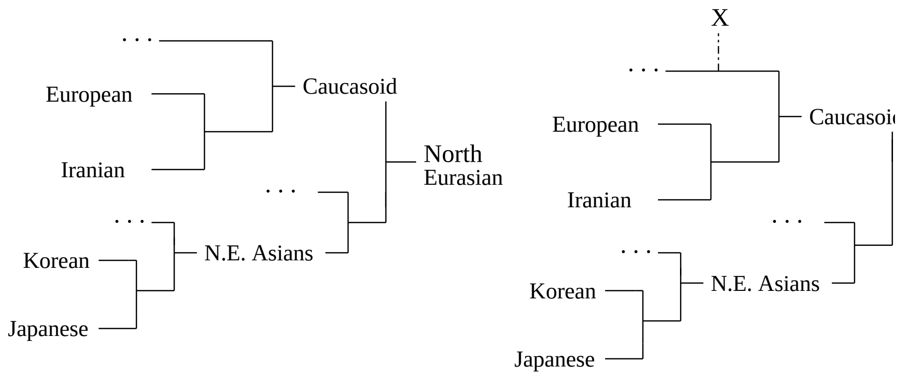


Figure 2: The left phylogeny is rooted at the node North Eurasian, elucidating the evolutionary history between the extant objects: European, Japanese, Korean, and Iranian. The right phylogeny (not including the dashed edge) is unrooted and as such we cannot determine the evolutionary history, only the relative relationship between the extant objects, such as that European is genetically closer to Iranians than to Koreans for example. We can even induce a fictitious evolutionary history by having the root $X$ as given in the right phylogeny, which says that *N.E. Asians* are a sub-group of the Caucasoids, whereas Europeans are not. Figures are sub-trees taken from the diagrams from L. Cavalli-Sforza, *Genes, peoples, and languages.* Scientific American. 1991.

Note that the size of the **tree space** grows exponentially with respect to the size of the input (number of leaves). Given a set of $n$ leaves, there are

$$(2n-5)!/(2^{n-3}(n-3)!)$$

possible unrooted phylogenetic tree topologies [34], and

$$(2n-3)!/(2^{n-2}(n-2)!)$$

possible rooted tree topologies. The exponential complexity in the number of candidate hypothesis trees on a set of data is one of the major factors in the difficulty of the phylogenetic inference problem.

In this thesis, we only consider evolutionary trees under a biological context, (i.e. objects in question are either a set of organisms, genomes, or genes). We note that phylogenetic analysis on genes, genomes, and species are inherently different. Phylogenetics at the species level typically implies the actual evolutionary history of the organisms under question, and can be investigated using fossil data, morphological data, or their genetic data (genomes and/or homologous gene sequences). This however does not imply that in general, a species tree is the same as, say, its genome tree, or its gene tree [34].

In the context of inferring the phylogeny on a set of entire genome sequences however the tree model is limited in that it lacks the expressiveness to succinctly describe multi-ancestral relationships known as *horizontal gene transfers*. This occurs when the genes of a particular organism's genome arises from several ancestors, usually from closely related species. In such cases, a graph is required to capture of evolutionary relationship. In nature, horizontal gene transfer generally occurs through non-reproductive means such as *transduction*, where genetic material is carried across species by viruses, or *transformation* where the genetic material is taken up by a species from the environment and subsequently integrated into its own genome. Figure (3) shows an example of horizontal gene transfer. For a more thorough treatment on the biological background of evolution, phylogenetics and established techniques, consult [53], [34].

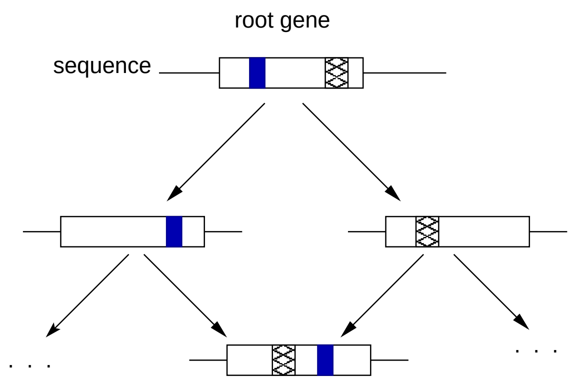


Figure 3: A graph must be used to expound the evolutionary relationship between genes having experienced horizontal gene transfer

.

In biology, the classical study of evolution seeks to fulfill Darwin's dream of deducing the resolved (if possible) evolutionary tree on the entire set of species of organisms found in Nature. The study of phylogenetics at the gene or genome sequence level is also becoming prevalent in various applications of comparative biology and genomics, where reliable and effective analysis of phylogenetic relationships is required. The following lists a few such applications.

**Applications of phylogenetics in genomics and proteomics:**

- Database search methods for homologous sequences, can be made more efficient using an evolutionary tree as a guide [42].
- Detecting potential gene expression regulatory elements by finding conserved regions in a set of orthologous, non-coding DNA sequences [5], requires accurate phylogenetic information.
- Phylogenetic analysis can provide valuable metrics for governing gene function prediction algorithms. [39, 13].
- Evolutionary trees can be used as valuable roadmaps for inferring gene duplication events in genomes [14], [9] inferring gene rearrangements [4],

and horizontal gene transfer problems [23], [48].

## 1.2 Motivation

We have listed a few of the many biological motivations for studying the phylogenetic inference problem. To date the problem is considered a 'hard' problem from both a biological and computational perspective. From a biological perspective, the challenge lies in our limited understanding of the evolutionary process. Consequently we are unable to construct a sufficiently realistic model of evolution that can always correctly interpret the input data describing our set of organisms in question as to resolve their evolutionary relationship. Figure

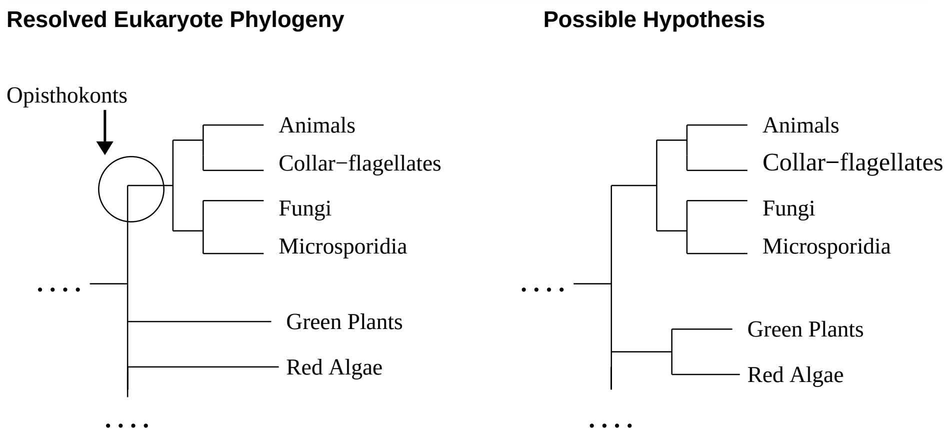


Figure 4: The phylogeny on the left shows the current resolved relationship between some Eukaryotes. The tree on the right is a possible hypothesis that is a slightly more resolved version of the evolutionary relationship of the Eukaryotes, where *Green Plants, Red Algae* are evolutionarily closer to each other than any other Eukaryotes.

(4) shows the currently known evolutionary relationship [38] between some Eukaryotes (i.e. organisms possessing a nucleus in their cells). Our current understanding of evolution and availability of the necessary data prevents us from determining say the evolutionary relationship between *Green Plants, Red Algae*, and the resolved subtree *Opisthakonts.* From a computational perspective, any algorithm for solving the phylogentic inference problem must deal with two implicit complexities:

1. The exponential size of the tree solution space wrt the size of the input. This rules out any algorithm which tries to exhaustively search and score the entire set of candidates from the tree space.

2. Biological modelling. As we will see in section 2, there is an implicit tradeoff in any algorithm for solving the phylogenetic inference problem, between the sophistication/realism of the incorporated biological model and the algorithm's complexity. Algorithms which incorporate complex models in general are superior at inferring the correct evolutionary relationships than those with simple models of evolution, but at a cost of increased complexity which sometime makes the method computationally intractable.

Thus faced with both the biological problem of not having a realistic model of evolution and the computational problems of having an exponentially large solution space as well as the evolution model/computational complexity tradeoff, the phylogenetic problem is in general very hard.

In this thesis we explore a method called HyperCleaning$^*$ which is based on the Quartet Method Paradigm (see section 2.5). HyperCleaning$^*$ tackles the phylogenetic inference problem purely from the computational complexity view. We introduce a space/time tradeoff mechanism that enables HyperCleaning$^*$ to scale up to large input sets. This mechanism is general and can be applied to other quartet based methods for speedup. Moreover, we address the accuracy of the quartet set inference stage using ensemble learning techniques.

## 1.3 Structure of This Thesis

The contribution of this thesis are:

- Generalize the HyperCleaning algorithm to handle weighted quartets.
- Introduce a space/time tradeoff (*collapsing*) mechanism that enables hypercleaning$^*$ to scale up to infer evolutionary trees over large data sets.
- A modular implementation of HyperCleaning$^*$ in C++.

In chapter 2, we give a paradigm based overview of several phylogenetic inference algorithms, with emphasis on some well known methods: Neighbour Joining, Maximum Likelihood, Parsimony, and some Quartet Based methods. In chapter 3, the hypercleaning$^*$ algorithm is presented in detail. Chapter 4 gives a detailed account of the collapsing mechanism that makes hypercleaning$^*$ efficient on large datasets. In Chapter 5 we present the use of ensemble learning techniques for the quartet set inference stage, and how to improve the quality of the witness quartet set. In chapter 6 we present our experimental results, further avenues of research and the conclusion.

# 2 Phylogenetic Inference

## 2.1 Evolutionary Model - A Biological Perspective

Due to the recent exponential accumulation of sequence data as well as the advances in sequencing technology, biological sequences are becoming an ubiquitous source of data from which informative knowledge about the evolutionary process can be gleaned. All biological information on a given organism is stored in raw form in its DNA sequence (called its *genome*), on an alphabet of size 4. Encoded in these sequences are all the instructions necessary for a particular organism to survive in its current environment. Among the DNA sequences are genes which contain instructions for synthesizing proteins, which are the building blocks of crucial biological processes and cellular machinery in an organism. During a stage called *transcription*, certain portions of the DNA sequence (i.e. the relevant protein coding genes) are transcribed into amino acid sequences on an alphabet of size 20 that encodes protein building information. The cellular machinery subsequently builds proteins from these amino acid sequences in a process called *translation*. Most biological sequences under study are either DNA or amino acid sequences. For simplicity we call the individual letters of a given sequence *nucleotides*. Consult Li [34] for a biological treatment on sequences, and Sankoff [46], or Eddy et. al. [12] on sequence analysis techniques. Models of evolution on biological sequences fundamentally based on the assumption that evolution induces change or substitution of nucleotides in sequences in a stochastic fashion. As such we can view the evolutionary model as a distribution of change on the nucleotide level parameterized by time, expressed as the probability:

$$p(a \to b \mid t) \tag{1}$$

that nucleotide $a$ evolves to nucleotide $b$ within $t$ units of time, where $a, b \in \Sigma$, and $\Sigma$ is a finite sequence alphabet,

The following two assumption are also made on our sequence model of evolution. First, the *memoryless* property assumes that our model of evolution on sequences satisfies the following:

$$p(a \rightarrow b \mid t_1 + t_2) = \sum_c \Big( p(a \rightarrow c \mid t_1) + p(c \rightarrow b \mid t_2) \Big) \tag{2}$$

Secondly, we assume evolution is *time reversible* which implies that

$$p(a)p(a \rightarrow b \mid t) = p(b)p(b \rightarrow a \mid t) \tag{3}$$

Numerous computational models have been proposed to estimate the true distribution of equation (1). Some models, such as Jukes-Cantor, are simple and assume that the evolutionary process across a sequence is site independent, i.e. given sequences $x = (x_1, .., x_n), y = (y_1, .., y_n)$ :

$$p\Big( (x_1, x_2, ..., x_n) \rightarrow (y_1, .., y_n) \mid t \Big) = \prod_{i=1}^{n} p(x_i \rightarrow y_i \mid t) \tag{4}$$

Other models are more complicated and take into account correlations across multiple nucleotides to model evolutionary patterns such as mutation variance among coding vs. non-coding sites, codon position bias, etc. Consult [53], [36] or [33] for a more thorough treatment.

## 2.2 Computational Methods

Most computational algorithms for solving the phylogenetic inference problem proposed throughout the years can be characterized by whether they explicitly incorporate the model of evolution into their inference procedure, or employ some implicit evolutionary model. Two broad class of methods illustrating this are *sequence* and *distance* based methods.

Sequence based methods work directly with the aligned gene sequences or genomes as input, while distance based methods on input $S$ of size $n$ requires a $n \times n$ distance matrix $M$ where entry $M_{ij}$ encodes the estimated evolutionary distance between sequences $i, j \in S$. The distance matrices themselves are often obtained from sequence data, but this stage is removed from the inference methods.

In general, most distance based methods such as neighbour joining [45] and Weighbor [6] are computationally fast lending it to handle large data sets, but are limited in accuracy since the only information available are the estimated pair-wise distances. Methods following the explicit model based approach, such as maximum likelihood [17], parsimony [16], and structural EM [20] tend to be more accurate and robust across the input distribution, at the price of being computationally more expensive.

Another set of methods are based on the quartet method paradigm and we defer their discussion until section 2.5. Quartet methods are meta-methods in that they use existing methods (distance, or sequence based) to infer the set of all quartet sub-trees and then re-combine these pieces of a puzzle into a larger picture. For the remainder of this section, we examine in closer detail several established methods under the sequence based and distance based framework. Through these methods, we examine the intrinsic coupling between the computational complexity of a method and the amount of biological information or the sophistication of the biological model a computational method has to work with. This illustrates the complexity/effectiveness tradeoff inherent to all phylogenetic inference methods.

## 2.3 Sequence Based Approaches

In a sequence based approach to phylogenetic inference, we are given a set of $n$ aligned sequences $S$, which we view as extant or observable data generated by the evolutionary process. Our inference algorithm, on input $S$ and some explicit model on sequence evolution, will proceed to infer the underlying phylogeny. Figure (5) shows the inference process.

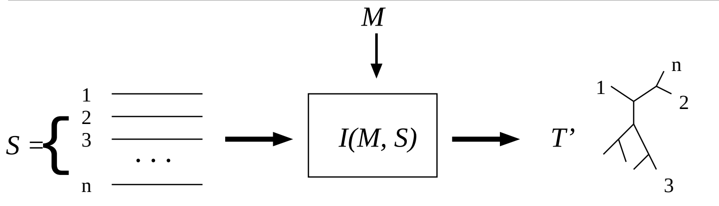


Figure 5: High level diagram of phylogenetic sequence based inference.

One simplifying assumption that most model based methods make is the aforementioned site independence (see equation 4), also called *point mutation* process. This implies that given our input set $S$ of aligned sequences, each nucleotide site amongst the sequences evolve independent of the other sites. Thus evolution on a set of sequences can be viewed simply as $l$ independent, simultaneous evolutionary processes effecting the sequences in question, where $l$ is the length of each of the gene sequence. In reality point mutation does not hold since some sites in genes will correlate with the evolutionary rate of other sites further along the same gene.

### 2.3.1 Maximum Likelihood

The method known as maximum likelihood [17] is a computationally expensive method that has a formal statistical framework. Maximum likelihood (ML) represents a candidate hypothesis (i.e. phylogenetic tree) as a parameterized naive bayes net. The naive bayes net is parameterized by some model of evolution $M$. The underlying objective is to assign a likelihood score to a hypothetical tree $T$ given the data $S$. In essence, ML calculates the likelihood that a particular tree $T$, under some model of evolution $M$, produced the observed data set $S = \{S_1, S_2, \ldots, S_n\}$ in question. This can be expressed by [3]

$$Pr(S \mid T, M) \tag{5}$$

which can be expanded to the following:

$$Pr(S_1, S_2, ..., S_n \mid T, M) \tag{6}$$

As such the hypothesis (i.e. tree) that produces the maximum likelihood of generating the data with respect to the above expression is selected as the most probable phylogeny, denoted by $T^{ML}$:

$$T^{ML} = \text{argmax}_T Pr(S|T, M) \tag{7}$$

Thus expression (5) outlines the metric to use for scoring a particular tree and (7) dictates how to pick the best tree from all possible trees from the tree space.

In order to reduce the computational complexity of calculating (5), most implementations of ML assumes that the sequences of $S$ have evolved under point mutation. Consequently for a particular sequence $S_i \in S$, we have

$$Pr(S_i \mid M, T) = \prod_{j=1}^{n} Pr(S_i^j \mid M, T) \tag{8}$$

[3]This method is also known as maximum relative likelihood, since we assume a uniform distribution of the tree space. If we computed instead $P(T)P(S \mid T, M)$, then this is another form of ML known as a maximum posterior probability (MAP) estimate [50]

where $S_i^j$ is the $j$th nucleotide in the sequence $S_i$. As such we can simplify equation (5) to

$$Pr(S_1, S_2, ...S_n \mid M, T) = \prod_{j=1}^{m} Pr(S_1^j, S_2^j, ..., S_n^j \mid M, T) \tag{9}$$

Thus from (5), (6) and (9), we have

$$Pr(S \mid M, T) = \prod_{j=1}^{m} Pr(S_1^j, ..., S_n^j \mid M, T) \tag{10}$$

To compute the above probability, it suffices that we can compute the following, for all $j = 1, \dots, n$:

$$Pr(S_i^j, ...S_n^j \mid M, T) \tag{11}$$

The joint in (11) can be computed efficiently by decomposing it into the product of local dependencies. This general technique have been studied extensively in the statistical machine learning community. Consult [11], [19] for a thorough treatment.

Given a set of sequences $S = \{S_1, .., S_n\}$, and a hypothetical tree $T$ with leaves labelled by $S$ describing its possible evolutionary history, it turns out that the local independence structure for computing (11) is almost described by the topology of $T$.

To convert the hypothesis tree $T$ into a bayesian network that describes the local independencies for efficiently computing (11) requires picking of a 'starting' leaf $v_0$ in tree $T$, either by randomly picking a node if the underlying evolutionary process is assumed time reversible 4, [17], or by determining an outgroup, see [25]. Then visit all edges of $T$ and put a direction indicating the direction of evolution assuming that $v_0$ is the ancestral node. Having converted hypothesis tree $T$ into a bayesian network $T_G$, we can solve for (11) using the standard *forward algorithm* on parameterized bayesian networks as described in

[19]. We will not go into the details of the forward algorithm here.

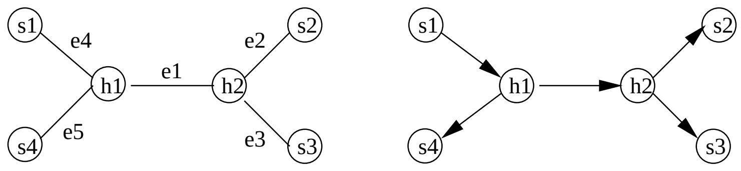


Figure 6: The left is a tree $T$ on leaves $S = s_1, .., s_4$, with edge lengths $e_1, e_2, e_3, e_4$. On the right is the corresponding parameterized bayes net (assuming $s_1$ is the starting node), describing the distribution $Pr(s_1, .., s_4, h_1, h_2|\theta)$ in terms of local conditionals, where the conditional say, $P(s_3|h_2)$, depends on the parameter $e_3$.

For illustration consider a simple example as given in Figure (6). The candidate tree on 4 leaves on the left has been converted to the bayes net on the right. As such using the *forward* algorithm, equation (11) is simplified into the following product of local conditionals:

$$Pr(s_1^j, .., s_4^j \mid M, T) = \tag{12}$$

$$\sum_{h_1^j} \sum_{h_2^j} Pr(s_1^j, .., s_4^j, h_1^j, h_2^j \mid M, T) = \tag{13}$$

$$Pr(h_1^j) \sum_{h_1^j} Pr(h_1^j|s_1^j) Pr(s_4^j|h_1^j) \sum_{h_2^j} Pr(h_2^j|h_1^j) Pr(s_2^j|h_2^j) Pr(s_3^j|h_2^j) \tag{14}$$

where we marginalize across all possible values for the variables: $h_1^j, h_2^j$ (representing the nucleotide values of the internal unseen nodes).

The only remaining issue is how to get the conditional values in (14) of the form:

$$Pr(v_i|v_j); v_i, v_j \in \{s_1^k, s_2^k, s_3^k, s_4^k, h_1^k, h_2^k\} \tag{15}$$

In a parameterized bayes net, conditionals are explicitly defined by a function $f$ on the edge length parameters $e_1, e_2, ... \in \theta$, such that

$$Pr(v_i|v_j) = f(v_i, v_j, e_k) \tag{16}$$

, where $e_k$ is the edge connecting the adjacent nodes $v_i, v_j$.

To make a good choice for the function $f(v_i, v_j, e_k)$, first consider the biological implication of a conditional, say

$$Pr(h_2|h_1) \tag{17}$$

Let $e_k$ be the edge length between nodes $h_1, h_2$, then the above conditional denotes the likelihood of the state of $h_1^k$ evolving to the state of $h_2^k$ given time $e_k$. Thus conditional (17) is precisely the likelihood of nucleotide substitution between two sequences, as given by the parameter $M$.

The computational difficulties associated with ML is in determining the tree parameter $T$ and its edge weights that maximizes (16). Since the size of the tree space grows exponentially wrt the number of input taxas, it is both computationally impractical and infeasible to search through every tree $T$, and simultaneously determine the corresponding edge weight parameters to maximize the ML score [53].

### 2.3.2 Parsimony

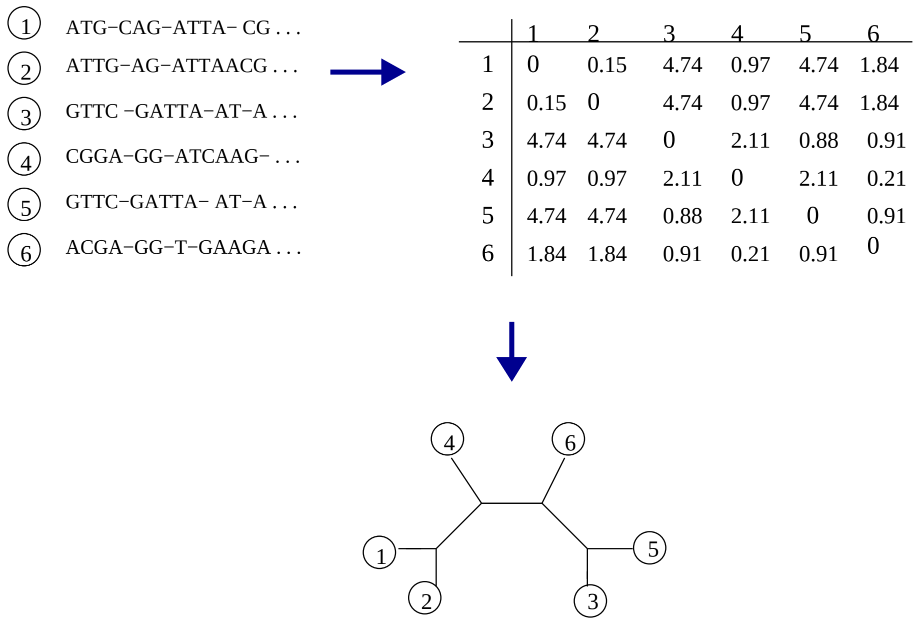
Parsimony is a sequence based method which assumes a simpler model of evolution than ML. Unlike ML, which allows the user to specify its model $M$ of evolution, parsimony incorporates an implicit model of evolution. Parsimony assumes that evolutionary processes act on sequences in a divergent, stationary, and point mutation manner. Consequently we may employ the notion of Ockham's Razor in that the tree (i.e. hypothesis) which takes the minimum number of mutations to explain is the most probable one. Based on the simple, implicit model of evolution adopted by parsimony, several discrete algorithmic techniques have been developed for parsimony that find the maximum likelihood tree $T$ [18], [16]. In essence finding the most parsimonious tree is equivalent to solving the “Hamming-distance Steiner tree problem”. Parsimony is effective as an estimator provided that the data in question does not violate the Okham Razor assumption too much, which is not unrealistic in biological evolution due to back substitutions and convergent evolution. Although parsimony techniques are computationally much faster than maximum likelihood, searching through the entire tree space is still exponential on the size of the input. Similar to ML, most parsimony algorithms used today, such as in PAUP [52], rely on heuristics to constrain the number of tree candidates searched from the tree space.

## 2.4 Distance Based Approaches

The motivation behind distance based methods is that the evolutionary distance $d_T(x, y)$ between leaves $x, y$ in the underlying phylogeny $T$ is unique to $T$ [54]. Various methods exist for estimating pair-wise evolutionary distances. Pairwise evolutionary distances between sequences can be estimated by adopting some sequence based model of evolution such as Jukes-Cantor [28], Kimura-Two, etc.. Note that this approach requires that the extant set $S$ be an aligned set of gene sequences. Other approaches such as [33] uses an information theoretic estimate of the pair-wise evolutionary distances between a set $S$ of whole genomes, rather than a set of aligned sequences. Figure (7) illustrates the high level idea behind distance based methods.

### 2.4.1 Neighbour Joining

Neighbour Joining (NJ) [45] is a greedy, computationally fast, distance based method that iteratively builds the phylogeny estimate rather than search across the tree space. NJ on a set of input leaves $S$, requires the evolutionary distances



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0.15 | 4.74 | 0.97 | 4.74 | 1.84 |
| 2 | 0.15 | 0 | 4.74 | 0.97 | 4.74 | 1.84 |
| 3 | 4.74 | 4.74 | 0 | 2.11 | 0.88 | 0.91 |
| 4 | 0.97 | 0.97 | 2.11 | 0 | 2.11 | 0.21 |
| 5 | 4.74 | 4.74 | 0.88 | 2.11 | 0 | 0.91 |
| 6 | 1.84 | 1.84 | 0.91 | 0.21 | 0.91 | 0 |

Figure 7: Distance based inference on a set of six extant organisms as described by an alignment of their gene sequences. First pairwise sequence evolutionary distances are estimated and stored in a distance matrix. Then an estimate of the phylogeny is produced.

$D_{ij}$ between all pairs $i, j \in C$. These distances are stored in a *distance matrix* $M$, where $M_{ij} = D_{ij}$. The method used for estimating these distances varies and partly depends on the data that describes the input set $S$. If the leaves of $S$ are described as aligned gene sequences, then sequence based distance models such as Jukes-Cantor [28], or Kimura-Two model [32] may be employed.

Given a distance matrix $M$ on input set $S$, NJ proceeds to build an estimated phylogeny by successively finding the next 'neighbouring' leaf to append to the current partially resolved tree.

A partially resolved tree is captured by a clade, which is simply some estimate of a subtree of the underlying phylogeny. A trivial clade is simply some given leaf of $S$. Given clade $C$, the leaves of $C$, denoted $Leaves(C)$ are a subset of the input leaves $S$. When joining two clades $C_i, C_j$ to form a single clade $C_{ij}$, graphically we introduce a new internal node $root_{ij}$, and without loss of generality, attach the root of clade $C_i$ as the left descendant to $root_{ij}$ and attach the root node of $C_j$ as the right descendant of $root_{ij}$. Then the new node $root_{ij}$ becomes the root node of clade $C_{ij}$. The root node of a trivial clade is just the leaf itself. Figure (8) illustrates this process.

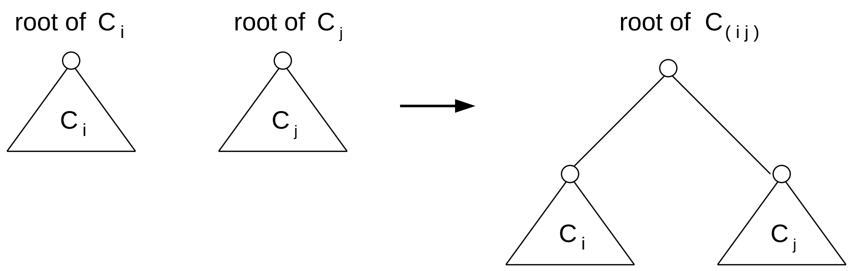


Figure 8: The joining of two clades $C_i, C_j$ into a new clade $C_{(ij)}$.

```
$T \longleftarrow$ NEIGHBOUR-JOINING $(S, M)$

1   Start with clade set $C = \{C_1, .., C_n\}$, where each $C_i$
```

```
-   is simply the leaf $i \in S$.
2   WHILE ( $C$ has more than 2 elements )
3       compute for clade $i$, the normalizing constant
-       $u_i = \sum_{k \neq i} \frac{D_{ik}}{n-2}$
4       choose $i, j$ for which $D_{(C_i, C_j)} - u_i - u_j$ is smallest.
5       Join cluster $C_i, C_j$ into new cluster $(C_{ij})$, and update
-       tree $T$ as described in Figure(8).
6       Compute and store the distances between the new cluster $(C_{ij})$
-       with all the other clusters $C_k$: $D_{(ij),k} = \frac{1}{2}(D_{ik} + D_{jk} - D_{ij})$
7       $C = C - C_i - C_j + C_{(ij)}$
8   return $T$.
```

The NJ algorithm is guaranteed to terminate since each iteration of lines 3-7 results in a net decrease of one element of the clade set $C$. Thus NJ will run for $n-2$ iterations where $n$ is the size of the original set $S$ (i.e. no of leaves). During each iteration of lines 4-5, we must perform pairwise computations of the form $D(x, y)$ between all pairs $x, y \in S$. This involves at most $n^2$ lookups from the distance matrix $M$. In line-6, we must perform an additional $O(n)$ computations and updates to the distance matrix $M$. This yields a total worst running time of $O(n^3)$, but realistically has a running time of $O(n^2)$. Thus NJ is a very efficient algorithm.

Moreover, if the inferred evolutionary distances are true relative to each other, then an *additive* relationship is induced.

**Definition 3 -** *Additive Matrix* If a distance matrix $M$ on set $S$ is additive, then the following condition must hold:

In the underlying phylogeny $T$ with leaves labelled by $S$, two leaves $i, j$ are neighbours iff

$$D(i,j) + D(x,y) \leq D(i,x) + D(j,y) \ \forall x, y \neq i, j \in S \qquad \odot$$

NJ has the property that if the distance matrix $M$ is additive, then the estimate tree it produces will be the real underlying phylogeny. The difficulty is real life is that most datasets used to describe the leaves set $S$ does not hold enough information for existing distance estimation methods to generate the set of additive distances required by NJ.

## 2.5 Quartet Based Methods

Quartet methods is a hybrid method based on the observation that a phylogeny is uniquely described by its set of *induced quartets*. A *quartet* is a set of 4 sequences and a *quartet topology* is simply the phylogeny (i.e. the actual unrooted evolutionary tree) on these 4 sequences. A given quartet $(a, b, c, d)$ has four possible topologies, denoted $ab|cd, ac|bd, ad|bc, abcd$. A quartet topology $ab|cd$ is

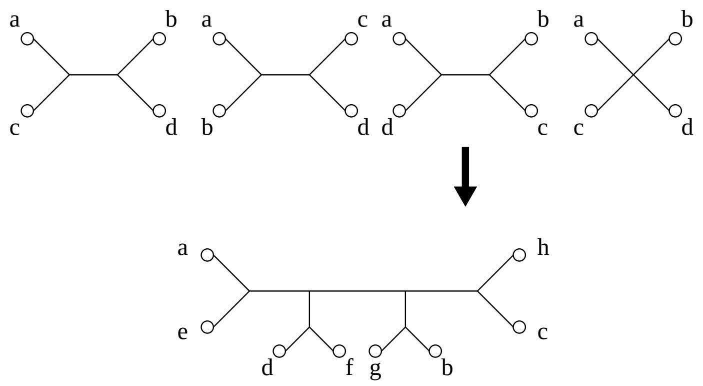


Figure 9: A quartet a,b,c,d, its four topologies, and its induced topology in tree $T$

induced in a phylogeny $T$ iff $P_T(a,b) \cap P_T(c,d) = \emptyset$, where $P_T(a,b)$ is the path in tree $T$ between leaves $a, b$. Figure (9) shows a quartet, its four topologies and its induced topology in a tree. Quartet methods in general consists of two stages, a quartet set inference stage ($QI$), and a re-combination or tree inference ($TI$) stage. Figure (10) shows a diagram of quartet based methods for inference.

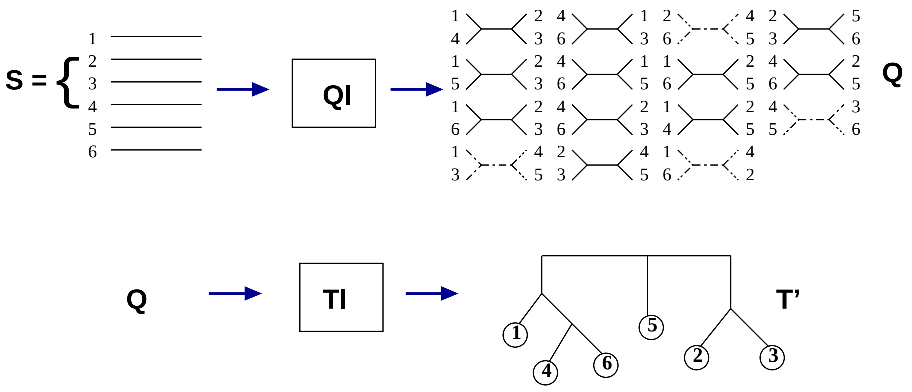


Figure 10: A quartet based method consists of two components ($QI, TI$).

The first stage consists of inferring all $\binom{n}{4}$ quartet topologies over an input data set $S$ of $n$ objects. In essence we are inferring the phylogenetic relationship on all possible quartets of $S$, denoted by set $Q$. Any existing phylogenetic method can be used for constructing $Q$, depending partly on the type of data characterizing $S$. Any sequence based methods such as ML or parsimony can be used to infer the set of quartet topologies. Distance based methods on quartets such as the popular *weak four point* condition, or the *ordinal quartet method* [29] can also be used. Let us examine the *weak four point* condition:

**Definition 4** - *Weak Four Point Condition*

Given 4 organisms, $a, b, c, d$ along with the pairwise estimated distances $D(i, j)$, $i, j \in \{a, b, c, d\}$, the following condition enables us to infer the quartet topology:

$$D(a,b) + D(c,d) < min\{D(a,c) + D(b,d), D(a,d) + D(b,c)\} \Longrightarrow ab \mid cd$$

$$D(a,c) + D(b,d) < min\{D(a,d) + D(b,c), D(a,b) + D(c,d)\} \Longrightarrow ac \mid bd$$

$$D(a,d) + D(b,c) < min\{D(a,b) + D(c,d), D(a,c) + D(b,d)\} \Longrightarrow ad \mid bc$$

Else *abcd* $\odot$

The theoretical justification for the weak four point condition on inferring quartet topologies is based on the following observation:

If $D$ provides an accurate estimate of the true pairwise evolutionary distances (in either a relative or absolute sense), then for any quartet $a, b, c, d$, exactly one of the above condition will be observed. This is called the *four point condition.*[8].

The recombination stage takes the information provided by $Q$ and combines these quartets to form an estimate $T'$ of the unknown phylogeny $T$. Most of the existing quartet methods re-combine the quartets using a tree-building scheme by recovering or inferring the edges of the unknown phylogeny.

**Definition 5 -** *edge of a phylogeny*
An edge $e$ in an evolutionary tree $T$ is defined by the bipartition $(X, Y)$ where $X, Y$ denotes the leaves set of the two disjoint sub-trees of $T$ resulting from removing $e$. $\odot$

We can view the inferred quartet set $Q$ as containing many small pieces of clues to the big picture. The difficulty is that some of the smaller pieces of the puzzle contains errors. Specifically set $Q$ will contains *quartet errors*.

**Definition 6 -** *Quartet error*
Given an inferred quartet set $Q$ on input $S$, a quartet topology $ab|cd \in Q$ is a quartet error if $ab|cd \notin Q_T$, where $Q_T$ is the quartet set induced by the unknown phylogeny $T$. $\odot$

Current quartet based methods rely on the underlying assumption that although set $Q$ will contain errors, and most of the quartet topologies in $Q$ are indeed induced in $T$. Thus to best infer $T$, the recombination stage will build an estimate $T'$, whose induced quartet set $Q_{T'}$ maximizes its consistency with $Q$. Formally, quartet based methods try to solve some variation of the following optimization problem:

**Maximum Quartet Consistency (MQC)**
Instance: A set $Q$ of inferred quartet topologies over input set $S$.
Goal: Find an evolutionary tree $T'$ on input $S$ that maximizes $\mid Q_{T'} \cap Q \mid$. This then estimates the unknown phylogeny $T$.

An equivalent optimization problem is the *Minimum Quartet Error* problem:

**Minimum Quartet Error (MQE)**
Instance: A set $Q$ of inferred quartet topologies over input set $S$.
Goal: Find an evolutionary tree $T'$ that minimizes $\mid Q_{T'} - Q \mid$.

The above two optimization problems infers an estimate $T'$ of the real phylogeny $T$, such that T' maximizes the support or minimizes the distance from the guide set $Q$. Thus if $Q$ is reasonably close to the quartet set $Q_T$, then the quartet method will produce a tree very close to the true phylogeny. The following theorem [27] relates the amount of quartet error allowable on the inferred quartet set $Q$, but still retain enough information about $T$.

**Theorem 1**
Consider input set $S$ and its phylogeny $T$. Assume we have a quartet set $Q$ estimating $Q_T$. If each edge $e = (X, Y) \in T$ has less than

$$(\mid X \mid -1)(\mid Y \mid -1)/2 \tag{18}$$

quartet errors, then $T$ is the unique evolutionary tree that minimizes $\mid Q_T - Q \mid$. $\odot$

*proof-* Consult [27].

In essence, Theorem 1 justifies quartet based methods in the context of the

MQE or MQC problem. Consider a quartet method and input dataset $S$. Theorem 1 stipulates that on input $S$, if the quartet method infers the quartet set $Q$, satisfying (18) wrt the unknown $T$, then $Q$ contains enough information for the re-combination stage of the method to recover the phylogeny $T$. Moreover this is accomplished if the re-combination stage solves the MQE (or MQC) problem.

Thus the effectiveness of a quartet method is dictated by the following:

- In the first stage, its estimation $Q$ of $Q_T$ should be sufficiently close, ideally satisfying equation (18).
- In the re-combination stage, it should as best as possible solve the MQE/MQC problem.

On the first point, the accuracy of the inferred quartet set $Q$ depends on the effectiveness of the phylogenetic inference method employed on a particular quartet set. Some quartets where one or more of its sequences have undergone convergent evolution might result in a quartet error. Or a quartet may have two long branches in its topology in $Q_T$, thus making it hard for most phylogenetic methods to correctly infer its topology. The concept of taxanomic sampling have been proposed to improve the accuracy of $Q$, by specifically handling some of these pathological cases. The idea of quartet taxanomic sampling is to infer the topology of a quartet $(a, b, c, d) \subset S$ by considering a larger set $(a, b, c, d, e, f, ..) \subseteq S$, and infer the evolutionary tree on this larger set, then extracting the quartet topology on $(a, b, c, d)$. Taxanomic sampling have been shown to affect certain pathological cases such as long branch attraction cases [24, 25], as well as on quartets in the Felsenstein zone [21]. But in general, taxanomic sampling does not seem to be effective on the overall accuracy of quartets in $Q$, as shown by an experimental study by [2].

In [1], a method is been developed to identify those sequences that might contribute to quartet errors. These so called 'bad apples' can be isolated, and quartets involving them can be dealt with differently than other sequences. For example, taxanomic sampling can be employed on quartets involving the bad apples and normal inference methods can be employed on the other sequences.

The art of designing quartet based methods falls mostly on the algorithm for facilitating the re-combination stage. In essence if all quartet methods use the same set of inferred quartets, it is the re-combination stage that dictates both its computational efficiency and its effectiveness at solving the MQC/MQE problem. Next we examine several quartet based methods, specifically their re-combination stages.

### 2.5.1 Quartet Puzzling

Quartet puzzling [51] uses a heuristic approach to solving the MQE problem in its quartet re-combination stage. Given a set of inferred quartets $Q$ on input $S$, puzzling first randomly permutes the set of leaves in some random order $O = \{s_1, s_2, .., s_n\}$. Then a tree estimate is constructed by iteratively adding the leaves of the input set $S$ to the existing tree. Initially the tree is on the first four leaves in $O$, $s_1, s_2, s_3, s_4$, as induced by $Q$. The rest of the sequences of $S$ are then successively inserted as given by the ordering in $Q$, onto the branches of the core tree. For each sequence $s$ to be added to the existing tree $T'$, puzzle penalizes each edge of $T'$ for accepting $s$ for insertion based on the consensus of the quartets in $Q$ involving $s$. Specifically, for all quartets involving $s \in Q$ of the form $sa \mid bc$, each edge in $T'$ along the path from $b$ to $c$, (i.e. $\forall e \in T'_P(b, c)$), gets a penalty of 1. In other words, we penalize all edges of $T'$ that disagree with a particular quartet topology of $Q$ involving $s$. When done for all quartets of $Q$ involving $s$, we are in essence implicitly evaluating all the edges of $T'$ as candidates for insertion of $s$, with the objective of minimizing the number of quartet errors. The sequence $s$ is then inserted into the edge with the minimum penalty score. Thus quartet puzzling is a heuristic for solving the MQE problem. This randomization of sequences and subsequent constructing of the estimated tree is repeated several times and finally those compatible edges that occur in more

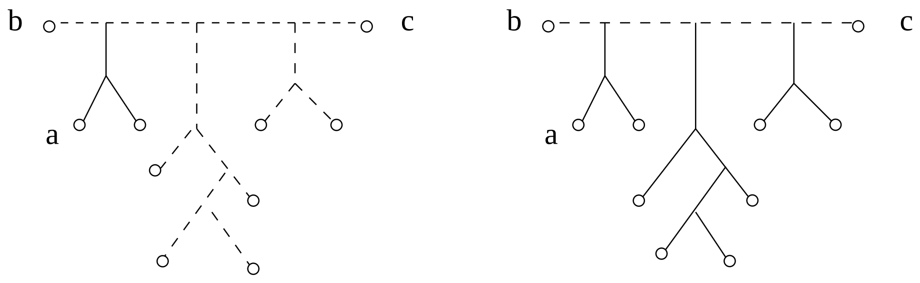


Figure 11: Suppose $sa \mid bc \in Q$. The dashed edges on the left tree shows those edges which should be penalized since inserting a sequence $s$ into them would contradict the support of $sa \mid bc$. The same tree on the right shows those edges that actually gets penalized using puzzling's penalty.

than half of the estimated trees are returned as the resulting phylogeny. Note that the resulting tree might not be fully resolved, and is not guaranteed to be the optimal solution to the MQE problem.

The effectiveness of the puzzling heuristic for solving the MQE problem has not been formally investigated. However one shortcoming in the edge penalty scoring might adversely affect its effectiveness for the MQE problem, as described in [3], [30]. Note that the goal of puzzling at some step in the tree construction stage is to insert the new sequence $s$ into an edge of $T'$ such that the number of quartets in $Q$ disagreeing with it is minimized. To achieve this, Puzzling penalize, for every given quartet $sa \mid bc \in Q$, the edges in $T'$ on the path from $b$ to $c$. In fact, puzzling can underestimate the penalty of inserting $s$ onto some edge(s). The example in figure (11) is a modified version of the one from [30] that illustrates this.

### 2.5.2 Hypercleaning

Hypercleaning [3] solves the MQE problem using two stages. Stage 1 consists of recovering the set, denoted $Best(Q, m)$, of all possible edges $e = (X, Y)$ with less than a certain amount of quartet errors with respect to $Q$. The parameter $m$ defines the error tolerance. Stage 2 then consists of some method of selecting a subset of compatible edges of $Best(m)$ as the estimated phylogeny. We use the notion of a bipartition for an edge as described in definition 5.

*Stage 1*- Hypercleaning recovers all edges with less than certain amount of quartet errors wrt $Q$. This requires a scoring function on edges, and a threshold on the scores for deciding which edges should be selected based on their scores.

**Definition 7 -** *scoring function* $\sigma((X, Y), Q)$
Given an edge $e = (X, Y)$, its score, denoted $\sigma((X, Y), Q)$ is given by

$$\sigma((X, Y), Q) = \frac{4 \mid Q(X, Y) - Q \mid}{\mid X \mid (\mid X \mid -1) \mid Y \mid (\mid Y \mid -1)} \tag{19}$$

where $Q(X, Y)$ is the set of all quartet topologies in $Q$ induced by the edge $(X, Y)$ as described below

$$Q(X, Y) = \{\text{all quartet topologies, } ab \mid cd, \text{ such that } a, b \in X, c, d \in Y\} \tag{20}$$

and the term

$$\mid Q(X, Y) - Q \mid$$

is simply the number of quartet errors across edge $e$ wrt $Q$, defined by those quartet topologies in $Q(X, Y)$ not appearing in $Q$. $\odot$

Note that an edge $e = (X, Y)$ has $\binom{X}{2}\binom{Y}{2}$ induced quartet topologies. Thus, the scoring on an edge defined in equation (19) is in essence the normalized number of quartet error across $e$ wrt $Q$. This normalization is necessary for comparing the support of two edges with different bipartition sizes can be pairwise compared. Under such a scoring function $\sigma$, HyperCleaning on parameter $m$ returns the set, denoted $Best(m, Q)$, of all edges supported by $Q$ under the following scoring threshold:

$$Best(m, Q) = \{(X, Y) :: \sigma((X, Y), Q) < \frac{2m}{\mid X \mid |Y|}\} \tag{21}$$

Combining Equation (19) and Equation (21), we see that Hypercleaning recovers all edges $e = (X, Y)$ with the following quartet error bound:

$$\begin{aligned} & \sigma((X, Y), Q) && (22) \\ & = \frac{4 \mid Q(X, Y) - Q \mid}{\mid X \mid (\mid X \mid -1) \mid Y \mid (\mid Y \mid -1)} && (23) \\ & < \frac{2m}{\mid X \mid\mid Y \mid} \text{ such that} && (24) \\ & \mid Q(X, Y) - Q \mid < m\frac{(\mid X \mid -1)(\mid Y \mid -1)}{2} && (25) \end{aligned}$$

Thus HyperCleaning returns all edges whose number of quartet errors wrt $Q$ is less than the bound in Equation(25). Consequently, for $m = 1$, HyperCleaning recovers all edges whose quartet error is less than $\frac{|X|-1|Y|-1}{2}$. Note that this cutoff is precisely the amount of quartet errors allowed on the edges of the true phylogeny wrt $Q$, such that solving the MQE problem on $Q$ will indeed recover the real edges of $T$. Thus if indeed the edges of the underlying true phylogeny $T$ has fewer than the specified amount of quartet (errors) discrepancies wrt $Q$, then the set $Best(m = 1, Q)$ returned by Hypercleaning will contain the edges of the true phylogeny.

However if the inferred quartet set $Q$ is not an accurate estimate of $Q_T$, then the pre-condition specified in Theorem 1 might not be met. Thus there may be edges in the underlying $T$ whose number of quartet errors may be larger than $\frac{(|X|-1)(|Y|-1)}{2}$ with respect to the inferred quartet set $Q$. In this case the set $Best(m = 1, Q)$ returned by HyperCleaning will not contain all the edges of $T$. But by increasing the parameter $m$, HyperCleaning has the following guarantee:

**Theorem 2**

Given an inferred quartet set $Q$, HyperCleaning on input parameter $m$ will correctly recover all edges of the underlying phylogeny $e_T = (X, Y) \in T$ whose quartet errors wrt $Q$ is bounded by

$$Q(X,Y) - Q < \frac{m(\mid X \mid -1)(\mid Y \mid -1)}{2}$$

consult [3] for details. $\odot$

This makes HyperCleaning a powerful tool for recovering the edges of the underlying phylogeny $T$.

On input $m$, the HyperCleaning algorithm runs in bounded polynomial time on the order of

$$O(n^5 f(2m) + n^7 f(m)) \tag{26}$$

where $f(x) = 4m^2(1 + 2m)^{4m}$ [3]. This theoretical upper bound is however very loose, and in practice (e.g. in the implementation given in [3]), the actual running time is much lower.

One interesting property of the HyperCleaning algorithm is that its running time depends on the amount of error in the inferred quartets $Q$ from input $S$. This is not observed in other (current) phylogenetic methods. Although no formal analysis has been performed, the following intuitive argument illustrates.

The running time of HyperCleaning is dictated by 2 parameters, the size of the input dataset $n$, and the value $m$. From Theorem 2, we see that $m$ basically controls the threshold on the amount of error (as defined in Equation (19)), an edge $e$ can have with respect to $Q$ and still be recovered. The question then becomes, given a certain $Q$, how large must we set $m$ such that the set $Best(m, Q)$ returned by HyperCleaning contains all or most of the edges of the underlying phylogeny $T$. Clearly if $Q$ is very strongly correlated with $Q_T$ then a small $m$ value suffices. In one extreme if $Q = Q_T$ (i.e. the inferred quartet set $Q$ contains no errors), then an arbitrary small $m \to 0$ is sufficient since HyperCleaning (see Theorem 2) will return all edges with zero quartet errors with respect to $Q$, which are exactly those edges of $T$.

Consequently the accuracy of the quartet set $Q$ with respect to $Q_T$ dictates how large we need to set $m$ and thus affects the running time of HyperCleaning. Realistically, under input data set of moderate difficulty, the inferred quartet set $Q$ will contain quartet errors wrt $Q_T$. Our simulation shows that under randomly sampled dataset $S$, in the form of aligned sequences from the Ribosomal Database Project (RDP) [35] the inferred quartet set $Q$ on $S$ has on average $15\% - 30\%$ quartet error wrt $Q_T$. This along with Theorem-2 implies that in order for HyperCleaning to recover all the underlying edges of $T$, we must raise the value of $m$ accordingly, which in turn increases the running time.

Consider a reasonable situation where we are given a dataset with more than 50 leaves. Moreover this dataset (say in the form of aligned sequences) imposes sufficiently many quartet errors on $Q$ with respect to $Q_T$, as to require $m = 20$ for HyperCleaning to recover all edges of $T$. Running HyperCleaning on $n \geq 50$ and $m \geq 20$ renders the running time computationally intractable (in the realistic running time and not theoretical sense).

Note that any method for improving the accuracy of the inferred quartet set $Q$ with respect $Q_T$ not only improves the accuracy of the estimate as in any other quartet methods, but will also improve HyperCleaning's running time.

In chapter 3 we introduce a major component of this thesis, a space/time tradeoff mechanism which at the expense of increasing memory, will significantly reduce the running time as to make HyperCleaning tractable on large datasets.

## 2.6 Effectiveness Assessment of Phylogenetic Methods

The effectiveness of a phylogenetic inference method (i.e. an estimator) can be measured by a combination of several factors. In the following section we describe these factors, which will be subsequently used in chapter 6 as effectiveness metrics for our comparative analysis.

The four important metrics for measuring effectiveness are: *consistency*, *robustness*, *accuracy*, and *scalability*. [4]

i) *Consistency* - an estimator is consistent if the probability of a correct approximation approaches 1 as the number of input data approaches infinity. In other words, it is desirable that an estimator satisfies the *monotonic improving* property that as the estimator receives more information, its estimation gets more accurate. A phylogenetic inference method is consistent if the probability that the 'inferred' phylogeny is the actual phylogeny approaches 1 as the amount of data per taxa (i.e. amount of data per leaf) increases. As stressed in [31], the two main problems with consistency is that

- Since we cannot run an estimator with its input data approaching infinity, it is impossible to empirically conclude if it is consistent or not. The only way to show consistency is through mathematical proofs which is feasible only with unrealistic simplifying assumptions, such as the assumption of *additivity* in the neighbour joining method.
- Even if an inference method is consistent, it is not sufficient to conclude that it is monotonic improving, which is what we are really after in an inference method [31], [26].

ii) *Robustness* - most phylogenetic inference methods have a set of assumptions under which they perform optimally. For example Neighbour Joining assumes that the distances given to it are additive, and Quartet Methods assumes that the inferred quartet set $Q$ on input set $S$ contains no or very little quartet errors. Robustness is concerned about the degradation of the accuracy of the inference method when its assumptions are violated.

iii) *Accuracy* - the accuracy of an inference method tries to categorize the

[4]Note that although the effectiveness measure is not governed by a method's efficiency, in practise the computational efficiency is a major factor governing the usefulness of a method.

amount of ‘deviation’ of the inferred tree from the actual tree. Many metrics have been proposed for accuracy, such as the number of edges not shared [44], [43] , or the number of cut-paste operations required to turn the inferred tree into the true phylogeny.

Another notion of accuracy is that of *relative accuracy* which can be measured as the ratio between the number of correct edges shared between an estimate tree and the real phylogeny and the total number of edges recovered in an estimate tree. This measure is useful for comparing two estimators where one or both produces estimates that are not fully resolved trees.

iv) *Scalability* - this deals with the important issue of the accuracy of the estimator as the size of the input problem grows. [31] supports the doctrine that estimators tend to decrease in accuracy as the size of the tree grows. The notion of scalability directly affects accuracy and we are mostly interested in the ‘rate’ at which accuracy or relative accuracy is affected (degrades) as the size of the tree grows.

# 3 HyperCleaning$^*$

In this section, we provide an overview of the HyperCleaning$^*$ algorithm and how it extends the HyperCleaning algorithm. Then we focus on the first extension of the algorithm - incorporating weighted quartets. We will formally introduce the notion of a weighted quartet set, and propose an algorithm along with the necessary definitions for recovering the best edges with respect to the weighted quartet set, with the accuracy guarantees similar to those given by HyperCleaning on an unweighted quartet set. Finally we formally introduce the concept of collapsing an unresolved phylogeny and the subsequent update of the quartet set to reduce the size of the input but without loss of information. This enables us to formulate a collapsing mechanism to decrease the running time of our algorithm. The details of incorporating collapsing into the algorithm presented here will be covered in chapter 4.

## 3.1 Overview of HyperCleaning$^*$

HyperCleaning$^*$ (HC*) is an extension of the HyperCleaning algorithm (HC). Similar to HC, HC* recovers all edges below a certain error threshold with respect to the inferred quartet set $Q$. HC* extends HC in two aspects:

**i)** HC$^*$ introduces a more general notion of a quartet set, called a weighted quartet set (or witness set). Consequently the definition of an edge score $\sigma(e, Q)$, the error threshold on returned edges as in Equation 64, and the algorithm for efficiently constructing the set $Best(m, Q)$ must be modified in light of the new weighted quartet set $W$. The motivation is in line with what the Short Quartet Method [15] hints at, that some quartets are more informative about the underlying phylogeny than others, and should have more weight in the re-combination stage. Moreover given a single quartet, we should also allow for non-zero weights, representing confidence, to all three of its topologies. Weighted quartets provide a context for facilitating these notions.

**ii)** Introducing a ‘collapsing’ mechanism that enables a speedup in running time of recovering edges, at the expense of more memory. Recall that HC on low values of $m$ might not return enough edges in $Best(m, Q)$ for deriving a fully resolved tree. But on high values of $m$, HC might be too time consuming given its high degree polynomial running time as given in Equation (26). Under the new collapse mechanism, we could run HC* on a low value of $m$, and take only the top scoring edges. This induces an unresolved tree, also called a *cluster tree*. Figure(12) shows a cluster tree and its induced clusters.

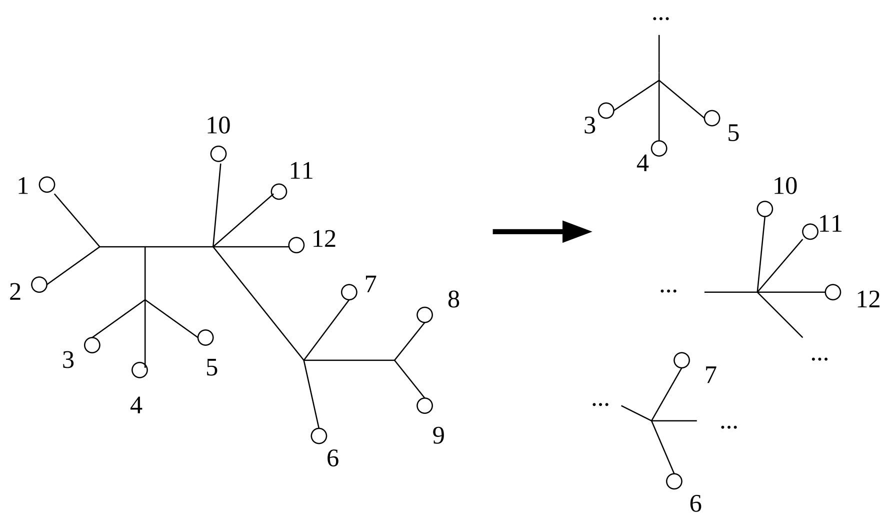


Figure 12: An unresolved cluster tree on $S$ with 12 leaves, and 3 of its clusters.

**Notion -** *Cluster Tree : Given set $S$ of $n$ leaves, an unrooted resolved binary tree on $S$ has $n - 3$ internal edges (i.e. edge not incident to a leaf). Any set of $l < n - 3$ compatible edges on $S$ induces a cluster tree. A cluster in a cluster tree consists of an internal vertex with its adjacent leaves.*

Note that this notion of a cluster tree which is only used to motivate the following discussion. In the next section, we formally define cluster tree in the context of HC*'s collapsing mechanism. In essence, the collapsing mechanism enables HC* to run at a low value of $m$, returning the set $Best(m, W)$ of edges which will induce an unresolved tree, or a cluster tree. The collapsing mechanism will then collapse all but one of the clusters in this unresolved tree. The remaining single cluster can be viewed as simply a star topology on a set of leaves and collapsed nodes. Moreover this cluster set should be much smaller than the original input set $S$. We can then run HC* on this smaller cluster set, and attempt to resolve edges that lie in that cluster by using a higher value of $m$. In other words, we reduce the size of input size $n$, such that we can raise the size of input parameter $m$, allowing more edges to be resolved without accruing the high cost of added computational time. Figure (13) shows the idea. The collapsing mechanism guarantees no information loss during the collapse, such that any edges recovered by HC$^*$ under a collapsed cluster, will have the same score with respect to $W$ as under no collapsing with the original input set $S$, by raising the value of $m$ to the necessary level.

In the next two sections, we present the Hypercleaning* algorithm.

## 3.2 HC* using Weighted Quartets

HC* adopts a more general notion of a quartet set, and subsequently generalized notions of quartet errors/support of edges with respect to the quartet set. From here in we use the definition of an edge as a bipartition (see Def 5 on page 24). HC* uses the notion of a weighted quartet set on input dataset $S$:

**Definition -** *Weighted Quartet Set (Witness Set)*

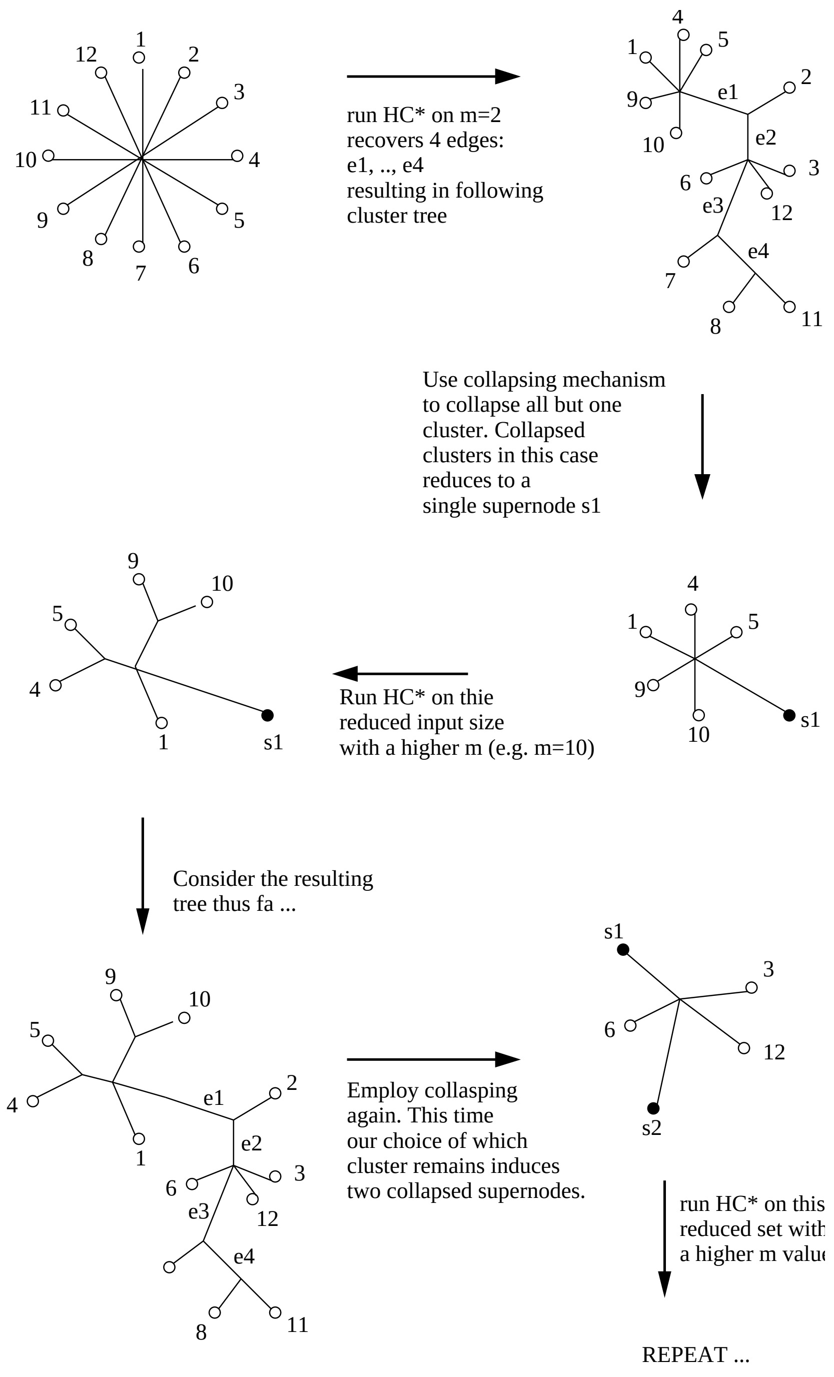


Figure 13: Running HC* with low values of $m$ and further resolution of the unresolved tree by recursively collapsing the problem into smaller instances and then running HC* with higher values of $m$ on these smaller problems.

A weighted quartet set $W$ on input $S$ consists of the following set:

$$\begin{aligned} W = \{ \qquad & (ab \mid cd, w(ab \mid cd)), \\ & (ac \mid bd, w(ac \mid bd)), \\ & (ad \mid bc, w(ad \mid bc))) :: \text{ for all quartets } a, b, c, d \in S\}, \end{aligned}$$

where $w(ab \mid cd) \geq 0$ is a weighting that W assigns to the quartet topology $ab \mid cd$ being induced in the underlying phylogeny $T$. One restriction on the weights is that they be non-negative. If the quartet weights are arrived at using probabilistic methods, where the weights represents confidence, then another restriction might be that the weights of the three topologies on a given quartet sum to 1. HC$^*$ however does not require such a restriction to hold. $\odot$

The weighted quartet set $W$ can be viewed as a generalized version of $Q$, where in $Q$ each quartet $a, b, c, d$ has exactly one topology with weight 1. The purpose of $W$ is to allow degrees of confidence to be assigned to quartet topologies. Note that the cardinality of $W$ on input set $S$ is $3\binom{|S|}{4}$.

We defer the actual construction of the witness set $W$ from the given input data until section 5.

Under this more general notion of quartet set, we must also generalize the definition of a support and its dual, the distance, of an edge with respect to $W$.

**Definition -** *distance function of an edge* $e = (X, Y)$
Given an edge $e = (X, Y)$, its **quartet error** with respect to $W$, is given by

$$\sigma(e = (X,Y), W) = \frac{\sum_{ab|cd \in Q(X,Y)} w(ac \mid bd) + w(ad \mid bc)}{\binom{|X|}{2}\binom{|Y|}{2}} \tag{27}$$

where $Q(X, Y)$ is the set of quartet topologies induced by $e$. $\odot$

HC* on input set $S$ and parameter $m$ returns the set denoted $Best(m, W)$ of all edges on $S$ whose distance with respect to $W$ is less than $\frac{2m}{|X||Y|}$. Specifically, HC* returns

$$Best(m, W) = \{(X, Y) :: \sigma((X, Y), W) < \frac{2m}{\mid X \mid\mid Y \mid}\} \tag{28}$$

Consequently from (27), (28) yields the following lemma:

**Lemma 1**

Given input $S$ and a corresponding inferred weighted quartet set $W$, HC* on input $m$, will recover an edge set containing all edges $e = (X, Y)$ in the underlying phylogeny, provided the following error bound it met:

$$\sigma(e, W) < m\frac{\mid X - 1 \mid\mid Y - 1 \mid}{2}$$

$\odot$

### 3.2.1 The Algorithm

HC* on input $S$, $m$ produces the edge set $Best(m, W)$ satisfying Equation (28) by first producing for every two leaves $x, y \in S$, the set:

$$Best_{xy}(m, W) = \{ \quad e = (X, Y) \text{ such that } x \in X, y \in Y, \tag{29}$$

$$\sum_{ax|by \in W(X,Y)} w(ay \mid bx) + w(ab \mid xy) < m\} \tag{30}$$

This set $Best_{xy}$ in essence consists of edges with leaves $x, y$ in its left and right partites respectively, whose set of induced quartets $Q(X, Y)$ has a quartet error score bounded by $m$ with respect to $W$. The construction of this set is recursive and is defined in the following theorem. Moreover the set $Best_{xy}(m, W)$ will also allow trivial edge $(\{x\}, \{y\})$. Note that this trivial edge induce no quartets.

**Procedure 1 -** Constructing $Best_{xy}(m, W_k)$

Given leaves $x, y \in S$, and $1 \leq k \leq n$. The following recurrence relationship defines the set $Best_{xy}(m, W_k)$ with respect to the set $Best_{xy}(m, W_{k-1})$, where we denote $Best_{xy}(m, W) = Best_{xy}(m, W_n)$. $W_k$ are those quartets of $W$ induced by the leaves in the sequence $S_k = \{x, y, v_1, v_2, .., v_{k-2}\} \subseteq S$.

If $k = 1$ then $Best_{xy}(m, W_1) = \emptyset$.
If $k = 2$ then

$$\forall x, y \in S, Best_{xy}(m, W_2) = \{ (\{x\}, \{y\}) \}$$

For $k \geq 3$

$$Best_{xy}(m, W_k) = \text{ all edges } e \in L_{xy} \cup R_{xy} \text{ satisfying Equation (30) where } W = W_k \tag{31}$$

where the construction of $L_{xy}, R_{xy}$ are as follows:

$$L_{xy} = \{(X \cup \{s_k\}, Y) \mid (X, Y) \in Best_{xy}(m, W_{k-1})\}$$
$$R_{xy} = \{(X, Y \cup \{s_k\}) \mid (X, Y) \in Best_{xy}(m, W_{k-1})\}$$

and leaf $\{s_k\}$ is the $k$th leaf drawn from the sequence $S_k = (x, y, v_1, v_2, ...v_{k-2}) \subseteq S$. $\odot$

**Theorem 3 -** Given input set $S$ and weighted quartet set $W$, the set returned by Procedure 1, i.e. $Best_{xy}(m, W_k)$ for $k = n$, defines $Best_{xy}(m, W)$ as given in (30).

*proof* - Consider an arbitrary edge $e = (X \cup \{s_k\}, Y) \in Best(m, W_k)$ (similar arguments can be made for the symmetric case $(X, Y \cup \{s_k\})$ ). We have three possible cases:

*case i)* Non-trivial case, where both $\mid X \cup \{s_k\} \mid, \mid Y \mid \geq 1$. For this case let us define

$$Err_{with} = \sum_{s_k x | y' y \in Q(X \cup \{s_k\}, Y)} w(s_k y' \mid xy) + w(xy' \mid s_k y) \tag{32}$$

$$Err_{without} = \sum_{x' \neq s_k, x \neq s_k | y' \neq y \in Q(X \cup s_k, Y)} w(x'y \mid xy') + w(x'y' \mid xy) \quad (33)$$

Since $(X \cup \{s_k\}, Y) \in Best(m, W_k)$, Equation (30) implies

$$Err_{with} + Err_{without} < m, \text{ therefore,} \quad (34)$$

$$Err_{without} < m \quad (35)$$

which implies that

$$(X, Y) \in Best_{xy}(m, W_{k-1})$$

as required.

*case ii)* Trivial case where $\mid X \mid = \emptyset$. In this case $e = (X \cup \{s_k\}, Y) = (\{s_k\}, Y)$. By definition, $e \in Best_{xy}(m, W_{k-1})$ since there are no quartets running across $e$.

*case iii)* Trivial case where $\mid Y \mid = \emptyset$. By definition $e \in Best_{xy}(m, W_{k-1})$. $\odot$

The ultimate goal of HC* is to return the set of edges $Best(m, W)$ as described in Equation (28). The goal of the sets $Best_{xy}(m, W_k)$ for all $x, y \in S$ and $1 \leq k \leq n$ defined above is to facilitate an efficient bottom up construction of the set $Best(m, W)$. This is achieved by the following recursive construction.

**Procedure 2 -** Constructing the set $Best(m, W_k)$

On input set $S$ of $n$ leaves, the set $Best(m, W_k)$, is defined recursively from $Best(m, W_{k-1})$, for $2 \leq k \leq n$, where $Best(m, W) = Best(m, W_n)$. $W_k$ is the subset of quartets of $W$ induced by the subset of leaves $S_k = \{v_1, v_2, ..., v_k\} \subseteq S$.

If $k = 1$ then $Best(m, W_1) = \emptyset$.
If $k \geq 2$ then

$Best(m, W_k) =$ all edges $e \in L \cup R \cup M$satisfying Equation(21) where $Q = W_k$

where the sets $L, R, M$ are constructed as follows:

$$L = \{(X \cup \{s_k\}, Y) :: (X, Y) \in Best(m, W_{k-1})\}$$
$$R = \{(X, Y \cup \{s_k\}) :: (X, Y) \in Best(m, W_{k-1})\}$$
$$M = \bigcup_{x \in S_{k-1}} Best_{xs_k}(m, W_k)$$

**Theorem 4 -** Given input $S$ and $W$, the resulting set $Best(m, W_k)$ for $k = n$ defines the set $Best(m, W)$ as given by Equation (28).

*proof* - Consider an arbitrary edge $e = (X \cup \{s_k\}, Y)$

$$(X \cup \{s_k\}, Y) \in Best(m, W_k) \tag{36}$$

If $(X, Y) \in Best(W_{k-1}, m)$ then we are done. Otherwise we must show that $(X, Y) \in M$.

Since $(X \cup \{s_k\}, Y) \in Best(m, W_k)$ we have

$$\sigma((X \cup \{s_k\}, Y), W_k) < \frac{2m}{\mid X \cup \{s_k\} \mid \cdot \mid Y \mid} \tag{37}$$

Since $(X, Y) \notin Best(m, W_{k-1})$ then we have

$$\sigma(W_{k-1}, (X, Y)) \geq \frac{2m}{\mid X \mid \cdot \mid Y \mid} \tag{38}$$

Now we define the term $s_{in}$, denoting the sum of errors (distances) of all quartets across $((X \cup \{s_k\}), Y)$ involving $s_k$. Also let us denote $s_{out}$ as the sum of errors of all quartets across $((X \cup \{s_k\}), Y)$ not involving $s_k$.

$$s_{in} = \sum_{as_k|cd \in Q(X \cup s_k, Y)} w(ac|s_k d) + w(ad|s_k c) \tag{39}$$

$$s_{out} = \sum_{a \neq s_k, b \neq s_k|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc) \tag{40}$$

From our assumption (37), we have the following

$$s_{in} + s_{out} < \frac{m(\mid X \cup \{s_k\} \mid -1) \cdot (\mid Y - 1 \mid)}{2} \tag{41}$$

But from assumption (38), we have

$$s_{out} \geq \frac{m(\mid X \mid -1) \cdot (\mid Y \mid -1)}{2} \tag{42}$$

From (41), (42), we conclude that

$$s_{in} < \frac{m \mid \{s_k\} \mid \cdot (\mid Y \mid -1)}{2} \tag{43}$$

Consider the sum of quartet errors across edge $(X \cup \{s_k\}, Y)$ involving both $y \in Y$ and $s_k$, denoted $error(y, s_k)$:

$$error(y, s_k) = \sum_{xs_k \mid yy' \in Q(X \cup \{s_k\}, Y)} w(s_k y \mid xy') + w(s_k y' \mid xy) \tag{44}$$

Let us assume that $error(y, s_k) \geq m,$ for all $y \in Y$ which implies that

$$\sum_{xs_k \mid yy' \in Q(X \cup \{s_k\}, Y)} w(s_k y \mid xy') + w(s_k y' \mid xy) \geq m \tag{45}$$

Consequently, we make the following observation

$$\sum_{y \in Y} \left( \sum_{xs_k \mid yy' \in Q(X \cup \{s_k\}, Y)} w(s_k y \mid xy') + w(s_k y' \mid xy) \right) \geq m(\mid Y \mid) \tag{46}$$

Since for non-trivial cases, $\mid Y \mid \geq 2$, we make note of the following

$$2s_{in} \geq \sum_{y \in Y} \left( \sum_{xs_k \mid yy' \in Q(X \cup \{s_k\}, Y)} w(s_k y \mid xy') + w(s_k y' \mid xy) \right) \tag{47}$$

Thus, (43), (46), and (47) gives us

$$\begin{aligned} & 2\frac{m \mid \{s_k\} \mid \cdot \mid Y - 1 \mid}{2} \\ & > 2s_{in} \\ & \geq \sum_{y \in Y} \left( \sum_{xs_k \mid yy' \in Q(X \cup \{s_k\}, Y)} w(s_k y \mid xy') + w(s_k y' \mid xy) \right) \\ & \geq m \mid Y \mid \end{aligned}$$

The above says that $m \mid \{s_k\} \mid \cdot \mid Y-1 \mid = m \mid Y-1 \mid > m \mid Y \mid$, such that we have contradiction $\mid Y-1 \mid > \mid Y \mid$. Thus our assumption that $error(y, s_k) > m$ for all $y \in Y$ must be false. Specifically we can find at least one $y \in Y$ such that

$$error(y, s_k) < m$$

Thus the above along with (30) and (44) says that

$$(X \cup \{s_k\}, Y) \in Best_{s_k y}(m, W_k)$$

Thus $(X \cup \{s_k\}, Y) \subseteq M$ as required. $\odot$

Procedures 1 and 2 in essence describe the recursive construction of the set $Best(m, W)$. The efficiency is that at each stage for $1 \leq k \leq n$, the cardinality of the set $Best(m, W)_k$ is inherently constrained by the size of the sets $Best(m, W_{k-1})$ and $Best_{xy}(m, W_k)$, with the guarantee that $Best(m, W)$ does not miss any edges. This in essence constrains the edge space to be searched through, whereas the naive search on $n$ leaves involves searching through $2^n$ edges.

### 3.2.2 Cluster Tree, Collapsing and HC*

In the beginning of this chapter we intuitively motivated the reason for introducing the collapsing mechanism. Figure 13 shows an example of how collapsing can reduce existing problem into several instances of smaller problems. In this section we provide some formal definitions and the goal behind collapsing. The next chapter will examine the collapsing mechanism in detail, and its affect on how HC* recovers edges, as defined by Theorems 3 and 4. The goal of collapsing is to enable HC* to run on a small value of $m$ which is both fast and memory efficient, since $m$ directly effects the amount of quartet errors permitted and in turn the size of the sets $Best(m, W_k), Best_{xy}(m, W_k), 1 \leq k \leq n$. On small values of $m$ however, the set $Best(m, W)$ returned by HC$^*$ might not contain $n-3$ compatible edges to yield a tree estimate, inducing a cluster tree. The goal is to iteratively collapse the cluster tree into instances of a single cluster (i.e. a star tree) on a smaller set of vertices. Then we can run HC$^*$ on these star trees with higher values of $m$ to recover the unresolved edges.

Let us now formally define some terminologies required for introducing the collapse mechanism. The following three definitions are mutually referencing.

**Definition -** *Cluster Tree*

A cluster tree $T$ is a tree consisting of edges and vertices, where each vertex is either a leaf, an internal vertex (i.e. non-leaf), or a supernode. Note that a regular tree is a cluster tree whose vertices are either leaves or internal vertices. $\odot$

**Definition -** *Cluster*

Given a cluster tree $T$, a cluster $C_i \in T$ is the set of vertices consisting of exactly one internal vertex, along with its neighbouring leaves and supernodes. The leaves of a cluster $C_i$ are denoted $Leaves(C_i)$, and the supernodes (see following definition) in $C_i$ are denoted $SuperNodes(C_i)$. Thus we have $C_i =$ an internal node$\cup SuperNodes(C_i) \cup Leaves(C_i)$. An internal vertex with no adjacent leaves or supernodes is a *trivial cluster*. Let $Clu(T)$ be the set of all clusters in $T$. $\odot$

**Definition -** *Supernode*

A supernode $\mathbf{c}$ of a cluster tree $T$, is a degree-1 vertex, denoting some collapsed cluster $C_i$. $\odot$

**Definition -** *Cluster Collapse*

Given a cluster tree $T$ and cluster $C_i \in T$, the cluster can be collapsed into a single node $\mathbf{c_i}$, only if the resulting node $\mathbf{c_i}$ is a degree-1 vertex in the remaining

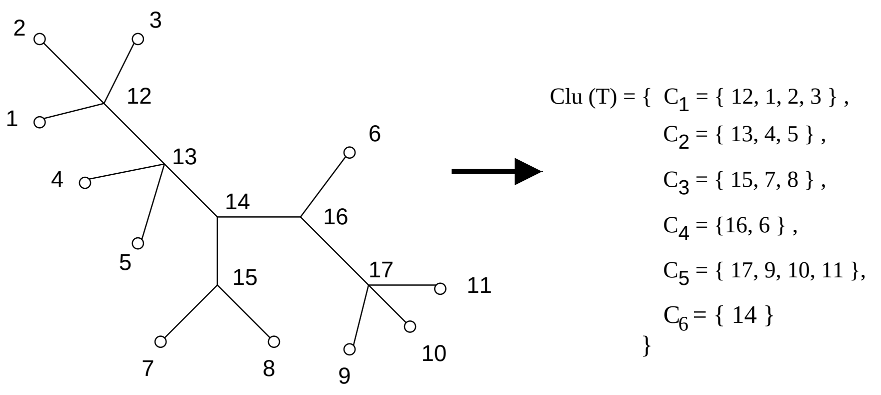


Figure 14: An unresolved cluster tree, with clusters $C_1, .., C_6$. $C_6$ is a trivial cluster

tree. After the collapse, the tree remains a connected tree, but has one more supernode and one less cluster. $\odot$

Figure (15) illustrates a valid and an invalid cluster collapse.

**Definition -** *Full Collapse of a tree : Full* $\bullet$ $(T \mid C_i)$
Given a (cluster) tree $T$, its full collapse with respect to cluster $C_i$, denoted *Full* $\bullet$ $(T \mid C_i)$ is the ordering of all the clusters $C_j \neq C_i \in Clu(T)$, and the subsequent collapse in turn of these ordered clusters. The ordering must be valid, in the sense that each cluster, on its turn to collapse, must be collapsible as in previous definition, wrt $T$. $\odot$

The result of the sequence of collapses is single cluster $C_i^* = C_i \cup \{\mathbf{s_1}, ..., \mathbf{s_k}\}$, $k <\mid S \mid$, and the corresponding tree topology on $C_i^*$ is simply the star topology.

We will now show that it is always possible to correctly collapse the clusters of a cluster tree until only one (pre-determined) cluster remains.

**Lemma -** *Generating a valid collapse ordering for Full* $\bullet$ $(T \mid C_i)$

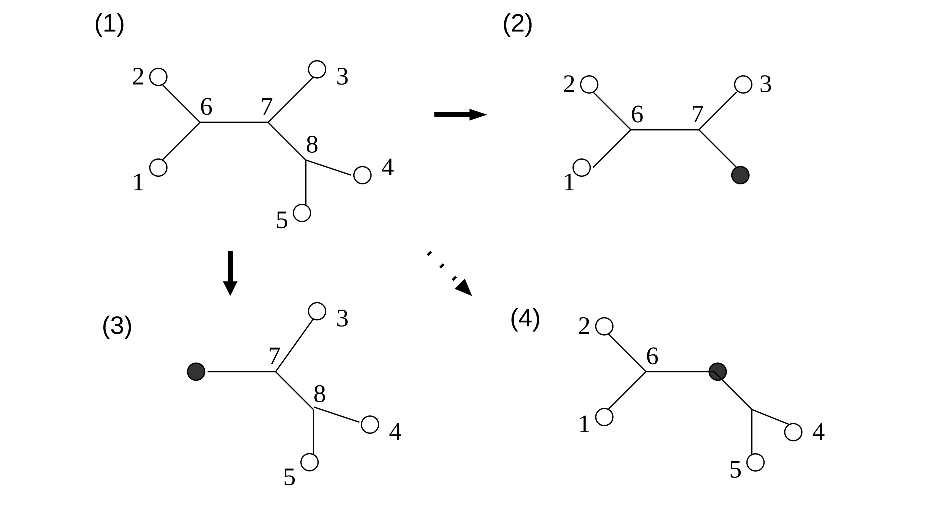


Figure 15: The cluster tree $T$ in (1) has 3 clusters: $C_6 = \{6, 1, 2\}$, $C_7 = \{7, 3\}$, and $C_8 = \{8, 4, 5\}$. (2) denotes the valid collapse of cluster $C_8$ since the resulting vertex satisfies the definition of a supernode. (3) shows the valid collapse of cluster $C_6$. In (4), the collapse of cluster $C_7$ is not valid since the resulting vertex does not satisfy the definition of a supernode.

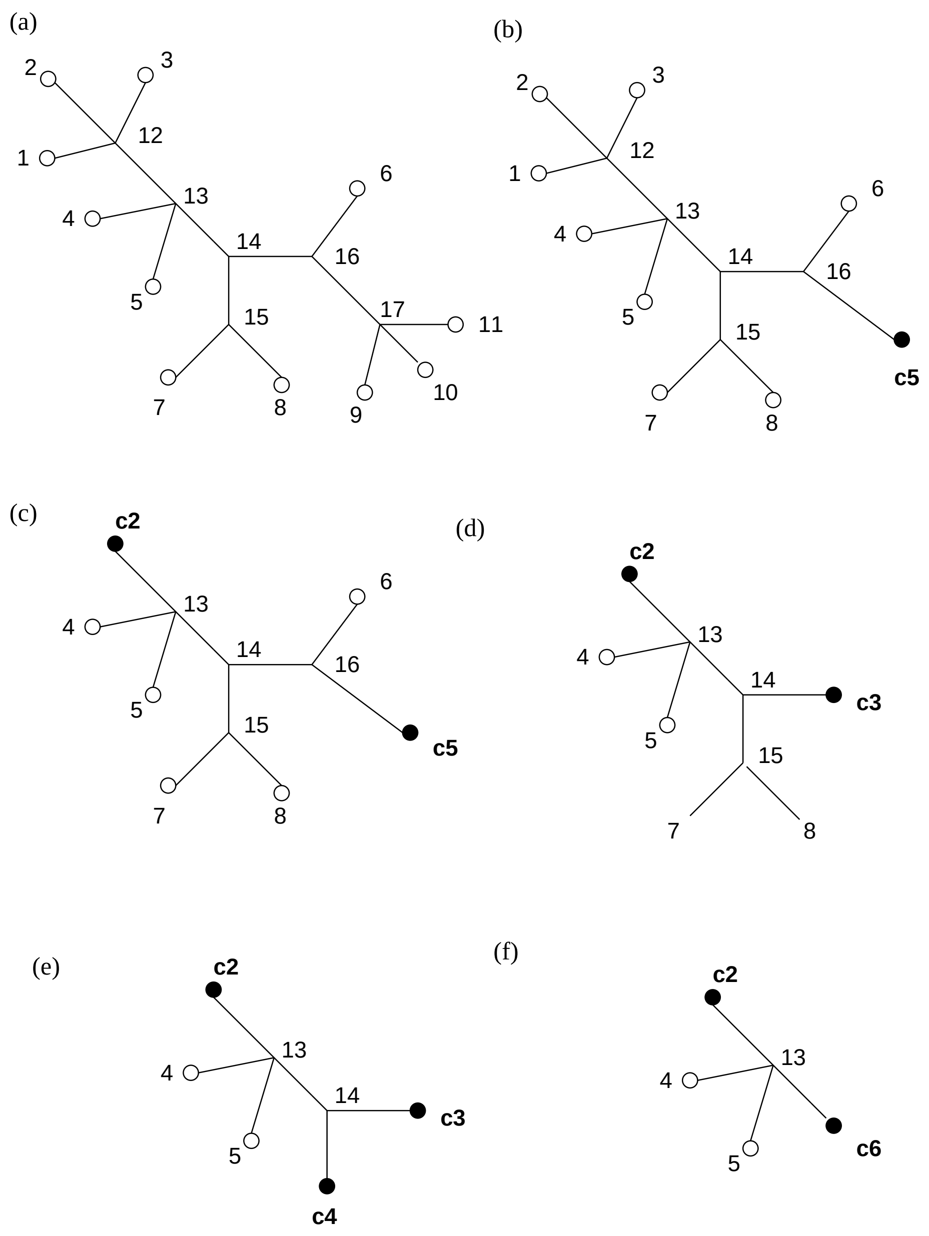


Figure 16: The collapsing of cluster tree $T$ given in (Fig 14) into a single cluster, denoted by $Full \bullet (T \mid C_2)$. A valid collapse ordering is $(C_5, C_1, C_3, C_6, C_4)$. (a) is the original tree, (b) collapses cluster $C_5$ into supernode **c5**. (c) collapses cluster $C_1$ into supernode **s2**. (d) collapses cluster $C_3$ into supernode **c3**. (e) collapses cluster $C_4$ into supernode **c4**. (f) cluster $C_6$, which now has 2 adjacent supernodes as a result of previous rounds of collapsing, is itself collapsed into supernode **c6**. The result is a cluster tree consisting of a single cluster (4,5 **c2**, **c6**), of size 4.

Given tree $T$ with $k$ clusters, a valid ordering of the clusters can be obtained by sorting all clusters $\neq C_i$ by decreasing distance from $C_i$, where the $dist(C, D)$ between two clusters is simply the number of edges between their closest vertices. Ties in the ordering can be settled arbitrarily.

*proof*$-$Suppose an invalid collapse occurs at some given point during a full collapse of some cluster tree: $Full \bullet (T \mid C_i)$, say during collapsing cluster $C_k \neq C_i$. Specifically, assume that collapsing cluster $C_k$ yields a vertex $c_k$ of degree $> 1$ (i.e. a supernode). This means that cluster $C_k$ had two or more distinct adjacent clusters $C_{p_1}, C_{p_2}, ..., C_{p_d}$ prior to its collapse. Since the cluster tree $T$ does not have a cycle, exactly one of these clusters, say $C_{p_1}$ is closer to cluster $C_i$ than $C_k$, and all other adjacent clusters $C_{p_2}, ..., C_{p_d}$ are one edge further from $C_i$ than $C_k$. But these adjacent clusters under the above collapsing scheme should already have been themselves collapsed into supernodes. Thus we have a contradiction, as required..

$\odot$

Figure (16) shows a collapsing of a cluster tree into a single cluster using a valid collapse ordering.

**Definition -** *Size and Cardinality of Vertices*

Given a vertex $s$, we define its size as

$$Size(s) = \begin{cases} 1 & \text{if } s \text{ is a leaf or internal vertex} \\ \sum_{v \in C_s} Size(v) & \text{if } s \text{ is a supernode, and } C_s \\ & \text{is the cluster collapsed into } s \end{cases}$$

The cardinality of a vertex $s$, denoted $\mid s \mid$ is 1 if $s$ is a leaf or an internal vertex, and the number of elements in its corresponding cluster if $s$ is a supernode. $\odot$

Given a supernode **s** denoting some collapsed cluster $C_s$, we can also define its expansion, as follows:

**Definition -** *Expanding a supernode: exp(**s**)*

Consider a supernode **s** resulted from collapsing a cluster $C_s$. The expansion of **s**, denoted $exp(\mathbf{s})$ is a recursive procedure, which returns a set of leaves as follows:

$$exp(\mathbf{s}) = \left\{ Leaves(C_s) \bigcup_{\forall s' \in SuperNode(C_s)} exp(s') \right\}$$

$\odot$

In the context of collapsing, an edge $e = (X, Y)$ defined by its bipartition of vertices also becomes more general. In a normal tree $T$, an edge $e = (X, Y)$ consists of two disjoint but pairwise complete sets on the leaves $S$. In a cluster tree $T_C$, an edge $e' = (X', Y')$ also consists of two disjoint, pairwise complete sets on all degree-1 vertices in $T_C$. In other words the bipartites of an edge are the leaves and supernodes in $T_C$. Given an edge $e = (X, Y)$ in a cluster tree $T_C$, consider the following definition:

**Definition -** *Expansion of an edge $e = (X, Y)$ in a cluster tree*

Given an edge $e = (X, Y)$ in cluster tree $T_C$, its full expansion, denoted $Exp(e = (X, Y))$ returns the edge $e' = (X', Y')$ where $X', Y'$ is a bipartition on the original leaves set $S$. Formally, the $Exp(e = (X, Y))$ is recursively defined as follows:

$$\begin{aligned}
& Exp(e = (X, Y)) \\
& = (X', Y') \\
& = \Bigg( Leaves(X) \bigcup_{s \in SuperNode(X)} exp(s)\,, \\
& \qquad\qquad\qquad Leaves(Y) \bigcup_{s \in SuperNode(Y)} exp(s) \Bigg)
\end{aligned}$$

Having the above definitions on a cluster tree, collapsing, and expanding under a cluster tree, we are ready to formally state the definition of what it means for

HC* to achieve information loss-less collapsing of the cluster tree into a single cluster, and subsequent resolution of edges on that cluster.

**Definition -** *HC* Information loss-less edge recovering under collapsing*
Given input set of leaves $S$ for which we wish to infer the phylogeny, and inferred quartet set $W$, assume that HC* is at some stage of collapsing some semi-resolved cluster tree into a star topology tree, whose leaves are on the cluster set:

$$C = \{l_1, l_2, ..l_k, \mathbf{s1}, .., \mathbf{sp}\} \tag{48}$$

consisting of leaves and supernodes. We wish to construct the weighted quartet set $W^*$ on the vertices of $C$, such that running HC* on input $C$ using $W^*$, will return the edge set $Best(m, W^*)$ with the following accuracy guarantee:

$$\forall \text{ edges } e \in Best(m, W^*) :: \sigma(e, W^*) = \sigma(e^* = Exp(e), W) \tag{49}$$

Such a weighted quartet set $W^*$ is the precondition for information lossless resolution of the cluster $C$. $\odot$

### 3.2.3 The HC* algorithm under collapsing

Consider an input cluster set $C$ consisting of vertices (i.e. leaves and supernodes), and a weighted quartet set $W$ on these vertices. [5] The objective is then to modify the existing HC* algorithm in the previous chapter, such that running HC* on input set $C$ with some parameter $m$ will return the set of edges $Best(m, W)$, satisfying the accuracy bound given in Lemma-1. As we will see, it suffices to make modifications to the definitions ( given in Equations 27, 28) of quartet error bounds across an edge in sets $Best_{xy}(m, W_k)$, and $Best(m, W_k)$.

[5]The initial input set of leaves $S$ is a specific instance of the problem, where $S$ is simply a cluster of leaves. But in general the input cluster after some rounds of collapsing consists of leaves and/or supernodes.

The algorithms for constructing the sets as given in Theorems 3, 4 will extend to this general case.

### 3.2.4 Constructing $Best_{xy}(m, W)$

On input set of vertices $C$, parameter $m$, and the weighted quartet set $W$ on $C$, the set $Best_{xy}(m, W_k)$ is defined as

$$Best_{xy}(m, W_k) = \{(X, Y) :: \sum_{ax|by \in Q(X,Y)} \frac{w(ay|bx) + w(ab|xy)}{Size(x)Size(y)} < m\} \quad (50)$$

where $W_k$ is the subset of $W$ induced by the sequence of vertices. $S_k = \{x, y, v_1, v_2, ..., v_{k-2}\} \subseteq C$, and $Size(x)$ as given by definition in section 3.2.2. Moreover any trivial edges with either zero or one element in either of its two partites belongs to $Best_{xy}(m, W_k)$ The only modification of this definition is the normalizing constant, which now takes into account the 'actual' number of leaves in the partites, and not just the number of vertices. In particular we observe that $Size(x)Size(y) = (\mid X' \mid)(\mid Y' \mid)$, where $(X', Y') = Exp(e = (X, Y))$.

Note that under collapsing, some of the nodes might become supernodes, therefore a quartet involving one or more supernodes must have some appropriate weight definition. In short, this is accomplished by updating the quartet set $W$ and is a central issue with loss-less collapsing, which will be addressed in the next chapter.

Having modified our definition of the amount of quartet errors across an edge, the algorithm presented in Procedure 2 for constructing $Best_{xy}(m, W_k)$ now generalizes to the collapsing case. We state Theorem-3 again, and prove it under the new definition of edge error.

**Procedure 3 -** Constructing $Best_{xy}(m, W)$ under collapsing

If $k = 1$ then $Best_{xy}\{m, W_k\} = \emptyset$, since $W_1$ does not contain any bipartitions.

Else If $k = 2$, then $Best_{xy}\{m, W_k\} = \{(\{x\}, \{y\})\}$.

Else If $k \geq 3$, then

$$Best_{xy}(m, W_k) = \text{all edges } e \in L_{xy} \cup R_{xy} \text{ satisfying Equation (50)}$$

where

$$L_{xy} = \{(X \cup \{s_k\}, Y) \mid (X, Y) \in Best_{xy}(W_{k-1}, m)\}$$
$$R_{xy} = \{(X, Y \cup \{s_k\}) \mid (X, Y) \in Best_{xy}(W_{k-1}, m)\}$$

and the vertex $\{s_k\}$ is the kth element drawn from the sequence
$S_k = \{x, y, v_1, v_2, ..., v_{k-2}\} \subseteq C$. $\odot$

**Theorem 5 -** Given input $S$ and $W$, the set constructed under Procedure 3, i.e. $Best_{xy}(m, W_k)$, for $k = n$ satisfies the definition of $Best_{xy}(m, W)$ as given in definition (50) .

*proof*$-$ Consider a bipartition $e = (X \cup \{s_k\}, Y) \in Best_{xy}(m, W_k)$ ( similar arguments can be made for edge $e = (X, Y \cup \{s_k\}) \in Best(m, W_k)$ ).

We have three cases :

*case i)* Non-trivial case, where both $|X \cup \{s_k\}|, |Y| \geq 1$. In this case, $(X, Y)$ is non-trivial. Moreover let us define

$$Err_{with} = \sum_{s_k x | y'y \in Q(X \cup \{s_k\}, Y)} \frac{(w(s_k y'|xy) + w(xy'|s_k y))}{Size(x)Size(y)}$$
$$Err_{without} = \sum_{x' \neq s_k, x \neq s_k | y'y \in Q(X \cup \{s_k\}, Y)} \frac{(w(x'y|xy') + w(x'y'|xy))}{Size(x)Size(y)}$$

Such that by our assumption, since $(X \cup \{s_k\}, Y) \in Best_{xy}(m, W_k)$, then

$$Err_{with} + Err_{without} < m, \quad \text{such that} \tag{51}$$

$$Err_{without} < m \tag{52}$$

which implies that

$$(X, Y) \in Best_{xy}(m, W_{k-1})$$

as required.

*case ii)* Trivial case where $\mid X \mid = \emptyset$, such that the edge in question $e = (X, Y) = (\{s_k\}, Y)$. By our construction, $\{s_k\}$ must be the vertex $\{x\}$, such that $e = (\{x\}, Y) \in Best_{xy}(W_{k-1}, m)$. This can be seen by considering the initial edge $(\{x\}, \{y\}) \in Best_{xy}(m, W_2)$ for $k = 2$, and the iterative construction of the set $R$ as $k = 3, .., k-2$.

*case iii)* Trivial case where $\mid Y \mid = \emptyset$ is not possible for $k \geq 1$ due to our construction. $\odot$

### 3.2.5 Constructing $Best(m, W_k)$

The algorithm for constructing the set $Best(m, W_K)$ does not need to be modified, provided we make the following simple modification to the definition given in (28) on the scoring of an edge (i.e. the number of quartet errors across an edge).

The set $Best(m, W_k)$ is defined as follows

$$Best(m, W_k) = \{(X, Y) \mid \sigma(W_k, (X, Y)) < \frac{2m}{Size(X)Size(Y)}\} \tag{53}$$

where $W_k$ are those quartets in W induced by $S_k$.

**Procedure 4 -** Constructing $Best(m, W_k)$ under collapsing

If $k = 1$ then $Best(m, W_k) = \emptyset$, Else if $k \geq 2$, then

$$Best(m, W_k) = \text{all edges } e \in L \cup R \cup M \text{ satisfying Equation(53)}$$

where

$$L = \{(X \cup \{s_k\}, Y) \mid (X, Y) \in Best(m, W_{k-1})\}$$
$$R = \{(X, Y \cup \{s_k\}) \mid (X, Y) \in Best(m, W_{k-1})\}$$
$$M = \bigcup_{y \in S_{k-1}} Best_{ys_k}(m, W_k)$$

**Theorem 6 -** Given input $S, W$, the resulting set returned by Procedure 4, i.e. $Best(m, W_k)$ for $k = n$ defines the set $Best(m, W)$ as given in definition (53).

*proof*$-$ Assume we have a bipartition $(X \cup \{s_k\}, Y) \in Best(W_k, m)$ (similar arguments can be made for the symmetric case $(X, Y \cup \{s_k\})$ ). If $(X, Y) \in Best(W_{k-1}, m)$ then we are done. But if $(X, Y) \notin Best(W_{k-1}, m)$ then

We must show that $(X, Y) \in M$.

First we observe the following facts:

$$((X \cup \{s_k\}, Y) \in Best(W_k, m) \tag{54}$$
$$\Leftrightarrow \sigma(W_k, (X \cup \{s_k\}, Y)) < \frac{2m}{Size(X \cup \{s_k\})Size(Y)} \tag{55}$$
$$\Leftrightarrow \frac{\sum_{ab|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc)}{\binom{Size(X)}{2}\binom{Size(Y)}{2}} < \frac{2m}{Size(X \cup \{s_k\})Size(Y)} \tag{56}$$
$$\Leftrightarrow \frac{4\sum_{ab|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc)}{Size(X \cup \{s_k\})(Size(X \cup \{s_k\}) - 1)Size(Y)(Size(Y) - 1)} \tag{57}$$
$$< \frac{2m}{Size(X \cup \{s_k\})Size(Y)} \tag{58}$$
$$\Leftrightarrow \frac{2\sum_{ab|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc)}{(Size(X \cup \{s_k\}) - 1)(Size(Y) - 1)} < m \tag{59}$$

$$\Leftrightarrow \sum_{ab|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc) \tag{60}$$

$$< \frac{m(Size(X \cup \{s_k\}) - 1)(Size(Y) - 1)}{2} \tag{61}$$

it follows that

$$(X, Y) \notin Best(W_{k-1}, m) \tag{62}$$

$$\Leftrightarrow \sum_{ab|cd \in Q(X,Y)} w(ac|bd) + w(ad|bc) \tag{63}$$

$$\geq \frac{m(Size(X) - 1)(Size(Y) - 1)}{2} \tag{64}$$

Now we define the term $s_{in}$, denoting the sum of errors (distances) of all quartets across $((X \cup \{s_k\}), Y)$ involving $s_k$. Also let us denote $s_{out}$ as the sum of errors of all quartets across $((X \cup \{s_k\}), Y)$ not involving $s_k$.

$$s_{in} = \sum_{as_k|cd \in Q(X \cup s_k, Y)} w(ac|s_k d) + w(ad|s_k c) \tag{65}$$

$$s_{out} = \sum_{a \neq s_k, b \neq s_k|cd \in Q(X \cup s_k, Y)} w(ac|bd) + w(ad|bc) \tag{66}$$

We see from (61), that

$$s_{in} + s_{out} < \frac{m(Size(X \cup \{s_k\}) - 1)(Size(Y) - 1)}{2} \tag{67}$$

and from (64), that

$$s_{out} \geq \frac{m(Size(X) - 1)(Size(Y) - 1)}{2} \tag{68}$$

From (67), (68), we conclude that

$$s_{in} < \frac{m(Size(\{s_k\})(Size(Y) - 1)}{2} \tag{69}$$

At this stage, we are ready to show that $(X, Y) \subseteq M$. Let us define $error(y)$ as the sum of quartet errors across $(X \cup \{s_k\}, Y)$ involving both $y \in Y$ and $s_k$.

$$error(y) = \sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} \frac{(w(s_k y|xy') + w(s_k y'|xy))}{Size(y)Size(s_k)}$$

Let us assume that

$$error(y) \geq m, \text{ for all } y \in Y \tag{70}$$

which implies that for all $y \in Y$,

$$\sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} (w(s_k y|xy') + w(s_k y'|xy)) \geq m\,Size(y)\,Size(s_k) \tag{71}$$

such that,

$$\sum_{y \in Y} ( \sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} (w(s_k y|xy') + w(s_k y'|xy))) > \sum_{y \in Y} (mSize(y)Size(s_k)) \tag{72}$$

Moreover, we notice that

$$2s_{in} \geq \sum_{y \in Y} ( \sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} (w(s_k y|xy') + w(s_k y'|xy))) \tag{73}$$

Inequality (69), along with (72) and (73) gives us

$$\begin{aligned}
&2(\frac{m(Size(Y)-1)(Size(\{s_k\}))}{2}) \\
&> 2s_{in} \\
&= m(Size(Y)-1)(Size(\{s_k\}) \\
&\geq \sum_{y \in Y} ( \sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} (w(s_k y|xy') + w(s_k y'|xy))) \\
&> \sum_{y \in Y} (mSize(y)Size(s_k)) \\
&= mSize(Y)Size(s_k)
\end{aligned}$$

From the above set of equations, we have

$$\begin{aligned}
m(Size(Y)-1)(Size(\{s_k\})) &> m\,Size(Y)\,Size(\{s_k\}), \quad \text{which implies} \\
(Size(Y)-1) &> Size(Y)
\end{aligned}$$

which is a contradiction.

Thus our assumption that $error(y) \geq m$ for all $y \in Y$, must be false. Specifically, we can find at least one $y \in Y$ such that $error(y) < m$, i.e.

$$\Leftrightarrow \sum_{xs_k|yy' \in Q(X \cup \{s_k\}, Y)} \frac{(w(s_k y|xy') + w(s_k y'|xy))}{Size(y)Size(s_k)} < m$$

But this implies that

$$(X \cup \{s_k\}, Y) \in Best_{s_k y}(W_k, m)$$

Such that $(X \cup \{s_k\}, Y) \subseteq M$ as required. $\odot$

# 4 HC* collapsing mechanism

We examine the collapsing mechanism as introduced in Chapter 3 for speeding up the weighted Hypercleaning algorithm. WLOG, consider the resulting cluster tree $T$ induced by the edges returned by Hypercleaning on input set $S$ of vertices. We can reduce the input size $n$ of vertices by collapsing all but one of the clusters in $T$. The collapsing mechanism that achieves this has the following abstract signature

$$(W', T') \leftarrow COLLAPSE(W, T)$$

with the original cluster tree and the corresponding quartet set as input, it returns an updated quartet set on the reduced vertex set $V(T')$. Moreover, the collapser must guarantee that any subsequent edges recovered by Hypercleaning on $T', W'$, should have the same distance score as under the original $(T, W)$ pair. See Figure(17) for an illustration. Recall that two input parameters $(m, n)$, where $m$ stipulates the error allowance on returned edges, and $n$ is the size of input, dictate the running time of the algorithm. Under collapsing, we effectively reduce the input size of the problem, thereby allowing us to increase the $m$ value, in a computationally efficient manner, for resolving additional edges within some unresolved cluster on the current tree estimate.

## 4.1 Updating W for achieving loss-less collapsing

Phylogenetic inference of a given leaves set $S$ by HC* becomes an iterative process, where each iteration resolves some subset of the $n - 3$ unrooted edges of an estimate phylogeny. The resulting cluster tree is then collapsed into a single cluster of vertices $C$. The cluster set $C$ is then fed into HC* as input, along with quartet set $W'$ induced by $C$. As such, any subsequent edge $e$ resolved by HC* (e.g. $e \in Best(m, W')$ ) must satisfy Equation (49) of definition of a

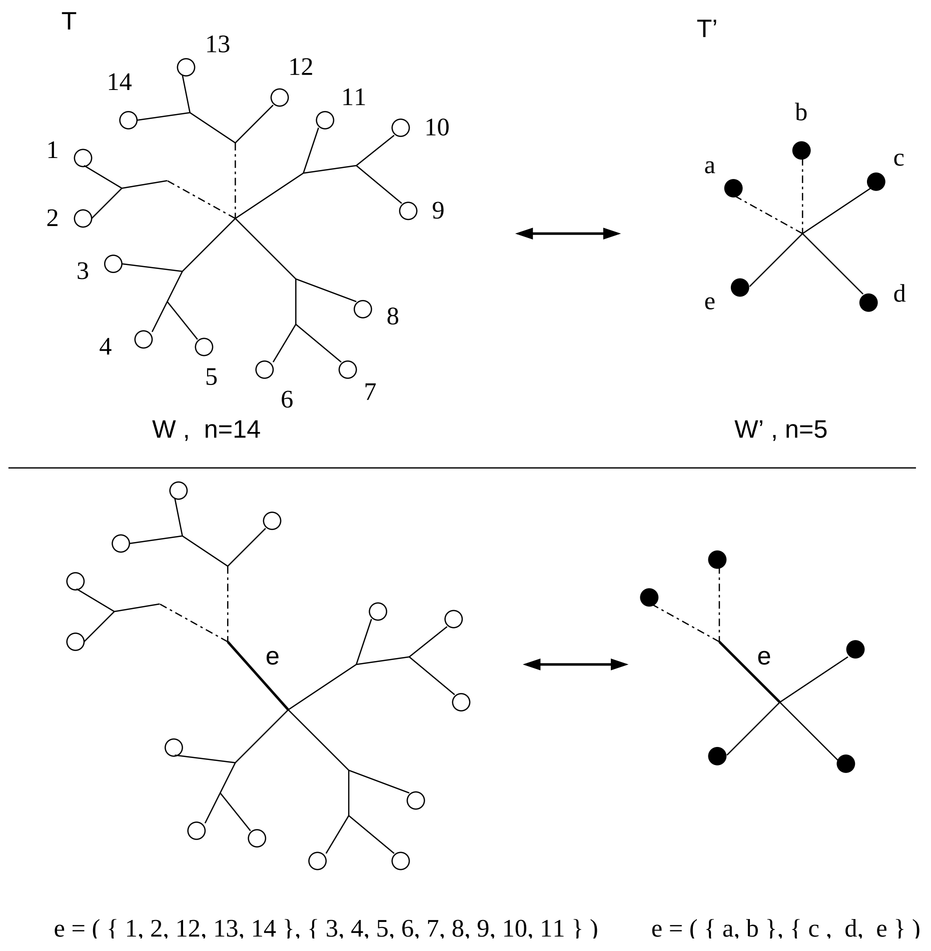

T
13
12
14
11
10
1
9
2
3
8
4
5
7
6
W , n=14
T'
b
a
c
e
d
W' , n=5
e
e
e = ( { 1, 2, 12, 13, 14 }, { 3, 4, 5, 6, 7, 8, 9, 10, 11 } )
e = ( { a, b }, { c , d, e } )

Figure 17: The top half of the diagram shows the reduction of the input size from $n = 14$ to $n = 5$ vertices by the collapsing mechanism. The main requirement of the collapser is that any subsequent resolution of an edge under $W'$ must have the same distance/support score as under the original quartet set $W$, as illustrated by the bottom half. In essence $\sigma(e, W) = \sigma(e, W')$, where $W, W'$ are weighted quartet sets defined under the $n = 14, n = 5$ vertex sets respectively.

loss-less edge recovering under collapsing. The following pseudo-code outlines the HC* inference procedure under collapsing:

COLLAPSE-RESOLVE

INPUT: $T'$: cluster tree, $HC^*$: algorithm, $W$: quartet set

OUTPUT: Resolved tree: $T$

```
0 $k = 0$;
1 $T^k = T'$;
2 $V^k = V(T)$; $W^k = W$;
3 WHILE ( $T^k$ not fully resolved )[6]
4     FOR some cluster $C_i \in Clu(T^{(k)})$
5         $W^{k+1}, V^{k+1} \leftarrow COLLAPSE^k \Big( Full \bullet (T^{(k)} \mid C_i), V^k, W^k \Big)$
6         $Best(m, W^{k+1}) = HC^*(V^{k+1}, m, W^{k+1})$
7     $T^{k+1} \leftarrow$ insert compatible edges $e \in Best(m, W^{k+1})$ into $T^k$
8     $k = k + 1$
9     $V^k = V(T^k)$
```

For the remaining of this chapter, we examine the $COLLAPSE$ procedure.

### 4.1.1 Algorithm for Updating W

Recall that in an effort to reduce the size of our leaves set, we collapse some unresolved cluster tree $T_C$ into a single cluster $C_i$ on a smaller set of vertices. To do so, we must first define a valid collapse ordering: $Full \bullet (T_C \mid C_i)$, and collapse each of the clusters in turn until only a single cluster remains. Then we proceed to run HC* on this small cluster of vertices in an attempt to resolve

[6]Alternatively, we could demand that $T^k$ is resolved to some user specified threshold, and then apply other methods to resolve the few remaining edges. Some applications might only require that only most of the edges of the underlying phylogeny be estimated.

more edges. However, any edge recovered this way must satisfy the information loss-less criteria. Consider an inductive case, where we have already done $k$ number of collapses in our $Full \bullet (T_C \mid C_i)$ operation, resulting in the current cluster tree $T_C^{(k)}$, with vertex set $V^{(k)}$, and the weighted quartet set $W^{(k)}$ on $V^k$. Assume that the quartet set $W^k$ satisfies the loss-less condition. Consequently for any edge $e = (X, Y)$ on $V^k$, we have:

$$\sigma(e = (X, Y), W^k) = \sigma(Exp(e), W)$$

where $W$ is the quartet set on the original input set $S$ of leaves. Now consider collapsing a new (non-trivial) cluster $C$ into supernode $\mathbf{c}$, such that the updated vertex set becomes: $V^{k+1} = V^k - C + \mathbf{c}$. We then need to construct an updated weighted quartet set $W^{k+1}$ on $V^{k+1}$ satisfying the loss-less condition. In the following procedure, we define this COLLAPSE mechanism.

**Procedure** $W^{k+1}, V^{k+1} \leftarrow COLLAPSE^k(Full \bullet (T_C^k \mid C_i), V^k, W^k)$
Given some unresolved cluster tree $T_C^k$ on vertex set $V^k$ with weighted quartet set $W^k$, we wish to collapse all the clusters of $T_C^k$ except $C_i$. The collapse ordering is defined by $Full \bullet (T_C^k \mid C_i)$. Without loss of generality, suppose the collapse ordering is given by:

$$C_1, C_2, ..., C_{i-1}, C_{i+1}, ..., C_q$$

The procedure $COLLAPSE$ is parametrized by the aforementioned ordering and can be characterized by a sequence of corresponding function calls:

$$\begin{aligned}
&W_{(1)}, V_{(1)} \leftarrow COLLAPSE_1^k(V^k, W^k, C_1)\\
&W_{(2)}, V_{(2)} \leftarrow COLLAPSE_2^k(V_{(1)}, W_{(1)}, C_2)\\
&\qquad\qquad\qquad \cdots\\
&W_{(q-1)}, V_{(q-1)} \leftarrow COLLAPSE_{q-1}^k(V_{(q-2)}, W_{(q-2)}, C_{q-2})\\
&W^{k+1}, V^{k+1} \leftarrow COLLAPSE_q^k(V_{(q-1)}, W_{(q-1)}, C_{q-1})
\end{aligned}$$

The following defines the actual algorithm for the procedure:

$$COLLAPSE^k_{j+1}(V^k_{(j)}, W^k_{(j)})$$

for $j = 0, \ldots, q-1$, while in next theorem we prove its correctness in satisfying the information loss-less precondition. Note that $V^0, W^0$ corresponds to the original input vertex set and its quartet set $V^k, W^k$.

**Procedure** $W_{(j+1)}, V_{(j+1)} \leftarrow \quad COLLAPSE^k_{(j+1)}\Big(V_{(j)}, W_{(j)}, C\Big),$

Assume we are in the $k$-th iteration of $Full \bullet (T_C) \mid C_i)$. Given some vertex set $V_{(j)}$, and quartet set $W_{(j)}$, $j = 0, ..., q-1$, where we assume that $W_{(j)}$ on $V_{(j)}$ satisfies the information loss-less precondition. Assume we want to collapse a cluster $C \neq C_i \subseteq V_{(k)}$ into supernode $\mathbf{s}$, such that we have $V_{(j+1)} = V_{(j)} - C + \mathbf{s}$. Thus, we construct an updated weighted quartet set $W_{(j+1)}$ on $V_{(j+1)}$ as follows, to satisfy the loss-less precondition:

Consider the following combinations of quartet: $(a, b, c, d) \in V_{(j+1)} = V_{(j)} - C + \mathbf{s}$, where these vertices do not have to be all distinct [7]

**C1** $(a, b, c, d)$, where $a \neq b \neq c \neq d \in V_{(j+1)} - \mathbf{s}$ :
For all such quartets, assign:

$$w_{(j+1)}(ab \mid cd) = w_{(j)}(ab \mid cd) \quad (74)$$

$$w_{(j+1)}(ac \mid bd) = w_{(j)}(ac \mid bd) \quad (75)$$

$$w_{(j+1)}(ad \mid bc) = w_{(j)}(ad \mid bc) \quad (76)$$

[7] Quartets can be of the form $(a, a, b, c)$ since $a$ might be a supernode which semantically represents collapsed leaves and or other supernodes. As such when we consider the distance score for a quartet topology $a, a \mid b, c$, we take into account the following alternative: $a, b \mid a, c$ which represents all quartets of the form $a_1, b \mid a_2, c$, where $a_1, a_2$ are two collapsed vertices in supernode $a$. A quartet of the form $(a, a, a, b)$ is impossible since no supernode can span across an edge.

**C2** $(a, a, c, d)$, where $a \neq c \neq d \in V_{(j+1)} - \mathbf{s}$ :

For quartets of this form, assign:

$$w_{(j+1)}(ac \mid ad) = w_{(j)}(ac \mid ad) \tag{77}$$

**C3** $(a, b, c, c)$, where $a \neq b \neq c \in V_{(j+1)} - \mathbf{s}$ :

For quartets of this form, assign:

$$w_{(j+1)}(ac \mid bc) = w_{(j)}(ac \mid bc) \tag{78}$$

**C4** $(a, a, b, b)$, where $a \neq b \in V_{(j+1)} - \mathbf{s}$ :

For quartets of this form, assign:

$$w_{(j+1)}(ab \mid ab) = w_{(j)}(ab \mid ab) \tag{79}$$

**C5** $(a, a, \mathbf{s}, \mathbf{s})$, where $a \in V_{(j+1)} - \mathbf{s}$

For these quartets, assign:

$$w_{(j+1)}(ab \mid ab) = \sum_{c,d \in C} w_{(j)}(ac \mid ad) + \sum_{c \in C} w_{(j)}(ac \mid ac) \tag{80}$$

**C6** $(a, b, c, \mathbf{s})$, where $a \neq b \neq c$, $a, b, c \in V_{(j+1)} - \mathbf{s}$:

For quartets of this form, assign:

$$w_{(j+1)}(ab \mid c\mathbf{s}) = \sum_{d \in C} w_{(j)}(ab \mid cd) \tag{81}$$

$$w_{(j+1)}(ac \mid b\mathbf{s}) = \sum_{d \in C} w_{(j)}(ac \mid bd) \tag{82}$$

$$w_{(j+1)}(bc \mid a\mathbf{s}) = \sum_{d \in C} w_{(j)}(bc \mid ad) \tag{83}$$

**C7** $(a, b, \mathbf{s}, \mathbf{s})$, where $a \neq b \in V_{(j+1)} - \mathbf{s}$

For these quartets, assign:

$$w_{(j+1)}(a\mathbf{s} \mid b\mathbf{s}) \quad = \frac{1}{2} \sum_{c,d \in C} \Big( w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc) \Big) \tag{84}$$

$$+ \sum_{c \in C} w_{(j)}(ac \mid bc) \tag{85}$$

**C8** $(a, a, b, \mathbf{s})$, where $a \neq b \in V_{(j+1)} - \mathbf{s}$

For these quartets, assign:

$$w_{(j+1)}(ab \mid a\mathbf{s}) = \sum_{d \in C} w_{(j)}(ab \mid ad) \tag{86}$$

$\odot$

**Theorem**

Given the above definition of $COLLAPSE^{k}_{(j+1)}$ on the $(j+1)$ st cluster collapse in the sequence of collapses as given by $Full \bullet (T_C \mid C_i)$, the resulting weighted quartet set $W_{(j+1)}$ on the updated vertex set $V_{(j+1)}$ satisfies the information loss-less precondition.

*proof-* Consider an arbitrary edge $e = (X, Y)$ on vertex set $V_{(j)}$, and all the quartet topologies that span across $e$. Let $C$ be the cluster to be collapsed, and assume without loss of generality $C \subseteq Y$. Specifically, these quartet topologies must fall into one of the ten cases illustrated in Fig(18).

**Case 1)** Quartet topology of the form $(ab \mid cd)$, where

$$a \neq b \in X; c \neq d \in Y; a, b, c, d \in V_{(j)} \setminus C \tag{87}$$

This quartet topology contributes the following amount to the score $\sigma(e, W_{(j)})$

$$w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc)$$

But since $a, b, c, d \notin C$, it follows that $a, b, c, d \in V_{(j+1)} \setminus \mathbf{s}$. Thus these quartets will appear in $W_{(j+1)}$ and their contribution to the distance score $\sigma(e, W_{(j+1)}$ will be:

$$w_{(j+1)}(ac \mid bd) + w_{(j+1)}(ad \mid bc) \tag{88}$$

The quartet weights in Equation (88) are covered by **C1** in the procedure for constructing $W_{(j+1)}$.

**Case 2)** Quartet topology of the form $(aa \mid cd)$, where

$$a \in X; c \neq d \in Y; a, c, d \in V_{(j)} \setminus C \tag{89}$$

This quartet topology contributes the following amount to the score $\sigma(e, W_{(j)})$

$$2w_{(j)}(ac \mid ad)$$

But since $a, c, d \notin C$, it follows that $a, c, d \in V_{(j+1)} \setminus \mathbf{s}$. Thus all quartets of the form $(aa \mid cd) \in W_{(j)}$ will also be in $W_{(j+1)}$, contributing the same distance score to $\sigma(e, W_{(j+1)})$:

$$2w_{(j+1)}(ac \mid ad) \tag{90}$$

The quartet weights in Equation (90) are covered by **C2** in the procedure for constructing $W_{(j+1)}$.

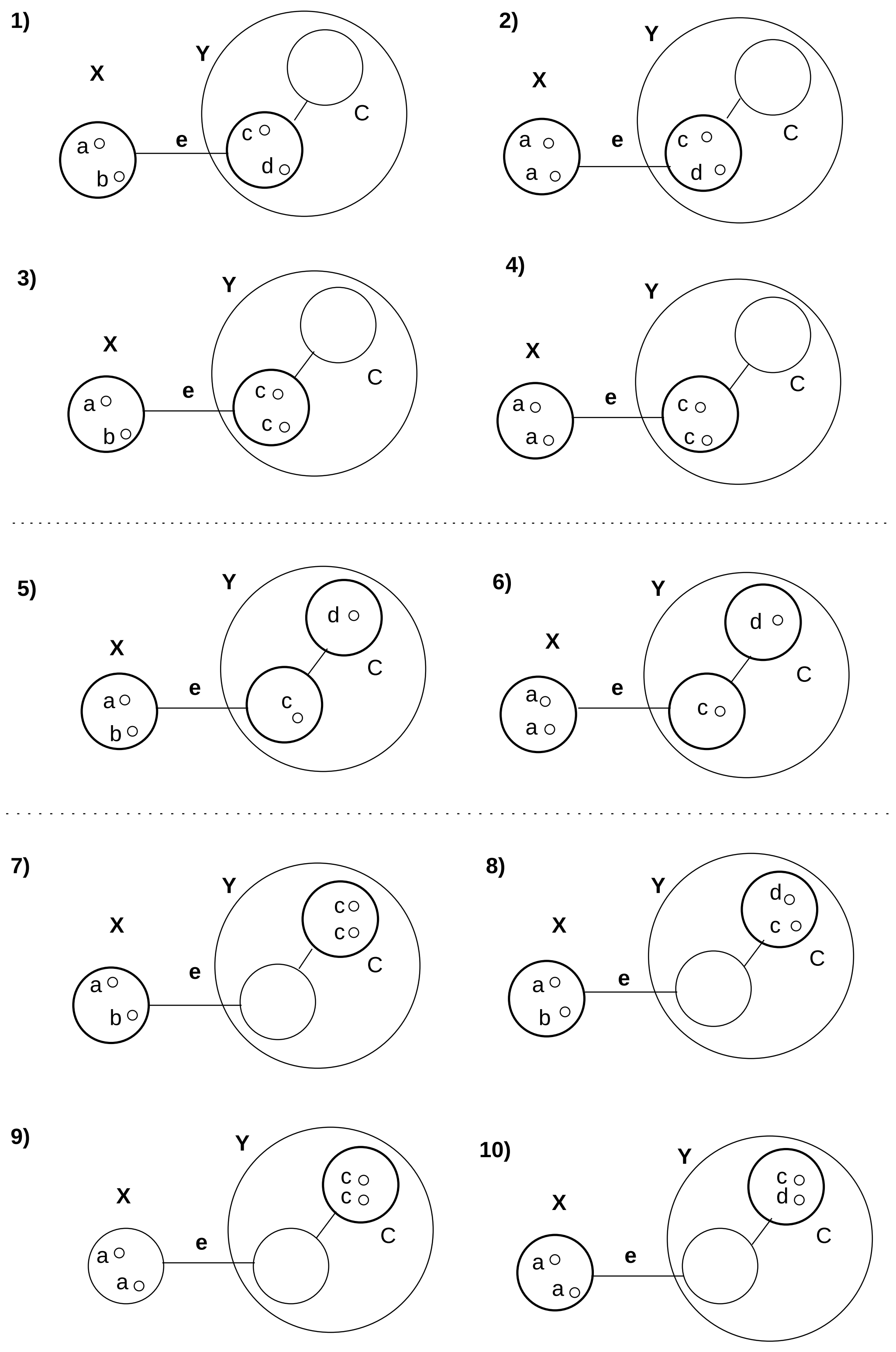


Figure 18: Given some vertex set $V_{(j)}$, consider an edge $\mathbf{e} = (X, Y)$ , where $X \cup Y = V_{(j)}$. Any quartet across edge $\mathbf{e}$ falls into one of the above 10 categories.

**Case 3)** Quartet topology of the form $(ab \mid cc)$, where

$$a \neq b \in X; c \in Y; a \neq b \neq c \in V_{(j)} \setminus C \tag{91}$$

This quartet topology $(ab \mid cc)$ contributes the following distance score to $\sigma(e, W_j)$:

$$2w_{(j)}(ac \mid bc)$$

But since $a, b, c \notin C$, it follows that all quartets of the form $(ab \mid cc) \in W_{(j)}$ will also be in $W_{(j+1)}$, contributing the same distance score to $\sigma(e, W_{(j+1)})$:

$$2w_{(j+1)}(ac \mid ad) \tag{92}$$

The quartet weights in Equation (92) are covered by **C3** in the procedure for constructing $W_{(j+1)}$.

**Case 4)** Quartet topology of the form $(aa \mid bb)$, where

$$a \in X; b \in Y; a, b \in V_{(j)} \setminus C \tag{93}$$

This quartet topology contributes the following amount to the distance score $\sigma(e, W_{(j)})$ :

$$2w_{(j)}(ab \mid ab)$$

But since $a, b \notin C$, it follows that $a, b \in V_{(j+1)} \setminus \mathbf{s}$. Thus all quartets of this form in quartet set $W_j$ will also be in quartet set $W_{j+1}$, contributing the same amount to the distance score $\sigma(e, W_{(j+1)})$:

$$2w_{(j+1)}(ab \mid ab) = 2w_{(j)}(ab \mid ab) \tag{94}$$

$$\Leftrightarrow \qquad w_{(j+1)}(ab \mid ab) = w_{(j)}(ab \mid ab) \tag{95}$$

The quartet weights in Equation (95) are covered by **C4** in the procedure for constructing $W_{(j+1)}$.

**Case 5)** Quartet topology of the form $(ab \mid cd)$ where

$$a \neq b \in X; c \neq d \in Y; a, b, c \in V_{(j)} \setminus C, \text{ and } d \in C \tag{96}$$

This quartet topology contributes the following amount to the distance score $\sigma(e, W_{(j)})$:

$$w_{(j)}(ad \mid bc) + w_{(j)}(ac \mid bd)$$

For a given fixed $a, b, c$, consider the collection of all quartet topologies $(ab \mid cd)$, for all $d \in C$. The total contribution of these quartets to distance $\sigma(e, W_{(j)})$ is given by

$$\sum_{d \in C} w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc)$$

Thus when cluster $C$ gets collapsed into supernode $\mathbf{s}$, then all the above quartet topologies in $W_{(j)}$ reduce to a single quartet topology $(ab \mid c\mathbf{s}) \in W_{(j+1)}$. Thus in quartet set $W_{(j+1)}$, for every quartet topology $(ab \mid c\mathbf{s})$, we must set its distance to

$$w_{(j+1)}(ac \mid b\mathbf{s}) + w_{(j+1)}(bc \mid a\mathbf{s}) = \sum_{d \in C} w_{(j)}(ac \mid bd) + w_{(j)}w(ad \mid bc) \tag{97}$$

The quartet weights in Equation (97) are covered by **C6** in the procedure for constructing $W_{(j+1)}$.

**Case 6)** Quartet topology of the form $(aa \mid bc)$ where

$$a \in X; b \neq c \in Y; a, b \in V_{(j)} \setminus C, \text{ and } c \in C \tag{98}$$

This quartet topology contributes the following amount to the distance score $\sigma(e, W_{(j)})$

$$2w_{(j)}(ab \mid ac)$$

For some fixed $a, b, c$, consider the collection of all quartet topologies $(aa \mid bc)$, $\forall c \in C$. These quartets contribute the following amount to the distance $\sigma(e, W_{(j)})$

$$\sum_{c \in C} 2w_{(j)}(ab \mid ac)$$

Thus when the cluster $C$ gets collapsed into supernode $\mathbf{s}$, then all the above quartets of the form $(aa \mid bc)$ in $W_{(j)}$ reduce to a single quartet topology $(aa \mid b\mathbf{s}) \in W_{(j+1)}$. Thus in $W_{(j+1)}$, we set the distance for $(aa \mid b\mathbf{s})$ to:

$$2w_{(j+1)}(ab \mid a\mathbf{s}) = \sum_{c \in C} 2w_{(j)}(ab \mid ac) \tag{99}$$

$$\Leftrightarrow \qquad w_{(j+1)}(ab \mid a\mathbf{s}) = \sum_{c \in C} w_{(j)}(ab \mid ac) \tag{100}$$

The quartet weights in Equation (100) are covered by **C8** in the procedure for constructing $W_{(j+1)}$.

**Case 7)** Quartet topology of the form $(ab \mid cc)$, where

$$a \neq b \in V_{(j)} \setminus C; c \in C. \tag{101}$$

This quartet topology $(ab \mid cc)$ contributes the following distance score to $\sigma(e, W_j)$:

$$2w_{(j)}(ac \mid bc)$$

The collection of all such quartet topologies $(ab \mid cc)$, for some fixed $a, b$ and for all $c \in C$, then makes the following total contribution to the distance score $\sigma(e, W_{(j)})$

$$\sum_{c, c \in C} 2w_{(j)}(ac \mid bc) \tag{102}$$

When $C$ is collapsed into supernode $\mathbf{s}$, all of the above quartet topologies in $W_{(j)}$ get reduced to a single quartet topology $(ab \mid \mathbf{ss}) \in W_{(j+1)}$, Thus in quartet set $W_{(j+1)}$, for every quartet of the above form, part of its distance score comes from the set of all collapsed quartets as described above. The remaining contribution to the distances for quartet topology of the form $(ab \mid \mathbf{ss})$ is covered in Case 8 (see below).

**Case 8)** Quartet topology of the form $(ab \mid cd)$, where

$$a \neq b \in V_{(j)} \setminus C, c \neq d \in C$$

This quartet topology $(ab \mid cd)$ contributes the following distance score to $\sigma(e, W_j)$:

$$w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc)$$

The collection of all such quartet topologies $(ab \mid cd)$, for some fixed $a, b$ and for all $c, d \in C$, contributes the following amount to the distance score across edge $e$. Thus when $C$ collapses into supernode $\mathbf{s}$, all of the above quartet topologies reduces into a single quartet $(ab \mid \mathbf{ss}) \in W_{(j+1)}$ , which we note is the same topology as in Case 7). In quartet set $W_{(j+1)}$, for every quartet of the form: $(ab \mid \mathbf{ss})$, part of its distance score comes from the set of collapsed quartets as mentioned above, which contributes the following total:

$$\sum_{c,d \in C} w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc)$$

Thus the total distance score on quartet topology $ab \mid \mathbf{ss}$ (as covered by this and Case-7) is as follows:

$$2w_{(j+1)}(a\mathbf{s} \mid b\mathbf{s}) = \tag{103}$$
$$\left( \sum_{c,d \in C} w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc) \right) + 2 \sum_{c \in C} w_{(j)}(ac \mid bc) \tag{104}$$
$$\Leftrightarrow w_{(j+1)}(a\mathbf{s} \mid b\mathbf{s}) = \tag{105}$$
$$\left( \frac{1}{2} \sum_{c,d \in C} w_{(j)}(ac \mid bd) + w_{(j)}(ad \mid bc) \right) + \sum_{c \in C} w_{(j)}(ac \mid bc) \tag{106}$$

The quartet weights in Equation (106) are covered by **C7** in the procedure to construct the weighted quartet set $W_{(j+1)}$. $\odot$

**Case 9)** Quartet topology of the form $(aa \mid cc)$, where $a \in V_{(j)} \setminus C$, and $c \in C$. This quartet contributes the following score to $\sigma(e, W_{(j)})$

$$2w_{(j)}(ac \mid ac), \text{ if } c = d$$

Thus the set of all such quartets contribute a total of following to $\sigma(e, W_{(j)})$

$$\sum_{c \in C} \left( 2w_{(j)}(ac \mid ac) \right)$$

When $C$ is collapsed into supernode $\mathbf{s}$, then all these quartets reduces to a single quartet $(aa \mid \mathbf{ss})$ in $V_{(j+1)}$. The above distances contribute partially to all the quartet distances across the quartet topology $aa \mid \mathbf{ss}$. The remaining distances are covered in Case-10.

**Case 10** Quartet topology of the form $(aa \mid cd)$, where $a \in V_{(j)} \setminus C$, and $c \neq d \in C$. This quartet contributes the following score to $\sigma(e, W_{(j)})$

$$2w_{(j)}(ac \mid ad), \text{ if } c \neq d$$

Thus the set of all such quartets contribute a total of following to $\sigma(e, W_{(j)})$

$$\sum_{c,d \in C} \left( 2w_{(j)}(ac \mid ad) \right)$$

When $C$ is collapsed into supernode $\mathbf{s}$, then all these quartets reduces to a single quartet $(aa \mid \mathbf{ss})$, in $V_{(j+1)}$. Note this is the same quartet topology as in Case-9.

Thus in total, the sum of all distances across quartet $(ac \mid \mathbf{ss})$ is given by:

$$2w_{(j+1)}(a\mathbf{s} \mid a\mathbf{s}) = \sum_{c,d \in C} 2w_{(j)}(ac \mid ad) + \sum_{c \in C} 2w_{(j)}(ac \mid ac) \qquad (107)$$

$$\Leftrightarrow w_{(j+1)}(a\mathbf{s} \mid a\mathbf{s}) = \sum_{c,d \in C} (w_{(j)}(ac \mid ad) + \sum_{c \in C} (w_{(j)}(ac \mid ac) \qquad (108)$$

The quartet weights in Equation (108) are covered by **C5** in the procedure to construct the weighted quartet set $W_{(j+1)}$. $\odot$

Given an arbitrary edge $e = (X, Y), X \cup Y = V_{(j)}$, the above cases cover all possibilities of a quartet across $e$. Moreover we construct the quartet set $W_{(j+1)}$ such that the distance scores of all quartets across $e$ on $W_{(j)}$ is accounted for in some corresponding quartet across $e$ on $W_{(j+1)}$. Thus through our construction, we have

$$\sigma(e = (X, Y \cup \mathbf{c}), W_{(j+1)}) = \sigma(e = (X, Y \cup C), W_{(j)}) \qquad (109)$$

But from our inductive assumption on $j$, we have that

$$\sigma(e = (X, Y \cup C), W_{(j)}) = \sigma(Exp(e), W) \qquad (110)$$

Thus we have

$$\sigma(e, W_{(j+1)}) = \sigma(Exp(e), W)$$

as required by the information loss-less precondition.

$\odot$

Thus the collapsing mechanism is defined by a collapsing scheme $Full\bullet(T_C \mid C_i)$, on our original unresolved tree $T_C$, initially on quartet set $W$. The procedure $COLLAPSE$ is defined by a sequence of procedure calls:

$$COLLAPSE_1, COLLAPSE_2, ..., COLLAPSE_q$$

where $COLLAPSE_j^k, j = 1, \ldots, q$ executes the $j$th collapse as dictated by $FULL \bullet (T_C \mid C_I)$. When collapsing the $j$-th cluster, procedure $COLLAPSE_j^k$, ensures that an updated quartet set $W_{(j)}$ on the resulting vertex set $V_{(j)}$ will satisfy the information loss-less condition. After all clusters except $C_i$ have been collapsed, we have a single cluster $C_i^*$ on vertex set $V_{(q)}$ along with the corresponding weighted quartet set $W_{(q)}$.

# 5 Results, Future Work and Conclusion

In this last section, we summarize the performance of the HC$^*$ algorithm in terms of its efficiency and effectiveness. The experiments were performed on real data sets sampled from the Ribosomal Database (RDP) [35]. All experiments are carried out using a PentiumII-350MHz PC with 256Megs of memory running Linux. Finally we list some related open problems/futures works followed by the conclusion.

## 5.1 Efficiency of HyperCleaning$^*$

One aspect of efficiency is the measure the speedup of HC$^*$ over HC. To show this, we designed the following experiment which utilizes a feature of HC$^*$'s collapsing mechanism: the ability to incorporate external knowledge about the inference problem at hand to significantly speedup its run time. Consider some given set $S$ of $n$ objects on which we wish to infer the phylogeny. Suppose that by some external means such as biological experiment, we resolve a deep edge (i.e. $e = (X, Y)$ where $\mid X \mid$ and $\mid Y \mid$ are rough $\frac{1}{2}n$). Under HC$^*$ we can incorporate this edge into the inference procedure by having it induce a cluster tree consisting of two large clusters of roughly equal size. HC$^*$ can then collapse one of the clusters into a supernode and resolve edges from the remaining cluster. This effectively reduces the original input space by half. Moreover, we can initially pick a small $m$ value for resolving high confidence edges from the one cluster. After one round of collapsing, HC$^*$ incorporates all compatible edges from the resulting set $Best(m, W)$ into the estimate tree and proceeds to pick the next largest cluster to resolve. The procedure iterates until all $n - 3$ non-trivial edges of the tree are inferred. Figure (19) depicts screen captures of the iterative collapse and resolution of a randomly sampled RDP 30-taxon tree under this scheme. The top-left window shows the original star tree as resolved by a single deep edge, inducing two large clusters. The top-right window shows one round of collapse, resolution, where some edges in the top cluster are resolved. The middle-left window shows the collapse and resolution of the bottom cluster. This iterative process continues until the final resolved tree as given in the bottom-right window. Note that for the original HC algorithm, the knowledge of a deep edge cannot be exploited in this fashion.

For the experiment we randomly sampled 30-taxon and 45-taxon trees from the RDP database. We ran both HC and HC$^*$ on these trees under the condition of resolving almost all of the $n-3$ underlying non-trivial edges. For HC we picked a sufficiently high $m$ value ($m = 20$ for 30-taxon trees, and $m = 25$ for 45-taxon trees) and then selected those compatible edges of the resulting $Best(m, W)$ set in a greedy fashion. For HC$^*$ we incorporated one deep edge to facilitate the iterative process for inferring the compatible edges in the tree estimate. For each sample tree, we compared the running time (CPUtime), the amount of edges recovered and the accuracy on both HC and HC$^*$. Tables (1), (2) shows the results on the 30-taxon, 45-taxon samples respectively. The metric used for accuracy is based on the Robinson-Foulds (RF) measure, where the RF accuracy is the number of edges (i.e. bipartitions) shared between the candidate tree and the true target tree.

Another aspect of efficiency is the comparison of the run-time of HC$^*$ compared to the run-time of efficient implementations of the established methods, specifically Parsimony, Neighbour Joining, and Quartet Puzzling. Note that Maximum Likelihood methods will not be considered, since even the fastest implementation of ML employing heuristics (e.g. fastDNAML, or ML in the PAUP package) become computationally impractical on large data sets. We ran HC$^*$ against the PAUP [52] package implementation of neighbor joining (paupNJ), parsimony (paupPars), and quartet puzzling (papuPuzPars) which uses parsi-

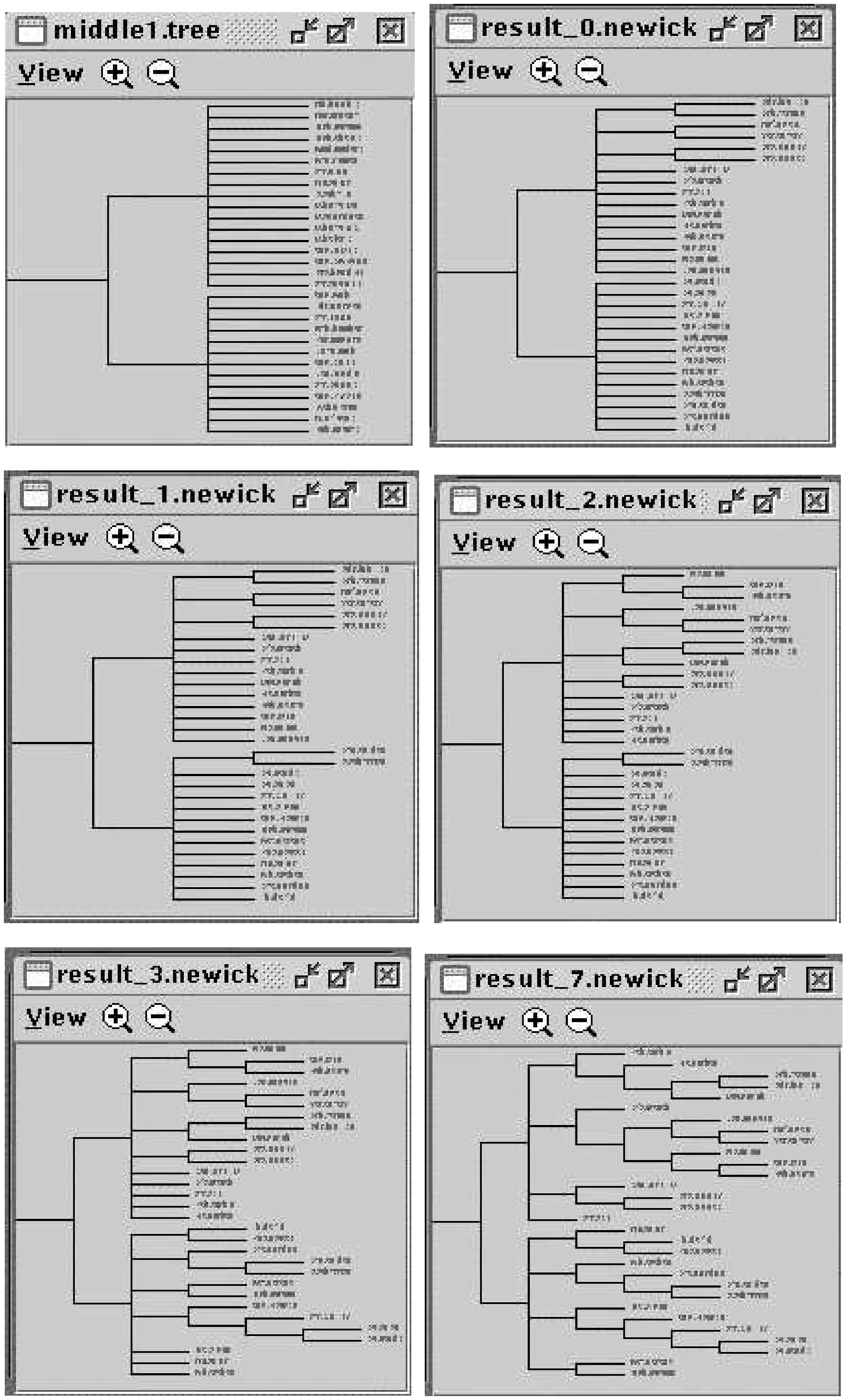


Figure 19: The iterative collapse and resolution of a 30-taxon tree. Images generated by InVest.

| Comparison of running time: HC vs. HC$^*$ on 30-taxon trees | | | | | | |
|---|---|---|---|---|---|---|
| | HC | | | HC$^*$ | | |
| tree | time (s) | res | accuracy | time (s) | res | accuracy |
| 1 | 492 | 27 | 12 | 10 | 27 | 14 |
| 2 | 298 | 24 | 8 | 17 | 27 | 7 |
| 3 | 342 | 26 | 10 | 15 | 26 | 13 |
| 4 | 349 | 25 | 13 | 10 | 27 | 16 |
| 5 | 290 | 24 | 16 | 14 | 26 | 17 |

Table 1: Efficiency measure on 30-taxon tree. The time column measures the CPU usage in seconds, 'res' is the amount of edges resolved, and accuracy is the number of correctly inferred edges. For HC, the $m$ value was set to 20.

| Comparison of running time: HC vs. HC$^*$ on 45-taxon trees | | | | | | |
|---|---|---|---|---|---|---|
| | HC | | | HC$^*$ | | |
| tree | time (m:s) | res | accuracy | time (m:s) | res | accuracy |
| 1 | 212:57 | 36 | 18 | 1:43 | 40 | 20 |
| 2 | 246:13 | 38 | 19 | 1:15 | 37 | 20 |
| 3 | 201:24 | 36 | 20 | 1:43 | 39 | 21 |
| 4 | 217:22 | 38 | 21 | 1:49 | 40 | 23 |
| 5 | 230:18 | 37 | 19 | 1:44 | 41 | 21 |

Table 2: Efficiency measure on 45-taxon tree

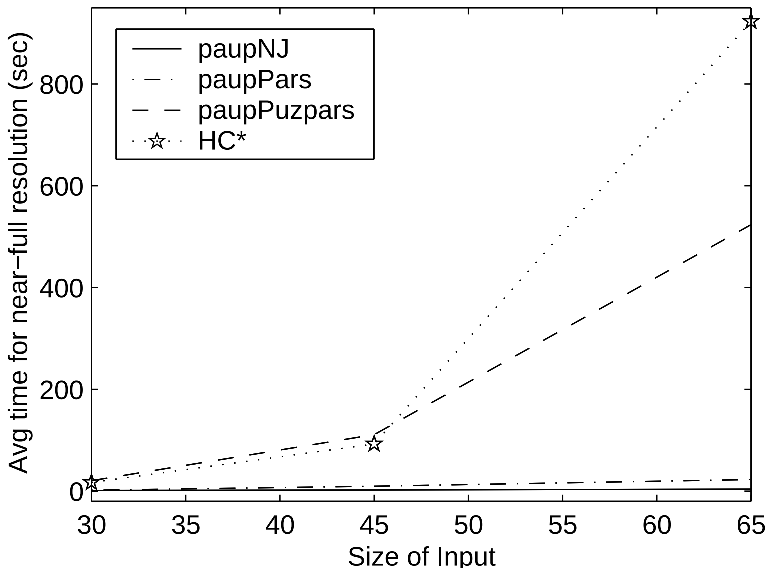


Figure 20: Experimental run time of the inference methods with respect to input size

| avg run time on | paupNJ | paupPars | paupPuzPars | HC* |
|---|---|---|---|---|
| 30-taxon | 1.2 | 1.9 | 20.4 | 17.1 |
| 45-taxon | 2.3 | 9.6 | 110.9 | 92.5 |
| 65-taxon | 4.0 | 22.8 | 523.7 | 923.4 |

Table 3: Comparison of average run time in seconds across methods

mony to construct its quartet set. We randomly sampled six trees from the RDP of sizes 30 and 45. Then we took the average (CPUtime) runtime of each of the methods. Table (3) summarizes the results and Figure( 5.1 ) shows the plot of the rough approximation of the average runtime across these methods as a function of input size.

## 5.2 Effectiveness of HyperCleaning$^*$

To test the accuracy of HC$^*$, we designed an experiment using real dataset by taking a random sample of subtrees of various sizes of the RDP database. Next we ran our implementation of HC$^*$ against paupNJ, paupPars and paupPuzPars. We also used fastDNAml implementation [37] of ML on the smaller datasets, which became computationally intractable for larger datasets with greater than 30 leaves. Tables (4) and (5) lists the accuracy of the methods on 30-taxon and 45-taxon trees. The witness quartet set $Q$ that HC$^*$ used was generated with fastDNAml.

| Tree | Edges recovered : max (/ 27) | | | | |
|---|---|---|---|---|---|
| | paupNJ | paupPars | paupPuzPars | HC$^*$ | fastDNAml |
| 0 | 8 | 12 | 13 | 14 | 16 |
| 1 | 11 | 11 | 11 | 14 | 13 |
| 3 | 13 | 13 | 13 | 15 | 12 |
| 5 | 5 | 9 | 7 | 8 | 7 |
| 7 | 12 | 12 | 11 | 14 | 13 |
| 8 | 11 | 15 | 15 | 15 | 19 |
| 10 | 10 | 15 | 11 | 16 | 17 |
| 11 | 16 | 15 | 17 | 16 | 15 |
| 12 | 11 | 15 | 18 | 17 | 15 |
| 14 | 7 | 8 | 8 | 10 | 11 |
| 15 | 15 | 13 | 15 | 13 | 13 |
| 16 | 10 | 11 | 13 | 14 | 14 |
| mean(%) | 39.81 | 46.00 | 46.91 | 54.00 | 51.00 |

Table 4: Comparison of Methods across 12 Randomly Sampled 30-Taxon RDP Trees

| Tree | Edges recovered : max (/ 42) | | | |
|---|---|---|---|---|
| | paupNJ | paupPars | paupPuzPars | HC* (w fastDNAml) |
| 0 | 11 | 17 | 10 | 20 |
| 1 | 19 | 20 | 21 | 26 |
| 2 | 20 | 24 | 22 | 21 |
| 3 | 19 | 21 | 20 | 23 |
| 4 | 13 | 18 | 17 | 21 |
| 5 | 21 | 19 | 16 | 19 |
| 6 | 12 | 15 | 17 | 23 |
| 7 | 14 | 21 | 21 | 20 |
| 8 | 20 | 23 | 21 | 24 |
| 9 | 17 | 19 | 21 | 22 |
| 10 | 18 | 19 | 19 | 19 |
| 11 | 19 | 24 | 19 | 24 |
| mean(%) | 40.29 | 49.60 | 44.45 | 54.00 |

Table 5: Comparison of Methods across 12 45-Taxon RDP Trees

## 5.3 Future Work

There are many interesting problems and extensions on HC$^*$ algorithm that remains unanswered. First we examine some practical problems which the author intends to follow up on, some practical but open issues, and finally we present some open issues that are of theoretical interest.

**Follow Ups -** The reported results were performed on several small scale experiments where the datasets are randomly sampled from real (difficult) datasets. In would also be very useful to test HC$^*$ on various input parameters such as various shallow and deep trees (i.e. trees with small and large sequence divergence amongst their edges), or select trees that exhibit cases of long-branch attraction amongst some subset of their leaves. Moreover we could run simulations studies, whereby we can control indirectly the quality of the inferred quartet sets by tweaking the parameter space such as the sequence length, edge lengths and various assumptions on the models of evolution. A comparative analysis of the robustness of HC$^*$ vs. the other quartet methods (e.g. Puzzling) across the various parameter space might be interesting.

Currently a more substantial simulation study is in progress, where ten random topologies on taxon set of size: 30, 50, 70, and 100 are randomly sampled from the RDP database. For each topology, we adopt the HYK model [33] of evolution of sequences using the SeqGen [40] sequence generator along the tree topologies using a wide spectrum of parameter values on the HYK model of evolution as follows:

- The default transversion/translation rate of: $\{4 : 1\}$.
- A rate heterogeneity of: $\alpha = \{0.5, 1.0, 2.0\}$.
- Leaves of sequence lengths: $\{500, 1000, 2000\}$.
- Branch length scaling factor of: $\{0.4, 1.0, 2.0\}$.

Thus in total, a set of 10*3*3*3=270 datasets are produced for each of the taxon set sizes: 30, 50. For computational efficiency reason, we randomly picked a subset of 54 out of the 270 datasets on taxon sets of sizes: 70, 100. On these (simulated) true data sets, we proceed to test the effectiveness of HC$^*$ as an approximation method for ML, in that we use $fastDNAml$ to infer the quartet sets, and subsequently employ HC$^*$ to assemble the quartet set into a phylogeny hypothesis. For comparison, we ran the data sets on the PAUP-Parsimony, PAUP-NJ, and TREE-PUZZLE [51] methods. The measure of accuracy is the standard Robinson-Foulds measure. Results and interpretation will be appended as errata when made available (scheduled for early 2002).

**Open Issues -** Given the set $Best(m, W)$ of best supported edges returned by HC$^*$, it is not guaranteed that all edges are mutually compatible. We adopted the greedy solution of iteratively adding the next compatible edge with respect to the existing compatible set in a top down manner. In essence we have the following optimization problem: given set $Best(m, W)$, we wish to select the maximal set of compatible edges, denoted $CE$ that minimizes

$$\sum_{(X,Y)\in CE} \sigma(W, (X, Y))$$

. Note this can be thought of as the sum of $L_1$ norms on the bipartitions. For a variation on the above optimization problem, consult [7] where the objective function is based on $L_\infty$ rather than $L_1$ norms on the bipartitions.

Another open issue is the construction of weighted quartets for improving the accuracy and/or robustness of the latter inference step. Given a set of $n$ leaves, clearly some subset of these quartets are more important or dependable than others. An interesting problem is how to recognize these so called ‘good’ and ‘bad’ apples and construct the weighted quartet set to reflect this. Another interesting idea is to employ ensemble learning techniques to construct a combined classifier of multiple experts (i.e. inference methods) to improve the baseline quality of the quartet set. To date not many work has been done in this area.

The HC$^*$ method effectively takes the ‘art’ out of constructing the edges from the quartet witness set, in that it returns the set of all edges in the order of decreasing support from the quartet set. Moreover as seen in chapter 2, the performance of HC$^*$ rests on the degree of quality of the quartet set. The challenge now rests on the efficient and accurate inference of the quartet set. Further work in improving the quartet inference stage includes:

- The quartet inference stage is inherently an $O(n^4)$ complexity task given input of size $n$. Often on input sizes of moderate to large size such as $n > 300$, the quartet inference stage itself becomes a dominant process over the HC$^*$ algorithm. As such it would be beneficial to identify from the input set $S$, a subset $S' \subseteq$ of 'bad apples' that can be problematic in that they induce a relatively larger proportion of the quartet errors in the resulting quartet set. In essence we are trying to remove those pathologically hard leaves for which to infer quartet topologies on. The benefit of this is twofold: i) the remaining input set $S/S'$ is smaller and thus more computationally efficient to infer the corresponding quartet set, and ii) the resulting edges recovered by HC$^*$ should be more accurate. However the tradeoff is that the resulting tree will cannot be fully resolved with respect to the original input set $S$, and thus we must perform post processing of those 'bad apple' leaves to elucidate their position in the resulting unresolved phylogeny.

- In an attempt to improve the quality of the inferred quartet set, an ensemble learning approach may be adopted. As such we construct an ensemble $E = \{E_1, .., E_k\}$ of a mixture of individual experts (i.e. phylogenetic inference methods) and combine their collective evidence in an attempt to 'boost' the accuracy of the quartet set. Many strategies for combining expert evidence may be tried such as weighted voting [11], linear combination of experts using Kernel based methods such as Maximum Margins [22], [47], or Support Vector Machines [10].

**Theoretical Issues -** HC$^*$ lacks theoretical analysis both in terms of runtime and memory requirements. In the worst case, we proposed a worst case memory bound of $3k(\binom{n}{4})$ on input size $n$ (see appendix) and small constant $k$. However the author believe that this bound is very loose and in reality that HC$^*$ should take no more than $2k(\binom{n}{4})$ memory under amortized analysis. The average runtime complexity is non-trivial to analyze since the average case depends on several factors, such as the average size of the clusters, average number of clusters in an unresolved tree, and the quality of the quartet set.

# References

[1] Badger, J., Hu, M., and Kearney, P. Identifying bad apples in inferring phylogenetic trees. *Manuscript* (2002).

[2] Badger, J., and Kearney, P. Picking fruit from the tree of life: Comments on taxanomic sampling and the quartet method. In *Proceedings of the 16th ACM Symposium on Applied Computing* (2001), ACM Press, pp. 61–67.

[3] Berry, V. e. a. A practical algorithm for recovering the best supported edges of an evolutionary tree. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms* (2000), ACM Press, pp. 287–296.

[4] Blanchette, M. Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution 49* (1999), 191–203.

[5] Blanchette, M. Algorithms for phylogenetic footprinting. In *Proceedings of the 5th Conference on Research in Computational Molecular Biology* (2001), ACM Press, pp. 49–58.

[6] Bruno, W., Socci, N., and Halpern, A. Weighted neighbour joining: A likelihood-based approach to distance based phylogeny reconstruction. *Molecular Evolution Biology 17* (2000), 189–197.

[7] Bryant, D. Structures in biological classifications. *PhD Thesis, Department of Mathematics, University of Canterbury* (1997).

[8] Buneman, P. *The Recovery of trees from measures of dissimilarity.* Edinburgh University Press, 1971.

[9] Chen, K., Durand, D., and Colton, M. Notung: Dating gene duplications using gene family trees. In *Proceedings of the 4th Conference on Computational Molecular Biology* (2000), ACM Press, pp. 96–106.

[10] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20* (1995), 273.

[11] Duda, R., and Hart, P. *Pattern Classification.* Wiley and Sons, 2001.

[12] DURING, R., EDDY, S., KROGH, A., AND MITCHISON, G. *Biological Sequence Analysis: Probabilistic Models for Protein and Nucleic Acids.* Cambridge University Press, 1998.

[13] EISEN, J. Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Research 8* (1998), 163–167.

[14] EL-MABROUK, N., BRYANT, D., AND SANKOFF, D. Reconstructing the pre-doubling genome. In *Proceedings of the 3rd Conference on Computational Molecular Biology* (1999), ACM Press, pp. 154–163.

[15] ERDOS, P., RICE, K., STEEL, M., SZEKELY, L., AND WARNOW, T. The short quartet method. *International Congress on Automata, Languages and Programming* (1997).

[16] FARRIS, J. Method for computing wagner trees. *Systematic Zoology 34* (1970), 21–34.

[17] FELSENSTEIN, J. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution 17* (1981), 368–376.

[18] FITCH, W. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology 20* (1971), 406–416.

[19] FREY, B. *Graphical Models for Machine Learning and Digital Communications.* MIT press, 1998.

[20] FRIEDMAN, N., NINIO, M., AND PE'ER, I. A structural EM algorithm for phylogenetic inference. In *Proceedings of the 5th Conference on Computational Molecular Biology* (2001), ACM Press, pp. 132–140.

[21] GRAYBEAL, A. Is it better to add taxa or characters to a difficult phylogenetic problem? *Systematic Botany 47* (1998), 9–17.

[22] GROVE, A., AND SCHUURMANS, D. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the 15-th National Conference on Artificial Intelligence* (1998), vol. 15, AAAI.

[23] Hallett, M., and Lagergren, J. Efficient algorithms for lateral gene transfer problems. In *Proceedings of the 5th Conference on Computational Molecular Biology* (2001), ACM Press, pp. 149–156.

[24] Hendy, M., and Penny, D. A framework for the quantitative study of evolutionary trees. *Systematic Zoology 38* (1989), 297–309.

[25] Hillis, D. Approaches for assessing phylogenetic accuracy. *Systematic Biology 44* (1995), 3–16.

[26] Hillis, D., and Huelsenbeck, J. Hobgoblin of phylogenetics. *Nature 369* (1994), 363–364.

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

[28] Jukes, C., and Cantor, C. *Evolution of Protein Molecules in Mammalian Protein Metabolism.* Academic Press, 1969.

[29] Kearney, P. The ordinal quartet method. *Proceedings of the 2nd Annual International Conference on Computational Molecular Biology* (1998), 125–134.

[30] Kearney, P. *Phylogenetics and the Quartet Method.* Current Topics in Computational Biology. Springer-Verlag and Tsinghua University Press, 2000.

[31] Kim, J. Large scale phylogenetics and measuring the performance of phylogenetic estimators. *Systematic Biology 47* (1998).

[32] Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies in nucleotide sequences. *Journal of Molecular Biology 16* (1980), 111–120.

[33] Li, M., Badger, J., Xin, C., Kwong, S., Kearney, P., and Zhang, H. An information based sequence distance and its applications to whole genome mitochondrial phylogeny. *Bioinformatics* (2001).

[34] Li, W. *Molecular Evolution.* Sinauer Associates Inc., Sunderland, MA, US, 1997.

[35] Maidak, B., Cole, J., Lilburn, T., Parker, C., Saxman, P., Farris, R., Garrity, G., Olsen, G., Schmidt, T., and Tiedje, J. The RDP-II (Ribosomal Database Project). *Nucleic Acids Research 29* (2001), 171–173.

[36] Nei, M., and Kumar, S. *Molecular Evolution and Phylogenetics.* Oxford University Press, 2000.

[37] Olsen, G., Matsuda, H., Hagstrom, R., and Overbeek, R. Fastdnaml: a tool for construction of phylogenetic trees of dna sequences using maximum likelihood. *Current Applications in Biosciences 10* (1994), 41–48.

[38] Patterson, D. *Tree of Life.* http://ag.arizona.edu/tree/eukaryotes/eukaryotes.html, 2000.

[39] Pellegrini, M., Marcotte, E., Thompson, M., Eisenberg, D., and Yeates, T. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of Natural Academy of Science, USA 96* (1999), 4285–4288.

[40] Rambaut, A., and Grassly, N. Seq-gen: An application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Comput. Appl. Biosci. 13* (1997).

[41] Ranwez, V., and Gascuel, O. Quartet based phylogenetic inference: Improvements and limits. *Mol. Biol. Evol.* (2001), 1103–1116.

[42] Rehmsmeier, M., and Vingron, M. Phylogeny meets sequence search. *Proceedings of the 14th German Conference on Bioinformatics* (1999), 66–72.

[43] Robinson, D., and Foulds, L. Comparison of phylogenetic trees. *Mathematical Biosciences 53* (1981), 131–147.

[44] Rohlf, F. Consensus indices for comparing classifications. *Mathematical Biosciences 59* (1981), 131–144.

[45] Saitou, N., and Rei, M. The neighour joining method: A new method for reconstructing phylogenetic trees. *Mol. Bio. Evol. 4* (1987), 406–425.

[46] Sankoff, D. *Time Warps, String Edits, and Macromolecules.* CSLI Publications, 1982.

[47] Schapire, R., Freund, Y., Bartlett, P., and Lee, W. Boosting the margin: A new explanation for the effectiveness of voting methods. *Machine Learning 14* (1997).

[48] Stanhope, M., Lupas, A., Italia, M., Koretke, K., Volker, C., and Brown, J. Phylogenetic analysis do not support horizontal gene transfers from bacteria to vertebrates. *Nature 411* (2001), 940–944.

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

[50] Steel, M., and Penny, D. Parsimony, likelihood, and the role of models in molecular phylogenetics. *Mol. Biol. Evol. 17* (2000), 839–850.

[51] Strimmer, K., and Haeseler, A. Quartet puzzling: a quartet maximum likelihood method for reconstructing tree topologies. *Mol. Biol. Evol. 13* (1996), 964–969.

[52] Swofford, D. *PAUP Version 4.0.* Sinauer Associates, Sunderland, MA, http://www.lms.si.edu/PAUP/.

[53] Swofford, D., Olsen, G., Waddell, P., and Hillis, D. *Molecular Systematics*, 2 ed. Sinauer Associates, MA, 1996.

[54] Waterman, S., Smith, T., Singh, M., and Beyer, W. Additive evolutionary trees. *Journal of Theoretical Biology 64* (1977), 199–213.

# A Implementation Issues

In the next few pages, we examine some data structure and implementation related issues one should consider in implementing HyperCleaning$^*$. The following serves as a roadmap and is neither a formal specification or a claim to the most efficient way of doing things.

## A.1 Implementations of $Best(W_k, m)$, and $Best_{xy}(W_k, m)$

Given our input set $S$ on $n$ leaves, consider the construction of the sets $Best_{xy}(m, W_K)$ for all $x, y \in 1, .., n$ and $Best(W_k, m)$ for $k = 1, .., n$. For a given value of $k$, observe that we only require the sets $Best_{xy} W_{k-1}, m, \forall x, y \in 1, .., n$, and $Best_(W_{k-1}, m)$. i.e. we only need the information from the previous $k$ value. We can implement this using a slice approach where during each iteration on $k$, we keep only the necessary information on the previous slice. Figure (21) is a diagrammatic illustration of the requirements for constructing the set $Best(W_k, m)$. The diagram also shows the structure of the sets $Best_{xy}(W_k, m), Best(W_k, m)$. We examine several structure related issues:

1. For a set $Best_{xy}(W_k, m), k = 1, .., n$, note that $Best_{x=i,y=j}(W_k, m)$ is equivalent to the set $Best_{x=j,y=i}$ since both represent the set of edges involving quartets on labels $i, j, 1, 2, .., k$ where two of vertices on each quartet are $i, j$. Thus in its implementation, we only need to construct and keep one copy which can be done using a lexicographical ordering

$$2 \leq x \leq n, 1 \leq y < x$$

   . This is reflected in figure (21).

2. As shown in figure (21), each cell in the table representation of $Best_{xy}(m, W_k)$ and $Best(m, W_k)$ corresponds to a set of edges. Since the most frequent

operation on the sets $Best_{xy}(m, W_k)$ and $Best(m, W_k)$ is membership testing, it seems reasonable that we should implement a split set using a hashtable. One concern about hashing an edge/biparition is the evaluation of its hash function value. Note that two edges can be the same, but have different left/right partite ordering and ordering of labels within a paritite. For example the following two edges $e_1, e_2$ are the same:

$$e_1 = (\{1, 2, 3\}, \{4, 5, 6, 7\}) \;\; e_2 = (\{5, 4, 7, 6\}, \{1, 3, 2\})$$

Thus we must adopt a hash function that accounts for this situation. To get around the ambiguity of the left and right partite, we maintain the invariant that the left partite of an edge is the larger of the two. This invariant is checked and updated if necessary after any operation that changes the sizes of the partites of an edge. The exception is when both partites have the same cardinality, which must be dealt with separately. Since two given partites can have the same elements but in an different order, we must examine all the elements when evaluating the hash function value on a split. In our implementation, we used two bitvectors $V_L, V_R$ , for the left and right partitie respectively. For every element $i$ in the partite $X$, we would set the $ith$ bit in vector $V_X$. To compute the hash value, we chopped both bitvectors into segments of length $sizeof(int)$, padding it with zeros if necessary. Then we took the XOR of the segments of $V_L$ followed by summing the result with each and every segment of $V_R$. The above scheme of representing a split and evaluating its hash value seems to to work well in terms of minimizing the number of collisions between different splits.

## A.2 Collapsing and Shifting Indices

To facilitate the collapsing algorithm, we introduce a Collapser class, that should be responsible for the following:

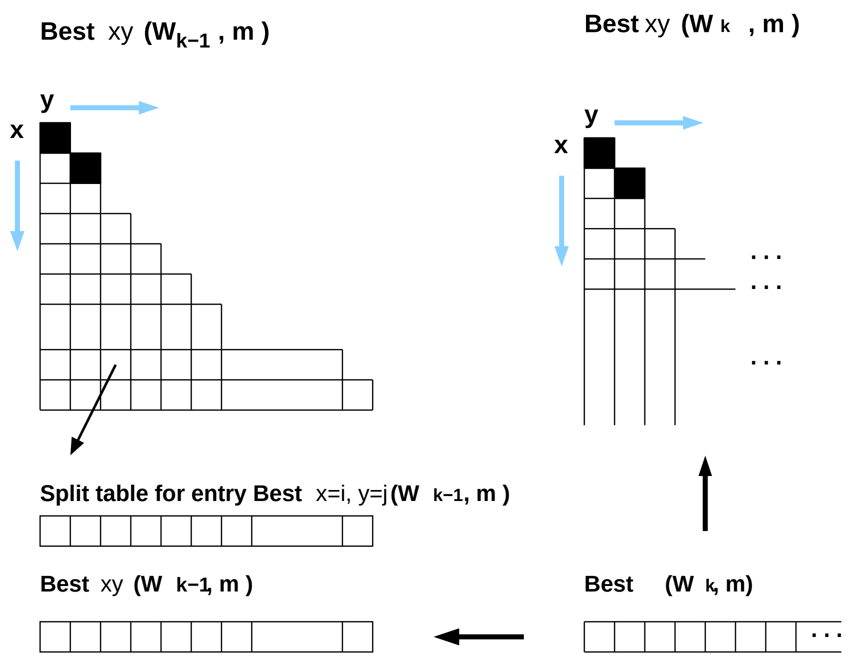


Figure 21: Constructing the set $Best(W, k)$ for some current value of $k$ requires only information from the previous slice $k - 1$. i.e. the sets: $Best_{xy}(W, k - 1), \forall x, y \in 1, ..n$, and $Best_(W, k - 1)$.

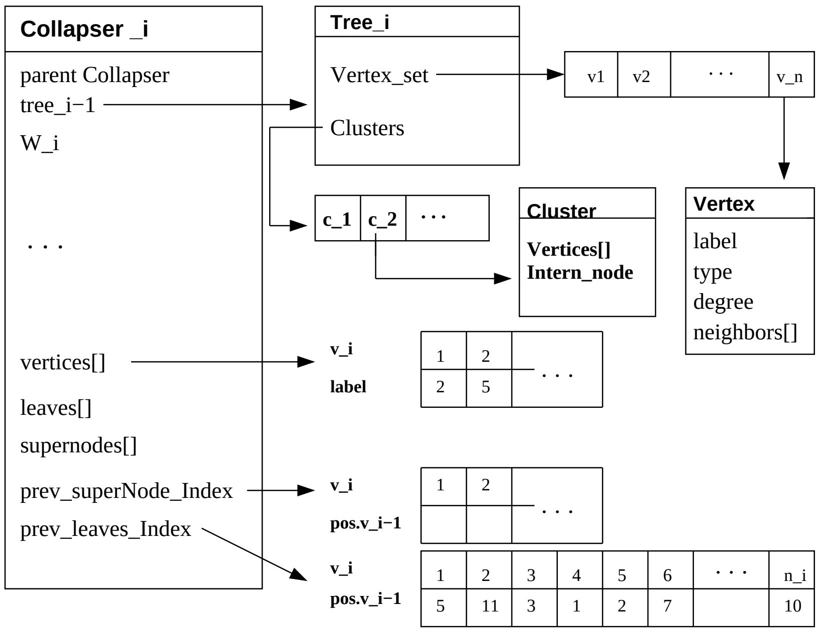


Figure 22: The Collapser class diagram, with its major attributes.

1. Given a cluster tree $T^{i-1}$, induce the individual clusters and evaluate a valid collapse ordering

2. The collapse of the tree into a star topology based on some ordering

3. Updating the quartet set $W^{i-1}$ into the appropriate new set $W^i$

We can then run Hypercleaning on the labels of the start topology using quartet set $W^i$.

Figure (22) is a class diagram of Collapser, its major attributes, and their structures. The following lists the major attributes:

- *parent-Collapser*: references the Collapser object from the previous round of collapsing

- $tree_{i-1}$: references the cluster tree object that will be collapsed by the current Collapser. A tree object has two major attributes: 1)*Vertex-Set*[]: a vector of Vertices (both internal and leaf nodes) of the tree, and

2)$Clusters[]$: a vector [8] A valid collapse ordering generated by the Collapser object uses the information from its $tree_{i-1} \rightarrow Clusters[]$ field.

- $leaves[]$: a vector of leaves remaining **after** the corresponding cluster tree ($tree_{i-1}$) have been collapsed. This field is set to NULL prior to the collapse of the cluster tree.

- $supernodes[]$: a vector of supernodes remaining **after** the collapse of the cluster tree. This field is set to NULL prior to the cluster tree's collapse. Each *supernode* object in this vector MUST have a record of its *size* (i.e. number of leaf nodes collapsed into it thus far).

- $vertices[]$: a vector of the label of all the vertices after the collapse of the cluster tree. This vector can be simply a concatenation of the labels of the nodes from the $leaves[]$ and $supernodes[]$ vectors. This vector makes it convenient when we construct the updated quartet set $W^i$ to lexicographically generate all $\binom{n}{4}$ quartets by nested loops along the nodes of this vector.

- The vectors $prev\_superNode\_index[]$, $prev\_leaves\_Index[]$, provides a mapping on the vertices (both leaves and supernodes) between the current and previous round of collapsing. This mapping scheme of keeping track of the index of the leaf/supernode labels from one round of collapsing to the next is necessary since for a particular round, those leaves and/or supernodes in a cluster to be collapsed can be located anywhere in the $leaves[]$, $supernodes[]$ and $vertices[]$ vectors. Consequently in the updated vectors, the indices in these vectors are now shifted, and a correspondence between the before/after collapsing must be established. Thus $this \rightarrow prev\_leaves\_Index[i] = k$ says that the leaf label in $this \rightarrow leaves[i]$ on the current round (i.e. **after** the collapsing) can be found in the $k$th entry of $this \rightarrow parentCollapser \rightarrow leaves[]$. Or that the leaf label has its corresponding entry in index $k$ in the previous round. Figure(23) illustrates

[8]We use terms like vectors, tables, structures, classes, and use of class diagrams only for descriptive purposes and do not adopt any assumptions about their implementation or structural properties, unless explicitly stated.

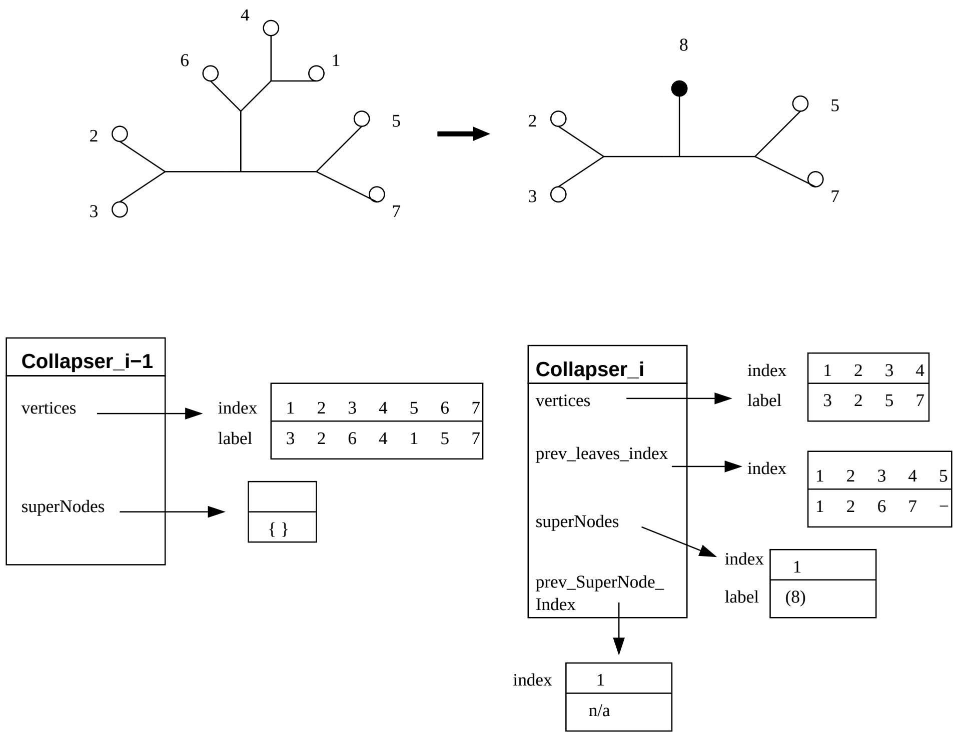


Figure 23: Reordering of vertex indices after one round of collapse.

such a case.

## A.3 Maintaining collapse history to perform $Exp()$

Consider the resulting set $Best(m, W^m)$ as a result of $m$ rounds of collapsing. Thus we have in total $m$ collapser objects, each keeping a legacy of the necessary data attributes from the previous rounds of collapse. In order to perform the $Exp()$ operation on the edges $e \in Best(m, W^m)$, it is necessary for us to maintain all $m$ collapser objects in memory, although it is not necessary to keep all of their attributes. For example, when going from $i$ to $i+1$ collapsing where $i+1 \leq m$, we require only the quartet set $W^i$ for constructing the next quartet set $W^{i+1}$. Thus at any given time, we only need to keep the current collapser and its parent collapser object's weighted quartet sets. However we must keep all the other attributes such as the $tree_i$, and the indices mapping the current and previous vertex indices since these information are required to perform the $Exp()$ operation on all the edges returned by $Best(m, W^m)$. Thus in the worst case, we need $3Mem(W)$ amount of memory, where $Mem(W)$ is the amount of memory required to store the initial quartet set $W$ on $\binom{n}{4}$ quartets, since during collapsing, we only need enough memory to store two other weighted quartet sets with equal or less number of quartets. Since the memory requirement for storing these quartets are of much higher order than the memory to store the other attributes of the collapser objects or to store the edges in the $Best(m, W_k^i)$ and $Best_{xy}(m, W_k^i)$ sets, then a rough estimate of the memory required to run $HC^*$ on set $S$ of $n$ elements is in the worst case $O(3\ \binom{n}{4}\ )$. Given that a quartet can be represented by four shorts (for the four labels) and three floats (for the scores of the three topologies) say taking 20 bytes in total, then the memory requirements is approximately in the worst case $O(60\ \binom{n}{4}\ )$ bytes.