Answer Set Programming or Hypercleaning: Where does the Magic Lie in Solving Maximum Quartet Consistency?

Fathiyeh Faghih and Daniel G. Brown

David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
e-mail: {ffaghihe, browndg}@uwaterloo.ca

Technical Report CS-2010-20

Abstract. Answer set programming (ASP) approach is an efficient approach for solving the maximum quartet consistency problem. We distinguish two factors that affect the efficiency of a recent ASP approach for solving maximum quartet consistency problem; answer set programming itself and a variety of preprocessing steps. In this paper, we propose a method for applying one of the preprocessing steps used in the ASP approach to an alternative algorithm. Our results show that the preprocessing step gives the main contribution to efficiency of the ASP approach for data sets with less than 20 taxa. We also identify an issue in the ASP approach for solving the maximum quartet consistency problem.

Key words: Maximum Quartet Consistency, Answer Set Programming, Edge Detection Strategy, Efficiency

1 Introduction

An important question in computational biology is to determine the evolutionary relationship of a set of taxa. There are different methods for phylogenetic reconstruction, such as distance methods, parsimony, quartet, likelihood and Bayesian approaches. Among these methods, quartet-based algorithms for reconstructing phylogenies have received a considerable amount of attention in the last two decades [1,2].

Given a taxon set S, each subset of four taxa of S is called a quartet of S. A quartet topology is an unrooted phylogeny of a quartet. A common approach in quartet methods is to estimate the phylogeny of each quartet and then infer a global phylogeny from them. The ideal case is when all quartet topologies are consistent with a single tree topology. In this situation, reconstructing the overall phylogeny is not complicated and can be done in $O(n^4)$ time [3], where n is is the number of the taxa. However, this is not always the case.

In practice, some quartets may be erroneous, so the whole set of quartet topologies may not be mutually consistent. We must then construct the whole

phylogeny based on inconsistent quartet topologies. One solution is to find a phylogeny that respects as many quartet topologies as possible: *Maximum Quartet Consistency (MQC)* problem, which is the *NP*-hard [4]. The algorithms proposed for MQC form two groups:

- Exact algorithms [5,6,1], which guarantee to find the phylogeny that is compatible with the maximum possible number of quartet topologies, but may require super-polynomial runtime.
- Near-optimal algorithms, which do not offer this guarantee, but may have a better efficiency.

Our focus in this paper is on exact algorithms that include a dynamic programming algorithm by Ben-Dor *et al.* [5], a fixed-parameter method proposed by Gramm and Niedermeier [6], and an Answer Set Programming (ASP) approach by Wu *et al.* [1]. There is also an approach using pseudo-boolean optimization and other logic formulations [7].

ASP is one of the most efficient approaches for optimally solving the MQC problem. ASP uses answer set programming [1], and the results show that it outperforms previous algorithms considerably. It also employs some preprocessing steps to make the algorithm more efficient. This strategy raises a question in mind: Is it answer set programming that makes the ASP approach so efficient, or is it the "magic" of the proposed preprocessing steps? In this paper, we answer this question by applying one of the preprocessing steps presented by Wu *et al.* to the Gramm and Niedermeier (GN) algorithm [6] (another algorithm for solving the MQC problem). We find that employing the preprocessing step can make the GN algorithm as efficient as the ASP approach. So, we can conclude that this step should have the main contribution in the efficiency of the ASP approach.

2 ASP Approach for Solving the MQC Problem

In this section, we briefly introduce the ASP approach proposed by Wu *et al.* [1].

2.1 General Approach

The authors of the ASP method propose a new representation for MQC, where solving the MQC problem becomes searching for the corresponding ultrametric matrix that satisfies the maximum number of given quartet topologies.

An ultrametric phylogeny is a rooted tree in which each internal node is labeled with a positive number, and on every path from the root to any leaf, the labels are strictly decreasing. [8]. The corresponding ultrametric matrix (M) is an $n \times n$ matrix for which $M(i, j)$ is the label of the least common ancestor (LCA) of taxa s_i and s_j. The least common ancestor of two leaves, s_i and s_j in a tree is the common ancestor of s_i and s_j farthest from the root.

Wu *et al.* prove the following theorem to show the correspondence of searching for the phylogeny and finding the corresponding ultrametric matrix.

Theorem 1. *A quartet topology $[s_i, s_j|s_k, s_l]$ is consistent with a phylogeny T, if and only if any ultrametric labeling scheme M of T satisfies:*

$$min\{M(i,k), M(j,l)\} > min\{M(i,j), M(k,l)\}$$

They also note that from an ultrametric matrix, we can find its tree:

Theorem 2. *Let M be an $n \times n$ ultrametric matrix, then there exists a unique ultrametric phylogeny with a labeling scheme M; moreover, this phylogeny can be constructed in $O(n^2)$ time [8].*

Theorem 1 shows how to check the consistency of an ultrametric labeling with a quartet. Theorem 2 shows that from the matrix, we can construct the phylogeny.

2.2 Answer Set Programming

Wu *et al.* formulate the problem of finding the ultrametric matrix in answer set programming. In this section, we want to give a brief introduction to answer set programming.

In answer set programming, we have a set of constrained Boolean variables, and the goal is to assign truth values to these variables in a way that the constraints are satisfied. Answer set programming solves the problems in a declarative way: the user specifies what constraints to be satisfied and the answer set programming system is responsible for the efficient implementation. The constraints in answer set programming are a set of rules of the form: $a \leftarrow b_1 \cdots b_m$,not $c_1 \cdots$ not c_n, where a, b_i and c_i are atoms and not c_i is called a not-atom. An atom has two possible truth-values: true and false. Intuitively, such a rule implies that, if all b_is are true and all c_js are false in a solution, then the atom a must be true in the same solution.

There are other types of constraint specification which can be used to ask for optimal solutions, e.g., by specifying $maximize[a_1 = w_1; \cdots; a_n = w_n]$, where w_is are integers representing the weights of the atoms to their left. This rule forces the generation of only those solutions in which the sum of the weights of atoms a_is is maximized.

In the MQC problem, our input is a set of n taxa and a set of quartet topologies. We also have n^2 variables for the entries of the corresponding ultrametric matrix M.

There are three groups of constraints: symmetry, ultrametricity, and quartet consistency. The symmetry constraints require that $M(i,j) = M(j,i)$. Remember that $M(i,j)$ corresponds to the label of the least common ancestor of taxa s_i and s_j, and the labels are integers between 1 and n.

In an ultrametric matrix, for every triplet (i,j,k) of distinct values, where i, j, among $M(i,j)$, $M(j,k)$, and $M(i,k)$, there are two equal values that must be greater than the third value. The ultrametricity constraints formulate this property on every subset of three taxa from the given set of taxa.

The quartet consistency constraints encode our goal of finding an ultrametric matrix consistent with the maximum number of quartets. Theorem 1 is used to write the quartet consistency rules in the answer set program.

The overall program has $O(n^4)$ constraints in $O(n^2)$ variables.

2.3 Edge Detection

In addition to their answer set programming, Wu *et al.* also propose three strategies to make this computation more efficient. Our work has focused on one of them, for detecting edges in the optimal solution before solving the answer set program.

They first generate a set of candidate edges and then find edges certain to be in the solution from the candidate edges. The candidate set is constructed using the hypercleaning algorithm [9].

Hypercleaning is a polynomial-time algorithm for constructing a (polynomial-sized) collection of bipartitions that are most strongly supported by quartet data. In this procedure, each edge is presented as a bipartition. A phylogenetic tree T contains the bipartition (X, Y), if there is an edge e in T such that $T-e$ consists of two trees, where one contains the taxa in X and the other contains those in Y.

Hypercleaning defines the following normalized metric to find out how much the phylogeny disagrees with the set of quartet topologies if it includes the bipartitions (X, Y).

$$\delta(Q, (X, Y)) = \frac{4|(Q_{(X,Y)} - Q)|}{|X|(|X|-1)|Y|(|Y|-1)}, \tag{1}$$

In this metric, $Q_{(X,Y)}$ is the set of quartet topologies of the form $xx'|yy'$ where $x, x' \in X$ and $y, y' \in Y$, and $|(Q_{(X,Y)} - Q)|$ is the defined distance from a set of quartets Q to a bipartition (X, Y) [9]. Here, a large value of δ indicates a bipartition in conflict with many quartets. Based on this metric, the following set of edges can be computed by increasing m.

$$Best(Q, m) = \left\{ (X, Y) \mid \delta(Q, (X, Y)) < \frac{2m}{|X||Y|} \right\} \tag{2}$$

By increasing m, the bipartitions chosen are less supported by the quartet data: edges conflicting with no quartets will be in $Best(Q, 0)$. Moreover, note that the edges in $Best(Q, 0)$ are also included in $Best(Q, 1)$ based on their definitions. Wu *et al.* use the edges in $Best(Q, 1)$ as their candidate edge set.

The ASP paper also proves that for each edge in this set, if the following equation holds, the edge must be included in the optimal solution.

$$2p_1 + (l-1)p_2 + (n-l-1)p_3 \leqslant (l-1)(n-l-1) \tag{3}$$

Suppose that (X, Y) is the corresponding bipartition of an edge with $|X| = l \geq 2$ and $|Y| = n - l \geq 2$. $Q_{X,Y}$ is defined to be the set of quartet topologies in the form of $[x, x'|y, y']$, where $x, x' \in X$ and $y, y' \in Y$. If the quartet topology $q \in Q$ for x, x', y, y' is not in the form of $[x, x'|y, y']$, then q is a quartet error across the bipartition (X, Y). The parameter p_1 is the number of quartet errors across the edge or the corresponding bipartition.

Fixing three taxa from Y, the subset of l quartet topologies from Q, where each quartet topology contains these three taxa and another taxon from X is called an l-subset with respect to (X, Y). For each l-subset, if ignoring the difference of the taxa from X gives rise to one unique quartet topology, then this l-subset is exchangeable on X; otherwise, it is nonexchangeable on X. The parameters p_2 and p_3 are the number of nonexchangeable l-subsets on X and the number of nonexchangeable $(n-l)$-subsets on Y, respectively. There is an issue in using this equation to find the edges in the optimal solution. We will discuss this equation, as well as the corresponding issue with it in more details in Section 6.

3 The Algorithm by Gramm and Niedermeier

We now apply this edge detection strategy to the algorithm by Gramm and Niedermeier [6], which we call the GN algorithm. In this section, we will give an introduction to the GN algorithm.

The main idea of the GN algorithm is to consider local conflicts and try to resolve them in order to construct the tree. A local conflict is a size-three set of quartet topologies that involves only five taxa. For such a set of topologies, if equation 4 does not hold, then it is not possible to construct a single tree consistent with all of them, and they form a local conflict.

$$[ab|cd] \in Q_S \Rightarrow [ab|ce] \in Q_S \text{ or } [ae|cd] \in Q_S \tag{4}$$

Gramm and Niedermeier also prove that a local conflict must always exist if a set of quartets are not consistent.

Lemma 1. *If we are given a set S of taxa, some taxon $f \in S$, and a complete set Q_S of quartet topologies that is not tree-consistent, then Q_S has at least one local conflict involving f.*

A set of quartet topologies Q is tree-consistent, if there exists a tree T such that for the set Q_T of quartet topologies induced by T; $Q \subseteq Q_T$ [6]. Simply speaking, the lemma says that it is sufficient to look for local conflicts involving a special taxon. A trick is also introduced in [6] for finding the local conflicts more efficiently.

The algorithm builds the conflict list C of local conflicts. By Lemma 1, it is sufficient to build a list of local conflicts containing some arbitrarily chosen taxon. Then, it uses a search tree to resolve local conflicts by searching through all the possibilities, recursively. At each possible resolution, the algorithm updates the conflict list. Whenever the conflict list is empty, we have found a solution, and the output is the list of quartets that are consistent with each other.

4 Applying the Preprocessing Step to the GN Algorithm

In this paper, we want to study the algorithm presented by Wu *et al.*, to see the contribution of the different proposed ideas to its efficiency. We apply the

preprocessing step to the GN algorithm. If application of the preprocessing step to the GN algorithm makes its computation as efficient as the ASP method, the preprocessing step plays the main role, not the answer set programming approach.

We implemented the GN algorithm in Java using a hash table data structure to store the quartet topologies. We also implemented the hypercleaning algorithm used in a preprocessing step by Wu *et al.*. We use a recursive procedure proposed by Zhang [10] to implement the hypercleaning algorithm, so as to return $Best(Q, 1)$.

In each recursion step of the procedure, one taxon is added to all edges found so far, and the base case of the procedure returns all possible bipartitions for a tree with three taxa. So, the main idea is to start with the trivial tree of three taxa and add one leaf at a time to all edges we have so far. For each edge, the constraint for being included in $Best(Q, 1)$ is also checked at the end of each recursion. We then check each edge in $Best(Q, 1)$ with equation 3 to find edges guaranteed to be included in the optimal solution.

For an edge e found by this preprocessing step, we know that all quartets inferred by this bipartition must be included in the optimal tree in the topology required by the edge. For each such bipartition, we find all of its quartets and change their topology in our database of quartet topologies for those that were inferred incorrectly. We mark all these quartets as unchangeable by the search procedure. So during the GN procedure for finding a set of consistent quartet topologies, these quartet topologies are not changed. This idea can lead to reduce the search space of the GN algorithm considerably.

Finally, we use the set of quartets at the end of the algorithm to construct the correct tree, using PhyloQuart [3].

For testing the ASP approach [1], we used the source code that the authors have made available. We used the code to generate the logic program for each set of quartets. For computing the stable model of each logic program, we used lparse and smodels [11].

Smodels is an implementation of the stable model semantics for logic programs. Smodels can be used either as a C++-library that can be called from user programs or as a stand-alone program together with a suitable front-end. The main front-end is lparse (`http://www.tcs.hut.fi/Software/smodels/`).

5 Results

We compared the results of the with the running time of the Gramm and Niedermeier algorithm with and without edge detection with that of the ASP approach.

5.1 Synthetic Data Generation

For the evaluation part of our study, we used synthetic data sets, using the method introduced by Wu *et al.* [1].

Given a set of n taxa, a phylogeny is generated by recursively joining randomly selected subtrees (through one edge in each subtree). The subtrees are selected from a set that initially contains only the one-node subtrees (each corresponding to one of the given taxa). When two subtrees are joined, they are replaced in the set by the newly generated subtree. This procedure yields a phylogeny on n taxa.

After generating a random phylogeny, the set of quartet topologies are derived from the phylogeny. After that, p percent of the $\binom{n}{4}$ quartets are picked and the corresponding topologies are altered, so that we will have a set of quartets with the number of quartet errors upper bounded by $\frac{p}{100}\binom{n}{4}$. The number of errors is not necessarily equal to $\frac{p}{100}\binom{n}{4}$, since some quartet topology alterations might give rise to a new compatible set of quartet topologies. Wu *et al.* [12] have given an $O(n^4 logn)$ algorithm that finds the MQC solution with high probability in this error model; recently, Brown and Truszkowski have given an $O(nlogn)$ algorithm with similar guarantees [13].

Every generated data set has two parameters, n (the number of taxa) and p (the percentage of altered quartet topologies). The error percentages that we used in our research are $p = 1\%$ and 10% . For every pair of parameters (n, p), we generated 10 data sets and reported the results as the average of running time on all of them.

5.2 Computational Results

Table 1 shows the results of our study. The available source code for the ASP paper does not work when the number of taxa is greater than 20 due to memory faults.

	(10,1%)	**(10,5%)**	**(15,1%)**	**(20,1%)**
GN	13 sec	212 sec	31 min	$>$ 94 min
GN + Edge Detection	0.277 sec	0.39 sec	6.76 sec	8.199 sec
ASP	0.2 sec	0.31 sec	1.34 sec	22.121 sec

Table 1. Running time for the GN algorithm, our proposed procedure, and the ASP approach

The results are in Table 1. It can be easily seen that the edge detection strategy is key to success here: adding the preprocessing to the GN algorithm reduces the running time considerably, especially as the number of taxa increases. Indeed, the running time of our proposed procedure is near to the running time of the ASP approach. Based on the data sets with fewer than 20 taxa, our conclusion is that the main contribution of the ASP approach efficiency is the edge detection strategy and not the answer set programming approach.

6 Issue with the Edge Detection Strategy

During our research, we found an issue with the edge detection strategy empirically that can be discussed theoretically too. Recovering the edges in the optimal tree using equation 3 only works well for small numbers of taxa, i.e. less than thirty taxa (with the error percentage of 1 percent). For large data sets, hypercleaning will recover edges in the optimal solution only if the error rate is very small compared to the number of quartets.

We can also investigate the problem from the the theoretical point of view. As mentioned, an equation is presented in [1] for checking whether an edge must be included in the optimal tree or not. For explaining the issue, we should review the equation in more details.

$$2p_1 + (l-1)p_2 + (n-l-1)p_3 \leqslant (l-1)(n-l-1)$$

The constants p_1, p_2, and p_3 increase by increasing the number of taxa and their increase is related to the number of errors in the quartet set. In particular, in the error model we use, the number of quartet errors for a given partition increases as n^4 (assuming that the two sides of the partition are not small). However, the right hand side of the equation increases as n^2, since it is the multiplication of the sizes of two partitions. Therefore, for the large number of taxa, the equation will only hold when the number of errors (not the percentage of errors) is very small comparing to the number of taxa, or if n is very small.

Moreover, the reported results in [1] on recovered edges seem surprising: they claim that by increasing the number of taxa, with the same error percentage, more edges are recovered.

7 Conclusion

ASP, as proposed by Wu *et al.*, is currently one of the most efficient approaches for solving maximum quartet consistency problem. In this paper, we studied this approach to see the contribution of different proposed ideas in its efficiency. We distinguish two parameters in the efficiency of the ASP approach: answer set programming and its preprocessing steps.

Our results show that for data sets with less than 20 taxa, the main factor of the ASP approach efficiency is the edge detection strategy and not the answer set programming. We also found an issue in the ASP approach empirically and explained it theoretically.

References

1. G. Wu, J. You, and G. Lin, "Quartet-based phylogeny reconstruction with answer set programming," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 139–152, 2007.
2. K. Strimmer and A. Von Haeseler, "Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies," *Molecular Biology and Evolution*, vol. 13, no. 7, pp. 964–969, 1996.
3. V. Berry and O. Gascuel, "Inferring evolutionary trees with strong combinatorial evidence," *Computing and Combinatorics*, pp. 111–123, 1997, Software available through http://www.lirmm.fr/ vberry/PHYLOQUART/phyloquart.html.
4. T. Jiang, P. Kearney, and M. Li, "Orchestrating quartets: approximation and data correction," in *focs*, p. 416, Published by the IEEE Computer Society, 1998.
5. A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg, "From four-taxon trees to phylogenies: the case of mammalian evolution," in *Proceedings of the second annual international conference on Computational molecular biology*, pp. 9–19, 1998.
6. J. Gramm and R. Niedermeier, "A fixed-parameter algorithm for minimum quartet inconsistency," *Journal of Computer and System Sciences*, vol. 67, no. 4, pp. 723–741, 2003.
7. A. Morgado and J. Marques-Silva, "A pseudo-boolean solution to the maximum quartet consistency problem," in *WCB08-Workshop on Constraint Based Methods for Bioinformatics*, 2008.
8. D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Univ Pr, 1997.
9. V. Berry, D. Bryant, T. Jiang, P. Kearney, M. Li, T. Wareham, and H. Zhang, "A practical algorithm for recovering the best supported edges of an evolutionary tree (extended abstract)," in *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 287–296, Society for Industrial and Applied Mathematics, 2000.
10. H. Zhang, "Design, implementation, and analysis of a novel quartet-based phylogenetic reconstruction method," 2000.
11. P. Simons, "Smodels: An implementation of the stable model semantics for logic programs." `http://www.tcs.hut.fi/Software/smodels`, 2000.
12. G. Wu, M. Kao, G. Lin, and J. You, "Reconstructing phylogenies from noisy quartets in polynomial time with a high success probability," *Algorithms for Molecular Biology*, vol. 3, no. 1, p. 1, 2008.
13. D. G. Brown and J. Truszkowski, "Fast error-tolerant quartet phylogeny algorithms," *ArXiv e-prints*, Oct. 2010.