Quartet-Based Phylogeny Reconstruction with Answer Set Programming

Gang Wu, Jia-Huai You, and Guohui Lin

Abstract—In this paper, a new representation is presented for the Maximum Quartet Consistency (MQC) problem, where solving the MQC problem becomes searching for an ultrametric matrix that satisfies a maximum number of given quartet topologies. A number of structural properties of the MQC problem in this new representation are characterized through formulating into answer set programming, a recent powerful logic programming tool for modeling and solving search problems. Using these properties, a number of optimization techniques are proposed to speed up the search process. The experimental results on a number of simulated data sets suggest that the new representation, combined with answer set programming, presents a unique perspective to the MQC problem.

Index Terms—Phylogeny, quartet, Maximum Quartet Consistency (MQC), Answer Set Programming (ASP), ultrametric matrix.

1 Introduction

Evolution is an important subarea of study in the biological sciences, where knowledge of the evolutionary history, or *phylogeny*, of the taxa under consideration plays a very important role in the studies. Evolution study started with taxon-specific information such as physical characteristics (morphological data). In the last few decades, with the availability of a large number of molecular sequences, the study has involved using this additional information to provide phylogenetic biologists with more accurate knowledge of the evolutionary history. It is now routine for biologists to conduct evolutionary analyses of large DNA and protein sequence data sets or even whole genomes. It is expected that such a study would be able to carry out more accurate phylogeny that models the course of evolution and speciation over time for the set of taxa under consideration.

In a phylogeny on a set of taxa, the leaves are labeled with the given taxa and the internal nodes represent extinct or hypothesized ancestors. There are *rooted* and *unrooted* phylogenies. In a rooted phylogeny, an edge specifies the parent-child relationship and the root represents a common ancestor of all the taxa. A rooted phylogeny is called *binary* or *resolved* if every internal node except the root has degree exactly 3 and the root has degree 2. In an unrooted phylogeny, there is no parent-child relationship specified for an edge and it is called *resolved* if every internal node has degree exactly 3. There are a lot of works on both rooted and unrooted phylogeny reconstruction, depending on the detailed biological applications. It is already known that rooted phylogenies and unrooted phylogenies can be easily transformed into each other [21], for example, by using an outgroup.

In this paper, we consider the reconstruction of rooted resolved phylogenies. We adopt the general notions from graph theory. For example, the *path* from node u to node v in the phylogeny T is a sequence of distinct nodes $u_0, u_1, \ldots, u_k$ such that $u_0 = u$, $u_k = v$, and, for each $i \in \{0, 1, \ldots, k-1\}$, (u_i, u_{i+1}) is an edge in T. If node u is on the path from root to node v, then u is called an *ancestor* of node v and v is called a *descendant* of node u.

1.1 Quartet-Based Phylogeny Reconstruction

In the last two decades, quartet-based methods for reconstructing phylogenies have received a considerable amount of attention in the computational biology community. Given a taxon set S, each subset of four taxa of S is called a *quartet* of S. An unrooted phylogeny (or topology) of a quartet is called its *quartet topology*. The quartet-based phylogeny reconstruction is to first build a phylogeny for every quartet and then try to infer an overall unrooted phylogeny for the whole set of taxa. After building a topology for every quartet, quartet-based methods all rely on some combinatorial algorithms to construct a phylogeny on the entire set of taxa. In the ideal case where all quartet topologies are "correct," namely, they agree with each other, the task of assembling an overall phylogeny is easy and can be done in $O(n^4)$ time [8], where $n = |S|$ is the number of taxa under consideration.

In practice, however, some quartet topologies might be ambiguous (and, thus, missing) or even erroneous and, thus, the given quartet topology set might be *incomplete*, i.e., it contains less than $\binom{n}{4}$ quartet topologies (if there is exactly one topology for each quartet, then the quartet topology set is *complete*) and might contain conflicting quartet topologies. This property complicates the phylogeny reconstruction problem but also raises the computational interests. Under the parsimony assumption, the goal is to construct a phylogeny that respects as many quartet topologies as possible. Such an optimization problem turns out to be hard [16]. Exhaustive search is generally infeasible as there are $\frac{(2n-5)!}{(n-3)!2^{n-3}}$ unrooted resolved phylogenies on n leaves to choose from [10]. A few attempts have been made to solve

• *The authors are with the Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada.*
E-mail: {wgang, you, ghlin}@cs.ualberta.ca.

Manuscript received 26 May 2005; revised 27 Nov. 2005; accepted 16 Jan. 2006; published online 9 Jan. 2007.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0053-0505.
Digital Object Identifier no. 10.1109/TCBB.2007.1008.

this optimization problem optimally. Ben-Dor et al. presented a dynamic programming algorithm that evaluates the number of quartet topologies that are consistent with a bipartition of the taxon set and thereby determines a phylogeny that satisfies a maximum number of quartet topologies [2]. The running time of the algorithm is $O(3^n n^4)$. The inconsistent quartet topologies with respect to a bipartition are referred to as *quartet errors* (across the bipartition), which are identified and changed in order to be compatible to other quartet topologies. In general, those given quartet topologies that are not respected by the output phylogeny are referred to as quartet errors, which need to be identified and changed. When the number of quartet errors is known ahead of time, then the fixed parameter algorithm proposed by Gramm and Niedermeier [13], which, for simplicity, is referred to as the GN algorithm in the sequel, would be able to detect and correct them and to return an associated phylogeny. The GN algorithm has a running time of $O(4^k n + n^4)$, where k is the number of quartet errors.

Another line of research has been carried out where near-optimal solutions are pursued instead of the optimal ones under some time constraint. To name a few, the heuristics of Sattath and Tversky [22] and Bandelt and Dress [1] combine some clustering procedures with pairwise similarity scores derived from the quartet topologies. One novel variation on the scoring approach is described by Ben-Dor et al. [2], where, instead of constructing a similarity score for clustering, they embed the n taxa as points in the n-dimensional euclidean space R^n using semidefinite programming and then apply a nearest neighbor clustering procedure to finish the task. Strimmer and von Haeseler designed the *Quartet Puzzling* heuristic [25] in which the tree building (or "puzzling") part works by 1) ordering the leaves arbitrarily, 2) constructing a phylogeny on the first four leaves, and then 3) adding new leaves one at a time by attaching it to the edge that gives the optimal quartet score. Dekker proposed another method for constructing phylogenies from quartet topologies and other subphylogenies using several quartet inference rules [7]. The *Short Quartet Method* [9] constructs phylogenies using several inference rules and greedy selection of quartet topologies. In the special case where the set of quartet topologies is complete, the problem of constructing a phylogeny to satisfy a maximum number of quartet topologies has been shown to admit a *polynomial time approximation scheme* (PTAS) [15]. The same group of researchers also proposed a number of *quartet cleaning* algorithms [3], [4], [15] that can tolerate varying degrees of errors presented in the quartet set. In particular, Berry et al. formalized the quartet cleaning algorithms into four categories: *global/local edge/vertex cleaning*, based on the type of quartet errors the algorithm trying to correct [4]. A step further, a *hypercleaning* heuristic algorithm is proposed in [3]. Instead of maximizing the number of consistent quartet topologies, hypercleaning constructs a new quartet distance model to measure the quartet errors and tries to minimize this distance. These quartet cleaning algorithms typically run in polynomial time and are well suited for the situation where the given quartet topology set is unweighted and almost tree-like [3], [15].

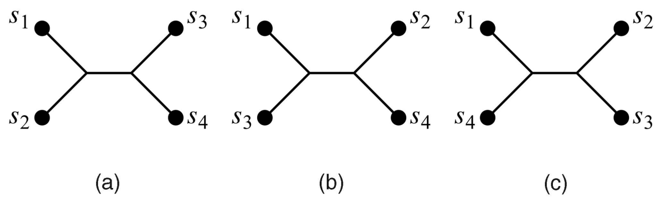


Fig. 1. The three possible topologies for quartet $\{s_1, s_2, s_3, s_4\}$.

1.2 Organization

We introduce in Section 2 the detailed definitions for a number of objects we are going to use in this paper. We then describe the combinatorial optimization problems we are trying to solve. In Section 3, a new representation is given for the phylogeny reconstruction problem to satisfy a maximum number of quartet topologies. In Section 4, we present a brief introduction to *Answer Set Programming* (ASP) and describe how the new representation of the MQC problem can be encoded as an answer set program, which can subsequently be run in an efficient ASP system, *Smodels* [23]. In Section 5, we present a number of nice structural properties of the new representation that can be taken advantage of in pruning the search space effectively. Section 6 summarizes our experimental results on a number of simulated data sets and various comparison results. Finally, we conclude the paper in Section 7.

2 Problem Descriptions

Given a taxon set S and a quartet $\{s_1, s_2, s_3, s_4\}$ of S, there are three possible topologies associated with the quartet, up to symmetry. These three quartet topologies are shown in Fig. 1. For simplicity, we use $[s_1, s_2 | s_3, s_4]$ to denote the quartet topology in which the path connecting s_1 and s_2 doesn't intersect the path connecting s_3 and s_4 (see Fig. 1a). The other two quartet topologies are $[s_1, s_3 | s_2, s_4]$ and $[s_1, s_4 | s_2, s_3]$. Given a phylogeny T on S, for every quartet, one may trim all the other nodes (including the root if the phylogeny is rooted) from T to obtain a topology for the quartet. Therefore, only one of the three possible quartet topologies can be present in the given quartet topology set Q. Suppose Q contains the quartet topologies built in the first step of a quartet-based phylogeny reconstruction, which can be done by various quartet inference methods. If there exists one phylogeny T on S such that the quartet topology q in Q is the same as the one derived from T, then we say T *satisfies* q or q is *consistent* with T; if there exists one phylogeny T satisfying all quartet topologies in Q, then we say Q is *compatible* and T is the phylogeny *associated* with Q.

The recognition problem to determine if a given set Q of quartet topologies on $S = \{s_1, s_2, \ldots, s_n\}$ is compatible or not is called the *Quartet Compatibility Problem* (QCP).

INPUT: A set Q of quartet topologies on $S = \{s_1, s_2, \ldots, s_n\}$.
QUESTION: Is Q compatible? Equivalently, is there a phylogeny T on S that satisfies all the quartet topologies in Q?

It is known that, when Q is complete, QCP can be answered in polynomial time and, if Q is compatible, the associated phylogeny is unique and can be constructed within the same time [8]. However, if Q is incomplete, then QCP is NP-complete [24]. The more computationally interesting problem is, when Q is not compatible, to construct a phylogeny satisfying as many quartet topologies as possible.

MAXIMUM QUARTET CONSISTENCY (MQC):
INPUT: A set Q of quartet topologies on $S = \{s_1, s_2, \ldots, s_n\}$.
GOAL: Find a phylogeny T on S that satisfies a maximum number of quartet topologies in Q.

The MQC problem is the target problem in this paper. When Q is complete, MQC has been proven to be NP-hard yet admits a PTAS [15]. When Q is incomplete, MQC has been proven to be MAX SNP-hard [15]. In this paper, we assume the given quartet topology set Q is complete. In the following, we will state some equivalent problems to MQC, one of which can be readily solved by answer set programming.

2.1 Ultrametric Phylogeny

Given a set of taxa, $S = \{s_1, s_2, \ldots, s_n\}$, and a rooted phylogeny T on S, the *least common ancestor* of two leaf nodes s_i and s_j in T is the common ancestor of s_i and s_j farthest away from the root, denoted as $\mathrm{LCA}(s_i, s_j)$. $\mathrm{LCA}(s_i, s_j)$ can also be interpreted as the internal node that is on the path connecting s_i and s_j and is the closest to the root.

A *labeling scheme* for the rooted phylogeny T is a mapping from the set of internal nodes in T to the set of integers $\{1, 2, \ldots, n-1\}$. Note that, since T is a binary tree, there are exactly $(n-1)$ internal nodes in T. Two internal nodes can be labeled by the same number in the set $\{1, 2, \ldots, n-1\}$, that is, the mapping is not necessarily a bijection—in fact, to speed up the search process, one of our goals is to use a small number of labels and, thus, one label is usually shared by multiple internal nodes. Let $M(i,j)$ denote the label of the internal node $\mathrm{LCA}(s_i, s_j)$. Without loss of generality, $M(i,i)$ is set to 0 for every s_i. A labeling scheme is *ultrametric* if, along any root to leaf path, the labels of the internal nodes on the path are strictly decreasing [14]. One phylogeny, together with an ultrametric labeling scheme, is called an *ultrametric phylogeny* [14]. Note that this definition of ultrametric phylogeny differs from the concept of "ultrametric tree," which refers to a rooted edge weighted tree satisfying the three point condition.

Let M be an $n \times n$ symmetric matrix with its entry values taken from the set $\{0, 1, \ldots, n-1\}$. M is *ultrametric* if $M(i,i) = 0$ for all i, $M(i,j) > 0$ for $i \neq j$, and, for every triplet (i,j,k), there are two equal values among $M(i,j)$, $M(j,k)$, and $M(i,k)$ and they are greater than the third value. It is not difficult to see the following property:

Theorem 2.1. *Given a set of taxa $S = \{s_1, s_2, \ldots, s_n\}$ and a rooted phylogeny T on S, there exists an ultrametric labeling scheme for T and the resultant labeling matrix M is an ultrametric matrix.*

3 PROPERTIES OF MQC AND A NEW REPRESENTATION

Recall that our goal is to construct a phylogeny T that satisfies a maximum number of quartet topologies in the given set Q. In this section, we will first examine some structural properties of this optimal phylogeny T. Using these structural properties, we will be able to transform the phylogeny searching problem into an ultrametric matrix searching problem. The ultrametric matrix searching problem can then be written into an answer set program, which can be solved by an existing efficient ASP solver Smodels [23]. We want to remark that, nonetheless, our ultimate goal is to design a quartet-specific ASP solver with the current set of results being our first step toward this goal.

Theorem 3.1. *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:*

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

Proof. It should be noted that, in fact, the ultrametric labeling scheme satisfies:

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

Nonetheless, we choose the stated inequality to minimize the number of comparisons in computation since the complete one requires two extra comparisons, which are unnecessary.

In phylogeny T, let T' denote the minimal (rooted, resolved) subphylogeny containing leaf nodes s_i, s_j, s_k, and s_l with all the other leaf nodes removed and degree 2 internal nodes (except the root of T') ignored. It is clear now that, for at least one of the two pairs (s_i, s_j) and (s_k, s_l), the taxa are sibling leaves in T'. Without loss of generality, assume that (s_i, s_j) is such a pair, that is, s_i and s_j are siblings in T'. It follows that, for every pair of leaf nodes among (s_i, s_k), (s_i, s_l), (s_j, s_k), and (s_j, s_l), its least common ancestor must be an ancestor of $\mathrm{LCA}(s_i, s_j)$. Therefore, the minimum among $M(i,k)$, $M(i,l)$, $M(j,k)$, and $M(j,l)$ must be strictly greater than $M(i,j)$. This proves the "only if" part. The "if" part can be proven by a simple argument by contradiction and it is omitted. □

The following theorem is the inverse of Theorem 1:

Theorem 3.2 [14]. *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.*

An $n \times n$ ultrametric matrix M *satisfies* a quartet topology $[s_i, s_j | s_k, s_l]$ or the quartet topology is *consistent* with M if $\min\{M(i,k), M(j,l)\} > \min\{M(i,j), M(k,l)\}$ holds. The next two theorems we are going to introduce say that the construction of a phylogeny that satisfies a maximum number of quartet topologies is equivalent to the problem of searching for an ultrametric matrix to satisfy the same maximum number of quartet topologies.

Corollary 3.3. *An $n \times n$ ultrametric matrix M satisfies a quartet topology if and only if its associated ultrametric phylogeny satisfies the quartet topology.*

Theorem 3.4. *Given a set Q of quartet topologies on a set of taxa $S = \{s_1, s_2, \ldots, s_n\}$, Q is compatible if and only if there is an $n \times n$ ultrametric matrix M on S that satisfies all the quartet topologies in Q.*

Proof. The theorem follows easily from Theorems 2.1, 3.1, 3.2, and Corollary 3.3. □

Similarly, the following theorem holds:

Theorem 3.5. *Given a set Q of quartet topologies on a set of taxa $S = \{s_1, s_2, \ldots, s_n\}$ and an ultrametric phylogeny T on S, T satisfies k quartet topologies in Q if and only if its corresponding ultrametric matrix M on S satisfies the same k quartet topologies in Q.*

4 Formulating MQC into Answer Set Programming

According to Theorem 3.5, the MQC problem is equivalent to the problem of finding an ultrametric matrix that satisfies a maximum number of quartet topologies. Finding an $n \times n$ ultrametric matrix is to assign values from $\{0, 1, 2, \ldots, n-1\}$ to matrix entries, which can then be formulated into an answer set program. In this section, we briefly introduce answer set programming (ASP) and then provide a complete ASP formulation of the MQC problem.

4.1 Answer Set Programming

Answer set programming is a new form of logic programming with strength in solving constraint problems in a declarative way [20]. In ASP, a given problem is expressed as a logic program and each answer set of the program corresponds to a solution to the given problem. Declarative knowledge representation stems from the fact that, while the user specifies what should be satisfied, an implemented system is responsible for efficient computation. In this subsection, we provide a brief introduction to ASP.

In ASP, an *atom* is of the form $p(a_1, a_2, \ldots, a_n)$, where $p(\cdot)$ is an n-ary predicate symbol and $a_1, a_2, \ldots, a_n$ are *terms*. A term is either a constant, a variable, or a function $f(t_1, t_2, \ldots, t_m)$, where $t_1, t_2, \ldots, t_m$ are terms. An atom without variables is called a *ground* atom. If $p(\cdot)$ is a 0-ary predicate, the atom then can simply be expressed as p. A *logic program* in this context (also called *an answer set program* or just *a program*) is a finite collection of rules of the form

$$a \leftarrow b_1, \ldots, b_m, \texttt{not}\ c_1, \ldots, \texttt{not}\ c_n,$$

where a, b_i, and c_i are function-free *atoms* and $\texttt{not}\ c_i$ is called a *not-atom*. Atoms and not-atoms are referred to as *literals*. An atom has two possible truth values, *true* and *false*. In the above rule, atom a is the *head* of the rule, which when empty means false. To the right of sign "$\leftarrow$ " is the *body* of the rule, which, when empty, means the head is a fact. Intuitively, such a rule says that, if all b_is are true and all c_js are false in a solution, then atom a must be true in the same solution.

A program is *ground* if every atom in it is ground. An answer set program is typically instantiated to a ground program for the computation of answer sets. In the following, we assume atoms are function-free and programs are ground. Answer sets are defined as *stable models* [11]. Let S be a set of atoms and P a program. An atom a is true if and only if it is in S and is false otherwise. A rule in P is *satisfied* by S if either its head is in S or some literal in the rule body is false. Informally, S is a stable model of P if every rule in P is satisfied by S and, for any atom a in S, there is a rule "$a \leftarrow b_1, \ldots, b_m, \texttt{not}\ c_1, \ldots, \texttt{not}\ c_n$" in P such that all b_is are in S and none of the c_js is in S, by a *noncircular* reasoning. For example, rule "$a \leftarrow a$" cannot be used to justify a even if a is already in S. In other words, a stable model is a set of atoms that satisfy every rule in the program and every atom in the stable model has a "reason" to be there.

ASP systems have made enumeration and selection easier with several extensions, one of which is called the *cardinality rule* in the form [23]:

$$L\{a_1, a_2, \ldots, a_m\}U \leftarrow body,$$

where *body* is shorthand for a conjunction of literals. This rule says that if all the literals in the body are in an answer set, then the rule is satisfied by the answer set only when the number of atoms among $a_1, a_2, \ldots, a_m$ in the answer set is between integers L and U, inclusive. One can think of a cardinality rule as a high-level language construct supported by the underlying language semantics.

In general, an answer set program can be translated into a SAT instance and solved by an existing SAT solver [18]. It can also be solved directly by an answer set solver. In this latter category, one of the most efficient implementations is the *Smodels* system [23], which theoretically can be used to solve all NP problems.

There are a number of reasons for us to choose Smodels in our research work. First, Smodels comes with useful tools and is more user-friendly in comparison to SAT solvers. For example, Smodels' *Lparse* grounds a function-free program efficiently and, thus, allows the user to write compact programs. The grounding process embodies a number of preprocessings in constraint handling. Second, Smodels has both language constructs and an efficient implementation for the support of optimization. In Smodels, one can ask for optimal solutions, e.g., by specifying

$$\text{maximize } [a_1 = w_1, \ldots, a_n = w_n],$$

where w_is are integers representing the weights of the atoms to their left. This specification forces Smodels to generate only those stable models in which the sum of the weights of atoms a_is is maximized (Smodels does this by branch-and-bound). When all the weights are 1 (i.e., $w_i = 1$ for all $i \in [1..n]$), Smodels computes the stable models containing a maximum number of a_is. Built-in constructs for optimization can also be found in SAT solvers such as Maxsat [5] and MaxSolver [26].

More importantly, Smodels is among a very few ASP/SAT solvers that come with a full implementation of the more powerful constraint propagation mechanism called *lookahead*: At any choice point, each (unassigned) atom is assumed a

TABLE 1
Formulating the MQC Problem into a Constraint Programming Problem

INPUT:
• A set of n taxa $S = \{s_1, s_2, \ldots, s_n\}$. n^2 variables $M(i,j)$, where $1 \leq i,j \leq n$. The domain of each variable $M(i,j)$, where $i \neq j$, is $\{1, 2, \ldots, n-1\}$; $M(i,i) = 0$, for all i.
• A set Q of quartet topologies on S. Each quartet topology $[s_i, s_j \vert s_k, s_\ell]$ is transformed into a quartet consistency constraint $quar(i,j,k,\ell)$.
CONSTRAINTS:
• Symmetry Constraint: $M(i,j) = M(j,i)$, for all $1 \leq i,j \leq n$, i,j distinct;
• Ultrametric Constraint: $ultra(i,j,k)$, for all $1 \leq i,j,k \leq n$, i,j,k distinct;
• Quartet Consistency Constraint: $quar(i,j,k,\ell)$, for quartet topology $[s_i, s_j \vert s_k, s_\ell] \in Q$.
GOAL:
• Find a value for $M(i,j)$ for all $1 \leq i,j \leq n$, such that all the symmetry and ultrametric constraints are satisfied and the number of satisfied quartet consistency constraints is maximized.

truth value and, if a contradiction is derived, the atom is assigned with the opposite truth value. In this way, the truth value of such an atom is obtained by constraint propagation, not by search. Due to the high overhead, no leading SAT solvers (such as *Sato* [28], *zChaff* [19], etc.) employ this mechanism. Nevertheless, we found in our experiments that lookahead is particularly effective in solving the MQC problem under our formulation. This was verified by running our programs in Smodels, with lookahead turned off, on the data sets used in our experiments. Typically, the average running time without lookahead on the data sets of 15 taxa is 20 minutes, while the same problem instances can be solved within seconds. Theoretically, without lookahead, the constraint propagation in Smodels is similar to *unit propagation* in SAT solvers. The computational results showed that lookahead in Smodels prunes a significant amount of search space and, as a result, can make search hundreds of times faster. In fact, we believe this is one of the main factors that makes our programs outperform the other approaches. In this regard, one of the contributions of this paper is the identification of a practical constraint program for which the high overhead of lookahead pays off.

Constraint satisfaction is another formalism for solving constraint problems. The MQC problem under the ultrametric matrix representation can also be formulated as a constraint satisfaction problem (CSP) and solved by a CSP solver. Popular CSP solvers are currently almost all based on Constraint Logic Programming (CLP), where the user has to carefully exercise the "control" of a program. In comparison, the ASP and SAT users only need to express the logic correctly and let the solver search for solutions. With regard to search efficiency, all CSP solvers employ the constraint propagation mechanism that enforces *arc-consistency* during search, which requires that any instantiation of one variable can be consistently extended to any of the unassigned variables. It is known that lookahead (as implemented in Smodels) is more powerful than arc-consistency in pruning the search space [27]. A more general notion of consistency is called *k-consistency*, which, roughly speaking, requires that any $k-1$ consistent instantiation can be extended to all kth variables. In the literature, higher-level consistencies are usually considered too expensive. For the constraint propagation mechanism (known as *singleton* arc-consistency) that has the same pruning power as lookahead, while the former is considered too expensive, the latter has been a key to our success.

4.2 Problem Formulation

Based on Theorem 3.5, the MQC problem can be described as shown in Table 1 (note that all the values are integers). In the following, we will show how this problem description can be presented to Smodels so that the answer sets computed by Smodels correspond to the solutions to the MQC problem.

We now give explanations on the ultrametric constraints and quartet consistency constraints and then write them into ASP rules. By definition, for every triplet (i,j,k), where i,j,k are distinct and satisfy $1 \leq i,j,k \leq n$, among $M(i,j)$, $M(j,k)$, and $M(i,k)$, there are two equal values that must be greater than the third value. We denote the ultrametric constraint involving these three indices as $ultra(i,j,k)$. It is easy to verify that an ultrametric constraint $ultra(i,j,k)$ is satisfied if and only if one of the following three constraints is satisfied:

- $M(i,j) = M(i,k) > M(j,k)$,
- $M(i,j) = M(j,k) > M(i,k)$, or
- $M(j,k) = M(i,k) > M(i,j)$.

In the answer set program, we use atom $m(i,j,k)$ to denote the fact that $M(i,j) = k$. The following cardinality rule expresses that $M(i,j)$ has exactly one solution:

$$1\{m(i,j,k) \mid k = 1, 2, \dots, n-1\}1.$$

Similarly, we use atom $equal(i,j,k,\ell)$ to denote the fact that $M(i,j) = M(k,\ell)$:

$$\begin{array}{lcl} equal(i,j,k,\ell) & \leftarrow & m(i,j,1), m(k,\ell,1) \\ equal(i,j,k,\ell) & \leftarrow & m(i,j,2), m(k,\ell,2) \\ \dots & \dots & \dots \\ equal(i,j,k,\ell) & \leftarrow & m(i,j,n-1), m(k,\ell,n-1) \end{array}$$

and atom $gter(i,j,k,\ell)$ to denote the fact that $M(i,j) > M(k,\ell)$:

$$\begin{array}{lcl} gter(i,j,k,\ell) & \leftarrow & m(i,j,2), m(k,\ell,1) \\ gter(i,j,k,\ell) & \leftarrow & m(i,j,3), m(k,\ell,1) \\ gter(i,j,k,\ell) & \leftarrow & m(i,j,3), m(k,\ell,2) \\ gter(i,j,k,\ell) & \leftarrow & m(i,j,4), m(k,\ell,1) \\ \dots & \dots & \dots \\ gter(i,j,k,\ell) & \leftarrow & m(i,j,n-1), m(k,\ell,n-2). \end{array}$$

It follows that an ultrametric constraint $ultra(i,j,k)$ can be written as:

$$\begin{array}{lcl} ultra(i,j,k) & \leftarrow & equal(i,j,i,k), gter(i,j,j,k) \\ ultra(i,j,k) & \leftarrow & equal(i,j,j,k), gter(i,j,i,k) \\ ultra(i,j,k) & \leftarrow & equal(i,k,j,k), gter(i,k,j,k). \end{array}$$

According to Corollary 3.3, one quartet topology $[s_i, s_j | s_k, s_\ell]$ is satisfied if and only if at least one of the following two constraints is satisfied:

- $M(i,k) > M(i,j)$ and $M(j,\ell) > M(i,j)$;
- $M(i,k) > M(k,\ell)$ and $M(j,\ell) > M(k,l)$.

Such a quartet consistency constraint $quar(i,k,j,\ell)$ can be written as:

$$\begin{array}{lcl} quar(i,j,k,\ell) & \leftarrow & gter(i,k,i,j), gter(j,\ell,i,j) \\ quar(i,j,k,\ell) & \leftarrow & gter(i,k,k,\ell), gter(j,\ell,k,\ell). \end{array}$$

Finally, we can use a *maximize* rule to find a stable model that contains a maximum number of quartet consistency atoms $quar(i,j,k,\ell)$:

$$\text{maximize } \big[quar(i,j,k,\ell) = 1 \mid [s_i, s_j \mid s_k, s_\ell] \in Q\big].$$

In the supplementary materials (http://www.cs.ualberta.ca/~ghlin/src/WebTools/quartet.php), we have made available a complete answer set program formulated for a small instance of the MQC problem. The instance was constructed out of a pairwise distance matrix on 10 taxa, where the quartet topologies were inferred using the four-point method [8]. The solution to the answer set program by Smodels is also included, from which a phylogeny was written and can be readily fed to TreeView (http://taxonomy.zoology.gla.ac.uk/rod/treeview.html).

5 Strategies to Speed Up the Computation

We present three speedup strategies specific to our answer set program in the following three subsections, each of which takes advantage of some structural properties of the optimal phylogeny. Most of these strategies should be implemented prior to writing the actual answer set program, while being considered as preprocessing to produce extra constraints for the writing. Our experimental results show that they all help reduce the running time significantly.

5.1 Breaking the Symmetry

Symmetry breaking might not be quartet specific but, rather, ultrametric matrix specific. The observation is that an ultrametric matrix M is symmetric and, therefore, instead of putting the symmetry as constraints, we would rather use it to reduce the number of atoms in the answer set program. To this purpose, only $M(i,j)$ with $1 \le i < j \le n$ becomes a variable, which gives only $\frac{1}{2}(n^2 - n)$ variables at the end. Consequently, we remove all symmetry constraints from the constraint set. Similarly, we would only consider ultrametric constraints $ultra(i,j,k)$ such that $1 \le i < j < k \le n$ and quartet consistency constraints $quar(i,j,k,\ell)$ such that $1 \le i < j \le n$, $1 \le k < \ell \le n$, and $1 \le i < k \le n$.

5.2 Reducing the Domain Sizes

We define the *height* of an internal node v in a rooted phylogeny to be the maximum number of internal nodes along any path from v to a leaf node in the subtree rooted at v, including v. For any rooted phylogeny T, one natural ultrametric labeling scheme is to label each internal node by its height. Therefore, the domain of variables $M(i,j)$ for the target ultrametric matrix can be represented as $\{1, 2, \dots, h\}$, where h denotes the height of the root. What complicates the search of the target ultrametric matrix is that we do not know the exact value for h in advance and have to set h to be $(n-1)$, which is the largest possible number of internal nodes for a rooted phylogeny on n taxa. Observe that the quartet consistencies with respect to a rooted phylogeny will not change when the root of the phylogeny is changed. We can take advantage of this property to move the root to a proper place so that the height of the root becomes less than $(n-1)$.

Theorem 5.1. *Given a set Q of quartet topologies on taxon set $S = \{s_1, s_2, \dots, s_n\}$, there exists a rooted phylogeny T that satisfies a maximum number of quartet topologies in Q and the height of the root is at most $\lceil \frac{n}{2} \rceil$.*

Proof. The proof is done by rerooting phylogeny T, if the height of its root is greater than $\lceil \frac{n}{2} \rceil$. We first discard the root of T to get an unrooted phylogeny denoted as T'. Then, we root T' on the edge that is in the middle of a longest path in T'. In such a way, we obtain a new rooted phylogeny T' that satisfies the same number of quartet topologies as T and every path from the root to a leaf node contains at most $\lceil \frac{n}{2} \rceil$ internal nodes. □

From Theorem 5.1, we conclude that, in the search for a target ultrametric matrix, we can limit the domain of each matrix variable $M(i,j)$ to $\{1, 2, \dots, \lceil \frac{n}{2} \rceil\}$. Furthermore, since setting the matrix variable $M(i,j)$ to be the height of $\mathrm{LCA}(s_i, s_j)$ is an ultrametric labeling, we can further reduce the domain of $M(i,j)$ if we have knowledge of the height of $\mathrm{LCA}(s_i, s_j)$. For instance, if the height of $\mathrm{LCA}(s_i, s_j)$ is greater than k_1 ($k_1 \ge 1$) and less than k_2 ($k_2 \le \frac{n}{2}$) in an optimal phylogeny, we can set the domain for $M(i,j)$ to be $\{k_1+1, k_1+2, \ldots, k_2-1\}$. The rest of this section is devoted to obtaining better k_1 and k_2 to narrow down the search space.

For this purpose, we need to introduce some definitions. Note that, before we move on to the writing of the answer set program, we may take advantage of some existing fast quartet-based phylogeny construction heuristics (such as hypercleaning [3]) to get a near-optimal phylogeny on the input set Q. The number of quartet topologies in Q conflicting with this near-optimal phylogeny is an upper bound on the number of quartet errors for the MQC problem. In other words, suppose the near-optimal phylogeny dissatisfies E_{apx} quartet topologies in Q, then any optimal phylogeny will dissatisfy at most E_{apx} quartet topologies in Q. Based on the value of E_{apx}, we can calculate an upper bound on the number of quartet errors involving taxon s_i or s_j.

A *local conflict* [13] is a set of three incompatible quartet topologies on a subset of exactly five taxa. For example, $\{[s_a, s_b|s_c, s_d], [s_a, s_c|s_b, s_e], [s_a, s_c|s_d, s_e]\}$ is a local conflict. If we change the quartet topology on $\{s_a, s_b, s_c, s_d\}$ from $[s_a, s_b|s_c, s_d]$ to $[s_a, s_c|s_b, s_d]$ in Q, then the above local conflict is *resolved*. One may check that there are multiple ways to resolve a local conflict and our search goal can be rephrased as to change a minimum number of quartet topologies to resolve all local conflicts.

Given Q, we generate the list of all local conflicts on Q and let C denote the number of local conflicts involving none of s_i and s_j. From Lemma 6 in [13], we need to change at least $\frac{C}{6(n-4)}$ quartet topologies to resolve these C local conflicts. Therefore, an upper bound on the number of quartet errors on (s_i, s_j) is $U_{i,j} = E_{apx} - \frac{C}{6(n-4)}$. In other words, to get an optimal phylogeny, we need to change at most $U_{i,j}$ quartet topologies involving s_i or s_j or both.

For a pair of taxa (s_i, s_j) and a quartet topology q involving both of them (and two other taxa), *q conflicts* with pair (s_i, s_j) if q is not in the form of $[s_i, s_j|*, *]$. For pair (s_i, s_j), "ignoring the difference of s_i and s_j" means we treat s_i and s_j as a common taxon. For two quartet topologies on $\{s_i, s_a, s_b, s_c\}$ and $\{s_j, s_a, s_b, s_c\}$, respectively, if ignoring the difference of s_i and s_j gives rise to one unique quartet topology, then these two quartet topologies are *exchangeable* on pair (s_i, s_j); otherwise, they are *nonexchangeable* on pair (s_i, s_j). Let p_1 denote the number of quartet topologies not conflicting with (s_i, s_j), p_2 denote the number of quartet topologies conflicting with (s_i, s_j), p_3 denote the number of exchangeable pairs on (s_i, s_j), and p_4 denote the number of nonexchangeable pairs on (s_i, s_j). We need the following lemma to bound these p-values:

Lemma 5.2. *Given a complete quartet topology set Q on taxon set $S = \{s_1, s_2, \ldots, s_n\}$ and a phylogeny T on S rooted as in the proof of Theorem 5.1, if the height of $\mathrm{LCA}(s_i, s_j)$ in T is k, then the number of taxa in the subtree rooted at $\mathrm{LCA}(s_i, s_j)$ is at least $k+1$ but at most $\min\{n-k, 2^k\}$.*

Proof. We exclude the case where $\mathrm{LCA}(s_i, s_j)$ is the root. Since the height of $\mathrm{LCA}(s_i, s_j)$ is k and the subtree rooted at $\mathrm{LCA}(s_i, s_j)$ is binary, there are at least $k+1$ but at most 2^k taxa in the subtree.

Since phylogeny T is rooted at the middle of its longest path, there is at least one internal node outside of the subtree and its height is at least $k-1$. It follows that there are at least k taxa outside of the subtree. In other words, there are at most $n-k$ taxa in the subtree. This proves the lemma. □

Theorem 5.3. *Given a complete quartet topology set Q on taxon set $S = \{s_1, s_2, \ldots, s_n\}$ and a phylogeny T on S rooted as in the proof of Theorem 5.1, if the height of $\mathrm{LCA}(s_i, s_j)$ in T is k ($1 < k \le \lceil \frac{n}{2} \rceil$), then the following inequalities hold:*

- $\binom{n-\overline{k}}{2} \le p_1 \le \binom{k-1}{2} + \binom{n-k-1}{2}$;
- $(k-1)(n-k-1) \le p_2 \le \binom{\overline{k}-2}{2} + (\overline{k}-2)(n-\overline{k})$;
- $p_3 \ge \binom{n-\overline{k}}{3}$;
- $p_4 \le \binom{\overline{k}-2}{3} + \binom{\overline{k}-2}{2}(n-\overline{k})$,

where $\overline{k} = \min\{n-k, 2^k\}$.

Proof. Since the height of $\mathrm{LCA}(s_i, s_j)$ in T is k, there are at least $n-\overline{k}$ taxa outside of the subtree T_{ij} rooted at $\mathrm{LCA}(s_i, s_j)$. Therefore,

$$\binom{n-\overline{k}}{2} \le p_1 \text{ and } p_3 \ge \binom{n-\overline{k}}{3}.$$

Let S_{ij} denote the taxon set in T_{ij} and let $m = |S_{ij}| - 2$. For each quartet $\{s_i, s_j, s_a, s_b\}$, if one of s_a and s_b is in S_{ij} and another one is in $S - S_{ij}$, then the quartet topology for $\{s_i, s_j, s_a, s_b\}$ must conflict on (s_i, s_j). Therefore, if the quartet topology for $\{s_i, s_j, s_a, s_b\}$ does not conflict on (s_i, s_j), then s_a and s_b should be both in S_{ij} or both in $S - S_{ij}$. In other words,

$$p_1 \le \binom{m}{2} + \binom{n-2-m}{2} = \frac{2m^2 - (2n-4)m + (n-2)(n-3)}{2}.$$

Since $k-1 \le m \le \overline{k}-2 \le n-k-2$, one can check that the right-hand side of the above inequality gets its maximum when $m = k-1$ and, subsequently,

$$p_1 \le \binom{k-1}{2} + \binom{n-k-1}{2}.$$

This proves the first and the third items.

For the same reason, we conclude that

$$\binom{m}{2} + m(n-2-m) \ge p_2 \ge m(n-2-m),$$

the right-hand side gets its minimum $(k-1)(n-k-1)$ when $m = k-1$, and the left-hand side gets its maximum $\binom{\overline{k}-2}{2} + (\overline{k}-2)(n-\overline{k})$ when $m = \overline{k}-2$. This proves the second item.

Given two quartet topologies on $\{s_i, s_a, s_b, s_c\}$ and $\{s_j, s_a, s_b, s_c\}$, respectively, if they are nonexchangeable, then one of the following two cases happens: 1) All three taxa s_a, s_b, s_c are in S_{ij} or 2) two taxa of s_a, s_b, s_c are in S_{ij} and the third one is in $S - S_{ij}$. Therefore,

$$p_4 \leq \binom{m}{3} + \binom{m}{2}(n-2-m) = \frac{m(m-1)(-2m+3n-8)}{6},$$

where the right-hand side gets its maximum when $m = \overline{k} - 2$. In other words,

$$p_4 \leq \binom{\overline{k}-2}{3} + \binom{\overline{k}-2}{2}(n-\overline{k}).$$

This proves the fourth item. □

With the upper bound $U_{i,j}$ on the number of quartet errors on pair (s_i, s_j) computed, we can calculate an upper bound k_2 and a lower bound k_1 according to Theorem 5.3 for the height of LCA(s_i, s_j) or, equivalently, the domain of ultrametric variable $M(i,j)$ can be narrowed down to $[k_1, k_2]$ (from the trivial one $[1, \lceil \frac{n}{2} \rceil]$ by Theorem 5.1).

5.3 Reducing the Number of Taxa

Given a taxon set S, a *bipartition* is a pair of nonempty subsets (X, Y) such that $X \cup Y = S$ and $X \cap Y = \emptyset$. For a phylogeny T on S, removing one edge e from T gives rise to two subtrees, of which the taxon sets X_e and Y_e form a bipartition (X_e, Y_e). In this sense, we do not distinguish the edge e and the bipartition (X_e, Y_e). Reducing the number of taxa is done through proving that some bipartition (X, Y) is in an optimal phylogeny. Consequently, one may turn to constructing optimal phylogenies on $X \cup y$ and $x \cup Y$, respectively, where x and y are the supertaxa to replace X and Y, and then combine them into an optimal phylogeny on S (through overlapping x with y). We note that we are interested in nontrivial cases where $|X| \geq 2$ and $|Y| \geq 2$ (a dynamic programming based on a similar idea can be found in [2]). We also remark that the extent of success of this optimization strategy depends on the quality of the quartet topology set Q. In what follows, we first give a general definition of exchangeability, then provide a theorem with which we can detect bipartitions (or, equivalently, edges) in an optimal phylogeny.

Given a bipartition (X, Y) of the taxon set S with $|X| = \ell \geq 2$ and $|Y| = n - \ell \geq 2$, define $Q_{(X,Y)}$ 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) (or across the edge) [4]. Let $p_1 = |Q_{(X,Y)} - Q|$, which is the number of quartet errors across (X, Y). 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). Clearly, there are, in total, $\binom{n-\ell}{3}$ such ℓ-subsets. For an 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. Let p_2 and p_3 denote the number of nonexchangeable ℓ-subsets on X and the number of nonexchangeable $(n-\ell)$-subsets on Y, respectively.

Lemma 5.4. *Given n $(n > 1)$ positive integers $a_1, a_2, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i = m$, then $\sum_{1 \leq i < j \leq n} a_i a_j \geq \frac{(n-1)(2m-n)}{2}$ and the equality holds if and only if $a_i = m - (n-1)$ for some i.*

Proof. Note that, since a_is are all positive and $\sum_{i=1}^{n} a_i = m$, there must be $m \geq n$. We prove the lemma by induction on n. When $n = 2$, $a_1 a_2 = -a_1^2 + m a_1 \geq m - 1$ and the equality holds if and only if $a_1 = m - 1$ or $a_2 = m - 1$.

Assume $\sum_{1 \leq i < j \leq k} a_i a_j \geq \frac{(k-1)(2m-k)}{2}$ and the equality holds if and only if $a_i = m - (k-1)$ for some i. We have

$$\begin{aligned}
&\sum_{1 \leq i < j \leq k+1} a_i a_j \\
&= \sum_{1 \leq i < j \leq k} a_i a_j + a_{k+1} \sum_{1 \leq i \leq k} a_i \\
&\geq \frac{(k-1)[2(m - a_{k+1}) - k]}{2} + a_{k+1}(m - a_{k+1}) \\
&= -a_{k+1}^2 + (m-k+1)a_{k+1} + m(k-1) - \frac{k(k-1)}{2} \\
&\geq (m-1)k - \frac{k(k-1)}{2} \\
&= \frac{k[2m - (k+1)]}{2}.
\end{aligned}$$

Also note that the equality holds if and only if $a_{k+1} = 1$ (second inequality becomes equality) and $a_i = m - k$ for some $1 \leq i \leq k$ (first inequality becomes equality) or $a_{k+1} = m - k$ (second inequality becomes equality and it forces $a_i = 1$ for all other i). This proves the lemma. □

Theorem 5.5. *Let Q be a complete set of quartet topologies on an n-taxon set S. For a bipartition (X, Y) of S, where $|X| = \ell \geq 2$ and $|Y| = n - \ell \geq 2$, let p_1 be the number of quartet errors in Q across (X, Y), p_2 be the number of nonexchangeable ℓ-subsets on X, and p_3 be the number of nonexchangeable $(n-\ell)$-subsets on Y. If $2p_1 + (\ell-1)p_2 + (n-\ell-1)p_3 \leq (\ell-1)(n-\ell-1)$, then bipartition (X, Y) must be in an optimal phylogeny.*

Proof. With respect to (X, Y), we divide set Q into five disjoint parts:

- Part 1: Every quartet topology is on four taxa in Y.
- Part 2: Every quartet is on one taxon in X and three taxa in Y.
- Part 3: Every quartet is on two taxa in X and two taxa in Y.
- Part 4: Every quartet is on three taxa in X and one taxon in Y.
- Part 5: Every quartet is on four taxa in X.

Furthermore, the quartet topologies in Part 2 are grouped into a list of ℓ-subsets on X; the quartet topologies in Part 4 are grouped into a list of $(n-\ell)$-subsets on Y. In the following, we will prove that, for any phylogeny T_1 on S which does not contain edge (X, Y), we can construct a new phylogeny T_2 on S out of T_1 that contains edge (X, Y) and satisfies at least as many quartet topologies as T_1 satisfies.

Let T_X and T_Y denote phylogenies on X and Y, respectively. We define an operation to join T_X and T_Y to produce a phylogeny on S as follows: Pick one edge e in T_X and create a degree-2 node v_e at the middle of the edge; pick one edge f in T_Y and create a degree-2 node v_f at the middle of the edge as well; connect v_e and v_f via an edge. Such an operation joins

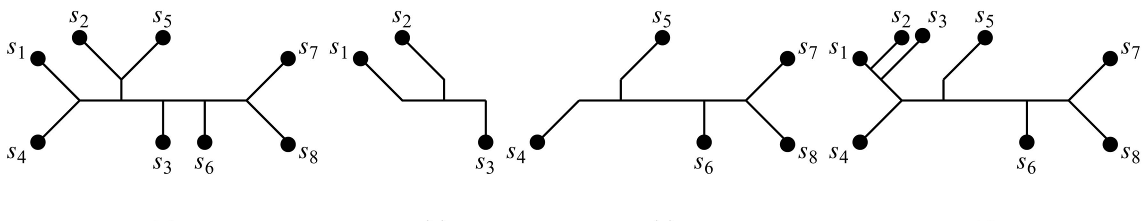


Fig. 2. Construction of a new phylogeny T_2 from T_1. (a) T_1. (b) T_{1X}. (c) T_{1Y}. (d) T_2.

tree T_X to tree T_Y at edge f using edge e. Fig. 2 shows an example of construction T_2 out of T_1, where $X = \{s_1, s_2, s_3\}$ and $Y = \{s_4, s_5, s_6, s_7, s_8\}$. From T_1, trimming all the leaf nodes in Y, we get a phylogeny T_{1X} on X (Fig. 2b); similarly, we get a phylogeny T_{1Y} on Y as shown in Fig. 2c. Through joining T_{1X} to T_{1Y} at the edge incident at s_4 using the edge incident at s_3, we get another phylogeny T_2.

We will be constructing a new phylogeny T_2 out of T_1 following the way described in the last paragraph. It follows that, for each exchangeable ℓ-subset in Part 2, if T_1 satisfies all its quartet topologies, then T_2 also satisfies all of them, no matter which edge in T_{1X} is used for joining to T_{1Y}. Conversely, if T_1 satisfies none of them, T_2 satisfies none of them either. However, if T_1 satisfies ℓ' of them for some ℓ' such that $0 < \ell' < \ell$, then T_2 satisfies either all (ℓ) of them or none of them, depending on which edge in T_{1X} is used for joining to T_{1Y}. Therefore, we can always find an edge in T_{1X} for joining to T_{1Y} such that T_2 satisfies at least as many quartet topologies in exchangeable ℓ-subsets in Part 2 as T_1 satisfies. Similarly, we can always find an edge in T_{1Y} for joining to T_{1X} such that T_2 satisfies at least as many quartet topologies in exchangeable $(n-\ell)$-subsets in Part 4 as T_1 satisfies. Another obvious fact according to the way we construct T_2 is that T_1 and T_2 satisfy the same number of quartet topologies in Part 1 and Part 5.

For each nonexchangeable ℓ-subset in Part 2, from the way T_2 is constructed, at least one of the ℓ quartet topologies is satisfied by T_2; similarly, for each nonexchangeable $(n-\ell)$-subset in Part 4, at least one of the $(n-\ell)$ quartet topologies is satisfied by T_2. It follows that, among the quartet topologies in the nonexchangeable ℓ-subsets in Part 2, T_1 can satisfy at most $(\ell-1)p_2$ more quartet topologies than T_2; among the quartet topologies in the nonexchangeable $(n-\ell)$-subsets in Part 4, T_1 can satisfy at most $(n-\ell-1)p_3$ more quartet topologies than T_2.

Now, we look from another angle to compare the number of quartet topologies in Part 3 that are satisfied by T_1 and T_2. Let us review the configuration of phylogeny T_{1X} in T_1. First of all, T_{1X} is on ℓ taxa and, thus, contains exactly $2\ell-3$ edges. Second, in T_1, taxa in Y are connected as (maximal) subtrees whose roots are connected to edges in T_{1X}. Suppose there are ℓ' such subtrees and the numbers of taxa in them are $m_1, m_2, \ldots, m_{\ell'}$, respectively (see Fig. 3a). These values have the following properties:

- $\ell' > 1$ since, otherwise, T_1 would contain edge (X, Y).
- $m_i > 0$, for every i: $1 \le i \le \ell'$.
- $m_1 + m_2 + \ldots + m_{\ell'} = n - \ell$.

Consider the ith subtree and the jth subtree with their taxon sets S_i and S_j. If their roots are connected to a common edge in T_{1X}, then the number of quartet errors in T_1 across (X, Y), each of which involves one taxon in S_i and one taxon in S_j, is at least $m_i m_j(\ell-1)$; if their roots are connected to separated edges in T_{1X}, then the number of quartet errors in T_1 across (X, Y), each of which involves one taxon in S_i and one taxon in S_j, is at least $m_i m_j(n_1n_2 + n_1n_3 + \ldots + n_{s-1}n_s)$, where n_x denotes the size of the taxon set of the xth (maximal) subtree whose root is connected to the path connecting the roots of the ith subtree and the jth subtree (see Fig. 3b). Exactly the same argument says:

- $s > 1$.
- $n_x > 0$, for every x: $1 \le x \le s$.
- $n_1 + n_2 + \ldots + n_s = \ell$.

Therefore, by Lemma 2,

$$n_1n_2 + n_1n_3 + \ldots + n_{s-1}n_s \ge \frac{(s-1)(2\ell-s)}{2} \ge \ell - 1.$$

Consequently, the total number of quartet errors in T_1 across (X, Y) is at least

$$\begin{aligned}(\ell-1)\sum_{1\le i<j\le(n-\ell)} m_i m_j &\ge (\ell-1)\frac{(\ell'-1)(2(n-\ell)-\ell')}{2}\\ &\ge (\ell-1)(n-\ell-1).\end{aligned}$$

Since there are p_1 quartet errors in Q across (X, Y), at least $(\ell-1)(n-\ell-1) - p_1$ quartet topologies in Part 3 of Q are not satisfied by T_1. In other words, among the

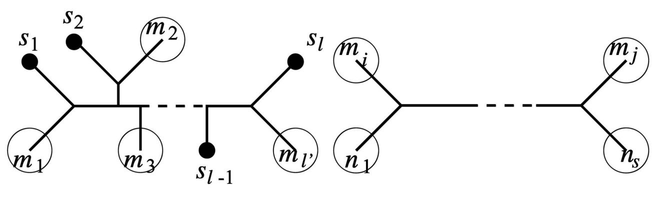


Fig. 3. (a) The maximal subtrees on taxa in Y whose roots are connected to edges in T_{1X}. (b) The maximal subtrees on taxa in X whose roots are connected to the path connecting the roots of the ith and the jth subtrees.

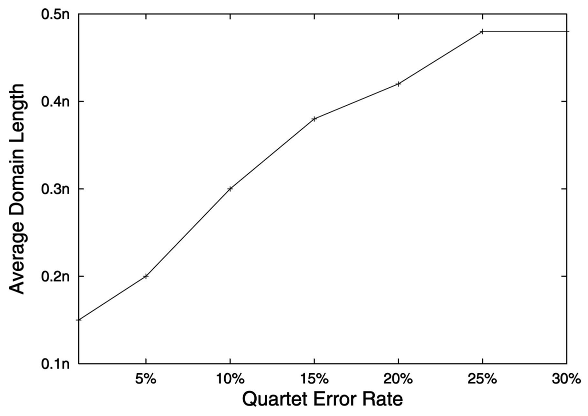


Fig. 4. Average domain lengths after domain reduction (n is the number of taxa).

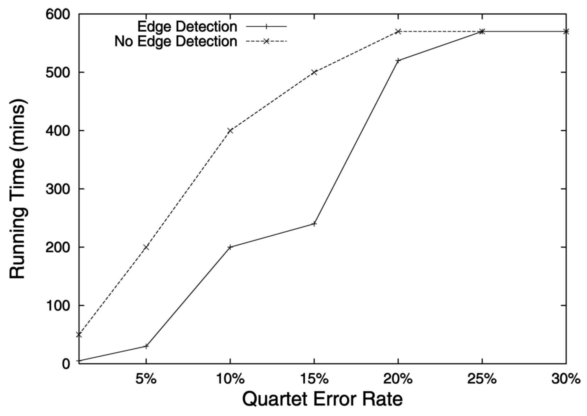


Fig. 5. The performance of Theorem 5.5 and Corollary 5.6 on data sets of 20 taxa.

quartet topologies in Part 3, T_2 can satisfy at least $(\ell-1)(n-\ell-1)-2p_1$ more quartet topologies than T_1.

In summary, as long as

$$(\ell-1)(n-\ell-1)-2p_1 \geq (\ell-1)p_2+(n-\ell-1)p_3,$$

the constructed T_2 satisfies at least the same number of quartet topologies in Q as T_1 satisfies. This proves the theorem. □

When $|X|=2$ and (X,Y) is in a phylogeny T, we call those two taxa in X *siblings* in T. In this special case, $p_3=0$ and we have the following corollary:

Corollary 5.6. *Let Q be a complete set of quartet topologies on an n-taxon set S. For a pair of taxa s_i and s_j, let p_1 denote the number of quartet errors in Q across bipartition $(\{s_i,s_j\}, S-\{s_i,s_j\})$ and p_2 denote the number of nonexchangeable 2-subsets on $\{s_i,s_j\}$. If $2p_1+p_2 \leq n-3$, then s_i and s_j must be siblings in an optimal phylogeny.*

6 Computational Results

To investigate how well the proposed ASP method works in practice, we ran it on several synthetic data sets to examine the strength of the speedup strategies and to make comparisons to three other exact algorithms for the MQC problem in the literature and to one of the currently best heuristics for the MQC problem. Note that we did not examine the quality of the solution of the MQC problem in the biological context, though experiments on real data sets could be set up for this purpose. The reported experimental results were done on a computer with an AMD 2.2 GHz Opteron processor and 2.5 GB main memory.

6.1 Synthetic Data Set Generation

For a set of n taxa, we generated a phylogeny by recursively joining randomly selected subtrees (again, through one edge in each subtree). The subtrees were selected from a set that initially contained only the one-node subtrees, each corresponding to a given taxon. When two subtrees were joined, we replaced them in the set by the newly generated subtree. This procedure yielded a phylogeny on n taxa and we derived the set of quartet topologies from the phylogeny. We then arbitrarily picked p percent of the $\binom{n}{4}$ quartet topologies and altered the topologies to make them potential quartet errors. We remark that this process only guarantees that the number of quartet errors in the data set is upper bounded by $\frac{p}{100}\binom{n}{4}$, but not necessarily equal to since some combinations of quartet topology alterations might give rise to a new compatible set of quartet topologies. We generated data sets defined by a pair (n,p). We used quartet error percentage $p = 1\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%$. For every pair (n,p), we generated 10 data sets. The following reported results are the average performance over them.

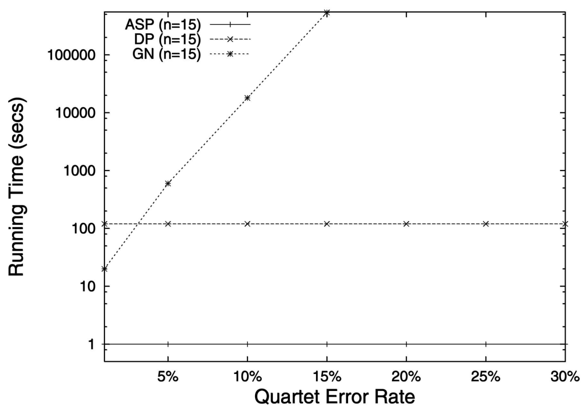
6.2 Experiments on the Speedup Strategies

The following experiments were intended to find out how much the speedup strategies proposed in Section 5 contribute to the efficiency of our method. The symmetry breaking strategy obviously can reduce the running time significantly since they can eliminate many variables. Fig. 4 shows the average domain sizes of variables $M(i,j)$ with respect to the size of input taxon set after we applied Theorems 5.1 and 5.3. From this figure, we can see that the smaller the quartet error rate was, the smaller domains that we were able to generate were.

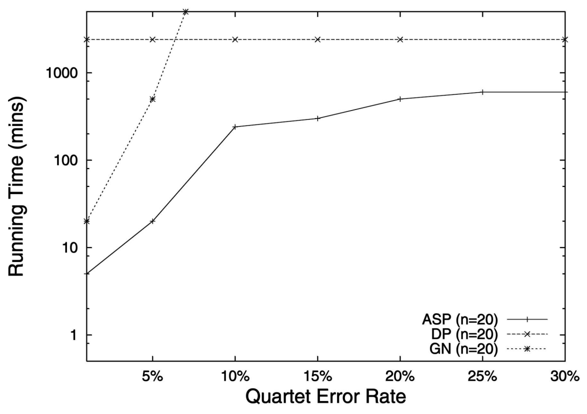
Given a taxon set S and a bipartition type $(k, n-k)$, there are $\binom{n}{k}$ possible bipartitions of the type. Therefore, testing all the possible bipartitions is essentially infeasible. We only tested all possible bipartitions with $k=2$, i.e., all possible sibling pairs. For bipartitions with $k>2$, we used the hypercleaning algorithm [3] to generate a set of candidate bipartitions first and used Theorem 5.5 on those bipartitions. For a data set of 20 taxa, hypercleaning finished in less than 2 minutes, which is short compared to the running time of our ASP method. Fig. 5 shows the computational results on the 10 data sets of 20 taxa with and without the application of edge detection strategy (Corollary 5.6, hypercleaning plus Theorem 5.5). Intuitively, the smaller the quartet error rate is, the more bipartitions could be discovered. In our experiments, we found that, when the quartet error rate is less than 1 percent, we could find all the sibling pairs in the optimal phylogeny.

6.3 Comparisons among Exact Algorithms

A fundamental idea in constraint programming systems like Smodels is that of constraint propagation during search, which, given some atoms whose truth values are already determined, quickly forces some of the remaining



(a)



(b)

Fig. 6. The running times of three exact algorithms for the MQC problem. (a) Running times for data sets of 15 taxa. (b) Running times for data sets of 20 taxa.

atoms to commit truth values. In this way, the search space could be pruned significantly. In Smodels, a powerful mechanism of *lookahead* is also employed, which assigns an atom a truth value without search if assuming the opposite truth value for the atom leads to a contradiction.

It is clear that the main search mechanisms in answer set programming systems like Smodels are distinguished from the existing exact algorithms for the MQC problem. Therefore, it is interesting to do an experimental comparison among the exact algorithms proposed previously for the MQC problem, including the dynamic programming algorithm by Ben-Dor et al. [2] (denoted as DP), the fixed-parameter algorithm by Gramm and Niedermeier [13] (denoted as GN), and our answer set program (denoted as ASP). For the GN algorithm, we used $k = \frac{p}{100} \times \binom{n}{4}$ as the first upper bound on the number of quartet errors.

When $n \leq 10$, all three algorithms could solve the MQC problem within several minutes. Fig. 6 gives the comparisons of running times on a logarithmic scale for data sets of 15 and 20 taxa, which shows that our ASP method outperformed the other two algorithms in all data sets. The computational time of the DP algorithm is independent of the quartet error rate. However, the performance of our ASP method and the GN algorithm is affected by quartet error rate. Fig. 6b shows that the running time of the GN algorithm grows too fast to be feasible, whereas our ASP method has a much slower running time growth with respect to quartet error rate and can solve the MQC problem more efficiently when the error rate is less than 10 percent.

When $n > 20$, the DP algorithm could not generate an optimal phylogeny within 7 days. GN could generate an optimal phylogeny if the number of quartet errors was very small compared to the total number of input quartet topologies. In [13], computational results on data sets of up to 50 taxa with up to 150 quartet errors are reported. Fig. 7 compares the running times of the GN algorithm and our ASP method on the data sets of 20 to 80 taxa with 100 quartet errors. It also gives the computational times of our ASP method when the quartet error percentage is 1 percent. We note that our ASP method is similar to the GN algorithm in that, for fixed values of quartet error number, it requires a shorter computational time on data sets on a larger number of taxa. This is mainly due to the practical speedup strategies proposed in Section 5, which become more effective when the quartet error rate is small. In our experiments, when $n > 30$, the GN algorithm could not finish the computation in 7 days even when the quartet error rate was only 1 percent, whereas our method could produce optimal phylogenies quickly and the computational time was almost in polynomial scale. These results show that some of the hard instances that were not reported to be solved by the GN and DP algorithms can now be solved by our method.

6.4 Experiments on the Use of Hypercleaning

The heuristic *hypercleaning* developed by Berry et al. [3] is an extension of the quartet cleaning methods. Given a bipartition (edge) (X, Y) of the taxon set S, we define $Q_{(X,Y)}$ to be the set of quartet topologies in the form of $[x, x'|y, y']$, where $x, x' \in X$ and $y, y' \in Y$. The normalized distance from Q to (X, Y) is defined as $\delta(Q, (X, Y)) = \frac{4|Q_{(X,Y)} - Q|}{|X|(|X|-1)|Y|(|Y|-1)}$, which basically measures the extent of disagreement of Q if edge (X, Y) is included in the output phylogeny. The

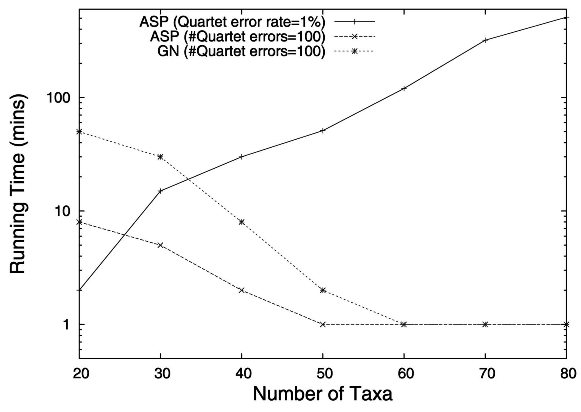


Fig. 7. The running times (in a logarithmic scale) of GN and ASP on data sets with small numbers of quartet errors.

TABLE 2
The Average Numbers of Edges in $Best(Q,0)$ and $Best(Q,1)$

Problem Size		$Best(Q,0)$	$Best(Q,1)$
$n = 10$	$p = 1\%$	5	7
	$p = 3\%$	4	7
	$p = 5\%$	1	7
	$p = 10\%$	0	5
	$p = 15\%$	0	1
$n = 20$	$p = 1\%$	3	17
	$p = 3\%$	0	16
	$p = 5\%$	0	3
	$p = 10\%$	0	0
	$p = 15\%$	0	0
$n = 30$	$p = 1\%$	2	26
	$p = 3\%$	0	19
	$p = 5\%$	0	2
	$p = 10\%$	0	0
	$p = 15\%$	0	0
$n = 40$	$p = 1\%$	0	32
	$p = 3\%$	0	26
	$p = 5\%$	0	0
	$p = 10\%$	0	0
	$p = 15\%$	0	0

hypercleaning algorithm computes the following sets of edges in order:

$$Best(Q,m) = \left\{ (X,Y) | \delta(Q,(X,Y)) \leq \frac{2m}{|X||Y|} \right\},$$
$$m = 0,1,2,\ldots.$$

Note that the edges in $Best(Q,0)$ are also in $Best(Q,1)$ based on their definitions. It has been shown that edges in $Best(Q,0)$ and $Best(Q,1)$ are always compatible with each other, while edges from $Best(Q,m)$ with greater values of m might not be compatible with edges in $Best(Q,0)$ and $Best(Q,1)$. The hypercleaning uses edges in $Best(Q,0)$ and $Best(Q,1)$ to form a base set of edges in the final phylogeny and tries to add edges in $Best(Q,2)$ to it greedily to grow the phylogeny and, if the phylogeny is not resolved yet, then tries to add edges in $Best(Q,3)$ to it greedily to grow the phylogeny, and so on, until a resolved phylogeny is formed. The overall running time of the hypercleaning algorithm is $O(n^5 f(2m) + n^7 f(m))$, where $f(m) = 4m^2(1+2m)^{4m}$ and m denotes the greatest value to which the set $Best(Q,m)$ has to be computed in the execution.

Although, in theory, the edges in $Best(Q,0)$ may not be in the optimal phylogeny for the MQC problem, we show that they have a high probability of being included in the optimal phylogeny. Suppose an edge (X,Y) ($|X| \leq |Y|$) in $Best(Q,0)$ is not in the optimal phylogeny, where the optimal edge should be (X',Y'). Then, the taxon set $(X-X') \cup (Y-Y')$ contains at least two distinct taxa, s_i and s_j, where $s_i \in X$ and $s_j \in Y$. For three taxa, s_i, s_j and another one $s_k \in X$, there are at least $n/2$ quartets, namely, $\{s_i, s_j, s_k, s_l\}$ for all $s_l \in Y$, that would make the same wrong prediction for the topologies of $\{s_i, s_j, s_k, s_l\}$, i.e., $[s_i, s_k | s_j, s_l]$. This is very unlikely in the quartet inference on real data, even in our synthetic data set. For example, the probability of uniformly randomly choosing those $n/2$ quartet topologies to change is less than $\frac{1}{n^3}$ as we are working on a complete set of quartet topologies. For each quartet topology, the probability of changing to the specified topology is 1/2 as we have two alternatives. Then, the overall probability of making (X,Y) be not included in the optimal phylogeny is less than $\frac{1}{n^3} \cdot (\frac{1}{2})^{\frac{n}{2}}$, which is very small.

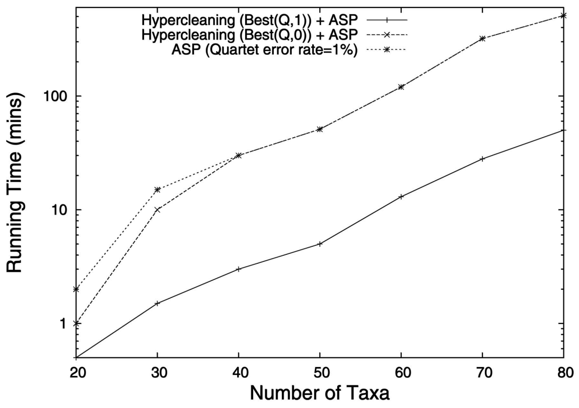


Fig. 8. The running times of ASP on data sets of 1 percent quartet error rate, with and without using hypercleaning as a preprocessing.

Table 2 collects average numbers of edges in $Best(Q,0)$ and $Best(Q,1)$ found in different test data sets. Experiments on small instances, e.g., up to 20 taxa, showed that all the edges in $Best(Q,0)$ and $Best(Q,1)$ were included in the optimal phylogenies. However, the number of edges in $Best(Q,0)$ and $Best(Q,1)$ is greatly affected by the quartet error rate. When the quartet error rate exceeds 5 percent, we could rarely get an edge in $Best(Q,1)$. Fig. 8 compares the running times of our answer set program on data sets with 1 percent quartet errors, with and without using hypercleaning as a preprocessing. It can be seen that, through including the edges in $Best(Q,1)$, the computational time of our answer set program is dramatically reduced, i.e., at a logarithmic scale.

7 Conclusions and Future Work

We have proposed a new equivalent formulation of the MQC problem in answer set programming and a number of optimization strategies for this new formulation. The formulation, together with our speedup strategies, might lead us to a new perspective of the problem as our experiments on a number of simulated data sets showed that the proposed method outperformed previously proposed exact algorithms for the MQC problem. Although, in the worst case, our program still takes exponential time, it allows the incorporation of the domain knowledge into the search process, where the search space can be significantly reduced by constraint propagation in ASP. In the ideal case, we might be able to encode the target matrix variables such that the exponential behavior becomes a rare occurrence and the average behavior is acceptable for practical use.

We note that there are a few existing works on phylogeny construction using ASP, however, in which the phylogeny construction problems considered are not quartet-based. For example, Brooks et al. encoded the *Maximum Compatibility* problem by an answer set program. In the Maximum Compatibility problem, a set of characters of taxa are given and the goal is to construct a phylogeny with a maximum number of compatible characters [6]. The encoded answer set program is run by Cmodels [17], which in turn calls the SAT solver zChaff [19] for the computation of answer sets. Gent et al. encoded another phylogeny construction problem by an answer set program, where the problem is, given a set of rooted triples of taxa, to compute a rooted phylogeny such that a maximum number of them are satisfied [12]. A rooted triple is of form $((i,j),k)$ that says, in the output rooted phylogeny, $\mathrm{LCA}(s_i, s_j)$ must be a descendant of $\mathrm{LCA}(s_i, s_k)$. It should be noted that, although this encoding gives an interesting perspective of the phylogeny construction problem, the problem is known to be solved in polynomial time [12]. In both [6] and [12], the constructed phylogenies are rooted binary trees and, in the encoding, they are ultrametric too.

One of the most important future works we want to pursue is to improve our encoding scheme to further speed up the computation. Our goal is set for solving harder instances containing 80 taxa within a day and, thus, to provide another fast way to optimal phylogeny construction. The ultimate goal is to fully explore the structured properties of the MQC problem and to design a quartet-specific constraint programming solver.

8 Supplemental Materials

The complete set of synthetic data sets that were used in our experiments are available on the Web at http://www.cs.ualberta.ca/~ghlin/src/WebTools/quartet.php. From the same Web page, readers may download our source code (and a link to the Smodels). Also available is a sample answer set program formulated for a small instance of the MQC problem. The instance was constructed out of a pairwise distance matrix on 10 taxa, where the quartet topologies were inferred using the four-point method. The solution to the answer set program by Smodels is also included, from which a phylogeny was written and can be readily fed to TreeView.

Acknowledgments

Gang Wu's research was partially supported by NSERC and CFI. Jia-Huai You's research was partially supported by NSERC and a grant from NSFC under project 60573009. Guohui Lin's research was partially supported by NSERC and CFI.

References

[1] H. Bandelt and A. Dress, "Reconstructing the Shape of a Tree from Observed Dissimilarity Data," *Advances in Applied Math.*, vol. 7, pp. 309-343, 1986.

[2] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg, "From Four-Taxon Trees to Phylogenies: The Case of Mammalian Evolution," *Proc. Fourth Ann. Int'l Computing and Combinatorics Conf. (RECOMB)*, pp. 9-19, 1998.

[3] 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," *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pp. 287-296, Jan. 2000.

[4] V. Berry, T. Jiang, P.E. Kearney, M. Li, and H.T. Wareham, "Quartet Cleaning: Improved Algorithms and Simulations," *Proc. Seventh Ann. European Symp. Algorithms (ESA '99)*, pp. 313-324, 1999.

[5] B. Borchers and J. Furman, "A Two-Phase Exact Algorithm for MAX-SAT and Weighted MAX-SAT Problems," *J. Combinatorial Optimization*, vol. 2, pp. 299-306, 1999.

[6] D.R. Brooks, E. Erdem, J.W. Minett, and D. Ringe, "Character-Based Cladistics and Answer Set Programming," *Proc. Seventh Int'l Symp. Practical Aspects of Declarative Languages (PADL '05)*, pp. 37-51, 2005.

[7] M.C.H. Dekker, "Reconstruction Methods for Derivation Trees," master's thesis, Vrije Univ., Amsterdam, 1986.

[8] P. Erdos, M. Steel, L. Szekely, and T. Warnow, "A Few Logs Suffice to Build (Almost) All Trees (Part 1)," *Random Structures and Algorithms*, vol. 14, pp. 153-184, 1999.

[9] P.L. Erdos, M.A. Steel, L.A. Szekely, and T.J. Warnow, "Inferring Big Trees from Short Sequences," *Proc. Int'l Congress on Automata, Languages, and Programming*, pp. 827-837, 1997.

[10] J. Felsenstein, "The Number of Evolutionary Trees," *Systematic Zoology*, vol. 27, pp. 27-33, 1978.

[11] M. Gelfond and V. Lifschitz, "The Stable Model Semantics for Logic Programming," *Proc. Fifth Int'l Conf. Logic Programming*, pp. 1070-1080, 1988.

[12] I.P. Gent, P. Prosser, B.M. Smith, and W. Wei, "Supertree Construction with Answer Set Programming," *Proc. Ninth Int'l Conf. Principles and Practice of Constraint Programming (CP '03)*, pp. 837-841, 2003.

[13] J. Gramm and R. Niedermeier, "A Fixed-Parameter Algorithm for Minimum Quartet Inconsistency," *J. Computer and System Sciences*, vol. 67, pp. 723-741, 2003.

[14] D. Gusfield, *Algorithms on Strings, Trees, and Sequences.* Cambridge, 1997.

[15] T. Jiang, P.E. Kearney, and M. Li, "Orchestrating Quartets: Approximation and Data Correction," *Proc. 39th Symp. Foundations of Computer Science*, pp. 416-425, 1998.

[16] T. Jiang, P.E. Kearney, and M. Li, "Some Open Problems in Computational Molecular Biology," *J. Algorithms*, vol. 34, pp. 194-201, 2000.

[17] Y. Lierler and M. Maratea, "Cmodels-2: SAT-Based Answer Set Solver Enhanced to Non-Tight Programs," *Proc. Seventh Int'l Conf. Logic Programming and Nonmonotonic Reasoning (LPNMR '04)*, pp. 346-350, 2004.

[18] F. Lin and Y. Zhao, "ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers," *Artificial Intelligence*, vol. 157, pp. 115-137, 2004.

[19] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver," *Proc. 38th Design Automation Conf. (DAC '01)*, pp. 530-535, 2001.

[20] I. Niemelä, "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm," *Annals of Math. and Artificial Intelligence*, vol. 25, pp. 241-273, 1999.

[21] D. Pelleg, "Algorithms for Constructing Phylogenies from Quartets," master's thesis, Israel Inst. of Technology, 1998.

[22] S. Sattath and A. Tversky, "Additive Similarity Trees," *Psychometrika*, vol. 42, pp. 319-345, 1977.

[23] P. Simons, "Smodels: An Implementation of the Stable Model Semantics for Logic Programs," http://www.tcs.hut.fi/Software/smodels/, 2000.

[24] M. Steel, "The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees," *J. Classification*, vol. 9, pp. 91-116, 1992.

[25] K. Strimmer and A. von Haeseler, "Quartet Puzzling: A Quartet Maximum-Likelihood Method for Reconstructing Tree Topologies," *Molecular Biology and Evolution*, vol. 13, pp. 964-969, 1996.

[26] Z. Xing and W. Zhang, "MaxSolver: An Efficient Exact Algorithm for (Weighted) Maximum Satisfiability," *Artificial Intelligence*, vol. 164, pp. 47-60, 2005.

[27] J. You, G. Liu, L. Yuan, and C. Onuczko, "Lookahead in Smodels Compared to Local Consistencies in CSP," *Proc. Eighth Int'l Conf. Logic Programming and Nonmonotonic Reasoning (LPNMR '05),* pp. 266-278, 2005.

[28] H. Zhang, "SATO: An Efficient Propositional Prover," *Proc. Int'l Conf. Automated Deduction,* pp. 272-275, 1997.

Gang Wu received the MSc degree from the Information Communication Institute of Singapore, Nanyang Technological University, in 2002. He is a PhD student of computing science at the University of Alberta. His research interests include phylogeny construction, answer set programming, and constraint programming. He is a student member of the IEEE Computer Society.

Jia-Huai You received the PhD degree from the University of Utah in 1985. He is a professor in the Department of computing science at the University of Alberta and has worked in a faculty position at Rice University. His main research interests are in logic-based programming paradigms, including answer set programming, propositional satisfiability, and various approaches to constraint programming.

Guohui Lin received the PhD degree in theoretical computer science from the Chinese Academy of Sciences in 1998. He joined the University of Alberta as an assistant professor of computing science in July 2001. His research interests include bioinformatics, computational biology, and algorithm design and analysis, and his recent work focuses on algorithmic developments for protein structure determination and comparison, whole genome phylogenetic analysis, RNA structure prediction and comparison, and putative gene finding. He is a member of the ACM and the IEEE Computer Society.