A-003

A Polynomial-Time Algorithm for Ordered Triplets Consistency

Shuo Wang* Jesper Jansson* Shin-ichi Minato*

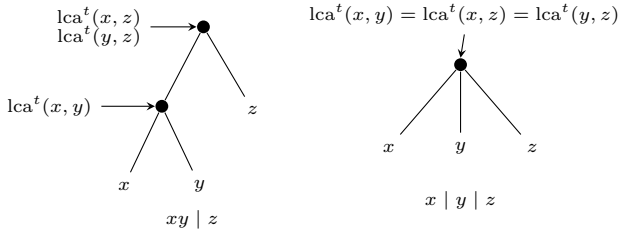
1. Introduction

Phylogenetic trees represent ancestral relationships among entities such as species, genes, or languages. Since constructing a phylogenetic tree directly can be difficult, the so-called supertree approach first builds smaller trees from local information and then combines them into a single tree. In this context, the rooted triplets consistency problem is to construct a phylogenetic tree that is consistent with all the required triplets and no forbidden triplets from the input, or to determine that no such tree exists. We present a polynomial-time algorithm based on interval dynamic programming for the ordered version of this problem.

2. Problem Definition

A *phylogenetic tree* is a rooted tree in which each leaf has a distinct label and each internal node has at least two children. For a phylogenetic tree T, let $L(T)$ denote its leaf-label set. For any two nodes u and v in T, their lowest common ancestor is denoted by $\mathrm{lca}^T(u, v)$.

A *rooted triplet* is a phylogenetic tree with exactly three leaves. Let t be a rooted triplet with $L(t) = \{x, y, z\}$. If t is binary, then t is called a *resolved triplet*. We write $t = xy \mid z$ if $\mathrm{lca}^t(x, y)$ is a proper descendant of $\mathrm{lca}^t(x, z) = \mathrm{lca}^t(y, z)$. Otherwise, t is called a *fan triplet* and is written as $t = x \mid y \mid z$, where $\mathrm{lca}^t(x, y) = \mathrm{lca}^t(x, z) = \mathrm{lca}^t(y, z)$.



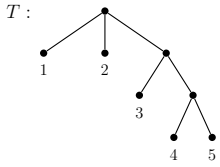
Given a phylogenetic tree T and three distinct leaves $x, y, z \in L(T)$, the resolved triplet $xy \mid z$ is *consistent with* T if $\mathrm{lca}^T(x, y)$ is a proper descendant of $\mathrm{lca}^T(x, z) = \mathrm{lca}^T(y, z)$. Similarly, the fan triplet $x \mid y \mid z$ is *consistent with* T if $\mathrm{lca}^T(x, y) = \mathrm{lca}^T(x, z) = \mathrm{lca}^T(y, z)$. Let $t(T)$ denote the set of all rooted triplets consistent with T.

The $R^{+-}F^{+-}$-Consistency problem is defined as follows. Given two sets R^+ and R^- of resolved triplets and two sets F^+ and F^- of fan triplets over a leaf-label set L, output a phylogenetic tree T with $L(T) = L$ such that $R^+ \cup F^+ \subseteq t(T)$ and $(R^- \cup F^-) \cap t(T) = \emptyset$, if one exists; otherwise, output NULL. Below, we regard each triplet in $R^+ \cup F^+$ as a required constraint on the solution and each triplet in $R^- \cup F^-$ as a forbidden constraint.

We now define the ordered version considered in this paper. An *ordered phylogenetic tree* is a phylogenetic tree in which the children of each internal node are ordered from left to right. Scanning the leaves of an ordered phylogenetic tree T from left to right yields a sequence called the *frontier* of T, denoted by $\mathrm{fr}(T)$. Let $\sigma = (\ell_1, \ell_2, \ldots, \ell_n)$ be a prescribed ordering of the leaf-label set L. We say that T respects σ if $\mathrm{fr}(T) = \sigma$.

The ordered $R^{+-}F^{+-}$-Consistency problem is defined as follows. Given a prescribed ordering σ of a leaf-label set L, two sets R^+ and R^- of resolved triplets over L, and two sets F^+ and F^- of fan triplets over L, output an ordered phylogenetic tree T with $L(T) = L$ such that $\mathrm{fr}(T) = \sigma$, $R^+ \cup F^+ \subseteq t(T)$, and $(R^- \cup F^-) \cap t(T) = \emptyset$, if one exists; otherwise, output NULL.

The following example illustrates the ordered $R^{+-}F^{+-}$-Consistency problem:



Consider the input: $L = \{1, 2, 3, 4, 5\}$, $\sigma = (1, 2, 3, 4, 5)$, $R^+ = \{34 \mid 1\}$, $R^- = \{12 \mid 3\}$, $F^+ = \{1 \mid 2 \mid 4\}$, and $F^- = \{1 \mid 4 \mid 5,\ 3 \mid 4 \mid 5\}$. The output tree T satisfies $\mathrm{fr}(T) = \sigma$. Also, $34 \mid 1 \in t(T)$, $12 \mid 3 \notin t(T)$, $1 \mid 2 \mid 4 \in t(T)$, $1 \mid 4 \mid 5 \notin t(T)$, and $3 \mid 4 \mid 5 \notin t(T)$. Thus, T is a valid solution.

3. Previous Work and Main Result

Rooted triplets consistency has been extensively studied in the unordered setting; see, e.g., [3] for a survey. The most basic variant is the R^+-Consistency problem, in which the input contains only required resolved triplets. In 1981, Aho et al. [1] showed how to solve this problem in polynomial time. Subsequent studies considered variants that allow different combinations of the four constraint types. Jansson et al. [3] cataloged the computational complexity of the $R^{+-}F^{+-}$-Consistency problem and its restricted variants. Their results, summarized in Table 1, show that the complexity depends on the allowed types of triplet constraints: some variants are polynomial-time solvable, whereas others are NP-hard.

Table 1 Complexity of unordered $R^{+-}F^{+-}$-Consistency variants.

Consistency	$\emptyset$	F^+	F^-	F^{+-}
$\emptyset$	–	P	P	NP-hard
R^+	P	P	P	NP-hard
R^-	P	P	NP-hard	NP-hard
R^{+-}	P	P	NP-hard	NP-hard

"P" means solvable in polynomial time.

To the best of our knowledge, previous work on the ordered setting has considered only the R^+-Consistency problem. Gąsieniec et al. [2] gave an $O((m + n) \log n)$-time algorithm for this problem, where $m = |R^+|$ and $n = |L|$.

This paper considers more general combinations of constraints in the ordered setting, allowing the input to contain any of the four constraint types R^+, R^-, F^+, and F^-. The $\mathbf{P}^*$ entry in the bottom-right cell of Table 2 represents the main result of this paper, which is a new algorithm that solves the ordered $R^{+-}F^{+-}$-Consistency problem in polynomial time. Since every other ordered variant is a special case of the full problem, it follows that all variants in the table are polynomial-time solvable.

Table 2 Complexity of ordered $R^{+-}F^{+-}$-Consistency variants.

Consistency	$\emptyset$	F^+	F^-	F^{+-}
$\emptyset$	–	P	P	P
R^+	P [2]	P	P	P
R^-	P	P	P	P
R^{+-}	P	P	P	$\mathbf{P}^*$

∗ Graduate School of Informatics, Kyoto University, Japan.

Copyright © 2026 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

4. The New Dynamic Programming Algorithm

Index the leaves according to the prescribed order as $1 < 2 < \cdots < n$. In an ordered phylogenetic tree, the descendant leaves of every node form a contiguous interval. Hence, the children of the root of any subtree with leaf set $[P, Q]$ induce a left-to-right partition

$$[P, Q] = I_1 \dot\cup I_2 \dot\cup \cdots \dot\cup I_k, \qquad k \geq 2.$$

We use a two-level dynamic programming solution (DP) based on intervals. The outer DP processes intervals in increasing order of length, while the inner DP determines whether a fixed interval can be partitioned into root-child blocks that are recursively feasible and compatible with the constraints checked at the current root.

For each interval $[P, Q]$, define the Boolean variable

$$\text{TreeFeasible}[P, Q] = \text{true}$$

if and only if there exists an ordered phylogenetic tree with leaf set $[P, Q]$ that satisfies every input constraint whose three leaves lie in $[P, Q]$.

4.1 Candidate Blocks and Root Compatibility

To compute TreeFeasible$[P, Q]$, consider each proper subinterval $[i, j] \subsetneq [P, Q]$ as a candidate root-child block. For a constraint on $a < b < c$ with $b \in [i, j]$, the positions of a and c relative to $[i, j]$ determine whether the constraint is handled recursively or at the current root. If $a, c \in [i, j]$, the constraint is deferred to the subtree on $[i, j]$. Otherwise, the current root induces one of the three ordered triplet topologies $ab \mid c$, $a \mid bc$, and $a \mid b \mid c$, as shown in Fig. 1.

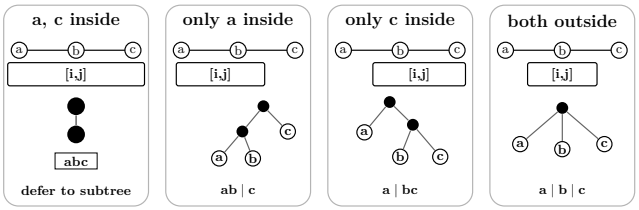


Fig. 1 Four root-level cases for a candidate block $[i, j]$.

For the fixed interval $[P, Q]$, define

$$\text{RootCompatible}_{P,Q}(i, j) = \text{true}$$

if and only if using $[i, j]$ as one root-child block of $[P, Q]$ does not violate any input constraint whose three leaves lie in $[P, Q]$. To test this condition, consider each input constraint on leaves $a < b < c$ whose three leaves lie in $[P, Q]$ and whose middle leaf b lies in $[i, j]$. The validity conditions for the six constraint forms are listed in Table 3.

Table 3 Validity requirements for a candidate block $[i, j]$.

Constraint	Condition for $[i, j]$ to be valid
$R^+ : ab \mid c$	$a \in [i, j]$
$R^+ : a \mid bc$	$c \in [i, j]$
$F^+ : a \mid b \mid c$	$\big(a \in [i, j] \wedge c \in [i, j]\big) \vee \big(a \notin [i, j] \wedge c \notin [i, j]\big)$
$R^- : ab \mid c$	$\neg\big(a \in [i, j] \wedge c \notin [i, j]\big)$
$R^- : a \mid bc$	$\neg\big(a \notin [i, j] \wedge c \in [i, j]\big)$
$F^- : a \mid b \mid c$	$a \in [i, j] \vee c \in [i, j]$

A candidate block $[i, j]$ is usable in a root partition of $[P, Q]$ exactly when it has a feasible subtree and is compatible at the current root. Define

$$\text{Usable}_{P,Q}(i, j) = \text{TreeFeasible}[i, j] \wedge \text{RootCompatible}_{P,Q}(i, j).$$

4.2 Inner DP

For the fixed interval $[P, Q]$ and each $P \leq y \leq Q$, define

$$\text{PrefixPartitionable}_{P,Q}[y] = \text{true}$$

if and only if the prefix $[P, y]$ can be partitioned into one or more usable blocks. Set $\text{PrefixPartitionable}_{P,Q}[P-1] = \text{true}$ for the empty prefix. The variables are computed from left to right by

$$\text{PrefixPartitionable}_{P,Q}[y] = \bigvee_{\substack{P-1 \leq x < y \\ (x,y) \neq (P-1,Q)}} \Big(\text{PrefixPartitionable}_{P,Q}[x] \wedge \text{Usable}_{P,Q}(x+1, y)\Big).$$

Here, $[x + 1, y]$ is the last block appended to a partition of $[P, x]$. The condition $(x, y) \neq (P - 1, Q)$ excludes the partition consisting only of $[P, Q]$, since the root must have at least two children.

Whenever $\text{PrefixPartitionable}_{P,Q}[y]$ is set to true for some value x, also store the predecessor $\text{pred}_{P,Q}[y] = x$. The value x is the endpoint of the prefix preceding the last block $[x + 1, y]$. If $\text{PrefixPartitionable}_{P,Q}[Q] = \text{true}$, following these predecessor values from Q back to $P - 1$ recovers a feasible root partition of $[P, Q]$.

4.3 Outer DP and Tree Reconstruction

Initialize $\text{TreeFeasible}[P, P] = \text{true}$ for $1 \leq P \leq n$ and $\text{TreeFeasible}[P, P + 1] = \text{true}$ for $1 \leq P < n$, since intervals with at most two leaves contain no triplet constraints. The remaining intervals are processed in increasing order of length. For each interval $[P, Q]$, set

$$\text{TreeFeasible}[P, Q] = \text{PrefixPartitionable}_{P,Q}[Q].$$

Every value $\text{TreeFeasible}[i, j]$ used in $\text{Usable}_{P,Q}(i, j)$ has already been computed because $[i, j]$ is a proper subinterval of $[P, Q]$. After all intervals have been processed, the original instance is feasible if and only if $\text{TreeFeasible}[1, n] = \text{true}$. If this value is false, the algorithm outputs NULL. Otherwise, it recovers a root partition of $[1, n]$ from the stored predecessor values. It recursively constructs a subtree for each block and attaches the resulting subtrees, from left to right, to a new root. Repeating this process yields a solution tree.

4.4 Computational Complexity Analysis

Let n be the number of leaf labels and m the number of input triplets. The outer DP has $O(n^2)$ choices of the parent interval $[P, Q]$. For each $[P, Q]$, the inner DP computes $O(n)$ prefix variables and examines $O(n)$ preceding boundaries for each variable, giving $O(n^2)$ candidate blocks. Testing each candidate block takes $O(n + m)$ time. Therefore, the total running time is $O(n^4(n + m))$. The TreeFeasible table and the stored predecessor values require $O(n^3)$ space. Thus, the algorithm runs in polynomial time and space.

This work was partly supported by JSPS KAKENHI Grant Number JP25H01114.

References

[1] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput.*, 10:405–421, 1981.

[2] L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. Inferring ordered trees from local constraints. In *Proceedings of CATS'98*, pages 67–76, 1998.

[3] J. Jansson, A. Lingas, R. Rajaby, and W.-K. Sung. Determining the consistency of resolved triplets and fan triplets. *J. Comput. Biol.*, 25:740–754, 2018.

Copyright © 2026 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.