STELAR-X: Scaling Coalescent-Based Species Tree Inference to 100,000 Species and Beyond

Anik Saha[1] and Md. Shamsuzzoha Bayzid[1,*]

[1]Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka-1205, Bangladesh
[*]Corresponding author: shams_bayzid@cse.buet.ac.bd

Abstract

Summary methods reconstruct species trees from collections of gene trees by accounting for gene tree discordance, and offer a statistically consistent framework for phylogenomic inference under the multispecies coalescent model. While existing triplet- and quartet-based approaches such as ASTRAL and STELAR have provable statistical consistency, their running time and memory usage restrict their applicability to ultra-large datasets. We introduce STELAR-X, a highly scalable triplet-based phylogenetic inference algorithm that achieves an asymptotically optimal memory complexity of $O(nk)$ for n species and k gene trees–essentially matching the input size and allowing analyses to remain feasible as long as the input trees fit in memory–while also substantially reducing running time. STELAR-X achieves this by a comprehensive re-engineering of the underlying data structures and algorithms. We introduce a novel, compact integer tuple-based encoding of tree bipartitions and efficient procedures for rapid pre-computation of bipartition weights. We further leverage GPU parallelism for fast pre-computation of necessary weights. This improved and redesigned computational framework underpins a dynamic programming algorithm with substantially reduced computational overhead. Extensive experiments demonstrate that STELAR-X achieves unprecedented scalability. On simulated datasets with 10,000 taxa and 1,000 gene trees, STELAR-X runs 712× faster than ASTRAL-MP (the most scalable variant of ASTRAL) while using 7.5× less CPU memory. Most significantly, STELAR-X analyzed a dataset of 100,000 taxa and 1,000 genes in 8.5 hours using 86 GB RAM, and a 100,000-gene dataset with 1000 taxa in just 4 minutes using 106 GB RAM – scales that were previously intractable for statistically consistent summary methods. STELAR-X is publicly available at `https://github.com/aaniksahaa/STELAR-X`.

1 Introduction

Widespread genome-wide discordance poses a major challenge for phylogenomic studies based on multi-locus datasets [1, 2]. A species tree represents the branching pattern of species through speciation events, while a gene tree describes the evolutionary history of a particular gene across those species. Various biological processes can cause different loci to follow different evolutionary paths, so estimating a species tree typically involves analyzing multiple genes and alignments across the genome. Among these processes, incomplete lineage sorting (ILS) is widely recognized as a dominant source of discordance, best explained under the multispecies coalescent model [3–6]. In the presence of gene tree heterogeneity, traditional approaches such as concatenation–where gene alignments are merged into a single supermatrix to infer one tree–can be statistically inconsistent [7,8], and produce incorrect trees with high support [9].

To overcome these limitations, a range of methods have been developed that explicitly account for ILS, many of which are provably statistically consistent, meaning that they will converge in probability to the true species tree given sufficiently large numbers of genes and sites per gene [10–14]. The most scalable among these are "summary methods", which first infer gene trees for individual loci and then combine them to estimate the species tree. Notable examples of statistically consistent, coalescent-based summary methods include MP-EST [14], ASTRAL [15], BUCKy [16], GLASS [17], STEM [18], SVDquartet [13], STEAC [19], NJst [20], ASTRID [21], STELAR [22].

Among summary methods, ASTRAL has emerged as the most widely used and accurate approach for large-scale phylogenomic analyses under the multispecies coalescent model. By maximizing quartet consistency across input gene trees, ASTRAL provides statistically consistent species tree estimates. ASTRAL and its parallel implementation ASTRAL-MP [23] have been shown to scale to datasets with up to ∼10,000 species with several days of running time. Despite these advances, both ASTRAL and ASTRAL-MP face scalability bottlenecks when the number of species and genes exceeds several tens of thousands.

Improving the scalability of species tree inference has long been a key focus in phylogenomics. Divide-and-conquer strategies have proven particularly effective in extending summary methods to larger datasets. For example, the disk-covering method [24] demonstrated that partitioning taxa into overlapping subsets ("disks"), inferring local trees, and merging them can substantially enhance scalability of summary methods without sacrificing accuracy. A notable recent development in this direction is uDANCE [25], a divide-and-conquer framework for inferring and incrementally updating large genome-wide phylogenetic trees. Unlike traditional phylogenomic methods that reconstruct trees *de novo* each time, uDance can update and expand an existing "backbone" tree as new genomes are added. The backbone-growth paradigm allows uDANCE to analyze much larger taxon sets than standard summary methods. However, constructing a "tree of life" may require building backbone trees consisting of hundreds of thousands of taxa–a challenge that remains unresolved. Furthermore, because uDANCE is not explicitly designed to ensure statistical consistency under the multispecies coalescent model, the scalability of provably consistent summary methods (e.g., ASTRAL) still falls far short of the needs of ultra-large phylogenomics involving hundreds of thousands of species.

In this work, we address this gap by introducing STELAR-X, an extended version of the statistically consistent, coalescent-based method STELAR (Species Tree Estimation by maximizing tripLet AgReement) [22]. STELAR-X achieves unprecedented scalability through a combination of algorithmic redesign and efficient data representations. We introduce a new, compact integer tuple-based representation of tree bipartitions that avoids traditional $O(n^2k)$ bitset-based encoding used in widely used methods like ASTRAL. We also used a customized double-hashing scheme based on permutation-invariant and associative hash functions with provably negligible collision probability. In addition, we introduce optimized CPU and GPU parallelization to accelerate computation. These innovations yield an asymptotically optimal memory complexity of $O(nk)$, ensuring that analyses remain feasible whenever the input gene trees fit in memory. Combined with additional algorithmic optimizations, STELAR-X reduces the running time of the original STELAR from $O(n^4k^3)$ to $O(n^2k^2)$. In comparison with ASTRAL-MP, the most scalable variant of ASTRAL, STELAR-X demonstrated dramatic performance improvements: on datasets with 10,000 species and 1,000 genes, it ran 712× faster and used 7.5× less memory. ASTRAL-MP was shown to scale to a 10,000-species dataset [23] which required 15 hours of computation and 123GB memory in our experiment, whereas STELAR-X completed the same scale of analysis within only two minutes using only 20GB memory. Remarkably, STELAR-X can analyze datasets with 100,000 species and thousands of genes in under 8.5 hours using 86 GB of memory, and datasets with 100,000 genes and 1,000 taxa in just 4 minutes using 106 GB memory – a level of scalability that represents a significant advance and establishes a new benchmark for statistically consistent species tree inference.

2 Materials and Methods

2.1 Problem Definition

Let $\mathcal{G} = \{g_1, g_2, \ldots, g_k\}$ be a collection of k binary rooted gene trees, each defined on the same set $\mathcal{X}$ of n taxa. For any set of three taxa $\{a, b, c\}$, let $ab|c$ denote the triplet tree where a and b form a cherry and c is the outgroup. The three possible triplet topologies for a set $\{a, b, c\}$ of taxa are ($ab|c$, $bc|a$, $ca|b$). Let $\tau(t)$ represent the set of all triplets induced by a tree t. The problem of constructing species trees by maximizing triplet consistency with the input gene trees is defined as follows. The species tree reconstruction problem by maximizing triplet consistency seeks for a species tree T on $\mathcal{X}$ that maximizes the triplet score

$$\text{score}(T, \mathcal{G}) \;=\; \sum_{i=1}^{k} \big|\tau(g_i) \cap \tau(T)\big|.$$

2.2 Background: overview of STELAR

For a rooted, binary tree t, each internal node u induces a subtree bipartition $(A|B)$ where A and B are the leaf sets of the two child subtrees of u [26]. Let $\mathcal{SBP}(t)$ denote the set of internal bipartitions of t. Then the collection of subtree-bipartitions in the set $\mathcal{G}$ of gene trees is $\mathcal{GB} = \bigcup_i \mathcal{SBP}(g_i)$. We further define $\mathcal{UGB}$ as the set of unique subtree-bipartitions from $\mathcal{GB}$. STELAR infers a species tree ST using a dynamic programming approach that recursively identifies the optimal (in terms of maximizing the triplet consistency) subtree-bipartitions at each level in a top-down manner [22]. Note that any triplet on $\{a, b, c\}$ is mapped to a unique internal node u in T (the least common ancestor of $\{a, b, c\}$ in T), and this node in turn induces a subtree-bipartition $(A|B)$. Consequently, the triplet consistency score of a candidate species tree ST can be equivalently expressed as a sum over the scores for the subtree-bipartitions in $\mathcal{SBP}(ST)$. Let $w_{\mathcal{G}}(x)$ denote the weight of a subtree-bipartition x (the number of triplets in $\mathcal{G}$ that map to x). STELAR reconstructs the species tree by dynamically selecting subtree-bipartitions from a candidate set $\mathcal{CB}$ according to the following recursive formulation:

$$V(A) = \begin{cases} 0 & \text{if } |A| = 1, \\ \max_{A' \subset A,\, (A'|A-A') \in \mathcal{CB}} V(A') + V(A - A') + w_{\mathcal{G}}(A'|A - A') & \text{if } |A| > 1, \end{cases} \tag{1}$$

where $V(A)$ is the optimal triplet score achievable on cluster $C \subseteq \mathcal{X}$. If $\mathcal{CB}$ contains all possible bipartitions, we obtain an exact but exponential time algorithm. In the constrained version of STELAR, $\mathcal{CB}$ is restricted to only those bipartitions observed in the input gene trees, i.e., $\mathcal{CB} = \mathcal{UGB}$. $\mathcal{CB}$ can also be considered as a modest extension of $\mathcal{UGB}$ by adding extra bipartitions for handling extensively incomplete gene trees. STELAR, including its constrained version, is a statistically consistent approach for species tree estimations [22].

2.3 Overview of STELAR-X

To extend STELAR for the analysis of ultra-large phylogenetic datasets, we introduce algorithmic and data-structure innovations in STELAR-X: a compact tuple representation of subtree bipartitions, a permutation-invariant double-hashing scheme for frequency mapping, GPU-accelerated precomputation of bipartition weights, and an efficient dynamic-programming solver that uses the precomputed maps.

As we demonstrate theoretically (Section 2.6), the computation of weights constitutes the primary bottleneck in terms of running time. The structure of this computation, however, makes it particularly well-suited for Single Instruction Multiple Data (SIMD) execution [27]. We therefore isolate this step as a standalone routine and run it with GPU parallelization beforehand (discussed in Section 2.8).

In the earlier version of STELAR, all subtree-bipartitions were represented as bitsets, the unique ones were extracted subsequently, and bitwise operations were used to calculate weights. With bitset representations, even a single taxon is represented by a bit vector of length n. For datasets with hundreds of thousands of taxa, this strategy leads to prohibitive RAM consumption and running time.

To address this scalability challenge, STELAR-X represents each subtree bipartition by a fixed-length integer tuple derived from the postorder leaf array of its source gene tree (Section 2.4). It maps equivalent subtree-bipartitions (which may appear in different gene trees and in swapped order) to a unique hash signature using a customized double hashing, which makes collisions vanishingly rare, and accumulates frequencies for a subtree-bipartition in a hash table (Section 2.5). Using this frequency

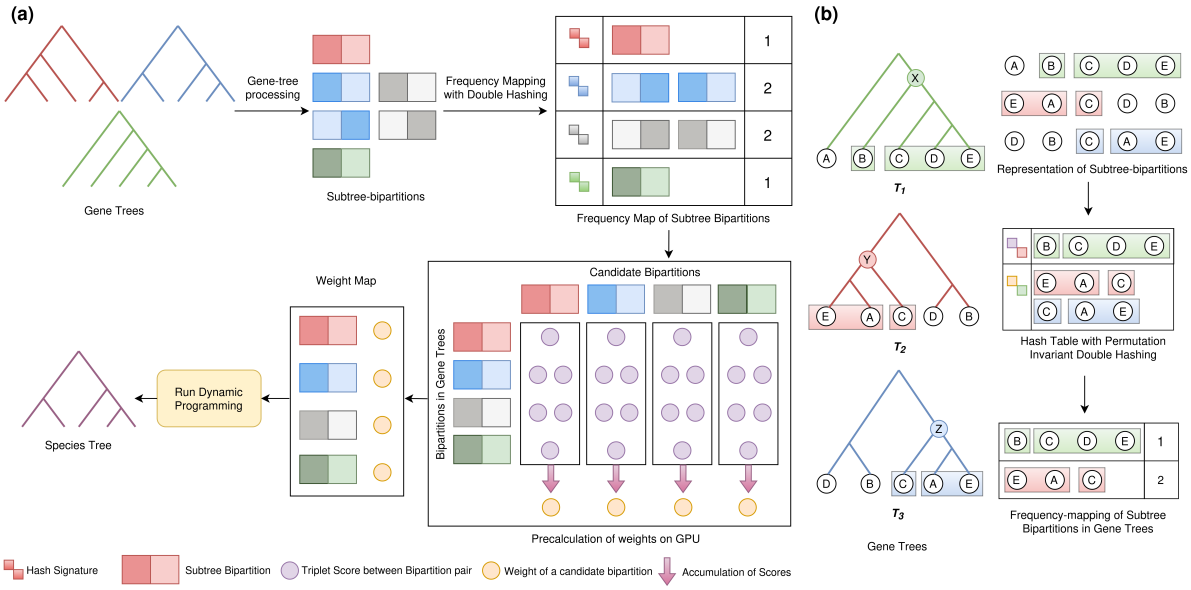

Figure 1: (a) Algorithmic framework of STELAR-X integrating double-hashing-based frequency mapping, GPU-accelerated weight precomputation, and dynamic programming for species tree inference. (b) Illustration of frequency mapping using double hashing. Three subtree bipartitions $(B|CDE)$, $(EA|C)$, and $(CA|E)$ are considered from three gene trees. Since the bipartitions $(EA|C) \in \mathcal{SBP}(T_2)$ and $(C|AE) \in \mathcal{SBP}(T_3)$ are equivalent, they share the same hash signature, yielding a frequency of 2.

map, STELAR-X precomputes the weight $w_{\mathcal{G}}(x)$ for all $x \in \mathcal{CB}$ by summing contributions from all unique subtree-bipartitions. This embarrassingly parallel reduction is offloaded to the GPU (Section 2.8). Finally, we build a cluster→subtree-bipartition map and run the DP (Equation 1) using table lookups for precomputed weights (Section 2.7). Figure 1a shows the overall algorithmic flow of STELAR-X.

2.4 Compact tuple representation of subtree bipartitions

For each gene tree g_i we form an array $\mathcal{A}_i$ of its leaf labels in a postorder traversal. Any internal node in g_i induces a bipartition $(A|B)$ where A and B, by construction, correspond to two adjacent contiguous subarrays of $\mathcal{A}_i$. Let $\mathcal{A}_{i,l,r}$ denote the subarray of $\mathcal{A}_i$ from indices l to r (i.e., $\mathcal{A}_i[l : r]$). Then, if $A = \mathcal{A}_{i,l_1,r_1}$ and $B = \mathcal{A}_{i,l_2,r_2}$, the bipartition $(A|B)$ can be represented as the tuple (i, l_1, r_1, l_2, r_2), where $l_1 < r_1 < l_2 < r_2$. In fact, since $l_2 = r_1 + 1$, four integers suffice to store the bipartition.

This tuple representation reduces storage from $O(n)$ bits per bipartition to a constant number of machine words, giving an overall space of $O(|\mathcal{UGB}|)$ words. Because $|\mathcal{UGB}| = O(nk)$, the storage for the unique bipartitions is $O(nk)$ words, while the usual bitset representations would take $O(n^2k)$ space. We present an example of this representation in Supplementary Material (Section 1).

2.5 Frequency mapping of subtree-bipartitions using permutation-invariant and associative double hashing

We now describe the efficient construction of the frequency map $f_{\mathcal{G}}(x)$ of a subtree-bipartition $x \in \mathcal{UGB}$ across all gene trees in $\mathcal{G}$ under the compact integer-tuple representation.

Equivalence of bipartitions. Let P and Q be sets of taxa; we write $P \sim Q$ if they contain the same taxa, regardless of order. Two subtree-bipartitions $x = (A_1 \mid B_1)$ and $y = (A_2 \mid B_2)$ are equivalent $(x \equiv y)$ if either $A_1 \sim A_2$ and $B_1 \sim B_2$, or $A_1 \sim B_2$ and $B_1 \sim A_2$. Thus, equivalence is invariant under both taxon ordering and side swapping.

For the tuple representation of bipartitions $x = (i_x, l_{1x}, r_{1x}, l_{2x}, r_{2x})$ and $y = (i_y, l_{1y}, r_{1y}, l_{2y}, r_{2y})$, equivalence is possible only when $i_x \neq i_y$ (i.e., two bipartitions originate from different gene trees). Bipartitions within the same gene tree $(i_x = i_y)$ are either disjoint or one contains the other. A naive cross-tree comparison to detect equivalence would require $O(n)$ time per pair. To eliminate this overhead, we adopt a hashing-based approach that detects equivalence in $O(1)$ time.

Permutation-invariant double hashing. Let ϕ be permutation-invariant and associative hash function that maps a set of taxa to a single hash value independent of order. If $x \equiv y$, by definition of equivalence, either $\mathcal{A}_{i_x,l_{1x},r_{1x}} \sim \mathcal{A}_{i_y,l_{1y},r_{1y}}$ and $\mathcal{A}_{i_x,l_{2x},r_{2x}} \sim \mathcal{A}_{i_y,l_{2y},r_{2y}}$, or the similar relations with the two sides swapped. Since ϕ is permutation-invariant, these relations imply

$$\phi(\mathcal{A}_{i_x,l_{1x},r_{1x}}) = \phi(\mathcal{A}_{i_y,l_{1y},r_{1y}}) \quad \text{and} \quad \phi(\mathcal{A}_{i_x,l_{2x},r_{2x}}) = \phi(\mathcal{A}_{i_y,l_{2y},r_{2y}}),$$

or the corresponding equalities under a side swap. Although equal hash values suggest possible equivalence, hash collisions prevent a strict guarantee. To make the probability of collisions negligible, we employ two safeguards.

First, we apply double hashing: two independent permutation-invariant and associative hash functions, ϕ_1 and ϕ_2, produce a composite signature (ϕ_1, ϕ_2) for any set of taxa. Second, we apply a per-element hash function $\mathcal{H}$ to each taxon identifier, mapping it into $\mathbb{Z}_{2^k}$ via multiplication and bitwise operations modulo 2^k so that 2^k is large. This sparse projection mitigates structural collisions common to contiguous integer labels. For any taxon $t \in \mathcal{X}$, let $\mathrm{id}(t) \in \{0, \ldots, n-1\}$ denote its identifier. For a subset $S \subseteq \mathcal{X}$, we define its set-level signature as $\sigma(S) = \big(\phi_1(\{\mathcal{H}(\mathrm{id}(t)) : t \in S\}),\ \phi_2(\{\mathcal{H}(\mathrm{id}(t)) : t \in S\})\big)$. Consequently, for a bipartition $x = (A \mid B)$, the composite signature is $(\sigma(A), \sigma(B))$.

Now, since the sets A and B represent subarrays of $\mathcal{A}_i$, we need to efficiently compute $\phi_1(\{\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))\}_{l \le j \le r})$ and $\phi_2(\{\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))\}_{l \le j \le r})$ for any required subarray $\mathcal{A}_{i,l,r}$. To this end, we precompute prefix-scan arrays for ϕ_1 and ϕ_2. For an associative hash function ϕ admitting a cancellation operator $\ominus$ (e.g., subtraction for addition, XOR for XOR), we define for each array $\mathcal{A}_i$ the prefix array $\mathcal{P}_i$ where, $\mathcal{P}_{i,m}^{(\phi)} = \phi(\{\,\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))\,\}_{1 \le j \le m})$ for $1 \le m \le n$. Then for any subarray $\mathcal{A}_{i,l,r}$, the hash value of the corresponding taxon set is obtained in constant time as,

$$\phi(\{\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))\}_{l \le j \le r}) \;=\; \mathcal{P}_{i,r}^{(\phi)} \;\ominus\; \mathcal{P}_{i,l-1}^{(\phi)},$$

with the convention that $\mathcal{P}_{i,0}^{(\phi)}$ is the identity element of ϕ. We construct such prefix arrays independently for both ϕ_1 and ϕ_2, enabling $O(1)$ evaluation of any required subarray hash value. Further details on the hashing scheme, along with an example hash-signature computation, are provided in Supplementary Material (Section 2).

This combination of double hashing and sparse integer mapping makes collisions vanishingly rare (see the collision analysis in Theorem 2.1). In our implementation, ϕ_1 is addition and ϕ_2 is XOR over $\mathbb{Z}_{2^k}$, and we observed no hash collisions over extensive evaluation. All arithmetic is performed modulo 2^k so that 2^k is large (we use $k = 64$). A modulus of the form 2^k is convenient for two reasons: it preserves the associativity and group structure of bitwise operations, and it is computationally efficient because it naturally arises from the wrap-around behavior of fixed-width integer types. Finally, the frequency $f_{\mathcal{G}}(x)$ of each unique bipartition $x \in \mathcal{UGB}$ is derived by inserting the computed signatures into a hash table and counting the number of bipartitions that map to the same hash value. Figure 1b schematically illustrates this process.

Theorem 2.1 (Collision Probability Bound)**.** *Let $\sigma(S) = (\phi_1(\{r_t : t \in S\}), \phi_2(\{r_t : t \in S\}))$ be the double-hash signature of a set S of taxa, where ϕ_1, ϕ_2 are independent permutation-invariant hash functions modulo M and each $r_t = \mathcal{H}(\mathrm{id}(t))$ is independently and uniformly distributed in $\mathbb{Z}_M$. Among B total bipartitions, the probability that at least one collision will occur is bounded by*

$$\Pr(\exists\ \textit{collision}) \le \frac{B^2}{M^2}.$$

Proof. See Supplementary Material (Section 3, Theorem 3.1)

Thus, to keep this probability below a prescribed tolerance δ, it suffices to ensure that $M \ge \frac{B}{\sqrt{\delta}}$. Hence, choosing M ($M = 2^k$) sufficiently larger than B ($B = \Theta(nk)$) renders global collisions negligible in practice.

2.6 Precomputation of Bipartition Weights

We next justify and describe the precomputation of weights associated with all candidate bipartitions in $\mathcal{CB}$. This step is essential for the dynamic programming (DP) phase of STELAR, whose state space is defined by the recursive formulation in Equation (1). The DP begins with the full taxon set $\mathcal{X}$ and recursively partitions each cluster using subtree-bipartitions. Consequently, every transition depends on the availability of subtree-bipartition weights. We now show in Theorem 2.2 that, in the constrained version of STELAR, this process inevitably requires the weights of all subtree-bipartitions in $\mathcal{CB}$.

Theorem 2.2. *For the constrained version (i.e., $\mathcal{CB} = \mathcal{UGB}$), the DP state space of STELAR spans every bipartition $x \in \mathcal{CB}$.*

Proof. The proof is provided in the Supplementary Material (Section 4, Theorem 4.1). □

Precomputing these weights therefore does not introduce redundant computation. Moreover, even when $\mathcal{UGB} \subsetneq \mathcal{CB}$, precomputing weights for all $x \in \mathcal{CB}$ remains practical, provided that $|\mathcal{CB}|$ is not asymptotically larger than $|\mathcal{UGB}|$.

Weight formulation. Let $x = (A_1 \mid B_1) \in \mathcal{CB}$ be a candidate bipartition and $y = (A_2 \mid B_2) \in \mathcal{SBP}(g_i)$ a bipartition from gene tree g_i. Following [22], the number of triplets simultaneously mapped to both x and y is

$$\mathcal{M}(x, y) = \mathcal{NT}(|A_1 \cap A_2|, |B_1 \cap B_2|) + \mathcal{NT}(|A_1 \cap B_2|, |A_2 \cap B_1|) \tag{2}$$

where $\mathcal{NT}(n_1, n_2)$ is the number of triplets mapped to a bipartition $(A \mid B)$ with $|A| = n_1$, $|B| = n_2$:

$$\mathcal{NT}(n_1, n_2) = \binom{n_1}{2}\binom{n_2}{1} + \binom{n_2}{2}\binom{n_1}{1} = \frac{n_1 n_2 (n_1 + n_2 - 2)}{2}. \tag{3}$$

The total weight of x across all gene trees is then $w_{\mathcal{G}}(x) = \sum_{i=1}^{k} \sum_{y \in \mathcal{SBP}(g_i)} \mathcal{M}(x, y)$. Using the precomputed frequency mapping $f_{\mathcal{G}}(y)$ of unique bipartitions $y \in \mathcal{UGB}$ produced by the double hash table, this can be written more efficiently as,

$$w_{\mathcal{G}}(x) = \sum_{y \in \mathcal{UGB}} f_{\mathcal{G}}(y)\, \mathcal{M}(x, y). \tag{4}$$

Note that each $M(x, y)$ requires computing the four intersection cardinalities between clusters $|A_1 \cap A_2|, |B_1 \cap B_2|, |A_1 \cap B_2|, |A_2 \cap B_1|$ (Equations (2) and (3)).

Efficient intersection counting using tuples. To avoid memory-intensive bitset operations, we employ our proposed integer-tuple representation of subtree-bipartitions. Let $X = \mathcal{A}_{i,l_x,r_x}$ and $Y = \mathcal{A}_{j,l_y,r_y}$ denote two clusters ($i \neq j$). We have $|X| = r_x - l_x + 1$ and $|Y| = r_y - l_y + 1$. The intersection $|X \cap Y|$ is computed by iterating over each taxon in the smaller cluster and checking taxon membership in the other cluster. To optimize this membership testing, we precompute index maps $\pi_i(v)$, where $\pi_i(v)$ gives the position of taxon v in the post order array representation $\mathcal{A}_i$ of g_i. Thus, for each taxon $v \in \mathcal{A}_{i,l_x,r_x}$, we check whether $l_y \leq \pi_j(v) \leq r_y$ (assuming $|X| < |Y|$) and accumulate the intersection count. This procedure runs in $O(\min(|X|, |Y|))$ time per intersection.

$$|X \cap Y| = \sum_{v \in \mathcal{A}_{i,l_x,r_x}} \mathbf{1}_{[\, l_y \leq \pi_j(v) \leq r_y \,]}.$$

Supplementary Section 1 presents an example showing the computation of $M(x, y)$. Assuming balanced gene trees, the total time for precomputing all $w_{\mathcal{G}}(x)$ over $\mathcal{UGB}$ is $O(n^2k^2)$ (Theorem 2.3). Therefore, this is the dominant part of the overall running time of STELAR-X (see Section 2.9). This justifies isolating the weight precomputation as a separate stage, which we further accelerate using GPU parallelization (see Section 2.8).

Theorem 2.3. *For the constrained version (i.e., $\mathcal{CB} = \mathcal{UGB}$), if each gene tree is a perfectly balanced rooted binary tree with $n = 2^d$ taxa, then the precomputation of weights requires $O(n^2k^2)$ time.*

Proof. The proof is provided in the Supplementary Material (Section 4, Theorem 4.2). □

Although the analysis assumes balanced binary trees for mathematical simplicity, the practical runtime mainly depends on the smaller of the two subtrees in each bipartition pair. As a result, the observed running time typically remains close to the theoretical bound. In the extreme case of highly imbalanced fully caterpillar-like gene trees, the precomputation requires $\mathcal{O}(n^3k^2)$ time (Theorem 2.4). In practice, however, this effect is alleviated because the number of unique bipartitions $|\mathcal{UGB}|$ is generally much smaller than nk, making the balanced-tree assumption a good approximation of empirical behavior.

Theorem 2.4. *Assume $\mathcal{CB} = \mathcal{UGB}$. If each gene tree is a rooted caterpillar on n taxa, then the precomputation of weights (evaluating $\mathcal{M}(x, y)$ for all ordered pairs $x, y \in \mathcal{UGB}$) requires $\mathcal{O}(n^3k^2)$ time.*

Proof. The proof is provided in the Supplementary Material (Section 4, Theorem 4.3). □

2.7 Execution of Dynamic Programming

As the final stage, we execute the dynamic programming (DP) procedure defined by Equation (1). Each DP state corresponds to a cluster of taxa, and transitions require identifying all subtree-bipartitions $x \in \mathcal{CB}$ that partition the taxa in the current cluster. To enable efficient lookup, we precompute a state-space mapping Ψ from clusters $C \subseteq \mathcal{X}$ to the set of candidate bipartitions.

$$\Psi : \{\text{clusters of taxa}\} \longrightarrow \mathcal{CB}, \qquad \Psi(C) = \{\, x = (A|B) \in \mathcal{CB} \mid A \cup B = C \,\}.$$

For each subtree-bipartition $x = (A|B) \in \mathcal{CB}$, we define its associated cluster $C = A \cup B$ and construct a mapping. Here, again, we employ the double-hashing scheme as before and avoid the memory-intensive bitset representation. Each cluster C is encoded using the pair of hash signatures $(\phi_1(C), \phi_2(C))$.

During DP (Eq. 1) we enumerate $\Psi(C)$ to obtain allowed subtree-bipartitions and fetch $w_{\mathcal{G}}(A|B)$ in $O(1)$ time per bipartition using the precomputed weight map discussed earlier (Section 2.6). Because both the state space (clusters) and candidate bipartitions are encoded via compact signatures, memory demand remains $O(|\mathcal{CB}| + |\mathcal{UGB}|) = O(nk)$. The DP itself then runs in time proportional to the number of state transitions (i.e., the total number of candidate bipartitions considered), plus any bookkeeping overhead.

2.8 Parallelization

We next identify the stages of our algorithm that are amenable to CPU and GPU parallelization and describe their implementations on multicore CPUs and GPUs. We assume $T_{\mathcal{C}}$ CPU threads and $T_{\mathcal{G}}$ GPU threads, representing the degrees of parallelism under ideal load balancing and negligible communication overhead.

CPU Parallelization CPU parallelism is applied primarily to the processing of the k input gene trees. The collection of gene trees is divided into $T_{\mathcal{C}}$ approximately equal partitions, each assigned to an independent thread running in parallel. Within each thread, the parsing, subtree–bipartition extraction, and frequency mapping steps are executed sequentially on the assigned subset. When necessary, thread-safe containers and synchronization primitives are employed to ensure correct aggregation of shared data across threads.

GPU Parallelization The most computationally intensive stage, computing the weight $w_{\mathcal{G}}(x)$ for each candidate bipartition $x \in \mathcal{CB}$, requires evaluating $\mathcal{M}(x, y)$ for all $y \in \mathcal{UGB}$. This corresponds to executing the same operation for $O(|\mathcal{CB}||\mathcal{UGB}|)$ bipartition pairs, followed by accumulation along the $\mathcal{UGB}$ dimension to obtain the weights of $O(|\mathcal{CB}|)$ candidate bipartitions. Therefore, we offload this highly regular computation pattern to $T_{\mathcal{G}}$ GPU threads. We transfer the post-order leaf traversal arrays, the frequency map, and the index map to GPU. For each candidate bipartition $x \in \mathcal{CB}$, each GPU thread computes the intersection counts for a distinct (x, y) pair, and results are accumulated by a reduction across $\mathcal{UGB}$ to obtain the aggregated weights. Additional optimizations, such as shared-memory tiling and coalesced access patterns, are used to reduce memory latency and improve throughput. The resulting weight map of size $O(|\mathcal{CB}|)$ feeds the dynamic programming algorithm.

2.9 Complexity Analysis

Theorem 2.5 (Running time and memory complexity)**.** *For n taxa, k gene trees, $T_{\mathcal{C}}$ CPU threads and $T_{\mathcal{G}}$ GPU threads, STELAR-X runs in $O\left(\frac{nk}{T_{\mathcal{C}}} + \frac{n^2k^2}{T_{\mathcal{G}}} + |\mathcal{CB}|\right)$ time (under the balance gene tree assumption), $O(nk + |\mathcal{UGB}| + |\mathcal{CB}|)$ CPU memory, and $O(|\mathcal{UGB}| + |\mathcal{CB}|)$ GPU memory. In the worst case, when all gene trees are caterpillars, the running time becomes $O\left(\frac{nk}{T_{\mathcal{C}}} + \frac{n^3k^2}{T_{\mathcal{G}}} + |\mathcal{CB}|\right)$.*

Proof. A detailed derivation of these bounds is provided in Supplementary Material (Section 5, Theorem 5.1). □

In practice, the precomputation complexity depends on $|\mathcal{UGB}|$ and $|\mathcal{CB}|$. Under lower levels of ILS, $|\mathcal{UGB}|$ tends to be substantially smaller than nk, leading to a reduced effective running time. Consequently, the degree of ILS directly influences the practical computational cost of the algorithm.

In case of memory usage, since both $|\mathcal{UGB}|$ and $|\mathcal{CB}|$ can be bounded by $O(nk)$ in the constrained case, the overall CPU and GPU memory consumption scales as $O(nk)$. If only the unique bipartitions

were represented using traditional bitset-based representations for weight computation, the memory usage would depend on the number of distinct bipartitions, and thus on the degree of ILS. In contrast, STELAR-X employs a compact tuple-based representation throughout, achieving asymptotically optimal memory usage of $O(nk)$ that matches the input size, irrespective of the ILS level. This claim regarding the impact of ILS on running time and memory usage is further confirmed in our empirical results (see Section 3.2 and Supplementary Figure S2).

3 Results

3.1 Experimental Setup

We evaluated STELAR-X on both simulated and empirical datasets. To assess scalability, we compared its performance against ASTRAL-MP [23], the only existing multi-parallel summary method. For accuracy assessment, we further compared STELAR-X with ASTRAL-MP [23], and WQFM-TREE [28]. All experiments were conducted on a Linux server equipped with an NVIDIA RTX 4090 GPU (24 GB VRAM) and an AMD EPYC 7742 processor with 32 active cores and 129 GB of RAM. Each analysis was allocated a maximum runtime of 48 hours.

Simulated Datasets. For the scalability analysis, we simulated large phylogenomic datasets under the Multispecies Coalescent (MSC) model using SimPhy [29]. In one set of experiments, we fixed the number of gene trees (k) at 1,000 and varied the number of taxa (n) from 1,000 to 100,000. In another set, we fixed the number of taxa at 1,000 and varied the number of gene trees from 1,000 to 100,000. All datasets were generated under four model conditions corresponding to different levels of incomplete lineage sorting (ILS), ranging from low (ILS-L1) to medium-high (ILS-L4). The average gene tree discordance corresponding to these ILS levels is reported in Supplementary Table S2. Additionally, to evaluate accuracy, we used smaller simulated datasets containing 37, 100, 200, and 500 taxa since existing methods do not scale to very large datasets. Simulation parameters are provided in Supplementary Material (Section 6).

Biological Datasets. We analyzed two biological datasets: the avian dataset comprising 48 taxa and 14,446 genes [30], and an extended Avian dataset comprising 363 taxa and 14,972 genes [31].

3.2 Results on Scalability Analysis

To assess the scalability of STELAR-X, we performed experiments on simulated datasets with large numbers of taxa and gene trees under medium ILS level (ILS-L2). We report the running time, CPU memory usage, and GPU VRAM consumption averaged over 5 replicates. In the first set of experiments, we fixed the number of gene trees at $k = 1{,}000$ and varied the number of taxa n from 1,000 to 100,000. Within our resource limits (128 GB CPU memory and a 48-hour of analyses), ASTRAL-MP could be executed only up to 10,000 taxa, consistent with the scalability reported in the original ASTRAL-MP study. Consequently, we compared STELAR-X and ASTRAL-MP on datasets up to this size (Figure 2a).

Across all tested cases, STELAR-X consistently and substantially outperformed ASTRAL-MP in both running time and CPU memory usage, while exhibiting a comparable trend in GPU VRAM utilization. Specifically, STELAR-X analyzed the dataset with 10,000 taxa in only 77 seconds, achieving a 712× speedup over ASTRAL-MP and a 7.5× reduction in CPU memory. Notably, ASTRAL-MP reached the resource limit (128 GB of CPU memory) at 7,500 and 10,000 taxa, whereas STELAR-X required only 11.9 GB and 16.4 GB for these two cases. As evident in Figure 2, both the running time and memory growth rates of STELAR-X are markedly lower than those of ASTRAL-MP, reflecting the asymptotically optimal $O(nk)$ space usage achieved by our new tuple-based representation and algorithmic re-engineering. Given the observed trend, ASTRAL-MP would require prohibitive time and memory to analyze datasets beyond 10,000 taxa, while STELAR-X remains computationally feasible well beyond this range.

Next, we evaluated the scalability with respect to the number of genes while fixing the number of species at $n = 1{,}000$. ASTRAL-MP reached the 128 GB memory limit at 50,000 genes. As shown in Figure 2b, STELAR-X was consistently faster and more memory-efficient across all tested dataset sizes. For example, on the 50,000-gene dataset, ASTRAL-MP required 75 minutes and 123.5 GB of CPU RAM, whereas STELAR-X completed the analysis in 140 seconds using 60.7 GB of CPU RAM—an approximate 32.2× speedup with a 2× reduction in memory usage. GPU VRAM usage remained moderate (0.4∼0.8 GB) for both methods, with STELAR-X showing slightly higher requirements.

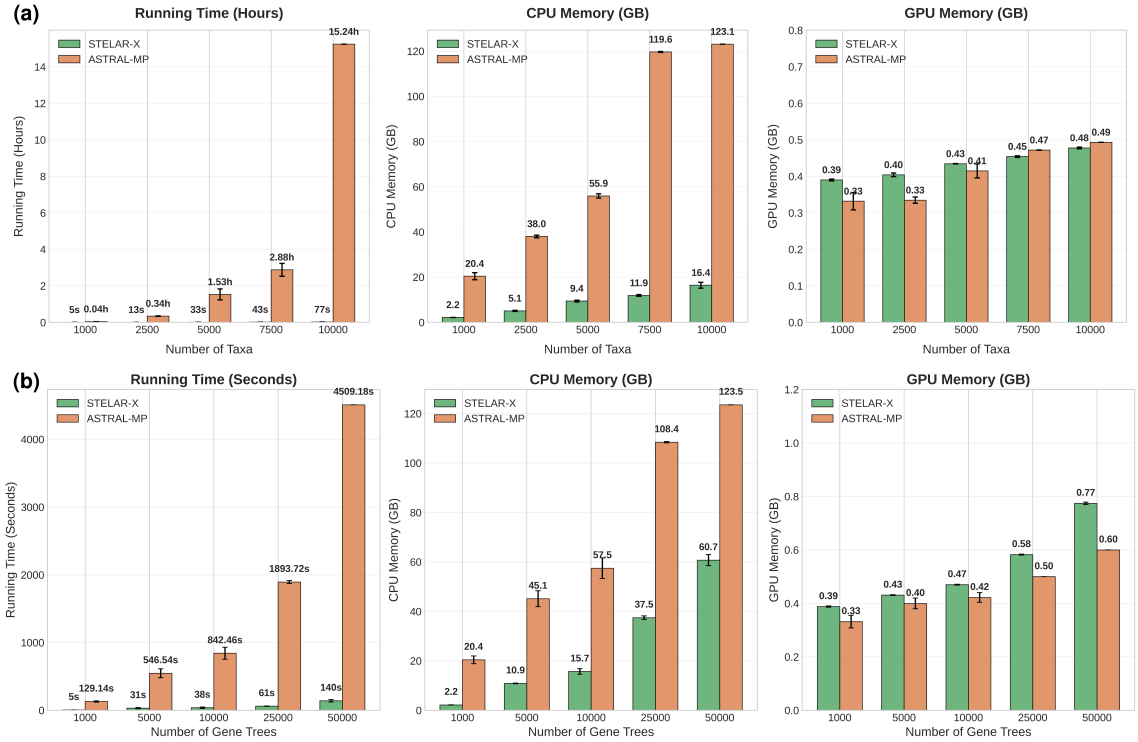

Figure 2: Average running time, CPU memory usage, and GPU VRAM requirements of STELAR-X and ASTRAL-MP over 5 replicates (except for 10,000-taxon and 50,000-gene conditions where we ran ASTRAL-MP on one replicate due to its high computational demands). (a) Datasets with 1,000 gene trees and 1,000–10,000 taxa, (b) Datasets with 1,000 taxa and 1,000–50,000 genes.

Scaling up the number of taxa and gene trees. To further evaluate our performance at a larger scale (beyond the limits of existing methods), we evaluated STELAR-X on datasets with up to 100,000 taxa and 100,000 genes. In the first experiment, we fixed $k = 1{,}000$ and increased n from 1,000 to 100,000 (Figure 3a). Consistent with our theoretical expectations, CPU memory usage exhibited an approximately linear growth, while GPU VRAM usage followed an almost perfectly linear trend. STELAR-X successfully analyzed the 100,000-taxon dataset in about 8.5 hours using 86 GB of CPU RAM and 1.3 GB of GPU VRAM. For comparison, ASTRAL-MP required 120 GB of CPU memory to process only 7,500 taxa, and extrapolations from its growth curve suggest that analyzing 100,000 taxa would be practically infeasible for ASTRAL-MP. In contrast, given the observed linear scaling of memory and substantially faster runtime, STELAR-X is expected to handle substantially larger datasets (>100,000 taxa) on machines with a few hundred gigabytes of RAM (e.g., 256 GB) and modest multi-day runtimes.

In the second experiment, we fixed n at 1,000 and varied k from 1,000 to 100,000 (Figure 3b). Notably, STELAR-X scaled even more efficiently with increasing numbers of genes, exhibiting nearly linear growth in runtime. Remarkably, it analyzed 100,000 genes in just 240 seconds using 106 GB CPU RAM– an unprecedented scale not achievable by ASTRAL-MP or other existing summary methods. CPU RAM and GPU VRAM usage remained low and continued to follow the same linear pattern.

Impact of varying ILS. Next, we analyzed the effect of varying levels of ILS on the running time and memory consumption of STELAR-X using simulated datasets comprising 10,000–40,000 taxa with 1,000 genes and 1,000 taxa with 1,000–100,000 genes under four model conditions, ranging from low (ILS-L1) to medium-high (ILS-L4) levels of ILS (see Supplementary Figure S2). Consistent with our complexity analysis (Section 2.9), the running time increases with higher levels of ILS, whereas CPU RAM and GPU VRAM usage remain nearly constant, showing only minor variations due to small constant factors, thus maintaining the same linear scaling trend. For example, with 40,000 taxa, the running time increased by 3.7× from ILS-L2 to ILS-L4, while CPU RAM and GPU VRAM consumption rose only slightly, from 49 GB to 53 GB and from 0.78 GB to 0.85 GB, respectively. As the number of gene trees increased, this trend became more pronounced, with memory usage showing only negligible variation across ILS levels.

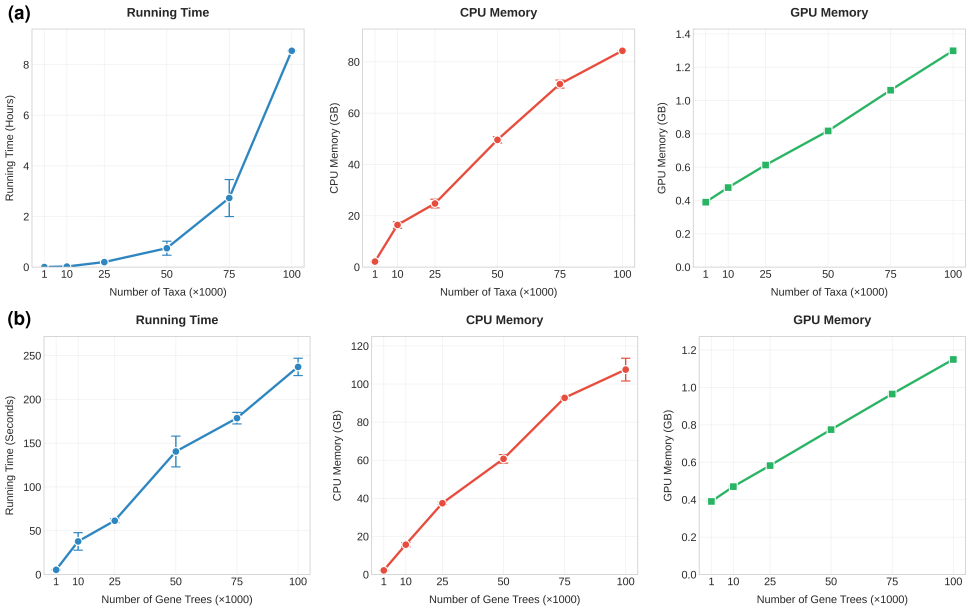

Figure 3: Average running time, CPU RAM, and GPU VRAM usage of STELAR-X on large datasets over 5 replicates (except for the 100,000-taxon case where we analyzed one replicate). (a) 10,000 to 100,000 taxa with 1,000 gene trees, (b) 1,000 taxa with 10,000 to 100,000 gene trees.

3.3 Results on Accuracy of Inference

We conducted experiments on simulated datasets with 100, 200, and 500 taxa and compared the average performance of STELAR-X with the leading quartet-based methods ASTRAL-MP and WQFM-TREE [28]. We also analyzed a biologically-based 37-taxon simulated mammalian dataset previously studied in [32]. On the 37-taxon datasets (Figures 4a,b), STELAR-X showed accuracy comparable to both ASTRAL-MP and wQFM-TREE across varying levels of ILS and gene tree estimation error (varied by sequence length), with no statistically significant differences among methods ($p > 0.05$). STELAR-X was slightly more accurate than ASTRAL-MP on model conditions with high (0.5X) and moderate (1X) levels of ILS. wQFM-TREE achieved the highest accuracy in almost all cases. Similar patterns were observed for the 100–500-taxon datasets (Figure 4c), although STELAR-X displayed slightly higher RF rates on 100 and 200 taxa.

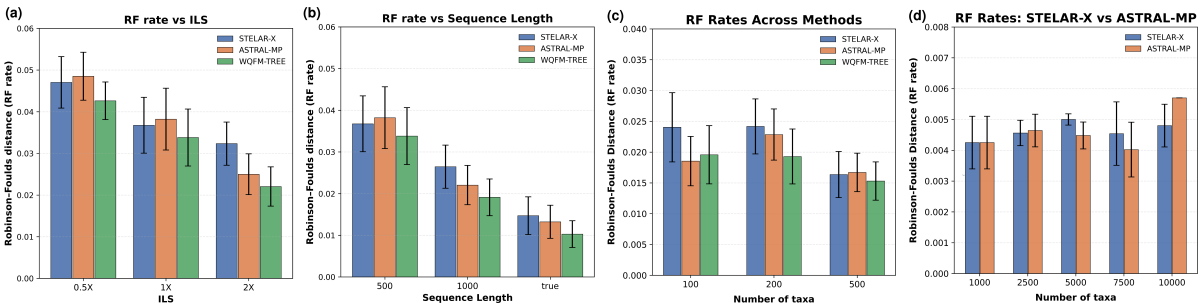

Figure 4: (a)-(b) Average RF rates of STELAR-X, ASTRAL-MP, and WQFM-TREE on 37-taxon simulated datasets over 10 replicates (except for ASTRAL-MP on 10,000 taxa. (a) varying ILS (b) varying sequence lengths (c) 100–500 taxa dataset (d) RF rates on 1,000–10,000 taxa.

To further assess accuracy at larger scales, we compared STELAR-X and ASTRAL-MP on datasets spanning 1,000–10,000 taxa (Figure 4d). wQFM-TREE could not be run on these datasets within our resource limits and was therefore excluded. For the 1,000-, 2,500-, and 10,000-taxon datasets, STELAR-X matched or exceeded ASTRAL-MP in accuracy, while modestly higher RF error rates were observed in the other two cases. Nonetheless, overall RF errors remained low, as the true gene trees (instead of estimated gene trees) under moderate levels of ILS were used, as sequence simulation and gene tree

estimation for these large dataset sizes were computationally prohibitive.

3.4 Results on Biological Datasets

We evaluated the avian dataset comprising 48 taxa and 14,446 genes [30]. STELAR-X analyzed the avian dataset in only 12 seconds, achieving a 30× speedup over ASTRAL-MP. As shown in Figure 5, both STELAR-X and ASTRAL-MP accurately reconstructed the major clades Australaves, Afroaves, Cuprimulgimorphae, Core Waterbirds and Columbimorphae. None of the methods could reconstruct the Cursores and Otidimorphae clades. To demonstrate scalability in recent large phylogenomic datasets, we also analyzed an extended avian dataset comprising 363 taxa and 14,972 genes [31] (Supplementary Table S4 and Figure S3).

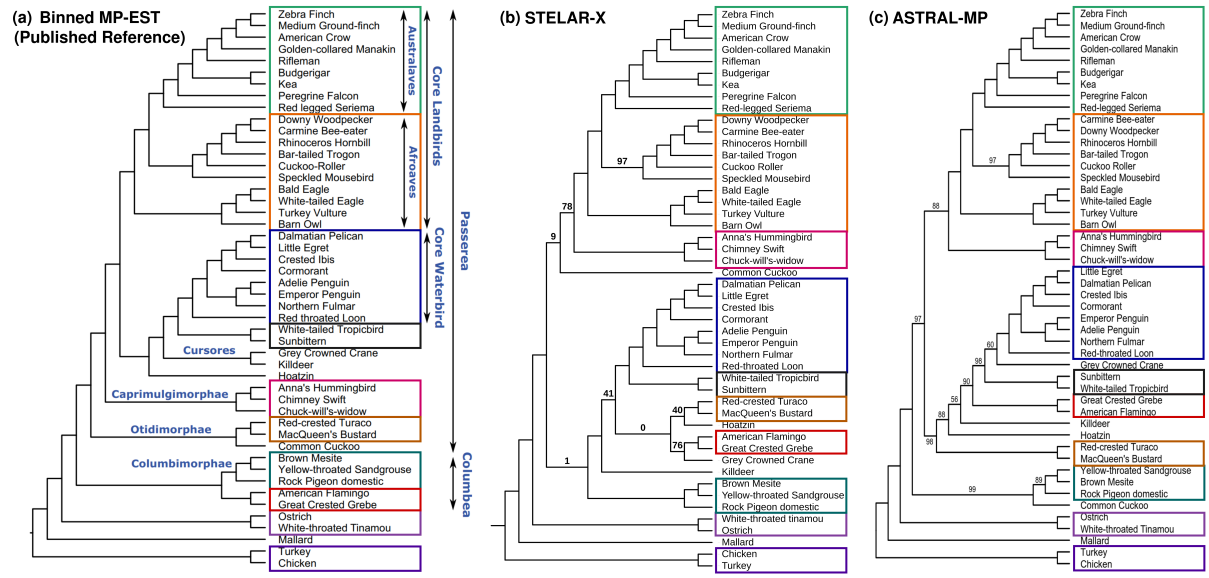

Figure 5: (a)-(c) Estimated trees on the avian dataset with identical clades indicated. Branch supports (BS) based on quartet-based local posterior probabilities [33] are shown on branches. All BS values are 100% except where noted.

4 Conclusions

We presented a statistically consistent coalescent-based summary method STELAR-X, a substantially redesigned and computationally efficient extension of the triplet-based summary method STELAR. At the core of STELAR-X is a compact integer-tuple representation of subtree-bipartitions that replaces the bitset encodings used in established methods such as ASTRAL. This redesign enables extensive algorithmic optimizations and re-engineering, including efficient pre-computation of bipartition weights using a customized double-hashing technique and a streamlined dynamic programming strategy with greatly reduced computational overhead. By further utilizing GPU acceleration for highly regular weight pre-computations, STELAR-X delivers substantial improvements in both running time and memory efficiency. Our experiments demonstrate that STELAR-X maintains competitive accuracy with ASTRAL-MP and wQFM-TREE. Most notably, STELAR-X delivers unprecedented scalability, and can infer species trees from datasets comprising 100,000 taxa and 100,000 genes in few hours of computation using only 100 GB of memory, whereas current approaches encounter prohibitive memory and runtime limitations well below this scale. Moreover, STELAR-X can be used within divide-and-conquer frameworks [24, 25] to analyze millions of taxa. This opens the door to statistically consistent phylogenomic analyses for ambitious community initiatives to resolve the tree of life across diverse lineages, such as constructing a comprehensive tree of life for all ∼330,000 known species of flowering plants (angiosperms) [34, 35].

Beyond scalability, the new data structures and algorithmic redesign introduced here can benefit other coalescent-based tools, including ASTRAL. As a next step, we will extend STELAR-X to unrooted and multi-copy gene family trees to account for both orthology and paralogy, and evaluate it on large empirical datasets spanning complex evolutionary histories.

References

[1] Erich D Jarvis, Siavash Mirarab, Andre J Aberer, Bo Li, Peter Houde, Cai Li, Simon YW Ho, Brant C Faircloth, Benoit Nabholz, Jason T Howard, et al. Whole-genome analyses resolve early branches in the tree of life of modern birds. Science, 346(6215):1320–1331, 2014.

[2] Norman J Wickett, Siavash Mirarab, Nam Nguyen, Tandy Warnow, Eric Carpenter, Naim Matasci, Saravanaraj Ayyampalayam, Michael S Barker, J Gordon Burleigh, Matthew A Gitzendanner, et al. Phylotranscriptomic analysis of the origin and early diversification of land plants. Proceedings of the National Academy of Sciences, 111(45):E4859–E4868, 2014.

[3] R. R. Hudson. Testing the constant-rate neutral allele model with protein sequence data. Evolution, 37:203 – 217, 1983.

[4] M. Nei. Stochastic errors in dna evolution and molecular phylogeny. In In H. Gershowitz, D. L. Rucknagel, and R. E. Tashian, editors, Evolutionary Perspectives and the New Genetics, pages 133 – 147, 1986.

[5] M. Nei. Molecular evolutionary genetics. New York, 1987. Columbia University Press.

[6] N. Rosenberg. The Probability of Topological Concordance of Gene Trees and Species Trees. Theoretical Population Biology, 61(2):225–247, March 2002.

[7] Sebastien Roch and Mike Steel. Likelihood-based tree reconstruction on a concatenation of aligned sequence data sets can be statistically inconsistent. Theoretical Population Biology, 100:56–62, 2015.

[8] J H Degnan, M DeGiorgio, D Bryant, and N A Rosenberg. Properties of consensus methods for inferring species trees from gene trees. Systematic Biology, 58:35–54, 2009.

[9] L S Kubatko and J H Degnan. Inconsistency of phylogenetic estimates from concatenated data under coalescence. Systematic Biology, 56:17, 2007.

[10] J Heled and A J Drummond. Bayesian inference of species trees from multilocus data. Molecular Biology and Evolution, 27:570–580, 2010.

[11] A.J. Drummond and A. Rambaut. BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evolutionary Biology, 7:214, 2007.

[12] L Liu. BEST: Bayesian estimation of species trees under the coalescent model. Bioinformatics, 24:2542–2543, 2008.

[13] Julia Chifman and Laura Kubatko. Quartet from snp data under the coalescent model. Bioinformatics, 30(23):3317–3324, 2014.

[14] Liang Liu, Lili Yu, and Scott V Edwards. A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evolutionary Biology, 10:302, 2010.

[15] Siavash Mirarab, Rezwana Reaz, Md S Bayzid, Théo Zimmermann, M Shel Swenson, and Tandy Warnow. ASTRAL: genome-scale coalescent-based species tree estimation. Bioinformatics, 30(17):i541–i548, 2014.

[16] B Larget, S K Kotha, C N Dewey, and C Ané. BUCKy: Gene tree/species tree reconciliation with the Bayesian concordance analysis. Bioinformatics, 26(22):2910–2911, 2010.

[17] E Mossel and S Roch. Incomplete lineage sorting: consistent phylogeny estimation from multiple loci. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 7(1):166–171, 2011.

[18] L S Kubatko, B C Carstens, and L L Knowles. Stem: Species tree estimation using maximum likelihood for gene trees under coalescence. Bioinformatics, 25:971–973, 2009.

[19] Liang Liu, Lili Yu, Dennis K Pearl, and Scott V Edwards. Estimating species phylogenies using coalescence times among sequences. Systematic Biology, 58(5):468–477, 2009.

[20] Liang Liu and Lili Yu. Estimating species trees from unrooted gene trees. Systematic Biology, 60(5):661–667, 2011.

[21] Pranjal Vachaspati and Tandy Warnow. Astrid: accurate species trees from internode distances. BMC Genomics, 16(10):S3, 2015.

[22] Mazharul Islam, Kowshika Sarker, Trisha Das, Rezwana Reaz, and Md Shamsuzzoha Bayzid. Stelar: A statistically consistent coalescent-based species tree estimation method by maximizing triplet consistency. BMC Genomics, 21(1):1–13, 2020.

[23] John Yin, Chao Zhang, and Siavash Mirarab. ASTRAL-MP: scaling ASTRAL to very large datasets using randomization and parallelization. Bioinformatics, 35(20):3961–3969, 2019.

[24] Md Shamsuzzoha Bayzid, Tyler Hunt, and Tandy Warnow. Disk covering methods improve phylogenomic analyses. BMC Genomics, 15(6):S7, 2014.

[25] Metin Balaban, Yueyu Jiang, Qiyun Zhu, Daniel McDonald, Rob Knight, and Siavash Mirarab. Generation of accurate, expandable phylogenomic trees with udance. Nature biotechnology, 42(5):768–777, 2024.

[26] M. S. Bayzid, S. Mirarab, and T. Warnow. Inferring optimal species trees under gene duplication and loss. In Proc. of Pacific Symposium on Biocomputing (PSB), volume 18, pages 250–261, 2013.

[27] Xinyao Yi. A study of performance programming of cpu, gpu accelerated computers and simd architecture. ArXiv, abs/2409.10661, 2024.

[28] Abdur Rafi, Ahmed Mahir Sultan Rumi, Sheikh Azizul Hakim, Sohaib, Md Toki Tahmid, Rabib Jahin Ibn Momin, Tanjeem Azwad Zaman, Rezwana Reaz, and Md Shamsuzzoha Bayzid. wqfm-tree: highly accurate and scalable quartet-based species tree inference from gene trees. Bioinformatics Advances, 5(1):vbaf053, 2025.

[29] Diego Mallo, Leonardo De Oliveira Martins, and David Posada. Simphy : Phylogenomic simulation of gene, locus, and species trees. Systematic Biology, 65(2):334–344, 11 2015.

[30] Erich D. Jarvis, Siavash Mirarab, Andre J. Aberer, Bo Li, Peter Houde, Cai Li, Simon Y. W. Ho, Brant C. Faircloth, Benoit Nabholz, Jason T. Howard, Alexander Suh, Claudia C. Weber, Rute R. da Fonseca, Jianwen Li, Fang Zhang, Hui Li, Long Zhou, Nitish Narula, Liang Liu, Ganesh Ganapathy, Bastien Boussau, Md. Shamsuzzoha Bayzid, Volodymyr Zavidovych, Sankar Subramanian, Toni Gabaldón, Salvador Capella-Gutiérrez, Jaime Huerta-Cepas, Bhanu Rekepalli, Kasper Munch, Mikkel Schierup, Bent Lindow, Wesley C. Warren, David Ray, Richard E. Green, Michael W. Bruford, Xiangjiang Zhan, Andrew Dixon, Shengbin Li, Ning Li, Yinhua Huang, Elizabeth P. Derryberry, Mads Frost Bertelsen, Frederick H. Sheldon, Robb T. Brumfield, Claudio V. Mello, Peter V. Lovell, Morgan Wirthlin, Maria Paula Cruz Schneider, Francisco Prosdocimi, José Alfredo Samaniego, Amhed Missael Vargas Velazquez, Alonzo Alfaro-Núñez, Paula F. Campos, Bent Petersen, Thomas Sicheritz-Ponten, An Pas, Tom Bailey, Paul Scofield, Michael Bunce, David M. Lambert, Qi Zhou, Polina Perelman, Amy C. Driskell, Beth Shapiro, Zijun Xiong, Yongli Zeng, Shiping Liu, Zhenyu Li, Binghang Liu, Kui Wu, Jin Xiao, Xiong Yinqi, Qiuemei Zheng, Yong Zhang, Huanming Yang, Jian Wang, Linnea Smeds, Frank E. Rheindt, Michael Braun, Jon Fjeldsa, Ludovic Orlando, F. Keith Barker, Knud Andreas Jønsson, Warren Johnson, Klaus-Peter Koepfli, Stephen O'Brien, David Haussler, Oliver A. Ryder, Carsten Rahbek, Eske Willerslev, Gary R. Graves, Travis C. Glenn, John McCormack, Dave Burt, Hans Ellegren, Per Alström, Scott V. Edwards, Alexandros Stamatakis, David P. Mindell, Joel Cracraft, Edward L. Braun, Tandy Warnow, Wang Jun, M. Thomas P. Gilbert, and Guojie Zhang. Whole-genome analyses resolve early branches in the tree of life of modern birds. Science, 346(6215):1320–1331, December 2014.

[31] Josefin Stiller, Shaohong Feng, Al-Aabid Chowdhury, Iker Rivas-González, David A. Duchêne, Qi Fang, Yuan Deng, Alexey Kozlov, Alexandros Stamatakis, Santiago Claramunt, Jacqueline M. T. Nguyen, Simon Y. W. Ho, Brant C. Faircloth, Julia Haag, Peter Houde, Joel Cracraft, Metin Balaban, Uyen Mai, Guangji Chen, Rongsheng Gao, Chengran Zhou, Yulong Xie, Zijian Huang, Zhen Cao, Zhi Yan, Huw A. Ogilvie, Luay Nakhleh, Bent Lindow, Benoit Morel, Jon Fjeldså, Peter A. Hosner, Rute R. da Fonseca, Bent Petersen, Joseph A. Tobias, Tamás Székely, Jonathan David Kennedy, Andrew Hart Reeve, Andras Liker, Martin Stervander, Agostinho Antunes, Dieter Thomas

Tietze, Mads F. Bertelsen, Fumin Lei, Carsten Rahbek, Gary R. Graves, Mikkel H. Schierup, Tandy Warnow, Edward L. Braun, M. Thomas P. Gilbert, Erich D. Jarvis, Siavash Mirarab, and Guojie Zhang. Complexity of avian evolution revealed by family-level genomes. Nature, 629(8013):851–860, April 2024.

[32] Siavash Mirarab, Md Shamsuzzoha Bayzid, Bastien Boussau, and Tandy Warnow. Statistical binning enables an accurate coalescent-based estimation of the avian tree. Science, 346(6215):1250463, 2014.

[33] Erfan Sayyari and Siavash Mirarab. Fast coalescent-based computation of local branch support from quartet frequencies. Molecular Biology and Evolution, 33(7):1654–1668, 04 2016.

[34] Alexandre R. Zuntini, Tom Carruthers, Olivier Maurin, Paul C. Bailey, Kevin Leempoel, Grace E. Brewer, Niroshini Epitawalage, Elaine Françoso, Berta Gallego-Paramo, Catherine McGinnie, Raquel Negrão, Shyamali R. Roy, Lalita Simpson, Eduardo Toledo Romero, Vanessa M. A. Barber, Laura Botigué, James J. Clarkson, Robyn S. Cowan, Steven Dodsworth, Matthew G. Johnson, Jan T. Kim, Lisa Pokorny, Norman J. Wickett, Guilherme M. Antar, Lucinda DeBolt, Karime Gutierrez, Kasper P. Hendriks, Alina Hoewener, Ai-Qun Hu, Elizabeth M. Joyce, Izai A. B. S. Kikuchi, Isabel Larridon, Drew A. Larson, Elton John de Lírio, Jing-Xia Liu, Panagiota Malakasi, Natalia A. S. Przelomska, Toral Shah, Juan Viruel, Theodore R. Allnutt, Gabriel K. Ameka, Rose L. Andrew, Marc S. Appelhans, Montserrat Arista, María Jesús Ariza, Juan Arroyo, Watchara Arthan, Julien B. Bachelier, C. Donovan Bailey, Helen F. Barnes, Matthew D. Barrett, Russell L. Barrett, Randall J. Bayer, Michael J. Bayly, Ed Biffin, Nicky Biggs, Joanne L. Birch, Diego Bogarín, Renata Borosova, Alexander M. C. Bowles, Peter C. Boyce, Gemma L. C. Bramley, Marie Briggs, Linda Broadhurst, Gillian K. Brown, Jeremy J. Bruhl, Anne Bruneau, Sven Buerki, Edie Burns, Margaret Byrne, Stuart Cable, Ainsley Calladine, Martin W. Callmander, Ángela Cano, David J. Cantrill, Warren M. Cardinal-McTeague, Mónica M. Carlsen, Abigail J. A. Carruthers, Alejandra de Castro Mateo, Mark W. Chase, Lars W. Chatrou, Martin Cheek, Shilin Chen, Maarten J. M. Christenhusz, Pascal-Antoine Christin, Mark A. Clements, Skye C. Coffey, John G. Conran, Xavier Cornejo, Thomas L. P. Couvreur, Ian D. Cowie, Laszlo Csiba, Iain Darbyshire, Gerrit Davidse, Nina M. J. Davies, Aaron P. Davis, Kor-jent van Dijk, Stephen R. Downie, Marco F. Duretto, Melvin R. Duvall, Sara L. Edwards, Urs Eggli, Roy H. J. Erkens, Marcial Escudero, Manuel de la Estrella, Federico Fabriani, Michael F. Fay, Paola de L. Ferreira, Sarah Z. Ficinski, Rachael M. Fowler, Sue Frisby, Lin Fu, Tim Fulcher, Mercè Galbany-Casals, Elliot M. Gardner, Dmitry A. German, Augusto Giaretta, Marc Gibernau, Lynn J. Gillespie, Cynthia C. González, David J. Goyder, Sean W. Graham, Aurélie Grall, Laura Green, Bee F. Gunn, Diego G. Gutiérrez, Jan Hackel, Thomas Haevermans, Anna Haigh, Jocelyn C. Hall, Tony Hall, Melissa J. Harrison, Sebastian A. Hatt, Oriane Hidalgo, Trevor R. Hodkinson, Gareth D. Holmes, Helen C. F. Hopkins, Christopher J. Jackson, Shelley A. James, Richard W. Jobson, Gudrun Kadereit, Imalka M. Kahandawala, Kent Kainulainen, Masahiro Kato, Elizabeth A. Kellogg, Graham J. King, Beata Klejevskaja, Bente B. Klitgaard, Ronell R. Klopper, Sandra Knapp, Marcus A. Koch, James H. Leebens-Mack, Frederic Lens, Christine J. Leon, Étienne Léveillé-Bourret, Gwilym P. Lewis, De-Zhu Li, Lan Li, Sigrid Liede-Schumann, Tatyana Livshultz, David Lorence, Meng Lu, Patricia Lu-Irving, Jaquelini Luber, Eve J. Lucas, Manuel Luján, Mabel Lum, Terry D. Macfarlane, Carlos Magdalena, Vidal F. Mansano, Lizo E. Masters, Simon J. Mayo, Kristina McColl, Angela J. McDonnell, Andrew E. McDougall, Todd G. B. McLay, Hannah McPherson, Rosa I. Meneses, Vincent S. F. T. Merckx, Fabián A. Michelangeli, John D. Mitchell, Alexandre K. Monro, Michael J. Moore, Taryn L. Mueller, Klaus Mummenhoff, Jérôme Munzinger, Priscilla Muriel, Daniel J. Murphy, Katharina Nargar, Lars Nauheimer, Francis J. Nge, Reto Nyffeler, Andrés Orejuela, Edgardo M. Ortiz, Luis Palazzesi, Ariane Luna Peixoto, Susan K. Pell, Jaume Pellicer, Darin S. Penneys, Oscar A. Perez-Escobar, Claes Persson, Marc Pignal, Yohan Pillon, José R. Pirani, Gregory M. Plunkett, Robyn F. Powell, Ghillean T. Prance, Carmen Puglisi, Ming Qin, Richard K. Rabeler, Paul E. J. Rees, Matthew Renner, Eric H. Roalson, Michele Rodda, Zachary S. Rogers, Saba Rokni, Rolf Rutishauser, Miguel F. de Salas, Hanno Schaefer, Rowan J. Schley, Alexander Schmidt-Lebuhn, Alison Shapcott, Ihsan Al-Shehbaz, Kelly A. Shepherd, Mark P. Simmons, André O. Simões, Ana Rita G. Simões, Michelle Siros, Eric C. Smidt, James F. Smith, Neil Snow, Douglas E. Soltis, Pamela S. Soltis, Robert J. Soreng, Cynthia A. Sothers, Julian R. Starr, Peter F. Stevens, Shannon C. K. Straub, Lena Struwe, Jennifer M. Taylor, Ian R. H. Telford, Andrew H. Thornhill, Ifeanna Tooth, Anna Trias-Blasi, Frank Udovicic, Timothy M. A. Utteridge, Jose C. Del Valle, G. Anthony Verboom, Helen P. Vonow, Maria S. Vorontsova, Jurriaan M. de Vos, Noor Al-Wattar, Michelle Waycott, Cassiano A. D. Welker, Adam J. White, Jan J. Wieringa, Luis T.

Williamson, Trevor C. Wilson, Sin Yeng Wong, Lisa A. Woods, Roseina Woods, Stuart Worboys, Martin Xanthos, Ya Yang, Yu-Xiao Zhang, Meng-Yuan Zhou, Sue Zmarzty, Fernando O. Zuloaga, Alexandre Antonelli, Sidonie Bellot, Darren M. Crayn, Olwen M. Grace, Paul J. Kersey, Ilia J. Leitch, Hervé Sauquet, Stephen A. Smith, Wolf L. Eiserhardt, Félix Forest, and William J. Baker. Phylogenomics and the rise of the angiosperms. Nature, 629(8013):843–850, April 2024.

[35] William J Baker, Paul Bailey, Vanessa Barber, Abigail Barker, Sidonie Bellot, David Bishop, Laura R Botigué, Grace Brewer, Tom Carruthers, James J Clarkson, et al. A comprehensive phylogenomic platform for exploring the angiosperm tree of life. Systematic biology, 71(2):301–319, 2022.

Supplementary material for STELAR-X: Scaling Coalescent-Based Species Tree Inference to 100,000 Species and Beyond

Anik Saha[1] and Md. Shamsuzzoha Bayzid[1,*]

[1]Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka-1205, Bangladesh
[*]Corresponding author: shams_bayzid@cse.buet.ac.bd

1 Example: Tuple Representation of Bipartitions and Computation of $\mathcal{M}(x, y)$)

We assume 1-based array indexing for this example. Consider two trees T_1 and T_2 defined over the same taxa set $\{A, B, C, D, E\}$:

$$T_1 = (A, (B, (C, (D, E)))), \qquad T_2 = (((E, A), C), (D, B)).$$

We consider the following two subtree-bipartitions (x in T_1 and y in T_2):

$$x = (A_1|B_1) = (B|CDE), \qquad y = (A_2|B_2) = (EA|C).$$

We first illustrate how bipartitions are represented under the post-order leaf traversal. For T_1, the traversal array is $\mathcal{A}_1 = \{A, B, C, D, E\}$. Let an internal node X induce the subtree bipartition $x = (B|CDE)$. This bipartition x is represented as the integer tuple $(1, 2, 2, 3, 5)$, where $i = 1$, $l_x = 2$, $r_x = 2$, $l_y = 3$, and $r_y = 5$. Similarly, for T_2, the traversal array is $\mathcal{A}_2 = \{E, A, C, D, B\}$. Let an internal node Y induce the subtree bipartition $y = (EA|C)$. This bipartition is represented as the integer tuple $(2, 1, 2, 3, 3)$, where $i = 2$, $l_x = 1$, $r_x = 2$, $l_y = 3$, and $r_y = 3$.

We now compute $\mathcal{M}(x, y)$, which involves determining the sizes of intersections between the corresponding sides of these two bipartitions (Equation 2 in the main text).

$$\mathcal{M}(x, y) = \mathcal{NT}(|A_1 \cap A_2|, |B_1 \cap B_2|) + \mathcal{NT}(|A_1 \cap B_2|, |A_2 \cap B_1|)$$

To facilitate efficient comparison between bipartitions originating from different trees, we first compute the index maps π_1 and π_2. These mappings record, for each taxon, its position in the respective post-order traversal arrays $\mathcal{A}_1$ and $\mathcal{A}_2$:

$$\mathcal{A}_1 = \{A, B, C, D, E\} \;\Rightarrow\; \pi_1(A) = 1,\ \pi_1(B) = 2,\ \pi_1(C) = 3,\ \pi_1(D) = 4,\ \pi_1(E) = 5,$$
$$\mathcal{A}_2 = \{E, A, C, D, B\} \;\Rightarrow\; \pi_2(A) = 2,\ \pi_2(B) = 5,\ \pi_2(C) = 3,\ \pi_2(D) = 4,\ \pi_2(E) = 1.$$

For the intersection $A_2 \cap B_1$, we iterate over the smaller cluster $A_2 = \{E, A\}$ and test membership in $B_1 = \{C, D, E\}$ using the index maps π_1 and π_2:

$$\text{Check whether } E \in B_1: \quad \pi_1(E) = 5 \text{ and } 3 \le 5 \le 5 \;\Rightarrow\; E \in B_1 \;\Rightarrow\; \mathbf{1}_{B_1}(E) = 1,$$
$$\text{Check whether } A \in B_1: \quad \pi_1(A) = 1 \text{ and } 1 < 3 \;\Rightarrow\; A \notin B_1 \;\Rightarrow\; \mathbf{1}_{B_1}(A) = 0.$$

Summing these indicators yields

$$|A_2 \cap B_1| = \mathbf{1}_{B_1}(E) + \mathbf{1}_{B_1}(A) = 1 + 0 = 1.$$

The remaining intersections are computed similarly by iterating over the smaller of the two clusters and summing membership indicators:

$$|A_1 \cap A_2| = |\{B\} \cap \{E, A\}| = 0,$$

$$|B_1 \cap B_2| = |\{C, D, E\} \cap \{C\}| = 1,$$

$$|A_1 \cap B_2| = |\{B\} \cap \{C\}| = 0.$$

These intersection counts are then used to compute $\mathcal{M}(x, y)$.

2 Computation of Double Hash Signatures

2.1 Specification of Single-element Hash Function Implementation

We implement the single-element hash function $\mathcal{H}$ using a standard 64-bit multiplicative mixing scheme. For an integer taxon identifier m, the function applies a sequence of XOR–shift and multiplication steps:

$$\begin{aligned}
&x \leftarrow m,\\
&x \leftarrow x \oplus (x \gg 16),\\
&x \leftarrow x \times C_1,\\
&x \leftarrow x \oplus (x \gg 13),\\
&x \leftarrow x \times C_2,\\
&x \leftarrow x \oplus (x \gg 16).
\end{aligned}$$

Here $C_1 =$ `0x85ebca6b` and $C_2 =$ `0xc2b2ae35` are fixed 32-bit multiplicative constants commonly used in high-quality non-cryptographic hash functions (e.g., MurmurHash3 [1]). To avoid degenerate cases in prefix-based computations, we map the output value 0 to 1. The resulting function provides good dispersion of contiguous integer identifiers and minimizes structural collisions.

2.2 Example: Calculation of Double Hash Signature of Bipartitions with Prefix Scan Arrays

We assume 1-based array indexing in this example. Consider the gene tree $g_i = ((D, B), (C, (A, E)))$ containing the set $\mathcal{X} = \{A, B, C, D, E\}$ of taxa. We consider a subtree-bipartition $x = (DB|CAE)$ in g_i, and illustrate how its double-hash signature is computed using prefix-scan arrays.

We use two hash functions ϕ_1 and ϕ_2, instantiated as addition and XOR, respectively. The taxa A–E are assigned integer identifiers 0–4 sequentially. For simplicity, let the single-element hash function be defined as

$$\mathcal{H}(m) = (101m + 7) \bmod 128.$$

(see the specification of the actual single-element hash function in Section 2.1). We consider a small modulus $2^7 = 128$ for convenience of illustration.

The post traversal array of g_i is

$$\mathcal{A}_i = \{D, B, C, A, E\}.$$

The bipartition $x = (DB|CAE)$ is therefore encoded as the integer tuple $(i, 1, 2, 3, 5)$. Prefix-sum and prefix-XOR arrays over $\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))$ are constructed as shown in Table S1.

We now compute the hash values for the two sides of the bipartition $x = (DB \mid CAE)$ by applying the corresponding cancellation operators— subtraction for ϕ_1 and XOR for ϕ_2—using the prefix-scan arrays.

Side $\{D, B\}$ of x. This corresponds to the range $\mathcal{A}_i[1:2]$. Using the prefix arrays,

$$\phi_1(\{D, B\}) = (34 - 0) \bmod 128 = 34, \qquad \phi_2(\{D, B\}) = (90 \oplus 0) \bmod 128 = 90.$$

One can verify that the sum of $\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))$ over the range $[1:2]$ is $(54+108) \bmod 128 = 34$. Likewise, the XOR over the same range is $(54 \oplus 108) \bmod 128 = 90$.

Thus, the signature of $\{D, B\}$ is $(34, 90)$.

Table S1: Example calculation of prefix scan arrays for two hash functions: addition and XOR on the traversal array $\mathcal{A}_i = \{D, B, C, A, E\}$. Integer identifiers 0–4 are sequentially assigned to the taxa A–E. The single element hash function $\mathcal{H}$ is assumed to be $\mathcal{H}(m) = (101m + 7) \bmod 128$. All calculations are shown modulo $2^7 = 128$.

j	$\mathcal{A}_{\mathbf{i,j}}$	$\mathrm{id}(\mathcal{A}_{\mathbf{i,j}})$	$\mathcal{H}(\mathrm{id}(\mathcal{A}_{\mathbf{i,j}}))$	**prefix sum over** $\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))$	**prefix XOR over** $\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))$
0				0	0
1	D	3	54	54	54
2	B	1	108	34	90
3	C	2	81	115	11
4	A	0	7	122	12
5	E	4	27	21	23

Side $\{C, A, E\}$ of x. This corresponds to the range $\mathcal{A}_i[3:5]$. From the prefix arrays,

$$\phi_1(\{C, A, E\}) = (21 - 34) \bmod 128 = 115, \qquad \phi_2(\{C, A, E\}) = (23 \oplus 90) \bmod 128 = 77.$$

One can verify that the sum of $\mathcal{H}(\mathrm{id}(\mathcal{A}_{i,j}))$ over the range $[3:5]$ is $(81 + 7 + 27) \bmod 128 = 115$. Likewise, the XOR over the same range is $(81 \oplus 7 \oplus 27) \bmod 128 = 77$.

Thus, the signature of $\{C, A, E\}$ is $(115, 77)$.

Final signature. Combining both sides, the double-hash signature of the bipartition $x = (DB|CAE)$ is

$$((34, 90), (115, 77)).$$

3 Theoretical Analysis of Collision Probability

Theorem 3.1 (Collision Probability Bound)**.** *Let $\sigma(S) = (\phi_1(\{r_t : t \in S\}), \phi_2(\{r_t : t \in S\}))$ be the double-hash signature of a set S of taxa, where ϕ_1, ϕ_2 are independent permutation-invariant hash functions modulo M and each $r_t = \mathcal{H}(\mathrm{id}(t))$ is independently and uniformly distributed in $\mathbb{Z}_M$. Among B total bipartitions, the probability that at least one collision will occur is bounded by*

$$\Pr(\exists \textit{ collision}) \leq \frac{B^2}{M^2}.$$

Proof. The key intuition is that collisions are rare because whenever two sets differ, at least one taxon contributes an independent random value, making accidental equality highly unlikely.

Let $\mathcal{X}$ be the taxon set with $|\mathcal{X}| = n$. Let B be the number of subtree-bipartitions produced across all gene trees (in our setting, $B = \Theta(nk)$). We choose a large modulus M and work in the ring $\mathbb{Z}_M = \{0, 1, 2, \ldots\ldots, M - 1\}$.

Setup. A single-element hash

$$\mathcal{H} : \{0, \ldots, n - 1\} \to \mathbb{Z}_M$$

maps each taxon identifier $\mathrm{id}(t)$ to

$$r_t \coloneqq \mathcal{H}(\mathrm{id}(t)) \in \mathbb{Z}_M.$$

We idealize that the values $\{r_t\}_{t \in S}$ are independent and uniformly distributed in $\mathbb{Z}_M$. We then apply two associative, permutation-invariant set-level functions ϕ_1, ϕ_2, each operating directly on subsets of $\{r_t\}$. The signature of a set S is

$$\sigma(S) \coloneqq \big(\phi_1(\{r_t : t \in S\}),\, \phi_2(\{r_t : t \in S\})\big).$$

Single-set collision probability. Let $S \neq T$ and define $U = S \setminus T$, $V = T \setminus S$. Associativity ensures that ϕ_a admits a corresponding cancellation operator $\ominus$ (e.g. subtraction for addition, XOR for XOR, etc.), so that

$$\phi_a(S) \ominus \phi_a(T) \;=\; \phi_a(U) \ominus \phi_a(V).$$

Because $U \cup V \neq \varnothing$, at least one r_t appears only in one side. Conditioning on all other r_t, the difference reduces to a function of the uniform random variable r_t. This equals zero with probability at most $1/M$. Thus

$$\Pr\big(\phi_a(S) = \phi_a(T)\big) = \frac{1}{M}. \tag{S1}$$

Pairwise signature collision. Suppose the two set-level functions ϕ_1 and ϕ_2 are constructed so that their collision events are statistically independent (e.g. by using independent salts or structurally distinct definitions of ϕ_a). Then for distinct sets S, T,

$$\Pr\big(\sigma(S) = \sigma(T)\big) = \Pr\big(\phi_1(S) = \phi_1(T)\big) \cdot \Pr\big(\phi_2(S) = \phi_2(T)\big) = \frac{1}{M^2}.$$

Collision probability for bipartitions. Let bipartitions $x = (A_1 \mid B_1)$ and $y = (A_2 \mid B_2)$ be distinct (non-equivalent). A collision occurs if and only if

$$\big(\sigma(A_1) = \sigma(A_2) \ \wedge \ \sigma(B_1) = \sigma(B_2)\big) \quad \text{or} \quad \big(\sigma(A_1) = \sigma(B_2) \ \wedge \ \sigma(B_1) = \sigma(A_2)\big).$$

Since each matching holds with probability $1/M^2$, we obtain

$$\Pr\big(\text{collision between } x \text{ and } y\big) \ \leq \ \frac{2}{M^2}.$$

Global (any-pair) collision probability. Among B bipartitions there are at most $\binom{B}{2}$ pairs. By a union bound,

$$\Pr(\exists \text{ collision}) \ \leq \ \binom{B}{2} \cdot \frac{2}{M^2} \ \leq \ \frac{B^2}{M^2}.$$

□

4 Theoretical analysis of the precomputation of subtree-bipartition weights

Theorem 4.1. *For the constrained version (i.e., $\mathcal{CB} = \mathcal{UGB}$), the DP state space of STELAR spans every bipartition $x \in \mathcal{CB}$.*

Proof. We argue by contradiction. Suppose there exists a bipartition $x = (A|B) \in \mathcal{CB}$ that is not explored in the state space of the DP. We consider two possible cases:

Case 1: $A \cup B = \mathcal{X}$ Since $\mathcal{CB} = \mathcal{UGB}$, we have $x \in \mathcal{GB}$. As $A \cup B$ covers all taxa, x corresponds to a root-level bipartition in one of the gene trees. By construction, such bipartitions are included in the search space at the very first state transition. This contradicts our assumption that x is not explored.

Case 2: $A \cup B \neq \mathcal{X}$ Again, $x \in \mathcal{GB}$ and hence arises from some internal node u of a gene tree g_i, where the taxa covered by u form the set $\mathcal{Y} = A \cup B$. Since $\mathcal{Y} \subsetneq \mathcal{X}$, u must have a parent node p in g_i. Let the bipartition induced by p be $y = (C|D)$. By construction, either $C = \mathcal{Y}$ or $D = \mathcal{Y}$. Thus, if y is explored in the DP state space, then the algorithm must also explore its child bipartition x, by design of the transition rules. Hence, the assumption that x is not explored implies that y is also not explored. Repeating this reasoning along the ancestor chain of u, we eventually reach the root of g_i, where Case 1 applies and yields a contradiction.

Since both cases lead to contradictions, our assumption is false. Therefore, every subtree-bipartition $x \in \mathcal{UGB}$ is necessarily explored in the constrained DP state space. □

Theorem 4.2. *For the constrained version of STELAR (i.e., $\mathcal{CB} = \mathcal{UGB}$), if each gene tree is a perfectly balanced rooted binary tree with $n = 2^d$ taxa, then the precomputation of weights requires $\mathcal{O}(n^2k^2)$ time.*

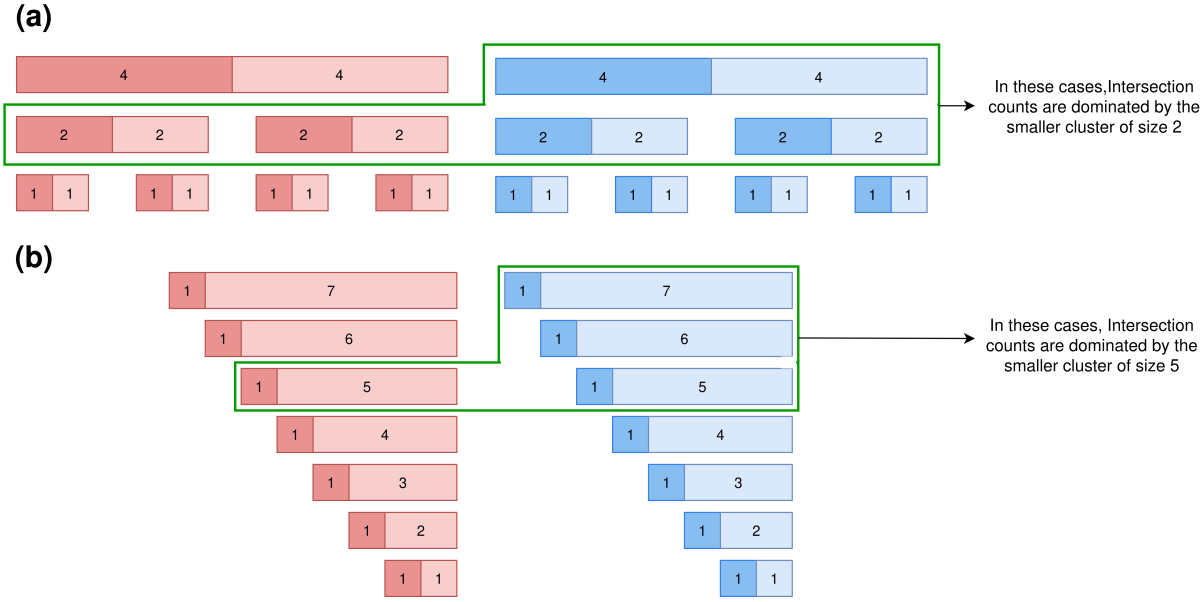

Figure S1: Running time analysis for computing intersection counts between two 8-taxon trees: (a) balanced binary trees and (b) caterpillar trees. Subtree bipartitions from each tree are shown, with numbers on partitions indicating the corresponding sizes.

Proof. We analyze the worst-case scenario in which every bipartition across all gene trees is distinct, i.e., $\mathcal{UGB} = \mathcal{GB}$. The complexity for all other cases then follows immediately as a direct relaxation of this setting.

By Equation 4 in the main manuscript ($w_{\mathcal{G}}(x) = \sum_{y \in \mathcal{UGB}} f_{\mathcal{G}}(y)\, \mathcal{M}(x, y)$), computing the weight of each candidate bipartition requires evaluating $\mathcal{M}(x, y)$ against every $y \in \mathcal{UGB}$. Thus we must evaluate $\mathcal{M}(x, y)$ for every ordered pair $(x, y) \in \mathcal{UGB} \times \mathcal{UGB}$. We bound the total cost by first bounding the cost between two fixed balanced trees and then summing over the tree-pairs.

In a perfectly balanced rooted binary tree on $n = 2^d$ leaves, an internal node at depth ℓ (root at depth 0) has subtree size $n/2^\ell$, and each of its two child subtrees has size $s = n/2^{\ell+1}$. Thus the set of possible child-subtree sizes is

$$\mathcal{S} = \{\, 2^0, 2^1, \ldots, 2^{d-1} \,\},$$

and for a given child-subtree size $s \in \mathcal{S}$ the number of internal nodes whose child-subtrees have size s equals the number of disjoint blocks of size $2s$ that partition the n leaves, i.e. $\frac{n}{2s}$. Therefore the number of subtree-bipartitions in one balanced tree whose parts (child-subtrees) have size s is $n/(2s)$.

Per-pair cost in terms of sizes. We consider two bipartitions $x = (A_1|B_1)$ in tree g_p and $y = (A_2|B_2)$ in tree g_q, and let the child-subtree sizes be s and t, respectively. So $|A_1| = |B_1| = s$ and $|A_2| = |B_2| = t$). By Equation 2 in the main manuscript, $\mathcal{M}(x, y)$ depends on four intersection counts among the parts of x and y. Each such intersection can be computed by iterating over the smaller argument using the index map π, hence the time to compute each intersection is $O(\min(s, t))$. Since there are only a constant number of such intersections, the total cost to evaluate $\mathcal{M}(x, y)$ is

$$\text{cost}(\mathcal{M}(x, y)) \;=\; O(\min(s, t)).$$

Cost between two balanced trees. We now group bipartitions of g_p and g_q by their child-subtree sizes $s = 2^i$ and $t = 2^j$ (with $0 \le i, j \le d - 1$). The number of bipartitions in g_p with part-size s is $n/(2s)$, and similarly for g_q with part-size t. Thus the number of ordered bipartition pairs with sizes (s, t) is $\frac{n}{2s} \cdot \frac{n}{2t}$, and the contribution of this size-class pair to the cost is

$$\frac{n}{2s} \cdot \frac{n}{2t} \cdot O(\min(s, t)) = O\left(\frac{n^2 \min(s, t)}{4st}\right) = O\left(\frac{n^2}{4 \max(s, t)}\right).$$

Replacing $s = 2^i$ and $t = 2^j$, the total cost between the two trees is

$$S_{pq} \;=\; O\left(n^2 \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} 2^{-\max(i,j)} \right).$$

An illustration of this analysis is shown in Figure S1a for a pair of balanced binary trees of 8 taxa.

We now evaluate the double sum by symmetry. Splitting the domain into the two triangles $i \le j$ and $i > j$:

$$\begin{aligned}
\sum_{i=0}^{d-1} \sum_{j=0}^{d-1} 2^{-\max(i,j)} &= \sum_{j=0}^{d-1} \sum_{i=0}^{j} 2^{-j} + \sum_{i=0}^{d-1} \sum_{j=0}^{i-1} 2^{-i} \\
&= \sum_{j=0}^{d-1} (j+1) 2^{-j} + \sum_{i=0}^{d-1} i\, 2^{-i} \\
&= 2 \sum_{k=0}^{d-1} \frac{k}{2^k} + \sum_{k=0}^{d-1} \frac{1}{2^k}.
\end{aligned}$$

Both finite sums are uniformly bounded as d grows, since $\sum_{k\ge 0} k/2^k = 2$ and $\sum_{k\ge 0} 1/2^k = 2$. Hence

$$\sum_{i=0}^{d-1} \sum_{j=0}^{d-1} 2^{-\max(i,j)} \le 6,$$

and in fact the exact value is

$$\sum_{i=0}^{d-1} \sum_{j=0}^{d-1} 2^{-\max(i,j)} = 6 - \frac{2d+3}{2^{d-1}}.$$

Therefore, the double sum is $O(1)$. Hence $S_{pq} = O(n^2)$. Thus, computing all pairwise intersection costs between bipartitions of two balanced trees costs $O(n^2)$.

Extension to k trees. There are k gene trees and thus k^2 ordered pairs of trees. Summing the $O(n^2)$ cost over all ordered pairs yields the total precomputation cost $O(n^2) \cdot k^2 = O(n^2 k^2)$, which completes the proof. □

Theorem 4.3. *Assume $\mathcal{CB} = \mathcal{UGB}$. If each gene tree is a rooted caterpillar on n taxa, then the precomputation of weights (evaluating $\mathcal{M}(x,y)$ for all ordered pairs $x, y \in \mathcal{UGB}$) requires $\mathcal{O}(n^3 k^2)$ time.*

Proof. By Equation 4 in the main manuscript, the precomputation requires computing $\mathcal{M}(x,y)$ for every ordered pair $(x,y) \in \mathcal{UGB} \times \mathcal{UGB}$. We first bound the cost between two fixed caterpillar trees and then sum over all ordered tree pairs.

Bipartitions in a rooted caterpillar. Fix a rooted caterpillar on n leaves arranged along the backbone. Its internal nodes (listed from the root downwards) induce the bipartitions of sizes

$$(1 \mid n-1),\ (1 \mid n-2),\ \ldots,\ (1 \mid 1),$$

. Equivalently, for each $L \in \{1, \ldots, n-1\}$ there is exactly one bipartition whose large side has size L (and the small side is size 1). Thus each caterpillar has exactly $n-1$ subtree-bipartitions and the multiset of large-side sizes is $\{1, 2, \ldots, n-1\}$ (each appearing once).

Per-pair computation of $\mathcal{M}(x,y)$. For two bipartitions $x = (A_1 \mid B_1)$ and $y = (A_2 \mid B_2)$ the equation 2 in main manuscript requires four intersection counts: $|A_1 \cap A_2|$, $|A_1 \cap B_2|$, $|B_1 \cap A_2|$, $|B_1 \cap B_2|$. In a rooted caterpillar at least three of these intersections involve a singleton side (size 1) and are therefore computable in $O(1)$ time. The remaining intersection is the one between the two large sides (say sizes s and t); this intersection can be computed in $O(\min(s,t))$ time by iterating over the smaller large-side and checking membership via the index map. Thus the dominant cost for evaluating $\mathcal{M}(x,y)$ is

$$\text{cost}\big(\mathcal{M}(x,y)\big) \;=\; O\big(\min(s,t)\big),$$

where s and t are the sizes of the large sides of x and y, respectively.

Cost between two caterpillar trees. Each of the two caterpillar trees has exactly one bipartition whose large-side size equals L for each $L \in \{1, \ldots, n-1\}$. Hence the total cost between two trees equals

$$S_{pq} \;=\; \sum_{s=1}^{n-1}\sum_{t=1}^{n-1} O\big(\min(s,t)\big) \;=\; O\left(\sum_{s=1}^{n-1}\sum_{t=1}^{n-1} \min(s,t)\right).$$

An illustration of this analysis is shown in Figure S1b for a pair of caterpillar trees of 8 taxa.

Now, the inner double sum is a standard combinatorial sum; counting by the minimum value $m = \min(s,t)$ gives

$$\sum_{s=1}^{n-1}\sum_{t=1}^{n-1} \min(s,t) = \sum_{m=1}^{n-1} m \cdot \#\{(s,t) : \min(s,t) = m\}.$$

For a fixed m, the number of ordered pairs (s,t) with $\min(s,t) = m$ equals $2(n-1-m)+1 = 2n-2m-1$. Therefore

$$\begin{aligned}\sum_{s=1}^{n-1}\sum_{t=1}^{n-1} \min(s,t) &= \sum_{m=1}^{n-1} m(2n-2m-1) = 2n\sum_{m=1}^{n-1} m - 2\sum_{m=1}^{n-1} m^2 - \sum_{m=1}^{n-1} m \\ &= \Theta(n^3).\end{aligned}$$

(Using the closed forms $\sum_{m=1}^{n-1} m = \Theta(n^2)$ and $\sum_{m=1}^{n-1} m^2 = \Theta(n^3)$ one obtains the cubic bound.) Hence $S_{pq} = O(n^3)$.

Extension to k trees. There are k gene trees and thus k^2 ordered tree-pairs. Summing the per-pair bound $O(n^3)$ over all ordered pairs yields the total precomputation cost

$$O(n^3) \cdot k^2 = O(n^3 k^2),$$

which completes the proof. □

5 Complexity analysis

We now provide a theoretical analysis of the running time and memory requirement of STELAR-X.

Theorem 5.1. *For n taxa, k gene trees, $T_{\mathcal{C}}$ CPU threads and $T_{\mathcal{G}}$ GPU threads, STELAR-X runs in $O\Big(\frac{nk}{T_{\mathcal{C}}} + \frac{n^2k^2}{T_{\mathcal{G}}} + |\mathcal{CB}|\Big)$ time (under the balance gene tree assumption), $O(nk + |\mathcal{UGB}| + |\mathcal{CB}|)$ CPU memory, and $O(|\mathcal{UGB}| + |\mathcal{CB}|)$ GPU memory. In the worst case, when all gene trees are caterpillars, the running time becomes $O\Big(\frac{nk}{T_{\mathcal{C}}} + \frac{n^3k^2}{T_{\mathcal{G}}} + |\mathcal{CB}|\Big)$.*

Proof. We first derive bounds for the running time of STELAR-X. The parsing of the k input trees over $T_{\mathcal{C}}$ CPU threads requires $O\Big(\frac{nk}{T_{\mathcal{C}}}\Big)$ time. The subsequent computation of the frequency mapping, using the double-hashing approach, also requires $O\Big(\frac{nk}{T_{\mathcal{C}}}\Big)$ time. This efficiency follows from the use of prefix-scan arrays, which enable constant-time range hash evaluations. Next, the precomputation of weights, distributed across $T_{\mathcal{G}}$ GPU threads, requires $O\Big(\frac{n^2k^2}{T_{\mathcal{G}}}\Big)$ time under the balanced gene tree assumption as derived in Theorem 4.2.

Finally, we analyze the running time of the dynamic programming (DP) algorithm. By Equation 1 in main manuscript and Theorem 4.1, the size of the DP state space is at most $O(|\mathcal{CB}|)$. Since both the state-space map and the weight map are precomputed, the DP algorithm itself executes in $O(|\mathcal{CB}|)$ time.

Putting all components together, the total running time of our algorithm under the assumptions that the trees are balanced and all the bipartitions are unique is,

$$O\Big(\frac{nk}{T_{\mathcal{C}}} + \frac{n^2k^2}{T_{\mathcal{G}}} + |\mathcal{CB}|\Big).$$

In the worst case, when all the trees are caterpillars, as derived in Theorem 4.3, the precompution complexity becomes $O\left(\frac{n^3k^2}{\mathcal{T}_\mathcal{G}}\right)$, and hence the total running time becomes, $O\left(\frac{nk}{T_\mathcal{C}} + \frac{n^3k^2}{T_\mathcal{G}} + |\mathcal{CB}|\right)$.

We now analyze the memory requirements of our algorithm in both CPU RAM and GPU VRAM.

On the CPU side, storing the k input gene trees over n taxa requires $O(nk)$ memory. The efficient representation of subtree-bipartitions, together with their subsequent processing, also requires $O(nk)$ memory. Storing the unique subtree bipartitions in $\mathcal{UGB}$ and $\mathcal{CB}$ requires $O(|\mathcal{UGB}| + |\mathcal{CB}|)$ space. The dynamic programming algorithm requires $O(|\mathcal{CB}|)$ additional space. Overall, the total CPU memory usage is therefore

$$O(nk + |\mathcal{UGB}| + |\mathcal{CB}|)\,.$$

On the GPU side, memory usage is dominated by the precomputation of weights. Each bipartition is represented as a constant-length integer tuple. Consequently, the total GPU VRAM required for storing all $O(|\mathcal{UGB}| + |\mathcal{CB}|)$ bipartitions is

$$O(|\mathcal{UGB}| + |\mathcal{CB}|)\,.$$

Since both $|\mathcal{UGB}|$ and $|\mathcal{CB}|$ can be bounded by $O(nk)$ in the constrained case, the overall CPU and GPU memory consumption scales as $O(nk)$, matching the asymptotic input-space complexity.

□

6 Dataset

The average gene tree-gene tree (GT-GT) discordance and gene tree-species tree (GT-ST) discordance (in terms of RF rates) corresponding to four different levels of ILS are shown in Table S2. Our simulation parameters corresponding to ILS-L2 are presented in Table S3. For generating these four different ILS levels, we vary the maximum value of the uniform distribution of Effective Population Size from 150000 to 300000.

Table S2: Discordance levels under different ILS conditions analyzed in this study. We show the average gene tree-gene tree (GT-GT) and gene tree-species tree (ST) discordance, measured by average RF rates, under various levels of ILS.

ILS Level	**GT–GT Discordance (%)**	**GT–ST Discordance (%)**
ILS-L1	25.4	17.5
ILS-L2	31.7	22.1
ILS-L3	37.5	26.4
ILS-L4	42.8	30.7

Table S3: Simulation parameters used to generate gene trees and species trees using Simphy.

Parameter Name	**Parameter Value**
Speciation rate	0.000001
Extinction rate	0
Duplication rate	0
Horizontal Gene Transfer rate	0
Ingroup divergence to the ingroup ratio	1.0
Generations	LogN(1.470055e+01,2.500000e-01)
Haploid effective population size	Uniform(100000,200000)
Global substitution rate	LogN(-1.727461e+01,6.931472e-01)
Species Tree Height	LogN(16.2,1)
Seed	42

7 Additional Results

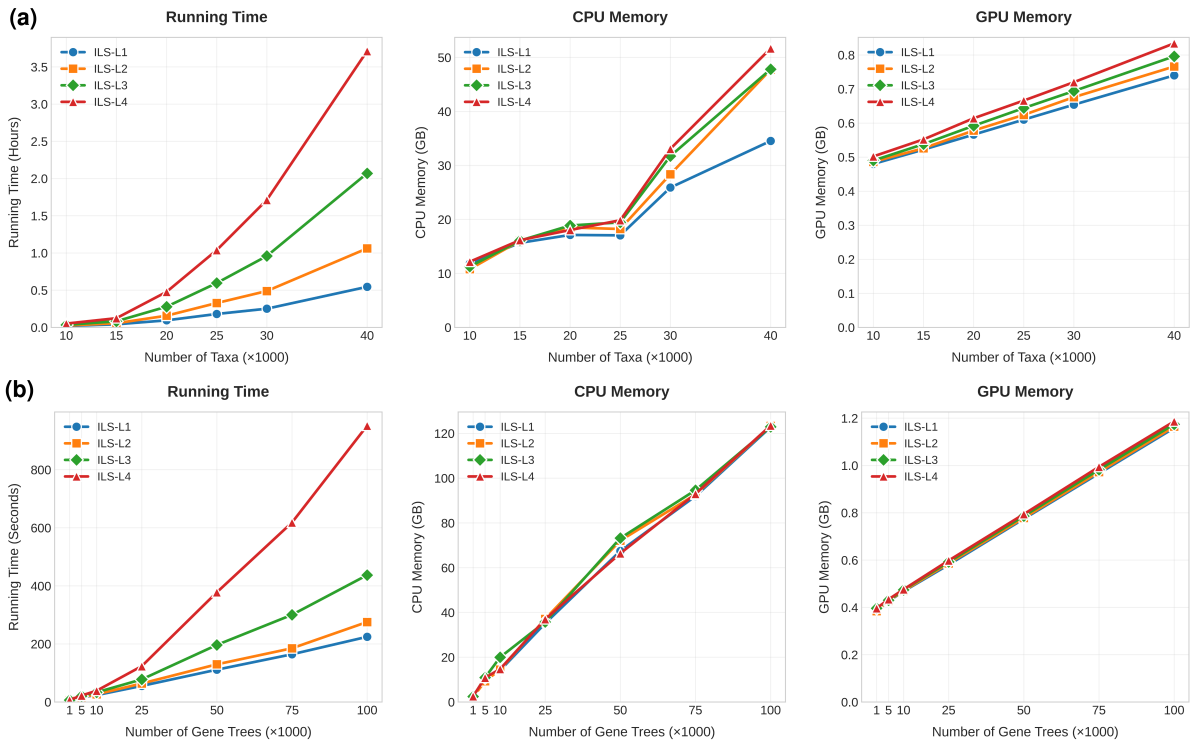

Figure S2: Effect of varying levels of ILS on the running time and memory usage of STELAR-X across 4 model conditions, using simulated datasets with (a) 10,000–40,000 taxa and 1,000 gene trees (b) 1,000 taxa and 1,000-100,000 gene trees.

Table S4: Comparison of running time and memory usage between STELAR-X (SX) and ASTRAL-MP (AM) across three biological datasets

Dataset	#Taxa	#Gene Trees	Running Time (s)		Max CPU (MB)		Max GPU (MB)	
			SX	AM	SX	AM	SX	AM
Avian-48	48	14446	12.043	357.537	2959.773	3156.684	134.144	176.248
Avian-363	363	14972	1063.009	9601.098	20396.316	34318.273	549.888	556.032

Figure S3: Species tree estimated by STELAR-X on the extended avian dataset with 363 taxa and 14,972 genes. Although the gene trees were highly incomplete, we did not augment the set of subtree bipartitions in the gene trees by adding extra subtree-bipartitions, as our primary objective here is to demonstrate scalability.

References

[1] Austin Appleby. Murmurhash3. https://github.com/aappleby/smhasher/wiki/MurmurHash3. Accessed: 2025-11-22.