Rabiee and Mirarab *BMC Genomics* 2020, **21**(Suppl 2):218
https://doi.org/10.1186/s12864-020-6607-z

BMC Genomics

RESEARCH Open Access

Forcing external constraints on tree inference using ASTRAL

Maryam Rabiee[1] and Siavash Mirarab[2*]

From 17th RECOMB Satellite Conference on Comparative Genomics
Montpellier, France. 1–4 October 2019

Abstract

Background: To account for genome-wide discordance among gene trees, several widely-used methods seek to find a species tree with the minimum distance to input gene trees. To efficiently explore the large space of species trees, some of these methods, including ASTRAL, use dynamic programming (DP). The DP paradigm can restrict the search space, and thus, ASTRAL and similar methods use heuristic methods to define a restricted search space. However, arbitrary constraints provided by the user on the output tree cannot be trivially incorporated into such restrictions. The ability to infer trees that honor user-defined constraints is needed for many phylogenetic analyses, but no solution currently exists for constraining the output of ASTRAL.

Results: We introduce methods that enable the ASTRAL dynamic programming to infer constrained trees in an effective and scalable manner. To do so, we adopt a recently developed tree completion algorithm and extend it to allow multifurcating input and output trees. In simulation studies, we show that the approach for honoring constraints is both effective and fast. On real data, we show that constrained searches can help interrogate branches not recovered in the optimal ASTRAL tree to reveal support for alternative hypotheses.

Conclusions: The new algorithm is added ASTRAL to all user-provided constraints on the species tree.

Keywords: ASTRAL, Constrained tree search, Tree completion, RF distance

Background

Phylogeny inference requires solving an optimization problem over the space of all trees. The super-exponential growth of the tree topology space makes examining all trees impossible, even for moderately large datasets. As a result, tree inference algorithms have adopted several heuristics strategies, including iterative search (e.g., hill-climbing), used by most maximum parsimony and maximum likelihood methods.

An increasingly popular alternative is searching the tree space using dynamic programming (DP). For an optimization score of interest, we need a recursion formulating how the optimal tree on a subset of leaves (or similar constructs) can be computed from the optimal trees on smaller subsets. With a recursive equation, DP can be used to compute the optimal solution in the classic fashion, typically implemented using memoization. Since the powerset grows exponentially with the set cardinality, this DP requires exponential running time. However, a restricted version of DP can be designed where each set is divided into only some of its subsets; the restricted DP can have polynomial running time with respect to the number of the leaves.

Phylogenetic inference using this particular DP approach has been known at least as early as 1996 [1] and has been used for many optimization criteria, including duplication and loss [2–4], deep coalescence [5], Robinson

*Correspondence: smirarab@ucsd.edu
[2]Department of Electrical and Computer Engineering, UC San Diego, 9500 Gilman Dr, 92093 La Jolla, USA
Full list of author information is available at the end of the article

Foulds (RF) distance [6], quartet score [7, 8], and others [9]. Among these, ASTRAL [8], which estimates a species tree from a set of gene trees by minimizing the quartet distance, has found increasing popularity [10]. DP is mostly used for problems where the input is a set of trees, and the output is a tree with the minimum total distance to the input trees. The popularity of DP for these problems is because the input set of trees create natural ways for restricting the space explored by DP. For example, the set of bipartitions observed in input trees can be used as the restriction set. In addition, ASTRAL has introduced heuristics to enrich the set of allowable bipartitions [11] while keeping the size of the search space polynomial [12].

The restrictions imposed on DP are not to be confused with the related concept of user-imposed constrained inference (we use "restricted" DP instead of "constrained" used in previous publications to avoid confusion). Systematists often would like to infer the best possible tree among trees that are compatible with a *constraint tree* of their choice, thereby completing and resolving the constraint tree. Constrained tree inference is needed for hypothesis-driven analyses that aim to choose the best among a set of hypotheses available by prior knowledge [13–16]. Constrained searches can help in model selection, testing whether a polytomy [17] or the monophyly of a group [18] can be rejected. Similarly, they can help gauge the "hidden" support for branches not recovered in the main analysis. Moreover, constrained searches have been successfully used to combine the results of multiple methods [19]. More recently, constrained trees were used in taxonomic profiling [20]. Finally, constrained searches enable updating existing trees without recomputing trees from scratch. For these reasons, most phylogenetic inference tools allow user-provided constraints.

To our knowledge, the DP paradigm has not been adopted to perform tree search with user-defined constraints. Performing constrained searches in the DP paradigm may appear easy: one needs to make sure the restricted set of bipartitions explored by DP are consistent with the constraints. As we show, there are roadblocks when the user-provided input is allowed to be arbitrary. The challenge is to find a large-enough search space that satisfies the user-provided constraints. Here, building on two recent advances [21, 22], we propose a solution to this challenge. We implement our solution inside the ASTRAL software for species tree inference, thereby enabling it to perform constrained searches for the first time. In extensive tests, we show that the constrained searches remain as accurate as unconstrained searches while reducing the running time, can improve accuracy in the presence of external knowledge about individual relationships, and can reveal hidden support.

Method

Our goal is to extend ASTRAL so that it can honor a user-provided constraint tree. We start by reviewing ASTRAL and the RF(+) algorithm that we will use.

Notations We are given a set of k potentially multifurcating input trees $\mathcal{T}$ on subsets of a leafset $\mathcal{L}$ of size n and a potentially multifurcating constraint tree, $\bar{T}$ on a subset of $\mathcal{L}$. Let $l(t)$ or $l(u)$ be the set of leaves of a tree t or leaves below a node u. We use $s(u)$ to denote sister(s) of u; i.e., the set of all nodes sharing a parent with u. Let $\mathcal{L}' = \mathcal{L} \setminus \{o\}$ where $o \in \mathcal{L}$ is a fixed arbitrarily chosen species. Denote $A \subset \mathcal{L}'$ as a cluster. Each edge in an unrooted tree corresponds to a bipartition of leaves, which corresponds to a cluster (the side missing o). A cluster A (i.e. the bipartition $A|\mathcal{L} \setminus A$) is called compatible with a tree T iff a tree exists that includes the bipartition and induces a resolution of T when restricted to same leaves as T. Two clusters are compatible iff they can be in the same tree [23].

Background: tree completion

Completing a tree based on a reference tree is a well-studied problem and is often formulated as minimizing the distance to the reference tree while maintaining compatibility with the original tree [4, 21, 22, 24]. A natural objective is to find a complete tree with the minimum RF distance (i.e., the total number of bipartitions that differ between the two trees) to the reference tree [25]. OCTAL was the first quadratic time solution to this RF completion problem [21]. Bansal later introduced a linear time solution [22], which we call B-RF(+) algorithm. Both methods take as input an incomplete backbone tree, T_b, and a complete and binary reference tree, T_r, and output a binary and complete tree compatible with T_b such that the RF distance to T_r is minimized among all allowable trees.

The B-RF(+) algorithm [22] achieves linear time using constant time least-common-ancestor (LCA) lookups made possible after a linear time preprocessing using the Schieber–Vishkin technique [26]. Both trees are rooted on an arbitrary shared leaf. For every fully-missing node u of T_r (i.e., $\forall u : l(u) \cap l(T_b) = \emptyset$), the subtree below u is added intact as the sister to the LCA of $l(s(u))$ inside T_b (Fig. 1). This placement of the subtree below u preserves all its bipartitions, the bipartition above it, and potentially the bipartition above the parent of u (we will come back to this point). The order of additions to T_b is determined by a pre-order traversal of T_r, adding each fully-missing node u when we visit the parent node of u. Note that the topology of the backbone tree will not change by the addition of new subtrees. Bansal proved that this algorithm minimizes the RF distances between T_r and any possible *binary* tree that is compatible with T_b.

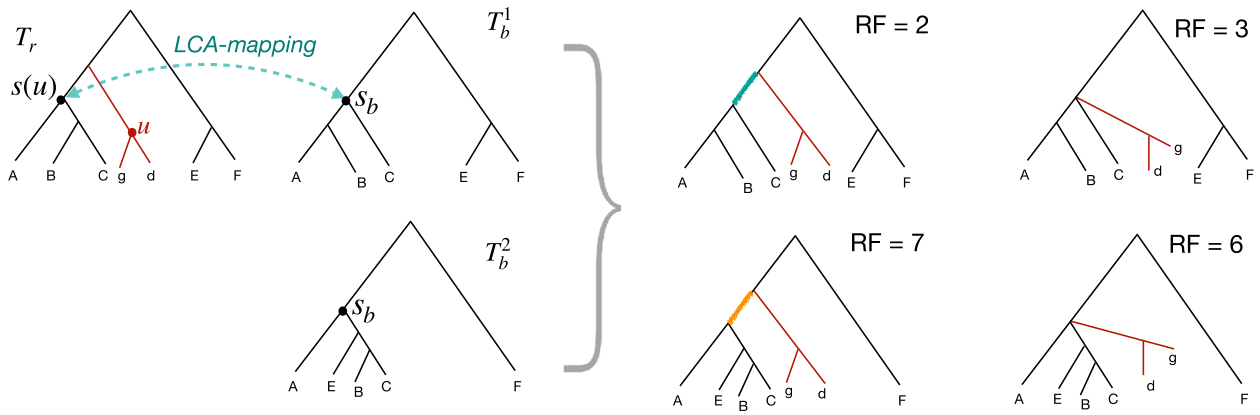

Fig. 1 Updates need for the tree completion algorithm. T_b^1 and T_b^2 are both completed based on T_r, generating either a binary tree or a multifurcating tree. In case 1 (T_b^1), subtree under u should be added as sister to s_b to minimize RF; the highlighted green branch matches T_r, but creating a polytomy would result in a FN. In case 2 (T_b^2), the subtree under u should be added as a polytomy under s_b; otherwise, the highlighted orange branch will be a FP. In the second case, restricting the output to be binary (as in the B-RF(+) algorithm) leads to suboptimal RF distances

Background: DP algorithm implemented in ASTRAL

ASTRAL estimates an unrooted (species) tree given a set of unrooted (gene) trees $\mathcal{T}$ and is statistically consistent under the multi-species coalescent model [27] of incomplete lineage sorting (ILS) given a sample of true gene trees. ASTRAL seeks the tree T with the maximum quartet score to $\mathcal{T}$ defined as $\sum_{t\in\mathcal{T}} |Q(T) \cap Q(t)|$, where $Q(.)$ is the set of quartet topologies of a tree. Let $S(A)$ be the score for an optimal subtree on the cluster A. Defining $S(\{x\}) = 0$ for $x \in \mathcal{L}$, the recursion is:

$$S(A) = \max_{A'\in X, A\setminus A'\in X} S(A') + S(A\setminus A') + w_{\mathcal{T}}(A'|A\setminus A'|\mathcal{L}\setminus A) \quad (1)$$

where X is a set of clusters and $w_{\mathcal{T}}$ is a function assigning weights to tripartitions of $\mathcal{L}$ such that the sum of all weights for any tree gives its quartet score. If X is set to $2^{\mathcal{L}'}$, the recursion tests all ways of dividing A into two smaller clusters and under this condition, $S(\mathcal{L}')$ (Eq. 1) gives the optimal quartet score [8] for $\mathcal{T}$ in time growing exponentially with n (expected, as the problem is NP-Hard [28]).

Forming set X: heuristics and restrictions

To handle large datasets, we need X to have a manageable size, preferably growing polynomially with n and k. At the same time, we ideally want X to have all clusters of the optimal tree. An obvious way of building X is to set it to all clusters in all trees in $\mathcal{T}$, hoping that all clusters in the optimal tree appear in at least one input. However, two difficulties emerge. Firstly, simulations under very high levels of gene tree discordance have shown this heuristic to be insufficient as bipartitions in the optimal tree can frequently be absent from gene trees [11]. To deal with this issue, starting from ASTRAL-II, set X is enhanced using a set of heuristic methods, and since ASTRAL-III, the size of X is restricted to grow linearly with n and k [12, 29]. These heuristics (among other techniques) build consensus trees from input trees and add resolutions of polytomies of consensus trees to X.

The second difficulty is having full resolutions. Equation 1 is well-defined only if for every non-singleton $A \in X$, there is $A' \subset A$ such that $A' \in X$ and $A \setminus A' \in X$. More generally, a cluster A in X is useful *only if* there exists a fully binary tree on $\mathcal{L}'$ that includes A and all of its clusters are in X. Including any other cluster in X is a waste of computation. Thus, set X (which needs to be non-empty) needs to satisfy this property (recall $o \notin A \subset \mathcal{L}'$ and $2n-3$ is the number of clusters in a fully resolved tree):

P1: $\forall A_1 \in X, \exists\{A_1, A_2, \ldots, A_{2n-3}\} \subset X$ s.t.$\forall(i,j)$: A_i is compatible with A_j.

Building X using bipartitions of input trees $\mathcal{T}$ can fail to satisfy this property unless all trees are complete and binary. Thus, starting from ASTRAL-II, three steps are taken. *i*) Before adding bipartitions from $\mathcal{T}$ to X, it first completes each tree with respect to other trees using a distance matrix computed from quartet frequencies in $\mathcal{T}$ and an algorithm based on the four-point condition [30]. *ii*) Polytomies in input trees are resolved once [11] or more [12] using heuristic methods that sample leaves around polytomies and use the distance matrix mentioned earlier. *iii*) Heuristic enhancements of set X employed in

ASTRAL-II and ASTRAL-III are all explicitly designed such that P1 is automatically satisfied, a feat that has been particularly challenging for multi-individual datasets [29]. Thus, in effect, the set X includes all clusters from each tree in a set of binary and complete trees; the set includes modified input trees and others that ASTRAL heuristically selects.

Enabling input constraints in ASTRAL

Given a constraint tree $\bar{T}$ and a set of gene trees, $\mathcal{T}$, our goal is to find the tree among all trees compatible with $\bar{T}$ that has the maximum quartet score with respect to $\mathcal{T}$. Compatibility with $\bar{T}$ is achieved if we enforce a second property on X.

P2: $\forall A \in X : A$ is compatible with $\bar{T}$.

Existing methods for forming X are not guaranteed to satisfy P2. One may think that we can follow standard methods of forming X and simply refuse to add clusters when they violate P2. Unfortunately, that approach, in addition to being slow, can violate property P1 and is not viable. Thus, the main challenge in building set X is maintaining P1 and P2 simultaneously and doing so in a scalable fashion.

Forming set X using tree completion

Our solution relies on completing and resolving the tree $\bar{T}$ using every tree in X. We require a tree completion method $Comp(T_b, T_r)$ that adds to T_b leaves that are present in T_r but are absent from T_b. The algorithm should only add missing leaves to T_b and can also resolve (some of) its polytomies. In other words, the output restricted to leaves of T_b is a resolution of T_r; thus, $Comp(\bar{T}, t)$ will be compatible with $\bar{T}$.

Tree completion/resolution methods were traditionally proposed for completing a gene tree using the species tree [4, 21, 22]. However, in our algorithm, we turn the problem on its head and complete the constraint species tree $\bar{T}$ using individual gene trees and use these mixed trees to build the set X. This uncommon use of the completion method is the main algorithmic idea that enables us to satisfy P1 and P2.

Given a $Comp(T_b, T_r)$ method, we propose Algorithm 1 for forming set X. The first step is the primary new step and forces gene trees to be compatible with $\bar{T}$ (by definition of $Comp(T_b, T_r)$). Step 2 is identical to ASTRAL-III, where, compatible gene trees are completed in a reference-free fashion with respect to each other; any completion method such as the distance-based method used by ASTRAL [30] is valid here. Step 3, like ASTRAL-III, creates a set of multifurcating consensus trees $\mathcal{C}$ from the compatible gene trees $\bar{\mathcal{T}}'$. Then, in Step 4, like Step 2, we force consensus trees to be compatible with $\bar{T}$. Thus, all trees in $\bar{\mathcal{T}}'$ and $\bar{\mathcal{C}}$ are compatible with $\bar{T}$ and any resolution of these trees is also compatible with $\bar{T}$. Step 5 uses sampling methods of ASTRAL-III to resolve these trees heuristically and adds their bipartitions to X. Thus,

Algorithm 1 FormX($\mathcal{T}, \bar{T}$) algorithm for computing set X from input gene trees $\mathcal{T}$ such that every cluster in X is compatible with $\bar{T}$.

1. $\mathcal{T}' = \{$Completed t with reference to $\mathcal{T} \mid t \in \mathcal{T}\}$
   (any completion method is valid)
2. $\bar{\mathcal{T}}' = \{Comp(\bar{T}, t) | t \in \mathcal{T}'\}$
3. $\mathcal{C} = \{$consensus trees computed from completed trees $\bar{\mathcal{T}}'\}$
   (these trees should only include bipartitions found in $\bar{\mathcal{T}}'$)
4. $\bar{\mathcal{C}} = \{Comp(\bar{T}, t) | t \in \mathcal{C}\}$
5. $\mathcal{T}^{\mathcal{X}} = \bigcup_{\bar{t} \in (\bar{\mathcal{T}}' \cup \bar{\mathcal{C}})} \{$ one or more binary resolutions of $\bar{t}$ $\}$
   (any method of resolving polytomies is valid)
6. Root every trees of $\mathcal{T}^{\mathcal{X}}$ at o and add all of its clusters to X.

Claim 1 *All bipartitions of X created by Algorithm 1 are part of a fully binary and complete tree (P1) and are compatible with the constraint tree $\bar{T}$ (P2).*

After forming X (Algorithm 1), DP proceeds as before, computing $w_{\mathcal{T}}$ using the original gene trees $\mathcal{T}$. Since all bipartitions in X are compatible with $\bar{T}$, any tree formed by DP will be compatible with $\circ\bar{T}$; thus, no other changes are needed.

Tree completion with non-binary input/output

We now describe our choice for the $Comp(T_b, T_r)$ method. We base our solution on the B-RF(+) algorithm [22] described earlier, which we re-implemented inside ASTRAL. However, several changes to the algorithm were necessary.

Multifurcating output. The B-RF(+) algorithm and OCTAL force the output to be binary. As a result, the output can include arbitrary branches that increase false positive (FP) edges (branches in the output missing from T_r) without reducing false negative (FN) edges (branches in T_r missing from the output) (Fig. 1). Thus, if the output tree is allowed to be multifurcating, neither algorithm is optimal (shown by a counter-example; Fig. 1). As mentioned earlier, ASTRAL has several heuristics to resolve polytomies in the input trees (Step 5 of Algorithm 1), and these heuristics are preferable to an arbitrary resolution. Thus, we changed the B-RF(+) algorithm so that $Comp(T_b, T_r)$ avoids adding arbitrary resolutions in

Steps 2 and 4, leaving resolving polytomies to heuristics of Step 5. Our experiments show that this change substantially reduced the RF of completed trees (Fig. S1).

Recall that the B-RF(+) algorithm adds each fully-missing node u in T_r as sister to the LCA of $l(s(u))$ in T_b; denote this LCA node as s_b. Let $\mathcal{L}^m = l(T_r) \setminus l(T_b)$ be the set of leaves missing from T_b and let $\mathcal{L}^s = l(T_r) \cap l(T_b)$ be shared leaves. Two cases arise.

Case 1. $l(s(u)) \setminus \mathcal{L}^m = l(s_b)$. In this case, the optimal placement of the subtree below u in T_b is as sister to s_b, creating a new node above s_b. The reason is that this placement leads to the bipartition identified by $s(u)$ to be identical between T_r and the completed tree, thereby avoiding a FN edge.

Case 2. $l(s(u)) \setminus \mathcal{L}^m \neq l(s_b)$. Here, no placement of the subtree below u onto T_b can avoid the FN penalty associated with missing the bipartition associated with $s(u)$ in T_b. However, placing the subtree as sister to s_b by creating a new internal node does lead to an unnecessary FP edge in the completed tree, separating $l(s_b)$ from other leaves (Fig. 1). To avoid these FP edges, we can simply create a polytomy in the completed tree by putting the new subtree as another child of s_b.

Our new algorithm (called B-RF(*) here) is a straightforward change of the B-RF(+) algorithm. We compute the LCA mapping both ways. When inserting the u subtree into T_b at s_b, we check if the LCA of s_b in T_r matches $s(u)$. If it does, we create a new internal node above s_b; otherwise, we create a polytomy in T_b by adding the subtree as a child of s_b. Let T^* be the output of our updated algorithm and T^+ be the output of the original B-RF(+) algorithm. By construction, every branch in T^* is also present in T^+, and thus, T^* is a contraction of T^+.

Claim 2 *The output of B-RF(*) has the minimum RF distance to the binary tree T_r among all (potentially multifurcating) trees compatible with T_b.*

We prove Claim 2 by showing that T^* has the minimum possible number of FN and FP branches; the optimality of RF follows. We rely on the result that T^+ is optimal among all binary trees [22]. By optimality of T^+, it has the minimum possible FN that any tree (binary or multifurcating) can achieve (a multifurcating tree cannot have a lower FN than T^+ because binary resolution of that tree would also have a lower FN than T^+). Also, T^* is a contraction of T^+, where, a branch is contracted *only if* it is a FP branch due to Case 2. Thus, the number of FN branches in T^* equals those of T^+ and hence is the minimum possible.

Now, let the number of FP branches in the T_b compared to $T_r{\restriction}_{\mathcal{L}^s}$ be f. Since adding $\mathcal{L}^m$ to T_b cannot reduce its FP branches, f is a lower bound on the number of FP branches in the optimal tree. We claim that the number of FP branches in T^* is also f. Starting with T_b, every addition of a subtree u to the current tree (T_i) keeps the number of FP branches fixed. To see this, first, consider a branch b of T_i that matches T_r (restricted to common leaves) before u is added and assume u is not placed on b; then, b cannot become a FP because u is placed on the correct side of b by the proof of the B-RF(+) algorithm. Next, consider the branch b where u is placed. If b matches T_r before the addition, u is placed here only if $l(s(u))$ matches leaves below b, as in Case 1, where two true positive branches are created. If b does not matches T_r before the addition, Case 2 ensures that we create a polytomy and avoid adding a new FP branch. Thus, every step keeps the number of FP branches fixed.

Multifurcating backbone. We defined B-RF(*) for binary T_b, but we can have multifurcating T_b (here, $\bar{T}$). To adopt B-RF(*) to multifurcating backbones, prior to completion, we need to add to T_b those bipartitions in T_r that are compatible with T_b (or else we will have unnecessary FN branches). This can be done using the same LCA mapping of the B-RF(*) algorithm. In a post-order traversal of T_r, for every node u that maps to a polytomy v in T_b, we check whether all children of u have a LCA mapping to a child of v. If they do, we create a new node below v and move mapped children under v to be children of the new node. It is easy to see that this method adds all missing bipartitions from T_r to T_b.

Multifurcating reference. Changing B-RF(*) to handle multifurcating T_r is straightforward. In the pre-order traversal, for any polytomy node u encountered in T_r, when there are multiple fully-missing nodes under u, we add them as a group to the same position (as a polytomy) in T_b. Other cases are naturally handled by the LCA mapping used by the B-RF(*) algorithm (note that we defined "sister" as the *set* of nodes sharing a parent with a node).

Results: simulations

We first test constrained ASTRAL on an existing [11] simulated dataset with 201 species. This SimPhy [31] dataset has three model conditions with moderate, high, or very high levels of ILS, controlled by setting the maximum species tree height to 10^7, 2×10^6, or 5×10^5 generations; mean RF distance (normalized by total number of branches in both trees) between the true species tree and true gene trees are 15%, 34%, and 69%, respectively. We use gene trees inferred using FastTree-II [32] from sequence data in our analysis. The estimated gene trees have relatively high levels of gene tree error (the average RF distance between estimated and true gene trees are 25%, 31%, and 47% for the three model conditions). Following previous publications [11], three replicates of high

ILS dataset are removed due to a extreme lack of gene tree resolution.

We compare both constrained and unconstrained ASTRAL to the true tree. We measure the topological error using the normalized RF distance, and also report the change in quartet score and running time between constrained and unconstrained ASTRAL. Note that quartet score of the constrained tree can be higher than unconstrained ASTRAL, as the default version of ASTRAL has a heuristic definition of X and is not guaranteed to find the optimal solution. We ask whether our method of forming the constrained and restricted set X is effective in providing the same level of accuracy as unconstrained (but restricted) searches while improving the running time. We then ask if the use of constraints can benefit accuracy.

Is set *X* restricted to a constraint tree sufficiently large?

Constraint tree $\bar{T}$ with missing leaves

We built constraint trees that include $\frac{1}{4}$, $\frac{1}{2}$, or $\frac{3}{4}$ of species by taking the ASTRAL-III tree on the full dataset and pruning leaves uniformly at random. Since the unconstrained tree induces the constraint tree, the relative accuracy of constrained and unconstrained search is entirely a function of the completeness of X.

The accuracy of the constrained ASTRAL in most condition matches that of the unconstrained ASTRAL (Fig. 2). For moderate (10^7) and high (2×10^6) levels of ILS, the drop in average accuracy for different numbers of genes never exceeds 0.4%. Only with very high ILS (5×10^5) and constraint trees with 50 species do we start to see small but noticeable drops in the accuracy of constrained ASTRAL. For example, with 50 genes and very high ILS, the error goes up from 20% with no constraints to 26% with a $\bar{T}$ that has 50 leaves. Consistent with this, the quartet score of the ASTRAL tree also remains largely unchanged in most cases, except, again, for the case of very high ILS and backbone trees that include only 50 leaves (Fig. S2).

Constrained searches also impact the running time and the search space X (Fig. 3). With very high ILS and $\bar{T}$ including 150 of leaves (pruning 50), we get from 4 to 8x improvement in running times and substantial reduction in the size of the search space. This reduction explains the small reduction in the accuracy of ASTRAL-III with constrained searches under these conditions. As backbone size becomes smaller, the running time converges to unconstrained ASTRAL; however, even when $\bar{T}$ includes a quarter of the leaves, we still have 1.2 to 3x improvement in the running time. With moderate and high levels of ILS, improvements are less pronounced but still substantial. Running time improvements mirror the change in the search space size, which is dramatically reduced (Fig. 3).

We also study a scenario where missing leaves in the constraint tree form clades instead of being uniformly distributed. Results of the clade-based removal did not substantially differ from the random removal (Fig. S3). With moderate or high ILS, random and clade-based removal were indistinguishable, and for very high ILS, only small differences were observed.

To summarize, our method of forming X for constraints with missing leaves retains accuracy and reduces running time, with only small reductions in the accuracy in the most extreme conditions, namely very high ILS and small constraint trees.

Constraint tree $\bar{T}$ with multifurcation

We next collapse randomly chosen branches from the unconstrained ASTRAL-III tree to create a complete but unresolved constraint tree. With these multifurcating constraint trees, constrained ASTRAL search is as accurate as the unconstrained ASTRAL even for very high levels of ILS (Fig. 4). Differences in mean accuracy are no more than 0.1% in any of the 27 conditions we tested. Remarkably, in the case of very high ILS, we even see a small but noticeable improvement in quartet score (but not accuracy) when the constraint tree includes only 50 branches (Fig. S4). Once again, the running time and the size of the search space both reduce dramatically in the constrained searches (Figure S5). Thus, our method of forming X is effective in the face of multifurcating constraint trees.

Can a constrained search help accuracy?

Constrained searches have the power to improve accuracy if prior knowledge of parts of the tree is available. To test this proposition in simulations, we study the accuracy of constrained ASTRAL when the constraint tree $\bar{T}$ is a subset or a contraction of the *true* species tree. In both cases (Figs. 2 and 4), the accuracy of the ASTRAL tree improves, and changes are dramatic when the constraint tree is missing only 50 leaves or branches. The improvements are especially strong for the case of complete but multifurcating true species trees (Fig. 4) where a constraint tree with only 50/198 branches can reduce the error from 19% to 13% with 50 genes with very high ILS. If $\bar{T}$ includes 150/198 branches, the error reduces down to 4%.

The dramatic improvements in accuracy are perhaps not surprising given the fact that parts of the tree are fixed to match the true tree. More interesting is whether adding constraints to some part of the tree improves the accuracy of the *remaining* parts of the tree. We thus evaluated the accuracy of trees only restricted to the leaves that are not part of the constraint tree. We observe that the accuracy of the remaining leaves has also increased dramatically as a result of having constraints (Fig. 5). For example, in the very high ILS case with 50 genes, when 50 species

are missing from the correct constraint tree, the error for the placement of these 50 species has reduced from 21% with no constraints to only 7% with constraints. Similarly strong levels of improvement are observed across all conditions, except when the constraint tree includes only 50 leaves. To summarize, the result demonstrates that given correct prior knowledge about parts of the phylogeny, a constraint ASTRAL search can improve the accuracy of the remaining parts.

Results: biological dataset

We reanalyze the avian phylogenomic dataset [33] with 48 bird species and more than 14,000 loci. The statistical binning method has been proposed to enable coalescent-based analyses of this dataset despite the low phylogenetic signal [34]. The main novel result found using this dataset is the division to Passerea and Columbea at the base of Neoaves, a relationship that was recovered in most analyses of the dataset, including concatenation

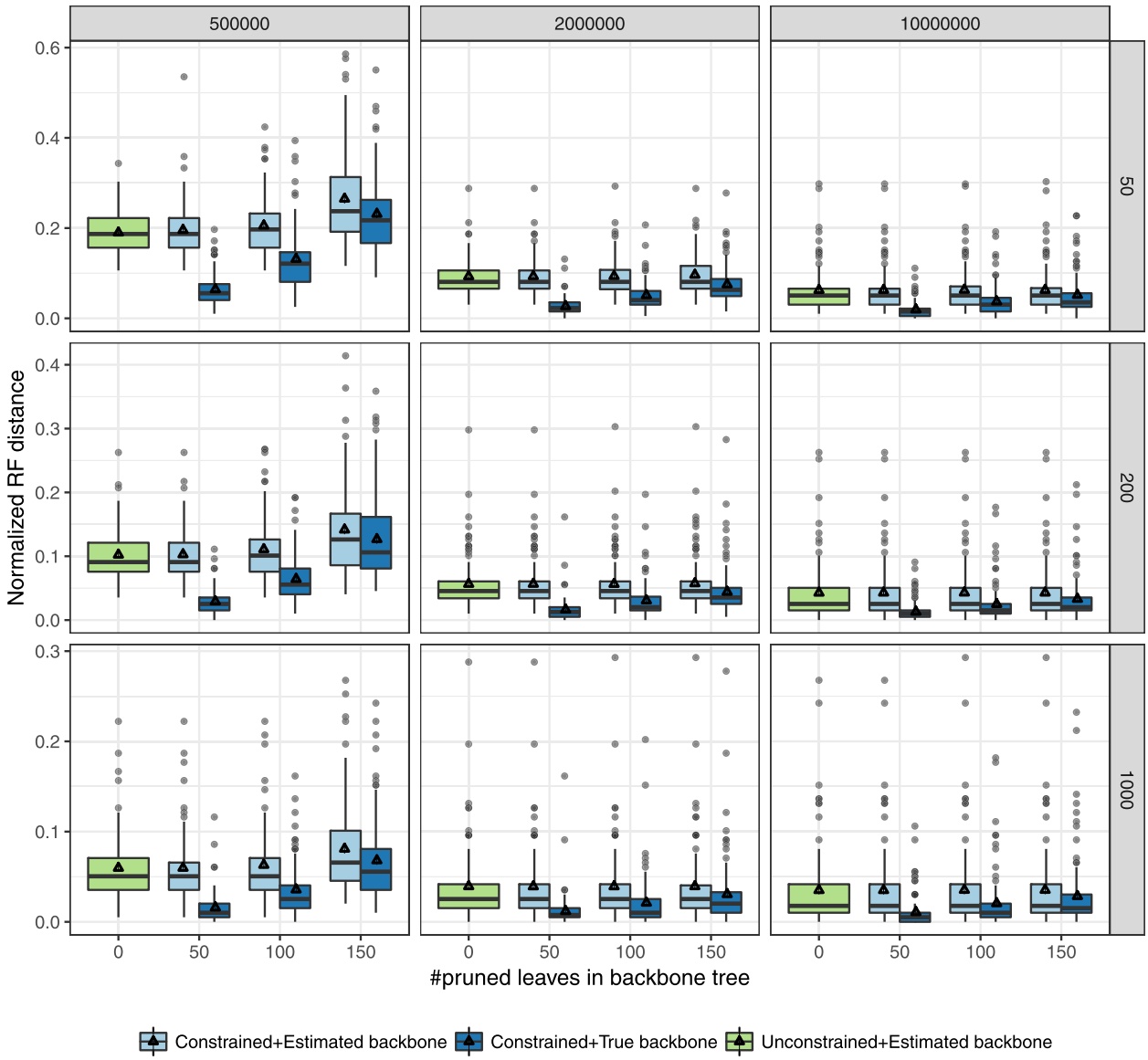

Fig. 2 Effectiveness of constrained ASTRAL with constraint trees that have randomly distributed missing leaves. ASTRAL-III species trees are compared with and without constraints using the Normalized RF distance between inferred species tree and true species tree. Boxplots show distribution (over 100 replicates) and triangles show the mean. Panels correspond to three different levels of ILS (500K, 1M and 2M generations, corresponding to very high, high, and moderate ILS, respectively) and varying number of genes (50, 200, and 1000). The constraint trees are obtained by pruning 50, 100, or 150 (x-axis) randomly chosen leaves from the unconstrained ASTRAL tree or the true species tree

(TENT), MP-EST [35] run on binned gene trees (MP-EST*), and ASTRAL run on unbinned gene trees with low support branches contracted [12]. However, running ASTRAL on 2022 binned gene trees failed to recover Passerea and Columbea and placed Otidimorphae (a clade within Passerea) within Columbea (Fig. 6). Nevertheless, the localPP support [36] for this alternative relationship is low. Thus, using constrained searches, we now ask whether there is support for Columbea/Passerea in the binned gene trees.

Constraining the ASTRAL tree to include Passerea results in recovering Columbea and placing Otidimorphae as sister to other Passerea. The Columbea clade, absent from unconstrained ASTRAL, has high support (0.97 localPP) in the constrained tree. Moreover, the support of the Columbimorphae, a clade universally supported in modern analyses, increases from 0.9 in the unconstrained tree to 1.0. On the other hand, the localPP support for Passerea is only 0.37, which is barely above the expected support of a random resolution (0.33),

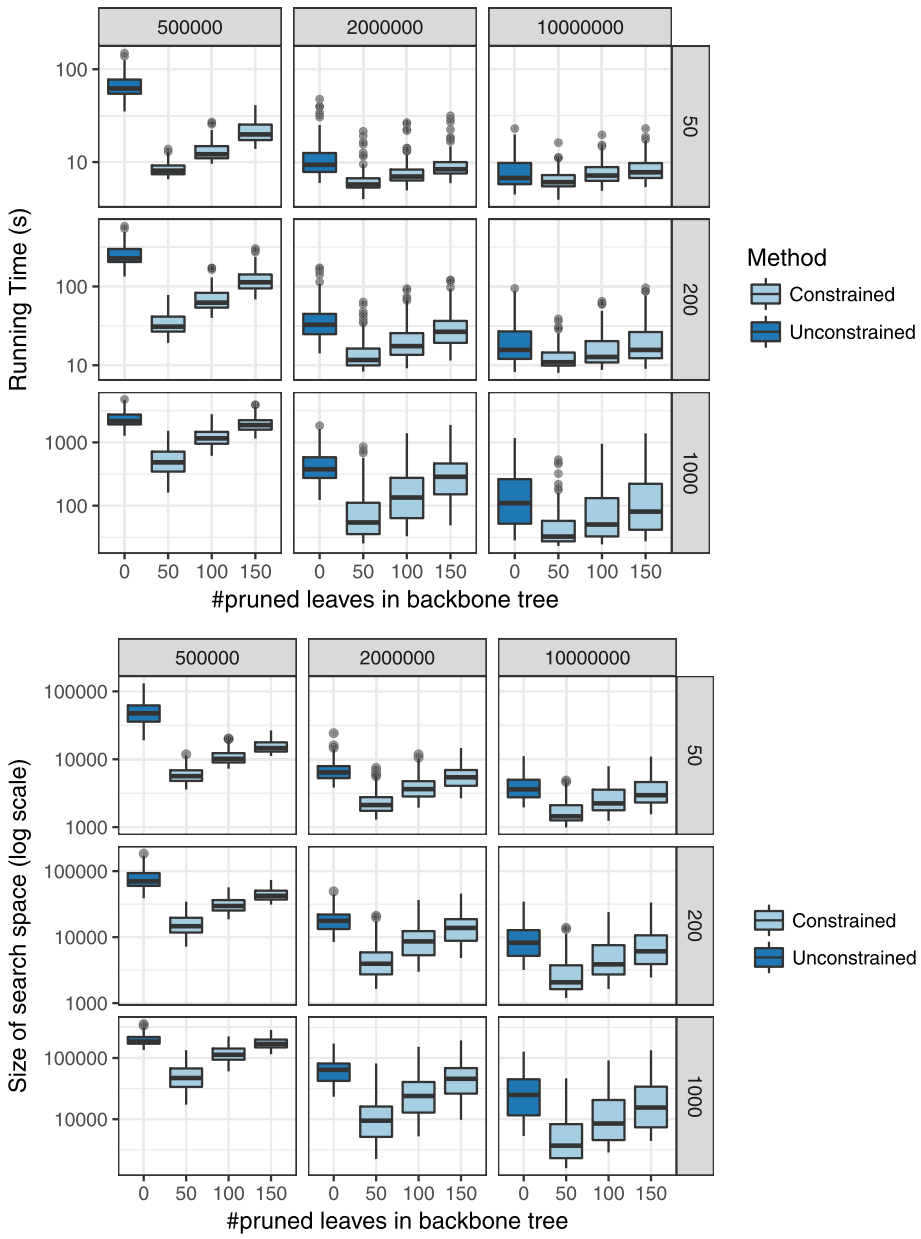

Fig. 3 Impact of constrained search on the running time and search space. The running time (top) and search space size $|X|$ (bottom) are compared between constrained and unconstrained ASTRAL-III. Other settings of the figure are identical to Fig. 2

and the total quartet score of the tree is reduced by 37230 quartets (0.015%). We then performed another constrained analysis forcing Otidimorphae to be with Caprimulgimorphae (as in TENT). This constraint leads to Passerea and Columbea both becoming monophyletic with 0.99 and 0.97 localPP (Fig. 6b). However, the Otidimorphae+Caprimulgimorphae clade itself has low localPP (0.13) and total quartet score reduces by 0.045%. Overall, while the unconstrained ASTRAL tree does not recover Columbea and Passerea, gene trees strongly support Columbea (if not Passerea).

We next tested four other clades recovered in TENT but absent from the unconstrained ASTRAL (Fig. 6b). Several patterns were observed. Forcing Afroaves to be monophyletic reveals a total lack of support for the monophyly of that clade (localPP= 0 and 0.07% reduction in quartet score). Forcing Cuckoo to be sister to Bustard or Hoatzin to be sister to Cursorimorphae shows a case where neither the constrained nor the unconstrained tree have strong support, and thus, results are inconclusive. Most interestingly, owl fits quite well with Coraciimorphae (localPP 0.99 in constrained analyses) as

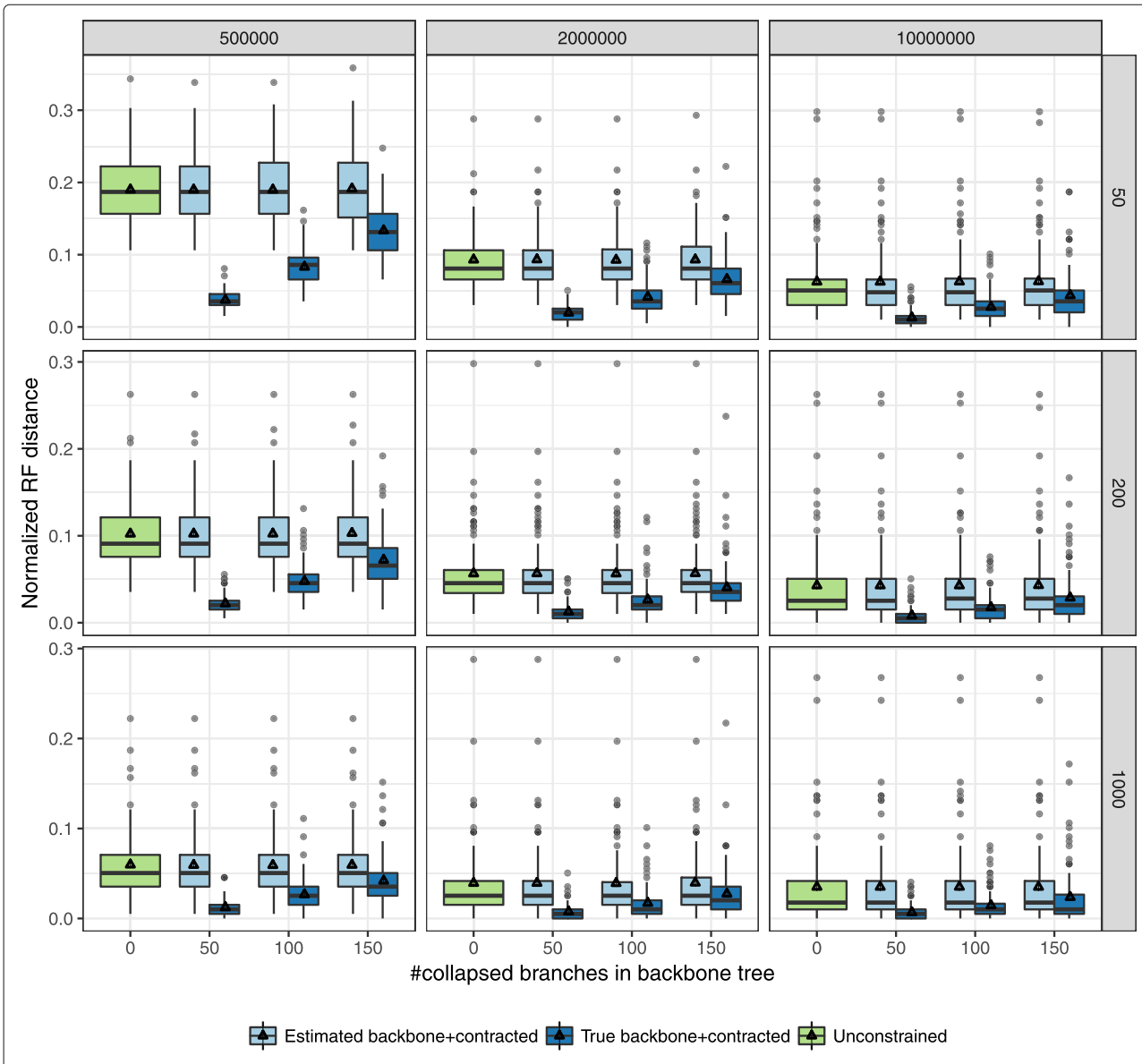

Fig. 4 Effectiveness of constrained ASTRAL with constraint trees that have randomly distributed contracted branches. Settings are similar to Fig. 2. The constraint trees are obtained by collapsing 50, 100, or 150 (x-axis) randomly chosen branches from the unconstrained ASTRAL tree or the true species tree. Note that the set of branches contracted from the true and estimated species trees are not identical

well as its unconstrained position as sister to birds of prey (localPP 1.0); this observation creates a suspicion of gene tree discordance due to processes other than ILS such as hybridization.

Discussion

The method we introduced for honoring user-provided constraints relies on an extension of a tree completion algorithm by Bansal [22]. Our choice of the RF-based tree completion was driven by the availability of the fast B-RF(+) algorithm, which we adopted. Note that our method of forming X is heuristic and the appropriateness of a criterion (such as RF) is an empirical question. However, there is no reason to think that other tree completion/resolution criteria could not have worked equally well or better.

From an algorithmic perspective, constrained search can be considered an extension of the phylogenetic placement problem [37–40]. Unlike placement, constrained search also infers the relationship between query genomes and hence is more informative. Like placement, constrained searches can be used to grow existing trees in an automated fashion and regular basis, without the need to redo the analysis from scratch each time. Taking this idea one step further, constrained ASTRAL can perhaps help develop new divide-and-conquer meta-methods that allow ASTRAL to scale to much larger datasets than what it can currently handle.

By completing and resolving the constrained species tree using input gene trees, we produce a "hybrid" between individual gene trees and the incomplete/unresolved species tree. We showed these hybrid trees are effective in defining search space but we also wondered about accuracy of these trees. Recall that our estimated gene trees have high estimation error (25% to 47% mean RF). Inspired by tree fixing methods [41], we ask if the hybrid gene trees (which, by construction, have reduced RF to the species tree) are better estimates of the true gene trees than the original gene trees. To answer this question, we estimated unconstrained ASTRAL-III trees, collapsed branches with ≤ 0.99 support, and used the resulting tree as T_b and each gene tree as T_r as input to the tree completion algorithm. The resulting hybrid trees had mixed accuracy (Fig. 7). With moderate ILS, hybrid gene trees have reduced error compared to original estimated gene trees; here, true gene trees are similar to the species tree (15% mean RF), and reducing distance to the (collapsed) species tree reduces the error. However, with higher ILS, original gene trees are more accurate perhaps because true gene trees are dissimilar to the species tree (69% mean RF) and making gene trees similar to the species tree is not beneficial. Using the hybrid gene trees as input to ASTRAL increases the species tree error in all cases (from 4.3%, 5.6%, 10.2% mean RF to 4.9%, 8.3%, 20.5%, respectively, for moderate, high, and very high ILS and 200 genes).

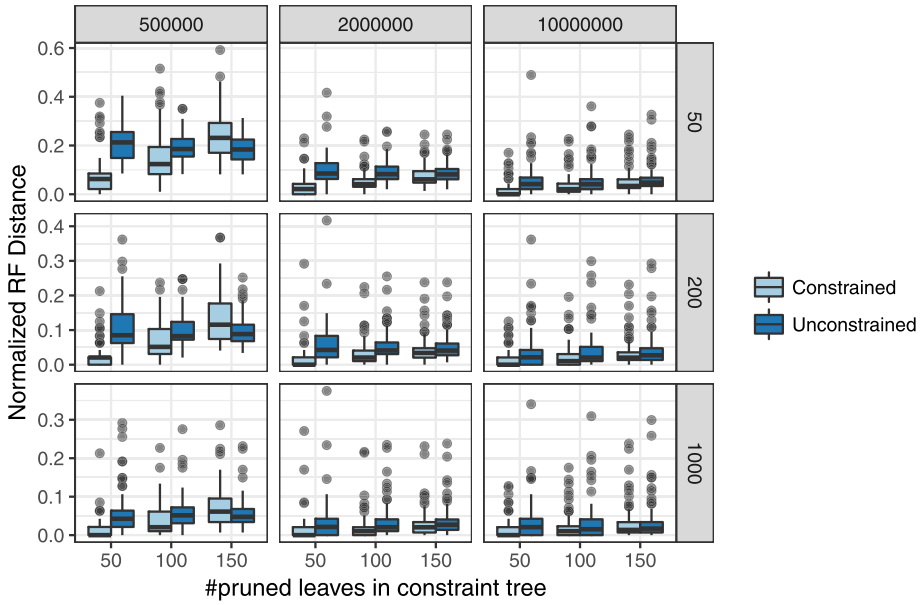

Fig. 5 Impact of correct constraints on some branches on the remaining branches. We show the distribution of the RF distance between (constrained or unconstrained) ASTRAL trees and the true species tree when both trees are restricted to the set of leaves that are *not* present in the constraint tree. As in Fig. 2, constraint trees are defined by pruning 50, 100, or 150 leaves from the true tree

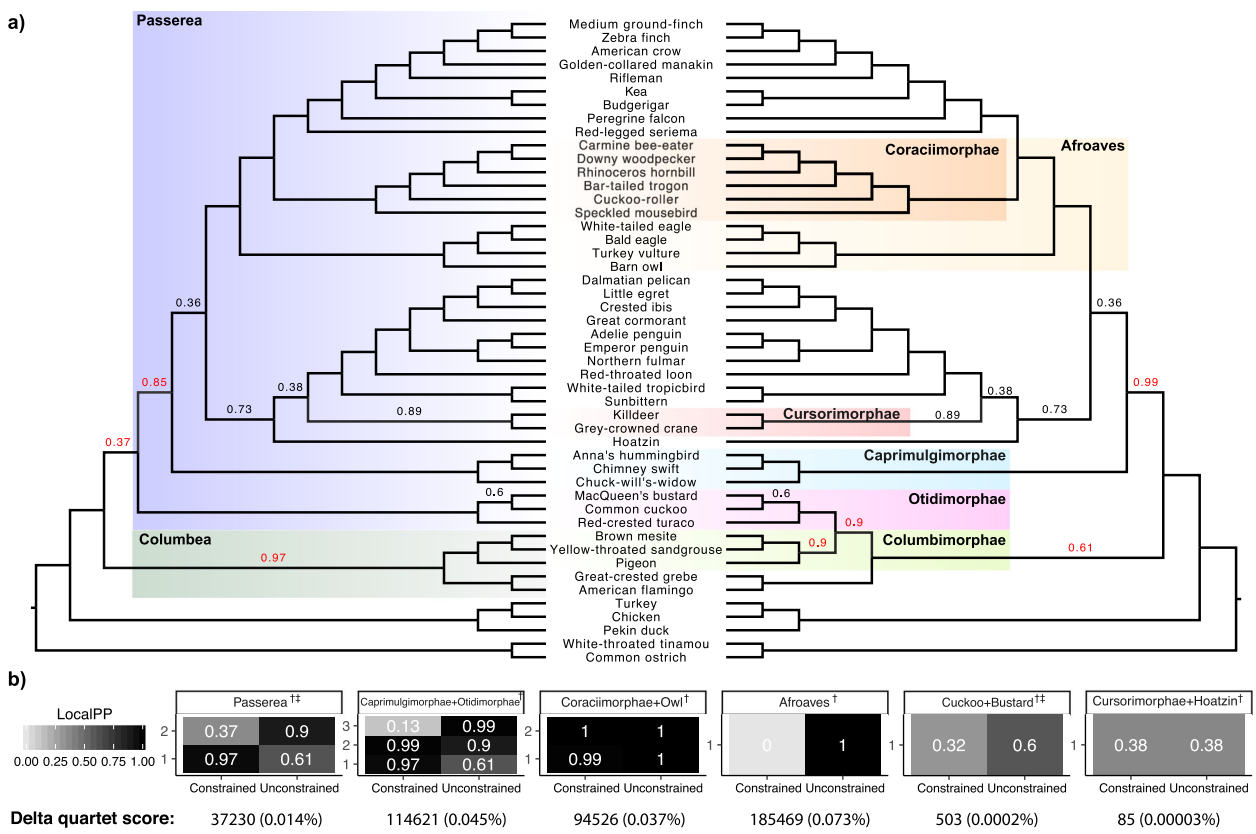

Fig. 6 Constrained analyses reveal support for alternative clades. **a** On the avian genomics dataset [33], we estimated ASTRAL-III trees with no constraints (right) using 2022 binned (super)gene trees. The tree did not include the Passerea vs. Columbea division that most other analyses of this dataset reveal. Constraining the ASTRAL tree to include Passerea as a clade resulted in a tree (left) with two new branches and 0.97 localPP support for Columbea. LocalPP support values below 1.0 are shown on branches (red: change in support across the two trees). **b** Similar to (**a**), constrained analyses are performed to find support for five other clades found in the TENT (†) and (in some cases) in MP-EST* (‡) but not found in the unconstrained analyses. For each clade, we show localPP support for branches that differ between the constrained and unconstrained trees (one to three branches change in constrained searches). We also show the reduction in the quartet score in the constrained analyses as an abosulte number and percentage

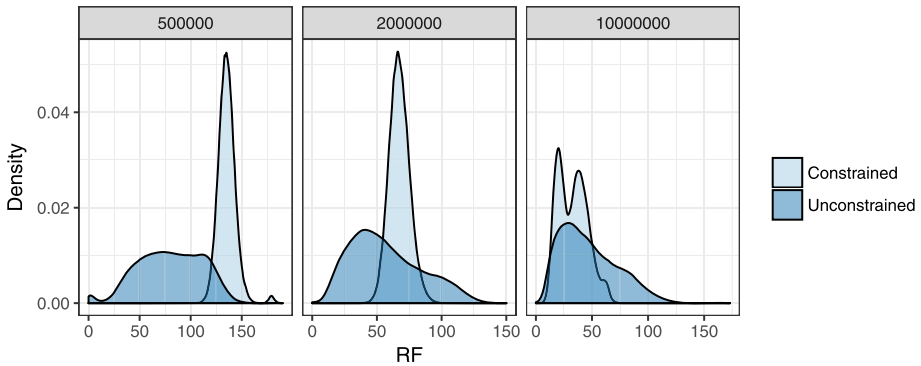

Fig. 7 Accuracy of hybrid gene trees. We resolve the ASTRAL-III tree with branches that have support ≤ 0.99 contracted (as T_b) with respect to each gene tree (as T_r) using our extended tree completion algorithm and call the resulting tree a hybrid gene tree. Density plots show the error of all 1000 hybrid (Constrained) and original (Unconstrained) gene trees, measured using RF distance to true gene trees (out of 396)

The constrained ASTRAL search opens the door for several downstream biological analyses. As shown in our reanalyses of the avian dataset, we can now perform hypothesis-driven analyses, as attempted often by systematists. These analyses have the potential to reveal support for some branches that are not recovered in the main ASTRAL tree. The presence of alternative support can also be visualized using tools such as DiscoVista [42]. Moreover, with constrained searches, we can now test if fixing parts of the tree can impact the resolution or support for the other parts. These analyses can help users understand how robust their estimated species trees are. Finally, methods of interrogating the impact of individual genes, such as Partitioned Coalescent Support (PCS) [43] can also benefit from constrained search. Currently, these methods limit themselves to scoring hand-curated trees but they can instead use automatically generated constrained ASTRAL trees.

Conclusions

We described an algorithm for genome-wide inference of species trees from gene trees while honoring user-provided constraint tree. We have implemented and tested this approach for ASTRAL; however, the same strategy should work for other similar DP methods. Our results showed that the constrained ASTRAL is effective in searching the tree topology space. More interestingly, we showed that constrained search using ASTRAL can help biologists obtain a better understanding of the complexities of genome-wide evolution by revealing support for conflicting resolutions in the species tree.

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s12864-020-6607-z.

Additional file 1: Supplementary material.

Abbreviations

ASTRAL: Accurate species tree algorithm; DP: Dynamic programming; FN: False negative; FP:False positive; ILS: Incomplete lineage sorting; LCA: Least common ancestor; MP-EST: Maximum Pseudo-likelihood for estimatingsSpecies trees; RF distance: Robinson-foulds distance

Acknowledgements

The authors thank anonymous reviewers.

About this supplement

This article has been published as part of *BMC Genomics Volume 21 Supplement 2, 2020: Proceedings of the 17th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop: genomics.* The full contents of the supplement are available online at https://bmcgenomics.biomedcentral.com/articles/supplements/volume-21-supplement-2.

Authors' contributions

Both authors designed the algorithms and contributed to writing. M.R implemented the code and performed analyses. The author(s) read and approved the final manuscript.

Funding

MR and SM were both supported through the National Science Foundation (NSF) grant III-1845967. Computations were performed on the San Diego Supercomputer Center (SDSC) through XSEDE allocations, which is supported by the NSF grant ACI-1053575. The publication cost of this article was funded by the NSF grant IIS-1845967.

Availability of data and materials

Constrained ASTRAL code along with a brief documentation and example for running the code is available on GitHub (https://github.com/maryamrabiee/Constrained-search). Datasets and results are also available on GitHub (https://github.com/maryamrabiee/Constrained-search-data)

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Department of Computer Science and Engineering, UC San Diego, 9500 Gilman Dr, 92093 La Jolla, USA. [2] Department of Electrical and Computer Engineering, UC San Diego, 9500 Gilman Dr, 92093 La Jolla, USA.

Published: 16 April 2020

References

1. Bryant D. Hunting for trees in binary character sets: efficient algorithms for extraction, enumeration, and optimization. J Comput Biol. 1996;3(2):275–88.
2. Hallett MT, Lagergren J. New algorithms for the duplication-loss model. In: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology - RECOMB '00. New York: ACM Press; 2000. p. 138–46. https://doi.org/10.1145/332306.332359.
3. Chang W-C, Górecki P, Eulenstein O. Exact solutions for species tree inference from discordant gene trees. J Bioinforma Comput Biol. 2013;11(05):1342005. https://doi.org/10.1142/S0219720013420055.
4. Bayzid MS, Warnow T. Gene Tree Parsimony for Incomplete Gene Trees. In: 17th International Workshop on Algorithms in Bioinformatics (WABI 2017). Leibniz International Proceedings in Informatics (LIPIcs), volume 88. Dagstuhl: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; 2017. p. 2–1213. https://doi.org/10.4230/LIPIcs.WABI.2017.2.
5. Than C, Nakhleh L. Species Tree Inference by Minimizing Deep Coalescences. PLoS Comput Biol. 2009;5(9):1000501. https://doi.org/10.1371/journal.pcbi.1000501.
6. Vachaspati P, Warnow T. FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. Bioinformatics. 2017;33(5):631–9. https://doi.org/10.1093/bioinformatics/btw600.
7. Bryant D, Steel M. Constructing Optimal Trees from Quartets. J Algorithm. 2001;38(1):237–59. https://doi.org/10.1006/jagm.2000.1133.
8. Mirarab S, Reaz R, Bayzid MS, Zimmermann T, Swenson MS, Warnow T. ASTRAL: genome-scale coalescent-based species tree estimation. Bioinformatics. 2014;30(17):541–8. https://doi.org/10.1093/bioinformatics/btu462.
9. Vachaspati P, Warnow T. SVDquest: Improving SVDquartets species tree estimation using exact optimization within a constrained search space. Mol Phylogenet Evol. 2018;124:122–36. https://doi.org/10.1016/j.ympev.2018.03.006.
10. Mirarab S. Species Tree Estimation Using ASTRAL: Practical Considerations. Arxiv preprint. 2019;1904.03826:.
11. Mirarab S, Warnow T. ASTRAL-II: Coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. Bioinformatics. 2015;31(12):44–52. https://doi.org/10.1093/bioinformatics/btv234.
12. Zhang C, Rabiee M, Sayyari E, Mirarab S. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. BMC Bioinformatics. 2018;19(S6):153. https://doi.org/10.1186/s12859-018-2129-y.

13. Medina M, Collins AG, Silberman JD, Sogin ML. Evaluating hypotheses of basal animal phylogeny using complete sequences of large and small subunit rRNA. Proc Natl Acad Sci. 2001;98(17):9707–12. https://doi.org/10.1073/pnas.171316998.
14. Planet PJ. Tree disagreement: measuring and testing incongruence in phylogenies,. J Biomed Inform. 2006;39(1):86–102. https://doi.org/10.1016/j.jbi.2005.08.008.
15. Tarver JE, dos Reis M, Mirarab S, Moran RJ, Parker S, O'Reilly JE, King BL, O'Connell MJ, Asher RJ, Warnow T, Peterson KJ, Donoghue PCJ, Pisani D. The Interrelationships of Placental Mammals and the Limits of Phylogenetic Inference. Genome Biol Evol. 2016;8(2):330–44. https://doi.org/10.1093/gbe/evv261.
16. Arcila D, Ortí G, Vari R, Armbruster JW, Stiassny MLJ, Ko KD, Sabaj MH, Lundberg J, Revell LJ, Betancur R-R. Genome-wide interrogation advances resolution of recalcitrant groups in the tree of life. Nat Ecol Evol. 2017;1(January):0020. https://doi.org/10.1038/s41559-016-0020.
17. Poe S, Chubb AL. Birds in a Bush : Five Genes Indicate Explosive Evolution of Avian Orders. Evolution. 2004;58(2):404–15.
18. Burleigh JG, Mathews S. Phylogenetic signal in nucleotide data from seed plants: implications for resolving the seed plant tree of life. Am J Bot. 2004;91(10):1599–613. https://doi.org/10.3732/ajb.91.10.1599.
19. Crandall KA, Fitzpatrick JF. Crayfish Molecular Systematics: Using a Combination of Procedures to Estimate Phylogeny. Syst Biol. 1996;45(1):1–26. https://doi.org/10.1093/sysbio/45.1.1.
20. Nguyen N-p, Mirarab S, Liu B, Pop M, Warnow T. TIPP: taxonomic identification and phylogenetic profiling. Bioinformatics. 2014;30(24):3548–55. https://doi.org/10.1093/bioinformatics/btu721.
21. Christensen S, Molloy EK, Vachaspati P, Warnow T. OCTAL: Optimal Completion of gene trees in polynomial time. Algorithm Mol Biol. 2018;13(1):6. https://doi.org/10.1186/s13015-018-0124-5.
22. Bansal MS. Linear-time algorithms for some phylogenetic tree completion problems under robinson-foulds distance. In: RECOMB International Conference on Comparative Genomics. Springer; 2018. p. 209–26.
23. Warnow T. Textbook for 394C : Algorithms for Computational Biology.
24. Christensen S, Molloy EK, Vachaspati P, Warnow T. Optimal completion of incomplete gene trees in polynomial time using OCTAL. In: Leibniz International Proceedings in Informatics, LIPIcs; 2017. https://doi.org/10.4230/LIPIcs.WABI.2017.27.
25. Kupczok A. Split-based computation of majority-rule supertrees. BMC Evol Biol. 2011;11(1):205. https://doi.org/10.1186/1471-2148-11-205.
26. Schieber B, Vishkin U. On finding lowest common ancestors: Simplification and parallelization. In: VLSI Algorithms and Architectures. Berlin/Heidelberg: Springer; 1988. p. 111–23. https://doi.org/10.1007/BFb0040379. http://www.springerlink.com/index/10.1007/BFb0040379.
27. Pamilo P, Nei M. Relationships between gene trees and species trees. Mol Biol Evol. 1988;5(5):568–83.
28. Lafond M, Scornavacca C. On the Weighted Quartet Consensus problem. Theoret Comput Sci. 2019;769:1–17. https://doi.org/10.1016/j.tcs.2018.10.005.
29. Rabiee M, Sayyari E, Mirarab S. Multi-allele species reconstruction using ASTRAL. Mol Phylogenet Evol. 2019;130:286–96. https://doi.org/10.1016/j.ympev.2018.10.033.
30. Mirarab S. Novel scalable approaches for multiple sequence alignment and phylogenomic reconstruction. PhD thesis. 2015.
31. Mallo D, De Oliveira Martins L, Posada D. SimPhy: Phylogenomic Simulation of Gene, Locus, and Species Trees,. Syst Biol. 2016;65(2):334–44. https://doi.org/10.1093/sysbio/syv082.
32. Price MN, Dehal PS, Arkin AP. FastTree-2 – Approximately Maximum-Likelihood Trees for Large Alignments. PLoS ONE. 2010;5(3):9490. https://doi.org/10.1371/journal.pone.0009490.
33. Jarvis ED, Mirarab S, Aberer AJ, Li B, Houde P, Li C, Ho SYW, Faircloth BC, Nabholz B, Howard JT, Suh A, Weber CC, da Fonseca RR, Li J, Zhang F, Li H, Zhou L, Narula N, Liu L, Ganapathy G, Boussau B, Bayzid MS, Zavidovych V, Subramanian S, Gabaldon T, Capella-Gutierrez S, Huerta-Cepas J, Rekepalli B, Munch K, Schierup M, Lindow B, Warren WC, Ray D, Green RE, Bruford MW, Zhan X, Dixon A, Li S, Li N, Huang Y, Derryberry EP, Bertelsen MF, Sheldon FH, Brumfield RT, Mello CV, Lovell PV, Wirthlin M, Schneider MPC, Prosdocimi F, Samaniego JA, Velazquez AMV, Alfaro-Nunez A, Campos PF, Petersen B, Sicheritz-Ponten T, Pas A, Bailey T, Scofield P, Bunce M, Lambert DM, Zhou Q, Perelman P, Driskell AC, Shapiro B, Xiong Z, Zeng Y, Liu S, Li Z, Liu B, Wu K, Xiao J, Yinqi X, Zheng Q, Zhang Y, Yang H, Wang J, Smeds L, Rheindt FE, Braun M, Fjeldsa J, Orlando L, Barker FKK, Jonsson KA, Johnson W, Koepfli K-P, O'Brien S, Haussler D, Ryder OA, Rahbek C, Willerslev E, Graves GR, Glenn TC, McCormack J, Burt D, Ellegren H, Alstrom P, Edwards SV, Stamatakis A, Mindell DP, Cracraft J, Braun EL, Warnow T, Jun W, Gilbert MTPTP, Zhang G. Whole-genome analyses resolve early branches in the tree of life of modern birds. Science. 2014;346(6215):1320–31. https://doi.org/10.1126/science.1253451.
34. Mirarab S, Bayzid MS, Boussau B, Warnow T. Statistical binning enables an accurate coalescent-based estimation of the avian tree. Science. 2014;346(6215):1250463. https://doi.org/10.1126/science.1250463.
35. Liu L, Yu L, Edwards SV. A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evol Biol. 2010;10(1):302.
36. Sayyari E, Mirarab S. Fast Coalescent-Based Computation of Local Branch Support from Quartet Frequencies. Mol Biol Evol. 2016;33(7):1654–68. https://doi.org/10.1093/molbev/msw079.
37. Matsen FA, Kodner RB, Armbrust EV. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinformatics. 2010;11(1):538. https://doi.org/10.1186/1471-2105-11-538.
38. Balaban M, Sarmashghi S, Mirarab S. APPLES: Fast Distance-based Phylogenetic Placement. bioRxiv. 2018475566. https://doi.org/10.1101/475566.
39. Barbera P, Kozlov AM, Czech L, Morel B, Darriba D, Flouri T, Stamatakis A. EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences. Syst Biol. 2019;68(2):365–9. https://doi.org/10.1093/sysbio/syy054.
40. Rabiee M, Mirarab S. INSTRAL: Discordance-aware Phylogenetic Placement using Quartet Scores. bioRxiv. 2018;432906:. https://doi.org/10.1101/432906.
41. Wu Y-C, Rasmussen MD, Bansal MS, Kellis M. TreeFix: Statistically Informed Gene Tree Error Correction Using Species Trees. Syst Biol. 2013;62(1):110–20. https://doi.org/10.5061/dryad.44cb5.
42. Sayyari E, Whitfield JB, Mirarab S. DiscoVista: Interpretable visualizations of gene tree discordance. Mol Phylogenet Evol. 2018;122:110–5. https://doi.org/10.1016/j.ympev.2018.01.019.
43. Gatesy J, Sloan DB, Warren JM, Baker RH, Simmons MP, Springer MS. Partitioned coalescence support reveals biases in species-tree methods and detects gene trees that determine phylogenomic conflicts. Mol Phylogenet Evol. 2019;139:106539. https://doi.org/10.1016/j.ympev.2019.106539.

Publisher's Note