Lafond et al. BMC Bioinformatics 2015, 16(Suppl 14):S4
http://www.biomedcentral.com/1471-2105/16/S14/S4

BMC Bioinformatics

RESEARCH Open Access

Reconstructing a SuperGeneTree minimizing reconciliation

Manuel Lafond[1*], Aïda Ouangraoua[2], Nadia El-Mabrouk[1]

From 13th Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics
Frankfurt, Germany. 4-7 October 2015

Abstract

Combining a set of trees on partial datasets into a single tree is a classical method for inferring large phylogenetic trees. Ideally, the combined tree should display each input partial tree, which is only possible if input trees do not contain contradictory phylogenetic information. The simplest version of the supertree problem is thus to state whether a set of trees is compatible, and if so, construct a tree displaying them all. Classically, supertree methods have been applied to the reconstruction of species trees. Here we rather consider reconstructing a super gene tree in light of a known species tree *S*. We define the supergenetree problem as finding, among all supertrees displaying a set of input gene trees, one supertree minimizing a reconciliation distance with S. We first show how classical exact methods to the supertree problem can be extended to the supergenetree problem. As all these methods are highly exponential, we also exhibit a natural greedy heuristic for the duplication cost, based on minimizing the set of duplications preceding the first speciation event. We then show that both the supergenetree problem and its restriction to minimizing duplications preceding the first speciation are NP-hard to approximate within a $n^{1-\epsilon}$ factor, for any $0 < \epsilon < 1$. Finally, we show that a restriction of this problem to uniquely labeled speciation gene trees, which is relevant to many biological applications, is also NP-hard. Therefore, we introduce new avenues in the field of supertrees, and set the theoretical basis for the exploration of various algorithmic aspects of the problems.

Introduction

A fundamental task in evolutionary biology is to combine a collection of rooted trees on partial, possibly overlapping, sets of data, into a single rooted tree on the full set of data. This is the goal of supertree methods, mainly designed and used for the purpose of reconstructing a species supertree from a set of species trees (see overviews of early methods in [4-6], and more recent methods in [2,10,21,24,25,28,29]).

Ideally, the combined supertree should "display" each of the input tree, in the sense that by restricting the supertree to the leaf set of an input tree, we obtain the same input tree. However, this is not always possible, as the input trees may contain conflicting phylogenetic information. Note that considering a set of input trees that are not all compatible leads to the questions of correcting input gene trees or finding a subset of compatible input trees or subtrees [26]. Here, we leave open these questions and study the more direct formulation of the supertree problem that is to consider a set of compatible input trees and find a supertree displaying them all. The BUILD algorithm by Aho *et al.* [1] can be used to test, in polynomial time, whether a collection of rooted trees is compatible, and if so, construct a compatible supertree, not necessarily fully resolved. This algorithm has been generalized in [9,20] to output all compatible supertrees, and adapted in [27] to output all minimally resolved compatible supertrees.

Although supertree methods are classically applied to the construction of species trees, they can be used as well for the purpose of constructing gene trees. Several gene tree databases are available (see for example

* Correspondence: lafonman@iro.umontreal.ca
[1]Département d'informatique et de recherche opérationnelle, Université de Montréal, Québec, Canada
Full list of author information is available at the end of the article

Ensembl Compara [30], Hogenom [22], Phog [11], MetaPHOrs [23], PhylomeDB [14], Panther [19]). For a gene family of interest, many different gene trees can therefore be available, and finding one single supertree displaying them all leads to a supertree question. On the other hand, given a gene of interest, a homology-based search tool is usually used to output all homologs in a set of genomes. The resulting gene family may be very large, involving distant gene sequences that may be hard to align, leading to weakly supported trees - or even worse, highly supported gene trees that are in fact incorrect. A standard way of reducing such errors is then to use a clustering algorithm based on sequence similarity, such as OrthoMCL [18], InParanoid [3], Proteinortho [17] or many others (see Quest for Orthologs links at http://questfororthologs.org/), to group genes into smaller sets of orthologs or inparalogs (paralogs that arose after a given speciation). Trees obtained for such partial gene families can then be combined by using a supertree method.

Considering input trees as parts of gene trees rather than as parts of species trees does not make any difference regarding the compatibility test procedure. However, for reconstructing a compatible "super gene tree", if a species tree is known for the taxa of interest, then it can be used as an additional information to choose among all possible supertrees displaying the input partial gene trees. Indeed, a natural optimization criterion is to minimize the reconciliation cost, i.e. either the duplication or the duplication plus loss cost, induced by the output tree. We call the problem of finding a compatible supertree minimizing a reconciliation cost *the supergenetree problem*.

In this paper, we first show how the exact methods developed for the supertree problem can be adapted to the supergenetree problem. As for the original algorithms, all the extensions have also exponential worst-time complexity. We then exhibit a heuristic, which can be seen as a greedy approach classically used for the supertree problems, that consists in constructing progressively the tree from its root to its leaves. The main module of this heuristic is to infer the minimum number of duplications preceding the first speciation, which we call the *Minimum pre-Speciation Duplication* problem. We show that the supergenetree problem for the duplication cost, and even its restricted version the Minimum pre-Speciation Duplication problem, are NP-hard to approximate within a $n^{1-\epsilon}$ factor, for any $0 < \epsilon < 1$ (n being the number of genes). Moreover, these inapproximability results even hold for instances in which there is only one gene per species in the input trees. Finally we consider the supergenetree problem with restrictions on input trees that are relevant to many biological applications. Namely, we require each gene to appear in at most one tree, and genes of any tree to be related through orthology only. This is for example the case of gene trees obtained for OrthoMCL clusters called orthogroups [18]. We show that even for this restriction, the supergenetree problem remains NP-hard for the duplication cost.

The following section introduces preliminary notations that will be required in the rest of the paper.

Preliminaries

Notations on trees

Given a set L, a *tree T for L* is a rooted tree whose leafset $\mathcal{L}(T)$ is in bijection with L. We denote by $V(T)$ the set of nodes and by $r(T)$ the root of T. Given an internal node x of T, the subtree of T rooted at x is denoted T_x. The *degree* of an internal node x of T is the number of children of x. If T is binary, we arbitrarily set one of the two children of x as the left child x_l and the other as the right child x_r. We call $\left(\mathcal{L}\left(T_{x_l}\right), \mathcal{L}\left(T_{x_r}\right)\right)$ the bipartition of a node x of degree 2 (note that the term 'bipartition' is sometimes used, in the context of unrooted trees, to denote the nodes or leaves of the two components obtained after removing a given edge. To avoid confusion, note that this is not what we mean here by 'bipartition').

A node x is an *ancestor* of y if x is on the (inclusive) path between y and the root, and we then call y a descendant of x. Two nodes x and y are *separated* in T if none is an ancestor of the other. The *lowest common ancestor* (lca) of a subset L' of $\mathcal{L}(T)$, denoted $lca_T(L')$, is the ancestor common to all nodes in L' that is the most distant from the root. The restriction $T|_{L'}$ of T to L' is the tree with leafset L' obtained from the subtree of T rooted at $lca_T(L')$ by removing all leaves that are not in L', and contracting all internal nodes of degree 2, except the root. We generalize this notation to a set of trees: For a set $\mathcal{T}$ of trees on L, $\mathcal{T}|_{L'} = \{T|_{L'} : T \in \mathcal{T}\}$. Let T' be a tree such that $\mathcal{L}(T') = L' \subseteq \mathcal{L}(T)$. We say that T displays T' iff $T|_{L'}$ is the same tree as T'.

A *triplet* is a binary tree on a set L with $|L| = 3$. For $L = \{x, y, z\}$, we denote by $xy|z$ the unique triplet t on L with root $r(t)$ for which $lca_t(x, y) \neq r(t)$ holds.

A *polytomy* (or star tree) over a set L is a tree for L with a single internal node, which is of degree $|L|$.

A *resolution* $B(T)$ of a non-binary tree T is a binary tree respecting all the ancestral relations given by T. More precisely, $B(T)$ is a binary tree such that $\mathcal{L}\left(B(T)\right) = \mathcal{L}(T)$, and for any $u, v \in V(T)$, if u is an ancestor of v in T, then $lca_{B(T)}\left(\mathcal{L}\left(T_u\right)\right)$ is an ancestor of $lca_{B(T)}\left(\mathcal{L}\left(T_v\right)\right)$.

Gene and species trees

Figure 1 is an illustration of the notations defined in this section.

A species tree S for a set $\Sigma = \{\sigma_1,...,\sigma_t\}$ of species represents an ordered set of speciation events that have led to Σ: an internal node is an ancestral species at the moment of a speciation event, and its children are the new descendant species. Inside the species' genomes, genes undergo speciation when the species to which they belong do, but also duplications and losses (other events such as transfers can happen, but we ignore them here). A *gene family* is a set of genes Γ accompanied with a *mapping function* $s : \Gamma \to \Sigma$ mapping each gene to its corresponding species.

Consider a gene family Γ where each gene $x \in \Gamma$ belongs to a species $s(x)$ of Σ. The evolutionary history of Γ can be represented as a *gene tree* T for Γ, which is a rooted binary tree with its leafset in bijection with Γ, where each internal node refers to an ancestral gene at the moment of an event (either speciation or duplication). The mapping function s is generalized as follows: if x is an internal node of T, then $s(x) = lca_S\left(\left\{s\left(x'\right) : x' \in \mathcal{L}\left(T_x\right)\right\}\right)$.

An internal node x of T is called a *speciation node* if $s(x_l)$ and $s(x_r)$ are separated in S. Otherwise, x is a *duplication node* preceding the speciation event $lca_S(s(x_l), s(x_r))$ if $lca_S(s(x_l), s(x_r))$ is an internal node of S, otherwise it is a duplication inside the extant species $lca_S(s(x_l), s(x_r))$. A duplication node x such that $s(x) = r(S)$ is called a *pre-speciation duplication* node. A gene tree T with all internal nodes being speciation nodes is called a *speciation tree*. Two genes x, y of $\mathcal{L}(T)$ are *orthologs in* T if their $lca_T(x, y)$ is a speciation node.

The *duplication cost* of T is the number of duplication nodes of T. It reflects the minimum number of duplications required to explain the evolution of the gene family inside the species tree S according to T. A well-known reconciliation approach [7,8] allows to further recover, in linear time, the minimum number of losses underlined by such an evolutionary history. We refer to the minimum number of duplications and losses required to explain T with respect to S as the *reconciliation cost* of T with respect to S, or simply the reconciliation cost if there is no ambiguity on the considered trees.

Supergenetree problem statement

A set $\mathcal{T}$ of gene trees is said *consistent* if there is a tree T, called a *supergenetree* for $\mathcal{T}$ displaying each tree of $\mathcal{T}$, and *inconsistent* otherwise. A supergenetree T for $\mathcal{T}$ is said *compatible* with $\mathcal{T}$. For example, the four triplets in Figure 2 are consistent, and the gene tree T of Figure 1 is compatible with them. However, adding the dotted tree to the set of triplets makes the gene tree set incompatible. Consistency of a set of trees can be tested in polynomial time [1]. For a consistent set of trees, the problem considered here is to find a compatible gene tree of minimum reconciliation cost with respect to a given species tree. A formal statement of the general problem follows.

MINIMUM SUPERGENETREE PROBLEM (MINSGT PROBLEM):

Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,\ 1 \le i \le k}$ of subsets of Γ, and a set $\mathcal{T} = \{G_1, G_2,..., G_k\}$ of consistent gene trees where, for each $1 \le i \le k$, G_i is a tree for Γ_i.

Output: Among all gene trees for Γ compatible with $\mathcal{T}$, one tree T of minimum reconciliation cost.

When the considered cost is the duplication cost, the problem is called the Minimum Duplication SuperGeneTree Problem (MinDUPSGT problem).

From the SuperTree to the SuperGeneTree Problem

The classical supertree problem is to state whether or not a set of partial trees are consistent, and if so construct a tree containing them all. Here, we introduce the classical methods for solving this problem, and explore natural generalizations to the supergenetree problem.

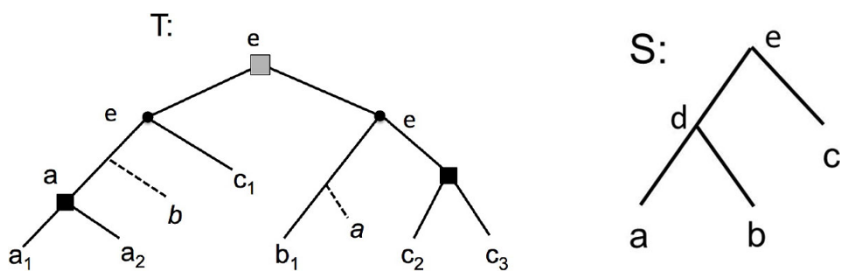

Figure 1 A gene tree T for the gene family $\Gamma = \{a_1, a_2, b_1, c_1, c_2, c_3\}$ and a species tree S for the set of species $\Sigma = \{a, b, c\}$ and where, for any x and any i, x_i is a gene in genome x. The label of an internal node x of T corresponds to $s(x)$. Speciation nodes are represented by circles and duplication nodes by squares. The pre-speciation duplication nodes (here only one node) are grey-colored. The dotted lines represent losses that are inferred by a most parsimonious reconciliation algorithm. The duplication cost of T is 3 and its reconciliation cost is 5.

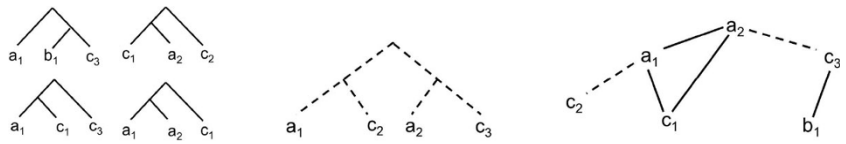

Figure 2 Genes trees (left and middle) and their corresponding triplet graphs (right). Plain edges of the graph correspond to the four triplet trees, while dotted edges correspond to the triplets of the four-leaves tree.

Let Γ be a set of n taxa (usually species in case of the supertree problem, and genes in case of the supergenetree problem), $\Gamma_{i,\ 1 \le i \le k}$ be a set of possibly overlapping subsets of Γ, and $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ be a set of trees where, for each $1 \le i \le k$, G_i is a tree for Γ_i. Let $tr(\mathcal{G})$ be the set of triplets of $\mathcal{T}$ defined as $tr(\mathcal{G}) = \left\{xy|z : \exists\ 1 \le i \le k \text{ such that } G_i|_{\{x,\ y,\ z\}} = xy|z\right\}$. Let $\mathcal{T}(\Gamma, E)$ be the triplet graph with the set of vertices Γ and the set of edges $E = \{xy : \exists\ z \in \Gamma$ such that $xy|z \in tr(\mathcal{G})\}$ (see Figure 2 for an example).

The classical BUILD algorithm [1] determines, in polynomial time, whether a set of triplets is consistent and if so constructs a tree T, possibly non-binary, compatible with them. The algorithm takes as input the graph $\mathcal{T} = \mathcal{T}(\Gamma, E)$. Let $\mathcal{C}(\mathcal{T}) = \{C_1, \cdots, C_m\}$ be the set of connected components of $\mathcal{T}$. If $\mathcal{T}$ has at least three vertices and $|\mathcal{C}(\mathcal{T})| = 1$, then $\mathcal{T}$ is inconsistent, and the algorithm terminates. For example, the set of five gene trees of Figure 2 is inconsistent, as the corresponding triplet graph (including dotted lines) is connected. Otherwise, if $|V(\mathcal{T})| \ge 3$, a polytomy is created over $\mathcal{C}(\mathcal{T})$, the internal node of the polytomy being the root $r(T)$ of the compatible tree T under construction and its children being m subtrees with leafsets $V(C_1)$,..., $V(C_m)$, with their topology yet to be determined (where $V(C_i) \subseteq \Gamma$ denotes the set of taxa appearing in C_i). The algorithm then recurses into each connected component, i.e. the subtree for $V(C_i)$ is determined recursively from the graph $\mathcal{T}\left(V(C_i), E|_{C_i}\right)$ defined by $E|_{C_i} = \{xy : \exists\ z \in \Gamma$ such that $xy|z \in tr\left(\mathcal{G}|_{V(C_i)}\right)\}$. If, at any step, the considered graph has a single component containing more than two vertices, then $\mathcal{T}$ is reported as an inconsistent set of trees and the algorithm terminates. Otherwise, recursion terminates when the graph has at most two vertices, eventually returning a supertree T. See Figure 3 for an example.

The BUILD algorithm has been generalized in an algorithm called AllTrees [20] to output all supertrees compatible with a set of triplets in case consistency holds. Instead of taking each element of $\mathcal{C}(\mathcal{T})$ as a separate leaf of $r(T)$, all possible groupings, in other words all partitions of $\mathcal{C}(\mathcal{T})$, are considered (see Figure 3, right, for a choice of bipartitions). For each partition $\mathcal{P}(\mathcal{C}(\mathcal{T}))$ of $\mathcal{P}(\mathcal{C}(\mathcal{T}))$, a polytomy is created over $\mathcal{P}(\mathcal{C}(\mathcal{T}))$. The algorithm then iterates by considering each possible partition of each subgraph induced by each element of $\mathcal{P}(\mathcal{C}(\mathcal{T}))$. The algorithm is polynomial in the size of the output that may be exponential in the size of the input.

A tree T compatible with $\mathcal{T}$ such that no internal edge of T can be contracted so that the resulting tree is also compatible with $\mathcal{T}$ is called a minimally resolved supertree. Minimally resolved supertrees contain all the information about all supertrees compatible with $\mathcal{T}$ but in a "compressed" format. By exhibiting some properties on graph components, Semple shows in [27] how some partitions of the triplet graph components can be avoided without loss of generality. The new developed algorithm, named AllMinTrees [27], outputs a minimally resolved tree in polynomial time. However, it was shown in [15] that the cardinality of the solution space can be exponential in $n = |\Gamma|$, leading to an exponential time algorithm with $\Omega\left(\frac{n}{2}^{\frac{n}{2}}\right)$.

Notice that, in general, the trees output by all these methods are non-binary trees.

Extensions to the SuperGeneTree problem

Natural exact solutions for the supertree problem can be extended to the supergenetree problem as follows:

(1) Use AllMinTrees to output all minimally resolved supertrees, and for each one which is non-binary in general, find in linear time a resolution minimizing the reconciliation [16,32] or duplication [31] cost. Among all optimally resolved trees, select one of minimum cost. Clearly this approach has the same complexity as the AllMinTrees algorithm, multiplied by a factor of n to resolve each tree, which is $\Omega\left(n \cdot \frac{n}{2}^{\frac{n}{2}}\right)$.

(2) As we are seeking a binary tree, each created node x of the supergenetree T under construction should determine a bipartition $\left(\mathcal{L}\left(T_{x_l}\right), \mathcal{L}\left(T_{x_r}\right)\right)$. Therefore, the AllTrees algorithm can be simplified by considering, instead of all partitions of $\mathcal{C}(\mathcal{T})$,

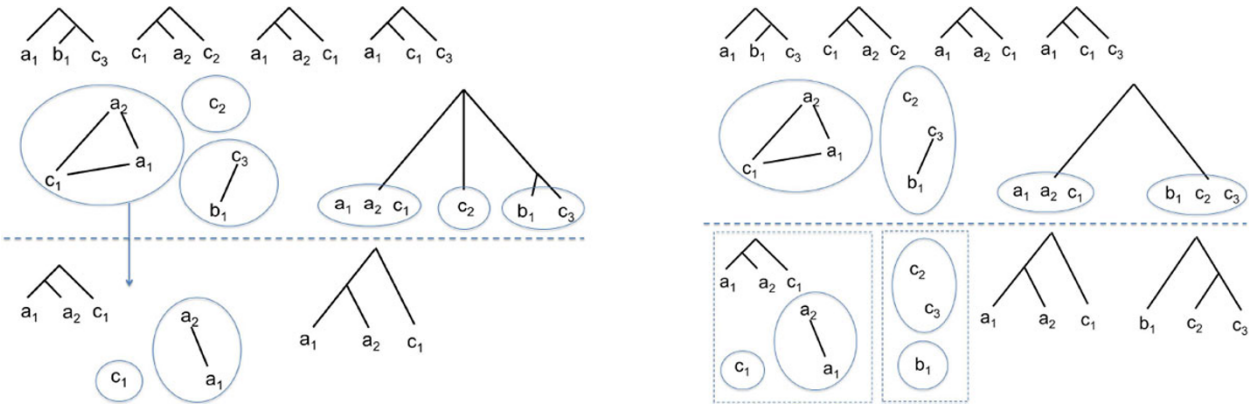

Figure 3 Left: Execution of the BUILD algorithm on the set of given four triplets. This example requires two iterations of the algorithm (delimited by a dotted line). At the first iteration, the triplet graph contains three components, leading to a polytomy with three leaves. The algorithm then iterates on the component $\{a_1, a_2, c_1\}$, which terminates the supertree reconstruction procedure. Notice that the gene tree of Figure 1, which is compatible with the four triplets, is not a resolution of this non-binary tree; Right: A variant of the BUILD algorithm, with the triplet graph components grouped into bipartitions - in this case leading to a fully resolved tree. This tree is actually the gene tree T of Figure 1.

only all bipartitions of the triplet graph components set. See an example in Figure 3, right. Notice that this simplification approach is not applicable to the AllMinTrees algorithm, as by imposing bipartitions, the minimum resolution condition cannot be guaranteed.

A branch-and-bound approach

The tree space which is explored by the two exact methods described above can be reduced by using a branch-and-bound approach. Consider for example method (1) using the AllMinTrees algorithm. At each iteration of computing one minimally resolved tree, resolve the intermediate non-binary tree obtained at this step, using for example the linear-time algorithm presented in [16]. If its reconciliation cost is greater than the cost of a full tree already obtained at a previous stage of the AllMinTrees algorithm, then stop expanding this tree as this can only increase the reconciliation cost.

A dynamic programming approach

The recursive top-down method (2) can instead be handled by a dynamic programming approach computing the minimum reconciliation cost of a tree on a subset of Γ according to the reconciliation costs of trees on smaller subsets, similarly to the wrok done in [13].

More precisely, let P be an arbitrary subset of Γ, and denote by $R(P)$ the minimum duplication cost of a tree $T|_P$ having leafset P and compatible with the set $\mathcal{G}|_P = \{G_1|_P, G_2|_P, \cdots, G_k|_P\}$. Let $\mathcal{T}(P, E|_P)$ be the BUILD graph restricted to P and $\mathcal{G}|_P$, and $C(\mathcal{T}) = \{C_1, \cdots, C_m\}$ the set of its connected components. If $C \subseteq \mathcal{C}(\mathcal{T})$, by $V(C)$ we mean $\cup_{C_i \in C} V(C_i)$. Denote the complement of C by $\bar{C} = \mathcal{C}(\mathcal{T}) \setminus C$. Finally set $d(V(C), V(\bar{C}))$ to 0 if $s(V(C))$ and $s(V(\bar{C}))$ are separated in S, in which case $(V(C), V(\bar{C}))$ is the bipartition of a speciation node, and 1 otherwise i.e. if $(V(C), V(\bar{C}))$ is the bipartition of a duplication node. Then:

$$R(P) = \min_{C \subset \mathcal{C}(\mathcal{T})} R(V(C)) + R\left(V\left(\bar{C}\right)\right) + d\left(V(C), V\left(\bar{C}\right)\right)$$

the value of interest being $R(\Gamma)$. First note that, assuming constant-time *lca* queries over S, $d(V(C), V(\bar{C}))$ can be computed in constant time if $s(V(C))$ and $s(V(\bar{C}))$ can be accessed in constant time, since if suffices to check that the *lca* of $s(V(C))$ and $s(V(\bar{C}))$ differs from both. To achieve this, we precompute $s(X)$ for every subset X of Γ of size 1, 2,..., n in increasing order. Noting that if $|X| > 1$, then for any $x \in X$, $s(X) = lca_S(s(X \setminus \{x\}), s(x))$, $s(X)$ can be computed in constant time assuming that $s(X \setminus \{x\})$ was computed previously and assuming constant-time *lca* queries. As there are 2^n subsets of Γ, each computed in constant time, this preprocessing step takes time $O(2^n)$.

As for $R(\Gamma)$, we can simply ensure that each $R(P)$ is computed at most once by storing its value in a table for subsequent accesses (i.e. when $R(P)$ is needed, we use its value if it has been computed, or compute it and store it otherwise). In this manner, each subset P takes time, not counting the recursive calls, proportional to $|P||\mathcal{G}| + |P| + 2^{|\mathcal{C}(\mathcal{T})|}$ to construct $\mathcal{T}(P, E|_P)$, find $\mathcal{C}(\mathcal{T})$, and evaluate each bipartition of $\mathcal{C}(\mathcal{T})$. We will simply use the fact that $|P||\mathcal{G}| + |P| + 2^{|\mathcal{C}(\mathcal{T})|}$ is in $O(2^n)$. As this has to be done for, at worst, each of the 2^n subsets of Γ, we get a total time $O(2^n + 2^n \cdot 2^n) = O(4^n)$. Note that this analysis probably overestimates the actual

complexity of the algorithm, as we are assuming that each subset P and each component set $\mathcal{C}(\mathcal{T})$ are both always of size n. It is also worth mentioning that the $R(P)$ recurrence can easily adapted to the mutation cost (duplications + losses).

A greedy heuristic for the duplication cost

Instead of trying all partitions of the triplet graph components set at each step of the AllTrees or AllMinTrees algorithms, if the goal is to minimize the duplication cost, then a natural greedy approach would be to choose the best partition at each iteration, namely the one allowing to minimize the number of duplications preceding each speciation event. Such an approach would result in pushing duplications down the tree. It leads to the following restricted version of the supergenetree problem.

MINIMUM PRE-SPECIATION DUPLICATION PROBLEM (MINPRESPEDUP PROBLEM):

Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,1\leq i\leq k}$ of subsets of Γ, and a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of consistent gene trees where, for each $1 \leq i \leq k$, G_i is a tree on Γ_i.

Output: Among all gene trees for Γ compatible with $\mathcal{T}$, one tree T with minimum pre-speciation duplication nodes.

We will show in the following section that even this restricted version of the supergenetree problem is hard. Here, we give the intuition of a natural way of solving this problem, that reduces to repeated applications of the Max-Cut problem. Although known to be NP-hard, efficient heuristics exist (up to a factor of 0.878 [12]), that can be used for our purpose.

For the supertree problem, the triplet graph $\mathcal{T} = \mathcal{T}(\Gamma, E)$ represents all triplets of the input trees that have to be combined. In the case of the supergenetree problem, another tree is available, the species tree S. A triplet $xy|z$ found in the input trees $\mathcal{T}$ can be reconciled with S, and if $r(xy|z)$ is a duplication, then any tree compatible with G must contain this duplication. Say that $r(xy|z)$ is a required duplication mapped to $r(S)$ if $s(r(xy|z)) = r(S)$ and $r(xy|z)$ is a duplication. Let us include this information in $\mathcal{T}$. More precisely, let $\mathcal{C} = \mathcal{C}(\mathcal{T})$ denote the set of connected components of $\mathcal{T}$, and let $\mathcal{T}(\mathcal{C})$ be the graph whose vertex set is $\mathcal{C}$, and $C_1, C_2 \in \mathcal{C}$ share an edge if C_1 has vertices x, y and C_2 has a vertex z such that $xy|z$ is a triplet in $\mathcal{T}$ with $r(xy|z)$ being a required duplication mapped to $r(S)$. If there are, say, d distinct such triplets, one can possibly set a weight of d to the C_1C_2 edge. See Figure 4 for an example.

Consider the problem of clustering the components of $\mathcal{T}(\mathcal{C})$ into two parts B_1, B_2 of a bipartition in a way minimizing the number of duplications preceding the speciation event $r(S)$. For each $C_1 \in B_1$ and $C_2 \in B_2$ such that C_1C_2 is an edge of $\mathcal{T}(\mathcal{C})$, a tree T rooted at the bipartition (B_1, B_2) contains the required duplications mapped to $r(S)$ represented by the C_1C_2 edge. If there are k such edges between B_1 and B_2 totalizing a weight of w, the single duplication at the root of T contains those w required duplications. In other words, we have "merged" w required duplications into one. It then becomes natural to find the bipartition of $\mathcal{T}(\mathcal{C})$ that merges a maximum of duplications, i.e. that contains a set of edges crossing between the two parts of maximum weight. This is the well-known Max-Cut problem. For instance in Figure 4, the Max-Cut has a weight of 3 and leads to the optimal tree T_1. Any other bipartition sends a required duplication to a lower level and is hence suboptimal. The T_2 tree is obtained from first taking the suboptimal $(\{a_1, b_1, d_1\}, \{c_1, e_1, f_1\})$ bipartition, which creates a duplication at the root and defers the $c_1e_1|f_1$ required duplication for later.

Note however that the components of $\mathcal{T}$ may contain required duplications themselves, which are not represented by the edges of $\mathcal{T}(\mathcal{C})$. Thus, a Max-Cut must then be applied recursively on both parts of the chosen bipartition. Therefore, this method does not benefit directly from the efficient approximation factor known for the Max-Cut problem, as the approximation error stacks with each application. In the next section, we show that, unlike Max-Cut, the MinPreSpeDup problem cannot admit a constant factor approximation (unless P = NP).

Inapproximability of the MinDupSGT and MinPreSpeDupSGT problems

Through the rest of this section, we denote by $n = |\Gamma|$ the size of the considered gene family. We show that both the MinDupSGT problem and its restriction the MinPreSpeDupSGT problem are NP-hard.

Theorem 1 *The* MinDupSGT *and* MinPreSpeDupSGT *problems are both NP-hard to approximate within a factor of* $n^{1-\epsilon}$ *for any constant* $0 < \epsilon < 1$. *Moreover, this result holds for both problems even when restricted to instances having at most one gene per species in* Γ.

Proof We use a reduction from the minimum k-colorability problem. Recall that a graph $H = (V, E)$ is k-colorable if there is a partition $\{V_1, V_2, \ldots, V_k\}$ of V into independent sets (i.e. if $x, y \in V_i$ for some $1 \leq i \leq k$, then $xy \notin E$). It is now well-known [33] that the smallest k for which H is k-colorable cannot be approximated within a factor of $|V|^{1-\epsilon}$ unless $P = NP$.

Now, given a graph $H = (V, E)$, we construct a gene set Γ, a set of rooted triplet gene trees $\mathcal{T}$ and a species

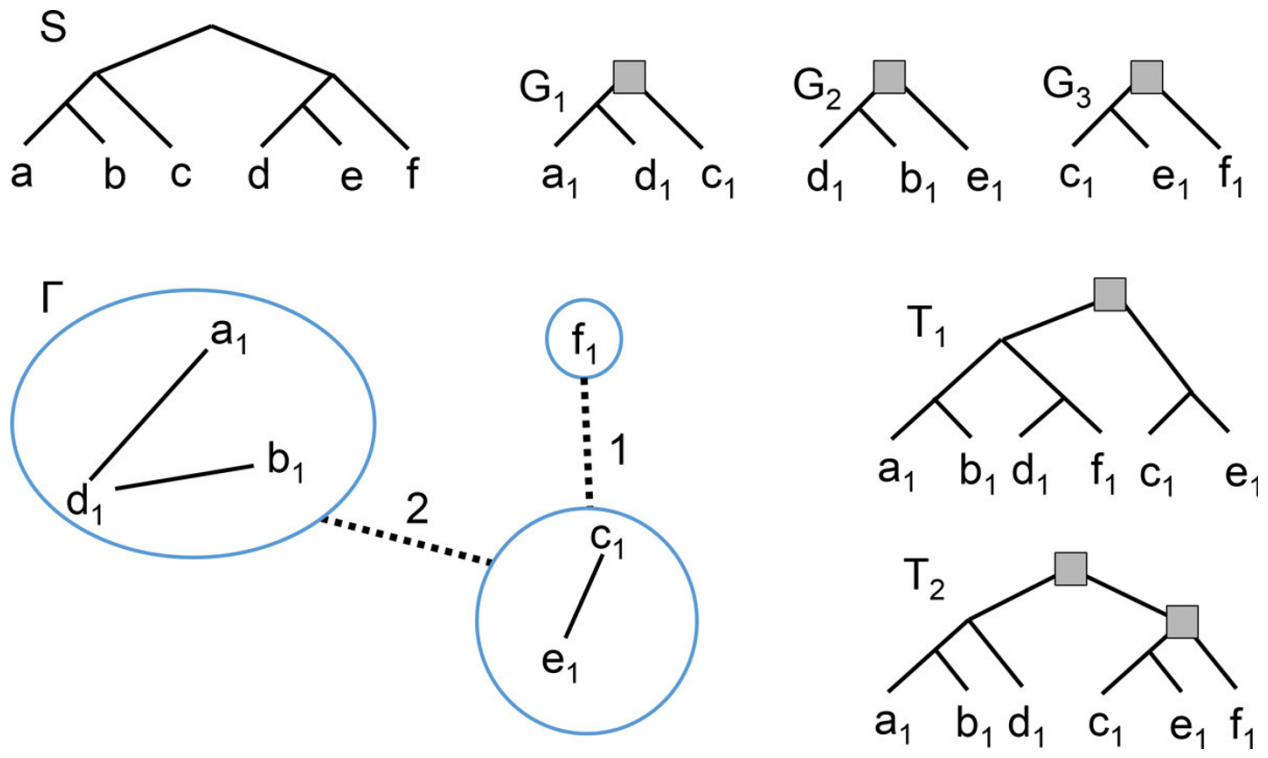

Figure 4 Example of how Max-Cut can be applied to the MinPreSpeDup problem. S is a species tree, $\mathcal{G} = \{G_1, G_2, G_3\}$ and $\mathcal{T}$ is the BUILD graph (solid edges). Its connected components are enclosed in circles, and the dotted edges represent required duplications mapped to $r(S)$. The edge of weight 2 is explained by the $a_1d_1|c_1$ and $d_1b_1|e_1$ triplets, whereas the edge of weight 1 is explained by the $c_1e_1|f_1$ triplet. A Max-Cut creates the bipartition ($\{a_1, b_1, d_1, f_1\}$, $\{c_1, e_1\}$), leading to the T_1 tree which merges all required duplications at its root. The tree T_2 is obtained from the suboptimal bipartition ($\{a_1, b_1, d_1\}$, $\{c_1, e_1, f_1\}$) and has 2 duplications.

tree S such that H is k-colorable if and only if $\mathcal{T}$ is compatible with some gene tree T having at most $k - 1$ duplications when reconciled with S. Using the same construction, we also show that H is k-colorable if and only if $\mathcal{T}$ is compatible with some gene tree T having at most $k - 1$ pre-speciation duplications when reconciled with S. In both cases, the gene-species mapping s is bijective, proving the second part of the theorem statement.

Let $\Gamma = \{v_1, v_2 : v \in V\}$ and for each edge $vw \in E$, add the triplets $v_1v_2|w_1$, $v_1v_2|w_2$, $w_1w_2|v_1$ and $w_1w_2|v_2$ to $\mathcal{T}$. Observe that this forces any tree T that displays $\mathcal{T}$ to display the tree $((v_1, v_2), (w_1, w_2))$. Add one species to Σ for each gene of Γ so that the gene-species mapping s is bijective. As for S, first let S_1 be any binary tree with one leaf for each member of $\{s(v_1) : v \in V\}$, and in the same manner let S_2 be any binary tree with one leaf for each member of $\{s(v_2) : v \in V\}$. Obtain S by connecting the root of S_1 and the root of S_2 under a common parent $r(S)$. Thus $s(v_1)$ and $s(v_2)$ are separated by $r(S)$ for any $v \in V$. Clearly, $\mathcal{T}$ and S can be constructed in polynomial time.

Claim 1 : if H is k-colorable, then we can find a tree T compatible with $\mathcal{T}$ having at most $k - 1$ duplications. Moreover each such duplication x is a pre-speciation duplication (i.e. $s(x) = r(S)$).

Let $\{V_1, V_2, ..., V_k\}$ be a k-coloring of H. For each $1 \leq i \leq k$, let T_i be the tree with leafset $V_i' = \{v_1, v_2 : v \in V_i\}$ that has only speciations, i.e. T_i is $S|_{s(V_i')}$ (because all genes in V'_i belong to a different species). Notice that $s(r(T_i)) = r(S)$, since $r(S)$ separates v_1 from v_2 for all $v \in V$. Obtain T by taking any binary tree on k leaves (and hence $k - 1$ internal nodes), then replacing each leaf by a distinct T_i. In this manner, T has $k - 1$ duplications since only the internal nodes of T that do not belong to any T_i need to be duplications. Moreover, each duplication node x has $s(x) = r(S)$. It remains to show that T is compatible with $\mathcal{T}$. It suffices to observe that all triplets of $\mathcal{T}$ are of the form $v_1v_2|w_h$ with $h \in \{1, 2\}$, and that such a triplet being in $\mathcal{T}$ implies that $vw \in E$. For such a triplet, we must then have $v \in V_i$ and $w \in V_j$ with $i \neq j$, implying $v_1, v_2 \in V_i'$ and $w_h \in V_j'$. By the construction of T, $v_1v_2|w_h$ must be a triplet of T, as desired.

Claim 2 : if there is a tree T compatible with $\mathcal{T}$ having $k - 1$ duplications, then H is k-colorable. Moreover if T has $k - 1$ duplications such that each duplication x has $s(x) = r(S)$, then H is k-colorable.

Let T be a tree compatible with $\mathcal{T}$ having $k - 1$ duplications. Call a node x of T *S-maximal* if x is not a duplication node mapped to $r(S)$ but every proper ancestor of x is a duplication mapped to $r(S)$. Let $X = \{x_1, x_2, ..., x_m\}$ be the set of S-maximal nodes of T. Note that if $y \neq r(T)$ is a duplication mapped to $r(S)$, then so is the parent of y. This implies that every leaf ℓ of T has at least one ancestor x_i in X, since x_i is the highest (i.e. closest to the root) ancestor of ℓ that is not a duplication mapped to $r(S)$ (such an x_i always exists, since ℓ is itself one such node). Moreover, x_i is unique, as no other $x_j \in X$ can be the ancestor of x_i. Therefore, $\{\mathcal{L}(T_{x_1}), \cdots, \mathcal{L}(T_{x_m})\}$ is a partition of $\mathcal{L}(T)$. We next show that $m \leq k$. Let T' be the tree obtained by removing all descendants of x_i in T, for all $1 \leq i \leq m$. Then T' is a binary tree with m leaves, and all its $m - 1$ internal nodes are duplications mapped to $r(S)$. Since T has no more than $k - 1$ duplications (in either cases of the claim), T' has at most $k - 1$ internal nodes and therefore at most k leaves. We deduce that $m \leq k$.

Observe that if $vw \in E$, then $\alpha = lca(v_1, v_2, w_1, w_2)$ must be a duplication such that $s(\alpha) = r(S)$. Indeed, α separates $lca(v_1, v_2)$ from $lca(w_1, w_2)$ since T displays $((v_1, v_2), (w_1, w_2))$. But since $s(lca(v_1, v_2)) = s(lca(w_1, w_2)) = r(S)$ by the construction of S, $s(\alpha)$ can only be $r(S)$ as well, and so α must be a duplication.

Now, let $V_i = \{v : v_1$ is a descendant of $x_i\}$ for each $1 \leq i \leq m$. Take $v, w \in V_i$ for some i. We show that $vw \notin E$, and thus that $\{V_1, ..., V_m\}$ forms a coloring of H with at most k colors. The argument applies whether each duplication maps to $r(S)$ or not, proving both parts of the claim. Suppose for the sake of contradiction that $vw \in E$, but $v, w \in V_i$. In T, $lca(v_1, w_1)$ must be a descendant of x_i, since x_i is a common ancestor of v_1 and w_1 by the definition of V_i. Moreover, $lca(v_1, w_1) \neq x_i$ since $lca(v_1, w_1) = lca(v_1, v_2, w_1, w_2)$ is a duplication mapped to $r(S)$, as shown above, while x_i is not such a duplication, by its definition. Therefore, $lca(v_1, w_1)$ is a proper descendant of x_i. But $s(lca(v_1, w_1)) = r(S) = s(x_i)$ implies that x_i is a duplication mapped to $r(S)$, a contradiction. We conclude that $\{V_1, ..., V_m\}$ with $m \leq k$ is a proper coloring of H.

This reduction, together with the fact that the k-coloring problem is NP-hard to approximate within a $n^{1-\epsilon}$ factor, proves the Theorem. □

Independent Speciation trees

We now consider the MinDupSGT problem in the special case where the input gene trees are *independent speciation trees*, meaning: (1) each gene of Γ appears in at most one gene tree leafset, and (2) gene trees of $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ are all speciation trees with respect to the species tree S. Our objective is to find a gene tree T compatible with $\mathcal{T}$ minimizing duplications that also maintains the orthology relationships specified by $\mathcal{T}$. In other words, we require that for every $G_i \in \mathcal{G}$, $T|_{\mathcal{L}(G_i)}$ has only speciations. We say that a gene tree T that satisfies this property *preserves the speciations* of $\mathcal{T}$. Note that if T preserves the speciations of $\mathcal{T}$, then it is necessarily compatible with $\mathcal{T}$. We call $T|_{\mathcal{L}(G_i)}$ the *copy of* G_i in T.

MINIMUM SPECIATION SUPERGENETREE (MINSPECSGT PROBLEM):

Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,1 \leq i \leq k}$ of disjoint subsets of Γ, and a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of consistent *independent speciation trees* such that, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.

Output: Among all gene trees for Γ that preserve the speciations of $\mathcal{T}$, one tree T of minimum duplication cost.

Notice that, since no gene of Γ appears more than once in the set of input trees, $\mathcal{T}$ always admits a solution. Indeed, taking any binary tree on k leaves and replacing each leaf by a distinct G_i achieves the desired result. However, while apparently easier, we show that finding such a gene tree T minimizing the number of duplications is still hard.

Theorem 2 *The decision version of the MinSpecSGT problem is NP-Complete, i.e. it is NP-Complete to decide if a species tree S and a set of independent speciation trees G admit a supertree T that preserves its speciations with at most k duplications.*

Proof The problem is easily seen to be in NP, as it is easy to verify in polynomial time that a given gene tree T is compatible with $\mathcal{T}$, preserves its speciations and has k duplications. For NP-hardness, we turn to the decision version of the k-colorability problem. That is, for a given k, deciding if a graph $H = (V, E)$ is k-colorable is NP-hard. We create from H a species tree S and a set of independent speciation trees $\mathcal{T}$ such that H is k-colorable if and only if S and G admit a supertree T with at most $k - 1$ duplications.

Let $n = |V|$, and denote $V = \{v_1, ..., v_n\}$. To create S, start with any binary tree S' on $\binom{n}{2}$ leaves. denote this leafset $W = \{w_{i,j}: 1 \leq i < j \leq n\}$ so that there is a one-to-one correspondence between W and the unordered pairs of V. Then, add a special leaf a by joining it with the root of S' under a common parent p, and finally obtain S by adding another special leaf b by joining it with p under a common parent. Therefore, the species set is $\sum = \mathcal{L}(S) = W \cup \{a, b\}$.

For the construction of each gene tree $G \in \mathcal{G}$, we ease up notation by labeling each leaf g of G by $s(g)$ directly (e.g. if we say that G is of the form (a, b), we mean that $\mathcal{T}$ has two leaves g_a, g_b such that $s(g_a) = a$ and $s(g_b) =$

b). In this manner, since all trees of $\mathcal{T}$ contain only speciations, each tree $G \in \mathcal{G}$ must be a subtree of S (or it is obtained from such a subtree by contracting edges). Also recall that we are assuming that each gene appears in at most one gene tree of $\mathcal{T}$, and so the genes from two distinct trees must also be distinct (even if they share the same label).

In $\mathcal{T}$, we first add k trees of the form (a, b), plus one tree G_i for each vertex v_i in H. The tree G_i corresponding to $v_i \in V$ is a copy of S from which we remove every leaf except those $w_{j,k}$ for which one of $j = i$ or $k = i$ holds, and $(v_j\ v_{k)} \in E$ (i.e. we keep the leaves of W that correspond to an edge incident to v_i). Also contract the degree 2 nodes of G_i. Notice that if $v_iv_j \in E$ and $i < j$, then both G_i and G_j contain a gene in the $w_{i,j}$ species. Also, if $v_iv_j \notin E$ then G_i and G_j have no genes from a common species.

Claim 1 : if H is k-colorable, then S and $\mathcal{T}$ admit a supertree T having at most $k - 1$ duplications.

Let $\{V_1,..., V_k\}$ be a k-partition of V into independent sets. Take any h such that $1 \leq h \leq k$. Recall that if $v_i, v_j \in V_h$, then G_i and G_j share no gene from a common species (since $v_iv_j \notin E$). Thus the trees in $\mathcal{G}_h = \{G_i : v_i \in V_h\}$ are all disjoint in terms of species. Let Σ_h be the set of species that appear in some tree of $\mathcal{G}_h$. Then, the tree $S|_{\Sigma_h}$ contains a copy of each tree in $\mathcal{G}_h$, and none of these copies overlap. Obtain T_h by joining a gene labeled a to $r(S|_{\Sigma_h})$ under a common parent p, then joining a gene labeled b to p under a new common parent. Now, T_h contains a copy of each tree in G_h and a copy of one of the (a, b) trees. By taking a tree with k leaves (where at worst, each $k - 1$ internal node is a duplication), and replacing each leaf by the speciation trees $T_1,..., T_k$, we obtain a gene supertree T, which preserves the speciations of $\mathcal{T}$ and has at most $k - 1$ duplications.

Claim 2 : if S and $\mathcal{T}$ admit a supertree T having $k - 1$ duplications, then H is k-colorable.

We first show that if T has $k - 1$ duplications, then it must have exactly k speciations mapped to $r(S)$. It cannot have more, as there would then be more than $k - 1$ duplications. Suppose instead that there are $k' < k$ such speciations, and denote them $x_1,...,x_{k'}$. Note that there must be at least $k' - 1$ duplications in the ancestors of the x_is. Now, for $1 \leq i \leq k'$, T_{x_i} must contain a certain number of copies of a and b. Let $m_i(a)$ and $m_i(b)$ denote, respectively, the number of copies of a and b contained in T_{x_i}, noting that in total, there are k copies of each since there are k subtrees of the form (a, b) in $\mathcal{T}$. Since x_i is a speciation mapped to $r(S)$, it separates the a copies from the b copies, thus the T_{x_i} subtree must contain at least $m_i(a) - 1 + m_i(b) - 1$ duplications. Denote by $d(T)$ the number of duplications in T. It follows that $d(T) \geq k' - 1 + \sum_{i=1}^{k'} (m_i(a) + m_i(b) - 2) = k' - 1 - 2k' + \sum_{i=1}^{k'} m_i(a) + \sum_{i=1}^{k'} (m_i(b) = -k' - 1 + k + k = 2k - k' - 1 > k - 1$ when $k' < k$, a contradiction.

Now, we can let $x_1,..., x_k$ be the k speciation nodes of T mapped to $r(S)$. The $k - 1$ duplications of T must then all be ancestors of the x_i, and they are all mapped to $r(S)$. Therefore the $T_{x_1}, ..., T_{x_h}$ subtrees each contain only speciations. For any $G_i \in \mathcal{G}$ corresponding to v_i, one of the T_{x_h} must contain the copy of G_i (for otherwise, the root of the copy of G_i in T would be a duplication, while it should be a speciation). Take any h such that $1 \leq h \leq k$. We claim that $V_h = \{v_i : T_{x_h}$ contains the copy of $G_i\}$ forms an inpedendent set. Since T_{x_h} contains only speciations, it cannot contain genes from the same species. Thus for any G_i, G_j contained in T_{x_h} we must have $v_iv_j \notin E$, as otherwise G_i and G_j would share a gene from the same species. Therefore V_h is an independent set. Thus $\{V_1,..., V_k\}$ form a k-coloring of H, and the proof is completed. □

It is interesting to note that this does not show the NP-hardness of the special case in which the input trees are only triplets. Indeed, a tree G_i created in this reduction has as many leaves as the number of neighbors of its corresponding vertex v_i. Therefore, if H is a cubic graph (ie. 3-regular), one can generate an input with only triplets. However, deciding if a cubic graph is k-colorable can be done in linear time, and thus the triplets case cannot be shown NP-hard through this reduction. The 3-colorability problem is NP-hard on 4-regular graph though, showing the NP-hardness of the problem on input trees having at most 4 leaves.

Conclusion

We introduce the supergenetree problem which aims at constructing a supertree that displays a set of input gene trees while minimizing the reconciliation cost with respect to an input species tree. This problem is a natural formulation of the question of combining a set of gene trees obtained for subsets of a gene family into a full gene tree for the whole gene family.

The supergenetree problem is an extension of the classical supertree problem on a set of input leaf-labeled trees, where the input trees are gene trees and a species tree is used in order to evaluate the reconciliation/duplication cost of a supergenetree. We show how existing exact and greedy heuristic algorithms for the supertree problem can be used to devise approaches for solving the supergenetree problem. The resulting approaches have exponential worst-time complexity as the original supertree algorithms.

We show that the supergenetree problem for the duplication cost is NP-hard to approximate within a factor essentially better than n, and this complexity remains

the same even when the problem is restricted, in a greedy approach, to finding a supertree with a minimum number of duplications before each speciation of the species tree. We also consider a restriction of the supergenetree problem relevant to many biological applications where subsets of orthologs are studied separately and then amalgamated into a single tree. Even this restriction is shown to be NP- Complete. The reconciliation cost remains to be studied, although we conjecture all of the above mentioned problems are hard in this case also.

These negative complexity results are not surprising though as they extend an already large set of problems related to supertrees that are known to be NP-hard. We think that appropriate heuristics for various classes of input trees are worth to be considered in future projects. Removing the assumption that the input gene trees are compatible would also lead to new interesting problems. A promising avenue would be to consider constructive FPT algorithms that can be integrated in greedy heuristics or dynamic programming algorithms. Also other restrictions on the input gene trees can be explored, hopefully leading to polynomial problems. Constructing gene trees by amalgamating smaller trees for subsets of orthologous genes is a natural way of constructing large trees that would benefit from a thorough theoretical and algorithmic analysis.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
ML, AO, NE devised the proofs and algorithms and wrote the paper.

Declarations
Publication of this work is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds de Recherche Nature et Technologies of Quebec (FRQNT) and the Canada Research Chair (CRC) in Biological and Computational Biology.
This article has been published as part of *BMC Bioinformatics* Volume 16 Supplement 14, 2015: Proceedings of the 13th Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics: Bioinformatics. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcbioinformatics/supplements/16/S14.

Authors' details
[1]Département d'informatique et de recherche opérationnelle, Université de Montréal, Québec, Canada. [2]Département d'informatique, Université de Sherbrooke, Québec, Canada.

Published: 2 October 2015

References
1. Aho AV, Yehoshua S, Szymanski TG, Ullman JD: **Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions.** *SIAM J Comput* 1981, **10(3)**:405-421.
2. Bansal M, Burleigh J, Eulenstein O, Fernández-Baca D: **Robinson-foulds supertrees.** *Alg Mol Biol* 2010, **5(18)**.
3. Berglund AC, Sjolund E, Ostlund G, Sonnhammer EL: **InParanoid 6: eukaryotic ortholog clusters with inparalogs.** *Nucleic Acids Research* 2008, **36**:D263-D266.
4. Bininda-Emonds O, editor: **Phylogenetic Supertrees combining information to reveal The Tree Of Life.** *Computational Biology* Kluwer Academic, Dordrecht, the Netherlands; 2004.
5. Bininda-Emonds ORP, Gittleman J, Steel MA: **The super tree of life: Procedures, problems and prospects.** *Annu Rev Ecol Syst* 2002, **33**:265-289.
6. Bryant D: **A classification of consensus methods for phylogenetics.** *DIMACS series in Discrete Math and Theo Comput Sci* 2003.
7. Chauve C, El-Mabrouk N: **New perspectives on gene family evolution: losses in reconciliation and a link with supertrees.** *RECOMB of LNCS, Springer* 2009, **5541**:46-58.
8. Chen K, Durand D, Farach-Colton M: **Notung: Dating gene duplications using gene family trees.** *Journal of Computational Biology* 2000, **7**:429-447.
9. Constantinescu M, Sankoff D: **An efficient algorithm for supertrees.** *J Classif* 1995, **12**:101-112.
10. Cotton JA, Wilkinson M: **Majority-rule supertrees.** *Syst Biol* 2007, **56(3)**:445-452.
11. Datta RS, Meacham C, Samad B, Neyer C, Sjölander K: **Berkeley phog: Phylofacts orthology group prediction web server.** *Nucleic Acids Res* 2009, **37**:W84-W89.
12. Goemans XMichel, Williamson PDavid: **Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.** *Journal of the ACM (JACM)* 1995, **42(6)**:1115-1145.
13. Hallett Mike, Lagergren Jens, Tofigh Ali: **Simultaneous identification of duplications and lateral transfers.** *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology ACM* 2004, 347-356.
14. Huerta-Cepas J, Capella-Gutierrez S, Pryszcz LP, Denisov I, Kormes D, Marcet-Houben M, Gabald'on T: **Phylomedb v3.0: an expanding repository of genome-wide collections of trees, alignments and phylogeny-based orthology and paralogy predictions.** *Nucleic Acids Res* 2011, **39**:D556-D560.
15. Jansson J, Lemence RS, Lingas A: **The complexity of inferring a minimally resolved phylogenetic supertree.** *SIAM J on computing* 2012, **41(1)**:272-291.
16. Lafond M, Swenson KM, El-Mabrouk N: **An optimal reconciliation algorithm for gene trees with polytomies.** *LNCS, of WABI* 2012, **7534**:106-122.
17. Lechner M, Findeib SSven, Steiner L, Marz1 M, Stadler PF, Prohaska SJ: **Proteinortho: detection of (co-)orthologs in large-scale analysis.** *BMC Bioinformatics* 2011, **12**:124.
18. Li L, Stoeckert CJ Jr, Roos DS: **OrthoMCL: identification of ortholog groups for eukaryotic genomes.** *Genome Research* 2003, **13**:2178-2189.
19. Mi H, Muruganujan A, Thomas PD: **Panther in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees.** *Nucleic Acids Res* 2012, **41**:D377-D386.
20. Ng MP, Wormald NC: **Reconstruction of rooted trees from subtrees.** *Discrete Appl Math* 1996, **69**:19-31.
21. Nguyen N, Mirarab S, Warnow T: **MRL and SuperFine+MRL: new supertree methods.** *J Algo for Mol Biol* 2012, **7(3)**.
22. Penel Simon, Arigon Anne-Muriel, Dufayard Jean-François, Sertier Anne-Sophie, Daubin Vincent, Duret Laurent, Gouy Manolo, Perrière Guy: **Databases of homologous gene families for comparative genomics.** *BMC Bioinformatics* 2009, **10(Suppl 6)**:S3.
23. Pryszcz LP, Huerta-Cepas J, Gabaldón T: **MetaPhOrs: orthology nd paralogy predictions from multiple phylogenetic evidence using a consistency-based confidence score.** *Nucleic Acids Research* 2011, **39**:e32.
24. Ranwez V, Berry V, Criscuolo A, Fabre P, Guillemot S, Scornavacca C, Douzery E: **PhySIC: a veto supertree method with desirable properties.** *Syst Biol* 2007, **56(5)**:798-817.
25. Ranwez V, Criscuolo A, Douzery EJ: **SuperTriplets: a triplet-based supertree approach to phylogenomics.** *Bioinformatics* 2010, **26(12)**:i115-i123.
26. Scornavacca Celine, Jacox Edwin, Szöllősi JGergely: **Joint amalgamation of most parsimonious reconciled gene trees.** *Bioinformatics* 2014, btu728.
27. Semple C: **Reconstructing minimal rooted trees.** *Discrete Appl Math* 2003, **127(3)**.
28. Steel M, Rodrigo A: **Maximum likelihood supertrees.** *Syst Biol* 2008, **57(2)**:243-250.
29. Swenson MS, Suri R, Linder CR, Warnow T: **SuperFine: fast and accurate supertree estimation.** *Sys Biol* 2012, **61(2)**:214-227.
30. Vilella AJ, Severin J, Ureta-Vidal A, Heng L, Durbin R, Birney E: **EnsemblCompara gene trees: Complete, duplication-aware phylogenetic trees in vertebrates.** *Genome Research* 2009, **19**:327-335.

31. Zheng Y, Wu T, Zhang L: **A linear-time algorithm for reconciliation of non-binary gene tree and binary species tree.** *Combinatorial Optimization and Applications of LNCS* 2013, **8287**:190-201.
32. Zheng Yu, Zhang Louxin: **Reconciliation with non-binary gene trees revisited.** *Research in Computational Molecular Biology, Springer* 2014, 418-432.
33. Zuckerman David: **Linear degree extractors and the inapproximability of max clique and chromatic number.** *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing ACM* 2006, 681-690.

doi:10.1186/1471-2105-16-S14-S4
Cite this article as: Lafond *et al.*: Reconstructing a SuperGeneTree minimizing reconciliation. *BMC Bioinformatics* 2015 **16**(Suppl 14):S4.