Gene Tree Construction and Correction using SuperTree and Reconciliation

Manuel Lafond
Département d'Informatique (DIRO)
Universié de Montréal
Montreal, QC, Canada
Email: lafonman@iro.umontreal.ca

Cedric Chauve
Department of Mathematics
Simon Fraser University
Burnaby, BC, Canada
Email: cedric.chauve@sfu.ca

Nadia El-Mabrouk
Département d'Informatique (DIRO)
Universié de Montréal
Montreal, QC, Canada
Email: mabrouk@iro.umontreal.ca

Aïda Ouangraoua
Département d'Informatique
Université de Sherbrooke
Sherbrooke, QC, Canada
Email: aida.ouangraoua@usherbrooke.ca

***Abstract*—The supertree problem asking for a tree displaying a set of consistent input trees has been largely considered for the reconstruction of species trees. Here, we rather explore this framework for the sake of reconstructing a gene tree from a set of input gene trees on partial data. In this perspective, the phylogenetic tree for the species containing the genes of interest can be used to choose among the many possible compatible "supergenetrees", the most natural criteria being to minimize a reconciliation cost. We develop a variety of algorithmic solutions for the construction and correction of gene trees using the supertree framework. A dynamic programming supertree algorithm for constructing or correcting gene trees, exponential in the number of input trees, is first developed for the less constrained version of the problem. It is then adapted to gene trees with nodes labeled as duplication or speciation, the additional constraint being to preserve the orthology and paralogy relations between genes. Then, a quadratic time algorithm is developed for efficiently correcting an initial gene tree while preserving a set of "trusted" subtrees, as well as the relative phylogenetic distance between them, in both cases of labeled or unlabeled input trees. By applying these algorithms to the set of Ensembl gene trees, we show that this new correction framework is particularly useful to correct weakly-supported duplication nodes. The C++ source code for the algorithms and simulations described in the paper are available at** `https://github.com/UdeM-LBIT/SuGeT`**.**

1. Introduction

The supertree problem consists in combining a set of input phylogenetic trees on possibly overlapping sets of data, into a single one for the whole set (see for example [1], [2], [3], [4], [5], [6], [7]). Ideally, the obtained tree should display each of the input trees, which is only possible if they are "consistent" i.e. if they do not contain conflicting phylogenetic information. The simplest formulation of the supertree problem is therefore to state whether an input set of trees is consistent, and if so, find a "compatible" tree, called a *supertree*, displaying them all. This problem is NP-complete for unrooted trees [8], [9], but solvable in polynomial time for rooted trees [10], [11], [12], [13]. However, even for rooted trees the set of all possible supertrees may be exponential in the number of genes.

Supertree methods have been mainly designed to reconstruct a species tree from gene trees obtained for various gene families. However, they can have applications for gene tree reconstruction as well. Indeed, they may be used to combine partial trees on overlapping gene sets available from various sources (various databases, various reconstruction tools, *etc*). Alternatively, in the case of large gene families, they may be used to combine gene trees for smaller sets of orthologs, usually obtained from clustering algorithms such as OrthoMCL [14], InParanoid [15] or Proteinortho [16]. In such a case, ideally, orthology relations should be preserved in the final tree. More generally, given a set of input "labeled gene trees", i.e. gene trees with internal nodes labeled as duplication or speciation, we may be interested in a supertree preserving this labeling. As far as we know, no automated method accounting for labeling constraints has never been proposed. Here, we consider the problem of reconstructing a "supergenetree" in both cases of a labeled or unlabeled set of input gene trees.

In this paper, we also show that the supertree principle can be used for gene tree correction. For various reasons related to the considered model, method or data, gene trees can contain many errors (see for example [32] for a link with dubious high duplication nodes), and trees frequently exhibit branches with low statistical support. Two main approaches exist to correct gene trees, based on a local exploration principle to identify closely related trees that might have a better statistical support [**?**], a better reconciliation cost [17], [18], [23] or a combination of both [**?**], [**?**]. In the present work, we consider the second approach, based on the reconciliation cost with a given species tree. A way of correcting a gene tree is to remove weakly-supported branches, leading

to a set of subtrees, that should then be merged into a new one, according to some criterion. The most commonly considered criterion is a best fit with the species tree. A simple way is to consider the set of subtrees as the leaves of a polytomy (star-tree), and to resolve the polytomy in a way minimizing the reconciliation cost with the species tree (see NOTUNG [17], the Zheng and Zhang algorithm [18], PolytomySolver [19]). Such a correction method, not only preserves the input subtrees, but also the gene clusters inside the subtrees. In other words, the exhibited monophily of input gene clusters is not challenged by a polytomy resolution method. However, it has been shown that genes under negative selection, while exhibiting the true topology, may be wrongly grouped into monophyletic groups (see for example [20], [21], [22], [23]). In this perspective, using a supertree method may be beneficial, as it preserves the topology of subtrees while allowing to group genes from different subtrees.

In [24], we introduced under the name of MINIMUM SUPERGENETREE ($MinSGT$) the problem of finding, for a set of gene trees, a supertree that minimizes the reconciliation cost with a given species tree. Under the duplication cost, we have shown that this problem is NP-hard to approximate within a $n^{1-\epsilon}$ factor, for any $0 < \epsilon < 1$, even for instances in which there is only one gene per species in the input trees, and even if each gene appears in at most one input tree. In this paper, we carry out on $MinSGT$ but for the more general reconciliation cost. Although NP-hardness proofs for the duplication cost do not apply to the duplication plus loss cost, the problem is conjectured NP-hard for this more general reconciliation cost, as accounting for losses in addition to duplications is unlikely to make the problem simpler. Given a set of consistent input gene trees, we provide various algorithmic results depending on the additional information we have on the trees.

In Section 3, we first exhibit a dynamic programming algorithm for the general case, exponential in the number of input trees. We show how this algorithm can be adapted to compute a supertree preserving the input trees labeling, as motivated above. In Section 4, we then consider the correction problem with as input a gene tree together with a set of subtrees which topology should be preserved in the final supertree. To avoid having a supertree grouping genes that are far apart in the original tree, the relative phylogenetic distance between gene clusters is considered as an additional constraint. Inpired by the MINIMUM TRIPLET RESPECTING HISTORY introduced in [23], we define the MINIMUM TRIPLET RESPECTING SUPERGENETREE PROBLEM asking for a supertree displaying all input subtrees, while preserving the topology of any triplet of genes taken from three different subtrees. We develop a quadratic-time algorithm for this problem. Finally, in Section 5, by applying these algorithms to a set of a few hundreds Ensembl vertebrate gene trees, we show that this new correction framework is particularly useful to correct weakly-supported upper duplication nodes, as we observe that the correction carried out by our algorithms often improves significantly the likelihood scores.

2. Preliminaries

All considered trees are rooted and binary. We denote by $r(T)$ the root, by $V(T)$ the set of nodes, and by $\mathcal{L}(T) \subseteq V(T)$ the leafset of a tree T. We say that T is a tree for $L = \mathcal{L}(T)$. Given a node x of T, the subtree of T rooted at x is denoted $T[x]$. When there is no ambiguity on the considered tree, we simply write $\mathcal{L}(x)$ instead of $\mathcal{L}(T[x])$. We arbitrarily set one of the two children of an internal node x as the left child x_l and the other as the right child x_r, and denote by $(\mathcal{L}(x_l), \mathcal{L}(x_r))$ the bipartition induced by x. Also for the sake of simplicity, we just denote by T_l and T_r the left and right subtrees of the root of T. A node x is an *ancestor* of a node y if x is on the path between y and $r(T)$. If x is an ancestor of y, $inter(x, y)$ is the number of nodes located on the path between x and y, excluding x and y. Two nodes x and y are *separated* in T iff none is an ancestor of the other. In this case, we also say that the two subtrees $T[x]$, $T[y]$ of T are *separated*.

The *lowest common ancestor* (lca) of $L' \subset \mathcal{L}(T)$, denoted $lca_T(L')$, is the ancestor common to all leaves in L' that is the most distant from the root. $T|_{L'}$ 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 then all internal nodes of degree 2, except the root. 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 isomorphic to T' while preserving the same leaf-labeling.

Gene and species trees. A species tree S for a set Σ of species represents an ordered set of speciation events that have led to Σ. 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* G for Γ. For example, in Figure 1, G is a gene tree for $\Gamma = \{s_1, s_2, b_1, b_2, h_1, h_2, h_3, m_3, r_3\}$. Each internal node of G refers to an ancestral gene at the moment of an event, either speciation ($Spec$) or duplication (Dup). The mapping function s is generalized as follows: if x is an internal node of G, then $s(x) = lca_S(\{s(x') : x' \in \mathcal{L}(x)\})$.

When the type of event is known for each internal node, the gene tree G is said *labeled*. Formally, a *labeled gene tree* for Γ is a pair (G, ev_G), where G is a tree for $\mathcal{L}(G) = \Gamma$, and $ev_G : V(G) \setminus \mathcal{L}(G) \to \{Dup, Spec\}$ is a function labeling each internal node of G as a duplication or a speciation node.

According to the Fitch [25] terminology, given a labeled gene tree (G, ev_G), we say that two genes x, y are *orthologs* if $ev_G(lca_G(x, y)) = Spec$, and *paralogs* if $ev_G(lca_G(x, y)) = Dup$. For example, from the set of labeled gene trees in Figure 1, s_1, h_1 are orthologs while s_1, h_2 are paralogs.

While a history for Γ can be represented as a labeled gene tree, the converse is not always true, as a labeled tree (G, ev_G) for Γ does not necessarily represent a valid history in agreement with a species tree S. For this to hold,

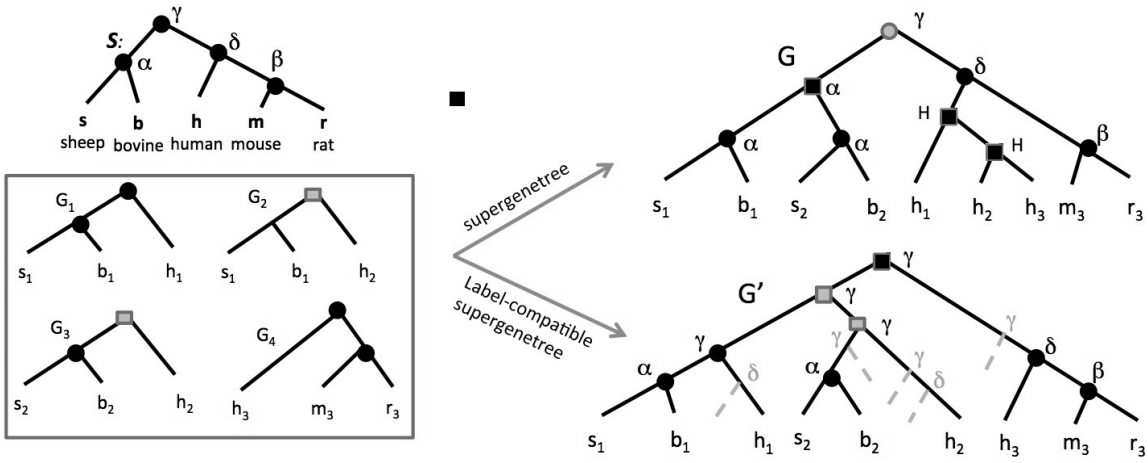

Figure 1. A species tree S on $\Sigma = \{s, b, h, m, r\}$, and a set of labeled gene trees $\mathcal{G} = \{G_1, G_2, G_3, G_4\}$, where each leaf x_i denotes a gene belonging to x. Square nodes are duplications and circular nodes are speciations. Internal nodes are labeled according to corresponding ancestral species in S. Dotted lines are losses. G is a supergenetree for $\mathcal{G}$ of minimum LCA-reconciliation cost (cost of 3), while G' is a label-compatible supergenetree for $\mathcal{G}$ of reconciliation cost 8 (3 duplications + 5 losses). G is not a label-compatible supergenetree due to the roots of G_2 and G_3 which are duplications in G_2 and G_3 (light gray squares), but are mapped to a speciation node in G (light gray circle). In G', these nodes are correctly mapped to duplication nodes (light gray squares in G').

(G, ev_G) should be *S-consistent*, i.e. any speciation node of (G, ev_G) should reflect the same clustering of species as in S (see [26] for a formal definition of S-consistency).

Reconciliation. The *LCA-reconciliation* of G with S is the labeled tree (G, ev_G) obtained by labeling each node x of G as $Spec$ if and only if $s(x_l)$ and $s(x_r)$ are separated in S, and as Dup otherwise. It follows that the LCA-reconciliation of G with S is an S-consistent tree. In Figure 1, G is labeled according to the LCA-reconciliation.

Given a labeled gene tree (G, ev_G), the *duplication cost* of (G, ev_G) is its number of duplication nodes. It reflects the number of duplications required to explain the evolution of the gene family inside the species tree S according to G. A well-known reconciliation approach [17], [27], [28] allows to further recover, in linear time, the minimum number of losses underlined by such an evolutionary history. We refer to the number of duplications and losses underlined by a labeled gene tree as its *reconciliation cost* in the general case, and as its *LCA-reconciliation cost* if the tree is labeled according to the LCA-reconciliation.

Supertree problems. Given a set $\mathcal{G}$ of trees for possibly overlapping subsets of Γ, the goal is to find a single tree displaying them all. This is possible only if the input trees are pairwise *consistent*. The consistency problem of rooted trees has been largely studied. For trees to be consistent, each triplet of data should exhibit the same topology in all trees. The BUILD algorithm [10] can be used to test, in polynomial-time, whether a collection of rooted trees is consistent, and if so, construct a compatible, not necessarily fully resolved, supertree. This algorithm has been generalized to output all compatible supertrees [11], [12], [13], which may be exponential in the number of genes.

3. Algorithms for Minimum SuperGeneTree Problems

We begin with the less constrained version of the problem. Given a set $\mathcal{G}$ of consistent input gene trees, we ask for a *compatible* tree, also called *supergenetree* G for $\mathcal{G}$, i.e. a tree displaying each tree of $\mathcal{G}$. In addition, among all supergenetrees for $\mathcal{G}$, G should be of minimum LCA-reconciliation cost (see G in Figure 1).

MINIMUM SUPERGENETREE ($MinSGT$) PROBLEM:
Input: A species set Σ and a species tree S for Σ; a gene family Γ of size n, a set $\Gamma_{i,1\leq i\leq k}$ of subsets of Γ such that $\bigcup_{i=1}^{k} \Gamma_i = \Gamma$, and a consistent set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of gene trees such that, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.
Output: Among all trees G for Γ compatible with $\mathcal{G}$, one of minimum LCA-reconciliation cost.

Suppose now that the input trees are labeled, and consider this labeling as an additional constraint. The problem becomes one of finding a labeled supergenetree preserving the input gene trees node labeling. As a labeled gene tree induces a full orthology and paralogy relation on the set of its leaves, this is possible only if the set of relations is *satisfiable*, i.e. if there is a labeled tree (G, ev_G) displaying the relations induced by all the input trees, and if there is such a tree which is S-consistent. Satisfiability is a well-studied problem. It reduces to verifying if a relation graph R (vertices are genes and edges link orthologous genes) is P_4-free, i.e. no four vertices of R induce a path of length 3 [29]. On the other hand, a cubic-time algorithm was developed in [26] for deciding whether a set of relations is S-consistent. Hereafter, we assume that the relations induced

by the input trees are satisfiable and S-consistent.

Let G and G' be two trees with $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ such that G displays G'. Then (G, ev_G) is said *label-compatible* with $(G', ev_{G'})$ iff, for any internal node x of G and x' of G' such that $x = lca_G(\mathcal{L}(x'))$, $ev_G(x) = ev_{G'}(x')$. A labeled supergenetree G for a set $\mathcal{G}$ of trees is said *label-compatible* with $\mathcal{G}$ iff it is label-compatible with each of the labeled trees of $\mathcal{G}$. An illustration is provided by the supergenetree G' in Figure 1. We are now ready to formulate our second problem.

MINIMUM LABELED SUPERGENETREE ($MinLSGT$) PROBLEM:
Input: A species set Σ and a species tree S for Σ; a gene family Γ of size n, a set $\Gamma_{i,1\leq i\leq k}$ of subsets of Γ such that $\bigcup_{i=1}^{k} \Gamma_i = \Gamma$, and a consistent set $\mathcal{G} = \{(G_1, ev_1), (G_2, ev_2), \cdots, (G_k, ev_k)\}$ of satisfiable and S-consistent labeled gene trees where, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.
Output: Among all labeled supergenetrees (G, ev_G) for Γ label-compatible with $\mathcal{G}$, one of minimum reconciliation cost.

The $MinSGT$ and $MinLSGT$ problems for the duplication cost were both shown NP-Hard in [24], even in the case where no two input trees have a gene in common and the trees only contain speciations.

3.1. The $MinSGT$ problem

We describe a dynamic programming algorithm for the $MinSGT$ problem leading to the following result.

Theorem 1. *The $MinSGT$ problem can be solved in $O((n+1)^k \times 4^k \times k)$ time complexity.*

The algorithm constructs the supergenetree G from the root to the leaves. At each step, i.e. for each internal node x being constructed in G, all possible bipartitions $(\mathcal{L}(x_l), \mathcal{L}(x_r))$ that could be induced by x are tried, and the iteration continues on each of $\mathcal{L}(x_l)$ and $\mathcal{L}(x_r)$. For example, at the root, the goal is to find the best bipartition of Γ, i.e. the one leading to the minimum LCA-reconciliation cost. At each step, this cost is computed from a *local reconciliation cost* at x (as defined in Lemma 1), and from the best reconciliation cost of the two created clusters. A key observation is that the constraint of being compatible with the input gene trees induces a strong constraint on the bipartitions, hence only a subset of the bipartition set has to be tested at each step.

First, a formulation of the reconciliation cost in terms of the sum of local reconciliation costs at each internal node x is given. The next lemma is a reformulation of the reconciliation cost, as described in many papers [17], [28].

Lemma 1. *The LCA-reconciliation cost of a gene tree G is the sum of* local LCA-reconciliation costs *$cost(L_l, L_r)$ for all internal nodes x of G, where $L = \mathcal{L}(x)$, and $(L_l, L_r) = (\mathcal{L}(x_l), \mathcal{L}(x_r))$, and $cost(L_l, L_r)$ equals to:*

- $inter(s(L), s(L_l)) + inter(s(L), s(L_r))$ *if $s(L) \neq s(L_l)$ and $s(L) \neq s(L_r)$;*
- $1 + inter(s(L), s(L_l)) + inter(s(L), s(L_r))$ *if $s(L) = s(L_l)$ and $s(L) = s(L_r)$;*
- $2 + inter(s(L), s(L_l)) + inter(s(L), s(L_r))$ *if $s(L) = s(L_l)$ and $s(L) \neq s(L_r)$ or conversely.*

The node $x = (L_l, L_r)$ is a speciation node in the first case, and a duplication node in the two last cases (thus adding 1 duplication to the LCA-reconciliation cost, plus 1 loss in the third case). Note that $inter(s, t) = 0$ if $s = t$.

For example, the root of G in Figure 1 fulfills the conditions of the first case, and thus it is a speciation node, whereas the root of G' fulfills the condition of the third case.

Lemma 1 allows to recursively compute a minimum LCA-reconciliation cost supergenetree, by exploring, for each node x from the root to the leaves, all "valid" bipartitions of $\mathcal{L}(x)$, remaining to be characterized formally. In the following, we define the properties of a bipartition (L_l, L_r) induced by the root of a supergenetree G. It directly follows from the definition of a supergenetree that should display each individual gene tree.

Property 1. *Let $\mathcal{G} = \{G_1, \ldots, G_k\}$ be a set of gene trees. The root of a supergenetree G compatible with $\mathcal{G}$ subdivides $\bigcup_{i=1}^{k} \mathcal{L}(G_i)$ into a* compatible bipartition *(L_l, L_r), i.e. a bipartition such that, for each i s.t. $1 \leq i \leq k$, either: 1) $\mathcal{L}(G_i) \subseteq L_l$; or 2) $\mathcal{L}(G_i) \subseteq L_r$; or 3) $\mathcal{L}(G_{i_l}) \subseteq L_l$ and $\mathcal{L}(G_{i_r}) \subseteq L_r$; or 4) $\mathcal{L}(G_{i_l}) \subseteq L_r$ and $\mathcal{L}(G_{i_r}) \subseteq L_l$.*

For example, the root of the supergenetree G in Figure 1 satisfies the third condition for G_1, G_2 and G_3, and the second for G_4.

$\mathcal{B}(G_1, \ldots, G_k)$ denotes the set of all bipartitions of $\bigcup_{i=1}^{k} \mathcal{L}(G_i)$ compatible with $\mathcal{G}$. For example, the two bipartitions defined by the roots of G and G' in Figure 1 are both compatible with the given set of gene trees. Figure 2 illustrates the set of all valid bipartitions compatible with two given trees.

Lemma 2. $|\mathcal{B}(G_1, \ldots, G_k)| \leq (\frac{4^k}{2}) - 1$.

Proof. For each tree G_i, there are four possibilities for placing $\mathcal{L}(G_{i_r})$ and $\mathcal{L}(G_{i_l})$ in a bipartition (L_l, L_r): either they are both in L_l, or both in L_r, or one in L_l and the other in L_r. Therefore, 4^k distributions of left and right subtrees of the k trees in (L_l, L_r). However, as the left and right characterization of nodes is arbitrary, each distribution is counted twice, and thus the total number of different bipartitions is $\frac{4^k}{2}$. One of these bipartitions has a part that is empty. We discard it and the total number is then $\frac{4^k}{2} - 1$. However, a set (L_l, L_r) obtained from such distribution of the G_i subtrees is not necessarily a bipartition, as a same gene can be present in two different input trees, and end up placed in both L_l and L_r. Therefore, $(\frac{4^k}{2}) - 1$ is only an upper bound of the number of compatible bipartitions. □

The constructive proof of Lemma 2 induces an algorithm for enumerating the members of $\mathcal{B}(G_1, \ldots, G_k)$, which is

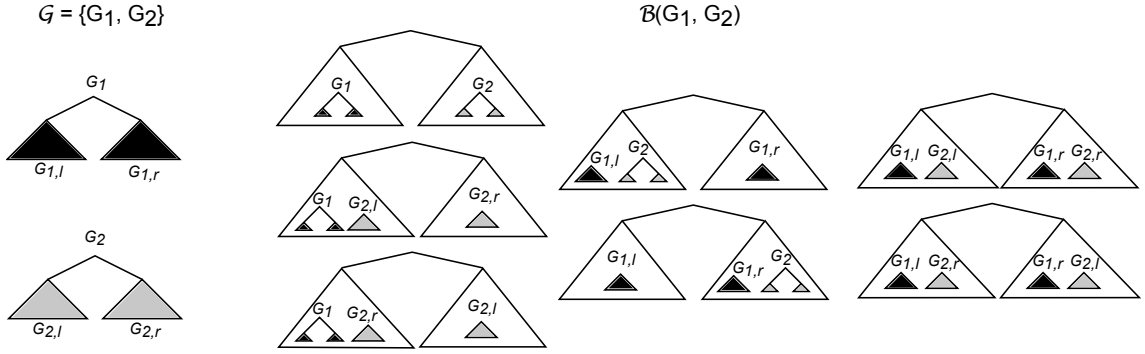

Figure 2. An illustration of the seven valid bipartitions for two trees G_1 and G_2. Each bipartition is obtained by "sending" $L_1 \in \{\mathcal{L}(G_1), \mathcal{L}(G_{1,l}), \mathcal{L}(G_{1,r}), \emptyset\}$ in the left part, and the complement $\mathcal{L}(G_1) \setminus L_1$ in the right part. The same process is then applied to G_2. The set $\mathcal{B}(G_1, G_2)$ consists in the set of all possible combinations of choices, after eliminating symmetric cases and partitions with an empty side. For each bipartition (L_l, L_r) of $\mathcal{B}(G_1, G_2)$, an optimal solution is recursively computed for L_l and L_r, the reconciliation cost is computed for the tree obtained by joining the roots of the two trees under a common parent, and the tree yielding a minimum cost among all possibilities is returned.

illustrated in Figure 2 for the case of two trees. Intuitively, to construct a bipartition (L_l, L_r), each tree G_i of $\mathcal{G}$ can choose to "send" in L_l either its left subtree G_{i_l}, its right subtree G_{i_r}, the whole tree G_i or nothing at all. What has not been sent in L_l is sent in L_r. Then $\mathcal{B}(G_1, \ldots, G_k)$ is the set of all possible combinations of choices. However, not every bipartition constructed in this manner yields a valid bipartition. For instance in Figure 2, the top-left bipartition cannot be valid if G_1 and G_2 share a leaf with the same label, as a gene cannot be sent both left and right. These cases, however, can be detected easily by verifying the sizes of L_l and L_r.

We are now ready to give the main recurrence formula of our dynamic programming algorithm. Denote by $MinSGT(G_1, \ldots, G_k)$ the minimum LCA-reconciliation cost of a supergenetree compatible with $\mathcal{G} = \{G_1, \ldots, G_k\}$. The next lemma directly follows from Lemma 1 and Property 1.

Lemma 3. *Let $\mathcal{G} = \{G_1, \ldots, G_k\}$ be a set of gene trees.*

1) $MinSGT(G_1, \ldots, G_k) = 0$ *if* $\mid \bigcup_{i=1}^{k} \mathcal{L}(G_i) \mid = 1$ *(Stop condition);*
2) *Otherwise,* $MinSGT(G_1, \ldots, G_k) =$

$$\min_{(L_l, L_r) \in \mathcal{B}(G_1, \ldots, G_k)} \left\{ \begin{array}{l} cost(L_l, L_r) + \\ MinSGT(G_{1|L_l}, \ldots, G_{k|L_l}) + \\ MinSGT(G_{1|L_r}, \ldots, G_{k|L_r}) \end{array} \right\}$$

Note that, given a bipartition $(L_l, L_r) \in \mathcal{B}(G_1, \ldots, G_k)$, for each i such that $1 \leq i \leq k$, $G_{i|L_l}$ and $G_{i|L_r}$ are equal either to $\emptyset$ or G_i or G_{i_l} or G_{i_r}. Thus, $G_{i|L_l}$ and $G_{i|L_r}$ are either empty trees or complete subtrees of G_i.

Note also that, at each step, the existence of a compatible bipartition follows from the fact that the input gene trees are assumed to be consistent, as stated in the formulation of the $MinSGT$ problem. In the absence of this assumption, we have to add a third equation to Lemma 3: If $\mid \bigcup_{i=1}^{k} \mathcal{L}(G_i) \mid > 1$ and $\mid \mathcal{B}(G_1, \ldots, G_k) \mid = 0$, $MinSGT(G_1, \ldots, G_k) = +\infty$.

Complexity. We now address the complexity of the dynamic programming algorithm defined by the recurrences of Lemma 3. Each call to the recursive procedure $MinSGT$ receives as input at most one subtree from each tree G_i. Let n be the maximum number of node in a tree G_i. As each tree has at most n possible subtrees, there are at most $(n+1)^k$ possible calls to $MinSGT$. Next, for any set of gene trees $\{G_1, \ldots, G_k\}$, the number of distinct bipartitions $(L_l, L_r) \in \mathcal{B}(G_1, \ldots, G_k)$ to be tested is at most $\frac{4^k}{2} - 1$ (Lemma 2). Finally, the value of $cost(L_l, L_r)$ can be computed in time $O(k)$ provided that the mapping s is precomputed for all nodes of the trees $G_1, \ldots, G_k$, and $lca(x, y)$ and $inter(x, y)$ are precomputed for any pair (x, y) of nodes in S. The time complexity of the overall algorithm is therefore $O((n+1)^k \times 4^k \times k)$, which completes the proof of Theorem 1.

3.2. The $MinLSGT$ Problem

The algorithm for the $MinSGT$ problem can be adapted to solve the $MinLSGT$ problem, leading to the following result.

Corollary 1. *The $MinLSGT$ problem can be solved in $O((n+1)^k \times 4^k \times k)$ time complexity.*

The intuition behind the $MinLSGT$ algorithm is quite simple. We proceed as in the $MinSGT$ algorithm, but each time a bipartition (L_l, L_r) is considered, we verify whether the root of a tree separating L_l and L_r should be a speciation or a duplication. If there are two genes $g_l \in L_l$ and $g_r \in L_r$ that disagree with this event, we treat (L_l, L_r) as an invalid bipartition and do not consider it further.

Before describing this adaptation, we need few additional definitions and properties. Given a set of labeled gene trees $\mathcal{G} = \{(G_1, ev_{G_1}), \ldots, (G_k, ev_{G_k})\}$ and a bipartition $(L_l, L_r) \in \mathcal{B}(G_1, \ldots, G_k)$, for any i s.t. $1 \leq i \leq k$, we say that G_i is *separated* by (L_l, L_r) iff G_i satisfies the third or fourth condition of Property 1. We denote by $\mathcal{G}(L_l, L_r)$

the set of gene trees G_i, $1 \leq i \leq k$, that are separated by (L_l, L_r).

Lemma 4. *Let $\mathcal{G} = \{(G_1, ev_{G_1}), \ldots, (G_k, ev_{G_k})\}$ be a set of labeled gene trees. Then, for any labeled supergenetree (G, ev_G) label-compatible with $\mathcal{G}$, the label $ev_G(x)$ of its root x equals the label of the root of any gene tree G_i, $1 \leq i \leq k$, such that $G_i \in \mathcal{G}(\mathcal{L}(x_l), \mathcal{L}(x_r))$.*

Proof. Let G_i, $1 \leq i \leq k$ be a genetree such that $G_i \in \mathcal{G}(\mathcal{L}(x_l), \mathcal{L}(x_r))$, and let x_i be the root of G_i. Then $lca_G(\mathcal{L}(x_i)) = x$, and thus by definition of the label-compatibility of G with G_i, we have $ev_G(x) = ev_{G_i}(x_i)$. □

From Lemma 4, we define a bipartition of $\bigcup_{i=1}^{k} \mathcal{L}(G_i)$ label-compatible with $\mathcal{G}$ as follows.

Definition 1. *Let $\mathcal{G} = \{(G_1, ev_{G_1}), \ldots, (G_k, ev_{G_k})\}$ be a set of labeled gene trees. A bipartition (L_l, L_r) of $\bigcup_{i=1}^{k} \mathcal{L}(G_i)$ is* label-compatible *with $\mathcal{G}$ if it is compatible with $\mathcal{G}$ and verifies:*

1) *if $\mid \mathcal{G}(L_l, L_r) \mid > 0$, the roots of all gene trees in $\mathcal{G}(L_l, L_r)$ have the same label denoted by $ev_{\mathcal{G}(L_l, L_r)}$.*
2) *if $\mid \mathcal{G}(L_l, L_r) \mid > 0$ and $ev_{\mathcal{G}(L_l, L_r)} = Spec$, then $lca_S(\{s(x) : x \in L_l \cup L_r\}) \neq lca_S(\{s(x) : x \in L_l\})$ and $lca_S(\{s(x) : x \in L_l \cup L_r\}) \neq lca_S(\{s(x) : x \in L_r\})$.*

For example, the bipartition determined by the root of the supergenetree G $(\{s_1, s_2, b_1, b_2\}, \{h_1, h_2, h_3, m_3, r_3\})$ in Figure 1 is not label-compatible with the set of gene trees, as it separates both G_1 and G_2 which do not have the same root label.

The $MinLSGT$ algorithm for solving the $MinSGT$ problem is based on the same general dynamic programming framework as the $MinSGT$ algorithm: at each step, iterate over all possible bipartitions, and then proceed recursively for each partition. The two differences are: (1) given a set of labeled gene trees $\mathcal{G} = \{(G_1, ev_1), \ldots, (G_k, ev_k)\}$, we only test a subset of compatible bipartitions of $\mathcal{B}(G_1, \ldots, G_k)$ that are label-compatible with $\mathcal{G}$; (2) computing local reconciliation costs should not be done on the basis of the LCA-reconciliation, as some nodes that would be labeled as speciation nodes from the LCA-mapping should rather be duplication nodes in order to be label-compatible with some input gene trees. For example, in Figure 1, $lca_{G'}(\{s_2, b_2, h_2\})$ would be labeled $Spec$ by the LCA-mapping. However, it should be labeled Dup to be label-compatible with G_3. The following Lemma is required, in place of Lemma 1.

Lemma 5. *Let x be an internal node of a labeled supergenetree (G, ev_G), $L = \mathcal{L}(x)$ and $(L_l, L_r) = (\mathcal{L}(x_l), \mathcal{L}(x_r))$. The local reconciliation cost of x, $cost_{\mathcal{G}}(L_l, L_r)$ is equal to:*

- $3 + inter(s(L), s(L_l)) + inter(s(L), s(L_r))$ *if $s(L) \neq s(L_l)$, $s(L) \neq s(L_r)$, $\mid \mathcal{G}(L_l, L_r) \mid > 0$ and $ev_{\mathcal{G}}(L_l, L_r) = Dup$;*
- $cost(L_l, L_r)$ *Otherwise.*

In the first case, the node x is a duplication node adding 1 duplication plus at least 2 losses to the reconciliation cost, and in the second case the local reconciliation cost is computed as for the LCA-reconciliation.

The complexity of the $MinLSGT$ algorithm remains in $O((n+1)^k \times 4^k \times k)$ provided that the sets of label-compatible bipartitions (L_l, L_r) are constructed simultaneously with the sets $\mathcal{G}(L_l, L_r)$.

3.3. Improved complexity from a core set of trees

Last, we show that the principles underlying the two algorithms described above can be improved to reduce the dependency in k. The key remark is that all bipartitions to consider can be identified by considering only a subset of the input trees provided they span the set of all genes of Γ.

Call $\mathcal{G}' \subseteq \mathcal{G}$ a *core* of $\mathcal{G}$ if $\bigcup_{G \in \mathcal{G}'} \mathcal{L}(G) = \Gamma$. We introduce the following modified $MinSGT$ algorithm, that we call $MinSGT\text{-}core$:

1) Find a core $\mathcal{G}' = \{G'_1, \ldots, G'_\ell\}$ of $\mathcal{G}$.
2) Apply the $MinSGT$ algorithm on $\mathcal{G}'$, with the exception that, when considering a bipartition $B = (L_l, L_r)$ compatible with $\mathcal{G}'$:
   • Verify that B is also compatible with $\mathcal{G} \setminus \mathcal{G}'$. If not, then do not proceed recursively on B;
   • Compute $cost(L_l, L_r)$ on the whole set $\mathcal{G}$.

Theorem 2. *Let $\mathcal{G}'$ be a core of $\mathcal{G}$ composed of k' trees. The $MinSGT$ problem can be solved in $O((n+1)^{k'} \times 4^{k'} \times k)$ time complexity.*

Proof. The difference between the executions of a call of the $MinSGT\text{-}core$ algorithm on the input $\mathcal{G}'$ and a call of the $MinSGT$ on $\mathcal{G}$ lies in the set of bipartitions considered at each step of the recursion. At a given step of the recursion, let $\mathcal{B}'$ and $\mathcal{B}$ be the set of bipartitions compatible with $\mathcal{G}'$ and $\mathcal{G}$ respectively, and let $\mathcal{B}^*$ be the set of bipartitions considered by $MinSGT\text{-}core$. We show that $\mathcal{B} = \mathcal{B}^*$.

Clearly $\mathcal{B} \subseteq \mathcal{B}'$, as a bipartition compatible with $\mathcal{G}$ is also compatible with $\mathcal{G}' \subseteq \mathcal{G}$. Suppose that there is some $(L_l, L_r) \in \mathcal{B}$ such that $(L_l, L_r) \notin \mathcal{B}^*$. This implies that (L_l, L_r) was filtered out of $\mathcal{B}'$, meaning that it is not compatible with some tree $G \in \mathcal{G} \setminus \mathcal{G}'$. Therefore (L_l, L_r) cannot be in $\mathcal{B}$, a contradiction. We deduce that $\mathcal{B} \subseteq \mathcal{B}^*$. To see that $\mathcal{B}^* \subseteq \mathcal{B}$, observe that $\mathcal{B}^*$ contains only bipartitions compatible with $\mathcal{G}' \cup (\mathcal{G} \setminus \mathcal{G}') = \mathcal{G}$, and that $\mathcal{B}$ contains every such bipartition. So both algorithms consider the same set of bipartitions at each step, and lead to the same solution. □

It thus remain to describe how to find a core $\mathcal{G}'$, as small as possible, as the size of the core is now the main complexity parameter. This problem is equivalent to the MINIMUM SET COVER PROBLEM known to be NP-hard. However, a natural heuristic is the following: choose a gene tree G_i with the largest subset of Γ as leafset, say of size $n-p$, and "complete" it with at most p additional gene trees

from $\mathcal{G}$ each containg at least one of the missing genes. This obviously provide a core, leading to the following result.

Corollary 2. *The $MinSGT$ problem can be solved in $O((n+1)^{p+1} \times 4^{p+1} \times k)$ time complexity, where p is the smallest integer such that a gene tree of $\mathcal{G}$ contains $n-p$ genes.*

The same technique applies to $MinLSGT$ and the same result could be stated for this problem.

4. Triplet Respecting Supergenetrees

We now consider a problem related to the correction of a gene tree. Assume that input gene trees $G_1, G_2, \cdots, G_k$ are separated subtrees (i.e. leaf-disjoint) of a given gene tree G^{Init}. The $MinSGT$ and $MinLSGT$ problems can also be considered in this context to infer an alternative gene tree displaying them all and minimizing a reconciliation cost. However, this may lead to a new tree exhibiting a complete reorganization of the input subtrees and possibly grouping genes that were far apart in the initial tree. Therefore, assume in addition that we trust the hierarchy of upper branches. Then we ask for a supergenetree of minimum reconciliation cost which preserves the phylogenetic relation between subtrees, as given by G^{Init}. Formally, we seek for a triplet respecting supergenetree, as defined bellow.

Definition 2. *Let $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ be a set of separated subtrees of a gene tree G^{Init} for Γ such that $\bigcup_{i=1}^{k} \mathcal{L}(G_i) = \Gamma$. A tree G^{TR} compatible with $\mathcal{G}$ is* triplet respecting *iff, for any triplet of trees $\mathcal{L}(G_{i_1})$, $\mathcal{L}(G_{i_2})$ and $\mathcal{L}(G_{i_3})$ in $\mathcal{G}$ and any triplet of genes $x \in G_{i_1}$, $y \in G_{i_2}$ and $z \in G_{i_3}$, G^{Init} and G^{TR} display the same topology for the triplet (x, y, z), i.e. $G^{Init}|_{\{x,y,z\}} = G^{TR}|_{\{x,y,z\}}$.*

For example in Figure 3, the supergenetree G is not triplet respecting as for the triplet of genes (h_1, m_1, m_2), G does not display the same topology as the tree G^{Init}.

MINIMUM TRIPLET RESPECTING SUPERGENETREE ($MinTRS$) PROBLEM:
Input: A species set Σ and a species tree S for Σ; a gene family Γ and a gene tree G^{Init} for Γ; a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of separated subtrees of G^{Init} such that $\bigcup_{i=1}^{k} \mathcal{L}(G_i) = \Gamma$.
Output: Among all triplet respecting gene trees for Γ compatible with $\mathcal{G}$, one of minimum LCA-reconciliation cost.

A natural extension of the $MinTRS$ Problem is the MINIMUM LABELED TRIPLET RESPECTING SUPERGENETREE ($MinLTRS$) Problem, where we are given a set of labeled separated subtrees of G^{Init} and we seek for a labeled triplet respecting supergenetree of minimum reconciliation cost. Here we focus on $MinTRS$, though all results extend naturally to $MinLTRS$, as briefly explained at the end of this section. Note that the $MinTRS$ and $MinLTRS$ problems can be reduced to the $MinSGT$ and $MinLSGT$ problems by considering as input of $MinSGT$ and $MinLSGT$ the set of subtrees $\mathcal{G}$ of G^{Init} augmented with the set of all rooted triplet trees that should be respected by the output supergenetree. However, the algorithms for $MinTRS$ and $MinLTRS$ problems induced by these reductions would remain exponential in the number of input subtrees.

We describe a more efficient recursive algorithm that solves the $MinTRS$ problem by making use of the $MinSGT$ solution. This algorithm leads to the following result.

Theorem 3. *The $MinTRS$ and $MinLTRS$ problems can be solved in $O(n^2)$ time complexity.*

The high-level description of the algorithm is as follows. The triplet-respecting property only allows a limited number of ways to combine the subtrees of $\mathcal{G}$ together. We distinguish two possible cases. First, if two subtrees G_1, G_2 of $\mathcal{G}$ form a "cherry" in G^{Init}, meaning that $r(G_1)$ and $r(G_2)$ share the same parent in G^{Init}, then G_1 and G_2 can be mixed together in any way without contradicting the triplet-respecting property. The optimal way of mixing the two trees is to compute $MinSGT(G_1, G_2)$, which gives a solution for the subtree of G^{Init} rooted at the parent of $r(G_1)$ and $r(G_2)$. For instance in Figure 3, the two children of the β node in G^{Init} form a cherry of subtrees. Second, if instead a subtree G_1 of $\mathcal{G}$ is not part of such a cherry, then let x be the sibling of $r(G_1)$ in G^{Init}. Then we show that the following procedure can be performed: recursively compute G_x, an optimal solution for the subtree of G^{Init} rooted at x, then try grafting G_x on G_1 in every possible way and keep the solution that minimizes the reconciliation cost. This gives a solution for the subtree of G^{Init} rooted at the parent of $r(G_1)$ and x. These two cases describe all the possible subtree mixings that can occur, and the rest of the G^{Init} topology must be conserved. For example in Figure 3, from a bottom-up point of view, the algorithm would compute $G_{3,4} = MinSGT(G_3, G_4)$, then obtain $G_{2,3,4}$ by finding the best place on which to graft $G_{3,4}$ on G_2 (in this case, above the parent of m_1 and r_1), then obtain a solution by grafting $G_{2,3,4}$ somewhere on G_1 (in this case above h_1). In the following we rather describe the algorithm in a top-down manner, i.e. we start at the root of G^{Init}, obtain a solution recursively for its two child subtrees and combine them appropriately.

Before describing the algorithm in full detail, we give a few additional definitions and properties. Let G and G' be two gene trees for Γ. Define $s_{G' \to G}$ as the mapping from the nodes of G' to the nodes of G such that $s_{G' \to G}(x) = lca_G(\mathcal{L}(x))$. For example in Figure 3, the image of the lowest δ node of G^{Init} by $s_{G^{Init} \to G}$ is the highest δ node of G, since it is the lca of m_1, r_1, h_2.

The algorithm for $MinTRS$ constructs the triplet respecting supergenetree G^{TR} by building recursively and independently the bipartitions $(\mathcal{L}(y_l), \mathcal{L}(y_r))$ induced by each internal node y of G^{TR} from the root to the leaves. The nodes of G^{TR} can be considered independently in the algorithm because the constraint of being triplet respecting strongly predetermines the set of leaves $\mathcal{L}(y)$ associated to some nodes y of G^{TR} as shown in Lemma 6.

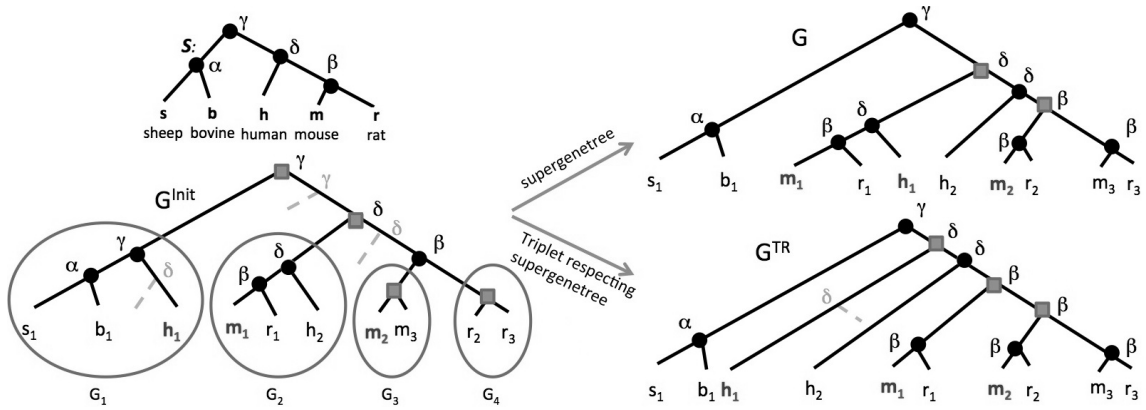

Figure 3. A species tree S on the set of species $\Sigma = \{s, b, h, m, r\}$, and a gene tree G^{Init} with a set of separated subtrees $\mathcal{G} = \{G_1, G_2, \cdots, G_4\}$. The name of genes and the form of internal nodes and lines follow the same rules as in Figure 1. G is a supergenetree for $\mathcal{G}$ of minimum LCA-reconciliation cost (cost of 2) and G^{TR} is a triplet respecting supergenetree for $\mathcal{G}$ of LCA-reconciliation cost 4 (3 duplications + 1 loss). G is not a triplet respecting supergenetree because for example, for the triplet of genes (h_1, m_1, m_2), G^{Init} displays the topology $(h_1, m_1 m_2)$ while G displays the topology $(h_1 m_1, m_2)$. In G^{TR} however, all triplet genes topologies are respected.

Given a node x of G^{Init}, we denote by $\mathcal{G}(x)$ the subset of $\mathcal{G}$ that are subtrees of $G^{Init}[x]$. If there exists a node y in G^{TR} such that $\mathcal{L}(y) = \mathcal{L}(x)$, then we also define $\mathcal{G}(y) = \mathcal{G}(x)$. For example, call x the highest δ node of G^{Init} in Figure 3. Then $\mathcal{G}(x) = \{G_2, G_3, G_4\}$. Now, for y being the lowest (non-loss) δ node of G^{TR}, $\mathcal{L}(y) = \mathcal{L}(x)$ and so $\mathcal{G}(y) = \{G_2, G_3, G_4\}$.

Lemma 6. *Let G^{TR} be a triplet respecting supergenetree for G^{Init} and $\mathcal{G} = \{G_1, \ldots, G_k\}$. For any node x of G^{Init} such that $\mid \mathcal{G}(x) \mid \geq 2$, there exists a node y of G^{TR} such that $\mathcal{L}(y) = \mathcal{L}(x)$.*

Proof. Let x be a node of G^{Init} such that $\mid \mathcal{G}(x) \mid \geq 2$. Each of the subtrees $G^{Init}[x_l]$ and $G^{Init}[x_r]$ then contains at least one tree of $\mathcal{G}$. Assume that (*) there exists no node y in G^{TR} such that $\mathcal{L}(y) = \mathcal{L}(x)$. Let x' be the node of G^{TR} such that $s_{G^{Init} \to G^{TR}}(x) = x'$, and let x'' be the node of G^{Init} such that $s_{G^{TR} \to G^{Init}}(x') = x''$. The assumption (*) implies that $x'' \neq x$, so x'' is a strict ancestor of x. Suppose w.l.o.g. that x belongs to the subtree $G^{Init}[x''_l]$ and pick any gene $c \in \mathcal{L}(x') \cap \mathcal{L}(x''_r)$. There exists a tree G_h of $\mathcal{G}$ such that $c \in \mathcal{L}(G_h)$ and G_h is contained in $G^{Init}[x''_r]$. Now, let a and b be two genes such that $a \in \mathcal{L}(x_l) \cap \mathcal{L}(x'_l)$ and $b \in \mathcal{L}(x_r) \cap \mathcal{L}(x'_r)$, or $a \in \mathcal{L}(x_l) \cap \mathcal{L}(x'_r)$ and $b \in \mathcal{L}(x_r) \cap \mathcal{L}(x'_l)$. Such two genes necessarily exist because $s_{G^{Init} \to G^{TR}}(x) = x'$. So, there exist two trees G_i and G_j of $\mathcal{G}$ such that $(a, b) \in \mathcal{L}(G_i) \times \mathcal{L}(G_j)$, G_i is contained in $G^{Init}[x_l]$ and G_j is contained in $G^{Init}[x_r]$. So G^{Init} displays the topology (ab, c) for the triplet of genes (a, b, c) while G^{TR} displays a different topology, either (a, bc) or (ac, b). The assumption (*) is then impossible. □

We denote by $V_{cons}(G^{TR})$ the subset of nodes y of G^{TR} such that there exists a node x of G^{Init} satisfying $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mid \mathcal{G}(y) \mid \geq 2$. For example, in Figure 3, $V_{cons}(G^{TR})$ contains three nodes, the root, the lowest δ node and the lowest duplication node of G^{TR}. Lemma 6 allows to predetermine the sets of leaves $\mathcal{L}(y)$ associated to the nodes $y \in V_{cons}(G^{TR})$. We now describe how to find the best subtree $G^{TR}[y]$ for each node $y \in V_{cons}(G^{TR})$, i.e. one leading to the minimum reconciliation cost.

Note that if $y \in V_{cons}(G^{TR})$ and $\mid \mathcal{G}(y) \mid = 2$, say $\mathcal{G}(y) = \{G_i, G_j\}$, $1 \leq i < j \leq k$, then the $MinSGT$ algorithm can be applied to build the subtree $G^{TR}[y]$ as a minimum reconciliation cost supergenetree for G_i and G_j. It then remains to describe a recursive procedure for finding the subtree $G^{TR}[y]$ for a node $y \in V_{cons}(G^{TR})$ such that $\mid \mathcal{G}(y) \mid > 2$.

In order to compute the reconciliation cost of the tree G^{TR}, we need to account for the local reconciliation costs for the nodes $y \in V_{cons}(G^{TR})$, and also for the internal nodes z of G^{TR} such that $z \notin V_{cons}(G^{TR})$. To do so, given a node $y \in V_{cons}(G^{TR})$, we define $cost_{TR}(y)$ as the local reconciliation cost for y, plus the local reconciliation costs for all internal nodes $z \in V(G^{TR})$ such that $z \notin V_{cons}(G^{TR})$, y is an ancestor of z and there exists no node $y' \in V_{cons}(G^{TR})$ on the path between y and z. For example in Figure 3, call y the root of G^{TR}. Then, $y \in V_{cons}(G^{TR})$ and $cost_{TR}(y) = cost(\mathcal{L}(y_l), \mathcal{L}(y_r)) + cost(\mathcal{L}(y_{l_l}), \mathcal{L}(y_{l_r})) + cost(\mathcal{L}(y_{r_l}), \mathcal{L}(y_{r_r}))$ counting the local reconciliation costs for y, $y_l \notin V_{cons}(G^{TR})$ and $y_r \notin V_{cons}(G^{TR})$. We then obtain a formulation of the reconciliation cost of G^{TR} as the sum of $cost_{TR}(y)$ for all nodes $y \in V_{cons}(G^{TR})$.

Lemma 7 describes the "valid" configurations of a subtree $G^{TR}[y]$ rooted at a node $y \in V_{cons}(G^{TR})$ such that $\mid \mathcal{G}(y) \mid > 2$, and the formula for computing $cost_{TR}(y)$ in each case. The following notations are used in Lemma 7. Given a node x of G^{Init}, and a node x^* of $G^{Init}[x_l]$, $A(x^*)$ is the set of all strict ancestors of x^* in $G^{Init}[x_l]$, $A_l(x^*)$ is the subset of $A(x^*)$ such that $u \in A_l(x^*)$ if $x^* \in V(G^{Init}[u_l])$ and $A_r(x^*) = A(x^*) \setminus A_l(x^*)$.

Lemma 7. *Let G^{TR} be a triplet respecting supergenetree*

for $\mathcal{G} = \{G_1, \ldots, G_k\}$. Let y be a node of G^{TR} such that $y \in V_{cons}(G^{TR})$ and $\mid \mathcal{G}(y) \mid > 2$. Let x be the node of G^{Init} such that $\mathcal{L}(y) = \mathcal{L}(x)$.

1) *If $\mid \mathcal{G}(x_l) \mid = 1$ and $\mid \mathcal{G}(x_r) \mid \geq 2$, let $y^* \in V_{cons}(G^{TR})$ be the node of G^{TR} such that $\mathcal{L}(y^*) = \mathcal{L}(x_r)$. The subtree $G^{TR}[y]$ can be obtained by taking the tree $G^{Init}[x_l]$ and grafting the tree $G^{TR}[y^*]$ onto it such that the root of $G^{TR}[y^*]$ appears as the sibling of a node x^* of $G^{Init}[x_l]$. The cost $cost_{TR}(y)$ is then given by the following formula:*
   $cost_{TR}(y) = cost_{TR}(x_l, x_r, x^*) =$
   $\sum_{u \in A_l(x^*)} cost(\mathcal{L}(u_l) \cup \mathcal{L}(x_r), \mathcal{L}(u_r))$
   $+ \sum_{u \in A_r(x^*)} cost(\mathcal{L}(u_l), \mathcal{L}(u_r) \cup \mathcal{L}(x_r))$
   $+ \sum_{u \in (V(G^{Init}[x_l]) \setminus A(x^*))} cost(\mathcal{L}(u_l), \mathcal{L}(u_r))$
   $+cost(\mathcal{L}(x_l), \mathcal{L}(x_r))$ *(if $x^* = x_l$)*
2) *If $\mid \mathcal{G}(x_l) \mid \geq 2$ and $\mid \mathcal{G}(x_r) \mid = 1$, then this case is symmetric to the previous case.*
3) *If $\mid \mathcal{G}(x_l) \mid \geq 2$ and $\mid \mathcal{G}(x_r) \mid \geq 2$, then $G^{TR}[y]$ is such that $\mathcal{L}(y_l) = \mathcal{L}(x_l)$ and $\mathcal{L}(y_r) = \mathcal{L}(x_r)$, and $cost_{TR}(y) = cost(\mathcal{L}(x_l), \mathcal{L}(x_r))$.*

Proof. In Case 1, we first deduce from Lemma 6 that there must exist a node $y^* \in V_{cons}(G^{TR})$ such that $\mathcal{L}(y^*) = \mathcal{L}(x_r)$. Next, $G^{Init}[x_l]$ is one of the gene trees of the set $\mathcal{G}$. So, it must be displayed by $G^{TR}[y]$ and then, $G^{TR}[y]$ can be obtained by taking $G^{Init}[x_l]$ and grafting $G^{TR}[y^*]$ onto it. Finally, Case 2 is symmetric to Case 1 and Case 3 follows directly from Lemma 6. The formulas for $cost_{TR}(y)$ follows directly from the definition of $cost_{TR}$. □

For example in Figure 3, the root and the highest δ node of G^{TR} fulfills the conditions of the first case. There are no node $y \in V_{cons}(G^{TR})$ satisfying $\mid \mathcal{G}(y) \mid > 2$ and fulfilling the conditions of the second or third case.

We are now ready to describe the recurrence formula of the recursive algorithm solving the $MinTRS$ problem. Given a node x of G^{Init} such that $\mid \mathcal{G}(x) \mid \geq 2$, we denote by $MinTRS(G^{Init}[x])$ the minimum LCA-reconciliation cost of a triplet respecting supergenetree compatible with $\mathcal{G}(x)$.

Lemma 8. *Let $\mathcal{G} = \{G_1, \ldots, G_k\}$ be a set of separated subtrees of a gene tree G^{Init} for Γ such that $\bigcup_{i=1}^{k} \mathcal{L}(G_i) = \Gamma$. Let x be a node of G^{Init} such that $\mid \mathcal{G}(x) \mid \geq 2$.*

1) *(Stop condition) If $\mid \mathcal{G}(x) \mid = 2$ ($\mathcal{G}(x) = \{G_i, G_j\}$), $MinTRS(G^{Init}[x]) = MinSGT(G_i, G_j)$.*
2) *Otherwise (i.e $\mid \mathcal{G}(x) \mid > 2$),*
   a) *If $\mid \mathcal{G}(x_l) \mid = 1$ and $\mid \mathcal{G}(x_r) \mid \geq 2$, $MinTRS(G^{Init}[x]) =$*
      $\min_{x^* \in V(G^{Init}[x_l])}\{cost_{TR}(x_l, x_r, x^*)\}$
      $+MinTRS(G^{Init}[x_r])$
   b) *If $\mid \mathcal{G}(x_l) \mid \geq 2$ and $\mid \mathcal{G}(x_r) \mid = 1$, this case is symmetric to the previous case.*
   c) *If $\mid \mathcal{G}(x_l) \mid \geq 2$ and $\mid \mathcal{G}(x_r) \mid \geq 2$, $MinTRS(G^{Init}[x]) =$*
      $cost_{TR}(\mathcal{L}(x_l), \mathcal{L}(x_r))$
      $+MinTRS(G^{Init}[x_l])$
      $+MinTRS(G^{Init}[x_r])$

Proof. The proof follows from Lemmas 6 and 7, and the fact that each call to the recursive procedure $MinTRS$ receives as input a subtree $G^{Init}[x]$ such that $\mid \mathcal{G}(x) \mid \geq 2$, starting with the whole tree rooted at $r(G^{Init})$. Case 1 is trivial. For Case 2(a) (and symmetrically Case 2(b)), following Lemma 7, there are $\mid V(G^{Init}[x_l]) \mid$ possible configurations for the subtree $G^{TR}[y]$ rooted at $y = s_{G^{Init} \to G^{TR}}(x)$, depending on which node x^* of $G^{Init}[x_l]$ is chosen to be the sibling of $y^* = s_{G^{Init} \to G^{TR}}(x_r)$. Since G^{TR} must be of minimum reconciliation cost, the configuration for $G^{TR}[y]$ must be one that locally minimizes the cost $cost_{TR}(x_l, x_r, x^*)$. Finally, Case 3 follows directly from Lemma 7. □

Complexity. we claim that Lemma 8 leads to a $O(n^2)$ algorithm for $MinTRS$, where $n = |V(G^{Init})|$. Let x be a node of G^{Init}, let n_x be the number of nodes in $G^{Init}[x]$ and let n_l and n_r be the number of nodes in the left and right subtrees of x, respectively. As a base case, if x falls into case 1 of Lemma 8, then running $MinSGT$ on the two child subtrees of x takes time $O(\max\{n_l, n_r\}^2) = O(n_x^2)$. Suppose instead that x falls into case 2.a, and thus $\mathcal{G}(x_l) = 1$ and $\mathcal{G}(x_r) \geq 2$. We may assume by induction that computing $MinTRS(G^{Init}[x_r])$ requires time $O(n_r^2)$. Afterwards, grafting the resulting tree is done on each $O(n_l)$ branch of $G^{Init}[x_l]$, and computing the cost can be done in time $O(n_l)$ for each grafting. Thus in total, case 2.a can be handled in time $O(n_r^2 + n_l^2) = O(n_x^2)$. The case 2.b of Lemma 8 is symmetric, and the case 2.c can be handled in constant time. As the quadratic bound holds for every node, we get a bound of $O(n_x^2) = O(n^2)$ when x is the root.

Algorithm for $MinLTRS$. The algorithm for the $MinTRS$ problem can be adapted to solve the $MinLTRS$ problem. The adaptation consists in (1) replacing the calls to $MinSGT(G_i, G_j)$ in the stop condition of Lemma 8 by calls to $MinLSGT((G_i, ev_i), (G_j, ev_j))$, and (2) replacing the use of $cost(L_l, L_r)$ in order to define $cost_{TR}(y)$ in Lemma 7 by the use of $cost_{\mathcal{G}}(L_l, L_r)$. Moreover, Lemma 5 must be extended such that $cost_{\mathcal{G}}(L_l, L_r) = +\infty$ if (L_l, L_r) is not label-compatible with $\mathcal{G}$. The complexity of the algorithm remains unchanged in $O(n^2)$.

5. Experiments

In the context of gene tree correction, we wanted to evaluate: (1) the benefit of the new supertree approach allowing to merge clades from different subtrees, compared with the more constrained polytomy resolution approach [19] which conserves input subtrees separated; (2) the benefit of the additional triplet preservation requirement of $MinTRS$ and $MinLTRS$. Both evaluations were made based on the conjecture of dubious highest duplication nodes in gene trees [23], [32].

	% of modified trees	Avg. running time	Avg rec. cost reduction	Trees with better AU value
$MinSGT$	211 97.2%	205240 ms	22.5 (24.8%)	
$MinLSGT$	207 95.3%	113 ms	19.5 (21.5%)	
$MinTRS$	211 97.2%	3031 ms	15.5 (17.1%)	68.6%
$MinLTRS$	207 95.3%	60 ms	13.5 (14.9%)	66.4%
$PolytomySolver$	20 ms 9.2%	3 ms	3.65 (4.0%)	50.0%

TABLE 1. RESULTS FOR THE 217 ENSEMBL TREES (SEE TEXT FOR ALL DETAILS). SECOND COLUMN: NUMBER AND PERCENTAGE OF CORRECTED TREES; THIRD COLUMN: MEAN RUNNING TIME FOR A TREE IN MS; FOURTH COLUMN: MEAN VALUE AND PERCENTAGE OF THE RECONCILIATION COST REDUCTION, I.E. DIFFERENCE IN RECONCILIATION COST BETWEEN THE ORIGINAL AND CORRECTED TREE; LAST COLUMN: PERCENTAGE OF CORRECTED TREES THAT HAVE A BETTER AU VALUE THAN THE ORIGINAL ENSEMBL TREES (DUE TO PHYML'S LONG COMPUTATION TIME, WE COULD NOT OBTAIN THE AU VALUES FOR $MinSGT$ AND $MinLSGT$).

For this purpose, we considered the gene trees of the Ensembl vertebrate database Release 84 rooted at a duplication node. For each tree G^{Init}, $\mathcal{G}$ was defined as the set of all subtrees of G^{Init} rooted at the "highest speciation nodes", i.e. speciation nodes with only duplication nodes as ancestors. On average, G^{Init} contains 121.2 leaves and is partitioned into 7.5 subtrees. Aiming at comparing all developed algorithms, including the exponential time $MinSGT$ and $MinLSGT$, we restricted the sample to the 217 gene trees with at most 200 leaves and partitioned into at most 5 subtrees. We also applied, on these 217 trees, $PolytomySolver$ [19] which, given a set of trees $G_1, \ldots, G_k$, finds a binary tree with leafset $\{G_1, \ldots, G_k\}$ such that the reconciliation cost of the resulting tree is minimum. Results are given in Table 5.

While the four supertree algorithms correct more than 200 trees, corresponding to more than 95% of the 217 trees, $PolytomySolver$ only corrects 20 trees corresponding to about 9% of the trees. Additionally, $PolytomySolver$ reduces the reconciliation cost by only 4% on average on the 20 corrected trees, compared to more than 15% for supertree algorithms. Clearly, by exploring a larger solution space, $MinSGT$ and $MinLSGT$ allow to obtain the best solutions in terms of reconciliation cost.

As for $MinTRS$ and $MinLTRS$, although more constrained than $MinSGT$ and $MinLSGT$, they lead surprisingly to almost as much correction as these two algorithms, while the correction achieved by $PolytomySolver$ is clearly less. The triplet preserving constraint appears to be less stringent than the conservation of the subtrees. In particular, for the trees leading to only two subtrees, $PolytomySolver$ conserves the initial tree. Notice that introducing the labeling constraint ($MinSGT$ versus $MinLSGT$ and $MinTRS$ versus $MinLTRS$) only leads to a slight decrease of the correction rates.

Finally, in order to assess the benefit of the triplet respecting constraint and the quality of the correction achieved, the trees corrected by $MinTRS, MinLTRS$ and $PolytomySolver$ were evaluated according to their statistical support. PhyML [30] was executed to obtain the log-likelihood values per site (note that 150 trees were included in this evaluation, as PhyML was very time-consuming on the larger trees). Consel [31] was then run to evaluate, using the AU (Approximately Unbiased) test, if the likelihood differences of pairs of Ensembl and corrected gene trees were significant enough to statistically reject one of them. A tree can be rejected if its AU value, interpreted as a p-value, is under 0.05. Otherwise, no significant evidence allows us to reject one of the two trees.

Interestingly, when compared to the tree output by $MinTRS$ (respectively $MinLTRS$), 48.5% (resp. 46.5%) of the Ensembl gene trees are rejected compared to only 11.9% (resp. 11.0%) of the corrected gene trees. More than 68.6% (resp. 66.4%) of the corrected trees have better AU values than original trees. As for PolytomySolver, 5% of the Ensembl trees were rejected, as 25% of the corrected trees were rejected, with 50% of the corrected trees obtaining a better AU value. The performance of $MinTRS$ and $MinLTRS$ is rather surprising as our correction, based on the phylogenetic information of the species tree, is not expected to improve tree likelihood based on sequence similarity. This may be an indication that high duplications are actually dubious and that a correction specifically focusing on such duplications is able to significantly improve the accuracy of the tree. This observation is further supported by the fact that the number of highest duplications is lower for corrected trees than for initial trees (data not shown), showing that our correction algorithms have the general tendency of deleting high duplications.

6. Conclusion

This paper introduces a new methodology combining the supertree and reconciliation frameworks with the purpose of constructing a gene tree by combining a set of trees on partial, possibly overlapping data. We also show how this new paradigm is useful for gene tree correction. In particular, the artifact of duplications wrongly inferred close to the root of a gene tree has been reported in the literature. Here, we propose a new method for correcting a gene tree, by

first removing the higher duplication nodes and then finding the supertree best fitting the species tree, that preserves the remaining "trusted" subtrees, and possibly their hierarchical position in the initial gene tree. This supertree approach is shown to correct more trees than the approach based on resolving a polytomy, as the first correction allows the clustering of genes from different input subtrees. The corrected Ensembl gene trees are shown to exhibit less highest duplication nodes and a lower reconciliation cost. Corrected gene trees are also shown to have a better likelihood support.

This new gene tree construction and correction paradigm leads to many new open problems. In particular, no proof currently exists on the complexity of the problem of finding a supergenetree minimizing the reconciliation cost, although it is likely to be NP-hard, based on the fact that minimizing the duplication cost is hard. The two problems (reconciliation versus duplication costs) probably also share the same inapproximability properties. However, it is possible that the supertree problems presented here are fixed-parameter tractable with respect to parameters such as the number of trees, the minimum reconciliation cost or the size of the intersection between the leafset of the trees. This is an area that deserves a more in-depth investigation. In addition, while the extension to the labeled case has been done with the same exponential complexity, adding the label restriction strongly constrains the set of explored bipartitions, and we can expect a more efficient algorithm in this case.

The problems we consider are build upon strong underlying assumptions, such as the consistency of input gene trees, the compatibility and S-consistency of input gene relations. A natural extension is then to integrate the notion of a minimal correction of input trees to fit these preliminary conditions. Finally, from an application point of view, rather than removing higher duplication nodes, other types of gene tree pruning can be envisaged and used to select the initial "trusted" phylogenetic information that can then be combined using our supergenetree and reconciliation framework.

Funding

M.L. acknowledges the support of the Fonds de Recherche du Québec Nature et Technologie (FRQNT). C.C. acknowledges the support of the Natural Sciences and Engineering Research Council (NSERC) (Discovery Grant RGPIN-249834). N.E. acknowledges the support of NSERC. A.O. acknowledges the support of the Canada Research Chairs (CRC) (CRC Tier2 Grant 950-230577).

References

[1] O. Bininda-Emonds, Ed., *Phylogenetic Supertrees combining information to reveal The Tree Of Life*, ser. Computational Biology. Dordrecht, the Netherlands: Kluwer Academic, 2004.

[2] M. Bansal, J. Burleigh, O. Eulenstein, and D. Fernández-Baca, "Robinson-foulds supertrees," *Alg. Mol. Biol.*, vol. 5, no. 18, 2010.

[3] N. Nguyen, S. Mirarab, and T. Warnow, "MRL and SuperFine+MRL: new supertree methods," *Alg. Mol. Biol.*, vol. 7, no. 3, 2012.

[4] V. Ranwez, V. Berry, A. Criscuolo, P. Fabre, S. Guillemot, C. Scornavacca, and E. Douzery, "PhySIC: a veto supertree method with desirable properties," *Syst. Biol.*, vol. 56, no. 5, pp. 798- 817, 2007.

[5] V. Ranwez, A. Criscuolo, and E. Douzery, "SuperTriplets: a triplet-based supertree approach to phylogenomics," *Bioinformatics*, vol. 26, no. 12, pp. i115- i123, 2010.

[6] M. Steel and A. Rodrigo, "Maximum likelihood supertrees," *Syst. Biol.*, vol. 57, no. 2, pp. 243- 250, 2008.

[7] M. Swenson, R. Suri, C. Linder, and T. Warnow, "SuperFine: fast and accurate supertree estimation," *Sys. Biol.*, vol. 61, no. 2, pp. 214-227, 2012, Special issue RECOMB-CG 2012.

[8] M. Steel, "The complexity of reconstructing trees from qualitative characters and subtrees," *J. Classif.*, vol. 9, pp. 91 - 116, 1992.

[9] C. Scornavacca, L. van Iersel, S. Kelk, and D. Bryant, "The agreement problem for unrooted phylogenetic trees is FPT," *J. Graph Algorithms Appl.*, vol. 18, no. 3, pp. 385 - 392, 2014.

[10] A. Aho, S. Yehoshua, T. Szymanski, and J. Ullman, "Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions," *SIAM J. Comput.*, vol. 10, no. 3, pp. 405-421, 1981.

[11] M. Constantinescu and D. Sankoff, "An efficient algorithm for supertrees," *J. Classif.*, vol. 12, pp. 101- 112, 1995.

[12] M. Ng and N. Wormald, "Reconstruction of rooted trees from subtrees," *Discrete Appl. Math*, vol. 69, pp. 19- 31, 1996.

[13] C. Semple, "Reconstructing minimal rooted trees," *Discrete Appl. Math.*, vol. 127, no. 3, 2003.

[14] L. Li, C. J. Stoeckert, and D. Roos, "OrthoMCL: identification of ortholog groups for eukaryotic genomes," *Genome Res.*, vol. 13, pp. 2178- 2189, 2003.

[15] A. Berglund, E. Sjolund, G. Ostlund, and E. Sonnhammer, "InParanoid 6: eukaryotic ortholog clusters with inparalogs," *Nucleic Acids Res.*, vol. 36, pp. D263 - D266, 2008.

[16] M. Lechner, S. Findeib, L. Steiner, M. Marz1, P. Stadler, and S. Prohaska, "Proteinortho: detection of (co-)orthologs in large-scale analysis," *BMC Bioinformatics*, vol. 12, p. 124, 2011.

[17] M. Hahn, "Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution," *Genome Biol.*, vol. 8, no. R141, 2007.

[18] Y.-C. Wu, M. D. Rasmussen, M. S. Bansal, and M. Kellis, "TreeFix: Statistically Informed Gene Tree Error Correction Using Species Trees," *Syst. Biol.*, vol. 62, no. 1, pp. 110–120, 2013.

[19] K. Chen, D. Durand, and M. Farach-Colton, "Notung: Dating gene duplications using gene family trees," *J. Computat. Biol.*, vol. 7, pp. 429–447, 2000.

[20] Y. Zheng and L. Zhang, "Reconciliation with non-binary gene trees revisited," in *Proceedings of RECOMB 2014*, ser. Lecture Notes Comput. Sci., vol. 8394, 2014, pp. 418-432, proceedings of RECOMB.

[21] K. M. Swenson and N. El-Mabrouk, "Gene trees and species trees: Irreconcilable differences," *BMC Bioinformatics*, vol. 13, no. (Suppl 19), p. S15, 2012.

[22] M. S. Bansal, Y. Wu, E. J. Alm, and M. Kellis, "Improved gene tree error correction in the presence of horizontal gene transfer," *Bioinformatics*, vol. 31, no. 8, pp. 1211–1218, 2015.

[23] E. Noutahi, M. Semeria, M. Lafond, J. Seguin, B. Boussau, L. Guguen, N. El-Mabrouk, and E. Tannier, "Efficient gene tree correction guided by genome evolution," *Plos One*, 2016, to appear.

[24] M. Lafond, E. Noutahi, and N. El-Mabrouk, "Efficient Non-Binary Gene Tree Resolution with Weighted Reconciliation Cost," in *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 54, 2016, pp. 14:1–14:12.

[25] S. Massey, A. Churbanov, S. Rastogi, and D. Liberles, "Characterizing positive and negative selection and their phylogenetic effects," *Gene*, vol. 418, pp. 22- 26, 2008.

[26] M. Skovgaard, J. Kodra, D. Gram, S. Knudsen, D. Madsen, and D. Liberles, "Using evolutionary information and ancestral sequences to understand the sequence-function relationship in GLP-1 agonists," *J. Mol. Biol.*, vol. 363, pp. 977- 988, 2006.

[27] S. Taylor, K. de la Cruz, M. Porter, and M. Whiting, "Characterization of the long-wavelength opsin from Mecoptera and Siphonaptera: does a flea see?" *Mol. Biol. Evol.*, vol. 22, pp. 1165- 1174, 2005.

[28] M. Lafond, A. Ouangraoua, and N. El-Mabrouk, "Reconstructing a supergenetree minimizing reconciliation," *BMC Genomics*, vol. 16, p. S4, 2015, Special issue of RECOMB-CG 2015.

[29] W. M. Fitch, "Homology. a personal view on some of the problems," *Trends Genet.*, vol. 16, no. 5, pp. 227- 231, 2000.

[30] M. Lafond and N. El-Mabrouk, "Orthology and paralogy constraints: satisfiability and consistency," *BMC Genomics*, vol. 15, no. Suppl 6, p. S12, 2014, Special issue RECOMB-CG 2014.

[31] L. Zhang, "On a Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies," *J. Comput. Biol.*, vol. 4, no. 2, pp. 177- 187, 1997.

[32] C. Chauve and N. El-Mabrouk, "New perspectives on gene family evolution: losses in reconciliation and a link with supertrees," in *Proceedings of RECOMB 2009*, ser. Lecture Notes Comput. Sci., vol. 5541. Springer, 2009, pp. 46-58.

[33] M. Hellmuth, M. Hernandez-Rosales, K. Huber, V. Moulton, P. Stadler, and N. Wieseke, "Orthology relations, symbolic ultrametrics, and cographs," *J. Math. Biol.*, vol. 66, no. 1–2, pp. 399–420, 2013.

[34] S. Guindon and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood," *Syst. Biol.*, vol. 52, no. 5, pp. 696–704, 2003.

[35] H. Shimodaira and M. Hasegawa, "CONSEL: for assessing the confidence of phylogenetoc tree selection," *Bioinformatics*, vol. 17, pp. 1246- 1247, 2001.

Manuel Lafond received the PhD degree from the University of Montreal, Canada in 2016. He is now a NSERC postdoctoral fellow in the Department of Mathematics and Statistics at the University of Ottawa, Canada, where he is developing new models and methods to distinguish orthologous genes from paralogous genes. His research interests mainly reside in the intersection of computational biology, algorithms and graph theory.

Cedric Chauve is a professor in mathematics at the Simon Fraser University. His research interests include algorithms design and analysis, genome evolution, genome rearrangements, and ancestral genome reconstruction.

Nadia El-Mabrouk is a full professor at the computer science department (DIRO) of the University of Montreal. She joined the department in 1998, after her Ph.D. in theoretical computer science from University "Paris VII" in France. Her research projects are related to the computational biology aspects of comparative genomics. She publishes in various computer science, bioinformatics and life science journals, such as "Bioinformatics", "Journal of Computational Biology" or "Molecular Biology and Evolution". She is involved, every year, in the program committees of several well-known bioinformatics conferences such as RECOMB, ISMB, WABI or APBC.

Aïda Ouangraoua received the master's degree and PhD degree in Computer Science from Université de Bordeaux, France, in 2004 and 2007, respectively. She did her postdoctoral training at Simon Fraser University and Université du Québec à Montréal, Canada. She was a researcher at INRIA Lille, France from 2009 to 2014. She is currently a professor at the Department of Computer Science of Université de Sherbrooke, Québec, Canada. Her research interests include Algorithms and Computational Molecular Biology.