Generation of level-k LGT Networks

Joan Carles Pons, Celine Scornavacca, Gabriel Cardona

Generation of level-k LGT Networks

Joan Carles Pons, Celine Scornavacca, Gabriel Cardona

***Abstract*—Phylogenetic networks provide a mathematical model to represent the evolution of a set of species where, apart from speciation, reticulate evolutionary events have to be taken into account. Among these events, lateral gene transfers need special consideration due to the asymmetry in the roles of the species involved in such an event. To take into account this asymmetry, LGT networks were introduced.**

Contrarily to the case of phylogenetic trees, the combinatorial structure of phylogenetic networks is much less known and difficult to describe. One of the approaches in the literature is to classify them according to their *level* and find *generators* of the given level that can be used to recursively generate all networks.

In this paper we adapt the concept of generators to the case of LGT networks. We show how these generators, classified by their level, give rise to simple LGT networks of the specified level, and how any LGT network can be obtained from these simple networks, that act as building blocks of the generic structure.

The stochastic models of evolution of phylogenetic networks are also much less studied than those for phylogenetic trees. In this setting, we introduce a novel two-parameter model that generates LGT networks. Finally, we present some computer simulations using this model in order to investigate the complexity of the generated networks, depending on the parameters of the model.

***Index Terms*—Phylogenetic network, LGT network, level-k, generator, blobbed tree, evolutionary model.**

I. Introduction

The evolution of species is often modeled as a branching mechanism and represented via phylogenetic trees. However, the presence of reticulation events (hybridisations, recombinations and lateral gene transfers to name a few) makes *phylogenetic networks* a more accurate model [Doolittle and Bapteste, 2007]. These networks, in their broadest sense, can simply be thought as graphs (directed or undirected) with their leaves labelled by species. Among the different kinds of phylogenetic networks that exist in the literature, a model that recently has gained much attention from the scientific community [Anaya et al., 2016], [Francis et al., 2018a], [Francis et al., 2018b], [Jetten and van Iersel, 2018], [Pons et al., 2018], [Bordewich and Semple, 2018] is given by *tree-based networks* [Francis and Steel, 2015]. In these networks, the underlying backbone of tree-like evolution is represented by a phylogenetic tree — the base tree— to which reticulate events are added. Specially designed to highlight the asymmetricity of lateral gene transfer events, [Cardona et al., 2015] introduced *LGT networks*, i.e. tree-based networks with a base tree representing the main line of evolution of the organisms under study and with a subset of its arcs representing LGT events.

JCP and GC have been partially supported by the Spanish Ministry of Economy and Competitiveness and European Regional Development Fund project DPI2015-67082-P (MINECO/FEDER). CS has been partially supported by a research bursary for visiting lecturers from the University of the Balearic Islands.

In order to use phylogenetic networks in a phylogenomics context —e.g. reconstructing them using the information contained in a set of phylogenetic networks— while avoiding the combinatorial explosion, some topological restrictions have been introduced [Cardona et al., 2008], [Cardona et al., 2009], [Huson et al., 2010], [Huber et al., 2016]. One of these restrictions is to focus only on level-k networks, which are phylogenetic networks where the number of reticulation events in each biconnected component is bounded by k [Choy et al., 2005]. The characterization of the combinatorial structure of level-k networks was firstly introduced in [van Iersel et al., 2009a], [van Iersel et al., 2009b] and further developed in [Gambette et al., 2009], where a polynomial time algorithm was given in order to build all level-$(k+1)$ generators from the level-k ones. This process allows to generate the whole set of generators for any k, although the process becomes soon prohibitively slow.

We begin this paper reviewing in Section II some basic definitions on phylogenetic networks and LGT networks. Then, in Section III we introduce LGT generators as an adaptation of generators to the case of LGT networks; we give their definition, properties and algorithms to generate them. These algorithms have been implemented in python and can be downloaded from https://github.com/bielcardona/LGTGenerators. In Section IV we show how the building blocks of level-k LGT networks can be obtained from the generators constructed in the previous section, and in Section V we prove that gluing together these building blocks we can effectively generate all level-k LGT networks. Finally, in Section VI we give a stochastic model of evolution that generates LGT networks and make some computer simulations to analyze the complexity of the generated networks.

II. Preliminaries

A. Phylogenetic networks

The mathematical objects used in this paper to depict evolution are directed acyclic graphs (DAGs, for short). A node in a DAG D is a *reticulation* if it has indegree greater than one, and *principal* otherwise. A node with indegree zero is called a *root*, and a node with outdegree zero is called a *leaf*. We shall always assume that DAGs are *rooted*, i.e. they have a single root r, and we will say that they are rDAGs. A principal node with outdegree 1 is called *elementary*. The *subdivision* of an arc (u,v) in D is the result of replacing (u,v) by two arcs (u,w) and (w,v), where w is a new (elementary) node. A *subdivision* of D is a directed graph obtained from D by a sequence of arc subdivisions. Conversely, *contracting* an elementary node $w \neq r$ in D is the operation of replacing the two arcs, say (u,w) and (w,v), incident with w by the single arc (u,v), and finally deleting the node w. If the

elementary node is the root r, then contracting it coincides with its removal. *Binary* DAGs are DAGs in which all nodes have both indegree and outdegree ≤ 2.

A *binary (rooted) phylogenetic network*, from now on simply refered to as a *network*, is a binary rDAG without elementary nodes and whose reticulations have outdegree 1. Notice that the different kinds of nodes, classified by their pair of degrees $d(u) = (\text{indeg}\, u, \text{outdeg}\, u)$ are: the root, with $d(u) = (0, 2)$, the (internal) principal nodes, with $d(u) = (1, 2)$, the leaves, with $d(u) = (1, 0)$, and the reticulations, with $d(u) = (2, 1)$. Notice also that the term phylogenetic tree/network is usually reserved for trees/networks whose leaves are labeled by a set X of taxa. In this paper, however, no such labeling is assumed; we sometimes emphasize this fact saying that the trees or networks are *unlabeled*.

Throughout this paper, whenever we consider the connectivity of a directed graph we will mean the connectivity of the underlying undirected graph. A *cut node* (resp. *cut arc*) in a directed graph is a node (resp. arc) whose removal increases the number of connected components of the graph. A *biconnected component*, or a *blob*, of a network is a non-trivial maximal connected subgraph without cut nodes. In this setting, by non-trivial we mean that we exclude the case of two nodes connected by an arc. It follows from the definition that different blobs are arc-disjoint; in this paper we also assume that they are node-disjoint, which is the case for phylogenetic networks as we have defined them. A network is of *level-k* when all their biconnected components have at most k reticulation nodes [van Iersel et al., 2009a]. Roughly speaking, the level of a network measures the extent of interlacing between the reticulation nodes in the network. It can be used as a measure for how tree-like a network is. A level-k network which is not a level-$(k-1)$ network is called a *strict* level-k network.

A network can thus be seen as having tree-like parts and non-tree-like parts, the blobs, containing the reticulations. Once the set of blobs in a network N is determined, contracting each blob to a single node, the network becomes a tree because every arc outside of a blob is a cut arc. This tree is referred to as the *blobbed tree* of N and it is denoted by $B(N)$ [Gusfield and Bansal, 2005], [Gusfield, 2014]. We remark that the blobbed tree may not be a phylogenetic tree (in the sense of the definition above) since it could be non binary and contain elementary nodes. See Fig. 1 for an example.

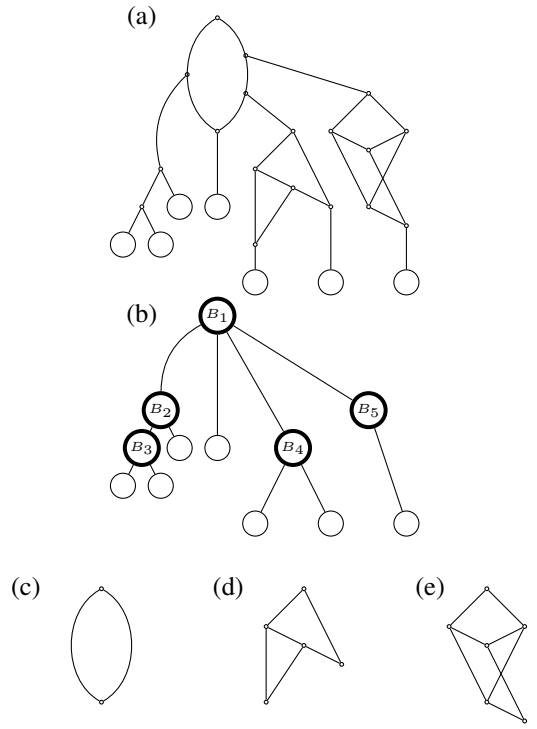

Fig. 1. (a) A level-3 phylogenetic network N; (b) The blobbed tree of the network N where every node with thick borders represents an internal blob; (c-e) The three non trivial blobs of the network, B_1, B_4 and B_5, respectively – B_2 and B_3 are trivial.

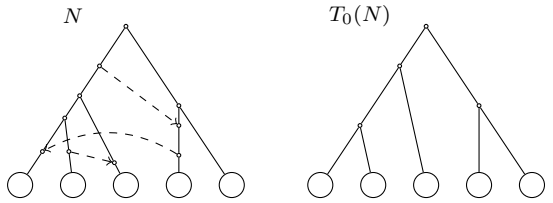

Fig. 2. A level-3 LGT network N with three secondary arcs (dashed) and its principal subtree $T_0(N)$ after contracting its elementary nodes.

B. LGT networks

An *LGT network* $N = (V, E)$ is a network along with a partition of E in a set of *principal* arcs E_p and a set of *secondary* arcs E_s, such that its *principal subtree* $T_0(N) = (V, E_p)$ is a tree (maybe with elementary nodes), see Fig. 2. It is straightforward to check that the following two conditions characterize LGT networks among phylogenetic networks:

1) except for the root, every node has exactly one principal incoming arc, and
2) except for the leaves, every node has at least one principal outgoing arc.

Note that our definition of LGT network differs slightly from the definition in [Cardona et al., 2015] because here we focus on binary networks. Note also that LGT networks are *tree-based* networks, where $T_0(N)$ is a *distinguished base tree* [Francis and Steel, 2015]. A path in an LGT network N is *principal* if it consists only of principal arcs. We say that an LGT network is of level k if the underlying network (i.e., with no distinction of principal and secondary arcs) is of level k.

III. Generators for LGT networks

The combinatorial structure of level-k phylogenetic networks is based on the concept of *generators* [van Iersel et al., 2009a]. These generators allow for the construction of *simple level-k networks*, which are defined as level-k networks containing no cut arc except the trivial ones leading to leaves, and constitute the basic building blocks for all level-k networks. Formally, a *level-k generator* is a binary rooted directed acyclic multigraph without cut or elementary nodes, with k reticulations, and whose leaves are reticulations. In particular, the only possibilities for $d(u) = (\text{indeg}\, u, \text{outdeg}\, u)$

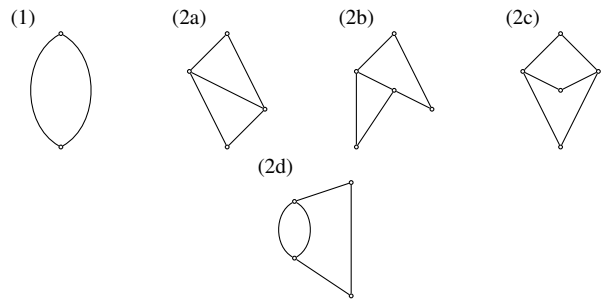

Fig. 3. The only level-1 generator (1) and the four level-2 generators (2a-d).

are $(0,2)$, $(1,2)$, $(2,0)$, and $(2,1)$. There is a single level-1 generator and 4 level-2 generators, which are depicted in Figure 3. This number greatly increases for higher levels: 65 level-3 generators, 1993 level-4 generators, etc. We denote by S_k the set of level-k generators and by $g_k = |S_k|$ the number of such generators.

In this section we adapt the definition of generators above to the case of LGT networks, obtaining what we call *LGT generators*. For small values of the level, we will show how to compute all those generators, up to isomorphisms.

Definition 1: A level-k LGT generator is a level-k generator, as defined above, together with a tagging of some of its arcs using the labels `SEC` and `MIX`, with the following restrictions:

1) For each reticulation, exactly one of its incoming arcs is tagged, and if it is not a leaf, then its single outgoing arc is not tagged with `SEC`.
2) For each principal node, if both outgoing arcs are tagged, then their tags are different, and if only one of them is tagged, then it is tagged with `SEC`. Also, if it is not the root, then its single incoming arc is not tagged.

The rationale behind this definition is that we will construct LGT networks from these generators; arcs not tagged will be replaced by principal paths, arcs tagged with `SEC` will represent secondary arcs, and arcs tagged with `MIX` will become paths formed by a principal path followed by a secondary arc.

In order to ease notations, we will indicate by $\hat{G}$ a level-k LGT generator and by G the underlying generator (that is, forgetting the tagging).

Given a level-k generator G, all possible taggings that make it an LGT generator can be obtained by the following procedure:

1) For each reticulation in G, choose one of its incoming arcs and tag it with `SEC`.
2) For each reticulation in G, if it has a tagged outgoing arc, change the tag to `MIX`.
3) For each principal node in G, if both outgoing arcs are tagged with `SEC`, choose one of them and change the tag to `MIX`.

Proposition 1: The procedure described above generates a level-k LGT generator. Conversely, every level-k LGT generator can be obtained from a level-k generator using this procedure.

Proof: Let us assume that we tag a level-k generator using the procedure. Let u be one of its reticulations. Step 1 in the procedure ensures that exactly one of its incoming arcs are tagged, and notice that steps 2 and 3 can change tags, but do not tag untagged arcs or viceversa. If u is not a leaf, then its outgoing arc could have been tagged with `SEC` in step 1, but in such a case, its tag would be changed to `MIX` in step 2. Hence, condition 1 in the definition of level-k LGT generator is satisfied.

Now, let us assume that u is a principal node. If both of its outgoing arcs are tagged in step 1, then exactly one of them changes its tag in step 3. If only one of them is tagged in step 1, then its tag will be `SEC` and will not be changed in the other steps. Finally, if u has one incoming arc, then it will not be tagged, since step 1 only applies to arcs whose endpoint is a reticulation and the other steps can change tags, but do not tag untagged arcs or viceversa. Thus, condition 2 is also satisfied.

If we have a level-k LGT generator $\hat{G}$, then it can be obtained using the described procedure as follows: In step 1, choose as the arc to be tagged the same one that is tagged in $\hat{G}$ with `SEC` or `MIX`. In step 3, choose as the arc to have its tag changed the one whose tag is `MIX` in $\hat{G}$. ■

We denote by $\hat{S}_k(G)$ the set of all level-k LGT generators that can be constructed from G and by $\hat{S}_k$ the set of all level-k LGT generators. Also, let $\hat{g}_k(G) = |\hat{S}_k(G)|$ and $\hat{g}_k = |\hat{S}_k|$.

Proposition 2: $\hat{g}_k \leq g_k \cdot 2^{k+\lfloor k/2 \rfloor}$.

Proof: Since there are g_k level-k generators, it is enough to count how many taggings can be defined on each of them. Notice that in the tagging procedure above the first step can be done in exactly 2^k ways. The second step can only be done in one way. Finally, the third step can be done in 2^h ways, where h is the number of principal nodes having both outgoing arcs tagged, but since there are at most k tagged arcs, there can be at most $\lfloor k/2 \rfloor$ such nodes. ■

Notice that Propositions 2.4 and 2.5 in [Gambette et al., 2009] prove that $2^{k-1} \leq g_k \leq k!^2 \cdot 50^k$ and hence the number of level-k LGT generators is bounded from above by $k!^2 \cdot 50^k \cdot 2^{k+\lfloor k/2 \rfloor}$. As for a lower bound, notice that the different taggings applied to a generator could give isomorphic LGT generators, in the sense that there exists an isomorphism of directed graphs that preserves the taggings. Hence, in principle only the bound $\hat{g}_k \geq 2^{k-1}$ can be given.

The procedure presented in this section can be used to compute the set of all level-k LGT generators for small values of k; it has been implemented in python using the package `phylonetwork` [Cardona and Sánchez, 2012] and can be downloaded from https://github.com/bielcardona/LGTGenerators. This script takes as input the set of level-k generators [Gambette et al., 2009], then uses the procedure described in this section to find all possible taggings and finally checks for isomorphism (of arc-labeled directed multigraphs) to remove duplicates. The number of LGT generators for small values of the level is summarized in Table I, and Figure 4 shows the drawings of all the LGT generators of level 2.

IV. Simple level-k LGT networks

In this section we show how we can obtain all simple level-k LGT networks from the level-k LGT generators defined in the previous section.

TABLE I
NUMBER OF GENERATORS AND LGT GENERATORS FOR SMALL VALUES OF THE LEVEL k.

k	g_k	$\hat{g}_k$
1	1	1
2	4	14
3	65	546
4	1,993	39,257
5	91,454	4,052,250

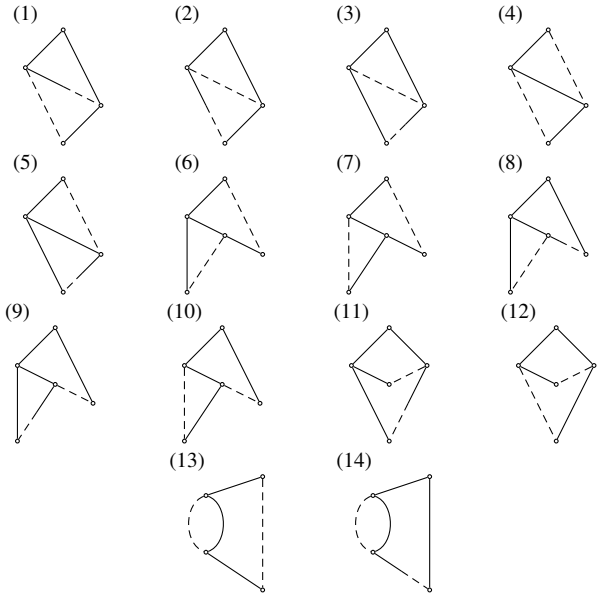

Fig. 4. The set of non-isomorphic level-2 LGT generators. In this figure, arcs not tagged are drawn with solid lines, arcs tagged with SEC are drawn with dashed lines, and those tagged with MIX are drawn with mixed lines (the first half with solid line and the second one with dashed line). The LGT generators (1-5) are associated with generator (2a) in Figure 3, the ones in (6-10) with the generator (2b), the ones in (11-12) with generator (2c) and the LGT generators (13-14) with generator (2d).

Given a level-k LGT generator $\hat{G}$, consider the following construction:

1) For each arc $e = (u, v)$ tagged with MIX, remove the tag, subdivide it by introducing an elementary node w and tag the second arc (w, v) with SEC.
2) For each node w that is either a reticulation or elementary, add a newly created node l (which will be a leaf) and an arc (w, l).

Let $M(\hat{G}) = (V, E)$ the directed graph constructed so far, together with the partition of its arcs obtained defining E_p as the set of arcs not tagged, and E_s as the set of tagged arcs (notice that at this point only the tag SEC is used). See Figure 5 for a depiction of this construction.

Proposition 3: Let $\hat{G}$ be a level-k LGT generator. Then, the directed graph $M(\hat{G})$ constructed above is an unlabeled simple level-k LGT network.

Proof: It follows from the definition of generators that $M(\hat{G})$ is an unlabeled phylogenetic network; also, since the definition of the level for an LGT network does not depend on the partition of arcs into principal and secondary ones, if G has level k, so has $\hat{G}$ and $M(\hat{G})$.

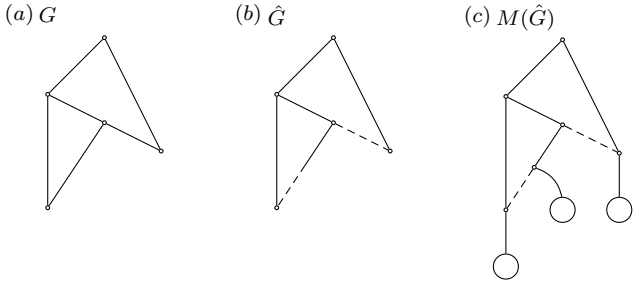

Fig. 5. (a) A level-2 generator; (b) A level-2 LGT generator $\hat{G}$ constructed from G; (c) The (unlabeled) level-2 LGT network $M(\hat{G})$ obtained from $\hat{G}$. Again, secondary arcs are drawn using dashed lines and principal arcs with solid lines.

Finally, notice that the set of principal arcs, build up a tree, since in the construction of $M(\hat{G})$ each node has one single incoming principal arc (except for the root) and at least one outgoing principal arc (except for the leaves). ■

This construction can be generalized to obtain all simple level-k LGT networks taking as input the set of all generators by means of the following construction.

Let $\hat{G}$ be a level-k LGT generator and fix a mapping $n : E \to \mathbb{Z}_{\geq 0}$ that assigns to each arc e a non-negative integer $n(e)$ with the only condition that $n(e) \geq 1$ if e is tagged with MIX. With this data we define a level-k LGT network according to the following construction:

1) For each arc $e = (u, v)$:
   a) If $n(e) > 0$, subdivide it by introducing $n(e)$ elementary nodes $w_1, \ldots, w_{n(e)}$.
   b) If, moreover, e is tagged, tag the last arc $(w_{n(e)}, v)$ —and only this last arc— with SEC.
2) For each node w that is either a reticulation or elementary, add a newly created node l (which will be a leaf) and an arc (w, l).

Let $M(\hat{G}, n) = (V, E)$ the directed graph constructed so far, together with the partition of its arcs obtained by defining E_p equal to the set of arcs not tagged, and E_s as the set of tagged arcs (notice that at this point only the tag SEC is used). Notice that $M(\hat{G})$ in Proposition 3 corresponds to the particular case that $n(e) = 1$ if e is tagged with MIX and $n(e) = 0$ otherwise.

Theorem 1: Let $\hat{G}$ be a level-k LGT generator with set of edges E, and $n : E \to \mathbb{Z}_{\geq 0}$ a mapping such that $n(e) = 1$ if e is tagged with MIX. Then, the directed graph $M(\hat{G}, n)$ constructed above is an unlabeled simple level-k LGT network. Conversely, given an unlabeled simple level-k LGT network N, then there exists a level-k LGT generator $\hat{G}$ and a mapping $n : E(\hat{G}) \to \mathbb{Z}_{\geq 0}$ such that $M(\hat{G}, n)$ is isomorphic to N.

Proof: The first part of the theorem is proved analogously to Proposition 3.

As for the converse, let N be an unlabeled simple level-k LGT network. Remember that the different possibilities for the pair $(\operatorname{indeg} u, \operatorname{outdeg} u)$ for any node u of N are $(0, 2)$, $(1, 2)$, $(1, 0)$, and $(2, 1)$, corresponding respectively to the root, the principal nodes, the leaves and the reticulations. Let N_1 be the directed graph obtained by removing the leaves of N. We recall that N has neither cut nodes nor cut arcs, except

those whose target is a leaf. Hence, N_1 has neither cut nodes nor cut arcs. For each leaf of N that we remove, its parent u can either be a reticulation or a principal node. After the removal of the leaf, the respective possibilities for the pair $(\operatorname{indeg} u, \operatorname{outdeg} u)$ are $(2,0)$ and $(1,1)$. Hence, all the nodes of N_1 have their pair of indegree and outdegree equal to $(0,2)$, $(1,2)$, $(1,1)$, $(2,1)$, or $(2,0)$. In particular, the elementary nodes of N_1 are principal nodes in N. As a consequence, the elementary paths in N_1 are formed by principal arcs, except for the last arc, that can be (or not) secondary. Notice also that N can be reconstructed, up to isomorphism, from N_1 by simply hanging leaves to the elementary nodes and the nodes with outdegree zero of N_1.

Let G be the directed graph obtained from N_1 by contracting all the elementary nodes. In G all nodes have their pair of degrees equal to $(0,2)$, $(1,2)$, $(2,1)$, or $(2,0)$. Also, since G does not have neither cut arcs nor cut nodes, it follows that G is a level-k generator since no reticulation has been removed.

We define now a tagging on the set E of arcs of G, and a mapping $n : E \to \mathbb{Z}_{\geq 0}$ as follows:

1) If $e \in E$ was a principal arc in N_1, then e gets no tag and we define $n(e) = 0$.
2) If $e \in E$ was a secondary arc in N_1, then we tag e with `SEC` and we define $n(e) = 0$.
3) If $e \in E$ has been obtained by contraction of an elementary path $u = u_0 \to u_1 \to \cdots \to u_r \to v$ ($r \geq 1$) in N_1 we set $n(e) = r$. If, moreover, (u_r, v) is secondary, we tag e with `MIX`.

Since N is an LGT network, the tagging we have just defined satisfies the conditions in Definition 1 for reticulations, but not necessarily for principal nodes. In order to satisfy all conditions, we modify it as follows: For every principal node u, if either (1) both of its outgoing arcs are tagged with `MIX`—say e is one of them—, or (2) only one of its outgoing arcs is tagged and the tag is `MIX`—say e is this arc—, then change the tag of e to `SEC`.

Let $\hat{G}$ be the generator G together with the obtained tagging. It is straightforward to check that $\hat{G}$ is an LGT generator and that it has the same level as N. Also, N_1 can be recovered from $(\hat{G}, n)$. Indeed, the only differences between N_1 and G are (1) that elementary paths are substituted with arcs and (2) the partition of arcs into principal and secondary arcs is lost. The lengths of the paths can be recovered by looking at the mapping n, while the set of secondary arcs is determined by the tagging. Finally, notice that this reconstruction process is exactly the definition of $M(\hat{G}, n)$. ∎

Note also that the characterization given by this theorem is not unique, since different generators and mappings can produce the same simple LGT network. See Figure 6.

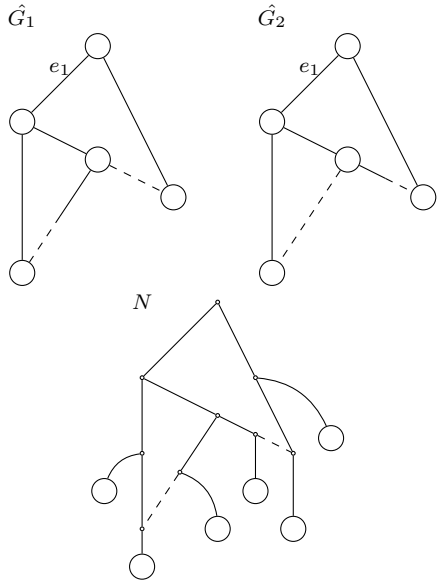

Fig. 6. Two level-2 LGT generators $\hat{G}_1$ and $\hat{G}_2$ and a simple level-2 LGT network N. N can be obtained both as $M(\hat{G}_1, n)$ and as $M(\hat{G}_2, n)$, with $n(e_1) = 0$ and $n(e) = 1$ for every $e \neq e_1$.

V. Level-k LGT networks

Let $\mathcal{N}_k^1$ be the set of (unlabeled) simple LGT networks of level at most k. For convenience, we let $\mathcal{N}_0^1 = \{T_0, T_2\}$, where T_0 is the trivial graph with a single node, and T_2 is the binary rooted tree with 2 leaves. We define recursively the sets $\mathcal{N}_k^m$ ($m > 1$) as

$$\mathcal{N}_k^m = \{R(N, \ell, N') \mid N \in \mathcal{N}_k^{m-1}, \ell \text{ is a leaf of } N, N' \in \mathcal{N}_k^1\},$$

where $R(N, \ell, N')$ is the network that is obtained by substituting in N the node ℓ with the network N' (more formally, it is the disjoint union of N and N' modulo identifying ℓ with $r(N')$). We also define

$$\mathcal{N}_k = \bigcup_{m \geq 1} \mathcal{N}_k^m.$$

That is, $\mathcal{N}_k$ is the set of LGT networks that can be obtained by taking simple level-k LGT networks and substituting a leaf with another simple level-k LGT network and iterating the process.

Theorem 2: Let N be an unlabeled level-k LGT network, and let m be the number of internal nodes of its blobbed tree. Then, $N \in \mathcal{N}_k^m$.

Proof: Let $B = B(N)$ be the blobbed tree of N. Each node of B corresponds to a biconnected component (or blob) of N, which might be trivial, i.e. a single node. For each blob U of N, we shall denote by $[U]$ the corresponding node of B. Recall that there is an arc $([U], [U'])$ in B, if and only if there is an arc in N connecting a node $u \in U$ to a node of $u' \in U'$, and in such a case the nodes u, u' are unique. Also, arcs joining different blobs must necessarily be principal, since its target will never be a reticulation. Hence N can be recovered up to isomorphism by taking the disjoint union of its blobs (which might contain both principal and secondary arcs) and adding to this disjoint union one (suitably chosen) principal arc for each arc of B. Notice that adding such an arc, say from u to v, is equivalent to: (1) adding a pendant leaf $\ell(u)$ to u and (2) identifying $\ell(u)$ and v.

Let $[U_1], \ldots, [U_n]$ be the nodes of B ordered according to a topological sort, so that if $([U_i], [U_j])$ is an arc of B, then $i < j$. We can also assume that the first m nodes $[U_1], \ldots, [U_m]$ are the internal nodes of B and that the other ones $[U_{m+1}], \ldots, [U_n]$ are the leaves. In particular, $[U_1]$ is the root of B. For each blob U_j ($j \neq 1$) with root $r(j)$ there exists

another one $U_{\iota(j)}$ (with $\iota(j) < j$) and a node $p(j) \in U_{\iota(j)}$ such that $(p(j), r(j))$ is an arc of N; also both $\iota(j)$ and $p(j)$ are uniquely determined by j. Due to our restrictions of the degrees of the nodes, the node $p(j)$ must either be (1) an elementary node, (2) a reticulation without outgoing arcs, or (3) the single node in its blob.

For each blob U_i we define the network $\overline{U}_i$ as follows:

- If U_i is non-trivial, then $\overline{U}_i$ is the result of adding to U_i, for each of its nodes that is either elementary or a reticulation without outgoing arcs, a pendant leaf to it.
- If U_i is trivial (i.e. it is a single node) and $[U_i]$ is internal in B, then $\overline{U}_i$ is the result of adding two pendant leaves to this node.
- If U_i is trivial and $[U_i]$ is a leaf in B, then $\overline{U}_i = U_i$.

Note that in any case $\overline{U}_i$ belongs to $\mathcal{N}_k^1$.

For each blob U_j ($j \neq 1$), consider $p(j) \in U_{\iota(j)}$ as defined above. In the construction of $\overline{U}_{\iota(j)}$, at least one pendant arc has been added to $p(j)$. Notice also that if there are two different blobs j, j' with $p(j) = p(j')$, then the blob containing $p(j)$ must contain one single node and hence $\overline{U}_{\iota(j)}$ has been obtained by adding two pendant leaves to $p(j)$. Therefore, we can construct in any case an injective mapping $j \mapsto \ell(j)$ such that $\ell(j)$ is a leaf in $\overline{U}_{\iota(j)}$ and there is an arc in N from its parent $p(j)$ to the root $r(j)$ of U_j.

Let $\tilde{U}_i$ be defined recursively as:

- $\tilde{U}_1 = \overline{U}_1$.
- $\tilde{U}_i = R(\tilde{U}_{i-1}, \ell(i), \overline{U}_i)$ if $i > 1$.

Notice that if $i > m$, then $\overline{U}_i$ is a single node, and hence $R(\tilde{U}_{i-1}, \ell(i), \overline{U}_i) \simeq \tilde{U}_{i-1}$. Therefore, we can stop the construction of these networks with $\tilde{U}_m$. Notice that at the end we get a network $\tilde{U}_m \in \mathcal{N}_k^m$. Also (see the first paragraph in this proof) the operations we have made to construct $\tilde{U}_m$ are equivalent to adding the required arcs to the disjoint union of its blobs in order to recover N and hence $N \simeq \tilde{U}_m$. ■

As a consequence, we find that all level-k LGT networks can be constructed gluing together simple ones.

Corollary 1: The set $\mathcal{N}_k$ is equal to the set of all level-k LGT networks.

VI. An evolutionary model for LGT networks

In this section we introduce a simple stochastic evolutionary model that accounts for speciation and lateral gene transfer (LGT) events and hence generates LGT networks that are, moreover, time-consistent [Baroni et al., 2006]. To our knowledge, this is the first model where LGT events are generated alongside speciations; other models of evolution consider a fixed base tree and then model LGT events between its branches [Steel et al., 2013], [Roch and Snir, 2013], [Linz et al., 2007]. We have performed multiple simulations using this model in order to generate networks with a given number of events and have made a statistical study of the level and the number of biconnected components (blobs) of the obtained networks. Both the scripts and the results obtained can be downloaded from https://github.com/bielcardona/LGTGenerators.

The model we are going to describe depends on two different parameters $\alpha \in [0, 1]$ and $\beta \geq 0$. Roughly speaking –see below for details– the first parameter controls how likely an LGT event is, as opposed to a speciation event, and the second one how likely it is that the ancestors of the species involved in an LGT event have already participated in previous LGT events.

We first describe the two evolutionary processes.

1) Let l be a leaf of a network N. The *speciation event on* l, denoted by $S(N, l)$, consists in splitting l into two leaves. That is, adding two new leaves l_1 and l_2 and two principal arcs (l, l_1) and (l, l_2) to N.
2) Given two leaves l, m of a network N, the *LGT event from* l *to* m, denoted by $L(N, l, m)$, consists in adding two new leaves l_1 and m_1, two principal arcs (l, l_1) and (m, m_1), and the secondary arc (l, m). Such an LGT event can be *internal* if the parents of the leaves l and m belong to the same blob of N and *external* otherwise. Hence, an LGT event is internal if, in the evolutive history of the leaves involved in the process, some previous LGT event has taken place, and external otherwise.

The stochastic model of evolution can be described as follows. Starting with the binary rooted tree with two leaves $N_1 = T_2$, at each step construct N_{k+1} from N_k according to these rules: With probability α choose the next event to be an LGT event, and with probability $1 - \alpha$ a speciation event.

- If a speciation event is going to take place, choose randomly and uniformly l, one of the leaves of N_k, and compute $N_{k+1} = S(N_k, l)$.
- If an LGT event is going to take place, consider all (ordered) pairs of different leaves, and assign to each pair (l, m) a probability of being chosen equal to p if the respective parents of l and m belong to the same blob of N_k and $\beta \cdot p$ otherwise. Obviously, p must be chosen so that the sum of all the probabilities is 1. Then, choose one of these pairs (l, m) with the given probability distribution and compute $N_{k+1} = L(N_k, l, m)$.

Figure 7 shows an example of an LGT network produced using the evolutionary model we have described by means of the events $N_2 = S(N_1, 3)$, $N_3 = S(N_2, 4)$, $N_4 = L(N_3, 7, 5)$, $N_5 = L(N_4, 8, 6)$. Note that the first LGT event is external, while the second one is internal.

We have performed several simulations using the described model, in order to study the complexity of the generated networks.

In our first experiment we have considered $\beta \in \{0.01, 10\}$, $\alpha \in \{0, 0.1, 0.2, 0.3, 0.4\}$ and $n \in \{10, 30\}$. For each value of n we have considered the experiment of generating a network using n iterations of the stochastic model with the chosen parameters α and β. We have run this experiment 500 times and computed both the average value of the level and of the number of nontrivial blobs of the resulting networks. These results are shown in Figures 8 and 9. Notice that for each choice of n and β, the graphs show the dependence on α of the indicators of complexity under study. From the observation of the plots it follows that the average level has a linear dependence on α, while the number of blobs increases but tends to stabilize. This phenomena should be

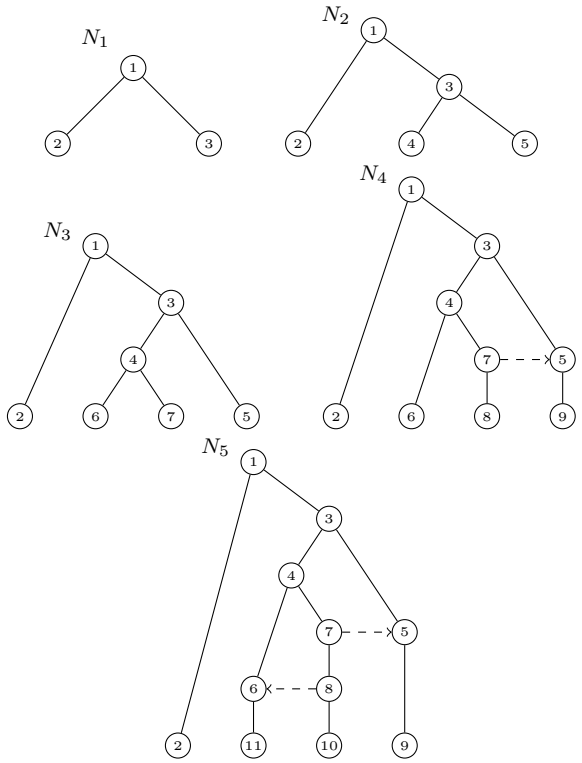

Fig. 7. An example of how the evolution processes generate an LGT network.

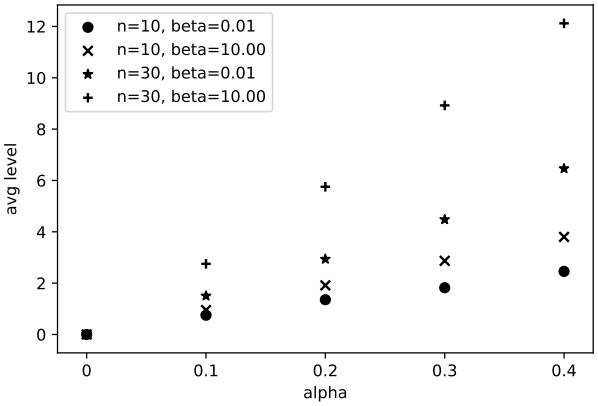

Fig. 8. Dependence on α of the average level of the simulated networks after 500 iterations of the experiment, with different values of the other parameters.

expected, as α represents the probability that an LGT event takes place, and hence simulations with increasing values of this parameter should give higher level. As for the number of blobs, increasing the value of α has two different effects. At first, it creates new nontrivial blobs, but when more blobs are created, the probability that an external LGT event takes place increases, and hence different blobs get connected and reduced to a single one, which decreases the number of blobs.

In our second experiment we follow the same approach but now we are interested in the dependence on β. Hence, we have performed the simulations with $n \in \{30, 50\}$, $\alpha \in \{0.1, 0.3\}$ and $\beta \in \{0.01, 0.02, 0.05, 0.1, 0.2, 1, 5, 10, 20, 50, 100\}$. The results are presented in Figures 10 and 11. For the number of

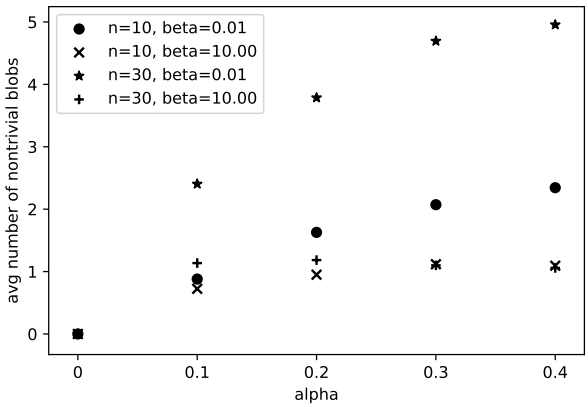

Fig. 9. Dependence on α of the average number of nontrivial blobs of the simulated networks after 500 iterations of the experiment, with different values of the other parameters.

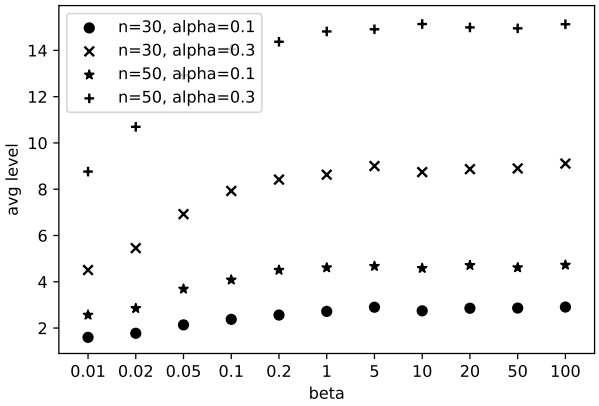

Fig. 10. Dependence on β of the average level of the simulated networks after 500 iterations of the experiment, with different values of the other parameters.

blobs, notice that as β increases, external LGT events are more likely, and hence the resulting networks tend to have fewer but bigger blobs. Notice that this effect stabilizes at $\beta = 1$: the fact that external LGT events are as likely as internal ones makes that after many simulations of events (we take $n = 30$ or $n = 50$) the blobs will get connected and reduced to a single one. As for the level, as we have already argued, when β increases, the blobs get bigger and hence with higher level. Finally, when the number of blobs stabilizes, also does the average level of the generated networks.

VII. Conclusions

In this paper, we have described how generators of level-k networks can be adapted to consider LGT networks and how to obtain, from those, first the set of simple LGT networks and then the full set of networks. We have seen how the complexity of an LGT network can be described by how intricate are their simple components —measured by their level— and the number of such (nontrivial) components. We have introduced a two-parameter stochastic model of evolution that generates LGT networks and made an experimental study of how the aforementioned measures of complexity depend on the parameters of the model.

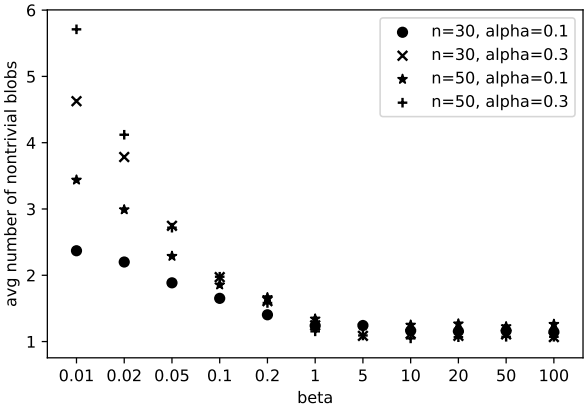

Fig. 11. Dependence on β of the average number of nontrivial blobs of the simulated networks after 500 iterations of the experiment, with different values of the other parameters.

REFERENCES

[Anaya et al., 2016] Anaya, M., Anipchenko-Ulaj, O., Ashfaq, A., Chiu, J., Kaiser, M., Ohsawa, M. S., Owen, M., Pavlechko, E., John, K. S., Suleria, S., et al. (2016). On determining if tree-based networks contain fixed trees. *Bulletin of mathematical biology*, 78(5):961–969.

[Baroni et al., 2006] Baroni, M., Semple, C., and Steel, M. (2006). Hybrids in real time. *Systematic biology*, 55(1):46–56.

[Bordewich and Semple, 2018] Bordewich, M. and Semple, C. (2018). A universal tree-based network with the minimum number of reticulations. *Discrete Applied Mathematics*, 250:357–362.

[Cardona et al., 2008] Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2008). A distance metric for a class of tree-sibling phylogenetic networks. *Bioinformatics*, 24(13):1481–1488.

[Cardona et al., 2015] Cardona, G., Pons, J. C., and Rosselló, F. (2015). A reconstruction problem for a class of phylogenetic networks with lateral gene transfers. *Algorithms for Molecular Biology*, 10(1):1–15.

[Cardona et al., 2009] Cardona, G., Rossello, F., and Valiente, G. (2009). Comparison of tree-child phylogenetic networks. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(4):552–569.

[Cardona and Sánchez, 2012] Cardona, G. and Sánchez, D. (2012). PhyloNetworks: A Python library for phylogenetic networks.

[Choy et al., 2005] Choy, C., Jansson, J., Sadakane, K., and Sung, W.-K. (2005). Computing the maximum agreement of phylogenetic networks. *Theoretical Computer Science*, 335(1):93–107.

[Doolittle and Bapteste, 2007] Doolittle, W. F. and Bapteste, E. (2007). Pattern pluralism and the tree of life hypothesis. *Proceedings of the National Academy of Sciences*, 104(7):2043–2049.

[Francis et al., 2018a] Francis, A., Huber, K. T., and Moulton, V. (2018a). Tree-based unrooted phylogenetic networks. *Bulletin of mathematical biology*, 80(2):404–416.

[Francis et al., 2018b] Francis, A., Semple, C., and Steel, M. (2018b). New characterisations of tree-based networks and proximity measures. *Advances in Applied Mathematics*, 93:93–107.

[Francis and Steel, 2015] Francis, A. R. and Steel, M. (2015). Which phylogenetic networks are merely trees with additional arcs? *Systematic biology*, 64(5):768–777.

[Gambette et al., 2009] Gambette, P., Berry, V., and Paul, C. (2009). The structure of level-k phylogenetic networks. In *Combinatorial Pattern Matching*, pages 289–300. Springer.

[Gusfield, 2014] Gusfield, D. (2014). *ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks*. The MIT Press.

[Gusfield and Bansal, 2005] Gusfield, D. and Bansal, V. (2005). A fundamental decomposition theory for phylogenetic networks and incompatible characters. In Miyano, S., Mesirov, J., Kasif, S., Istrail, S., Pevzner, P. A., and Waterman, M., editors, *Research in Computational Molecular Biology*, pages 217–232, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Huber et al., 2016] Huber, K. T., Moulton, V., Steel, M., and Wu, T. (2016). Folding and unfolding phylogenetic trees and networks. *Journal of mathematical biology*, 73(6-7):1761–1780.

[Huson et al., 2010] Huson, D. H., Rupp, R., and Scornavacca, C. (2010). *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press.

[Jetten and van Iersel, 2018] Jetten, L. and van Iersel, L. (2018). Nonbinary Tree-Based Phylogenetic Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(1):205–217.

[Linz et al., 2007] Linz, S., Radtke, A., and von Haeseler, A. (2007). A Likelihood Framework to Measure Horizontal Gene Transfer. *Molecular Biology and Evolution*, 24(6):1312–1319.

[Pons et al., 2018] Pons, J. C., Semple, C., and Steel, M. (2018). Tree-based networks: characterisations, metrics, and support trees. *Journal of Mathematical Biology*.

[Roch and Snir, 2013] Roch, S. and Snir, S. (2013). Recovering the Treelike Trend of Evolution Despite Extensive Lateral Genetic Transfer: A Probabilistic Analysis. *Journal of Computational Biology*, 20(2):93–112.

[Steel et al., 2013] Steel, M., Linz, S., Huson, D. H., and Sanderson, M. J. (2013). Identifying a species tree subject to random lateral gene transfer. *Journal of Theoretical Biology*, 322:81–93.

[van Iersel et al., 2009a] van Iersel, L., Keijsper, J., Kelk, S., Stougie, L., Hagen, F., and Boekhout, T. (2009a). Constructing level-2 phylogenetic networks from triplets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(4):667–681.

[van Iersel et al., 2009b] van Iersel, L., Kelk, S., and Mnich, M. (2009b). Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks. *Journal of Bioinformatics and Computational Biology*, 7(04):597–623.

Joan-Carles Pons graduated in mathematics in 2010, and received his PhD in mathematics in 2016, all from the University of the Balearic Islands (Spain). He is an Assistant Professor at the Department of Mathematics and Computer Science and member of the Computational Biology and Bioinformatics Research Group at the University of the Balearic Islands.

Celine Scornavacca graduated in mathematical engineering at the University of Roma 2 in 2006, and received the PhD degree in computer science from the Montpellier University in 2009. After a postdoctoral position at the University of Tübingen, she joined the Institut des Sciences de l'Evolution de Montpellier (ISE-M) as a research associate (CNRS), in October 2011. Her research interests are parameterized complexity, supertree and network methods for phylogenetics, combinatorics and bioinformatics.

Gabriel Cardona graduated in mathematics in 1996, in telecommunications engineering in 1997 and received his PhD in mathematics in 2002, all from the Polytechnic University of Catalonia. He is an Associate Professor at the Department of Mathematics and Computer Science and member of the Computational Biology and Bioinformatics Research Group at the University of the Balearic Islands (Spain).