Version dated: January 16, 2019

OSF-Builder: A new tool for constructing and representing evolutionary histories involving introgression

Guillaume E. Scholz[1], Andrei-Alin Popescu[1], Martin I. Taylor[2], Vincent Moulton[1], and Katharina T. Huber[1]

[1]*School of Computing Sciences, University of East Anglia, Norwich, United Kingdom*

[2]*School of Biological Sciences, University of East Anglia, Norwich, United Kingdom*

Corresponding author: Katharina T. Huber, School of Computing Sciences, University of East Anglia, Norwich, United Kingdom; E-mail: k.huber@uea.ac.uk.

Abstract.— Introgression is an evolutionary process which provides an important source of innovation for evolution. Although various methods have been used to detect introgression, very few methods are currently available for constructing evolutionary histories involving introgression. In this paper we propose a new method for constructing such evolutionary histories whose starting point is a species forest (consisting of a collection of lineage trees, usually arising as a collection of clades or monophyletic groups in a species tree), and a gene tree for a specific allele of interest, or allele tree for short. Our method is based on representing introgression in terms of a certain 'overlay' of the allele tree over the lineage trees, called an overlaid species forest (OSF). OSFs are similar to phylogenetic networks although a key difference is that they typically have multiple roots because each monophyletic group in the species tree has a different point of origin. Employing a new model for introgression, we derive an efficient algorithm for building OSFs called OSF-Builder that is guaranteed to return an optimal OSF in the sense that the number of potential introgression events is minimized. As well as using simulations to assess the performance of OSF-Builder, we illustrate its use on a butterfly dataset in which introgression has been previously inferred. The OSF-Builder software is available for download from

https://www.uea.ac.uk/computing/software/OSF-Builder

(Keywords: introgression, allele, lineage, phylogenetic network, OSF-Builder, Fitch-Hartigan algorithm)

Introgression, an evolutionary process in which foreign genetic material (typically some genes) is introduced into a genome, is an important evolutionary process that usually arises through hybridization between species of distinct evolutionary lineages (Goulet *et al.* 2017). It is a widespread phenomenon occurring in both plants and animals and can play a fundamental role in speciation (Mallet 2005). In the case of adaptive introgression, beneficial alleles are transmitted between species (Zhang *et al.* 2016). This has the potential to promote adaptive diversification (Grant and Grant 2016), improve the resilience of endemic species to environmental change (Becker *et al.* 2013), and facilitate range expansion of invasive species (Muhlfeld *et al.* 2014). It has been highlighted as a concern for genetically modified crops that hybridize with their wild relatives (Stewart Jr *et al.* 2003). Perhaps most controversially and interestingly, introgressed Neanderthal alleles may have helped modern humans adapt to non-African environments (Sankararaman *et al.* 2014). Several other examples of introgression are presented in Mallet (2005); Zhang *et al.* (2016) and the references therein.

Most methods for analyzing introgression tend to focus on its detection. For example, the ABBA/BABA test (Sousa and Hey 2013) (see also Martin *et al.* (2015)), which was designed to detect interbreeding between modern humans and Neanderthals (Durand *et al.* 2011), has also been used to detect adaptive introgression between butterflies, lizards, and birds (see Zhang *et al.* (2016) and the references therein). It is a statistical test that is based on a D-statistic and it essentially works by comparing frequencies of rooted tree topologies on four taxa. Using posterior predictive checking, this idea is extended in the JML software (Joly 2012) (see also Joly *et al.* (2009)) to test for the presence of hybridization in multispecies sequence data. Realising that for certain models frequency patterns are in fact phylogenetic invariants, the recently introduced HyDe software tool (Blischak *et al.* 2018) builds on these ideas to detect introgression under the coalescent model with hybridization. However, to date very few methods have been designed to explicitly construct evolutionary histories involving introgression. Notable exceptions to this are Wen *et al.* (2016) in which the PhyloNet software (Than *et al.* 2008) is used to represent introgression between mosquitoes in terms of a phylogenetic network and the SNaQ software (Solis-Lemus and Ané 2016) which employs level-1 phylogenetic networks to capture introgression between swordtails and platyfishes (Xiphophorus: Poeciliidae).

In this paper, we present a new tool to construct and represent evolutionary histories involving introgression. To do this, we introduce the concept of an *Overlaid Species Forest* (or *OSF*, for short), examples of which are presented in Figures 1c and 1d. An OSF is essentially a leaf-labelled multiply rooted directed graph that does not contain directed cycles (see Section Definitions for details). Our definition of OSFs was motivated in part by diagrams such as the one presented in Wallbank *et al.* (2016, Figure 5), which represents introgression of two alleles between *Heliconius* butterfly species. Formally speaking, an OSF may be regarded as a type of phylogenetic network, although it differs from typical phylogenetic networks in that it may have more than one root (as opposed to a single root corresponding to a last common ancestor). This occurs because the construction superimposes an *allele tree* (i. e. a phylogenetic tree constructed for the gene that has potentially introgressed) over lineage trees represented in the species tree, and each of the lineage trees has a different ancestral root node in the species tree.

The lineage trees can be chosen by, for example, considering bootstrap values of branches on the species tree, cutting the species tree at a certain level, or choosing taxa that have some trait of interest (see, e.g., Wallbank *et al.* (2016)). Moreover, at least for the applications we are interested in, the gene tree represents the evolutionary history of a specific allele of interest for the gene in question, and so we call this an *allele tree*.

Using the above terms our main problem can be stated as follows: How should the allele tree be laid over the species forest to indicate the path by which an allele may have introgressed between lineages? Figure 1a shows three lineage trees (indicated in bold branches) that are components of a larger, fully resolved species tree. Figure 1b shows an allele tree. Figure 1c shows an OSF obtained by overlaying the allele tree over three lineage trees pictured in Figure 1a. Figure 1d shows the OSF obtained by overlaying the allele tree over the two lineage trees obtained from the species tree in Figure 1a by deleting the outgoing branches of the root node.

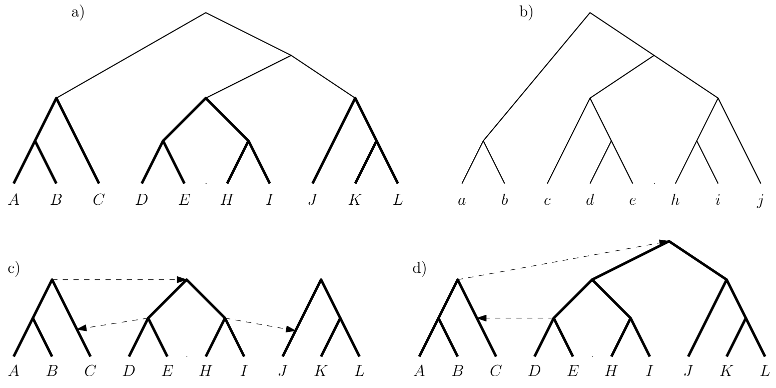

Figure 1: a) A species tree with three 'lineage trees' highlighted in bold. b) An allele tree, with a leaf labelled by the same letter as the species in which it is contained. c) An OSF constructed by overlaying the allele tree on the lineage trees $((A, B), C)$, $((D, E), (H, I))$ and $(J, (K, L))$. d) An OSF constructed by overlaying the allele tree over the two lineage trees $((A, B), C)$ and $(((D, E), (H, I)), (J, (K, L)))$.

A key feature of our approach is that our starting point is a *species forest* and an allele tree and that neither that tree nor the trees in that forest are assumed or required to be fully resolved. In practice, this forest will usually be a collection of lineage trees (i. e. clades or monophyletic groups – see, e. g. , Figure 1a) from a reconstructed species tree that might not be fully resolved. This has an advantage when studying introgression because obtaining fully resolved species trees can be problematic (see, e. g. , (Ge *et al.* 1999)).

The problem of constructing OSFs is closely related to that of reconciling a gene tree with a species tree, which is extensively studied in the literature (see, e.g., Doyon *et al.*

(2011)). The key methodological distinction between approaches developed for tackling this problem and our method is the replacement of the species tree by a species forest. This is critical since, in essence, the species forest forces extra constraints (namely that introgression must occur between and not within the specified lineages) which would not necessarily be respected in a reconciliation approach. To illustrate this difference, consider again the species tree and allele tree in Figure 1a and b. Then using the definition of a reconciliation of a gene tree with a species tree sensu Tofigh *et al.* (2011) (see also e. g. Huber *et al.* (2018)) it follows that the network depicted in Figure 2d is a reconciliation for them. In particular, and contrary to the introgression scenario pictured in Figure 2c for that allele tree and the obtained species forest in Figure 2b, that reconciliation scenario postulates two duplication events at the root of the network and one horizontal gene transfer event (indicated by a dashed branch).

Our method for building an OSF essentially works as follows. Starting with a species forest F and an allele tree G, we use a bottom-up approach to add in branches, which we call *contact arcs*, between different lineage trees of F so that the resulting graph simultaneously displays G and all the lineage trees in F. If there is discordance between the species forest and G, then we interpret contact arcs as indicators of potential gene flow between the lineage trees. We also employ a new model for introgression which captures the idea that an allele can spontaneously arise within a lineage tree and that introgression is infrequent. This is formalized by allowing the root of the allele tree G to be mapped to any node of a lineage tree (i.e. not necessarily its root) of F. In addition, we invoke parsimony so as to minimize the number of contact arcs necessary to reconcile G with the trees in F. Adapting the Fitch-Hartigan algorithm (Fitch 1971) enables us to then find an *optimal* OSF (i.e., an OSF with a minimal number of contact arcs) relative to our model.

We now summarize the contents of the rest of the paper. In the next section, we present our model for introgression and the OSF-Builder algorithm. This section also includes a brief discussion of theoretical properties of OSF-Builder and its usage. We describe a simulation study used to investigate the performance of OSF-Builder in the case where there is uncertainty in either the allele tree or the species forest. To illustrate the use of OSF-Builder, we apply it in the Results section to a Heliconius butterfly dataset from Wallbank *et al.* (2016) and Kozak *et al.* (2015), in which there is evidence for introgression having played a role in wing pattern evolution. We also present the results from the simulation study. We conclude with a brief discussion of some possible future directions.

Methods

An Evolutionary Model for Introgression

We begin by introducing our model for introgression. The starting point for our model is the observation that in many introgression studies lineage information is available for the species under consideration and that alleles from one lineage are distinct from alleles from another lineage. In the case of introgression, lineages can carry alleles specific to other lineages, making these useful indicators for introgression. A useful way to represent introgression by a graph is to add lineage tree connecting branches to the species forest so that the information provided by both the lineage trees and the allele tree is displayed in a single structure.

Model assumptions.— To obtain our model, we make the following additional biologically motivated assumptions.

(M1) Introgression of an allele being studied is an infrequent event.

(M2) An allele can only originate in one lineage.

(M3) If an allele has introgressed from one lineage into a different lineage then it cannot introgress back into the first lineage unless the start of the first introgression event predates the end of the second one (i.e., we do not allow time inconsistent events).

(M4) The allele composition of a species x comprises an allele inherited by descent that uniquely identifies species x and all the alleles that species x has obtained via introgression events (in particular, we do not allow for loss of alleles).

(M5) The only other permissible evolutionary events are allele sequence divergence accompanying speciation and whole genome duplication.

Assumption (M1) motivates the use of a parsimony framework for modelling introgression, Assumption (M2) is motivated by the observation that lineages are sometimes identifiable from a specific allele, and Assumption (M3) reflects time consistency. Assumption (M4) captures the idea that a species can carry lineage-specific and lineage-foreign alleles.

Definitions

We now present some definitions that we will use to formalize the model presented in the previous section. As mentioned in the Introduction, we will use directed graphs as a tool to vizualize OSFs. For a directed graph H, denote by $V(H)$ the set of nodes of H. To emphasize that the branches of H have a direction, we refer to them as *arcs* rather than branches. If v is a node of H then we refer to the number of arcs coming into v as the *indegree* of v and the number of arcs leaving v as the *outdegree* of v. If the outdegree of v is zero then we call v a *leaf* of H. If the indegree of v is 0 we call v a *root*. We denote the set of leaves of H by $L(H)$. For example, in the graph in Figure 1c, the indegree of the node at the top of the figure to where the top dashed arrow points is zero, and its outdegree is two. The node labelled D in that graph is a leaf and the set of leaves of that graph is $\{A, \ldots, E, H \ldots, L\}$.

A *(rooted) phylogenetic tree* T on a set X of taxa is a rooted directed tree with leaf set X that has precisely one node of indegree 0, and no nodes that simultaneously have indegree one and outdegree one. If T is such a tree, and u and v are nodes of T such that there exists a directed path from u to v in T, then we say that u is an *ancestor* of v in T. Note that $u = v$ might hold. If v is a leaf of T that has u as an ancestor then we call v an *offspring (taxon)* of u and if v is an ancestor of u that is directly above u then we call v a *parent* of u. In that case we also call u a *child* of v. For example, the tree in Figure 1a is a phylogenetic tree on $\{A, \dots, E, H, \dots L\}$. The right arc emanating from its root (i.e. the top node) ends in an ancestor of $D, E, H, \dots, L$ and all seven of them are offspring taxa of that node.

We call a phylogenetic tree representing a clade a *lineage tree* and a phylogenetic tree representing the evolutionary history of an allele of a gene an *allele tree*. Note that the leaves of a lineage tree are species whereas the leaves of an allele tree are not. We also call a collection of one or more lineage trees on pairwise distinct leaf sets a *species forest*. Note that the node set (leaf set) of a forest is the set of all nodes (leaves) in its lineage trees. We refer to a map that assigns a leaf a of the allele tree to the species in the forest that carries a as a *leaf-map*. To facilitate readability, we always assume that a leaf of the allele tree is assigned to the leaf of the species forest by the leaf map that is labelled by the capitalization of that letter (i.e., a maps to A in this case).

We say that a phylogenetic tree T is *displayed* by an OSF if T is either the allele tree or one of the trees in the species forest from which the OSF was constructed. Continuing with the example in Figure 1, the two trees in Figure 1d obtained by deleting the two dashed arrows are lineage trees that, taken together, form a species forest. The node set of that forest comprises the nodes in both trees and its leaves are $A, \dots, E$, and $H, \dots, L$. The leaf set of the allele tree is $\{a, \dots, e, h, i, j\}$ and the leaf-map assigns leaf a to species A, leaf b to species B, and so on.

Overlaid Species Forests

We now formalize our model. We assume that we are given a species forest F (i.e., a collection of lineage trees), an allele tree G, and a leaf-map ϕ for G and F. We shall call a map ψ from the node set of G to the node set of F an *Overlaid Species Forest* or *OSF*, for short, for F and G (and ϕ) if ψ satisfies the following three properties:

(P1) When restricted to the leaves of G the maps ψ and ϕ coincide.

(P2) If u and v are two nodes of G such that u is an ancestor of v in G, and the nodes $\psi(u)$ and $\psi(v)$ belong to the same lineage tree in F, then $\psi(u)$ is an ancestor of $\psi(v)$ in that tree.

(P3) For any node v of G there exists an offspring taxon h of v such that species $\phi(h)$ and $\psi(v)$ are nodes in the same lineage tree in F.

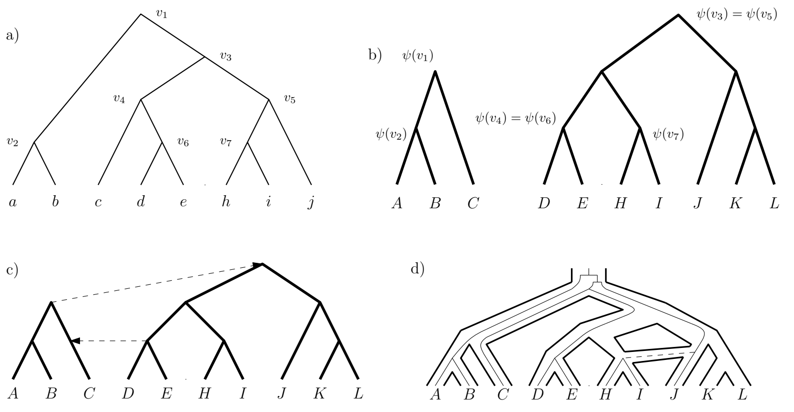
a)
v_1
v_3
v_4
v_5
v_2
v_6
v_7
a b c d e h i j
b)
$\psi(v_1)$
$\psi(v_2)$
$\psi(v_4) = \psi(v_6)$
$\psi(v_3) = \psi(v_5)$
$\psi(v_7)$
A B C D E H I J K L
c)
A B C D E H I J K L
d)
A B C D E H I J K L

Figure 2: a) The allele tree from Figure 1b with interior nodes labelled $v_1, \ldots, v_7$. b) A species forest obtained from the species tree in Figure 1a in which an OSF ψ for both is indicated in terms of the images of $v_1, \ldots, v_7$ under ψ. c) A depiction of ψ in terms of the OSF in Figure 1d. d) A reconciliation in terms of a phylogenetic network for the allele tree in Figure 1b and the species tree in Figure 1a. The two duplication events are indicated by horizontal bars and the horizontal gene transfer event is indicated by a dashed branch.

For example, the map ψ for the OSF in Figure 2c is given in Figure 2b.

It can be shown that we can always represent an OSF by a multiply rooted, directed acyclic graph H by adding contact arcs connecting distinct lineage trees such that H displays the given allele tree. For simplicity we also consider an OSF to be such an object (so Figure 2c and Figure 2b are equivalent).

Note that in the case in which the species forest only contains one lineage tree, Property (P2) implies that the problem of constructing an OSF boils down to finding a reconciliation map for a lineage tree and a gene tree under parsimony where the only two permissible evolutionary events are divergence and duplication (see, e.g., Baudet *et al.* (2015)). In this case, the OSF does not have any contact arcs. Also note that the map that assigns a leaf of an allele tree to its corresponding species in a species forest and all its other nodes to the root of an arbitrarily chosen lineage tree in the forest is an OSF. Consequently, for any allele tree and any species forest there always exists an OSF. However it is straight-forward to see that that OSF is not necessarily minimal, i.e., there might exist OSFs that postulate fewer contact arcs for that allele tree and species forest.

The OSF-BUILDER *Algorithm*

To minimize the number of contact arcs when we construct an OSF, we adapt the well-known *Fitch-Hartigan algorithm* (Fitch 1971). For a given phylogenetic tree T on some set X and a sequence alignment for X, this algorithm is used to compute an optimal labelling of the interior notes of T in terms of the character states or alphabet that make up the sequences.

To build an OSF, rather than assigning character states to the nodes of a phylogenetic tree, we assign the lineage trees to the nodes of an allele tree. To make this more precise, suppose F is a species forest, G is an allele tree, and ϕ is a leaf-map that relates their leaf sets. Then, in the bottom-up step of the Fitch-Hartigan algorithm we record for every node v of G all potential trees of F that could accommodate v by an OSF for F and G. Once this is done for all nodes of G, we use the top-down step of that algorithm to decide for every node v in G which one of the potential trees recorded for v to choose. Using a modification of the 'lca-map' which assigns to a collection of leaves of a phylogenetic tree its last common ancestor, we then identify the nodes in the species forest to which contact arcs are added.

The Algorithm.— We now give a more detailed description of the OSF-Builder algorithm. We refer to the next section for a worked example and the Supplementary Material (see doi:10.5061/dryad.1nt8t75) for a proof of its correctness. OSF-Builder takes as input a species forest F, an allele tree G and a leaf-map ϕ from G to F. Put $\mathcal{F} = (F, G, \phi)$. Then OSF-Builder finds an overlaid species forest $\psi = \psi_{\mathcal{F}} : V(G) \to V(F)$ by adding a minimal number of contact arcs connecting distinct trees in F such that Properties (P1) – (P3) are satisfied by the associated multiply rooted directed acyclic graph.

OSF-Builder works by first initializing each leaf w of G with the lineage tree P_w in F that contains the species associated to w via ϕ in its leaf set (Step 0). Referring to this assignment as a map P from the leaves of G into the trees of F defined by putting $P(w) = P_w$, the algorithm then applies the Fitch-Hartigan algorithm to G and F to find an extension $\overline{P}$ of P to a map from the nodes of G to the trees in F (Step 1). The purpose of this map is to select for all nodes v of G, a lineage tree among all lineage trees that could potentially accommodate v. We use this information as the starting point for our construction of the contact arcs. Before presenting details concerning this construction we remark that by virtue of the Fitch-Hartigan algorithm, the number of contact arcs that we add to the species forest to obtain an OSF for F and G is as small as possible.

To complete the construction of the contact arcs, OSF-Builder considers each interior node v of G in turn to obtain the node in the lineage tree $\overline{P}(v)$ that v is assigned to via ψ. More precisely, OSF-Builder first associates to every node v of G the set U_v of all species in F that carry an offspring taxon of v. After this, it considers the subset U'_v of species in U_v that are also contained in the leaf set of lineage tree $\overline{P}(v)$ (i.e., $U'_v = U_v \cap L(\overline{P}(v))$). Note that it can be shown that the set U'_v must contain at least one species of F. Next, the last common ancestor of the species in U'_v with regards to the tree $\overline{P}(v)$ is found by OSF-Builder and taken to be the node that v is mapped to under ψ. To ensure that no contact arcs are between leaves of F which would affect readability, all leaves that are involved in a contact arc are pushed out in the final post-processing phase of the algorithm.

An illustrative example

Using the lineage trees and allele tree pictured in Figures 3a, we now explain how OSF-Builder generates the OSF depicted in Figure 3d so as to illustrate how the algorithm works. Let G denote the allele tree pictured in Figure 3a(bottom) and let ϕ denote the leaf-map for F and G, that is, an assignment of the taxa in G to the species in F. Then OSF-Builder employs the bottom-up step of the Fitch-Hartigan algorithm to recursively assign to each node v of G all lineage trees that could potentially accommodate v as follows. Starting with the taxa of G and considering each taxon in turn OSF-Builder assigns to each taxon v the lineage tree that contains the species that v is mapped to under ϕ (Step 0). This is the tree $\overline{P}(v)$. Now assume that v is a node in G such that the assignment has already been made for all children of v. Let S comprise all lineage trees in F that can accommodate a maximum of the children of v. Then OSF-Builder assigns the lineage trees in S to v as lineage trees that can potentially accommodate v.

We illustrate this assignment for all non-leaf nodes of G in Figure 3b. For example, the tree L_2 is assigned to node w_1 by OSF-Builder because both children of that node can be accommodated by lineage tree L_2, that is, are mapped to a species in L_2 by our leaf-map. In contrast, because one child of node v_1 has been assigned to lineage tree L_1 whereas the other has been assigned to lineage tree L_2, OSF-Builder marks L_1 and L_2 as potential candidates for accommodating v_1 and defers making a choice between them until the top-down step of the Fitch-Hartigan algorithm. Once assignments of allele tree nodes have been made in the bottom-up step, OSF-Builder carries out the top-down step of the Fitch-Hartigan algorithm. Starting with the root of G, OSF-Builder selects a lineage tree amongst all possible lineage trees that can accommodate that root. Now assume that v is a node in G for which a lineage tree has been selected by OSF-Builder in its top-down phase. Then if u is a child of v that can also be accommodated by that tree OSF-Builder also selects that tree for u. If u cannot be accommodated by that tree then OSF-Builder selects one of the trees that can accommodate u, breaking ties in favour of one of them – see the Results and Discussion Sections for more on this.

We illustrate that step for the non-leaf nodes of G in Figure 3c. Since both L_1 and L_2 can accommodate the root ρ of G, OSF-Builder arbitrarily breaks this tie in favour of L_1. This is the tree $\overline{P}(\rho)$ selected for ρ by OSF-Builder. Since each child of ρ was assigned a unique lineage tree in the bottom-up phase, OSF-Builder selects that tree for each child. In the case of the child that is an ancestor of leaf b that tree is L_1. For the child labelled u_1 that tree is L_2 implying that the tree $\overline{P}(u_1)$ is L_2. To minimize the number of contact arcs and also break the tie as to which tree to assign to the children of u_1, OSF-Builder chooses L_2 as both children can be accommodated by that tree. Thus, the tree assigned to both of them via $\overline{P}$ is L_2. Since w_1 is a child of v_1 and w_1 can be accommodated by L_2, OSF-Builder also chooses L_2 for w_1. Thus, the tree $\overline{P}(w_1)$ is again L_2. Applying the same rational to the remaining non-leaf node of G as for w_1 implies that OSF-Builder chooses L_2 for that node too. The resulting assignment of lineage trees to the nodes of G is the map $\overline{P}$ in the previous section.

Since G contains three branches for which the lineage trees assigned to their end nodes differ, i.e., the branches ending in leaves c and j, respectively, and the branch coming into u_1, it follows that the OSF generated by OSF-BUILDER must have three contact arcs. To find the nodes in the forest connected by them, OSF-BUILDER considers every interior node v of G in turn to find the node in the tree $\overline{P}(v)$ assigned to v. To do this, OSF-BUILDER first finds all offspring taxa of v that are accommodated by the same lineage tree as v. These taxa make up the set U'_v. Then it finds the last common ancestor in $\overline{P}(v)$ of the taxa in U'_v. This ancestor is then assigned by OSF-BUILDER to v. For example, since only the offspring taxa d, e, h, and i of node u_1 are mapped to species contained in the same lineage tree as u_1 (i.e., L_2), the set U'_{u_1} comprises the species D, E, H, and I only. Since their last common ancestor in L_2 is the root of L_2, that node is assigned to u_1 by OSF-BUILDER. Note that since the node c of G is assigned to species C via this process, the arc leading to C is subdivided by a new node and the contact arc is attached to that node (rather than C) in OSF-BUILDER's post-processing phase.

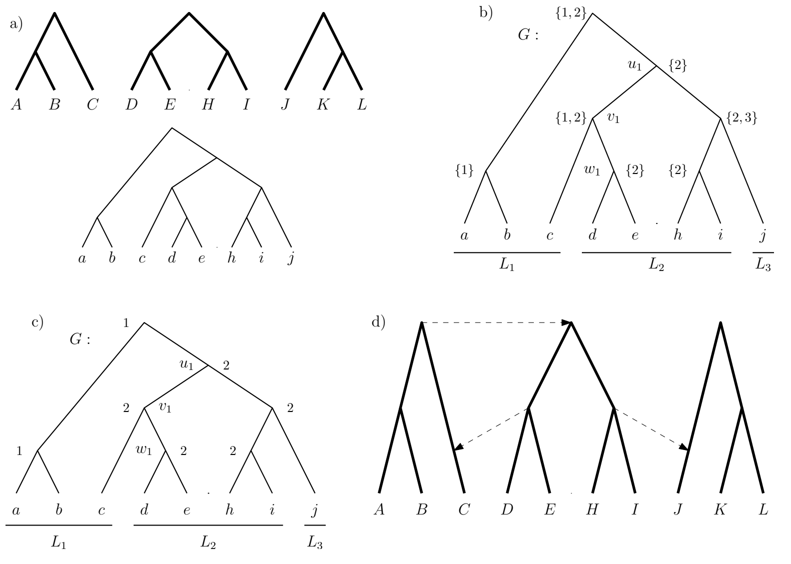

Figure 3: a) The species forest indicated in Figure 1a given in bold and the allele tree in Figure 1b. b) The bottom-up phase of the Fitch-Hartigan part of the OSF-BUILDER algorithm applied to the set of lineage trees and the allele tree in a). The leaf set corresponding to each of the three lineage trees L_1, L_2, and L_3 is indicated by a horizontal line below the leaves. For ease of readability inside the allele tree, we denote, for all $i = 1, 2, 3$, a lineage tree L_i assigned to a non-leaf node by i. c) Employing the same notation as in b), the top-down phase of the Fitch-Hartigan part of the OSF-BUILDER algorithm. d) The resulting OSF (which is also the OSF in Figure 1c.)

Properties and Usage of OSF-BUILDER

The OSF-Builder algorithm has some attractive theoretical properties. First, OSF-Builder will always find an OSF that minimizes the number of contact arcs, for any given input (Supplementary Material, Lemma 1 and Theorem 2). Furthermore, if F and G are the species forest and the allele tree obtained from that OSF by deleting its contact arcs, then OSF-Builder will recover that OSF from F and G provided the information is available as to how ties were resolved.

It should be noted however that the OSF need not be unique, and also that there might be OSFs that cannot be found by OSF-Builder (see, e.g., Supplementary Material, Figure 1). Another property of OSF-Builder is that, for any subtree of an allele tree, an OSF found by OSF-Builder for the allele tree always restricts to an optimal OSF for that subtree. Thus, even if an allele tree G is only known in parts (i.e. in terms of a subtree G'), then the OSF returned by the algorithm for G' and a species forest can be employed as a basis for finding an optimal OSF for G and that forest. However it should be noted that this OSF might not be found by OSF-Builder.

We have implemented OSF-Builder in Python 2 (see Supplementary Material for details) and it is freely available from the website given in the abstract. It takes as input a triple consisting of a species forest and an allele tree (all in newick format) and a leaf map for them, and generates an OSF in the form of a multiply rooted directed graph. This graph is currently returned by OSF-Builder as a .gve file and also in e-newick notation (Morin and Moret 2006). The .gve file can be represented using the freely available Graphviz software (Ganser and North 2000), which we used to generate Figure 5. For technical reasons, in case the e-newick format is chosen, we artificially turn the found OSF into a phylogenetic network by arbitrarily joining all roots of the OSF with a new single root. This can then be read by popular software packages such as Dendroscope (Huson and Scornavacca 2012) and manually post-processed to obtain the OSF found by our approach by removing the arbitrarily placed arcs and the root.

Simulations

In this section, we describe the methodology behind our simulation studies which we use to assess the performance of OSF-Builder in the case of uncertainty either in the allele tree or the species forest. We refer to Figure 4 for a schematic overview. Essentially, we begin by first randomly generating a species forest and an allele tree together with a minimal OSF for this pair (called OSF_1). We then perturb the allele tree by applying subtree-prune-and-regraft (SPR) operations in order to simulate inaccurate estimates of this tree within the input to the OSF-Builder algorithm. SPR-operations are widely used in phylogenetics to explore tree space and amount to pruning a subtree of a given tree and regrafting that subtree elsewhere onto the resulting tree (see, e.g., Semple and Steel (2003) for a formal definition). By applying varying numbers of SPR-operations we can simulate trees that are close to (few operations) or far from (many operations) the allele tree. After this, we run our algorithm on the original species forest and perturbed allele tree to obtain an OSF (called OSF_2), and we measure how close OSF_2 is to OSF_1. Following the same protocol, we also use SPR-operations to introduce uncertainty in the species forest.

We now provide more details. Using the software SimCoal (Excoffier *et al.* 2000), we first simulate a phylogenetic tree with 100 leaves. Using this tree we then generate a species forest F containing seven lineage trees by randomly deleting non-pendant arcs, and checking that this selection gives a valid collection of phylogenetic trees. To generate an OSF, which we call OSF_1, we randomly add eight contact arcs in-between the lineage trees in F by connecting nodes between pairs of trees, ensuring that we do not create a directed cycle (cf. Figure 2 in the Supplementary Material). We then also create an allele tree G from this OSF by randomly picking a root and taking the tree with leaves in F that can be reached from that root via directed paths. We also ensure that running our algorithm on the pair F and G gives OSF_1.

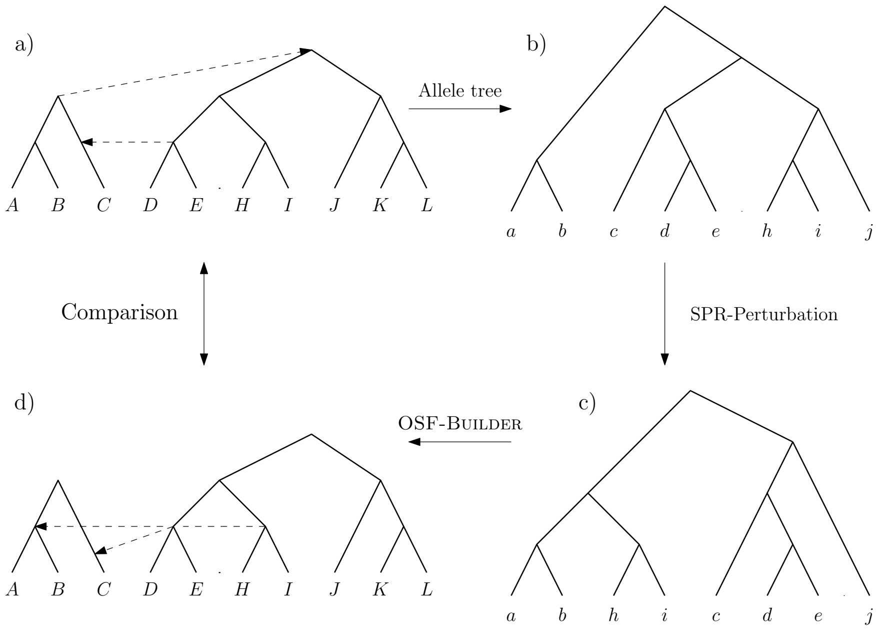

Figure 4: A schematic outline of our simulation study which models of uncertainty in the input allele tree. a) OSF_1 and b) An allele tree G. Note that to ensure that G is a valid input allele tree for OSF-Builder we derive it from OSF_1. c) The allele tree G' obtained from G by application of one SPR operation to G. d) The OSF OSF_2 generated by OSF-Builder when given G' and F as input.

To simulate uncertainty in the input data, we apply 1, 3 or 5 random SPR-operations to the allele tree G, one after the other, to generate a perturbed allele tree G'. In the case of the species forest, we proceed as for the allele tree ensuring that a pruned subtree is not regrafted to the same tree from which it was pruned. The reason for the latter is that such forests would suggest introgression events within a lineage which is not permitted by our model.

We then use G', F and G, F' (with implicit leaf maps) as input to OSF-Builder to obtain an OSF which we denote by OSF_2. For each case, we repeated this process 100 times to obtain a distribution. To measure how close OSF_2 was to OSF_1, we compute the difference between the number of contact arcs.

Results

In this section, we present our findings for a biological dataset and our simulation study. Unless specified otherwise, we refer to an optimal OSF (in the sense of postulating a minimal number of contact arcs) as just an OSF.

The Heliconius Butterflies Dataset

In The Heliconius Genome Consortium (2012) evidence was presented for introgression having played a role in the evolution of wing pattern in *Heliconius* butterfly species. To investigate this further Wallbank *et al.* (2016) studied the evolutionary relationships between *Heliconius* butterfly species based on the so-called dennis and ray alleles of the *optix* gene, which are known to be implicated in wing pattern production. They determined that the dennis allele introgressed from *Heliconius melpomene* into *Heliconius timareta* and from an ancestor of *Heliconius luciana*, *Heliconius pardalinus* and *Heliconius elevatus* into *Heliconius melpomene*, and that the ray allele introgressed from *H.melpomene* into *H.timareta*, and also into *H.elevatus* (Wallbank *et al.* 2016, Fig. 5).

To run OSF-Builder on this data we obtained a species forest and allele tree as follows. For the allele tree, we used the tree constructed for the dennis allele in Wallbank *et al.* (2016) (see Supplementary Material for the link to dryad). To obtain the species forest we used the species phylogeny presented in Kozak *et al.* (2015, Fig. 1) containing ten *Heliconius* butterfly lineages. Out of these ten lineages, only five contain species carrying the dennis allele, we thus only considered these five for our study. Amongst these, the *melpomene* and *silvaniforms* lineages are studied in more detail in Wallbank *et al.* (2016, Fig. 4). In particular, both contain more species than their counterparts in Kozak *et al.* (2015), and lineage *melpomene* is split into two distinct sublineages *melpomene* and *timareta*. We incorporated this additional information into our species forest, and, since the *elevatus* sublineage of the *silvaniforms* lineage plays a key role in introgression, we made it into a lineage tree in it own right. The resulting species forest therefore has 7 lineages.

We depict the OSF generated by OSF-builder in Figure 5. We used the relative ages of the lineages as indicated by their underlying species phylogeny (Kozak *et al.* 2015, Fig. 1) to resolve ties. As can be seen, all four of the aforementioned introgression events were identified by OSF-Builder. Furthermore, Figure 5 suggests that introgression between these lineages did not only occur once but multiple times including backcrossing and that the dennis allele might have introgressed into *H.melpomene* via *H.timareta*. The former is interesting given that Wallbank *et al.* (2016) raised the question as to whether multiple introgression events might have occurred between these lineages. We repeated this analysis using the tree constructed for the ray allele in Wallbank *et al.* (2016), and obtained similar results (OSF presented in the Supplementary Material).

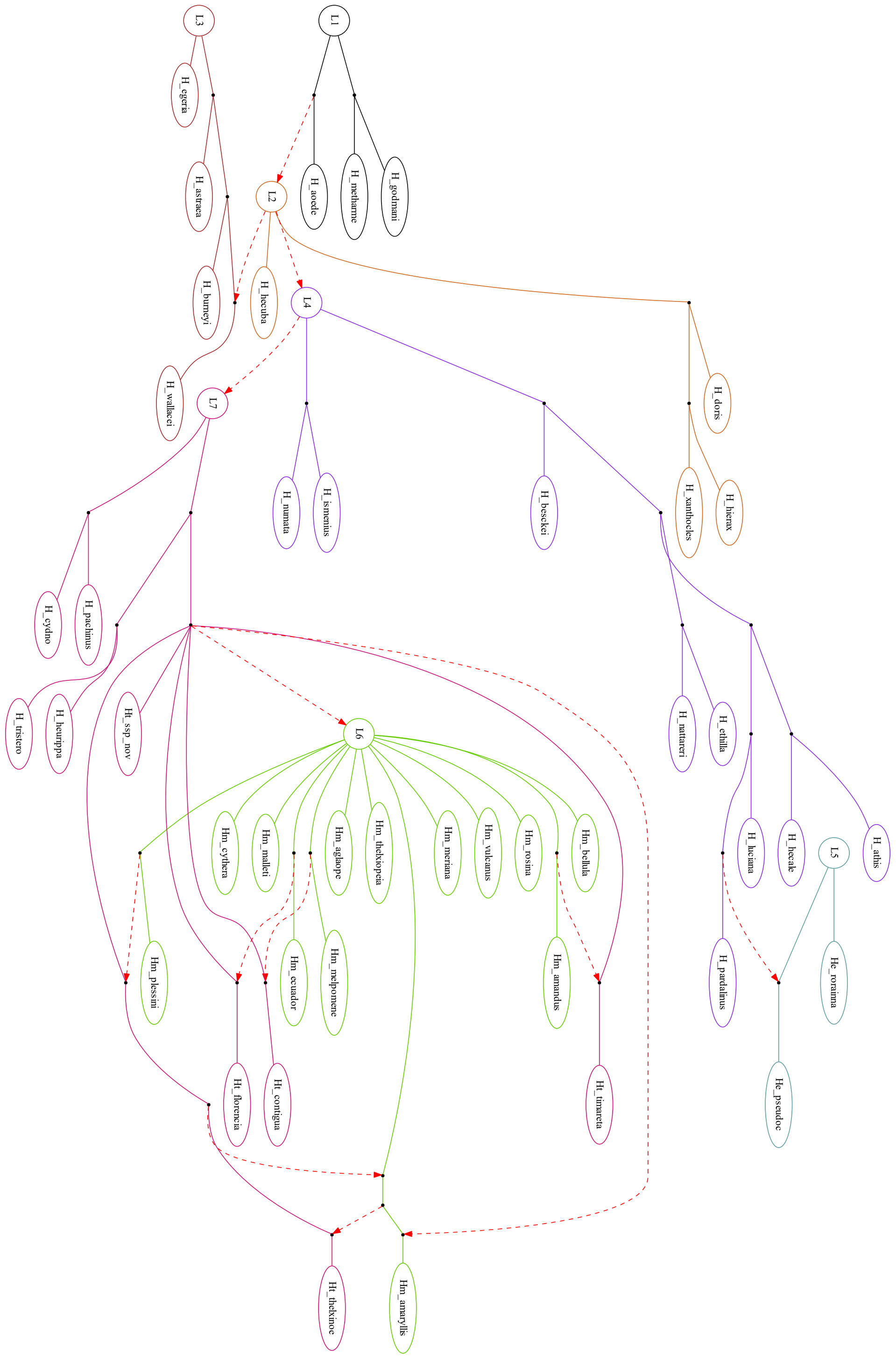

Figure 5: For the Heliconius butterfly dataset from Wallbank *et al.* (2016) and Kozak *et al.* (2015), we picture the OSF found by OSF-Builder for the dennis allele of the optix gene. Each lineage tree is rooted by a node of the form Li where $1 \leq i \leq 7$. In the online version, we represent each of them by a different colour. Potential introgression events are indicated as dashed arrows.

Given that OSF-Builder might have had to resolve ties in the construction of the depicted OSF, we also investigated what effect the approach that we used in this example for tie resolution might have had on its structure. For this we considered all possible ways in which ties could be resolved in this dataset (random sampling from the set of all possible ways to resolve ties could be used for larger datasets). We depict our findings in Figure 6a in terms of a heat map generated by measuring the number of times a contact arc occurred in an OSF and normalized over all OSFs to obtain a proportion. These proportions were used as input to the *Heatmapper* software (Babicki *et al.* 2016) where higher proportions are represented as darker cells. In particular, a dark cell in row i and column j indicates that a high proportion of the OSFs had a contact arc from lineage tree L_i to lineage tree L_j. In Figure 6b, we present an augmented version of the species tree in (Kozak *et al.* 2015, Fig. 1) in which we have added in all contact arcs that are in more than 50% of the OSFs found by OSF-Builder. Due to the size of the species tree we have shrunk the lineages down to leaves.

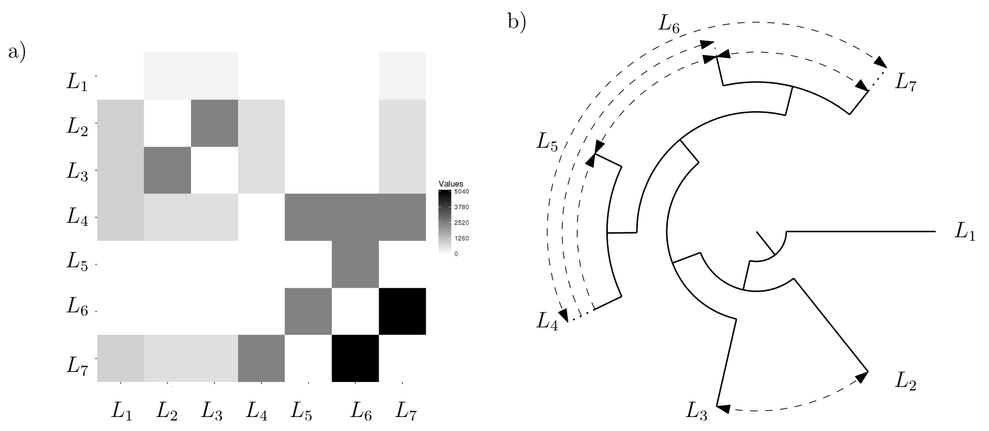

Figure 6: Heliconius butterflies dataset. (a) Heat map representing the proportion of OSFs containing contact arcs found over all possible ways to resolve ties in the top-down phase of the Fitch-Hartigan algorithm. (b) A simplified version of the *Heliconius* butterflies species tree in which we have added all contact arcs (dashed arrows) that were contained in more than 50% of the OSFs found. All seven lineage trees have been shrunk to single leaves and double arrows indicate contact arcs going both ways.

Interestingly, out of the $42 = 7 \times 6$ possible contact arcs between the seven lineage trees we only observe 26 different ones and out of these only 10 occur more than half the time. Out of those 10, the contact arcs from lineage *H.melpomene* (L_6) to its sister lineage *H.timareta* (L_7) and from L_7 to L_6 were recovered every time suggesting that there is strong signal in the data for introgression between these two lineages. The remaining contact arcs which appear a high proportion of times are quite symmetric (in the sense that introgression is supported strongly in both directions between the lineages). A notable exception to this is the *silvaniforms* lineage (L_4), which was the source of a high proportion of contact arcs ending in lineages L_5 and L_6 although no contacts arcs were found with either of these lineages as a source and ending in lineage L_4.

Effect of Input Perturbations on OSF-Builder

To study the effect of uncertainty in the input data we simulated two scenarios as described in Methods. We start with the case where the species forest F is fixed and the allele tree G is varied. We let $n_{(F,G)}$ denote the number of contact arcs of an OSF for a species forest F and an allele tree G.

In Figure 7, we present the distribution of the differences $n_{(F,G')} - n_{(F,G)}$ between the true OSF_1 and the computed OSF_2 when the species forest F is fixed and the allele tree G is varied to give a perturbed tree G'. Note that in case the OSF for F and G' has fewer contact arcs than the OSF for F and G then that difference is negative. As explained in Methods, G' is obtained from G by applying 1, 3 or 5 SPR operations. As might be expected, we see that more SPR operations result in a greater difference in contact arcs between OSF_1 and OSF_2. Having said this, in the majority of the cases and largely irrespective of the considered number of operations, this difference is at most one contact arc. Furthermore, Figure 8 suggests that if G' is not too dissimilar from the true allele tree then 60% of the OSFs constructed by OSF-Builder contained all 8 contact arcs of OSF_1 and, reassuringly, more than 90% contained at least 7 of them.

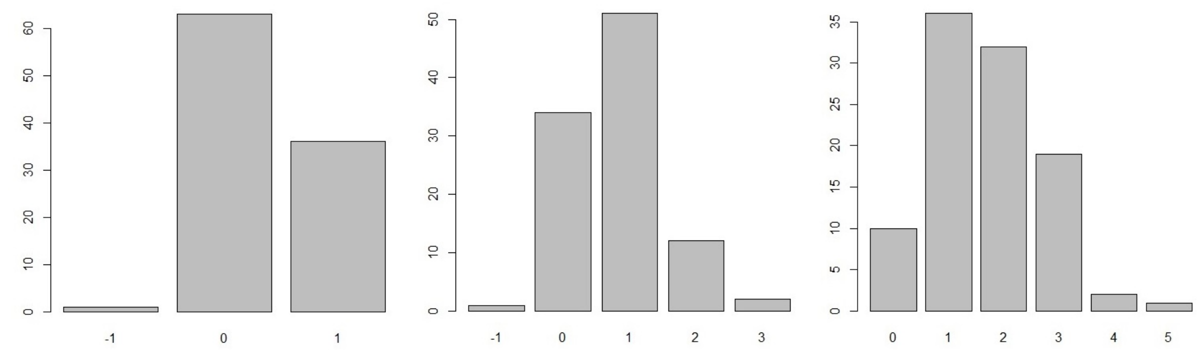

Figure 7: For F and G the true species forest and allele tree, respectively, and G' the perturbed allele tree, we depict the distribution of the difference $n_{(F,G')} - n_{(F,G)}$ in the number of contact arcs over 100 runs for 1 (left), 3 (middle), and 5 (right) SPR-operations (see Methods for details). The x-axis is labelled by $n_{(F,G')} - n_{(F,G)}$ and the y-axis gives the percentage of times that a difference is observed. Note that the value of -1 on the x-axis means that G' has one contact arc less than G.

We now turn our attention to the case where the allele tree is fixed and the uncertainty affects the species forest. We depict in Figure 9 the distribution of the differences $n_{(F',G)} - n_{(F,G)}$, where F' is a forest obtained from F by applying 1, 3, or 5 SPR-operations to the lineage trees in F, respectively. The distributions highlighted by Figure 9 are similar to the ones observed in Figure 7 for the allele tree, in the sense that the number of SPR-operations directly influences the number of contact arcs. Put

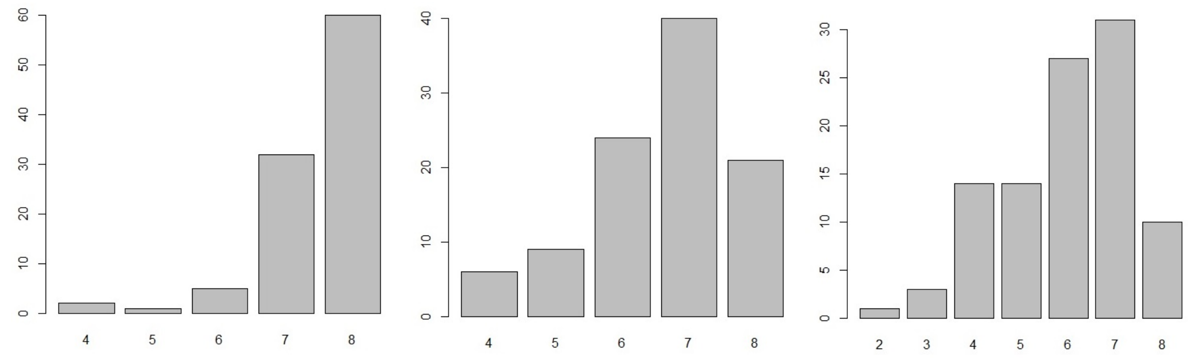

Figure 8: For F and G the true species forest and allele tree, respectively, and G' the perturbed allele tree, we depict the distribution of the number of original contact arcs recovered, over 100 runs, for 1 (left), 3 (middle), and 5 (right) SPR-operations. The x-axis is labelled by the number of contact arcs and the y-axis gives the percentage of times that a number was observed. For ease of readability, contact arc numbers which were not observed are omitted.

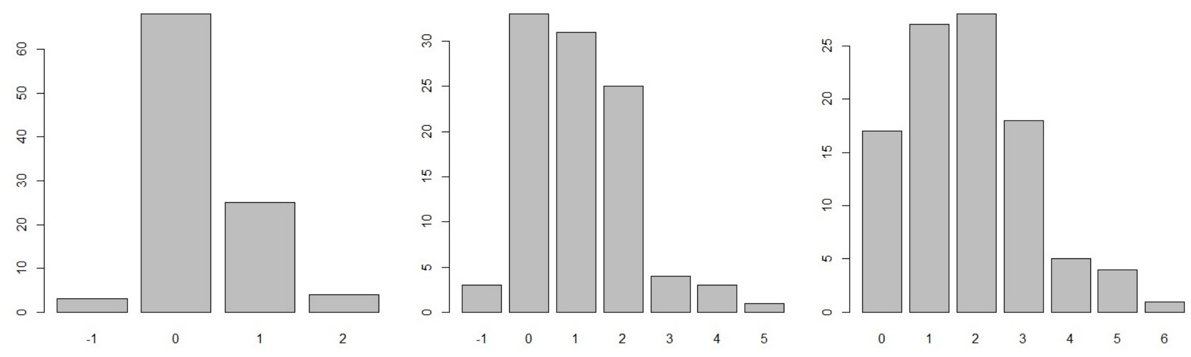

Figure 9: For F and G the true species forest and allele tree, respectively, and F' the perturbed species forest, we depict the distribution of the difference $n_{(F',G)} - n_{(F,G)}$ in the number of contact arcs over 100 runs for 1 (left), 3 (middle), and 5 (right) SPR-operations. The labelling of the axis is as in Figure 7.

differently, the larger the number of SPR-operations is between the lineage trees in F' and the lineage trees in F the more the species forests differ in their number of contact arcs.

Taken together, Figures 7 and 9 indicate a high percentage of cases where the true OSF had the same number of contact arcs as OSF_2, and this is independent of the number of SPR-operations performed. Furthermore, the difference $n_{(F',G)} - n_{(F,G)}$ when perturbing the species forest is higher than the difference $n_{(F,G')} - n_{(F,G)}$ when perturbing the allele tree using the same number of SPR-operations. In other words, it seems that uncertainty in the species forest has a greater effect on OSF-Builder's ability to recover an OSF than uncertainty in an allele tree. Having said this, in cases where uncertainty is low OSF-Builder might provide a attractive tool for shedding light into introgression independent of whether this uncertainty affects the allele tree or the lineage trees.

Discussion

We have presented a novel approach for representing and exploring evolutionary scenarios where introgression is suspected to have occurred. It is implemented in the OSF-Builder software which is freely available for download from the webpage mentioned in the abstract. It takes as input a species forest and an allele tree and constructs an OSF for them that is minimal in the number of contact arcs. We have assessed the performance of OSF-Builder in the presence of uncertainty by means of a simulation study, and have also illustrated its use on a biological dataset. Our results are promising and indicate that OSF-Builder could be a useful new tool for studying introgression, in combination with existing tools for identifying introgression, especially as it is quick to run in practice.

For OSF-Builder to give informative results it is clearly important to take as input good estimates of the species forest and allele tree. Estimating the allele tree is usually a matter of building a phylogenetic tree from a multiple sequence alignment for which there are numerous tools available. For the species forest, data such as morphological characters or geographical locations are commonly used to infer lineage information (see e.g. (The Heliconius Genome Consortium 2012; Barej *et al.* 2015)). If however lineages are not known, then methods such as those described in (Parker *et al.* 2008; Alexandrou *et al.* 2011) might be used. Alternatively, one could also run OSF-Builder with varying choices of (species-tree induced) lineage trees to better understand the effect of lineage tree choice. Guided by the species tree, this clearly boils down to either merging or splitting up lineages trees. Provided that the resulting species forest is such that every lineage tree in the forest contains at least one species for which information is available in the allele tree then the inner workings of OSF-Builder ensure that the number of contact arcs increases in case lineage trees are split up and decreases in case they are merged.

Our simulation studies suggest that OSF-Builder is resilient to small amounts of uncertainty in the allele tree or in the lineage trees. This not only holds with regard to recovering the correct number of contact arcs but also to reconstructing the true contact arcs. Since deeper signals tend to get confounded by more recent ones it might however be difficult to detect events which occurred further back in time, a problem faced by many tools used for phylogenetic analysis. In the case of OSF-Builder, the most likely reason for this occurring is its bottom-up nature which implies that decisions as to how to resolve ties close to leaves can affect the breaking of ties closer to the root of the allele tree. In regards to our simulation methodology, we have used random SPR moves to generate trees that are increasingly far away from either a fixed lineage or allele tree in terms of the number of SPR operations. It may be of interest to investigate other ways to simulate random introgression scenarios which take into account the underlying introgression process, but this would probably first require a careful analysis of what the most common kinds of error are in lineage and allele tree estimates.

OSF-Builder is quick to run in practice (e.g. the *Heliconius* butterflies dataset took 3.8s to run on a HP laptop running Windows 10), and can be run on large datasets. Even so, as noted above, there could potentially be several optimal OSFs (just as with maximum parsimony where there can be several optimal most parsimonious trees – see, e. g. , Kurtzman and Robnett (2003)). In case an OSF is derived from a species tree, we recommend that the lineage trees be input in the order suggested by their relative ages in the species tree, with the oldest one coming first. If this information is not available or the user wants to understand the effect of other orderings the program can be run several times, each time changing tie resolving preferences to get an idea which contact arcs are more common than others (as illustrated in the *Heliconius* butterfly dataset). Even so, it might be of interest to develop alternative ways to allow OSF-Builder to resolve ties. For example, weighting schemes could maybe be developed for the lineage trees based on either confidence values for the trees or the size of their leaf sets (or a combination of both). As different OSFs might be computed it might also be of interest to develop tools that would allow for comparing and assessing different OSFs, along the lines of metrics used for phylogenetic trees or networks (see, e.g., Cardona *et al.* (2009); Huber *et al.* (2011)). This could potentially provide more insight into differences between OSFs supported by a dataset rather than just taking the difference in the number of contact arcs, as in our simulation studies. Another potential direction to understand multiple OSFs may be to investigate the adaptation of approaches that deal with multiple optima in lateral gene transfer studies (see e.g. Bansal *et al.* (2013)).

Although the parsimony model appears to be useful in practice it is based on the assumption that introgression is infrequent, an assumption whose validity may become clearer as the number of genome studies for introgression grows (Koyama *et al.* 2016; Zhang *et al.* 2016). Therefore, it could be of interest to develop more sophisticated models for introgression which could try to take this into account. For example, it could be worth exploring whether our optimization criterion could be replaced with a more sophisticated one that takes into account costs of introgression events (e. g. , different costs could be assigned to introgression events based on their source trees and/or their recipient trees). In another direction, it should be noted that our model does not allow for losses, which can be an important source of genetic variation (Albalat and Canestro 2016). It may be interesting to try and incorporate losses into our model, along similar lines to approaches used for modelling tree reconciliation some of which, for example, employ probabilistic methods (see, e.g., Baudet *et al.* (2015); Doyon *et al.* (2011)). Another important underlying evolutionary process that should be taken into account is incomplete lineage sorting, whose affects can also greatly complicate matters (Choleva *et al.* 2014). Including this process into network models appears to be a challenging problem. In Holland *et al.* (2008), an approach is described for distinguishing incomplete lineage sorting and hybridization using a special type of (unrooted) phylogenetic network called a split network. Recent progress has also been made on this problem for rooted networks using parsimony (Yu *et al.* 2013) and both could provide clues for its solution.

OSF-Builder is a method that is designed to analyse a single allele tree relative to a species forest. In practice, however, it could be of interest to analyse several allele trees simultaneously. This could, for example, provide more evidence of large-scale introgression events, where several genes introgressed at a similar time between lineages. This could be done one allele tree at a time, and the resulting OSFs compared in a pairwise fashion, potentially using methods for comparing OSFs as mentioned above. However, if a large number of allele trees are involved this could potentially become quite cumbersome. In the area of lateral gene transfer, methods have been developed for identifying 'highways' for lateral gene transfer between species which consider multiple gene trees (see, e. g. , Bansal *et al.* (2011)). Thus, it might be of interest to see if these approaches could be adapted to deal with multiple allele trees. These methods could be useful for teasing apart whether or not multiple, tightly-linked genes or single genes are responsible for certain traits (see e.g. (Nadeau 2016, p.28) for a discussion of such a phenomenon in butterfly patterning).

In summary, we believe that OSF-Builder is a helpful new tool for analysing introgression, and we expect that it should provide a useful basis for developing new methods to study this important phenomenon.

Supplementary Material

Supplementary Material is available from the Dryad Digital Repository: `http://dx.doi.org/10.5061/dryad.1nt8t75`

Acknowledgements

The authors would like to thank Annabel Whibley, John Innes Center, UK, for stimulating discussions on the *Heliconius* butterfly dataset, and the reviewers and the associate editor for their very helpful and constructive comments. KTH, GS, and VM would also like to thank the Biomathematics Research Center, University of Canterbury, New Zealand.

*

References

Albalat, R. and Canestro, C. 2016. Evolution by gene loss. *Nat. Rev. Genet.*, 17(7): 379–91.

Alexandrou, M. A., Oliveira, C., Maillard, M., McGill, R. A. R., Newton, J., Creer, S., and Taylor, M. I. 2011. Competition and phylogeny determine community structure in Mllerian co-mimics. *Nature*, 469: 84–88.

Babicki, S., Arndt, D., Marcu, A., Liang, Y., Maciejewski, A., and Wishart, D. S. 2016. Heatmapper: Web-enabled heat mapping for all. *Nucleic Acids Res.*, 44(W1): W147–W153.

Bansal, M., Banay, G., Gogarten, J., and Shamir, R. 2011. Detecting highways of horizontal gene transfer. *J. Comput. Biol.*, 18(9): 1087–114.

Bansal, M., Alm, E., and Kellis, M. 2013. Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. *J. Comput. Biol.*, 20(10): 738–54.

Barej, M., Penner, J., Schmitz, A., and Röde, M.-O. 2015. Multiple genetic lineages challenge the monospecific status of the West African endemic frog family Odontobatrachidae. *BMC Evolutionary Biology*, 15: 67.

Baudet, C., Donati, B., Sinaimeri, B., Crescenzi, P., Gautier, C., Matias, C., and Sagot, M.-F. 2015. Cophylogeny reconstruction via an approximate Bayesian computation. *Syst. Biol.*, 55: 130.

Becker, M., Gruenheit, N., Steel, M., Voelckel, C., Deusch, O., Heenan, P., McLenachan, P., Kardailsky, O., Leigh, J., and Lockhart, P. 2013. Hybridization may facilitate in situ survival of endemic species through periods of climate change. *Nature Climate Change*, 3: 1039–1043.

Blischak, P., Chifman, J., Wolfe, A., and Kubatko, L. 2018. HyDe: a Python package for genome-scale hybridization detection. *Syst. Biol*, 67(5): 821–829.

Cardona, G., Llabés, M., Rosselló, F., and G., V. 2009. Metrics for phylogenetic networks ii: Nodal and triplets metrics. *IEEE/ACM Trans Comput Biol Bioinform*, 6(3): 454–69.

Choleva, L., Musilova, Z., Kohoutova-Sediva, A., Paces, J., Rab, P., and Janko, K. 2014. Distinguishing between incomplete lineage sorting and genomic introgressions: Complete fixation of allospecific mitochondrial dna in a sexually reproducing fish (cobitis; teleostei), despite clonal reproduction of hybrids. *PLoS One*, 9(6): e80641.

Doyon, J.-P., Ranwez, V., Daubin, V., and Berry, V. 2011. Models, algorithms and programs for phylogeny reconciliation. *Briefings in Bioinformatics*, 12(5): 392–400.

Durand, E. Y., Patterson, N., Reich, D., and Slatkin, M. 2011. Testing for ancient admixture between closely related populations. *Mol. Biol. Evol.*, 28(8): 2239–2252.

Excoffier, L., Novembre, J., and Schneider, S. 2000. Simcoal: A general coalescent program for the simulation of molecular data in interconnected populations with arbitrary demography. *Hered.*, 91(6): 506–509.

Fitch, W. M. 1971. Toward defining the course of evolution: Minimum change for a specific tree topology. *Syst. Zool.*, 20: 406–416.

Ganser, E. R. and North, S. C. 2000. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11): 1203–1233.

Ge, S., Sang, T., Lu, B.-R., and Hong, D.-Y. 1999. Phylogeny of rice genomes with emphasis on origins of allotetraploid species. *Proc. Natl. Acad. Sci. USA*, 96(25): 14400–14405.

Goulet, B., Roda, F., and Hopkins, R. 2017. Hybridization in plants: Old ideas, new techniques. *Plant Physiology*, 173: 65–78.

Grant, P. and Grant, B. 2016. Introgressive hybridization and natural selection in darwin's finches. *Biological Journal of the Linnean Society*, 117(4): 812–822.

Holland, B. R., Benthin, S., Lockhart, P. J., Moulton, V., and Huber, K. T. 2008. Using supernetworks to distinghish hybridization from lineage-sorting. *BMC Evolutionary Biology*, 8: 202pp.

Huber, K. T., Spillner, A., Suchecki, R., and Moulton, V. 2011. Metrics on multilabeled trees: Interrelationships and diameter bounds. *IEEE/ACM Trans Comput Biol Bioinform.*, 8(4): 1029–40.

Huber, K. T., Moulton, V., Sagot, M.-F., and Sinaimeri, B. 2018. Exploring and visualising spaces of tree reconciliations. *Syst. Biol.*

Huson, D. and Scornavacca, C. 2012. Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks. *Syst. Biol.*, 61(6): 1061–7.

Joly, S. 2012. JML:Testing hybridization from species trees. *Mol. Ecol. Resour.*, 12: 179–184.

Joly, S., McLenachan, P. A., and Lockhart, P. J. 2009. A statistical approach for distinguishing hybridization and incomplete lineage sorting. *Am. Nat.*, 174: E54–E70.

Koyama, T., Ito, H., Fujisawa, T., Ikeda, H., Kakishima, S., Cooley, J., Simon, C., Yoshimura, J., and Sota, T. 2016. Genomic divergence and lack of introgressive hybridization between two 13-year periodical cicadas support life cycle switching in the face of climate change. *Mol. Ecol.*, 25(21): 5543–5556.

Kozak, K. M., Wahlberg, N., Neild, A. F. E., Dasmahapatra, K. K., Mallet, J., and Jiggins, C. D. 2015. Multilocus species trees show the recent adaptive radiation of the mimetic heliconius butterflies. *Syst. Biol.*, 64(3): 505524.

Kurtzman, C. P. and Robnett, C. 2003. Phylogenetic relationships among yeasts of the 'Saccharomyces complex' determined from multigene sequence analyses. *FEMS Yeast Res*, 3(4): 417–32.

Mallet, J. 2005. Hybridization as an invasion of the genome. *Trends in Ecology & Evolution*, 20(5): 229–237.

Martin, S. H., Davey, J. W., and Jiggins, C. 2015. Evaluating the use of ABBA - BABA statistics to locate introgressed loci. *Mol. Biol. Evol.*, 32(1): 244–257.

Morin, M. M. and Moret, B. M. E. 2006. NETGEN: Generating phylogenetic networks with diploid hybrids. *Bioinformatics*, 22(15): 19211923.

Muhlfeld, C., Kovach, R., Jones, L., Al-Chokhachy, R., Boyer, M., Leary, R., Lowe, W. H., Luikart, G., and Allendorf, F. W. 2014. Invasive hybridization in a threatened species is accelerated by climate change. *Nature Climate Change*, 4: 620–624.

Nadeau, N. 2016. Genes controlling mimetic colour pattern variation in butterflies. *Current Opinion in Insect Science*, 17: 24–31.

Parker, J., Rambaut, A., and Pybus, O. 2008. Correlating viral phenotypes with phylogeny: Accounting for phylogenetic uncertainty. *Infection, Genetics and Evolution*, 8(3): 239–46.

Sankararaman, S., Mallick, S., Dannemann, M., Prüfer, K., Kelso, J., Pääbo, S., Patterson, N., and Reich, D. 2014. The genomic landscape of Neanderthal ancestry in present-day humans. *Nature*, 507(7492): 354.

Semple, C. and Steel, M. 2003. *Phylogenetics*. Oxford University Press.

Solis-Lemus, C. and Ané, C. 2016. Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting. *PLoS Genet*, 12(3): e1005896.

Sousa, V. and Hey, J. 2013. Understanding the origin of species with genome-scale data: Modelling gene flow. *Nat. Rev. Genet.*, 14: 404–414.

Stewart Jr, C. N., Halfhill, M. D., and Warwick, S. I. 2003. Genetic modification: Transgene introgression from genetically modified crops to their wild relatives. *Nat. Rev. Genet.*, 4(10): 806.

Than, C., Ruths, D., and Nakhleh, L. 2008. PhyloNet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, 9: 322.

The Heliconius Genome Consortium, . 2012. Butterfly genome reveals promiscuous exchange of mimicry adaptations among species. *Nature*, 487(7405): 94–8.

Tofigh, A., Hallett, M., and Lagergren, J. 2011. Simultaneous identification of duplications and lateral gene transfer. *IEEE/ACM Trans. Comp. Bio. Bioinf.*, 8(2): 517–535.

Wallbank, R. W. R., Baxter, S. W., Pardo-Diaz, C., Hanly, J. J., H., M. S., Mallet, J., Dasmahapatra, K. K., Salazar, C., Joron, M., Nadeau, N., McMillan, W. O., and Jiggins, C. D. 2016. Evolutionary novelty in a butterfly wing pattern through enhancer shuffling. *PLoS Biol.*, 14(1): e1002353.

Wen, D., Yu, Y., Hahn, M. W., and Nakhleh, L. 2016. Reticulate evolutionary history and extensive introgression in mosquito species revealed by phylogenetic network analysis. *Mol. Ecol.*, 25(11): 2361–72.

Yu, Y., Barnett, R. M., and Nakhleh, L. 2013. Parsimonious inference of hybridization in the presence of incomplete lineage sorting. *Syst. Biol.*, 62: 738–751.

Zhang, W., Dasmahapatra, K. K., Mallet, J., Moreira, G. R., and Kronforst, M. R. 2016. Genome-wide introgression among distantly related heliconius butterfly species. *Genome Biology*, 17(1): 25.