Advancing admixture graph estimation via maximum likelihood network orientation

Erin K. Molloy[1,*], Arun Durvasula[2], Sriram Sankararaman[1,2,3,4,*]

[1] *Dept. of Computer Science, Univ. of California, Los Angeles, 90095, USA*

[2] *Dept. of Human Genetics, David Geffen School of Medicine, Univ. of California, Los Angeles, 90095, USA*

[3] *Bioinformatics Interdepartmental Program, Univ. of California, Los Angeles, 90095, USA*

[4] *Dept. of Computational Medicine, Univ. of California, Los Angeles, 90095, USA*

**Email: ekmolloy@cs.ucla.edu, sriram@cs.ucla.edu*

February 2, 2021

Abstract

Motivation: Admixture, the interbreeding between previously distinct populations, is a pervasive force in evolution. The evolutionary history of populations in the presence of admixture can be modeled by augmenting phylogenetic trees with additional nodes that represent admixture events. While enabling a more faithful representation of evolutionary history, *admixture graphs* present formidable inferential challenges. A key challenge is the need for admixture graph inference algorithms that are accurate while being completely automated and computationally efficient. Given the challenge of exhaustively evaluating all topologies, search heuristics have been developed to enable efficient inference. One heuristic, implemented in the popular method TreeMix, consists of adding admixture edges to an initial tree while optimizing a suitable objective function.

Results: Here, we present a demographic model (with one admixed population incident to a leaf) where TreeMix and any other starting-tree-based maximum likelihood heuristic using its likelihood function is *guaranteed* to get stuck in a local optimum and return the incorrect network topology. To address this issue, we propose a new search strategy based on reorientating the admixture graph that we term the maximum likelihood network orientation (MLNO) problem. We augment TreeMix with an exhaustive search for MLNO, referred to as OrientAGraph. In evaluations using previously published admixture graphs, OrientAGraph outperforms TreeMix on 4/8 models (there are no differences in the other cases). Overall, OrientAGraph finds graphs with higher likelihood scores and topological accuracy while remaining computationally efficient. Lastly, our study reveals important directions for improving maximum likelihood admixture graph estimation.

Availability: OrientAGraph is available on Github (`https://github.com/ekmolloy/OrientAGraph`) under the GNU General Public License v3.0.

1 Introduction

Admixture, the exchange of genes between distinct populations, has emerged as an important process in shaping genetic diversity within and between populations. Evidence of gene flow events in many species, including humans, (Green *et al.*, 2010), wolves (Pilot *et al.*, 2019) and butterflies (Edelman *et al.*, 2019), has led to increasing interest in estimation of the demographic history of populations while accounting for admixture events.

The demographic history of populations in the presence of admixture can be naturally modeled by augmenting phylogenetic trees with additional nodes that represent admixture events. These *admixture graphs* cannot be directly observed and need to be estimated from genomic data. Admixture graphs model changes in allele frequencies as a Wright-Fisher process, parameterized by the pair $(N, \mathbf{\Theta})$, where N is

a *phylogenetic network* on a set S of populations (Definition 1) and $\boldsymbol{\Theta}$ is a real-valued vector describing parameters associated with the edges of N, namely branch lengths in units of genetic drift and admixture proportions. The objective of admixture graph inference is to estimate $(N, \boldsymbol{\Theta})$ from genetic variation data measured across the populations in S. Typically, these methods utilize as input summary statistics computed from allele frequencies in each of the S populations such as the vector $\boldsymbol{X}$ of f-statistics.

One of the most widely used tools, `qpGraph` (Patterson *et al.*, 2012), takes $\boldsymbol{X}$ as well as a network topology N as input and then fits the numerical parameters $\boldsymbol{\Theta}$. When the number of populations is very small, it is possible to evaluate the likelihood of all network topologies with a fixed number of admixture events (e.g. using the `admixturegraph` R package from Leppälä *et al.*, 2017). Otherwise, the researcher must propose potential networks and evaluate them individually (see Harney *et al.*, 2020 for discussion).

Brute force and manual procedures quickly become intractable because the number of possible network topologies grows super-exponentially in the number of populations (McDiarmid *et al.*, 2015), motivating the development of heuristics for estimating admixture graphs from f-statistics. For example, `MixMapper` (Lipson *et al.*, 2013, 2014) is a pipeline in which the user identifies a set of unadmixed populations $R \subseteq S$, estimates a submodel (i.e. a population tree) on $S \setminus R$, and then sequentially adds populations in R to the submodel. This approach works best when there are only a few admixed populations and can be viewed as a "semi-automated" version of `qpGraph` (see Lipson, 2020 for discussion).

A recent approach to automating admixture graph inference, called `miqograph` (Yan *et al.*, 2020), is unique from prior methods in that it simultaneously estimates the network topology N and numerical parameters $\boldsymbol{\Theta}$, requiring the user to specify some additional parameters. `miqograph` is similar to `admixturegraph` in that it is guaranteed to find the highest scoring admixture graph with a user-specified number of admixture events; however, unlike `admixturegraph`, the network space evaluated by `miqograph` is constrained to networks in which all admixture nodes are either a leaf node or are the parent of an admixture node (see Yan *et al.*, 2020 for further discussion).

To the best of our knowledge, the only method that is fully automated and (at least in principle) allows for the greatest flexibility in exploring network topologies is `TreeMix` (Pickrell and Pritchard, 2012). The only topological constraint imposed by `TreeMix` is that the output is *tree-based*, meaning that it can be drawn as a tree annotated with additional edges representing gene flow (Francis and Steel, 2015; see Definition 2). At a high level, `TreeMix` uses a maximum likelihood (ML) search heuristic that operates with two major phases: first, a ML population tree T is estimated on S, and second, edges are added sequentially as long as the edges improve a measure of model fit (hill climbing). We refer to methods that operate in this fashion as starting-tree-based maximum likelihood (STB-ML) methods. Empirical evidence suggests that `TreeMix` can be biased by its starting tree, especially when many populations are admixed (Lipson *et al.*, 2013) so that many researchers are wary of relying solely on `TreeMix`, despite its scalability relative to other approaches (Lipson, 2020).

In this paper, we consider the problem of admixture graph estimation using STB-ML methods. To explore the limitations of this class of methods, we consider a simple admixture graph $(N^*, \boldsymbol{\Theta}^*)$ with just one admixture event incident to a leaf. Given the f-statistics implied by the true model $(N^*, \boldsymbol{\Theta}^*)$ (which is equivalent to assuming that the number of independent sites goes to infinity), any STB-ML method that succeeds in finding the ML tree (which in this case is also the Neighbor Joining tree) as it starting tree, is guaranteed to get stuck in a local optimum and return an incorrect network topology N'. Our case study highlights that STB-ML methods can be inconsistent even under a demographic model with a single admixture event.

Examining our case study in more detail, we observe that the incorrect topology N' can be transformed into the correct topology N^* simply by changing which population is admixed and redirecting the edges of N' accordingly. This graph transformation is commonly performed by researchers manually searching for the admixture graph topology using `qpGraph` (Lipson, 2020) and is also related to recent theoretical results on phylogenetic network orientation by Huber *et al.* (2019). Utilizing the terminology of Huber *et al.* (2019), we observe that N' and N^* are two different *orientations* of the same undirected network topology (Definition 4), with N^* yielding in a higher likelihood score than N'. While Huber *et al.* (2019) look at network orientation from a graph theory perspective (e.g. addressing questions such as: is the orientation of a phylogenetic network uniquely determined by the position of the root and admixed populations?), in this paper, we explore network orientation as a search strategy for efficiently identifying a ML admixture graph, leading us to propose the *maximum likelihood network orientation* (MLNO) problem.

To evaluate the utility of solving the MLNO problem for estimating admixture graphs, and more specifically, improving the accuracy of STB-ML methods, we augment `TreeMix` with an exhaustive search for the MLNO after the addition of every edge. In line with theoretical expectations, our approach, which we refer to as `OrientAGraph`, recovers the correct network in our motivating case study, whereas the original `TreeMix` method does not. More importantly, we benchmark `OrientAGraph` versus `TreeMix` and `miqograph` using six model admixture graphs that were estimated from biological data sets in recent studies. We find that `OrientAGraph` improves the accuracy of the original `TreeMix` method and is more accurate than `miqograph` either when admixture occurs deep in time or when the data set is large enough (*e.g.*, 10 populations and 2 admixture events) so that `miqograph` does not solve its problem to optimality within the allowed time frame for our study. Although our focus is on estimating admixture graphs from f-statistics, the MLNO problem may be relevant to other methods, including `PhyloNet` (Yu *et al.*, 2014; Wen *et al.*, 2018) and `GTMix` (Wu, 2020), that implement an ML search over network space from a starting tree.

2 Terminology and Background

Throughout this paper, we refer to the approximate likelihood functions utilized within `TreeMix` (Pickrell and Pritchard, 2012) and `qpGraph` (Patterson *et al.*, 2012) as well as theoretical results for explicit phylogenetic networks. We review these results and define the relevant terminology in this section.

2.1 Phylogenetic Networks

First and foremost, we use the term *phylogenetic network* to refer to a graphical object, which we now define.

Definition 1 (Phylogenetic Network)**.** *A* phylogenetic network *N is a triplet (n, S, ϕ), where n is a graph, S is a set of labels (typically denoting species or populations), and ϕ is a bijection mapping the leaves (i.e. vertices with out-degree 0) of n to the labels in S. Unless otherwise noted, N is* directed*, meaning that n is a directed acyclic graph with a directed path between the root (i.e. a special vertex with in-degree 0) and all other vertices in n and without any parallel arcs or self-loops. We assume that the root has out-degree 2, leaf vertices have in-degree 1, and all other vertices, referred to as internal vertices, have in-degree and out-degree summing to 3; put simply, we assume that n is binary. For convenience, we typically do not make an explicit distinction between a phylogenetic network N and its graph n. Instead, we say that N is a network on S and denote its vertices as $V(N)$, edges as $E(N)$, and leaves as $L(N)$.*

The definition given here is consistent with Francis and Steel (2015), Huber *et al.* (2019), and much of the literature on explicit phylogenetic networks. It is worth noting that we use the terms admixture or gene flow instead of reticulation or hybridization. Specifically, we refer to any internal vertex with in-degree 2 (and thus out-degree 1) as an *admixture node*. An *admixture edge* is an arc with an admixture node as its head (or target); all other arcs are referred to as *tree edges*. A directed phylogenetic network with zero admixture nodes (and thus edges) is simply a directed *phylogenetic tree*.

A phylogenetic network N *contains* a phylogenetic tree T if T can be obtained from N through a sequence of edge deletions and edge contractions. Recall that an edge *contraction* removes edge $e = (u, v)$ from $E(N)$ and then identifies its endpoints, that is, u and v are replaced with a single vertex uv and then the edges of $E(N)$ are updated in the natural way. A network containing trees should not to be confused with a *tree-based network*, a class of phylogenetic networks introduced by Francis and Steel (2015)

Definition 2 (Tree-based Phylogenetic Network)**.** *We say that a directed phylogenetic network N is* tree-based *if it can be obtained by sequentially adding admixture edges to a (binary, directed) phylogenetic tree T, referred to as the base tree. Specifically, every admixture edge must be added so that both its head and tail subdivide tree edges. An important point is N can have multiple base trees.*

This class of networks is relevant to our study, as `TreeMix` constrains its search to tree-based networks. `TreeMix` searches the space of tree-based phylogenetic networks by adding admixture edges and executing *tail moves*.

Definition 3 (Tail Move)**.** *A tail move between two edges e and f in a phylogenetic network N involves relocating the tail (or source) of edge e so that it subdivides edge f.*

Importantly, the space of rooted phylogenetic networks is connected under tail moves (Theorem 3.4 in Janssen *et al.*, 2018; also see Gambette *et al.*, 2017).

Thus far, the definitions and results have been for directed phylogenetic networks; we conclude this section by discussing recent results for *undirected* phylogenetic networks. A phylogenetic network $N' = (n', S, \theta)$ is undirected if its n' is undirected and connected, with no parallel edges or self-loops (Gambette *et al.*, 2012). Again, we assume that N' is binary, so leaves are vertices with degree 1 and all other vertices have degree 3. It is easy to transform a directed phylogenetic network N into an undirected network N' simply by ignoring edge directions and suppressing the vertex previously designated as the root (Gambette *et al.*, 2012).

Definition 4 (Network Orientation)**.** *We say that a directed network N is an* orientation *of an undirected network N' if the undirected version of N, denoted $N|_u$, is isomorphic to N'.*

Note that two undirected networks $N_1' = (n_1, S, \phi_1)$ and $N_2' = (n_2, S, \phi_2)$ are *isomorphic* if there exists a bijection $f : V(N_1') \to V(N_2')$ such that $(u, v) \in E(N_1')$ if and only if $(f(u), f(v)) \in E(N_2')$ and $\phi_1(u) = \phi_2(f(u))$ for all $u \in L(N)$.

The reverse operation of transforming an undirected network N' into a directed network N, referred to as orienting the network, is not so straightforward, because the location of the root does not uniquely determine the direction of all edges in an unrooted network (Gambette *et al.*, 2012; Huber *et al.*, 2019). Notably, Huber *et al.* (2019) recently showed that specifying the the location of the root (i.e. the edge e_r that the root subdivides) and the admixture nodes (i.e. the nodes $A \subseteq V(N')$ that should have in-degree 2) in an undirected phylogenetic network N' results in a unique orientation of N, provided that an orientation exists (Theorem 2 in Huber *et al.*, 2019). Furthermore, given e_r and A, this orientation can be can be found in linear time in the number of edges (Algorithm 1 in Huber *et al.*, 2019).

In the remainder of this paper, all phylogenetic networks are directed, unless otherwise noted, and the term "phylogenetic" is often omitted when referring to phylogenetic trees and networks.

2.2 Likelihood Function

In the previous section, we discussed phylogenetic networks and defined tail moves, which can be used to traverse network space. However, a phylogenetic network N is just one parameter in an admixture graph $(N, \boldsymbol{\Theta})$, so in order to search network space, we must define a procedure for estimating the numerical parameters $\boldsymbol{\Theta}$ (i.e. the branch lengths in drift units and admixture proportions) as well as a procedure for computing the likelihood of the resulting model given the input data. In this section, we review the aspects of likelihood function and parameter optimization used by `TreeMix` (and related methods e.g. `qpGraph`) that are necessary to understand our case study and computational experiments.

Without loss of generality, we can define the likelihood function in terms of f_2 statistics. Given a model admixture graph $(N, \boldsymbol{\Theta})$, we can compute the expected value of the f_2-statistics directly, up to a scaling factor (Pickrell and Pritchard, 2012; Patterson *et al.*, 2012; Peter, 2016). If N is a tree (i.e. there is no admixture), then the f_2-statistic for populations $i, j \in S$ is the sum of the lengths (in units of genetic drift) of edges on the path from the leaf labeled i to the leaf labeled j in N. If there is admixture, then this formula is more complex, as we must consider all paths between i and j and use the admixture proportions to weight these paths appropriately.

In practice, $(N, \boldsymbol{\Theta})$ are unknown, and each f_2-statistic is estimated from allele frequency data, along its standard error. We let $\boldsymbol{X}$ and $\boldsymbol{Z}$ denote the vector of observed f-statistics and variances, respectively. Given our input $\boldsymbol{X}$ and a network N, we can optimize $\boldsymbol{\Theta}$ to minimize the difference between the observed f_2-statistics $\boldsymbol{X}$ and the expected value $\boldsymbol{Y}$ of the f_2-statistics for $(N, \boldsymbol{\Theta})$. Assuming the difference between the expected value Y_i and observed value X_i is normally distributed with mean 0 and standard deviation Z_i, the composite likelihood function is

$$\log \mathcal{L}(N) = -\frac{1}{2} \sum_i \left(\frac{(X_i - Y_i)^2}{Z_i^2} + 2\log(Z_i) + \log(2\pi) \right) \tag{1}$$

This (approximate) log-likelihood function used by `TreeMix` is related to the log-likelihood score used by `qpGraph` and `miqograph`.

As discussed by Pickrell and Pritchard (2012) and others (*e.g.*, Lipson, 2020), the three branch lengths incident to an admixture node cannot be estimated simultaneously (because there are more unknowns than

equations). Automated methods typically estimate one of the three branch lengths, setting the other two to zero. For example `admixturegraph` sets the lengths of the incoming edges to 0, and `TreeMix` sets the outgoing branch and one of the two incoming branches to zero. Such choices in parameter estimation can change the likelihood of the fitted model (N, Θ). Lastly, we note that the position of the root is not identifiable, other than that it must be ancestral to admixture nodes in N.

3 Pitfalls of Starting-Tree-based ML Methods

In this section, we will show that there exist simple demographic models for which starting-tree-based maximum likelihood (STB-ML) methods, such as `TreeMix`, are guaranteed to get trapped in a local optimum and thus return the incorrect network.

The model network we consider, denoted $(N^*, \mathbf{\Theta}^*)$, is shown in Figure 1a. N^* is on a population set $S = \{A, B, C, D, E\}$, with A labeling the admixed population and E labeling the outgroup. Given $(N^*, \mathbf{\Theta}^*)$, it is possible write down the f-statistics implied by this model, denoted $\boldsymbol{X}^*$. As the amount of data generated under $(N^*, \mathbf{\Theta}^*)$ goes to infinity, the estimated f-statistics converge to $\boldsymbol{X}^*$, up to a scaling factor, and the standard error for every f-statistic goes to zero (note that it is typically shifted by a small constant to avoid numerical issues).

Evaluating the likelihood of a candidate network N requires optimizing the numerical parameters $\mathbf{\Theta}$. In section 2.2, we noted that there are differences in how these parameters are fit by automated methods. To show that our results are agnostic to this optimization choice, we perform computational experiments using the `admixturegraph` R package, which fixes the lengths of all admixture edges to 0, and `TreeMix`, which fixes the length of one of the two incoming edges to each admixture node and its outgoing edge to 0. For `TreeMix`, the log-likelihood of (N, Θ) depends on which of the two admixture edges is set to 0; therefore, we evaluate both scenarios and report the higher of the two log-likelihoods. All scripts used in this study are available on Github: `https://github.com/ekmolloy/mlno-study`.

We begin our computational study by addressing whether the network topology N^* has the highest likelihood of all phylogenetic networks on S with 1 admixture event. We found that N^* has the highest likelihood through brute force, using the `all_graphs` function in `admixturegraph` to generate all such

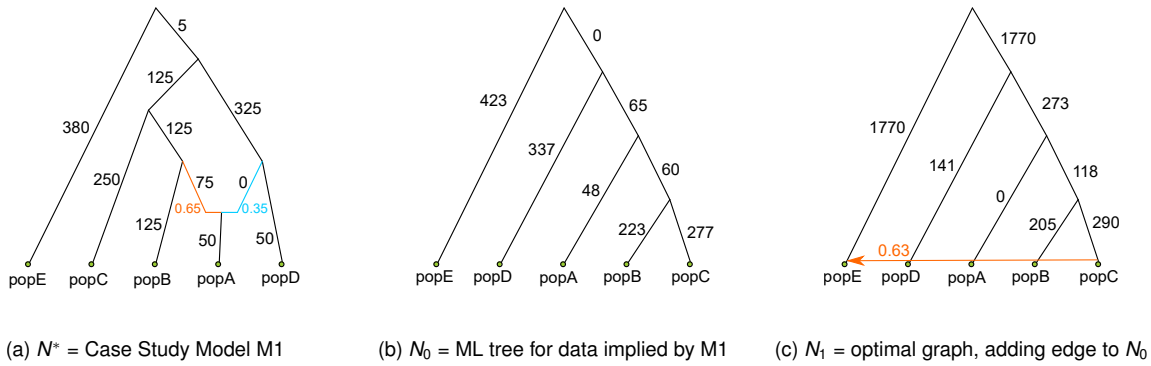

Figure 1: Subfigure (a) shows the model admixture graph $(N^*, \mathbf{\Theta}^*)$ discussed in Section 3; we refer to this model as M1. Subfigure (b) shows the maximum likelihood tree, denoted N_0, for the data implied by M1. Subfigure (c) shows the network that adds one admixture edge to N_0 so that the likelihood of the resulting graph is maximized; we denote this network as N_1. N_0 and N_1 are drawn rooted at the outgroup E, although recall that the root does not change the likelihood score, provided the root is in a valid position: the root needs to be ancestral to the admixture nodes. Although N^* has the highest likelihood of all networks with one admixture edge, `TreeMix` gets stuck in a local optimum, returning N_1. The log-likelihood scores, computed using `TreeMix`, of N_0, N_1, and N^* are -1,302,625, -366,944, and 83, respectively. The numerical parameters shown in subfigure (b) and (c) were also computed using `TreeMix`. Branch lengths are shown multiplied by $1,000$, as this is the convention in the population genetics literature.

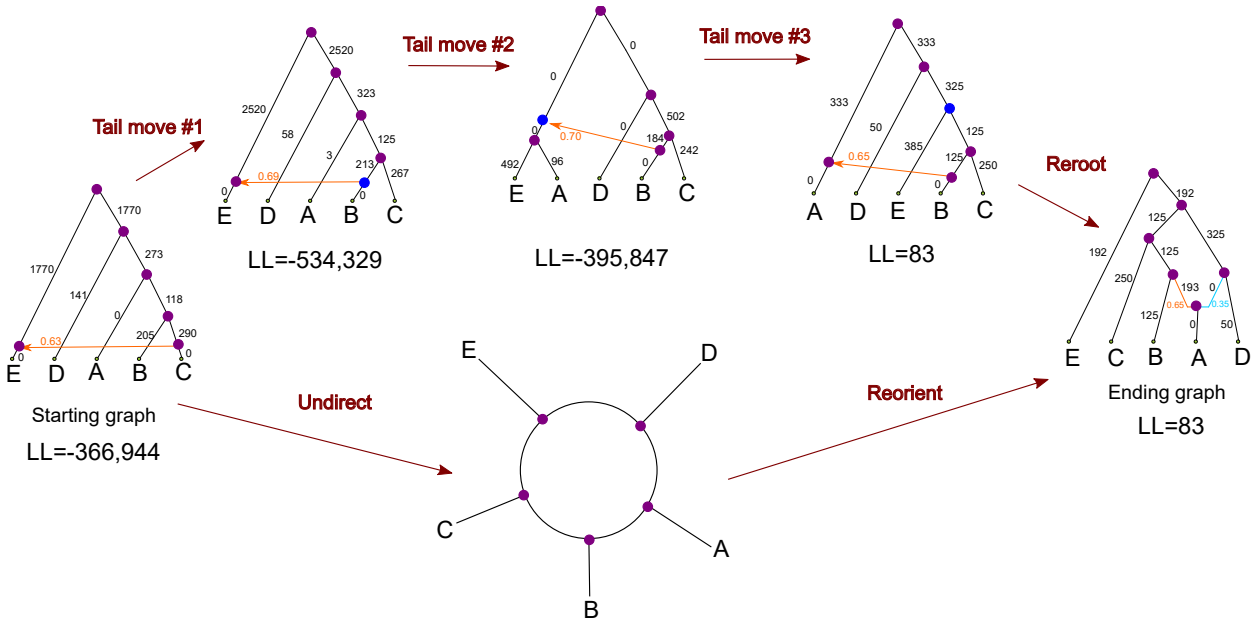

Figure 2: The top of the figure shows the sequence of three tail moves that transforms the starting network N_1 computed in Step 2 into the ending network N^*. We found, through brute force analyses, that this is the minimum number of tail moves required and furthermore that any sequence of tail moves from N_1 to N^* must traverse through a network with a lower likelihood score than N_1. Alternatively, N_1 can be transformed into N^* by selecting A as the admixed population (instead of E) and redirecting the edges of the network accordingly. Adopting the language of Huber *et al.* (2019), we refer to this as reorientating the network. This is not to be confused with rerooting the network, which simply relocates the position of the root and does not change which populations are admixed. The numerical parameters and log-likelihood scores shown above were computed using `TreeMix`. Note that the branch lengths in these subfigures are not drawn to scale and should be divided by 1,000.

topologies and computing their likelihoods with both `admixturegraph` and `TreeMix`.

We now turn to the performance of STB-ML methods given $\boldsymbol{X}^*$ as input. Because $M1$ contains a single admixture event, a generic ML-STB method would operate by (1) estimating a maximum likelihood starting tree N_0, (2) adding an admixture edge to N_0 so that the likelihood of the resulting network, denoted N_1, is maximized, and (3) searching for a network with a higher likelihood from N_1, for example by applying tail moves (Definition 3). Below, we use brute force to determine the outcome of these three steps. The log-likelihood scores and numerical parameters reported in this section were computed using `TreeMix`, selecting the direction of the gene flow edge to maximize the likelihood.

Step 1: The first step of a STB-ML method is to estimate a starting tree on the full set of populations; this is typically done using an ML heuristic or the Neighbor Joining (NJ) algorithm (Saitou and Nei, 1987). The NJ tree for $\boldsymbol{X}^*$ is shown in Figure 1b, where we have rooted the tree at the outgroup population E (recall that the position of the root is not identifiable). In our case study, the NJ tree is topologically equivalent to the ML tree, which we confirmed using a brute force search (as described above). It is worth noting that the starting tree, denoted N_0, is contained in the undirected version of N^*, but the outgroup-rooted version of N_0 is *not* contained in N^*. Because N_0 cannot be transformed into N^* simply through edge additions, any STB-ML method that utilizes the NJ tree or successfully finds the ML tree will need to utilize additional search moves to recover N^*.

In practice, `TreeMix` uses a random population heuristic to build a starting tree and may not be successful in finding the optimal tree. We explore this issue further in Section 5; in this section, we assume that optimal tree is identified by the STB method.

Step 2: The second step of a STB-ML method is to add an admixture edge to the starting tree, in this case N_0, so that the likelihood of the resulting network is maximized. Treating N_0 as unrooted, an admixture edge can be added between any pair of edges in N_0 by sub-dividing one edge with the admixture source and the other edge with the admixture target. Again, we used brute force, using the `add_an_admixture` function in `admixturegraph`; the optimal graph, denoted N_1, is shown in Figure 1c. Importantly, N_1 selects E to be the admixed population (instead of A) and has a lower likelihood score than N^*.

Step 3: The third step of an STB-ML method is to search from N_1 for a network with a higher likelihood. For this step, we assume that the STB-ML method employs a hill climbing approach using standard search moves. In our analysis of all networks with one admixture node, we found N_1 to have the second highest likelihood score, second only to N^*; therefore, it is sufficient to show that N_1 cannot be transformed into N^* via a single search move. It is easy to see that this cannot be achieved with head moves, which relocate the head of admixture edges. To evaluate whether this can be achieved with tail moves (Definition 3), we implemented a subroutine that finds all networks within one tail move of the input network. Applying this subroutine iteratively, we found that a minimum of three tail moves are required to transform N_1 into N^*, modulo the position of the root, which as previously mentioned does not impact the likelihood score. As expected from our brute force analysis, this sequence of tail moves, shown in Figure 2, requires moving from N_1 to graphs of lower likelihood, so it is impossible to reach N^* from N_1 via hill climbing with standard search moves.

To summarize, we have shown through a series of computational experiments that even for a simple model admixture graph, with just one admixed population, generic STB-ML methods are guaranteed to get trapped in a local optimum and return the incorrect network. This statement holds for any hill climbing approach in which we employ search moves that cannot move from N_1 to N^* in a single step; we expand upon this in the next section.

4 Maximum Likelihood Network Orientation (MLNO) Problem and OrientAGraph

For the case study from the previous section, we show that generic starting-tree-based maximum likelihood (STB-ML) methods are guaranteed to find a network N_1 from which it is impossible to reach the correct network N^* using standard search moves, without traversing through a network of lower likelihood than N_1. In order to find N^* in a purely hill climbing approach, we need to move from N_1 to N^* in a single step. One possible solution is to perform three tail moves before computing the likelihood score. However, the number of networks that are three tail moves away from N_1 is quite large, so it is unlikely that the correct set of moves would be chosen to transform N_1 to N^*. It is, however, possible to reach N^* from N_1 in a single "move" by selecting A to be the admixed population and reorienting edges of the network accordingly (Figure 2). This search strategy of selecting different populations as admixed populations has been used to manually explore different network topologies, as described in Lipson (2020). There is a connection between this approach and network orientation, because the admixture nodes (along with a valid root position) uniquely determines the orientation of an undirected network, as shown by Huber *et al.* (2019). In other words, researchers, such as Lipson (2020), manually search for the network orientation with the highest likelihood. Integrating this manual process into automated methods is appealing and leads us to propose the maximum likelihood network orientation (MLNO) problem.

Definition 5 (Maximum Likelihood Network Orientation (MLNO) Problem)**.** *Let N' be an undirected phylogenetic network, and let $\mathcal{N}(N')$ be the orientation neighborhood of N' i.e. the set of directed phylogenetic networks such that $N \in \mathcal{N}(N')$ implies that the undirected version of N, denoted $N|_u$, is isomorphic to N'. We say that a directed network N^* is a maximum likelihood orientation of N', if N^* is in the set* $\arg\max_{N \in \mathcal{N}} \mathcal{L}(N)$.

We take $\mathcal{L}(N)$ to be the likelihood function given in Equation 1.

The most straightforward approach to finding the MLNO is brute force, typically starting from some directed network N with h admixture nodes. The orientation neighborhood of $N|_u$ is defined by all ways to select h admixture nodes from the set $V(N|_u)$ and all ways of selecting a root edge from $E(N|_u)$ (Huber

et al., 2019). The position of the root is not identifiable, so for our purposes, we only consider networks that are rooted at the outgroup g, denoting this set $\mathcal{N}_g(N|_u) \subset \mathcal{N}(N|_u)$. In a brute force approach, we evaluate the likelihood of every directed network $M \in \mathcal{N}_g(N|_u)$; this requires optimizing the parameters, denoted $\mathbf{\Theta}_M$. As discussed in Sections 2.2 and 3, methods such as `admixturegraph` and `TreeMix` differ in how they optimize parameters. `TreeMix` optimizes the length of the edge that will be included as part of its base tree (Definition 2) and the length of the other edge (representing gene flow) is set to 0. When adding an admixture event, `TreeMix` evaluates the likelihood for both directions of gene flow, fixing the edge included in the base tree in subsequent calculations. To evaluate the likelihood of a reorientated network, there are two issues: (1) the likelihood scores changes depending on which edges are forced to have length 0 (i.e. labeled as gene flow edges) and (2) `TreeMix` can only be used to compute the likelihood of tree-based networks. To find the likelihood of M, we consider all 2^h trees contained in M and evaluate the likelihood of those that are valid base trees for M (note that determining whether a tree contained in M is a valid base tree for M scales linearly in $|E(M)|$; see Francis and Steel, 2015). This procedure can be applied when h is small; however, we acknowledge that it is computationally intractable for large h and can be circumvented by choosing to optimize the length of the outgoing edge incident to an admixture node, instead of one of the two incoming edges. This connection between the computing the likelihood of an admixture graph and combinatorial results for tree-based phylogenetic networks is interesting and relates more broadly to parameter optimization and likelihood computation for admixture graph inference.

It is also the case that finding the MLNO network is only feasible when h and $|V(N|_u)|$ are sufficiently small. For $h = 1$, the orientation neighborhood $\mathcal{N}_g(N|_u)$ can be generated in $O(|E(N|_u)| \times |V(N|_u)|$ time, because there are $O(|V(N|_u)|)$ possible admixture nodes and the algorithm from Huber *et al.* (2019), which scales linearly in the $|E(N|_u)|$, can be used to reorient the network for each selection. Furthermore, the likelihood of both bases trees for each $M \in \mathcal{N}_g(N|_u)$ can be evaluated when using `TreeMix`. We conjecture that finding the MLNO of $N|_u$ is NP-hard when h is not fixed, in which case heuristics would need to be developed for when $V(N|_u)$ or h are large. As many published admixture graphs, including the ones that we study in Section 5, have a small number of leaves and admixture events, we do not address the issue of scalability in our study. Rather, we focus on the utility of solving the MLNO problem when estimating admixture graphs with STB-ML methods.

We conclude this section by summarizing the subroutines that we have implemented within the latest version of `TreeMix` (v1.13 Revision 231) and evaluate in our experimental study. Like the original code from Pickrell and Pritchard (2012), our implementation, which we refer to as `OrientAGraph`, is provided under the GNU General Public License v3.0 (`https://github.com/ekmolloy/OrientAGraph`). `OrientAGraph` estimates an admixture graph $(N, \mathbf{\Theta})$ with h admixture events in the following steps:

1. Estimate an ML starting tree N_0, rooting it at the outgroup g.
2. For $i = 1, 2, \ldots, h$:
   (a) $(N_i, \mathbf{\Theta}_{N_i}) \leftarrow$ Find the ML network that can be obtained by adding an admixture edge e to N_{i-1}
   (b) $(N_i, \mathbf{\Theta}_{N_i}) \leftarrow$ Find the ML network in orientation neighborhood of $N_i|_u$.
   (c) $(N_i, \mathbf{\Theta}_{N_i}) \leftarrow$ Search from N_i for a network of a higher likelihood.
3. Return $(N_i, \mathbf{\Theta}_{N_i})$

Steps 1 and 2c are the *same* as `TreeMix`. Step 2a is *different* than `TreeMix`, because we evaluate the likelihood of all legal edge additions rather than using a heuristic to evaluate a subset of them. Step 2b is *new* to our method (note that our current implementation evaluates the likelihood of all base trees). We acknowledge that our method does more work that `TreeMix` and thus will always be slower. However, current trends focus on evaluating all possible admixture graphs with a fixed h, so evaluating more network topologies seems reasonable given that `TreeMix` is one of the most scalable methods at present.

5 Experimental Study

While we know that the MLNO problem is important to our case study, our goal is to evaluate whether it is important to realistic models of admixture. We utilized models published in prior studies and benchmark

methods on the exact f-statistics implied by the models, representing the case of infinite data. To evaluate the impact of finite data on our results, we also benchmark methods on f-statistics (and standard errors) computed from genome sequence data simulated under these models. We refer to the former as 'model data sets' and the latter as 'simulated data sets'.

5.1 Model Data Sets

We used the `admixturegraph` R package (Leppälä *et al.*, 2017) to create the f-statistic data sets for eight models. Model M1 is based on our case study (Figure 1a), M2 is based on the toy example shown in Figure 5 of Patterson *et al.* (2012) used to benchmark `qpGraph` (Supplementary Figure S1), and all other models are based on admixture graphs estimated from biological data sets in prior studies (see Figure 3 for details). For these experiments, the standard error of the f-statistic was set to 0.0001 to avoid numerical problems.

5.2 Simulated Data Sets

We simulated data sets from three demographies from models: M1, M5, and M6 using `ms` (Hudson, 2002) with the number of loci set to $3,000$, the number of sites per locus set to 1 Megabase, the effective population size above the root (i.e. N_0) set to $10,000$, the sample size for each population set to 20, the mutation rate μ set to 1.25×10^{-8}, and the recombination rate r set to 1×10^{-8}. The models based on biological data sets had branch lengths given in units of genetic drift and were not dated, so we did not know the timing of

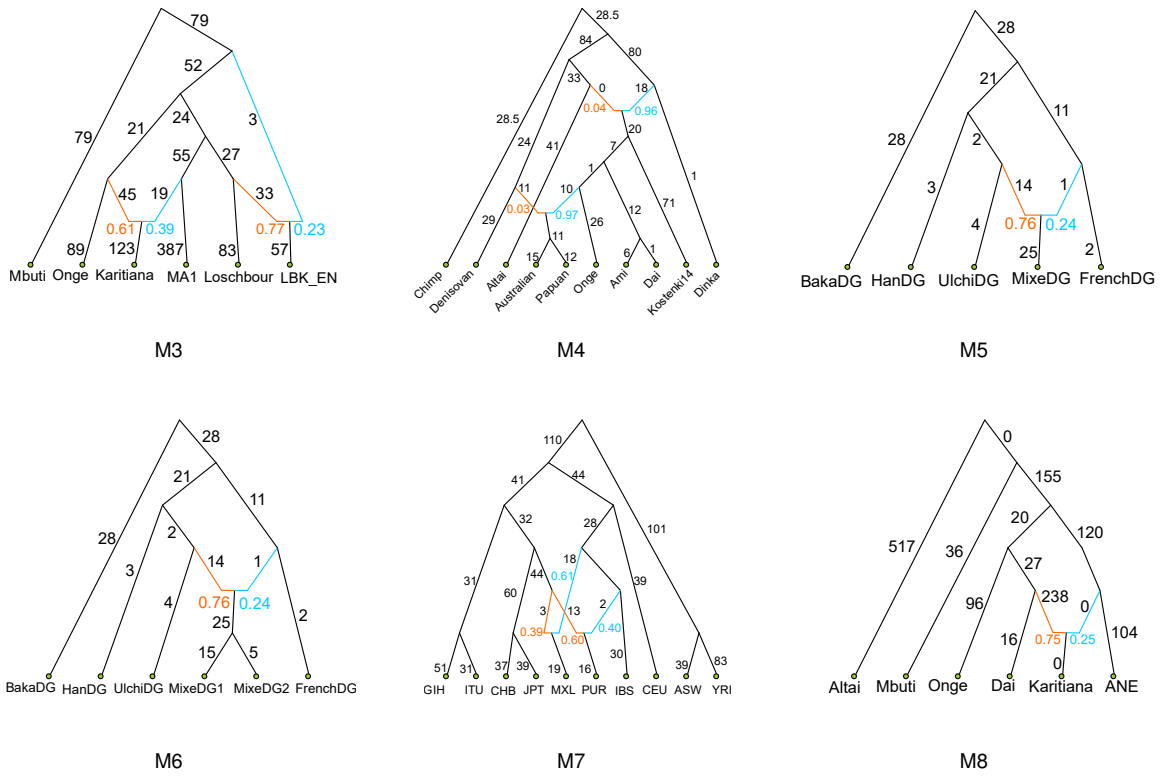

Figure 3: This figure shows the model demographies evaluated in our experimental study. M3 is Figure S8.1 from Haak *et al.* (2015), M4 is Figure S11.1 from Mallick *et al.* (2016) but simplified so that there are only two admixture events, M5 is Figure 5a from Lipson (2020), M6 is M5 but extended so that MixeDG splits into two populations, M7 is Figure 7a from Wu (2020), and M8 is Figure 2 from Yan *et al.* (2020). All of these admixture graphs were estimated from biological data sets. Note that the branch lengths in these subfigures are not drawn to scale and should be divided by 1,000.

each internal node. To simulate data with `ms`, we assigned timings at the internal nodes and then re-scaled the population size on each branch so that it had the correct length in drift units. The maximum number of SNPs per locus was approximately 6000, 3000, and 3500 for M1, M5, and M6, and we set the block size accordingly, when using `TreeMix` to estimate f_2-statistics, along with their standard error (note that we also performed experiments computing the covariance matrix instead of f_2-statistics to confirm that this did not impact results). Lastly, we repeated this process of estimating matrices with `TreeMix` using SNPs from the first 100, 500, 1000, 1500, 2000, and 2500 loci. This produced a total of 18 simulated data sets.

5.3 Method Evaluation

Our primary focus was comparing `TreeMix` to `OrientAGraph`, which is based on `TreeMix`. As discussed in as see Section 4, there are two subroutines that differ between these methods (MLNO and the number of admixture edge additions that are evaluated). We ran `OrientAGraph` with both of these subroutines, as well as with each of the subroutine separately, to understand their impact. Otherwise, we ran all methods with same options: specifying the outgroup population, specifying the (correct) number of admixture events to be added, and executing additional search moves from the starting tree, after reconstructing it with random population addition. For model data sets, we generated all possible population addition orders, selected 100 uniformly at random, and ran `TreeMix` and `OrientAGraph` using each of these orders. For simulated data sets, we ran `TreeMix` and `OrientAGraph` using all possible population orders.

We also benchmarked a recently published method, `miqograph` (Yan *et al.*, 2020), which is guaranteed to find the true admixture graph topology, provided that it meets some topological constraints. Our primary goal was to benchmark `miqograph` for running time, but it was difficult to set-up a large number of experiments (e.g. using a cluster system), because of how the industrial solver, Gurobi, handles academic licenses. Therefore, we ran `miqograph` on the model data sets only, and all experiments where running time is reported were performed on the same shared computing resource (Intel(R) Xeon(R) CPU 2.10GHz server with 128 GB RAM and 32 cores). We ran `miqograph` given the same information as `TreeMix`, including the outgroup and the (correct) number of admixture events. `miqograph` also requires the user to specify the granularity with which admixture proportions can be estimated and the depth of its "search tree." We set the granularity to 4 and the depth to half the number of leaves (i.e. populations in S). Lastly, we allowed `miqograph` to use all available threads (note that `TreeMix` and `OrientAGraph` are single threaded).

Methods were compared in terms of three measures: the log-likelihood score of the estimated admixture graph (for `TreeMix` and `OrientAGraph` only), the topological accuracy (specifically the triplet distance) between the true and estimated admixture graph topology, both rooted at the outgroup (Jansson *et al.*, 2019), and the runtime (in seconds); see the Supplementary Text for details.

6 Results and Discussion

In this section, we report the results of running `TreeMix`, `OrientAGraph`, and `miqograph` on model datasets (Table 1) and the results of running `TreeMix` and `OrientAGraph` on simulated data sets (Figure 4). Similarities and differences between the methods are also discussed.

6.1 Topological Accuracy and Likelihood Score for Model Data Sets

Models M2, M3, and M8. For these three models, all methods recovered the true admixture graph topology.

Model M1. For our case study model, we observed the results expected based on Section 3; `TreeMix` recovered the incorrect admixture graph topology whereas `OrientAGraph` recovered the correct topology.

Models M5 and M6. We observed similar results to our case study (M1) on models M5 and M6. Model M5 was estimated manually with `qpGraph` by Lipson (2020) using data from the Simmons Genome Diversity Project (Mallick *et al.*, 2016). We found that `TreeMix` returned the incorrect admixture graph topology. Both `OrientAGraph` and `miqograph` returned the correct admixture graph topology. This difference in

Table 1: Results for estimating admixture graphs given the f-statistics implied by the models in Figure 3 using `TreeMix`, `OrientAGraph`, and `miqograph`. We report the log-likelihood score (for `TreeMix` and `OrientAGraph` methods only), the topological error between the model admixture graph and the estimated admixture graph, and the running time in seconds. Running time is averaged across 100 runs for `TreeMix` and `OrientAGraph` using different population orders. *Indicates there was a difference in the log-likelihood score or triplet distance for the graphs returned on different runs, in which case, we report the average. ⋆Indicates that `miqograph` cannot get the topology correct, because of the position of the admixture event(s). †Indicates that `miqograph` did not complete within the time limit of 10,000 seconds (2.78 hours) and thus was not solved to optimality.

Model	Log-likelihood		Topological Error			Running Time		
	`TreeMix`	`OrientAGraph`	`TreeMix`	`OrientAGraph`	`miqograph`	`TreeMix`	`OrientAGraph`	`miqograph`
M1	-366944	83	9	0	0	0.06	0.19	398
M2	83	83	0	0	0	0.04	0.15	802
M3	124	124	0	0	0	0.28	2.12	2340
M4⋆	103*	373	14	0	81†	1.28	11.72	10000
M5	-33	83	9	0	0	0.05	0.16	170
M6⋆	-30	124	15	0	2	0.11	0.31	556
M7	-2883	-2883	18.6*	20	82†	5.58	13.20	10000
M8	124	124	0	0	0	0.09	0.34	264

topology was also reflected in the log-likelihood scores of the returned graphs. Exploring this data set further, we found that the NJ tree computed for this data set and rooted at the outgroup cannot be transformed into the correct network through the addition of an edge. We expect that the results of our case study are applicable to M5, although we did not perform exhaustive search to confirm.

It is worth noting that we were not aware of M5 when we formulated our case study, but after finding this result, we extended M5 to have two populations descending from the admixed population (Mixe) to explore whether this phenomenon was unique to the depth of the admixed population (Model M6). This does not seem to be the case, as we observed similar trends for M6. Note that `miqograph` is guaranteed to recover the incorrect network topology for M6, because of its topological constraint that admixture events be at the leaves or else nested; this is reflected in our results.

For each of the six models discussed above, there was either no difference between `TreeMix` and `OrientAGraph` or else `OrientAGraph` improved the accuracy of `TreeMix`. This is expected as `OrientAGraph` is `TreeMix` with the MLNO subroutine and an exhaustive the search for the best admixture edge addition. By running these options separately, we confirmed that these results for models M1, M5, and M6 are driven by the MLNO subroutine option (Supplementary Table 1).

Model M4. Model M4 is an admixture graph presented with the publication of Simmons Genome Diversity data set (Mallick *et al.*, 2016); however, we simplified the published model so that it only had two admixture events: gene flow from Denisova into the ancestors of Australians and Papuans as well as gene flow from Neanderthals into modern human, non-Africans. After 2.78 hours, the graph returned by `miqograph` graph had a triplet distance of 81 to the correct admixture graph topology; however, it is unclear how much this score would be improved with a longer running time, given that these admixture events are not incident to the leaves or nested (so `miqograph` is guaranteed to return the incorrect topology). In contrast, `TreeMix` and `OrientAGraph` completed in less than 15 seconds, with a triplet distance of 15 and 0 to the correct admixture graph respective. This difference was also reflected in the log-likelihood scores of the returned graph. Unlike the results for models M1, M5, and M6, these results were driven by the exhaustive search for the best admixture edge addition rather than the MLNO subroutine, suggesting that heuristics employed by `TreeMix` add admixture edges are not sufficient for this model.

Model M7. This model admixture graph was produced by running `GTMix` (Wu, 2020) on data from the third phase of the 1000 Genomes project (The 1000 Genomes Project Consortium, 2015). This model is interesting as `TreeMix` produced a different topology from `GTMix` on this data set, although it should be noted that `GTMix` takes estimated genealogies rather than f-statistics as input. Our question is whether the issue with `TreeMix`'s performance could be related to its search procedure, because `TreeMix` can in principle

recover the correct admixture graph, as it is tree-based. In addition, `miqograph` should recover the correct topology if run to optimality, as both admixture events are incident to the leaves. `miqograph` did not solve its problem to optimality within our allowed time frame of 10,000 seconds, returning a graph with a triplet distance of 82 from the true graph. This suggests that users may want to be wary of the results produced by `miqograph`, when it does not succeed in solving its problem to optimality. `TreeMix` and `OrientAGraph` also failed to recover the correct topology. Both methods returned graphs with a log-likelihood score of -2883 for all 100 runs using different population orders; however, the triplet distance of these estimated graph topologies to the correct graph varied: either 16 or 20. We estimated numerical parameters for the correct graph topology using `OrientAGraph` and found the resulting model had a log-likelihood score of 373. This suggests that these `TreeMix` and `OrientAGraph` failed to recover the correct topology due to falling into a local optimum. It is worth noting that model M7 is the only graph that we studied where a single vertex is a source for two different admixture edges (using terminology from phylogenetic networks, it is the only network that we studied that is not tree-child). Overall, these results suggest that M7 is an interesting for method developers.

6.2 Runtime Comparison for Model Data Sets

Differences in runtimes were not pronounced for 6/8 models, which had at most 7 populations at the leaves and 2 admixture events (see Table 1). For the other two models with 10 populations and 2 admixture events, there were notable differences in the runtime of methods. Notably, `miqograph` did not solve its problem to optimality within the allowed time frame for our study (10,000 seconds), whereas `TreeMix` and `OrientAGraph` always completed in less than 30 seconds. `OrientAGraph` was up to 10x slower than `TreeMix`, but this difference did not impact the usability of `OrientAGraph` for the data sets evaluated in this study.

6.3 Simulated Data Sets

We also evaluated `TreeMix` and `OrientAGraph` on genome-scale data sets simulated from models M1, M5, and M6 to evaluate the impact of finite numbers of loci. Provided the number of loci was greater than 1000, `OrientAGraph` recovered the correct topology and `TreeMix` recovered the incorrect topology (Figure 4), which is consistent with our results from the model data sets. For M5 and M6, `TreeMix` and `OrientAGraph` were not always able to recover the correct graph given small numbers of loci ($< 1,000$), reflecting larger differences between the estimated and the true f_2-statistics for smaller numbers of loci.

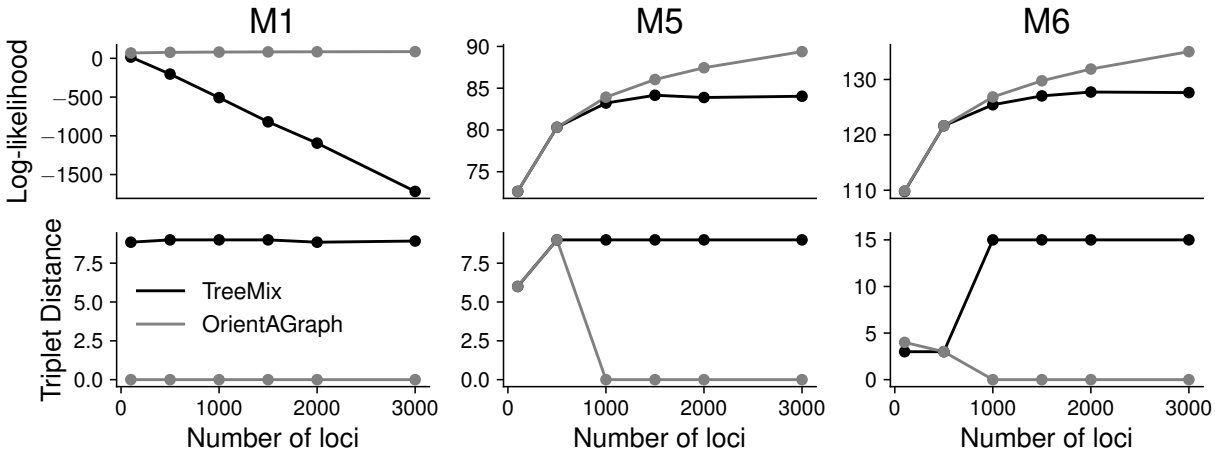

Figure 4: Results from simulated data ranging from 100 loci to 3000 loci. Note that we are showing averages across runs of `TreeMix` for each population order (120 combinations for M1 and M5 and 720 combinations for M6). On these data sets, the results are consistent across runs so we only show the average. While we show both log-likelihood and triplet distance, we do not expect these two measures are not fully correlated (i.e. graphs with lower triplet scores may not have a lower log-likelihood scores).

7 Conclusion

Our work proposes a new search strategy based on network reorientation in the context of starting-tree-based maximum likelihood methods. While our results suggest that such a strategy leads to improved admixture graph inference, fully automated admixture graph inference presents a number of additional challenges as well as directions for future work.

Our current implementation of `OrientAGraph` relies on exhaustive search to solve the MLNO problem. Exhaustive search is also used to compute the likelihood for different base trees in the reoriented admixture graph, pointing to a challenge of estimating numerical parameters for admixture graphs in an automated fashion. While `OrientAGraph` remains efficient for the populations and admixture graphs considered here, we expect that `OrientAGraph` will not scale to very large numbers of populations and admixture events. Developing inference methods that are accurate while scaling to large numbers of populations is an important direction for future work.

In our current study, we have focused on admixture graph inference methods that use f-statistics as input. Recent work has shown how admixture graphs can be inferred from estimated genealogical trees (Wu, 2020). It would be of interest to compare our method to these approaches and to assess whether network orientation can improve admixture graph inference in this setting, although it is worth noting that some empirical evidence suggests that starting trees are not always problematic for methods that use genealogical trees (Cao *et al.*, 2019).

Funding

This work was funded by the following grants: NSF GRFP DGE-1650604 (A.D.), NIH R35GM125055 (S.S. and E.M.), and an Alfred P. Sloan Research Fellowship (S.S.).

References

Cao, Z., Zhu, J., and Nakhleh, L. (2019). Empirical performance of tree-based inference of phylogenetic networks. In K. T. Huber and D. Gusfield, editors, *19th International Workshop on Algorithms in Bioinformatics, WABI 2019, September 8-10, 2019, Niagara Falls, NY, USA*, volume 143 of *LIPIcs*, pages 21:1–21:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Edelman, N. B., Frandsen, P. B., Miyagi, M., *et al.* (2019). Genomic architecture and introgression shape a butterfly radiation. *Science*, **366**(6465), 594–599.

Francis, A. R. and Steel, M. (2015). Which Phylogenetic Networks are Merely Trees with Additional Arcs? *Systematic Biology*, **64**(5), 768–777.

Gambette, P., Berry, V., and Paul, C. (2012). Quartets and Unrooted Phylogenetic Networks. *Journal of Bioinformatics and Computational Biology*, **10**(04), 1250004.

Gambette, P., van Iersel, L., Jones, M., Lafond, M., Pardi, F., and Scornavacca, C. (2017). Rearrangement moves on rooted phylogenetic networks. *PLOS Computational Biology*, **13**(8), 1–21.

Green, R. E., Krause, J., Briggs, A. W., *et al.* (2010). A Draft Sequence of the Neandertal Genome. *Science*, **328**(5979), 710–722.

Haak, W., Lazaridis, I., Patterson, N., *et al.* (2015). Massive migration from the steppe was a source for Indo-European languages in Europe. *Nature*, **522**, 207–211.

Harney, É., Patterson, N., Reich, D., and Wakeley, J. (2020). Assessing the Performance of qpAdm: A Statistical Tool for Studying Population Admixture. *bioRxiv*.

Huber, K. T., van Iersel, L., Janssen, R., Jones, M., Moulton, V., Murakami, Y., and Semple, C. (2019). Rooting for phylogenetic networks. *arXiv*, **CoRR**, abs/1906.07430.

Hudson, R. R. (2002). Generating samples under a Wright–Fisher neutral model of genetic variation . *Bioinformatics*, **18**(2), 337–338.

Janssen, R., Jones, M., Erdös, P., van Iersel, L., and Scornavacca, C. (2018). Exploring the Tiers of Rooted Phylogenetic Network Space Using Tail Moves. *Bulletin of Mathematical Biology*, **80**(8), 2177–2208.

Jansson, J., Mampentzidis, K., Rajaby, R., and Sung, W.-K. (2019). Computing the Rooted Triplet Distance Between Phylogenetic Networks. In C. J. Colbourn, R. Grossi, and N. Pisanti, editors, *Combinatorial Algorithms*, pages 290–303, Cham. Springer International Publishing.

Leppälä, K., Nielsen, S. V., and Mailund, T. (2017). admixturegraph: an R package for admixture graph manipulation and fitting. *Bioinformatics*, **33**(11), 1738–1740.

Lipson, M. (2020). Applying f4-statistics and admixture graphs: Theory and examples. *Molecular Ecology Resources*, **20**(6), 1658–1667.

Lipson, M., Loh, P.-R., Levin, A., Reich, D., Patterson, N., and Berger, B. (2013). Efficient Moment-Based Inference of Admixture Parameters and Sources of Gene Flow. *Molecular Biology and Evolution*, **30**(8), 1788–1802.

Lipson, M., Loh, P.-R., Patterson, N., Moorjani, P., Ko, Y.-C., Stoneking, M., Berger, B., and Reich, D. (2014). Reconstructing Austronesian population history in Island Southeast Asia. *Nature Communications*, **5**(1), 4689.

Mallick, S., Li, H., Lipson, M., *et al.* (2016). The Simons Genome Diversity Project: 300 genomes from 142 diverse populations. *Nature*, **538**, 201—-206.

McDiarmid, C., Semple, C., and Welsh, D. (2015). Counting Phylogenetic Networks. *Annals of Combinatorics*, **19**, 205–224.

Patterson, N., Moorjani, P., Luo, Y., Mallick, S., Rohland, N., Zhan, Y., Genschoreck, T., Webster, T., and Reich, D. (2012). Ancient Admixture in Human History. *Genetics*, **192**(3), 1065–1093.

Peter, B. M. (2016). Admixture, Population Structure, and F-Statistics. *Genetics*, **202**(4), 1485–1501.

Pickrell, J. K. and Pritchard, J. K. (2012). Inference of Population Splits and Mixtures from Genome-Wide Allele Frequency Data. *PLOS Genetics*, **8**(11), 1–17.

Pilot, M., Moura, A., Okhlopkov, I., *et al.* (2019). Global Phylogeographic and Admixture Patterns in Grey Wolves and Genetic Legacy of An Ancient Siberian Lineage. *Scientific Reports*, **9**, 17328.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4**(4), 406–425.

The 1000 Genomes Project Consortium (2015). A global reference for human genetic variation. *Nature*, **526**, 68–74.

Wen, D., Yu, Y., Zhu, J., and Nakhleh, L. (2018). Inferring Phylogenetic Networks Using PhyloNet. *Systematic Biology*, **67**(4), 735–740.

Wu, Y. (2020). Inference of population admixture network from local gene genealogies: a coalescent-based maximum likelihood approach. *Bioinformatics*, **36**(Supplement_1), i326–i334.

Yan, J., Patterson, N., and Narasimhan, V. (2020). miqoGraph : Fitting admixture graphs using mixed-integer quadratic optimization. *Bioinformatics*. btaa988.

Yu, Y., Dong, J., Liu, K. J., and Nakhleh, L. (2014). Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences*, **111**(46), 16448–16453.