ScisTree2: An Improved Method for Large-scale Inference of Cell Lineage Trees and Genotype Calling from Noisy Single Cell Data

Haotian Zhang[1], Yiming Zhang[1], Teng Gao[2], and Yufeng Wu[1,3]

1. School of Computing, University of Connecticut Storrs, CT 06269, U.S.A.

2. Department of Biomedical Informatics, Harvard Medical School, Boston, MA, USA

3: Corresponding author: yufeng.wu@uconn.edu

Abstract

In a multicellular organism, cell lineages share a common evolutionary history, which has important applications in the study of development, aging, and cancer. Cell lineage tree is a model for the evolutionary history of cells sampled from an organism. The inference of cell lineage trees has been made possible by recent developments in single-cell sequencing. However, single-cell data is well known to be sparse and noisy. Moreover, the size of single cell data is increasing rapidly, both in the number of assayed cells and the amount of genomic information captured per cell. Thus, accurate inference of cell lineage tree from large single-cell data is a computational challenge. In this paper, we present ScisTree2, a fast and accurate cell lineage tree inference and genotype calling approach based on the infinite sites model. Similar to existing maximum likelihood-based inference approaches such as ScisTree, ScisTree2 relies on local search for finding optimal trees. Compared to the original ScisTree, the main new feature of ScisTree2 is a local search based on subtree prune and regraft (SPR) that is not implemented in the original ScisTree due to its high computational cost. This new local search is more general than that in ScisTree and significantly improves the accuracy of cell lineage tree inference. The key technical contributions of this work are the efficient SPR local search algorithms that can scale to very large data. ScisTree2 can call single cell genotypes based on the inferred cell lineage tree. Experiments on simulated and real biological data show that ScisTree2 achieves better overall accuracy while significantly outperforms in efficiency when compared with existing methods. To the best of our knowledge, ScisTree2 is the first model-based cell lineage tree inference and genotype calling approach that can infer trees with very large data with, e.g., tens of thousands of cells.

Keywords. Phylogeny inference, cell lineage tree, single cell sequencing, probabilistic inference, perfect phylogeny and algorithms.

1 Introduction

An adult human body contains about 30 trillion cells, which perform diverse functions. These cells are interconnected via a shared cellular division history that spans development, aging and disease. This evolutionary history of sampled cells from an organism can be depicted by a cell lineage tree, a fundamental model integral to the study of tumor evolution and developmental biology [1, 2, 13, 15]. A cell lineage tree is a rooted leaf-labeled tree. The leaves of the tree are the extant cells. The internal nodes are the progenitor cells that are ancestral to the extant cells. To simplify notations, in the following, we use tree and cell lineage tree interchangeably.

Reconstructing cell lineage tree from noisy single cell DNA data has been actively studied in the literature recently. Existing methods differ in their modeling assumptions and also computational approaches. First, mutational models are important for single cell DNA data analysis because mutations lead to genomic variants which are the main type of data for cell lineage tree inference. The main genomic variant studied in this paper is single nucleotide variant (SNV). We do not use more complex variants such as copy number variations (CNVs) for cell lineage tree inference. There are two main modeling assumptions regarding to the mutational models for SNV variants: the infinite sites (IS) model and the finite sites model. The IS model assumes each mutant variant only arises *once* in the evolutionary history (i.e., there is no recurrent mutations). For example, in the cell lineage tree shown in Fig. 1(a), there is a single mutation for each SNV site. Many existing approaches (e.g., [4, 18, 6]) assume the IS model. An alternative model is the finite sites model (e.g., [19, 5]), which allows recurrent mutations at a genomic site. Existing cell lineage tree reconstruction methods also differ in the high-level computational approaches. Several methods are based on probabilistic inference and use Markov chain Monte Carlo (MCMC) [4, 19]. A major downside of these methods is that they cannot handle large data. An alternative to MCMC is using maximum likelihood and local search for finding optimal trees and is much faster than MCMC. Several newer methods including *ScisTree* [18] and *CellPhy* [5] adopt this approach. There are also parsimony-based approaches. For example, *Huntress* [6] aims at making the fewest changes to genotypes so that the changed data satisfies the infinite sites model.

In [18], we developed a cell lineage tree inference method called *ScisTree*. *ScisTree* assumes the IS model and finds optimal trees through local search of tree space based on maximum likelihood. *ScisTree* first builds an initial tree from the called genotypes using the well-known neighbor joining algorithm. It then iteratively evaluates and finds the tree that has the maximal likelihood among trees that are topologically similar to the current tree. *ScisTree* uses a likelihood model that allows efficient computation for a given tree. Simulation shows that *ScisTree* can infer accurate trees and is much faster than several existing methods, including *SCITE* [4]. However, *ScisTree* becomes slow when the number of cells exceeds $1,000$. Single cell DNA data with tens of thousands of cells are becoming available. For example, a recently developed variant caller, SComatic, can detect somatic SNVs from scRNA-seq and scATAC-seq, which typically contain 1,000s to 10,000s of cells [12]. At present, no existing cell lineage tree inference methods can work for very large data. Thus, a new cell lineage tree inference method that can perform accurate lineage inference for very large data is needed.

In this paper, we present *ScisTree2*, an improved cell lineage tree inference approach over the original *ScisTree*. *ScisTree2* has the following features.

1. *ScisTree2* implements efficient subtree prune and regraft (SPR) local search which is not implemented the original *ScisTree*. The SPR local search can work for trees with as many as $10,000$ cells and can lead to much more accurate inference than the NNI local search as implemented in the original *ScisTree*.
2. *ScisTree2* implements a fast branch and bound method that significantly reduces the computational time.
3. *ScisTree2* calls single cell genotypes based on the inferred cell lineage tree.

Through experiments on both simulated and real biological data, we show that *ScisTree2* outperforms the existing methods for cell lineage tree inference and genotype calling. The main advantage of *ScisTree2* is that it runs much faster than all existing cell lineage tree inference methods (except the classic distance-based tree inference methods such as neighbor joining). *ScisTree2* can infer cell lineage trees with $10,000$ or more cells.

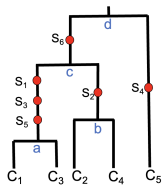
2 Preliminaries

We are given single cell DNA sequencing data (reads). The first step is using a genotype caller (e.g., GATK) to call the genetic variants from the single cell sequence reads. Most genotype callers output called genotypes along with the genotype probability for each possible genotype of each cell at a site. For simplicity, *ScisTree2* only considers two genotypes: wild-type (allele 0) and mutant (allele 1). It is well known that single cell data is very noisy, which can lead to wrongly called genotypes. One of the major source of noises is allelic dropout (ADO), which is quite common in single cell data. ADO can lead to less or even no sequence reads at a specific SNV site for a cell. The rate of ADO depends on specific single cell sequencing technologies and may vary from experiments to experiments. In the current single cell data, ADO can be 50% or even higher. ADO may lead to a wrongly called wild-type genotype while the true genotype is the mutant (i.e., producing a false negative error). Sequencing error in single cell DNA data is also common, which may lead to a wild-type allele being wrongly called to a mutant (i.e., producing a false positive error). There are also other kinds of errors, such as amplification error and doublets. Noise in single cell data implies that the called genotypes from single cell data has significant uncertainty, which poses a major challenge for cell linage tree inference.

The most common way for dealing with the noise and uncertainty in single cell genotypes in the existing cell lineage tree inference literature is assuming each genotype has the *uniform* error probability. A main disadvantage of the uniform genotype error model is the loss of uncertainty information contained in individual genotypes. *ScisTree* and *ScisTree2* take the so-called genotype probability matrix M as input. For each cell c and each site s, $M[c, s]$ is equal to the *posterior* probability of the cell s having the *wild-type* genotype at the site s. Part 1(c) of Fig. 1 shows an example of the genotype probability matrix. There are five cells and six SNV sites. The infinite sites assumption imposes a restriction on the pattern of SNV genotypes. The most straightforward way of calling genotypes from M is taking the allele (wild-type or mutant) that has larger probability. However, genotypes called this way may violate the IS model. As an example, for the site S_6 of the cell C_3, $M[3, 6] = 0.7$, which implies this genotype is the wild-type (0) since the wild-type has higher probability at this position. However, the called genotype is different from the true genotype shown in Fig. 1(b). Note that the noise (e.g., ADO or sequencing error) in single cell data implies that we cannot simply take the genotypes with larger probability at each position.

2.1 Computing the probability of uncertain genotypes on a fixed tree

Maximum likelihood is a popular phylogenetic inference approach. Computing the likelihood of single cell sequence data D for a given tree T is in principle straightforward to compute: $Pr(D|T) = \sum_G Pr(D|G)Pr(G|T)$. Here, G refers to the genotypes at the leaves of T. Directly computing the likelihood by enumerating G using the above equation is infeasible. *CellPhy* computes the likelihood $Pr(G|T)$ by incorporating the uncertainty of G at each leaf of T using a modified version of the well-known Felsenstein's algorithm. One major computational difficulty for the *CellPhy* likelihood is its complexity: T has branch lengths and $Pr(D|T)$ needs to (implicitly) sum over all possible alleles at internal nodes. This leads to relatively complex search for optimal trees. This is perhaps the reason why *CellPhy* becomes slow for large number of cells (say $10,000$) despite the fact that *CellPhy* builds on top of RAxML-NG, one of the most well-engineered phylogenetic tree inference tools. Moreover, the *CellPhy* likelihood can not be easily extended to allow the IS model.



(a) True cell lineage tree

	S_1	S_2	S_3	S_4	S_5	S_6
C_1	1	0	1	0	1	1
C_2	0	1	0	0	0	1
C_3	1	0	1	0	1	*1*
C_4	0	1	0	0	0	1
C_5	0	*0*	0	1	0	0

(b) True genotypes

	S_1	S_2	S_3	S_4	S_5	S_6
C_1	0.01	0.8	0.02	0.9	0.01	0.05
C_2	0.6	0.02	0.8	0.9	0.8	0.02
C_3	0.08	0.7	0.02	0.98	0.01	**0.7**
C_4	0.8	0.01	0.8	0.8	0.8	0.05
C_5	0.7	**0.3**	0.9	0.02	0.9	0.9

(c) M: the probability matrix of genotypes being the wild-type

Figure 1: *Illustration of cell lineage tree, genotypes and data with uncertainty. Part 1(a): the true cell lineage tree with five cells and six sites (with mutations labeling branches). Infinite sites model: one mutation per site. Four internal nodes: a to d. Part 1(b): the true (binary) genotypes. Part 1(c): probabilities of genotypes being the wild-type (0). The two boldfaced positions: true genotypes don't agree with that called by taking the alleles with larger probability at each position.*

ScisTree, on the other hand, uses a much simpler (but more heuristic) probabilistic model, which is designed for the IS model. *ScisTree* aims at maximizing the **posterior** probability $Pr(G^*|D)$ for some genotypes G^* among all genotypes G that satisfy the IS model. We say G satisfy the IS model if there exists a (possibly multifurcating) phylogeny T (called *perfect phylogeny* [3]) where leaves are labeled by the cells (rows) in G, and each column (site) c of G labels a single branch of T such that exactly the cells below this branch are the mutants of c. See Figs. 1(a) and 1(b) for an illustration. The probability of G is calculated as follows. We let $G(c, s)$ be the (binary) genotype of the cell c at the site s. Then, $Pr(G|D) = \prod_{c=1}^{n} \prod_{s=1}^{m} M[c, s]^{1-G(c,s)}(1 - M[c, s])^{G(c,s)}$, where G satisfies the IS model, n is the number of cells and m is the number of SNV sites. Here, we assume each genotype at a site s for a cell c is independent. Finding $G^* = \arg\max_G Pr(G|D)$ for the given genotype probability matrix M is known to be NP complete [18]. A key insight made in *ScisTree* is, if the underlying (rooted and binary) tree T is *given*, then G^* can be found in $O(nm)$ time by finding the best branches to place mutations on the given T as follows. Recall that the set of mutants of a site s must form a *single* clade (subtree) in T. Then, for each branch ending at node v in T, we define $Q_v(s)$ as

$$Q_v(s) = \prod_{c \in taxa(v)} \frac{1 - M[c, s]}{M[c, s]} \tag{1}$$

where $taxa(v)$ is the set of leaves below v. Recall that $M[c, s]$ is the probability of the genotype of the cell c and the site s is the wild-type. $Q_v(s)$ only concerns the cells within the subtree rooted at v, and has a simple product form. Let v_1 and v_2 be the two children of v. Then, $Q_v(s) = Q_{v_1}(s)Q_{v_2}(s)$. Thus, we can calculate all $Q_v(s)$ in $O(n)$ time for each site in a bottom-up way using dynamic programming. We denote the set of nodes of T as $nodes(T)$. The maximum posterior probability $Pr(G^*|T, D)$ of genotypes conditional on T is then:

$$P(T) = Pr(G^*|T, D) = \prod_{s=1}^{m} \left[\prod_{c=1}^{n} M[c, s] \right] \max_{v \in nodes(T)} Q_v(s) \tag{2}$$

Example We use the site S_1 in Fig. 1(c) to show how to find where its mutation is located on the tree in Fig. 1(a). We write $Q_v(1)$ as Q_v here to simplify the notations. At leaves, $Q_{C_1} = \frac{1-0.01}{0.01} = 99, Q_{C_2} = \frac{2}{3}, Q_{C_3} = \frac{23}{2}, Q_{C_4} = \frac{1}{4}$ and $Q_{C_5} = \frac{3}{7}$. For internal nodes, $Q_a = Q_{C_1}Q_{C_3} = \frac{2277}{2}, Q_b = Q_{C_2}Q_{C_4} = \frac{1}{6}, Q_c = Q_aQ_b = \frac{2277}{12}$, and $Q_d = Q_cQ_{C_5} = \frac{2277}{28}$. We place the mutation for S_1 at the branch right above a because Q_a is the *largest* among all the Q values for S_1.

ScisTree finds the T^* that gives the maximum posterior probability $P(T^*)$ using local search. At each local search step, *ScisTree* examines all the $O(n)$ trees within a single nearest neighbor interchange (NNI) move. Each tree takes $O(mn)$ to compute its posterior probability. Therefore, the running time is $O(Kn^2m)$ for NNI local search where K is the number of iterations which depends on how far the initial tree is from a local optima. When data size is relatively large (say $n = m = 1,000$), *ScisTree* becomes slow.

3 Method

3.1 The high-level approach

On the high level, *ScisTree2* uses the same local search approach as *ScisTree*:

1. Construct an initial cell lineage tree (see Sect. A.5 in the Supplemental Materials).
2. Iteratively search for an optimal cell lineage tree by examining and choosing the tree that gives the maximal posterior probability among all "neighboring" trees that are within a single tree rearrangement move from the current tree.
3. Once optimal cell lineage tree is inferred, genotypes of cells at each site are called by placing the single mutation on the tree to obtain the optimal posterior probability.

While the local search is simple in principle, developing an effective local search approach that can scale to tens of thousands of cells is not so straightforward. The most important aspect of a local search approach is that evaluating neighboring tree should be performed efficiently. The original *ScisTree* only implements the NNI local search, but not the SPR local search due to its high computational cost. The key insight by *ScisTree2* is that the running time of the SPR local search can be reduced to $O(Kn^2m)$ from the naïve $O(Kn^3m)$ time (where K is the number of iterations). Moreover, the SPR local search can be further speeded up by a branch and bound heuristic. This allows *ScisTree2* to perform SPR local search faster than the NNI search in the original *ScisTree*. Moreover, the SPR local search in *ScisTree2* finds more accurate cell lineage trees and genotypes than the original *ScisTree*.

3.2 Efficient SPR local search

We call the set of trees that can be obtained by a single tree rearrangement operation (e.g., NNI or SPR) as the 1-operation neighborhood of the current tree. The original *ScisTree* only implements the NNI local search. A main disadvantage of the NNI local search is that the 1-NNI neighborhood is rather restricted. This makes NNI local search easily get trapped at local optima. An alternative to NNI is (rooted) subtree prune and regraft (SPR), which is more commonly used in phylogeny inference. An SPR operation prunes (i.e., detaches) a subtree of a tree T and regrafts (i.e., re-attaches) to another branch of T. While the 1-NNI neighborhood of a tree T with n taxa is of size $O(n)$, the size of 1-SPR neighborhood of T is $O(n^2)$, and contains the 1-NNI neighborhood as a subset. Naïvely, one can calculate the likelihood for each of the $O(n^2)$ trees in the 1-SPR neighborhood. But this leads to $O(n^3m)$ running time for n cells and m SNV sites, which is too slow even for data of moderate size. We now present a novel algorithm that finds the best tree in the 1-SPR neighborhood in $O(n^2m)$ time.

High-level idea We focus on computing the posterior probability $P(T')$ where T' is obtained by a *single* SPR operation on the current tree T . This is illustrated in Fig. 2. By Equation 2, we need to find the maximum over all the Q values in T'. The key observation is that we don't need to calculate all Q values of T' from scratch. This is because T and T' are very similar topologically. Almost *all* the nodes in T are also in T'. At these shared nodes, Q values in T' either are the same as those in T, or can be easily computed from the Q values in T. Careful consideration of the relations between the Q values in T and T' values leads an efficient algorithm for finding the largest Q value in T', if some preprocessing on T is performed. We now give the details of this algorithm.

We consider the SPR operation in Fig. 2 that prunes a subtree T_u (rooted at the node u) of T and regrafts so that the parent of u is a new node w in T', which breaks the edge (w_1, w_2) into two edges. We let v_2 be the sibling of u and v_1 be the parent of v in T. We let the node r be the lowest common ancestor (LCA) of v and w_2 in T. Recall that the posterior probability is the product of the maximum $Q_s(a)$ (over each node a of T) times a constant factor for each site s (Equation 2). To simplify the notation, we often omit the subscript s by writing $Q_s(a)$ as $Q(a)$ throughout

this paper with the understanding that $Q(a)$ is for a specific site. We denote $Q'(x)$ be the updated $Q(x)$ value for the node x on the new tree T' after the SPR move shown in Figure 2.

We now show that we can find the maximum of $Q'(x)$ values efficiently *without* calculating $Q'(x)$ values explicitly. Note that T' has a new node w that is not in T, while v is in T but not in T'. All the other nodes are the same in T and T'. Recall that $Q(a)$ is equal to the product of the ratios between the probability of the taxon $t(b)$ being the mutant (allele 1) and that of being the wild-type (allele 0) for each leaf b within the subtree T_a (Equation 1). We have the following observations about $Q'(x)$ for a node $x \in T'$.

1. x is a node in T that either (i) is *not* on the paths $r \to v$ and $r \to w_1$, or (ii) $x = r$. Then, $Q'(x) = Q(x)$. This is because the SPR move doesn't change the set of taxa under node x and thus $Q'(x)$ is equal to product of the same set of ratios as in $Q(x)$.
2. $x \neq r$ and is on the path $r \to v_1$, $Q'(x) = \frac{Q(x)}{Q(u)}$. This is again because $Q(x)$ is the product of ratios of probabilities of genotypes within T_x being 1 vs. being 0; after the SPR move, the leaves under u are removed from T'_x and thus are absent from the product for $Q'(x)$.
3. $x \neq r$ and is along the path $r \to w_1$, $Q'(x) = Q(x)Q(u)$. This is because now leaves under u are inserted to T'_x.
4. $x = w$ (i.e., x is the only new node in T' added by SPR). Then $Q'(w) = Q(w_2)Q(u)$.

We now perform some preprocessing steps on T that enable the evaluation of *multiple* nodes together that fall into the same case as listed above. The preprocessing is conducted only once before the local search starts.

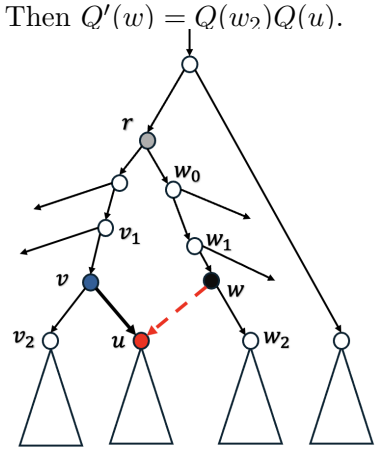


Figure 2: An illustration of (rooted) SPR local search. The subtree rooted at u (whose parent is v) is pruned and re-grafted to the edge right above w. Note: the lowest common ancestor of v and w_2 in the tree before SPR is r.

3.2.1 Preprocessing of the original T

We have the following definitions over T. For each node a of T, $M_1(a) \stackrel{\text{def}}{=} \max_{x \in nodes(T_a)} Q(x)$. That is, $M_1(a)$ is the maximum $Q(x)$ for each node x within the subtree T_a (it is allowed $x = a$). For a leaf a_0, $M_1(a_0) = Q(a_0)$. For an internal node a with two children nodes a_1 and a_2,

$$M_1(a) = max(Q(a), M_1(a_1), M_1(a_2))$$

Note that $M_1(a)$ values are for the nodes a in the original T *before* the SPR move. Recall that the $Q(a)$ values are already computed. Therefore, all $M_1(a)$ values can be computed in $O(n)$ time by performing a bottom-up traversal of T.

We denote the parent of the node b in T as $p(b)$. For each node a and a node $b \neq a$ where $b \in nodes(T_a)$ (i.e., b is within the subtree T_a), define $M_2(a, b) \stackrel{\text{def}}{=} \max_{x \in path(a,b)} Q(x)$. Here, $path(a, b)$ is the set of nodes along the path from a (inclusive) to b (inclusive). That is, $M_2(a, b)$ is the maximum $Q(x)$ at the node x that is on the path from a to b. $M_2(a, b)$ values can be computed in $O(n)$ time using the recurrence: $M_2(a, a) = Q(a)$; for $b \neq a$:

$$M_2(a, b) = \max(M_2(a, p(b)), Q(b))$$

Moreover, for each node a and a node $b \neq a$ where $b \in nodes(T_a)$, we define:

$$M_3(a, b) \stackrel{\text{def}}{=} \max_{x \in nodes(T_a), x \notin nodes(T_b), x \notin path(a,b)} Q(x)$$

That is, $M_3(a, b)$ is the maximum $Q(x)$ at the node x that is within T_a but is not within T_b nor along the path from a to b. As an example, in Fig. 2, when calculating $M_3(r, v)$, the node u is excluded (because $u \in T_v$); similarly, the right child node of the topmost node in Fig. 2 is also not included because it is not in T_r; on the other hand, w_1 is included because $w_1 \in T_r$, $w_1 \notin T_v$

and $x \notin path(r, v)$. All $M_3(a, b)$ values can be computed in $O(n)$ time. To see this, we first iterate over each node b; then walk the way up to the root from b; for each node a that is ancestral to b, let a_1 be the child of a that is on the path from $a \to b$, and let a_2 be the sibling of a_1 but *not* on the path $a \to b$. Then $M_3(a, a) = -\infty$. If $a \neq b$,

$$M_3(a, b) = \max(M_1(a_2), M_3(a_1, b))$$

3.2.2 Constant-time calculation of posterior probability for a single SPR move

We are now ready to show how to calculate the posterior probability for T' obtained by a single SPR move in $O(1)$ time *per SNV site*. We define $child(a, b)$ for a node a and a node b that is a descendant of a where $child(a, b)$ is the immediate child of a that is also ancestral to b. For example, in Fig. 2, $child(v_1, v_2) = v$. Now consider a single SPR move that prunes a subtree rooted at u (u's parent is v, v's parent is v_1 and u's sibling is v_2) to form a new node w by breaking an edge (w_1, w_2) as in Figure 2. With these definitions, we have:

$$M_1(root(T')) = \max_{x \in nodes(T')} Q(x) = \max(M_1(v_2), M_1(u), M_1(w_2), M_3(child(r, v), v), M_3(child(r, w_1), w_1),$$
$$M_2(root(T), r), M_3(root(T), r), \frac{M_2(r, v_1)}{Q(u)}, M_2(r, w_2)Q(u)) \quad (3)$$

The posterior probability $P(T')$ can be computed from $M_1(root(T'))$ by Equation 2. Here, $root(T')$ is the root node of T'. Due to the space limit, we give the detailed algorithm (Algorithm 2) for the SPR local search in the Supplemental Materials (Sect. A.1).

Correctness Due to the space limit, the correctness of Algorithm 2 is given in the Supplemental Materials (Sect. A.2).

3.2.3 Time analysis

The maximum Q' values for each SPR operation and each site s in $O(1)$ time (Equation 3), when the M_1, M_2 and M_3 values are pre-computed. There are $O(n^2)$ SPR operations and m sites. Pre-computing M_0, M_2 and M_3 values takes $O(n^2)$ time for each site. Therefore, finding the maximum posterior probability for a single SPR takes $O(n^2m)$ time in total.

3.3 Branch and bound speedup of SPR local search

Exhaustive search over all the $O(n^2)$ trees in the 1-SPR neighborhood can be slow when n is large. We now show that a *branch and bound* approach can in practice significantly speedup the SPR local search. Here is the high level idea. We use Fig. 2 for illustration again. We need to find the best branch (w_1, w_2) to regraft for a subtree that is rooted at u. We let the LCA node of u and w_2 be r. Let w_0 be the child of r that is ancestral to w_2. That is, (w_1, w_2) is within T_{w_0}. Naïvely, examining all branches under T_{w_0} takes $O(n)$ time because there are $O(n)$ edges in T_{w_0}. The key idea is that an *upper bound* of the maximum SPR Q' value among all SPR moves that prune T_u to somewhere *within* T_{w_0} can lead to significant speed up: we can discard the *entire* subtree T_{w_0} if this upper bound is no larger than the current maximal SPR Q' value. Due to the space limit, we provide the details of the branch and bound in the Supplemental Materials (Sect. A.3).

3.4 Genotype calling

Once the optimal tree T_{opt} is inferred, genotype calling becomes straightforward. We consider each SNV site s, and find the branch b of T_{opt} to place the single mutation for s that maximizes the $Q_v(s)$ value in Equation 2: the cells that are descendants of b are the mutants, and the rest are the wild-types. b can be found through the standard trace-back technique in dynamic programming.

Additional methods *ScisTree2* improves the construction of initial trees. Also, the running time of the NNI local search can be reduced to $O(Knm)$ from the naïve $O(Kn^2m)$ time using similar techniques as the SPR search. Due to the space limit, these additional methods are provided in the Supplemental Materials (Sects. A.4 and A.5).

4 Results

ScisTree2 is downloadable from `https://github.com/yufengwudcs/ScisTree2`.

4.1 Simulated data

We compare *ScisTree2* with the following methods on simulated data: (i) *CellPhy*, (ii) *Huntress*, (iii) *Sifit* and (iv) the original *ScisTree*. Most of these methods are developed recently and can handle relatively large data. Because *ScisTree2* is developed to handle large data, we don't compare with approaches (e.g., *SCITE*) that cannot scale to large data. In addition to comparing running time, we use the following metrics (all between 0 and 1, with 1 being the best) for accuracy comparison.

1. Genotype accuracy: the percentage of correctly called genotypes.
2. Tree accuracy: the percentage of the shared clades between the inferred trees and the true trees (which is equal to one minus the normalized Robinson-Foulds distance).
3. Ancestor-descendant (AD) F score: for the percentage of the pairs of mutations (m_a, m_b) where m_a is ancestral to m_b in the true tree but *not* so in the inferred tree.
4. Different-lineage (DL) F score: for the percentage of pairs of mutations (m_a, m_b) where m_a is *not* ancestral to m_b in the true tree but m_a is ancestral to m_b in the inferred tree.

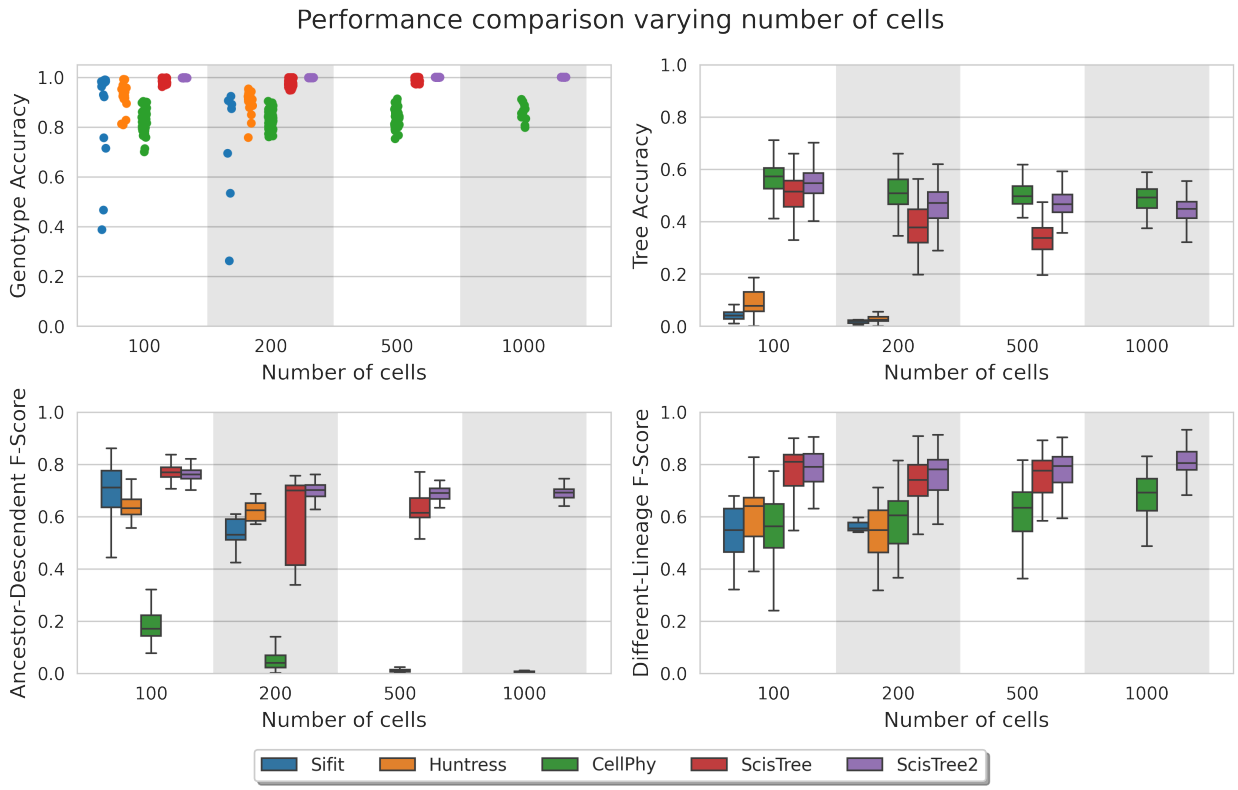


Figure 3: Comparison of inference accuracy of *ScisTree2* and four other methods: *CellPhy*, *Huntress*, *Sifit* and the original *ScisTree*. Vary the number of cells. Four metrics: genotype accuracy (scatter plot used for better visualization), tree accuracy, AD and DL F scores. X axis: number of cells (100, 200, 500 and 1,000); number of sites: five times of number of cells. Y axis: performance metric (1.0: the best).

We use *CellCoal* [14] to simulate single cell DNA sequence data. *CellCoal* is run under the diploid infinite sites model. The most important simulation parameters (along with their default values) are: (i) the number of cells n (200), (ii) the number of SNV sites m (five times of n), (iii) allelic dropout (ADO) rate (0.2), (iv) sequencing error rate (0.01), and (v) reads coverage (10x). Other parameters include amplification error rate and doublet rates. We run *CellPhy* -GL with the simulated VCF files from *CellCoal*. We run *Huntress* with the maximum likelihood genotypes extracted from the VCF files. *ScisTree2* and *ScisTree* requires posterior probability $Pr(G|D)$ of each genotype. Here, the single cell data D is in the form of sequence reads counts for different alleles. *CellCoal* only calculates the likelihood $Pr(D|G)$. We use Bayes Theorem to calculate $Pr(G|D)$ from $Pr(D|G)$. The details are given in the Supplemental Materials (Sect. A.6).

Accuracy Fig. 3 shows the comparison the inference accuracy of different methods by varying the number of cells. We show the four metrics: genotype accuracy, tree accuracy, AD and DL F scores. Overall, *ScisTree2* and *CellPhy* have the highest tree accuracy. *CellPhy* is sometimes slightly more accurate than *ScisTree2* in tree accuracy. However, *CellPhy* performs much worse in terms of AD/DL accuracy and genotype accuracy than *ScisTree2*. *Huntress* performs worse in tree accuracy than other methods, but is competitive in AD/DL and genotype accuracy. *ScisTree2* outperforms the original *ScisTree* in almost all experiments.

Running time *Sifit* and the original *Scistree* have no parallelization implemented. *Huntress*, *CellPhy*, and *ScisTree2* support multi-threading, and are run using 30 threads. *CellPhy* is run on its "FAST" mode that performs fast tree search from a single starting tree. *Sifit* runs with 10,000 iterations. Other methods are under their default settings. All methods are tested on a Linux server with Intel(R) Xeon(R) W-2195 CPU (36 cores). We test four settings with varying numbers of cells and sites. The results are shown in Fig. 4. Overall, *ScisTree2* is clearly the fastest, especially for data with large number of cells. *CellPhy* scales well with the number of sites, but appears to become very slow when the number of cells is large. *Huntress* scales well with the number of cells, but is very slow for data with large number of sites. *Sifit* and the original *ScisTree* are both significantly slower than the other methods.

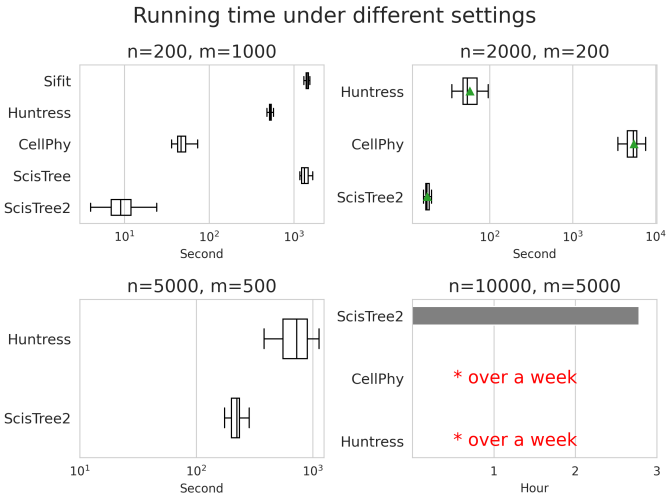


Figure 4: Comparison of the running time between *ScisTree2* and other methods on simulated data with the varying numbers of cells and sites, Time: elapsed time (in hours). 30 threads used for methods supporting multi-threading. Methods too slow: not reported.

Simulation for targeted sequencing Current single cell targeted DNA sequencing usually generates data with a large number of cells but relatively small number of SNVs. To evaluate the performance of *ScisTree2* on targeted single-cell data, we simulate reads using *CellCoal* with $1,000, 2,000$ and $5,000$ cells while keeping the number of SNVs to be 500. The results are shown in Fig. 5. *ScisTree2* clearly outperforms both *Huntress* and *CellPhy* for genotype accuracy and AD/DL F-scores for data with more cells than sites. For data with small number of sites, no method performs well in tree topology inference.

4.2 A high-grade serous ovarian cancer single cell DNA sequencing data

A low-coverage (with about 0.15x coverage) targeted sequencing dataset [7] with 891 cells from three clonally related high-grade serous ovarian cancer (HGSOC) cell lines from the same patient

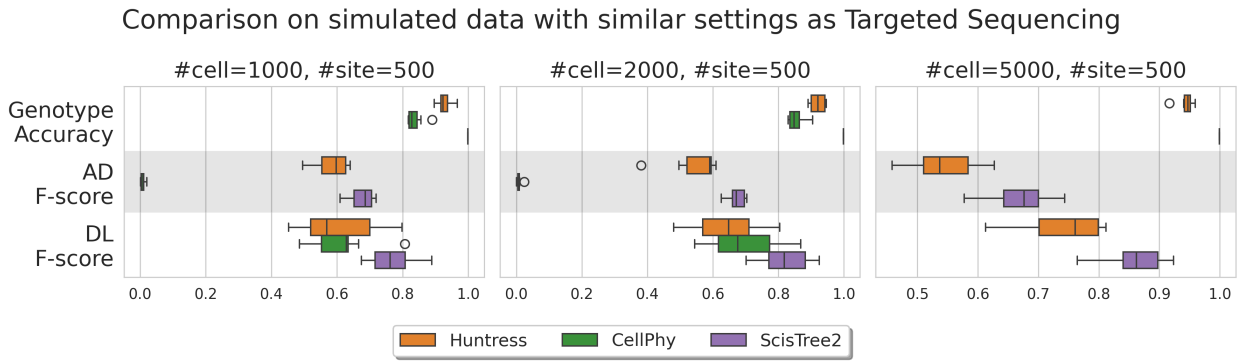


Figure 5: Comparison on simulated data with settings similar to targeted single cell sequencing data with more cells and fewer sites. Tree accuracy not shown: very low for all methods.

and processed by rigorous quality control were generated in [7]. The clonal tree of nine clones for the HGSOC data in [7] is shown in Fig. 6. The labels on the branches of the clonal tree are the genes where called mutations occur. Six genes, including three ancestral genes (*TP53*, *FOXP2* and *C7orf10*) and three clade genes (*HTR1D*, *INSL4* and *ZHX1*), are reported in [7]. Here, ancestral genes refer to the mutated genes shared by all tumor cells, and clade genes are those shared by a subset of clones. The true cell lineage tree for the HGSOC data is unknown. So we use the ordering of these six genes on the clonal tree (possibly the most confident ones reported in [7]) as the ground truth to test some aspects of inference accuracy. After dimensionality reduction and clustering, 891 cells are split into 9 clones based on copy number profiles, where the median clonal coverage is 15x. The resulting data contains the sequence read counts of 14,068 SNVs. We calculate the likelihood and posterior probabilities of the genotypes from the read counts (see the Supplemental Materials, Sect. B.1). We only run *ScisTree2* on the complete HGSOC data (with 14,068 SNVs), because *Huntress* is very slow on data with large number of sites and *CellPhy* is also very slow for this data which may be due to its large number of missing values.

Tree inference and mutated genes calling We run *ScisTree2* to infer the cell lineage tree and call genotypes for the complete HGSOC data. The called genotypes are for the 2,337 (16%) ancestral mutations that are shared by all 891 cells, 1,891 (14%) clonal mutations that are within a single clone, and 9,840 (70%) clade mutations that are located along branches of the clonal tree. Genes affected by ancestral mutations may have significant impact on early development of tumors, so we further use ANNOVAR [17] to annotate these ancestral mutations and find 48 exonic genes, which may potentially be the key driver genes in ovarian cancer oncological pathways.

Evaluation of the called mutated genes and the inferred cell lineage tree We use the inferred cell lineage tree to obtain the ordering of the highlighted six genes in [7] and then compare with the gene ordering from the reported clonal tree (Fig. 6 Left) The orderings of these six gene match *perfectly* in the two studies. As expected, *TP53*, a commonly recognized cancer driver gene, appears before all other mutated genes. In addition, we extract 49 ancestral genes including the six reported genes based on the provided clonal tree in [10] from the original study. All 48 genes found by *ScisTree2* are among them, with only one gene, *XXYLT1*, which is not identified as an ancestral gene in *ScisTree2*. We also investigate the topological concordance between the inferred cell lineage tree (Fig. 6 Right) and the clonal tree in [7]. Each of the three cell line forms a distinct clade in the cell lineage tree as expected. Within each cell line, the inferred cell lineage tree overall agrees with the clonal tree for the cell line OV2295 and OV2295(R2), but differs in many parts for cell line TOV2295(R). Note that the clonal tree in [7] was constructed from copy number variations only. Our results show that *ScisTree2* may potentially be useful to clonal analysis from SNV data.

Analysis of a reduced HGSOC data To compare with *Huntress* and *CellPhy*, we now construct a smaller data from the HGSOC data so that all three methods can run. We follow the same

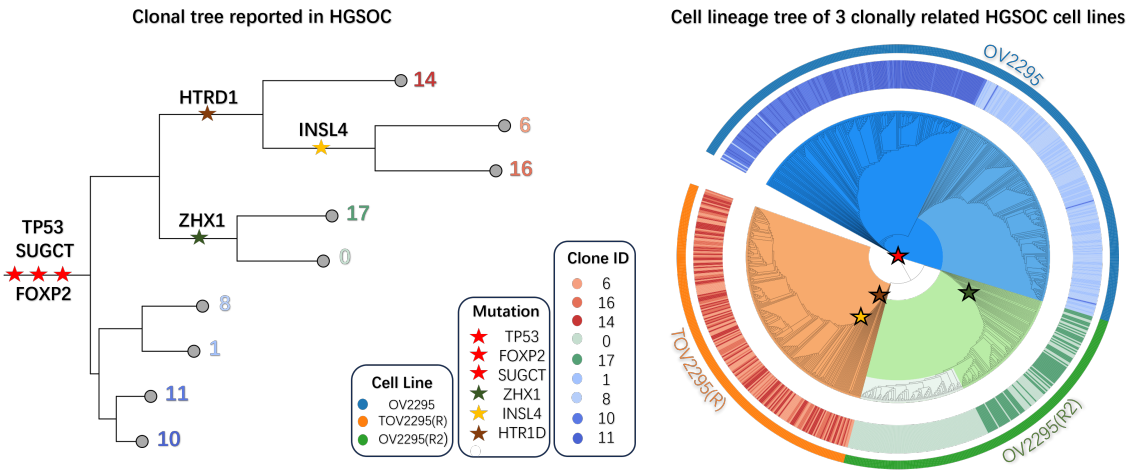


Figure 6: Analysis of the HGSOC data by *ScisTree2*. Left: the clonal tree of HGSOC with six mutated genes considered important for cancer development in the original study [7]. Number at a leaf: clone id (with a distinct color); Right: reconstructed cell lineage tree of tumor cells (colored w.r.t. their clones) with the same genes mapped on. Outer ring: 3 cell lines OV2295, TOV2295(R), and OV2295(R2). Inner ring: (colored) clones from the original study; Genes marked as red star are ancestral genes, others are clade genes.

filtering approach used in [6]. We filter out sites with more than 650 missing values, which leads to a reduced-size dataset with only 789 SNV sites. The original study in [7] mapped these SNVs to the branches of the clonal tree, which we use as the ground truth for comparing different methods. We collect the AD/DL F scores and the running time of *ScisTree2*, *Huntress* and *CellPhy* respectively. The results are shown in Fig. 7. *ScisTree2* and *Huntress* perform much better in inference accuracy w.r.t. AD and DL F scores compared with *CellPhy*. *ScisTree2* outperforms by 5% over *Huntress* in accuracy and is also faster on this smaller data.

Additional results in the Supplemental Materials: (i) results on simulated data by varying simulation parameters e.g., the dropout rate; genotype calling by *ScisTree2* for low coverage data (say 0.1x). (ii) robustness testing on semi-simulated data, and (iii) analysis of a dataset from metastatic colorectal cancer patient CRC2 [8] using *ScisTree2*.

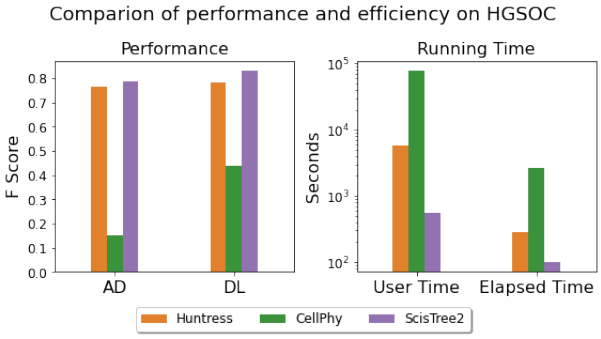


Figure 7: Compare *ScisTree2* with *Huntress* and *CellPhy* on a reduced HGSOC data. Left: AD and DL F scores of three methods for mutation ordering. Right: running time (in seconds) for user (CPU) time and elapsed (wall clock) time.

5 Conclusion

In this paper, we present *ScisTree2*, an improved cell lineage tree inference and genotype calling method. Through simulation and real data analysis, we show that *ScisTree2* can infer trees and call genotype accurately for large data. The main technical contribution of *ScisTree2* is showing that efficient local search of cell lineage tree on a simple probabilistic model runs faster than existing methods while achieving higher accuracy at the same time. Efficiency can be important for analyzing large data that is becoming available by the latest variant calling methods for single-cell DNA sequencing data.

ACKNOWLEDGMENTS

This work is partly supported by U.S. National Science Foundation grant IIS-1909425 (to YW).

References

[1] Sara Bizzotto, Yanmei Dou, Javier Ganz, Ryan N Doan, Minseok Kwon, Craig L Bohrson, Sonia N Kim, Taejeong Bae, Alexej Abyzov, Peter J Park, and Christopher A Walsh. Landmarks of human embryonic development inscribed in somatic mutations. *Science*, Mar 2021.

[2] Tim H. H. Coorens, Luiza Moore, Philip S. Robinson, Rashesh Sanghvi, Joseph Christopher, James Hewinson, Moritz J. Przybilla, Andrew R. J. Lawson, Michael Spencer Chapman, Alex Cagan, Thomas R. W. Oliver, Matthew D. C. Neville, Yvette Hooks, Ayesha Noorani, Thomas J. Mitchell, Rebecca C. Fitzgerald, Peter J. Campbell, Iñigo Martincorena, Raheleh Rahbari, and Michael R. Stratton. Extensive phylogenies of human development inferred from somatic mutations. *Nature*, 2021.

[3] Dan Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 1991.

[4] Katharina Jahn, Jack Kuipers, and Niko Beerenwinkel. Tree inference for single-cell data. *Genome Biology*, 2016.

[5] Alexey Kozlov, Joao M. Alves, Alexandros Stamatakis, and David Posada. Cellphy: accurate and fast probabilistic inference of single-cell phylogenies from scdna-seq data. *Genome Biology*, 2022.

[6] Can Kızılkale, Farid Rashidi Mehrabadi, Erfan Sadeqi Azer, Eva Pérez-Guijarro, Kerrie L. Marie, Maxwell P. Lee, Chi-Ping Day, Glenn Merlino, Funda Ergün, Aydın Buluç, S. Cenk Sahinalp, and Salem Malikić. Fast intratumor heterogeneity inference from single-cell sequencing data. *Nature Computational Science*, 2022.

[7] E. Laks, A. McPherson, H. Zahn, et al. Clonal decomposition and DNA replication states defined by scaled single-cell genome sequencing. *Cell*, 2019.

[8] Marco. L. Leung, Alexander Davis, Ruli Gao, Anna Casasent, Yong Wang, Emi Sei, Eduardo Vilar, Dipen Maru, Scott Kopetz, and Nicholas E. Navin. Single-cell DNA sequencing reveals a late-dissemination model in metastatic colorectal cancer. *Genome research*, 2017.

[9] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M.A. DePristo. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome Research*, 2010.

[10] A. W. McPherson, Clonal decomposition and DNA replication states defined by scaled single cell genome sequencing (Version v2) [Data set]. *Zenodo.* https://doi.org/10.5281/zenodo.5725635, 2019

[11] Bettina P. Mihalas, et al., Age-dependent loss of cohesion protection in human oocytes *Current Biology, Volume 34, Issue 1*, Jan 2024.

[12] Francesc Muyas, Carolin M. Sauer, Jose Espejo Valle-Inclán, Ruoyan Li, Raheleh Rahbari, Thomas J. Mitchell, Sahand Hormoz, and Isidro Cortés-Ciriano. De novo detection of somatic mutations in high-throughput single-cell profiling data sets. *Nature Biotechnology*, 2023.

[13] Nicholas E. Navin. Cancer genomics: one cell at a time. *Genome Biology*, 2014.

[14] David Posada. CellCoal: Coalescent Simulation of Single-Cell Sequencing Samples. *Molecular Biology and Evolution*, 2020.

[15] Kamen P Simeonov, China N Byrns, Megan L Clark, Robert J Norgard, Beth Martin, Ben Z Stanger, Jay Shendure, Aaron McKenna, and Christopher J Lengner. Single-cell lineage tracing of metastatic cancer reveals selection of hybrid emt states. *Cancer Cell*, Aug 2021.

[16] Alexandros Stamatakis. Raxml version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 2014.

[17] Kai Wang, Mingyao Li, and Hakon Hakonarson. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 2010.

[18] Yufeng Wu. Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. *Bioinformatics*, 2020.

[19] Hamim Zafar, Anthony Tzen, Nicholas Navin, Ken Chen, and Luay Nakhleh. Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome Biology*, 18(1):178, 2017.