# A Multi-labeled Tree Edit Distance for Comparing "Clonal Trees" of Tumor Progression

**Nikolai Karpov**
Department of Computer Science, Indiana University, Bloomington, IN, USA
nkarpov@iu.edu

**Salem Malikic**
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
smalikic@sfu.ca

**Md. Khaledur Rahman**
Department of Computer Science, Indiana University, Bloomington, IN, USA
morahma@iu.edu

**S. Cenk Sahinalp**
Department of Computer Science, Indiana University, Bloomington, IN, USA
cenksahi@iu.edu

## Abstract

We introduce a new edit distance measure between a pair of "clonal trees", each representing the progression and mutational heterogeneity of a tumor sample, constructed by the use of single cell or bulk high throughput sequencing data. In a clonal tree, each vertex represents a specific tumor clone, and is labeled with one or more mutations in a way that each mutation is assigned to the oldest clone that harbors it. Given two clonal trees, our multi-labeled tree edit distance (MLTED) measure is defined as the minimum number of mutation/label deletions, (empty) leaf deletions, and vertex (clonal) expansions, applied in any order, to convert each of the two trees to the maximal common tree. We show that the MLTED measure can be computed efficiently in polynomial time and it captures the similarity between trees of different clonal granularity well. We have implemented our algorithm to compute MLTED exactly and applied it to a variety of data sets successfully. The source code of our method can be found in: `https://github.com/khaled-rahman/leafDelTED`.

**2012 ACM Subject Classification** Applied computing → Computational genomics, Computing methodologies → Combinatorial algorithms

**Keywords and phrases** Intra-tumor heterogeneity, tumor evolution, multi-labeled tree, tree edit distance, dynamic programming

**Digital Object Identifier** 10.4230/LIPIcs.WABI.2018.22

**Funding** NK is supported by NSF CCF-1525024 and IIS-1633215. SM is supported by a Vanier Canada Graduate Scholarship.

## 1 Introduction

According to the *clonal theory of cancer evolution* [26], cancer originates from a single cell which had acquired a set of mutations that provide it proliferative advantage compared to the neighboring healthy cells. As tumor grows, cancer cells acquire new mutations and some of them might accumulate set of mutations conferring further selective advantage or disadvantage compared to the other cells. This continues over a period of time and at the time of the clinical diagnosis, tumors are usually heterogeneous consisting of multiple cellular

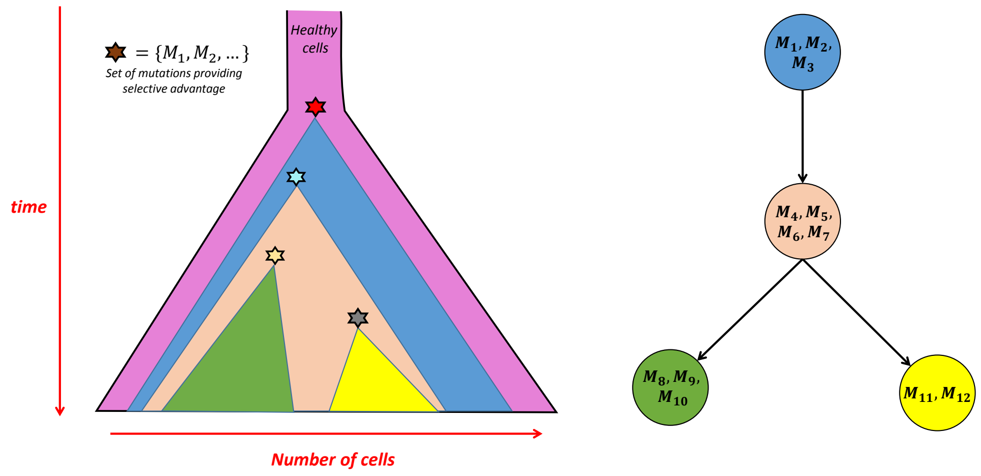


**Figure 1** Graphical overview of tumor initiation and growth (left) and the corresponding clonal tree of tumor evolution (right). Sets of mutations providing proliferative advantage and driving the emergence of new clones are denoted as stars in the left and as sets of corresponding mutations in the right panel (e.g. red star from the left panel represents the set of mutations $\{M_1, M_2, M_3\}$.) Node corresponding to the healthy cells is omitted as it would be non-informative.

populations, harboring distinct sets of mutations, leading to different phenotypes. Each such cellular population is considered to be a clone. The whole process of tumor initiation and growth is illustrated in Figure 1 (left panel).

One of the most widely used ways of depicting mutational heterogeneity and tumor progression over time is by the use of a *clonal tree of tumor evolution*. Here, each individual vertex represents a distinct clone and each mutation (i.e. its label) is placed as part of the label of clone where it occurs for the first time in evolutionary history. In this work we focus on trees built by the use of single nucleotide variants (SNVs), which represent the most widely used type of mutations in reconstructing trees of tumor evolution [12]. We also assume that each SNV occurs exactly once during the course of tumor evolution and never gets lost (infinite sites assumption, usually abbreviated as ISA). Some recently introduced methods (e.g. SiFit [10]) allow for the violations of ISA and, in such cases, we expect that labels corresponding to mutations vioalating ISA are removed from the trees prior to distance calculation. In order to simplify our figures, in each figure in this work we omit the node representing population of healthy cells. Namely, such node would be non-informative as it would always be label-free (since healthy cells are assumed to contain none of the mutations relevant to cancer progression) and attached as the parent of root node in each of the figures presented in this work. See Figure 1 for an illustration of tumor growth (left panel) and the corresponding clonal tree of tumor evolution (right panel). Note that the children of a vertex in a clonal tree are unordered.

A popular alternative to the clonal tree is the *mutation tree*, a special case of the clonal tree, where along each edge there is exactly one mutation [21, 14] - thus a mutation tree is a clonal tree with the highest possible granularity. As can be expected, any clonal tree can be easily converted to the mutation tree as follows. Consider an arbitrary edge $(u, v)$ and assume WLOG that a set of all mutations assigned to it is $\{M_1, M_2, \dots, M_k\}$. Now replace each edge $(u, v)$ by a path with vertices $\{w_0 = u, w_1, w_2, \dots, w_{k-1}, w_k = v\}$ and edges $\{(w_0, w_1), (w_1, w_2), \ldots, (w_{k-1}, w_k)\}$, such that exactly one mutation, WLOG $M_i$, is assigned to the edge $(w_{i-1}, w_i)$ for each $i \in \{1, 2, \ldots, k\}$. Note that from a given clonal tree which is not mutation tree (i.e. contains at least one node with two or more labels), multiple different mutation trees can be obtained. There are additional tree representations [14] for tumor evolution but in this work we focus on clonal trees only.

## Similarity/distance measures between tree representations of tumor evolution

In the past few years, we have witnessed rapid developments in computational methods for inferring trees of tumor evolution from both bulk and single cell high throughput sequencing (HTS) data [21, 14, 9, 18, 11, 6, 15, 27, 16, 25, 8, 5].

In order to assess the accuracy of the proposed method, many of these studies use simulated HTS data extracted from synthetic tumor compositions. The inferred tree is then compared against the (synthetic) ground truth. (We will call the *ground truth tree* the *true tree.*) Other studies, such as the Pan Cancer Analysis of Whole Genomes Project (PCAWG) compare trees inferred by participating methods on real tumor samples to reach a consensus tree. In order to compare clonal trees with varying granularity (granularity can be measured in terms of the average number of mutations assigned to a clone) the measure(s) used should be versatile enough to discriminate real topological differences between trees from those differences due to the type and coverage of the HTS data used by a method; e.g. such a distance measure should be equal to 0 between any clonal tree and its corresponding mutation tree (obtained using the procedure described above).

Unfortunately, comparing trees of tumor evolution has turned out to be a very challenging problem and available measures fail to capture the differences/similarities between inferred or true trees. In fact many of existing measures only aim to compare the relative placement of pairs of mutations across two trees, e.g. whether the two mutations maintain an ancestor-descendant relationship in both trees [27, 16]. Such measures can not capture topological differences between distinct trees, e.g. a simple topology with two vertices, where all but one of the mutations is assigned to the non-root vertex, v.s. a star topology where each vertex is assigned a single mutation. As a result it is of high importance to have measures of tree similarity that not only consider the relative placement of mutations but also the topological structure of the trees.

The standard measure to compare combinatorial objects - such as strings, especially in bioinformatics, is the edit distance. This measure has numerous applications and a large number of variants, not only for strings but also for labeled trees, have been considered in the past. The classical Levenstein edit distance between two strings is defined as the minimum number of single symbol insertions, deletions and replacements to convert one of the strings to the other and has a well established dynamic programming algorithm to compute it (e.g. [35]). The running time of this algorithm is proportional to the product of the lengths of the two input strings and the existence of a sub-quadratic algorithm is unlikely [1]. In general, the complexity of computing an edit distance strictly depends on the set of allowed edit operations. E.g. if we consider a variant of the problem where only single character mismatches and block reversals are allowed, then the running time reduces to $O(n \log^2 n)$ [29] - here $n$ is the total length of the strings; on the other hand, the variant where only mismatches, block deletion and move operations are allowed is $NP$-hard [32].

Edit distance measures for rooted trees have typically been defined for those with ordered vertices, each with a single label; here the goal is to transform one tree to the other by the use of vertex insertions, vertex deletions and vertex label replacements [31]. Based on the tree edit distance, a notion of tree alignment has also been introduced, both for vertex ordered as well as unordered trees [13]. For many of the vertex ordered cases, there are polynomial time algorithms that can solve the distance/alignment problem [33, 31, 22, 4, 38, 37, 30, 13, 20, 3], whereas for several unordered cases, the problems are NP-hard [24, 34] or MAX SNP-hard [39, 13].

Unfortunately for many of variants of the problem, the time complexity is still unknown [2]. As importantly none of the existing algorithms deal with trees where vertices may have more than a single (mutational) label - which is at the heart of clonal tree comparison problem.

In this paper we introduce for the first time a tree edit distance measure to compare trees where vertices may have one or more (mutational) labels. The multi-labeled tree edit distance (MLTED) measure we introduce in this paper successfully captures the differences between clonal trees: for example it satisfies a key condition that two clonal trees from which it is possible to produce two identical mutation trees have a distance 0. Even though MLTED is defined for (the more general) vertex unordered trees, we show that there is an algorithm to exactly compute it in polynomial time. We have implemented this algorithm and applied it to a number of synthetic and real data sets to compare trees inferred by some of the available tumor history reconstruction methods with success.

## 2 Definitions

A rooted tree $T = (V, E)$ is a connected, acyclic, undirected graph with set of vertices $V$ (also denoted as $V(T)$) and edges $E$ (also denoted as $E(T)$), with a particular vertex, $r$ identified as the root. For each non-root vertex $v$, any vertex $u$ that lies on the simple path between $v$ and the root is considered to be its ancestor; in particular, the node $u = p(v)$ on this path which has an edge to $v$ is considered to be its parent. The depth of vertex $v$ denoted $d(v)$, is thus defined as the number of its ancestors. The lowest common ancestor of any pair of vertices $u$ and $v$, denoted $\text{lca}(u, v)$, is defined as a common ancestor of both $u$ and $v$ whose depth is maximum possible. The structure of a tree induces partial order $\preceq$ on its vertices: $u \preceq v$ denotes that $u$ is an ancestor of $v$.

In this work, we consider multi-labeled trees in which each vertex $v$ has a subset $L_v$ of labels from a universe $\mathbb{L}$. Additionally, each label is unique to a vertex, i.e. $L_u \cap L_v = \emptyset$ for each pair of distinct vertices $u$ and $v$. We denote the set of all labels assigned to the vertices of $T$ as $L(T)$. In other words, $L(T) = \bigcup\limits_{v \in V(T)} L_v$.

Consider the following types of edit operations on multi-labeled tree:

- *deleting a label* where one of the labels is removed from some set $L_v$,
- *deleting a vertex* where a vertex is removed from the tree. This operation is allowed to be performed only for unlabeled leaves, i.e. vertices with no labels and no children,
- *expanding a vertex* where vertex $v$ is replaced by two vertices $v_1$ and $v_2$ such that all children of $v$ after this operation are children of $v_2$, and the parent of $v$ is the parent of $v_1$, and $v_1$ is the parent of $v_2$. Each of the labels from $L_v$ is assigned to exactly one of the $L_{v_1}$ and $L_{v_2}$.

We define the edit distance between two multi-labeled trees as the minimal number of label deletions, such that together with an arbitrary number of vertex deletions and vertex expansions (which do not contribute to the distance), they transform both trees to a common third tree (which will be defined as the *maximal common tree* of the two input trees). As can be observed, the distance between two trees is upper bounded by the total number of labels in the two trees.[1]

[1] Note that typically edit distance measures are based on symmetric edit operations, in a way that each

## 3 Set Alignment Problem

Our algorithm for computing the distance between trees is based on finding a *maximal common tree*, as will be defined below. We first show connection between this notion and our edit distance measure and then demonstrate how this notion works for paths, which form the base case. We later extend this definition to the general case and provide an efficient algorithm for computing the maximal common tree.

A *Common tree* of arbitrary multi-labeled trees $T_1$ and $T_2$ is any multi-labeled tree which can be obtained from each of $T_1$ and $T_2$ by the use of edit operations defined above. A *maximal common tree* of $T_1$ and $T_2$ is a common tree of $T_1$ and $T_2$ having the largest number of labels among all common trees of $T_1$ and $T_2$. Our MLTED measure between $T_1$ and $T_2$ is, by definition, equal to the difference between the total number of labels in $T_1$ and $T_2$ and twice the number of labels in their maximal common tree. In other words, tree edit distance is defined as the total number of labels required to be removed from the two trees in order to transform them to their maximal common tree.

If the two trees are simple paths, then any common tree between them is also a path. Let the ordered sequence of vertices of the first tree/path be $v_1, v_2, \ldots, v_n$ with respective label sets $S_1, S_2, \ldots, S_n$, and the ordered sequence of vertices of the second tree/path be $w_1, w_2, \ldots, w_m$ with respective label sets $P_1, P_2, \ldots, P_m$. (Assume that $S_i, P_j$ are subsets of $\mathbb{L}$ and that any label $u \in \mathbb{L}$ occurs exactly in one of $S_1, S_2, \ldots, S_n$ and exactly in one of $P_1, P_2, \ldots, P_m$.) Let $f : \mathbb{L} \to \{1, 2, \ldots, n\}$ and $g : \mathbb{L} \to \{1, 2, \ldots, m\}$ be the functions that map labels to vertex indices, respectively in the first and the second tree such that $v_{f(a)}$ denotes the vertex of label $a$ in the first tree and $w_{g(a)}$ denotes the vertex of the label $a$ in the second tree.

It is easy to see that computing a maximal common tree in this special case is equivalent to the following generalized version of the string edit distance problem for a pair of ordered sets.

Set Alignment Problem
**Instance:** Two ordered set of labels: $(S_1, S_2, \ldots, S_i)$ and $(P_1, P_2, \ldots, P_j)$ where $1 \leq i \leq n$ and $1 \leq j \leq m$.
**Task:** Find set $A(i,j) \subseteq (\bigcup_{p=1}^{i} S_p) \bigcap (\bigcup_{q=1}^{j} P_q)$ of maximal size such that, for each pair $(a, b)$ of labels from $A(i,j)$, the following holds: $f(a) \leq f(b) \iff g(a) \leq g(b)$.

The following lemma offers an efficient algorithm for solving the Set Alignment Problem. Our approach for computing edit distance between two arbitrary trees (presented in Section 4) uses this algorithm as a subroutine.

▶ **Lemma 1.** *Let* $\mathrm{D}(i,j)$ *be the size of the set which is answer of the* Set Alignment Problem *for the instance where input sequences are* $(S_1, \ldots, S_i)$ *and* $(P_1, \ldots, P_j)$ *(i.e. according to the notation from the above* $D(i,j) = |A(i,j)|$*). Then the following hold:*

- $\mathrm{D}(i,0) = \mathrm{D}(0,j) = 0$, *for all non-negative integers* $i$ *and* $j$.
- $\mathrm{D}(i,j) = \max(\mathrm{D}(i,j-1), \mathrm{D}(i-1,j)) + |S_i \cap P_j|$, *for all positive integers* $i$ *and* $j$.

operation is complemented by a reverse operation (e.g. deleting a label is the reverse of inserting the same label). In such cases, the edit distance is defined as the minimum number of operations required to transform one combinatorial object into another. Although it is possible to define our edit distance measure similarly (with label insertions complementing label deletions), we chose to present our distance by specifying deletions only for keeping the description compact.

**Proof.** The first equation easily follows from the fact that $A(i,0) \subseteq \emptyset$ and $A(0,j) \subseteq \emptyset$.

For the second equation, we first prove that $\mathrm{D}(i,j) \geq \max(\mathrm{D}(i,j-1),\mathrm{D}(i-1,j))+|S_i \cap P_j|$. In order to prove this, observe that each of $A(i,j-1) \cup (S_i \cap P_j)$ and $A(i-1,j) \cup (S_i \cap P_j)$ represent a valid candidate solution for the instance of Set Alignment Problem with the input sequences $(S_1,\ldots,S_i)$ and $(P_1,\ldots,P_j)$. Namely, in the case of set $A(i,j-1) \cup (S_i \cap P_j)$ (analogous applies to the set $A(i-1,j) \cup (S_i \cap P_j)$), if we consider two arbitrary labels $a$ and $b$ of this set, then:

- If $a \in A(i,j-1)$ and $b \in A(i,j-1)$ then $f(a) \leq f(b) \iff g(a) \leq g(b)$ holds by the definition of $A(i,j-1)$.
- If $a \in A(i,j-1)$ and $b \in S_i \cap P_j$ then $f(a) \leq i$ and $g(a) \leq j-1$. On the other hand, $f(b) = i$ and $g(b) = j$ hence $f(a) \leq f(b) \iff g(a) \leq g(b)$ is obviously satisfied.
- Case where $a \in S_i \cap P_j$ and $b \in A(i,j-1)$ is analogous to the previous case.
- Case where both $a$ and $b$ are from $S_i \cap P_j$ is trivial since in this case $f(a) = f(b) = i$ and $g(a) = g(b) = j$ implying that $f(a) \leq f(b) \iff g(a) \leq g(b)$ holds in this case as well.

Now it suffices to prove that $\mathrm{D}(i,j) \leq \max(\mathrm{D}(i,j-1),\mathrm{D}(i-1,j)) + |S_i \cap P_j|$. In order to prove this, consider the partition of $A(i,j)$ into $A(i,j) \setminus (S_i \cap P_j)$ and $S_i \cap P_j$. We claim that at most one of the sets $S_i$ and $P_j$ has non-empty intersection with the set $A(i,j) \setminus (S_i \cap P_j)$. To prove this, assume on contrary that there exists $a \in S_i \cap (A(i,j) \setminus (S_i \cap P_j))$ and $b \in P_j \cap (A(i,j) \setminus (S_i \cap P_j))$. Since $a \in S_i$ we have $f(a) = i$. For $b$ we have that $b \in A(i,j)$ and $b \notin S_i$ implying that $f(b) \leq i-1$. Similarly, $g(a) \leq j-1$ and $g(b) = j$. By the above assumption, both $a$ and $b$ belong to $A(i,j)$ but obviously they violate constraint $f(a) \leq f(b) \iff g(a) \leq g(b)$ which is, by definition of $A(i,j)$ satisfied for all of its labels. This contradiction directly implies our latest claim. To finalize the proof of inequality $\mathrm{D}(i,j) \leq \max(\mathrm{D}(i,j-1),\mathrm{D}(i-1,j)) + |S_i \cap P_j|$ assume WLOG that the intersection of $S_i$ and $A(i,j) \setminus (S_i \cap P_j)$ is the empty set. This implies that $A(i,j)$ does not contain any label from $S_i \setminus (S_i \cap P_j)$. Therefore $D(i,j) \leq D(i-1,j)+|S_i \cap P_j| \leq \max(\mathrm{D}(i,j-1),\mathrm{D}(i-1,j))+|S_i \cap P_j|$ which completes our proof. ◀

Lemma 1 provides a dynamic programming formulation for calculating distance $D(n,m)$ between trees $T_1$ and $T_2$.

▶ **Observation 2.** *Total time and total space required for calculating number of labels in each of the sets $S_i \cap P_j$, where $i \in [n]$ and $j \in [m]$ are both $O(\sum_{i=1}^{n} |S_i| + \sum_{j=1}^{m} |P_j| + nm)$.*

**Proof.** For each label from $u \in L$ we can store two indices $f(u)$ and $g(u)$. This can be implemented in the above time and space by using a hash table. If we know these indices, we can fill the table $I_{ij}$, where $I_{ij} = |S_i \cap P_j|$, by iterating through elements of $\mathbb{L}$ and increasing the value of $I_{f(x)g(x)}$ by one for each $x \in \mathbb{L}$. ◀

▶ **Lemma 3.** *The* Set Alignment Problem *is solvable in $O(\sum_{i=1}^{n} |S_i| + \sum_{j=1}^{m} |P_j| + nm)$ time and space.*

**Proof.** Follows straightforwardly from Lemma 1 and Observation 2. ◀

## 4 Computing a maximal common tree in the general case

We now describe an efficient algorithm for computing a maximal common tree. Note that in the remainder of the paper we call all vertices in a tree with exactly one child as *non-crucial* vertices and all other vertices, i.e. leaves, and vertices with two or more children, as *crucial* vertices. Now consider the sequence of edit operations applied to a tree $T_1$ in the process to reaching a common tree $T$ with another tree $T_2$.

▶ **Observation 4.** *Each edit operation applied to any vertex deletes at most one crucial vertex and creates at most one (new) crucial vertex; no edit operation can increase the number of crucial vertices.*

**Proof.** The proof follows through a simple case analysis on the edit operations we allow.

- The edit operation of deleting a label does not change the topology of the tree or the set of crucial vertices in the tree.
- The edit operation of deleting a leaf $u$ does change the topology of a tree, but with respect to the set of crucial vertices, the only update is that $u$ is lost, and, (i) provided that $u$ was the only child of $p(u)$, $p(u)$ becomes crucial, or (ii) provided that $u$ was one of the two children of $p(u)$, $p(u)$ becomes non-crucial, or (iii) provided that $u$ was one of more than two children of $p(u)$, $p(u)$ stays crucial. All other vertices remain unaltered.
- Finally, the edit operation of expanding, i.e., splitting a vertex $v$ into $v_1$ and $v_2$ does change the topology of the tree but it does not create a new crucial vertex if $v$ is non-crucial; however, if a vertex $v$ is crucial, then $v_2$ becomes crucial after the edit operation, but $v_1$ stays non-crucial. ◀

The observation above indicates that an edit operation applied to a crucial vertex $u$ may create a new crucial vertex $v$. In that case, we say that the crucial vertex $u$ in $T_1$ *corresponds to* a crucial vertex $v$ in $T_1'$ (if latter was created). In case of an expansion of vertex $u$ in $T_1$ to two vertices $u_1$ and $u_2$, we say that $u$ corresponds to $u_2$ in $T_1'$. In case of a deletion of a leaf $u$, if $p(u)$ which was originally non-crucial, became crucial, then we say that $u$ in $T_1$ corresponds to $p(u)$ in $T_1'$. For any vertex $v$ which remains unedited and crucial in $T_1'$, we say that $v$ in tree $T_1$ corresponds to $v$ in the tree $T_1'$.

Finally, we say that $v$ in $T_1$ corresponds to $v'$ in $T$ if for the sequence of trees $T_1 = T_1^0, T_1^1, \ldots, T_1^l = T$ (where $T_1^{i+1}$ is obtained from $T_1^i$ by an edit operation) there exists the sequence of vertices $v = v^0, v^1, \ldots, v_l = v$ (where $v^l \in V(T_1^l)$) such that $v^i$ corresponds to $v_{i+1}$ for all $i$. We extend the notion of correspondence to $T_2$ in a similar manner.

Given trees $T_1$ and $T_2$, their common tree $T$ and the vertices in $T_1$ and $T_2$ that correspond to every crucial vertex in $T$, it is straightforward to establish the edit operations to transform $T_1$ and $T_2$ to $T$. The algorithm to compute $T$ makes use of this observation.

▶ **Observation 5.** *Given two sets of crucial vertices $u_1, \ldots, u_l$ and $v_1, \ldots, v_l$ in $T_1$ and $T_2$ respectively such that $u_i$ and $v_i$ correspond to same crucial vertex in the common tree $T$ for each $i$, we can reconstruct a common tree $T'$ such that the number of labels in $T'$ is at least that in $T$.*

**Proof.** Here we describe the procedure of reconstructing the tree $T'$ in two steps.

In the first step we delete each label which cannot belong to $T$ in a trivial manner: let $S_1$ ($S_2$) be the set of vertices which do not lie on a path from the root of $T_1$ ($T_2$) to some $u_i$ ($v_i$). Then we delete all vertices from $S_1$ (and $S_2$) together with their labels. Note that no label which is present in tree $T$ will be deleted: if a vertex $v$ does not belong to a path from the root to some crucial vertex in $T$, then any label from $L_v$ cannot be present in $T$. However, if any label in $T$ that is in $L_v$ for some vertex $v$ which lies on a path from the root to a leaf $w$ (which is necessarily crucial) then there must exist a pair of vertices $u_i, v_i$ which correspond to the leaf $w$.

Thus, starting from the leaf level, we can delete all vertices which do not belong to a path from the root to any $u_i$ (and $v_i$). It is easy to see that this first step transforms $T_1$ and $T_2$ into isomorphic trees - the isomorphism $\phi$ on $r_1, u_1, \ldots, u_l$ which transforms $T_1$ into $T_2$ is $\phi(r_1) = r_2, \phi(u_1) = v_1, \ldots, \phi(u_l) = v_l$ (here $r_i$ is the root of $T_i$).

Let $T'_1$ and $T'_2$ denote the trees respectively produced from $T_1$ and $T_2$ after applying the first step. Notice that, $T'_1$ and $T'_2$ are also topologically isomorphic to $T$ and $T'$.

In the second step, for each pair of vertices $v_i$ and $u_i$ we consider the pair of "maximal" paths from $v_i$ and $u_i$ to the associated root, which do not contain other vertices from $v_1, \ldots, v_l$ and $u_1, \ldots, u_l$. For this pair of paths we apply a sequence of edit operations that expand vertices and delete labels, such that the resulting paths will be identical with the maximum possible number of labels.

$T'$ is the tree produced as a result of the second step. Note that on any pair of paths from the vertex pair $u_i$ and $v_i$ to the respective root, the set of labels observed will be identical. This implies that $T'$ is a common tree with number of labels necessarily lower bounded by that of $T$. ◀

The above observation implies that we can reduce the problem of computing a maximal common tree between two multi-labeled trees to the problem of finding an *optimal pair of sequences* of vertices $u_1, \ldots, u_l$ and $v_1, \ldots, v_l$ corresponding to the maximal common tree.

Our general algorithm for computing the distance between two multi-labeled trees requires constant time access to the solutions to many instances of the Set Alignment Problem, which we compute in a preprocessing step.

Solving Set Alignment Problem for all pairs of sequences $u_1, \ldots, u_l$ and $v_1, \ldots, v_l$ is impractical. Fortunately, special conditions with respect to the structure of these sequences help us develop an efficient algorithm for finding an optimal pair of sequences as explained below.

The algorithm for computing an optimal pair of sequences will need the solutions to Set Alignment Problem for all possible downward paths; we call this auxiliary problem Pairwise Alignments on a Tree.

Given a pair of vertices $u, v$ such that $u \preceq v$, let the following sequence of sets of vertex labels be denoted as $\mathrm{P}(u, v) = (L_{w_1}, \ldots, L_{w_k})$ where $w_1(= u), w_2, \ldots, w_k(= v)$ is called the downward path between $u$ and $v$. Then we can define Pairwise Alignments on a Tree problem formally as follows.

Pairwise Alignments on a Tree
**Instance:** Two rooted unordered multi-labeled trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, T_2)$ with associated sets of labels for each vertex.
**Task:** For each 4-tuple $(a, b, c, d)$ such that $a, b \in V_1$, $c, d \in V_2$, $a \preceq b$ and $c \preceq d$, compute and store the answer for Set Alignment Problem on $\mathrm{P}(a, b), \mathrm{P}(c, d)$.

In the next lemma, we introduce equations for computing Pairwise Alignments on a Tree which forms the basis of our dynamic programming algorithm.

▶ **Lemma 6.** *Given $a, b \in V(T_1)$; $c, d \in V(T_2)$; $a \preceq b$; $c \preceq d$, let $\mathrm{D}(a, c, b, d)$ be the solution for the instance $\mathrm{P}(a, b)$, $\mathrm{P}(c, d)$ of* Set Alignment Problem. *Then*
1. *If $a = b$ and $c = d$ then $\mathrm{D}(a, c, b, d) = |L_b \cap L_d|$.*
2. *If $a = b$ and $c \neq d$ then $\mathrm{D}(a, c, b, d) = \mathrm{D}(a, c, b, p(d)) + |L_b \cap L_d|$.*
3. *If $a \neq b$ and $c = d$ then $\mathrm{D}(a, c, b, d) = \mathrm{D}(a, c, p(b), d) + |L_b \cap L_d|$.*
4. *Otherwise $\mathrm{D}(a, c, b, d) = \max(\mathrm{D}(a, c, p(b), d), \mathrm{D}(a, c, b, p(d))) + |L_b \cap L_d|$.*

**Proof.** Each of the cases above holds true as a direct consequence of Lemma 1. ◀

Through a straightforward application of the above lemma, we obtain the following.

▶ **Lemma 7.** *If $I_1$ and $I_2$ denote the heights of $T_1$ and $T_2$, respectively,* Pairwise Alignments on a Tree *is solvable in $O\left(|V_1||V_2|I_1 I_2 + |L(T_1)| + |L(T_2)|\right)$ time and space.*

**Proof.** The algorithm is a straightforward implementation of Observation 2 and Lemma 6. Namely, from Observation 2 it follows that the values of $|L_a \cap L_b|$, for all $a \in V_1$ and $b \in V_2$, can be computed by the use of algorithm having time and space complexity $O\left(|V_1||V_2| + |L(T_1)| + |L(T_2)|\right)$. After computing these values, all entries in D can be computed in the time and space that are proportional to the number of all possible combinations of $a, b, c, d$, which is bounded by $|V_1||V_2|I_1 I_2$. Now, combining the above with the obvious inequality $|V_1||V_2|I_1I_2 \geq |V_1||V_2|$, we have that the overall time and space complexity of the proposed algorithm is $O\left(|V_1||V_2|I_1I_2 + |L(T_1)| + |L(T_2)|\right)$. ◀

Let $M : V(T_1) \cup V(T_2) \to V(T_1) \cup V(T_2)$ be the (partial) bijective mapping between vertices $v$ and $w$ of $T_1$ and $T_2$, respectively, which correspond to the same crucial vertex $u$ in their common tree $T$; i.e. $M(v) = w$ and $M(w) = v$.

▶ **Observation 8.** *For any pair of vertices $a, b \in V_1$ (or $V_2$) the lowest common ancestor of $a$ and $b$, namely $\text{lca}(a, b)$, has a mapping, $M(\text{lca}(a, b))$ which is equal to $\text{lca}(M(a), M(b))$. For any triplet of vertices $a, b, c \in V_1$ (or $V_2$), the lowest common ancestor of $a, b$ is equal to the lowest common ancestor of $b, c$ if and only if $\text{lca}(M(a), M(b)) = \text{lca}(M(b), M(c))$.*

We now present our algorithm for computing the size of a maximal common tree, which is a combination of dynamic programming and an algorithm for finding a maximum cost matching.

▶ **Theorem 9.** *The mapping which corresponds to a maximal common tree can be computed in time $O(|V_1||V_2|(|V_1| + |V_2|)\log(|V_1| + |V_2|) + |V_1||V_2|I_1I_2 + |L(T_1)| + |L(T_2)|)$.*

**Proof.** For $i \in \{1, 2\}$ and $x \in V_i$, let $T_i(x)$ denote multi-labeled tree which is identical to subtree of $T_i$ rooted at $x$, except that the set of labels assigned to the root node in $T_i(x)$ is empty. We first define function $G : V_1 \times V_2 \to \mathbb{N}$, such that $G(a, b)$ denotes the size of the maximal common tree between trees $T_1(a)$ and $T_2(b)$, where we are assuming that $M(a) = b$. The maximal common tree is then indicated by $\max\limits_{(a,b)\in V_1\times V_2} [G(a, b) + \text{D}(r_1, r_2, a, b)]$. The key observation of our algorithm is that the computation of $G(a, b)$ can be reduced to finding a maximum cost matching for an auxiliary graph. Let $a_1, \ldots, a_n$ be the children of $a$, and $b_1, \ldots, b_m$ be the children of $b$. The structure conditions on mapping provide the guarantee that all vertices which are leaves of downward paths from $a$ without internal crucial vertices, lie in distinct subtrees. Using the Observation 8 this implies that each such vertex lies in distinct subtrees with roots $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$. For the simplicity of notation, for $i \in \{1, \ldots, n\}$, denote the subtree of $T_1$ with root at $a_i$ as $F_i$. Analogously, we denote subtree of $T_2$ rooted at $b_j$, where $j \in \{1, 2, \ldots, m\}$, by $H_j$. For a pair of subtrees, we define the score of an optimal choice of pairs of mapped vertices; $c(F_i, H_j) = \max\limits_{c\in V(F_i), d\in V(H_j)} (\text{D}(a, b, c, d) + G(c, d))$. Thus the cost of edges in this graph is equal to the size of a maximal common tree if we choose an optimal mapped pair of vertices in these subtrees. However, we should find an optimal choice such that a subtree corresponds to a subtree in another tree, under the condition that we cannot construct correspondence of two subtrees into single subtree; this problem is the well known maximum weighted bipartite matching problem, which can be solved in a polynomial time [19]. Finally, we can construct a bipartite graph on the set of vertices $a_1, \ldots, a_n, b_1, \ldots, b_m$ with the cost of an edge $(a_i, b_j)$ equal to $c(F_i, H_j)$. Thus our algorithm returns the score of an optimal assignment in this graph where the optimal value of the $G(a, b)$ can be reduced to the computation of $G(x, y)$ for all $x$ and $y$ such that $a \preceq x$ and $b \preceq y$ and finding the maximum weighted matching on auxiliary graph (which is complete bipartite graph with $n + m$ vertices and $nm$ edges). Thus we can summarize our algorithm as follows.

- If $a$ or $b$ is a leaf then $G(a,b)=0$.
- Otherwise let $Q$ be a complete weighted bipartite graph on vertices $a_1,\dots,a_n,b_1,\dots,b_m$ ($a_i$ is the root of $F_i$, and $b_j$ is the root of $H_j$) in which the weight of an edge $(a_i,b_j)$ is equal to $\max_{c\in V(F_i),d\in V(H_j)}(\mathrm{D}(a_i,b_j,c,d)+G(c,d))$;then $G(a,b)$ is equal to the cost of an optimal assignment in $Q$.

The time to construct auxiliary graphs is bounded by $O(|V_1||V_2|I_1I_2)$, the most time consuming part of this algorithm is the solution to the assignment problem. However, the time needed to solve this problem for a graph with $n$ vertices and $m$ edges is bounded by $O(nm\log n)$. If $n_a$ is the number of children of a vertex $a$ in the first tree, and $n_b$ is the number of children of a vertex $b$ in the second tree then the total time is bounded by $O(\sum_{a,b}(n_a+n_b)n_an_b\log(n_a+n_b))$ which can be bound (up to constant factor) by $O(|V_1||V_2|(|V_1|+|V_2|)\log(|V_1|+|V_2|))$ or $O((|V_1|\sum_j n_b^2+|V_2|\sum_i n_a^2)\log(|V_1|+|V_2|))$. The second bound is significantly better if the maximal degree of a vertex is bounded by a small value. ◀

The above theorem finalizes the description of our algorithm for computing the edit distance between two multi-labeled trees.

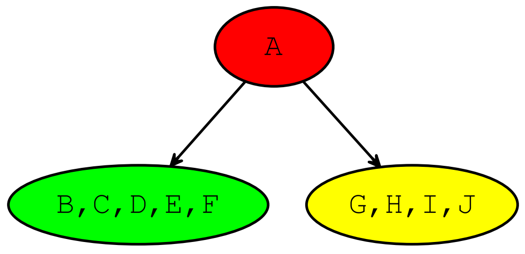
## 5 Discussion and an application

### 5.1 The existing measures and their limitations

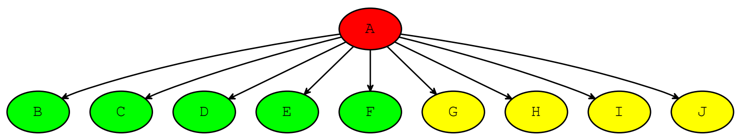
There are number of measures in the literature that are being used to compare clonal trees. Two of the most widely used measures include: (1) Ancestor-Descendant Accuracy (ADA), measure which considers only mutations originating at nodes(clones) which are in ancestor-descendant relationship in the true tree and returns the fraction of pairs of such mutations for which the relationship is preserved in the inferred tree. (2) Different-Lineage Accuracy (DLA), defined analogously as ADA, where only pairs of mutations originating from different clones which are in neither ancestor-descendant nor descendant-ancestor relationship are considered. In addition to these two measures, used in [27, 15, 16, 17] and elsewhere, (3) Clustering Accuracy (CA) [15] and (4) Co-Clustering Accuracy (CCA) [17] were also introduced in order to measure the accuracy in the placement of mutations originating from the same clone in true tree. CA measures the fraction of label pairs that are both co-located in the same vertex in both trees, whereas CCA measures the proximity in the inferred tree of pairs of mutations originating from the same clone in true tree (see [15] and [17] for definitions of CA and CCA). Finally, (5) Pair-wise Marker Shortest Path Distance (PMSPD) [25] is (symmetric) distance measure calculated as the sum, over all label pairs, of the absolute difference of path length between the two labels in true tree with the equivalent length calculated in the inferred tree.

All of the above mentioned are designed to compare inferred tree against the given true tree and no single measure can capture the overall similarity/difference between two arbitrary trees. Furthermore, for each of the measures there exist cases where it returns high similarity for topologically very different true and inferred trees. We will illustrate this below by presenting several examples using trees from Figure 2 where true tree and four trees inferred by (hypothetical) methods are shown. Each vertex in any one of these trees have one or more labels (corresponding to mutations in clonal trees) represented by $A,B,C,\dots,J$.

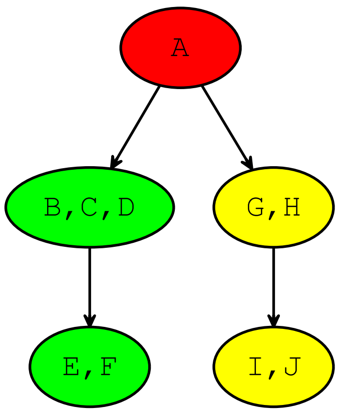
For ADA measure, one needs to consider all pairs of labels in the true tree: $\{(A,B),(A,C),$ $(A,D),(A,E),(A,F),(A,G),(A,H),(A,I),(A,J)\}$. We see that ‘Inferred tree 1’ has the maximum score despite being topologically very different from ’True tree’. The same tree can be



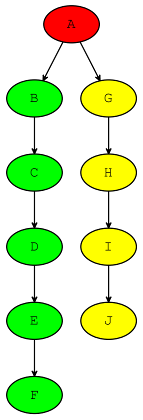
(a) True tree



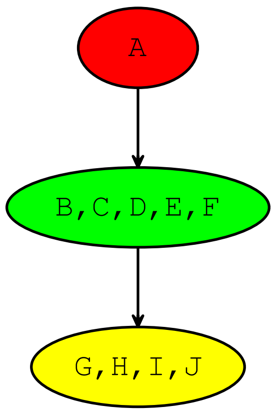
(b) Inferred tree 1



(c) Inferred tree 2



(d) Inferred tree 3



(e) Inferred tree 4

**Figure 2** (a) True clonal tree depicting the evolution of hypothetical tumor. (b)-(e) Hypothetical trees inferred by methods for reconstructing history of tumor evolution (input data to these methods is assumed to be obtained from the hypothetical tumor mentioned in the description of 'True tree'). These trees are used as examples which demonstrate limitations of the existing measures for calculating similarity/distance between true and each of the four inferred trees (details provided in Section 5.1). In Section 5.2.1 we discuss the application of MLTED in calculating similarities between these pairs of trees.

used as an illustration for the limitations of DLA measure where the following set of label pairs need to be considered in true tree $\{(B, G), (B, H), (B, I), (B, J), (C, G), (C, H), (C, I), (C, J),$ $(D, G), (D, H), (D, I), (D, J), (E, G), (E, H), (E, I), (E, J), (F, G), (F, H), (F, I), (F, J)\}$. Clustering of mutations in 'Inferred tree 4' is in the perfect agreement with the clustering in the 'True tree' hence both CA and CCA measures will return maximum score for this tree, even though it is also topologically very different from 'True tree'. Finally, the calculation of the PMSPD measure between the 'True tree' and 'Inferred tree 1', as well as 'Inferred tree 2', is shown in Figure 3. This measure assigns the same score to these two inferred trees, despite the fact that 'Inferred tree 2' is, from the perspective of interpreting tumor evolution, much closer to 'True tree'.

## 5.2 Applications of MLTED

In order to facilitate the interpretation of results, for two arbitrary trees $T_1$ and $T_2$, in addition to the MLTED similarity measure which returns the number of mutations in common tree of $T_1$ and $T_2$ and is denoted here as $MLTED(T_1, T_2)$, we also introduce MLTED-normalized($T_1$, $T_2$) defined as $\frac{MLTED(T_1,T_2)}{\max(a,b)}$, where $a$ and $b$ denote number of mutations in $T_1$ and $T_2$. MLTED-normalized can be interpreted as similarity measure which takes values from $[0, 1]$, with higher values denoting higher similarity among trees. In the discussion of results below, all presented scores represent MLTED-normalized similarity measure, although it is obviously equivalent to MLTED (assuming that the sets of node labels are known for both trees, which is true in all of our comparisons).

### 5.2.1 Application to the synthetic examples with the available ground truth

In this section we discuss similarity between true and inferred trees shown in Figure 2.

‘Inferred tree 1’ has relatively low score equal to 0.3 which rewards the proper placement of mutation A and correctly inferred phylogenetic relations for pairs of mutations originating from different clones, but penalizes for extensive branching which leads to the inaccurate placement to different branches of mutations originating from the same clone, as well as to significant topological differences between this and true tree. In contrast, and as expected based on our discussion from the introduction, ‘Inferred tree 2’ (which represents slightly refined version of ‘True tree’ where green and yellow clones are each split into two adjacent clones belonging to the same branch) and ‘Inferred tree 3’ (which represents fully resolved mutation tree that can be obtained from ‘True tree’) both have score 1. ‘Inferred tree 4’, having score 0.6, is rewarded for the proper placement of mutation A and large cluster of mutations appearing for the first time at green clone, but is penalized for inaccurate placement of yellow clone from where 4 out of 10 mutations originate.

### 5.2.2 Application to real data

In order to demonstrate the application of measure developed in this work in real settings where true tree is usually not available, we analyzed two datasets obtained by sequencing real samples of triple-negative breast cancer (TNBC) and acute lymphoblastic leukemia (ALL). For each sample, we inferred trees of tumor evolution by the use of SCITE [14], SiFit [10] and PhISCS[2]. We provide more details about these methods and parameters used in running them, as well as details of obtaining real data, in the Appendix. Inferred trees and very detailed discussion of the calculated MLTED-normalized scores for pairs of inferred trees are shown in Figure 4 and Figure 5 (for the TNBC sample) and Figure 6 (for the ALL sample). We show that MLTED-normalized score recognizes high similarity in the placement of vast majority of mutations between two trees (as demonstrated for trees inferred by PhISCS and SiFit for TNBC sample where score equals 0.82), but also penalizes for topological differences and different sorting of mutations along linear chains (as demonstrated for trees inferred by SCITE and SiFit for ALL sample where the score equals 0.69).

[2] Available at `https://github.com/haghshenas/PhISCS`

## 6 Conclusion and Future Work

Comparing clonal and mutation trees inferred by different methods and/or by the use of different types of input data, currently represents an important challenge in intra-tumor heterogeneity and evolution studies. In this work, we show that the most widely used measures for comparing clonal/mutation trees usually capture only some specific aspects of tree similarity and can also report the similarity score which is in sharp contrast to the expected result (e.g. perfect score is sometimes given to inferred tree which is very different from the true tree). Motivated by these, we introduce new measure, termed MLTED, for calculating similarity between trees of tumor evolution. From the above discussion and results on synthetic and trees obtained by analyzing real datasets, we conclude that MLTED successfully captures similarities and differences, relevant in proper interpretation of tumor evolution, between two given trees. The proposed similarity measure represents a promising alternative to the existing measures and has wider domain of applications (as it does not require the presence of reference tree). We expect that MLTED becomes a part of the gold standard in assessing the performance of methods for inferring trees of tumor evolution on simulated data, as well as in comparing similarity and finding consensus tree of trees reported by different methods in the cases where real data is used as the input and ground truth is usually not available.

In the current implementation, MLTED is designed for trees built by the use of SNVs under the ISA. Such trees are an output of the vast majority of the existing methods for studying tumor evolution. However, some recent studies based on the analysis of single-cell sequencing datasets suggest the possibility of ISA violation in some cases [23]. With the rapid advancements in the field of single-cell sequencing, we expect developments of sophisticated computational methods based on finite sites model and using SNVs, copy number and other structural aberrations in inferring clonal/mutation trees. These will also require some future work in extending MLTED to properly capture the key differences among trees reported by such methods.

### References

1 A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015. doi:10.1145/2746539.2746612.

2 P. Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, 2005. doi:10.1016/j.tcs.2004.12.030.

3 W. Chen. More efficient algorithm for ordered tree inclusion. *J. Algorithms*, 26(2):370–385, 1998. doi:10.1006/jagm.1997.0899.

4 W. Chen. New algorithm for ordered tree-to-tree correction problem. *J. Algorithms*, 40(2):135–158, 2001. doi:10.1006/jagm.2001.1170.

5 Nilgun Donmez, Salem Malikic, Alexander W. Wyatt, Martin E. Gleave, Colin Collins, and S Cenk Sahinalp. Clonality inference from single tumor samples using low-coverage sequence data. *Journal of Computational Biology*, 24(6):515–523, 2017. doi:10.1089/cmb.2016.0148.

6 A. G. Deshwar et al. Phylowgs: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome biology*, 16(1):35, 2015.

7 C. Gawad et al. Dissecting the clonal origins of childhood acute lymphoblastic leukemia by single-cell genomics. *Proceedings of the National Academy of Sciences*, 111(50):17947–17952, 2014.

8 El-Kebir M. et al. Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell systems*, 3(1):43–53, 2016.

9 F. Strino et al. Trap: a tree approach for fingerprinting subclonal tumor composition. *Nucleic acids research*, 41(17):e165–e165, 2013.

10 H. Zafar et al. Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome biology*, 18(1):178, 2017.

11 Hajirasouliha I. et al. A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data. *Bioinformatics*, 30(12):i78–i86, 2014.

12 J. Kuipers et al. Advances in understanding tumour evolution through single-cell sequencing. *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, 1867(2):127–138, 2017.

13 Jiang T. et al. Alignment of trees - an alternative to tree edit. *Theor. Comput. Sci.*, 143(1):137–148, 1995. doi:10.1016/0304-3975(95)80029-9.

14 K. Jahn et al. Tree inference for single-cell data. *Genome biology*, 17(1):86, 2016.

15 M. El-Kebir et al. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics*, 31(12):i62–i70, 2015.

16 S. Malikic et al. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics*, 31(9):1349–1356, 2015.

17 S. Malikic et al. Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. *To appear in proceedings of RECOMB*, 2018.

18 W. Jiao et al. Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC bioinformatics*, 15(1):35, 2014.

19 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

20 J. Jansson and A. Lingas. A fast algorithm for optimal alignment between similar ordered trees. *Fundam. Inform.*, 56(1-2):105–120, 2003. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi56-1-2-07.

21 R. Kim, K.I. & Simon. Using single cell sequencing data to model the evolutionary history of a tumor. *BMC bioinformatics*, 15(1):27, 2014.

22 P.N. Klein. Computing the edit-distance between unrooted ordered trees. In *Algorithms - ESA ’98, 6th Annual European Symposium, Venice, Italy, August 24-26, 1998, Proceedings*, pages 91–102, 1998. doi:10.1007/3-540-68530-8_8.

23 Jack Kuipers, Katharina Jahn, Benjamin J Raphael, and Niko Beerenwinkel. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome research*, 27(11):1885–1894, 2017.

24 P. Kilpeläinen & H. Mannila. Ordered and unordered tree inclusion. *SIAM J. Comput.*, 24(2):340–356, 1995. doi:10.1137/S0097539791218202.

25 E. M. Ross & F. Markowetz. Onconem: inferring tumor evolution from single-cell sequencing data. *Genome biology*, 17(1):69, 2016.

26 P.C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194(4260):23–28, 1976.

27 Victoria Popic, Raheleh Salari, Iman Hajirasouliha, Dorna Kashef-Haghighi, Robert B West, and Serafim Batzoglou. Fast and scalable inference of multi-sample cancer lineages. *Genome biology*, 16(1):91, 2015.

28 Daniele Ramazzotti, Alex Graudenzi, Luca De Sano, Marco Antoniotti, and Giulio Caravagna. Learning mutational graphs of individual tumor evolution from multi-sample sequencing data. *arXiv preprint arXiv:1709.01076*, 2017.

29 S. Muthukrishnan & S.C. Sahinalp. An efficient algorithm for sequence comparison with block reversals. *Theor. Comput. Sci.*, 321(1):95–101, 2004. doi:10.1016/j.tcs.2003.05.005.

**30** S. M. Selkow. The tree-to-tree editing problem. *Inf. Process. Lett.*, 6(6):184–186, 1977. doi:10.1016/0020-0190(77)90064-3.

**31** K. Zhang & D.E. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989. doi:10.1137/0218082.

**32** D. Shapira & J.A. Storer. Edit distance with block deletions. *Algorithms*, 4(1):40–60, 2011. doi:10.3390/a4010040.

**33** Kuo-Chung T. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979. doi:10.1145/322139.322143.

**34** J. Matoušek & R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992. doi:10.1016/0012-365X(92)90687-B.

**35** R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974. doi:10.1145/321796.321811.

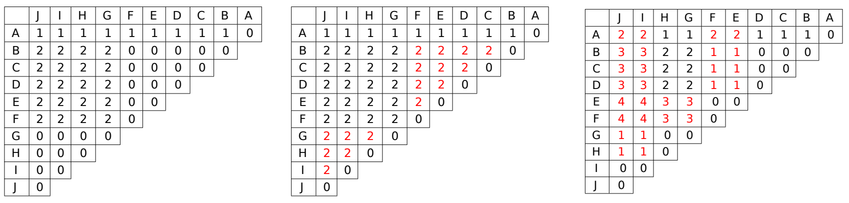
**36** Yong Wang, Jill Waters, Marco L Leung, Anna Unruh, Whijae Roh, Xiuqing Shi, Ken Chen, Paul Scheet, Selina Vattathil, Han Liang, et al. Clonal evolution in breast cancer revealed by single nucleus genome sequencing. *Nature*, 512(7513):155, 2014.

**37** D. Shasha & K. Zhang. Fast algorithms for the unit cost editing distance between trees. *J. Algorithms*, 11(4):581–621, 1990. doi:10.1016/0196-6774(90)90011-3.

**38** K. Zhang. Algorithms for the constrained editing distance between ordered labeled trees and related problems. *Pattern Recognition*, 28(3):463–474, 1995. doi:10.1016/0031-3203(94)00109-Y.

**39** K. Zhang and T. Jiang. Some MAX snp-hard results concerning unordered labeled trees. *Inf. Process. Lett.*, 49(5):249–254, 1994. doi:10.1016/0020-0190(94)90062-0.

## Supplementary Figures



| | J | I | H | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| B | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | |
| C | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | | |
| D | 2 | 2 | 2 | 2 | 0 | 0 | 0 | | | |
| E | 2 | 2 | 2 | 2 | 0 | 0 | | | | |
| F | 2 | 2 | 2 | 2 | 0 | | | | | |
| G | 0 | 0 | 0 | 0 | | | | | | |
| H | 0 | 0 | 0 | | | | | | | |
| I | 0 | 0 | | | | | | | | |
| J | 0 | | | | | | | | | |

(a) Distances for 'True tree'

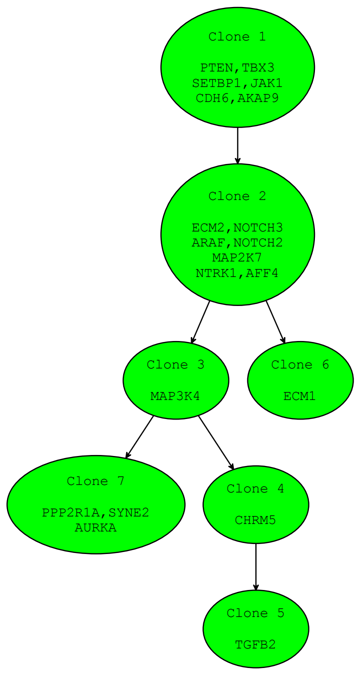
| | J | I | H | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| B | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | |
| C | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | | |
| D | 2 | 2 | 2 | 2 | 2 | 2 | 0 | | | |
| E | 2 | 2 | 2 | 2 | 2 | 0 | | | | |
| F | 2 | 2 | 2 | 2 | 0 | | | | | |
| G | 2 | 2 | 2 | 0 | | | | | | |
| H | 2 | 2 | 0 | | | | | | | |
| I | 2 | 0 | | | | | | | | |
| J | 0 | | | | | | | | | |

(b) Distances for 'Inferred tree 1'

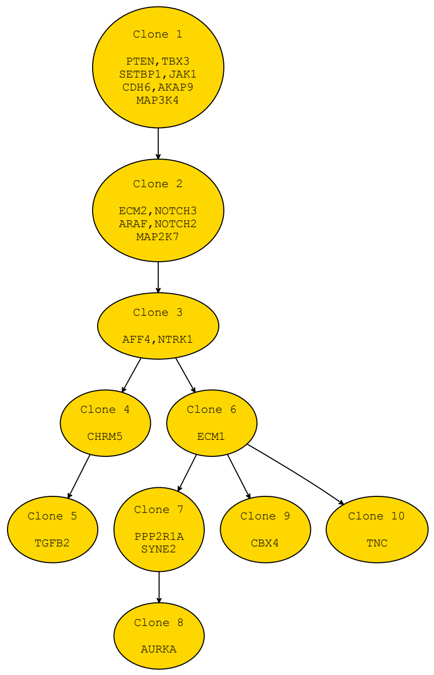
| | J | I | H | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 0 |
| B | 3 | 3 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | |
| C | 3 | 3 | 2 | 2 | 1 | 1 | 0 | 0 | | |
| D | 3 | 3 | 2 | 2 | 1 | 1 | 0 | | | |
| E | 4 | 4 | 3 | 3 | 0 | 0 | | | | |
| F | 4 | 4 | 3 | 3 | 0 | | | | | |
| G | 1 | 1 | 0 | 0 | | | | | | |
| H | 1 | 1 | 0 | | | | | | | |
| I | 0 | 0 | | | | | | | | |
| J | 0 | | | | | | | | | |

(c) Distances for 'Inferred tree 2'

**Figure 3** Distances between pairs of labels required for calculating Pair-wise Marker Shortest Path Distance (PMSPD) for trees from Figure 2. Entries in each matrix represent length of path between labels (note that labels are shown in the first row and the first column of each matrix). Distance is calculated as the sum of absolute values of differences between pairs of entries which are at the same position in both matrices. Red colored entries in labels pairwise distance matrix shown in (b)/(c) differ from the corresponding entries in matrix for true tree shown in (a) and therefore contribute to the overall distance. PMSPD assigns the same score to 'Inferred tree 1' and 'Inferred tree 2', despite the fact that 'Inferred tree 2' is, from the perspective of interpreting tumor evolution, much closer to 'True tree'.



(a) Tree for TNBC dataset inferred by SiFit



(b) Tree for TNBC dataset inferred by PhISCS

**Figure 4** Clonal trees of tumor evolution, inferred by SiFit and PhISCS, for triple-negative breast cancer (TNBC) dataset originally published in [36] and consisting of the binary presence/absence profile of 22 mutations across 16 single cells. Names of the clones are assumed not to be included as part of the node label. Trees are very similar to each other in placement of the vast majority of mutations: (i) Clone 1 in the SiFit tree is almost identical (with respect to the set of mutations assigned to its label) to Clone 1 in PhISCS tree (ii) Clone 2 in SiFit tree is split into two adjacent clones, namely Clone 2 and Clone 3, in PhISCS tree. Analogous applies to Clone 7. (iii) The order of mutations in genes CHRM5 and TGFB2, as well as in most other pairs of mutations (including the pairs where both mutations are at the same node), is same among the trees. Notable exceptions leading to some dissimilarities between the trees include mutations in genes MAP3K4 and ECM1. In addition, mutations in genes CBX4 and TNC are absent in tree reported by SiFit. Removing these four mutations and their corresponding nodes from each tree (if present) and assigning each of the Clone 4 and Clone 7 in SiFit tree as child of Clone 2, and Clone 7 as child of Clone 3 in PhISCS tree, we obtain trees which are same up to the existence of splits of single into two adjacent clones belonging to the same lineage (see (ii) from above). MLTED-normalized score for the two trees equals 0.82, which well reflects the overall high topological similarity and concordance in ordering pairs of mutations.

,

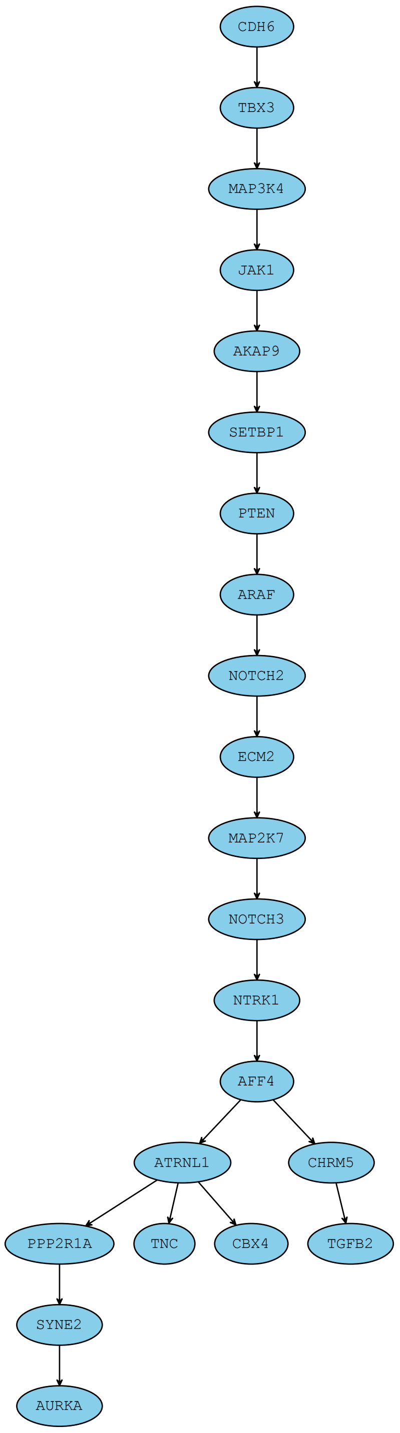
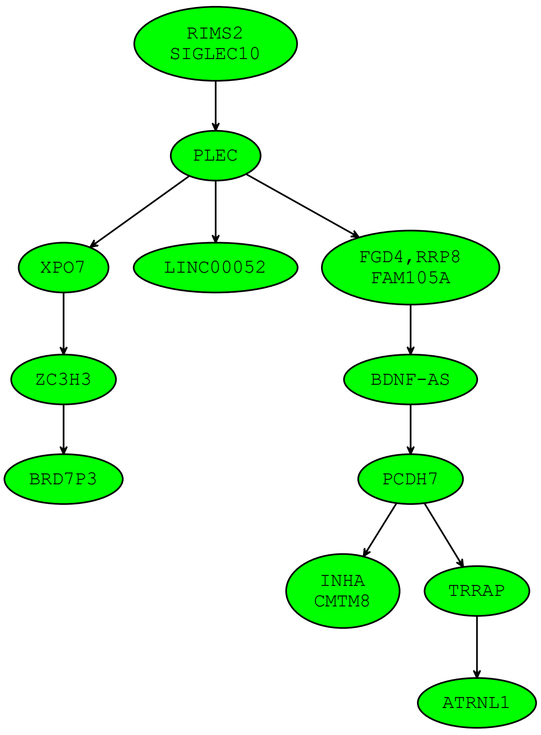
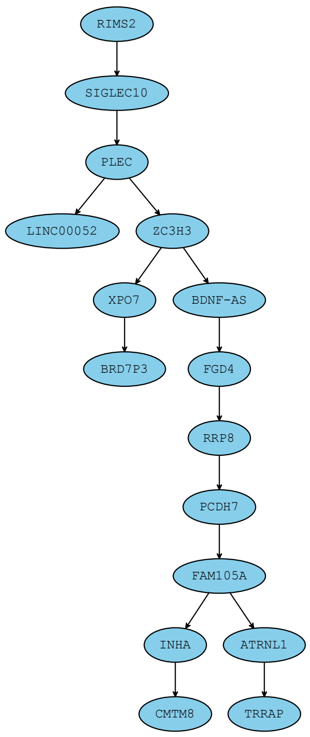


**Figure 5** Mutation tree for TNBC dataset (see Figure 4 for details) inferred by SCITE. This tree can be obtained from PhISCS tree by expanding nodes having more than one label, hence MLTED-normalized score between the two trees is maximum possible (i.e. equals 1). Compared with tree inferred by SiFit, SCITE tree has analogous topological similarities and differences as tree inferred by PhISCS, and MLTED-normalized score for these two trees is also equal to 0.82.

,



(a) Tree for ALL dataset inferred by SiFit



(b) Tree for ALL dataset inferred by SCITE

**Figure 6** Trees inferred by SCITE and SiFit for acute lymphoblastic leukemia (ALL) patient dataset from [7] consisting of 115 single cells and 16 mutations. Unsurprisingly, due to large number of single-cells in this dataset, sequencing noise and similarities in the scoring schemes used in PhISCS and SCITE (see Section A) both methods report the same mutation tree so we only focus on SCITE in this discsussion. The most notable difference among the two trees is in the placement and ordering of mutations in genes ZC3H3, XPO7 and BRD7P3 as well as in the ordering of mutations in genes FGD, RRP8, FAM105A, BDNF-AS and PCDH7. Furthermore, the relative order also differs for mutations in genes TRRAP and ATRNL1. However, in contrast to these important differences, the trees still share most of the major branching events in tumor evolution and have consistent ancestor-descendant order for most of the pairs of mutations. All these are reflected in MLTED-normalized score of 0.69 assigned to this pair of trees.

,

## A Details of obtaining trees of tumor evolution for the real data sets

### A.1 Summary of methods used for inferring trees of tumor evolution

In this work, we inferred trees of tumor evolution by the use of SCITE [14], SiFit [10] and PhISCS[3]. Each of the methods takes as the input single-cell sequencing (SCS) data matrix and estimated noise rates of SCS experiment. The underlying scoring used in PhISCS is analogous to that in SCITE and the major difference among the two methods is in the type of tree returned in the output. While SCITE searches for the maximum likelihood mutation tree, PhISCS reports the maximum likelihood clonal tree. Due to the equivalence in tree scoring, assuming that both methods find the optimal solution, clonal tree reported by PhISCS is expected to represent compressed version of mutation tree reported by SCITE (i.e. we expect that tree reported by SCITE belongs to the set of mutation trees which can be obtained from clonal tree reported by PhISCS). Similarly as PhISCS, SiFit also returns clonal tree of tumor evolution but uses different tree search methodology and does not necessarily yield the same output as PhISCS nor SCITE (as demonstrated in [10]).

### A.2 Details of obtaining input data and running SCITE, SiFit and PhISCS

We obtained binary SCS data mutation matrix for TNBC patient sample from [28] and for ALL patient sample from [7]. For each sample, false positive and false negative rates of sequencing experiment were estimated in the original studies [36, 7] and provided as the input to the methods used in the analysis. In order to obtain better convergence, we run MCMC based methods SiFit and SCITE for very large number of iterations. For SiFit, we set number of iterations to 5,000,000. For SCITE we set number of repetitions of the MCMC to 3 and chain length of each MCMC repetition to 1,000,000. PhISCS is combinatorial optimization based method which provided guarantee of the optimality for each solution.

[3] Available at `https://github.com/haghshenas/PhISCS`