Study of Phylogenetic Methods — Independent Study

Karan Dhingra
MT19025

I. Abstract

Phylogeny is the study of organisms, but it has gained quite an attraction in cancer study because of tumor cells being heterogeneous, which implies that the morphological behavior of two cells differs, but their genetic structure changes. Studying tumor cell's phylogeny aims to get more information about the driving mutation, which caused an uncontrolled cell division. In some studies, it has been shown that driving mutation acts as a better biomarker. This study starts from phylogeny basics, followed by a state of the art techniques to get a phylogenetic tree from mutation data and different evaluation techniques to compare the methods.

II. Introduction

Phylogeny is the study of relationships among different groups of Organisms. It helps us understand how an organism has evolved, which is quite essential in cancer study as tumor cells are heterogeneous, i.e., tumor cells can show different morphological and phenotypic profiles. This behavior occurs both inside a tumor (intra-tumor heterogeneity) and between tumor cells (inter-tumor heterogeneity). This heterogeneity is the result of an imperfection DNA replication between different tumor cells. The phylogeny of tumor cells helps us understand how the tumor evolved, and biomarkers in therapy have been proven effective to stop cancer when the marker is set to the tumor cell, driving the mutations[1]. In this study, we explore different techniques to understand the phylogeny of tumor cells. Section III explains the importance of phylogeny in cancer study and medications, while section IV explains the basics of phylogeny, followed by rules defining perfect phylogeny and different algorithms used to achieve perfect phylogeny. Finally, we discuss the implementation of different algorithms, followed by evaluation techniques, datasets.

III. Tumor Heterogeneity

In our body, cell division is a normal process that happens for growth and repair. A cell becomes tumorous if it divides uncontrollably, leading to uncontrolled growth. A similar treatment can show different results to different patients with similar clinical and pathological characteristics like the location of a tumor, type, and grade. It happens due to the tumor cells being heterogeneous, i.e., different tumor cells show different morphological and phylogenetic traits. Morphological traits imply the structure of the tumor cells, and phylogenetic traits represent genetic structure. There are two kinds of heterogeneity, (i) intratumor heterogeneity, which exists due to imperfect replication of the DNA between cells, and (ii) intertumor heterogeneity, which can be because of imperfect replication in metastasis tumor (tumor which can jump locations), or due to presence of a different kind of tumor naturally. It has been observed that the tumor cell which is driving the mutations[1] forms a useful biomarker, which can then be used to decide an effective therapy. In phylogeny, this cell is also known as the parent cell or the cell, which started the uncontrolled growth. Finding which mutations lead to the start of this uncontrolled growth can lead to a better understanding of the tumor and how to cure it effectively.

IV. Concept of Phylogeny

Phylogeny is the study of relationships amount different organisms. In the case of a tumor, it is a phylogenetic study of tumor cells. The phylogenetic tree of a tumor has a root node, an unmutated cell, followed by multiple child nodes depicting mutations at specific sites(figure 1).

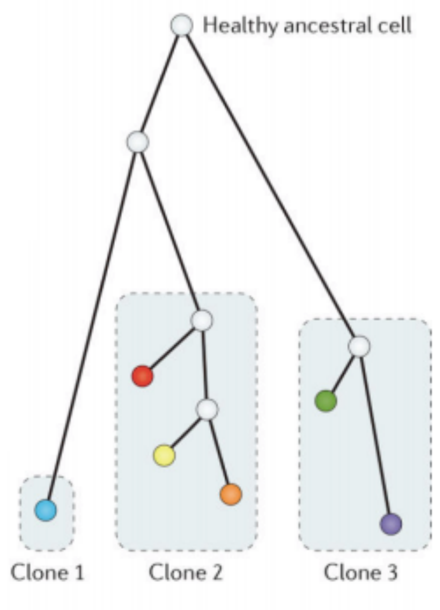


Figure 1. Phylogenetic Tree

One way to understand the phylogeny would be to sequence a single cell over time and observe its transition(known as single-cell sequencing), but it is technological and cost-ineffective; millions of cells present in a tumor. Thus, instead, we bulk sequence cells to understand the phylogeny to cluster subpopulations with similar somatic mutations.

These sub-populations constitute the tumor's phylogeny, but to understand its architecture (or evolution), we need to restructure it into a perfect phylogeny configuration. A perfect phylogenetic tree follows the assumption of infinite-sites[2], i.e., a mutation occurs over a particular site only once during the tumor cell's lifetime. This principle ensures there are no cycles in a phylogenetic tree. This assumption holds true in the real world because of a significantly lower percentage of somatic mutations than the total number of genome positions. e.g., In table-I, we have four examples (or subpopulations), and in each instance, we have sequenced four sites.

-	C1	C2	C3	C4
R1	1	1	1	0
R2	1	0	0	1
R3	1	0	1	0
R4	1	1	0	0

Table I
EXAMPLE : PERFECT PHYLOGENY

As we can see, that samples {R1, R2, R3}, or {R1, R2, R4} follows perfect phylogeny, and it is because sample R1 has a mutation in both C2, and C3, while there is no mutation in R3 for C2, and R4 for C3. Sample R3 and R4 show that C2 and C3 mutations happened independently, while sample R1 shows that C3 and C4 mutations occurred together, and this would break the perfect phylogeny's infinite-sites assumption. Thus, a phylogeny algorithm aims to enforce the infinite-sites assumption by decomposing a few mutations in some samples.

A. Algorithms

There are different kinds of an algorithm, based on the resolution of input it takes, and the resolution of the tree the algorithm can generate.

1) Maximum Parsimony Algorithm: It takes boolean matrix M of shape $[N_{samples}, N_{sites}]$, where true represents the presence of a mutation at that location, and zero means absence, respectively. It generates a similar matrix M of shape $[N_{clones}, N_{sites}]$, where N_{clones} represents decomposed samples. It works on Occam's razor principle rather than infinite-sites assumptions and aims to generate a tree that minimizes the homoplasy (or parallel mutation). The maximum parsimony algorithm[3] solves a smaller parsimony problem, i.e., given a phylogenetic tree with $N_{samples}$ leaf nodes, it generates mutations for internal nodes.

Algorithm 1 Maximum Parsimony Algorithm - Back pass

```
Result: Labelled internal nodes of the tree
Input:
  Tree with $N_{samples}$ leaf nodes, corresponding to input;
Backward Pass:
  for node in tree_nodes do
      if node is leaf then
          state(node) = mutation(node);
      else
          if disjoint(state(node.children)) then
              state(node) = union(state(node.children));
          else
              state(node) = and(state(node.children));
          end
      end
end
```

We first do a backward pass over the input tree, which sets the node's state, according to algorithm-1, where union, disjoint & and are the logical operators. The state of a node is a vector of length N_{sites} which stores a set at each location, so this computation happens for each site independently.

Algorithm 2 Maximum Parsimony Algo - Forward pass

```
Forward Pass:
label(Root) = [random(e) for e in state(Root)];
for node in tree_nodes do
    if label(node) is Null then
        for (site in $N_{sites}$) do
            node_state = state(node)[site];
            parent_label = label(node.parent))[site];
            if (parent_label in node_state) then
                label(node)[site] = parent_label;
            else
                label(node)[site] = random(node_state);
            end
        end
    end
end
```

In the forward pass(2), we randomly assign mutation to the node from its state if the parent has been assigned a different mutation; otherwise, we copy the parent mutation. In practice, for testing, we biased setting 0 in case of random assignment when both $\{0, 1\}$ mutations character were available in the state for a given site.

The maximum parsimony algorithm only solves the small parsimony problem. Still, the input only consists of un-ordered mutation samples; we do not have the tree's structure (or which sample is closer to the other sample). To generate tree structure, we use a distance-based clustering algorithm[4] (4)

Algorithm 3 Distance based clustering

```
Input: Matrix M of shape [$N_{samples}$, $N_{sites}$];
for i, j in $N_{Samples}$ do
    distance[i, j] = count(disjoint(M[i], M[j]))
end
while $len(distance) > 1$ do
    first_node, second_node = argmin(distance)
    pop(second_node), pop(first_node)
    push(merge(first_node, second_node))
end
```

Where $argmin$ returns nodes with minimum distance, the pop function removes a node from the distance matrix. In contrast, $merge$ functions create a node with $first\_node$ and $second\_node$ as its children, and $push$ functions push the merged node into a distance matrix to structure the samples based on closeness in somatic mutations.

2) MixPhy[5]: Similar to Maximum Parsimony Algorithm, it takes a matrix M of shape $[N_{samples}, N_{sites}]$, and generates a matrix of shape $[N_{clones}, N_{sites}]$. It aims to generate a perfect phylogenetic tree, i.e., which follow infinite sites assumption. It places N_sites into k different buckets, such that there is no intra bucket conflict, and there is inter bucket conflict present. It then decomposes each sample, S_i, into k different samples S_i^k, where each sub-sample S_i^p can only have somatic mutations belonging to that bucket.

for any two sites, Si_i, Si_j, there is a conflict if either $Si_i \nleq Si_j$ or $Si_j \nleq Si_i$ $\forall$ samples, and $intersection(Si_i, Si_j) \neq \phi$. This is the same condition as that of perfect phylogeny. Still, there is no polynomial algorithm that can find the appropriate buckets, as we would have to check $2^{N_{sites}}$ possibilities.

In mixphy, this condition is relaxed by creating a comparable graph, which has sites as vertices, and there is a directed edge from Vertex $Si_i \rightarrow Si_j$ if $Si_i \leq Si_j \forall$ samples. In another term, there is an edge from S_i to S_j if site S_i has no conflicts with site S_j. In figure 1, we will have an edge from $C_2 \rightarrow C_1, C_3 \rightarrow C_1, C_4 \rightarrow C_1$ but not from $C_2 \rightarrow C_3$, or $C_3 \rightarrow_2$ which are conflicting in nature. It leads to exclusion of valid cases like $C_4 \rightarrow C_3, C_4 \rightarrow C_2$ from the graph. This relaxation leads to the graph structure, becoming co-comparable in nature, and its undirected version always forms clique in nature. The minimum number of cliques that covers all of the vertices are the k different buckets, with each bucket having columns associated with each clique.

Example: For a matrix shown in figure:2, we have a directed graph shown in fig:2. In the undirected co-comparable graph, we would have three unique cliques corresponding to $C_{13}, C_{12},\ and, C_{14}$. Thus, the perfect phylogeny of this example would be(table:II)

Thus, the mixphy algorithm decomposed the mutation,

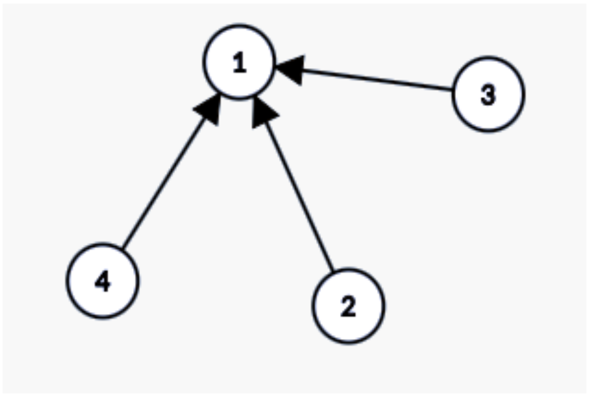


Figure 2. Directed Similarity graph

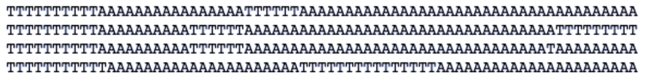
-	C1	C2	C3	C4
$R1_1$	1	1	0	0
$R1_2$	1	0	1	0
$R1_2$	1	0	0	0
$R2_1$	1	0	0	0
$R2_2$	1	0	0	0
$R2_3$	1	0	0	1
$R3_1$	1	0	0	0
$R3_2$	1	0	1	0
$R3_3$	1	0	0	0
$R4_1$	1	1	0	0
$R4_2$	1	0	0	0
$R4_3$	1	0	0	0

Table II
OUTPUT - EXAMPLE 1

which was saying C_2, C_3 happened simultaneously, and ensure there is no homoplasy in the phylogeny.

Changes: While going through the work of Mixphy and its implementation, we observed that the authors had solved this problem using the Network-flow problem, but it should be easily solved using the longest path in the undirected graph as the graph would always form a clique, and the longest path would be the size of that clique.

While observing the algorithm's output, they assign distinct clique id to a vertex while observing (as shown in the example below) that a vertex can have more than one clique id. We came to this conclusion when we were testing the output on a bigger dataset (G12); results from both implementations (ours' clique and author's network-flow) were not correct, and once we assigned more than one clique id to a vertex, we got quite improved results.



```
TTTTTTTTTATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATAAAAAAATTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTTAAAAAAAAATTTTTTAAAAATAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTTAAAAAAAAATTTTTTTTTTTAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTT
TTTTTTTTTTAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAA
TTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAATTTTTTTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAA
```

Figure 3. G12: Ground Truth

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTTTTTTAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTT
AAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAATAAAAAAATTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAATAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 4. G12: Mixphy (Author's Implementation)

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTTTAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAA
AAAAAAAAAAAAAAAAAATTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAATTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 5. G12: Mixphy (Our's - Before Correction)

As we can observe that both of the implementations generated an absence of mutation in initial sites for many clones, which are not at all present in the ground truth. Also, it does not match the results from the maximum parsimony algorithm.

```
TTTTTTTTTATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATAAAAAAAATTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTAAAAATAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTTTTTTAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTT
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAATTTTTTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 6. G12: Maximum Parsimony Algorithm

But after doing the correction as mentioned earlier, we were able to generate similar (if not better) results, though it needs more evaluation over different datasets.

```
TTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATAAAAAAAATTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTATTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTAAAAATAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTAAAAAAAAATTTTTTTTTTAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAATTTTTTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTTTTTT
TTTTTTTTTTAAAAAAAAAATTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATAAAAAAAAA
```

Figure 7. G12: Mixphy (Our's - After Correction)

Another way to see this is to observe the co-comparability graph of the G12 dataset(figure: 8). There are nine input samples in the dataset with 69 sites, but mixphy observes only 14 unique sites as it is not dependent on the ordering of the sites. We have seven colors (or unique cliques) present in the dataset, and there are three cliques that only consists of only one site because bigger cliques occupy other sites. If we do not add these sites to the clique, those cliques will lose the somatic structure of the G12 phylogeny.

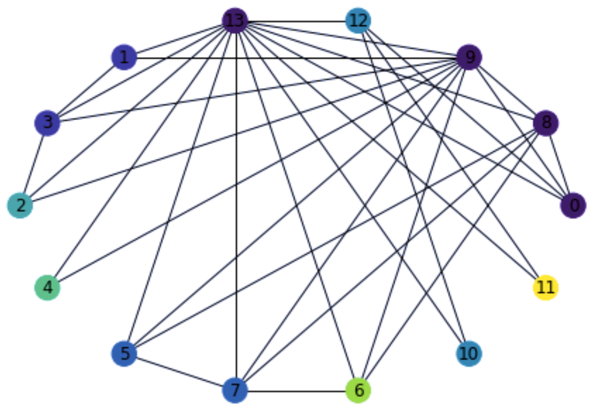


Figure 8. G12: Co comparability Graph

3) Clustering Based Algorithms: These algorithms take the M matrix as input, but the matrix is not boolean but $\in [0, 1]$ (also known as VAFs), representing the presence of mutation infraction. The output from the algorithm is of shape $[N_{clusters}, N_{clusters}]$, where $N_{clusters}$ are the number of unique structures found in the data, and thus provide the only coarse phylogenetic structure of the phylogeny.

- **Beta Mixture Modeling[6]**: In this clustering techniques, the aim is to model input distribution using a mixture of multiple beta models. It uses a number of beta models(N_B) as a hyper-parameter and uses the EM algorithm to find each beta model's optimal parameters.

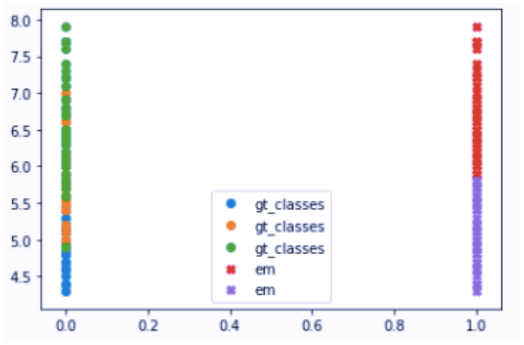


Figure 9. Beta Model Experiment on 1D synthetic data

- **Kernel Density Estimation**: In this clustering techniques, the aim is to directly model the distribution of the input data rather than estimating its mixture of different models. It does not have number of clusters as hyper-parameter, which is un-intuitive on a higher dimension. Still, it has bandwidth as a hyper-parameter, i.e., till how far from the peak the elements should be in the same clusters.

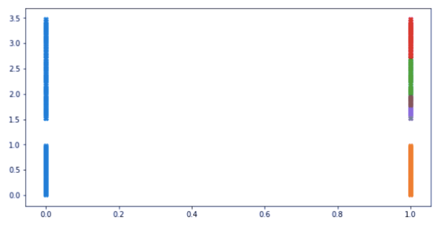


Figure 10. ClusterKDE : Bandwidth is the eucledian distance

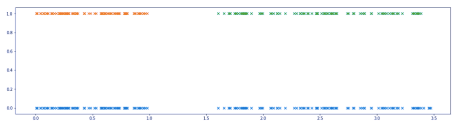


Figure 11. ClusterDP : Bandwidth is the Density

In clusterKDE[7], points are cut-off from a cluster based on a threshold distance, which does not work very well in every condition (fig:IV-A3). While clusterDP[8] uses the distance between KDE values of different points as a threshold, which allows it to work well in conditions where the density of two clusters is the same but points in one cluster are distributed farther as compared to other(fig:IV-A3).

- **SciClone and QuantClone**: SciClone[9] uses Variational Bayesian Mixture model(similar to Beta Mixture model), which provides straightforward computational techniques rather than monitoring in case of MCMC. Their model, has been tested by community and uses CloneEvol to provide the structure of the phylogenetic tree. QuantClone[10] aims to improve upon the performance of SciClone by incorporating genotype information along with VAFs as:

$$\theta = (VAF) * \frac{N_C + N_{CN} + \frac{1-p}{p}}{N_C} \tag{1}$$

where N_C is number of copies in tumor cell, while N_{CN} is number of copies in normal cell, p is the impurity, and θ denotes the genotype information encoded with the VAF.
To compare cluster-based methods with others, we need to find fine-level information; thus, we look for the possibility of whether SciClone, when combined with CloneEvol[11], can also provide the information for each site, but upon searching through the codebase of CloneEvol, QuantClone, and SciClone, it turns out that when we construct the cluster, we loose all of the fine information up to site level.

4) Clonefinder[6]: This algorithm takes VAFs as input, and computes maximum parsimonious tree with fine information, i.e. it takes a matrix of shape $[N_{samples}, N_{sites}]$, and returns a matrix of shape $[N_{clones}, N_{sites}]$. It uses regression to minimize the loss between $||V_i - V_c||$ where V_i is the input VAF, and $V_c = F.M$ where F is the frequency matrix of shape $[N_{samples}, N_{clones}]$, and M is the output phylogeny matrix of shape $[N_{clones}, N_{sites}]$.

Algorithm 4 Clonefinder

```
Input: Matrix $V_i$ of shape [$N_{samples}$, $N_{sites}$];
Algorithm:
$M = V_i > 0$;
$M = Unique(M, axis = 0)$
Base Case:
for $i$ in $N_{samples}$ do
    $freq[i] = regression(M, V[i])$
end
$V_c = freq.M$
if ( $\|V_c - V_i\| < threshold).all()$ then
    return;
end
$M = max\_parsimony(M)$
while ~(—$V_c - V_i| < threshold).all()$ do
    for ($i$ in $N_{samples}$) do
        if ~($\|V_c[i] - V_i[i]\| < threshold).all()$ then
            $V_i[i] = decomposes(V_i[i], V_c[i])$
            M = $V_i > 0$
            $M = max\_parismony(M)$
        end
    end
end
```

Where decomposes(V_i, V_c) takes predicted vs real values for each sample and cluster it into a different region so that the decomposed regions have less variance altogether.

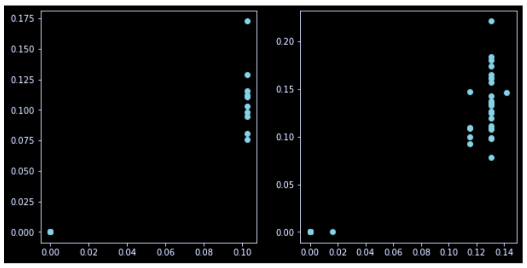


Figure 12. V_c(x-axis) vs V_i(y-axis) for two samples

B. Generation of Tree from Perfect Phylogeny[4]

Given M, which has no conflict present in the columns, ancestors are found using the following algorithm:

```
Algorithm 5 Tree from Perfect Phylogeny
Input: Matrix $M$ of shape [$N_{samples}$, $N_{sites}$];
Algorithm:
sort(M, axis=1) over columns in descending order;
append $(_0, M$, where $_0 \in \{1\}$;
for C, C' in columns, where $C \neq C'$ do
    if $C' \subseteq C$ then
        draw an edge in T from $C' \to C$.
    end
end
```

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

$$M_{sorted} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

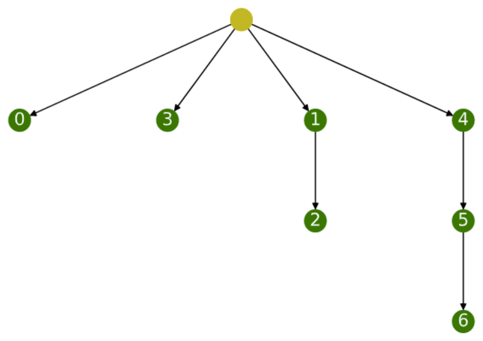


Figure 13. Output : Tree from Perfect Phylogeny

Where each number denotes the sample from matrix M.

V. Implementations

A. Clonefinder

There was an implementation available, but it used MEGA-X software for almost all of the computation. We re-implemented all of the MEGA components to aid us in understanding the importance of each component. While the original implementation uses the Maximum parsimony algorithm from MEGA, which has very little information regarding implementation. Out implementation uses distance-based clustering and the Fitch-Hartigan algorithm[4], which forms an improved version of Maximum parsimony.

B. Mixphy

Again, for Mixphy, an implementation was available, but it was (i) written in C and (ii) solved the complete problem as a network flow problem. We re-implemented the API in python and computed the longest path in the graph to color the graph. We observed similar performance in G7 and G12 datasets. The other benefits of implementing Clonefinder and Mixphy was integrating Mixphy and Clonefinder algorithm in a single pipeline and solving the problem of Perfect phylogeny from the VAFs compared to a maximum parsimonious tree by Clonefinder.

C. QuantClone, Clonevol

To use both of the algorithms with Tree metrics, we needed to get site level information from these algorithms, which was missing. These algorithms do not solve the small-parsimony problem. To understand their codebase better to retrieve fine information, we tweaked a lot of the code, and output from different functions was tested to find that missing information. Still, upon detailed study, we concluded that this information is lost at the time of cluster formation.

VI. Metrics

We have a matrix of shape $[N_{clones}, N_{sites}]$ as ground truth, and $[N_{clones\_pred}, N_{sites}]$ as output from different algorithms. It is possible that for a given algorithm N_{clones_pred} is not same as N_{clones}. The first evaluation metric is mean absolute error, where we compute MAE between each clone of ground truth and take an average of minimum error, which covers both of the matrices. As the aim is to compute the phylogenetic tree, we have metrics that work on trees directly.

A. MLTED Score

MLTED[12] computes number of edits required to convert Tree A to Tree B. On comparison of Tree 1 (fig:14) with Tree 3, and 4 (fig:16, fig:17 respectively) the score is 1, and edit distance is 0 because Tree 0 is lacking information with Tree 3 and 4 has, while the edit distance between Tree 1,3,4 and Tree 2 (fig:15) is of 14 and similarity of 3 because only three nodes are similar (a, b, c) between any pair.

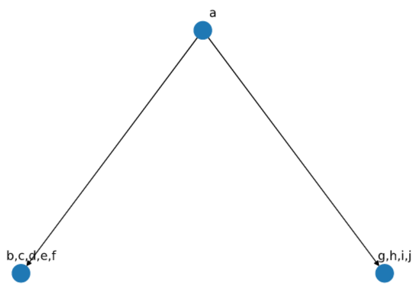


Figure 14. MLTED: Tree - 1

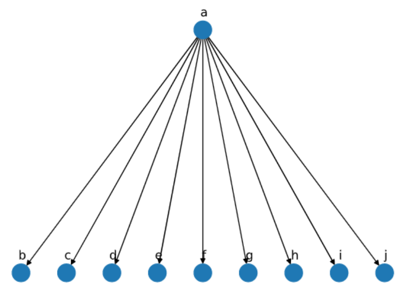


Figure 15. MLTED: Tree - 2

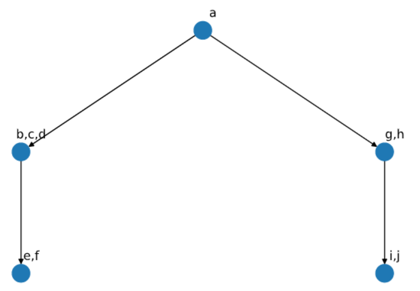


Figure 16. MLTED: Tree - 3

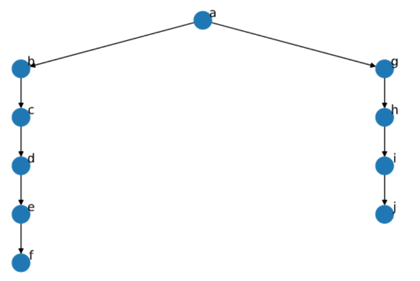


Figure 17. MLTED: Tree - 4

In Tree 2, b,c,d, mutations happened parallelly, while in Tree-4, the mutations are sequential, and Tree-1, 3 lacks this information altogether.

B. *RF metric*

RF metric[13] is implied by the sum of two terms, A+B, where A represents a number that partition the data implied by $T1$ but not $T2$, while B represents versa. RF metric ensures that we have the exact ordering of leaves in the tree, whereas MLTED only ensures that nodes(can contain multiple taxas) are incorrect order. It is impossible to compare the trees present in the above tables, as the RF metric requires the same set of leaf nodes in the tree. Therefore, the table given below will help in a better understanding of the RF metric.

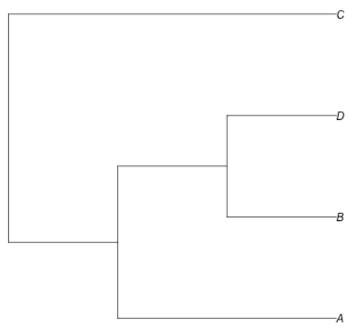


Figure 18. RF: Tree - 1

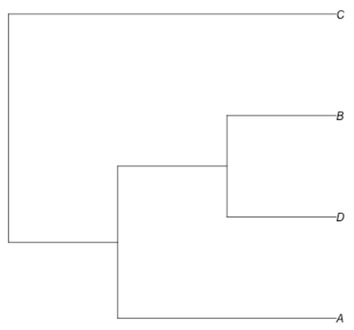


Figure 19. RF: Tree - 2

In RF distance, we must calculate the sum of splits present in tree 1 but not tree 2 and vice versa. Therefore using the edge labelling (in red color, fig:21). We define the split of each tree (q split is not required as it would be the same as that of x) as

There are no splits between Tree 1(fig:18) and Tree 2(fig:19) as it does not matter which branch splits it because

Tree1	Tree2	Tree3
C(x)ABD	C(x)ABD	B(x)ACD
D(y)ABC	B(y)ADC	C(y)ABD
B(z)ACD	D(z)ACB	D(z)ACB
A(r)BCD	A(r)BCD	A(r)BCD
BD(p)AC	BD(p)AC	CD(p)AB

Table III
EDGE LABELLING-RF METRIC

of being unrooted. While there are two unique splits in the case of Tree 3(fig:20) and Tree 1 or Tree 2. Thus, the RF distance would be 2 in this case. If needed, normalization is done by dividing it by the total number of splits present in each tree. Theoretically, we are not bounded by comparing a tree that does not have the same number of leaves, but no implementation is available.

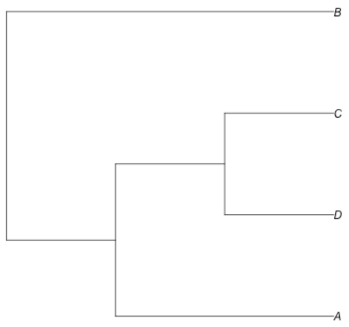


Figure 20. RF: Tree - 3

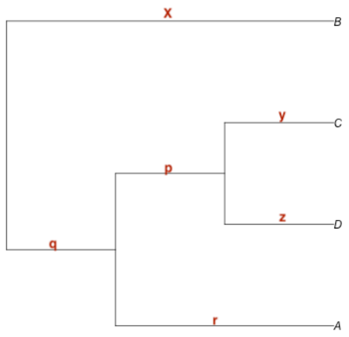


Figure 21. RF: Tree - Edge labelled

C. TreeVec

TreeVec[14] is also a metric for an unrooted tree and is primarily to pool different leaves based on their context. Basically, if the tree has leaf nodes as $a1, a2, b1, b2$. It implies $a1, a2$ belongs to category a while $b1, b2$ belongs to category b. No other method can help in this case. So, this method is very trivial.

Algorithm 6 TreeVec Metric

```
1) Compute minimum depth from the root (can be assigned to any node) to the pair of leaf nodes (for both trees)
2) Take mean of all depths for each category
3) Compute the euclidian distance between each category, and the final score would be the average of the distances
```

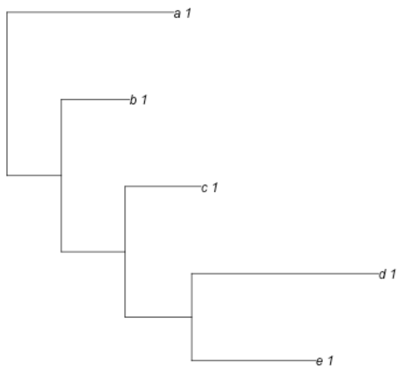


Figure 22. TreeVec: Tree 1

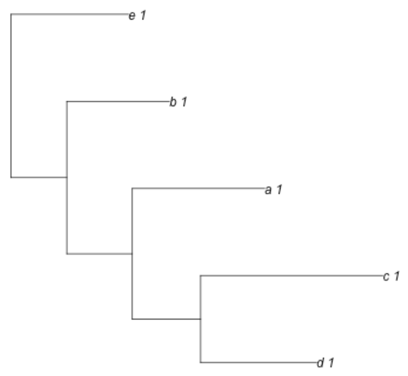


Figure 23. TreeVec: Tree 2

In this example, there are no examples in the subcategories. The minimum depth for each pair of the nodes is presented in the table below.

Pair	Tree1	Tree2
$a \to b$	0	1
$a \to c$	0	2
$a \to d$	0	2
$a \to e$	0	0
$b \to c$	1	1
$b \to d$	1	1
$b \to e$	1	0
$c \to d$	2	3
$c \to e$	2	0
$d \to e$	3	0

Table IV
EDGE LABELLING-RF METRIC

Thus, final distance between these two trees (eucleadian in this example), 4.89.

VII. DATASET

All of the metrics that we used required ground truth data to be available for comparison, which is impossible with real-world data; that's why synthetic data based on real-world data is generated.

The phylogeny of G7 and G12[15] dataset is based on ccRCCs, which are cancer cells present in the kidney, and displays intra-tumor heterogeneity. The data for testing was sequenced using exome-sequencing, which sequences all of the gene's protein-coding regions. In total, 679 nonsynonymous nucleotides (genetic) changes were observed in at least one region. A mutation was called present if nucleotide change was more than 5% at a given site in more than 1% reads. Also, sequencing at more sites might have led to extra subclones because there was a persistent increase in the number of detected mutations with each additional biopsy. Then the maximum parsimony algorithm from MEGA is used to construct the phylogeny tree.

To generate synthetic reads from a tree, a Markov model[16] is used. We assume that going from a root to any of its children is independent. A transition probability matrix is learned using the maximum likelihood algorithm such that each entry (i, j) gives the probability of jumping from ith branch to jth branch. For testing, pre-generated sequence data of G7 and G12 data from Clonephytester were used.

VIII. CONCLUSION AND FUTURE WORK

After studying the basics of phylogeny, a different state of the art algorithms, and techniques to evaluate the algorithms, I have gained enough confidence to explore new ideas in this field. There are two possible improvements over Clonefinder, and Mixphy, which is currently in work, aims to generate a perfect phylogeny tree rather than the maximum parsimonious tree (derived using Clonefinder). Also, while writing the report, an old work on small parsimony algorithms can help get site-level information from Clonevol and QuantClone.

REFERENCES

[1] M. Cusnir and L. Cavalcante, "Inter-tumor heterogeneity," *Hum Vaccin Immunother*, vol. 8, no. 8, pp. 1143–1145, Aug 2012.

[2] I. Hajirasouliha and B. J. Raphael, "Reconstructing mutational history in multiply sampled tumors using perfect phylogeny mixtures," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2014, pp. 354–367.

[3] T. Warnow, "Maximum parimony," https://tandy.cs.illinois.edu/MaximumParsimony-598.pdf.

[4] J. A. R. Elizabeth S. Allman, "The mathematics of phylogenetics," https://jarhodesuaf.github.io/PhyloBook.pdf.

[5] A. Hujdurovic, U. Kacar, M. Milanic, B. Ries, and A. I. Tomescu, "Complexity and Algorithms for Finding a Perfect Phylogeny from Mixed Tumor Samples," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 15, no. 1, pp. 96–108, 2018.

[6] S. Miura, K. Gomez, O. Murillo, L. A. Huuki, T. Vu, T. Buturla, and S. Kumar, "Predicting clone genotypes from tumor bulk sequencing of multiple samples," *Bioinformatics*, vol. 34, no. 23, pp. 4017–4026, 12 2018.

[7] S. R. dos Santos, "Clusterkde algorithm for clustering data based on kernel density estimation," in *MASCOT2018-15th MEETING ON APPLIED SCIENTIFIC COMPUTING AND TOOLS*, 2018.

[8] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[9] C. A. Miller, B. S. White, N. D. Dees, M. Griffith, J. S. Welch, O. L. Griffith, R. Vij, M. H. Tomasson, T. A. Graubert, M. J. Walter, M. J. Ellis, W. Schierding, J. F. DiPersio, T. J. Ley, E. R. Mardis, R. K. Wilson, and L. Ding, "SciClone: inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution," *PLoS Comput Biol*, vol. 10, no. 8, p. e1003665, Aug 2014.

[10] P. Deveau, L. Colmet Daage, D. Oldridge, V. Bernard, A. Bellini, M. Chicard, N. Clement, E. Lapouble, V. Combaret, A. Boland, V. Meyer, J. F. Deleuze, I. Janoueix-Lerosey, E. Barillot, O. Delattre, J. M. Maris, G. Schleiermacher, and V. Boeva, "QuantumClone: clonal assessment of functional mutations in cancer based on a genotype-aware method for clonal reconstruction," *Bioinformatics*, vol. 34, no. 11, pp. 1808–1816, 06 2018.

[11] H. X. Dang, B. S. White, S. M. Foltz, C. A. Miller, J. Luo, R. C. Fields, and C. A. Maher, "ClonEvol: clonal ordering and visualization in cancer sequencing," *Ann Oncol*, vol. 28, no. 12, pp. 3076–3082, Dec 2017.

[12] N. Karpov, S. Malikic, M. Rahman, S. C. Sahinalp *et al.*, "A multi-labeled tree dissimilarity measure for comparing clonal trees of tumor progression," 2019.

[13] D. Robinson and L. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1, pp. 131 – 147, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0025556481900432

[14] M. Kendall and C. Colijn, "Mapping phylogenetic trees to reveal distinct patterns of evolution," *Molecular Biology and Evolution*, vol. 33, no. 10, p. 27352743, Jun 2016. [Online]. Available: http://dx.doi.org/10.1093/molbev/msw124

[15] M. Gerlinger, S. Horswell, J. Larkin, A. J. Rowan, M. P. Salm, I. Varela, R. Fisher, N. McGranahan, N. Matthews, C. R. Santos, P. Martinez, B. Phillimore, S. Begum, A. Rabinowitz, B. Spencer-Dene, S. Gulati, P. A. Bates, G. Stamp, L. Pickering, M. Gore, D. L. Nicol, S. Hazell, P. A. Futreal, A. Stewart, and C. Swanton, "Genomic architecture and evolution of clear cell renal cell carcinomas defined by multiregion sequencing," *Nat Genet*, vol. 46, no. 3, pp. 225–233, Mar 2014.

[16] A. Rambaut and N. C. Grassly, "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees," *Comput Appl Biosci*, vol. 13, no. 3, pp. 235–238, Jun 1997.