Tree Encoding

Possible deadlines

1. RECOMB: October 30

TODO

1. Write the new definitions
2. Implement definition of previous point in the software
3. Write real case scenario
4. Define an **enumerate** algorithm

1 Introduction

2 Preliminaries

In this paper we use the standard graph theory terminology, following [2]. Moreover, given a vertex v we will use $N^+_G(v)$ to denote the set of vertices w such that in G there is an arc from v to w, and $N^-_G(v)$ to denote the set of vertices w such that in G there is an arc from w to v.

A tumor phylogeny, or simply *phylogeny* or *tree*, is a node-labeled tree such that no two nodes share the same label. Given a tree T, the set of its labels is called the set C of *characters*.

Let $G = \langle V_G, E_G, L_G \rangle$ be a node-labeled directed acyclic graph (shortly called *labeled dag*), with vertices V_G, arcs E_G, and some of its vertices — those belonging to the set $L_G \subseteq V_G$ — are called terminal. Moreover we will distinguish a node $r(G)$ of G, called its *root*, such that $r(G)$ is the only node of G with no incoming arcs.

Given a tree $T = \langle V_T, E_T \rangle$, we will say that G displays T if there exists a *mapping* $\phi : V_T \mapsto V_G$ from T to G such that:

1. the root of T is mapped to the root of G;
2. for each node $x \in V_T$, the label of x is also the label of $\phi(x)$
3. for each two vertices x and y of T such that x is the parent of y, then $(\phi(x), \phi(y))$ is an arc of G
4. each leaf of T is mapped to a terminal node of G, *i.e.*, a vertex in L_G.

With a slight abuse of language, we can say that ϕ also maps edges of T to arcs of G. We can now extend the notion of display to a set of trees, to obtain the notion of encoding, that is a weighted dag that displays all trees in the set in such a way that the weight of an arc is the maximum number of input edges that can be mapped to it.

Definition 1. Let $\mathcal{T} = \{T_1, \ldots, T_n\}$, be a set of trees that are labeled by the same set of characters C, and let $G = \langle V_G, E_G, L_G, w, l\rangle$ be a labeled dag, where $w : E \mapsto \mathbb{N}$ is the weight of each arc and $l : E \mapsto C$ is the character labeling each arc. Then G is an encoding for $\mathcal{T}$ if G displays all trees in $\mathcal{T}$ so that for each arc (x, y) of G there are exactly $w(x, y)$ mappings ϕ_i that have (x, y) in the image.

In the following we will only consider directed acyclic graphs that are node-labeled and edge-weighted. Moreover, we will assume that the order of mutation is consistent between the input trees, *i.e.*, if in one tree the mutation A is ancestor of mutation B, then in all the other tree A is an ancestor of B, or they are in different branches.

Definition 2 (Tree Encoding Problem)**.** The smallest tree encoding problem (TEP) asks, given a set $\mathcal{T} = \{T_1, \ldots, T_n\}$ labeled by the same set of labels, for a smallest — *i.e.*, with the fewest arcs — dag G that displays $\mathcal{T}$.

Given a dag G, the *decompressing* problem asks for a set $\mathcal{T}$ of trees such that G is an encoding for $\mathcal{T}$. Unfortunately, the corresponding decision problem is NP-hard and there can be more than one set of trees that are encoded by the same dag G.

Definition 3. Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of labeled trees and let G be a dag G. The decision version of the decompressing problem asks whether G is an encoding for $\mathcal{T}$.

Lemma 4. *The decompressing problem is NP-hard.*

But a relaxed version of the problem can be solved efficiently, by modifying the algorithms in [1, 3, 6].

Lemma 5. *Let T be a tree and let G be a dag G. Then we can compute in polynomial time whether G displays T.*

2.1 Building the encoding

In this section we will describe an algorithm to compute an encoding of a set of trees. The main idea of the algorithm is to incorporate iteratively a tree into the encoding, first by adding the tree to the encoding, then by merging some nodes. There are two fundamentally diverse possibilities to merge two nodes, corresponding to Definitions 6 and 7. The first one, called incorporation, essentially merges two nodes when the second node is connected to a subset of the nodes to which the first node is connected.

Definition 6. Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of labeled trees, let G be a dag, and let x, y be two nodes of G such that (1) x and y have the same label, and (2) $N^+_G(x) \subseteq N^+_G(y)$. If for each nonempty subset Z of $\subseteq N^+_G(y)$ there exists a tree T_z with a node z which has the same label as x and such that the children of z in T_z are exactly Z, then y *dominates* x. Morever, the incorporation of x into y results in the graph G_1 obtained from G by removing the vertex x and adding all arcs (z, y) with $z \in N^-_G(x) \setminus N^-_G(y)$.

The second class of merge is the **shrinking**, that allows to merge a node whose only child is an edge that connects it to a descendant of one of its duplicate.

Definition 7. Let G be an encoding, and let x, y be two nodes of G such that (1) x and y have the same label, (2) y has only an outgoing arc (y, z), (3) z is a descendant of x, (4) $N^-(x) = N^-(y)$, and (5) there is no path of G from x to y. Then the shrinking of y to x results in the encoding G_1 obtained from G by removing the vertex y and adding the arc (x, z).

Lemma 8. *Let G be an encoding, and let x, y be two nodes of G such that y dominates x. Let G_1 be the result of the incorporation of x into y. Then each tree displayed by G is also displayed by G_1.*

Proof. First of all notice that, since x and y have the same label, no tree displayed by G can have both x and y. Let T be any tree displayed by G that has the node y. Notice that $N^+_G(x) \subseteq N^+_G(x)$, hence the subtree of T rooted at y corresponds to a subtree rooted at x induced in G — just replace y with x. Moreover, let p be the parent of y in T. By construction, there is an arc (p, x) in G_1. Hnece, we can build a tree T_1 in G_1 by replacing the subtree rooted at y with the corresponding subtree rooted at x. □

Lemma 9. *Let G be an encoding, and let x, y be two nodes of G such that x can be shrunk to y. Let G_1 be the result of shrinking x into y. Then each tree displayed by G is also displayed by G_1.*

Proof. The proof is similar to that of Lemma 8. Again notice that, since x and y have the same label, no tree displayed by G can have both x and y. Let T be any tree displayed by G that has the node y. Moreover, let p be the parent of y in T. The edge (x, z) partitions the tree T into two trees T_1 and T_2. Since there is no path from x to y in G, for each vertex v on a path from x to z in G there is no path from x to y in G nor in G_1. Consequently, and also by construction of G_1, both T_1 and T_2 are trees displayed in G_1. Again, by construction, G_1 displays T, completing the proof. □

A consequence of Lemmas 8 and 9 is that Algorithm 1 computes an encoding, albeit not necessarily one with fewest arcs.

3 Real Case Scenario

One of the main motivations of this work is to provide an easy visualization method to biologist and medical doctors that allows them to compare different progression hypothesis of the same tumor created by either the same or different tools. The MegaTree is therefore a structure that contains all the inputs and display them in a succint way such that an evaluation of all the possible evolutions inferred can be done with an lower effort and in a small time by the experts, allowing them to make more accurate and informed decisions.

According to the goal we propose a possible scenario in which three different solutions are proposed by the same tool; in this particular case we used LICHeE [5] to infer three different progression, shown in Figure 3, for patient RMH004 of the clear cell renal cell carcinomas (ccRCC) sequencing study [4].

4 Visualization

Showcase of the software

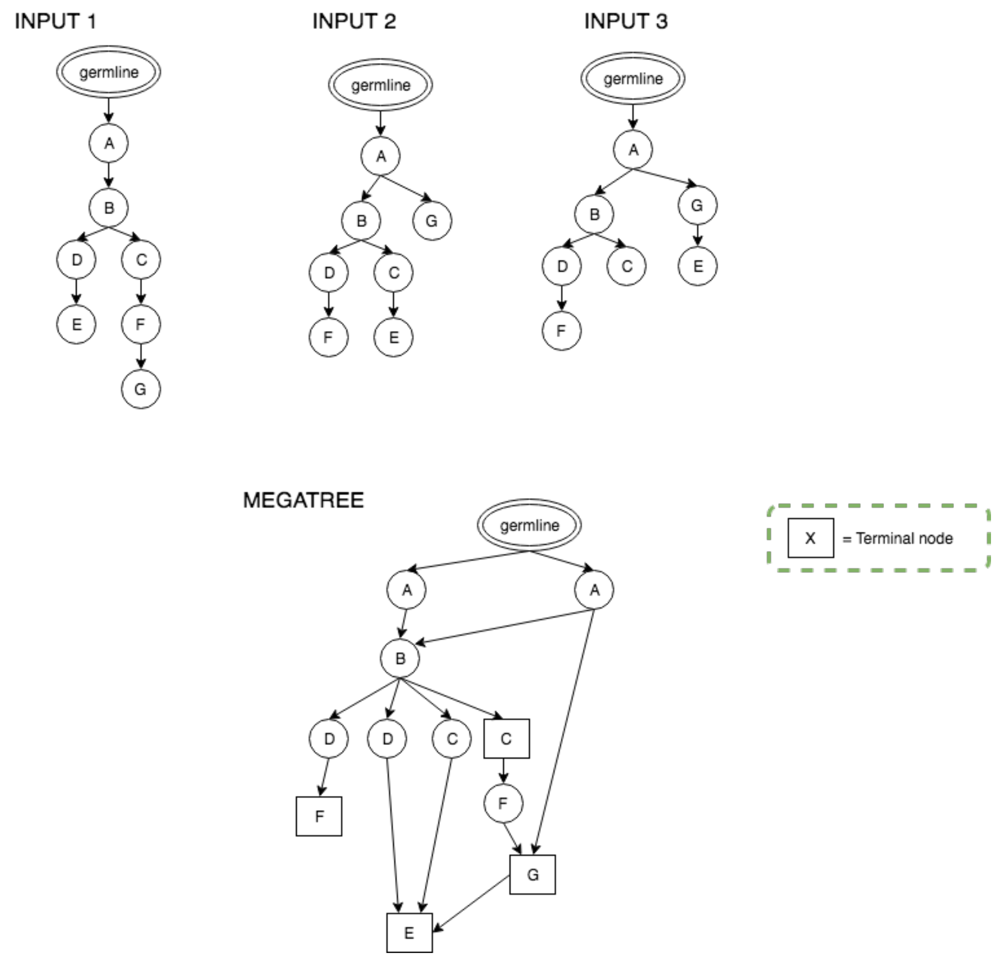


Figure 1: Sample instance and solution.

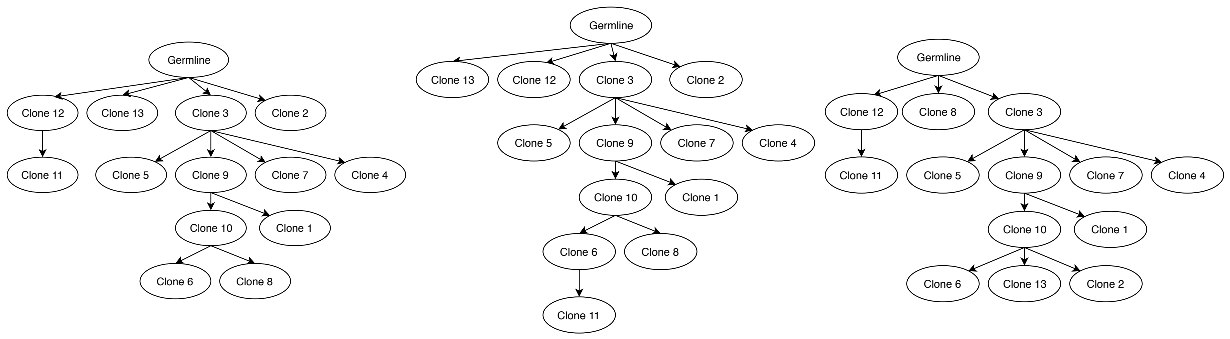


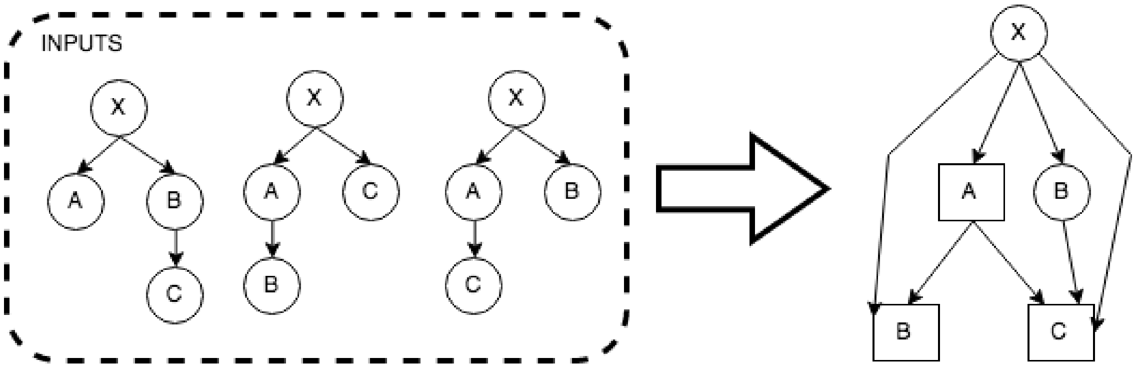
Figure 2: Three different cancer progression hypotheses inferred by LICHeE for ccRCC patient RMH004 of [4] sequencing study.

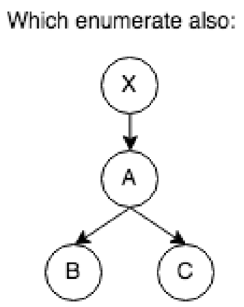
Algorithm 1: Build($\mathcal{T}$): compute an encoding for the set $\mathcal{T}$ of trees

```
   Input  : A set $\mathcal{T}$ of trees
   Output: An encoding $G$.
1  $G \leftarrow$ the graph with a distinguished node $\alpha$;
2  foreach $T \in \{T_1, \ldots, T_n\}$ do
3      $r(T) \leftarrow$ the root of $T$;
4      Add $T$ to $G$, as a disconnected subgraph, marking all leaves as terminal nodes;
5      Add the arc $(\alpha, r(T))$;
6      while there exist two nodes $v$, $w$ of $G$ such that $w$ dominates $x$ or $v$ can be shrunk to $x$ do
7          Find such nodes $v$, $w$ that are the furthest from $\alpha$;
8          if $w$ dominates $x$ then
9              $w$ incorporates $v$;
10         else
11             Shrink $v$ to $w$;
12 Set to 0 the weight of each arc of $G$;
13 foreach $T \in \{T_1, \ldots, T_n\}$ do
14     $\phi \leftarrow$ a mapping from $T$ to $G$;
15     foreach arc $e$ of $T$ do
16         $w(e) \leftarrow w(e) + 1$;
17 return $(G)$
```

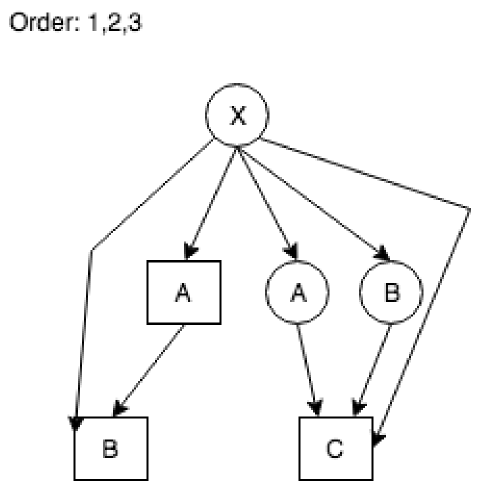
5 Figuring out terminal nodes (TO BE DELETED)

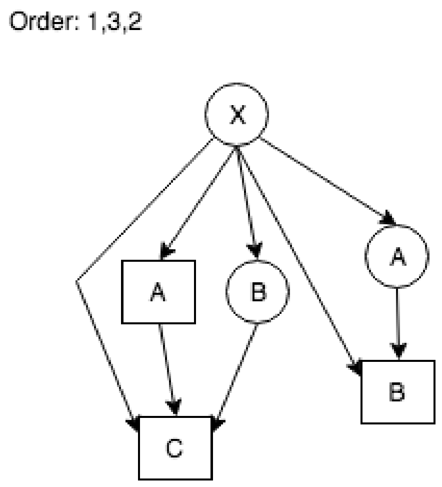
This is an example where terminal nodes are constructed as intended but they do create ambiguity:





When exactly a terminal node can have a children? An answer could be that they can only have one children, however the result of the BUILD change depending on the order of the inputs. The number of duplicated nodes stays the same, but the structure changes:



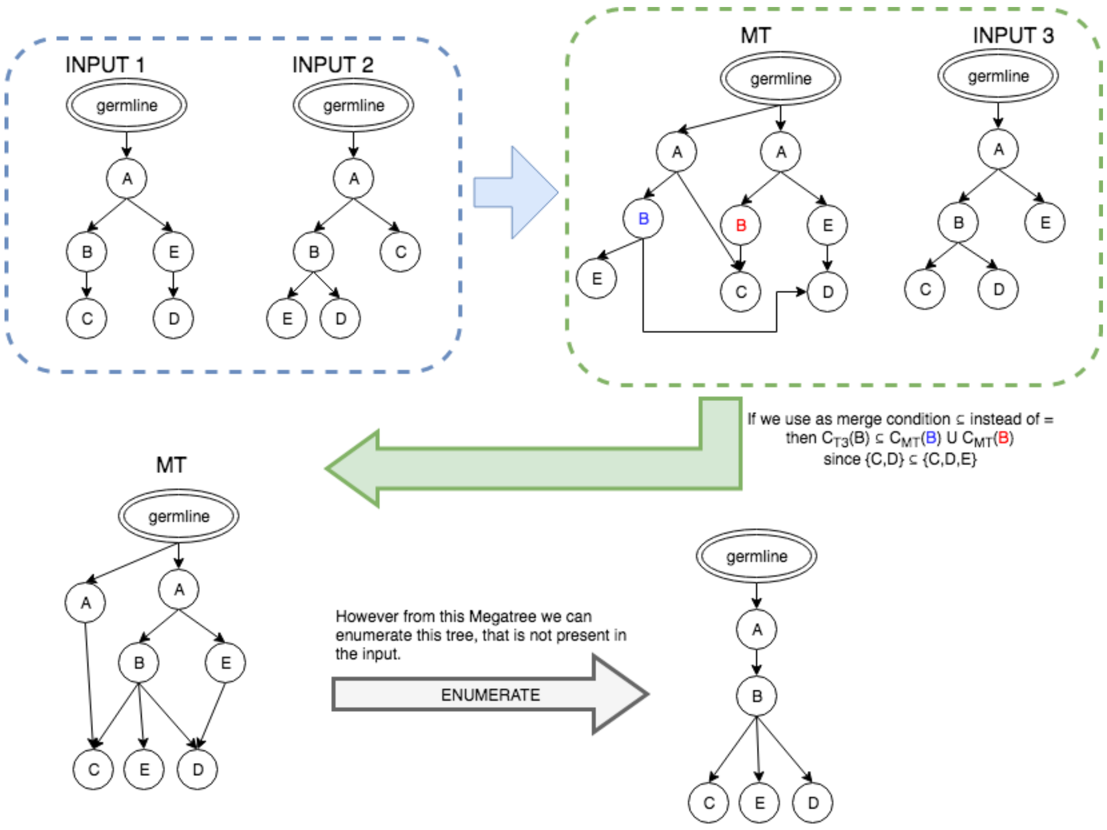


This shouldn't be an issue.

6 Not working 1 (TO BE DELETED)

NOTE: This section is identical to the previous version.

The following picture shows why the merge condition must be an equivalence and cannot be $\subseteq$:


INPUT 1
INPUT 2
germline
MT
INPUT 3
If we use as merge condition ⊆ instead of =
then $C_{T3}(B) \subseteq C_{MT}(B) \cup C_{MT}(B)$
since {C,D} ⊆ {C,D,E}
However from this Megatree we can enumerate this tree, that is not present in the input.
ENUMERATE

References

[1] M. Bordewich and C. Semple. Reticulation-Visible Networks. *arXiv:1508.05424 [cs, math]*, Aug. 2015. arXiv: 1508.05424.

[2] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

[3] P. Gambette, A. D. M. Gunawan, A. Labarre, S. Vialette, and L. Zhang. Locating a Tree in a Phylogenetic Network in Quadratic Time. In T. M. Przytycka, editor, *Research in Computational Molecular Biology*, number 9029 in Lecture Notes in Computer Science. Springer International Publishing, Apr. 2015.

[4] 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. *Nature Genetics*, 46(3):225–233, 2014.

[5] V. Popic, R. Salari, I. Hajirasouliha, D. Kashef-Haghighi, R. B. West, and S. Batzoglou. Fast and scalable inference of multi-sample cancer lineages. *Genome Biology*, 16:91, 2015.

[6] L. van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23), Nov. 2010.