Clonal Evolution Distances: Biologically Motivated Distances for Comparing Cancer Clonal Trees

by

Shohreh Golpaigani Fard

A thesis submitted to
Faculty of Graduate and Postdoctoral Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
December 2025

Thesis advisor
Olivier Tremblay-Savard

Author
Shohreh Golpaigani Fard

Clonal Evolution Distances: Biologically Motivated Distances for Comparing Cancer Clonal Trees

Abstract

Cancer evolution is driven by the accumulation of mutations in tumor cells, creating genetically diverse subclones. Clonal trees are useful for representing this evolutionary history, but existing distance methods often assume unique labels and no mutation loss—assumptions that rarely hold in real data from modern sequencing technologies.

In this thesis, we present the Clonal Evolution Distance (CED), a biologically motivated metric for comparing clonal trees that supports multi-labeled nodes, repeated mutations, and mutation loss, while following the principle that once a mutation is lost, it cannot be regained. CED is defined as the minimum total weight of deletion, rearrangement, and insertion operations required to transform one tree into another, where weights reflect the number of nodes affected. We show that computing exact CED is NP-hard.

To reduce complexity, we propose a semi-metric, δ_{CED}, that uses biologically informed path pairing and tie-breaking. Possible applications include comparing trees from different methods, datasets, or time points, and evaluating robustness with partial sequencing data.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor, Dr. Olivier Tremblay-Savard, for his guidance, support, and constant patience throughout this research. I am also grateful to the members of my thesis committee for their valuable and insightful feedback. I appreciate the members of the bioinformatics lab for their helpful discussions and support along the way.

Special thanks go to my husband, Omid, for always encouraging and supporting me to follow my dreams. I am deeply thankful to my three-year-old son, Arvid, for his mature understanding despite his young age.

Finally, I acknowledge the Department of Computer Science and the University of Manitoba for providing the resources and environment necessary to carry out this research.

Chapter 1

Introduction

Despite decades of progress in treatment development, cancer is a disease responsible for many deaths worldwide. Cancer patients' genes are studied to understand the genetic changes that cause cancer. These studies have shown that tumors are composed of a diverse population of cells called subclones, with each subclone carrying its own set of mutations (changes in DNA) that evolve within the tumor microenvironment. The microenvironment refers to the complex ecosystem surrounding cancer cells, including components such as immune cells, and blood vessels.

A single cell with a set of mutations that give it a survival advantage can start cancer, and it develops over the lifetime of a patient. Over time, cancer cells acquire more mutations, which may give their population either an advantage or a disadvantage. This evolutionary process is much more rapid and complex than the evolution of species across generations [Greaves and Maley, 2012]. Cancer evolution happens at the level of the tumor cell population, while the evolution of species occurs across large populations. This evolutionary process of mutation acquisition can be represented by a tree called clonal tree.

A clonal tree is a rooted tree in which nodes represent the subclones and are labelled by the sets of mutations acquired for the first time in the subclone, and edges represent

mutation acquisition. Labels can be represented using letters, numbers or a combination of both. The root represents the healthy stage of the tumor. Nodes closer to the root represent clones with mutations acquired in the earlier stages of the tumor, while nodes farther from the root represent subclones that acquired mutations later. A special case of clonal trees, where each node is labeled with exactly one label, is called a mutation tree.

Clonal trees are strong tools to represent the evolutionary status of a tumor, because they can depict the accumulation of mutations over time, and visualize the diversity within the tumor. The structure of trees implies a temporal order of mutation acquisition. Furthermore, they enable tumor evolution comparison.

Tree comparison is required because there are some cases where more than one clonal tree is available for a single tumor. One circumstance where this can occur is when some tree construction methods return multiple equally optimal trees. This happens when the computational cost of handling complex and noisy data prevents these methods from identifying a unique optimal tree. Another case arises when data from different sequencing techniques—techniques for reading DNA—are used. Finally, applying different tree construction methods to the same dataset may also result in different trees.

Comparing these trees across patients or time points helps with understanding tumor heterogeneity. For instance, comparing trees constructed from different regions of a tumor in a single patient can reveal whether the key (early) mutations are shared across regions or are region-specific. Note that subclones are not equivalent to tumor regions. A subclone is a group of genetically similar cells, while regions are anatomical locations. Multiple subclones can coexist in one region, or a subclone may extend across multiple regions. One can build a tree for each region or aggregate data from all regions to build a single tree. Comparing trees across regions is how spatial patterns and region-specific subclones are detected. Furthermore, by comparing trees generated by new tree construction methods with known and

trusted ones, we can determine whether new methods can closely represent the true evolutionary history. We can also compare trees constructed from subsets of the same dataset to assess the sensitivity of tree construction methods to the quality and size of data.

Normal tree comparison methods, such as the Robinson-Foulds (RF) distance [Robinson and Foulds, 1981], assume that only leaves are labeled and that trees are uniquely labeled with a single label per node—an assumption that rarely holds for clonal trees. In clonal trees, a node may carry multiple labels, reflecting uncertainty about the order in which mutations were acquired. Moreover, the same mutation may appear in different lineages of the tree due to independent acquisition in separate subclones.

Traditional methods also assume that if a mutation appears in the history content (i.e., in the label set of an ancestor node), it must appear in the history content of all of its descendant nodes. However, real cancer evolution often involves mutation loss, where this assumption breaks down. Mutation loss plays a crucial role in accurately characterizing tumor evolution, as certain treatments or the immune system may target and eliminate cells carrying specific mutations. [Jiang et al., 2022] As a result, descendant subclones may no longer inherit those mutations.

Another reason normal tree comparison methods do not work well for clonal trees is that they assume both trees have the same set of labels. However, clonal trees generated by different tree construction methods, trees from different patients, or even trees representing the same tumor at different time points may contain varying label sets [Jahn et al., 2021]. These characteristics of clonal trees make it practically impossible to directly apply traditional tree comparison methods to them.

To overcome these challenges, researchers have modified the definitions of some existing tree comparison methods to make them applicable to clonal trees. However, only a few of these methods can fully handle all the properties of clonal trees. Most of them still impose

some restrictions that limit their applicability to more realistic evolutionary scenarios.

To be more specific, most of the tree comparison methods for clonal trees are generalizations of traditional tree comparison methods applied to multilabeled trees, but they still do not allow a label to appear more than once, and they do not permit mutation loss. Even the methods that impose no restrictions do not always capture the full picture of how cancers evolve, because they often blindly generalize traditional tree comparison methods without considering biological relevance.

For instance, some methods only check if the ancestor-descendant relation between two mutations is the same in two trees [Karpov et al., 2019]. In Chapter 3 we explain in detail how these methods work and why they cannot fully represent the dissimilarities between two clonal trees.

Due to technical limitations and computational costs among other reasons, earlier tree construction methods often simplified tumor evolution. For instance, they assumed each mutation happens only once and ignored mutation loss. This constraint is known as the Infinite Sites Assumption (ISA). See Figure 2.1 for an example of a clonal tree generated under ISA, and Figure 2.2 for one that does not satisfy it. In Figure 2.2, the loss of mutation D is represented as "D". The ISA assumptions made building the evolutionary trees easier. It also justified applying similar restrictions to the early distance methods.

However, tree construction methods have evolved to capture a more realistic picture of tumor evolution. Methods such as SiFit allow violations of the Infinite Sites Assumption [Zafar et al., 2017]. This makes distance methods defined under ISA either ineffective or reliant on a preprocessing step that removes repeated labels and mutation losses before the method can be applied [Karpov et al., 2019]. Development of higher resolution sequencing techniques such as single cell sequencing and computationally efficient algorithms has helped researchers to build more detailed and realistic tree construction models. Consequently,

distance methods must also evolve to remain effective on these more complex trees.

We introduce the Clonal Evolution Distance (CED), a novel metric designed to quantify the differences between clonal trees while respecting the hierarchical nature of clonal evolution. Our method incorporates three key operations: deletion (deletes a label, which may result in node deletion if the set of labels of that node becomes empty), rearrangement (moves a label from its current node to another existing or new node), and insertion (inserts a label into an existing or new node). Similar to the edit distance [Bille, 2005], node deletion results in reattaching the child nodes to the parent of the deleted node. When inserting a new child node under an existing node, the parent's children can be redistributed between the parent and the new child. These different operations are described in depth in Chapter 4.

Each of these operations is weighted to reflect its biological significance in tumor progression. To be more specific, the weight of an operation corresponds to the number of nodes whose set of labels or history content changes as a result of applying the operation. Details are illustrated with examples in Chapter 4.

The CED is then defined as the minimum weight among all possible sequences of operations that transform one tree into another. We impose no restrictions on the number of repeated labels or mutation loss; however, a label that is lost on a lineage is never acquired again on that same lineage. This assumption is known as Dollo's law of irreversibility [Gould, 1970].

We prove that computing the CED's value, d_{CED}, is NP-hard and introduce a semi-metric variant, δ_{CED}, which has fixed-parameter tractable (FPT) algorithm in the number of pairwise consistent tree partitions (PCTP)—pairwise tree partitions which are consistent with regional similarities between tumors—as defined in 4. This semi-metric uses a set of operations similar to those of d_{CED}, but incorporates a path-mapping technique.

In Chapter 2, we cover all the background information needed to understand the project.

In Chapter 3, we discuss existing distance methods, along with their strengths and limitations. In Chapter 4 Section 4.1, we define the CED operations and present some examples. In Section 4.2 we formally define the CED distance method and prove that it is a metric. We also show in this section that finding d_{CED} is NP-hard. In Section 4.3, we determine the largest possible distance between two trees under specific restrictions, since without such restrictions, this value is infinite. In Section 4.4 we define δ_{CED} and show that it is a semi-metric. In Section 4.5, we introduce an algorithm to compute δ_{CED}. In Section 4.6, we show that the proposed algorithm runs in polynomial time with respect to all parameters except one. Finally, in Section 4.7, we present several proof-of-concept examples to illustrate the key ideas. In Chapter 5, we provide a summary of the project, discuss its limitations, and highlight aspects that can be explored further in the future.

Chapter 2

Background

Understanding cancer evolution requires a solid foundation in biological concepts such as DNA, genes, and mutations, as well as the computational methods used to study their evolution. In this section, we cover all the necessary background.

2.1 Fundamental Concepts of Molecular Genetics

Deoxyribonucleic acid, or DNA, is a molecule that acts as the blueprint of the cell, since it contains the information needed to construct other components of the cell. DNA is composed of two strands, where each of them is made up of simpler units called nucleotides. Nucleotides contain one of four bases: adenine (A), thymine (T), cytosine (C), or guanine (G). The two strands are held together by hydrogen bonds between complementary base pairs (A-T and C-G) and form a double-helix structure. [Watson and Crick, 1953]

The majority of cellular processes are carried out by proteins, which are sequences of amino acids. The specific sequence of amino acids determines a protein's structure and function [Havey, 2017]. Proteins are synthesized based on the information encoded in DNA through two main steps: transcription, where a segment of DNA called gene containing

functional and reproductive information [Havey, 2017] is used as a template to produce a single-stranded molecule called messenger RNA (mRNA) in the nucleus of the cell; and translation, where this mRNA moves to the cytoplasm, the gel-like substance filling the interior of the cell, and attaches to ribosomes, small cellular machines that assemble amino acids into protein chains using the mRNA template.

The long structure into which DNA is organized is called a chromosome, and the complete set of chromosomes in a cell is called the genome of that cell. During cell division, the cell replicates its genome so that each daughter cell receives the same genetic material. The duplication involves separating the two strands, and producing new complementary strands.

Cell division normally ensures the safe transfer of genetic information from one generation to the next. However, sometimes the genome is altered during replication. These changes to the DNA are called mutations. Cells are equipped with mechanisms to repair damage to DNA, such as mismatches, breaks, or chemical modifications, to maintain genomic stability and prevent mutations [Wood, 1999].

2.2 DNA Mutation Types, Their Repair, and Immune Response

DNA mismatches occur when incorrect nucleotides are paired during DNA replication. There are key proteins that identify these mismatches, remove the newly synthesized strand containing the incorrect base, and replace it by the correct base [Iyer et al., 2006].

DNA breaks are cuts in the DNA backbone, which can occur in one or both strands. Single-strand breaks are identified by specialized proteins; the damaged part is removed and filled in with the correct sequence. Double-strand breaks are more dangerous, as they can lead to chromosome loss or rearrangement. Cells have two different mechanisms to repair

double-strand breaks.

The first mechanism, homologous recombination (HR) occurs when specific proteins use an undamaged DNA sequence as a template to fix the break. This mechanism is usually accurate and safe, but it can only be used during certain phases of the cell's life cycle. The second mechanism is called non-homologous end joining (NHEJ). In this process, some proteins find the broken ends, others may trim the ends, and finally, the ends are attached. This process is faster, but it may result in small missing or extra nucleotides in the DNA. [Mao et al., 2008]

Finally, chemical modifications refer to the process where small chemical groups are added to DNA bases. This can alter how DNA functions. Special enzymes find these chemical groups and remove the damaged base, leaving a gap. Other enzymes then trim the ends of the gap if needed and fill in the gap with the correct sequence. [Chatterjee and Walker, 2017]

When DNA repair mechanisms fail to fix certain critical DNA damage, damaged DNA can activate molecular events called signaling pathways that alert the immune system. This immune response can lead to programmed cell death which helps to remove damaged cells. This provides an additional layer of protection beyond DNA repair mechanisms. [Nakad and Schumacher, 2016] This cell death response mainly happens through a process called apoptosis, which is a controlled way for cells to die without harming the surrounding tissue. During apoptosis, the cell carefully breaks down its internal parts, including its DNA, and then signals nearby cells to remove the remains safely. Important proteins like *p53* help decide when the damage is too severe and trigger this process. When DNA damage is detected, *p53* can pause the cell cycle to allow repair or initiate apoptosis if the damage is irreparable. This prevents damaged cells from dividing and passing on mutations. Alternatively, cells may undergo other forms of regulated death such as necroptosis, pyroptosis, or frerroptosis, which typically happens when apoptosis is blocked, during an infection, or in response to a

specific type of damage [Bertheloot et al., 2021; Tang et al., 2021].

2.3 Cancer Initiation and Tumor Heterogeneity

Unfortunately, some mutations may allow cells to evade being detected by the immune system, change their metabolism, and induce new blood vessel growth [National Cancer Institute, 2025]. Mutations that affect genes that control cell growth and division can lead to uncontrolled cell growth, resulting in cancer [Franjic, 2021]. These genes are called key drivers and their mutations are called mutation drivers [Bailey et al., 2018; Peters et al., 2024]. Mutations may be inherited or acquired later in life due to aging or exposure to harmful factors such as cigarette smoking.

A single cell with a set of mutations initiates cancer. Then, selective pressure within the tumor microenvironment leads to a Darwinian process of cancer evolution [McGranahan and Swanton, 2017], which means that if these mutations give the cell a better chance to grow and divide, the cell replicates more rapidly than normal and form a clone of cells sharing the same set of mutations. Over time, each of the cells in that clone may acquire additional mutations, creating new subclones. The subclones with fitness advantages expand, which results in complex evolutionary trajectories. Cancer cells evade normal cell death mechanisms, they may invade surrounding tissues, and even spread to other organs [National Cancer Institute, 2025].

The heterogeneity both between tumors and within a tumor leads to unique characteristics such as the potential to invade other organs (metastatic potential), gene expression, structure, and drug resistance. These complex and diverse characteristics make cancer treatment challenging.

2.4 Cancer Treatment

Cancer treatment has evolved rapidly. Nowadays, a wide range of therapies, including surgery, radiation, chemotherapy, immunotherapy, targeted therapy, hormone therapy, CAR-T cell therapy and cancer vaccines are offered. Each method has distinct mechanisms and applications and has its own benefits and limitations. Below is an overview of some traditional and modern cancer treatments.

Surgery is an effective treatment for solid tumors which are accessible and isolated. The process involves surgically removing the tumor and surrounding tissue. Advances in robotics and fluorescence imaging has increased precision and reduced the risk of incomplete tumor removal and hurting healthy tissues. However, when the tumor is metastatic or near to vital organs, surgery is not effective. [Zafar et al., 2025]

Chemotherapy involves using drugs to disrupt cell replication, leading to cell death. Unfortunately, these drugs cannot distinguish cancer cells from healthy ones. This often leads to causing damage to normal tissues. Furthermore, this method may cause side effects and drug resistance. The immune system may be activated as a side effect of the drug which can slow the drug's effectiveness. [Zafar et al., 2025]

Radiation therapy uses high-energy particles to destroy cancer cells by damaging their DNA. Similar to chemotherapy, this method may trigger side effects and therapy resistance. Despite advances in technologies, such as allowing radiation to beam according to the shape of the tumor or delivering the required dose of radiation in microseconds, the healthy cells surrounding the tumor are still jeopardized. [Zafar et al., 2025]

Hormone therapy is limited to hormone-dependent tumors such as breast cancer. The level of hormones associated with tumor growth—like estrogen in the case of breast cancer—is controlled by this therapy. This method can also induce resistance over time. [Zafar et al., 2025]

So far, we have discussed traditional categories of treatments. Next, we explore some modern options.

In targeted drug therapy, drugs designed to target molecules or genetic mutations driving the cancer are used. Healthy cells with targeted properties may be eliminated as a side effect of the drug. New mutations in cancer cells may also help them escape and create drug resistance. [Liu et al., 2024]

Antibody-based therapies use drugs to block proteins that stop the immune system from attacking cancer cells. In the long term, this method may increase the risk of immune-related side effects. These side effects are generally predictable because they are known from clinical trials, and patients are usually warned about them in advance. Other adverse effects, which are harmful, unintended, sometimes unexpected, and may be more serious requiring intervention or stopping the medication, are also possible. [Liu et al., 2024]

Gene therapy alters genetic material within cancer or immune cells to attack tumors effectively or overcome drug resistance. It is challenging to precisely alter the cancer cells, and targeting healthy cells may cause unintended mutations. Furthermore, the immune system may react to this method. [Liu et al., 2024]

For additional modern treatment options, see [Liu et al., 2024].

2.5 Sequencing Techniques

Despite advancements in cancer treatment methods, all of them share a common requirement: a clear understanding of the unique features of each tumor and a personalized profile of the patient. DNA sequencing, which is the process of determining the order of the four nucleotide bases that make up a DNA molecule [National Cancer Institute], plays a crucial role in constructing profiles for tumors. It provides fundamental genetic information about

mutations, large-scale structural changes such as when segments of DNA are missing or rearranged, and other genomic alterations, including increases or decreases in the number of copies of particular genes or DNA segments, all of which contribute to tumor development and progression [Sekar et al., 2014]. This direct access to the genome of individual cells, allows identifying differences (mutations) in an individual's DNA sequence compared to a reference genome. There are two sequencing approaches, bulk sequencing and single cell sequencing.

The sample in bulk sequencing is a mixture of cancerous and healthy cells, and the result is represented as an aggregate sequence. This sequence can be considered an average of the DNA of all cells in the sample, in the sense that rare clones appear in a small portion and are not easy to detect, while more common or repeated mutations are highlighted by the sequencing. This method provides an overview of genetic mutations across millions of cells efficiently, but tumor heterogeneity is sacrificed. [Lim et al., 2020] It is hard to infer the lineage of tumor mutations from analysing the aggregate sequence because the sequences of different subclones are merged [Zafar et al., 2017].

In the single-cell approach, on the other hand, individual cells are isolated and sequenced, so the rare clones hidden from the bulk approach are identified. By sampling both tumor and surrounding cells, this method provides a detailed view of the tumor microenvironment and the evolutionary lineage of mutations. Single-cell sequencing is more challenging and computationally intensive, but recent advancements in technology has reduced its cost remarkably. [Lim et al., 2020]

Single-cell sequencing has uncovered new layers of tumor heterogeneity. Findings from these advanced sequencing techniques demonstrate that tumor evolution often violates the Infinite Sites Assumption (ISA)[Zafar et al., 2017]. Real tumor evolution is more complex, with recurrent mutations, mutation losses, and convergent evolutionary events that violate

the conditions required for the ISA to hold[Zafar et al., 2017]. Phylogenetic trees are powerful tools for representing and analyzing sequencing data. Recent tree construction methods are designed to capture the complexities revealed by single-cell sequencing[Zafar et al., 2017].

2.6 Tree Construction Methods

Several computational methods have been developed to infer tumor evolutionary trees from sequencing data. These trees have many applications, such as finding the history of migration of metastatic cells [Kumar et al., 2020]. Recent studies have shown that, despite previous beliefs, metastatic cells are not the result of migration from a single cell or clone. In fact, they result from complex cell migration involving different clones [El-Kebir et al., 2018], and the history of migration of these cells can be traced using tumor evolutionary trees.

Figure 2.1 shows a tumor evolutionary tree. Each label A, B, C, D, E, F represents a mutation. The label A represents the mutation which is present in every cell of the tumor in its initial stage. Each edge represents the acquisition of one or more mutations that label the child (end-point) node it enters.

The infinite sites assumption (ISA) is a standard assumption which is made in many articles to simplify the calculations [Ciccolella et al., 2021b]. The name derives from the idea that if there were an infinite number of sites (positions) in the genome where mutations could occur, it would be highly unlikely for the same site to mutate more than once. In other words, every mutation would be unique and persist throughout tumor evolution without recurrence or reversal. However, the ISA is usually violated in real tumor evolution [Li et al., 2024]. Under the ISA model, the node labeled B in fact represents a subclone of cells in which the mutations B and A are present, because once mutations are acquired, they are never

lost. Similarly, all descendant nodes inherit the mutations acquired by their ancestor nodes, where ancestors of a descendant node like n are defined as the nodes on a path between the root and n.

Figure 2.2 represents a more realistic scenario. In this tree, mutation loss is represented by "", and the mutation B is gained in two different lineages. These properties are allowed by the finite-sites model. There are tree construction methods that infer tumor trees from single-cell sequencing data, which either follow the finite-sites model or allow trees to violate the ISA. In contrast, most bulk sequencing–based methods are under the ISA except a few of them. In what follows, we focus on the approaches that allow ISA violations.

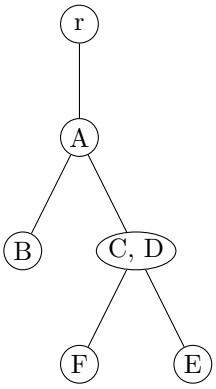


Figure 2.1: Clonal tree with ISA

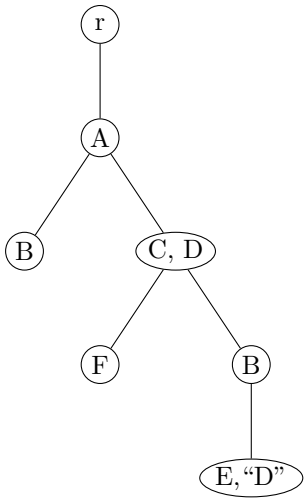


Figure 2.2: Clonal tree without ISA

SiFit [Zafar et al., 2017] operates on an input matrix where rows represent mutational sites. The matrix can be binary (indicating mutation presence or absence) or ternary (when the data for both chromosomes at each site is not the same: 0 means no mutation, 1 means a mutation in one copy of the chromosomes, and 2 means a mutation in both copies). This matrix is derived from the single-cell sequencing data, and false positive and false negative rates can also be obtained from it. SiFit employs a finite-sites probabilistic model of tu-

mor evolution, along with an error-rate model for single-cell sequencing, which incorporates statistical uncertainty and sequencing errors. It uses a heuristic search algorithm to find the phylogenetic tree that maximizes the likelihood of observing the data given the model parameters. The results of testing SiFit on a set of simulated data indicated that it outperformed the methods that infer tumor trees under the ISA, in terms of accuracy and ability to estimate error rates. [Zafar et al., 2017]

SiCloneFit [Zafar et al., 2019] is another tree construction model based on a finite-site model of evolution. SiCloneFit employs Markov chain Monte Carlo (MCMC) sampling, which explores the range of possible clonal trees by proposing iterative updates to clone assignments and tree topologies. These updates—such as adding or removing branches or rearranging tree structures—are accepted or rejected probabilistically based on how well they explain the observed data within a Bayesian framework. By iteratively performing these moves, the MCMC algorithm navigates the complex space of possible evolutionary trees and clone partitions. Ultimately, it converges to results that reflect the most plausible clonal trees.

SPhyR [El-Kebir, 2018] employs more restrictions. It assumes that each mutation is acquired once and lost at most k times (across different lineages), following the k-Dollo model. It begins with a binary $m \times n$ matrix, where each entry (i, j) represents the presence or absence of the jth mutation in the ith cell. The goal of SPhyR is to build a tree where each node represents a clone, labeled with a binary n-tuple indicating the status of the n mutations. Each leaf of the tree corresponds to a single cell. It uses the false negative and positive rates to correct real sequencing data errors. Then by integer linear programming searches for an assignment of states to matrix entries that both fit the observed (and possibly error-corrected) data and minimize the total number of losses (parsimony). Solution of this ILP yields a valid matrix consistent with k-Dollo evolutionary constraints. Once the matrix

is valid, combinatorial algorithms are used to reconstruct the tree. Mutations are assigned to edges. SPhyR outperforms SiFit, both in terms of accuracy and runtime.

Another tree construction model which employes the k-Dollo model is GRMT, generative reconstruction of mutation tree [Yu et al., 2021]. This method starts building the tree from the root and adds mutations in each iteration until all mutations are placed on the tree. At each step, it adds the node that fits best. Constructing the tree from scratch instead of exploring all possible trees allows GRMT to handle large datasets efficiently.

These tree construction methods do not consider evidence from sequencing data to confirm mutation loss, which may affect their accuracy. SCARLET [Satas et al., 2020] allows mutation loss, but only if it is supported by sequencing data evidence of a corresponding deletion. It refines a tree called a copy-number tree, which represents structural changes in the genome. These changes refer to cases where entire segments of DNA are duplicated or deleted compared to a reference genome. The copy-number tree is constructed by other established methods. The refined tree is a clonal tree whose losses are supported by the copy-number tree. The algorithm works recursively from the leaves (observed cells) up to the root. It infers mutation profiles for internal nodes by solving a phylogeny problem that models mutation presence and absence while respecting the constraints of unique gains and supported losses.

ConDoR [Sashittal et al., 2023] is another tree construction method which employs copy-number aberrations (CNAs). It defines a novel evolutionary model called the constrained k-Dollo model, which allows mutations to be gained once but lost up to k times, with the key constraint that losses can only occur between cells belonging to different copy-number clusters. It takes as input the clustering of cells based on their copy-number profiles, along with mutation variant read counts and the total number of DNA sequencing reads covering each mutation site. It searches for the optimal mutation matrix and tree structure

maximizing the likelihood of observed read counts under the constrained k-Dollo model.

Bonizzoni et al. [Bonizzoni et al., 2017] propose a bulk sequencing-based method which formulates tumor evolutionary history reconstruction as an Integer Linear Programming (ILP) problem. It allows mutations to be gained and lost a limited number of times. It takes as input mutation frequency data from multiple tumor samples, and aims to find a phylogenetic tree and clonal composition that best explain this data. The key idea is to construct extended matrices representing possible mutation acquisition and loss states and encode compatibility constraints as linear inequalities. The tumor evolutionary tree is then constructed by solving the ILP problem.

The method PhISCS [Malikic et al., 2019] addresses the tumor clonal tree construction problem by integrating single-cell and bulk sequencing data. At first, it uses the ISA, but then the method is generalized to allow limited number of losses. It formulates the problem as a combinatorial optimization task and uses integer linear programming (ILP) and constraint satisfaction problem (CSP) approaches to efficiently solve it. The input is a ternary matrix representing mutation presence, absence, or missing data in single cells. Additional constraints ensure that the inferred phylogenetic tree and clonal composition assign mutations to clones in such a way that the resulting expected mutation frequencies match the observed proportion of cells carrying each mutation in the bulk tumor sample.

The characteristic that traditional evolutionary clonal or mutation tree construction methods have in common with the recent methods is that they are not deterministic. Instead, they use heuristic or probabilistic approaches as described in this section. Hence, distance or dissimilarity measures are essential for evaluating and comparing the resulting trees.

As the tree construction methods for mutations are evolving to allow more realistic evolution scenarios such as violating the ISA, the distance methods should be able to compare

these trees.

Chapter 3

Literature Review

In this chapter, we review the existing distance and dissimilarity measures for comparing clonal trees in cancer evolution studies. We discuss how traditional approaches and recent advances accommodate different evolutionary assumptions and highlight the strengths and limitations of each method.

3.1 General Tree Comparison Measures for Leaf-Labeled Trees

The famous Robinson-Foulds (RF) distance [Robinson and Foulds, 1981] for leaf-labeled trees (phylogenetic trees) counts the number of edge contractions and expansions required to transform one tree into another. An alternative formulation defines the distance as the number of nodes whose descendant label sets are unique (no node on the other tree has the same set of descendant labels) [Llabrés et al., 2021].

Generalized Robinson-Foulds (GRF) distance [Llabrés et al., 2021] for phylogenetic trees is designed to address the low resolution of RF distance, where small and large differences

in label sets had a similar impact on the distance. Instead of treating set labels as equal or different, GRF measures how different each set label of subtrees in one tree is to the most similar set label of subtrees in the other tree. It averages these minimal differences to quantify overall tree dissimilarity.

A widely used class of tree comparison methods is operation-based distances, which quantify the minimum number of specific tree modification operations required to transform one tree into another. A well-known example of these distances methods is the Nearest Neighbor Interchange (NNI) [Allen and Steel, 2001]. It measures the minimum number of local subtree rearrangements needed to transform one tree into another. Due to the small range of the movements, reasonably similar trees may have a large distance. The Subtree Prune and Regraft (SPR) distance is a more flexible version of the same distance which allows cutting a subtree and pasting the cutted tip somewhere else on the remaining tree. This movement does not have to be local, but calculating the distance is computationally more expensive. The Tree Bisection and Reconnection (TBR) distance is even more flexible, since after cutting a subtree, it allows the regrafting to be between any two edges. Finding TBR distance is NP-hard. [Allen and Steel, 2001]

3.2 Distance Methods For Clonal Trees Under ISA

The Common Ancestor Set distance (CASet) compares the ancestor-descendant relationships between clonal trees [DiNardo et al., 2020]. For each pair of nodes in each tree, the set of common history mutations shared by those nodes is collected, and the distance value is the average Jaccard distance between all these sets. Nodes near the root of the tree can contribute to larger differences if labeled differently, since these labels (mutations) are shared by many descendants.

The same authors proposed another distance measure, the Distinctly Inherited Set Comparison distance (DISC). It focuses on the sets of mutations that are distinctly inherited in different lineages [DiNardo et al., 2020], and like CASet, it uses the Jaccard distance. None of these distance measures is a metric. Their author suggests that the CASet is more useful for the trees which are very different, and the DISC should be used to distinguish between the trees that are mostly similar. However, they can not achieve both purposes at the same time. These two distances can be computed in polynomial time in the number of labels, when they are applied to trees that have the exact same set of labels.

The multi-labeled tree dissimilarity measure (MLTD) [Karpov et al., 2019] uses an edit distance based method to transform two multi-labeled trees into a common tree [Karpov et al., 2019]. The operations can delete labels, delete unlabeled leaves (nodes with no children), and insert nodes with in-degree and out-degree of one as new parents of an existing node v, where the labels of v are redistributed between v and its newly inserted parent. Then the distance value is the number of the labels in each tree minus twice the number of the labels in the maximum common tree (the common tree with most number of labels).

The MLTD is not a metric since it does not satisfy the triangle inequality, and moreover, it does not distinguish between a tree with one node labelled with A, B and a tree with two nodes, one labelled with A and one labelled with B. This means the measure does not differentiate between tree construction methods that determine the order of mutation acquisition and those that leave it undecided by multi-labeling nodes. It is computed in polynomial time.

There are other distance methods for clonal trees that are specifically designed for trees that have the exact same set of mutation labels. For example, the path distance measures the total difference in the path length between each pair of mutations. The Parent-Child distance (PCD) counts the pairs of ordered parent and child mutations—the first mutation

labels a parent node and the second labels its child—that are induced from one tree but not from the other. The Ancestor-Descendant (AD) distance generalizes this by considering all ancestor-descendant mutation pairs instead of only direct parent-child relationships. The Clonal distance (CD) on the other hand, measures dissimilarity by comparing individual nodes and counting those whose set of labels does not exactly match the labels of any node in the other tree. [Govek et al., 2018]

The famous Robinson-Foulds (RF) distance [Robinson and Foulds, 1981], which was originally developed for phylogenetic trees, has various generalizations for the mutation trees [Jahn et al., 2021; Khayatian et al., 2024] . For instance, the Bourque distance is defined on clonal trees [Jahn et al., 2021]. If the trees being compared have the exact same set of labels, the Bourque distance reduces to the standard RF distance for labeled trees. Each edge defines a partition of labels: one part includes the child node and its descendants, and the other includes all remaining labels. The total number of partitions in the two trees considering shared and non-shared labels is calculated by counting bipartitions that appear uniquely in each tree, with partitions common to both trees counted only once. Next, the non-shared labels are ignored, the partitions are updated accordingly, and the symmetric difference—i.e., partitions that exist in only one of the trees—is calculated. The Bourque distance is the difference between the total number of partitions and the symmetric difference both defined above. The Bourque distance is a metric and can be computed in linear time, but it does not allow mutation loss.

High-order Bourque distances [Jahn et al., 2021] is defined on the same set of trees and under the same constraints as the original Bourque distance. The k-Bourque distance focus on comparing the Bourque distances between k-neighborhood subtrees——subtrees induced by the set of descendant nodes within k edges from a given node——of the two trees. These subtrees are rooted at nodes in each tree. To accommodate differences in tree sizes, perfect

matchings are constructed between the sets of k-neighborhood subtrees, augmenting the smaller set with dummy (empty) subtrees if necessary. The k-Bourque distance is then defined as the minimum total cost of a perfect matching in this bipartite graph, where each edge weight corresponds to the Bourque distance between a pair of k-neighborhood subtrees. The algorithm runs in cubic time with respect to the maximum size of the two trees.

3.3 Distance Methods Violating ISA

A recent generalization of the Robinson-Foulds distance is defined on clonal trees that may have different sets of labels and repeated labels. It is called is the k-Robinson–Foulds (k-RF) dissimilarity measure [Khayatian et al., 2024]. It finds the partitions for each edge but considers only the labels in the k-neighborhood of the endpoints. The distance is the symmetric difference of the set of these partitions. The restriction of the k-RF to the set of uniquely labeled mutation trees is a metric. The k-RF distance of zero does not imply having two identical trees which is an important drawback of the distance. Nevertheless, the distance can be computed in polynomial time.

The k-Bourque and k-RF distances compare two trees by looking at the labels of nodes within k nodes of each edge. In other words, they determine how similar the neighbourhoods around each edge are in both trees. This definition depends on the choice of k. When k is set to a small value, these measures only capture similarities in local regions of the trees. However, focusing on k-neighbourhoods allows the calculation of distances for trees with repeated labels in polynomial time.

The Generalized Robinson-Foulds (GRF) distance [Llabrés et al., 2020] for clonal trees is defined similarly to its definition for phylogenetic trees and captures dissimilarities with better resolution than the other generalizations of the RF distance. It computes the normalized

symmetric difference (Jaccard distance) between multisets of clones from two trees. It is a metric and can be calculated in polynomial time. However, it does not consider mutation loss.

The triplet-based similarity score MP3 [Ciccolella et al., 2021a] is defined on clonal trees and accounts for repeated labels and mutation loss. For every three mutation labels, it compares the set of labels of the minimal subtrees containing the three mutation labels in each tree. Because mutations can appear multiple times in different parts of the tree (repeated labels), a single triplet of mutations may correspond to multiple subtrees. MP3 handles this by taking the union of the labels from all such relevant subtrees. A key limitation of this method is that it treats mutation loss the same as regular mutation acquisition, overlooking the inheritance-based structure of clonal trees. For instance, let $L_1 = \{A, B, C, D\}$, $L_2 = \{A, B, C, D, \text{“}D\text{”}\}$ and $L_3 = \{A, B, C\}$ denote the label sets of the minimal subtrees containing A, B, C in T, T', and T'', respectively. Here “D” represents loss of the mutation D. The measure considers L_1 and L_3 more similar than L_2 and L_3. However, the extra label D is lost at some point in L_2, meaning that the descendant nodes no longer inherit the extra mutation. So, from the inheritance-based point of view, L_2 and L_3 are more similar.

Despite these advances, existing measures are not specifically designed to capture meaningful differences between clonal trees. They either struggle to balance computational efficiency and resolution or lack the ability to handle complex evolutionary scenarios such as mutation loss and repeated mutations. We propose the Clonal Evolution Distances (CED) as a new framework to address these limitations by introducing biologically motivated operations. Table 3.1 summarizes the key properties of distances defined for clonal trees.

Distance	Repeated labels	Mutation loss	Different set of labels	ISA	Runtime	Metric
CASet	x	x	✓	✓	Polynomial on trees with same labels	x
DISC	x	x	✓	✓	Polynomial on trees with same labels	x
MLTD	x	x	✓	✓	Polynomial	x
Path-distance	x	x	x	✓		
PCD	x	x	x	✓		x
AD	x	x	x	✓		x
CD	x	x	x	✓		
Bourque	x	x	✓	✓	Polynomial	✓
k-Bourque	x	x	✓	✓	Polynomial	
K-RF	✓	x	✓	x	Polynomial	x
GRF	✓	x	✓	x	Polynomial	✓
MP3	✓	✓	✓	x		
CED metric	✓	✓	✓	x	NP-hard	✓
CED semi-metric	✓	✓	✓	x	Polynomial in all variables except a rare one	x

Table 3.1: Comparison of distance measures for clonal trees

Chapter 4

Methodology: Clonal Evolution Distance Metric and Semi-Metric

We propose the following distance method, which quantifies both the number of subclones (nodes) identified differently by various methods and the extent of these differences. Whether the input data comes from the same method used on different patients, or on the same patient at different time points, it measures the number of subclones which contain different mutations. It uses specific operations to transform one tree into the other one.

A clonal tree $T(V(T), E(T), L(T))$ is a rooted tree where nodes in $V(T)$ represent subclones (sets of cells with a common set of mutations in a tumor), the edges in $E(T)$ represent the process of mutation acquisition or loss corresponding to the labels of the nodes they enter, and $L(T)$ is the multiset union of labels of the nodes of T. We impose no restrictions on the number of repeated labels or mutation loss; however, a label that is lost on a lineage is never acquired again on that same lineage, because if reacquisition occurred, the sequencing data would no longer contain any evidence of the earlier loss. In this document, mutation loss is denoted by quotation marks. For instance, "ℓ" indicates the loss of mutation ℓ. In Figure 2.1 we have $L(T) = \{A\} \cup \{B\} \cup \{C, D\} \cup \{F\} \cup \{E\}$. For each node $v \in V(T)$, the

notation $L(v)$ represent the set of mutations which v is labeled with. The root represents a dummy node with in-degree zero, labeled r, on which no operations can be applied. So the root's children represent the initial mutations. A node v is an ancestor of u if it appears on the path from the root to u. Consequently, u is referred to as a descendant of v. Even though a mutation can be acquired in different places and multiple nodes may have the same set of labels, the corresponding nodes are regarded as distinct if they differ in their ancestral histories or historical content.

Definition 1 (Historical Content)**.** *Let T be a clonal tree and v a node in T. The history content of v denoted by $\mathcal{HC}_T(v)$, is defined as the union of every label set of the ancestors of v, including the labels of v itself, where for any mutation A, if both A and its loss "A" label some ancestors of v, then neither A nor "A" is included in $\mathcal{HC}_T(v)$.*

Definition 2 (Mutation Content)**.** *Let T be a clonal tree. For each node $v \in T$, the pair $(L(v), \mathcal{HC}_T(v))$ denoted by $\mathcal{MC}_T(v)$ is referred to as the mutation content of v with respect to T.*

4.1 CED Operations

Before presenting the formal definition of the operations, we first illustrate how they work through a few examples.

- Deletion: This operation removes a label. Removing a label may result in an empty node. In such cases, as shown in Figure 4.1, the empty node is also deleted.
- Insertion: This operation acts as the inverse of deletion. So, as shown in the Figure 4.2, it may either add labels to existing nodes or create new nodes in which to insert the labels. A new node can be inserted in multiple ways, depending on which subtrees

Figure 4.1: The tree T' is the tree T after deleting C and D. Deleting the label C changes the label set of its node and the historical content of the nodes labeled F and E, but then deleting the label D removes one node too.

rooted at the original children of the new node's parent are reassigned to be rooted at the new node. In other words, inserting a new node enables the redistribution of the children of the parent node.

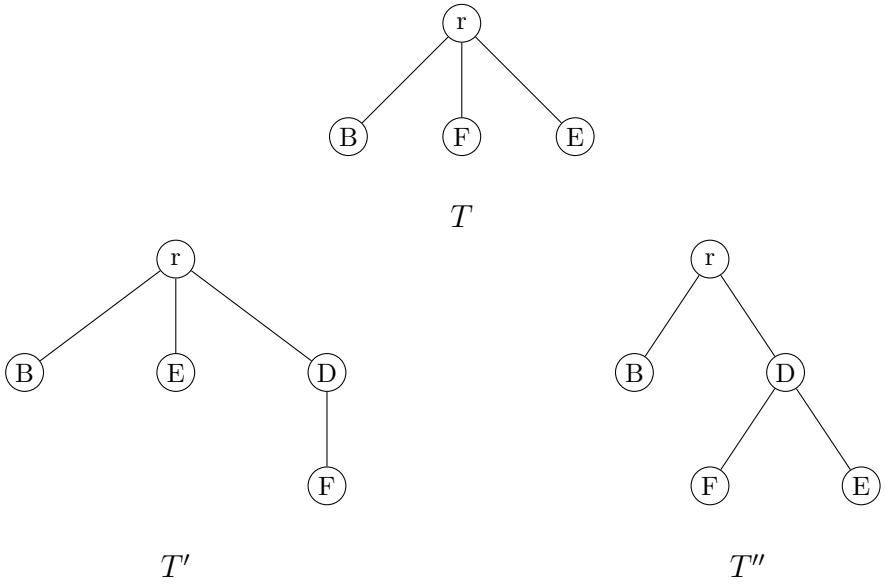
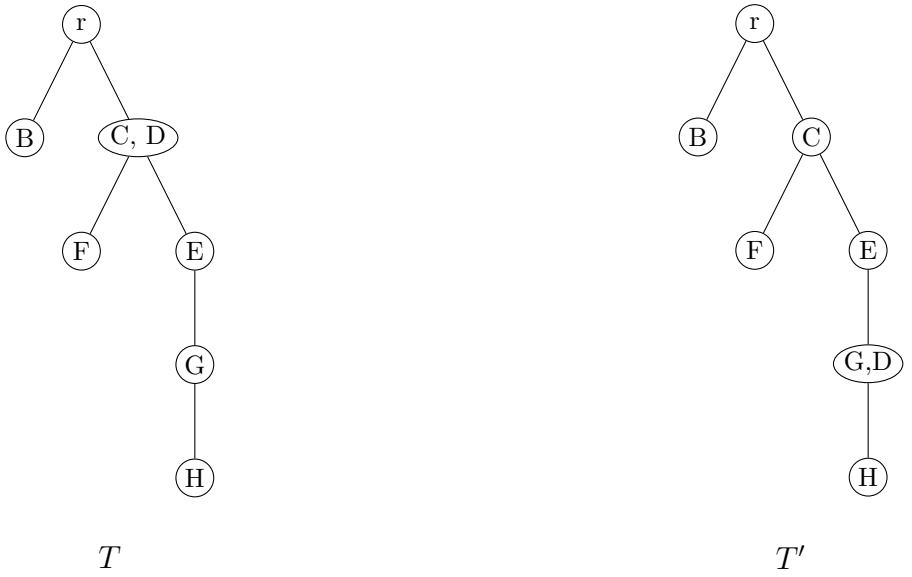


Figure 4.2: A new node can be inserted in multiple ways, depending on which subtrees rooted at the original children of the new node's parent are reassigned to be rooted at the new node; however, only one of these insertions is the inverse of a specific deletion. Two of the possible ways to insert the label D into the tree T are shown above. The tree T' shows an insertion which does not reverse the deletion of D in the Figure 4.1. This insertion adds a new node and changes the historical content of the node labeled F. The tree T'' shows the inverse of the deletion of the label D in the Figure 4.1. This insertion changes the historical content of the node labeled E as well.

- Rearrangements: This operation can move labels between existing nodes along a lineage (see Figure 4.3). Similar to a deletion, a rearrangement may result in an empty node as shown in the Figure 4.4; in such cases, the empty node is removed. It can also create a new node (see Figure 4.5), similar to the insertion operation; however, instead of inserting a new label, it relocates a label from another node into the newly created one.



T T'

Figure 4.3: Rearrangement between existing nodes: The tree T' is the tree T after moving the label D to the node labeled G. The rearrangement of the label D, as shown above, affects the label sets of both its source and destination nodes. It also changes the historical contents of the nodes labeled E, F and G, but has no effect on the mutation content of the node labeled H.

Once a mutation is deleted from a lineage, any subsequent mutation loss of that label on the same lineage becomes meaningless, as the label no longer exists to be lost. This is why the deletion of a mutation label should automatically result in the removal of its mutation losses from all descendant nodes. To maintain the reversibility of the operations, insertions should add the mutation losses intended for the descendant nodes. The same principle applies to rearrangements. Mutation loss labels can also be deleted, inserted, or rearranged independently, just like mutation labels.

Figure 4.4: Rearrangement between existing nodes resulting in node removal: The tree T' is the tree T after moving the label C to the node labeled E. The rearrangement of the label C, as shown above, removes its original node and changes the historical contents of the nodes labeled F and the label set of the node labeled with E.

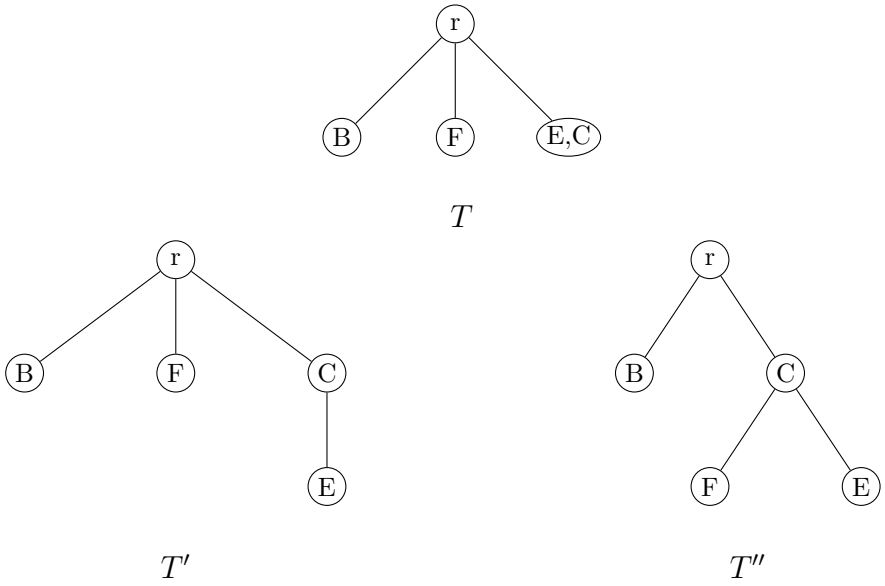


Figure 4.5: A label can be rearranged to a new node in different ways, depending on how the insertion of the new node redistributes the subtrees rooted at the parent node. The rearrangement resulting in the tree T'' is the reverse of the rearrangement of the label C in Figure 4.4.

Figure 4.6 illustrates how a mutation loss can alter the effect of an operation on the tree. Deleting label A which would normally affect the contents of all descendant nodes, does not impact those that are labeled with "A" as one of its labels and their descendant nodes.

Labels can be represented using letters, numbers or a combination of both. We refer to the nodes using lowercase letters.

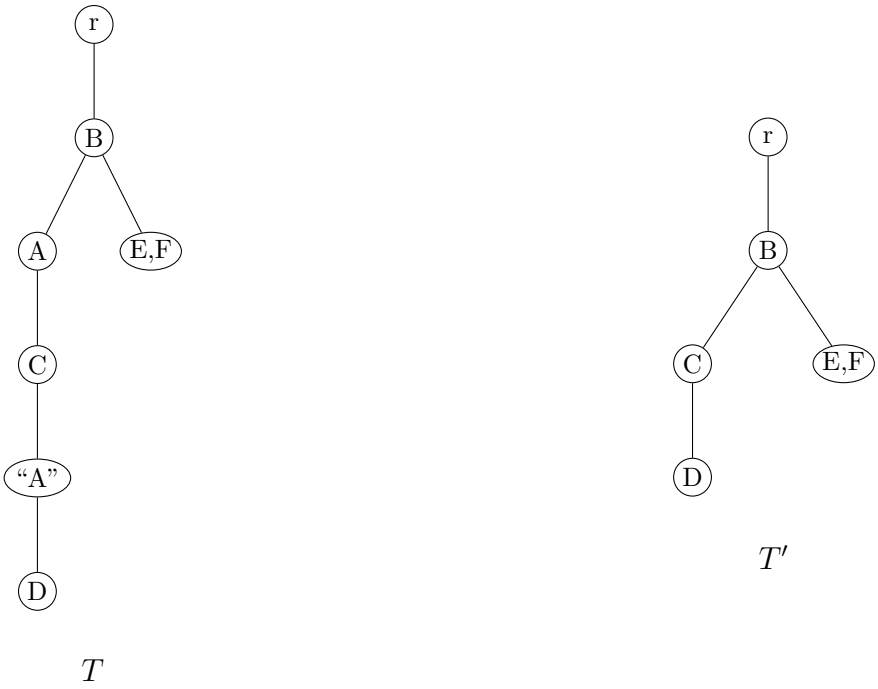


Figure 4.6: The tree T' is the tree T after deletion of A. Deleting Label A does not change the mutation content of the node labeled D

We are now ready to present the formal definition of the CED operations. Throughout this document, given a clonal tree T, a node v is said to be new if it does not already exist in T. In such a case, we denote v by $p \triangleright v$, where p is an existing node of T. This notation indicates that the node v should be introduced as a child of p.

Definition 3 (CED Operations)**.** *Let T be a clonal tree. We define three operations that can be applied to all nodes except the root r:*

- *Deletion $D_T(\ell, a)$ (or $D_T(\text{“}\ell\text{”}, a)$ when referring to a mutation loss): This operation first removes the label ℓ (“ℓ”) from $L(a)$. If a has only one label, the operation removes the node and reattaches its children to its parent. The operation then automatically removes “ℓ” from the set of labels of the descendant nodes of a. The operation $D_T(\text{“}\ell\text{”}, a)$ only removes “ℓ” from $L(a)$, because, as specified earlier, once a label is lost, it cannot be reacquired in the same lineage. Consequently, there is no descendant labeled with ℓ*

that needs to be automatically removed.

- *Insertion $I_T(\ell, a, B, C)$: where B is a set of pairs (b_i, D_i), with each b_i being a descendant of a, and D_i denoting a subset of the children of b_i's parent node.*

  This operation first inserts the label ℓ into node a. If a is an existing node, the parameter C is empty. If a is a new node (needed to be created on T), the parameter C represents a subset of the original children of a's parent. The subtrees rooted at nodes in C are detached from their parent and reattached to a, making them children of a. The operation then inserts "ℓ" into a subset of a's descendants, denoted by the first elements in B such as b_i. For each b_i, if b_i already exists on T, then D_i is empty. If b_i is a new node, then D_i denotes the children of b_i's parent whose subtrees should be reattached to b_i. Insertion of a mutation loss label, denoted $I_T(\text{"}\ell\text{"}, a, C)$, does not require the third parameter B used in regular label deletions, since a lost label cannot be reinserted into descendant nodes.

- *Rearrangement $R_T(\ell, a, I)$: where $I = I_{\mathcal{T}}(\ell, b, B, C)$ and a is either an ancestor or a descendant of b. Essentially, the rearrangement operation applies $D_T(\ell, b)$ and then I. To be specific, the rearrangement operation represents the removal of label ℓ from $L(a)$ and automatic removal of "ℓ" from the sets of labels of the descendant nodes, in the same way that the deletion operation does. Let $\mathcal{T}$ represent the transformed tree after these deletions. Then the insertion operation within the rearrangement operation adds ℓ to $L(b)$. The insertion operation $I_{\mathcal{T}}(\ell, b, B, C)$ specifies how the reinsertion of the label ℓ is performed, including the position of the inserted "ℓ" and the reattachment of the children of b's parent if b is a new node. Similar to the other operations, the rearrangement of a mutation loss, denoted by $R_T(\text{"}\ell\text{"}, a, I)$, only deletes "$\ell$" from $L(a)$ and adds it to $L(b)$.*

When there is no ambiguity we can drop T from the notations.

The following example shows how the insertion operation is used within a rearrangement operation.

Example 1. *Consider Figure 4.7. The operation $R_T(3, c, I)$ where $I = I_{\mathcal{T}}(3, r \triangleright f, \{(b \triangleright g, \emptyset)\}, \{c, b\})$ transforms T into T'. First, the labels 3 and "3" are removed from $L(c)$ and $L(e)$, respectively, resulting in the intermediate tree $\mathcal{T}$. Since $L(e)$ becomes empty, the node e must be eliminated. Next, I creates node f as a child of r and assigns it the label 3. It then creates node g as a child of b, adding label "3" to $L(g)$. Finally, the subtrees rooted at c and b are detached from their positions in $\mathcal{T}$ and reattached as children of f.*

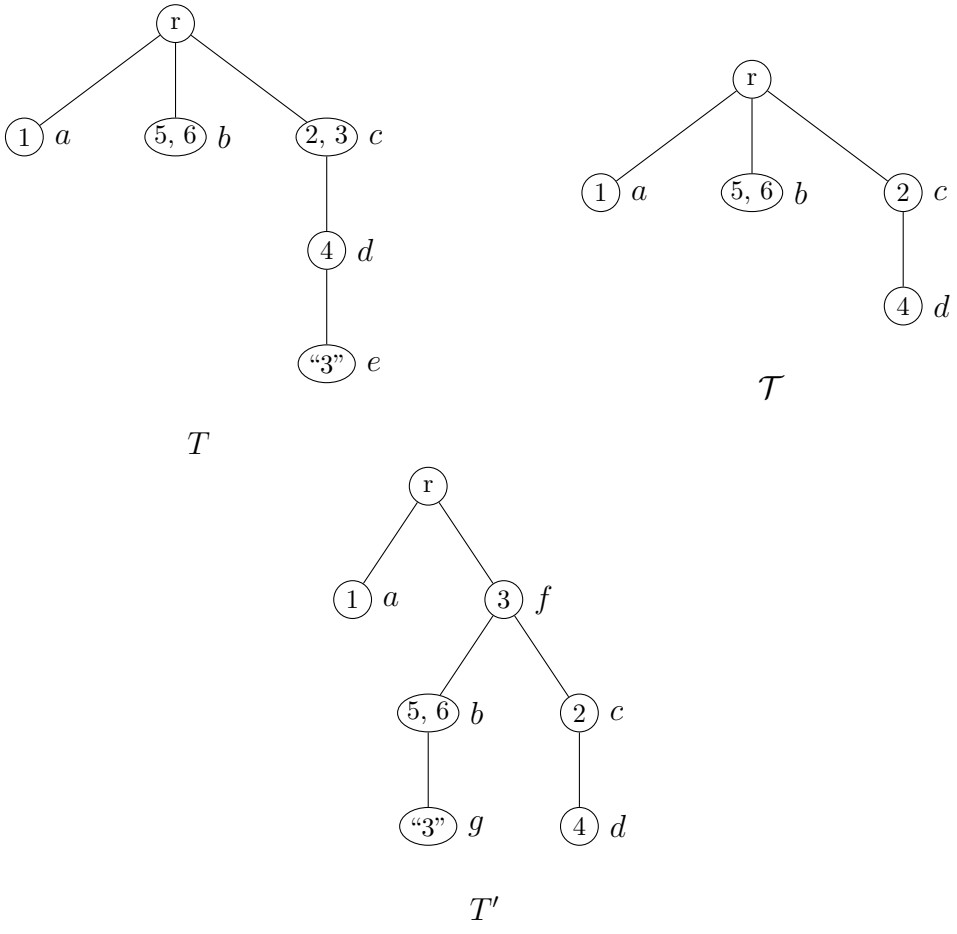


Figure 4.7: Nodes are labeled with lowercase letters, while numbers represent mutations. The operation $R_T(3, c, I)$ where $I = I_{\mathcal{T}}(3, r \triangleright f, \{(b \triangleright g, \emptyset)\}, \{c, b\})$ transforms T into T'.

Definition 4 (Valid Operation)**.** *A CED operation is called* valid *if its resulting tree remains a clonal tree. In other words, no two nodes within the same lineage share the same label.*

From this point onward, all references to CED operations will refer to valid operations.

When $S = f_1, \ldots, f_n$ is a sequence of CED operations, $T\langle S\rangle$ denotes the tree obtained by applying S to T, i.e applying f_1 through f_n successively.

Proposition 1. *Let T be a clonal tree. The CED operations are reversible.*

Proof. We will show below that each operation f can be reversed by an operation f^{-1}.

- **Deletion** $D_T(\ell, a)$: Let B denote the set of pairs (b_i, D_i) where b_i are the descendants of a with "ℓ" $\in L(b_i)$. For each b_i, if $L(b_i) = \{$"ℓ"$\}$ then D_i denotes the set of children of b_i; otherwise, $D_i = \emptyset$.

  If deleting label ℓ from $L(a)$ leads to the deletion of node a, then let C denote the set of children of a; otherwise, let $C = \emptyset$. Then,

  $$D_T^{-1}(\ell, a) = I_{T\langle D\rangle}(\ell, a, B, C),$$

  . Similarly,

  $$D_T^{-1}(\text{“}\ell\text{”}, a) = I_{T\langle D\rangle}(\text{“}\ell\text{”}, a, C).$$

- **Insertion** $I_T(\ell, a, B, C)$: This operation adds ℓ to $L(a)$ and "ℓ" to $L(b_i)$ for each i in B. Each time it creates a new node, the operation specifies how the children of the parent should be distributed. The reverse operation should delete the new nodes and reattach the children to their original parent. We claim that

  $$I_T^{-1}(\ell, a, B, C) = D_{T\langle I\rangle}(\ell, a).$$

  The new nodes have only one label (ℓ or "ℓ"), which is deleted by $D_{T\langle I\rangle}(\ell, a)$, leading to node deletion, and the children are reattached to their original parent. Now, we just

need to show that $D_{T\langle I\rangle}(\ell, a)$ does not remove any label not inserted by $I_T(\ell, a, B, C)$. We know that we only have repeated labels on different lineages, so the insertion operation could not insert ℓ in a lineage that already has this label. Therefore, the ℓ and all "ℓ" labels deleted by $D_{T\langle I\rangle}(\ell, a)$ were inserted by the insertion operation. Similarly,

$$I_T^{-1}(\text{“}\ell\text{”}, a, C) = D_{T\langle I\rangle}(\text{“}\ell\text{”}, a).$$

- **Rearrangement** $R_T(\ell, a, I)$: Another way to explain the rearrangement operation is that it applies $D_T(\ell, a)$ and then I. Now that we have found the reverse of the deletion and insertion operations, we can simply use them to see that

$$R_T^{-1}(\ell, a, I_{\mathcal{T}}(\ell, b, B, C)) = R_{T\langle R\rangle}(\ell, b, D_{\mathcal{T}}^{-1}(\ell, a)).$$

  The same formula applies to the rearrangement of mutation losses.

□

With respect to nodes, a CED operation f either deletes nodes, inserts nodes, deletes nodes and reinsert them (only possible with rearrangements that delete a node with label ℓ and insert a new node with label ℓ on the same lineage), or possibly modifies their sets of labels. Let V_f be the set of all nodes in $V(T)$ like v such that the operation f does not delete it, or deletes it but puts it back (not necessarily in the same position). We can represent the corresponding node in $T\langle f\rangle$ with $\Phi_f(v)$. So Φ_f defines an injective mapping from V_f into $V(T\langle f\rangle)$. We call $\Phi_f : V_f \to \Phi_f(V_f)$ the **effect function** of f.

Definition 5 (Weight Function)**.** *For a CED operation f on a clonal tree T, we define the weight function $W_T(f)$ as*

$$\left|\{v \in V_f \mid \mathcal{MC}_T(v) \neq \mathcal{MC}_{T\langle f\rangle}(\Phi_f(v))\} \cup (V(T) \setminus V_f) \cup (V(T\langle f\rangle) \setminus \Phi_f(V_f))\right|,$$

where $T\langle f\rangle$ denotes the resulting tree after applying f on T. When there is no ambiguity, we omit T in the notation and write $W(f)$.

In other words, the weight function counts the number of nodes whose mutation content is modified by operation f, plus the number of nodes that have been deleted (excluding those deleted by a rearrangement operation that reinserts the node) and the number of nodes that have been inserted (not reinserted by a rearrangement operation).

Proposition 2. *Let f be a CED operation on a clonal tree T and f^{-1} be its reverse operation. Then*

$$W_T(f) = W_{T\langle f\rangle}(f^{-1}).$$

Proof. The operation f^{-1} is the reverse of f, hence $T\langle f, f^{-1}\rangle = T$. Therefore,

$$W_{f^{-1}} = \left|\left\{u \in V_{f^{-1}} \mid \mathcal{MC}_{T\langle f\rangle}(u) \neq \mathcal{MC}_T(\Phi_{f^{-1}}(u))\right\} \cup (V(T\langle f\rangle) \setminus V_{f^{-1}}) \cup (V(T) \setminus \Phi_{f^{-1}}(V_{f^{-1}}))\right|,$$

where $\Phi_f : V_f \to \Phi_f(V_f) \subseteq V(T\langle f\rangle)$. On the other hand, according to the definition, $v \in V_f$ if and only if $\Phi_f(v) \in V_{f^{-1}}$, hence

$$V_{f^{-1}} = \Phi_f(V_f). \tag{4.1}$$

In other words, since Φ is injective, for each u in $V_{f^{-1}}$, there is a unique v in V_f such that $\Phi(v) = u$. Furthermore,

$$\Phi_{f^{-1}}(\Phi_f(v)) = v. \tag{4.2}$$

Let $N_1 = \left|\left\{u \in V_{f^{-1}} \mid \mathcal{MC}_{T\langle f\rangle}(u) \neq \mathcal{MC}_T\big(\Phi_{f^{-1}}(u)\big)\right\}\right|$. Hence, $\Phi_{f^{-1}} : \Phi_f(V_f) \to V_f$, and

$$\begin{aligned}
N_1 &\overset{\text{by (4.1)}}{=} \left|\left\{\Phi_f(v) \in V_{f^{-1}} \mid \mathcal{MC}_{T\langle f\rangle}(\Phi_f(v)) \neq \mathcal{MC}_T(\Phi_{f^{-1}}(\Phi_f(v))) \text{ and } v \in V_f\right\}\right| \\
&\overset{\text{by (4.2)}}{=} \left|\left\{\Phi_f(v) \in V_{f^{-1}} \mid \mathcal{MC}_{T\langle f\rangle}(\Phi_f(v)) \neq \mathcal{MC}_T(v) \text{ and } v \in V_f\right\}\right| \\
&= \left|\left\{v \in V_f \mid \mathcal{MC}_{T\langle f\rangle}(\Phi_f(v)) \neq \mathcal{MC}_T(v)\right\}\right|,
\end{aligned}$$

where the last equality holds since $\Phi_f : V_f \to \Phi_f(V_f)$ is bijective.

Furthermore, again according to the definition, $V_{f^{-1}} = \Phi_f(V_f)$ and $V_f = \Phi_{f^{-1}}(V_{f^{-1}})$. Therefore,

$$V(T\langle f\rangle) \setminus V_{f^{-1}} = V(T\langle f\rangle) \setminus \Phi_f(V_f),$$

and

$$V(T) \setminus \Phi_{f^{-1}}(V_{f^{-1}}) = V(T) \setminus V_f,$$

which completes our proof.

□

4.2 d_{CED} for the Trees Generated by Finite-Sites Models

We now use the CED operations to define a metric on clonal trees.

Definition 6 (Good Rearrangement)**.** *Let T and T' be two clonal trees. A* Good Rearrangement *(denoted by GR) of a label l in T (with respect to T') is a rearrangement operation that moves l from its current node in T to another node v, such that either:*

1. *the destination node v is a new node and contains no other labels,* or
2. *the destination node v contains at least one label l' such that l and l' appear together in the same node in T'.*

From this point forward, all references to CED operations implicitly assume that rearrangement operations are *Good Rearrangements* as defined above.

Definition 7 (Total Weight)**.** *Let $S = (f_1, \ldots, f_n)$ be a sequence of CED operations applied to a clonal tree T, denoted by $T\langle S\rangle$, where each operation f_i is applied before f_{i+1}.*

The total weight *of S, denoted by $W_T(S)$, is defined as*

$$W_T(S) = \sum_{i=1}^{n} W_{T\langle S_{i-1}\rangle}(f_i),$$

where, for each $1 \leq i \leq n$, we have $S_i = f_1, \ldots, f_i$, and $T\langle S_0\rangle = T$.

Corollary 1. *Let T be a clonal tree and $S = f_1, \cdots, f_n$ a sequence of CED operations. Let $T\langle S\rangle = T'$, and $S^{-1} = f_n^{-1}, \cdots f_1^{-1}$. Then $T'\langle S^{-1}\rangle = T$, and the total weight of S is equal the total weight of S^{-1}.*

Proof. According to Proposition 1, each CED operation f_i is reversible, and its reverse f_i^{-1} is also a CED operation on the resulting tree. In particular,

$$T\langle f^{-1}, f\rangle = T.$$

Let $T_0 = T$, $T_1 = T_0\langle f_1\rangle$, $T_2 = T_1\langle f_2\rangle$, and continue until $T_n = T_{n-1}\langle f_n\rangle = T'$. Therefore,

$$T_n\langle f_n^{-1}\rangle = T_{n-1}, \quad T_{n-1}\langle f_{n-1}^{-1}\rangle = T_{n-2}, \quad \ldots, \quad T_1\langle f_1^{-1}\rangle = T_0 = T.$$

Thus,

$$T'\langle S^{-1}\rangle = T.$$

The second part is a direct result of Proposition 2.

□

Definition 8 (Clonal Evolutionary Distance)**.** *Let T and T' be two clonal trees. The clonal evolutionary distance, denoted by $d_{CED}(T, T')$, is defined as*

$$d_{CED}(T, T') = \min\{\, W_T(S) \mid S \text{ transforms } T \text{ into } T' \,\}.$$

Theorem 1. *The Clonal Evolution Distance d_{CED} defines a metric on the space of clonal trees generated by finite-sites models.*

Proof. Let T, T', and T'' be three clonal trees. To prove that the Clonal Evolution Distance d_{CED} is a metric, we must show that it satisfies the following axioms:

1. Non-negativity: $d_{CED}(T, T') \geq 0$

   Proof. The weight of each sequence of CED operations is defined as a non-negative integer. Therefore, $d_{CED}(T, T') \geq 0$. □

2. $d_{CED}(T, T') = 0$ if and only if $T = T'$

   Proof. ($\Rightarrow$) If $d_{\text{CED}}(T, T') = 0$, then there exists a sequence of operations S such that $T\langle S\rangle = T'$ and $W(S) = 0$. Consequently, for every operation $f \in S$, we have $W(f) = 0$. This implies that no node is deleted without being put back, no new node is inserted, and neither the label set nor the history content of any node is modified by f. In other words, each operation in S has no effect on T. Therefore, $T = T'$.

   ($\Leftarrow$) If $T = T'$, then no operation is needed to transform one into the other, and the minimum weight of such a transformation is 0. Therefore, $d_{CED}(T, T') = 0$. □

3. Symmetry: $d_{CED}(T, T') = d_{CED}(T', T)$

   Proof. Let S be an optimal sequence of operations that transforms T' into T, with total weight $d_{CED}(T, T')$. By Corollary 1, the inverse sequence S^{-1} transforms T into T' with the same total weight. Therefore, $d_{CED}(T', T) \leq d_{CED}(T, T')$.

   By a similar argument $d_{CED}(T, T') \leq d_{CED}(T', T)$.

   Thus, we conclude that

   $$d_{CED}(T, T') = d_{CED}(T', T).$$

   □

4. Triangle Inequality: $d_{CED}(T, T') \leq d_{CED}(T, T'') + d_{CED}(T'', T')$

   Proof. Let S' and S'' denote optimal sequences transforming T into T'' and T'' into T', respectively. Define $S = S'S''$ as the concatenation of these two sequences. In other words,

   $$T\langle S\rangle = (T\langle S'\rangle)\langle S''\rangle.$$

   On the other hand, $T\langle S'\rangle = T''$. Since S'' continues directly from S', and the total weight is the sum of the individual weights, we have

   $$W(S) = W(S') + W(S'').$$

   Because S' and S'' are optimal,

   $$W(S) = d_{\text{CED}}(T, T'') + d_{\text{CED}}(T'', T').$$

   Finally, since d_{CED} is the minimum over the weights of all sequences transforming T into T' including S, it follows that

   $$\begin{aligned} d_{\text{CED}}(T, T') &\leq W(S) \\ &= d_{\text{CED}}(T, T'') + d_{\text{CED}}(T'', T'), \end{aligned}$$

   which establishes the triangle inequality.

□

□

Theorem 2. *The problem of computing the clonal evolutionary distance for mutation trees satisfying the Infinite Sites Assumption (ISA) is as hard as computing the tree editing distance [Zhang and Jiang, 1994] for such trees.*

Proof. Consider two unordered mutation trees T and T' (with no multi-labeled nodes) that satisfy the Infinite Sites Assumption (ISA), i.e., each mutation appears exactly once and is never lost. Let $e(T, T')$ denote the editing distance between T and T', where the editing distance is defined as the minimum total cost of transforming one tree into the other using the following operations:

- Deletion (D) and insertion (I), which are defined similarly to the CED operations. Note that the parameter B in CED insertion is absent since we do not allow mutation loss.
- Relabelling ($RL(\ell_1, a, \ell_2)$) changes the current label ℓ_1 of node a to label ℓ_2.

Note that since we are working under ISA, all parameters referring to mutation loss are eliminated from the notation.

We define the cost of each editing operation $C(f)$ as follows:

$$C(D(\ell, a)) = W(D(\ell, a)),$$

$$C(I(\ell, a, C)) = W(I(\ell, a, C)),$$

and

$$C(RL(\ell_1, a, \ell_2)) = W(D(\ell_1, a)) + W(I(\ell_2, a, C),$$

where C is defined by $D^{-1}(\ell_1, a)$.

Now we show that every optimal sequence S of editing distance operations that transforms T into T' corresponds to a sequence of CED operations S', whose total weight is equal to the total cost of S and transforms T into T'.

Let $S = f_1, \ldots, f_n$ be an optimal sequence of editing distance operations transforming T into T'. Let $S' = g_1, \ldots, g_m$ be the sequence of CED operations that preserves

the insertions and deletions in S, and replaces each relabelling operation with the corresponding deletion and insertion CED operations. For instance, $RL(\ell_1, a, \ell_2)$ is replaced with $D(\ell_1, a), I(\ell_2, a, C)$, where C is defined by $D^{-1}(\ell_1, a)$. In other words, $I(\ell_2, a, C)$ reverses $D(\ell_1, a)$ but with label ℓ_2 instead of the label ℓ_1. The sequence S' also transforms T into T' and by definition $C(S) = W(S')$.

The editing distance does not apply any restrictions on the intermediate trees, so at some point, the tree $T\langle f_1, \cdots, f_i\rangle$ may contain one label ℓ on two nodes a and b in a lineage where a is an ancestor of b while $T\langle f_1, \cdots, f_{i-1}\rangle$ does not. Since the sequence is optimal and T' does not have repeated labels, the label ℓ does not appear more than twice throughout the transformation, and there is an operation f_j, where $j > i$, such that f_j eliminates one of these ℓs. Therefore, one of them is applied on node a and the other is applied on b. Since the sequence is optimal, this extra ℓ is eliminated before any other operation count its node. Assume that f_i is applied on a and f_j is applied on b. We find the corresponding operations in S'. The operation f_i corresponds to an insertion, or a deletion and an insertion in case f_i is a relabelling operation. Similarly, the operation f_j may correspond to a deletion, or a deletion and insertion in S'. We replace the insertion operation corresponding to f_i with $R(\ell, b, I)$ where I inserts ℓ in a. Then we remove the deletion corresponding to f_j. If f_i is applied on b and f_j is applied on a, then we replace the deletion corresponding to f_j with $R(\ell, a, I)$ where I inserts ℓ in b, and remove the insertion operation corresponding to f_i. Next, we show that this updated version of S' has the same result and weight as the original S'. Repeating this process for all possible labels that appear twice in some intermediate tree would update S' to a sequence of valid CED operations with the same weight.

To prove our claim, consider the first case. The updated version omits $I(\ell, a, C)$ and $D(\ell, b)$ but instead includes $R(\ell, b, I(\ell, a, C))$, while the rest of the sequence remains unchanged. This rearrangement operation produces the same overall effect as the missing

insertion and deletion.

Since a is an ancestor of b, the insertion $I(\ell, a, C)$ alters the mutation content of all descendants of a, including a itself, except for b and its descendants. In the updated version, this insertion is replaced by a rearrangement that affects exactly the same nodes plus node b. Thus, the weight increases by one. However, because b is a descendant of a, the deletion $D(\ell, b)$ modifies only node b, contributing a weight of one. This deletion is omitted in the updated version, compensating for the extra weight introduced by the rearrangement.

By similar reasoning, the second case yields an updated version with the same overall effects and total weight as S'.

Now we will prove the equivalence in the other direction. Let $S'' = f_1, \ldots, f_n$ be a given sequence of CED operations that transforms T into T'. Let $g(S'')$ be the sequence of editing distance operations be defined as follows: for each rearrangement operation $f = R(\ell, a, I(\ell, b, C))$, let $g(f) = I(\ell, b, C), D(\ell, a)$. For CED operation f in S'' where f is a deletion or an insertion operation let $g(f) = f$. This sequence $g(S'')$ transforms T into T'.

We now show, by induction, that $C(g(S'')) = W(S'')$.

We are dealing with mutation trees in which nodes have one label. So, label insertion operations create new nodes and deletion operations delete nodes.

Let f_1 be the first operation in S''. If f_1 is a deletion or insertion, then $C(g(f_1)) = W(f_1)$ by our definition of cost.

If $f_1 = R(\ell, a, I(\ell, b, C))$, and a is an ancestor of b, Then $W(I(\ell, b, C)) = 1$ because it only counts the new node, since the mutation content of other nodes has not changed. Therefore, the cost of the corresponding editing insertion is equal 1. Next, when the deletion is applied, one node is deleted, and the mutation content of all descendants of a which are not descendants of b has changed because they no longer inherit ℓ. The weight $W(D(\ell, a))$ counts these nodes and $W(D(\ell, a))$ is equal the cost of the corresponding editing deletion

operation. On the other hand, $W(R(\ell, a, I(\ell, b, C)))$ counts a, b, and all nodes that were descendants of a in T, but are not descendants of b in $T\langle f_1\rangle$. So the total cost of the editing insertion and deletion is equal to $W(f_1)$. Thus $C(g(f_1)) = W(f_1)$. If $f_1 = R(\ell, a, I(\ell, b, C))$, and b is an ancestor of a, then the cost function of the editing insertion counts b and all nodes that are descendants of b but are not descendants of a. The cost function of the deletion only counts the deleted node a. Hence, similar to the previous case, $C(g(f_1)) = W(f_1)$.

Now assume that $C(g(f_k)) = W(f_k)$ for some $1 < k < n$. We aim to prove that $C(g(f_{k+1})) = W(f_{k+1})$. Since $T\langle f_1, \ldots, f_k\rangle = T\langle g(f_1), \ldots, g(f_k)\rangle$, let this tree be called T_k. We use the first case we proved to see that when $g(f_{k+1})$ and f_{k+1} are applied on T_k, we have $C(g(f_{k+1})) = W(f_{k+1})$.

By induction, for each k, we have $C(g(f_k)) = W(f_k)$, and thus $C(g(S'')) = W(S'')$.

Hence, we observe that if S is a solution to the CED problem transforming T into T', then $g(S)$, as defined above, is a solution for the editing distance problem, which completes the proof. □

Lemma 1. *The cost function defined in the proof of Theorem 2 is a metric on the set of mutation trees.*

Proof. Let T_1, T_2, and T_3 be mutation trees. We prove that the cost function C satisfies the following properties:

- If f is an editing operation and $T_1\langle f\rangle = T_2$, then $C(f) = 0$ if and only if $T_1 = T_2$.
- If $T_1\langle f\rangle = T_2$ and $T_2\langle f'\rangle = T_1$, then $C(f) = C(f')$.
- If $T_1\langle f\rangle = T_2$, $T_1\langle f'\rangle = T_3$, and $T_3\langle f''\rangle = T_2$, then $C(f) \leq C(f') + C(f'')$.

The first two properties follow directly from the definition of the weight function. It remains to prove the triangle inequality.

Suppose first that f deletes a label ℓ. Then either one of f' and f'' performs the same deletion while the other makes no change, or f' relabels ℓ and f'' deletes the resulting substitute label. In both cases, the cost of deleting ℓ is accounted for in $C(f') + C(f'')$. In particular, in the second case, the deletion cost is included in the relabelling cost by definition. Hence, the triangle inequality holds.

Next, let f be the insertion of a label ℓ. Again, either one of f' and f'' performs the insertion and the other makes no change, or f' relabels a label ℓ' to ℓ and the other inserts ℓ'. The inequality is immediate in the first case. In the second case, the insertion weight is included in the relabelling cost, so the inequality follows.

Finally, assume that f is a relabelling operation changing ℓ into ℓ'. Then either f' relabels ℓ to an intermediate label ℓ'' and f'' relabels ℓ'' to ℓ', or f' deletes ℓ and f'' inserts ℓ'. In the first case, the weight of deleting ℓ is included in the cost of the first relabelling and the weight of inserting ℓ' is included in the cost of the second, implying that $C(f) \leq C(f') + C(f'')$. In the second case, the sum of the deletion and insertion costs is exactly the cost of f. In both situations, the triangle inequality holds, completing the proof. □

Corollary 2. *The problem of finding the CED distance for clonal trees generated by finite sites model is NP-hard.*

Proof. Calculating the editing distance with a metric cost function such as the cost function defined in Theorem 2 is NP-hard [Zhang and Jiang, 1994].

Suppose there exists an algorithm, denoted as `Alg`, with time complexity $O(\texttt{Alg})$, that solves the general CED problem on clonal trees. The specific case discussed in Theorem 2 is a special instance of the general CED problem and can thus be solved by `Alg`. This implies that `Alg` must handle, at a minimum, the complexity of the editing distance problem addressed in Theorem 2 which is known to be NP-hard [**?**]. Therefore, the general CED problem is NP-hard.

□

4.3 Diameter of the CED Distance

Consider the following definition: Let $\mathcal{T}$ be the set of all clonal trees. The *diameter* of the CED distance over $\mathcal{T}$ is defined as

$$\text{diam}_{CED}(\mathcal{T}) = \max_{T,T' \in \mathcal{T}} d_{CED}(T, T'). \tag{4.3}$$

Without additional constraints, the diameter is infinite, since starting from a given clonal tree T, one can always insert more labels into T to obtain another tree T' with a larger distance. Therefore, it is necessary to impose constraints on the number of labels.

Definition 9 (Diameter of CED Distance)**.** *Let $\mathcal{T}$ be the set of all clonal trees T with $|L(T)| \leq n$ for some positive integer n. The* diameter *of the CED distance over $\mathcal{T}$ is defined by the equation 4.3*

Proposition 3. *Let $\mathcal{T}$ be the set of all clonal trees T with $|L(T)| \leq n$ for some positive integer n. The* diameter *of the CED distance over $\mathcal{T}$ is*

$$\text{diam}_{CED}(\mathcal{T}) = 2(n-1).$$

Proof. Let T and T' be two clonal trees with n labels (including the root) that have no labels in common except for the root and have no mutation loss. This means, each label on T except the root requires at least one operation to be deleted, and each label on T' except the root requires at least one operation to be inserted. Consider a sequence of CED operations that transforms T into T' by first deleting all labels in $L(T)$ (except for the root)

from the leaves to the root, and then inserting all labels in $L(T')$ (except for the root) from the root to the leaves.

Due to the bottom-up deletion and top-down insertion of labels, the weight of each operation is one. Therefore, the total weight of this sequence is:

$$\begin{aligned} |L(T)| - 1 + |L(T')| - 1 &= |L(T)| + |L(T')| - 2 \\ &= 2n - 2 \end{aligned}$$

This sequence has the least number of operations, and the weight of each operation is the least possible weight, hence it is the optimal sequence transforming T into T'. Consequently,

$$\text{diam}_{CED}(\mathcal{T}) \geq 2(n-1). \tag{4.4}$$

On the other hand, for any two given trees T and T', the bottom-up deletion and top-down insertion procedures described above generate a sequence that transforms T into T' with total weight $|L(T)| + |L(T')| - 2$, which is at most $2n - 2$. Hence,

$$d_{CED}(T, T') \leq 2n - 2.$$

Since the trees are fixed, it follows that

$$\text{diam}_{CED}(\mathcal{T}) \leq 2(n-1). \tag{4.5}$$

Combining Equation 4.5 with Equation 4.4 gives the desired result.

□

4.4 Clonal Evolution Semi-Metric

In this section, we introduce the *clonal evolution semi-metric* δ_{CED}, which is inspired by the clonal evolution distance metric. In Proposition 4, we will show that δ_{CED} satisfies the

properties of a semi-metric. Unlike the full metric, δ_{CED} is computed using an algorithm whose complexity is exponential in a parameter that we expect to be small in practical applications, while remaining polynomial with respect to all other variables. Therefore, the problem is fixed-parameter tractable (FPT) with respect to this parameter.

When comparing two clonal trees path by path, it is preferable to match paths that most likely originate from the same lineage or tumor region. Paths that share a larger set of labels or have greater overlap in mutation content are more likely to represent the same clonal lineage across the two trees. In other words, this approach to path matching better reflects regional similarities between tumors. This intuition narrows the space of possible path matchings and leads to a more biologically meaningful transformation, which is the central idea behind the clonal evolution semi-metric δ_{CED}. The semi-metric δ_{CED} prioritizes transformations that preserve structurally and label-wise similar paths. To formalize this approach, we introduce a set of definitions below.

In this section, mutation losses such as "ℓ" are not deleted or inserted as a side effect of deletion, insertion or rearrangement of ℓ. Later, we explain why this approach does not lead to counting extra nodes, unlike in the previous section.

Definition 10 (Semi-CED Operation)**.** *Let T be a clonal tree. We define three operations for all nodes except the root r:*

- *Deletion $D_T(\ell, a)$ (or $D_T(\text{“}\ell\text{”}, a)$ when referring to a mutation loss): This operation works in the same way as the CED deletion, except that it removes only the label specified by its parameter.*
- *Insertion $I_T(\ell, a, C)$: This operation works in the same way as the CED insertion, except that it inserts only the label specified by its parameter.*
- *Rearrangement $R_T(\ell, a, I)$: Where $I = I_{\mathcal{T}}(\ell, b, C)$ and either a is an ancestor or a*

descendant of b. This operation works in the same way as the CED rearrangement, except that it only relocates the label specified by its parameter and no other labels are touched by it.

When there is no ambiguity we can drop T from the notations.

In this section, we consider all operations as semi-CED operations.

Definition 11 (Tree Partition)**.** *A* tree partition *of a rooted tree T is a collection $P = \{T_0, T_1, \ldots, T_k\}$ of vertex-disjoint subtrees, each consisting of a single path in T satisfying the following conditions:*

- *T_0 = root, i.e., T_0 contains only the root of T,*
- *For each $i \geq 1$, T_i is a path of nodes starting from a child of a node already included in $T_0, \ldots, T_{i-1}$, and ending at a descendant leaf not already covered by any T_j for $j < i$.*
- *Every node in T belongs to exactly one of the subtrees in P.*

Example 2. *For the tree T' in Figure 4.7, the following are two possible tree partitions: $P_1 = \{r, f \hookrightarrow g, c, a\}$, and $P_2 = \{r, f \hookrightarrow c, b \hookrightarrow g\}$, where $n_1 \hookrightarrow n_2$ denotes the unique path between nodes n_1 and n_2 in the tree.*

Definition 12 (Pairwise Consistent Tree Partitions)**.** *Let T and T' be two clonal trees, and let the ordered sets $P = \{T_0, \ldots, T_k\}$ and $P' = \{T'_0, \ldots, T'_q\}$ be tree partitions of T and T', respectively, with $m = min\{k, q\}$. The pair (P, P') is called a* pairwise consistent tree partition *(PCTP) for T and T' if, for any other pair of tree partitions (ρ, ρ'), where*

$$\rho = \{\mathcal{T}_0, \ldots, \mathcal{T}_h\}, \quad \rho' = \{\mathcal{T}'_0, \ldots, \mathcal{T}'_s\},$$

the following conditions hold. For each $i \leq m$,

1. *if* $i \leq \min\{h, s\}$, *then:*

$$|L(\mathcal{T}_i) \cap L(\mathcal{T}'_i)| \leq |L(T_i) \cap L(T'_i)|.$$

2. *In case of a tie, the number of common labels whose corresponding nodes share a common historical content is greater than or equal to that in* (ρ, ρ').

3. *In case of a tie, the pair in the pairwise consistent tree partition has a smaller label intersection, denoted by* M, *with the paths of higher index. In other words,*

$$\begin{aligned} M &= \max_{j=1}^{q-i} \left|L(T_i) \cap L(T'_{i+j})\right| + \max_{j=1}^{k-i} |L(T_{i+j}) \cap L(T'_i)| \\ &\leq \max_{j=1}^{s-i} \left|L(\mathcal{T}_i) \cap L(\mathcal{T}'_{i+j})\right| + \max_{j=1}^{h-i} |L(\mathcal{T}_{i+j}) \cap L(\mathcal{T}'_i)|\,. \end{aligned}$$

4. *In case of a tie, among the nodes containing the shared labels mentioned in condition (3), the number of those in* (P, P') *with preserved historical content is less than or equal to that in* (ρ, ρ'). *In other words, in the pairwise consistent tree partition, the paths that have a match with more preserved historical content are left to be paired with them in the next steps.*

5. *In case of a tie, fewer or an equal number of rearrangement operations are needed to transform the* i*-th path in* P *into the corresponding path in* P' *than in* (ρ, ρ').

Remark 1. *It is important to note that multiple PCTPs arise only when two pairwise consistent tree partitions are tied across all five conditions specified in the definition.*

Example 3. *Consider trees* T *and* T' *in Figure 4.8 . A PCTP should assign* $T_1 = a \hookrightarrow e$ *and* $T'_1 = a' \hookrightarrow e'$, *since they share 8 labels—more than any other possible pairing.*

Next, the pairs $(f \hookrightarrow g,\ f' \hookrightarrow g')$, $(h \hookrightarrow k,\ f' \hookrightarrow g')$, *and* $(h \hookrightarrow k,\ h' \hookrightarrow j')$ *each have two labels in common. However, the labels shared between* $(h \hookrightarrow k,\ f' \hookrightarrow g')$ *do not preserve the historical content of their corresponding nodes in* T *and* T'. *Therefore,* (P_1, P'_1) *where*

$$P_1 = \{r, a \hookrightarrow e, h \hookrightarrow k, f \hookrightarrow g\}, \quad P'_1 = \{r, a' \hookrightarrow e', h' \hookrightarrow j', f' \hookrightarrow g'\}$$

forms a PCTP and (P_2, P_2') *where*

$$P_2 = \{r, a \hookrightarrow e, f \hookrightarrow g, h \hookrightarrow k\}, \quad P_2' = \{r, a' \hookrightarrow e', f' \hookrightarrow g', h' \hookrightarrow j'\}$$

forms the other one.

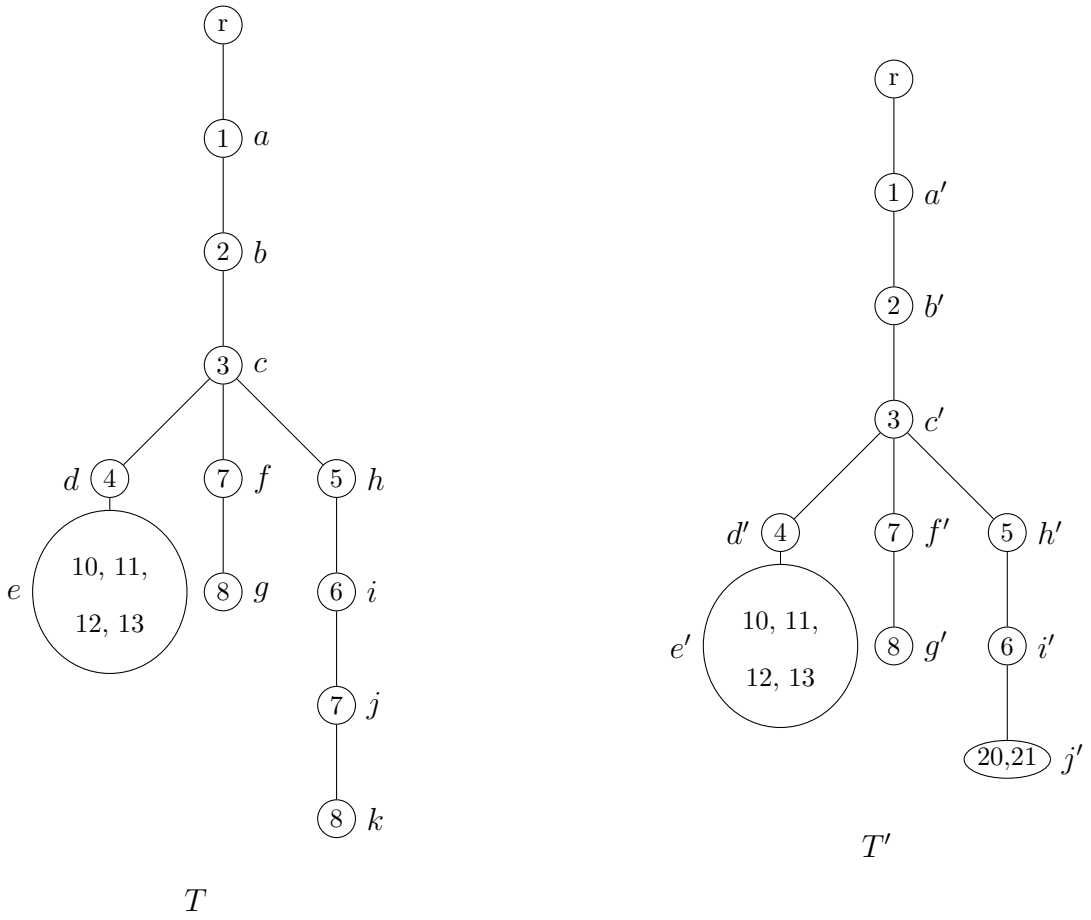


Figure 4.8: A PCTP between T and T' aligns their paths so that every node in a matched pair corresponds to a node with the same mutation content, except for nodes j, k, and j'.

Remark 2. *In some cases such as Example 3, two pairwise consistent tree partitions may differ only in the order of the same pairs. These redundant PCTPs could be filtered out by introducing a consistent ordering, such as sorting them alphabetically or alphanumerically by node names—for instance, by using the first node of each path.*

Definition 13 (Common Node)**.** *Let T and T' be two clonal trees, and let (P, P') be a pairwise consistent tree partition of them, where $P = \{T_0, \ldots, T_k\}$ and $P' = \{T'_0, \ldots, T'_{k'}\}$. For each i, nodes v and v' on T_i and T'_i, respectively, are said to be a common nodes if $\mathcal{MC}_{T_i}(v) = \mathcal{MC}_{T'_i}(v')$.*

Definition 14 (Node Worth)**.** *Let T and T' be two clonal trees, and let (P, P') denote a pairwise consistent tree partition, where $P = \{T_0, \ldots, T_k\}$ and $P' = \{T'_0, \ldots, T'_{k'}\}$. For each i such that the intersection L_i of $L(T_i)$ and $L(T'_i)$ is not empty, the* node worth *of the pair (T_i, T'_i) is defined as:*

$$\mathcal{NW}(T_i, T'_i) = \min\left(n_i, n'_i\right),$$

where n_i and n'_i denote the number of nodes in T_i and T'_i, respectively, that have at least one label in L_i and are not common nodes.

Remark 3. *For the distance d_{CED}, we count the* affected nodes *for each operation individually. This means that a single node may be counted multiple times if it is affected by multiple operations. For example, deleting a label ℓ and its descendant label "ℓ" via two separate operations (which can happen when the mutation-loss deletion is performed first), would result in double-counting the nodes below "ℓ", despite their labels or content not actually changing. To avoid this, such deletions must be handled by a single operation. In contrast, for δ_{CED}, a node is counted only once, and only if its mutation content differs between the source and target trees. The weight is not computed per operation, but by comparing the resulting paired paths. Therefore, we do not to handle mutation loss as a side effect of an operation.*

Definition 15 (Weight of Pairwise Consistent Tree Partitions)**.** *Let T and T' be two clonal trees, and let (P, P') be a pairwise consistent tree partition of them, where $P = \{T_0, \ldots, T_k\}$ and $P' = \{T'_0, \ldots, T'_{k'}\}$. Suppose that K is the largest index such that for each $1 \leq i \leq K$, the paths T_i and T'_i share at least one label.*

The weight $W_{PP'}$ of (P, P') is defined as the sum of:

1. *the summation of the node worths of $(T_1, T'_1), \ldots, (T_K, T'_K)$*

$$\sum_{i=1}^{K} \mathcal{NW}(T_i, T'_i),$$

2. *the minimum number of operations needed to transform each T_i into T'_i for $i \leq K$, and*

3. *the minimum number of deletion and insertion operations required to delete $T_{K+1}, \ldots, T_k$ and insert $T'_{K+1}, \ldots, T'_{k'}$.*

Definition 16 (CED Semi-Metric)**.** *Let T and T' be two clonal trees. The CED semi-metric $\delta_{CED}(T, T')$ is defined as*

$$\delta_{CED}(T, T') = \min_{(P,P') \in \mathcal{P}} W_{PP'},$$

where $\mathcal{P}$ denotes the set of all possible pairwise consistent tree partitions of T and T', and $W_{PP'}$ is the weight of (P, P').

Lemma 2. *Let T and T' be two clonal trees, and let (P, P') be a PCTP for (T, T') with $W_{PP'} = t$. Then, (P', P) is a PCTP for (T', T) with $W_{P'P} = t$.*

Proof. All conditions in Definition 12 are symmetric, which implies that (P', P) is a PCTP for (T', T). Let K be the largest index such that for each $1 \leq i \leq K$, the paths T_i and T'_i have at least one label in common. The weight is also preserved because:

- The node worth of (T_i, T'_i) is equal to the node worth of (T'_i, T_i) for each $i \leq K$.
- For each $i \leq K$, transforming T_i into T'_i requires the same number of operations as transforming T'_i into T_i, since the operations are reversible, as previously discussed.
- The total number of required deletions and insertions for paths that do not share any labels (T_i and T'_i where $i > K$) is preserved, because the labels that are deleted when transforming T into T' must be inserted when transforming T' into T, and vice versa.

This completes the proof. □

Proposition 4. *The distance method δ_{CED} is a semi metric.*

Proof. Let T, T', and T'' be three clonal trees. To prove that the Clonal Evolution Distance δ_{CED} is a semi-metric, we must show that it satisfies the following axioms:

1. Non-negativity: $\delta_{CED}(T, T') \geq 0$

   Proof. For each pairwise consistent tree partition (P, P'), the weight $W_{PP'}$ is defined as a non-negative integer. Therefore, $\delta_{CED}(T, T') \geq 0$. □

2. $\delta_{CED}(T, T') = 0$ if and only if $T = T'$

   Proof. ($\Rightarrow$) If $\delta_{CED}(T, T') = 0$, then no operations are needed to transform T into T'. Thus, there exists a pairwise consistent tree partition (P, P') such that $P = P'$. Moreover, no node is counted in the weight, implying all nodes preserved their ancestral histories. Hence, $T = T'$.

   ($\Leftarrow$) If $T = T'$, define P as a partition of T in which T_1 corresponds to the path containing the largest number of labels. Then let T_2 be the path with the next largest number of labels, and continue this process until all paths are included in P. Let $P' = P$. Then (P, P') is a PCTP with $W_{PP'} = 0$. Therefore, $\delta_{CED}(T, T') = 0$. □

3. Symmetry: $\delta_{CED}(T, T') = \delta_{CED}(T', T)$

   Proof. It is a direct result of Lemma 2. □

□

The following example shows that the distance method δ_{CED} does not satisfy the triangle inequality.

Example 4. *Consider the trees in Figure 4.9. Transforming T into T', since T has only one path $a \hookrightarrow f$, it can either be matched with $d' \hookrightarrow a'$ or $d' \hookrightarrow f'$. The first condition in the definition of PCTP breaks the tie. Hence, according to Remark 1, we have only one possible PCTP $(\{r, a \hookrightarrow f\}, \{r, d' \hookrightarrow a', e' \hookrightarrow f'\})$. Its weight is equal to 11, because two deletions are required to delete the labels 5 and 6, three rearrangements are required to fix the order of the remaining labels, all four remaining nodes are counted because their mutation content has changed, and finally, two insertion operations are required to reinsert the labels 5 and 6. Since this is the only possible pairwise consistent tree partition, we have $\delta_{CED}(T, T') = 11$.*

Similarly, transforming T into T'', we have only one possible PCTP and its weight is equal to 6, because it counts the nodes d, e, and f, since their ancestral history has changed, and three deletions are required. So, $\delta_{CED}(T, T'') = 6$.

Finally, the only PCTP between T'' and T' requires three insertions and nodes d'', e'', and f'' are common, which means $\delta_{CED}(T', T'') = 3$. Hence,

$$\delta_{CED}(T, T') \nleq \delta_{CED}(T, T'') + \delta_{CED}(T'', T').$$

Theorem 3. *Let $\mathcal{T}$ be the set of all clonal trees with n shared labels (including the root, involving no mutation loss) and n' distinct (non-shared) labels combined, where $n \geq 3$. Repeated labels are counted separately. Then the diameter of $\mathcal{T}$, denoted by $\mathrm{Diam}(\mathcal{T})$, which is the largest distance two trees in $\mathcal{T}$ can have, is equal to*

$$n' + 2n - 3.$$

Proof. Let T and T' be two given clonal trees with n shared labels and n' distinct labels. Let (P, P') be a pairwise consistent tree partition for (T, T'), where $P = \{T_0, \ldots, T_m\}$ and $P' = \{T'_0, \ldots, T'_{m'}\}$, and let M be the greatest index such that T_i and T'_i have some labels in

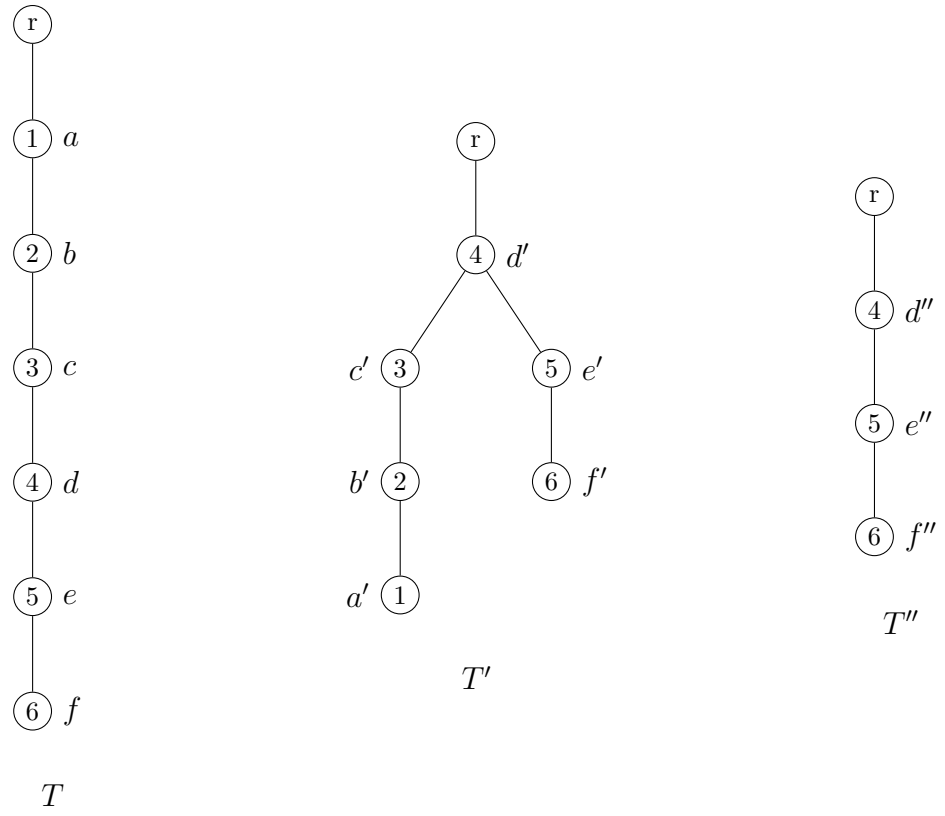


Figure 4.9: The triangle inequality does not hold for the trees T, T', and T''

common. Since n is not zero, according to the definition of pairwise consistent tree partition, $M \geq 1$.

Let T_i and T'_i have n_i shared labels and n'_i distinct labels. When transforming T_i into T'_i, the weight counts at most n_i for the node worth, at most $n_i - 1$ rearrangements to correct their positions along the path, and n'_i for deleting and inserting the distinct labels. When labels in common between both trees are not on matched paths, each contributes at most twice to the total weight, as they must be deleted and reinserted elsewhere.

Consequently, for each positive integer $i \leq M$, the weight of transforming T_i into T'_i is at most $n'_i + (n_i) + (n_i - 1)$ which means that $W_i = n'_i + 2n_i - 1$. Therefore,

$$\begin{aligned} \delta_{CED}(T, T') &\leq \textstyle\sum_{i=1}^{M} W_i + \sum_{i>M}^{m} |L(T_i)| + \sum_{i>M}^{m'} |L(T'_i)| \\ &= n' + 2(n-1) - M. \end{aligned}$$

Since $M \geq 1$, we get

$$\delta_{CED}(T, T') \leq n' + 2(n-1) - 1.$$

Trees T and T' were arbitrary, so

$$\text{Diam}(\mathcal{T}) \leq n' + 2n - 3.$$

Now, let T_0 be a path tree with n nodes labeled with r followed by the child node labeled with 1, to the last node labeled with $n-1$. Let T_0' have the same structure but with the labels arranged in reverse order (i.e., the root is followed by nodes labeled $n-1, \ldots, 1$). Let the n' non-shared labels to be distributed arbitrarily between T_0 and T_0' and placed randomly on them to get T and T', respectively. Below we see that the trees are already defined to achieve the maximum possible node worth before adding these labels, so the placement of distinct labels—whether on new nodes or existing ones—does not affect the result.

We claim that in this case,

$$\delta_{CED}(T, T') = n' + 2n - 3.$$

The distance value includes:

- n' operations for deleting the distinct labels from T and inserting the distinct labels of T',
- $n-2$ rearrangement operations to transform the common part of T into that of T', and
- $n-1$ nodes corresponding to the common part have their mutation content changed, so the node worth is $n-1$.

There is only one PCTP for (T, T'), so the total weight is

$$\delta_{CED}(T, T') = n' + (n-2) + (n-1) = n' + 2n - 3.$$

This completes the proof.

□

4.5 An Algorithm for the CED Semi-Metric δ_{CED}

In this section, we use several algorithms to generate all possible pairwise consistent tree partitions and compute their weights. Algorithm 1, which is the main algorithm, calls the function GenerateAllPCTPs on (T, T') to find all possible pairwise consistent tree partitions between them. Then, for each pair of resulting PCTP (P, P'), it calls the functions TotalNodeWorth, TotalRearrangementCost, and InsertionDeletionCost to compute the total node worth, the minimum number of required rearrangements, and the minimum number of required insertions and deletions, respectively.

The following example helps illustrate the logic of Algorithm 2 for generating all possible PCTPs.

Example 5. *Consider the trees in Figure 4.8. We know by the definition that $T_0 = r$, and $T'_0 = r$. Now we want to choose a pair of paths to asign to T_1, and T'_1. Our options for T_1 are $\mathcal{T}_1 = a \hookrightarrow e$, $\mathcal{T}_2 = a \hookrightarrow g$, and $\mathcal{T}_3 = a \hookrightarrow k$. Our options for T'_1 are $\mathcal{T}'_1 = a' \hookrightarrow e'$, and $\mathcal{T}'_2 = a' \hookrightarrow g'$, and $\mathcal{T}'_3 = a' \hookrightarrow j'$. First, we generate a table whose entries represent the labels that two paths have in common. The root does not appear in the table because it is already assigned in the partitioning. Repeated labels are assigned an additional index to distinguish them. For example, in Table 4.1, the first element of each index indicates which copy of the label in T is being referred to, and the second element indicates the corresponding copy of the label in T'. The copies of a label in a tree are numbered arbitrarily, but the numbers are fixed after they are assigned. This step is performed when Algorithm 2 initializes table B (as shown in Table 4.1).*

For each label appearing in this table, we store its origin node and the mutation content of that node.

	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$
$\mathcal{T}_1'$	1,2,3,4,10,11,12,13	1,2,3	1,2,3
$\mathcal{T}_2'$	1,2,3	1,2,3,$7_{1,1}$,$8_{1,1}$	1,2,3, $7_{2,1}$,$8_{2,1}$
$\mathcal{T}_3'$	1,2,3	1,2,3	1,2,3,5,6

Table 4.1: Each entry in this table represents the labels that the paths have in common.

The green color indicates that these labels in both trees originate from nodes with the same mutation content. According to Definition 12, the function TieBreak is called for Table 4.1 and the first elements of (P, P') after T_0 and T_0' should be $(\mathcal{T}_1, \mathcal{T}_1')$, since they share the largest number of labels. So $T_1 = \mathcal{T}_1$, and $T_1' = \mathcal{T}_1'$. Now, we need to update the table. Since $\mathcal{T}_1$ and $\mathcal{T}_1'$ are no longer needed, their corresponding column and row must be removed, and the labels of their nodes should also be eliminated from the remaining entries in the table. Similarly, we update the remaining paths $\mathcal{T}_2 = f \hookrightarrow g$, $\mathcal{T}_3 = h \hookrightarrow k$, $\mathcal{T}_2' = f' \hookrightarrow g'$, and $\mathcal{T}_3' = h' \hookrightarrow j'$. These steps are performed by the UpdateTable function.

	$\mathcal{T}_2$	$\mathcal{T}_3$
$\mathcal{T}_2'$	$7_{1,1}$, $8_{1,1}$	$7_{2,1}$, $8_{2,1}$
$\mathcal{T}_3'$		5, 6

Table 4.2: Updated table after removing row $\mathcal{T}_1'$ and column $\mathcal{T}_1$.

In the updated table, the number of shared labels is tied, so we proceed to the next condition in Definition 12. The options $(\mathcal{T}_2, \mathcal{T}_2')$ and $(\mathcal{T}_3, \mathcal{T}_3')$ are valid PCTPs because each pair shares two labels with another unpaired path, none of those labels occurs on a node with

preserved historical content, and neither pair requires any rearrangement. The algorithm continues for both options. Let us continue with the choice $T_2 = \mathcal{T}_2$ and $T'_2 = \mathcal{T}'_2$ and update the table by removing the chosen paths from the table and removing their labels from the remaining cells. The paths $\mathcal{T}_3$ and $\mathcal{T}'_3$ remain unchanged since they do not share a node with the removed paths. Table 4.3 is the last updated table. So the final paths in the partitions are $T_3 = \mathcal{T}_3$ and $T'_3 = \mathcal{T}'_3$.

	$\mathcal{T}_3$
$\mathcal{T}'_3$	5, 6

Table 4.3: Final updated table after removing $\mathcal{T}_2$, $\mathcal{T}'_2$.

After the final update, all table entries become empty, and every path is included in the partitions; therefore, no path remains to be processed by the function HandleUnmapped. Therefore, this partition is added to the set $\mathcal{P}$. If we end up having paths with no labels in common, or if one of the trees has more paths than the other, these paths will be matched to $\emptyset$ (the path $\mathcal{T}$ being matched with $\emptyset$ means that if $\mathcal{T}$ is the i-th element of P, then $\emptyset$ would be the i-th element of P'), where $\emptyset$ represents an empty path (i.e., a path with no nodes or edges), ensuring that P and P' have the same number of elements. Note that the empty entries are only used to support the following algorithms.

Algorithm 1 Compute $\delta_{CED}(T, T')$

Input: Two clonal trees T and T'

output: $\delta_{CED}(T, T')$

function DELTACED(T, T')

    $\mathcal{P} \leftarrow$ AllPCTPs(T, T')

    $\delta \leftarrow \infty$

    for all $(P, P') \in \mathcal{P}$ **do**

        $w_1 \leftarrow$ TotalNodeWorth(P, P')

        $w_2 \leftarrow$ TotalRearrangementCost(P, P')

        $w_3 \leftarrow$ InsertionDeletionCost(P, P')

        $W_{PP'} \leftarrow w_1 + w_2 + w_3$

        $\delta \leftarrow \min(\delta, W_{PP'})$

    end for

    return δ

end function

Algorithm 2 AllPCTPs

function AllPCTPs(T, T')
  $\mathcal{P} \leftarrow \emptyset$ ▷ Set of all valid partitions
  $n \leftarrow$ number of paths (leaves) in T
  $n' \leftarrow$ number of paths in T'
  Initialize $B = [b_{ij}]$ as an $n \times n'$ table of shared label sets (see Appendix A.2)
  Initialize empty arrays P and P', each of size $n + n'$
  RecursivePartitioning($B, T, T', P, P', \mathcal{P}, 0$)
  return $\mathcal{P}$
end function
function RecursivePartitioning($B, T, T', P, P', \mathcal{P}, t$)
  if all entries in B are empty **then**
    HandleUnmapped(B, P, P', t) ▷ Assign unmatched paths to $\emptyset$ (Appendix A.3)
    Add $(P[0{:}t], P'[0{:}t])$ to $\mathcal{P}$
    return
  end if
  $E \leftarrow$ TieBreak(B) ▷ Apply tie-breaking rules to select best (i, j) pairs (Algorithm 9)
  for each selected entry $(i, j) \in E$ **do**
    $P_{\text{new}} \leftarrow$ copy of P; $P'_{\text{new}} \leftarrow$ copy of P'
    $P_{\text{new}}[t] \leftarrow T_i$; $P'_{\text{new}}[t] \leftarrow T'_j$
    $B' \leftarrow$ copy of B
    UpdateTable(B', i, j) ▷ Remove shared labels and their nodes (Appendix A.4)
    RecursivePartitioning($B', T, T', P_{\text{new}}, P'_{\text{new}}, \mathcal{P}, t + 1$)
  end for
end function

Next, Algorithm 3 invokes the function ComputeNodeWorth for each pair (T_i, T'_i) that shares at least one label, in order to compute their node worth and accumulate the total. The function ComputeNodeWorth calls FindCommonNodes, which identifies the common nodes between the two subtrees by comparing the mutation content of their nodes in the original trees.

Algorithm 3 TotalNodeWorth(P, P')

Input: $P = \{T_0, \ldots, T_k\}$ and $P' = \{T'_0, \ldots, T'_{k'}\}$ (PCTP of T, T')
Output: $\mathcal{TNW}(P, P')$, the total node worth
$totalWorth \leftarrow 0$
for each (T_i, T'_i) in P and P' matched by the pairwise consistent tree partitioning **do**
    if T_i and T'_i share at least one label **then**
        $worth \leftarrow$ ComputeNodeWorth(T_i, T'_i)
        $totalWorth \leftarrow totalWorth + worth$
    end if
end for
return $totalWorth$

Algorithm 4 ComputeNodeWorth(T_i, T_i')

Input: Paths T_i and T_i'

Output: $\mathcal{NW}(T_i, T_i')$, the node worth of the pair

CommonNodes $\leftarrow$ FindCommonNodes(T_i, T_i')

$L_i \leftarrow L(T_i) \cap L(T_i')$

$n_i \leftarrow 0$

for each node a in T_i **do**

    if $a \notin$ CommonNodes **and** a has at least one label in L_i **then**

        $n_i \leftarrow n_i + 1$

    end if

end for

$n_i' \leftarrow 0$

for each node b in T_i' **do**

    if $b \notin$ CommonNodes **and** b has at least one label in L_i **then**

        $n_i' \leftarrow n_i' + 1$

    end if

end for

$\mathcal{NW}(T_i, T_i') \leftarrow \min(n_i, n_i')$

return $\mathcal{NW}(T_i, T_i')$

Algorithm 5 FindCommonNodes(T_i, T_i')

Input: Subtrees T_i and T_i'
Output: CommonNodes, the set of common nodes between T_i and T_i'
CommonNodes $\leftarrow \emptyset$
for each node a in T_i **do**
    for each node b in T_i' **do**
        if $\mathcal{MC}(a) = \mathcal{MC}(b)$ **then**
            Add a and b to CommonNodes
            break
        end if
    end for
end for
return CommonNodes

We need the following definition to continue:

Definition 17 (Relation Matrix)**.** *Given two multilabeled trees,* ***Source*** *and* ***Target****, each consisting of a single path, let* $L = \{\ell_1, \ell_2, \ldots, \ell_n\}$ *be the set of labels common to both trees (note that the total set of labels in each tree may differ). We define a square matrix* M *of size* $n \times n$ *such that:*

$$M_{ij} = \begin{cases} 0 & \text{if the ancestral-descendant relation between } \ell_i \text{ and } \ell_j \text{ is the same in } \textbf{\textit{Source}} \text{ and } \textbf{\textit{Target}} \\ 1 & \text{otherwise} \end{cases}$$

We call this matrix the relation matrix *of the pair* (***Target***, ***Source***)*.*

Algorithm 6 sends each pair of paths (T_i, T_i') in (P, P') that share at least two labels to the function MinNumberOfRearrangements, which computes the minimum number of rearrangement operations required to transform T_i into T_i'. If fewer than two labels are in

common, then no rearrangement is necessary. The algorithm then sums all the returned values.

Algorithm 7 constructs the relation matrix M of its input source and target paths which belong to (P, P'). To be precise, for the (i, j)-th entry of the matrix for $(\mathit{Source}, \mathit{Target}) \in (P, P')$, it checks whether the label ℓ_i is in the set of labels on an ancestor of the node containing ℓ_j, or vice versa, or if they both belong to the same node in the tree **Source**. Then it checks the same relationship for ℓ_i and ℓ_j in the tree **Target**. If the relation is the same for both paths, the (i, j)-th entry is set to 0; otherwise, it is set to 1. The relation matrix is constructed only for the labels that *Source* and *Target* share. Labels that appear in *Source* but not in *Target* must be deleted, and labels that appear in *Target* but not in *Source* must be inserted. These insertion and deletion steps are handled separately by Algorithm 8, as we explain later.

Lemma 3. *Let Source and Target be two multilabeled trees, each consisting of a single path and sharing the same set L of n labels. The* 1*-entries of the relation matrix above the diagonal (upper triangle) define a bipartite graph, and any minimum vertex cover of this graph specifies a minimum-sized sequence of rearrangements that transforms Source into Target.*

Proof. First, we claim that if there exists a subset $L' \subseteq L$ such that zeroing out all entries in the rows and columns corresponding to the labels in L' makes the matrix entirely zero, then it is possible to transform *Source* into *Target* through rearrangements that move exactly the labels in L'. Let $L' = \{\ell_1, \cdots, \ell_m\}$ be such subset of L. *Source* is updated to delete all the labels in L'. In the updated *Source* denoted by Source^u, all labels have the same ancestor–descendant relationships as in *Target*, since we assumed all 1-entries belong to rows or columns related to the labels in L' which no longer exist.

We prove by induction that for each $1 \leq i \leq m$, there is a rearrangement R_i such that, once applied to $\mathit{Source}\langle R_1, \cdots, R_{i-1}\rangle$, all label pairs in *Source* have the same ances-

tor–descendant relationships as in *Target*, except for pairs where at least one label belongs to $\ell_{i+1}, \ldots, \ell_m$.

Because the ancestor–descendant relation is transitive, there exists a position in $Source^u$ where ℓ_1 can be inserted while preserving correct relationships with all other labels. A rearrangement operation corresponds to this deletion and insertion of ℓ_1. Each rearrangement in $Source^u$ maps to a rearrangement in the original *Source* as follows: if the target node in $Source^u$ already exists, the rearrangement is identical in the original *Source*; if the target node is new, the corresponding target node in the original *Source* is a new node placed under the node corresponding to the parent of the target node in $Source^u$. If the new node must be inserted above all nodes in $Source^u$, and n_0 is the node in $Source^u$ closest to the root, then the new node in *Source* should be inserted as the parent of n_0. Note that all nodes in $Source^u$ correspond to a node in *Source*.

Let R_1 denote the rearrangement for ℓ_1. Update $Source^u$ again by inserting ℓ_1 on its target position.

Assume the claim is true for R_{k-1} with $k - 1 < m$. In other words, after applying rearrangements $R_1, \ldots, R_{k-1}$, all ancestor-descendant relationships involving only labels outside of $\ell_k, \ldots, \ell_m$ match those in *Target*. We proceed to prove the claim for R_k.

For $i = k$, again according to the transitivity of ancestor-descendant relation, there exists a position for ℓ_i in the path updated by deleting $\ell_1, \ldots, \ell_m$ and reinserting $\ell_1, \ldots, \ell_{i-1}$ such that ℓ_i satisfies the correct ancestor-descendant relationships with all labels in the updated *Source*. Let R_i be the corresponding rearrangement of ℓ_i in the original *Source*. This completes the proof for R_k.

Therefore, applying the sequence of rearrangements $S = R_1, \ldots, R_m$ to *Source* yields a path in which all labels satisfy the ancestor-descendant relationships defined by *Target*. Thus, this sequence transforms *Source* into *Target*. This completes the first part of the proof.

Since the relation matrix M is symmetric, it suffices to zero-out all 1-entries by considering only those in the upper triangle, i.e., entries where $M_{ij} = 1$ for $i < j$. Define two sets, *Row* and *Column*, so that for every pair (i, j) with $M_{ij} = 1$ and $i < j$, the label ℓ_i is included in *Row* and ℓ_j is included in *Column*. When a label appears in both sets, we can distinguish them via indexing. Construct the bipartite graph $G = (Row, Column, E)$, where E consists of all edges (i, j) such that $M_{ij} = 1$ and $i < j$. By definition, each edge corresponds to a 1-entry in the upper triangle, and each node to a label in L. Let L' denote a vertex cover for G. Since each edge is incident to at least one label in L', removing all labels in L', i.e., zeroing out their associated rows or columns, eliminates every 1-entry in the upper triangle of M. By combining this with the symmetry of the relation matrix and the result from the first part of the proof, it follows that L' yields a sequence of rearrangements, of cardinality $|L'|$, transforming *Source* into *Target*.

On the other hand, in an optimal sequence S, each label is moved at most once, so we can identify each rearrangement in S by the label it moves. Let $L'' = \{\ell'_1, \ldots, \ell'_k\}$ denote the set of labels corresponding to S. Zeroing out the rows and columns associated with L'' reduces the matrix to all zeros. Therefore, L'' corresponds to a vertex cover of G.

Therefore, finding the minimum number of rearrangement operations to transform *Source* into *Target* is equivalent to finding a minimum vertex cover for G. The graph G is constructed by Algorithm 7. □

Proposition 5. *Algorithm 7 introduces a sequence of operations that transforms the path Source into the path Target, and the length of this sequence is the minimum possible among all sequences that can achieve this transformation.*

Proof. By Lemma 3, a minimum vertex cover for the graph built by Algorithm 7 introduces a minimum-sized sequence of rearrangement operations. By Kőnig's theorem [Kőnig, 1931], in every bipartite graph, the size of the minimum vertex cover equals the size of the max-

imum matching. We compute the maximum matching using the Hopcroft–Karp algorithm [Hopcroft and Karp, 1973].

In Algorithm 7, the procedure MaximumMatching refers to this implementation. The minimum vertex cover is then constructed from the matching via a standard traversal (see, for example, [Lau et al., 2012]). This completes our proof. □

For the partition found in Example 5, all entries of the tree relation matrices are zero, which means no rearrangement is required.

Algorithm 6 TotalRearrangementCost(P, P')

Input: Two partitions $P = \{T_1, \ldots, T_n\}$ and $P' = \{T'_1, \ldots, T'_n\}$
Output: Total number of rearrangements over all matched pairs
totalCost $\leftarrow 0$
for $i \leftarrow 1$ to n **do**
    $L_i \leftarrow$ set of labels in T_i
    $L'_i \leftarrow$ set of labels in T'_i
    $L_c \leftarrow L_i \cap L'_i$
    if $|L_c| \geq 2$ **then**
        $r \leftarrow$ MinNumberOfRearrangements(T_i, T'_i)
        totalCost $\leftarrow$ totalCost + r
    end if
end for
return totalCost

Algorithm 7 MinNumberOfRearrangements(Source, Target)

Input: Two paths *Source* and *Target* with n common labels for some $n \geq 2$

Output: The minimum number of rearrangements needed to transform *Source* into *Target*

Map the common labels to the set $\{1, \cdots, n\}$ arbitrarily (bijection).

Construct the relation matrix M of size $n \times n$ for the n labels that *Source* and *Target* have in common according to the mapping above.

Initialize two empty sets: *Row* and *Column*.

Initialize an empty set *Edges*

for $i = 1$ to n **do**

    for $j = i + 1$ to n **do**

        if $M_{ij} = 1$ **then**

            Append the label corresponding to i to *Row*.

            Append the label corresponding to j to *Column*.

            Append the edge (i, j) to *Edges*, treating $i \in$ *Row* and $j \in$ *Column* as its endpoints.

        end if

    end for

end for

$M \leftarrow$ MaximumMatching(*Row*, *Column*, *Edges*)

$m \leftarrow |M|$

return m

Finally, Algorithm 8 compares pairs (*Source*, *Target*) in a pairwise consistent tree partition and counts the labels that appear only in *Source* (which need to be deleted) and those that appear only in *Target* (which need to be inserted). If *Source* or *Target* in a pair is paired with

the empty set, then all of its labels are counted. The next example shows how to compute this number for the trees in Example 5.

Example 6. *Consider the trees and the partition introduced in Example 5. The size of the symmetric difference for* (T_1, T_1') *as well as* (T_2, T_2') *is zero. For* (T_3, T_3')*, labels* 7 *and* 8 *belong only to* T_2*, and labels* 20 *and* 21 *belong only to* T_2'*. Therefore, Algorithm 8 counts a total of four operations.*

Since we have already seen earlier in this section that both node worth and minimum number of rearrangements are equal to zero, and the only two pairwise consistent tree partitions for these trees differ only in the order of the same pairs, we conclude that $\delta_{CED}(T, T') = 4$*.*

Algorithm 8 InsertionDeletionCost

1: **function** InsertionDeletionCost($P = \{T_1, \ldots, T_K\}, P' = \{T_1', \ldots, T_K'\}$)
2: **Input:** A pairwise consistent tree partition
3: **Output:** Total number of required deletion and insertion operations
4: $cost \leftarrow 0$
5: **for** $i \leftarrow 1$ to K **do**
6: $labels\_in\_T_i \leftarrow$ labels in T_i
7: $labels\_in\_T_i' \leftarrow$ labels in T_i'
8: $symmetricDiff \leftarrow (labels\_in\_T_i \setminus labels\_in\_T_i') \cup (labels\_in\_T_i' \setminus labels\_in\_T_i)$
9: $cost \leftarrow cost +$ |symmetricDiff|
10: **end for**
11: **return** *cost*
12: **end function**

Theorem 4. *The value computed by Algorithm 1 for the clonal trees* T *and* T' *is equal to* $\delta_{CED}(T, T')$*.*

Proof. Let $\mathbf{P} = (\{T_0, \ldots, T_n\}, \{T'_0, \ldots, T'_{n'}\})$ be a pairwise consistent tree partition for (T, T'), where $n \leq n'$, and let K be the smallest index such that T_K and T'_K do not have any label in common. Algorithm 2 returns a pairwise partition (P, P') such that

$$P = \{T_0, \ldots, T_{K-1}, S_1, S_2\}, \quad P' = \{T'_0, \ldots, T'_{K-1}, S'_1, S'_2\},$$

where:

- S_1 is an ordering of $T_K, \ldots, T_n$;
- S'_1 is a sequence of $n - K + 1$ empty sets, one for each unmatched path in S_1;
- S_2 is a sequence of $n' - K + 1$ empty sets, one for each unmatched path in S'_2;
- S'_2 is an ordering of $T'_K, \ldots, T'_{n'}$.

Each pair (T_i, T'_i) for $1 \leq i < K$ in $\mathbf{P}$ satisfies Definition 12, and the algorithm is deterministic and greedy by design, therefore the pairs are selected in the same order by the algorithm. The remaining paths, which are not selected by the main loop of the algorithm, are processed in the order described above by Algorithm A.3.

Now, we show that, for the given pairwise consistent tree partition, the total node worth, the minimum number of operations required to transform pairs that share at least one label, and the minimum number of insertions and deletions for the remaining pairs are all computed correctly by the algorithms described below.

According to Definition 14, node worth only considers those pairs (T_i, T'_i) that share at least one label in common. In other words, the node worth is computed only for the pairs where neither T_i nor T'_i is the empty set, as described in Algorithm 3. Since the paths and their order are exactly the same for the first $K - 1$ elements of both pairwise partitions, the NodeWorth computed by the algorithm for each corresponding pair matches the node

worth in $\mathbf{P}$. Consequently, the total node worth remains the same for both $\mathbf{P}$ and the pairwise tree partition computed by the algorithm.

The minimum number of operations is defined only for the first $K-1$ pairs and is composed of two components: the minimum number of rearrangements and the required deletions and insertions. Proposition 5 establishes that Algorithm 7 correctly computes the minimum number of rearrangements for each such pair. The deletion and insertion counts are then obtained using Algorithm 8, as previously described.

Algorithm 8 counts the symmetric difference of each pair, which corresponds to the number of deletions and insertions for the first $K-1$ pairs, all labels on the remaining paths of T (must be deleted), and all labels on the remaining paths of T' (must be inserted into T). Since these remaining paths are paired with the empty set in (P, P'), the symmetric difference is just the label set of the non-empty path, hence all their labels are accounted for by the symmetric difference of their corresponding pairs, regardless of their order. This completes the proof that the final value δ computed by Algorithm 1 is equal to $\delta_{CED}(T, T')$.

□

4.6 Time Complexity of Computing δ_{CED}

We analyze the time complexity of Algorithm 1 for computing $\delta_{CED}(T, T')$ under two separate cases: one where a unique pairwise consistent tree partition exists, and one where multiple pairwise consistent tree partitions must be explored because of ties.

Let:

- n be the maximum number of leaves in T and T',
- L be the maximum number of labels in any path from root to leaf on T and T'.

The tie-breaking process is invoked at most $\mathcal{O}(n)$ times—once for each partitioning step.

Each time it runs, the algorithm compares at most n^2 entries in table B to evaluate the hierarchy of tie-breaking conditions:

- Finding entries with the largest number of shared labels (line 2 of algorithm 9): $\mathcal{O}(n^2)$
- Comparing mutation contents (line 4 of algorithm 9): $\mathcal{O}(n^2)$
- Counting label occurrences elsewhere (line 6 of algorithm 9): $\mathcal{O}(n^2)$
- Matching mutation contents for elsewhere labels (line 8 of algorithm 9): $\mathcal{O}(n^2)$
- Computing MinNumberOfRearrangements for tied entries, each of which is $\mathcal{O}(L^3)$ as we will explain later in this section (line 10 of algorithm 9): Worst case $\mathcal{O}(n^2 \cdot L^3)$

Therefore, the total worst-case complexity of the tie-breaking phase over the full algorithm is:

$$\mathcal{O}(n^3 \cdot L^3)$$

Case 1: Unique Pairwise Consistent Tree Partition. Assume that all tie-breaking steps in Algorithm 9 resolve deterministically, and only one pairwise consistent tree partition (P, P') is generated.

Generating the Partition. Algorithm 2 generates a single pairwise consistent tree partition using a recursive table-based procedure. In the unique case, it processes at most n matchings, and each table update takes $\mathcal{O}(n^2)$ time. Thus, the complexity is:

$$\mathcal{O}(n^3)$$

Computing Cost Components.

- TotalNodeWorth:

  The time complexity of computing the total node worth is dominated by the repeated computation of mutation contents and their comparisons. However, this can be done in $\mathcal{O}(L)$ time, as the mutation contents are available by construction and the comparison of mutation contents can be done in linear time. The FindCommonNodes algorithm compares each node in a path with each node in the paired path, in $\mathcal{O}(L^2)$ time for each pair. The ComputeNodeWorth algorithm calls FindCommonNodes once, yielding $\mathcal{O}(L^2)$ time. Finally, the TotalNodeWorth algorithm sums over pairs with at least one common label, leading to a total time complexity of

$$\mathcal{O}\big(n \cdot L^3\big).$$

- TotalRearrangementCost:

  In the worst case, each matched pair with at least two common labels has $\mathcal{O}(L)$ labels in common. Consequently, Algorithm 7 constructs and processes an $L \times L$ relation matrix in the worst case. Matrix construction requires $\mathcal{O}(L^2)$ time. Computing the maximum matching via the Hopcroft–Karp algorithm for a graph with m edges and n' nodes takes $\mathcal{O}(\sqrt{n'}\, m)$ time. Since $m, n' \leq L^2$, the maximum matching can be computed in $\mathcal{O}(L^3)$ time. Consequently, determining the minimum number of rearrangements for a single pair requires $\mathcal{O}(L^3)$ time. Therefore, in the worst case across n pairs, the total number of required rearrangements is needs $\mathcal{O}(n \cdot L^3)$ time.

- InsertionDeletionCost:

  For each pair in the pairwise consistent tree partition, it compares label sets and computes the symmetric difference, each in $\mathcal{O}(L)$ time:

$$\mathcal{O}(n \cdot L)$$

Total Time Complexity (Unique Partitioning). Combining the above components, the total time complexity is:

$$\mathcal{O}(n^3 \cdot L^3) + \mathcal{O}(n^3) + \mathcal{O}(n \cdot L^3) + \mathcal{O}(n \cdot L^3) + \mathcal{O}(n \cdot L)$$

This simplifies to:

$$\mathcal{O}(n^3 \cdot L^3)$$

Case 2: Tie Between t Equal-Quality Options. Now suppose that at some point in the recursive partitioning procedure, the tie-breaking hierarchy fails to resolve ambiguity, and t equally optimal options must all be explored. Each of these options may lead to more ties as they continue pairing the remaining paths. This results in k complete pairwise consistent tree partitions that need to be evaluated.

Let $\mathcal{C}$ denote the time to evaluate a single pairwise consistent tree partition, as derived above.

If k distinct pairwise consistent tree partitions are generated, the total cost becomes:

$$\mathcal{O}(k \cdot \mathcal{C}) = \mathcal{O}(k \cdot n^3 \cdot L^3).$$

Remarks. In practice, the number of ties t and the number of resulting PCTPs is expected to be small. However, in the worst-case scenario, where at every step of finding a PCTP all possible pairs lead to a tie, k can be in $\mathcal{O}(n^{2n})$.

4.7 Behavior Analysis and Proofs of Concept for d_{CED} and δ_{CED}

We compared the CED distances against the following existing distance measures:

- CaSet (Common Ancestor Set distance) [DiNardo et al., 2020]: compares the ancestor-descendant relationships between clonal trees.

- Disc (Distinctly Inherited Set Comparison distance) [DiNardo et al., 2020]: counts the sets of mutations that are distinctly inherited in different lineages.

- MLTD (multi-labeled tree dissimilarity) [Karpov et al., 2019]: uses an edit-distance-based method to transform two multi-labeled trees into a common tree; the distance is then calculated as the number of labels in each tree minus twice the number of labels in the maximum common tree.

- MP3 (Triplet-based similarity score) [Ciccolella et al., 2021a]: for every set of three mutation labels, MP3 compares the labels of the minimal subtrees containing them in each tree.

Distance values were computed manually using the formulas provided in the references.

An important property of a good distance measure is that, for highly similar trees (label-wise), it should distinguish topological differences. Figure 4.10 shows that the CED distances satisfy this property. To make MP3 [Ciccolella et al., 2021a] (bounded by 1) comparable with distance measures, we convert it into a distance by subtracting its value from 1. In addition, we normalize the CED distances using the diameter, as presented in Proposition 3 and Theorem 3.

Let tree T in Figure 4.10 be the reference (true) tree. The other trees are compared to T. Tree T_1 is a subtree of T while tree T_2 is not. The CED metric and the semi-metric correctly capture this distinction. MP3, and DISC [DiNardo et al., 2020] fail to differentiate between these cases, returning identical distances for both pairs of trees. Although MLTD [Karpov et al., 2019] and CASet [DiNardo et al., 2020] assigns a smaller distance to (T, T_1), they

reach their maximum at (T, T_2), unlike the CED, where the maximum is reached when T is compared to a tree with only the root node (data not shown) .

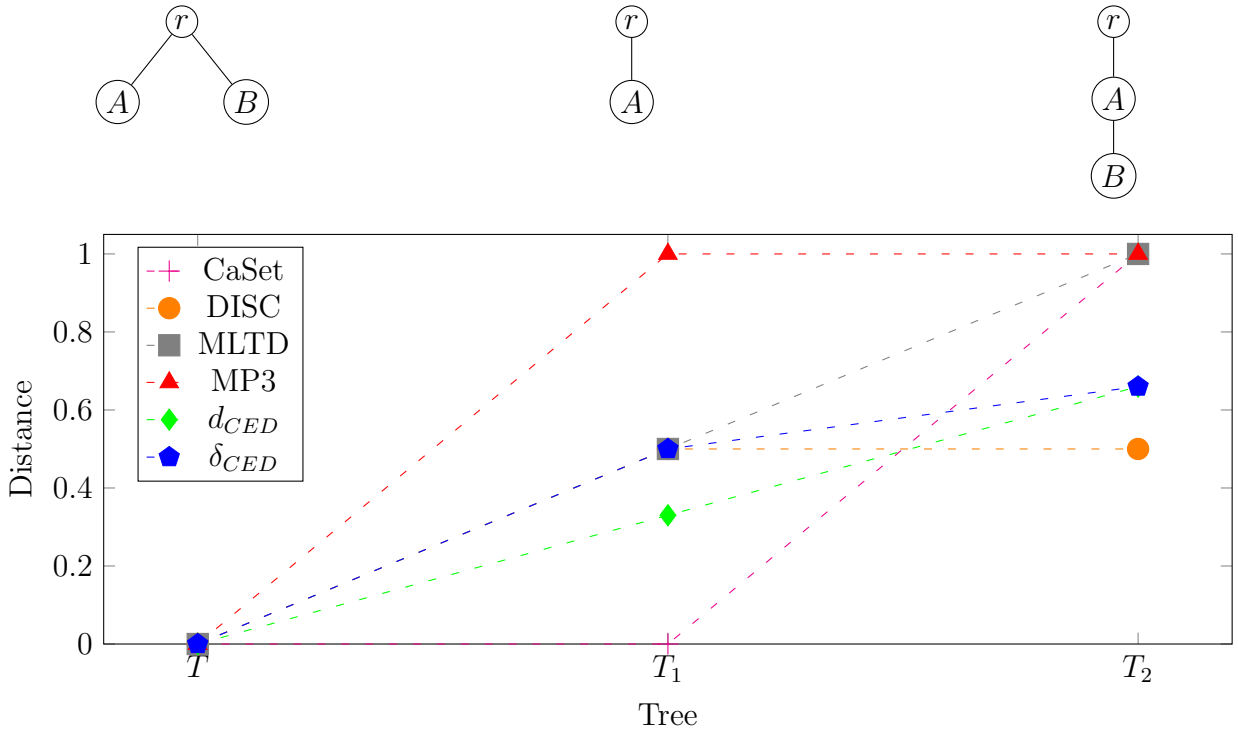


Figure 4.10: Comparison of distance measures reflecting the impact of dataset origin and topology of trees. Tree T is the reference tree, and the distances from T to the other trees T' and T'' are shown.

The next important feature of distance methods corresponds to the following observation: When a clone has an extra or missing mutation compared to a reference tree, all its descendants inherit that error. Consequently, changes in nodes with many descendants should contribute more to the distance than changes in nodes with fewer descendants. Figure 4.11 uses trees with the same set of labels to illustrate this property for several distance methods, including the CED distances. The only distance which fails to reflect this property is MLTD.

Some articles (for example, see [DiNardo et al., 2020]) propose a different test to illustrate the effects of changes closer to the root. They take a tree consisting of a single path and move its only leaf higher in the tree, attaching it as a leaf child of nodes closer to the root. The

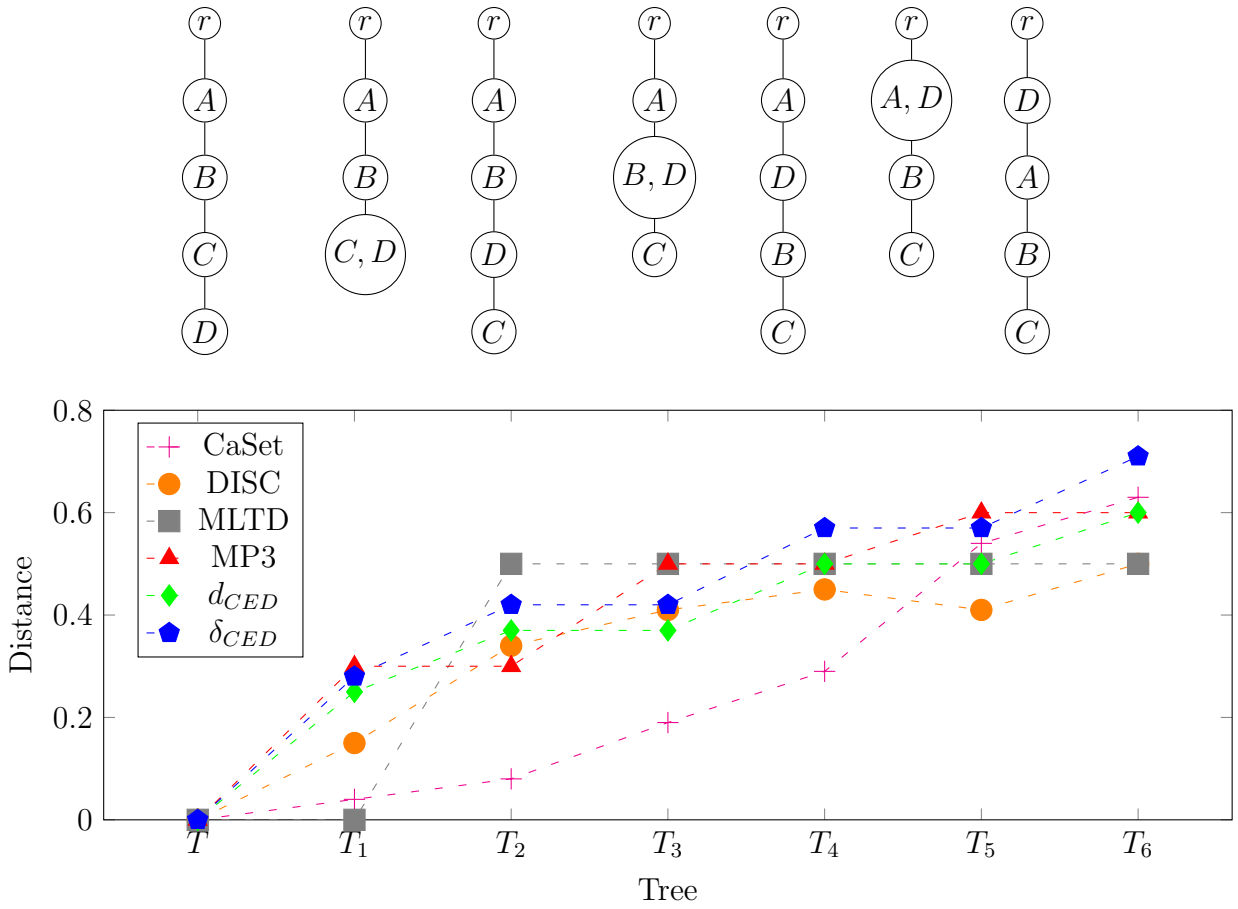


Figure 4.11: Effects of position of changes relative to the root. Tree T is the reference, and the other trees are compared against it.

authors expect the distance to increase. However, no matter how high we attach the leaf, it remains a leaf, and the changes only affect one clone. Therefore, we expect the distance to remain the same, as is the case with the CED distances.

Next, we compare the CED distance with the only other method (MP3) which allows mutation loss, to see how each of them respond to mutation loss. The reference tree in Figure 4.12 is T which does not acquire or lose mutation A. The other trees all acquire A at the first stage of the tumor, and lose it somewhere down the path. This comparison allows us to observe the importance of treating mutation loss differently, because when mutation loss happens closer to the acquisition, its descendants have similar mutation content to their

copy in the reference tree. Consequently, we should observe smaller distances for the cases where mutation loss happens closer to the root.

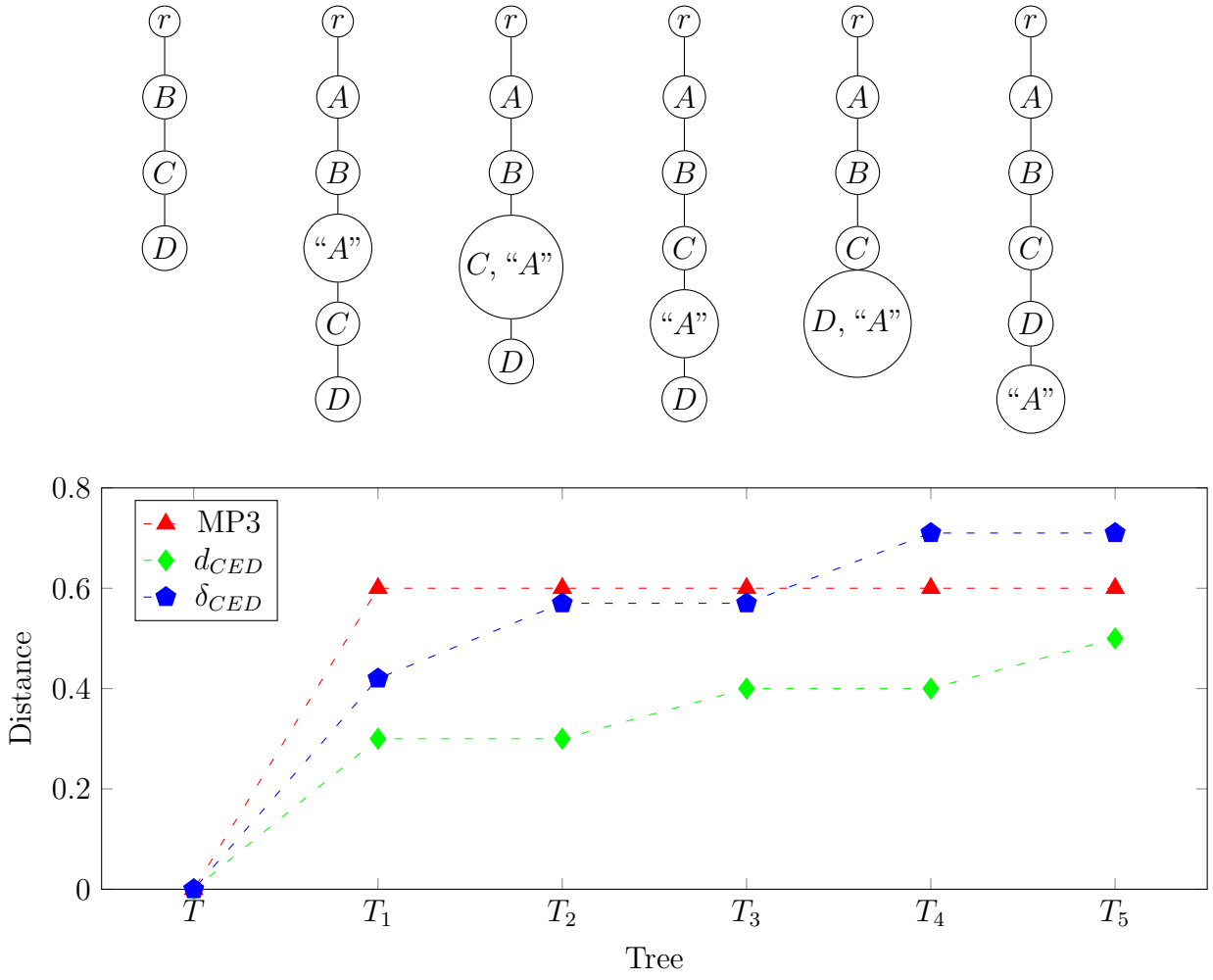


Figure 4.12: Effects of position of mutation loss relative to the root. Tree T is the reference, and the other trees are compared against it.

Another approach to see the effects of the position of changes relative to the root is shown in Figure 4.13. In this case, each of T_1 to T_6 contains an additional mutation E that is absent in the reference tree T. A comparison of Figure 4.12 and Figure 4.13 shows that MP3 treats mutation loss as the acquisition of a new mutation.

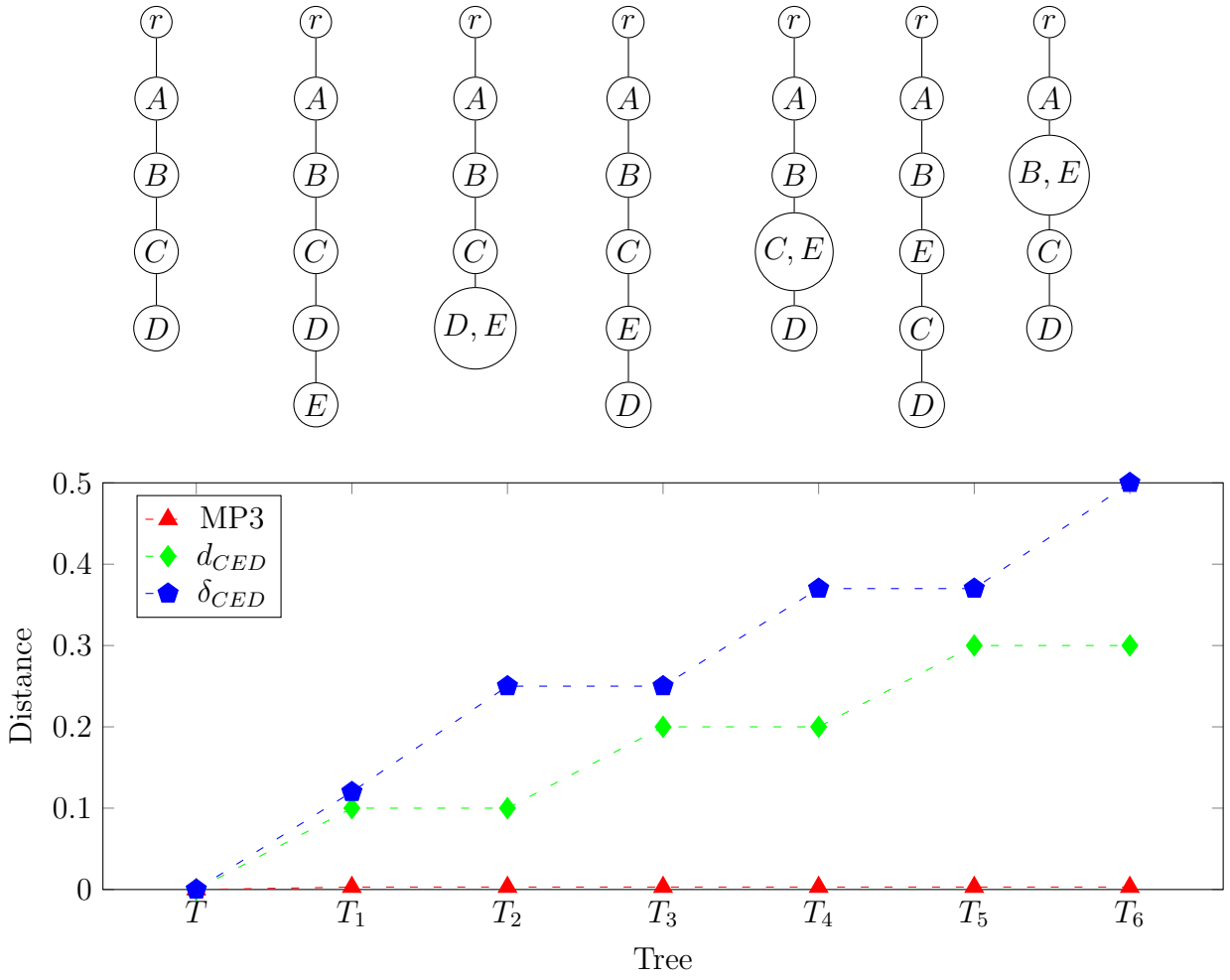


Figure 4.13: Effects of position of an extra mutation relative to the root. Tree T is the reference, and the other trees are compared against it.

Chapter 5

Conclusion

5.1 Summary

In this thesis we have presented the development, analysis, and theoretical foundation of the *Clonal Evolution Distances* (CED). The CED distances are novel measures designed to compare clonal trees derived from cancer sequencing data under realistic models of tumor evolution. The work began with a detailed review of the biological and computational context in which clonal trees arise. It focused on the evolutionary processes underlying tumor growth, the structural properties of clonal trees, and the limitations of traditional phylogenetic comparison methods in this domain.

Tumor evolution, unlike species evolution, occurs within an individual and involves heterogeneous populations of genetically distinct subclones that arise, expand, and sometimes vanish under selective pressures. The representation of this process as clonal trees symbolizes structural and temporal information about mutation acquisition and loss along evolutionary lineages. However, existing tree distance measures typically assume that each mutation occurs exactly once and is never lost (the Infinite Sites Assumption, ISA). Some of them assume further that all trees being compared contain identical label sets. These assumptions

are regularly violated by real tumor evolution, as revealed by high-resolution methods such as single-cell sequencing.

Recognizing this mismatch between existing comparison methods and biological reality, we established the need for a metric capable of directly handling clonal trees that may contain repeated occurrences of the same mutation, experience mutation losses, or differ in their mutation sets. In the literature review, we analyzed multiple types of prior work, from general-purpose phylogenetic distances such as Robinson–Foulds, Nearest Neighbor Interchange, Subtree Prune and Regraft, and Tree Bisection and Reconnection, to cancer-specific extensions for clonal trees like CASet, DISC, MLTD, Path, Parent–Child, Ancestor–Descendant, and Bourque distances. While these measures vary greatly in computational complexity, flexibility, and biological interpretability, none simultaneously support mutation loss, repeated mutation acquisition, arbitrary label sets, and a biologically grounded weighting of differences. Measures that relax the ISA often lack the resolution to discriminate larger-scale structural differences or fail to account for complex evolutionary changes. This identified gap motivated us to formulate a new distance metric adapted for clonal evolutionary analysis.

The core contribution, the CED metric, is defined through three biologically motivated edit-style operations: *deletion*, *insertion*, and *rearrangement* of mutation labels within the tree. Unlike traditional tree edit distances, CED assigns each operation a weight equal to the number of nodes whose label set or history content changes as a result of that operation. This reflects the impact on all descendant subclones. Deletion removes a mutation from a node (and, if necessary, the node itself), and triggers changes to the label set of any descendant nodes recording a mutation loss for that mutation. Insertion reverses this process by adding a mutation to the label set of an existing node or by introducing a new node labeled with the mutation. Rearrangement moves a mutation between nodes—including moving it into a new node. These operations are subject to Dollo's Law, which prohibits a mutation from

reappearing along the same lineage once lost. The CED restricts rearrangements to "good" rearrangements, in which the label is relocated to a node in the source tree that either has no other labels, or already contains at least one label that co-occurs with it in the target tree. The CED distance between two trees is defined as the minimum total weight among all sequences of the CED operations converting one tree into the other.

We proved that CED satisfies all metric axioms: non-negativity, identity, symmetry, and the triangle inequality. Furthermore, using a reduction to the unordered tree edit distance, we established that computing the exact CED value (denoted d_{CED}) is NP-hard even for restricted classes of trees. This highlights a need for tractable approximation methods in large-scale or high-complexity trees. An analysis of the metric's behaviour under label set constraints produced exact formulas, $2(n-1)$, for its diameter for clonal trees with less than or equal to n arbitrary labels.

In response to the computational complexity of CED metric computation, we proposed the δ_{CED} *semi-metric*, which uses a path-matching strategy motivated by lineage preservation. The δ_{CED} distance decomposes each tree into a set of subtrees, each consisting of a single path and then pairs paths from the two trees according to a hierarchical set of criteria: first, maximizing the number of shared labels in each pair; second, preserving the mutation content of those shared labels; third, minimizing the intersection of labels between a paired path and the rest of the tree; and, when ties still occur, selecting the pairing that requires the fewest rearrangements to transform one path into the other. This approach prunes the vast space of possible transformation sequences, focusing on plausible correspondences between evolutionary lineages. The distance δ_{CED} computes dissimilarity for each matched pair by summing: (1) the *node worth*, i.e., the number of nodes with altered label and historical content in corresponding paths; (2) the minimum rearrangement cost derived from a relation matrix of ancestor-descendant relationship disagreements; and (3) the insertion–deletion

cost for labels present in one but not the other. The overall distance is the sum of these components over all pairs in the matching.

While δ_{CED} preserves symmetry and is equal to zero if and only if the trees are identical, it does not satisfy the triangle inequality, and thus is a semi-metric. Bounds were established for its maximum value over trees with n common and n' unique labels, yielding a diameter of $n' + 2n - 3$. A complete set of algorithms was presented for partition generation, tie-breaking, node worth evaluation, rearrangement computation, and insertion–deletion counting, together with proofs that the algorithms are exact. In practice, the algorithm designed to find δ_{CED} is polynomial in the number of leaves and labels when the number of ties in the path-matching process is small, but can become exponential in worst-case tie situations.

The methodology chapter includes formal proofs, propositions, and illustrative examples demonstrating how the metric and semi-metric operate in realistic scenarios. Our framework is sensitive to both local and global differences in tree topology and labeling, naturally handling repeated labels and mutation losses. This degree of precision allows for meaningful comparison of clonal trees arising from different patients, tumor regions, or time points, as well as benchmarking of tree reconstruction algorithms against known or simulated ground truths. Although developed in the context of cancer genomics, the underlying principles may extend to other domains involving labeled rooted trees with complex label behaviour.

The contributions of this work are primarily theoretical, establishing a new mathematical formalism for clonal tree comparison, proving key properties, and providing algorithms with analyzed complexity.

5.2 Limitations

Although in this thesis we introduce a novel and biologically motivated framework for comparing clonal evolution trees, some limitations should be acknowledged. A primary limitation is that the present formulation treats all mutation labels the same. In reality, however, mutations differ greatly in their functional and clinical impact.

A small subset of *driver* mutations often plays a decisive role in tumor initiation, progression, and therapy resistance. In contrast, *passenger* mutations are alterations that arise during tumor evolution but do not contribute directly to cancer development; they are generally biologically neutral. Assigning identical costs to CED operations involving either type may understate the significance of structural changes affecting high-impact driver mutations, while exaggerating the importance of alterations involving biologically neutral passenger mutations.

Developing a weighting scheme informed by biological significance could improve the method's ability to reflect the true consequences of evolutionary events. Biological significance could be derived from cancer gene databases, collections of interacting genes/proteins with a shared function, or laboratory experiments that test whether a specific mutation alters cell behavior, growth, or drug sensitivity.

Another limitation of the current framework is that it treats all clones equally, regardless of their size or prevalence within the tumor population. In our proposed distance, each node contributes to the distance solely based on its labeling or location changes, without considering the proportion of cells that the corresponding subclone represents.

In reality, large clones encompassing a substantial fraction of the tumor mass may have a greater impact on disease progression, treatment resistance, and clinical outcome than small, rare clones. By ignoring clone size, our method may overemphasize alterations in minor clones, while underrepresenting significant evolutionary changes in dominant clones.

Incorporating clonal prevalence—potentially derived from bulk or single—cell sequencing data—into the distance calculation could provide a more clinically meaningful measure of evolutionary divergence between tumors.

5.3 Future Works

The Clonal Evolution Distance d_{CED} and its semi-metric variant δ_{CED} introduced in this thesis represent promising advances in the comparison of tumor clonal trees under realistic evolutionary models. However, several important future directions remain to fully validate, refine, and extend these methods for broader applications in cancer genomics.

5.3.1 Validation on Simulated and Real Data

A key next step is the systematic validation of d_{CED} and δ_{CED} using both simulated clonal trees with known ground-truth trees and trees generated for real sequencing datasets. Simulated data generated under controlled conditions can allow precise assessment of each measure's sensitivity to label dissimilarities such as mutation losses, and repeated mutations, and varying phylogenetic topologies. Real datasets from single-cell or multi-region sequencing will test the measures in clinical contexts.

An especially informative validation strategy we can take is to compare clonal trees reconstructed from *subsets* of sequencing data with those reconstructed from the *complete* dataset for the same patient. The degree to which our method recognizes these as similar, despite the subset lacking some mutations or regions, will provide a practical measure of its robustness to incomplete or noisy input. Such experiments can simulate realistic situations where technical constraints or cost limitations reduce sequencing depth or coverage.

5.3.2 Refinement of Tie-Breaking Criteria

We use a hierarchical tie-breaking procedure for δ_{CED} to select optimal path pairings when multiple equally good options exist. Although these criteria are biologically motivated, their impact on accuracy and runtime should be examined. With large-scale testing we can reveal which tie configurations most often affect final distance values or increase computational time. Insights from these experiments can guide us with refinement of tie-breaking priorities. By applying this iterative approach, we can both improve biological fidelity and reduce computational overhead.

Our longer-term goal is to identify a tie-breaking condition (or set of conditions) that uniquely resolves all ties in practice, effectively collapsing the exponential worst case into a polynomial-time procedure for δ_{CED}. This may require both theoretical exploration of the structural constraints in realistic tumor trees, and data-driven learning of tie-resolution rules using extensive collections of simulated and real-world comparisons. Our ultimate aim is to develop a version of δ_{CED} that maintains high biological relevance while achieving guaranteed tractability.

5.3.3 Comparative Analysis with Existing Distances

Comparing d_{CED} and δ_{CED} with the existing clonal tree distances is essential to clarify their unique strengths. The same benchmark datasets used for CED validation should be used for distances such as generalizations of the Robinson–Foulds distance, CASet, DISC, MLTD, and triplet-based approaches. Discrimination power between biologically distinct evolutionary scenarios, robustness to missing or noisy data, correlation with known ground truth, and computational efficiency are the performance metrics the we should consider. Comparing results for both complete and subset induced trees can highlight situations where CED offers distinct advantages over existing distances.

Appendix A

Supporting Algorithms

A.1 Tie-Breaking Details

Algorithm 9 Tie-Breaking Hierarchy

function TieBreak(B)
    Find all entries b_{ij} with the largest number of labels
    if tie **then**
        Among these, select those where more labels have matching sets of labels and content
        if tie **then**
            Among these, select those where fewer labels appear elsewhere in B
            if tie **then**
                Among these, select those where fewer elsewhere labels have matching historical content
                if tie **then**
                    Select those with minimum MinNumberOfRearrangements(T_i, T'_j)
                end if
            end if
        end if
    end if
    return all selected entries
end function

A.2 Table Initialization Details

Each table entry b_{ij} stores:

- Shared labels between paths except r the label of the root
- For each label:
  - Origin nodes (separate for T and T')
  - Mutation content of the origin nodes

A.3 Unmapped Path Handling

procedure HANDLEUNMAPPED(B, P, P', t)
  for each unmapped T_i with empty column in B **do**
    $P[t] \leftarrow T_i, \quad P'[t] \leftarrow \emptyset$
    $t \leftarrow t + 1$
  end for
  for each unmapped T'_j with empty row in B **do**
    $P[t] \leftarrow \emptyset, \quad P'[t] \leftarrow T'_j$
    $t \leftarrow t + 1$
  end for
end procedure

A.4 Table Update Procedure

1: **procedure** UpdateTable(B, i, j)

2: Remove column i and row j

3: Remove labels of b_{ij} from other entries

4: Remove corresponding nodes from remaining paths

5: **end procedure**

Bibliography

B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5(1):1–15, 2001. doi: 10.1007/s00026-001-8006-8.

M. H. Bailey, C. Tokheim, E. Porta-Pardo, S. Sengupta, D. Bertrand, A. Weerasinghe, A. Colaprico, M. C. Wendl, J. Kim, B. Reardon, P. K. Ng, K. J. Jeong, S. Cao, Z. Wang, J. Gao, Q. Gao, F. Wang, E. M. Liu, L. Mularoni, C. Rubio-Perez, N. Nagarajan, I. CortÃ©s-Ciriano, D. C. Zhou, W. W. Liang, J. M. Hess, V. D. Yellapantula, D. Tamborero, A. Gonzalez-Perez, C. Suphavilai, J. Y. Ko, E. Khurana, P. J. Park, E. M. Van Allen, H. Liang, M. W. Group, C. G. A. R. Network, M. S. Lawrence, A. Godzik, N. Lopez-Bigas, J. Stuart, D. Wheeler, G. Getz, K. Chen, A. J. Lazar, G. B. Mills, R. Karchin, and L. Ding. Comprehensive characterization of cancer driver genes and mutations. *Cell*, 173 (2):371–385.e18, April 2018. doi: 10.1016/j.cell.2018.02.060.

D. Bertheloot, E. Latz, and B. S. Franklin. Necroptosis, pyroptosis and apoptosis: an intricate game of cell death. *Cellular & Molecular Immunology*, 18(5):1106–1121, May 2021. doi: 10.1038/s41423-020-00630-3. Epub 2021 Mar 30. PMID: 33785842; PMCID: PMC8008022.

P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1–3):217–239, 2005. doi: 10.1016/j.tcs.2004.12.030.

P. Bonizzoni, S. Ciccolella, G. Della Vedova, and M. Soto. Beyond perfect phylogeny: Multisample phylogeny reconstruction via ilp. *ACM-BCB '17, Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, August 2017. doi: 10.1145/3107411.3107441. © 2017 Copyright held by the owner/author(s).

N. Chatterjee and G. C. Walker. Mechanisms of dna damage, repair, and mutagenesis. *Environmental and Molecular Mutagenesis*, 58(5):235–263, June 2017. doi: 10.1002/em.22087.

S. Ciccolella, G. Bernardini, L. Denti, P. Bonizzoni, M. Previtali, and G. Della Vedova. Triplet-based similarity score for fully multilabeled trees with poly-occurring labels. *Bioinformatics*, 37(2):178–184, April 2021a. doi: 10.1093/bioinformatics/btaa676.

S. Ciccolella, C. Ricketts, M. Soto Gomez, M. Patterson, D. Silverbush, P. Bonizzoni, I. Hajirasouliha, and G. Della Vedova. Inferring cancer progression from single-cell sequencing while allowing mutation losses. *Bioinformatics*, 37(3):326–333, February 2021b. doi: 10.1093/bioinformatics/btaa722.

Z. DiNardo, K. Tomlinson, A. Ritz, and L. Oesper. Distance measures for tumor evolutionary trees. *Bioinformatics*, 36(7):2090–2097, April 2020. doi: 10.1093/bioinformatics/btz869.

M. El-Kebir. Sphyr: tumor phylogeny estimation from single-cell sequencing data under loss and error. *Bioinformatics*, 34(17):i671–i679, 2018. doi: 10.1093/bioinformatics/bty589.

M. El-Kebir, G. Satas, and B. J. Raphael. Inferring parsimonious migration histories for metastatic cancers. *Nature Genetics*, 50(5):718–726, May 2018. doi: 10.1038/s41588-018-0106-z.

S. Franjic. Dna is the basic molecule of inheritance. *JSM Biotechnology and Bioengineering*, 7(1):1089, November 2021.

S. J. Gould. Dollo on dollo's law: irreversibility and the status of evolutionary laws. *Journal of the History of Biology*, 3(2):189–212, 1970. doi: 10.1007/BF00137351.

K. Govek, C. Sikes, and L. Oesper. A consensus approach to infer tumor evolutionary histories. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 63–72, Washington, DC, USA, 2018. Association for Computing Machinery. doi: 10.1145/3233547.3233584. URL `https://doi.org/10.1145/3233547.3233584`.

M. Greaves and C. C. Maley. Clonal evolution in cancer. *Nature*, 481(7381):306–313, January 2012. doi: 10.1038/nature10762.

M. J. Havey. Overview of deoxyribonucleic acid (dna). *IDOSR Journal of Biology, Chemistry and Pharmacy*, 1(1):1–6, January 2017.

J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973. doi: 10.1137/0202019.

R. R. Iyer, A. Pluciennik, V. Burdett, and P. L. Modrich. Dna mismatch repair: Functions and mechanisms. *Chemical Reviews*, 106(2):302–323, February 2006. doi: 10.1021/cr0404794.

K. Jahn, N. Beerenwinkel, and L. Zhang. The bourque distances for mutation trees of cancers. *Algorithms for Molecular Biology*, 16:9, June 2021. doi: 10.1186/s13015-021-00188-3.

C. Jiang, E. Schaafsma, W. Hong, Y. Zhao, K. Zhu, C.-C. Chao, and C. Cheng. Influence of t cell-mediated immune surveillance on somatic mutation occurrences in melanoma. *Frontiers in Immunology*, 12:703821, January 2022. doi: 10.3389/fimmu.2021.703821.

N. Karpov, S. Malikic, M. K. Rahman, and C. S. Sahinalp. A multi-labeled tree dissimilarity measure for comparing clonal trees of tumor progression. *Algorithms for Molecular Biology*, 14:17, July 2019.

E. Khayatian, G. Valiente, and L. Zhang. The k-robinson-foulds dissimilarity measures for comparison of labeled trees. *Journal of Computational Biology*, 31(4):328–344, April 2024. doi: 10.1089/cmb.2023.0312.

S. Kumar, A. Chroni, K. Tamura, M. Sanderford, O. Oladeinde, V. Aly, T. Vu, and S. Miura. Pathfinder: Bayesian inference of clone migration histories in cancer. *Bioinformatics*, 36 (Supplement_2):i675–i683, December 2020. doi: 10.1093/bioinformatics/btaa795.

D. Kőnig. Graphen und matrizen. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

L. C. Lau, R. Ravi, and M. Singh. Matching and vertex cover in bipartite graphs. In R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, editors, *Iterative Methods in Combinatorial Optimization*, pages 28–45. Cambridge University Press, Cambridge, UK, 2012. doi: 10.1017/CBO9780511977152.004.

L. Li, W. Xie, L. Zhan, S. Wen, X. Luo, S. Xu, Y. Cai, W. Tang, Q. Wang, M. Li, Z. Xie, L. Deng, H. Zhu, and G. Yu. Resolving tumor evolution: a phylogenetic approach. *Journal of the National Cancer Center*, 4(2):97–106, June 2024. doi: 10.1016/j.jncc.2024.03.001.

B. Lim, Y. Lin, and N. Navin. Advancing cancer research and medicine with single-cell genomics. *Cancer Cell*, 37(4):456–470, April 2020. doi: 10.1016/j.ccell.2020.03.008.

B. Liu, H. Zhou, L. Tan, K. T. H. Siu, and X.-Y. Guan. Exploring treatment options in cancer: tumor treatment strategies. *Signal Transduction and Targeted Therapy*, 9(1):175, July 2024. doi: 10.1038/s41392-024-01856-7.

M. Llabrés, F. RossellÃ³, and G. Valiente. A generalized robinson-foulds distance for clonal trees, mutation trees, and phylogenetic trees and networks. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, page 13, New York, NY, USA, September 2020. Association for Computing Machinery. doi: 10.1145/3388440.3412479.

M. Llabrés, F. RossellÃ³, and G. Valiente. The generalized robinson-foulds distance for phylogenetic trees. *Journal of Computational Biology*, 28(12):1181–1195, December 2021. doi: 10.1089/cmb.2021.0342.

S. Malikic, F. R. Mehrabadi, S. Ciccolella, M. K. Rahman, C. Ricketts, E. Haghshenas, D. Seidman, F. Hach, I. Hajirasouliha, and S. C. Sahinalp. Phiscs: a combinatorial approach for subperfect tumor phylogeny reconstruction via integrative use of single-cell and bulk sequencing data. *Genome Research*, 29(11):1860–1877, November 2019. doi: 10.1101/gr.234435.118. Epub 2019 Oct 18.

Z. Mao, M. Bozzella, A. Seluanov, and V. Gorbunova. Comparison of nonhomologous end joining and homologous recombination in human cells. *DNA Repair (Amst)*, 7(10):1765–1771, October 2008. doi: 10.1016/j.dnarep.2008.06.018.

N. McGranahan and C. Swanton. Clonal heterogeneity and tumor evolution: Past, present, and the future. *Cell*, 168(4):613–628, February 2017. doi: 10.1016/j.cell.2017.01.018.

R. Nakad and B. Schumacher. Dna damage response and immune defense: Links and mechanisms. *Frontiers in Genetics*, 7:147, August 2016. doi: 10.3389/fgene.2016.00147. Edited by Alexandros G. Georgakilas. Reviewed by Bjoern Schwer and Vassilis G. Gorgoulis.

National Cancer Institute. Definition of dna sequencing. `https://www.cancer.gov/`

publications/dictionaries/cancer-terms/def/dna-sequencing. Accessed: July 2025.

National Cancer Institute. What is cancer?, 2025. URL https://www.cancer.gov/about-cancer/understanding/what-is-cancer. Accessed July 2025.

L. Peters, A. Venkatachalam, and Y. Ben-Neriah. Tissue-predisposition to cancer driver mutations. *Cells*, 13(2):106, January 2024. doi: 10.3390/cells13020106.

D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1–2):131–147, February 1981. doi: 10.1016/0025-5564(81)90043-2.

P. Sashittal, H. Zhang, C. Iacobuzio-Donahue, et al. Condor: Tumor phylogeny inference with a copy-number constrained mutation loss model. *Genome Biology*, 24:272, November 2023. doi: 10.1186/s13059-023-03106-5.

G. Satas, S. Zaccaria, G. Mon, and B. J. Raphael. Scarlet: Single-cell tumor phylogeny inference with copy-number constrained mutation losses. *Cell Systems*, 10(4):323–332, April 2020. doi: 10.1016/j.cels.2020.04.001.

D. Sekar, K. Thirugnanasambantham, V. I. Hairul Islam, and S. Saravanan. Sequencing approaches in cancer treatment. *Cell Proliferation*, 47(5):391–395, October 2014. doi: 10.1111/cpr.12124.

D. Tang, X. Chen, R. Kang, and et al. Ferroptosis: molecular mechanisms and health implications. *Cell Research*, 31:107–125, February 2021. doi: 10.1038/s41422-020-00441-1.

J. D. Watson and F. H. C. Crick. The structure of dna. *Cold Spring Harbor Symposia on Quantitative Biology*, 18:123–131, 1953. doi: 10.1101/sqb.1953.018.01.020.

R. D. Wood. Dna damage recognition during nucleotide excision repair in mammalian cells. *Biochimie*, 81(1-2):39–44, January-February 1999. doi: 10.1016/s0300-9084(99)80036-4.

Z. Yu, H. Liu, F. Du, and X. Tang. Grmt: Generative reconstruction of mutation tree from scratch using single-cell sequencing data. *Frontiers in Genetics*, 12:692964, June 2021. doi: 10.3389/fgene.2021.692964.

A. Zafar, S. Khatoon, M. J. Khan, J. Abu, and A. Naeem. Advancements and limitations in traditional anti-cancer therapies: a comprehensive review of surgery, chemotherapy, radiation therapy, and hormonal therapy. *Discover Oncology*, 16(1):607, April 2025. doi: 10.1007/s12672-025-02198-8.

H. Zafar, A. Tzen, N. Navin, K. Chen, and L. Nakhleh. Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome Biology*, 18:178, September 2017. doi: 10.1186/s13059-017-1311-2.

H. Zafar, N. Navin, K. Chen, and L. Nakhleh. Siclonefit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. *Genome Research*, 29(11):1847–1859, November 2019. doi: 10.1101/gr.243121.118.

K. Zhang and T. Jiang. Some max snp-hard results concerning unordered labeled trees. *Information Processing Letters*, 49(5):249–254, 1994. doi: 10.1016/0020-0190(94)90062-0.

K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. Technical Report 3, 1992.