Improved Algorithms for Bi-Partition Function Computation

John D. Bridgers
Division of Intramural Research, National Library of Medicine, Bethesda, MD, USA

Jan Hoinka
Division of Intramural Research, National Library of Medicine, Bethesda, MD, USA

S. Cenk Sahinalp
Cancer Data Science Laboratory, National Cancer Institute, Bethesda, MD, USA

Salem Malikic[1]
Cancer Data Science Laboratory, National Cancer Institute, Bethesda, MD, USA

Teresa M. Przytycka[1]
Division of Intramural Research, National Library of Medicine, Bethesda, MD, USA

Funda Ergun[1]
Department of Computer Science, Indiana University, Bloomington, IN, USA

Abstract

The evolutionary history of a tumor, inferred from single-cell sequencing data, is typically represented as a tree in which each subtree corresponds to a clade of cells seeded by a specific set of mutations. Traditional methods typically identify a single most likely tree for downstream analyses, such as detecting driver mutations, studying mutation co-occurrence patterns and identifying common evolutionary trajectories. However, the reliability of such inferred trees, particularly their topology, clade composition, and mutational placements, often remains uncertain.

To quantify this uncertainty, the concept of a Bi-partition Function was recently introduced, providing a probabilistic measure of how reliably a mutation seeds a given clade of cells. The single available algorithm for estimating the Bi-partition Function relies on simplifying assumptions and uses sampling for limited exploration of the tree-space.

In this paper, we introduce the first exact algorithm for computing the Bi-partition Function. Our algorithm scales linearly with the number of mutations but exhibits super-exponential complexity with respect to the number of cells. Despite this complexity, it establishes crucial ground truth values, essential for accurately benchmarking and validating approximate methods. Additionally, we present a GPU-accelerated version of the available sampling-based algorithm, significantly boosting the computational performance through large-scale parallelization, enabling more accurate Bi-partition Function estimates via deeper exploration of the tree spaces. We compare our methods on synthetic datasets, demonstrating that especially when the number of mutations sufficiently exceed the number of cells, our GPU-accelerated sampling algorithm closely approximates the exact ground truth values.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Tumor Evolution, Bi-partition Function, Single-Cell Sequencing, Algorithms

Digital Object Identifier 10.4230/LIPIcs.WABI.2025.5

Supplementary Material *Software (Source Code)*: `https://github.com/john-db/bipartf-exact`

Funding J.D.B., J.H., and T.M.P. were supported by the Division of Intramural Research (DIR) of the National Library of Medicine (NLM), National Institutes of Health. S.C.S. and S.M. were supported by the Intramural Research Program of the National Cancer Institute (NCI), National Institutes of Health. F.E. was supported by NSF grant 2414736.

[1] Joint last authors

© John D. Bridgers, Jan Hoinka, S. Cenk Sahinalp, Salem Malikic, Teresa M. Przytycka, and Funda Ergun;
licensed under Creative Commons License CC-BY 4.0

Acknowledgements This research utilized the computational resources of the NIH Biowulf high performance computing cluster (`http://hpc.nih.gov`).

1 Introduction

Tumor phylogenies offer invaluable insights into tumor evolution, clonal dynamics, metastatic migration, and treatment resistance. Each path from the root of such a tree to its leaves represents a distinct evolutionary trajectory, reflecting critical aspects of tumor development and progression [13]. Typically, tumor phylogenies are inferred from mutations detected in single-cell sequencing data, which are inherently noisy [4, 12, 18, 10, 14, 15, 6, 5, 17]. Therefore, assessing the reliability of inferred tumor phylogenies – particularly their topological structures, clade compositions, and mutation placements – is crucial before conducting downstream analyses, such as identifying clonal versus subclonal drivers, mutation co-occurrences, and shared evolutionary paths across tumor samples.

To rigorously address this need, the recently introduced concept of the Bi-partition Function quantifies the probability that a specific set of mutations seeds a particular clade of cells [11]. Given a genotype matrix with rows representing cells and columns representing mutations, the Bi-partition Function for a mutation (column) ρ and a subset of cells (rows) R represents the probability that precisely the cells in R harbor mutation ρ under the perfect phylogeny model. Consequently, the Bi-partition Function serves as a powerful approach to assess whether an inferred cell clade genuinely represents a biologically distinct group, potentially associated with aggressive growth, immune evasion, or treatment resistance [7].

Currently, the only available method for estimating the Bi-partition Function relies on sampling evolutionary scenarios from a restricted subset of possible trees of tumor evolution [11]. Despite its theoretical guarantee of convergence to correct values in the limit, this sampling-based algorithm inherently involves approximations and assumptions aimed at computational tractability. However, the extent of its accuracy for finite datasets remained unknown, primarily because exact values of the Bi-partition Function have not been previously computable.

In this paper, we present the first exact algorithm for computing the Bi-partition Function from an input genotype matrix. Our algorithm exhibits linear complexity relative to the number of mutations but scales super-exponentially with the number of cells. While computationally intensive for large datasets, this algorithm provides a critical utility to benchmark the accuracy and assess the limitations of approximate methods based on sampling.

Furthermore, we introduce a GPU-accelerated implementation of the existing sampling algorithm [11], in which the scoring of evolutionary scenarios has been reformulated to allow large scale parallelization. This acceleration significantly increases the number of scenarios that can be evaluated within a given time limit and thus improves estimation accuracy, reduces sampling noise, and facilitates a precise assessment of algorithmic approximations.

We compare our algorithms on simulated and real datasets to evaluate the performance and accuracy of the sampling approach. Our results reveal that, particularly in datasets where mutations sufficiently outnumber cells, our sampling approach provides accurate estimates of the Bi-partition Function, validating its practical utility in tumor phylogeny evaluation.

2 Background, Motivation, and Our Problem

Since the concept of the Bi-partition Function for trees of tumor evolution was recently introduced, we first provide some background and motivation. While we present some general formulations in this section, the rigorous mathematical definitions and the details of the exact

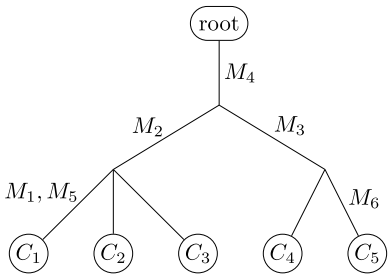
computation algorithm are deferred to the next section. Before discussing the Bi-partition Function, we provide an overview of how the evolutionary history of a tumor is represented and inferred from single-cell sequencing data.

2.1 Representing the Evolutionary History of a Tumor

In this work, we assume that we are given the results of a single-cell sequencing experiment, in which a set $C = \{C_1, C_2, \ldots, C_n\}$ of single cells was extracted from a tumor sample and sequenced. Following the mutation calling step, we are given a set $M = \{M_1, M_2, \ldots, M_m\}$ of somatic mutations, each reported to be present in at least one of the sequenced cells.

We also assume that the commonly used Infinite Sites Assumption (ISA) holds; i.e., each mutation is acquired at most once during the course of tumor evolution, and, once it is acquired in some cell, it never gets lost and is passed on to all of the cell's descendants.

Under the ISA, the evolutionary history of the sequenced cells can be represented by a tree in which each leaf represents exactly one of the sequenced cells – we will thus use the terms "cell" and "leaf" interchangeably. The root of such a tree corresponds to a population of normal cells, free of any somatic mutations. Each non-root internal node has at least two children, and the edges correspond to (and are labeled by) mutations. For each mutation placed on the edge connecting node v to its parent, we assume that the mutation is present in cell v (if v is a leaf) or in all cells that are descendants of v (if v is an internal node). In this work, we will refer to these trees as Cell Lineage Trees with Mutation assignment (CLT-M). An example of a CLT-M is shown in Figure 1a.



(a)

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	1	1	0	1	1	0
C_2	0	1	0	1	0	0
C_3	0	1	0	1	0	0
C_4	0	0	1	1	0	0
C_5	0	0	1	1	0	1

(b)

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	1	0	0	1	1	1
C_2	0	1	?	?	0	0
C_3	?	1	0	1	0	0
C_4	0	?	1	0	0	0
C_5	0	0	?	1	0	1

(c)

Figure 1 **(a)** A cell lineage tree with mutation assignment (CLT-M) depicting the evolutionary history of a tumor. **(b)** The ground truth genotype matrix corresponding to the tree in (a). **(c)** An observed genotype matrix with false negatives (in blue), false positives (in red), and missing entries (indicated by question marks).

Given any node v of a CLT-M, we define the *clade* of v as the set of cells (leaves) of the subtree rooted at v; by this definition a leaf is its own clade (i.e., a clade consisting of one cell). We say that the mutations labeling the edge connecting v to its parent are *seeding* mutations of the clade of v – thus, we say that mutations "correspond to," or, equivalently, "seed" a clade. For example, in the tree shown in Figure 1a, the set of cells $\{C_1, C_2, C_3\}$ forms a clade seeded by mutation M_2, while $\{C_1\}$ forms a clade seeded by M_1 and M_5.

Note that a set of mutations harbored by a cell (leaf) v in a CLT-M is the union of mutation labels on the path from v to the root. As a result, a CLT-M defines a unique *Ground Truth Genotype matrix* $\mathbf{X}$, which is a binary matrix with n rows corresponding to cells and m columns corresponding to mutations. In this matrix, $\mathbf{X}_{ij} = 1$ if mutation M_j is present in cell C_i; otherwise $\mathbf{X}_{ij} = 0$. The ground truth genotype matrix corresponding to the tree shown in Figure 1a is presented in Figure 1b.

Let $\mathbf{X}_\rho$ denote the ρth column of $\mathbf{X}$, representing mutation M_ρ. Observe that M_ρ seeds the clade consisting of cells $R \subseteq C$ if and only if $\mathbf{X}_{i\rho} = 1$ for all $C_i \in R$, and $\mathbf{X}_{i\rho} = 0$ otherwise. In other words, the ground truth matrix has 1s exactly at the rows corresponding to the cells in R. We denote this as $\mathbf{X}_\rho = R$.

2.2 Tree Inference from Single-Cell Sequencing Data

While a CLT-M can be inferred from the ground truth genotype matrix $\mathbf{X}$ in polynomial time [3], $\mathbf{X}$ is not known to us in practice. What we have instead is an *observed matrix* I, given as the output of a single-cell sequencing experiment. I is an $n \times m$ matrix with entries from $\{0, 1, ?\}$, where 1 and 0 indicate the presence and absence of a mutation in a cell respectively (as reported by the mutation calling), and "?" encodes *missing entries*, indicating insufficient information to call the presence or absence of a mutation. An example of the observed genotype matrix is shown in Figure 1c.

In addition to the missing entries, I typically has some incorrect (i.e., different from the ground truth) entries, due to false positive and false negative mutation calls. Consequently, it is likely that there is no CLT-M such that the presence pattern of mutations in cells as implied by the CLT-M is the same as the those given by I. For simplicity, we say that there does not exist any tree T such that I and T are *consistent* with each other.

Fortunately, one can succinctly and precisely characterize matrices that *are* consistent with some CLT-M. Any binary (or ternary) matrix A containing a triplet of rows/cells (i, j, h) and a pair of columns/mutations (a, b), in any order, such that

$$\begin{bmatrix} A_{ia} & A_{ib} \\ A_{ja} & A_{jb} \\ A_{ha} & A_{hb} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

is *n*ot consistent with any CLT-M under the ISA. Such a pair of mutations and triplet of cells are called a *conflict*; we call matrices not containing any conflicts *conflict-free matrices*. It is well known that any conflict-free matrix is consistent with some CLT-M [3].

Existing methods for reconstructing trees of tumor evolution, either implicitly (through a search in the space of trees, [4, 8, 18]) or explicitly (through a direct search in the space of matrices, [2, 10, 14, 1, 15]) convert some entries of I from 1 to 0 and from 0 to 1, and set the missing entries to 0 or 1, in order to obtain a conflict-free binary matrix E (see [9] for more details). The tree corresponding to E is then usually reported as the most likely evolutionary history of the sequenced cells and used in the downstream biological analyses.

2.3 Looking inside Trees of Tumor Evolution

As discussed in the Introduction, the tree inferred from single-cell sequencing data, which is not necessarily identical to the ground truth, features multiple clades, each with its own seeding mutations and possibly distinct phenotypes. For example, cells in some clade may have an increased proliferative advantage and higher aggressiveness compared to other cells, suggesting that the mutations seeding that clade are potential drivers of this aggressive behavior. However, the placement of some of these mutations may have low support in the data and in the alternative, near-optimal trees, these mutations may not be seeding the clade of interest. Therefore, before drawing major biological conclusions from inferred trees, it is advisable to assess the confidence that a given mutation of interest is specific to a particular clade. When this confidence is low, additional experimental validation is warranted.

The above observation has motivated us to introduce the *Bi-partition Function* for trees of tumor evolution [11]. Given the observed genotype matrix and the noise probabilities, one can draw conclusions regarding the likelihood of structures that make up the ground truth tree: The Bi-partition Function is defined as the probability that, in the ground truth tree a given mutation M_ρ seeds a given clade $R \subseteq C$, effectively partitioning the cells into two groups: R, arranged into a clade, which harbor mutation M_ρ, and $C \setminus R$, which do not. This is equivalent to the ground truth matrix $\mathbf{X}$ satisfying $\mathbf{X}_\rho = R$, hence the Bi-partition Function can be defined as the probability $\Pr[\mathbf{X}_\rho = R \mid \mathbf{X} \in \mathcal{G}]$, where $\mathcal{G}$ denotes the set of all conflict-free matrices of size $n \times m$. Note that

$$\Pr[\mathbf{X}_\rho = R \mid \mathbf{X} \in \mathcal{G}] = \frac{\Pr[\mathbf{X}_\rho = R \wedge \mathbf{X} \in \mathcal{G}]}{\Pr[\mathbf{X} \in \mathcal{G}]}.$$

To evaluate this equation, we need to add up the probabilities of all conflict-free matrices of size $n \times m$ that have column ρ equal to R (the numerator) and divide by the sum of the probabilities of all $n \times m$ conflict-free matrices (the denominator). The most straightforward way of doing this is iterating through the space of such binary matrices, discarding those with conflicts and focusing on those that are conflict-free. Since the number of distinct binary matrices is 2^{nm}, this approach is impractical, except for extremely small (n, m).

In this paper, we exploit the fact that all conflict free matrices correspond to some CLT-M and derive an algorithm to compute the above sum. Our algorithm computes the Bi-partition Function exactly for datasets containing a small number of cells (up to 8) and an arbitrary number of mutations. While in most practical applications the number of sequenced cells typically exceeds 8, it is common practice to pre-cluster cells into a small number of clusters to reduce the impact of noise. The resulting tree is then inferred based on these clusters. Therefore, our algorithm for exact computation of the Bi-partition Function remains relevant and practically valuable in a wide range of real-world settings.

3 The Formal Problem Definition and Methods

In this section we first formally define the Bi-partition Function and then describe our method for computing it.

3.1 Our Problem: the Bi-partition Function

As discussed above, in practice, we do not have direct access to the ground truth; rather, we have observations regarding the presence of mutations in cells. Such observations can be insufficient to make a call, or can be faulty with respect to a false positive rate α and

false negative rate β. Recall that our input is given as an observed matrix I of dimensions $n \times m$ with entries 0, 1, or "?", with "?" denoting a missing entry where we were unable to make a mutation call. With I (together with estimated false positive/negative probabilities) as an input, the underlying ground truth can be expressed as a distribution P over $n \times m$ binary matrices, with P represented as a matrix in $[0,1]^{n \times m}$: A genotype matrix $\mathbf{X}$ drawn from distribution P has a 1 in entry $\mathbf{X}_{ij}$ with probability P_{ij}, and 0 with probability $1 - P_{ij}$, independently from all other entries. (See Section A.1 for how P is constructed.)

With the ground truth a distribution rather than a single matrix, we now discuss *the probability that* a mutation ρ (shorthand for mutation M_ρ) is confined to a particular set of rows in a conflict-free matrix $\mathbf{X}$ chosen according to P. [2] Let $\mathcal{G}$ denote the set of conflict-free genotype matrices of dimensions $n \times m$. Our goal is to evaluate the Bi-partition Function, now defined formally:

▶ **Definition 1** (Bi-partition Function). *Let* $\mathbf{X}$ *denote a genotype matrix drawn from distribution* P *as described above. Given a set of cells (rows)* R *and a mutation (column label)* ρ*, the* Bi-partition Function *is the probability that column* ρ *of* $\mathbf{X}$ *contains* 1*s in rows corresponding to* R*, and* 0*s elsewhere, conditioned on* $\mathbf{X}$ *being conflict-free, i.e.,* $\Pr[\mathbf{X}_\rho = R \mid \mathbf{X} \in \mathcal{G}]$.

As already mentioned above, we have

$$\Pr[\mathbf{X}_\rho = R \mid \mathbf{X} \in \mathcal{G}] = \frac{\Pr[\mathbf{X}_\rho = R \wedge \mathbf{X} \in \mathcal{G}]}{\Pr[\mathbf{X} \in \mathcal{G}]}. \tag{1}$$

Our Approach. Since it is too inefficient to enumerate genotype matrices[3], we implicitly map the very large space of genotype matrices to that of Cell Lineage Trees (see next section for a formal definition) and evaluate the function in the latter, smaller space. Below, we investigate how the two spaces relate to one another.

3.2 Cell Lineage Trees

In what follows, we will use the notion of a Cell Lineage Tree (CLT), which is a generalized version of a CLT-M that we defined above: A CLT is a tree whose leaves represent a set C of cells, where each internal non-root node has at least two children – simply a CLT-M without edge (i.e., mutational) labels. Note that a CLT T can be turned into a number of different CLT-Ms through different labelings of its edges.

In order to better understand CLTs and their relationship to genotype matrices, we consider their components, clades, and classify them. We start with three types of special clades that we name *trivial clades*: (i) the clade of any leaf v, which is v itself, (ii) the clade of the root consisting of the entire set of cells, C, (iii) the *empty clade* with zero cells, assumed by convention to be part of any CLT in order to account for mutations that are absent from all cells.[4] Note that trivial clades are all part of every CLT on a given set of cells.

All other clades are said to be *nontrivial clades* of the CLT. Note that the set of nontrivial clades uniquely determine a CLT.

---

[2] We require $\mathbf{X}$ to be conflict-free; otherwise it cannot be consistent with any CLT and thus cannot correspond to the ground truth.

[3] In practice, such enumeration is only possible for some small values of n and m. In fact, we have also implemented the naive algorithm that enumerates all genotype matrices to compute the Bi-partition Function and verified our algorithm against it on matrices consisting of 5 cells and 5 mutations.

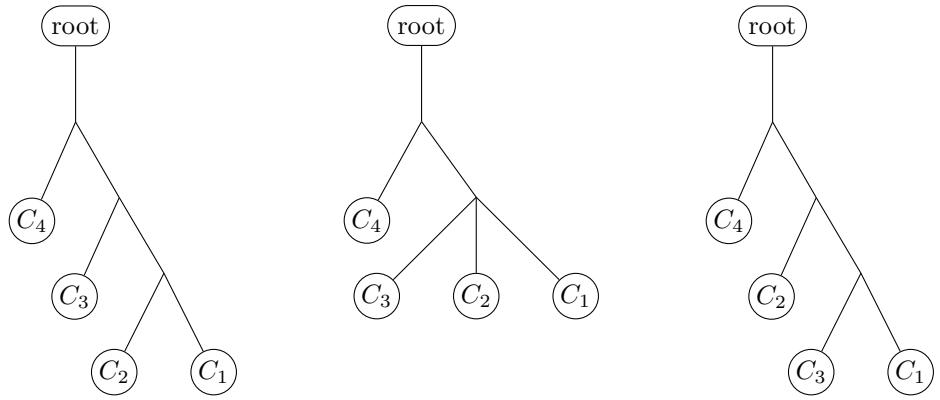
[4] While in the observed matrix, a given mutation is always reported as present in at least one cell, there is a small probability that it is absent in all cells in the true (but unknown) evolutionary history.

3.2.1 Linking Genotype Matrices to CLTs

We now explore the relationship between a genotype matrix G and CLTs on the same cells.

Let G to be a genotype matrix on cells $C = \{C_1, C_2, \ldots, C_n\}$ and mutations $M = \{M_1, M_2, \ldots, M_m\}$, and G_i denote the ith column of G (we use this column notation throughout the paper). Furthermore, let $[k]$ denote $\{1, 2, \ldots, k\}$ for any positive integer k. Let T be a CLT on cells (i.e., with leaves) C; recall that the edges of T are unlabeled. We say a mutation M_i in G "corresponds" to T if the cells harboring M_i in G form a clade in T.

▶ **Definition 2.** *G is consistent with a CLT T if every mutation in G corresponds to some empty or non-empty clade of T. The set of all genotype matrices consistent with T is denoted as $CM(T)$ (CM stands for "consistent matrices").*



(a) **(b)** **(c)**

	M_1	M_2	M_3	M_4	M_5
C_1	**1**	1	1	0	0
C_2	1	1	1	1	0
C_3	0	1	1	0	1
C_4	0	0	1	0	0

(d)

	M_1	M_2	M_3	M_4	M_5
C_1	**0**	1	1	0	0
C_2	1	1	1	1	0
C_3	0	1	1	0	1
C_4	0	0	1	0	0

(e)

Figure 2 (a) - (c) Three distinct CLTs on a common leafset. The left and right CLTs are binary, while the middle one is not (it can be obtained by removing clade $\{C_1, C_2\}$ from the left CLT, or clade $\{C_1, C_3\}$ from the right CLT). **(d)** A conflict-free genotype matrix that is consistent only with the left CLT – there is no edge where M_1 may be placed in (b) or in (c) so it is present only in C_1 and C_2. In fact, the left CLT is the unique CLT yielded by this matrix. **(e)** A conflict-free genotype matrix that is consistent with all three CLTs, however the unique tree yielded by it is the middle tree.

Figure 2 demonstrates that the notion of consistency between genotype matrices and CLTs is not sufficient to map conflict-free genotype matrices to CLTs as is desired for our approach to computing the Bi-partition Function. The matrix in Figure 2e is consistent with multiple CLTs, therefore we cannot use consistency to map it to a single CLT. This motivates us to describe a more restricted relationship between genotype matrix G and a CLT T.

▶ **Definition 3.** *A genotype matrix G yields a CLT T if G is consistent with T and every nontrivial clade of T corresponds to a mutation in G. The set of all genotype matrices which yield T is denoted as $Y(T)$.*

This notion of matrices yielding trees allows us to map conflict-free genotype matrices to CLTs since every conflict-free genotype matrix yields a unique CLT (see Section A.2 for a proof). For example, the matrices in Figures 2d and 2e yield the CLTs in Figures 2a and 2b respectively.

Observe that, in order to be consistent with or yield a CLT, G has to be conflict-free.

3.3 Evaluating the Bi-partition Function

We first provide a high-level description of how we evaluate the Bi-partition Function using Equation (1), by separately computing the numerator and denominator.

In order to compute the numerator (resp. denominator), we enumerate every possible CLT T on n nodes and compute the contribution of each such T to the numerator (resp. denominator) of Equation (1), expressed as:

$$\begin{aligned}\texttt{numerator\_contribution}(T, R, \rho) &= \Pr\left[\mathbf{X}_\rho = R \wedge \mathbf{X} \in Y(T)\right] \\ \texttt{denominator\_contribution}(T) &= \Pr\left[\mathbf{X} \in Y(T)\right]\end{aligned} \tag{2}$$

Since every conflict-free genotype matrix yields exactly one CLT, and every CLT is yielded by some conflict-free matrix, the probability space of conflict-free matrices can be partitioned by the CLTs that the matrices yield. Thus, we can simply add up the `numerator_contribution` for each possible T in order obtain the numerator of Equation (1). We can calculate the denominator similarly.

In order to enumerate all CLTs, we use a brute force algorithm that begins from the set of leaves and creates all possible partitions of those into two sets, then three sets, and so on. Each set in each partition is then partitioned recursively, terminating with singleton sets. This effectively enumerates all unique sets of nontrivial clades, allowing us to enumerate all CLTs.

3.3.1 Evaluating the Contribution of Each CLT T

We now show how each CLT T that we enumerate contributes to Equation (1).

Since a CLT is uniquely defined by its nontrivial clades, we represent T as the set of its nontrivial clades (thus c is a nontrivial clade). As before, we treat column $\mathbf{X}_i$ of $\mathbf{X}$ as the set of row indices where $\mathbf{X}_i$ has a 1.

$$\begin{aligned}&\Pr\left[\mathbf{X} \in Y(T)\right] = \Pr\left[\mathbf{X} \in CM(T) \wedge \forall c \in T, \exists j \in [m] \text{ s.t. } c = \mathbf{X}_j\right] \\ &= \Pr[\mathbf{X} \in CM(T)] - \Pr\left[\mathbf{X} \in CM(T) \wedge \exists c \in T \text{ s.t. } \forall j \in [m]\ c \neq \mathbf{X}_j\right],\end{aligned} \tag{3}$$

where

$$\begin{aligned}&\Pr\left[\mathbf{X} \in CM(T) \wedge \exists c \in T, \forall j \in [m] \text{ s.t. } c \neq \mathbf{X}_j\right] \\ &= \Pr\left[\bigvee_{c \in T} (\mathbf{X} \in CM(T) \wedge \forall j \in [m], c \neq \mathbf{X}_j)\right] \\ &= \Pr\left[\bigvee_{c \in T} \mathbf{X} \in CM(T \setminus \{c\})\right].\end{aligned} \tag{4}$$

Removing clade c from T collapses the edge between (u, v), where v is the root of clade c, and u is the parent of v (see Figure 2b which is the result of removing the clade with nodes C_1, C_2 from the tree in Figure 2a, or removing the clade with nodes C_1, C_3 from Figure 2c).

If $S \subseteq T$, then $T \setminus S$ represents the result of removing every clade in S from T. Let $\mathcal{P}(T)$ denote the powerset of T. By the inclusion-exclusion principle,

$$\begin{aligned}\Pr\left[\bigvee_{c\in T} \mathbf{X} \in CM(T\setminus\{c\})\right] &= \sum_{S\in\mathcal{P}(T)} \Pr\left[\mathbf{X}\in CM(T\setminus S)\right]\times(-1)^{|S|+1}\\ &= \sum_{c\in T}\Pr[\mathbf{X}\in CM(T\setminus\{c\})]\\ &\quad - \sum_{c_1,c_2\in\binom{T}{2}}\Pr[\mathbf{X}\in CM(T\setminus\{c_1,c_2\})]\\ &\quad + \sum_{c_1,c_2,c_3\in\binom{T}{3}}\Pr[\mathbf{X}\in CM(T\setminus\{c_1,c_2,c_3\})]\\ &\quad - \cdots\\ &\quad \pm \Pr[\mathbf{X}\in CM(\varnothing)]. \end{aligned} \tag{5}$$

($CM(\varnothing)$ is the set of matrices consistent with a tree that has no nontrivial clades, the star tree.)

The probability that a matrix is consistent with a tree T can be computed easily.

$$\begin{aligned}&\Pr[\mathbf{X}\in CM(T)] = \Pr\left[\forall j\in[m], \exists c\in T \text{ s.t. } \mathbf{X}_j = c\right]\\ &= \Pr\left[\bigwedge_{j\in[m]}\bigvee_{c\in T}\mathbf{X}_j = c\right]\\ &= \prod_{j\in[m]}\Pr\left[\bigvee_{c\in T}\mathbf{X}_j = c\right] \text{ because mutations are independent}\\ &= \prod_{j\in[m]}\sum_{c\in T}\Pr\left[\mathbf{X}_j = c\right] \text{ because events } \mathbf{X}_j = c \text{ are disjoint}\end{aligned} \tag{6}$$

With some abuse of a standard notation, let $\mathbf{X}\setminus\{\mathbf{X}_\rho\}$ represents the matrix resulting from deleting column ρ from $\mathbf{X}$. More precisely, $\mathbf{X}\setminus\{\mathbf{X}_\rho\} = (\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_{\rho-1}, \mathbf{X}_{\rho+1}, \ldots, \mathbf{X}_m)$. Next, let $Y'(T)$ be the set of $n\times(m-1)$ binary matrices which yield T. We have three disjoint cases when calculating $\texttt{numerator\_contribution}(T, R, \rho) = \Pr\left[\mathbf{X}_\rho = R \wedge \mathbf{X}\in Y(T)\right]$.

Firstly, if R is not a clade of T, then the $\Pr\left[\mathbf{X}_\rho = R\wedge\mathbf{X}\in Y(T)\right] = 0$ since $\mathbf{X}\in Y(T)$ implies that every column of $\mathbf{X}$ appears as a clade of T, which is contradicted by $\mathbf{X}_\rho$ being equal to a clade not found in T.

Then, if R is a clade of T we have two sub-cases. If R is a trivial clade, then if $\mathbf{X}_\rho = R$, column ρ does not impact whether or not $\mathbf{X}$ yields T since this is only determined by the nontrivial clades which may have mutations assigned. Therefore, $\mathbf{X}_\rho = R\wedge\mathbf{X}\in Y(T)$ if and only if $\mathbf{X}_\rho = R\wedge\mathbf{X}\setminus\{\mathbf{X}_\rho\}\in Y'(T)$ since whether or not $\mathbf{X}\in Y(T)$ is entirely determined by the other columns of the matrix (excluding column ρ). And since $\mathbf{X}_\rho$ and $\mathbf{X}\setminus\{\mathbf{X}_\rho\}$ are disjoint, $\mathbf{X}_\rho = R$ and $\mathbf{X}\setminus\{\mathbf{X}_\rho\}\in Y'(T)$ are independent events (since entries of the matrix are independent). Therefore we have $\Pr\left[\mathbf{X}_\rho = R\wedge\mathbf{X}\in Y(T)\right] = \Pr\left[\mathbf{X}_\rho = R\right]\times\Pr\left[\mathbf{X}\setminus\{\mathbf{X}_\rho\}\in Y'(T)\right]$ when R is a trivial clade in T.

Finally, we have the case arising when R is a nontrivial clade of T. This case differs slightly from the previous case because $\mathbf{X}_\rho$ does contribute toward $\mathbf{X}$ yielding T. If the rest of the matrix yields T (i.e., $\mathbf{X}\setminus\{\mathbf{X}_\rho\}\in Y'(T)$) then $\mathbf{X}$ yields T as in the previous case.

However, since $\mathbf{X}_\rho = R$, if the rest of the matrix yields instead the tree containing every nontrivial clade of T except for R (i.e., $\mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T \setminus \{R\})$), then $\mathbf{X}_\rho$ being equal to R adds this clade to the tree yielded by the rest of the matrix, so the entire matrix yields T in this case as well. Therefore when R is a nontrivial clade present in T we have:

$$\begin{aligned}
&\mathbf{X}_\rho = R \wedge \mathbf{X} \in Y(T) \\
\Leftrightarrow\ &\left(\mathbf{X}_\rho = R \wedge \mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T \setminus \{R\})\right) \vee \left(\mathbf{X}_\rho = R \wedge \mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T)\right) \\
\Leftrightarrow\ &\mathbf{X}_\rho = R \wedge \left(\mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T \setminus \{R\}) \vee \mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T)\right)
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\Pr\left[\mathbf{X}_\rho = R \wedge \mathbf{X} \in Y(T)\right] \\
&= \Pr\left[\mathbf{X}_\rho = R\right] \times \left(\Pr\left[\mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T \setminus \{R\})\right] + \Pr\left[\mathbf{X} \setminus \{\mathbf{X}_\rho\} \in Y'(T)\right]\right)
\end{aligned}$$

when R is a nontrivial clade of T.

Equations (3)–(6) show how to compute `denominator_contribution`(T), and from these three cases it can be seen that the computation of `numerator_contribution`(T, R, ρ) reduces to at most two computations of `denominator_contribution`(T).

3.3.2 Putting it All Together

We now consider the whole of Equation (1); let $CLT(n)$ be the set of all CLTs with n leaves. Recalling that the probability space of conflict-free matrices can be partitioned by which CLTs the matrices yield, we have the following:

$$\Pr\left[\mathbf{X}_\rho = R \wedge \mathbf{X} \in \mathcal{G}\right] = \sum_{T \in CLT(n)} \texttt{numerator\_contribution}(T, R, \rho)$$

$$\Pr\left[\mathbf{X} \in \mathcal{G}\right] = \sum_{T \in CLT(n)} \texttt{denominator\_contribution}(T) \tag{7}$$

by the law of total probability. This allows us to compute both the numerator and denominator, and, thus, the value of the RHS of Equation (1).

Algorithm 1: Denominator Contribution and Algorithm 2: Numerator Contribution compute the value that each T contributes to Equation (2). These contributions can be combined according to Equation (7) in order to obtain the numerator and denominator of Equation (1) (see Section A.3 for the algorithms). The overall time complexity of computation of Bi-partition function values using our algorithm is $O\left((n! \times 8^n) \times m\right)$ (see Section A.4 for derivation).

3.4 GPU-Accelerated Implementation

We provide a new implementation for computing the probability that the ground truth matrix is consistent with a given CLT (the quanity in Equation (6)). Since this calculation is simply a product of sums it may be naturally represented using matrix and vector operations. This allows us to take advantage of modern GPU's capability to massively parallelize such operations by stacking matrices on top of each other into tensors. We then replace the probability calculation algorithm used in the existing sampling based approach for estimating the Bi-partition Function with the GPU accelerated version.

4 Experimental Results

We evaluated our algorithms on synthetic datasets generated from simulated tumor progression trees, systematically varying the number of cells, mutations, false-negative rates, and missing entry rates, while keeping the false-positive rate fixed at 0.001.

First, we analyzed the runtime performance of our exact algorithm. Although the theoretical runtime complexity is linear in m and super-exponential in n, empirical results demonstrate a sublinear dependence on the number of mutations (Table 1), highlighting its potential use for benchmarking purposes.

We then used our exact algorithm to evaluate the accuracy of our GPU-accelerated sampling algorithm (Figures 3 and 4). As expected, the results clearly show accuracy improvements with increased sampling size. Importantly, the algorithm's accuracy significantly improves as the number of mutations increase. Additionally, the missing entry rate influences accuracy more substantially than the false-negative rate, suggesting the relative robustness of our sampling method to sequencing noise in contrast to coverage.

Finally, we assessed runtime improvements offered by our GPU-accelerated sampling algorithm compared to the original sampling method (Figure 5) on real mouse melanoma cell line data with 24 single cell derived sublines which were subjected to whole exome sequencing, which was used to evaluate the original sampling algorithm [11]. Note that both the original algorithm and the GPU-accelerated implementation are comprised of two major steps: tree sampling and probability calculation. Given that the probability calculation step dominates runtime in the original algorithm [11], we specifically evaluated the performance enhancement in this step. Remarkably, our GPU-accelerated implementation was observed to process approximately five orders of magnitude more trees within the same time frame, substantially improving scalability and enabling analysis of larger and more realistic datasets.

5 Conclusion

In this study, we introduced the first exact algorithm for computing the Bi-partition Function, a critical metric for assessing tumor phylogenies inferred from single-cell mutation data. Additionally, we developed a GPU-accelerated implementation of the existing sampling-based method for estimating the Bi-partition Function. Our evaluation using simulated datasets demonstrated that the exact algorithm's runtime scales super-polynomially with the number of cells, yet sublinearly with the number of mutations, highlighting its practicality primarily for smaller datasets or benchmarking purposes. On a real tumor cell line dataset, the GPU-accelerated sampling algorithm showed significant runtime improvements compared to the original implementation, making it suitable for larger, realistic inputs. As expected, its accuracy increases notably with a higher number of sampled trees. Importantly, we found that the accuracy improved substantially as the number of mutations in the simulated phylogenies increased, whereas false negative and missing entry rates had comparatively moderate impact.

Table 1 Table of average runtimes (in seconds) of the exact algorithm on synthetic inputs of varying size. Each row represents the number of cells ($n = 5$, 6, 7, or 8) while each column represents the number of mutations ($m = 8$, 20, 50, 100, 500, or 1000) used in the simulations. Each entry of the table presents the mean runtime of the exact algorithm across all mutations with 10 replicate simulations (i.e., when $m = 8$ it is the mean of 80 runs of the algorithm, when $m = 20$ it is the mean of 200 and so on). The runtime quickly blows up as n increases (as can be expected since the runtime of the exact algorithm is super-exponential in n), however when n is fixed the algorithm scales well as the number of mutations increase.

	8	20	50	100	500	1000
5	0.19	0.20	0.22	0.22	0.27	0.70
6	4.18	4.00	4.17	4.50	7.11	12.17
7	92.50	90.25	100.10	102.02	133.26	341.11
8	3325.37	3332.92	3709.91	3493.59	4892.56	11526.86

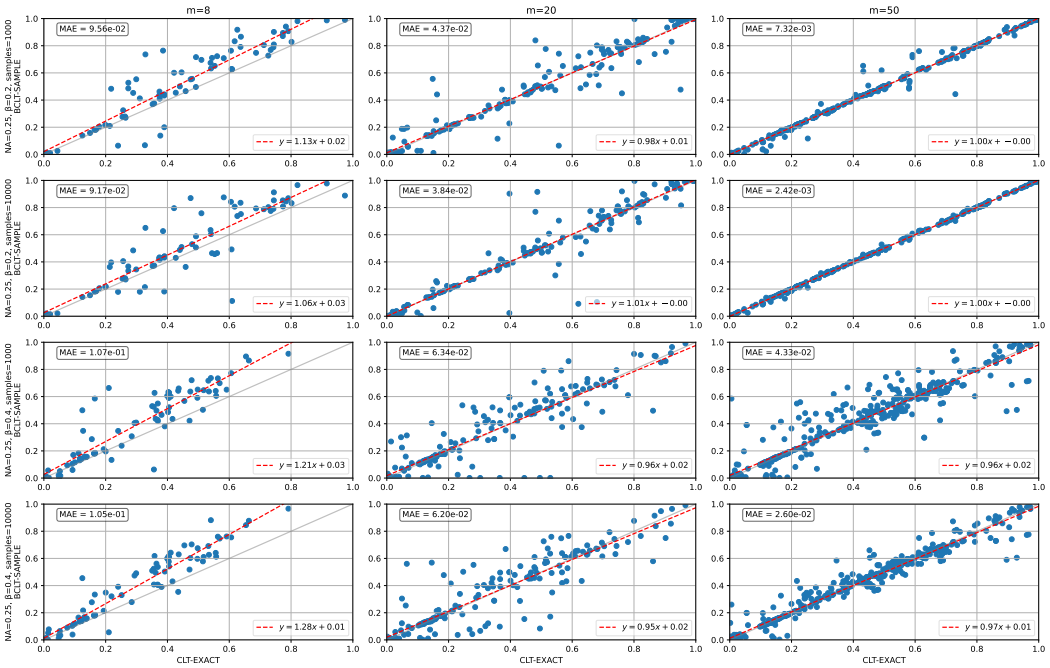


Figure 3 Accuracy of our GPU-accelerated sampling algorithm (y-axis: BCLT-SAMPLE) compared to the exact Bi-partition Function values obtained using our exact algorithm (x-axis: CLT-EXACT) on synthetic datasets. Each dataset consists of $n = 8$ cells with mutation counts varying across $m = 8, 20, 50$. The synthetic data was obtained with a simulated missing entry rate of 0.25 (in contrast to 0.5 in Figure 4), and false negative rate varying between $\beta = 0.2, 0.4$. For each setting, 10 replicate simulations were performed. The accuracy of the sampling algorithm is evaluated using $1,000$ and $10,000$ sampled trees per scenario. Each datapoint represents a comparison of the sampling algorithm's estimated Bi-partition Function for a specific mutation and its associated clade (in the ground truth tree) against the exact calculation. Points closer to the diagonal line ($y = x$) indicate higher accuracy. Each plot is annotated with mean absolute error (MAE) between the values of CLT-EXACT and BCLT-SAMPLE as well as a line of regression.

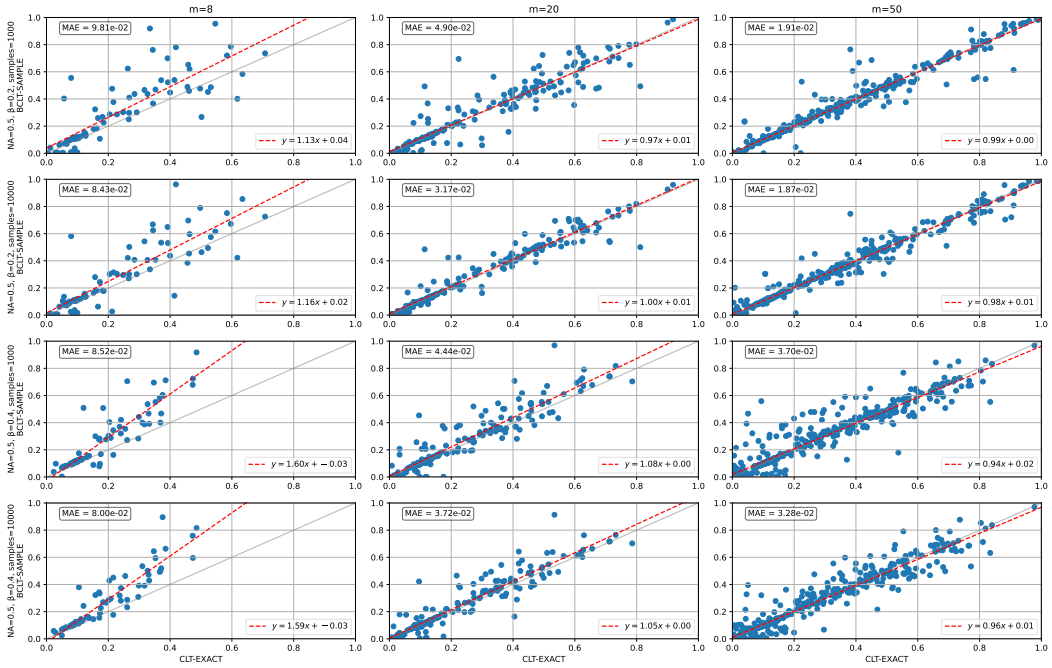


Figure 4 Accuracy of our GPU-accelerated sampling algorithm (y-axis: BCLT-SAMPLE) compared to the exact Bi-partition Function values obtained using our exact algorithm (x-axis: CLT-EXACT) on synthetic datasets. Each dataset consists of $n = 8$ cells with mutation counts varying across $m = 8, 20, 50$. The synthetic data was obtained with a simulated missing entry rate of 0.5 (in contrast to 0.25 in Figure 3), and false negative rate varying between $\beta = 0.2, 0.4$. For each setting, 10 replicate simulations were performed. The accuracy of the sampling algorithm is evaluated using $1,000$ and $10,000$ sampled trees per scenario. Each datapoint represents a comparison of the sampling algorithm's estimated Bi-partition Function for a specific mutation and its associated clade (in the ground truth tree) against the exact calculation. Points closer to the diagonal line ($y = x$) indicate higher accuracy. Each plot is annotated with mean absolute error (MAE) between the values of CLT-EXACT and BCLT-SAMPLE as well as a line of regression.

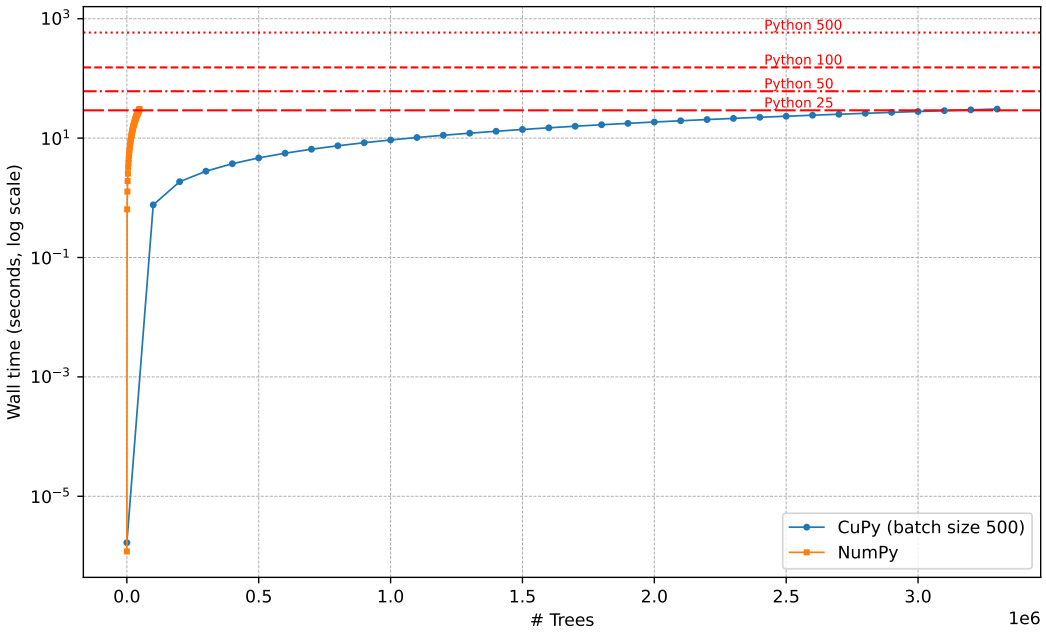


Figure 5 Wall clock runtime (seconds) of the GPU-accelerated tree scoring (CuPy) compared to both the original sampling algorithm and the accelerated algorithm when run on CPU (NumPy) on a real melanoma cell line dataset [11]. The x-axis represents the number of trees scored for the blue and orange line (CuPy and Numpy). The red horizontal lines represent the time it took the original sampling algorithm implemented using iterative statements in Python to score 25, 50, 100, and 500 trees respectively. We see that the GPU-accelerated algorithm can score millions of trees in time it took the original implementation to score just 25 trees. Both the NumPy and CuPy implementations were allowed to run for 30 seconds.

References

1 Mohammadamin Edrisi, Hamim Zafar, and Luay Nakhleh. A Combinatorial Approach for Single-cell Variant Detection via Phylogenetic Inference. In Katharina T. Huber and Dan Gusfield, editors, *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*, volume 143 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.WABI.2019.22.

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

3 Dan Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21(1):19–28, 1991. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230210104. doi:10.1002/net.3230210104.

4 Katharina Jahn, Jack Kuipers, and Niko Beerenwinkel. Tree inference for single-cell data. *Genome Biology*, 17(1), May 2016. doi:10.1186/s13059-016-0936-x.

5 Can Kızılkale, Farid Rashidi Mehrabadi, Erfan Sadeqi Azer, Eva Pérez-Guijarro, Kerrie L. Marie, Maxwell P. Lee, Chi-Ping Day, Glenn Merlino, Funda Ergün, Aydın Buluç, S. Cenk Sahinalp, and Salem Malikić. Fast intratumor heterogeneity inference from single-cell sequencing data. *Nature Computational Science*, 2(9):577–583, September 2022. doi:10.1038/s43588-022-00298-x.

6 Alexey Kozlov, Joao M. Alves, Alexandros Stamatakis, and David Posada. CellPhy: accurate and fast probabilistic inference of single-cell phylogenies from scDNA-seq data. *Genome Biology*, 23(1), January 2022. doi:10.1186/s13059-021-02583-w.

7 Marco L. Leung, Alexander Davis, Ruli Gao, Anna Casasent, Yong Wang, Emi Sei, Eduardo Vilar, Dipen Maru, Scott Kopetz, and Nicholas E. Navin. Single-cell DNA sequencing reveals a late-dissemination model in metastatic colorectal cancer. *Genome Research*, 27(8):1287–1299, May 2017. doi:10.1101/gr.209973.116.

8 Salem Malikic, Katharina Jahn, Jack Kuipers, S. Cenk Sahinalp, and Niko Beerenwinkel. Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. *Nature Communications*, 10(1), June 2019. doi:10.1038/s41467-019-10737-5.

9 Salem Malikić, Farid Rashidi Mehrabadi, Erfan Sadeqi Azer, Mohammad Haghir Ebrahimabadi, and Suleyman Cenk Sahinalp. Studying the History of Tumor Evolution from Single-Cell Sequencing Data by Exploring the Space of Binary Matrices. *Journal of Computational Biology*, July 2021. doi:10.1089/cmb.2020.0595.

10 Salem Malikic, Farid Rashidi Mehrabadi, Simone Ciccolella, Md. Khaledur Rahman, Camir Ricketts, Ehsan Haghshenas, Daniel Seidman, Faraz Hach, Iman Hajirasouliha, and S. Cenk 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, October 2019. doi:10.1101/gr.234435.118.

11 Farid Rashidi Mehrabadi, Erfan Sadeqi Azer, John D Bridgers, Eva Pérez-Guijarro, Kerrie L Marie, Howard H Yang, Charli Gruen, Chih Hao Wu, Welles Robinson, Huaitian Liu, et al. A partition function algorithm to evaluate inferred subclonal structures in single-cell sequencing data. In *Research in Computational Molecular Biology: 29th International Conference, RECOMB 2025, Seoul, South Korea, April 26–29, 2025, Proceedings*, volume 15647 of *Lecture Notes in Computer Science*, pages 409–413. Springer, 2025. doi:10.1007/978-3-031-90252-9.

12 Christopher A. Miller, Brian S. White, Nathan D. Dees, Malachi Griffith, John S. Welch, Obi L. Griffith, Ravi Vij, Michael H. Tomasson, Timothy A. Graubert, Matthew J. Walter, Matthew J. Ellis, William Schierding, John F. DiPersio, Timothy J. Ley, Elaine R. Mardis, Richard K. Wilson, and Li Ding. SciClone: Inferring Clonal Architecture and Tracking the Spatial and Temporal Patterns of Tumor Evolution. *PLoS Computational Biology*, 10(8):e1003665, August 2014. doi:10.1371/journal.pcbi.1003665.

13 Peter C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194(4260):23–28, October 1976. doi:10.1126/science.959840.

14 Erfan Sadeqi Azer, Farid Rashidi Mehrabadi, Salem Malikić, Xuan Cindy Li, Osnat Bartok, Kevin Litchfield, Ronen Levy, Yardena Samuels, Alejandro A Schäffer, E Michael Gertz, Chi-Ping Day, Eva Pérez-Guijarro, Kerrie Marie, Maxwell P Lee, Glenn Merlino, Funda Ergun, and S Cenk Sahinalp. PhISCS-BnB: a fast branch and bound algorithm for the perfect tumor phylogeny reconstruction problem. *Bioinformatics*, 36(Supplement_1):i169–i176, July 2020. doi:10.1093/bioinformatics/btaa464.

15 Gryte Satas, Simone Zaccaria, Geoffrey Mon, and Benjamin J. Raphael. SCARLET: Single-Cell Tumor Phylogeny Inference with Copy-Number Constrained Mutation Losses. *Cell Systems*, 10(4):323–332.e8, April 2020. doi:10.1016/j.cels.2020.04.001.

16 Jochen Singer, Jack Kuipers, Katharina Jahn, and Niko Beerenwinkel. Single-cell mutation identification via phylogenetic inference. *Nature Communications*, 9(1), December 2018. doi:10.1038/s41467-018-07627-7.

17 Leah L Weber, Chuanyi Zhang, Idoia Ochoa, and Mohammed El-Kebir. Phertilizer: Growing a clonal tree from ultra-low coverage single-cell dna sequencing of tumors. *PLoS Computational Biology*, 19(10):e1011544, 2023. doi:10.1371/JOURNAL.PCBI.1011544.

18 Yufeng Wu. Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. *Bioinformatics*, August 2019. doi:10.1093/bioinformatics/btz676.

A Appendix

A.1 Constructing Matrix P of Probabilities

Let α and β denote the false positive and false negative rates, respectively, of our single-cell sequencing data. For a given ground truth matrix $\boldsymbol{X}$, we have the following probabilities of observing the entries in I:

$$Pr(I_{ij} = 0 \mid \boldsymbol{X}_{ij} = 0) = (1-\alpha) \qquad Pr(I_{ij} = 0 \mid \boldsymbol{X}_{ij} = 1) = \beta$$
$$Pr(I_{ij} = 1 \mid \boldsymbol{X}_{ij} = 0) = \alpha \qquad Pr(I_{ij} = 1 \mid \boldsymbol{X}_{ij} = 1) = (1-\beta).$$

Since we have access to I and not to the ground truth, we are interested in reconstructing $\boldsymbol{X}$ from I. Consider an arbitrary I_{ij} of matrix I where $I_{ij} = 1$. Assuming that $\boldsymbol{X}_{ij}$ is equally likely to take values 0 or 1 before observing I, we can use Bayes' rule to calculate P_{ij}:

$$\begin{aligned} P_{ij} &= Pr[\boldsymbol{X}_{ij} = 1 | I_{ij} = 1] \\ &= \frac{Pr[I_{ij} = 1 | \boldsymbol{X}_{ij} = 1] Pr[\boldsymbol{X}_{ij} = 1]}{Pr[I_{ij} = 1 | \boldsymbol{X}_{ij} = 0] Pr[\boldsymbol{X}_{ij} = 0] + Pr[I_{ij} = 1 | \boldsymbol{X}_{i,j} = 1] Pr[\boldsymbol{X}_{ij} = 1]} \\ &= \frac{0.5 \cdot Pr[I_{ij} = 1 | \boldsymbol{X}_{ij} = 1]}{0.5 \cdot Pr[I_{ij} = 1 | \boldsymbol{X}_{ij} = 0] + 0.5 \cdot Pr[I_{ij} = 1 | \boldsymbol{X}_{ij} = 1]} \\ &= \frac{1-\beta}{\alpha + 1 - \beta}. \end{aligned}$$

Similarly, when $I_{ij} = 0$, we have $P_{ij} = Pr[\boldsymbol{X}_{ij} = 1 | I_{ij} = 0] = \frac{\beta}{\beta + 1 - \alpha}$. For missing entries, that is when $I_{ij} =?$, we assume that there is no informative prior on the true genotype state $\boldsymbol{X}_{ij}$ so we set $Pr(\boldsymbol{X}_{ij} = 1 \mid I_{ij} =?) = Pr(\boldsymbol{X}_{ij} = 0 \mid I_{ij} =?) = \frac{1}{2}$.

Note that there are other ways to obtain the matrix P from single-cell sequencing data, which may, for example, incorporate read counts for each cell-mutation pair [18, 16]. Our algorithms presented in this work are applicable to any such matrix P.

A.2 Proofs

▶ **Lemma 4.** *The set of conflict-free $n \times m$ genotype matrices $\mathcal{G}$ may be partitioned (i.e., assigned to mutually exclusive sets whose union is equal $\mathcal{G}$) by which n leaf CLTs they yield.*

Proof. To prove this we must show that (i) any conflict-free genotype matrix yields a CLT and (ii) the CLT yielded by a matrix is unique. If we create sets of matrices $Y(T)$ for each n leaf CLT T that each contain the matrices which yield T, then (i) implies that the union of these sets equals $\mathcal{G}$ and (ii) implies that these sets are mutually exclusive.

Let $X \in \mathcal{G}$ be a conflict-free genotype matrix. Recall that by the definition of conflict-free matrices X is consistent with some CLT T. Now assign the mutations of X to the edges of T such that every mutation is placed on the edge directed into the node (assuming that CLTs are represented as a directed graph with the direction of edges pointing away from the root) representing the most recent common ancestor of the leaves with that mutation (i.e., the node rooting the subtree containing the leaves with that mutation and no other leaves). Now, consider the edges which are not directed into a leaf, or attached to the root. If each of these edges has at least one mutation, then T is yielded by X. If not, contract each edge which has no mutations. To contract edge $e = (u \to v)$, delete e and add edges directed from u to each of the children of v. Once all the mutation-less edges have been contracted, the result will be a CLT that is consistent with T and has all internal edges annotated with mutations of X, therefore X yields T.

To show uniqueness, suppose that X yields two distinct CLTs T_1 and T_2. Since $T_1 \neq T_2$ there is some nontrivial clade C in one that is not in the other. Without loss of generality, suppose that $C \in T_1$ and $C \notin T_2$. Since X yields T_1, every nontrivial clade of T_1 has some mutation/column of X which is equivalent to it, therefore there exists $i \in [m]$ such that $X_i = C$. However, $C \notin T_2$, therefore there is no edge in T_2 to which mutation M_i from X may be assigned, therefore X is not consistent with T_2. This is a contradiction since T_2 is yielded by X, and X yielding a tree implies that X is consistent with it. ◀

A.3 Algorithms

Algorithm 1: Denominator Contribution Algorithm to compute the contribution of T to the denominator of Equation (1).

Input: Matrix $P \in [0,1]^{n \times m}$ from which $\mathbf{X}$ is drawn, CLT T (represented as a set of nontrivial clades)
Output: $\Pr[\mathbf{X} \in Y(T)]$

$p \leftarrow 0$
for $S \in \mathcal{P}(T)$ **do**
    $T' \leftarrow T \setminus S$
    $p \leftarrow p + (-1)^{|s|+1} \times \Pr[\mathbf{X} \in CM(T')]$
return p

A.4 Time Complexity

The number of CLTs with n leaves grows superexponentially in n; therefore, the runtime of the algorithm does as well since we enumerate every CLT. However, the algorithm remains linear in the number of mutations m. This enables us to compute the Bi-partition Function as m grows provided that n remains small (we have taken $n = 8$ in our experiments with

Algorithm 2: Numerator Contribution Algorithm to compute the contribution of T to the numerator of Equation (1).

Input: Matrix $P \in [0,1]^{n \times m}$ from which $\mathbf{X}$ is drawn, CLT T (represented as a set of nontrivial clades), clade R, mutation ρ
Output: $\Pr[\mathbf{X}_\rho = R \wedge \mathbf{X} \in Y(T)]$

```
if R is a trivial clade then
    p ← Pr[X \ {X_ρ} ∈ Y'(T)]
else if R ∈ T then
    p ← Pr[X \ {X_ρ} ∈ Y'(T)] + Pr[X \ {X_ρ} ∈ Y'(T \ {R})]
else
    p ← 0
p ← p × Pr[X_ρ = R]
return p
```

average runtimes of less than an hour, even with 1,000 mutations). A trivial upper bound[5] on the number of CLTs with n leaves is $2^{n-2}(2n-3)!!$ (where $k!!$ is the double factorial of k, or the product of the positive integers less than or equal to k with the same parity as k) since every CLT can be created from some binary CLT, of which there are $(2n-3)!!$, by contracting any subset of its $n-2$ non-root internal nodes. Computing `denominator_contribution`(T) for a single tree requires less than or equal to 2^{n-2} calls to the function which computes the value from Equation (6) (since a CLT with n leaves can have no more than $n-2$ nontrivial clades) which can be computed in $O(nm)$ time, therefore the time complexity of the exact algorithm is $O\left(2^{n-2} \times (2n-3)!! \times 2^{n-2} \times nm\right) \subseteq O\left((n! \times 8^n) \times m\right)$ (since $(2n-3)!! < (2n-2)!! = 2^{n-1} \times (n-1)!$).

[5] Exact values (and asymptotic behavior) of the number of CLTs with n leaves can be found in OEIS entry A000311 (`https://oeis.org/A000311`).