Network Optimization Algorithms and Applications to Molecular Biology

Alex Khodaverdian

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-262
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-262.html

December 13, 2022

Acknowledgement

First and foremost thank you to my advisor, Nir Yosef, in providing research direction, being patient with me, and just being awesome.\\

I truly believe research becomes much more exciting and much more fruitful when collaborating with other researchers, so thank you to all my colleagues and collaborators, including those from the Kuchroo lab.\\

Lastly, thank you to everyone in the Yosef Lab!

Network Optimization Algorithms and Applications to Molecular Biology

by

Alex V Khodaverdian

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nir Yosef, Chair
Professor Satish Rao
Professor Michael DuPage

Fall 2022

Network Optimization Algorithms and Applications to Molecular Biology

Abstract

Network Optimization Algorithms and Applications to Molecular Biology

by

Alex V Khodaverdian

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Nir Yosef, Chair

In this thesis, I pursue several problems in molecular biology through abstraction into network optimization algorithms.

In the first chapter of this thesis I consider our first flavor of network problems - sub-network optimization within a known dynamic network. In these instances, I introduce the notion of Condition and Time Conditioned networks, in which a network may dynamically change over time (i.e. vertices or edges). In our first set of problems, the goal becomes to find a global sub-network of minimum cost, which satisfies all local connectivity demands across conditions. In the second set of problems, I consider optimizing a singular walk demand beginning at source node a in timepoint t_1 and ending at a destination node b at a later timepoint t_2, while maintaining a notion of consistency over time. Lastly, I utilize these frameworks to investigate signal transduction in Th17 cells with the purpose of finding novel downstream proteins of interest involved in IL23 receptor signalling.

In the second chapter of this thesis, I consider the problem of lineage tracing in CRISPR/Cas9 models - given a set of terminal nodes or cells generated via CRISPR/Cas9 lineage tracing, what is the most likely tree that best represents the ground truth generative process. In particular, I begin by introducing two methods towards this analysis - a greedy approach, and an exact integer linear programming approach. I then test these methods via simulation and via ground truth trees generated in vitro. Lastly, I take a step back and consider the theoretical guarantees of our framework. That is, I explore the relationship between the number of characters/cut sites in our model against variables such as minimum cell division times, number of cells, and cutting rates. In particular I derive upper bounds for the number of characters required for exact reconstruction given perfect knowledge about the experimental setup.

In the third and final chapter of this thesis, I consider a network flow abstraction for estimating metabolic activity within cells. Given the relationship between metabolism and

immunological function, our goal becomes to discover tissue specific metabolic programs within Th17 cells. To achieve this goal, I leverage a Flux Balance Analysis approach to estimate network wide metabolic flux within Th17 cells collected from mice across various tissues, whereby I discover a novel gut specific metabolic target of interest responsible for regulating effector-like function and homeostasis.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost thank you to my advisor, Nir Yosef, in providing research direction, being patient with me, and just being awesome.

I truly believe research becomes much more exciting and much more fruitful when collaborating with other researchers, so thank you to all my colleagues and collaborators, including those from the Kuchroo lab.

Lastly, thank you to everyone in the Yosef Lab!

Chapter 1

Introduction and Background

A common approach for investigating biological systems is to first abstract the system as a network. Nodes within this network can represent a broad category of entities such as species at a higher resolution, or cell, proteins, or metabolites at a molecular resolution. Edges within this network represent varying relationships between these entities. For example within the context of lineage tracing in embryogenesis, edges represent the developmental relationship between descendent and ancestor cells, from the embryo, to the final organism of interest. Within other applications, such as protein-protein interaction networks, nodes represent proteins, while edges represent known physical binding or joint chemical reactions between proteins.

In all these instances, the underlying network represents biological information, and we leverage network optimization algorithms to further our biological understanding. These network optimization algorithms can span a plethora of problem types. In one case, the goal may be to optimize for a subnetwork of minimal cost, which best explains the relationship between a set of proteins. In another case, the goal may be to take a set of terminal nodes, and infer a possible phylogeney or tree that best explains the relationship between these entities. In yet another case, the goal may be take a well studied biological network, such a metabolic network, and infer activity amongst this network's edges given transcriptomic data. From the viewpoint of graph theory, network optimization algorithms have been well studied. However, not all of these algorithms are directly translatable to biological abstractions. Our goal, and thus the focus of this dissertation, has been to extend known network algorithms towards biological applications, with analysis of theoretical guarantees and applications to single cell and bulk transcriptomics.

1.1 Extracting Networks - Protein-protein Signalling Cascades

In molecular biology, there exist many known and well studied networks. One example in particular is in the form of protein-protein interaction networks(PPIN), where nodes represent proteins, and edges represent physical binding or joint chemical reactions. Within such abstraction, a researcher may investigate more specific questions. For example, within the PPIN, a known subset of proteins may be postulated to be active, and the goal becomes to find the most parsimonious sub-network connecting together all such proteins [89, 174, 186].

Alternatively, consider the case of cell receptor signalling. Cells commonly express a subset of protein receptors, variable by cell type and cell state. Once a receptor is stimulated via an extracellular ligand, a signalling cascade begins and through a series of protein-protein interactions and post-transnational modifications, modifies transcription factors in the nucleus. Although the start and end terminals of such signal are well known, the question becomes to find the most likely path in which the signal transduction occurred through the PPIN.

As described, these problems can be easily tackled via network optimization algorithms such as Djikstra's Algorithm in the case of a singular protein signal, or through a series of Steiner Network problems such as Steiner Tree, Steiner Forest, Directed Steiner Network, and Prize-Collecting Steiner Tree.

A natural biological extension to these problems comes in the form of the observation that biological systems are not static, but rather dynamic. In PPINs for example, proteins may simply not be present within a cell, or may not be active signalling candidates due to deactivation via post-transnational modification. As a result, a natural extension becomes to extend these optimization frameworks over dynamic networks, be it whether they are dynamic over time, or over drug treatment, or over cell type.

Several frameworks have been studied within this context of dynamic networks. One such notion comes in the form of temporal networks, whereby edges are traversed from timepoint t_1 to t_2, where $t_2 > t_1$. Several problems can be addressed over such network such as shortest path, minimum spanning tree, and discovering strongly connected components. However, these frameworks leave room for improvement. In particular, they fail to account for a totally dynamic network in both nodes and edges, and are limited in the scope of optimization queries that can be considered. In Chapter 2 of this dissertation I further explore this problem and propose a framework that I believe best addresses these limitations.

1.2 Building Networks - Single Cell Phylogenetics

The ability to construct phylogenetic trees has been of prime interest for investigators in understanding developmental relationships between related extant taxa. These phylogenetic trees can be defined over varying scales, for example representing broad processes such as speciation over organisms, or over more microscopic processes such as embryogenesis, cell differentiation, or cancer development. One such seminal example of embryogenesis comes in the work done by Sulston and colleagues, in which the entire embryogenic process of C. elegans were mapped via meticulous visual observation [44, 170]. More recent work by Yang and colleagues considers the phylodynamics, plasticity and paths of tumor evolution [193].

Recent advancements in both high-throughput single-cell sequencing and Crispr/Cas9 based lineage tracing technologies have allowed for tracking of relationships amongst more complex organisms and processes at a single cell resolution where visual observation is not feasible. In general, this process begins by engineering a single or small batch of cells with a handful of artificially inserted recording sites. Over the course of the biological process, such as embryogenesis or tumor development, these artificial recording sites accumulate irreversible heritable mutations in the form of Cas9-induced insertions or deletions. At the end of this process, the larger group of cells are collected, indels are read out via sequencing, and a phylogenetic reconstruction algorithm is used to infer clonal relationships between observed cells . These technologies have thus far enabled the study of zebrafish and mouse development, and cancer progression [129, 146, 147, 4, 165, 27, 98, 145].

When developing phylogenetic reconstruction algorithms under the Crispr/Cas9 regime, several considerations come into play. Firstly, the algorithm must be able to handle data on the scale tens of thousands or even hundreds of thousands of cells. Secondly, these algorithms must be able to deal with missing data in a subset of recording sites. Lastly, they should ideally leverage the design principles of Crispr/Cas9 lineage tracing systems, namely irreversible mutations. Traditional algorithms typically used for reconstructing phylogenies such as Camin Sokal or Neighbor Joining fall short in several of these criteria. In Chapter 3 of this dissertation, I address these concerns by proposing a new heauristic methodology for accurate reconstruction of Crispr/Cas9 lineages.

Zooming out, we can also consider the phylogenetic model used from an experimentalists perspective. That is, does the model contain theoretical guarantees that bound the amount of artificial recording sites needed for accurate reconstruction relative to the experiment's size? In other models, such as the Cavender-Farris-Neyman (CFN) 2-state model (a.k.a. binary Jukes-Cantor), such relationship has been well explored, and the number of recording sites k required is in the order of $O(\frac{log(n)}{\ell^2})$, where n is the number of cells, and ℓ is the minimum cell division time [71, 137, 41]. However, the CFN model does not readily translate to the Crispr/Cas9 setting, as it only deals with two states, and allows for reversible mutations.

This motivates the development of theoretical bounds under the Crispr/Cas9 model, which we address in Chapter 3 of this dissertation.

1.3 Inferring Network Activity - Cellular Metabolism

Cellular function and by extension abnormal cellular disease states are known to be regulated by cellular metabolism. Cellular metabolic networks are well studied, and are often abstracted as a hypergraph, where nodes represent metabolites, and hyperedges represent possible chemical reactions. The question of interest often thus becomes to understand the level of activity amongst various hyperedges, as well as within known subsystems in this network, to better characterize cellular function. For example, within effector T cells, loss of L-glutamine results in the inability for such cells to proliferate [23, 197].

To aid in the study of metabolic networks, computational frameworks, such as flux balance analysis (FBA), are often used to translate transcriptomics, coupled with network topology and stoichiometry into insillico predictions of metabolic activity within the cell. In particular, within such FBA frameworks reaction activity can be abstracted as flows over such hyperedges. In Chapter 4 of this work, we utilize Compass, the first FBA based algorithm to measure metabolic activity within single cells, to characterize cellular metabolic states based on scRNA-Sequencing within Th17 cells [178].

1.4 Dissertation Overview

In this dissertation, I present methods and tools for addressing several optimization frameworks in molecular biology.

In Chapter two, I extend the notion of the Steiner Network problems to a dynamic network. In particular I introduce the Condition Steiner Network (CSN) problem (alongside several variants). In this setting, I are given a weighted undirected graph G, a set of C conditions and a set of $k \geq C$ demands, at least one per condition. The conditions are specified over a sequence of graphs G_c defined over each condition, where vertices remain the same, but edges are allowed to change across conditions (I demonstrate that the results hold when both edges and vertices are variable). The demands are of the form $X = \{..., (u_i, v_i, c_i) ...\}$, where the goal becomes to connect u_i to v_i through a path of nodes all present in c_i. The goal is to find a minimum-weight subgraph of G that satisfies all the demands.

In addition, I extend this framework from unordered conditions to ordered time conditions to solve for a time-sensetive shortest walk problem that spans across a series of time-ordered subnetworks. In particular, I introduce the Time Conditioned Shortest Walk (TCSW), in which we are given a series of ordered networks G_t and ordered conditions $\{1, ..., T\}$ repre-

senting a discrete measurement of time, and as well as a pair of nodes ($a \in G_1, b \in G_T$). The goal is to find a walk from a source node a to target node b which begins in G_1, ends in G_T, and satisfies a notion of monotonic time consistency.

In both cases, we provide hardness of approximation results, and propose Integer Linear Program (ILP) solutions to solve instances of these problems for biological instances in feasible time.

Lastly, motivated by subnetwork optimization, we focus our attention to a case study. In particular, we investigate the biological process of signal transduction in Th17 cells via IL23 receptor signalling. Our goal within such investigation is to identify downstream genes of interest necessary for IL23 signal transduction, with the goal of tackling diseases of inflammation.

In Chapter three of this work we introduce Cassiopeia: a suite of three algorithms specifically aimed at reconstructing large phylogenies from lineage tracing experiments with special consideration for the Cas9-mutagenesis process and missing data. These algorithms range from a fast greedy algorithm, to a slow, but exact, Steiner Tree inspired algorithm, to a hybrid approach that blends the scalability of the greedy algorithm and the exactness of the Steiner tree approach to support massive single-cell lineage tracing phylogeny reconstruction.

In addition to Cassiopeia, we take a step back and consider the theoretical bounds within a single-cell CRISPR-Cas9 model. In particular, we develop two algorithms with theoretical guarantees for exact reconstruction of the underlying phylogenetic tree of a group of cells, showing that exact reconstruction can be achieved with high probability given sufficient information capacity in the experimental parameters. Using these asymptotic bounds we characterize the dependence between the necessary number of characters and other experimental parameters, such as the mutation rate (controlled by guide affinity), thus offering insight into how experimental design may be improved as the field develops.

In Chapter four of this work, we consider an application of network optimization within the metabolic model of Th17 cells within mice. For this, we use Compass, an in silico based approach, which uses cell transcriptomic data and models network wide metabolic flux using a Flux Balance Analysis (FBA). Our goal thus becomes to categorize and explore the heterogeneity of Th17 cells within various tissues at a metabolic level, with the goal of identifying metabolic programs or metabolites of interest. Our work identifies a gut-specific metabolic target creatine kinase, which we are validating experimentally in-vitro, responsible for regulating effector-like function and gut homeostasis

Chapter 2

Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling

2.1 Introduction

In this chapter, we consider the problem of subnetwork optimization, with applications to molecular biology. That is given a set of terminal nodes, our goal becomes to find the most probable or likely network which best connects these nodes, given biological constraints.

In the first section, we consider the classic Steiner tree problems extended to a multi-Condition setting. That is, given a set of conditions $c \in [C]$ (such as time, cells, etc), satisfy a set of connectivity demands in every condition, with minimal total network cost.

In the second section, we consider the classic Shortest Walk problems extended to a time-condition setting. That is, given a set of time ordered conditions, satisfy a singular connectivity demand from a source node a at timepoints t_1 to target node b at timepoint t_2, while maintaining a notion of consistency over time.

In the final section, we apply this concept to IL23 receptor signalling in Th17 cells. In particular, we contrast IL23 receptor signalling to IL12 receptor signalling in order to investigate and discover distinct and novel downstream targets of IL23 receptor.

2.2 Connectivity Problems on Heterogeneous Graphs

2.2.1 Authors and Contributions

Based on: Jimmy Wu, Alex Khodaverdian, Benjamin Weitz, Nir Yosef. Connectivity Problems on Heterogeneous Graphs. Algorithms for Molecular Biology 2019.

All authors conceived and designed the study. JW and BW derived the main hardness results. AK derived the monotonic hardness results and approximation algorithm. NY was the PI and oversaw the project.

2.2.2 Abstract

Background: Network connectivity problems are abundant in computational biology research, where graphs are used to represent a range of phenomena: from physical interactions between molecules to more abstract relationships such as gene co-expression. One common challenge in studying biological networks is the need to extract meaningful, small subgraphs out of large databases of potential interactions. A useful abstraction for this task turned out to be the Steiner network problems: given a reference "database" graph, find a parsimonious subgraph that satisfies a given set of connectivity demands. While this formulation proved useful in a number of instances, the next challenge is to account for the fact that the reference graph may not be static. This can happen for instance, when studying protein measurements in single cells or at different time points, whereby different subsets of conditions can have different protein milieu.

Results and Discussion: We introduce the *condition* Steiner network problem in which we concomitantly consider a set of distinct biological conditions. Each condition is associated with a set of connectivity demands, as well as a set of edges that are assumed to be present in that condition. The goal of this problem is to find a minimal subgraph that satisfies all the demands through paths that are present in the respective condition. We show that introducing multiple conditions as an additional factor makes this problem much harder to approximate. Specifically, we prove that for C conditions, this new problem is NP-hard to approximate to a factor of $C - \epsilon$, for every $C \geq 2$ and $\epsilon > 0$, and that this bound is tight. Moving beyond the worst case, we explore a special set of instances where the reference graph grows *monotonically* between conditions, and show that this problem admits substantially improved approximation algorithms. We also developed an integer linear programming solver for the general problem and demonstrate its ability to reach optimality with instances from the human protein interaction network.

Conclusion: Our results demonstrate that in contrast to most connectivity problems studied in computational biology, accounting for multiplicity of biological conditions adds considerable complexity, which we propose to address with a new solver. Importantly, our

results extend to several network connectivity problems that are commonly used in computational biology, such as Prize-Collecting Steiner Tree, and provide insight into the theoretical guarantees for their applications in a multiple condition setting.

Availability Our solver for the general Condition Steiner Network problem is available at `https://github.com/YosefLab/condition_connectivity_problems`

2.2.3 Background

In molecular biology applications, networks are routinely defined over a wide range of basic entities such as proteins, genes, metabolites, or drugs, which serve as nodes. The edges in these networks can have different meanings, depending on the particular context. For instance, in protein-protein interaction (PPI) networks, edges represent physical contact between proteins, either within stable multi-subunit complexes or through transient causal interactions (i.e., an edge (x, y) means that protein x can cause a change to the molecular structure of protein y and thereby alter its activity). The body of knowledge encapsulated within the human PPI network (tens of thousands of nodes and hundreds of thousands of edges in current databases, curated from thousands of studies [31]) is routinely used by computational biologists to generate hypotheses of how various signals are transduced in eukaryotic cells [15, 89, 160, 186, 198]. The basic premise is that a process that starts with a change to the activity of protein u and ends with the activity of protein v must be propagated through a chain of interactions between u and v. The natural extension regards a process with a certain collection of protein pairs $\{(u_1, v_1), \ldots, (u_k, v_k)\}$, where we are looking for a chain of interactions between each u_i and v_i [61]. In another set of applications, the notion of directionality is not directly assumed and instead, one is looking for a parsimonious subgraph that connects together a set S of proteins that are postulated to be active [89, 174].

In most applications, the identity of the so called terminal nodes (i.e., (u_i, v_i) pairs or the set S) is assumed to be known (or inferred from experimental data such as ChIP-seq [89, 174, 186]), while the identity of the intermediate nodes and interactions is unknown. The goal therefore becomes to complete the gap and find a probable subgraph of the PPI network that simultaneously satisfies all the connectivity demands, thereby explaining the overall biological activity. Since the edges in the PPI network can be assigned a probability value (reflecting the credibility of their experimental evidence), by taking the negative log of these values as edge weights, the task becomes minimizing the total edge weight, leading to an instance of the Steiner Network problem. We have previously used this approach to study the propagation of a stabilizing signal in pro-inflammatory T cells, leading to the identification of a new molecular pathway (represented by a sub-graph of the PPI network) that is critical for mounting an auto-immune response, as validated experimentally by perturbation assays and disease models in mice [186]. Tuncbag et al. [174] have utilized the undirected approach using the Prize-Collecting Steiner Tree model, where the input is a network G along with a penalty function, $p(v)$ for each protein (node) in the network (based on their importance; e.g., fold-change across conditions). The goal in this case is to find a probable subtree which

contains the majority of the high cost proteins in G, while accounting for penalties paid by both edge usage and missing proteins, in order to capture the biological activity represented in such a network [89, 174].

While these studies contributed to our understanding of signal transduction pathways in living cells, they do not account for a critical aspect of the underlying biological complexity. In reality, proteins (nodes) can become activated or inactivated at different conditions, thereby giving rise to a different set of potential PPIs that might take place [144]. Here, the term *condition* can refer to different points in time [131], different treatments [105], or, more recently, different cells [16]. Indeed, advances in experimental proteomics provide a way to estimate these changes at high throughput, e.g., measuring phosphorylation levels or overall protein abundance, proteome-wide for a limited number of samples [105]. A complementary line work provides a way to evaluate the abundance of smaller numbers of proteins (typically dozens of them) in hundreds of thousands of single cells [16].

The next challenge is therefore to study connectivity problems that take into account not only the endpoints of each demand, but also the condition in which these demands should be satisfied. This added complication was tackled by Mazza et al. [125], who introduced the "Minimum k-Labeling (MKL)" problem. In this setting, each connectivity demand comes with a label, which represents a certain experimental condition or time point. The task is to label edges in the PPI network so as to satisfy each demand using its respective label, while minimizing the number of edges in the resulting sub-graph and the number of labels used to annotate these edges. While MKL was an important first step, namely introducing the notion of different demands for each condition, the more difficult challenge still remains that of considering variability in the reference graph, namely different sets of proteins that may be active and available for use in each condition. To this effect, we note the existence of multi-layer networks in the data-mining space. In this context, studies have focused on networks which have edges that span across specified dimensions, or conditions [17, 113]. However, we could not find studies that tackle the problem of parsimonious connectivity in this domain.

2.2.4 Summary of main contributions

To address this open challenge, here we introduce the Condition Steiner Network (CSN) problem. In this setting, we are given a weighted undirected graph G, a set of C conditions and a set of $k \geq C$ demands, at least one per condition (note that we also cover the case of directed graphs, with similar results). The conditions are specified over a sequence of graphs G_c defined over each condition, where vertices remain the same, but edges are allowed to change across conditions (notably, our results also hold when G_c is defined with changing vertices rather than edges). Furthermore, demands are in the form of "connect node u to node v through a path of nodes that are present in condition c". The goal is to find a

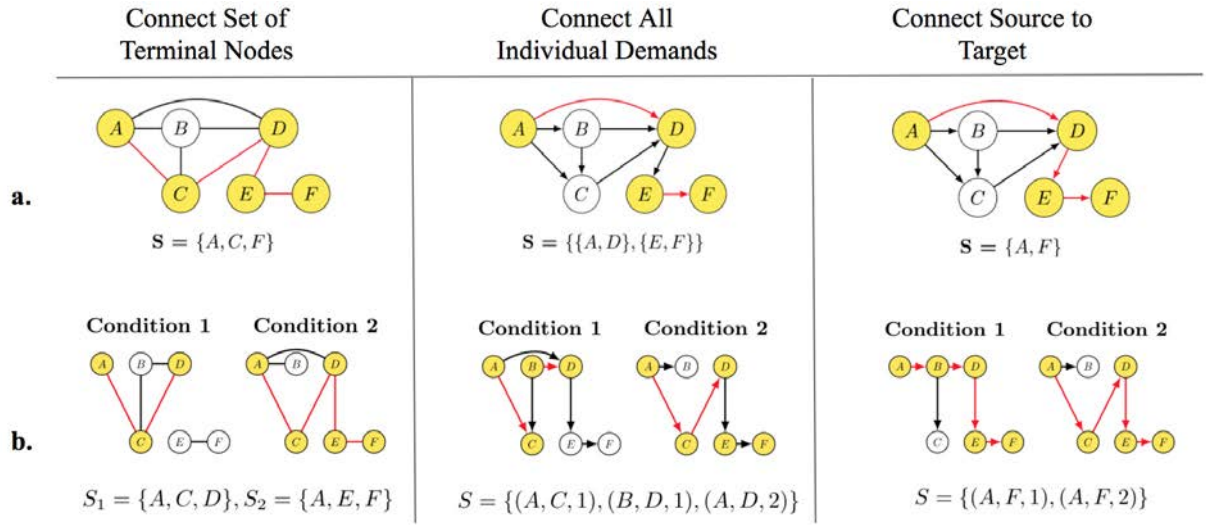


Figure 2.1: Examples of well studied network problems (**a**), and their corresponding extension with multiple conditions (**b**). The problems shown are: Undirected Steiner Tree, Directed Steiner Network, and Shortest Path, respectively. Yellow nodes and red edges correspond to nodes and edges used in the optimal solutions for the corresponding instances.

minimum-weight subgraph of G that satisfies all the demands.

We first show that it is NP-hard to find a solution that achieves a nontrivial approximation factor (by the "trivial" approximation, we mean the one obtained by solving the problem independently for each condition). This result extends to several types of connectivity problems and provides a theoretical lower bounds to the best-possible approximation guarantee that can be achieved in a multiple condition setting (Table 2.1). For instance, we can conclude that concomitantly solving the shortest path problem for a set of conditions is hard to approximate, and that the trivial solution (i.e., solving the problem to optimality in each condition) is, theoretically, the best that one can do. Another example, commonly used in PPI analysis, is the Prize-Collecting Steiner tree problem [89, 174]. Here, our results indicate that given a fixed input for this problem (i.e., a penalty function $p(v)$ for each vertex), it is NP-hard to solve it concomitantly in C conditions, such that the weight of the obtained solution is less than C times that of the optimal solution. Interestingly, a theoretical guarantee of $C \cdot (2 - \frac{2}{|V|})$[1] can be obtained by solving the problem independently for each time point.

While these results provide a somewhat pessimistic view, they rely on the assumption that the network frames G_c are arbitrary. In the last part of this paper, we show that for the specific case where the conditions can be ordered such that each condition is a subset of the next (namely, $G_c \subseteq G_{c'}$ for $c \leq c'$) then the CSN problem can be reduced to a stan-

[1] V is the set of nodes in the reference graph G

dard connectivity problem with a single condition, leading to substantially better theoretical guarantees. Finally, we develop an integer linear program for the general CSN problem, and show that provided with real-world input (namely, the human PPI) it is capable of reaching an optimal solution in a reasonable amount of time.

2.2.5 Introduction to Steiner problems

The Steiner Tree problem, along with its many variants and generalizations, form a core family of NP-hard combinatorial optimization problems. Traditionally, the input to one of these problems is a single (usually weighted) graph, along with requirements about which nodes need to be connected in some way; the goal is to pick a minimum-weight subgraph satisfying the connectivity demands.

In this paper, we offer a *multi-condition* perspective; in our setting, multiple graphs over the same vertex set (which one can think of as an initial graph changing over a set of discrete conditions), are all given as input, and the goal is to pick a subgraph satisfying condition-sensitive connectivity requirements. Our study of this problem draws motivation and techniques from several lines of research, which we briefly summarize.

Classic Steiner problems

A basic problem in graph theory is finding the *shortest path* between two nodes; this problem is efficiently solved using, for example, Dijkstra's algorithm.

A natural extension of this is the Steiner Tree problem: given a weighted undirected graph $G = (V, E)$ and a set of terminals $T \subseteq V$, find a minimum-weight subtree that connects all the nodes in T. A further generalization is Steiner Forest: given $G = (V, E)$ and a set of demand pairs $D \subseteq V \times V$, find a subgraph that connects each pair in D. Currently the best known approximation algorithms give a ratio of 1.39 for Steiner Tree [20] and 2 for Steiner Forest [3]. These problems are known to be NP-hard to approximate to within some small constant [35].

For directed graphs, we have the Directed Steiner Network (DSN) problem, in which we are given a weighted directed graph $G = (V, E)$ and k demands $(a_1, b_1), \ldots, (a_k, b_k) \in V \times V$, and must find a minimum-weight sub-graph in which each a_i has a path to b_i. When k is fixed, DSN admits a polynomial-time exact algorithm [51]. For general k, the best known approximation algorithms have ratio $O(k^{1/2+\epsilon})$ for any fixed $\epsilon > 0$ [52, 32]. On the complexity side, Dodis and Khanna [47] ruled out a polynomial-time $O(2^{\log^{1-\epsilon} n})$-approximation for this problem unless NP has quasipolynomial-time algorithms[2]. An important special case of

[2]Throughout this paper, $n := |V|$ denotes the number of nodes in the relevant graph.

DSN is Directed Steiner Tree, in which all demands have the form (r, b_i) for some root node r. This problem has an $O(k^\epsilon)$-approximation scheme [30] and a lower bound of $\Omega(\log^{2-\epsilon} n)$ [78].

Finally, a Steiner variant that has found extensive use in computational biology is the Prize-Collecting Steiner Tree problem, in which the input contains a weighted undirected graph $G = (V, E)$ and penalty function $p : V \to \mathbb{R}_{\geq 0}$; the goal is to find a subtree which simultaneously minimizes the weights of the edges in the tree and the penalties paid for nodes not included within the tree, i.e. $\text{cost}(T) := \sum_{e \in T} w(e) + \sum_{v \notin T} p(v)$. For this problem, an approximation algorithm with ratio 1.967 is known [9].

Condition Steiner problems

In this paper, we generalize the Shortest Path, Steiner Tree, Steiner Forest, Directed Steiner Network, and Prize-Collecting Steiner Tree problems to the multi-condition setting. In this setting, we have a set of *conditions* $[C] := \{1, \ldots, C\}$, and are given a graph for each condition.

Our main object of study is the natural generalization of Steiner Forest (in the undirected case) and Directed Steiner Network (in the directed case), which we call Condition Steiner Network:

Definition 1 (Condition Steiner Network (CSN))**.** *We are given the following inputs:*

1. *A sequence of undirected graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C)$, one for each* condition *$c \in [C]$. Each edge e in the underlying edge set $E := \bigcup_c E_c$ has a weight $w(e) \geq 0$.*

2. *A set of k* connectivity demands *$\mathcal{D} \subseteq V \times V \times [C]$. We assume that for every $c \in C$ there exists at least one demand and therefore that $k \geq |C|$.*

We call $G = (V, E)$ the underlying graph*. We say a subgraph $H \subseteq G$* satisfies *demand $(a, b, c) \in \mathcal{D}$ if H contains an a-b path P along which all edges exist in G_c. The goal is to output a minimum-weight subgraph $H \subseteq G$ that satisfies every demand in $\mathcal{D}$.*

Definition 2 (Directed Condition Steiner Network (DCSN))**.** *This is the same as CSN except that all the edges are directed, and a demand (a, b, c) must be satisfied by a directed path from a to b in G_c.*

We can also define the analogous generalizations of Shortest Path, (undirected) Steiner Tree, and Prize-Collecting Steiner Tree. We give hardness results and algorithms for these

problems by demonstrating reductions to and from CSN and DCSN.

Definition 3 (Condition Shortest Path (CSP), Directed Condition Shortest Path (DCSP))**.** *These are the special cases of CSN and DCSN in which the demands are precisely* $(a, b, 1), \ldots, (a, b, C)$ *where* $a, b \in V$ *are common source and target nodes.*

Definition 4 (Condition Steiner Tree (CST))**.** *We are given a sequence of undirected graphs* $G_1 = (V, E_1), \ldots, G_C = (V, E_C)$*, a weight* $w(e) \geq 0$ *on each* $e \in E$*, and sets of* terminal nodes $X_1, \ldots, X_C \subseteq V$*. We say a subgraph* $H \subseteq (V, \bigcup_c E_c)$ satisfies *the terminal set* X_c *if the nodes in* X_c *are mutually reachable using edges in* H *that exist at condition* c*. The goal is to find a minimum-weight subgraph* H *that satisfies* X_c *for every* $c \in [C]$*.*

Definition 5 (Condition Prize-Collecting Steiner Tree (CPCST))**.** *We are given a sequence of undirected graph* $G_1 = (V, E_1), \ldots, G_C = (V, E_C)$*, a weight* $w(e) \geq 0$ *on each* $e \in E$*, and a penalty* $p(v, c) \geq 0$ *for each* $v \in V, c \in [C]$*. The goal is to find a subtree* T *that minimizes* $\sum_{e \in T} w(e) + \sum_{v \notin T, c \in [C]} p(v, c)$*.*

Finally, in molecular biology applications, it is often the case that all the demands originate from a common root node. To capture this, we define the following special case of DCSN:

Definition 6 (Single-Source DCSN)**.** *This is the special case of DCSN in which the demands are precisely* $(a, b_1, c_1), (a, b_2, c_2), \ldots, (a, b_k, c_k)$*, for some* root $a \in V$*. We can assume that* $c_1 \leq c_2 \leq \cdots \leq c_k$*.*

It is also natural to consider variants of these problems in which nodes (rather than edges) vary across the conditions, or in which both nodes and edges vary. In Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we show that all three variants are in fact equivalent; thus we focus on the edge-based formulations.

2.2.6 Our results

In this work, we perform a systematic study of the condition Steiner problems defined above, from the standpoint of approximation algorithms—that is, algorithms that return subgraphs whose total weights are not much greater than that of the optimal subgraph—as well as integer linear programming (ILP). Since all of the condition Steiner problems listed in the previous section turn out to be NP-hard (and in fact all of them except Shortest Path are hard even in the classic single-condition setting) we cannot hope for algorithms that find

	Classic		*Condition*	
Problems	Lower Bound	Upper Bound	Lower Bound(s)	Upper Bound(s)
Steiner Forest	1.01 [35]	2 [3]	$C-\epsilon, k-\epsilon$	$2C, k$
Directed Steiner Network	$k^{1/4-o(1)}$ [46]	$k^{1/2+\epsilon}$ [52, 32]	$C-\epsilon, k-\epsilon$	$C \cdot k^{1/2+\epsilon}, k$
Undirected Shortest Path	N/A	1	$C-\epsilon$	C
Directed Shortest Path	N/A	1	$C-\epsilon$	C
Steiner Tree	1.01 [35]	1.39 [20]	$C-\epsilon$	$1.39C$
Prize-Collecting Steiner Tree	1.01 [35]	1.97 [9]	$C-\epsilon$	$1.97C$

Table 2.1: Approximation bounds for the various Steiner Network Problems in their classic setting and condition setting. For the classic problems, we have indicated the papers in which the bounds are shown. For the condition problems, all the lower bounds are developed in the present work; all the upper bounds are the naive bounds obtained from the "union of shortest paths" heuristic, or from applying the best known approximation algorithm for the appropriate classic Steiner problem to each condition, then taking the union of those solutions.

optimal solutions and run in polynomial time.

First, in Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we show a series of strong negative results, starting with (directed and undirected) Condition Steiner Network:

Theorem 1 (Main Theorem)**.** *CSN and DCSN are NP-hard to approximate to a factor of $C-\epsilon$ as well as $k-\epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Thus the best approximation ratio one can hope for is C or k; the latter upper bound is easily achieved by the trivial "union of shortest paths" algorithm: for each demand (a, b, c), compute the shortest a-b path at condition c; then take the union of these k paths. This contrasts with the classic Steiner network problems, which have nontrivial approximation algorithms and efficient fixed-parameter algorithms.

Next, we show similar hardness results for the other three condition Steiner problems. This is achieved by a series of simple reductions from CSN and DCSN.

Theorem 2. *Condition Shortest Path, Directed Condition Shortest Path, Condition Steiner Tree, and Condition Prize-Collecting Steiner Tree are all NP-hard to approximate to a factor of $C-\epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$.*

Note that each of these condition Steiner problems can be naively approximated by applying the best known algorithm for the classic version of that problem in each graph in the input, then taking the union of all those subgraphs. If the corresponding classic Steiner problem can be approximated to a factor of α, then this process gives an $\alpha \cdot C$-approximation for the condition version. Thus using known constant-factor approximation algorithms, each of the condition problems in Theorem 2 has an $O(C)$-approximation algorithm. Our result shows that in the worst case, one cannot do much better.

While these results provide a somewhat pessimistic view, the proofs rely on the assumption that the edge sets in the input networks (that is, $E_1, \ldots, E_C$) do not necessarily bear any relationship to one another. In Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we move beyond this worst-case assumption by studying a broad class of special cases in which the conditions are *monotonic*: if an edge e exists in some graph G_c, then it exists in all the subsequent graphs $G_{c'}, c' \geq c$. In other words, each graph in the input is a subgraph of the next. For these problems, we prove the following two theorems:

Theorem 3. *Monotonic CSN has a polynomial-time $O(\log k)$-approximation algorithm. It has no $\Omega(\log \log n)$-approximation algorithm unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log \log \log n})$.*

In the directed case, for monotonic DCSN with a single source (that is, every demand is of the form (r, b, c) for a common root node r), we show the following:

Theorem 4. *Monotonic Single-Source DCSN has a polynomial-time $O(k^\epsilon)$-approximation algorithm for every $\epsilon > 0$. It has no $\Omega(\log^{2-\epsilon} n)$-approximation algorithm unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathsf{polylog}(n)})$.*

These bounds are proved via approximation-preserving reductions to and from classic Steiner problems, namely Priority Steiner Tree and Directed Steiner Tree. Conceptually, this demonstrates that imposing the monotonicity requirement makes the condition Steiner problems much closer to their classic counterparts, allowing us to obtain algorithms with substantially better approximation guarantees.

Finally in Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we show how to model various condition Steiner problems as integer linear programs (ILPs). In experiments on real-world inputs derived from the human PPI network, we find that these ILPs are capable of reaching optimal solutions in a reasonable amount of time.

Table 2.1 summarizes our results, emphasizing how the known upper and lower bounds change when going from the classic Steiner setting to the condition Steiner setting.

2.2.7 Preliminaries

Note that the formulations of CSN and DCSN in the introduction involved a fixed vertex set; only the edges change over the conditions. It is also natural to formulate the condition Steiner network problem with nodes changing over condition, or both nodes and edges. However by the following proposition, it is no loss of generality to discuss only the edge-condition variant.

Proposition 1. *The edge, node, and node-and-edge variants of CSN are mutually polynomial-time reducible via strict reductions (i.e. preserving the approximation ratio exactly). Similarly all three variants of DCSN are mutually strictly reducible.*

We defer the precise definitions of the other two variants, as well as the proof of this proposition, to Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling.

In this edge-condition setting, it makes sense to define certain set operations on graphs, which will be of use in our proofs. To that end, let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs on the same vertex set. Their union, $G_1 \cup G_2$, is defined as $(V, E_1 \cup E_2)$. Their intersection, $G_1 \cap G_2$, is defined as $(V, E_1 \cap E_2)$. Subset relations are defined analogously; for example, if $E_1 \subseteq E_2$, then we say that $G_1 \subseteq G_2$.

Next we state the Label Cover problem, which is the starting point of one of our reductions to CSN.

Definition 7 (Label Cover (LC))**.** *An instance of this problem consists of a bipartite graph $G = (U, V, E)$ and a set of possible labels Σ. The input also includes, for each edge $(u, v) \in E$,* projection functions $\pi_u^{(u,v)} : \Sigma \to C$ *and* $\pi_v^{(u,v)} : \Sigma \to C$*, where C is a common set of colors; $\Pi = \{\pi_v^e : e \in E, v \in e\}$ is the set of all such functions. A labeling of G is a function $\phi : U \cup V \to \Sigma$ assigning each node a label. We say a labeling ϕ* satisfies *an edge $(u, v) \in E$, or (u, v) is* consistent *under ϕ, if $\pi_u^{(u,v)}(\phi(u)) = \pi_v^{(u,v)}(\phi(v))$. The task is to find a labeling that satisfies as many edges as possible.*

This problem was first defined in [11]. It has the following gap hardness, as shown by Arora et al. [10] and Raz
[149].

Theorem 5. *For every $\epsilon > 0$, there is a constant $|\Sigma|$ such that the following promise problem is NP-hard: Given a Label Cover instance (G, Σ, Π), distinguish between the following cases:*

- *(YES instance) There exists a* total labeling *of G; i.e. a labeling that satisfies every edge.*
- *(NO instance) There does not exist a labeling of G that satisfies more than $\epsilon|E|$ edges.*

In Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we use Label Cover to show $(2-\epsilon)$-hardness for 2-CSN and 2-DCSN; that is, when there are only two demands. To prove our main result however, we will actually need a generalization of Label Cover to partite hypergraphs, called k-Partite Hypergraph Label Cover. Out of space considerations we defer the statement of this problem and its gap hardness to Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, where the $(2-\epsilon)$-hardness result is generalized to show $(C-\epsilon)$-hardness and $(k-\epsilon)$-hardness for general number of conditions C and demands k.

2.2.8 Hardness of condition Steiner problems

Overview of the reduction

Here we outline our strategy for reducing Label Cover to the condition Steiner problems. First, we reduce to the CSN problem restricted to having only $C = 2$ conditions and $k = 2$ demands; we call this problem 2-CSN. The directed problem 2-DCSN is defined analogously. Later, we obtain similar hardness for CSN with more conditions or demands by using the same ideas, but reducing from k-Partite Hypergraph Label Cover.

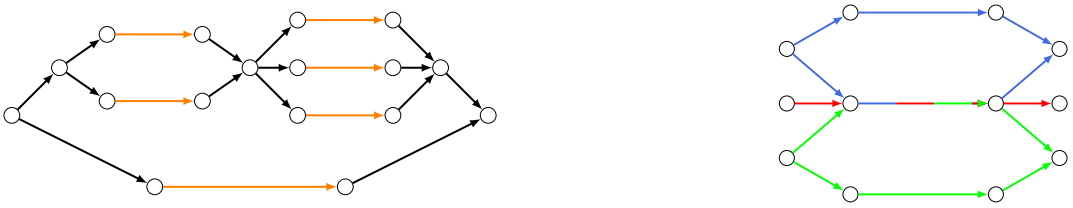


Figure 2.2: (Left) A bundle whose upper strand is a chain of two bundles; the lower strand is a simple strand. Contact edges are orange. (Right) Three bundles (blue, green, red indicate different conditions), with one strand from each merged together.

Consider the nodes $u_1, \ldots, u_{|U|}$ on the "left" side of the LC instance. We build, for each u_i, a gadget (which is a small sub-graph in the Steiner instance) consisting of multiple parallel directed paths from a source to a sink—one path for each possible label for u_i. We then chain together these gadgets, so that the sink of u_1's gadget is the source of u_2's gadget, and so forth. Finally we create a connectivity demand from the source of u_1's gadget to the sink of $u_{|U|}$'s gadget, so that a solution to the Steiner instance must have a path from u_1's gadget,

through all the other gadgets, and finally ending at $u_{|U|}$'s gadget. This path, depending on which of the parallel paths it takes through each gadget, induces a labeling of the left side of the Label Cover instance. We build an analogous chain of gadgets for the nodes on the right side of the Label Cover instance.

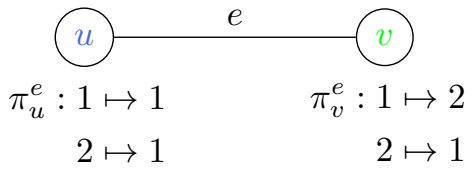
The last piece of the construction is to ensure that the Steiner instance has a low-cost solution if and only if the Label Cover instance has a consistent labeling. This is accomplished by setting all the u_i gadgets to exist only at condition 1 (i.e. in frame G_1), setting the v_j gadgets to exist only in G_2, and then merging certain edges from the u_i-gadgets with edges from the v_j-gadgets, replacing them with a single, shared edge that exists in both frames. Intuitively, the edges we merge are from paths that correspond to labels that satisfy the Label Cover edge constraints. The result is that a YES instance of Label Cover (i.e. one with a total labeling) will enable a high degree of overlap between paths in the Steiner instance, so that there is a very low-cost solution. On the other hand, a NO instance of LC will not result in much overlap between the Steiner gadgets, so every solution will be costly.

Let us define some of the building blocks of the reduction we just sketched:

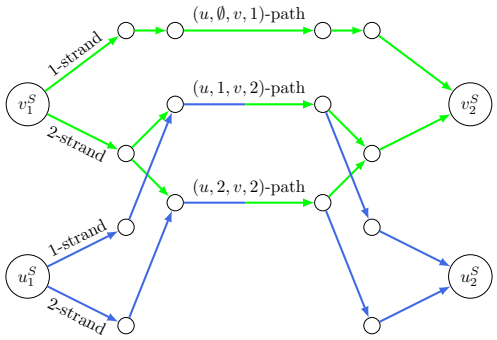
- A *simple strand* is a directed path of the form $b_1 \to c_1 \to c_2 \to b_2$.
- In a simple strand, we say that (c_1, c_2) is the *contact edge*. Contact edges have weight 1; all other edges in our construction have zero weight.
- A *bundle* is a graph gadget consisting of a source node b_1, sink node b_2, and parallel, disjoint *strands* from b_1 to b_2.
- A *chain* of bundles is a sequence of bundles, with the sink of one bundle serving as the source of another.
- More generally, a *strand* can be made more complicated, by replacing a contact edge with another bundle (or even a chain of them). In this way, bundles can be nested, as shown in Figure 2.
- We can *merge* two or more simple strands from different bundles by setting their contact edges to be the same edge, and making that edge existent at the union of all conditions when the original edges existed (Figure 2).

Before formally giving the reduction, we illustrate a simple example of its construction.

Example 1. *Consider a toy Label Cover instance whose bipartite graph is a single edge, label set is* $\Sigma = \{1, 2\}$*, color set is* $C = \{1, 2\}$*, and projection functions are shown:*



Our reduction outputs this corresponding 2-CSN instance:



G_1 comprises the set of blue edges; G_2 is green. The demands are $(u_1^S, u_2^S, 1)$ and $(v_1^S, v_2^S, 2)$. For the Label Cover node u, G_1 (the blue sub-graph) consists of two strands, one for each possible label. For the Label Cover node v, G_2 (green sub-graph) consists of one simple strand for the label '1', and a bundle for label '2', which branches out into two simple strands, one for each agreeing labeling of u. Finally, strands (more precisely, their contact edges) whose labels map to the same color are merged.

The input is a YES instance of Label Cover whose optimal labelings (u gets either label 1 or 2, v gets label 2) correspond to 2-CSN solutions of cost 1 (both G_1 and G_2 contain the $(u, 1, v, 2)$-path, and both contain the $(u, 2, v, 2)$-path). If this were a NO instance and edge e could not be satisfied, then the resulting 2-CSN sub-graphs G_1 and G_2 would have no overlap.

Inapproximability for two demands

We now formalize the reduction in the case of two conditions and two demands; later, we extend this to general C and k.

Theorem 6. *2-CSN and 2-DCSN are NP-hard to approximate to within a factor of $2 - \epsilon$ for every constant $\epsilon > 0$. For 2-DCSN, this holds even when the underlying graph is acyclic.*

Proof. Fix any desired $\epsilon > 0$. We describe a reduction from Label Cover (LC) with any parameter $\varepsilon < \epsilon$ (that is, in the case of a NO instance, no labeling satisfies more

than an ε-fraction of edges) to 2-DCSN with an acyclic graph. Given the LC instance $(G = (U, V, E), \Sigma, \Pi)$, construct a 2-DCSN instance ($\mathcal{G} = (G_1, G_2)$, along with two connectivity demands) as follows. Create nodes $u_1^S, \ldots, u_{|U|+1}^S$ and $v_1^S, \ldots, v_{|V|+1}^S$. Let there be a bundle from each u_i^S to u_{i+1}^S; we call this the u_i-*bundle*, since a choice of path from u_i^S to u_{i+1}^S in $\mathcal{G}$ will indicate a labeling of u_i in G.

The u_i-bundle has a strand for each possible label $\ell \in \Sigma$. Each of these ℓ-*strands* consists of a chain of bundles—one for each edge $(u_i, v) \in E$. Finally, each such (u_i, ℓ, v)-*bundle* has a simple strand for each label $r \in \Sigma$ such that $\pi_{u_i}^{(u_i,v)}(\ell) = \pi_v^{(u_i,v)}(r)$; call this the (u_i, ℓ, v, r)-*path*. In other words, there is ultimately a simple strand for each possible labeling of u_i's neighbor v such that the two nodes are in agreement under their mutual edge constraint. If there are no such consistent labels r, then the (u_i, ℓ, v)-bundle consists of just one simple strand, which is not associated with any r. Note that every minimal $u_1^S \to u_{|U|+1}^S$ path (that is, one that proceeds from one bundle to the next) has total weight exactly $|E|$.

Similarly, create a v_j-bundle from each v_j^S to v_{j+1}^S, whose r-strands (for $r \in \Sigma$) are each a chain of bundles, one for each $(u, v_j) \in E$. Each (u, r, v_j)-bundle has a (u, ℓ, v_j, r)-path for each agreeing labeling ℓ of the neighbor u, or a simple strand if there are no such labelings.

Set all the edges in the u_i-bundles to exist in G_1 only. Similarly the v_j-bundles exist solely in G_2. Now, for each (u, ℓ, v, r)-path in G_1, merge it with the (u, ℓ, v, r)-path in G_2, if it exists. The demands are $\mathcal{D} = \left\{ \left(u_1^S, u_{|U|+1}^S, 1 \right), \left(v_1^S, v_{|V|+1}^S, 2 \right) \right\}$.

We now analyze the reduction. The main idea is that any $u_i^S \to u_{i+1}^S$ path induces a labeling of u_i; thus the demand $\left(u_1^S, u_{|U|+1}^S, 1 \right)$ ensures that any 2-DCSN solution indicates a labeling of all of U. Similarly, $\left(v_1^S, v_{|V|+1}^S, 2 \right)$ forces an induced labeling of V. In the case of a YES instance of Label Cover, these two connectivity demands can be satisfied by taking two paths with a large amount of overlap, resulting in a low-cost 2-DCSN solution. In contrast when we start with a NO instance of Label Cover, any two paths we can choose to satisfy the 2-DCSN demands will be almost completely disjoint, resulting in a costly solution. We now fill in the details.

Suppose the Label Cover instance is a YES instance, so that there exists a labeling ℓ_u^* to each $u \in U$, and r_v^* to each $v \in V$, such that for all edges $(u, v) \in E$, $\pi_u^{(u,v)}(\ell_u^*) = \pi_v^{(u,v)}(r_v^*)$. The following is an optimal solution $\mathcal{H}^*$ to the constructed 2-DCSN instance:

- To satisfy the demand at condition 1, for each u-bundle, take a path through the ℓ_u^*-strand. In particular for each (u, ℓ_u^*, v)-bundle in that strand, traverse the (u, ℓ_u^*, v, r_v^*)-path.

- To satisfy the demand at condition 2, for each v-bundle, take a path through the r_v^*-strand. In particular for each (u, r_v^*, v)-bundle in that strand, traverse the (u, ℓ_u^*, v, r_v^*)-path.

In tallying the total edge cost, $\mathcal{H}^* \cap G_1$ (i.e. the sub-graph at condition 1) incurs a cost of $|E|$, since one contact edge in $\mathcal{G}$ is encountered for each edge in G. $\mathcal{H}^* \cap G_2$ accounts for no additional cost, since all contact edges correspond to a label which agrees with some neighbor's label, and hence were merged with the agreeing contact edge in $\mathcal{H}^* \cap G_1$. Clearly a solution of cost $|E|$ is the best possible, since every $u_1^S \to u_{|U|+1}^S$ path in G_1 (and every $v_1^S \to v_{|V|+1}^S$ path in G_2) contains at least $|E|$ contact edges.

Conversely suppose we started with a NO instance of Label Cover, so that for any labeling ℓ_u^* to u and r_v^* to v, for at least $(1-\varepsilon)|E|$ of the edges $(u, v) \in E$, we have $\pi_u^{(u,v)}(\ell_u^*) \neq \pi_v^{(u,v)}(r_v^*)$. By definition, any solution to the constructed 2-DCSN instance contains a simple $u_1^S \to u_{|U|+1}^S$ path $P_1 \in G_1$ and a simple $v_1^S \to v_{|V|+1}^S$ path $P_2 \in G_2$. P_1 alone incurs a cost of exactly $|E|$, since one contact edge in $\mathcal{G}$ is traversed for each edge in G. However, P_1 and P_2 share at most $\varepsilon|E|$ contact edges (otherwise, by the merging process, this implies that more than $\varepsilon|E|$ edges could be consistently labeled, which is a contradiction). Thus the solution has a total cost of at least $(2 - \varepsilon)|E|$.

It is thus NP-hard to distinguish between an instance with a solution of cost $|E|$, and an instance for which every solution has cost at least $(2 - \varepsilon)|E|$. Thus a polynomial-time algorithm for 2-DCSN with approximation ratio $2 - \epsilon$ can be used to decide Label Cover (with parameter ε) by running it on the output of the aforementioned reduction. If the estimated objective value is at most $(2-\varepsilon)|E|$ (and thus strictly less than $(2-\epsilon)|E|$) output YES; otherwise output NO. In other words, 2-DCSN is NP-hard to approximate to within a factor of $2 - \epsilon$.

To complete the proof, observe that the underlying directed graph we constructed is acyclic, as every edge points "to the right" as in Example 1. Hence 2-DCSN is NP-hard to approximate to within a factor of $2 - \epsilon$ for every $\epsilon > 0$, even on acyclic graphs. Finally, note that the same analysis holds for 2-CSN, by simply making every edge undirected; however in this case the graph is clearly not acyclic.

□

Inapproximability for general C and k

Theorem 1 (Main Theorem)**.** *CSN and DCSN are NP-hard to approximate to a factor of $C - \epsilon$ as well as $k - \epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Proof. We perform a reduction from k-Partite Hypergraph Label Cover, a generalization of Label Cover to hypergraphs, to CSN, or DCSN with an acyclic graph. Using the same ideas as in the $C = k = 2$ case, we design k demands composed of parallel paths corresponding to labelings, and merge edges so that a good global labeling corresponds to a large overlap between those paths. The full proof is left to Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling.

□

Note that a k-approximation algorithm is to simply choose $\mathcal{H} = \bigcup_{c_i} \tilde{P}_{c_i}$, where $\tilde{P}_{c_i}$ is the shortest $a_{c_i} \to b_{c_i}$ path in G_{c_i} for demands $\mathcal{D} = \{(a, b, c_i) : c_i \in [C]\}$. Thus by Theorem 1, essentially no better approximation is possible in terms of k alone. In contrast, most classic Steiner problems have good approximation algorithms [30, 78, 52, 32], or are even exactly solvable for constant k [51].

Inapproximability for Steiner variants

We take advantage of our previous hardness of approximation results in Theorem 1 and show, via a series of reductions, that CSP, CSN, and CPCST are also hard to approximate.

Theorem 2. *Condition Shortest Path, Directed Condition Shortest Path, Condition Steiner Tree, and Condition Prize-Collecting Steiner Tree are all NP-hard to approximate to a factor of $C - \epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$.*

Proof. We first reduce from CSN to CSP (and DCSN to DCSP). Suppose we are given an instance of CSN with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V, E)$, and demands $\mathcal{D} = \{(a_i, b_i, c_i) : i \in [k]\}$. We build a new instance $(\mathcal{G}' = (G'_1, \ldots, G'_k), G' = (V', E'), \mathcal{D}')$ as follows.

Initialize G' to G. Add to G' the new nodes a and b, which exist at all conditions G'_i. For all $e \in E$ and $i \in [k]$, if $e \in G_{c_i}$, then let e exist in G'_i as well. For each $(a_i, b_i, c_i) \in \mathcal{D}$,

1. Create new nodes x_i, y_i. Create zero-weight edges (a, x_i), (x_i, a_i), (b_i, y_i), and (y_i, b).
2. Let (a, x_i), (x_i, a_i), (b_i, y_i), and (y_i, b) exist only in frame G'_i.

Lastly, the demands are $\mathcal{D}' = \{(a, b, i) : i \in [k]\}$.

Given a solution $H' \subseteq G'$ containing an $a \to b$ path at every condition $i \in [k]$, we can simply exclude nodes a, b, $\{x_i\}$, and $\{y_i\}$ to obtain a solution $H \subseteq G$ to the original instance, which contains an $a_i \to b_i$ path in G_{c_i} for all $i \in [k]$, and has the same cost. The converse is

also true by including these nodes.

Observe that essentially the same procedure shows that DCSN reduces to DCSP; simply ensure that the edges added by the reduction are directed rather than undirected.

Next, we reduce CSP to CST. Suppose we are given an instance of CSP with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V, E)$, and demands $\mathcal{D} = \{(a, b, i) : i \in [C]\}$. We build a new instance of CST as follows:
$(\mathcal{G}' = (G'_1, \ldots, G'_C), G' = (V', E'), \mathcal{X} = (X_1, ..., X_C))$. Set $\mathcal{G}'$ to $\mathcal{G}$, and G' to G. Take the set of terminals in each condition to be $X_i = \{a, b\}$. We note that a solution $H' \subseteq G'$ to the CST instance is trivially a solution the CSP instance with the same cost, and vice-versa.

Finally, we reduce CST to CPCST. We do this by making an appropriate assignment of the penalties $p(v, c)$. Suppose we are given an instance of CST with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V, E)$, and terminal sets $\mathcal{X} = (X_1, ..., X_C)$. We build a new instance of CPCST, $(\mathcal{G}' = (G'_1, \ldots, G'_C), G' = (V', E'), p(v, c))$. In particular, set $\mathcal{G}'$ to $\mathcal{G}$, and G' to G. Set $p(v, c)$ as follows:

$$p(v, c) = \begin{cases} \infty, & v \in X_c \\ 0, & \text{otherwise} \end{cases}$$

Consider any solution $H \subseteq G$ to the original CST instance. Since H spans the terminals $X_1, \ldots, X_c$ (thus avoiding any infinite penalties), and since the non-terminal vertices have zero cost, the overall cost of H remains the same cost in the constructed CPCST instance. Conversely, suppose we are given a solution $H' \subseteq G'$ to the constructed CPCST instance. If the cost of H' is ∞, then H' does not span all the X_c's simultaneously, and thus H' is not a possible solution for the CST instance. On the other hand if H' has finite cost, then H' is also a solution for the CST instance, with the same cost.

To summarize: in the first reduction from CSN to CSP, the number of demands, k, in the CSN instance is the same as the number of the conditions, C, in the CSP instance; we conclude that CSP is NP-hard to approximate to a factor of $C - \epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$. Since C remains the same in the two subsequent reductions, we also have that CST and CPCST are NP-hard to approximate to a factor of $C - \epsilon$.

□

2.2.9 Monotonic special cases

In light of the strong lower bounds in the previous theorems, in this section we consider more tractable special cases of the condition Steiner problems. A natural restriction is that

the changes over conditions are *monotonic*:

Definition 8 (Monotonic {CSN, DCSN, CSP, DCSP, CST, CPCST})**.** *In this special case (of any of the condition Steiner problems), we have that for each $e \in E$ and $c \in [C]$, if $e \in G_c$, then $e \in G_{c'}$ for all $c' \geq c$.*

We now examine the effect of monotonicity on the complexity of the condition Steiner problems.

Monotonicity in the undirected case

In the undirected case, we show that monotonicity has a simple effect: it makes CSN equivalent to the following well-studied problem:

Definition 9 (Priority Steiner Tree [29])**.** *The input is a weighted undirected multigraph $G = (V, E, w)$, a* priority level *$p(e)$ for each $e \in E$, and a set of k demands (a_i, b_i), each with priority $p(a_i, b_i)$. The output is a minimum-weight forest $F \subseteq G$ that contains, between each a_i and b_i, a path in which every edge e has priority $p(e) \leq p(a_i, b_i)$.*

Priority Steiner Tree was introduced by Charikar, Naor, and Schieber [29], who gave a $O(\log k)$ approximation algorithm. Moreover, it cannot be approximated to within a factor of $\Omega(\log \log n)$ assuming NP $\notin \mathsf{DTIME}(n^{\log \log \log n})$ [37]. We now show that the same bounds apply to Monotonic CSN, by showing that the two problems are essentially equivalent from an approximation standpoint.

Lemma 1. *Fix any function $f : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$. If either Priority Steiner Tree or Monotonic CSN can be approximated to a factor of $f(k)$ in polynomial time, then so can the other.*

Proof. We transform an instance of Priority Steiner Tree into an instance of Monotonic CSN as follows: the set of priorities becomes the set of conditions; if an edge e has priority $p(e)$, it now exists at all conditions $t \geq p(e)$; if a demand (a_i, b_i) has priority $p(a_i, b_i)$, it now becomes $(a_i, b_i, p(a_i, b_i))$. If there are parallel multiedges, break up each such edge into two edges of half the original weight, joined by a new node. Given a solution $H \subseteq G$ to this CSN instance, contracting any edges that were originally multiedges gives a Priority Steiner Tree solution of the same cost. This reduction also works in the opposite direction (in this case there are no multiedges), which shows the equivalence.

□

Furthermore, the $O(\log k)$ upper bound applies to CST (We note that Monotonic CSP admits a trivial algorithm, namely take the subgraph induced by running Djikstra's Algorithm on G_1).

Lemma 2. *If Monotonic CSN can be approximated to a factor of $f(k)$ for some function f in polynomial time, then Monotonic CST can also be approximated to within $f(k)$ in polynomial time.*

Proof. We now show a reduction from CST to CSN. Suppose we are given a CST instance on graphs $\mathcal{G} = (G_1, \ldots, G_C)$ and terminal sets $\mathcal{X} = (X_1, ..., X_C)$. Our CSN instance has precisely the same graphs, and has the following demands: for each terminal set X_c, pick any terminal $a \in X_c$ and create a demand (a, b, c) for each $b \neq a \in X_c$. A solution to the original CST instance is a solution to the constructed CSN instance with the same cost, and vice-versa; moreover, if the CST instance is monotonic, then so is the constructed CSN instance. Observe that if the total number of CST terminals is k, then the number of constructed demands is $k - C$, and therefore an $f(k)$-approximation for CSN implies an $f(k - C) \leq f(k)$-approximation for CST, as required.

□

Monotonicity in the directed case

In the directed case, we give an approximation-preserving reduction from a single-source special case of DCSN to the Directed Steiner Tree (DST) problem (in fact, we show that they are essentially equivalent from an approximation standpoint), then apply a known algorithm for DST. Recall the definition of Single-Source DCSN:

Definition 6 (Single-Source DCSN)**.** *This is the special case of DCSN in which the demands are precisely $(a, b_1, c_1), (a, b_2, c_2), \ldots, (a, b_k, c_k)$, for some* root *$a \in V$. We can assume that $c_1 \leq c_2 \leq \cdots \leq c_k$.*

Lemma 3. *Fix any function $f : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$. If either Monotonic Single-Source DCSN or Directed Steiner Tree can be approximated to a factor of $f(k)$ in polynomial time, then so can the other.*

For the remainder of this section, we refer to Monotonic Single-Source DCSN as simply DCSN. Towards proving the theorem, we now describe a reduction from DCSN to DST. Given a DCSN instance $(G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C), \mathcal{D})$ with underlying graph $\mathcal{G} = (V, E)$, we construct a DST instance $(G' = (V', E'), D')$ as follows:

- G' contains a vertex v^i for each $v \in V$ and each $i \in [c_k]$. It contains an edge (u^i, v^i) with weight $w(u, v)$ for each $(u, v) \in E_i$. Additionally, it contains a zero-weight edge (v^i, v^{i+1}) for each $v \in V$ and each $i \in [c_k]$.
- D' contains a demand $(a^1, b_i^{c_i})$ for each $(a, b_i, c_i) \in \mathcal{D}$.

Now consider the DST instance (G', D').

Lemma 4. *If the DCSN instance $(G_1, \ldots, G_C, \mathcal{D})$ has a solution of cost C^*, then the constructed DST instance (G', D') has a solution of cost at most C^*.*

Proof. Let $\mathcal{H} \subseteq \mathcal{G}$ be a DCSN solution having cost C^*. For any edge $(u, v) \in E(\mathcal{H})$, define the *earliest necessary condition* of (u, v) to be the minimum c_i such that removing (u, v) would cause $\mathcal{H}$ not to satisfy demand (a, b_i, c_i).

Claim 1. *There exists a solution $\mathcal{C} \subseteq \mathcal{H}$ that is a directed tree and has cost at most C^*. Moreover for every path P_i in $\mathcal{C}$ from the root a to some target b_i, as we traverse P_i from a to b_i, the earliest necessary conditions of the edges are non-decreasing.*

Proof of Claim 1. Consider a partition of $\mathcal{H}$ into edge-disjoint sub-graphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$, where $\mathcal{H}_i$ is the sub-graph whose edges have earliest necessary condition c_i.

If there is a directed cycle or parallel paths in the first sub-graph $\mathcal{H}_1$, then there is an edge $e \in E(\mathcal{H}_1)$ whose removal does not cause $\mathcal{H}_1$ to satisfy fewer demands at condition c_1. Moreover by monotonicity, removing e also does not cause $\mathcal{H}$ to satisfy fewer demands at any future conditions. Hence there exists a directed tree $\mathcal{T}_1 \subseteq \mathcal{H}_1$ such that $\mathcal{T}_1 \cup \left(\bigcup_{i=2}^{k} \mathcal{H}_i\right)$ has cost at most C^* and still satisfies $\mathcal{T}$.

Now suppose by induction that for some $j \in [k-1]$, $\bigcup_{i=1}^{j} \mathcal{T}_i$ is a tree such that $\left(\bigcup_{i=1}^{j} \mathcal{T}_i\right) \cup \left(\bigcup_{i=j+1}^{k} \mathcal{H}_i\right)$ has cost at most C^* and satisfies $\mathcal{D}$. Consider the partial solution $\left(\bigcup_{i=1}^{j} \mathcal{T}_i\right) \cup \mathcal{H}_{j+1}$; if this sub-graph is not a directed tree, then there must be an edge $(u, v) \in E(\mathcal{H}_{j+1})$ such that v has another in-edge in the sub-graph. However by monotonicity, (u, v) does not help satisfy any new demands, as v is already reached by some other path from the root. Hence by removing all such redundant edges, we have $\mathcal{T}_{j+1} \subseteq \mathcal{H}_{j+1}$ such that $\left(\bigcup_{i=1}^{j+1} \mathcal{T}_i\right) \cup \left(\bigcup_{i=j+2}^{k} \mathcal{H}_i\right)$ has cost at most C^* and satisfies $\mathcal{D}$, which completes the inductive step.

We conclude that $\mathcal{T} := \bigcup_{i=1}^{k} \mathcal{T}_i \subseteq \mathcal{H}$ is a tree of cost at most C^* satisfying $\mathcal{D}$. Observe also that by construction, as $\mathcal{T}$ is a tree that is iteratively constructed by $\mathcal{T}_i \subseteq \mathcal{H}_i$, $\mathcal{T}$ has the property that if we traverse any $a \to b_i$ path, the earliest necessary conditions of the edges never decrease.

■

Now let $\mathcal{T}$ be the DCSN solution guaranteed to exist by Claim 1. Consider the subgraph $H' \subseteq G'$ formed by adding, for each $(u, v) \in E(\mathcal{T})$, the edge $(u^c, v^c) \in E'$ where c is the earliest necessary condition of (u, v) in $E(\mathcal{H})$. In addition, for all vertices $v^i \in H'$ where $v^{i+1} \in H'$, add the free edge (v^i, v^{i+1}). Since $w(u^c, v^c) = w(u, v)$ by construction, $\text{cost}(H') \leq \text{cost}(\mathcal{T}) \leq C^*$.

To see that H' is a valid solution, consider any demand $(a^1, b_i^{c_i})$. Recall that $\mathcal{T}$ has a unique $a \to b_i$ path P_i along which the earliest necessary conditions are nondecreasing. We added to H' each of these edges at the level corresponding to its earliest necessary condition; moreover, whenever there are adjacent edges $(u, v), (v, x) \in P_i$ with earliest necessary conditions c and $c' \geq c$ respectively, there exist in H' free edges $(v^t, v^{c+1}), \ldots, (v^{c'-1}, v^{c'})$. Thus H' contains an $a^1 \to b_i^{c_i}$ path, which completes the proof.

□

Lemma 5. *If the constructed DST instance (G', D') has a solution of cost C^*, then the original DCSN instance $(G_1, \ldots, G_C, \mathcal{D})$ has a solution of cost at most C^*.*

Proof. First note that any DST solution ought to be a tree; let $T' \subseteq G'$ be such a solution of cost C. For each $(u, v) \in G$, T' might as well use at most one edge of the form (u^i, v^i), since if it uses more, it can be improved by using only the one with minimum i, then taking the free edges (v^i, v^{i+1}) as needed. We create a DCSN solution $\mathcal{T} \subseteq \mathcal{G}$ as follows: for each $(u^i, v^i) \in E(T')$, add (u, v) to $\mathcal{T}$. Since $w(u, v) = w(u^i, v^i)$ by design, we have $\text{cost}(\mathcal{T}) \leq \text{cost}(T') \leq C$. Finally, since each $a^1 \to b_i^{t_i}$ path in G' has a corresponding path in $\mathcal{G}$ by construction, $\mathcal{T}$ satisfies all the demands.

□

Lemma 3 follows from Lemma 4 and Lemma 5. Finally we can obtain the main result of this subsection:

Theorem 4. *Monotonic Single-Source DCSN has a polynomial-time $O(k^\epsilon)$-approximation algorithm for every $\epsilon > 0$. It has no $\Omega(\log^{2-\epsilon} n)$-approximation algorithm unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathsf{polylog}(n)})$.*

Proof. The upper bound follows by composing the reduction (from Monotonic Single-Source DCSN to Directed Steiner Tree) with the algorithm of Charikar et al. [30] for Directed

Steiner Tree, which achieves ratio $O(k^\epsilon)$ for every $\epsilon > 0$. More precisely they give an $i^2(i-1)k^{1/i}$-approximation for any integer $i \geq 1$, in time $O(n^i k^{2i})$. The lower bound follows by composing the reduction (in the opposite direction) with a hardness result of Halperin and Krauthgamer [78], who show the same bound for Directed Steiner Tree. A quick note regarding the reduction in the opposite direction: Directed Steiner Tree is a precisely a Monotonic Single-Source DCSN instance with exactly one condition.

□

In Extracting Networks - Subnetwork Optimization and Applications to IL23 Signalling, we show how to modify the algorithm of Charikar et al. to arrive at a simple, explicit algorithm for Monotonic Single-Source DCSN achieving the same guarantee.

2.2.10 Application to protein-protein interaction networks

Methods such as Directed Condition Steiner Network can be key in identifying underlying structure in biological processes. As a result, it is important to assess the runtime feasibility of solving for an optimal solution. We show via simulation on human protein-protein interaction networks, that our algorithm on single-source instances is able to quickly and accurately infer maximum likelihood subgraphs for a certain biological process.

Building the protein-protein interaction network

We represent the human PPI network as a weighted directed graph, where proteins serve as nodes, and interactions serve as edges. The network was formed by aggregating information from four sources of interaction data, including Netpath [104], Phosphosite [86], HPRD [107], and InWeb [119], altogether, covering 16222 nodes and 437888 edges.

Edge directions are assigned where these annotations were available (primarily in Phopshosite and NetPath). The remaining edges are represented by two directed edges between the proteins involved. Edge weights were assigned by taking the negative logarithm of the associated confidence score, indicating that finding the optimal Steiner Network would be the same as finding the most confident solution (assuming independence between edges). Confidence data was available for the largest of the data sets (InWeb). For HPRD edges that are not in InWeb, we used the minimum nonzero confidence value by default. For the smaller and highly curated data-sets, Phopshosite and NetPath, we used the maximal confidence level.

Solving DCSN to optimality

Definition 6 (Single-Source DCSN)**.** *This is the special case of DCSN in which the demands are precisely* $(a, b_1, c_1), (a, b_2, c_2), \ldots, (a, b_k, c_k)$*, for some* root $a \in V$*. We can assume that* $c_1 \leq c_2 \leq \cdots \leq c_k$*.*

We can derive a natural integer linear program for the Single-Source Directed Condition Steiner Network in terms of network flows, with each demand being met by a flow from source to target:

$$
\begin{aligned}
&\text{minimize} \quad \sum_{(u,v)\in E} d_{uv} \cdot w(u,v) \\
&\text{subject to} \\
&\qquad k_c \cdot d_{uv} \geq d_{uvc} && \forall c,\ (u,v) \in E_c \\
&\sum_{(v,w)\in E_c} d_{vwc} = \sum_{(u,v)\in E_c} d_{uvc} + \delta_{vc} && \forall c,\ v \in V \\
&\qquad d_{uvc} \in \{0, 1, \ldots, k_c\} && \forall c,\ (u,v) \in E_c \\
&\qquad d_{uv} \in \{0, 1\} && \forall (u,v) \in E
\end{aligned}
$$

Figure 2.3: Integer linear program for Single-Source Conditon Steiner Network. $\delta_{vc} = 1$ for v at condition c if v is a target at condition c, $-k_c$ for v at condition c if v is the source node at condition c, 0 otherwise.

Each variable d_{uvc} denotes the flow through edge (u, v) at condition c, if it exists; each variable d_{uv} denotes whether (u, v) is ultimately in the chosen solution sub-graph; k_c denotes the number of demands at condition c . The first constraint ensures that if an edge is used at any condition, it is chosen as part of the solution. The second constraint enforces flow conservation, and hence that the demands are satisfied, at all nodes and all conditions.

We note that DCSN easily reduces DCSP, as outlined in Theorem 2. However, DCSP is a special case of Single-Source DCSN. Therefore, the integer linear program defined above can be applied to any DCSN instance with a transformation of the instance to DCSP.

Performance analysis of integer linear programming

Given the protein-protein interaction network G, we sample an instance of the node-variant Single-Source DCSN as so[3]:

[3]As previously mentioned, this variant reduces to the edge variant via reduction, and vice versa

- Instantiate a source node a.
- Independently sample β nodes reachable from a, for each of the C conditions, giving us $\{b_{1,1}, ..., b_{\beta,C}\}$.
- For each node $v \in V$, include $v \in V_c$ if v lies on the shortest path from a to one of $\{b_{1,c}, .., b_{\beta,c}\}$
- For all other nodes $v \in V$ for all c, include $v \in V_c$ with probability p.

Using a workstation running an Intel Xeon E5-2690 processor and 250GB of RAM, **optimal solutions** to instances of modest size (generated using the procedure just described) were within reach:

β	C	p	Time to solve
100	1	1.0	45s ± 5s
10	1	.25	1m ± 10s
100	1	.25	1m ± 10s
10	1	.75	1m ± 10s
100	1	.75	1m ± 10s
100	10	1.0	7m ± 30s
10	10	.25	9m ± 30s
10	10	.75	11m ± 30s
100	10	.25	12m ± 30s
100	10	.75	17m ± 2m
100	100	1.0	1h 40m ± 15m
10	100	.75	2h 30m ± 12m
100	100	.75	4h ± 40m

Table 2.2: ILP solve times for some random instances of SS-DCSN generated by our random model using the Gurobi Python Solver package[73].

We notice that our primary runtime constraint comes from C, the number of conditions. In practice, the number of conditions does not exceed 100.

In addition, we decided to test our DCSN ILP formulation against a simple algorithm of optimizing over each demand independently via shortest path. Theoretically, the shortest path method can perform up to k times worse than DCSN. We note that having zero weight edges complicates the comparison of algorithms' performance on real data. The reason is that we can have the same weight for a large and small networks. Instead, we wanted to also

take into account the size of the returned networks. To do that we added a constant weight for every edge. Testing over a sample set of instances generated with parameters $\beta = 100$, $C = 10$, $p = 0.25$, we found that the shortest path method returns a solution on average 1.07 times more costly.

Therefore, we present a model showing preliminary promises of translating and finding **optimal solutions** to real world biological problems with practical runtime.

2.2.11 Conclusion and discussion

In this paper we introduced the Condition Steiner Network (CSN) problem and its directed variant, in which the goal is to find a minimal subgraph satisfying a set of k condition-sensitive connectivity demands. We show, in contrast to known results for traditional Steiner problems, that this problem is NP-hard to approximate to a factor of $C-\epsilon$, as well as $k-\epsilon$, for every $C, k \geq 2$ and $\epsilon > 0$. We then explored a special case, in which the conditions/graphs satisfy a *monotonicity* property. For such instances we proposed algorithms significantly beating the pessimistic lower bound for the general problem; this was accomplished by reducing the problem to certain traditional Steiner problems. Lastly, we developed and applied an integer programming-based exact algorithm on simulated instances built over the human protein-protein interaction network, and reported feasible runtimes for real-world problem instances.

Importantly, along the way we showed implications of these results for CSN on other network connectivity problems that are commonly used in PPI analysis—such as Shortest Path, Steiner Tree, Prize-Collecting Steiner Tree—when conditions are added. We showed that for each of these problems, we cannot guarantee (in polynomial time) a solution with a value below $C - \epsilon$ times the optimal value. These lower bounds are quite strict, in the sense that naively approximating the problem separately in every condition, and taking the union of those solutions, already gives an approximation ratio of $O(C)$. At the same time, by relating the various condition Steiner problems to one another, we also obtained some positive results: the condition versions of Shortest Path and Steiner Tree admit good approximations when the conditions are monotonic. Moreover, all of the condition problems (with the exception of Prize-Collecting Steiner Tree) can be solved using a natural integer programming framework that works well in practice.

2.2.12 Proofs of main theorems

Problem variants

There are several natural ways to formulate the condition Steiner network problem, depending on whether the edges are changing over condition, or the nodes, or both.

Definition 10 (Condition Steiner Network (edge variant))**.** *This is the formulation described in the introduction: the inputs are* $G_1 = (V, E_1), \ldots, G_C = (V, E_C)$*,* $w(\cdot)$*, and* $\mathcal{D} = \{(a_i, b_i, c_i)\}$*. The task is to find a minimum-weight sub-graph* $\mathcal{H} \subseteq \mathcal{G}$ *that satisfies all of the demands.*

Definition 11 (Condition Steiner Network (node variant))**.** *Let the underlying graph be* $\mathcal{G} = (V, E)$*. The inputs are* $G_1 = (V_1, E(V_1)), \ldots, G_C = (V_C, E(V_C))$*,* $w(\cdot)$*, and* $\mathcal{D}$*. Here,* $E(V_c) \subseteq E$ *denotes the edges induced by* $V_c \subseteq V$*. A path satisfies a demand at condition* t *if all edges along that path exist in* G_c*.*

Definition 12 (Condition Steiner Network (node and edge variant))**.** *The inputs are precisely* $G_1 = (V_1, E_1), \ldots, G_C = (V_C, E_C)$*,* $w(\cdot)$*, and* $\mathcal{D}$*. This is the same as the node variant except that each* E_c *can be any subset of* $E(V_c)$

Similarly, define the corresponding directed problem Directed Condition Steiner Network (DCSN) with the same three variants. The only difference is that the edges are directed, and a demand (a, b, c) must be satisfied by a directed $a \to b$ path in G_c.

The following observation enables all our results to apply to all problem variants.

Proposition 2. *The edge, node, and node-and-edge variants of CSN are mutually polynomial-time reducible via strict reductions (i.e. preserving the approximation ratio exactly). Similarly all three variants of DCSN are mutually strictly reducible.*

Proof. The following statements shall hold for both undirected and directed versions. Clearly the node-and-edge variant generalizes the other two. It suffices to show two more directions:

(Node-and-edge reduces to node) Let (u, v) be an edge existent at a set of conditions $\tau(u, v)$, whose endpoints exist at conditions $\tau(u)$ and $\tau(v)$. To make this a node-condition instance, create an intermediate node $x_{(u,v)}$ existent at conditions $\tau(u, v)$, an edge $(u, x_{(u,v)})$ with the original weight $w(u, v)$, and an edge $(x_{(u,v)}, v)$ with zero weight. A solution of cost W in the node-and-edge instance corresponds to a node-condition solution of cost W, and

vice-versa.

(Node reduces to edge) Let (u, v) be an edge whose endpoints exist at conditions $\tau(u)$ and $\tau(v)$. To make this an edge-condition instance, let (u, v) exist at conditions $\tau(u, v) := \tau(u) \cap \tau(v)$. Let every node exist at all conditions; let the edges retain their original weights. A solution of cost W in the node-condition instance corresponds to an edge-condition solution of cost W, and vice-versa.

□

Proof of inapproximability for general C and k

Here we prove our main theorem, showing optimal hardness for any number of demands. To do this, we introduce a generalization of Label Cover to partite hypergraphs:

Definition 13 (k-Partite Hypergraph Label Cover (k-PHLC))**.** *An instance of this problem consists of a k-partite, k-regular hypergraph $G = (V_1, \ldots, V_k, E)$ (that is, each edge contains exactly one vertex from each of the k parts) and a set of possible labels Σ. The input also includes, for each hyperedge $e \in E$, a* projection function *$\pi_v^e : \Sigma \to C$ for each $v \in e$; Π is the set of all such functions. A labeling of G is a function $\phi : \bigcup_{i=1}^k V_i \to \Sigma$ assigning each node a label. There are two notions of edge satisfaction under a labeling ϕ:*

- ϕ strongly satisfies *a hyperedge $e = (v_1, \ldots, v_k)$ if the labels of all its vertices are mapped to the same color, i.e. $\pi_{v_i}^e(\phi(v_i)) = \pi_{v_j}^e(\phi(v_j))$ for all $i, j \in [k]$.*
- ϕ weakly satisfies *a hyperedge $e = (v_1, \ldots, v_k)$ if there exists some pair of vertices v_i, v_j whose labels are mapped to the same color, i.e. $\pi_{v_i}^e(\phi(v_i)) = \pi_{v_j}^e(\phi(v_j))$ for some $i \neq j \in [k]$.*

The following gap hardness for this problem was shown by Feige [50]:

Theorem 7. *For every $\epsilon > 0$ and every fixed integer $k \geq 2$, there is a constant $|\Sigma|$ such that the following promise problem is* NP*-hard: Given a k-Partite Hypergraph Label Cover instance (G, Σ, Π), distinguish between the following cases:*

- *(YES instance) There exists a labeling of G that strongly satisfies every edge.*
- *(NO instance) Every labeling of G weakly satisfies at most $\epsilon|E|$ edges.*

The proof of $(C - \epsilon)$-hardness and $(k - \epsilon)$-hardness follows the same outline as the $C = k = 2$ case (Theorem 6).

Theorem 8 (Main Theorem)**.** *CSN and DCSN are NP-hard to approximate to a factor of $C - \epsilon$ as well as $k - \epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Proof. Given the k-PHLC instance in the form $(G = (V_1, \ldots, V_k, E), \Sigma, \Pi)$, and letting $v_{c,i}$ denote the i-th node in V_c, construct a DCSN instance ($\mathcal{G} = (G_1, \ldots, G_k)$, along with k demands) as follows. For every $c \in [k]$, create nodes $v^S_{c,1}, \ldots, v^S_{t,|V_c|+1}$. Create a $v_{c,i}$-*bundle* from each $v^S_{c,i}$ to $v^S_{c,i+1}$, whose ℓ-strands (for $\ell \in \Sigma$) are each a chain of bundles, one for each incident hyperedge $e = (v_{1,i_1}, \ldots, v_{c,i}, \ldots, v_{k,i_k}) \in E$. Each $(v_{1,i_1}, \ldots, v_{c,i}, \ldots, v_{k,i_k})$-bundle has a $(v_{1,i_1}, \ell_1, \ldots, v_{c,i}, \ell_c, \ldots, v_{k,i_k}, \ell_k)$-path for each agreeing combination of labels—that is, every k-tuple $(\ell_1, \ldots, \ell_c, \ldots, \ell_k)$ such that: $\pi^e_{v_{1,i_1}}(\ell_1) = \cdots = \pi^e_{v_{c,i}}(\ell_c) = \cdots = \pi^e_{v_{k,i_k}}(\ell_k)$, where e is the shared edge. If there are no such combinations, then the e-bundle is a single simple strand.

For $c \in [k]$, set all the edges in the $v_{c,i}$-bundles to exist in G_c only. Now, for each $(v_{1,i_1}, \ell_1, \ldots, v_{k,i_k}, \ell_k)$, merge together the $(v_{1,i_1}, \ell_1, \ldots, v_{k,i_k}, \ell_k)$-paths across all G_c that have such a strand. Finally, the connectivity demands are $\mathcal{D} = \left\{ \left(v^S_{c,1}, v^S_{c,|V_c|+1}, c \right) : c \in [k] \right\}$.

The analysis follows the $k = 2$ case. Suppose we have a YES instance of k-PHLC, with optimal labeling ℓ^*_v to each node $v \in \bigcup_{t=1}^k V_c$. Then an optimal solution $\mathcal{H}^*$ to the constructed DCSN instance is to traverse, at each condition c and for each $v_{c,i}$-bundle, the path through the $\ell^*_{v_{c,i}}$-strand. In particular for each $(v_{1,i_1}, \ldots, v_{k,i_k})$-bundle in that strand, traverse the $(v_{1,i_1}, \ell^*_1, \ldots, v_{k,i_k}, \ell^*_k)$-path.

In tallying the total edge cost, $\mathcal{H}^* \cap G_1$ (the sub-graph at condition 1) incurs a cost of $|E|$, one for each contact edge. The sub-graphs of $\mathcal{H}^*$ at conditions $2, \ldots, k$ account for no additional cost, since all contact edges correspond to a label which agrees with all its neighbors' labels, and hence were merged with the agreeing contact edges in the other sub-graphs.

Conversely suppose we have a NO instance of k-PHLC, so that for any labeling ℓ^*_v, for at least $(1 - \epsilon)|E|$ hyperedges e, the projection functions of all nodes in e disagree. By definition, any solution to the constructed DCSN instance contains a simple $v^S_{t,1} \to v^S_{t,|V_c|+1}$ path P_c at each condition c. As before, P_1 alone incurs a cost of exactly $|E|$. However, at least $(1 - \epsilon)|E|$ of the hyperedges in G cannot be weakly satisfied; for these hyperedges e, for every pair of neighbors $v_{c,i_c}, v_{c',i_{c'}} \in e$, there is no path through the e-bundle in v_{t,i_c}'s $\ell^*_{v_{c,i_c}}$-strand that is merged with any of the paths through the e-bundle in $v_{c',i_{c'}}$'s $\ell^*_{v_{c,i_{c'}}}$-strand (for otherwise, it would indicate a labeling that weakly satisfies e in the k-PHLC instance). Therefore paths $P_2, \ldots, P_k$ each contribute at least $(1 - \epsilon)|E|$ additional cost, so the solution has total cost at least $(1 - \epsilon)|E| \cdot k$.

It follows from the gap between the YES and NO cases that DCSN is NP-hard to approximate to within a factor of $k - \epsilon$ for every constant $\epsilon > 0$; and since $C = k$ in our construction, it is also NP-hard for $C - \epsilon$. Moreover since The directed condition graph we constructed is acyclic, this result holds even on DAGs. As before, the same analysis holds for the undirected problem CSN by undirecting the edges.

□

Explicit algorithm for Monotonic Single-Source DCSN

We provide a modified version of the approximation algorithm presented in Charikar et al. [30] for Directed Steiner Tree (DST), which achieves the same approximation ratio for our problem Monotonic Single-Source DCSN.

We provide a similar explanation as of that presented in Charikar et al. Consider a trivial approximation algorithm, where we take the shortest path from the source to each individual target. Consider the example where there are edges of cost $C-\epsilon$ to each target, and a vertex v with distance C from the source, and with distance 0 to each target. In such a case, this trivial approximation algorithm will achieve only an $\Omega(k)$-approximation. Consider instead an algorithm which found, from the root, an intermediary vertex v, which was connected to all the targets via shortest path. In the case of the above example, this would find us the optimal sub-graph. The algorithm below generalizes this process, by progressively finding optimal substructures with good cost relative to the number of targets connected. We show that this algorithm provides a good approximation ratio.

Definition 14 (Metric closure of a condition graph)**.** *For a directed condition graph* $\mathcal{G} = (G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C))$*, define its* metric closure *to be* $\tilde{G} = (V, E, \tilde{w})$ *where* $E = \bigcup_c E_c$ *and* $\tilde{w}(u, v, c)$ *is the length of the shortest* $u \to v$ *path in* G_c *(note that in contrast with* w*,* $\tilde{w}$ *takes three arguments).*

Definition 15 ($V(T)$)**.** *Let* T *be a tree with root* r*. We say a demand of the form* (r, b, c) *is* satisfied *by* T *if there is a path in* T *from* r *to* b *at condition* c*.* $V(T)$ *is then the set of demands satisfied by* T*.*

Definition 16 ($D(T)$)**.** *The* density *of a tree* T *is* $D(T) = \frac{cost(T)}{|V(T)|}$*, where* $cost(T)$ *is the sum of edge weights of* T*.*

function A_i(transitive closure $G = (V, E, w)$, r, c, k, $\mathcal{D} \subseteq V \times [C]$)
  if (r, b_i, c_i) does not exist for least $k (b_i, c_i) \in \mathcal{D}, c_i \geq c$ **then return** NO SOLUTION
  $T \leftarrow \emptyset$
  while $k > 0$ **do**
    $T_{best} \leftarrow \emptyset$
    for all $(v, t') \in V \times [C], c' \geq c$ and $k', 1 \geq k' \geq k$ **do**
      $T' \leftarrow \mathsf{A}_{i-1}(G, v, c', k', \mathcal{D}) \cup (r, v, c')$
      if $d(T_{BEST}) > d(T')$ **then** $T_{BEST} \leftarrow T'$ ▷ Demand i satisfied only if edge to b_i at c_i (ie (x, b_i, c_i) for some x)
    $T \leftarrow T \cup T_{BEST}$; $k \leftarrow k - |D \cap V(T_{BEST})|$; $X \leftarrow X - V(T_{BEST})$
  return T

The way we will prove the approximation ratio of this algorithm is to show that it behaves precisely as the algorithm of Charikar et al. does, when given as input the DST instance produced by our reduction from Monotonic Single Source DCSN (Lemma 3).

Proposition 3. *The algorithm above is equivalent to the algorithm of Charikar et al., when applied to the DST instance output by the reduction of Lemma 3.*

Proof. To see this, note that in our reduced instance, we see a collection of vertices, $v^1, ..., v^{|C|}$. Therefore, the only equivalent modifications needed to the original algorithm are:

- In the input, rather than keeping track of the current root as some vertex v^i, keep track of v at the current condition instead, i.e. (v, i).
- The distance from some v^i to $x^j, j \geq i$ is simply the distance from v to x at condition j, i.e. $\tilde{w}(v, x, j)$.
- Instead of looping through all vertices in the form $v^1, \ldots, v^{|C|}$, we instead loop through all vertices, and all conditions.

Therefore this algorithm guarantees the same approximation ratio for Monotonic Single Source DCSN as the original algorithm achieved for DST. In particular for all $i > 1$, $A_i(G, a, 0, k, D)$ provides an $i^2(i-1)k^{1/i}$ approximation to DCSN, in time $O(n^i k^{2i})$ [30, 79][4].

□

[4]The first paper [30] incorrectly claims a bound of $i(i-1)k^{1/i}$; this was corrected in [79].

2.3 Directed Shortest Walk on Temporal Graphs

2.3.1 Authors and Contributions

Alex Khodaverdian, Nir Yosef. Directed Shortest Walk on Temporal Graphs. Submitted.

All authors conceived and designed the study. AK derived the main hardness results, the ILP formulation, and the benchmarking. NY was the PI and oversaw the project.

2.3.2 Abstract

Background: The use of graphs as a way of abstracting and representing biological systems has provided a powerful analysis paradigm. Specifically, graph optimization algorithms are routinely used to address various connectivity queries, such as finding paths between proteins in a protein-protein interaction network, while maximizing objectives such as parsimony. While present studies in this field mostly concern static graphs, new types of data now motivate the need to account for changes that might occur to the elements (nodes) that are represented by the graph on the relationships (edges) between them.

Results and Discussion: We define the notion of Directed Temporal Graphs as a series of directed subgraphs of an underlying graph, ordered by time, where only a subset of vertices and edges are present. We then build up towards the Time Conditioned Shortest Walk problem on Directed Temporal Graphs: given a series of time ordered directed graphs, find the shortest walk from any given source node at time point 1 to a target node at time $T \geq 1$, such that the walk is consistent (monotonically increasing) with the timing of nodes and edges. We show, contrary to the Directed Shortest Walk problem which can be solved in polynomial time, that the Time Conditioned Shortest Walk (TCSW) problem is NP-Hard, and is hard to approximate to factor $\lceil \frac{T}{2} \rceil - \epsilon$ for $T \geq 3$ and $\epsilon > 0$. Lastly, we develop an integer linear program to solve a generalized version of TCSW, and demonstrate its ability to reach optimality with instances of the human protein interaction network.

Conclusion: We demonstrate that when extending the shortest walk problem in computational biology to account for multiple ordered conditions, the problem not only becomes hard to solve, but hard to approximate, a limitation which we address via a new solver. From this narrow definition of TCSW, we relax the constraint of time consistency within the shortest walk, deriving a direct relationship between hardness of approximation and the allowable step size in our walk between time conditioned networks. Lastly we briefly explore a variety of alternative formulations for this problem, providing insight into both tractable and intractable variants.

Availability Our solver for the general k-Time Condition Shortest Walk problem is available at `https://github.com/YosefLab/temporal_condition_shortest_walk`

2.3.3 Background

A common approach for investigating biological systems is to first abstract the system as a network. These networks may be defined over broad and varying categories such as species or at a higher resolution over proteins or metabolites within cells. For example, proteins and their pairwise interactions can be modeled as protein protein interaction networks, where nodes represent proteins and edges represent physical binding or joint chemical reactions. Once such an abstraction has been made, an investigator can dig into more specific questions. For example, cells express a subset of protein receptors depending on the cell type and cell state. Once a receptor is stimulated via extracellular ligands, a signalling cascade begins at receptor on the cell surface and often ends by modifying several transcription factors in the nucleus. Although a broad range of known pairwise protein interactions are well studied in the protein-protein interaction network, the path in which the signal propagates through this network from a receptor or protein to a terminal protein or transcription factor is less well studied and is a newer question of interest [186, 88, 163, 148, 61]. Under the assumptions of a static protein-protein interaction network, one could simply apply out of the box algorithms such as Djikstra's algorithm or Steiner Tree to find the most likely path taken [160, 89]. In practice however it turns out that the network is dynamic with time, as only a subset of proteins are active signalling candidates at any given time, be it that the protein may not be present in the cell, or may be deactivated via post-translational modifications [144, 131, 158]. Therefore, the question of interest becomes to not only infer the sequence of protein protein interactions in which receptor signaling leads to changes in transcription factors, but how this signal propagates through a dynamic temporal network.

Temporal networks have been well studied, with applications to broad fields such as communication propagation [84, 85, 124]. The simplest of such temporal problems is the shortest path problem, with the following setup - a network is provided with weighted edges, which travel from timepoint t_1 to t_2, where $t_2 > t_1$. The goal becomes to find a path which optimizes over a constraint, and is time-monotonic (or increasing in time). Many such possible variants of time-monotonic shortest path exist over this network, for example the earliest arrival time from a to b, the latest departure time from a to b, or just simply the lowest cost path (by edge weight), all of which can be solved in polynomial time [187, 182, 188]. There also exist more difficult problems, such as finding strongly connected components [139], or finding time-monotonic spanning trees [90, 95, 152], both of which are intractable, although for the case of minimum spanning trees, there exists an approximation preserving reduction to Directed Steiner Tree, which can be non-trivially approximated [30, 20]. While these works underscore the importance of a singular optimization problem spanning multiple timepoints, they fail to consider several key points present in real biological applications. Firstly, in the aforementioned literature, nodes are always present and traversable. However, in practice, nodes, or proteins, can be in active forms at certain times, and inactive forms at other times. Secondly, a key notion in biology comes from the idea of Occam's Razor. That is we wish to explain a biological process with the least number of edges or nodes traversed, accounting

for the fact that the same edge or node can be used multiple times under different contexts (ex. time). This model fails to allow for that possibility due to only allowing for a path like singular edge traversal, rather than a walk.

Another work closely related to this problem comes from Wu et al, who explored the notion of Condition networks [189]. In this setting, a series of C directed networks G_c are provided, with the same set of nodes V, but a different set of edges E_c. In addition, each network comes with its own set of pairwise connectivity demands $X_c = \{(a_1, b_1), ..., (a_n, b_n)\}$. The goal becomes to identify a subgraph H s.t. all connectivity demands in X_c are satisfied in $H \cap G_c$, and H is of minimum cost. A special version of this problem is Condition Shortest Path (CSP), in which a singular connectivity demand $X = (a, b)$ is given for each condition. Both of these problems turn out to be NP-hard to approximate beyond a trivial factor $|C|$. Although this problem attempts to optimize for the shortest path between a given (a, b) pair per condition, with information reuse, we are instead interested in a very different scenario. In particular, the scenario we are interested in considers one singular network demand spanning over many graphs, motivated by dynamic biological networks in practice.

2.3.4 Summary of main contributions

To this end, we introduce the Time Conditioned Shortest Walk (TCSW) problem, which takes on a similar flavor as Condition Shortest Path (CSP) introduced by Wu et al. In this setting we are given a series of ordered networks G_t and ordered conditions $\{1, ..., T\}$ representing a discrete measurement of time, and as well as a pair of nodes $(a \in G_1, b \in G_T)$. For each time condition, our defined networks G_t have vertices V_t which are allowed to change across networks, but a set of global edges E which remain constant. The goal is to find a walk from a source node a to target node b which begins in G_1, ends in G_T, and satisfies the following constraints: transitions are allowed from $v \in G_i$ to $w \in G_i$ or G_{i+1} if $(v, w) \in E$. In addition, transitions are allowed from $v \in G_i$ to $v \in G_{i+1}$. Edge costs are only paid once globally; that is, if you reuse an edge, you do not pay a cost.

We first prove a simple algorithm for solving the case for $T = 2$ time conditions. We then move to the more difficult case where vertices are shifting, and show that it is NP-hard to find a solution that achieves an approximation factor better than $\lceil \frac{T}{2} \rceil$ for $T \geq 3$ via approximation preserving reduction to CSP. We then extend our results to the general setting, where the time differential between two nodes in a path can be an integer k greater than 1. We prove that this problem is similarly hard to approximate to a factor better than $\lceil \frac{T}{k+1} \rceil$, for an arbitrary step size k. Lastly, we provide a integer linear program for the general TCSW problem, and show that when provided with real-world input it is capable of finding optimal solutions in reasonable time.

2.3.5 Definitions and Preliminaries

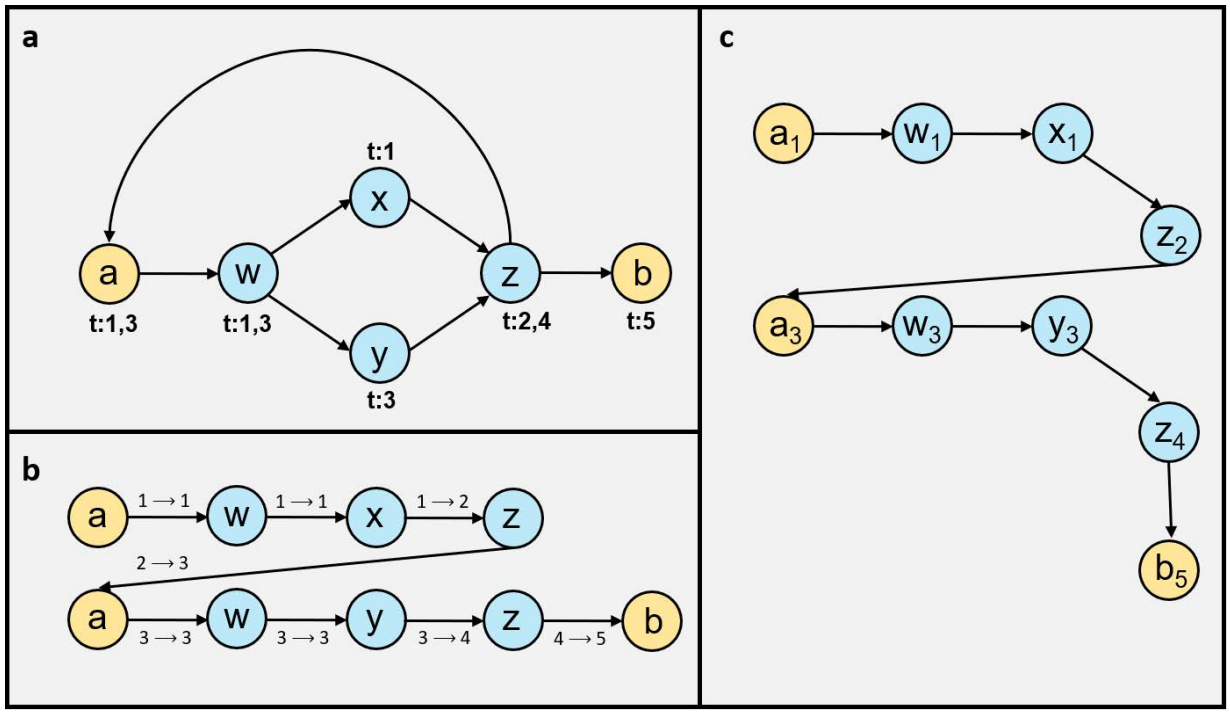


Figure 2.4: **(a)** Example TCSW instance, with active time points per node labeled and source a and target b highlighted in yellow. In particular, we note that z is inactive at timepoint 3, and therefore the only possible solution is to transition from z in timepoint 2 to a in timepoint 3, ultimately going through w y and z again to get to b at timepoint 5. **(b)** Solution to the example instance. **(c)** Solution to the example instance based on the alternate formulation of $\mathcal{G}$ presented in Definition 1

In graph theory, the shortest path problem is well studied in its many variants: from undirected networks, to weighted directed networks with nonnegative edge weights, to weighted directed networks without negative cycles. Each of these variants can be solved efficiently. For example the most common variant of the shortest path problem over weighted directed networks is solvable via Djikstra's algorithm in $O(E + V\log(V))$ complexity, or with more advanced techniques such as Thorup's algorithm in time $O(E + V\log(\log(V)))$.

In this paper we extend the generalization of network problems begun under Wu et al to the condition setting. Specifically, we generalize the shortest path problem to the time conditioned setting. Recall that in this setting we have a series of ordered time conditions $[T] = \{1, ..., T\}$, each with a corresponding network G_t.

Definition 17 (Time Condition Shortest Walk (TCSW))**.** *Given the following inputs:*

1. *A series of directed networks* $\{G_t = (V_t, E)\}_{t\in[T]}$ *each corresponding to a time condition* t*. Edges are positively weighted. Note that we denote* $V = \bigcup_t V_t$*. We denote a node* $v \in V_T$ *as* v_t

2. *A pair of nodes* (a_1, b_T) *s.t.* $a_1 \in V_1$ *and* $b_T \in V_T$ *which we wish to connect via a walk through* $G_1, ..., G_T$

Our goal is to find an $a - b$ walk from G_1 to G_T. We denote a walk W in the form $W = \{\ldots, (v_i, w_j), \ldots\}$, where (v_i, w_j) is a valid edge if $v \in V_i, w \in V_j$ and $i = j$ or $i = j + 1$, and $(v, w) \in E$ or $v = w$. We denote this "jump" constraint from G_i to G_{i+1} as the *temporal walk constraint*. The walk begins at a_1 and ends at b_T.

Equivalently, one can consider TCSW over a singular global network defined over all G_i. We define such network $\mathcal{G}$ with $v_i \in \mathcal{G}$ if $v \in V_i$, where v may be present in multiple V_t. An edge $e = (v_i, w_j)$ exists in $\mathcal{G}$ if $v_i \in V_i, w_j \in V_j$, $i = j$ or $i = j + 1$, and $(v_i, w_j) \in E$. In addition, we add zero weight edges for all nodes v_i, v_{i+1} if $v_i \in V_i$ and $v_{i+1} \in V_{i+1}$. The goal is to find the minimum cost path from a_1 to b_T where an edge (v, w) is only paid for once regardless of however many (v_i, w_j) pairs are traversed.

We offer the most formal presentation of TCSW, its variants, and $\mathcal{G}$ in the Appendix.

Definition 18 ($k-$Time Condition Shortest Walk ($k-$TCSW))**.** *In this variant, we relax the temporal walk constraint to allow for jumps between networks that aren't immediately subsequent in time. More specifically: For a given walk* W *in the form* $W = \{\ldots, (v_i, w_j), \ldots\}$*,* (v_i, w_j) *is a valid edge if* $v \in V_i, w \in V_j$ *and* $i \leq j \leq i + k$*. We denote this relaxed constraint as the k-temporal walk constraint. We note that in the definition for vanilla TCSW we operate under the 1-temporal walk constraint.*

Definition 19 (Directed Condition Shortest Path (CSP))**.** *We draw this definition from Wu et al.*

1. *A sequence of directed graphs* $G_1 = (V_1, E), G_2 = (V_2, E), ..., G_T = (V_T, E)$ *with positively weighted edges and* $V = \bigcup_t V_t$*.*

2. *A set of* C *connectivity demands* $\mathcal{D} \subseteq V \times V \times [C]$ *in the form* $(a, b, 1), ..., (a, b, C)$*.*

Let $G = (V, E)$ be the underlying network. The goal of this problem is to find a subgraph $H \subseteq G$ of minimal cost s.t. there exists a path from (a, b) in H amongst vertices that are active in G_c for all c. We note that CSP is hard to approximate to a factor of $C - \epsilon$ for every $C \geq 2$ and $\epsilon > 0$, a fact that we will exploit to prove $TCSW$ is hard to approximate.

Problem Variants

The description given above for $TCSW$ is just one possible way to describe the Temporal Walk problem. Here we describe several other variants, leaving their analysis for later in the manuscript.

Definition 20 (Strict Step Repay − Time Condition Shortest Walk (SSR−TCSW))**.** *Given the same inputs as $TCSW$, a demand pair and a series of networks $G_1...G_T$, the goal is to find an $a-b$ walk W from G_1 to G_T, where W in the form $W = \{\ldots,(v_i,w_j),\ldots\}$, where (v_i,w_j) is a valid edge if $v \in V_i, w \in V_j$ and $i = j+1$, and $(v,w) \in E$ or $v = w$. That is, every step must move up in time. In addition, edges are paid for per use, i.e. edges can be reused but must be paid for each time.*

Definition 21 (Strict Step − Time Condition Shortest Path (SS−TCSP))**.** *This variant is the same as $SSR-TCSW$ in that every edge used must move up in time. However, the difference in this variant is that the solution must be a path, not a walk (i.e. cannot traverse the same vertex twice, which implies no edge can be reused).*

Definition 22 (Repay − Time Condition Shortest Walk (R−TCSW))**.** *Given the same inputs as $TCSW$, the problem becomes to find an $a-b$ walk W from G_1 to G_T where edges are paid for per use (rather than just once).*

Definition 23 (Monotonic−Time Condition Shortest Walk (Mon−TCSW))**.** *This variant is similar to vanilla $TCSW$, except now (v_i,w_j) is a valid edge if $v \in V_i, w \in V_j$ and $i \leq j$. We note in $k-TCSW$, we allow this jump to be up to step size k, but in this variant jumps can be arbitrarily large as long as they are monotonic.*

Definition 24 (Multi−k Time Condition Shortest Walk (Multi−k−TCSW))**.** *In this variant, we allow for a set of n demands $(a_1,b_T^1),...,(a_1,b_T^n)$. Our goal is to find a set of n walks starting from a singular source a_1, and ending at b_T^i. The cost paid is the sum of the weight of all edges used in one or more walks (each edge is only paid for at most once). This variant is most suitable for protein signalling cascades, whereby often times a signal begins at a singular receptor, and through a series of interactions, many downstream proteins are affected. Naturally $k-TCSW$ is special case with only one demand.*

2.3.6 Our Results

In this work, we build off the results of Wu et al. In particular we aim to show that similar to the Condition Shortest Path problems, extending the Condition setting to TCSW, a singular global shortest path problem with the *temporal walk constraint*, becomes hard to solve and hard to approximate to any non-trivial factor. We then relax the *temporal walk constraint*, and show a neat tradeoff between the temporal walk constraint and the hardness of approximation. Lastly, we present an ILP formulation for these problems, and demonstrate the ability to find optimal solutions to the generalized k-TCSW problem in feasible

time on real world applications over the human Protein Protein Interaction Network (PPI).

We begin by proposing a rather simple algorithm for solving the $TCSW$ problem for $T \leq 2$.

Theorem 9. *TCSW can be solved in polynomial time for $T \leq 2$.*

While this result is promising, the regime of polynomial time algorithms ends here.

Theorem 10. *TCSW is hard to approximate to a factor of $\lceil \frac{T}{2} \rceil - \epsilon$ for every fixed $T \geq 3$ and $\epsilon > 0$.*

Thus the best approximation ratio one can hope for is $\lceil \frac{T}{2} \rceil$. Such ratio can easily be achieved by considering the global network $\mathcal{G}$ defined amongst all G_i as follows: Create a network $\mathcal{G}$ with node $v_i \in \mathcal{G}$ if $v_i \in V_i$. An edge $e = (v_i, w_j)$ exists in $\mathcal{G}$ if $v_i \in V_i, w_j \in V_j$, $i = j$ or $i = j + 1$, and $(v_i, w_j) \in E$. In addition, we add zero weight edges for all nodes v_i, v_{i+1} if $v_i \in V_i$ and $v_{i+1} \in V_{i+1}$. Find the shortest path in this network from a_1 to b_T (thus ignoring the benefits of re-using an edge). An edge may be re-used once per alternate level, thus giving us a solution as bad as $\lceil \frac{T}{2} \rceil$ away from OPT.

By relaxing the *temporal walk constraint*, we are left with a more general result

Theorem 11. *The general problem of k-TCSW is hard to approximate to a factor of $\lceil \frac{T}{k+1} \rceil - \epsilon$ for every fixed $k \geq 1$, $T \geq k + 2$, and $\epsilon > 0$.*

Similar to vanilla TCSW, we can generate a global network $\mathcal{G}$ with node $v_i \in \mathcal{G}$ if $v_i \in V_i$. An edge $e = (v_i, w_j)$ exists in $\mathcal{G}$ if $v_i \in V_i, w_j \in V_j$, $i \leq j \leq j + k$, and $(v_i, w_j) \in E$. In addition, we add zero weight edges for all self nodes up to k timepoints away (rather than 1). Similarly find the shortest path in this network from a_1 to b_T. In this instance, an edge may be reused once per $k+1$ level, thus giving us a solution as bad as $\lceil \frac{T}{k+1} \rceil$ away from OPT.

Although these results are rather pessimistic in the theoretical abstract we provide a more consoling view by formulating an integer linear program for the general TCSW problem. We then show in experiments on real-world inputs derived from the human PPI network, the ILPs are capable of finding optimal solutions in a reasonable amount of time.

2.3.7 Hardness of Time Condition Shortest Walk

Theorem 9. *TCSW can be solved in polynomial time for $T \leq 2$.*

Proof. Note that when $T = 1$ we're simply left with the vanilla shortest path problem, which can easily be solved by algorithms such as Djikstra's or Thorup's Algorithm.

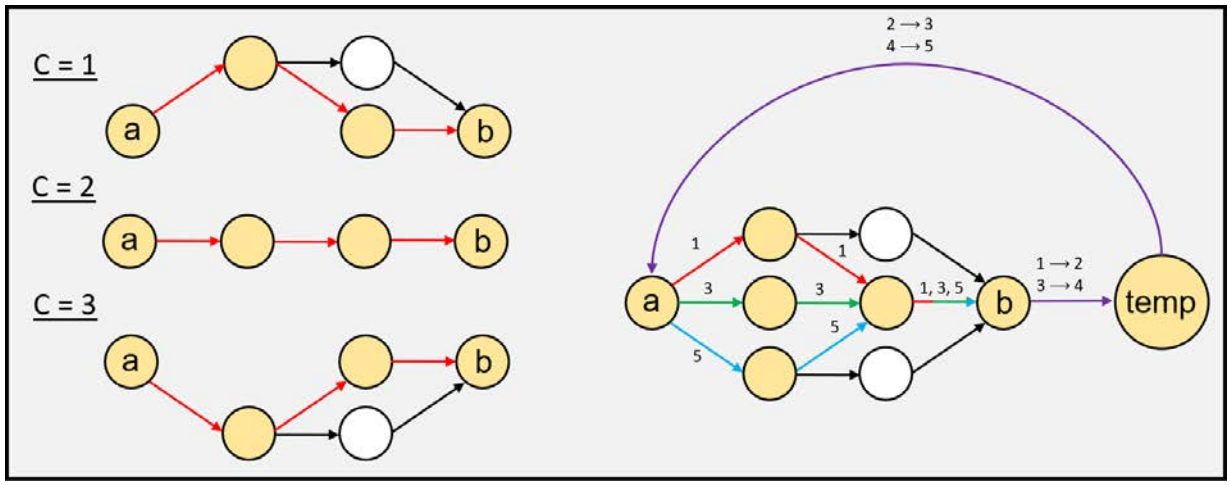


Figure 2.5: Example reduction from a Condition Shortest Path instance (left) with 3 conditions to an instance of TCSW with 5 time points (right). In the TCSW instance, red edges are traversed in timepoint 1, green edges in timepoints 3, blue edges in timepoint 5, and purple edges are traversed from timepoint i to timepoint $i+1$.

Now let us consider the case for $T = 2$. We argue that an optimal walk is in fact a path, and thus applying shortest path on the global network $\mathcal{G}$ will provide the optimal solution.

Consider the global network $\mathcal{G}$ as defined earlier in Definition 17. Assume for the sake of contradiction that there exists a walk W that has a lower cost than a shortest path P from a_1 to b_2 in $\mathcal{G}$. Due to our assumption and by definition of $\mathcal{G}$, this walk cannot be a simple path. Therefore that implies there exists at least one vertex $v \in V$ that was traversed twice in this walk, once from (v_i, w_j) and once from $(v_{i'}, x_{j'})$. Now consider the walk W' formed by concatenating this portion of the walk and simply taking the edge $(v_i, x_{j'})$. As we only have two time points, this is a still a valid traversal and satisfies the *temporal walk constraint*. We can therefore apply this concatenation process until every vertex is traversed only once, thus forming a simple path P', which has less than or equal cost to W. Therefore we arrive at a contradiction, as the optimal walk from a to b is in fact a path. □

Theorem 10. *TCSW is hard to approximate to a factor of* $\lceil \frac{T}{2} \rceil - \epsilon$ *for every fixed* $T \geq 3$ *and* $\epsilon > 0$.

We approach this proof via approximation preserving reduction from Condition Shortest Path to TCSW. Recall that in the Condition Shortest Path problem, we are given networks $G_1, ..., G_C$ and a source target pair (a, b). The goal is to find a subnetwork $H \subseteq G = \bigcup_c G_c$ of minimal cost such that there exists a path from a to b in H amongst vertices that are active in G_c, for all c.

Therefore, given an instance of CSP $(G_1, ..., G_C, (a, b))$, with underlying network $G = (V, E)$ the reduction works as follows:

- Construct an instance of TCSW with $2C - 1$ networks $G'_1, ..., G'_{2C-1}$. Let the odd networks in our constructed instance of TCSW $G'_{2c-1} = G_c$ for $c \in [C]$.
- For even instances G_{2c} for $c \in [C - 1]$, let G_{2c} be a network with exactly one node t^*.
- In addition to the edges E from our CSP instance, add two directed edges to our global edge set: (b, t^*) and (t^*, a), which we call our transition edges.
- Our singular demand is to find a walk from a_1 to b_{2C-1}
- The network $H \subseteq G$ that satisfies CSP is formed by taking the union of edges traversed in the TCSW instance solution W minus the transition edges (b, t^*) and (t^*, a)

We first note that by construction, the only way to go from a_1 to b_{2C-1} is to first go from a_1 to b_1, then take the transition edges from b_1 to t_2^* and t_2^* to a_3, and then repeat. Therefore W contains a path from a_c to b_c that only goes through edges in G_c for all $c \in [C]$, and therefore the union of edges $e \in W = H$ forms a valid solution for our CSP instance.

Now assume for the sake of contradiction that there exists a solution $\mathcal{H} \subseteq G$ of lower overall cost than the network H returned from our TCSW reduction. For each condition $c \in [C]$, consider an $a - b$ path going through H using only edges in G_c. Let the set of edges used in such path be called P_c. We construct a walk $\mathcal{W}$ as follows: $\mathcal{W} = [P_1, (b, t^*), (t^*, a), P_2, ..., P_C]$. This forms a valid walk in our TCSW instance, and has lower cost than W, as edge weights are only paid for once in both the CSP instance and the TCSW instance. This leads to a contradiction that W was our optimal walk.

Therefore, all CSP instances can be solved via a reduction to TCSW. Naturally, this leads us to the inapproximability of TCSW. Namely, CSP cannot be approximated to a factor of $C - \epsilon$ for all fixed $C \geq 2$ and $\epsilon > 0$. Therefore, by extension TCSW cannot be approximated to a factor $\frac{T+1}{2} - \epsilon$ for all fixed odd $T \geq 3$ and $\epsilon > 0$, as we have $2C - 1$ temporal conditions in our TCSW instance for a CSP instance of C conditions.

We note one caveat, which is that we're always reducing to an odd number of time conditions. It is straight forward to see that we could have equivalently reduced to an even number of time conditions by adding an extra network G_{2C} with just the node b, and instead solving for the walk from a_1 to b_{2C}, thus proving that even instances are inapproximable to a factor of $\frac{T}{2} - \epsilon$ for all fixed even $T \geq 4$ and $\epsilon > 0$. As a result, we can make a more general statement from these two that TCSW is inapproximable to a factor of $\lceil \frac{T}{2} \rceil - \epsilon$

Theorem 11. *The general problem of k-TCSW is hard to approximate to a factor of* $\lceil \frac{T}{k+1} \rceil - \epsilon$ *for every fixed* $k \geq 1$, $T \geq k + 2$, *and* $\epsilon > 0$.

This proof follows directly from Theorem 10. The reduction from CSP to k-TCSW follows the same flavor as the one above with the following key difference:

- Rather than constructing $2C-1$ networks, we construct $(k+1)C-k$ networks. We let every $(k+1)c-k$ network $G'_{(k+1)c-k} = G_c$.
- In addition, we let $G'_{(k+1)c}$ be a network with exactly one node t^*
- All other networks G'_i are empty
- We maintain the edge set formed previously, with the same two transition edges (b, t^*) and (t^*, a).
- Our singular demand is to find a walk from a_1 to $b_{(k+1)C-k}$
- The network $H \subseteq G$ that satisfies CSP is formed by taking the union of edges traversed in the TCSW instance solution W minus the transition edges (b, t^*) and (t^*, a)

We note that the exact same arguments work for why H is a valid and optimal solution for the CSP instance. One caveat that the reader may notice, similar to the odd even bifurcation in the prior section is that we always reduce CSP to a k-TCSW instance with $T = (k+1)C-k$ time conditions. However, with the same idea of adding dummy networks at the end of our k-TCSW, we encompass all other instances of k-TCSW. As CSP is hard with $C = 2$, TCSW is NP-hard for all instances $(k+1)2-k = k+2$. In addition, k-TCSW is hard to approximate to factor $\lceil \frac{T}{k+1} \rceil$ by similar argument.

Building the protein-protein interaction network

In order to construct a human protein protein interaction (PPI) network network for our simulations, we collected data to construct a weighted directed network from four sources. Our largest dataset came from InWeb [119], where protein-protein interactions were treated as bidirectional edges from the proteins used. Edge weights were set as the negative log confidence score collected. Similarly, PPIs from the Human Protein Reference Data (HPRD)[107] were treated as bidirectional, but assigned the minimum nonzero confidence values by default. For directed edges, collected from highly curated data-sets, we used Phosphosite[86] and NetPath[104], and assigned edges sourced from both our maximal confidence value

Solving k-TCSW to optimality

We can derive a natural linear program for k-TCSW in terms of network flows, by demanding a unit of flow from timepoint 1 for source s to arrive at timepoint T for target t, while maintaining the temporal walk constraint. We present the specific formulation in Figure 2.6.

$$\text{minimize} \quad \sum_{(u,v)\in E} d_{u,v} \cdot w(u,v)$$

subject to

$$\sum_{(u,v)\in E} \sum_{t'\in T} d_{uvtt'} = \delta_{vt} + \sum_{(v,w)\in E} \sum_{t'\in T} d_{vwtt'} \quad \forall t \in [T], v \in V$$

$$d_{uv} \geq d_{uvtt'} \quad \forall t, t' \in [T], e \in E$$

$$d_{uvtt'} = 0 \quad t' < t \text{ or } t' > t + k, \forall e \in E$$

$$d_{uvtt'} = 0 \quad u \notin V_t \text{ or } v \notin V_{t'}, \forall e \in E$$

$$d_{uvtt'} \in \{0, 1\}$$

$$d_{uv} \in \{0, 1\}$$

Figure 2.6: Integer linear program for k-Time Conditioned Shortest Walk. $\delta_{vt} = 1$ for v at time 1 if v is the source s, -1 if v is target t at time T, 0 otherwise. Each variable $d_{uvtt'}$ denotes the flow through edge (u, v) from time t to time t'; each variable d_{uv} denotes whether (u, v) is ultimately in the chosen walk solution;. The first constraint enforces flow conservation by demanding 0 flow through all nodes except the source s and target t. The second constraint ensures that if an edge is used at any condition, it is chosen as part of the solution. The third constraint ensures that a jump of no larger than k is taken by forcing 0 flow through edges of greater time length. The fourth constraint ensures that both ends u and v exist in V_t and $V_{t'}$ respectively.

Performance analysis of integer linear programming

Given the protein-protein interaction network G, we sample an instance of k-TCSW by sampling a source node $a \in V_1$ and target node $b \in V_T$ such that there exists a walk from a to b which satisfies the k temporal walk constraint. All other nodes exist in G_t with probability p.

Using a workstation running an Intel Xeon E5-2690 processor and 250GB of RAM, **optimal solutions** to instances of modest size (generated using the procedure just described) were within reach:

Table 2.3: ILP solve times for some random instances of k-TCSW generated by our random model using the Gurobi Python Solver package[73].

k	T	p	Time to solve
1	1	0.25	40s ± 5s
1	1	0.75	40s ± 5s
1	3	0.25	4m 30s ± 1m
1	3	0.75	20m ± 4m
1	5	0.25	14m ± 6m
1	5	0.75	24m ± 30s
2	5	0.25	20m ± 6m 30s
2	5	0.75	25m ± 30s
1	10	0.50	31m ± 1m 30s

We note that runtime seems to largely depend on the number of time conditions T, with some additional dependence on p, which loosely measures the size of our frames G_t. Through these simulations we present a model which is applicable to real world biological instances, and can solve for optimal solutions in a feasible amount of time.

2.3.8 Conclusion and discussion

In this manuscript we introduced the Time Condition Shortest Walk (TCSW) problem, in which the goal was to find an $a - b$ path beginning in an initial time conditioned frame G_1, and ending in G_T, with jumps of length at most one. Unlike the shortest path problem, which is tractable, we demonstrated via approximation preserving reduction to Condition Shortest Path (CSP) that TCSW is hard to approximate within a factor $\lceil \frac{T}{2} \rceil - \epsilon$ for $T \geq 3$ and $\epsilon > 0$. We then expanded this problem to a broader definition, k-TCSW, allowing for jumps of up to size k in this $a - b$ path. We demonstrated a direct inverse relation between the jump size k, and best approximation ratio achievable, with $k - TCSW$ being hard to approximate to $\lceil \frac{T}{k+1} \rceil - \epsilon$ for $T \geq 3$ and $\epsilon > 0$ $k \geq 1$. Lastly, we developed an integer linear program modeled on network flows, and applied it to solve for exact solutions over simulated instances on the human protein-protein interaction network, demonstrating feasibility runtimes for real-world instances.

In this work we also briefly explored a variety of alternative formulations for this problems, some being tractable, and others intractable. We believe a natural extension of this work would be to define the time conditioned frames over a series of variable edges, or both variable edges and vertices. Some modifications would have to be made to account for the ability to traverse self edges. Lastly, we believe given the feasibility demonstrated by our simulations, the next step would be to apply this method to cell signalling data spanning multiple time points.

2.3.9 Analysis of problem variants

Proposition 4. *Strict Step Repay - Time Condition Shortest Walk (SSR-TCSW) can be solved in polynomial time*

Proof. We provide a simple algorithm to solve the problem in $O(|T||E| + |T||V|log(|T||V|))$

Given the following inputs $(G_1, ..., G_T; a, b \in V; T)$, let $V' = \bigcup_{t\in[T]} V_t$. We create a new edge set E', where say there exists an edge e' in E' between $v_t, w_{t+1} \in V'$ if there is an edge e between $v, w \in E$, and $v \in V_t$ and $w \in V_{t+1}$. Let $w'(e') = w(e)$. We simply run Djikstra's algorithm on the new input as $W = Dijkstra(a, b, G' = (V', E', w'))$, and return W as an optimum walk.

Note that this is just a simple modification of the shortest walk problem where we must use exactly $T - 1$ edges. By construction each edge moves us up exactly one time point. The only difference is that we cannot pass through certain vertices at certain time points , which we account for by not including the corresponding vertices in our modified network G'.

□

Proposition 5. *Strict Step - Time Condition Shortest Path (SS-TCSP) is NP-Hard*

Proof. We will show a simple reduction from Hamiltonian Path to SS-TCSP. Consider an instance of Hamiltonian Path, where given a graph $G = (V, E)$, we are asked to find a simple path P s.t. P visits all vertices of G.

Let $|V| = n$. We now generate a new instance of SS-TCSP. First initialize $G_2, ..., G_{n+1} = G$. Initialize G_1 with a singular source node s and G_{n+2} with a target node t. Define $T = n + 2$. Initialize $E' = E$, and add to E' edges (s, v) and (v, t) for $v \in V$

Now consider the instance $I = \{G_1, ...G_{n+2}, (s, t), T\}$. We note that during time $\{2, .., |V|+ 1\}$ the path must stay in the original G, and based on the definition of Simple Path, we are not allowed to visit a node more than once. Therefore, if a solution exists to I, it must go through each node in G exactly once, which can only be the case if and only if there is a Hamiltonian Path in G. Therefore, any instance of Hamiltonian Path can be reduced to an instance of TCP, which implies TCP is NP-Hard. □

Proposition 6. *Repay-Time Condition Shortest Walk (Mon-TCSW) can be solved in polynomial time*

Proof. We note that the solution to this problem is rather trivial. Recall the network $\mathcal{G}$ from Definition 17. Run Djikstra's starting from a_1 to find the shortest path to b_T. By

construction of $\mathcal{G}$ the walk satisfies the temporal walk constraint. In addition, as edges are paid for per use this will find the optimal walk from a to b spanning G_1,...,G_T. □

Proposition 7. *Monotonic-Time Condition Shortest Walk (Mon-TCSW) can be solved in polynomial time*

Lemma 6. *An edge $e \in E$ is traversed more than once in W only if there is a cycle in W*

Proof. Let $e = \{v, w\}$. If e appears in W more than once, this implies that v was visited more than once, which implies there is a cycle in W. □

Lemma 7. *There exists an optimal walk W for Mon-TCSW that is a **Simple Path**.*

Proof. Assume for the sake of contradiction that there exists an walk W' that contains cycles that is better than W, our best Simple Path.

Let v be a node visited twice. This implies that $W' = \{..., \{v_1, w\}, ...\{v_2, y\}, ...\}$, $w, y \in V$. However as y appears temporally after v_2 in the path, it appears temporally after v_1, which implies we can concatenate the cycle and get $W^* = \{..., \{v_1, y\}, ...\}$ which has equal or lower cost than W' as all edge weights are positive. By applying this argument inductively, we arrive at a Simple Path P with no cycles that has lower cost than W'. Contradiction.

This implies the optimal walk W is a **Simple Path**. □

Lemma 8. *The optimal path does not visit each edge more than once. Alternatively, it is enough to consider* $\min \sum_{e \in W} w(e)$

Proof. By Lemma 2, the optimal walk W is a simple path. By Lemma 1, no edge is visited more than once.

Therefore, we simply suggest a modified version of the algorithm in for SSR-TCSW. Instead of simply introducing edges in SSR-TCSW from v_t to w_{t+1}, we introduce edges from v_t to $w_{t'}$ $\forall t' \geq t$ as long as $(v, w) \in E$. Therefore, this gives us a simple algorithm to solve the problem in $O(|T|^2|E| + |T||V|log(|T||V|))$. □

2.3.10 Formal Definition of TCSW and its variants

Definition 25 (Time Condition Shortest Walk (TCSW))**.** *Consider the following inputs:*

1. *A global weighted network $\mathcal{G} = (V, E, w)$*
2. *A temporal activity function $\rho : (V, \{1 \dots T\}) \rightarrow \{0, 1\}$, indicating whether a node $v \in V$ was active during time point $t \in \{1 \dots T\}$*
3. *A pair of nodes $(a, b) \in V$*

We define $\mathcal{G}^* = (V^*, E^*)$ as the *ρ-unwinding* of $\mathcal{G}$. In this graph, the set of nodes V^* includes multiple copies of every node in V, with one copy for each time point in which it was active. Formally, denote the copy of $u \in V$ at time t as u_t and let $V_t = \{u_t \text{ s.t. } u \in V \wedge \rho^t(u) = 1\}$. We then define $V^* = \bigcup_t V_t$. Similarly, we define E^* as the set of edges that connect node instances from the same time points or from adjacent time points. Formally, let $E_t = \{< u_t, v_{t+i} > \ s.t. \ (u = v \vee \ < u, v > \in E) \ \wedge \ \ (\ell \leq i \leq k)\}$

Notably, any path $P \in E^*$ would be time consistent - namely it will only include transitions between contemporary nodes or between nodes at adjacent time points, moving up in time. For a path $P \in E^*$ we denote by $c(P)$ the cost with edge repayment per use and $c^*(P)$ the cost without repayment per use (edges traversed at two different time points are considered the same edge). In the TCSW problem, we have a *ρ-unwinding* of $\mathcal{G}$ with $\ell = 0, \ k = 1$ and optimize for $c^*(P)$. In the SSR-TCSW problem, we use $\ell = 1, \ k = 1$ and optimize for $c^*(P)$. In the Mon-TCSW we use $\ell = 0, \ k = T - 1$ and optimize for $c(P)$.

2.4 Investigating Novel Downstream Targets of IL23R Signalling

2.4.1 Collaborators and Contributions

The last section of this dissertation describes a work in progress between the Yosef Lab and the Kuchroo Lab (Harvard University). While experimental validations are currently ongoing, the computational results look promising and are summarized below.

The primary collaborators of this work include: Alex Khodaverdian, Markus Schramm (Harvard), Sai Harsha Krovi (Harvard), Nandini Acharya (Harvard), Nir Yosef (Principal Investigator, Berkeley), and Vijay Kuchroo (Principal Investigator, Harvard).

2.4.2 Introduction

The IL17-IL23 axis plays a crucial role in pathologies characterized by chronic inflammation. IL-23 is secreted by dendritic cells and macrophages which contribute to the expansion and survival of Th17 cells by upregulating IL-23R expression, and activating subsequent downstream signaling cascades. In steady state there is a fine balance between inflammation and anti-inflammatory response. In the gut, differentiation of Th17 cells is influenced by various factors such as the presence of SFB and Candida albicans as previously shown by Littman and others. Such non-pathogenic Th17 response is crucial to maintain tissue homeostasis. However, induction of IL23 for prolonged periods of time leads to induction of anti-inflammatory cytokine IL-10 which prevents overt inflammation(cite papers). Interestingly, in certain viral infections and autoimmunity such as multiple sclerosis and inflammatory bowel disease (IBD) [183, 117, 106] this negative feedback loop is dysregulated by an unknown mechanism.

Although the surface receptor (IL23R) and ultimate endstream cytokine (IL17) are well known, the question of interest is understanding the manner in which signal transduction occurs once IL-23R has been stimulated. To this effect, several studies have been conducted. For example, in high-salt conditions, the loss of SGK1, NFAT5, or p38/MAPK nulls the development of stable Th17 cells [186]. In addition to these studies, therapeutics have been developed to tackle Th17 induced autoimmune diseases. From a therapeutic perspective, a monoclonal antibody secukinumab targeting the downstream cytokine IL-17A has been effective against diseases such as psoriatic arthritis and plaque psoriasis, while ineffective at dealing with IBD [91]. On the other hand, ustekinumab, targeting IL-12 and IL-23 has proven to be effective in treating IBD [156, 157]. Our goal therefore is to find other molecules of interest that are specific to IL-23R signaling, and in particular associated with pro-Th17 phenotypes, some of which may be useful as therapeutic targets moving forward.

To satisfy this goal, we turn towards Th1 cells. Th1 cells are very different in their function from Th17 cells, and in particular are induced via IL-12 stimulation via IL-12 receptor signaling. Interestingly, IL-12-receptor shares a subunit in the form of IL-12Rβ1 with IL-23 receptor, and has a separate subunit IL-12Rβ2. As a result, certain proteins such as STAT4 are found closely downstream of both receptors, whereas other proteins such as STAT3 are found only downstream of IL-23R, the second subunit of IL-23 receptor [59]. Therefore, IL-12Rβ2+ cells stimulated via IL-12 serve as a great contrast to compare against IL-23R+ cells stimulated by IL-23 to discover IL-23R specific signaling targets, and thus Th17 specific signaling targets.

Using the Th1/Th17 contrast to our advantage, we utilize a multi-omics approach to further our understanding of IL-23 specific signaling proteins. We began by collecting data from 3 individual human donor pools of 4-5 donors per pool. In particular CD4+CD25-CD45RO+ memory T cells were treated with anti-CD3/anti-CD28 beads and transduced with IL-23R or IL-12Rβ2 lentivirus and subsequently sorted for such receptors. Phosphoproteomics data was collected within the first hour, and transcriptomics data within the same day (Figure 2.7a). Through differential expression and enrichment analysis, we calculated significance scores across multiple measurements per gene, and applied a consensus methodology to identify two novel targets of interest. In particular, we identify glucocorticoid receptor (NR3C1), and the chromatin remodeling protein Chd1, both of which we are experimentally validating.

2.4.3 Results and Methodology

Given access to transcriptomic and phosphorylation data, our goal became to leverage both to identify gene targets that were significant across multiple measurements. To this effect, we devise a consensus approach leveraging two direct measurements of protein activity, and three indirect measurements of protein activity, to rank genes. We briefly describe our five criteria.

Differential Gene Expression Analysis

To assign direct gene expression significance scores, two comparisons were conducted. IL23-R cells treated with additional IL23 at the 4 hour mark were compared against IL23-R cells without additional IL23. The same was done at the 20 hour mark. Differential expression was done via a t-test, and false discovery rates were calculated via Benjamini-Hochberg correction. Each gene was assigned the minimum FDR across these two comparisons. Several genes of interest are highlighted in Figure 2.7c.

Differential Phosphorylation

To assign phosphoproteomics significance scores, three comparisons were considered. In particular, IL23-R enriched cells were compared against one another in time (at 45 min-

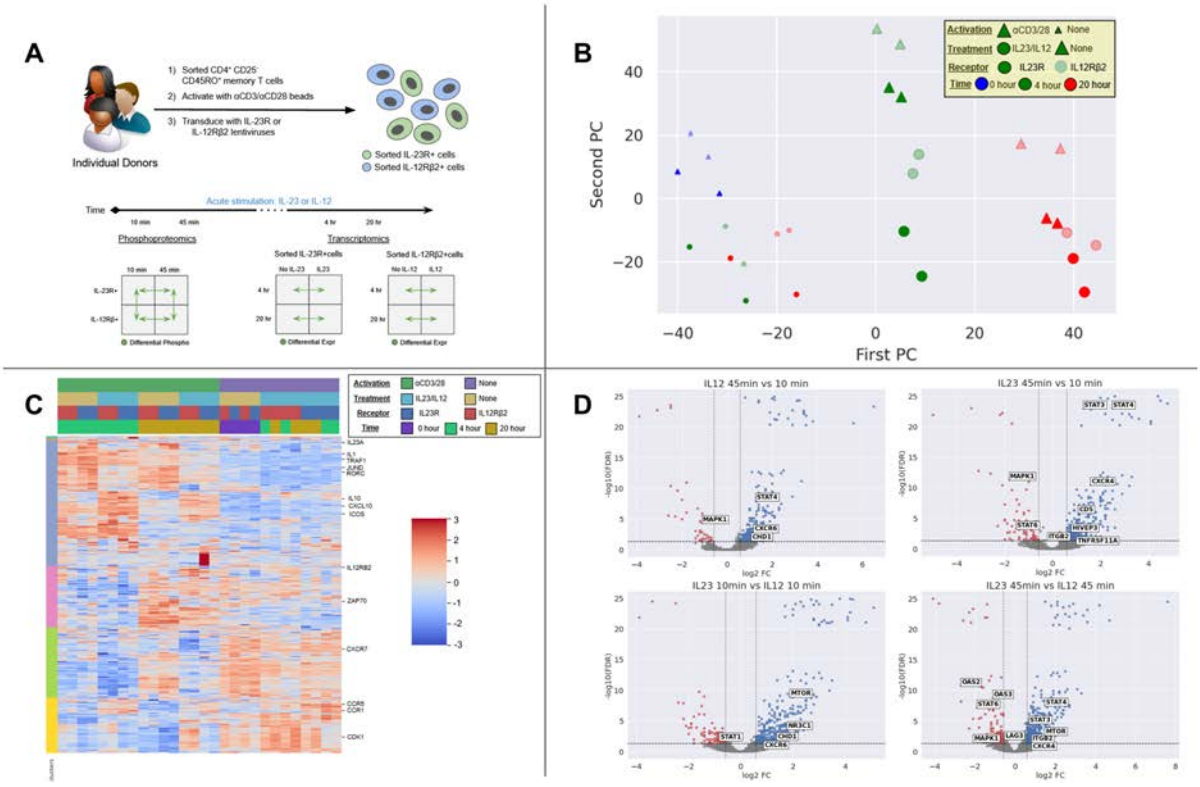


Figure 2.7: A) Experimental Setup: Given a donor pool of 3 individual donors, CD4+ CD25- CD45RO+ memory T cells were sorted, activated with aCD3/aCD28 beads, and transduced with IL-23R or IL-12Rb2 lentiviruses. These cells were then further sorted for IL-12Rb2 or IL-23R. Data was collected at four time points, 10 min and 45 min, and after being further stimulated with IL12 and IL23, at 4 hours and 20 hours. Given these measurements, a series of differential expression analyses were performed both on the transcriptomic and phosphoproteomic level. B) PCA plot of the raw gene expression data. We note the importance of the following metadata in order: Activation by aCD3/aCD28, time, treated with additional IL12/IL23, receptor. C) Gene expression heatmap of the most differentially expressed genes, with key genes of interest highlighted D) Volcano plot of all genes within the phosphoproteomics comparisons conducted, with key genes highlighted

utes versus 10 minutes), and versus IL12-R enriched cells at two timepoints (at 45 minutes and 10 minutes). This analysis was conducted with enterprise software SAS version 9.3, and once again the minimum FDR across all three comparisons was taken for every gene's phosphorylation score. Several genes of interest are highlighted in Figure 2.7d.

Inferring Transcription Factor Activity

In cell biology, a subset of genes are known as transcription factors. When present, these transcription factors affect expression of a group of downstream genes. Although a transcription factor may not show up as directly differentially expressed, our hope was that transcription factor activity may be inferred if enough downstream genes were active, thus marking a gene interesting if enough activity was observed.

Working towards this analysis we began by collecting a list of commonly known transcription factors and their downstream targets from ENCODE. Given this list of transcription factors, for each of the two comparisons in the gene expression, we conducted a hypergeometric test between the differentially expressed genes and the known downstream targets of the transcription factors. Similar to the gene expression, we assigned a transcription factor score to each gene as the minimum FDR across both comparisons.

Inferring Protein Protein Interaction Activity

Similar to the case of transcription factors, proteins are known to impact the activity of one another via physical binding. To this end, our goal was to leverage databases of known protein protein interactions to indirectly infer protein activity from differential phosphorylation analysis.

To this end a protein protein interaction network was generated from data from both Inweb and the Human Protein Reference Database (HPRD). Hypergeometric enrichment analysis was done for every gene in this network over neighboring genes across the three comparisons highlighted in our differential phosphorylation analysis. For every gene, we again took the minimum FDR across the three comparisons for each enrichment analysis.

Inferring Post Translational Modification Activity

In addition to bi-directional physical protein-protein interactions, proteins are known to impact the activity of other proteins in a unidirectional manner in the form of post translational modifications.

We considered protein activity indirectly inferred via post translational modifications by forming a network from interactions found in both Netpath and Phosphosite. Once again for every gene, hypergeometric enrichment analysis was conducted across the three phosphoproteomics comparisons, and FDRs were aggregated for each gene across such comparisons.

Consensus Proteins of Interest

At the end of such analysis, five scores were calculated per gene. Two scores directly came from differential expression analysis in the transcriptomic and phosphoproteomic data.

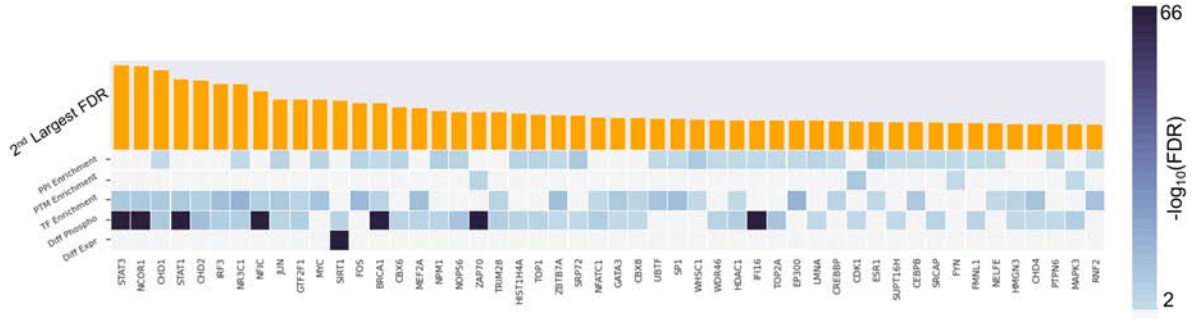


Figure 2.8: Visualization of the ranking procedure used to identify protein of interest. In particular 5 criteria were considered: differential gene expression, differential phosphorylation, transcription factor enrichment, protein-protein interaction enrichment, and post translational modification enrichment. The rankings were formed by taking the second lowest p-value from the five comparisons for each protein. We highlight in particular CHD1 and NR3C1 as top hits

Three scores came from indirect inference of gene activity, by considering neighboring activity in downstream transcription factor targets, physical protein binding, and post translational modifications. Genes were ultimately ranked by taking the second best FDR across these scores. Top genes are visualized in Figure 2.9.

Several of these gene targets, such as STAT3, STAT1 and IRF3 have been well studied in the context of Th17 cells. For example, IRF3 is a critical regulator of experimental autoimmune encephalomyelitis in mice, with mice deficient in IRF3 showing significantly reduced disease state [58].

Other top gene targets, such as Chd1 and NR3C1 on this list are far less studied in the context of Th17 cells. Within $CD8^+$ T cells, NR3C1 has been associated with T cell differentiation and dysfunction in the tumor microenvironment. Conditional deletion results in improved effector differentiation, and inhibition of the dysfunctional phenotype, resulting in tumor growth inhibition [1].

Inferring Protein Activity by Flow Optimization

In approaching this analysis, we also considered a different, and slightly more complicated approach to identify target genes. We began by considering the full protein protein interaction network discussed in the ILP-formulations of our theoretical network analysis. Our approach was the following: We can devise a flow program whereby n units of flow begin at IL23-Receptor, and must be sent to each of n differentially phosphorylated genes. We begin by calculating the cost, OPT of sending this flow. We then, for every gene g,

blockade that gene from being used in the flow solution, and calculate the cost, OPT_g given this constraint. From there, we consider $\frac{OPT_g}{OPT}$. If the cost is significantly increased, there is a high likelihood that g is a critical gene necessary for the IL23 signalling.

Unfortunately, this methodology proved to not work as anticipated, as $\frac{OPT_g}{OPT} = 1 + \epsilon$, which was too close to 1 for any significant signal. Although this methodology failed to identify genes in our specific instance, I consider it an approach that may be applied in future endeavors.

Preliminary Experimental Validation of CHD1 and NR3C1

Given the top hits of interest, our attention focused towards CHD1 and NR3C1, both of which were picked based on the fact that they were transcription factors, and were previously unstudied in Th17 cell models. Preliminary experimental results from actively induced EAE, as presented below, suggest that loss of Chd1 ameliorates EAE, while loss of NR3C1 worsens EAE.

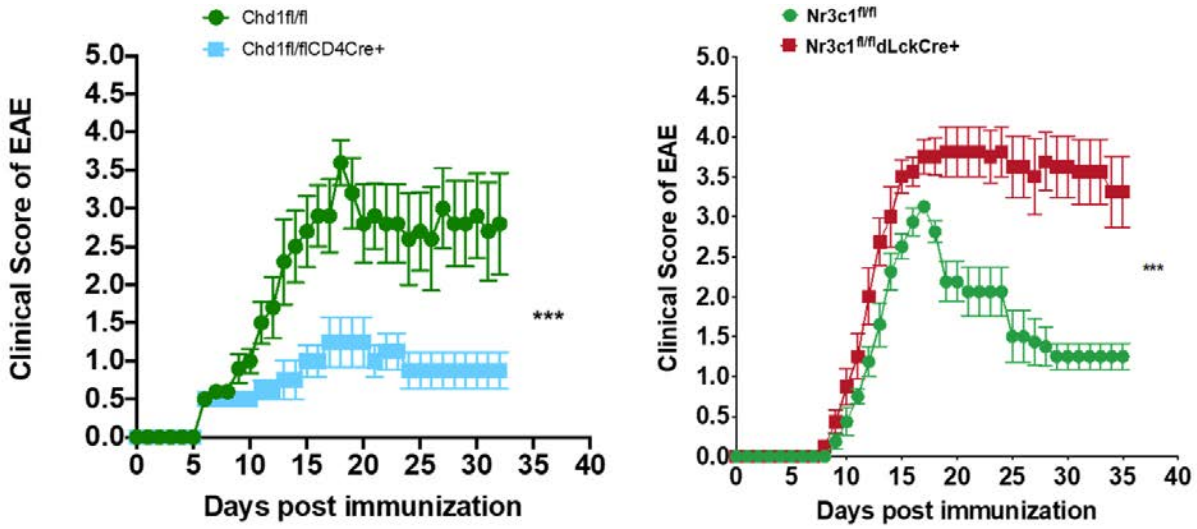


Figure 2.9: Clinical score of active EAE in female mice induced via 100mg of Myelin Oligodendrocyte Glycoprotein

Chapter 3

Building Networks - Reconstructing Phylogenies of Single Cell

3.1 Introduction

In this chapter, we consider the problem of lineage tracing in CRISPR/Cas9 models - given a set of terminal nodes or cells generated via CRISPR/Cas9 lineage tracing, what is the most likely tree that best represents the ground truth generative process.

In the first section, we approach this problem from a practical/heuristic perspective. In particular we introduce three methods towards this analysis - a greedy top down approach which iteratively groups cells with frequent mutations, an exact integer linear programming approach, and a hybrid approach. We validate the accuracy of these methods via simulation and via ground truth trees generated in vitro.

In the second section, we consider this problem from a theoretical perspective. In particular, we propose two algorithms and derive upper bounds for the amount of characters/cut sites required for exact reconstruction with high probability. Furthermore, we characterize the dependence between the necessary number of cut sites in our model against variables such as minimum cell division times, number of unique states, and cutting rates. Lastly, we validate these relationships between the number of cut sites and our various experimental parameters via simulations, and present empirical bounds on the number of cut sites required for exact reconstruction.

3.2 Inference of single-cell phylogenies from lineage tracing data using Cassiopeia

3.2.1 Authors and Contributions

Matthew G. Jones*, Alex Khodaverdian*, Jeffrey J Quinn*, Michelle M Chan, Jeffrey A Hussmann, Robert Wang, Chenling Xu, Jonathan S Weissman, Nir Yosef. Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. Genome Biology 2020

* signifies Equal Contribution

M.G.J., A.K., J.J.Q, N.Y, and J.S.W. contributed to the design of the algorithm, interpretation of benchmarking results, and writing of the manuscript. A.K., C.X., and N.Y. conceived of the multi-state greedy algorithm and Steiner-Tree adaptation for the phylogeny inference problem. A.K. and M.G.J. implemented the algorithms and all code relevant to the project. M.G.J. and A.K. conducted all stress tests on synthetic datasets. R.W. and A.K. conducted experiments and theoretical work regarding the greedy heuristics robustness in lineage tracing experiments. J.J.Q. generated the *in vitro* reference dataset. M.G.J., J.J.Q, M.C, and J.A.H. designed the processing pipeline for empirical lineage tracing data. M.G.J. and J.J.Q processed the reference dataset and M.G.J. reconstructed trees.

3.2.2 Abstract

The pairing of CRISPR/Cas9-based gene editing with massively parallel single-cell readouts now enables large-scale lineage tracing. However, the rapid growth in complexity of data from these assays has outpaced our ability to accurately infer phylogenetic relationships. First, we introduce Cassiopeia - a suite of scalable maximum parsimony approaches for tree reconstruction. Second, we provide a simulation framework for evaluating algorithms and exploring lineage tracer design principles. Finally, we generate the most complex experimental lineage tracing dataset to date, 34,557 human cells continuously traced over 15 generations, and use it for benchmarking phylogenetic inference approaches. We show that Cassiopeia outperforms traditional methods by several metrics and under a wide variety of parameter regimes, and provide insight into the principles for the design of improved Cas9-enabled recorders. Together these should broadly enable large-scale mammalian lineage tracing efforts. Cassiopeia and its benchmarking resources are publicly available at `www.github.com/YosefLab/Cassiopeia`.

3.2.3 Background

The ability to track fates of individual cells during the course of biological processes such as development is of fundamental biological importance, as exemplified by the ground-

breaking work creating cell fate maps in *C. elegans* through meticulous visual observation [170, 44]. More recently, CRISPR/Cas9 genome engineering has been coupled with high-throughput single-cell sequencing to enable lineage tracing technologies that can track the relationships between a large number of cells over many generations (Figure 3.1a, [127, 108]). Generally, these approaches begin with cells engineered with one or more recording "target sites" where Cas9-induced heritable insertions or deletions ("indels") accumulate and are subsequently read out by sequencing. A phylogenetic reconstruction algorithm is then used to infer cellular relationships from the pattern of indels. These technologies have enabled the unprecedented exploration of zebrafish [129, 147, 165, 180] and mouse development [103, 28].

However, the scale and complexity of the data produced by these methods are rapidly becoming a bottleneck for the accurate inference of phylogenies. Specifically, traditional algorithms for reconstructing phylogenies (such as Neighbor-Joining [153] or Camin-Sokal [21]) have not been fully assessed with respect to lineage tracing data and may not be well suited for analyzing large-scale lineage tracing experiments for several reasons. First, traditional algorithms were developed for the cases of few samples (in this case cells) and thus scalability is a major limitation (Additional file 1: Fig S1). Second, these algorithms are not well suited to handle the amount of missing data that is typical of lineage tracing experiments, which can be "heritable" (resulting from either large Cas9-induced resections that remove target sites or transcriptional silencing) or "stochastic" (caused by incomplete capture of target sites). Third, these approaches do not explicitly take into consideration the design principles of lineage-tracers, such as the irreversibility of mutations or the unedited state of the founder cell. Together, these reasons necessitate the development of an adaptable approach for reconstructing single-cell phylogenies and an appropriate benchmarking resource that can aid in the development of such algorithms.

Ideally, an algorithm for phylogeny inference from lineage tracing data would be robust to experimental parameters (e.g. rate of mutagenesis, the number of Cas9 target sites), scalable to at least tens of thousands of cells, and resilient to missing data. In this study, we introduce Cassiopeia: a novel suite of three algorithms specifically aimed at reconstructing large phylogenies from lineage tracing experiments with special consideration for the Cas9-mutagenesis process and missing data. Cassiopeia's framework consists of three modules: (1) a greedy algorithm (Cassiopeia-Greedy), which attempts to construct trees efficiently based on mutations that likely occurred earliest in the experiment; (2) a near-optimal algorithm that attempts to find the most parsimonious solution using a Steiner-Tree approach (Cassiopeia-ILP); and (3) a hybrid algorithm (Cassiopeia-Hybrid) that blends the scalability of the greedy algorithm and the exactness of the Steiner-Tree approach to support massive single-cell lineage tracing phylogeny reconstruction. To demonstrate the utility of these algorithms, we compare Cassiopeia to existing methods using two resources: first, we benchmark the algorithms using a custom simulation framework for generating synthetic lineage tracing datasets across varying experimental parameters. Second, enabled by a customizable target-site processing pipeline (Figure 3.1b), we assess these algorithms using a new reference *in*

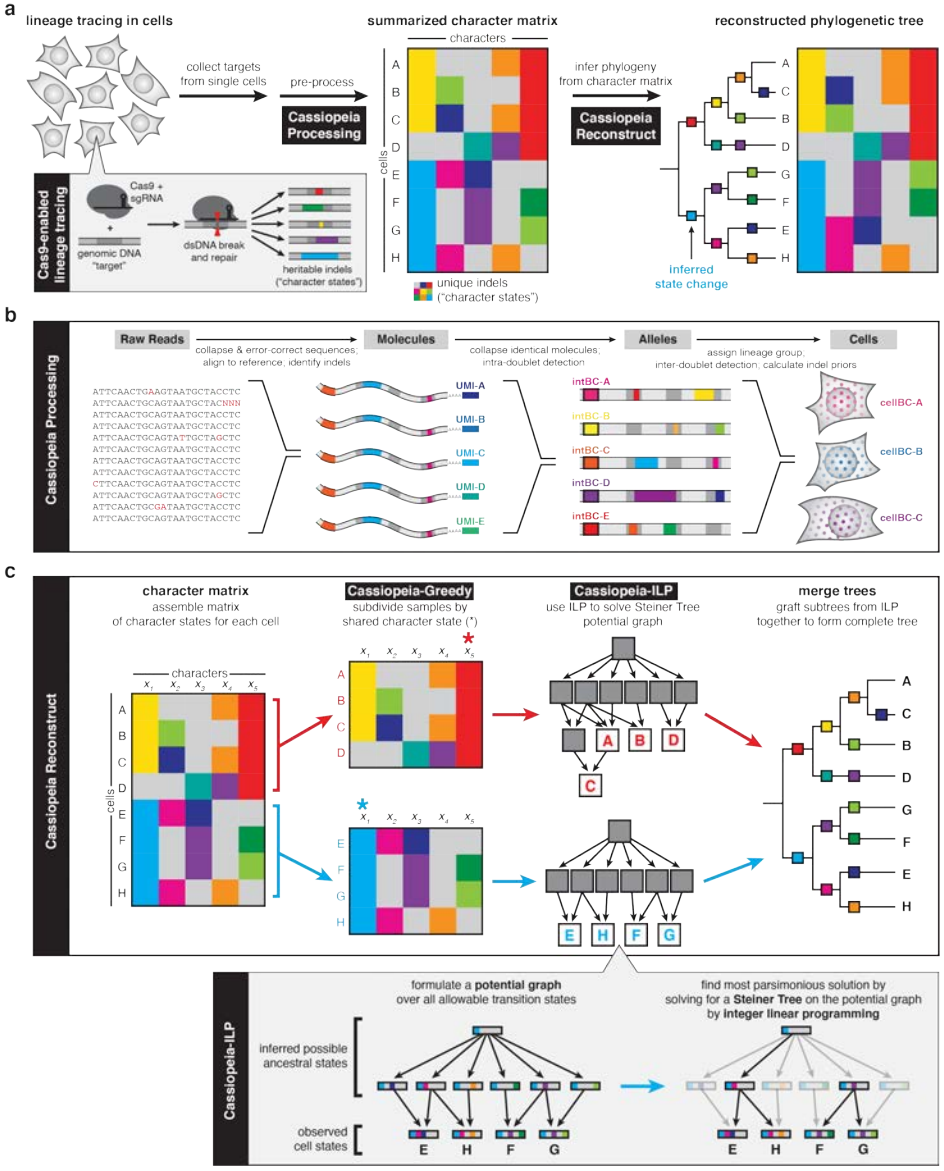


Figure 3.1: **A generalized approach to lineage tracing & lineage reconstruction.** (a) The workflow of a lineage tracing experiment. First, cells are engineered with lineage tracing machinery, namely Cas9 that cuts a genomic target site; the target site accrues heritable, Cas9-induced indels ("character states"). Next, the indels are read off from single cells (e.g. by scRNA-seq) and summarized in a "character matrix", where rows represent cells, columns represent individual target sites (or "characters") and values represent the observed indel (or "character state"). Finally, the character matrix is used to infer phylogenies by one of various methods. (b) The Cassiopeia processing pipeline. The Cassiopeia software includes modules for the processing of target-site sequencing data: first, identical reads are collapsed together and similar reads are error-corrected; second, these reads are locally aligned to a reference sequence and indels are called from this alignment; third, unique molecules are aggregated per cell and intra-doublets are called from this information; finally, the cell population is segmented into clones (or lineage groups) and inter-doublets are called. This clones are then passed to Cassiopeia's reconstruction module for phylogenetic inference. (c) The Cassiopeia reconstruction framework. Cassiopeia takes as input a "character matrix," summarizing the mutations seen at heritable target sites across cells. Cassiopeia-Hybrid merges two novel algorithms: the "greedy" (Cassiopeia-Greedy) and "Steiner-Tree / Integer Linear Programming" (Cassiopeia-ILP) approaches. First, the greedy phase identifies mutations that likely occurred early in the lineage and splits cells recursively into groups based on the presence or absence of these mutations. Next, when these groups reach a predefined threshold, we infer Steiner-Trees, finding the tree of minimum weight connecting all observed cell states across all possible evolutionary histories in a "potential graph", using Integer Linear Programming (ILP). Finally, these trees (corresponding to the maximum parsimony solutions for each group) are returned and merged into a complete phylogeny.

vitro lineage tracing dataset consisting of 34,557 cells over 11 clonal populations. Finally, we use Cassiopeia to explore experimental design principles that could improve the next generation of Cas9-enabled lineage tracing systems.

3.2.4 Results

Cassiopeia: A Scalable Framework for Single-Cell Lineage Tracing Phylogeny Inference

Typically, phylogenetic trees are constructed by attempting to optimize a predefined objective over characters (i.e. target sites) and their states (i.e. indels) [196]. Distance-based methods (such as Neighbor-Joining [153, 62, 134] or phylogenetic least-squares [25, 57]) aim to infer a weighted tree that best approximates the dissimilarity between nodes (i.e., the number of characters differentiating two cells should be similar to their distance in the tree). Alternatively, character-based methods aim to infer a tree of maximum parsimony [56, 49]. Conventionally, in this approach the returned object is a rooted tree (consisting of observed "leaves" and unobserved "ancestral" internal nodes) in which all nodes are associated with a set of character states such that the overall number of changes in character states (between ancestor and child nodes) is minimized. Finally, a third class of methods closely related to character-based ones takes a probabilistic approach over the characters using maximum likelihood [54, 143] or posterior probability [92] as an objective.

We chose to focus our attention on maximum parsimony-based methods due to the early success of applying these methods to lineage tracing data [147, 129] as well as the wealth of theory and applications of these approaches in domains outside of lineage tracing [115]. Our framework, Cassiopeia, consists of three algorithms for solving phylogenies. In smaller datasets, we propose the use of a Steiner-Tree approach (Cassiopeia-ILP) [201] for finding the maximum parsimony tree over observed cells. Steiner Trees have been extensively used as a way of abstracting network connectivity problems in various settings, such as routing in circuit design [72], and have previously been proposed as a general approach for finding maximum parsimony phylogenies [120, 184]. To adapt Steiner-Trees to single-cell lineage tracing, we devised a method for inferring a large underlying "Potential Graph" where vertices represent unique cells (both observed and plausible ancestors) and edges represent possible evolutionary paths between cells. Importantly, we tailor this inference specifically to single-cell lineage tracing assays: we model the irreversibility of Cas9 mutations and impute missing data using an exhaustive approach, considering all possible indels in the respective target sites (see methods). After formulating the Potential Graph, we use Integer Linear Programming (ILP) as a technique for finding near-optimal solutions to the Steiner Tree problem. Because of the NP-Hard complexity of Steiner Trees and the difficult approximation of the Potential Graph (whose effect on solution stability is assessed in Additional file 1: Fig S2), the main limitation of this approach is that it cannot in practice scale to very

large numbers of cells.

To enable Cassiopeia to scale to tens of thousands of cells, we apply a heuristic-based greedy algorithm (Cassiopeia-Greedy) to group cells using mutations that likely occurred early in the lineage experiment. Our heuristic is inspired by the idea of "perfect phylogeny" [171, 110] - a phylogenetic regime in which every mutation (here, Cas9- derived indels) are unique and occurred at most once. For the case of binary characters (i.e., mutated yes/ no without accounting for the specific indel), there exists an efficient algorithm [74] for deciding whether a perfect phylogeny exists and if so, to also reconstruct this phylogeny. However, two facets of the lineage tracing problem complicate the deduction of whether or not a perfect phylogeny exists: first, the "multi-state" nature of characters (i.e. each character is not binary, but rather can take on several different states; which makes the problem NP-Hard) [18, 166]; and second, the existence of missing data [75]. To address these issues, we first take a theoretical approach and prove that since the founder cell (root of the phylogeny) is unedited (i.e. includes only uncut target sites) and that the mutational process is irreversible (i.e. edited sites cannot be recut by Cas9), we are able to reduce the multi-state instance to a binary one so that it can be resolved using a perfect phylogeny-based greedy algorithm. Though Cassiopeia-Greedy does not require a perfect phylogeny, we also prove that if one does exist in the dataset, our proposed algorithm is guaranteed to find it (Theorem 1). Secondly, Cassiopeia-Greedy takes a data-driven approach to handle cells with missing data (see Methods). Unlike Cassiopeia-ILP, Cassiopeia-Greedy is not by design robust to parallel evolution (i.e. "homoplasy", where a given state independently arises more than once in a phylogeny in different parts of the tree). However, we demonstrate theoretically that in expectation, mutations observed in more cells are more likely to have occurred fewer times in the experiment for sufficiently small, but realistic, ranges of mutation rates (see Methods; Additional file 1: Fig S3), thus supporting the heuristic. Moreover, using simulations, we quantify the precision of this greedy heuristic for varying numbers of states and mutation rates, finding in general these splits are precise (especially in these regimes of realistic parameterizations; see Methods and Additional file 1: Fig S4). Below, we further discuss simulation-based analyses that illustrate Cassiopeia-Greedy's effectiveness with varying amounts of parallel evolution (Additional file 1: Fig S5).

While Cassiopeia-ILP and Cassiopeia-Greedy are suitable strategies depending on the dataset, we can combine these two methods into a hybrid approach (Cassiopeia-Hybrid) that covers a far broader scale of dataset sizes (Figure 3.1c). In this use case, Cassiopeia-Hybrid balances the simplicity and scalability of the multi-state greedy algorithm with the exactness and generality of the Steiner-Tree approach. The method begins by splitting the cells into several major clades using Cassiopeia-Greedy and then separately reconstructing phylogenies for each clade with Cassiopeia-ILP. This parallel approach on reasonably sized sub-problems ($\sim$ 300 cells in each clade) ensures practical run-times on large numbers of cells (Additional file 1: Fig S1). After solving all sub-problems with the Steiner Tree approach, we merge all clades together to form a complete phylogeny (Figure 3.1c).

A Simulation Engine Enables a Comprehensive Benchmark of Lineage Reconstruction Algorithms

To provide a comprehensive benchmark for phylogeny reconstruction, we developed a framework for simulating lineage tracing experiments across a range of experimental parameters. In particular, the simulated lineages can vary in the number of characters (e.g. Cas9 target sites), the number of states (e.g. possible Cas9-induced indels), the probability distribution over these states, the mutation rate per character, the number of cell generations, and the amount of missing data. We started by estimating plausible "default" values for each simulation parameter using experimental data (discussed below and indicated in Figure 3.2). In each simulation run, we varied one of the parameters while keeping the rest fixed to their default value. The probability of mutating to each state was found by interpolating the empirical distribution of indel outcomes (Additional file 1: Fig S6, see Methods). Each parameter combination was tested using a maximum of 50 replicates or until convergence, each time sampling a set of 400 cells from the total 2^D cells (where D is the depth of the simulated tree).

We compare the performance of our Cassiopeia algorithms (Cassiopeia-ILP, -Greedy, and -Hybrid) as well as an alternative maximum-parsimony algorithm, Camin-Sokal (previously used in lineage tracing applications [129, 147]), and the distance-based algorithm Neighbor-Joining. We assess performance using a combinatoric metric, "Triplets Correct" (Additional file 1: Fig S7, see Methods), which compares the proportion of cell triplets that are ordered correctly in the tree. Importantly, this statistic is a weighted-average of the triplets, stratified by the depth of the triplet (measured by the distance from the root to the Latest Common Ancestor (LCA); see Methods). As opposed to other tree comparison metrics, such as Robinson-Foulds [151], we reason that combinatoric metrics [39] more explicitly address the needs of fundamental downstream analyses, namely determining evolutionary relationships between cells (though the triplets correct statistic largely agrees with distance-based metrics; see Additional file 1: Fig S7b).

Overall, our simulations demonstrate the strong performance and efficiency of Cassiopeia. Specifically, we see that the Cassiopeia suite of algorithms consistently finds more accurate trees as compared to both Camin-Sokal and Neighbor-Joining (Figure 3.2a-e, Additional file 1: Fig S8a-e). Furthermore, not only are trees produced with Cassiopeia more accurate than existing methods, but also more parsimonious across all parameter ranges - serving as an indication that the trees reach a more optimal objective solution (Additional file 1: Fig S9). Importantly, we observe that Cassiopeia-Hybrid and -Greedy are more effective than Neighbor-Joining in moderately large sample regimes (Additional file 1: Fig S10). Notably, Cassiopeia-Greedy and -Hybrid both scale to especially large regimes (of up to 50,000 cells, a scale that includes the approximate upper limit of most current single-cell sequencing

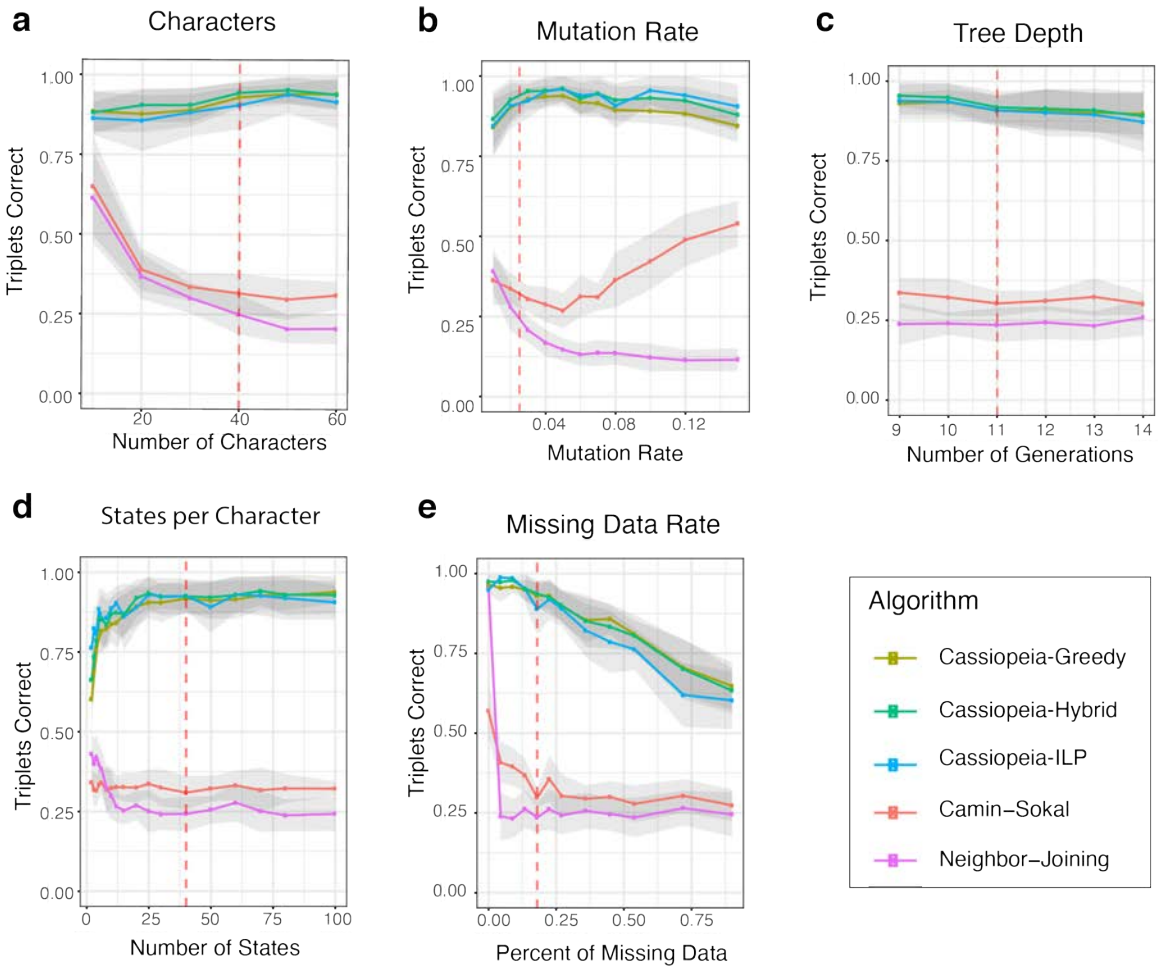


Figure 3.2: **Cassiopeia algorithms outperform other phylogenetic reconstruction methods on simulated lineages.** Accuracy is compared between five algorithms (Cassiopeia-Greedy, -ILP, and -Hybrid algorithms as well as Neighbor-Joining and Camin-Sokal) on 400 cells. Phylogeny reconstruction accuracy is assessed with the Triplets correct statistic across several experimental regimes: (a) the number of characters; (b) mutation rate (i.e. Cas9 cutting rate); (c) depth of the tree (or length of the experiment); (d), the number of states per character (i.e. number of possible indel outcomes); and (e) the dropout rate. Dashed lines represent the default value for each stress test. Between 10 and 50 replicate trees were reconstructed, depending on the stability of triplets correct statistic and overall runtime. Standard error over replicates is represented by shaded area.

experiments) without substantial compromise in accuracy (Additional file 1: Fig S11). In contrast, Camin-Sokal and Cassiopeia-ILP could not scale to such input sizes (Additional file 1: Fig S1). Finally, we observe that under a bootstrapping analysis, Cassiopeia's modules are robust to lineage-tracing data (Additional file 1: Fig S12a,b) as compared to Neighbor-Joining for reference (Additional file 1: Fig S12c, though Neighbor-Joining's stability may be improved with more sophisticated distance functions and feature selection).

These simulations additionally grant insight into critical design parameters for lineage recording technology. Firstly, we observe that the "information capacity" (i.e. number of characters and possible indels, or states) of a recorder confers an increase in accuracy for Cassiopeia's modules but not necessarily Camin-Sokal and Neighbor-Joining (though they do perform moderately well in low information capacity simulations; Figures 3.2a,d). This is likely because the greater size of the search space negatively affects the performance of these two algorithms (in other contexts referred to as the "curse of dimensionality" [177]). In addition to the information capacity, we find that indel distributions that tend towards a uniform distribution (and thus higher entropy) allow for more accurate reconstructions especially when the number of states is small or the number of samples is large (Additional file 1: Fig S13). Unsurprisingly, the proportion of missing data causes a precipitous decrease in performance (Figure 3.2e). Furthermore, in longer experiments where the observed cell population is sampled from a larger pool of cells, we find that the problem tends to become more difficult (Figure 3.2c).

Furthermore, these results grant further insight into how Cassiopeia-Greedy is affected in regimes where parallel evolution is likely: such as in low information capacity regimes (e.g. where the number of possible indels is less than 10, Figure 3.2d), or with high mutation rates (Figure 3.2b). In both of these regimes, the proportion of parallel evolution mutations of all mutations increases (Additional file 1: Fig S14). While Cassiopeia-ILP outperforms Cassiopeia-Greedy in these simulations, highlighting its utility to solve small, yet complex, datasets, we further explored Cassiopeia-Greedy's effectiveness in these regimes. To strengthen our previous theoretical results suggesting that indels observed in more cells are more likely to occur fewer times and earlier in the phylogeny (Additional file 1: Fig S3), we explored how parallel evolution affects Cassiopeia-Greedy empirically with simulation. Specifically, we simulated trees with varying numbers of parallel evolution events at various depths and find overall that while performance decreases with the number of these events, the closer these events occur to the leaves, the smaller the effect (Figure 3.11). Furthermore, we find that under the "default" simulation parameters (as determined by the experimental data; Additional file 1: Fig S6 and 3.3), Cassiopeia-Greedy consistently makes accurate choices of the first indel event by which cells are divided into clades (Additional file 1: Fig S4b). Of course in regimes where possible, Cassiopeia-ILP outperforms Cassiopeia-Greedy when there are few states (i.e. fewer than 10; Figure 3.2d) or high mutation rates (i.e. greater than 10%; Figure 3.2b).

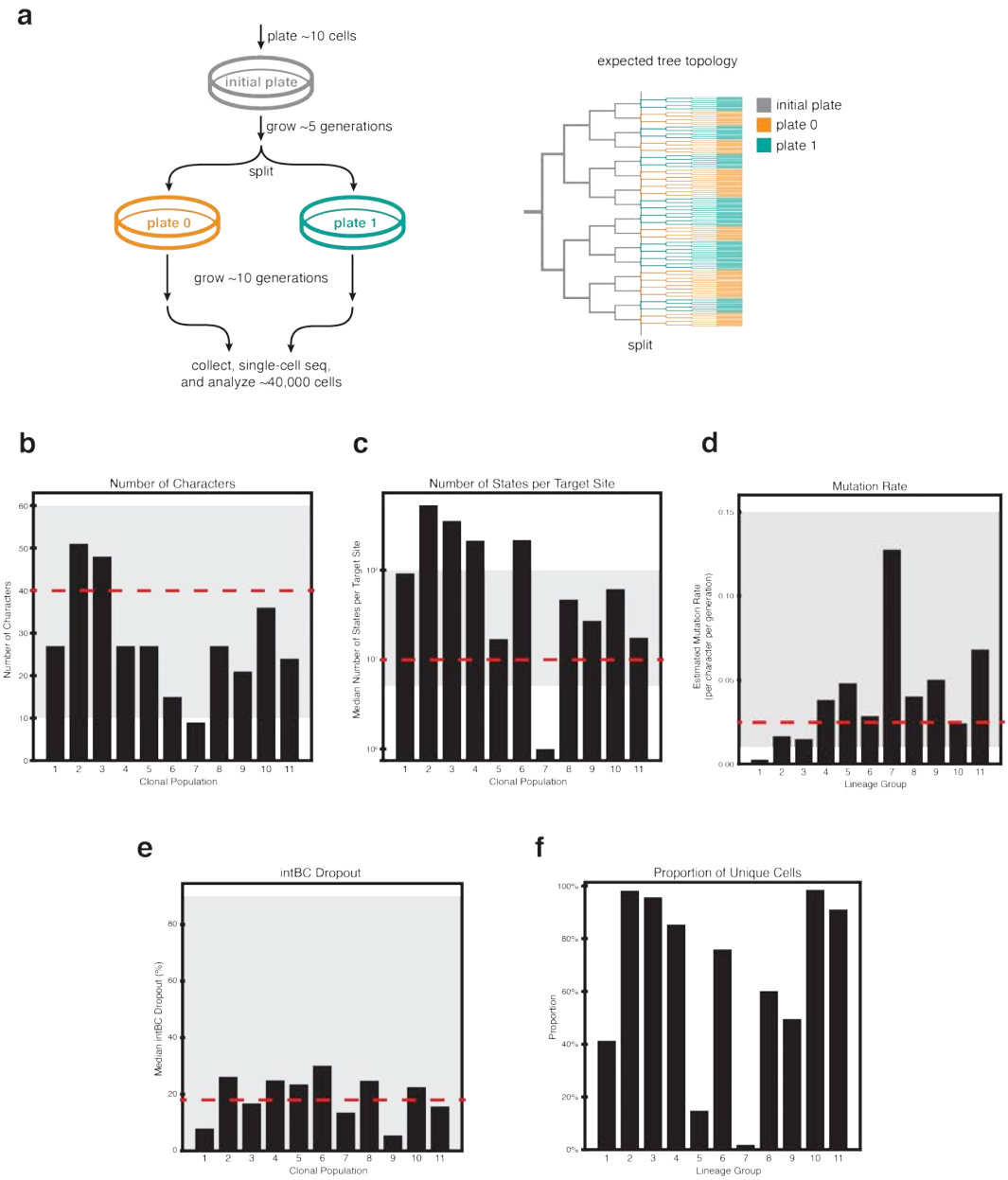


Figure 3.3: **An *in vitro* Reference Experiment.** (a) A reference lineage tracing dataset was generated using the technology proposed in Chan et al. [28] to human cells cultured *in vitro* for ~ 15 generations. A total of $34,557$ cells were analyzed after filtering and error correction. Only the initial split (into two plates) is shown. Analysis of the subsequent split (into four plates) is provided in Additional file 1: Fig S22. (b-f) Summary of relevant lineage tracing parameters for each clonal population in the experiment: (b) the number of characters per clone; (c) number of states per target site; (d) the estimated mutation rate per target site; (e) median dropout per target site; and (f) the proportion of uniquely marked cells. Gray shading denotes parameter regimes tested in simulations and red-dashed lines denote the default values for each synthetic benchmarks.

Practically, the issue of parallel evolution can be addressed to some extent by incorporating state priors (i.e. probabilities of Cas9-induced indel formation). Ideally, Cassiopeia-Greedy would use these priors to select mutations that are low-probability, but observed at high frequency. Theoretically, this would be advantageous as low-probability indels are expected to occur fewer times in the tree (3.1); thus if they appear at high frequency at the leaves, it is especially likely that these occurred earlier in the phylogeny. Furthermore, our precision-analysis indicates that Cassiopeia-Greedy's decisions are especially precise if it chooses an indel with a low prior (Additional file 1: Fig S4). To incorporate these priors in practice, we selected a link function (i.e. one translating observed frequency and prior probability to priority) that maximized performance for Cassiopeia-Greedy (Additional file 1: Fig S15; see Methods). After finding an effective approach for integrating prior probabilities, we performed the same benchmarks, and found that in cases of likely parallel evolution the priors confer an increase in accuracy (e.g. with high mutation rates; Additional file 1: Fig S16), especially in larger regimes (Additional file 1: Fig S11).

Here, we have introduced a flexible simulator that is capable of fitting real data, and thus can be used for future benchmarking of algorithms. Using this simulator and a wide range of parameters, we have demonstrated that Cassiopeia performs substantially better than traditional methods. Furthermore, these simulations grant insight into how Cassiopeia's performance is modulated by various experimental parameters, suggesting design principles that can be optimized to bolster reconstruction accuracy. Specifically, these simulations suggest that these technologies would benefit most from increases in information capacity, via more target sites or more diverse indel outcomes, and mutation rates tuned appropriately as to ensure low rates of parallel evolution. We anticipate that this resource will continue to be of use in exploring design principles of recorders and the effectiveness of novel algorithms.

An *In Vitro* Reference Experiment Allows Evaluation of Approaches on Empirical Data

Existing experimental lineage tracing datasets lack a defined ground truth to test against, thus making it difficult to assess phylogenetic accuracy in practice. To address this, we performed an *in vitro* experiment tracking the clonal expansion of human cells (A549 lung adenocarcinoma cell line) engineered with a previously described lineage tracing technology [28]. Here, we tracked the growth of 11 clones (each with non-overlapping target site sets for deconvolving clonal populations) over the course of 21 days (approx. 15 generations on average), randomly splitting the pool of cells into two plates every 7 days (Figure 3.3a; see Methods). At the end of the experiment, we sampled approximately $10,000$ cells from each of the four final plates. This randomized plate splitting strategy establishes a course-grained ground truth of how cells are related to each other. Here, cells within the same plate can be arbitrarily distant in their lineage, however there is only a lower bound on lineage dissimilarity between cells in different plates (since they are by definition at least separated by

the number of mutations that have occurred since the last split). Thus, overall, on average we expect cells within the same plate to be closer to each other in the phylogeny than cells from different plates. However, due to the considerations discussed above, we also expect to see some cells more closely related across plates than within (Figure 3.3a, right), and indels relating these cells across plates are likely to have occurred before the split.

Our lineage recorder is based on a constitutively expressed target sequence consisting of three evenly spaced cut sites (each cut site corresponding to a character) and a unique integration barcode ("intBC") which we use to distinguish between target sites and thus more accurately relate character states across cells (Figure 3.1b). The target sites are randomly integrated into the genomes of founder cells at high copy number (on average 10 targets per cell or a total of 30 independently evolving characters; Figure 3.3b, S18c). We built upon the processing pipeline in our previous work [28] to obtain confident indel information from scRNA-seq reads (Figure 3.1b, Additional file 1: Fig S18, & Additional file 1: Fig S17, see Methods for pre-processing procedures and guidelines, especially section "Guidelines for Final Quality Control"). In addition, we have added modules for the detection of cell doublets using the sets of intBCs in each clone and the indels detected within cells, and have determined an effective detection strategy using simulations (see Methods, Additional file 1: Fig S19). Importantly, though not directly applicable here, this doublet detection can be supplemented by other approaches when transcriptional data [126, 185] or multiplexing barcodes [167] are available. Additionally, we rely on a data-driven approach for estimating the likelihoods of each indel (see Methods; Additional file 1: Fig S20) because other approaches for indel-likelihood prediction [109, 34, 6] may be biased by cell-type or cell-state.

After quality control, error-correction, and filtering we proceeded with analyzing a total of $34,557$ cells across 11 clones. This diverse set of clonal populations represent various levels of indel diversity (i.e. number of possible states, Figure 3.3c), size of intBC sets (i.e. number of characters, Figure 3.3b and Additional file 1: Fig S18c), character mutation rates (Figure 3.3d, see Methods), and proportion of missing data (Figure 3.3e, see Methods). Most importantly, this dataset represents a significant improvement in lineage tracing experiments: it is the longest and most complex dataset to date in which the large majority of cells, over the entire cell population, have unique mutation states (71% after all quality-control and filtering; percentages of unique cells per clone is presented in Figure 3.3f), indicating a rich character state complexity for tree building.

We next reconstructed trees for each clone (excluding two which were removed through quality-control filters; see Methods) with our suite of algorithms, as well as Neighbor-Joining and Camin-Sokal (when computationally feasible). For both Cassiopeia-Greedy and Cassiopeia-Hybrid methods, we also compared tree reconstruction accuracy with or without prior probabilities. The tree for Clone 3, consisting of 7,289 cells, along with its character matrix and first split annotations (i.e. whether cells were initially split into plate 0 or plate 1, denoted as the plate ID), is presented in Figure 3.4. Interestingly, we find that certain

indels indeed span the different plates, thus suggesting that Cassiopeia-Greedy chooses as early splits indels which likely occurred prior to the first separation of plates (though this could also be due to parallel events that occurred independently at each plate). Moreover, the character matrix and the nested dissection of the tree illustrate the abundant lineage information encoded in this clone (96% of the 7,289 observed cells have unique mutation states) which allows Cassiopeia to infer a relatively deep tree (Figure 3.4d). Despite this complexity, Cassiopeia infers a tree that largely agrees with the observed mutations: cells close to one another in the tree tend to have similar mutations (Figure 3.4e).

By keeping track of which plate each cell came from we are able to evaluate how well the distances in a computationally-reconstructed tree reflect the distances in the experimental tree. Thus, we test the reconstruction ability of an algorithm using two metrics for measuring the association between plate ID and substructure: "Meta Purity" and "Mean Majority Vote" (see Methods). Both are predicated on the assumption that, just as in the real experiment, as one descends the reconstructed tree, one would expect to find cells more closely related to one another. In this sense, we utilize these two metrics for testing homogeneous cell labels below a certain internal node in a tree, which we refer to as a "clade".

We use these statistics to evaluate reconstruction accuracy for Clone 3 with respect to the first split labels (i.e. plate 0 or 1, Figure 3.5). In doing so, we find that Cassiopeia-Greedy and -Hybrid consistently outperform Neighbor-Joining. We find overall consistent results for the remainder of clones reconstructed (Additional file 1: Fig S21, and additionally when considering the subsequent split into four plates - Additional file 1: Fig S21), although Cassiopeia's modules have the greatest advantage in larger reconstructions. Specifically, Camin-Sokal and Neighbor-Joining perform similarly to Cassiopeia's modules on clones with few cells (e.g. Clone 11) or with low cell diversity (e.g. Clone 5, where target sites are "exhausted", possibly due to too-fast cutting, (Figure 3.3f, Additional file 1: Fig S23). Both cases indicate that in smaller and less complex clones traditional algorithms may be sufficient for reconstruction. Additionally, many of the issues described previously - parallel evolution, missing data, and information content - contribute to inferential errors in this empirical dataset (for example, Additional file 1: Fig S24).

Overall, we anticipate that this *in vitro* dataset will serve as a valuable empirical benchmark for future algorithm development. Specifically, we have demonstrated how this dataset can be used to evaluate the accuracy of inferred phylogenies and illustrate that Cassiopeia consistently outperforms Neighbor-Joining for the purposes of reconstructing trees from single-cell lineage tracing technologies. Moreover, we demonstrate Cassiopeia's scalability for reconstructing trees that are beyond the abilities of other maximum parsimony-based methods like Camin-Sokal as they currently have been implemented.

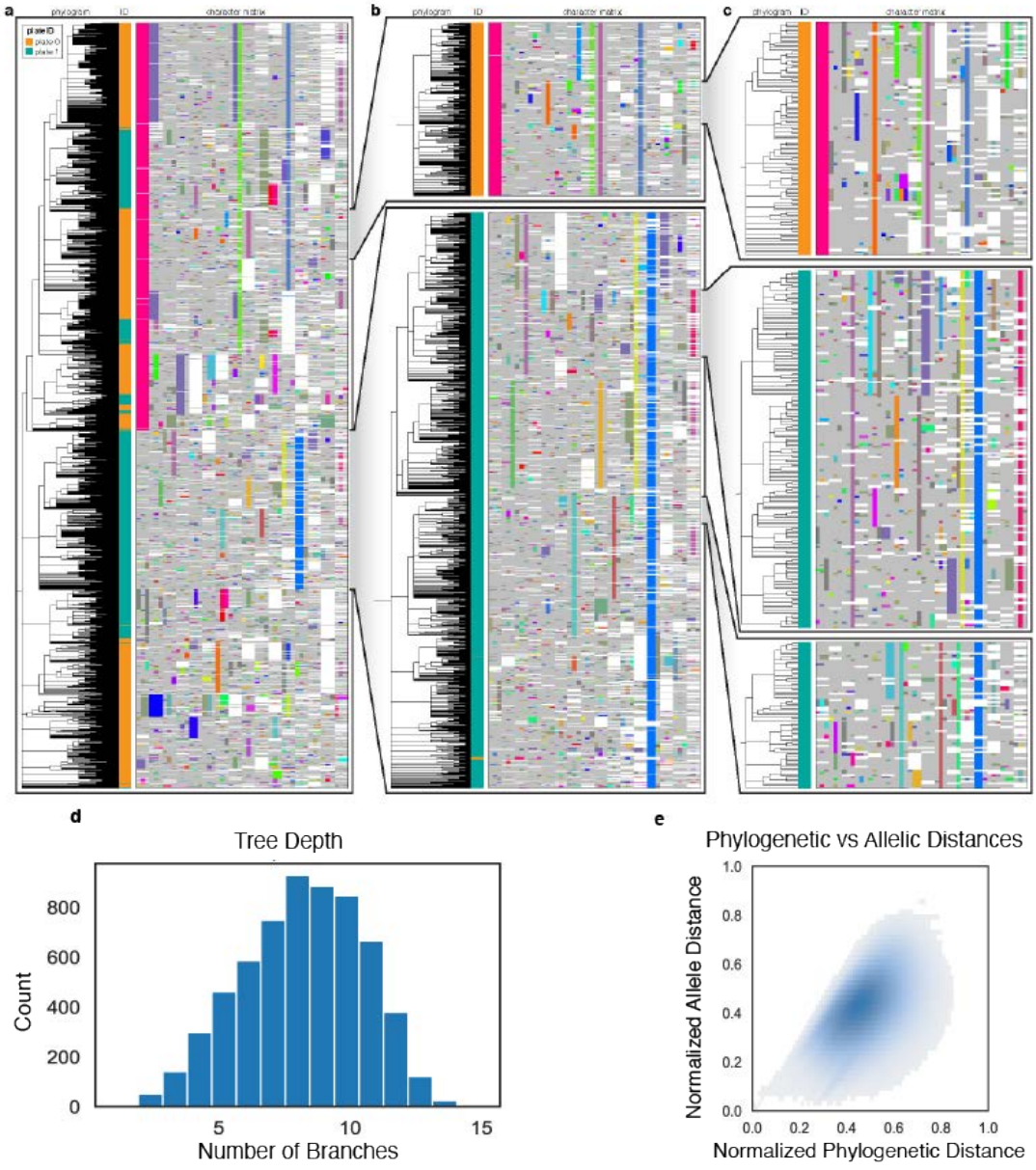


Figure 3.4: **Cassiopeia can reconstruct high-resolution phylogenetic trees from empirical lineage tracing data.** The full phylogenetic tree for Clone 3 (a), consisting of 7,289 cells, was reconstructed using Cassiopeia-Hybrid (with priors), and is displayed. The phylogram represents cell-cell relationships, and each cell is colored by sample ID at the first split (plate 0 or 1). The character matrix is displayed with each unique character state (or "indel") represented by distinct colors. (Light gray represents uncut sites; white represents missing values.) Of these 7,289 cells, 96% were uniquely tagged by their character states. (b-c) Nested, expanded views of the phylogram and character matrices. As expected, Cassiopeia correctly relates cells with similar character states, and closely related cells are found within the same culture plate. (d) A histogram of the tree-depth of each leaf from the root (mean = 8.22, max = 15). (e) Concordance between normalized allelic distance and normalized phylogenetic distance (see Methods; Pearson's correlation = 0.53).

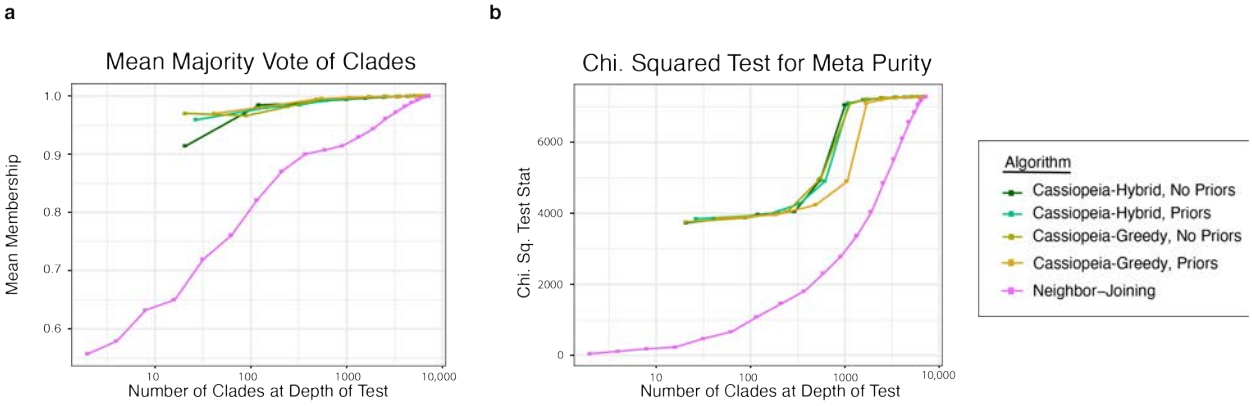


Figure 3.5: **Cassiopeia builds highly accurate trees from large empirical datasets.** The consistency between tree reconstructions are evaluated with respect to the first split. The Mean Majority Vote (a) and the Meta Purity test (b) were used for Cassiopeia-Hybrid and -Greedy (both with or without priors) and Neighbor-Joining. The statistics are plotted as a function of the number of clades at the depth of the test (i.e. the number of clades created by a horizontal cut at a given depth). All Cassiopeia approaches consistently outperform Neighbor-Joining by both metrics.

Generalizing Cassiopeia to Alternative & Future Technologies

While previous single-cell lineage tracing applications have proposed methods for phylogenetic reconstruction, they have been custom-tailored to the experimental system, requiring one to filter out common indels [165] or provide indel likelihoods [28]. We thus investigated how well Cassiopeia generalizes to other technologies with reconstructions of data generated with the GESTALT technology applied to zebrafish development [129, 147] (Figure 3.6a, Additional file 1: Fig S25). Comparing Cassiopeia's algorithms to Neighbor-Joining and Camin-Sokal (as applied in these previous studies [129, 147]), we find that Cassiopeia-ILP consistently finds the most parsimonious solution. Furthermore, the Mean Majority Vote statistic also indicates that there is strong tissue-type enrichment as a function of tree depth, agreeing with Camin-Sokal's reconstruction which was used in the original study [147] (Figure 3.6b). Together, these results clearly demonstrates Cassiopeia's effectiveness for existing alternative lineage tracing technologies.

After establishing Cassiopeia's generalizability, we turned to investigating plausible next-generation lineage tracers. Recently, base-editing systems (Figure 3.6c) have been proposed to precisely edit $A > G$ [65], $C > T$ [116, 66] or possibly $C > N$ (N being any base as in [82]). The promise of base-editing lineage recorders is three-fold: first, a base editor would increase the number of editable sites (as compared to the ones that rely on Cas9-induced double-strand breaks [28, 129, 165]) although at the expense of number of states (at best 4, corresponding to A, C, T, and G). Second, a base-editing system would theoretically result in less dropout, since target site resection via Cas9-induced double-strand breaks is far less likely [116]. Third, it is hypothesized that base-editors would be less cytotoxic as it does not

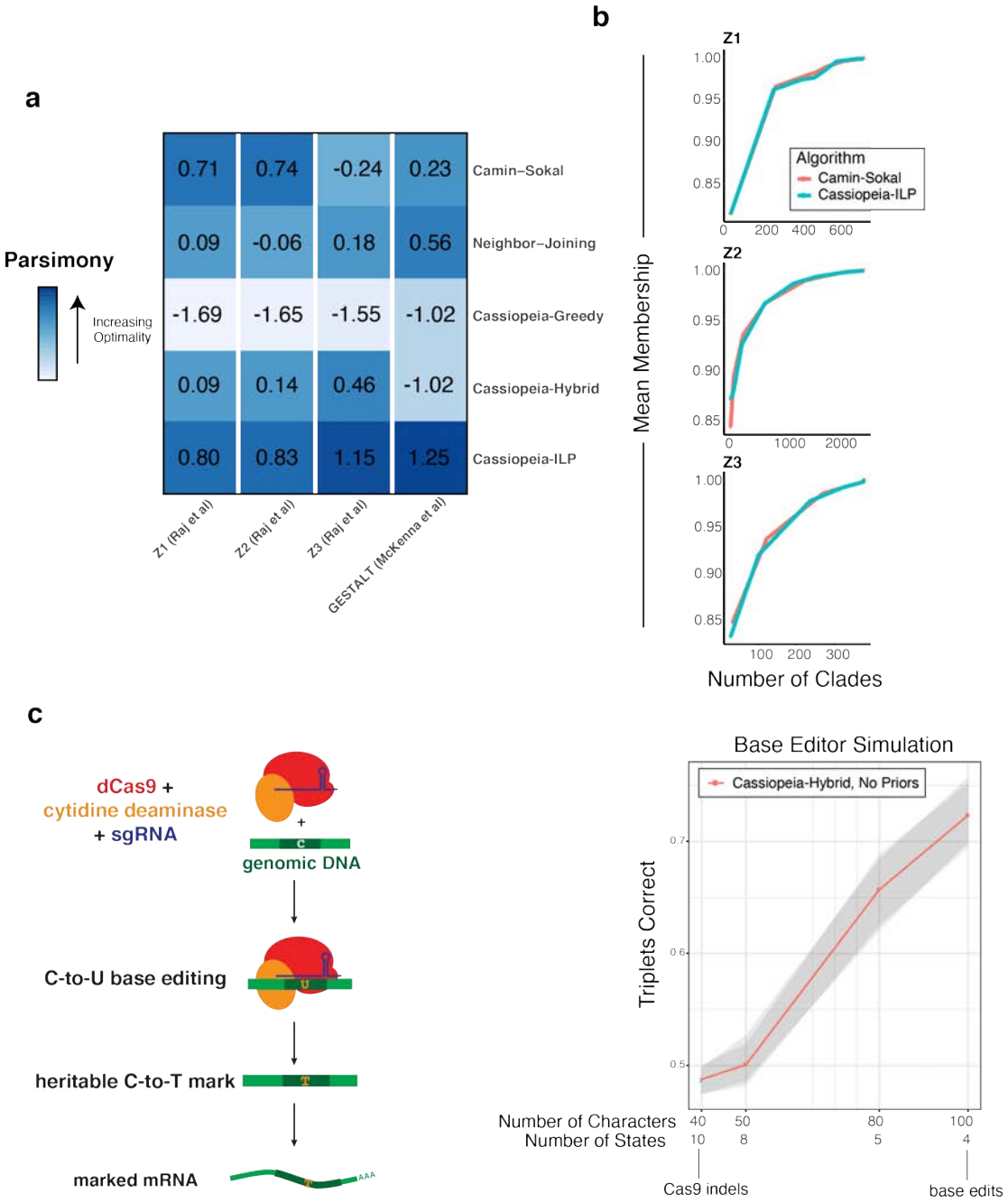


Figure 3.6: **Generalizing Cassiopeia & future design principles of CRISPR-enabled lineage tracers.** (a) Cassiopeia generalizes to alternative lineage tracing methods, as illustrated with the analysis of data from GESTALT technology [129, 147]). In a comparison of parsimony across Camin-Sokal, Neighbor-Joining, and Cassiopeia's methods, the Steiner-Tree approach consistently finds more parsimonious (i.e more optimal) solutions. Z-scores for each dataset are annotated over each tile. (b) Biological integrity of trees for each Zebrafish from Raj et al. [147], inferred with Cassiopeia-ILP, was assessed using the mean membership statistic (Methods) with respect to tissue type annotations from the original study. (c) Exploring information capacity of recorders with base-editors. A theoretical base-editor was simulated for 400 cells and reconstructions with Cassiopeia-Hybrid, with and without priors. We compared the accuracy of the reconstructions to the simulated tree using the triplets correct statistic. We describe the performance of Cassiopeia-Hybrid as the number of characters was increased (and consequently number of states was decreased.)

depend on inducing double strand breaks on DNA (although this relies on effective strategies for limiting off-target base-editing of DNA and RNA [194]). To evaluate the application of base editors for lineage tracing, we tested the performance of Cassiopeia in high-character, low-state regimes as would be the case in base editing (Figure 3.6c, see Methods). Using simulations with parameters deduced by a recent base editor application [82], we demonstrate that there appears to be an advantage of having more characters than states (Figure 3.6c). Of note, we did not observe any substantial deviation in these simulations from our initial scalability benchmarks in Additional file 1: Fig S1. This suggests that base-editors may be a promising future direction for lineage tracing from a theoretical perspective.

Another potentially promising design consideration concerns the range of character mutation rates and their variability across different target sites – a parameter that can be precisely engineered [101]. In this design, one would expect the variability to help distinguish between early and late branching points and consequently achieve better resolution of the underlying phylogeny [173, 103, 102]. We simulated "Phased Recorders" (Additional file 1: Fig S26) with varying levels of target-site cutting variability and observe that this design allows for better inference when the distributions of mutation probabilities are more dispersed (Additional file 1: Fig S26b). This becomes particularly useful when one can integrate accurate indel priors into Cassiopeia.

Overall, these results serve to illustrate how Cassiopeia and the simulation framework can be used to explore experimental designs. While there inevitably will be challenges in new implementations, these analyses demonstrate theoretically how design parameters can be optimized for downstream tree inference. In this way, the combination of our algorithms and simulations enables others to explore not only new algorithmic approaches to phylogenetic reconstruction but also new experimental approaches for recording lineage information.

3.2.5 Conclusions

In this study, we have presented three resources supporting future single-cell lineage tracing technology development and applications. Firstly, we described Cassiopeia, a scalable and accurate maximum parsimony framework for inferring high-resolution phylogenies in single-cell lineage tracing experiments. Next, we introduced a simulation approach for benchmarking reconstruction methods and investigating novel experimental designs. Finally, we generated the largest and most diverse empirical lineage tracing experiment to date, which we present as a reference for the systematic evaluation of phylogeny inference on real lineage tracing data. With the combination of these three resources, we have demonstrated the improved scalability and accuracy of Cassiopeia over traditional approaches for single-cell lineage tracing data and have explored design principles for more accurate tracing. To ensure broad use, we have made a complete software package, including the algorithms, simulation framework, and a processing pipeline for raw data, all publicly available

at `www.github.com/YosefLab/Cassiopeia`.

The results highlighted in this manuscript demonstrate the variability in reconstruction accuracy for each of Cassiopeia's modules depending on the parameters. As introduced here, we suggest using Cassiopeia-ILP for small regimes (fewer than 200 cells) especially where there is low information capacity, Cassiopeia-Greedy for extremely large regimes (10,000 cells and larger), and Cassiopeia-Hybrid for intermediate regimes. Ideally, Cassiopeia-Hybrid could be run in all situations and transition appropriately between Cassiopeia-Greedy and -ILP depending on the complexity of the data. While here we use the number of cells as the criterion for transitioning, we anticipate there is a more consistent statistic (e.g. the entropy of a group of cells) for controlling the Cassiopeia-Hybrid transition that will make Cassiopeia more intuitive and effective with handling real data.

Though we illustrate that Cassiopeia provides the computational foundation necessary for future large-scale lineage tracing experiments, there are several opportunities for future improvement. First, the inclusion of prior probabilities increases Cassiopeia's performance only when parallel evolution is likely (e.g. with a high per-character mutation rate or in low character-state regimes). While maximum parsimony methods are attractive due to their non-parametric nature, future studies may build on our work here by developing more powerful approaches for integrating prior mutation rates into maximum likelihood [54, 143] or Bayesian inference [93] frameworks, perhaps relying on recent literature that seeks to predict indel formation probabilities [109, 34, 6]. Future work in this space may also focus on using maximum parsimony solutions to further refine solutions in an effort to resolve branch length as with GAPML [55] or with paired transcriptomic observations [199]. Second, there exists a promising opportunity in developing new approaches for better handling of missing data. Determining a model which explicitly distinguishes between stochastic and heritable missing data may increase tree accuracy. Alternatively, adapting supertree methods (such as the Triple MaxCut algorithm [161]) for lineage tracing data may be an interesting direction as they have been effective for dealing with missing data (but only when this missing data is randomly distributed [190]). Aside from computational approaches for dealing with missing data, it is still unclear how much missing data is due to silencing, Cas9-resections, or stochastic dropout and experiments to elucidate the contributions of each will be helpful to the future design of lineage tracers. Third, while we provide theoretical and empirical evidence for our greedy heuristic, we note that there are opportunities for developing other heuristics - for example, by considering mutations in many characters rather than a single mutation as we do or using a distance-based heuristic.

The ultimate goal of using single-cell lineage tracers to create precise and quantitative cell fate maps will require sampling tens of thousands of cells (or more), possibly tracing over several months, and effectively inferring the resulting phylogenies. While recent studies [155] have highlighted the challenges in creating accurate CRISPR-recorders, our results suggest that with adequate technological components and computational approaches complex bio-

logical phenomena can be dissected with single-cell lineage tracing methods. Specifically, we show that Cassiopeia and the benchmarking resources presented here meet many of these challenges. Not only does Cassiopeia provide a scalable and accurate inference approach, but also our benchmarking resources enable the systematic exploration of more accurate algorithms as well as more robust single-cell lineage tracing technologies. Taken together, this work forms the foundation for future efforts in building detailed cell fate maps in a variety of biological applications.

3.2.6 Methods

3.2.7 *In vitro* lineage tracing experiment

Plasmid design and cloning

The Cas9-mCherry lentivector, pHR-UCOE-SFFV-Cas9-mCherry(to be added to Addgene), was designed for stable, constitutive expression of enzymatically active Cas9, driven by the viral SFFV promoter, insulated with a minimal universal chromatin opening element (minUCOE), and tagged with C-terminal, self-cleaving P2A-mCherry. PCTXX is derived from pMH0001 (Addgene Cat#85969, active Cas9) with the BFP tag exchanged with mCherry. The P2A-mCherry tag was PCR amplified from pHR-SFFV-KRAB-dCas9-P2A-mCherry (Addgene Cat #60954; forward: GAGCAACGGCAGCAGCGGATCCGGAGCTACTAACTTCAG; reverse: ATATCAAGCTTGCATGCCTGCAGGTCGACTTACTACT TACAGCTCGTCCATGC) and inserted using Gibson Assembly (NEB) into SbfI/BamHI-digested pMH0001 (active Cas9). Resulting plasmid was used for lentiviral production as described below.

The Target Site lentivector, PCT48 (available on Addgene), was derived from the reverse lentivector PCT5 (available on Addgene) containing GFP driven by the EF1a promoter. The sequence of the 10X amplicon with most common polyA location is the following:

AATCCAGCTAGCTGTGCAGCNNNNNNNNNNNNNNATTCAACTGCAGTAATGCT
ACCTCGTACTCACGCTTTCCAAGTGCTTGGCGTCGCATCTCGGTCCTTTGTAC
GCCGAAAAATGGCCTGACAACTAAGCTACGGCACGCTGCCATGTTGGGTCATA
ACGATATCTCTGGTTCATCCGTGACCGAACATGTCATGGAGTAGCAGGAGCTA
TTAATTCGCGGAGGACAATGCGGTTCGTAGTCACTGTCTTCCGCAATCGTCCA
TCGCTCCTGCAGGTGGCCTAGAGGGCCCGTTTAAACCCGCTGATCAGCCTCGA
CTGTGCCTTCTAGTTGCCAGCCATCTGTTGTTTGCCCCTCCCCCGTGCCTTCC
TTGACCCTGGAAGGTGCCACTCCCACTGTCCTTTCCTAATAAAAAAAAAAAAA
AAAAAAAAAA

where N denotes our 14bp random integration barcode. PCT5 was digested with SfiI and EcoRI within the 3'UTR of GFP. The Target Site sequence was ordered as a DNA fragment (gBlock, IDT DNA) containing three Cas9 cut-sites and a high diversity, 14-basepair randomer (integration barcode, or intBC). The fragment was PCR amplified with primers containing Gibson assembly arms compatible with SfiI/EcoRI-digested PCT5 (forward: GATGAGCTCTACAAATAATTAATTAAGAATTCGTCACGAATCCAGCTAGCTGT; reverse: GGTTTAAACGGGCCCTCTAGGCCACCTGCAGGAGCGATGG). The amplified Target Site fragment was inserted into the digested PCT5 backbone using Gibson Assembly. The assembled lentivector library was transformed into MegaX competent bacterial cells (Thermo Fisher) and grown in 1L of LB with carbenicillin at 100 μg/mL. Lentivector plasmid was recovered and purified by GigaPrep (Qiagen), and used for high-diversity lentiviral production as described below.

The triple-sgRNA-BFP-PuroR lentivector, PCT61 (available on Addgene), is derived from pBA392 (available on Addgene) as previously described [2, 100] containing three sgRNA cassettes driven by distinct U6 promoters and constitutive BFP and puromycin-resistance markers for selection. Importantly, the three PCT61 sgRNAs are complementary to the three cut-sites in the PCT48 Target Site. To slow the cutting kinetics of the sgRNAs to best match the timescale involved in the *in vitro* lineage tracing experiments [28], the sgRNAs contain precise single-basepair mismatches that decrease their avidity for the cognate cut-sites [68]. The triple-sgRNA lentivector was cloned using four-way Gibson assembly as described in [100]. Resulting plasmid was used for lentiviral production as described below.

Cell culture, DNA transfections, viral preparation, and cell line engineering

A549 cells (human lung adenocarcinoma line, ATCC CCL-185) and HEK293T were maintained in Dulbecco's modified eagle medium (DMEM, Gibco) supplemented with 10% FBS (VWR Life Science Seradigm), 2 mM glutamine, 100 units/mL penicillin, and 100 μg/mL streptomycin. Lentivirus was produced by transfecting HEK293T cells with standard packaging vectors and TransIT-LTI transfection reagent (Mirus) as described in ([2]). Target Site (PCT48) lentiviral preparations were concentrated 10-fold using Lenti-X Concentrator (Takara Bio). Viral preparations were frozen prior to infection. Triple-sgRNA lentiviral preparations were titered and diluted to a concentration to yield approximately 50% infection rate.

To construct the lineage tracing-competent cell line, A549 cells were transduced by serial lentiviral infection with the three lineage tracing components: (1) Cas9, (2) Target Site, and (3) triple-sgRNAs. First, A549 cells were transduced by Cas9 (mCherry) lentivirus and mCherry+ cells were selected to purity by fluorescence-activated cell sorting on the BD FACS Aria II. Second, A549-Cas9 cells were transduced by concentrated Target Site (GFP) lentivirus and GFP+ cells were selected by FACS; after sorting, Target Site infection and sorting were repeated two more times for a total of three serial lentiviral transfections,

sorting for cells with progressively higher GFP signal after each infection. This strategy of serial transfection with concentrated lentivirus yielded cells with high copy numbers of the Target Site, which were confirmed by quantitative PCR. Third, A549 cells with Cas9 and Target Site were transduced by titered triple-sgRNA (BFP-PuroR) lentivirus and selected as described below.

In vitro lineage tracing experiment, single-cell RNA-seq library preparation, and sequencing

One day following triple-sgRNA infection, cells were trypsinized to a single-cell suspension and counted using an Accuri cytometer (BD Biosciences). Approximately 25 cells were plated in a single well of a 96-well plate. Seven days post-infection, cells were trypsinized and split evenly into two wells of a 96-well plate. Cells stably transduced by triple-sgRNA lentivirus were selected by adding puromycin at 1.5 μg/mL on days 9 and 11 post-infection; puromycin-killed cells were removed by washing the plate with fresh medium. After 14 days, cells were trypsinized and split evenly for a second time into four wells of a 6-well plate. Finally, after 21 days in total, cells from the four wells were trypisinized to a single-cell suspension and collected.

Cells were washed with PBS with 0.04% w/v bovine serum albumin (BSA, New England Biolabs), filtered through 40 μm FlowMi filter tips filter tips (Bel-Art), and counted according to the 10x Genomics protocol. Approximately 14,000 cells per sample were loaded (expected yield: approximately 10,000 cells per sample) into the 10x Genomics Chromium Single Cell 3' Library and Gel Bead Kit v2, and cDNA was reverse-transcribed, amplified, and purified according to the manufacturer's protocol. Resulting cDNA libraries were quantified by BioAnalyzer, yielding the expected size distribution described in the manufacturer's protocol.

To prepare the Target Site amplicon sequencing library, resulting amplified cDNA libraries were further amplified with custom, Target Site-specific primers containing P5/P7 Illumina adapters and sample indices (forward: CAAGCAGAAGACGGCATACGAGATXXXXXXXX GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAGAATCCAGCTAGCTGTGCAGC; reverse: CAAGCAGAAGACGGCATACGAGATXXXXXXXXGTCTC GTGGGCTCGGAGATGTGTATAAGAGACAGGCATGGACGAGCTGTACAAGT; "X" denotes sample indices). PCR amplification was performed using Kapa HiFi HotStart ReadyMix, as in [2], according to the following program: melting at 95^{o}C for 3 minutes, then 14 cycles at 98^{o}C for 15 seconds and 70^{o}C for 20 seconds. Approximately 12 fmol of template cDNA were used per reaction; amplification was performed in quadruplicate to avoid PCR-induced library biases, such as jack-potting. PCR products were re-pooled and purified by SPRI bead selection at 0.9x ratio and quantified by BioAnalyzer.

Target Site amplicon libraries were sequenced on the Illumina NovaSeq S2 platform. Due to the low sequence complexity for the Target Site library, a phiX genomic DNA library was

spiked in at approximately 50% for increased sequence diversity. The 10x cell barcode and unique molecular identifier (UMI) sequences were read first (R1: 26 cycles) and the Target Site sequence was read second (R2: 300 cycles); sample identities were read as indices (I1 and I2: 8 cycles, each). Over 550M sequencing clusters passed filter and were processed as described below. All raw and processed data are available through GEO Series accession GSE146712 [99].

3.2.8 Processing Pipeline

Read Processing

Each target site was sequenced using the Illumina Nova-seq platform, producing 300bp long-read sequences. The Fastq's obtained were quantitated using 10x's cellranger suite, which simultaneously corrects cell barcodes by comparing against a whitelist of 10x's approved cell barcodes.

For each cell, a consensus sequence for each unique molecule identifier (UMI) was produced by collapsing similar sequences, defined by those sequences differing by at most 1 Levenshtein distance. A directed graph is constructed, where sequences with identical UMI's are connected to one another if the sequences themselves differ by at most one Levenshtein distance. Then, UMI's in this network are collapsed onto UMI's that have greater than or equal number of reads. This produces a collection of sequences indexed by the cell barcode and UMI information (i.e. there is a unique sequence associated with each UMI).

Before aligning all sequences to the reference, preliminary quality control is performed. Specifically, in cases where UMI's in a given cell still have not been assigned a consensus sequence, the sequence with the greatest number of reads is chosen. UMIs with fewer than 2 reads are filtered out, and cells with fewer than 10 UMIs are filtered out as well. Finally, a filtered file in Fastq format is returned.

Allele Calling

Alignment is performed with Emboss's Water local alignment algorithm. Optimal parameters were found by performing a grid search of gap open and gap extend parameters on a set of 1,000 simulated sequences, comparing a global and local alignment strategy. We found a gap open penalty of 20.0 and a gap extension penalty of 1.0 produced optimal alignments. The "indels" (insertions and deletions resulting from the Cas9 induced double-strand break) at each cut site in the sequences are obtained by parsing the cigar string from the alignments. To resolve possible redundancies in indels resulting from Cas9 cutting, the 5' and 3' flanking 5-nucleotide context is reported for each indel.

UMI Error Correction

To correct errors in the UMI sequence either introduced during sequencing, PCR preparation, or data processing, we leverage the allele information. UMIs are corrected within groups of identical cell barcode-integration barcode pairs (i.e. we assume that only UMIs encoding for the same intBC in a given cell can be corrected). We reason that ideally, for a given integration barcodes, a cell will only report one sequence, or allele. Within these "equivalence classes," UMIs that differ by at most 1 Levenshtein distance (although this number can be user-defined) are corrected towards the UMI with a greater number of reads.

Cell-based Filtering

With the UMI corrected and indels calculated, the new "molecule table" is subjected to further quality control. Specifically, UMIs are filtered based on the number of reads (dynamically set to be the 99th percentile of the reads divided by 10), integration barcodes (denoting a particular integration site) can be error corrected based on a minimum hamming distance and identical indels (referred to as alleles), and in the case where multiple alleles are associated with a given integration barcode a single allele is chosen based on the number of UMIs associated with it.

Calling Independent Clones

Collections of cells part of the same clonal population, are identified by the set of integration barcodes each cell contains. Because all cells in the same clone are clonal, we reasoned that cells in the same clone should all share the same set of integration barcodes that the progenitor cell contained. Because of both technical artifacts (e.g. sequencing errors, PCR amplification errors) and biological artifacts (e.g. bursty expression, silenced regions) however, rather than looking for sets of non-overlapping sets, we perform an iterative clustering procedure. We begin by selecting the intBC that is shared amongst the most cells and assign any cell that contains this barcode to a cluster and remove these cells from the pool of unassigned cells. We perform this iteratively until at most k percent (in our case defined as .5% of cells are unassigned, which we assign to a "junk" clone.

Using the set of integration barcodes for each clone, we are able to identify doublets that consist of cells from different clones. Finally, after identifying doublets, to further filter out low quality integration barcodes, for each clone integration barcodes that are not shared by at least 10% of cells in a given clone are filtered out, producing the final allele table.

Guidelines for Final Quality Control

The thresholds discussed above are heuristic choices determined based on our hands-on experience with this type of target-site library processing. However, these thresholds will undoubtedly change depending on the sequencer used, the sequencing depth of the library,

and the biological use case. For these reasons, we suggest that it is more effective to ensure that the final quality control numbers indicate that the library was processed sufficiently.

We present distributions for the metrics we find to be the most useful in Additional file 1: Fig S17: the UMIs per cellBC as a measure for how well sampled a cell is in (a), the reads per UMI as a measure for how confident one is of the UMI sequence in (b), UMIs per intBC as a measure for how confident one is of the called allele and intBC in (c), and a comparison of the number of UMIs versus the number of reads in (d), as a way of quickly assessing if there are any outlier UMIs.

Because this library was sequenced quite deeply, we do not expect typical applications to afford this degree of certainty. Instead, we suggest that cells should have at least 10 target-site UMIs, the reads/UMI distribution should have a mean at around 100-200 reads, and each intBC should have at least 5-10 UMIs associated with it. Cassiopeia's processing pipeline creates figures for each of these statistics after filtering and close attention should be paid to these figures during the processing of the target-site sequencing data.

Filtering of clones for Reconstruction

We filtered out clones upon two criteria: firstly, we removed clone 1 as we deduced that it had two defective guides; secondly, we removed lineages that reported fewer than 10% unique cells (thus removing clone 7). The remainder of clones were reconstructed.

Estimation of Per Character Mutation Rates

To estimate mutation rates per clone, we assume that every target site was mutated at the same rate and independently of one another across 15 generations. Assuming some mutation rate, p, per character, we know that the probability of not observing a mutation in d generations is $(1-p)^d$ in a given character and that the probability of observing at least 1 mutation in that character is $1-(1-p)^d$. Then, giving this probability $1-(1-p)^d = m$ can be used as a probability of observing a mutated character in a cell and model the number of times a character appears mutated in a cell as a binomial distribution where the expectation is simply nm where n is the number of characters. Said simply, given this model, one would expect to see nm characters mutated in a cell). In this case, the empirical expectation is the mean number of times a given character appeared mutated in a cell (averaged across all cells), which we denote as K and propose that

$$K = nm = n * (1 - (1-p)^d)$$

and thus p, the mutation rate, is

$$p = 1 - (1 - K/n)^d$$

3.2.9 Bulk Cutting Experiment to Determine Prior Probabilities of Indel Formation

Two and four days following triple-sgRNA (PCT61) infection, infected cells were selected by adding puromycin at 1.5 μg/mL; puromycin-killed cells were removed by washing the plate with fresh medium. Cells were split every other day, and 500k cells were collected on days 7, 14, and 28. Frozen cell pellets were lysed and the genomic DNA was extracted and purified by ethanol precipitation. The PCT48 Target Site locus was PCR amplified from genomic DNA samples (forward: TCGTCGGCAGCGTCAGATGTGTATAAGAGACAGAATCCAGCTAGCTGTGCAGC; reverse: GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAGTCGAGGCTGATCAGCG) and further amplified to incorporate Illumina adapters and sample indices (forward: AATGATACGGCGACCACCGAGATCTACACXXXXXXXXTCGTCGGCAGCGTCAG; reverse: CAAGCAGAAGACGGCATACGAGATXXXXXXXXGTCTCGTGGGCTCGGAG; "X" denotes sample indices). The subsequent amplicon libraries were sequenced on an Illumina MiSeq (paired end, 300 cycles each). Sequencing data was analyzed as described below.

3.2.10 Determining Prior Probabilities of Indel Formation

To determine the prior probabilities of edits, we leverage the fact that we have access to a large set of target sites (or intBCs) with a similar sequence (apart from the random barcode at the 5' end); namely, a total of 117 intBC across the 11 clones. To compute the prior probability for a given indel, we compute the empirical frequency of observing this mutation out of all unique edits observed. Specifically, we compute the prior probability of a given indel s, q_s as the following:

$$q_s = \frac{f(s)}{|I|}$$

where $f(s)$ is the number of intBC's that had s in at least one cell and $|I|$ is the number of intBCs that are present in the dataset.

As further support for this method, we used the bulk experiment consisting of many separately engineered A549 cells, as described in the previous section. The advantage of the bulk experiment is that we have access to substantially more intBCs ($> 10k$), thus providing a more robust estimation of q_s. We therefore employed the same approach to estimate indel formation rates from the bulk data and find that the resulting rates correlate well with the indel rates estimated from the single cell lineage tracing experiment (Additional file 1: Fig S20).

3.2.11 Doublet Detection

Methods to Detect Doublets

We hypothesized that doublets could come in two forms and that we could use various components of the intBC data structure to identify them. Namely, doublets could be of cells from the identical clone, here dubbed "intra-doublets", or doublets could be of cells from separate clones, here dubbed "inter-doublets."

In the case of "intra-doublets", we can utilize the fact that these cells will have a large overlap in their set of intBCs but will report "conflicting" alleles for each of these intBCs. Thus, to identify these doublets, we calculate the percentage of UMIs that are conflicting in each cell. Explicitly, for each cell we iterate over all intBCs and sum up the number of UMIs that correspond to an allele that conflicts with the more abundant allele for a given intBC; we then use the percentage of these UMIs to identify doublets. We perform this after all UMI and intBC correction in hopes of calling legitimate conflicts.

To deal with "inter-doublets", we developed a classifier that leverages the fact that cells from different clones should have non-overlapping intBC sets. While this is the ideal scenario, often times intBCs are shared between clones for one of two reasons (1) the clustering assignments are noisy or (2) the transfections of intBCs resulted in two cells receiving the same intBC, even though cells are supposed to be progenitors of separate clones. Our strategy is thus: for each cell $c_i \in C$ calculate a "membership statistic", $m_{i,k}$ for each clone $l_k \in L$. The membership statistic is defined as so:

$$m_{i,k} = \frac{\sum_{j \in I_k} \delta(i,j) p(j,k)}{\sum_{j \in I_k} (p(j,k))}$$

where I_k is the set of intBCs for the clone l_k and $p(j,k)$ is the prevalence rate of the intBC j in l_k. We use $\delta(i,j)$ as an indicator function for whether or not we observed the intBC j in the cell c_i. Intuitively, this membership statistic is a weighted similarity for how well the cell fits into each clone, where we are weighting by how much we are able to trust the intBC that is observed in the cell. To put all on the same scale, we normalize by total membership per cell, resulting in our final statistic, $m'_{i,k} = \frac{m_{i,k}}{\sum_{k'=0}^{k} m_{i,k'}}$ We then filter out doublets whose m' for their classified clone falls below a certain threshold.

Simulation of Doublets

We simulated two datasets to test our methods for identifying doublets and to find the optimal criterion on which to filter out doublets. To test this strategy, we took a single clone from our final Allele Table (the table relating all cells and their UMIs to clones) and formed 200 doublets by combining the UMIs from two cells. We generated 20 of these datasets, and noted which cells were artificially introduced doublets.

Contrary to the strategy for simulating doublets from the same clone, we created artificial "inter" doublets from the final Allele Table by combining doublets from two different clones. Similarly, we generated 20 synthetic datasets each with 200 of these artificial doublets.

Identification of Decision Rule

To identify the optimal decision rule for calling both types of doublets, we tested decision rules ranging from 0 to 1.0 at 0.05 intervals and calculated the precision and recall at each of these rules. Taking these results altogether, we provide an optimal decision rule where the F-measure (or the weighted harmonic mean of the precision and recall) of these tests is maximal.

3.2.12 Algorithmic Approaches For Phylogenetic Reconstruction

One way to approach the phylogenetic inference problem is to view each target site as a "character" that can take on many different possible "states" (each state corresponding to an indel pattern induced by a CRISPR/Cas9 edit at the target site). Formally, these observations can be summarized in a "character matrix", $M \in R^{n,m}$, which relates the n cells by a set of characters $\chi = \{\chi_1, ..., \chi_m\}$ where each character χ_i can take on some k_i possible states. Here, each sample, or cell, can be described as a concatenation of all of their states over characters in a "character string". From this character matrix, the goal is to infer a tree (or phylogeny), where leaf nodes represent the observed cells, internal nodes represent ancestral cells, and edges represent a mutation event.

We first propose an adaption of a slow, but accurate, Steiner-Tree algorithm via Integer Lineage Programming (ILP) to the lineage tracing phylogeny problem. Then, we propose a fast, heuristic-based greedy algorithm which simultaneously draws motivation from classical perfect phylogeny algorithms, and the fact that mutations can only occur unidirectionaly from the unmutated, or s_0 state. Lastly, we combine these two methods and present a hybrid method, which presents better results than our greedy approach, yet remains feasible to run over tens of thousands of cells.

Adaptation to Steiner Tree Problem

Steiner Trees are a general problem for solving for the minimum weight tree connecting a set of target nodes. For example, if given a graph $G = (V, E)$ over some V vertices and E edges, finding the Steiner-Tree over all $v \in V$ would amount to solving for the minimum spanning tree (MST) of G. While there exist polynomial time algorithms for the minimum-spanning tree, the general Steiner Tree problem, where the set of targets $T \subseteq V$ is designated, is NP-hard.

Previously, Steiner-Trees have been suggested to solve for the maximum parsimony solution to the phylogeny problem. Here, the graph would consist of all possible cells (both

observed and unobserved) and each edge would consist of a possible evolutionary event connecting two states (e.g. a mutation). Generally, given a set of length-l binary "character-strings" (recall that these are the concatenation of all character states for a given sample), we can solve for the maximum parsimony solution by finding the optimal Steiner Tree over the 2^l hypercube (i.e. graph). As a result, by converting our multi-state characters to binary characters via one hot encoding, theoretically, we should be able to compute the most parsimonious tree which best explains the observed data. However, in practice this method turns out to be infeasible, as we deal with hypercubes of size $O(2^{mn})$, where m is the number of characters, and n is the number of states. In the following, we will propose a method for estimating the underlying search space, providing us with a feasible solvable instance and a formulation of an Integer-Linear Programming (ILP) problem to solve for the optimal Steiner-Tree.

Approximation of Potential Graph

We first begin by constructing a directed acyclic graph (DAG) G, where nodes represent cells. We then take the source nodes, or nodes with in-degree 0, of G, and for each pair of source nodes, consider the latest common ancestor (LCA) they could have had. This LCA has an unmutated state for character χ_i if they disagree across two source nodes, and the same state as the two source nodes if they agree in value. If the edit distance between these two cells is below a certain threshold d, we add the LCA to G, along with directed edges to the two source nodes, weighted by the edit distance between the parent and the source. We repeat this process until only one node remains as a source: the root.

One may think that this step explodes with $O(n^2)$ complexity at each stage, where n is the number of source nodes in each prior stage, as we consider all pairs of source nodes. However, we note that the number of mutations per latest common ancestor is always less than both children, and therefore, we eventually converge to the root. Therefore, when dealing with several hundred cells, the potential graph is feasible to calculate.

Furthermore, to add scalability to the approximation of the Potential Graph, we allow the user to provide a "maximum neighborhood size" which will be used to dynamically solve for the optimal LCA distance threshold d to use. One may think of this as the maximum memory or time allowed for optimizing a particular problem. Since the size of the Potential Graph can grow quite large in regards to the number of nodes, we iteratively create potential graphs for various threshold d and at each step ensure that the number of nodes in the network does not exceed the maximum neighborhood size provided. If at any point the number of nodes does exceed this maximum size, we return the potential graph inferred for an LCA threshold of $d - 1$.

Formulation of Integer Linear Programming Problem

Given our initial cells, S , the underlying potential graph drawn from such cells, G, and the final source node, or root, r from G, we are interested in solving for $\mathcal{T} = SteinerTree(r, S, G)$.

We apply an integer linear programming (ILP) formulation of Steiner Tree, formulated in terms of network flows, with each demand being met by a flow from source to target. Below we present the Integer Linear Programming formulation for Steiner Tree. We use Gurobi [73], a standard ILP solver package

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in E} d^b_{uv} \cdot w(u,v) & & \\
\text{subject to} \quad & \sum_{(u,v)\in E} d_{uv} - \sum_{(v,w)\in E} d_{vw} = 0 & & \forall v \notin S \cup \{r\} \\
& \sum_{(r,w)\in E} d_{rw} = -|S| & & \\
& \sum_{(u,s)\in E} d_{us} = 1 & & \forall s \in S \\
& d^b_{uv} \geq \frac{d_{uv}}{|S|} & & \forall (u,v) \in E \\
& d_{uv} \in \{0,..,|S|\} & & \forall (u,v) \in E \\
& d^b_{uv} \in \{0,..,1\} & & \forall (u,v) \in E
\end{aligned}
$$

Each variable d_{uv} denotes the flow through edge (u, v), if it exists; each variable d^b_{uv} denotes whether (u, v) is ultimately in the chosen solution sub-graph. The first constraint enforces flow conservation, and hence that the demands are satisfied, at all nodes and all conditions. The second constraint requires $|S|$ units of flow come out from the *root*. The third constraint requires that each target absorb exactly one unit of flow. The fourth constraint ensures that if an edge is used at any condition, it is chosen as part of the solution.

Below we explicitly define the algorithm in pseudocode.

function ILP-SOLVER($cells = S$)
    Potential Graph $G \leftarrow$ BUILD-POTENTIAL-GRAPH(S)
    if $G == None$ **then**
        return GREEDY-SOLVER(S)
    $r \leftarrow$ root of G
    $\mathcal{T} \leftarrow$ STEINER-TREE(r, G, S) ▷ Steiner Tree ILP Solver
    return $\mathcal{T}$

function BUILD-POTENTIAL-GRAPH(cells = S, max lca length = k, max neighborhood size = N)
    $\mathcal{T}_0 = None$
    for all $d \in [1, k]$ **do**
        $\mathcal{T} \leftarrow$ DiGraph()
        for all $s \in S$ **do**
            $\mathcal{T} \leftarrow \mathcal{T} \cup \{s\}$
        $sources \leftarrow$ all source nodes in $\mathcal{T}$
        while len(sources) > 1 **do**
            for all $v_1, v_2 \in sources$ **do**
                $lca \leftarrow$ latest common ancestor of v_1, v_2
                if $dist(lca, v_1) + dist(lca, v_2) \leq d$ **then**
                    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(lca, v_1), (lca, v_2)\}$
            $sources \leftarrow$ all source nodes in $\mathcal{T}$
            if len(sources) $\geq N$ **then**
                return $\mathcal{T}_{d-1}$
        $\mathcal{T}_d \leftarrow \mathcal{T}$
    return $\mathcal{T}$

Stability Analysis of the Maximum Neighborhood Size Parameter

To evaluate the stability of the user-defined maximum neighborhood size parameter, we assessed the accuracy of the reconstructions for parameters varying from 800 to 14,000. We used trees simulated under default conditions (400 samples, 40 characters, 40 states per character, 11 generations, 2.5% mutation rate per character, and a mean dropout rate of 17%). The accuracy of trees were compared to the tree generated with a parameter of 14,000 using the triplets correct statistic. We used 10 replicates to provide a sense for how stable a given accuracy is.

In addition to providing measures of accuracy, we also provide the optimal LCA threshold d found for a given maximum neighborhood size during the inference of these potential graphs. Using these analysis, we found that a maximum neighborhood size of 10,000 nodes seemed to be an ideal tradeoff between scalability and accuracy (as it is in the regime where

accuracy saturates) for our default simulations. This corresponded to a mean LCA threshold, d, of approximately 5.

Heuristic-Based Greedy Method

On Perfect Phylogeny & Single Cell Lineage Tracing

In the simplest case of phylogenetics, each character is binary (i.e. $k_i = 2, \forall i \in m$) and can mutate at most once. This case is known as "perfect phylogeny" and there exist algorithms (e.g. a greedy algorithm by Dan Gusfield [74]) for identifying if a perfect phylogeny exists over such cells, and if so find one efficiently in time $O(mn)$, where m is the number of characters and n are the number of cells. However, several limitations exist with methods such as Gusfield's algorithm. One potential problem in using existing greedy perfect phylogeny algorithms for lineage tracing is that they require the characters to be binary. Indeed, if the characters are allowed to take any arbitrary number of states, the perfect phylogeny problem becomes NP-hard. However, while the number of states (CRISPR/Cas9-induced indels at a certain target site) in lineage tracing data can be large, these data benefit from an additional restriction that makes it more amenable for analysis with a greedy algorithm. Below, we show that because the founder cell (root of the phylogeny) is unedited (i.e. includes only uncut target sites) and that the mutational process is irreversible, we are able to theoretically reduce the multi-state instance (as observed in lineage tracing) to a binary one so that it can be resolved using a greedy algorithm.

A second remaining problem in using these perfect phylogeny approaches is that we cannot necessarily expect every mutation to occur exactly once. In theory, it may happen that the same indel pattern is induced in exactly the same target site on two separate occasions throughout a lineage tracing experiment, especially if a large number of cell cycles takes place. A final complicating factor is that these existing greedy algorithms often assume that all character-states are known, whereas lineage tracing data is generated by single-cell sequencing, which often suffers from limited sensitivity and an abundance of "dropout" (stochastic missing data) events.

The Greedy Algorithm

We suggest a simple heuristic for a greedy method to solve the maximum parsimony phylogeny problem, motivated by the classical solution to the perfect phylogeny problem and irreversibility of mutation. Namely, we consider the following method for building the phylogeny: Given a set of cells, build a tree top-down by splitting the cells into two subsets over the most frequent mutation. Repeat this process recursively on both subsets until only one sample remains.

Formally, we choose to split the dataset into two subsets, $O_{i,j}$ and $\overline{O}_{i,j}$, such that $O_{i,j}$ contains cells carrying mutation s_j in χ_i, and $\overline{O}_{i,j}$ contains cells without s_j in χ_i. We choose i, j based on the following criteria:

$$i, j = \arg\max_{i,j} n_{i,j}$$

where $n_{i,j}$ is the number of cells that carry mutation s_j in character χ_i. We continue this process recursively until only one sample exists in each subset. We note that this method operates over cells with non-binary states, solving the first of problems addressed earlier.

A major caveat exists with methods such as the greedy method proposed by Gusfield, as well as the one proposed by us thus far: namely, they assume all character states are known (i.e. no dropout). However, in our practice, we often encounter dropout as a consequence of Cas9 cutting or stochastic, technical dropout due to the droplet-based scRNA-seq platform. To address this problem in our greedy approach, during the split stage, these cells are not initially assigned to either of the two subsets, $O_{i,j}$ or $\overline{O}_{i,j}$. Instead, for each individual sample which contains a dropped out value for chosen split character χ_i, we calculate the average percentage of mutated states shared with all other cells in $O_{i,j}$ and $\overline{O}_{i,j}$ respectively, and assign the sample to the subset with greater average value.

Appending the dropout resolution stage with the initial split stage, we present our greedy algorithm below in its entirety.

---

function GREEDY-SOLVER($cells = S$, prior probabilities $= p$)
  if $len(S) = 1$ **then**
    return S
  $root \leftarrow$ latest common ancestor across all S
  $i, s_j \leftarrow$ maximally occurring character mutation pair in S weighted by priors p
  $O_{i,j} \leftarrow$ all cells in S with mutation s_j in χ_i
  $\overline{O}_{i,j} \leftarrow$ all cells in S without mutation s_j in χ_i and without dropout for χ_i
  $D_i \leftarrow$ all cells in S with dropout for χ_i ▷ Note $O_{i,j} \cup \overline{O}_{i,j} \cup D_i = S$
  for all $s \in D_i$ **do**
    if s shares more mutated states on average with cells in $O_{i,j}$ over $\overline{O}_{i,j}$ **then**
      $O_{i,j} \leftarrow O_{i,j} \cup \{s\}$
    else
      $\overline{O}_{i,j} \leftarrow \overline{O}_{i,j} \cup \{s\}$
  $\mathcal{T}_L, \mathcal{T}_R \leftarrow$ GREEDY-SOLVER($O_{i,j}, p$), GREEDY-SOLVER($\overline{O}_{i,j}, p$)
  $r_L, r_R \leftarrow$ root of $\mathcal{T}_L, \mathcal{T}_R$ respectively
  $\mathcal{T} \leftarrow \mathcal{T}_L \cup \mathcal{T}_R \cup \{root\}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(root, r_L), (root, r_R)\}$
  return $\mathcal{T}$

Overall, this method is very efficient, and scales well into tens of thousands of cells. Below, we show via proof below that this algorithm can find perfect phylogeny if one exists.

Cassiopeia-Greedy Algorithm Can Solve Multi-State Perfect Phylogeny

Here we show that while not required, Cassiopeia can solve the multi-state perfect phylogeny problem optimally. Importantly, however, Cassiopeia's effectiveness makes no assumption about perfect phylogeny existing in the dataset but rather leverages this concept to provide a heuristic for scaling into larger datasets.

To show how Cassiopeia's greedy method can solve perfect phylogeny optimally, we begin by introducing a few clarifying definitions prior to the main theorem. We define M as the original n cells by n character k-state matrix (i.e. entries $\in \{s_0, \ldots, s_{k-1}\}$). We say M has a zero root perfect phylogeny if there exists a tree $\mathcal{T}$ over its elements and character extensions such that the state of the root is all zeros and every character state are mutated into at most once. In addition, we assume that all non-leaf nodes of $\mathcal{T}$ have at least two children (i.e. if they only have one child, collapse two nodes into one node). Finally, we offer a definition for *character compatibility*:

Definition 26. *(Character Compatibility). For a pair of binary characters, (χ_1, χ_2), where the sets (O_1, O_2) contain the sets of cells mutated for χ_1 and χ_2, respectively, we say that they are compatible if one of the following is true:*

- $O_1 \subseteq O_2$
- $O_2 \subseteq O_1$
- $O_1 \cap O_2 = \emptyset$

This definition extends to multi-state characters as well, assuming they can be binarized.

Before proving the main theorem, we first prove the following lemma:

Lemma 9. *If M has a perfect phylogeny, then the most frequent character, mutation pair appears on an edge from the root to a direct child node.*

Proof. WLOG let $\chi_i : s_0 \rightarrow s_j$ denote the maximally occurring character, mutation pair within M. Suppose by contradiction that this mutation does not appear on an edge directly from root to a child, but rather on some edge (u, v) that is part of a sub-tree whose root r^*, is a direct child of the root. As r^* has at least two children, this implies that the mutation captured from the root to r^* must be shared by strictly more cells than $\chi_i : s_0 \rightarrow s_j$, thereby reaching a contradiction on $\chi_i : s_0 \rightarrow s_j$ being the maximally occurring mutation. □

Theorem 12. *The greedy algorithm accurately constructs a perfect phylogeny over M if one exists.*

Proof. We approach via proof by induction. As a base case, a single is trivially a perfect phylogeny over itself.

Now suppose by induction that for up to $n-1$ cells, if there exists a perfect phylogeny $\mathcal{T}$ over such cells, then the greedy algorithm correctly returns the perfect phylogeny. Consider the case of n cells. By the above lemma, we know we can separate these n cells into two subsets based on the most frequent character, mutation pair $\chi_i : s_0 \to s_j$, $O_{i,j}$ and $\overline{O}_{i,j}$, where $O_{i,j}$ contains cells with mutation s_j over χ_i, and $\overline{O}_{i,j} = M - O_{i,j}$. By induction, the greedy algorithm correctly returns two perfect phylogenies over $O_{i,j}$ and $\overline{O}_{i,j}$, which we can merge at the root, giving us a perfect phylogeny over n cells. □

Accounting for Prior Probability of Mutations

In most situations, the probability of mutation to each distinct state may not be uniform (i.e. character χ_1 mutating from the unmutated state s_0 to state s_4 may be twice as likely as mutating to state s_6). Therefore, we incorporate this information into choosing which character and mutation to split over based on the following criteria:

$$i, j = \arg\min_{i,j} p_i(s_0, s_j)^{f(n_{i,j})}$$

where $p_i(s_0, s_j)$ is the probability that character χ_i mutates from the unmutated state s_0 to s_j and $f(n_{i,j})$ is some transformation of the number of cells that report mutation j in character i that is supposed to reflect the future penalty (number of independent mutations of character i to state j) we will have to include in the tree if we do not pick i, j as our next split. After a comparison of 5 different transformations (Supp Figure 4), we find that $f(n_{i,j}) = n_{i,j}$ gives the best performance, leaving us with the following criteria for splittings:

$$i, j = \arg\min_{i,j} p_i(s_0, s_j)^{n_{i,j}}$$

A Hybrid Method for Solving Single Cell Lineage Tracing Phylogenies

Due to the runtime constraints of the Steiner Tree Method, it is infeasible for such method to scale to tens of thousand of cells. Therefore, we build a simple hybrid method which takes advantage of the heuristic proposed in the greedy algorithm and the theoretical optimality of the Steiner Tree method.

Recall that in the greedy method, we continued to choose splits recursively until only one sample was left per subset. In this method, rather than follow the same process, we choose a cutoff for each subset (e.g. 200 cells). Once a subset has reached a size lower than said cutoff, we feed each individual subset into the Potential Graph Builder and Steiner Tree solver, which compute an optimal phylogeny for the subset of cells. After an optimal subtree is found, we merge it back into the greedy tree. Therefore, we build a graph whose initial mutations are chosen from the greedy method, and whose latter mutations are chosen more precisely via the Steiner Tree approach.

Below we present a pseudo-code algorithm for the hybrid method. We note the slight difference in greedy from before. Namely, greedy additionally accepts a cutoff parameter, and in addition to returning a network built up to that cutoff, returns all subsets that are still needed to be solved.

1: **function** CASSIOPEIA-HYBRID(*cells* $= S$, greedy cutoff $= g$)
2: $\quad \mathcal{T}, \mathcal{S} \leftarrow$ GREEDY-SOLVER(S, g)
3: $\quad$ **for all** $S' \in \mathcal{S}$ **do**
4: $\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup$ ILP-SOLVER(S')
5: $\quad$ **return** $\mathcal{T}$

This approach scales well when each instance of Steiner Tree is ran on an individual thread, and thus often takes only a few hours to run on several thousand cells.

3.2.13 Theoretical Analysis of Parallel Evolution

Estimating First and Second Moments of Double Mutations

Expected Number of Double Mutations

Under the framework of our simulation, we assume that each at each generation, every cell divides, and then each character of each cell undergoes random mutation independently. Let p be the probability that a particular character mutates, and q be the probability the character took on a particular mutated state given that it mutated. Let T be the true phylogenetic tree over the samples. According to our model, T must be a full binary tree, and the samples are leaves of T. Let X be the total number of times a particular mutation occurred in the T. Let $X_{u,v}$ be an indicator variable for edge (u, v) such that:

$$X_{u,v} = \begin{cases} 1 & \text{if a mutation occurs on edge } (u,v) \\ 0 & \text{otherwise} \end{cases}$$

Let h be the height of the T, which is equalled to the number of generations. If v is at depth d in T, then the probability that a mutation occurs at (u, v) is $pq(1-p)^{d-1}$. Since there are 2^d nodes at depth d, we have:

$$\begin{aligned} E(X) &= \sum_{(u,v)\in T} E(X_{u,v}) \\ &= \sum_{d=1}^{h} 2^d pq(1-p)^{d-1} \\ &= \frac{2pq((2-2p)^h - 1)}{1-2p} \end{aligned} \tag{3.1}$$

Let $n = 2^h$ is the number of cells in our sample. If $p > 0.5$, $E(X) \leq 2pq/(2p-1)$, if $p = 0.5$, $E(X) = 2pqh = O(\log n)$, and if $p < 0.5$, $E(X) = O(n^{\frac{1}{\log_2 2-2p}})$. Moreover, for fixed h, $E(X)$ has a single peak for $p \in [0,1]$, meaning that it increases with p for sufficiently small values of p, and always increases with q. Intuitively, this is because $E(X)$ is small if 1) p is small enough that the character never mutates much throughout the experiment or 2) p is large enough that most mutations occur near the top of the tree, resulting in the extinction of unmutated cells early in the experiment. While $E(X)$ peaks for values of p in between, it is always directly proportional to q because X is simply equalled to q time the number of times the character mutated.

Variance of Double Mutations

We can compute the variance as:

$$\begin{aligned} Var(X) &= E(X^2) - E(X)^2 \\ &= 2 \sum_{(u,v)\neq(u',v')} E(X_{u,v}X_{u',v'}) + E(X) - E(X)^2 \end{aligned}$$

To compute $E(X_{u,v}X_{u',v'})$, we note that for a given pair of edges (u,v) and (u',v'), such that $LCA(u,u')$ is at depth d, u is at depth $d+l$, and u' is at depth $l+k$, the probability that a mutation occurred on both edges is $p^2q^2(1-p)^{d+l+k}$. Thus, we have:

$$\begin{aligned} \sum_{(u,v)\neq(u',v')} E(X_{u,v}X_{u',v'}) &= \sum_{d=0}^{h-1} 2^d \sum_{k=0}^{h-d-1} \sum_{l=0}^{h-d-1} 2^{l+k} p^2q^2(1-p)^{d+l+k} \\ &= p^2q^2 \sum_{d=0}^{h-1} (2-2p)^d (\sum_{k=0}^{h-d-1} (2-2p)^k)^2 \\ &= \frac{p^2q^2}{(2p-1)^2} \sum_{d=0}^{h-1} (2p-2)^d ((2p-2)^{(h-d)} - 1)^2 \\ &\leq \frac{p^2q^2}{(2p-1)^2} \sum_{d=0}^{h-1} (2p-2)^{2h-d} \\ &= (2p-2)^{h+1} \frac{p^2q^2}{(2p-1)^2} \sum_{d=0}^{h-1} (2p-2)^d \\ &\leq \frac{p^2q^2(2p-2)^{2h+1}}{(2p-1)^3} \end{aligned}$$

Thus, we can bound the variance as follows:

$$Var(X) \leq \frac{2p^2q^2(2p-2)^{2h+1}}{(2p-1)^3} + \frac{2pq(1-(2-2p)^h)}{2p-1} - \frac{4p^2q^2(1-(2-2p)^h)^2}{(2p-1)^2} \tag{3.2}$$

This means that in the case that $p > 0.5$:

$$Var(X) \leq \frac{2p^2q^2}{(2p-1)^3} + \frac{2pq}{2p-1} - \frac{4p^2q^2}{(2p-1)^2}$$

In the case that $p = 0.5$:

$$Var(X) = O(h^3) = O(\log^3(n))$$

In the case that $p < 0.5$:

$$Var(X) = O(n^{\frac{2}{\log_2 2-2p}})$$

Least Squares Linear Estimate & Negative Correlation Between Frequency and Number of Double Mutations

To justify the greedy, we must show that if a mutation occurs frequently, then it is likely to have occurred less times throughout the experiment. Let Y be the frequency of a particular mutation in the samples. We estimate X given Y using the least squares linear estimate (LLSE) as follows:

$$L(X|Y) = E(X) + \frac{CoV(X,Y)}{Var(Y)}(Y - E(Y)) \tag{3.3}$$

Since $CoV(X,Y) = E(XY) - E(X)E(Y)$, we need only to compute $E(XY)$, which we do by expressing X and Y in terms of the same indicators:

$$Y = \frac{1}{2^h} \sum_{(u,v)\in T} 2^{depth(v)} X_{u,v}$$

As a sanity check, it can easily be verified that $E(Y) = q(1-(1-p)^h)$ by computing $E(Y)$ using these indicators:

$$\begin{aligned} E(Y) &= 2^{-h} \sum_{d=1}^{h} 2^d (1-p)^{d-1} pq * 2^{h-d} \\ &= pq \sum_{d=1}^{h} (1-p)^{d-1} \\ &= q(1-(1-p)^h) \end{aligned}$$

Thus, we can compute $E(XY)$ similar to how we computed $E(X^2)$ for Variance.

$$\begin{aligned}
E(XY) &= 2^{-h}E((\sum_{(u,v)\in T} X_{u,v})(\sum_{(u,v)\in T} 2^{depth(v)}X_{u,v})) \\
&= 2^{-h}\Big(2\sum_{(u,v)\neq(u',v')} 2^{depth(v)}E(X_{u,v}X_{u',v'}) + \sum_{(u,v)\in T} 2^{depth(v)}E(X_{u,v}^2)\Big) \\
&= 2*2^{-h}\sum_{d=0}^{h-1}2^d\sum_{k=0}^{h-1}\sum_{l=0}^{h-1}2^{l+k}p^2q^2(1-p)^{d+l+k}*2^{h-d-l-1} + E(Y) \\
&= p^2q^2\sum_{d=0}^{h-1}\sum_{k=1}^{h-d-1}\sum_{l=0}^{h-d-1}(1-p)^d(2-2p)^k(1-p)^l + E(Y) \qquad (3.4) \\
&= \frac{pq^2}{1-2p}\sum_{d=0}^{h-1}(2-2p)^{h-d}-1)(1-(1-p)^{h-d})(1-p)^d + E(Y) \\
&= \frac{pq^2}{1-2p}\Big(2(2-2p)^h(1-2^{-h}) - \frac{(2-2p)(1-p)^h((2-2p)^h-1)}{1-2p} \\
&\qquad - \frac{1-(1-p)^h}{p} + h(1-p)^h\Big) + E(Y)
\end{aligned}$$

Assuming that is $p < 1 - 1/\sqrt{2} \approx 0.29$ (based on our estimation of Cas9-cutting rates, this seems to be a biologically relevant probability), we have:

$$\begin{aligned}
\lim_{h\to\infty} CoV(X,Y) &= \Big(2 - \frac{2-2p}{1-2p}\Big)\frac{pq^2(2(1-p)^2)^h}{1-2p} \\
&= -\infty
\end{aligned}$$

since $2 < (2-2p)/(1-2p)$ when $p < 0.5$.

$Var(Y)$ can be computed using the same indicators:

$$\begin{aligned}
Var(Y) &= 2\sum_{i,j} E(Y_iY_j) + \sum_i E(Y_i^2) - E(Y)^2 \\
\sum_{i,j} E(Y_iY_j) &= 2^{-2h}\sum_{d=0}^{h-1} 2^d(1-p)^d(\sum_{k=0}^{h-d-1} 2^k(1-p)^k pq * 2^{h-d-k-1})^2 \\
&= \frac{q^2}{4}\sum_{d=0}^{h-1}(\frac{1-p}{2})^d(\frac{1-(1-p)^{h-d}}{p})^2 \\
&= \frac{q^2}{4}\sum_{d=0}^{h-1}(\frac{1-p}{2})^d - \frac{2(1-p)^h}{2^d} + \frac{(1-p)^{2h}}{(2-2p)^d} \\
&= \frac{q^2}{4}\Big(\frac{2(1-(\frac{1-p}{2})^h)}{1+p} - 4(1-p)^h(1-2^{-h}) + \\
&\qquad \frac{(2-2p)(1-p)^{2h}(1-(\frac{1}{2-2p})^h)}{1-2p}\Big)
\end{aligned} \tag{3.5}$$

$$\begin{aligned}
\sum_i E(Y_i^2) &= 2^{-2h}\sum_{d=1}^{h} 2^d(1-p)^{d-1}pq * 2^{2(h-d)} \\
&= \frac{pq}{2}\sum_{d=0}^{h-1}(\frac{(1-p)}{2})^d \\
&= \frac{pq(1-(\frac{1-p}{2})^h)}{1+p}
\end{aligned}$$

Note that if $p < 0.5$, every term in $Var(Y)$ converges to a constant as $h \to \infty$. Thus, if $(1-p)^2 > 0.5$, then as the depth increases, X and Y become exponentially more negatively correlated. This means that for biologically relevant values of p, the frequency of a mutation in the samples is negatively correlated with number of times the mutation occurred, thus justifying the rationale of splitting the sample on more frequently occurring mutations.

Simulation For Tracking the Evolution of a Particular Mutation

To more efficiently simulate the number of occurrences of a particular mutation, we define $\{N_1, N_2, ...N_h\}$ as a Markov chain, where N_t is the number of unmutated cells at generation t, and $N_1 = 1$. Let $A_t \sim Bin(2N_t, p)$ be the number of cells that mutates at generation t, and $B_t \sim Bin(A_t, a)$ be the number of mutated cell that took on the particular state in question. The Markov chain evolves as $N_{t+1} = 2N_t - A_t$. Note that we assume, in this model, that mutation can only occur after cell division. Thus we have $X = \sum_{t=1}^{h} B_t$ and $Y = \sum_{t=1}^{h} 2^{t-h}B_t$.

3.2.14 Assessing the Precision of Greedy Splits.

To assess the precision of greedy splits, we first simulated 100 true phylogenies of 400 cells (without dropout) for all pairs of parameters in $num\_states = \{2, 10, 40\}$ and $p_cut = \{0.025, 0.1, 0.4\}$. For each network, we assessed the precision of the greedy split as follows:

1. We used the criteria $i, j = \arg\max_{i,j} n_{i,j}$ to select the character χ_i and state j to split on (as Cassiopeia-Greedy would do). This group of cells that have a mutation j in character χ_i is called G.

2. For define the a set of n subsets corresponding to cells that inherited the (character, state) pair (i, j) independently using the true phylogenies, and call this set $S = (s_1, s_2, ..., s_n)$ (this corresponds to there being n parallel evolution events for the (character, state) pair (i, j).

3. We presume that the largest group of cells in S is the "true positive" set (let this be defined as $s' = \arg\max_s |s_i|$. We then define the precision P as the proportion of true positives in the set G – i.e. $P = \frac{|s'|}{|G|}$.

3.2.15 Statistics for IVLT Analysis

Meta Purity Statistic

To calculate the agreement between clades (i.e. the leaves below a certain internal node of the tree) and some meta-value, such as the experimental plate from which a sample came from, we can employ a Chi-Squared test. Specifically, we can compute the following statistic: considering some M clades at an arbitrary depth d, we find the count of meta values associated with each leaf in each clade, resulting in a vector of values m_i comprised of these meta-counts for each clade i. We can form a contingency table summarizing these results, T, where each internal value is exactly $m_{i,j}$ - the counts of the meta item j in clade i. A Chi-Squared test statistic can be computed from this table.

To compare across different trees solved with different methods, we report the Chi-Squared Test Statistic as a function of the number of clades, or degrees of freedom of the test.

Mean Majority Vote Statistic

The Mean Majority Vote statistic seeks to quantify how coherent each clade is with respect to its majority vote sample at a give depth. For a given clade with leaves L_i where $|L_i| = n$, where every leaf $l_{i,j}$ corresponds to cell j in clade i has some meta label m_j, the majority vote of the clade is $v = argmax_{m' \in M} \sum_{j \in n} \delta(j, m')$. Here M is the full set of possible meta values and $\delta(m_j, m')$ is an indicator function evaluating to 1 iff $m_j = m'$. The membership of this clade is then simply $\frac{\sum_{j \in n} \delta(m_j, v)}{n}$. Then, the mean membership is the

mean of these membership statistics for all clades at a certain depth (i.e. if the tree were cut at a depth of d, the clades considered here are all the internal nodes at depth d from the root). By definition, this value ranges from $\frac{1}{|M|}$ to 1.0.

As above, to compare across different trees solved with various methods, we report this mean membership statistic as a function of the number of clades.

3.2.16 Triplets Correct Statistic

To compare the similarity of simulated trees to reconstructed trees, we take an approach which compares the sub-trees formed between triplets of the terminal states across the two trees. To do this, we sample $\sim 10,000$ triplets from our simulated tree and compare the relative orderings of each triplet to the reconstructed tree. We say a triplet is "correct" if the orderings of the three terminal states are conserved across both trees. This approach is different from other tree comparison statistics, such as Robinson-Foulds [151], which measures the number of edges that are similar between two trees.

To mitigate the effect of disproportionately sampling triplets relatively close to the root of the tree, we calculate the percentage of triplets correct across each depth within the tree independently (depth measured by the distance from the root to the Latest Common Ancestor (LCA) of the triplet). We then take the **average** of the percentage triplets correct across all depths. To further reduce the bias towards the few triplets that are sampled at levels of the tree with very few cells (i.e. few possible triplets), we modify this statistic to only take into account depths where there at least 20 cells. We report these statistics without this depth threshold in Additional file 1: Fig S8.

3.2.17 Allelic and Phylogenetic Distances

For the analysis in Figure3.4, we define two metrics to capture cell-to-cell similarity: a normalized allelic distance and normalized phylogenetic distance. The normalized allelic distance is calculated as follows: for all target sites $\chi_m \in \{\chi_1, ..., \chi_M\}$ in a pair of cells c_i and c_j:

1. if state in χ_m is the same in c_i and c_j, continue
2. else if state in χ_m is 0 or missing in either c_i or c_j increment the allelic distance by 1
3. else increment the allelic distance by 2

Finally, the allelic distance for a pair of cells is normalized by $2 * M$, where M is the number of target sites.

The phylogenetic distance is defined as simply the number of mutations separating the two cells c_i and c_j as implied by the tree (i.e. the number of mutations along the branches for the shortest path separating c_i and c_j). The normalized phylogenetic distances is this distance, divided by the diameter (defined as the maximum phylogenetic distance between all pairs of cells) of the tree.

3.2.18 Bootstrapping Analysis

Bootstrapping was done using a custom function for sampling M target sites (i.e. characters) from an $N \times M$ character matrix with replacement and reconstructing trees from these bootstrapped samples. After performing tree inference, we collapsed "singles" using the collapse.singles function in the R package "ape". For the purposes of our robustness analysis, we sampled $B = 100$ trees from $N = 10$ simulated trees and used the Transfer Bootstrap Expectation (TBE) [118] statistic for assessing branch support for each clade as implemented in Booster (available at https://github.com/evolbioinfo/booster/).

3.2.19 Application of Camin-Sokal

We applied Camin-Sokal using the "mix" program in PHYLIP [53] as done for reconstructions for McKenna et al [129] and Raj et al [147]. To use "mix" we first factorized the characters into binary ones (thus ending up with $\sum_i s_i$ binary characters total, where s_i is the number of states that character i presented). Then, we one-hot encoded the states into this binary representation where every position in the binary string represented a unique state at that character. We thus encoded every cell as having a 1 in the position of each binary factorization corresponding to the state observed at that character. If the cell was missing a value for character i, the binary factorization of the character was a series of '?' values (which represent missing values in PHYLIP "mix") of length s_i. Before performing tree inference, we weighted every character based on the frequency of non-zero (and non-missing values) observed in the character matrix. After PHYLIP "mix" found a series of candidate trees, we applied PHYLIP "consense" to calculate a consensus tree to then use downstream.

3.2.20 Application of Neighbor-Joining

We used Biopython's Neighbor-Joining procedure to perform all neighbor joining in this manuscript. We begun similarly to the Camin-Sokal workflow, first factorizing all of the characters into a binary representation. Then, we applied the Neighbor-Joining procedure using the "identity" option as our similarity map.

3.2.21 Application of Cassiopeia

Reconstruction of simulated data

We used Cassiopeia-ILP with a maximum neighborhood size of $10,000$ and time to converge of $12,600$s. Cassiopeia-Hybrid used a greedy cutoff of 200, a maximum neighborhood size of 6000 and 5000s to converge. Cassiopeia-Greedy required no additional hyperparameters. Simulations with priors applied the exact prior probabilities used to generate the simulated trees.

Reconstruction of *in vitro* clones

. For both Cassiopeia-Hybrid with and without priors, we used a cutoff of 200 cells and each instance of Cassiopeia-ILP was allowed 12,600s to converge on a maximum neighborhood size of 10,000. Cassiopeia-ILP was applied with a maximum neighborhood size of 10,000 and a time to converge of 12,600s.

Simulation of Target Site Sequences for Alignment Benchmarking

To determine an optimal alignment strategy and parameters for our target site sequence processing pipeline, we simulated sequences and performed a grid search using Emboss's Water algorithm (a local alignment strategy). We simulated 5,000 sequences. For each sequence, we begun with the reference sequence and subjected it to multiple rounds of mutagenesis determined by a Poisson distribution with $\lambda = 3$, and a maximum of 5 cuts. During each "cutting" event, we determined the outcomes as follows:

1. Determine the number of Cas9 proteins localizing to the target site in this iteration, where $n_{cas9} \sim min(3, Pois(\lambda = 0.4))$.

2. Determine the site(s) to be cut by choosing available sites randomly, where the probability of being chosen is $p = \frac{1}{n_{uncut}}$ and n_{uncut} is the number of sites uncut on that sequence.

3. If $n_{cas9} = 1$, we determined the type of the indel by drawing from a Bernoulli distribution with a probability of success of 0.75 (in our case, a "success" meant a deletion and a "failure" meant an insertion). We then determined by drawing from a Negative Binomial Distribution as so: $s \sim min(30, max(1,\ NB(0.5, 0.1)))$. In the case of an insertion, we added random nucleotides of size s to the cut site, else we removed s nucleotides.

4. In the case of $n_{cas9} \geq 2$, we performed a resection event where all nucleotides between the two cut sites selected were removed.

5. After a cut event, we appended the result of the Cas9 interaction to a corresponding CIGAR string

Our Water simulations were exactly 300bp, possibly extending past the Poly-A signal, as would be the case reading off a Nova-seq sequencer.

Upon simulating our ground truth dataset, we performed our grid search by constructing alignments with Water with a combination of gap open and gap extension penalties. We varied the gap open penalties between 5 and 50 and gap extension penalties between 0.02 and 2.02.

To score resulting alignments, we compared the resulting CIGAR string to our ground truth CIGAR string for each simulated sequence. To do so, we first split each cigar string

into "chunks", corresponding to the individual deletions or insertions called. For example, for some CIGAR string $40M2I3D10M$, the chunks would be $40M$, $2I$, $3D$, and $10M$. Then, beginning with a max score of 1, we first deducted the difference between the number of chunks in the ground truth and the alignment. Then, in the case where the number of chunks were equal between ground truth and alignment, we deducted the percent nucleotides that differed between CIGARs. For example, if the ground truth was $100M$ and the alignment gave $95M$, the penalty would be 0.05.

To find the optimal set of parameters, we selected a parameter pair that not only scored very well, but also located in the parameter space where small perturbations in gap open and gap extension had little effect.

Simulation of Lineages for Algorithm Benchmarking

We simulated lineages using the following parameters:

1. The number of characters to consider, C
2. The number of states per character, S
3. The dropout per characters, d_c $\forall c \in C$
4. The depth of the tree (i.e. the number of binary cell division), D
5. The probability that a site can be mutated, p. This is a general probability of cutting
6. The rate at which to subsample the data at the end of the experiment, M

To simulate the tree, we begin by first generating the probability of each character mutating to a state, here represented as $p_c(0, s), \forall s \in S$. In order to do this, we fit a spline function to the inferred prior probabilities from the lineage tracing experiment. (refer to the section entitled "Determining Prior Probabilities of Indel Formation" for information on how we infer prior probabilities). We then draw S values from this interpolated distribution. We then normalize these mutation rates to sum to p, therefore allowing in general a p probability of mutating a character and $1 - p$ probability of remaining uncut. In the case of the "State Distribution" simulations (Additional file 1: Fig S13), we say that p_c is distributed as:

$$p_c = \theta * Unif(0, 1) + (1 - \theta) * F'(x)$$

where $F'(x)$ is the interpolated empirical distribution and θ is the mixture component.

Then, we simulate D cell divisions, where each cell division consists of allowing a mutation to take place at each character with probability p. In the case a mutation takes place, we choose a state to mutate to according to their respective probabilities. Importantly, once a character has been mutated in a cell, that character cannot mutate again.

At the end of the experiment, we sample M percent of the cells resulting in $2^D * M$ cells in the final lineage.

We find that this method for simulating lineages (in particular the method for generating a set of priors on how likely a given state is to form) is able to closely recapitulate observed lineages (Additional file 1: Fig S6).

Metrics for Comparing Simulations to Empirical Data

We used three metrics of complexity to compare simulated clones to real clones:

- *Minimum Compatibility Distance*: For every pair of character, we define the Minimum Compatibility Distance as the minimum number of cells to be removed to obtain compatibility (Def. 26).
- *Number of Observable States per Cell*: The number of non-zero or non-missing values for each cell, across all characters (i.e. the amount of data that can be used for a reconstruction, per cell).
- *Number of Observable States per Character*: The number of non-zero or non-missing values across for each character, across all cells.

Parallel Evolution Simulations for Greedy Benchmarking

As shown above, our greedy approach should accurately reconstruct a lineage if a perfect phylogeny exists. In order to better quantify how much our greedy algorithm's performance is affected by parallel mutations, we decided to simulate "near perfect phylogenies", whereby we first began by simulating a perfect phylogeny, and afterwards introduced double mutated characters.

Specifically, we begin by simulating perfect phylogenies with $40 - k$ characters. We then fix a depth, d, and sample a node from said depth. We choose two grandchildren randomly from this node (one from each child) and introduce the same mutation on each of the edges from each child to grandchild, thereby violating the perfect phylogeny. We repeat this process k times. This thus creates an analysis, as presented in Additional file 1: Fig S5, whereby accuracy can be evaluated as a function of both depth of parallel evolution, d, and the number of events that occurred, k.

Simulation of "Base Editor" Technologies

We used the simulation framework described above to simulate base-editor technologies. To explore the trade off between the number of states and the number of characters, we simulated trees with $40, 50, 80$, and 100 characters while maintaining the product of characters and states equal at 400 (thus we had trees of $10, 8, 5$, and 4 states per character, respectively). The dropout per character was set to 10%, the mutation rate per character was set to 1.04% (a previously observed mutation rate [82]), and 400 cells were sampled from a tree of depth 10. For each character/state regime, we generated 10 trees for assessing

the consistency of results. We use a negative binomial model ($\sim NB(5, 0.5)$) as the editing outcome distribution (i.e. state distribution).

Simulation of "Phased Recorder" Technologies

To simulate the phased recorder, we used 5 different experiments varying mutation rates across 50 characters and 10 states per character. In each experiment, we chose a mutation rate for each character from one of 10 regimes, each differing in their relationship to the base mutation rate p_0. To systematically implement this, mutation rate for χ_i is described as such:

$$m_i = p_0 * (1 + e_j * \lfloor \frac{i}{5} \rfloor)$$

where $p_0 = 0.025$ and e_j is a experiment scalar in $\mathbf{e} = \{0, 0.05, 0.1, 0.25, 0.5\}$. This means that for characters $1 - 5$, $m_i = p_0$, for characters $6 - 10$, $m_i = e_j p_0$, for characters $11 - 15$, $m_i = 2e_j p_0$, etc. To summarize each experiment, we provide the ratio between the maximum and minimum mutation rates, which is by definition $1+10r_j$ (because we had 50 characters). We compare two models of indel formation rates - the first being a negative binomial model ($\sim NB(5, 0.5)$), and the second being the spline distribution fit from empirical data.

We simulated 10 trees per regime and reconstructed trees with Cassiopeia with and without priors.

3.2.22 Reconstructions of GESTALT Datasets

We downloaded data corresponding to the original GESTALT study [129] and the more recent scGESTALT study from https://datadryad.org/resource/doi:10.5061/dryad.478t9 and GSE105010, respectively. We created character matrices for input into Cassiopeia by creating pivot tables relating each cell the observed indel observed at each one of the 10 tandem sites on the GESTALT recorder. We then reconstructed trees from these character matrices using one of five algorithms: Camin-Sokal (used in the original studies), Neighbor-Joining, Cassiopeia's greedy method, Cassiopeia's Steiner Tree method, and Cassiopeia's hybrid method.

For each reconstruction, we record the parsimony of the tree, corresponding to the number of mutations that are inferred along the reconstructed tree. We display these findings in Figure 3.6a, where we have Z-normalized the parsimonies across the methods for each dataset to enable easier visualization of relative performances.

3.2.23 Visualization of Trees

To visualize trees we use the iTOL interface [38]. Colors in the heatmap denote a unique mutation, gray denotes an uncut site, and white denotes dropout.

3.2.24 Supplementary Figures

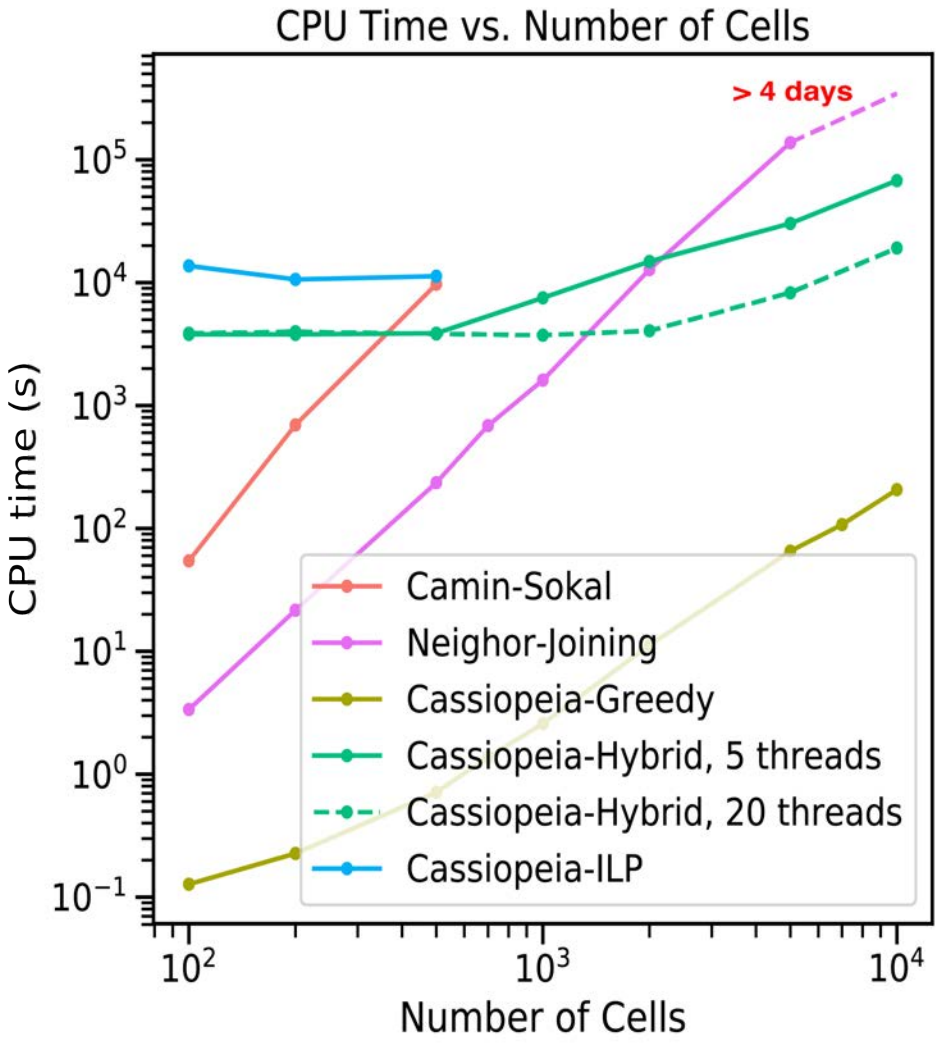


Figure 3.7: **Time complexity of lineage reconstruction approaches.** Time complexity, as measured in seconds, of each algorithm tested in this manuscript is compared using simulated datasets ranging from 100 cells to 10,000 cells. Default settings for the simulations were used (0.025 mutation rate, 40 characters, 10 states, and 0.18 median dropout rate). Cassiopeia was tested using default parameters of a maximum neighborhood size of 3000, time to converge of one hour, and a greedy cutoff of 200 cells. Cassiopeia was tested using 5 threads and 20 threads, illustrating the advantage of parallelizing the reconstruction algorithm. ILP, which was only run until 500 cells due to the infeasibility of running on larger datasets, was allowed 10000s to converge on a maximum neighborhood size of 20,000 (the default settings). Neighbor-Joining could not reconstruct a tree for 10,000 cells within 4 days when the reconstruction was terminated.

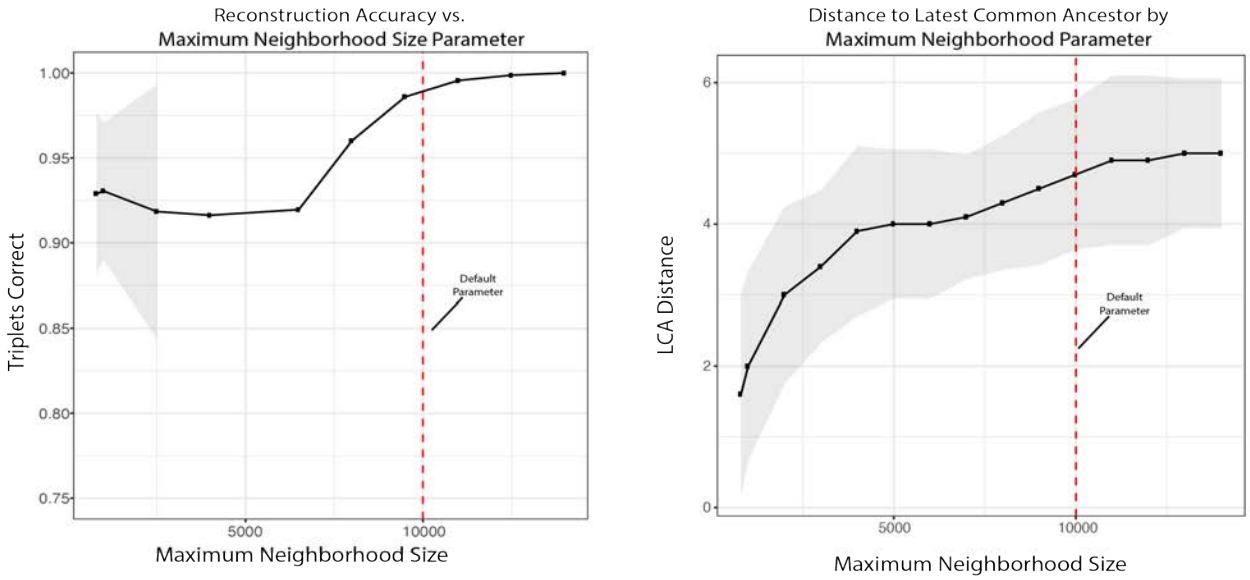


Figure 3.8: **Evaluation of the stability of the maximum neighborhood size parameter.** The maximum neighborhood size is a central parameter provided by the user when inferring the potential graph necessary as input to the Steiner-Tree solver (see methods). Here, we benchmark the stability of solutions with respect to several maximum neighborhood sizes using 10 trees with default parameters (40 characters, 40 states, 2.5% per-character mutation rate, depth of 11, and an average dropout rate of 17% per character). We quantify both the reconstruction accuracy with respect to the reconstructions found with the largest maximum neighborhood size (14,000 nodes) which displays a saturation at around 9,000 nodes. To provide intuition for the accuracy of the potential graph (represented as the maximum distance to the 'latest common ancestor' (LCA) which is dynamically solved for, given a maximum neighborhood size) we display the LCA allowed for each maximum neighborhood size parameter. In both figures, we display lines connecting the mean values; shaded regions are the standard deviation of the measurements across the 10 replicates.

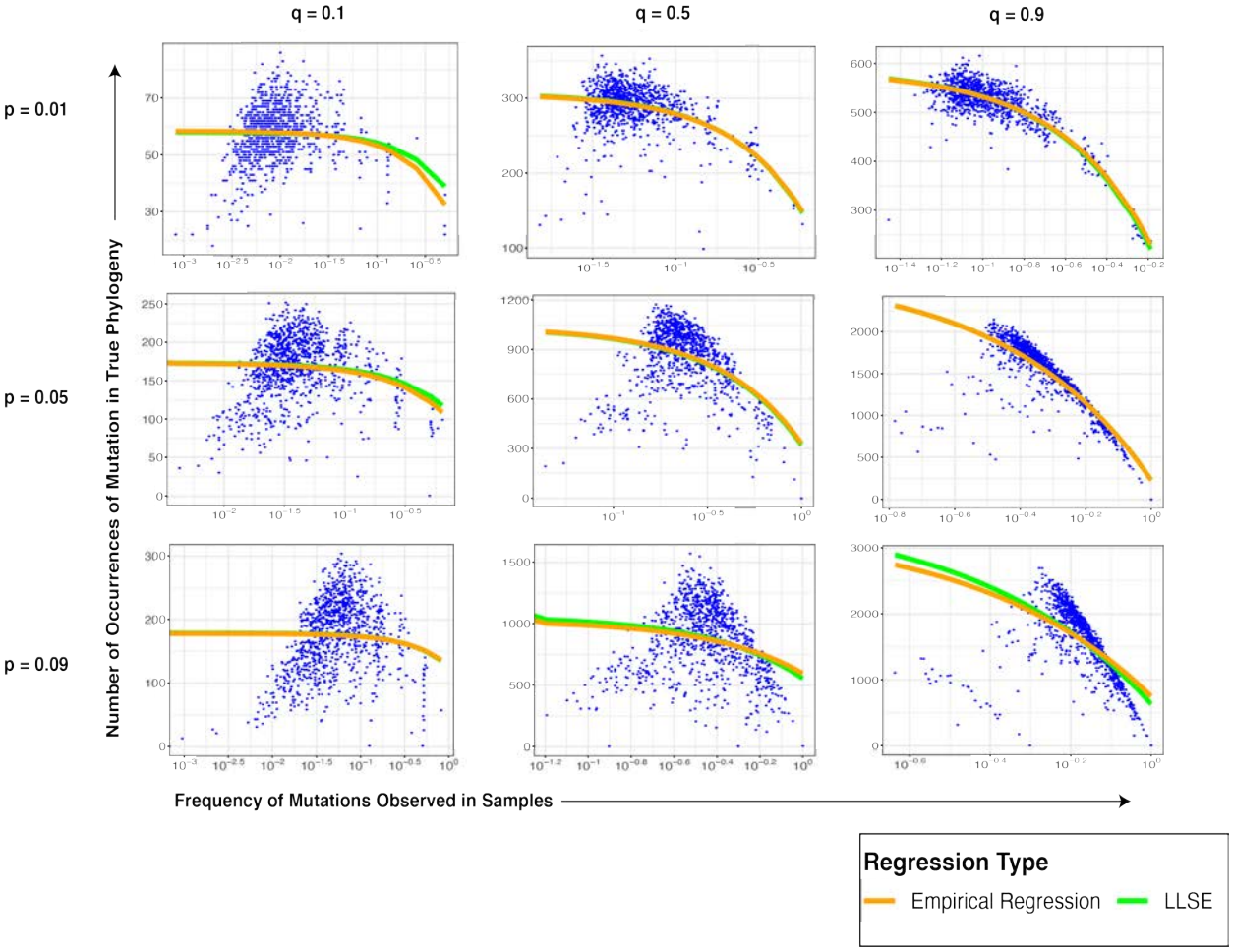


Figure 3.9: **Observed Frequency of Mutations is Measure of True Mutation Count.** The true number of occurrences of a mutation is estimated well by the observed frequency at leaves. We use a Linear Least Squares Estimate to quantify the relationship between the expected number of times a mutation occurred given the observed frequency at the leaves (Eq. 1). Using various rates for character and indel mutation rates (p and q, respectively) we show that this relationship is negative (i.e. greater observed frequencies tend to correspond to mutations that occurred few times near the top of the phylogeny) for a range of biologically-relevant values.

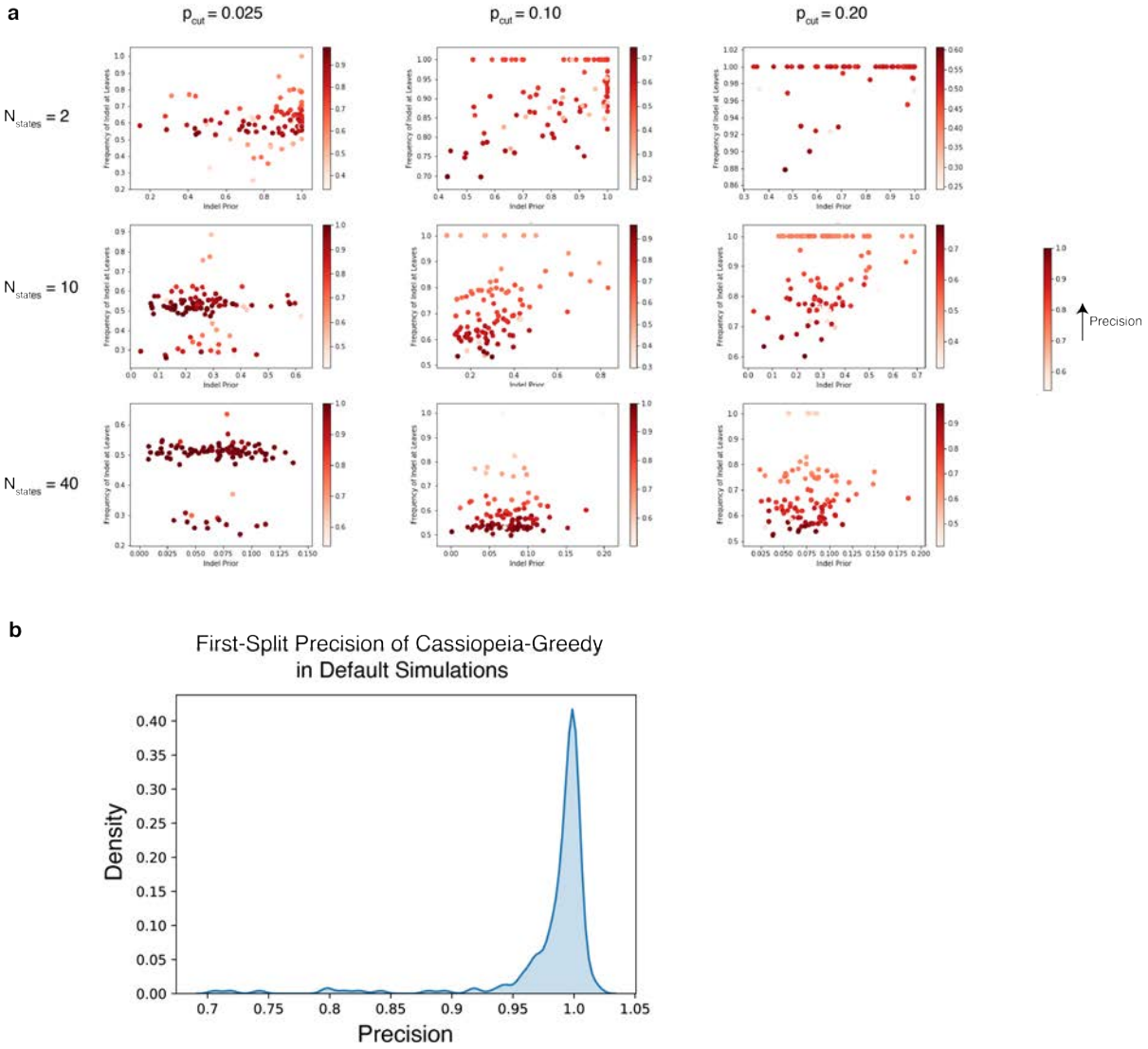


Figure 3.10: **Precision of Cassiopeia-Greedy First Split.** (a) The precision of greedy splits of 400 cells was measured with varying mutation rates and states per character, wihtout dropout. For each pair of parameters (number of states and mutation rate), we measure precision as a function of the conditional probability of the selected (character, state) pair and the frequency of that mutation observed in the 400 cells. (The conditional probability for state j, $q(j)$ is defined as $Pr(\chi \rightarrow j | \chi \ \ mutates)$). Precision was defined as the proportion of true positives in the greedy split (see Methods). Each point indicates a replicate (100 per plot) and the heat represents the precision. (b) The density histogram (smoothed using a kernel density estimation procedure) of all first-split precision statistics from Cassiopeia-Greedy on default simulations (i.e. 40 characters, 40 states, 2.5% mutation rate, 11 generations, 400 cells, and 18% dropout rate). We measured a median precision of 0.99 across all default simulations.

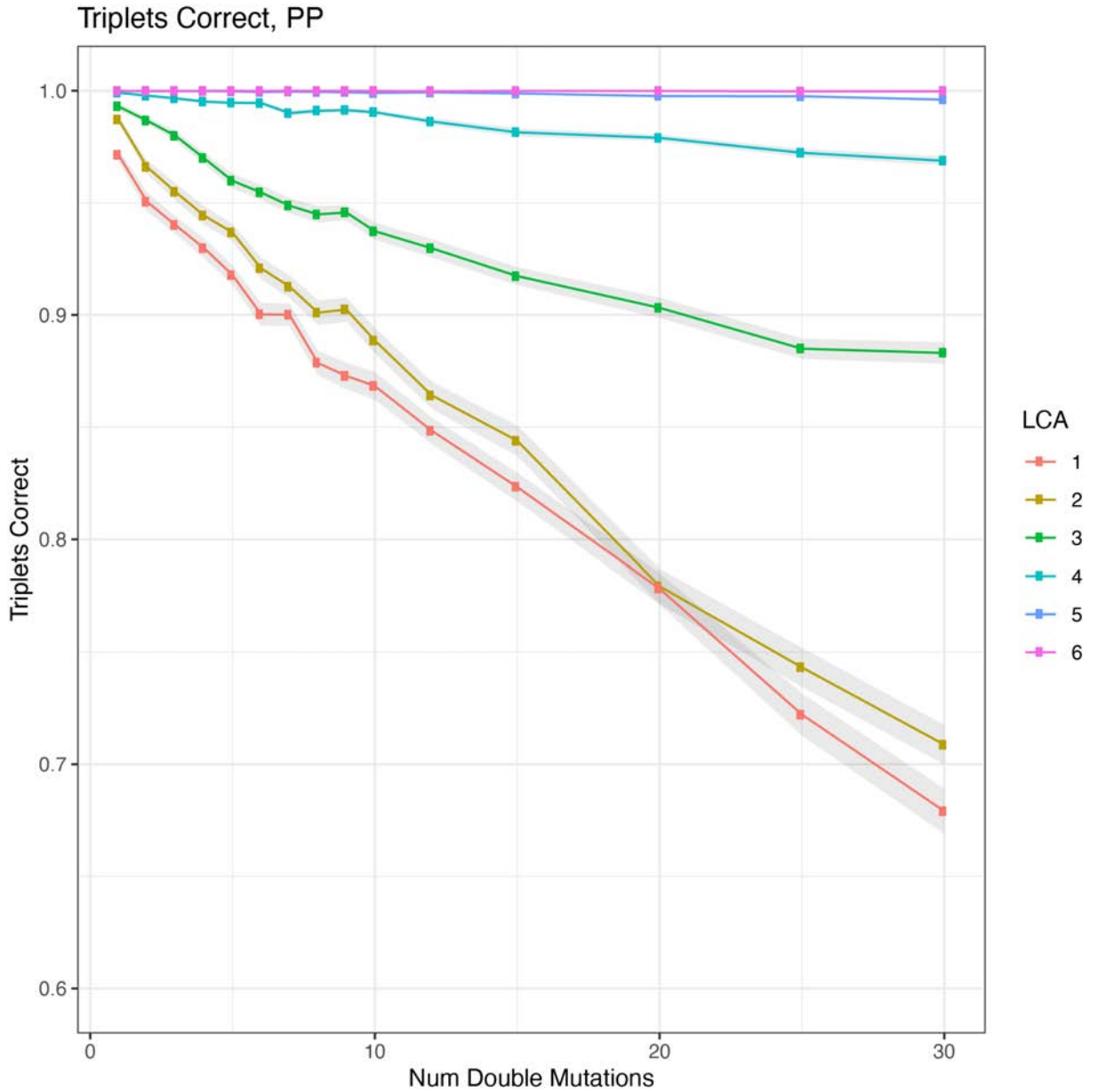


Figure 3.11: **Benchmarking of parallel evolution on the greedy heuristic.** The greedy heuristic, inspired by algorithms to solve the case of perfect phylogeny (see methods), is impacted by two factors: (1) the number of parallel evolution events (i.e. the same mutation occurs more than once in the experiment) and (2) the depth from the root these mutations occur at. Here, each line represents a series of experiments increasing the number of 'double mutations' (i.e. the simplest case of parallel evolution where a mutation occurs exactly twice) where the 'latest common ancestor' (LCA) is a set depth from the root.

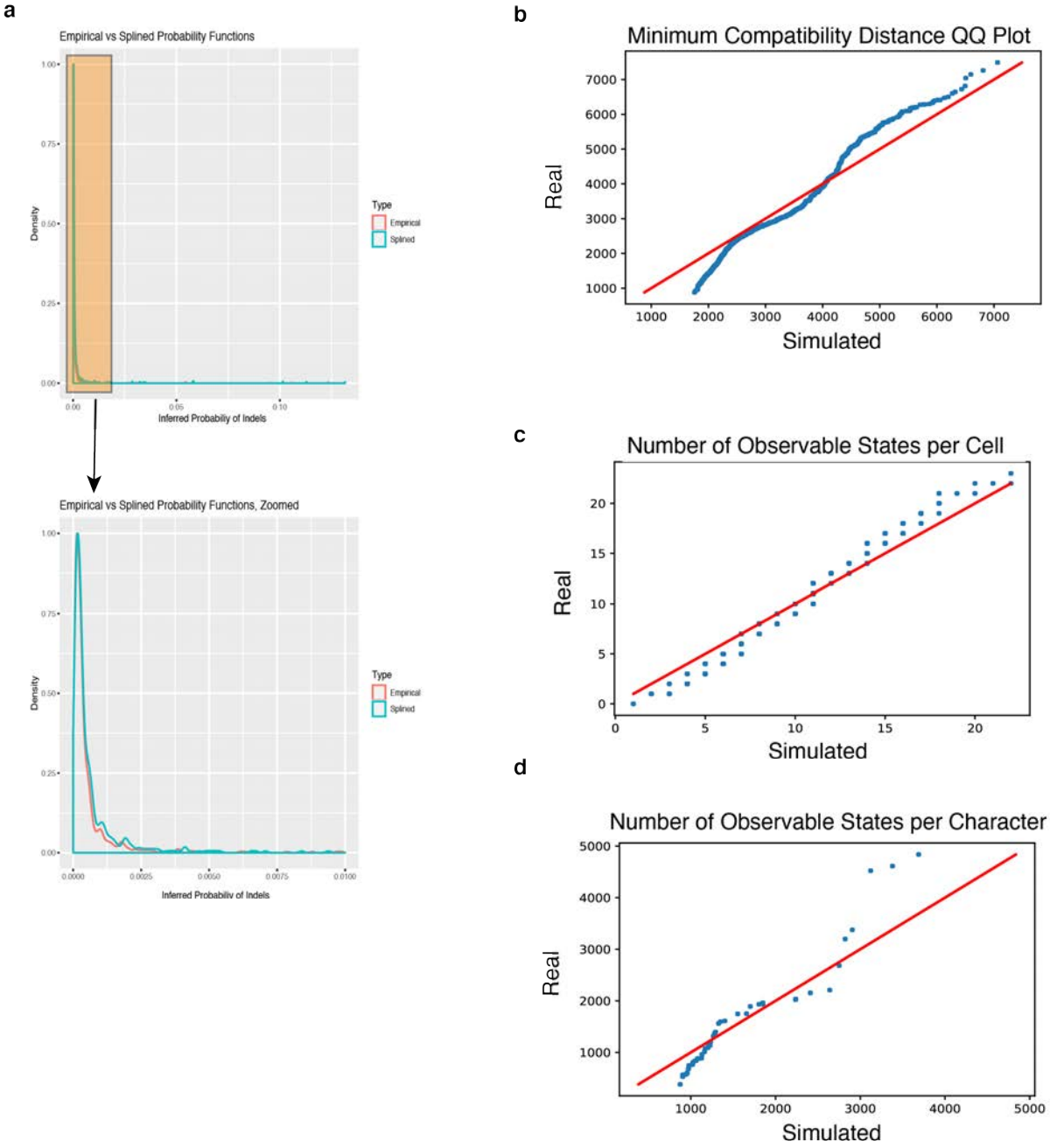


Figure 3.12: **Determination of mutation rates used in simulation.** We use an interpolation of the empirical indel distribution as input for the conditional probability of a state arising given a mutation. (a) A comparison of the empirical and 'splined' indel distributions; a zoomed in version is provided for comparison at low probabilities. (b-c) A comparison of three metrics between an observed clone (clone 3) and a simulated clone using inferred parameters. We used the number of character, states, per-character mutation rate, and dropout probabilities inferred from the empirical data; the indel formation rates were calculated using a polynomial spline function. (b) measures the 'minimum compatibility distance' for all pair-wise character combinations (see methods). (c) compares the number of observable states per cell. (d) compares the number of observable states per character.

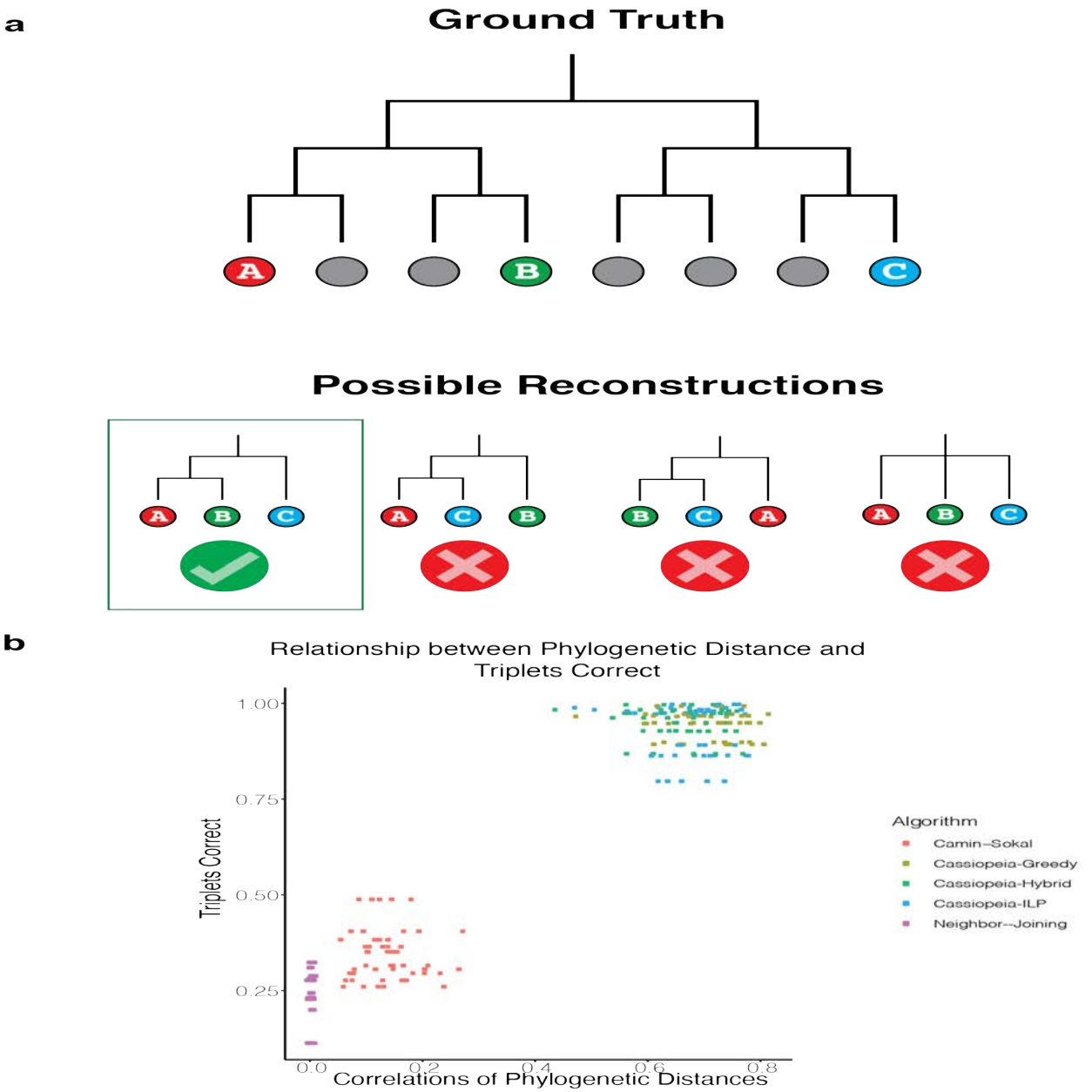


Figure 3.13: **Triplets Correct Statistic.** (a) Schematic for the Triplets Correct statistic, the combinatorial metric used to compare between trees. In this metric, we compare the relative orderings of three leaves between two trees (e.g. the "Ground Truth" and a reconstruction). There are four possible ways that a triplet could be ordered here, based on the relationship between each leaf and the Latest Common Ancestor (LCA) of the triplet. The statistic tallies the number of correct triplets and reports this value weighted by the depth of the LCA from the root. Importantly, this statistic is designed to avoid concerns of inappropriately weighting early splits as these might dominate the statistic. Specifically, the triplets are stratified in accordance to the depth of the LCA and the triplets correct is reported as an average across all LCA depths. This way, LCAs near the root will not dominate the score. (b) A comparison between the triplets correct statistic and the phylogenetic distance correlation (defined as the correlation of node-node distances between a simulated and reconstructed tree; see Methods) where we observe a Pearson correlation of 0.96.

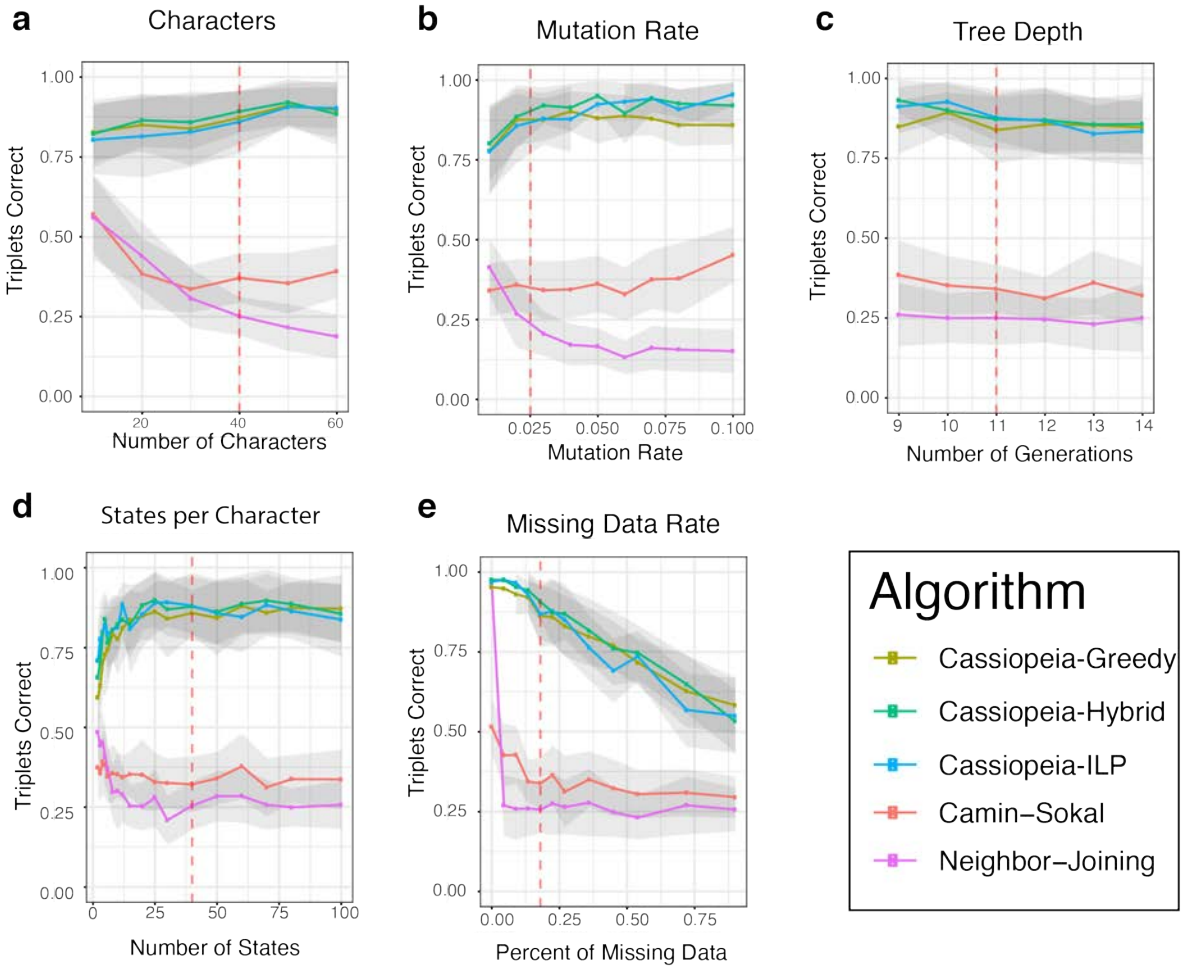


Figure 3.14: **Unthresholded Triplets Correct.** The Triplets Correct statistic reported for synthetic benchmarks presented in Figure 2 without removing triplets whose LCA-depth was sampled deeply enough (by default, a given triplet at depth D is only considered if a sufficient number of triplets at depth D is observed). Here, the effective threshold is 0.

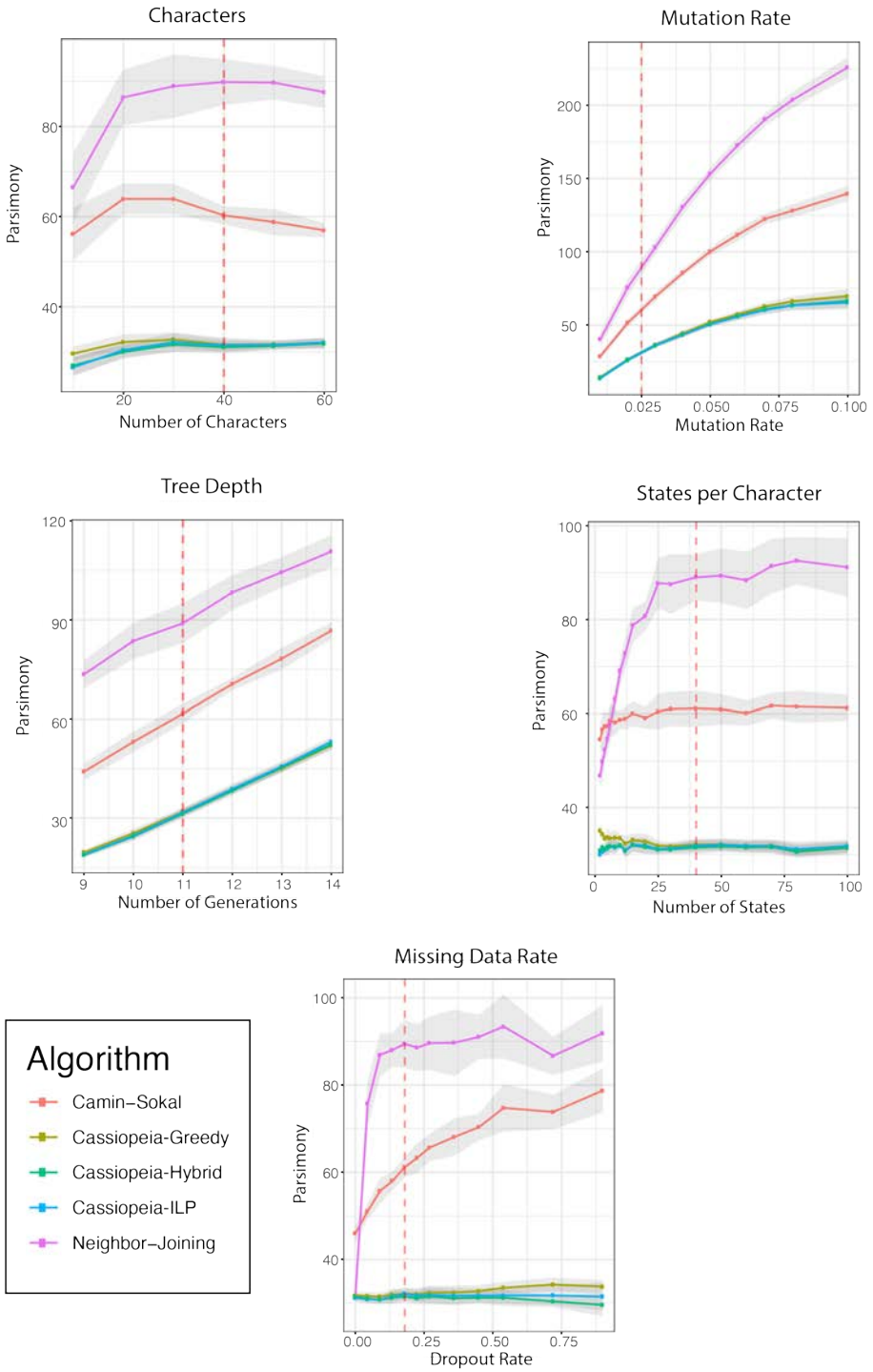


Figure 3.15: **Parsimony of reconstructed trees of 400 cell simulated datasets.** Parsimony scores (or number of evolutionary events) for each reconstructed network presented in Figure 2 were calculated and compared across phylogeny reconstruction methods. Results are presented for the number of characters, the mutation rate, tree depth, number of states and dropout rate for all five algorithms used in this study. Standard error is represented by shaded area.

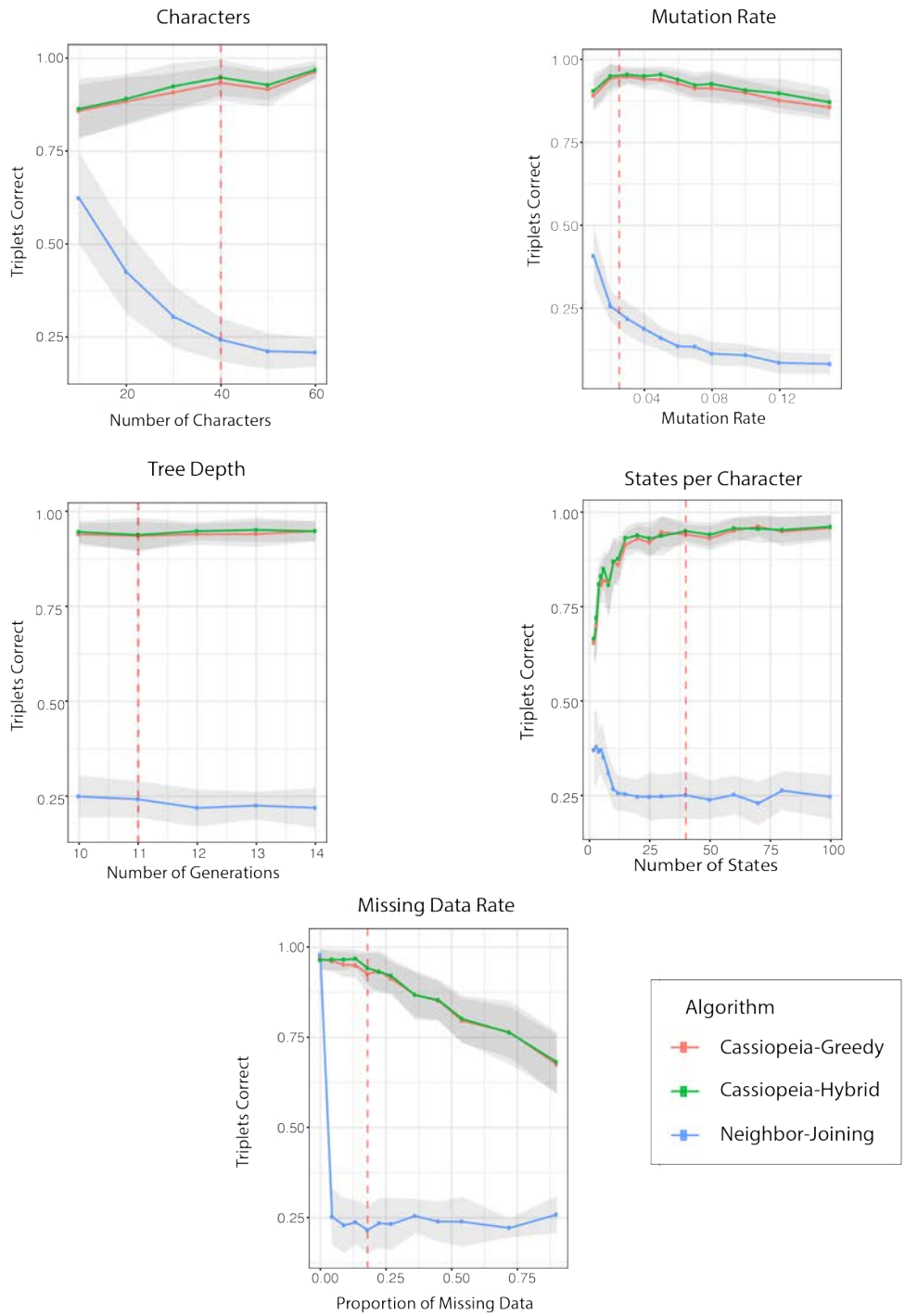


Figure 3.16: **Benchmarking of lineage tracing algorithms on 1000 cell synthetic datasets.** Phylogeny reconstruction algorithms were benchmarked on simulated trees consisting of 1,000 cells. The number of characters, character-wise mutation rate, length of experiment or tree depth, number of states, and dropout rate were tested. Due to scalability issues, only greedy, hybrid, and neighbor-joining were tested. Standard error is represented by shaded area.

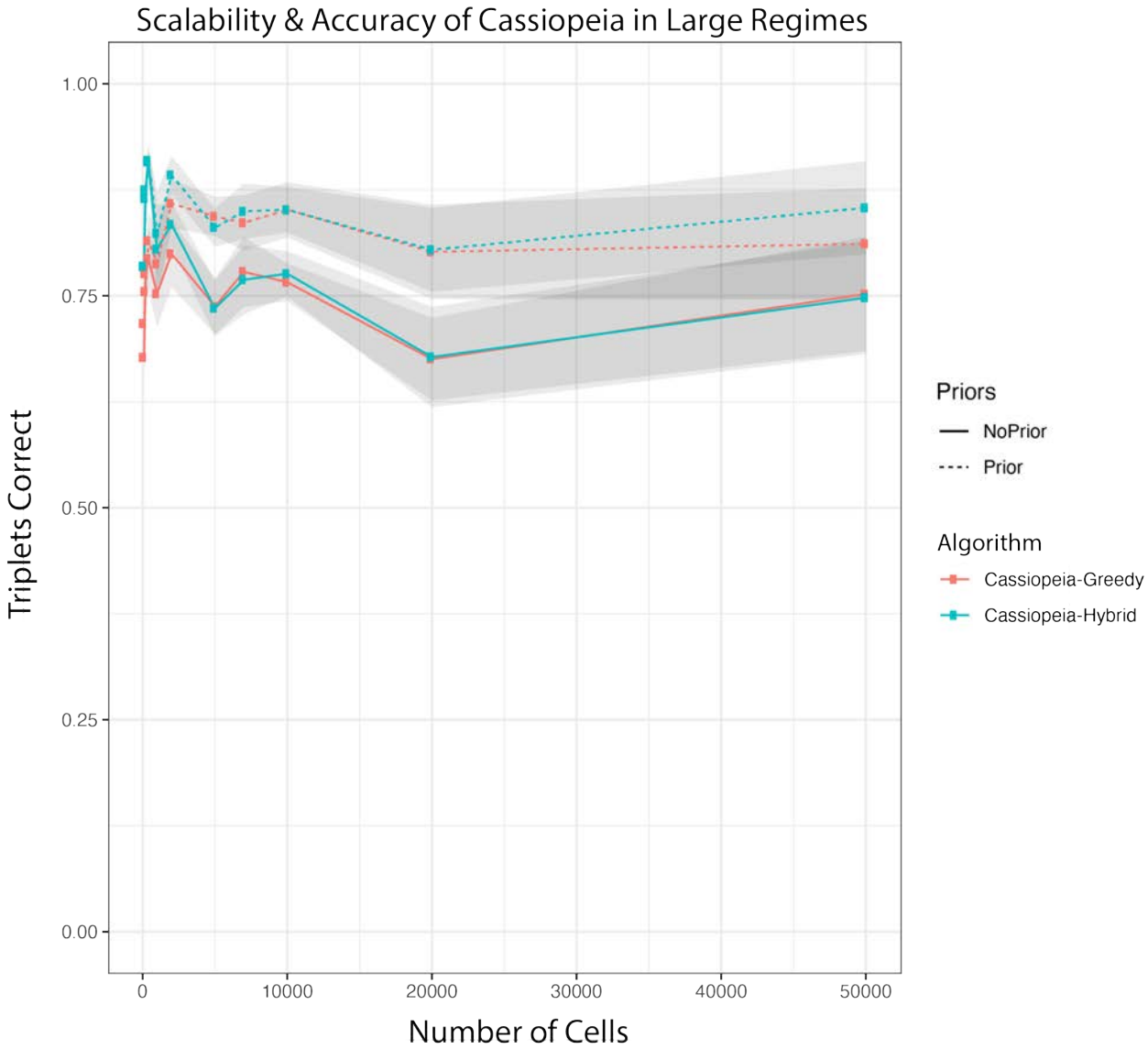


Figure 3.17: **Benchmarking of greedy and hybrid algorithms on large experiments.** Triplets correct is used to measure the accuracy of reconstructions using both hybrid and greedy algorithms on large trees (up to $50,000$ cells). Of note, hybrid and greedy have comparable results on larger trees, which remain accurate even in these massive regimes. In addition, the knowledge of prior probabilities of particular states confers a large increase in accuracy.

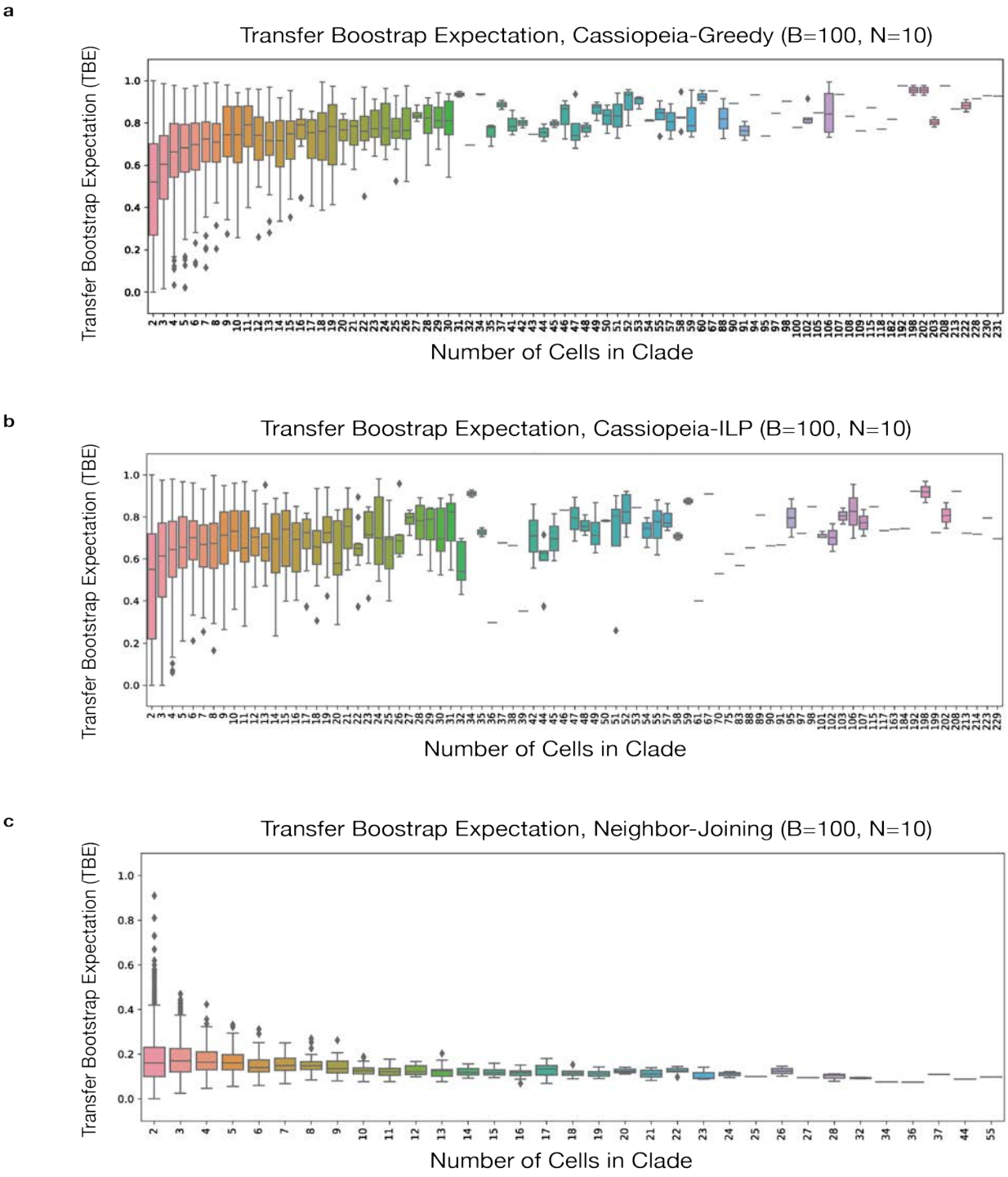


Figure 3.18: **Bootstrapping analysis of Cassiopeia and Neighbor-Joining with the Transfer Bootstrap Expectation statistic.** Bootstrap analysis of robustness for Cassiopeia-Greedy (a) , -ILP (b) , and Neighbor-Joining (c). 100 bootstrap samples (B = 100) were taken for 10 simulated trees (N = 10) by sampling characters with replacement and each matrix was used for reconstruction by each of the tree algorithms. The Booster software [61] was used to assess robustness of each clade in the original reconstruction, as measured with the Transfer Bootstrap Expectation (TBE) statistic. The distribution of TBE's is shown for each algorithm as a function of the size of the clade (i.e. a clade with two leaves underneath it will be of size 2

.

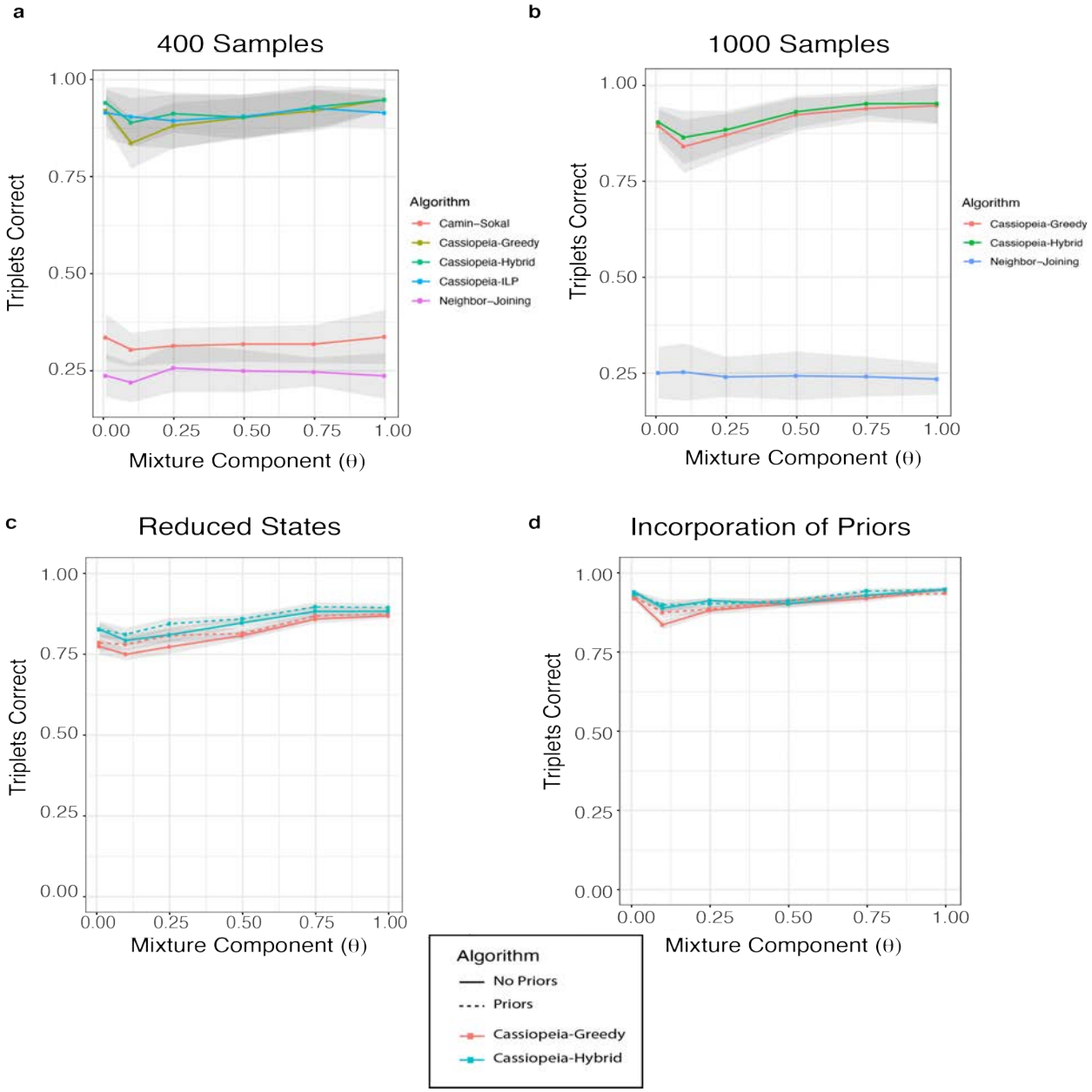


Figure 3.19: **Reconstruction accuracy under over-dispersed state distributions.** The effect of the indel distribution (i.e. the relative propensity for a given indel outcome) was explored in various regimes using a mixture model. Here, the mixture model consisted of mixing the inferred indel distribution with a uniform distribution between 0 and 1.0 with some probability θ (i.e. when $\theta = 1.0$, the indel distribution was uniform). In all simulations, we used default parameters for the simulated trees unless stated otherwise (40 characters, 40 states, depth of 11, median dropout rate of 17%, and a character mutation rate of 2.5%). (a) displays the results of all five algorithms over 400 samples. (b) displays results for simulations over 1000 samples for hybrid, greedy, and neighbor-joining methods. (c) Simulations for 400 samples using 10 states rather than 40 states per character. Dashed lines represent reconstructions performed with priors. (d) Simulations over 400 samples and 40 states, comparing results with and without priors. Dashed lines represent reconstructions performed with priors.

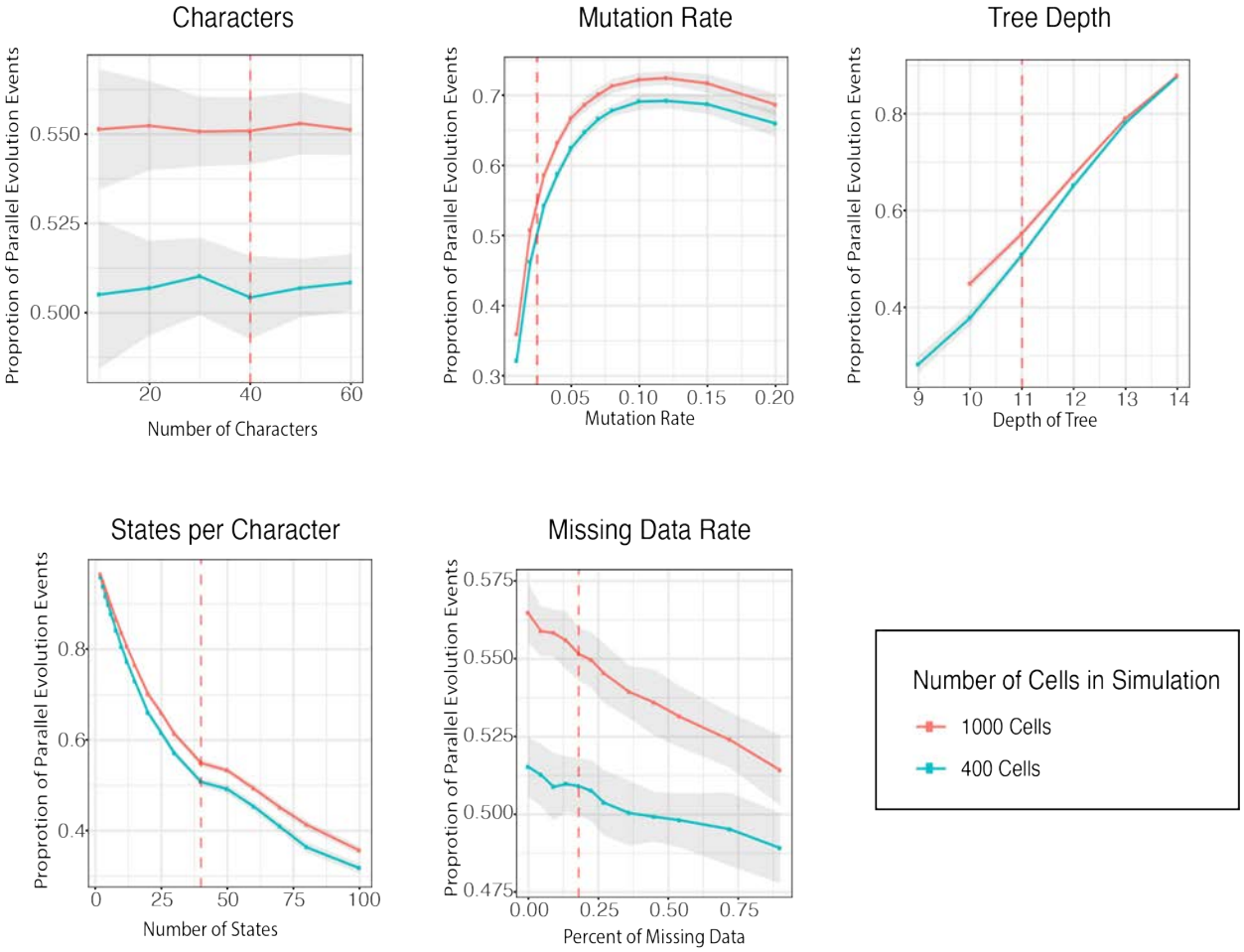


Figure 3.20: **Observed Proportion of Parallel Evolution in Simulations.** Inferred proportion of parallel evolution, as defined by teh proportion of mutations that are observed more than once in a given tree, for the simulations presented in Figure 2 and Fig S10.

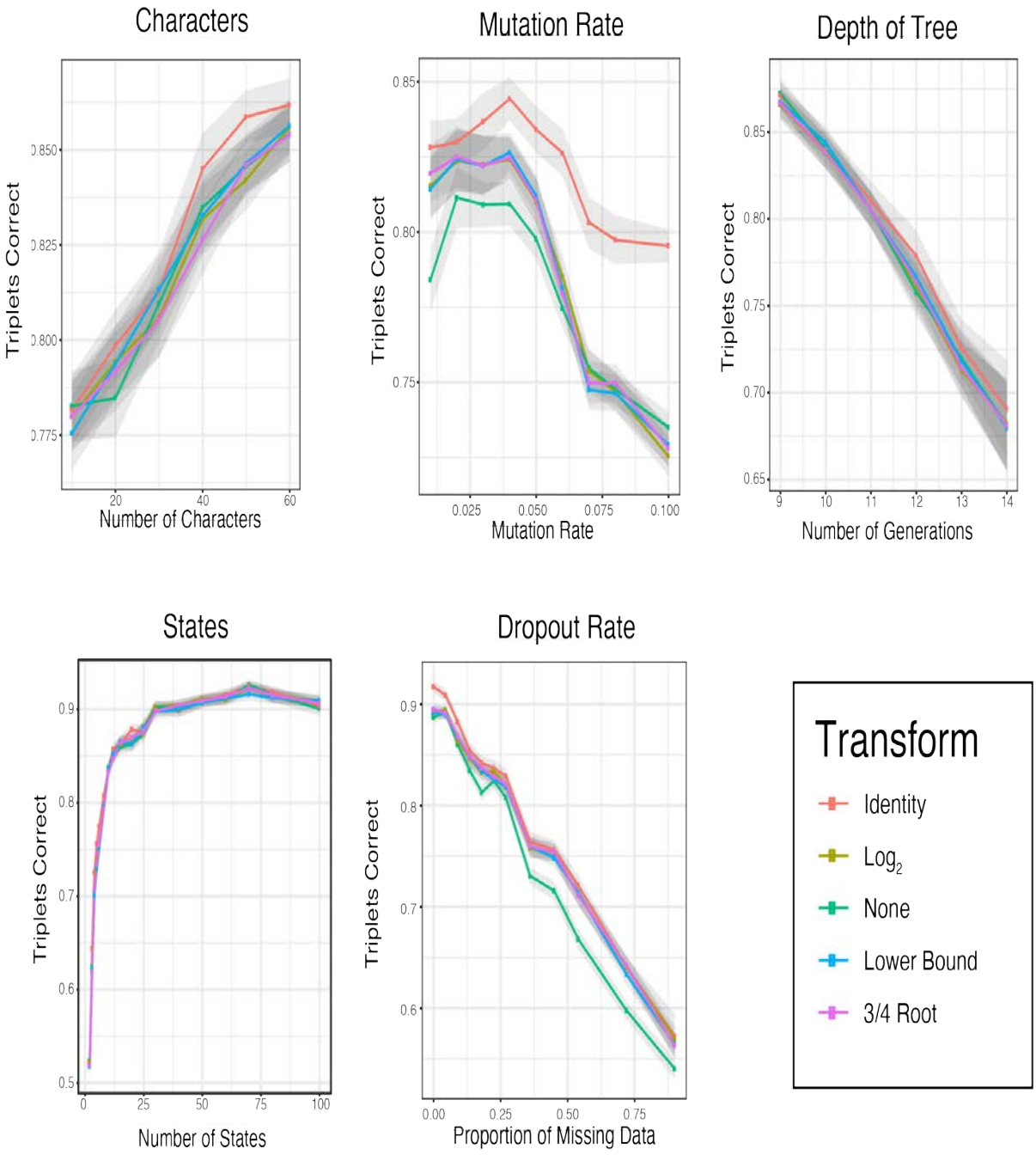


Figure 3.21: **Determination of the indel prior transformation function.** The effect of incorporating the prior probabilities of mutation events into the greedy algorithm is explored using synthetic datasets. The exact mutation probabilities used for simulations are used during reconstruction (i.e. the mutations drawn during simulation). Five possible transformations $f(n_{i,j})$, representing an approximation of the future penalty of not choosing this mutation (see methods) were tested for incorporation with the priors. The transformations were: (i) Identity ($f(n_{i,j}) = n_{i,j}$), (ii) Log2 ($f(n_{i,j}) = log_2(n_{i,j})$), (iii) None ($f(n_{i,j}) = 1$), (iv) Lower Bound ($f(n_{i,j}) = min(n_{i,j}, \frac{N}{20.0})$), and (v) $\frac{3}{4}$ root ($f(n_{i,j}) = (n_{i,j})^{\frac{3}{4}}$). $n_{i,j}$ denotes the number of cells which report the mutation j in character i and N is the total number of samples. To test these transformations, we evaluated the resulting tree accuracy via Triplets Correct. Standard error is represented by shaded area.

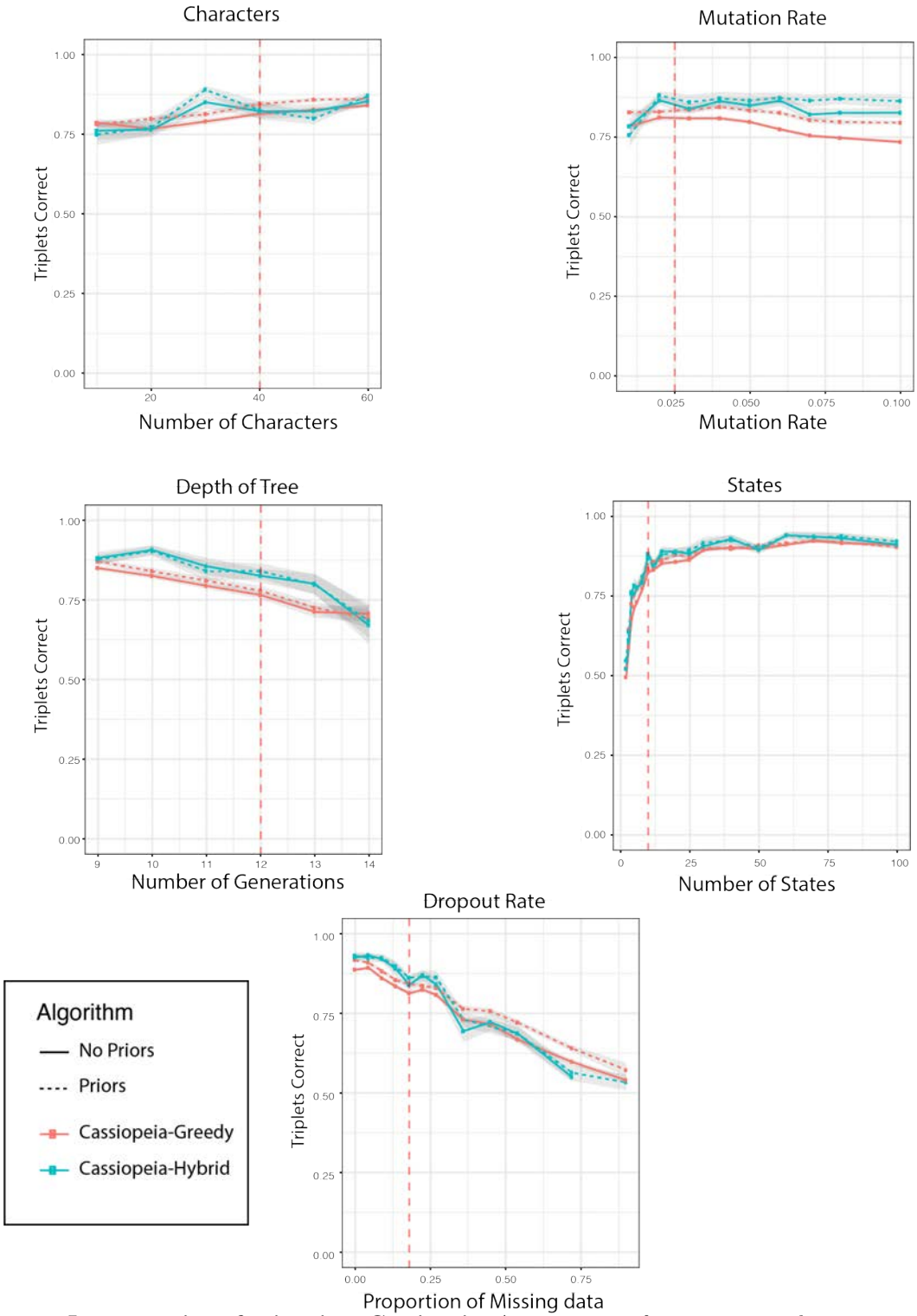


Figure 3.22: **Incorporation of priors into Cassiopeia.** A comparison of tree accuracy when using priors for both the greedy-only method and Cassiopeia. We compared performance as we varied the number of characters per cell, the mutation rate per character, the length of the experiment, the number of states per character, and the amount of missing data. Standard error is represented by shaded area.

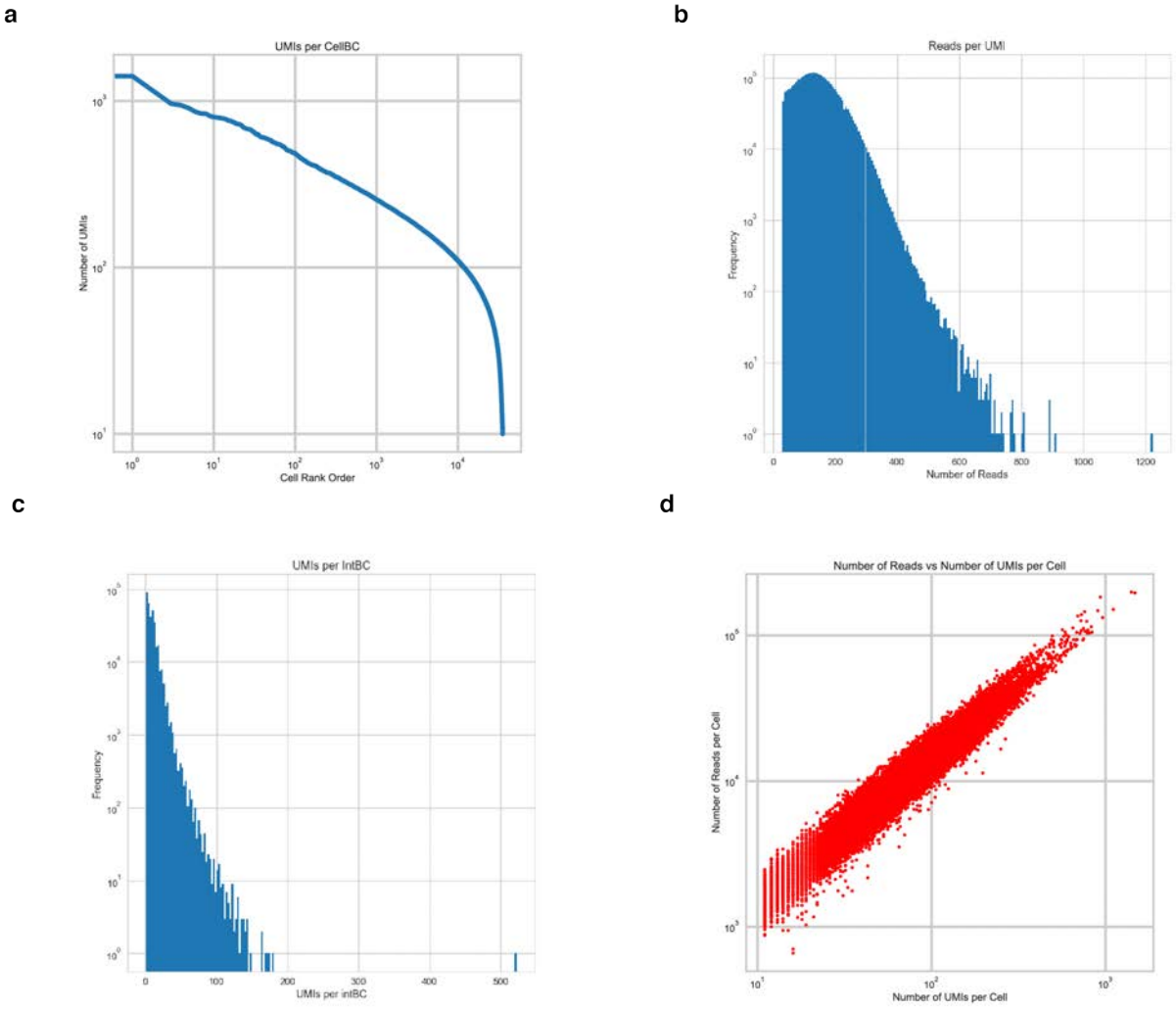


Figure 3.23: **Quality control metrics for the target site sequencing library processing pipeline.** (a-d) present quality control metrics after the processing pipeline. (a) Cells are ranked by the number of UMIs they contain, showing a median of 76; (b) The number of reads per UMI after UMI error correction and collaping, showing a median of 137; (c) The number of UMIs per integration barcode (intBC), showing a median of 7; (d) is the concordance between reads per cellBC and UMIs per cellBC, showing a pearson correlation of 0.96

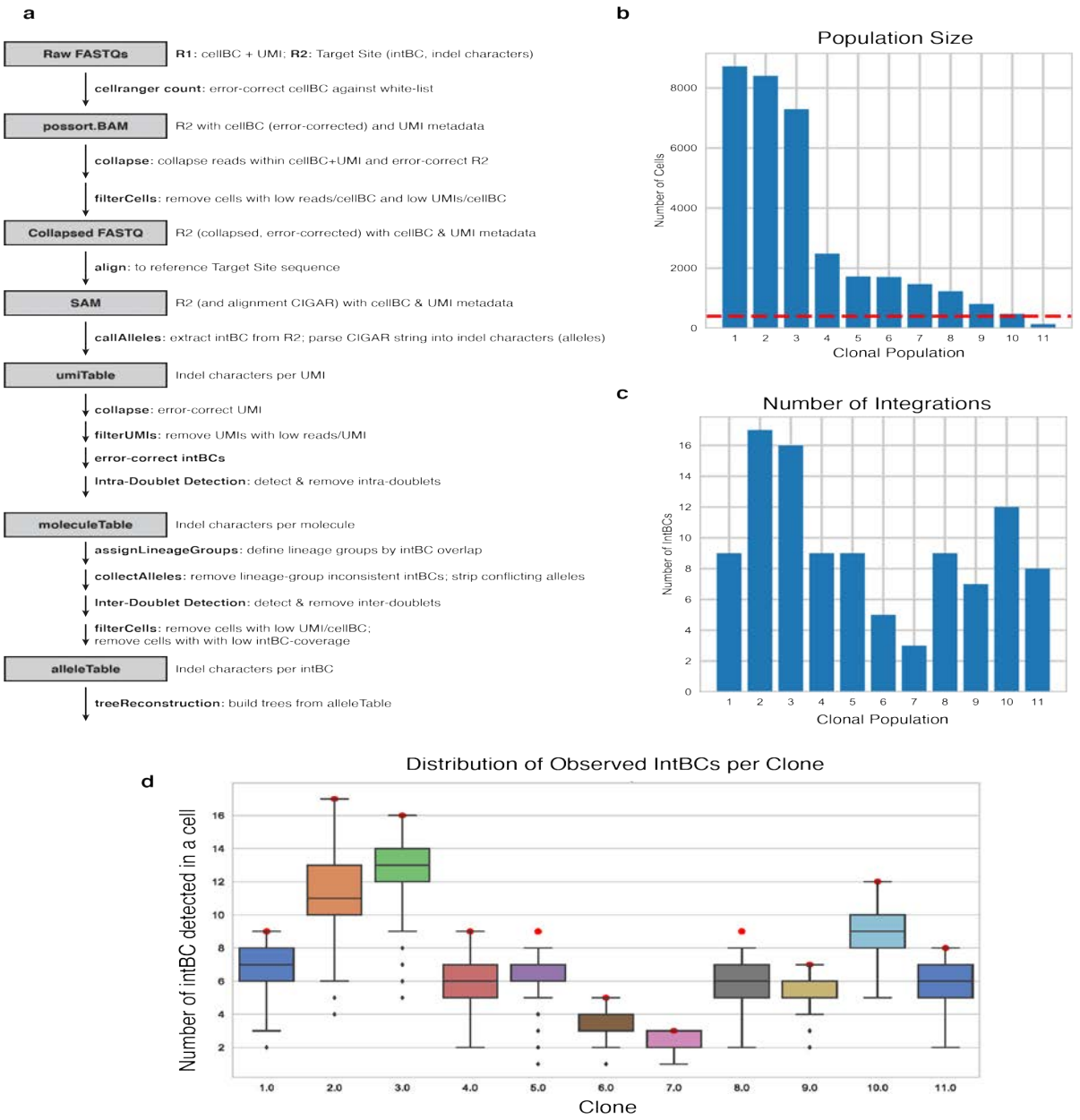


Figure 3.24: **Processing Pipeline for the *in vitro* dataset.** (a) shows a more in-depth flowchart of the Cassiopeia processing pipeline taking as input the raw FASTQs from a sequencing run and converting the observed reads into final trees. Cellranger "count" is used to map reads to dummy transcriptome (junk sequence that nothing will align to), filter cells, and read off the 10x cell barcodes and UMIs. The resulting BAM file is then passed through a series of cell filtering, UMI error correction, and allele mapping before becoming the final allele table that can be converted to character matrices for clone reconstruction. See methods for more detailed information for each step. (b-d) present additional summary statistics for the final allele table. (b) displays the number of cells per clone; (c) shows the median number of intBCs observed in each clone; (d) shows the distribution of the number of intBCs observed in each cell (red points are references to indicate the number of intBCs used to reconstruct the particular clone).

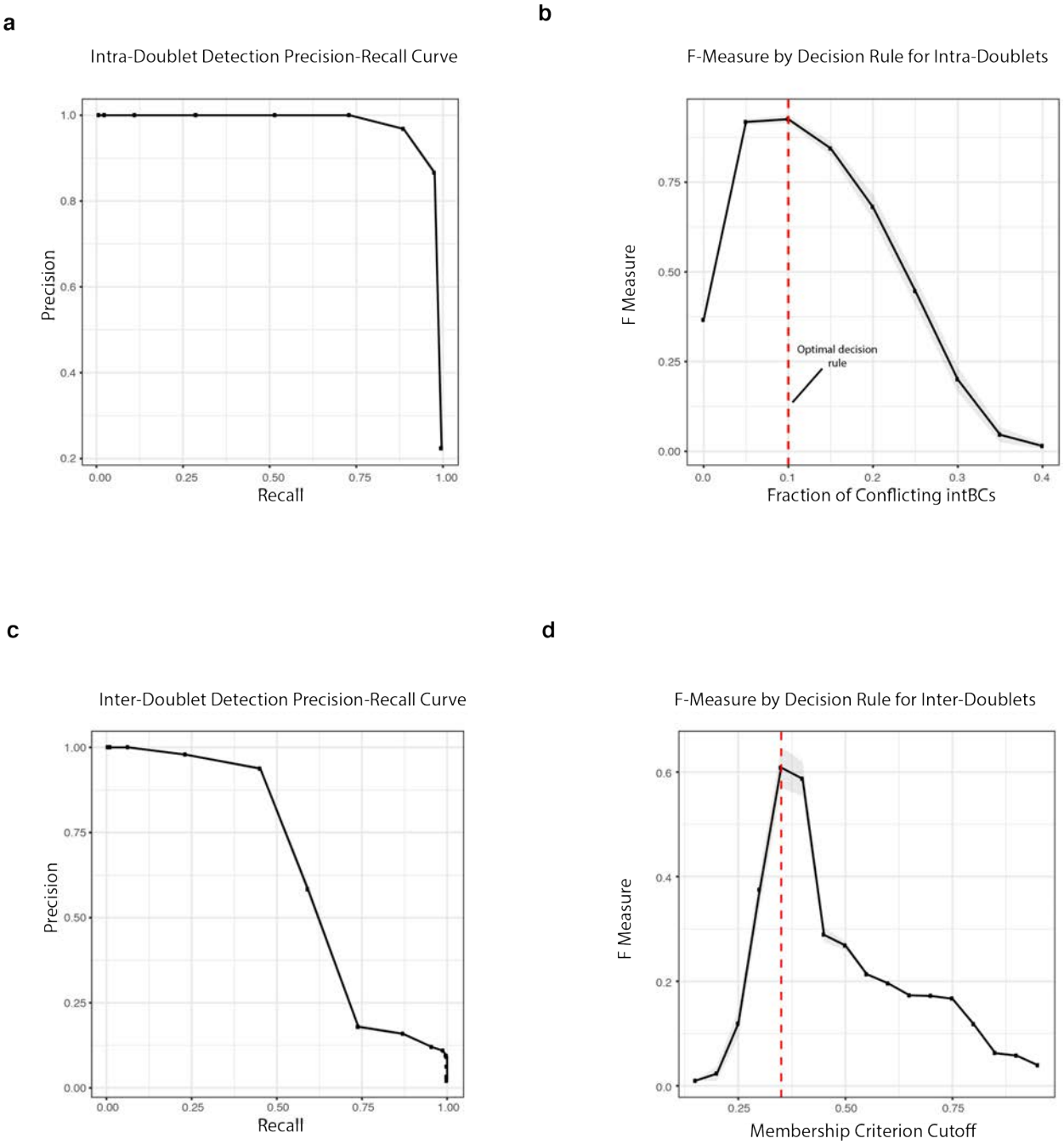


Figure 3.25: **Identification of doublets using intBCs.** IntBCs are used to identify doublets. (a-b) report the ability to identify doublets arising from the same clone, referred to as "intra"-doublets; (c-d) report the ability to identify doublets arising from different clones, referred to as "inter"-doublets. Doublets were simulated using the final allele table and 200 "intra"- and "inter"-doublets were created in each of 20 replicates. Precision-recall curves for intra- and inter-doublet detection methods are presented in (a) and (b), respectively. (c) and (d) present the F-measure (defined as the weighted harmonic mean between precision and recall) of detection methods for intra- and inter-doublets, respectively. Red-dashed lines denote the optimal decision rule for doublet detection. Standard error is represented by shaded area.

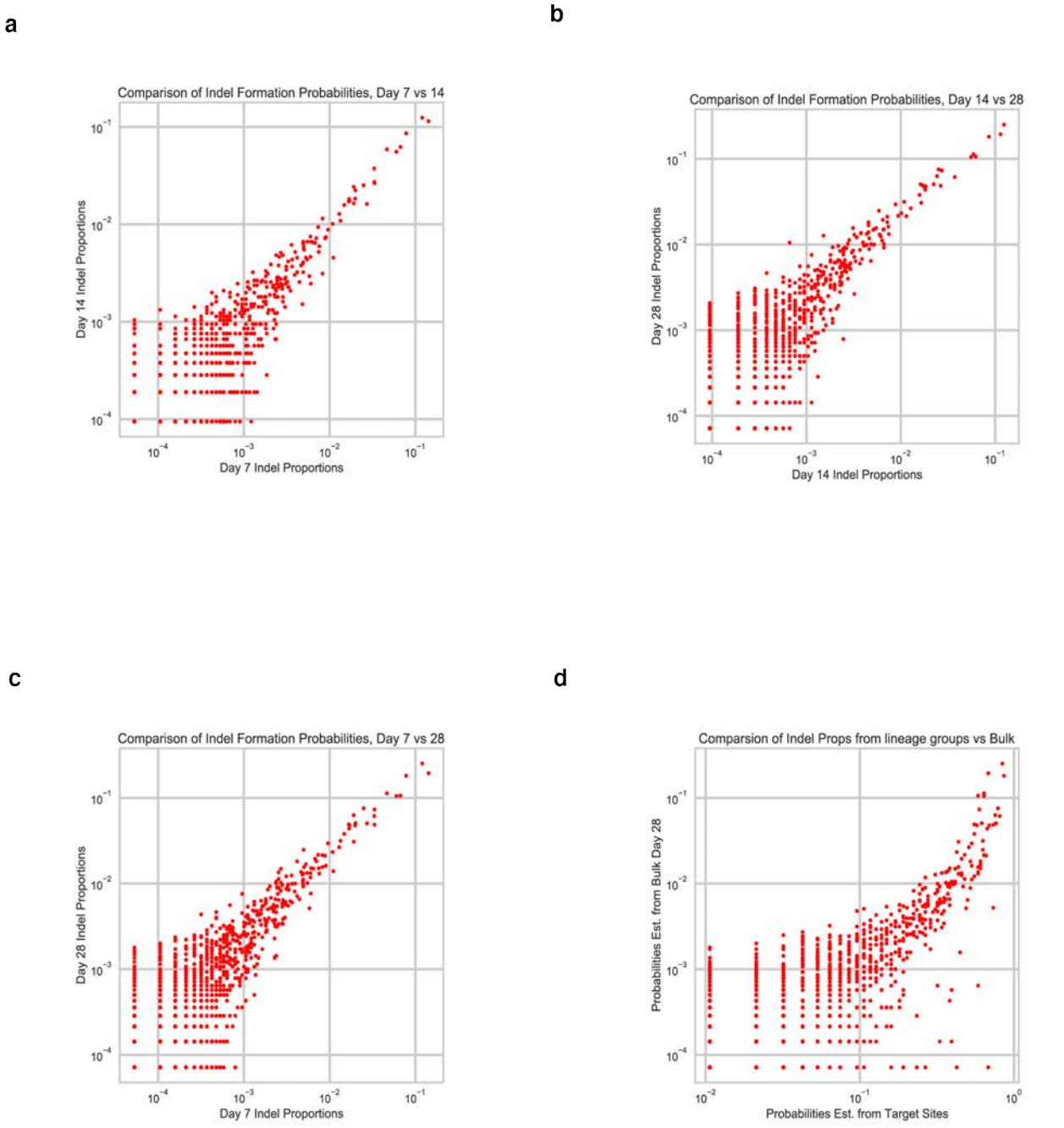


Figure 3.26: **Estimation of Prior Probabilities for Tree Reconstruction.** Prior probabilities to be used during tree reconstruction can be determined from both a bulk assay and independent clonal populations. Prior probabilities of mutations were determined by calculating the proportion of unique intBCs that report a particular indel (see methods). The bulk assay consisted of several independent clones with non-overlapping intBCs grown over the course of 28 days. (a-c) report the correlation of indel formation probabilities between various time points in the bulk experiment. A strong correlation is observed between all time points: 7 and 14 (a), 14 and 28 (b) and 7 and 28 (c). Indel formation probabilities can also be calculated using the intBCs from each clone as independent measurements. Using this method, (d) reports the correlation between this lineage-group specific probability calculation and the last time point of the bulk assay.

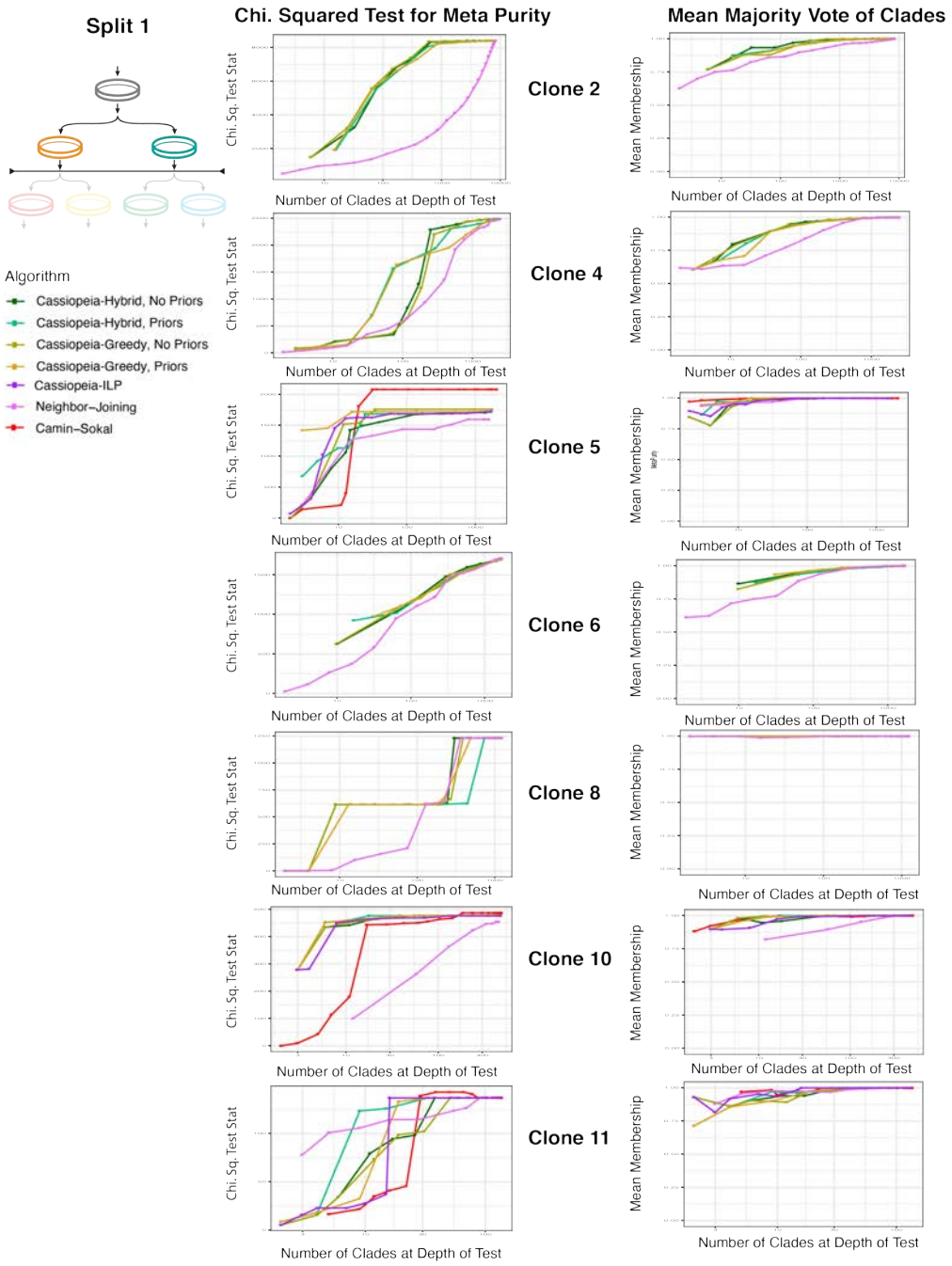


Figure 3.27: **Evaluation of algorithms on *in vitro* lineage tracing clones, First Split.** Trees were reconstructed for the remaining clones in the *in vitro* dataset that consisted of more than 500 unique cell states. LG2, LG4, LG6, and LG8 passed this threshold and were reconstructed with Cassiopeia (with and without priors), greedy-only (with and without priors) and Neighbor-Joining. The statistics provided were taken with respect to the first split ID (see methods). For both Cassiopeia with and without priors, we used a cutoff of 200 cells and each instance of the ILP was allowed 5000s to converge on a maximum neighborhood size of 6000. For example, for Clone 5 it is difficult to pinpoint a single reason for the observed variability other than the fact it has a very small proportion of unique cells, namely that every leaf represents multiple cells (which can come from different plates), thus potentially making the performance criteria less robust.

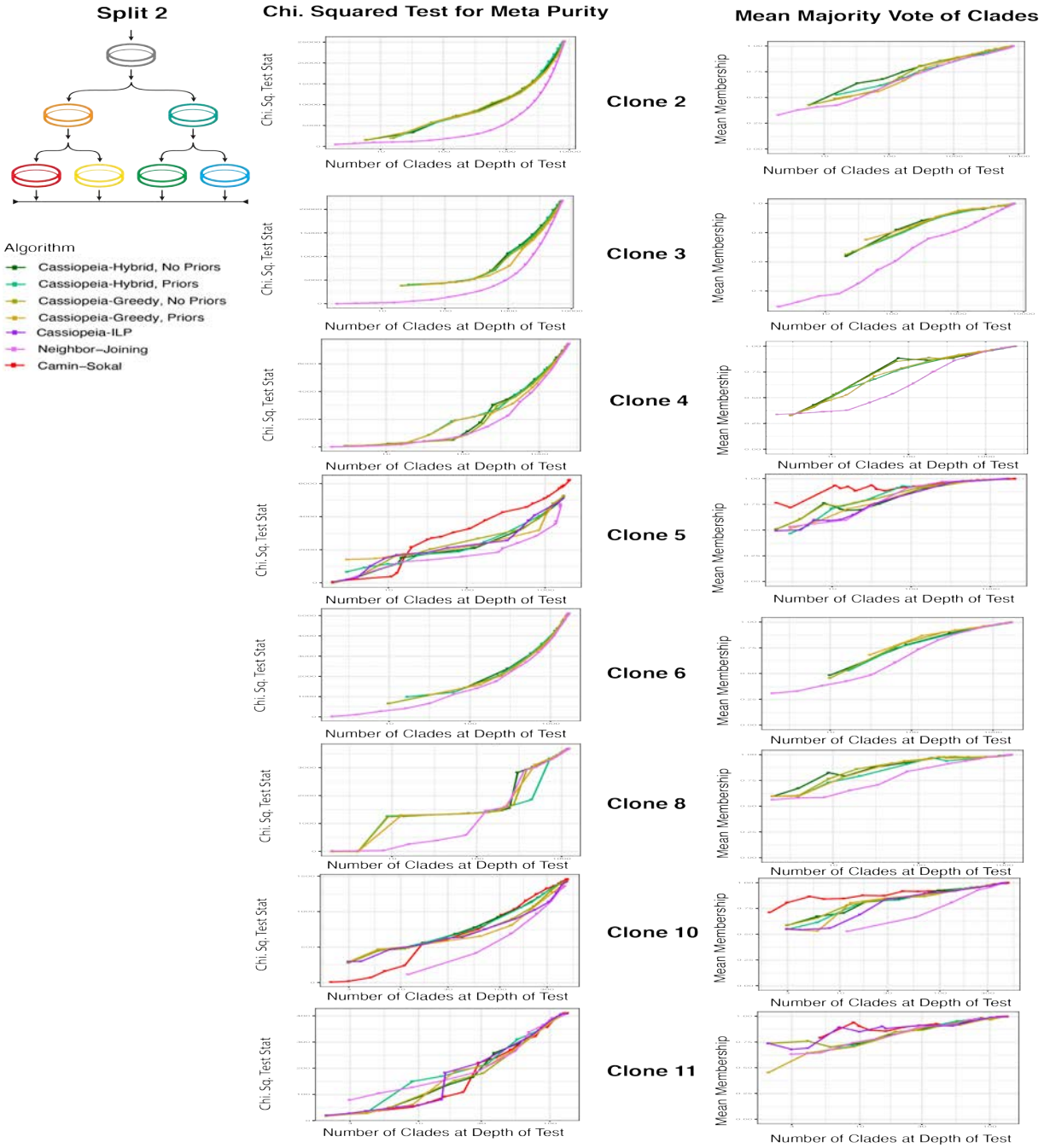
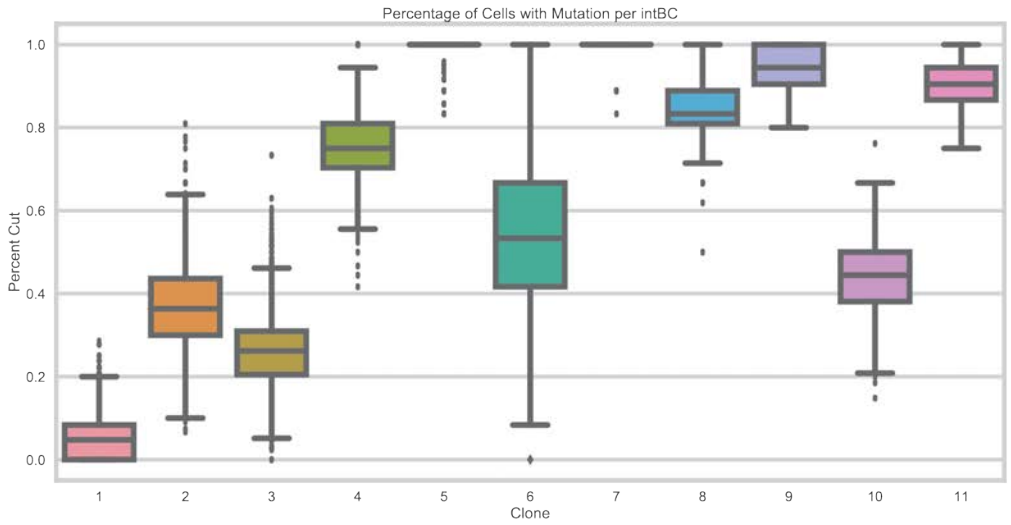


Figure 3.28: **Evaluation of algorithms on *in vitro* lineage tracing clones, Second Split.** Trees were reconstructed for the remaining clones in the *in vitro* dataset that consisted of more than 500 unique cell states. LG2, LG4, LG6, and LG8 passed this threshold and were reconstructed with Cassiopeia (with and without priors), greedy-only (with and without priors) and Neighbor-Joining. The statistics provided were taken with respect to the second split ID (see methods). For both Cassiopeia with and without priors, we used a cutoff of 200 cells and each instance of the ILP was allowed 5000s to converge on a maximum neighborhood size of 6000.

a



b

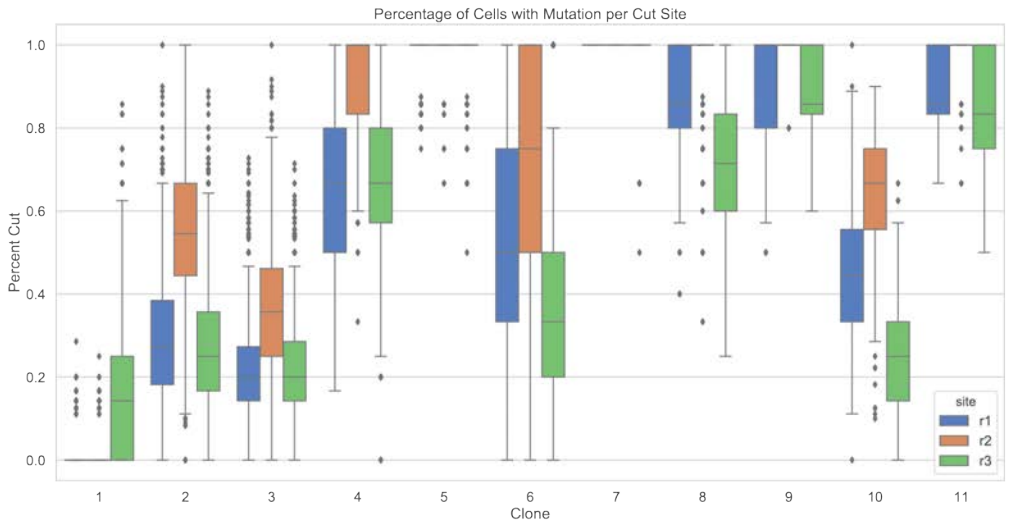


Figure 3.29: **Exhaustion of Target Sites across Clones.** Target site exhaustion for each clone, as measured by the proportion of sites observed as edited after the experiment. (a) presents the percentage of mutated cells across all cut sites per clone. (b) details the distribution of mutated cells per cut site in each clone.

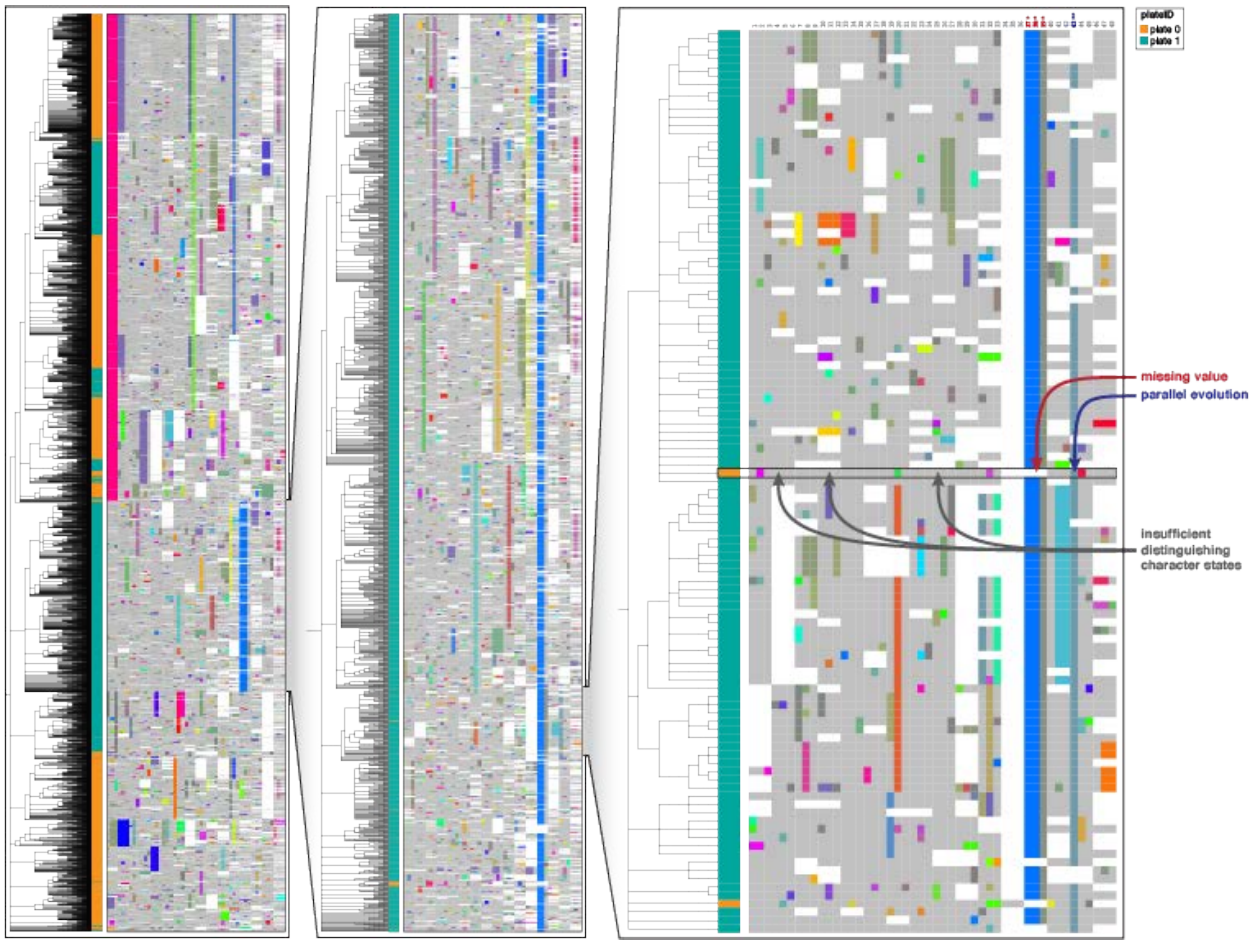


Figure 3.30: **Vignette of Inferential Mistakes for Clone 3.** An example from the reconstruction of Clone 3 with Cassiopeia-Hybrid where a cell has been misplaced in the tree due to several factors. In this case, it is clear that the cell was placed where it is due to an instance of parallel evolution of the state in character 43 (as annotated in the figure). Because the cell contained this state, it was grouped with cells of a different plate also containing this mutation. Furthermore, the cell contains few distinguishing mutations thus making it difficult to infer the true value of the missing values located in characters 37-39.

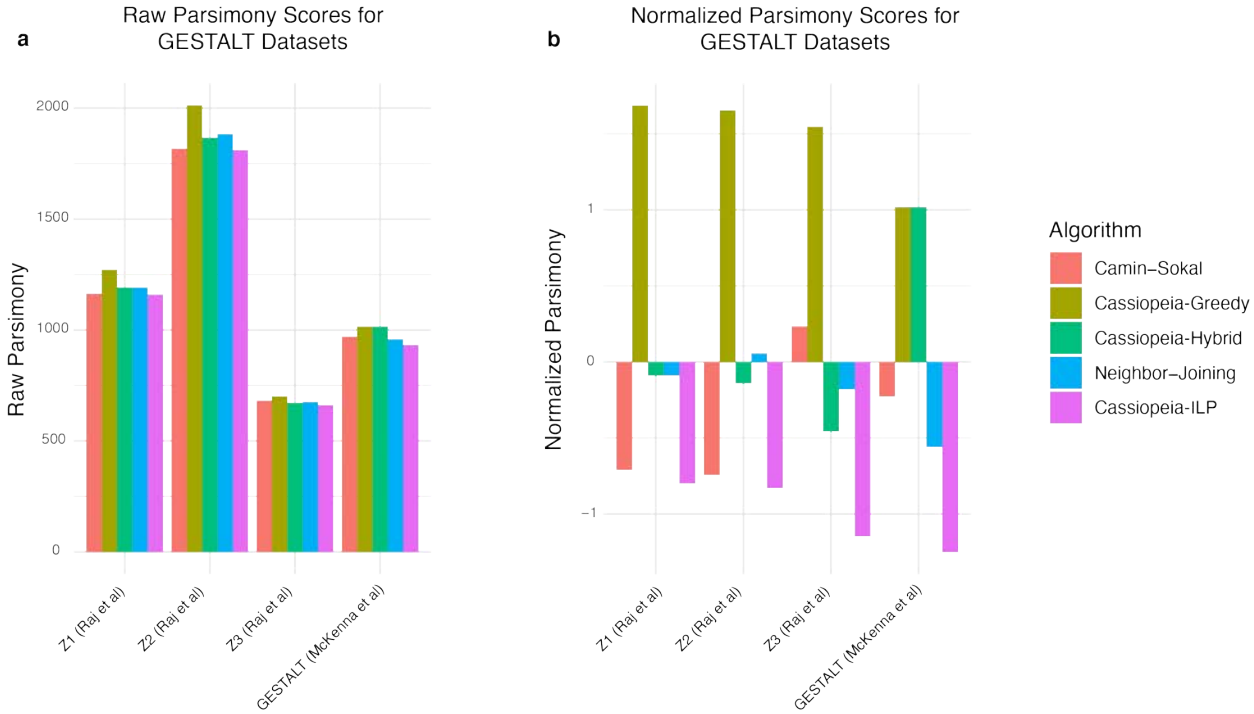


Figure 3.31: **Parsimony scores from reconstructions of the GESTALT datasets.** (a) Raw and (b) normalized parsimony scores for the parsimony scores from the GESTALT datasets. Camin-Sokal, Neighbor-Joining, Cassipeia-Greedy, -Hybrid, and -ILP were run on datasets from Raj et al [6] and McKenna et al [3]. Raw parsimony scores are calculated as the number mutations present in a phylogeny (summing over the mutations along every edge of the tree). The normalized scores correspond to z-scores for each dataset.

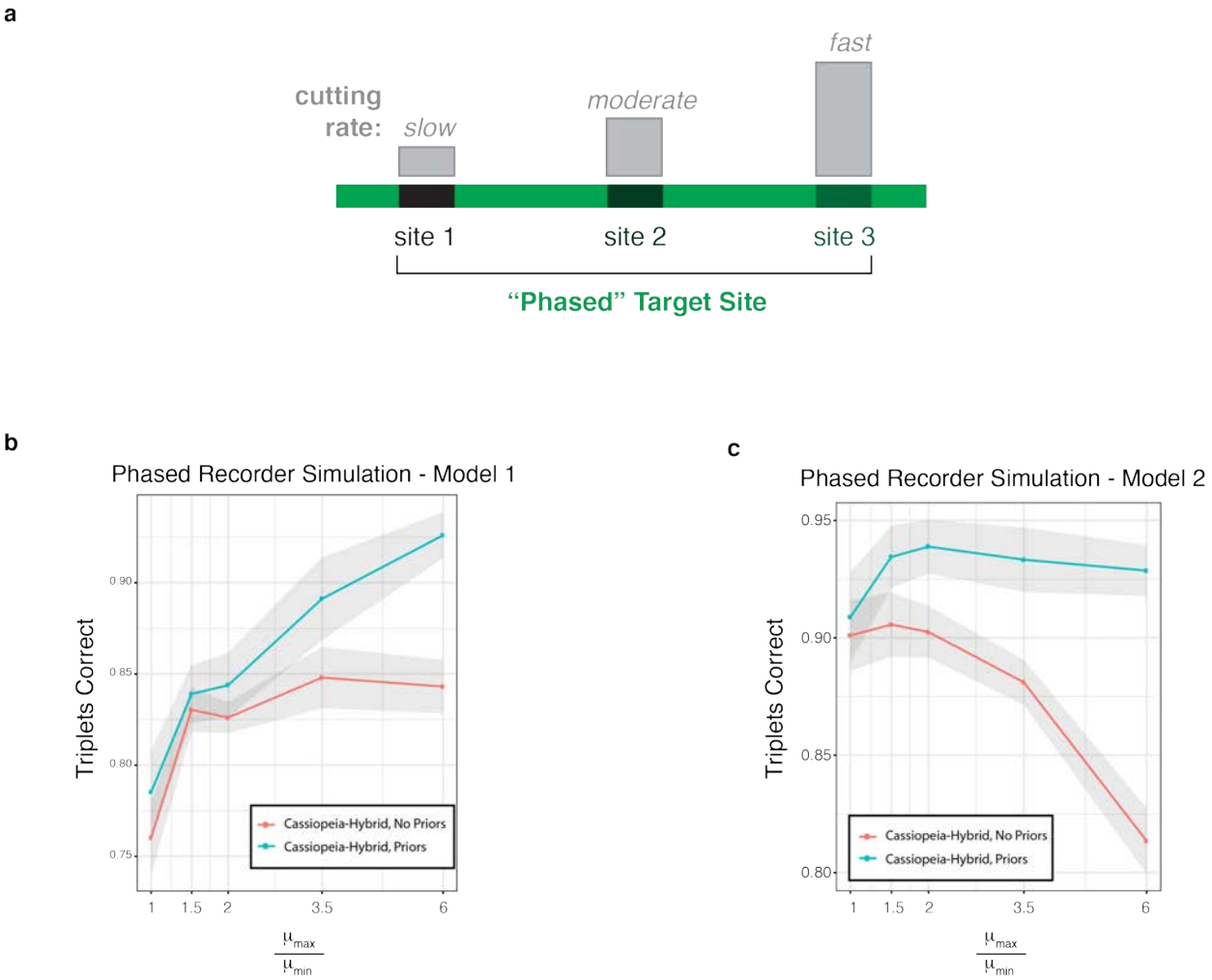


Figure 3.32: **"Phased Recorder" leverages variability across target sites.** (a) Design concept of the "Phased Recorder." (a) We simulated a "phased" editor, where each character is mutated at variable rates. (b-c) We varied the amount each character could very across 5 different experiments and simulated using two different indel formation rate models. Each cell had 50 characters with 10 states per character and a mean dropout of 10%. The amount of mutation variability is described with the ratio between the maximum and minimum mutation rates ($\frac{\mu_{max}}{\mu_{min}}$). Standard error is represented by shaded area. (b) Model 1 consists of drawing indels from a negative binomial distribution $NB(5, 0.5)$ where there are few "rare" indels. (c) Model 2 consists of drawing indels from the splined distribution of the empirical dataset's indel formation rates, as used in other synthetic benchmarks.

3.3 Theoretical Guarantees for Phylogeny Inference from Single-Cell Lineage Tracing

3.3.1 Authors and Contributions

Robert Wang*, Richard Zhang*, Alex Khodaverdian*, Nir Yosef. Theoretical Guarantees for Phylogeny Inference from Single-Cell Lineage Tracing. In Review.

* signifies Equal Contribution

RW conceived of the problem statement and led the development of the theoretical analyses. AK and RZ aided in the development of the theoretical analyses. The simulations were conceived by NY, RW, and RZ, and implemented by RZ. NY supervised the completion of the work. All authors contributed towards writing the manuscript.

3.3.2 Abstract

CRISPR-Cas9 lineage tracing technologies have emerged as a powerful tool for investigating development in single-cell contexts, but exact reconstruction of the underlying clonal relationships in experiment is plagued by data-related complications. These complications are functions of the experimental parameters in these systems, such as the Cas9 cutting rate, the diversity of indel outcomes, and the rate of missing data. In this paper, we develop two theoretically grounded algorithms for reconstruction of the underlying phylogenetic tree, as well as asymptotic bounds for the number of recording sites necessary for exact recapitulation of the ground truth phylogeny at high probability. In doing so, we explore the relationship between the problem difficulty and the experimental parameters, with implications for experimental design. Lastly, we provide simulations validating these bounds and showing the empirical performance of these algorithms. Overall, this work provides a first theoretical analysis of phylogenetic reconstruction in the CRISPR-Cas9 lineage tracing technology.

3.3.3 Introduction

Phylogenetic trees are routinely constructed to describe the developmental relationships within sets of extant taxa such as different organisms, proteins, or single cells. A landmark early example of using phylogenetics to describe cellular relationships was that of Sulston and colleagues reporting the development of C. elegans as deduced from meticulous visual observation [44, 170]. Recent progress in CRISPR-Cas9 based lineage tracing technologies now enables the inference of cellular lineage relationships in more complex organisms where visual observation is not possible. This is owing CRISPR-Cas9's ability to generate heritable, irreversible, and information-rich mutation events that can be read through single-cell assays (such as RNA-seq) and subsequently used to infer the underlying phylogeny [130, 146, 147,

4, 165, 27, 98, 145]. Typically, these technologies start by engineering a single progenitor cell with artificial transcribed recording sites that accumulate stable insertions or deletions ("indels") as a result of repair of Cas9 double-stranded breaks. These indel mutations are subsequently inherited by future descendants, and the accumulation of these mutations is used to infer the clonal relationships between the observed cells, stratifying them into clades of increasing resolution. Thus far, studies have paired these technologies with single-cell transcriptomic profiling [108] to study questions in development (e.g., inferring lineage relationships between cellular compartments) [4, 165, 147, 146, 27] and cancer progression (e.g., inferring rates and routes of metastases) [145].

Despite the many advantages of CRISPR-Cas9 lineage tracing systems, an outstanding goal is develop methods to accurately infer the underlying developmental process and to determine under what experimental conditions the problem is tractable. Exact reconstruction of the ground truth phylogenetic tree, defined here as having a reconstructed tree that exactly matches that of the ground truth clonal relationships, is plagued by various data-related complications (Fig.1). First, convergent evolution (or homoplasy) events can occur whereby the cells might appear to be incorrectly related to each other because the same indel occurs in unrelated clades. [128, 179, 98]. Second, substantial missing data is observed in these experiments in which the information at recording sites is lost due to partial RNA capture, recording site resection, or transcriptional silencing [98, 199, 155, 146, 147]. Finally, a mis-tuned "Cas9 editing rate" - the rate at which Cas9 induces heritable mutations used for lineage tracing - can lead to scenarios where there is a lack of mutation information sufficient for discerning relationships between cells. If the editing rate is too low, then "mutation-less edges" will occur in which a cell divides before it acquires a mutation, generating a irresolvable polytomy on the underlying phylogeny. If the editing rate is too high, then "mutation saturation" occurs in which the recording sites all acquire mutations before cell division ceases, making differentiation of the bottom of the phylogeny impossible [155]. As these complications are functions of parameters of the CRISPR-Cas9 lineage tracing system, a question of experimental design arises. Specifically, which configurations of the many experimental parameters involved in the CRISPR-Cas9 lineage tracing technology can alleviate these complications and make the problem of exact reconstruction more feasible? Our goal in this study is to address this question and identify configurations that are sufficient to theoretically guarantee exact reconstruction while also providing the accompanying phylogeny inference algorithms.

While there is extensive work establishing theoretical guarantees for exact reconstruction in other (arguably simpler) phylogenetic models, these models do not capture the specific features of CRISPR-Cas9 lineage tracing. One such example is the Cavender-Farris-Neyman (CFN) 2-state model (a.k.a. binary Jukes-Cantor). In this setting, for n taxa, algorithms have been developed to exactly reconstruct subtrees with the number of characters $k = O(\frac{log(n)}{\ell^2})$ [71, 137, 41] if the length (duration) of every edge is greater than some value ℓ, and complete reconstruction is possible if the length of each edge is also upper-

bounded by $\frac{log(2)}{4}$ [40, 135]. Unfortunately, the CFN model cannot be readily applied to data of this type as it differs from the CRISPR-Cas9 settings in several critical ways: there are only two states (mutated, non-mutated) as opposed to the arbitrary state space of indels caused by Cas9 repair, reversibility of mutations is allowed as opposed to the irreversible mutations of Cas9, and there is no missing data [26, 48]. This motivates the development of theoretical bounds for the CRISPR-Cas9 lineage tracing model in particular.

Despite this need, to our knowledge there has yet to be any work directly exploring guarantees for exact reconstruction in the general CRISPR-Cas9 lineage tracing model. There do exist a few regimes where theoretical guarantees exist (e.g., under perfect phylogeny where every mutation occurs exactly once [74, 98]), but these regimes rarely exist in experimental settings. When these conditions are not met, other methods rely on criteria defined over the reconstructed phylogeny to guide reconstruction. Maximum-likelihood [55, 199], parsimony-based [21, 129, 98, 165] and distance-based [154, 164] methods optimize over likelihood, the minimum number of mutations, and variations of the ME (minimal evolution) criterion respectively [63, 138]. Unfortunately, optimizing these criteria does not necessarily correspond to reconstructing the correct tree. While there have been results regarding exact reconstruction for Neighbor-Joining in particular given certain error bounds between the true and observed distance metric [136, 12], there has yet to be work characterizing how and when mutation-based distances meet these criteria. Additionally, while existing studies have used simulations to provide insight into the relationship between reconstruction accuracy and experimental parameters [155, 98, 69], there has been limited theoretical exploration into CRISPR-Cas9 lineage tracing experimental design and how to optimize parameters in order to achieve accurate reconstruction.

In this paper, we derive such bounds for the CRISPR-Cas9 lineage tracing model. We develop two algorithms with theoretical guarantees for exact reconstruction of the underlying phylogenetic tree of a group of cells, showing that exact reconstruction can indeed be achieved with high probability given sufficient information capacity in the experimental parameters. In particular, we begin with an algorithm for a lower bounded edge constraint model whereby we prove that, in the absence of missing data, high probability of exact reconstruction can be achieved in polynomial time with $O(\frac{log(n)}{\ell^2})$ characters, matching the CFN 2-state model. The lower bound assumption translates to a reasonable assumption over the minimal time until cell division [168]. We further extend this algorithm and bound to account for missing data, showing that the same bounds still hold assuming a constant probability of missing data. We then consider the case of imposing an additional upper bound on edge lengths in our tree, to which we apply a bottom up approach that decreases the asymptotic number of characters required to $O(\frac{log(n)}{\ell})$ characters, improving on previous bounds. The upper bound corresponds to an assumption on the maximum time until cell division, which can be evaluated in lineage-traced populations, as they by definition should not be post-mitotic. Using these asymptotic bounds we characterize the dependence between the necessary number of characters and other experimental parameters, such as the mutation

Parameters of Interest	
Parameters	Description
k	Number of characters
n	Number of leaves in the tree. i.e. number of observable samples
ℓ	The minimum edge length on the ground truth phylogeny, normalized to the height of the tree
λ	Rate at which each character mutates. Corresponds to rate at which recording sites are cut by Cas9
q	Probability of collision for independent mutation events
p_d	The probability that the state at a given character/leaf pair is indeterminable because of missing data
d^*	Depth such that any triplet rooted above d^* that is well separated enough can be recovered with high probability. Normalized to the height of the tree
$\delta(d)$	When all other parameters are fixed, $\delta(d)$ is a function that is proportional to the expected difference between ingroup-ingroup similarity and ingroup-outgroup similarity for triplet at depth d. It is the primary function used for the triplets oracle for making decision

Table 3.1: A summary of key model variables

rate (controlled by guide affinity [27]), thus offering insight into how experimental design may be improved as the field develops. Lastly, we validate these relationships between k and the experimental parameters via a large set of simulations and present empirical bounds on the k required for exact reconstruction.

Taken together, our results provide a first theoretical analysis into the feasibility of phylogeny inference in the single-cell CRISPR-Cas9 based setting. As this field continues to grow and generate excitement as exemplified by the recent Allen Institute Dream Challenge [69], we anticipate that this work will inform the emerging dependencies between the experimental parameters and will serve to guide future studies both in terms of technology development (which parameters should be optimized) and tailoring the CRISPR-Cas9 lineage tracing system for specific case studies (e.g., dependent on the expected number of cell generations in the entire process). The algorithms and simulation engine presented here are available as an open source software in `https://github.com/YosefLab/Cassiopeia`.

Problem Setup and Model Assumptions

In order to tackle the problems of guarantees on exact reconstruction and optimizing experimental design, it is helpful to consider a more abstract theoretical model. We begin with a single-cell with k unmutated characters, corresponding to k unedited recording sites

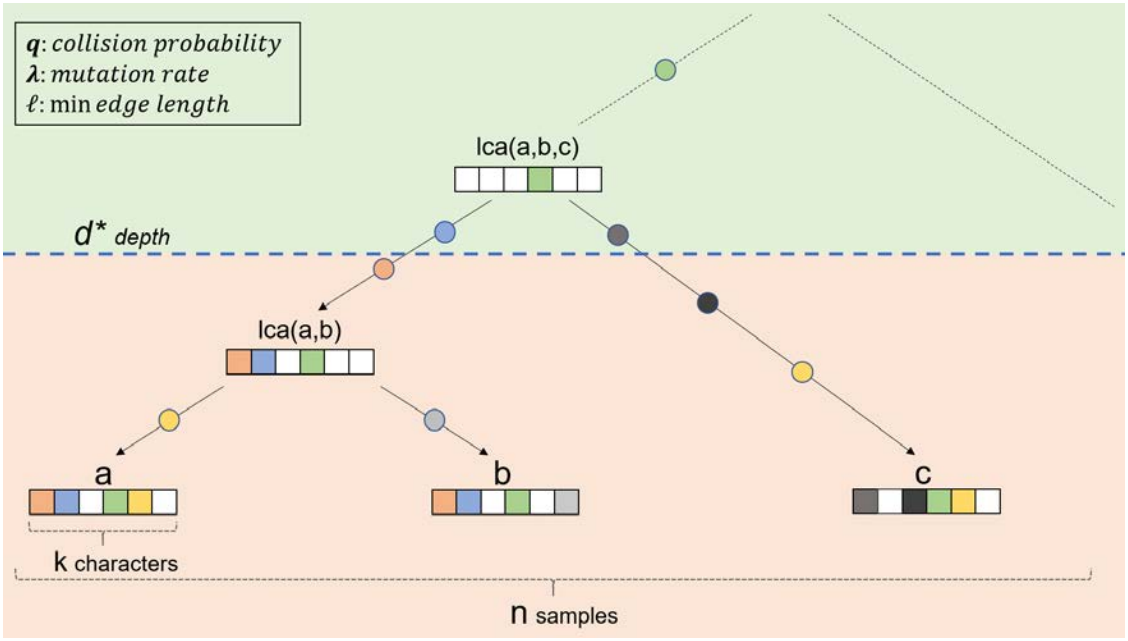


Figure 3.33: **General Problem Setup**. In particular, we note that a and b are the ingroup, whereas c is the outgroup. In addition, we point out some of the key variables used throughout this manuscript: d^*, k, n, q, λ, ℓ.

at which CRISPR-Cas9 can induce mutations. This cell then undergoes cell division. Over time, characters may mutate from their unedited states at instantaneous rate λ according to an exponential distribution. While previous models have assumed a per-generation mutation rate [98, 169, 199], in our model, mutations occur independently from cell division. We believe this is a more accurate representation of current experimental regimes.

When a mutation occurs, the respective character adopts a state (corresponding to an indel) according to some probability distribution over the space of possible indels. We assume that once a character mutates, it can never change its state again and that this mutation will be inherited by all descendants of this cell. This irreversibility assumption is derived from the fact that after an indel is introduced at a recording site by Cas9, the guide RNA no longer has affinity at that site preventing future edits [155, 87]. After a set period of time has elapsed, a subset of the contemporary cells (leaves of the tree) are collected for sequencing. We denote the size of this set by n.
Finally, some proportion with expectation p_d of each cell's characters will have their states rendered indeterminable. This missing data may be due to low capture at the sequencing step and affect only one cell, or through resection (excision) or transcriptional silencing events which are inherited throughout the cell division process and persist in all descendant cells [98, 155, 146, 55, 147]. We refer to the former as stochastic missing data and the latter

as heritable missing data.

Given the set of samples collected at the end of the experiment, the goal is to construct the phylogenetic tree that relates the observed cells to one another, based on their character/state information. Note that even though we collect a subset of the cells, the underlying "ground truth" phylogeny is still a binary tree. Formally, this problem can therefore be viewed as a character-based inference of a rooted phylogeny where the tree is binary, the root is unmutated, the mutations are irreversible, and some mutation data may be missing. In addition to offering algorithms to the problem, our goal is to shed light on the relationship between the various experimental parameters k, q, n, λ, and p_d (summarized in Table 3.1), finding the regimes in which exact lineage reconstruction is tractable. We next explore the generative model in more detail.

Generative Process

Let $\mathcal{T}$ be a binary tree representing the ground truth phylogeny in a CRISPR-Cas9 lineage tracing experiment. Let S be the set of leaves in $\mathcal{T}$, with each leaf representing a single cell in our input. Each edge (u, v) of $\mathcal{T}$ has a length, $l(u, v)$, representing the duration of time between the two respective cell division events. Furthermore, we define the distance $dist(u, v)$ between two vertices u, v as the sum of the lengths of the edges in the path between u and v. In addition, we denote the root node by r and say that a vertex u is at depth d if $dist(r, u) = d$. Finally we assume that the distance between the root and any leaf is equal to 1 (normalizing arbitrary time units), as all leaves are sampled at one time point, making the tree an ultrametric.

Each node in the tree has k independently evolving characters, each of which can take on states in $\{0, 1, ...m\}$. Each character starts in an unedited state (0) at the root. Once a cell acquires a mutation at a certain character, this mutation is inherited by all descendants of that cell, and mutations cannot occur at that character in these descendants (irreversibility). For each character, the time it takes for a mutation to occur on a path in the tree is exponentially distributed with rate λ, and is independent of cell division events. That is, if $r = 0^k$ is the root then the probability that a mutation occurs along the path from r to some downstream descendant vertex u for any particular character is

$$\int_0^{dist(r,u)} \lambda e^{-\lambda t} dt = 1 - e^{-\lambda dist(r,u)}$$

We assume that each character mutates independently of all other characters. Once a character mutates, it takes on state $j \in \{1, ...m\}$ with probability q_j. Let $q = \sum_{j=1}^{m} q_j^2$ be the probability that two independent mutations at the same character index arrive at the same state. At the end of the experiment, each character in the leaves has a p_d probability of becoming indeterminable and adopting the "missing" state. Finally, another measure of

similarity between nodes, which we will use throughout, is derived from their mutation profiles. Here, we define by $s(u, v)$ the number of mutations shared by nodes (cells) u and v. The definitions of the primary variables used in our analysis are summarized in Table 3.1 and Figure 3.33.

Additional Definitions

For a triplet of nodes $a, b, c \in \mathcal{T}$, we use the notation of $(a, b|c)$ to denote that c is the "outgroup", namely that $LCA(a, b)$ is a descendent of $LCA(a, b, c)$, where LCA gives the lowest common ancestor in $\mathcal{T}$. This leads us naturally to the concept of a triplets oracle.

Definition 27 (Triplets Oracle)**.** *We say that a function $O : S^3 \to S$ is a triplets oracle if for every leaf triplet $(a, b|c) \in S$, $O(a, b, c) = c$.*

It is known that $\mathcal{T}$ can be reconstructed exactly in $O(n \log n)$ time given a triplets oracle. Typically, it is unreasonable to expect to have exact triplet oracles in practical applications, so instead we define a relaxed version of this oracle.

Definition 28 ($(\ell^*, d^*)-$accurate partial oracle)**.** *We say that O is an $(\ell^*, d^*)-$accurate partial oracle, if for every triplet $(a, b|c)$ such that $depth(LCA(a, b, c)) \leq d^*$, then $O(a, b, c)$ returns either c or $Null$. In addition, if it also follows that $dist(LCA(a, b), LCA(a, b, c)) \geq \ell^*$, then the oracle is guaranteed to return the correct answer, i.e. $O(a, b, c) = c$.*

In other words, the partial oracle does not return the wrong answer for triplets whose LCA is close to the root (max depth d^*). If in addition the triplet is separated by a distance of at least ℓ^* it is guaranteed to return the correct outgroup. In the remaining cases (e.g., triplets with an LCA far from the root, which are thus more difficult to resolve), the partial oracle can be wrong.

Throughout the paper, we will use the first order approximation $1 - e^{-x} \approx x$ when suitable. We will also be using the following versions of Hoeffding's inequality:

If $Y \sim Bin(n, p)$, and $\mu = np = E(Y)$, then we have:

$$Pr[Y \geq (1 + \beta)\mu] \leq exp(-\frac{\beta^2 \mu}{2 + \beta}) \quad \text{for } \beta > 0$$

$$Pr[Y \leq (1 - \beta)\mu] \leq exp(-\frac{\beta^2 \mu}{2}) \quad \text{for } \beta \in (0, 1)$$

3.3.4 Results

In the first section of this paper we show that in an experiment where each sample has sufficiently high number of characters and states (where the required number of characters and states depends on λ, q, ℓ, d^* and p_d), that it is possible to construct $(\ell, d^*)-$partial oracles with high probability. Given these partial oracles, we have top-down algorithms that can exactly reconstruct $\mathcal{T}$ up to depth d^*, either with or without missing data. In particular, note that if $d^* = 1$ and ℓ is at most the minimum edge length, then an $(\ell, d^*)-$partial oracle is a triplets oracle, which leads to an exact reconstruction of $\mathcal{T}$ in $O(n \log n)$ time. In this paper, we study guarantees in full and partial exact reconstruction of the ground truth tree. We will give the guarantees about full reconstruction as corollaries of our main theorems as follows:

Corollary 1. *Given a ground truth tree, $\mathcal{T}$, of height normalized to* 1 *under the lineage tracing model with k characters, n samples, minimum edge length ℓ, and constant mutation rate and state space, there exists a polynomial time algorithm to reconstruct $\mathcal{T}$ with high probability when $k = O(\frac{\log n}{\ell^2})$.*

Accounting for the possibility of missing data (incomplete information on the mutation profile of each cells in our study), we get:

Corollary 2. *Under the same conditions above, assume that information of the state of any given character in a given cell can be masked with probability p_d independently for each character. In that case, there exists polynomial time algorithms to reconstruct $\mathcal{T}$ with high probability when $k = O(\frac{\log n}{\ell^2(1-p_d)^3})$.*

As another extension, we consider a less constrained case with no lower bound on edge length. We show that in that case we can still get partial recovery as follows:

Corollary 3. *Under the same conditions as in corollary 1 and with no lower bound on edge length ℓ, there exists a polynomial time algorithm that with high probability will return a tree which correctly resolves all triplets, $(a, b|c)$ such that $dist(LCA(a, b, c), LCA(a, b) \geq \ell^*$ when $k = O(\frac{\log n}{\ell^{*2}})$.*

Finally, we consider a more constrained case, where edge lengths are both upper- and lower-bounded. We demonstrate that it is possible to achieve a stronger lower bound on the number of characters required via a bottom-up algorithm. This part is based on an alternative strategy, in which we conduct a bottom up tree reconstruction without using an oracle. This alternative approach gives the following theoretical guarantee:

Corollary 4. *If we assume that edge lengths are between ℓ and $O(\sqrt{\ell})$ then for a sufficiently low mutation rate, there exists polynomial time algorithms to reconstruct $\mathcal{T}$ with high probability when $k = O(\frac{\log n}{\ell})$.*

Partial Reconstruction of Phylogenies with Top-down Oracle-based Algorithms

For a triplet a, b, c, suppose WLOG that $s(a,b) \geq s(b,c) \geq s(a,c)$. Our goal is to define a sufficiently high threshold t such if $s(a,b) - \max(s(b,c), s(a,c)) > t$ then with high probability, c is the outgroup. Since leaf node similarities can be readily computed from our input, this will help define a triplets oracle.
First, consider the case in which there is no missing data and all character states are determinable at the end of the experiment. Let $(a, b|c)$ be a triplet where

$$dist(LCA(a,b), LCA(a,b,c)) \geq \ell^* \text{ and } depth(LCA(a,b,c)) = d \leq d^*$$

Given our assumptions on the mutation process, the similarities $s(a,c)$ and $s(b,c)$ have the same distribution, since c is the outgroup. Thus, we will focus on analyzing the quantity $E[s(a,b) - s(b,c)]$ WLOG. Let $s_w(u,v)$ be the number of mutations shared by u and v that occurred after the point when the three lineages diverged from $LCA(u,v,w)$. Then we have that $s(a,b) - s(b,c) = s_c(a,b) - s_a(b,c)$ since any mutation that occurred before $LCA(a,b,c)$ is inherited by all three nodes and will contribute equally to $s(a,b)$ and $s(b,c)$. The number of mutations shared by a and b after their divergence is distributed according to Binomial(k,p) where p, the probability of a given character having the same mutation in a and b, is

$$\geq e^{-\lambda d}((1 - e^{-\lambda \ell^*}) + e^{-\lambda \ell^*}(1 - e^{-\lambda(1-\ell^*-d)})^2 q)$$

where $e^{-\lambda d}$ is the probability that the mutation did not occur before $LCA(a,b,c)$. The left term inside the parentheses is the probability that the shared mutation came from a single event that occurred on the path from $LCA(a,b,c)$ to $LCA(a,b)$. The right term is the probability that the shared mutation came from two independent events that happened downstream of $LCA(a,b)$ (i.e., homoplasy).
Considering there are k characters, we then have:

$$E[s_c(a,b)] \geq k e^{-\lambda d}((1 - e^{-\lambda \ell^*}) + e^{-\lambda \ell^*}(1 - e^{-\lambda(1-\ell^*-d)})^2 q)$$

A similar computation shows that:

$$E[s_a(b,c)] = k e^{-\lambda d}(1 - e^{-\lambda(1-d)})^2 q$$

Thus, we have that:

$$\begin{aligned}
E[s(a,b) - s(b,c)] &\geq k e^{-\lambda d}(1 - e^{-\lambda \ell^*} + q e^{-\lambda \ell^*}(1 - 2e^{-\lambda(1-\ell^*-d)} + e^{-2\lambda(1-\ell^*-d)}) \\
&\quad - q(1 - 2e^{-\lambda(1-d)} + e^{-2\lambda(1-d)})) \\
&= k e^{-\lambda d}(1 - e^{-\lambda \ell^*} + q(e^{-\lambda \ell^*} - 1 + e^{-2\lambda(1-d)+\lambda \ell^*} - e^{-2\lambda(1-d)})) \\
&= k e^{-\lambda d}((1 - e^{-\lambda \ell^*})(1 - q) + q e^{-2\lambda(1-d)}(e^{\lambda \ell^*} - 1)) \\
&\geq k e^{-\lambda d}((1 - e^{-\lambda \ell^*})(1 - q) + q e^{-2\lambda(1-d)} \lambda \ell^*) \\
&\approx k(e^{-\lambda d} \lambda \ell^*(1 - q) + q e^{-\lambda(2-d)} \lambda \ell^*) \\
&= k \lambda \ell^*(e^{-\lambda d}(1 - q) + q e^{-\lambda(2-d)})
\end{aligned}$$

Let $\delta(d) = e^{-\lambda d}(1-q) + qe^{-\lambda(2-d)}$. We then have that for any triplet $(a, b|c)$, where $depth(LCA(a,b,c)) = d$ and $dist(LCA(a,b,c),\ LCA(a,b)) \geq \ell^*$:

$$E[s(a,b) - s(b,c)] \geq k\lambda\ell^*\delta(d)$$

Defining a $(\ell^*, d^*)-$Oracle We are now ready to define the decision rule that will be used by the partial oracle. Let $d^* \in [0,1]$ be an arbitrary depth to which we expect the oracle to be correct, and let $\delta^* = \min_{x\in[0,d^*]} \delta(x)$. Notably, the δ^* function has a closed form that depends on λ, q, and d^* (see supplemental). For a particular triplet a, b, c, the oracle proceeds as follows:

i) Set a threshold $t = \frac{1}{2}k\lambda\ell^*\delta^*$.

ii) If there exists a pair a, b out of the triplet, such that $s(a,b) - \max(s(a,c), s(b,c)) > t$, then return c as the outgroup. Otherwise return $Null$.

In the following we will prove that for a sufficiently large k, the function defined above is a $(\ell^*, d^*)-$Oracle. In particular, let $(a, b|c)$ be any triplet where $LCA(a,b,c)$ is at depth $d < d^*$. We will prove that the following two conditions hold with high probability:

i) If $dist(LCA(a,b), LCA(a,b,c)) \geq \ell^*$, then $s(a,b) - \max(s(b,c), s(a,c)) > t$

ii) In all cases, $\min(s(b,c), s(a,c)) - s(a,b) < t$.

To see that conditions $i)$ and $ii)$ imply correctness of the (ℓ^*, d^*)-Oracle, note that the second condition guarantees that it is unlikely that the oracle will return the wrong answer when called on a triplet rooted at depth at most d^*. It will therefore return either the correct outgroup or $Null$. The first condition guarantees that if the triplet is also separated by a path of length at least ℓ^*, then the outgroup will be correctly returned.
Figure 3.36 in supplemental provides an empirical visualization of the (ℓ^*, d^*)-Oracle using simulations. The simulations mimic the CRISPR-Cas9 lineage tracing system, and are also described in supplemental.
Before computing the k necessary to make conditions $i)$ and $ii)$ hold, we first state the following lemma which allow us to derive a worst case bound on the probability of triplets failing to satisfy condition $i)$.

Lemma 10. *(proof in supplemental): Let $(a, b|c)$ be a triplet with $\alpha = dist(LCA(a,b,c), LCA(a,b))$. $P[s(a,b) - s(b,c) \geq t]$ is increasing with α.*

Now we can compute the k that ensures that both conditions $i)$ and $ii)$ are satisfied with high probability.

Lemma 11. *(proof in supplemental): Condition i) holds with probability at least* $1-\zeta$ *if we have the following guarantees on the parameters* q, λ, ℓ^*, d^* *and* k*:*

$$k \geq \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda \ell^{*2}\delta^*(1-q+qe^{-2\lambda})}$$

Both conditions i and ii hold with probability at least $1-\zeta$ *if we have:*

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)q}{\ell^{*2}\delta^{*2}}, \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda \ell^{*2}\delta^*(1-q+qe^{-2\lambda})}\Big)$$

An empirical demonstration and validation of the tightness of lemma 11 using simulations w.r.t. λ and q is provided in Figure 3.34, and w.r.t. n in supplemental Figure 3.37. The simulations are described in supplemental.

Sufficient Conditions for $(\ell^*, d^*)-$**Oracle in the Presence of Missing Data:** Now we consider the possibility of missing data (dropout) and give several simple strategies to handle it. In our analysis we consider two types of dropout events that may occur. A stochastic dropout is an event that occurs in and affects an individual cell (leaf), e.g., due to the limited sensitivity of single-cell RNA sequencing. A heritable dropout is an event that affects an entire clade, e.g., due to resection. Note that although we assume dropouts occur independently in each character, dropouts observed in the same character at two different cells are not necessarily independent as they could have originated from the same heritable dropout event. Now let p_d be the probability that a particular character of a particular cell suffers either heritable or stochastic dropout. Let $(a, b|c)$ be an arbitrary triplet and let ϵ be the probability that no dropout occurred in a particular character in either a, b or c. The probability that at least one cell of this triplet suffers a dropout at a particular character is maximized when the three cells share no common lineage and minimized when the three cells are phylogenetically proximal. We therefore have that $(1-p_d)^3 \leq \epsilon \leq 1-p_d$ (see proof in supplemental). To account for this when revising our oracle definition, we now define $s(a, b)$ as the number of characters shared by a, b that do not have dropout in either a, b or c. Note that in this case the definition of $s(a, b)$ depends on c but is well defined for every triplet a, b, c so we can consider the same threshold-based triplet oracle before using this new similarity function. This means that for triplet $(a, b|c)$, we have the following:

$$E[s(a,b) - s(b,c)] \geq k\epsilon\lambda\ell^*\delta(d) \geq k(1-p_d)^3\lambda\ell^*\delta(d)$$

$$E[s(b,c)] \leq k\lambda^2(1-p_d)q$$

We can similarly define conditions i) and ii) with the threshold $t = \frac{1}{2}(1-p_d)^3 k\lambda\ell^*\delta^*$, and see that if these conditions hold, then we have an (ℓ^*, d^*) partial oracle. We can also apply the same Chernoff bounds from the previous sections to get the following conditions on k to ensure conditions i) and ii) hold.

Lemma 12. *(proof in supplemental): In the presence of missing data at a rate of p_d, condition i) holds with probability at least $1-\zeta$ if we we have the following guarantees on the parameters q, λ, ℓ^*, d^* and k:*

$$k \geq \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda \ell^{*2}\delta^*(1-p_d)^3(1-q+qe^{-2\lambda})}$$

Both conditions i) and ii) hold with probability at least $1-\zeta$ if we have:

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)q}{\ell^{*2}\delta^{*2}(1-p_d)^5}, \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda \ell^{*2}\delta^*(1-p_d)^3(1-q+qe^{-2\lambda})}\Big)$$

Top-down Algorithms for Oracle-Based Partial Tree Inference Given our results on the correctness of the $(\ell^*, d^*)-$oracle, we are now ready to define the respective algorithm. Assuming $\ell-$bounded edge lengths in the ground truth tree, we use an oracle in which ℓ^* is set to ℓ. With that $(\ell, d^*)-$oracle, the algorithm guarantees accurate reconstruction up to depth d^* when given a sufficiently large k.

Theorem 13. *In the lineage tracing regime, if*

$$k \geq \frac{(96\log n + 32\log 1/\zeta)(\ell + (1-e^{-\lambda})q)}{\lambda \ell^{2}\delta^*(1-q+qe^{-2\lambda})}$$

and all edges in $\mathcal{T}$ at depth at most d^* have length at least ℓ, there exist polynomial time algorithms that return a tree which correctly resolves all triplets whose LCA is at depth at most d^* with probability at least $1-\zeta$.

Proof: By taking $\ell^* = \ell$, condition i) will hold for all triplets $(a, b|c)$ whose LCAs are at depth at most d^* on $\mathcal{T}$ as $dist(LCA(a,b,c), LCA(a,b)) \geq \ell$ for all triplets. The above bound on k, by lemma 11, implies condition i) is satisfied with probability $1-\zeta$. It then suffices to show that whenever condition i) is satisfied, there exists a polynomial time algorithm that constructs a tree which correctly resolves all triplets whose LCAs are at depth at most d^*. We present a simple top-down recursive splitting algorithm which is guaranteed to return correct splits up to depth d^*. This algorithm has a runtime of $O(kn^2)$ for each recursive call, where n is the size of the input to the call.

To prove correctness, let V be the set of samples at some recursive call in the algorithm. Let r' be the LCA of V in $\mathcal{T}$ at depth $d \leq d^*$. Let L and R be the samples in the left and right subtrees of r' respectively. Let $a \in L$ and $b \in R$ be arbitrary. By condition i), we know that for any $c \in L$, $s(a,c) > s(a,b)$ and for any $c' \in R$, $s(b,c') > s(a,b)$. This means that whenever (a,b) is in the graph, and right after (a,b) is deleted from the graph if it ever happens, all neighbors of a and b will remain connected to them. Thus, the graph

Algorithm 1 Threshold Algorithm

procedure SPLITSAMPLES(V)
  $G \leftarrow$ Complete graph over V
  while G is connected **do**
    $(u^*, v^*) = argmin_{(u,v)\in E} s(u, v)$
    Delete (u^*, v^*) from G
  $C_1, C_2 \leftarrow$ connected components of G
  $T_1, T_2 \leftarrow SplitSamples(C_1),\ SplitSamples(C_2)$
  Return binary tree with T_1 and T_2 as children of the root.

must remain connected until all all edges in the cut $L|R$ are deleted, and L and R will still remain connected immediately after all cut edges are deleted, giving us the correct split. This means that the algorithm will keep returning correct splits as long as the LCA of all samples in a recursive call has depth at most d^*.

Proof of Corollaries 1 and 2: If we take λ and q as constants and $d^* = 1$, Theorem 13 implies that it is asymptotically sufficient to have $k = O(\frac{\log n}{\ell^2})$ in order to ensure exact recovery of the entire ground truth tree with high probability. When dropouts are taken into account in the general case, the bound becomes $k = O(\frac{\log n}{\ell^2(1-p_d)^3})$ by lemma 12 because $1/(1-p_d)^3$ factor is needed to ensure condition i) still holds. Notably, when only stochastic dropouts are considered, the bound becomes $k = O(\frac{\log n}{\ell^2(1-p_d)^2})$ by lemma 18 (see supplemental).
In the next theorem, we see that when we don't have a lower bound on edge lengths, we can still construct a tree that correctly resolves all triplets that are well separated.

Theorem 14. *In the lineage tracing regime, if the number of characters satisfy*

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)q}{{\ell^*}^2{\delta^*}^2}, \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda{\ell^*}^2\delta^*(1-q+qe^{-2\lambda})}\Big)$$

and ℓ^* is an arbitrary parameter, then there exist polynomial time algorithms that return a tree which correctly resolves all triplets, $(a, b|c)$ such that $dist(LCA(a, b, c), LCA(a, b) \geq \ell^*$ and at $depth(LCA(a, b, c)) \leq d^*$ with probability at least $1 - \zeta$.

Proof: Again, by lemma 11, it suffices to show that if conditions i) and ii) both hold, then there exists an algorithm which resolves all triplets, $(a, b|c)$ such that $dist(LCA(a, b, c), LCA(a, b) \geq \ell^*$ and at $depth(LCA(a, b, c)) \leq d^*$. We can apply the classical Aho's algorithm to recover a tree that is consistent with all triplets resolved by the $(\ell^*, d^*)-$oracle, which is guaranteed to us by conditions i) and ii). The algorithm is specified below for completeness; other supertree algorithms can be used here as well.

Let $\mathcal{R}$ be the set of triplets for which we received a non-NULL answer from the oracle, and let V be the set of leaf nodes. Note that $\mathcal{R}$ must include all triplets that are at depth at

most d^* and whose internal nodes are separated by an path of length at least ℓ^*, since they satisfy condition i). It may also include incorrect triplets that are of depth more than d^*.

Algorithm 2 Aho's Algorithm

```
1: procedure AHO(R, V)
2:     E ← ∅;
3:     for (u, v|w) ∈ R do
4:         E ← E ∪ {(u, v)}
5:     if G = (V, E) is connected then
6:         If V is a singleton, return the single vertex.
7:         Otherwise, return an unresolved star-tree with V as leaves.
8:     else
9:         T ← new tree with root r'
10:        C_1, C_2, ...C_k ← connected components of G
11:        for i = 1 to k do
12:            R_i = {(a, b|c) ∈ R : a, b, c ∈ C_i}
13:            T_i ← AHO(R_i, C_i)
14:            Connect each T_i to T by adding an edge from r'.
15:        Return T
```

To prove the correctness of the algorithm, we must show that if $(a, b|c)$ is a triplet in the ground truth tree with $depth(LCA(a, b, c)) \leq d^*$ and $dist(LCA(a, b), LCA(a, b, c)) \geq \ell^*$, then $(a, b|c)$ is correctly resolved in the tree returned by the algorithm. First, assume by contradiction that the triplet is represented wrongly (WLOG) as $(a, c|b)$ in the returned tree. The presence of a wrong triplet $(a, c|b)$ in the returned tree means that at some point, there was a recursive call on a set of leaves, $V \ni a, b, c$ such that a, c were in a connected component of G not containing b. However, condition i), combined with the assumption that $dist(LCA(a, b), LCA(a, b, c)) \geq \ell^*$, implies that if $a, b, c \in V$ then there is an edge between a and b, which means b must be in the same connected component as a and c.

Next, assume by contradiction that a, b, c all have the same parent in the tree returned by the algorithm. First note that this cannot happen in line 15. This follows trivially since by definition a and b are initially in the same connected component. Therefore, the only way a trifurcation can happen is if a connected component that contains a, b and c is split into three or more component (with a, b and c on different components), each sent to a separate recursive call (line 14). This cannot happen since the presence of a, b and c entails the inclusion of an edge between a and b in that component (per line 4). This means that there was a recursive call on a set of leaves $V \ni a, b, c$ such that the connectivity graph, G over V is connected (i.e. the algorithm reached step 8). Let r' be the LCA of all vertices of V in $\mathcal{T}$. Let L and R be the vertices in V descended from the left and right children of r'

respectively. Since $depth(r') < d^*$ condition $ii)$ implies that any triplet with all leaves in V will be either correctly classified or assigned "Null" by the oracle. But, there are no edges between L and R, which means V is not connected, thus arriving at a contradiction once again. Thus, the only possibility is for $(a, b|c)$ to be correctly classified in the inferred tree.

Proof of corollary 3: We take λ and q as constants and $d^* = 1$. Additionally, we make no lower bound assumptions on the edge length ℓ and take ℓ^* to be an arbitrary parameter. Theorem 14 then implies that it is asymptotically sufficient to have $k = O(\frac{\log n}{\ell^{*2}})$ in order to ensure exact recovery of all triplets $(a, b|c)$ such that $dist(LCA(a, b, c), LCA(a, b) \geq \ell^*$ with high probability.

Simulations for the Threshold Algorithm:

Theorem 13 gives a lower bound on the number of characters k sufficient for exact phylogenetic reconstruction in the case where there is a minimum edge length ℓ. In order to both validate our asymptotic relationships between experimental parameters, as well as get a better sense for the number of characters that may be necessary in practice (a number that may be lower than our estimated of sufficiency), we turn to simulations. Specifically we chose to explore the empirical number of k necessary for exact reconstruction (all triplets resolved correctly) up to some depth d^* and as a function of the state collision probability q and the mutation rate λ (see supplemental for description of our simulations).

Figures 3.34A,C depict the dependence of k for high probability (0.9) of exact reconstruction with varying λ and q for $\ell = \frac{1}{9}$ or 0.05. We observe that in the regions where $q > \ell/\lambda$, the sufficient k increases sharply. This is since for lower values of q the asymptotic requirement for k becomes $O(\frac{log(n)}{\ell})$, whereas for higher values of q we get the general result of $O(\frac{log(n)}{\ell^2})$. Additionally, we observe interesting behaviors in λ. In particular when d^* is large enough (requiring exact reconstruction of the entire ground truth phylogeny or its top half), both excessively small and large values of λ lead to a larger requirement for k. Intuitively this is due to the lack of mutations or due to mutation saturation, in both cases leading to less informative input (Figure 3.34H). When the goal becomes partial reconstruction of only the top (20%) of the phylogeny and d^* is small, k no longer increases with large λ. This is because δ^* in the denominator of the bound shifts from $e^{-\lambda}$ to 1 as $d^* \to 0$. Intuitively, towards the top of the phylogeny characters are yet to be saturated, allowing cells whose LCAs are near the top of the tree to be resolved. This suggests that saturation is less problematic if only distal relationships need to be resolved correctly, i.e. in the case of $d^* << 1$.

In order to test the performance of the Threshold Algorithm in realistic settings, we simulate CRISPR-Cas9 induced phylogenies over two topological regimes: one with uniform edge lengths separating cell divisions with $\ell = \frac{1}{9}$ and one with an asynchronous cell division topology ($\ell = 0.05$) described in supplemental (Figure 3.34B,D). The first regime aims to

mimic a cell division process that has a regular molecular clock with bounded edges lengths both from above and below. The second regime is meant to mimic a more general stochastic cell division process that only has a minimum bound on edge length.

We find that the theoretical analysis and the simulations are consistent, both in terms of the direction dependence on the parameters, and on the inflection points at which the minimal k increases more rapidly. The largest discrepancies in the trends occur in the regions in which λq is high. In these regions the empirical increase in k is not nearly as sharp as the theoretical bound suggests. Hence the theoretical bound overestimates k relative to other values in these regions. Additionally, we observe that as d^* decreases the empirical k decreases (Figure 3.34I). This last result validates the trend in the bounds regarding d^*. Ultimately, the theoretical estimate predicts the empirical trends well, however, we do find that the absolute number of necessary characters (as found by the simulation) is much lower than the theoretical estimate.

Overall, these simulations provide validation that the asymptotic trends on k given by the theoretical parameters apply in realistic scenarios under the Threshold Algorithm. In addition, we provide necessary conditions for the number of characters required for exact reconstruction via the Threshold Algorithm for a series of parameter regimes.

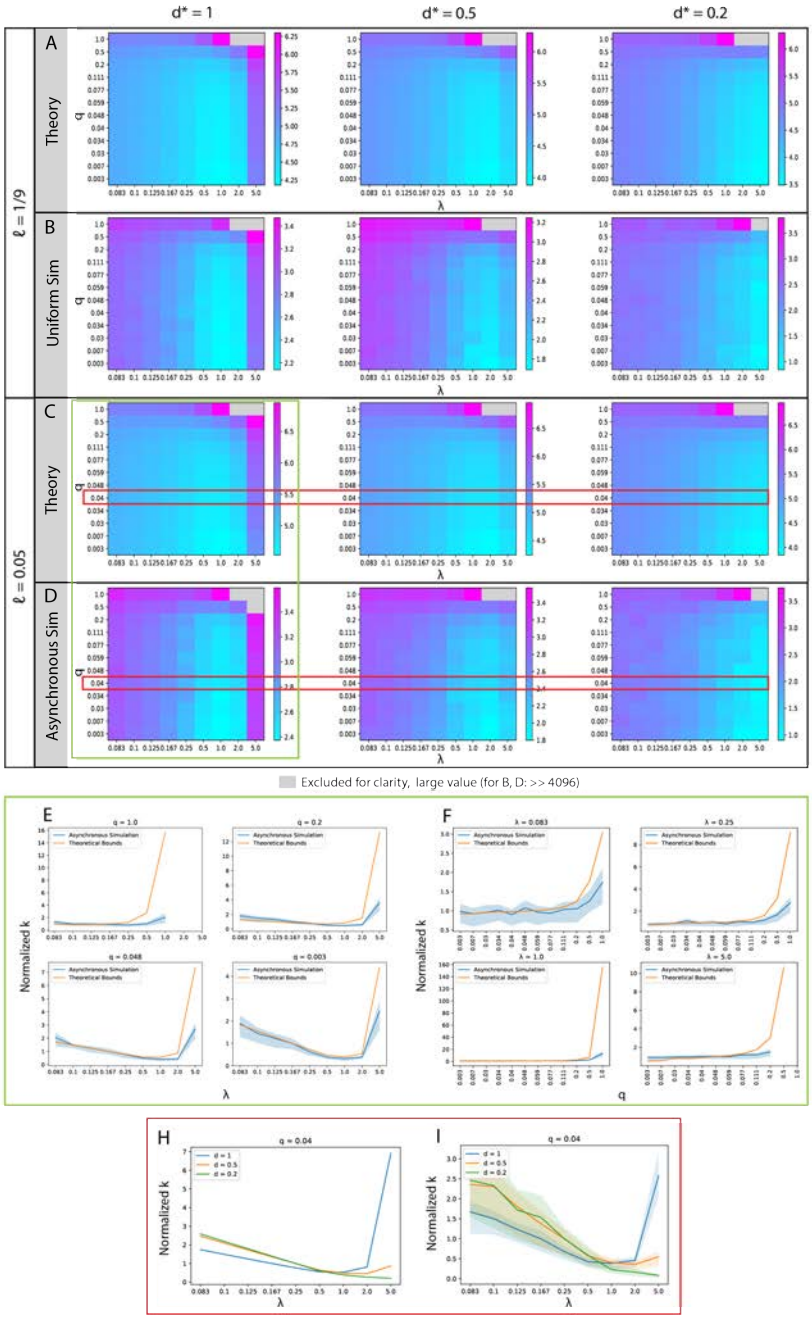


Figure 3.34: **Comparing the Threshold Algorithm in Theory and Simulation**. Simulated trees with 256 leaves, $n = 256$. (A, C) Theoretical sufficient lower bound on k required for 0.9 probability of perfect tree reconstruction up to depth d for varying values of d, q and λ for (A) $\ell = 1/9$ and (C) $\ell = 0.05$. (B, D) Minimum k required for 0.9 probability of perfect tree reconstruction in simulation with a cell division topology with (B) uniform edge lengths ($\ell = 1/9$) and with (D) an asynchronous cell division topology ($\ell = 0.05$). (A-D) Entries are log_{10} scaled. (E, F) Plots comparing the dependence of the minimum k in simulation with the theoretical bound on varying parameters (0.9 probability of perfect reconstruction, $\ell = 0.05$, $d^* = 1$). We report the dependence of k on (E) λ for fixed values of q and (F) q for fixed values of λ. (H) Comparison of the dependence of the bound on k for 0.9 probability of perfect reconstruction on λ for various values of d. (I) Comparison of the dependence of minimum k for 0.9 probability of perfect reconstruction in the asynchronous simulation on λ for various values of d. (E, F, I) For ease of comparison, the values of k are rescaled by the median value of k in each line. (E, F, I) Point-wise 95% confidence intervals for the minimum k in simulation are generated from the regression coefficients using the delta method, see supplemental.

Upper-Bounded Edge Lengths and Bottom-Up Approaches:

In the previous sections, we saw that when λ and q are fixed to be a constant, then the number of characters needed for exact recovery with high probability is $O(\frac{\log n}{\ell^2})$, where ℓ is the minimum edge length. It can also be shown that if we are able to bound λ and q such that $\lambda q \leq \ell$ then the bound becomes tighter: $k = O(\frac{\log n}{\ell})$. However, while λ can be controlled experimentally by calibrating the affinity of the lineage tracer's guide RNAs, there is currently no way to control the entropy of the indel state distribution - a quantity which relies on the endogenous DNA repair process.
In order to achieve this tighter bound without direct dependence on q, we instead use an additional assumption on ℓ - namely that there is some maximal (in addition to a minimal) possible period of time between the birth of a given node and the birth of its parent. If our set of leaves includes all the cells in the phylogeny, then this translates to an upper bound on the time between cell divisions. In the more common scenario of sampling only a small subset of cells from a given clone, each edge in the ground truth tree can correspond to a series of cell division events. However, in either case, a strict upper bound on the length always exists, corresponding to the duration of the lineage tracing experiment. In the following section, we show that with such an upper bound on edge length, we can achieve exact recovery with high probability when $k = O(\frac{\log n}{\ell})$, provided an upper bound on the probability of the most likely mutation ($\max_j(q_j)$) and on λq. The latter bound can be less strict than ℓ, depending on the ratio between the lengths of the longest and shortest edges. More importantly, under these revised assumptions, we can achieve a bound on the minimal required k that is independent of the value of q.

Theorem 15. *(proof in supplemental): Let $\mathcal{T}$ be a tree of height 1 over n leaves. Let ℓ and c be constants such that each edge, $(u, v) \in \mathcal{T}$ has $\ell \leq l(u, v) \leq \sqrt{c\ell}$. Suppose that for each character we have: $q_{\max} := \max_j(q_j) \leq \frac{3}{16(1-e^{-\lambda\sqrt{c\ell}})}$ and $\lambda q < \frac{\beta}{\max(1,c)}$ where $\beta < \frac{1}{1+C+2(e^{-\lambda\ell}+2\lambda\sqrt{c\ell}q_{\max})^2}$ and $C = 2\sqrt{c\ell}e^{-\lambda} + 4\lambda c\ell q_{\max}$. Then there exists an algorithm that, with high probability, will recover $\mathcal{T}$ if the number of characters satisfies:*

$$k \geq \frac{20\log n + 10\log(1/\zeta)}{\lambda e^{-\lambda}\ell(1-\beta(1+C))(1-\beta(1+C)-2\beta(e^{-\lambda\ell}+2\lambda\sqrt{c\ell}q_{\max})^2)}$$

If we take the limit as $\ell \to 0$ or $n \to \infty$ we get the following result:

Corollary 5. *Let $\mathcal{T}$ be a tree of height 1 over a sufficiently large number of leaves n. Define ℓ, c and $q_{\max}$ as in Theorem 15 with similar bounds. If $\lambda q \leq \frac{\beta}{\max(1,c)}$ where $\beta < \frac{1}{3}$ then there exists an algorithm that can, with high probability, recover $\mathcal{T}$ if the number of characters satisfies:*

$$k \geq \frac{20\log n + 10\log(1/\zeta)}{\lambda e^{-\lambda}\ell(1-\beta)(1-3\beta)}$$

Note that the bound on β is not the tightest possible, and it was chosen to simplify calculations. Additionally, we present an alternative analysis in supplemental (Theorem 16) that yields a bound which has a looser constraint on the λ and q parameters as ℓ tends to 0. Consider the following greedy algorithm which iteratively joins partially constructed subtrees by picking the pair with the most similar roots, and then joining them by inferring a new root by maximum parsimony. Let S denote the set of subtrees at any particular iteration. Let $T \in S$ denote an inferred subtree, and let $r(T)$ denote the root of that subtree. Let $r(T)_i$ denote the state of the inferred i^{th} character of $r(T)$.

Algorithm 3 Bottom-Up Algorithm

```
procedure BUILD TREE
    S ← leaves of 𝒯
    while |S| > 1 do
        T_1, T_2 = argmin_{(T,T')∈S} s(r(T), r(T'))
        r ← new node
        for i = 1 to k do
            if r(T_1)_i = r(T_2)_i then
                r_i ← r(T_1)_i
            else
                r_i ← 0
        T ← new tree with r as root and T_1 and T_2 as subtrees under the root
        S ← S ∪ {T}
```

(We note a similar algorithm has been presented in Sugino et. al [169], although no theoretical guarantees on accuracy are given in that work.) In the supplemental, we prove that under the conditions in Theorem 15, this algorithm correctly returns the ground truth tree $\mathcal{T}$ with high probability. An empirical demonstration and validation of the tightness of Theorem 15 using simulations w.r.t. λ and q is provided in Figure 3.35, and w.r.t. n in supplemental Figure 3.37. The simulations are described in supplemental.

Proof of Corollaries 4 and 5: The Bottom-Up Algorithm shows that there exists a polynomial time algorithm that ensures $k = O(\frac{\log n}{\ell})$ characters is sufficient asymptotically for exact recovery of the tree. As $n \to \infty$ we get that $\ell \to 0$ and $C \to 0$. With these we get that $\beta < \frac{1}{3}$. With the simpler bound on β, we also get that $k = O(\frac{\log n}{\ell})$ characters are sufficient asymptotically for exact recovery of the tree.

Simulations for the Bottom-Up Algorithm:

As above, we begin by examining the theoretical bounds for the necessary k. In particular, Figure 3.35A visualizes the bound for k across varying values of λ and q for high probability (0.9) of exact reconstruction. We consider two regimes: one with $\ell = 1/9, c = 1/9$ and one with $\ell = 0.05, c \approx 3.85$. Since q does not explicitly appear in the bound for k, we

instead use it to define a value for β, using its lower bound: $\beta := \lambda q \cdot max(1, c)$. Plugging this in provides a lower bound for the necessary k, which we plot here. Regions where the lower bound on β becomes larger than its upper-bound requirement (per theorem 15) are excluded.

From this figure it can be seen that k depends on λ in the same way as in Theorem 13. That is, k increases significantly for both excessively small and large values of λ. However, there is a contrast in the dependence of k on q. Although in the bound for Theorem 15 k does increase with q through the dependence of β, k is not as sensitive to large values of q as in Theorem 13. Further, as the bound is quadratic in $\frac{1}{1-\beta}$, the k increases rapidly with respect to $\beta := \lambda q \cdot max(1, c)$.

We tested the Bottom-Up Algorithm in the same simulation regimes (same tree and lineage tracing parameters) as the Threshold Algorithm (Figure 3.35B). Concordant with the theoretical results, we observed that the minimum required k is less sensitive to q, compared with the Threshold Algorithm. Furthermore, in both results we see similar trends in dependence on λ (Figure 3.35C, E), and q (Figure 3.35D, F). The main discrepancy between the theory and the simulation occurs where $\beta := \lambda q \cdot max(1, c)$ approaches our upper bound for β (i.e., the values that border the regions that were excluded from Figure 3.35A). In those cases,we see that the theoretical bound is looser and overestimates k relative to the simulations.

These simulations validate the relationships observed in the asymptotic trends on k and give tighter empirical conditions on the necessary k for exact reconstruction. We observe that the empirical necessary k in the Bottom-Up Algorithm is overall lower than that of the Threshold Algorithm, except the cases of non-uniform edge length with high value of c in which the minimal k is comparable (Figure 3.35A-B right, and Figure 3.35E-F). These results suggest that the Bottom-Up Algorithm can achieve exact reconstruction with fewer characters empirically than the Threshold Algorithm, but requires that the variance in the division times of the ground truth phylogeny to be small (corresponding to the assumption on upper bounded edge lengths).

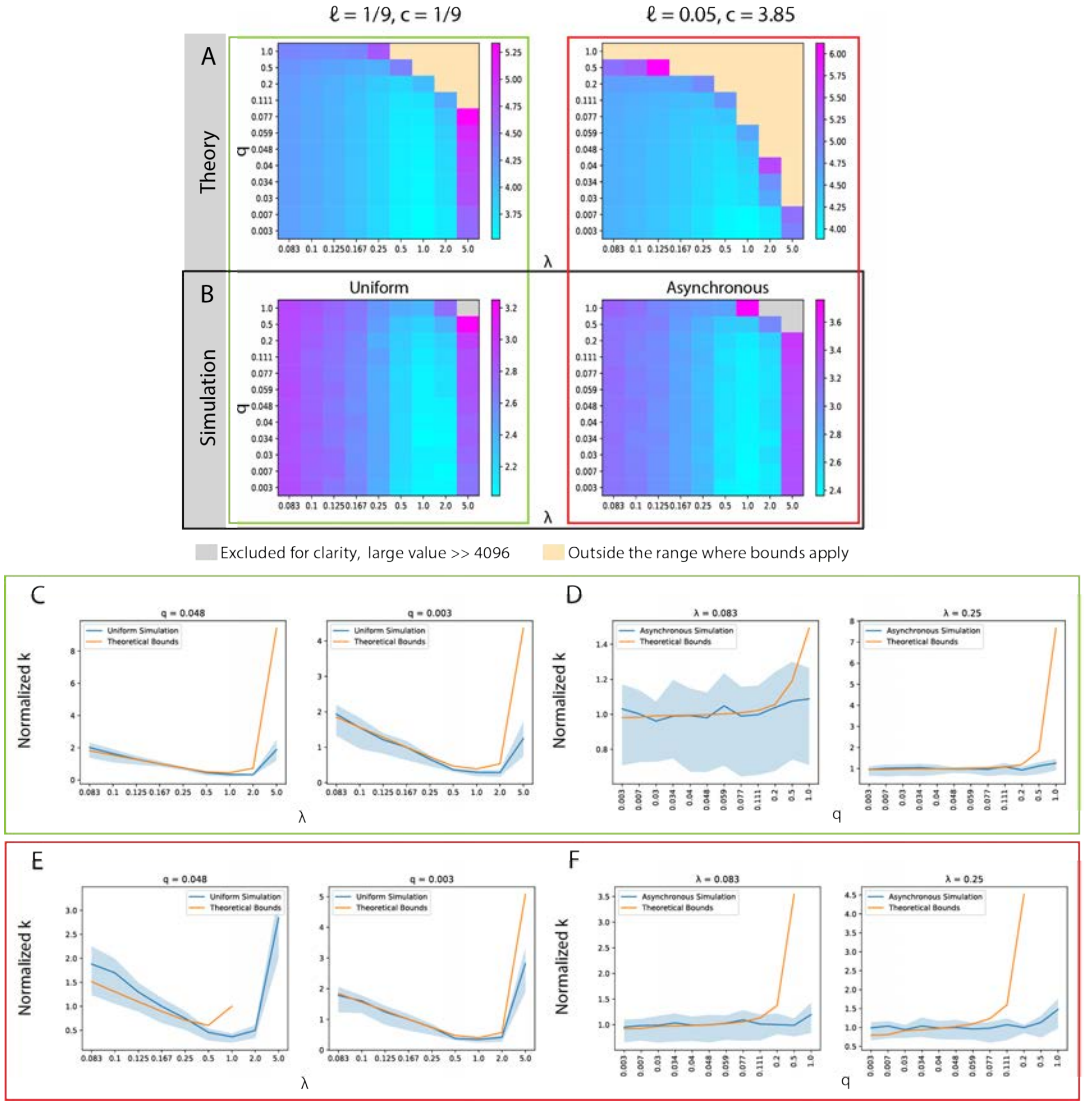


Figure 3.35: **Comparing the Bottom-Up Algorithm in Theory and Simulation**. Simulated trees with 256 leaves, $n = 256$. (A-B) Entries are log_{10} scaled. (A) Theoretical sufficient lower bound on k required for 0.9 probability of perfect tree reconstruction on varying values of q and λ, taking $\beta = \lambda q \max(1, c)$. As the state distributions are uniform, $q_j = q$ for each value of q. Left: $\ell = 1/9$ and $c = 1/9$ for comparison with the simulated case where the branch lengths are uniform. Right: $\ell = 0.05$ and $c \approx 3.85$ such that >99.99% of simulated branch lengths in the realistic simulation regime fall within the upper bound. (B) Minimum k required for 0.9 probability of perfect tree reconstruction in simulations, with Left: a cell division topology with uniform branch lengths, $\ell = 1/9$ and Right: an asynchronous cell division topology (description in supplemental), $\ell = 0.05$. (C-F) Plots comparing the dependence of the minimum k in simulation with the theoretical lower bound on varying parameters (0.9 probability of perfect reconstruction). We report the dependence on (C, E) λ for fixed values of q and on (D, F) q for fixed values of λ, in simulations with uniform edge lengths (C-D) and with asynchronous topologies (E-F). (C-F) For ease of comparison, the values of k are rescaled by the median value of k in each line. Point-wise 95% confidence intervals for the minimum k in simulation are generated from the regression coefficients using the delta method, see supplemental.

3.3.5 Discussion:

In this paper we have established sufficient conditions for high probability of exact reconstruction of the ground truth phylogeny in the CRISPR-Cas9 lineage tracing setting. These guarantees show that despite complications with the lineage tracing process such as homoplasy, missing data, and lack of mutation information, exact reconstruction can still be achieved given sufficient information capacity in the experiment (as measured by the number of recording sites). In addition to showing the feasibility of exact reconstruction, these theoretical results relate the difficulty of the reconstruction problem in the number of sufficient characters to the experimental parameters. We anticipate these results can inform researchers as to how to reduce the number of necessary characters or best aid downstream reconstruction of the phylogeny given the available number of characters through careful engineering of CRISPR-Cas9 lineage tracing experiments.

The theoretical results shown here provide insight into how the CRISPR-Cas9 lineage tracing experimental parameters relate to the reconstruction problem. One key insight is that for exact reconstruction, a mutation rate that is neither too high or low gives the best results, which is in line with the intuition that a middling rate balances mutation saturation and mutation-less edges. We also formalize the intuition that having a state distribution with a low rate of collision q makes the reconstruction problem easier in avoiding homoplasy. Additionally we show the difficulty that missing data poses for the problem of exact reconstruction. Using the presented methods, the number of additional characters needed to overcome missing data is cubic (quadratic in the case of only stochastic missing data) in $\frac{1}{1-p_d}$, where p_d is the probability of missing data. A final key result is that the (ℓ^*, d^*)-oracle in the Top-down Algorithms allows researchers to tailor the granularity of their reconstruction accuracy to what is achievable given the number of available characters. Here, substantially fewer characters are required if one is only interested in correctly resolving triplets that diverged early in the tree (small d^*) and well-separated triplets (large ℓ^*, regardless of the true minimum edge length ℓ).

Although having more characters is preferable, we recognize that currently there are practical limits on the number of recording sites that can be incorporated into CRISPR-Cas9 systems. Current methods to incorporate recording sites into the genome (such as lentiviral transduction [130, 98, 145] or transposition [146, 147, 27]) are limited by the low uptake of these sites into the progenitor cell, only offering on the scale of tens of recording sites [98, 147, 146, 145, 130, 27, 165, 4]. One alternate technology of particular interest is the base-editor, which uses a modified Cas9 complex to induce direct base-pair substitutions [7]. Base-editors, while yet to be explored in lineage tracing contexts, have the potential to offer one hundred or more editable sites [82], although careful engineering is required as q is high in these regimes owing to the limited state space of nucleotide outcomes. Ultimately though, we see in our simulations that even this increase in characters is far insufficient for exact reconstruction in most settings, especially considering the considerable amounts of missing data and the large number of samples (n) that we see in real CRISPR-Cas9 lineage tracing experiments. We thus challenge the field to develop systems that allow for considerably more

characters.

The limitations in adding more characters motivates the optimization of the other experimental parameters in engineering CRISPR-Cas9 lineage tracing experiments. We discuss here current and potential strategies for engineering the discussed parameters: the editing rate, the collision probability, and the missing data probability. There is a large body of literature showing that the editing activity of Cas9 (as in lineage tracing experiments) can be tuned with relative precision using mismatches between the guide RNA and recording site [27, 155, 98, 145]. In regards to the collision probability, experimenters are currently unable to dictate the collision rate in state outcomes due to the random indel outcomes of Cas9 editing. Recent strategies - such as pairing terminal deoxynucleotidyl transferase (TdT) with Cas9 [179] - have shown potential to increase indel diversity. Further, prime editors offer an avenue to more finely control the state distribution by dictating a-priori which indel will result in a given edit, though this technology has yet to be adopted for lineage tracing [8, 33, 36]. Fortunately, current CRISPR-Cas9 lineage tracing systems are capable of generating "un-problematic indel distributions". In those cases, the collision probabilities q lie outside of the range where k explodes with q [27, 98, 145]. Additionally, we see in the bounds and simulations that decreasing q has diminishing returns on k. Taken together, in designing CRISPR-Cas9 lineage tracing systems effort is better put on carefully engineering the Cas-9 cutting rate than optimizing the state distribution. Unfortunately, current strategies to control for missing data experimentally are more limited. Experimenters are at the mercy of the efficacy of single-cell assays in the case of low capture (leading to stochastic missing data), but can attempt to control the rate of resection and transcriptional silencing (both of which leading to heritable missing data). Recent designs have mostly relied on distributed designs to reduce the rate of resection, utilizing many "cassettes" (DNA segments that contain many proximal recording sites) with a small number of recording sites per cassette [165, 27, 98, 145]. Although not addressed in current designs, transcriptional silencing can be potentially limited by placing recording sites in regions of the genome that are more robust to silencing ("safe-harbor" regions) using emerging methods for guided transposition [114].

In addition to motivating the design of CRISPR-Cas9 lineage tracing experiments, our model motivates theoretical and algorithmic development for these systems. The sufficient bounds that we reach in our asymptotic analyses are not tight, as demonstrated by simulation. We believe that these bounds can likely be further improved to give a better sense of the necessary k analytically. Future approaches may take advantage of aspects of the model or engineering designs that are not leveraged in this work. For example, in our analysis we assume that the mutation rate λ is constant throughout the entire phylogeny and across recording sites. However, using a gradually increasing mutation rate or designing characters with variable rates, whose affinity is estimated a-priori may lead to better reconstruction results. Such a design can simultaneously alleviate issues of mutation saturation near the leaves of the tree as well as lack of sufficient mutations near the root. We also assume that the characters mutate and acquire missing data independently, although indels and missing data events can span multiple recording sites. Future approaches could take advantage of the structure present in these multi-site events. Finally, although our analysis handles missing

data by ignoring missing characters, the structure of heritable data offers additional information that could be better leveraged (i.e., utilized in the same way as any other mutation). The challenge, naturally, is to distinguish between the two types of missing data.
Here, we perform a first theoretical analysis of CRISPR-Cas9 phylogenetic reconstruction. In doing so, we have developed a generative model for this type of data, which we hope will frame future analysis of CRISPR-Cas9 lineage tracing systems, akin to the Jukes-Cantor model in other molecular phylogenetic studies. With this theoretical framework and the accompanying algorithms, our work naturally complements recent efforts to develop and understand algorithms for this lineage tracing data [69]. Ultimately, we believe that this work will continue to inform and orient both algorithmic and experimental methods as the technology and field evolve.

3.3.6 Materials and Methods:

Simulations and algorithms are implemented in Python in Cassiopeia software suite [98] (https://github.com/YosefLab/Cassiopeia), utilizing the NetworkX package [76]. Implementation specifics are provided in the supplemental.

Simulating Lineage Tracing Experiments:

In our simulations, we simulated forward-time lineage tracing experiments using our generative model. We split the simulation into two steps.

Simulating Cell Division Topologies: First, we simulate a continuous-time, binary, symmetric cell division topology. Then, we simulate CRISPR-Cas9 lineage tracing data over the given topology. The end result is a phylogenetic tree representing the single-cell lineage tracing experiment. The tree topology also records the ground truth phylogenetic relationships between the observed cells.
We begin by describing the two simulation schemes used for the tree topology. The first scheme simulates a cell division regime with regular cell division (uniform edge lengths). We start with a complete binary tree and add an implicit root, attaching this root to the root of the complete binary tree by an edge. The edge represents the lifetime of the root along which mutations can be acquired. For all figures besides Figure 3.37, we generated trees with 256 leaves representing cells observed at the end of the experiment. For Figure 3.37, we generated trees of various sizes, exponentially increasing n. Given that the number of edges in the path from the implicit root to each leaf has $log(n) + 1$ edges, each edge has uniform length, and the length of the experiment is normalized to one, each edge has length $\frac{1}{log(n)+1}$.
The second simulation scheme represents an asynchronous cell division regime, with stochastic waiting times between cell divisions and cell death. We model a forward-time Bellman-Harris model with extinction [77]. This generalizes the birth-death process [140], a commonly used phylogenetic model, such that the distribution of waiting times between division and death events are arbitrary. In our case, waiting times between division events are distributed

according to an exponential distribution that is shifted by a constant $a = 0.05$, representing minimum time between cell division events. The distribution of death waiting times is distributed exponentially, as we assume that cell death does not have a minimum time.The stopping condition is when all lineages reach time = 1, meaning that each lineage will have total path length from the root of 1.
To control for the number of leaves in the simulated trees, we take only trees that have between 205 and 307 leaves (if the procedure terminates with no living lineages, then we consider that tree to have 0 leaves). Note that in the trees generated by this process sister nodes need not have the same edge length and the root will have a singleton edge as in the binary case along which mutations can occur. We chose rates for the division and death waiting distributions (23.70 and 2.12, respectively) that gave an average of around 256 leaves over 1000 simulations. These rates were chosen assuming that the rate of division was 10 times that of the rate of death, and then correcting the death rate to increase the mean waiting time by $a = 0.05$ to match the shift in mean in the distribution of division times.
Due to the stochastic nature of this division process, we cannot exactly control for ℓ if we stop the experiment at a specified time. This is due to the fact that edges at the leaves of the tree may be very small if the stopping criterion is reached before the length can reach a, thus making the minimum edge length in the tree technically potentially smaller than a. We contend that these edges should not impact the analysis though. Small edges make it difficult to discern which neighboring clades are actually closer in relation. These small edges only occur at the bottom of the tree though and would only affect the edge lengths leading to single leaves, which are trivially discerned as a cherries with their neighbors, meaning that ℓ would still effectively be 0.05 in this case.

Simulating CRISPR-Cas9 Lineage Tracing Data: Given a tree topology, we simulate a CRISPR-Cas9 mutagenesis process over it. Along each edge with length t, independently for each character, we simulate the probability that a mutation will occur as $1 - e^{\lambda t}$. If a character has been chosen to mutate, we then draw from the state distribution to determine the state the character acquires. In this case, this is a uniform distribution with $m = 1/q$ states (note that in the uniform case, $q = \sum_{j=1}^{m} \frac{1}{m}^2 = 1/m$). Once a mutation is acquired on an edge, that mutation persists in all downstream nodes. Then the mutations acquired along the path from the implicit root is maintained for each leaf node, which then forms the observed character information for all observed cells. If the simulation involves missing data, then p_s proportion of characters are uniformly at randomly changed to a state representing missing data (-1). This character information is the input to the reconstruction algorithm.

Simulating the Minimum Necessary k for each Algorithm:

Here we describe the process by which we determined the minimum k necessary for 90% probability of a given reconstruction criterion in our simulations: either full reconstruction, partial reconstruction for triplets whose LCA is up to depth d, or triplets correct. For a given value of k and a given a set of parameters, we verify if it is sufficient for 90% probability

of full reconstruction as follows: we simulate 10 ground truth trees, reconstruct each tree from its observed cells (leaves) using the relevant algorithm, and compare the corresponding reconstructed tree to each ground truth tree. If ≥ 9 out of those 10 trees meet a scoring criterion, then we say that this k is sufficient. To alleviate the effect of noise, if 7-8 out of 10 trees meet the criterion, then we construct 20 additional trees and say k is sufficient if $\geq$ 18 of those trees meet the criterion.
To efficiently explore the space of k, we first exponentially (base 2) increase the value of k until a max value is reached (4098 in the case of no missing data and 16384 in the case of missing data). Once we find a sufficient k, we perform a binary search in the bin between that value and the value before it. Finally, we record the number of trees correctly reconstructed out of 10 for each value of k in the binary search and perform a logistic regression on these data points. We report the value of k that first reaches 90% reconstruction probability predicted by the logistic regression. If no k is sufficient up to the max value, then we deem that the necessary value of k is too large for our simulations to discern and we report a missing value. To calculate the point-wise confidence intervals used in Figures 3.34, 3.35, 3.37 for each regression on a set of parameters, we calculate the upper and lower bounds of the 95% confidence interval from the regression coefficients using the delta method. Then, we take the upper bound on the necessary k as the first k where the lower bound exceeds 90%, and we take the lower bound as the first k where the upper bound exceeds 90%.

3.3.7 Supplemental Information

Proofs

Analysis of the δ Function Since $\delta(d)$ is the only part of the bound that varies according to depth, any universal threshold to the oracle decision must take it into account. It can be verified that $\frac{d^2}{dx^2}\delta(x) \geq 0$ which means δ is convex. Setting $\delta'(x) = 0$, we have that the minimum of δ occurs at:

$$\begin{aligned} 0 &= -\lambda e^{-\lambda x}(1-q) + q\lambda e^{\lambda x - 2\lambda} \\ e^{-2\lambda x} &= \frac{q}{(1-q)e^{2\lambda}} \\ x &= \frac{1}{2\lambda}\ln(\frac{1-q}{q}) + 1 \end{aligned}$$

Let x^* be the minimum of δ. If $q < 1/2$, then $\ln(\frac{1-q}{q}) > 0$, which means that $x^* > 1$, so on the interval $[0, 1]$, the minimum occurs at $x = 1$, at which point $\delta(x) = e^{-\lambda}$. On the other hand, if $q \geq 1/2$, then the minimum will occur at $\delta(x^*) = 2e^{-\lambda}\sqrt{q(1-q)}$ if $x^* \in [0, 1]$ and $\delta(0) = 1 - q + qe^{-2\lambda}$ if $x^* < 0$. Note, x^* gets smaller as $q \to 1$, so if $q = 1$, then the minimum is precisely $e^{-2\lambda}$. Let $d^* \in [0, 1]$ be an arbitrary depth, and let $\delta^*(d^*, q, \lambda) = \min_{x\in[0,d^*]} \delta(x)$.

We then have:

$$\delta^* = \begin{cases} (1-q) + qe^{-2\lambda} & \text{if } \frac{1}{2\lambda}\ln(\frac{1-q}{q}) + 1 < 0 \\ 2e^{-\lambda}\sqrt{q(1-q)} & \text{if } \frac{1}{2\lambda}\ln(\frac{1-q}{q}) + 1 \in [0, d^*] \\ e^{-\lambda d^*}(1-q) + qe^{-\lambda(2-d^*)} & \text{if } \frac{1}{2\lambda}\ln(\frac{1-q}{q}) + 1 > d^* \end{cases}$$

Proof of lemma 10: Let $(a, b|c)$ be a triplet with $depth(LCA(a, b, c)) = d$, and let $\alpha = dist(LCA(a, b, c), LCA(a, b))$. Let $Y = s(a, b)$ and $X = s(b, c)$. For a particular character χ_i, let $Y_i = \mathbb{1}_{\chi_i(a)=\chi_i(b)}$ and let $X_i = \mathbb{1}_{\chi_i(b)=\chi_i(c)}$. We use the following results:

Lemma 13. *If for every character χ_i, $P[Y_i - X_i = -1]$ is a decreasing function of α and $P[Y_i - X_i = 1]$ is an increasing function of α for all $\alpha \in [0, 1]$, then for any t, $P[Y - X \geq t]$ is an increasing function of α.*

Proof: For a given character χ_i, $Y_i - X_i = \mathbb{1}_{\chi_i(a)=\chi_i(b)} - \mathbb{1}_{\chi_i(b)=\chi_i(c)}$ has 3 possible outcomes: $\{1, 0, -1\}$. Thus, if $P[Y_i - X_i = -1]$ is a decreasing function of α and $P[Y_i - X_i = 1]$ is an increasing function of α for all $\alpha \in [0, 1]$, then $P[Y_i - X_i \geq t]$ for any t is an increasing function of α. Stating that $P[Y_i - X_i \geq t]$ is an increasing function of α is identical to stating that for any α_1 and α_2 such that $\alpha_1 \geq \alpha_2$, $P_{\alpha_1}[Y_i - X_i \geq t] \geq P_{\alpha_2}[Y_i - X_i \geq t]$. We use the known result from probability theory that for random variables $A_i \overset{iid}{\sim} A$ and $B_i \overset{iid}{\sim} B$ such $P[A \geq t] \geq P[B \geq t]$ for all t, then $P[\sum_i A_i \geq t] \geq P[\sum_i B_i \geq t]$. Thus, as $Y - X = \sum_i Y_i - X_i$ and $Y_i - X_i$ is independent and identically distributed as we assume each character operates independently and identically, then $P_{\alpha_1}[Y - X \geq t] \geq P_{\alpha_2}[Y - X \geq t]$. Thus, $P[Y - X \geq t]$ is an increasing function of α for any t.

Lemma 14. *For every character χ_i, $P[Y_i - X_i = -1]$ is a decreasing function of α and $P[Y_i - X_i = 1]$ is an increasing function of α for all $\alpha \in [0, 1]$. Additionally, this result holds in both the general case and stochastic-only missing data cases.*

Proof: We prove this lemma in each case:
Case with no missing data: First, we examine $P[Y_i - X_i = -1]$. $Y_i - X_i = -1$ for a character χ_i corresponds to that character acquiring the same mutation on both the path from $LCA(a, b)$ to b and the path from $LCA(a, b, c)$ to c, and not acquiring that mutation on the path from $LCA(a, b)$ to a. Additionally, no mutation must be acquired at χ_i on the path from the r to $LCA(a, b, c)$ nor the path from $LCA(a, b, c)$ to $LCA(a, b)$. Thus we have:

$$P[Y_i - X_i = -1] = \sum_j e^{-\lambda d} e^{-\lambda\alpha}(1 - e^{-\lambda(1-d-\alpha)})q_j(1 - e^{-\lambda(1-d)})q_j(1 - (1 - e^{-\lambda(1-d-\alpha)})q_j)$$

Taking only the terms that depend on α, we have:

$$e^{-\lambda\alpha}(1 - e^{-\lambda(1-d-\alpha)})q_j(1 - (1 - e^{-\lambda(1-d-\alpha)})q_j)$$

To show that this value decreases with α, we show that the first derivative is negative with respect to α. We use the following form of the derivative:

$$\lambda(q_j - 1)q_j e^{2\lambda - \lambda(\alpha+2)} - \lambda q_j^2 e^{2\lambda(d+\alpha) - \lambda(\alpha+2)}$$

This value is positive owing to the fact that $q_j \in [0, 1]$ for all j. Thus, the function within the summation is decreasing in terms of α. Using the fact that the summation of decreasing functions is decreasing, the overall function is thus decreasing in terms of α.
Secondly, we examine $P[Y_i - X_i = 1]$. $Y_i - X_i = 1$ for χ_i corresponds to a mutation occurring in both a and b, but not in c. A mutation can occur in a and b if it appears on the path from $LCA(a, b, c)$ to $LCA(b, c)$, or if it appears independently in the paths from $LCA(a, b)$ to both a and b. Additionally, this mutation cannot appear on the path from $LCA(a, b, c)$ to c, and no mutations can occur on the path from r to $LCA(a, b, c)$. Thus, we have:

$$P[Y_i - X_i = 1] = \sum_j e^{-\lambda d}((1 - e^{-\lambda\alpha})q_j + e^{-\lambda\alpha}((1 - e^{-\lambda(1-d-\alpha)})q_j)^2)(1 - (1 - e^{-\lambda(1-d)})q_j)$$

Taking on the terms that depend on α, we have:

$$(1 - e^{-\lambda\alpha}) + e^{-\lambda\alpha}(1 - e^{-\lambda(1-d-\alpha)})^2 q_j$$

To show that this value is increasing with α, we show that the first derivative is positive with respect to α. We use the following form of the derivative:

$$\lambda q e^{2\lambda(d+\alpha) - \lambda(\alpha+2)} - \lambda(q - 1)e^{2\lambda - \lambda(\alpha+2)}$$

This value is positive owing to the fact that $q_j \in [0, 1]$. Thus, the function within the summation is decreasing in terms of α. Using the fact that the summation of decreasing functions is decreasing, the overall function is thus decreasing in terms of α.
General Missing Data Case: Next, we examine the general case with both stochastic and heritable missing data. In this case, we define $s(a, b)$ (analogously $s(b, c)$) as the number of characters shared by a, b that do not have dropout in either a, b or c. As we now simply condition on the fact that a, b, c must all be present, we add an additional $(1 - p_d)$ term to both $P[Y_i - X_i = 1]$ and $P[Y_i - X_i = -1]$. As this term does not depend on α, both functions depend on α as they do in the case without missing data.
Stochastic-only Missing Data Case: Finally, we examine the case with only stochastic missing data. Here we define $s(a, b)$ as the number of mutations shared by a and b in characters that did not suffer dropout in either sample. Thus, in analyzing $Y_i - X_i$ we must consider additional cases in which dropout in one cell can hide the fact that two cells inherited the same mutation.
First, we examine $P[Y_i - X_i = -1]$. $Y_i - X_i = -1$ for a character χ_i corresponds to that character acquiring the same mutation in b and c, not acquiring dropout in neither b nor c, and either observing dropout or not observing that mutation in a. For this to occur, a mutation can occur on the path from r to $LCA(a, b, c)$ while a acquires dropout,

or the same mutation can occur on the path from $LCA(a,b,c)$ to $LCA(a,b)$ and the path from $LCA(a,b,c)$ to c while a acquires dropout, or the mutation can occur on the path from $LCA(a,b)$ to b and the path from $LCA(a,b,c)$ to c while not appearing in a. The probability of this is:

$$\sum_j (1-p_d)^2(p_d(1-e^{\lambda d})+e^{\lambda d}(1-e^{-\lambda(1-d)})q_j(p_d(1-e^{-\lambda\alpha})q_j+e^{-\lambda\alpha}(1-e^{-\lambda(1-d-\alpha)})q_j(1-(1-p_d)(1-e^{-\lambda(1-d-\alpha)}))q_j$$

Taking the terms that depend on α:

$$p_d(1-e^{-\lambda\alpha})q_j + e^{-\lambda\alpha}(1-e^{-\lambda(1-d-\alpha)})q_j(1-(1-p_d)(1-e^{-\lambda(1-d-\alpha)}))q_j)$$

To show that this value decreases with α, we show that the first derivative is negative with respect to α. We use the following form of the derivative:

$$\lambda q_j(p_d-1)e^{-\lambda(\alpha+2)}(q_j e^{2m(d+\alpha)} - e^{2\alpha}(q_j-1))$$

This value is positive owing to the fact that $q_j \in [0,1]$ for all j and that $p_d \in [0,1)$. Thus, the function within the summation is decreasing in terms of α. Using the fact that the summation of decreasing functions is decreasing, the overall function is thus decreasing in terms of α.

Secondly, we examine $P[Y_i - X_i = 1]$. $Y_i - X_i = 1$ for χ_i corresponds to a mutation occurring in both a and b, not acquiring dropout in neither a nor b, and either observing dropout or not observing that mutation in c. For this to occur, a mutation can occur on the path from r to $LCA(a,b,c)$ while c acquires dropout, on the path from $LCA(a,b,c)$ to $LCA(a,b)$ while the mutation is not acquired in c or c acquires dropout, or the mutation occurs independently on the path from $LCA(a,b)$ to a and b while not appearing in c. The probability of this is:

$$\sum_j (1-p_d)^2(p_d(1-e^{-\lambda d})q_j+e^{-\lambda d}(1-(1-p_d)(1-e^{-\lambda(1-d)}))q_j((1-e^{-\lambda\alpha})q_j+e^{-\lambda\alpha}((1-e^{-\lambda(1-\alpha-d)})q_j)^2))$$

Taking on the terms that depend on α, we have:

$$(1-e^{-\lambda\alpha}) + e^{-\lambda\alpha}(1-e^{-\lambda(1-d-\alpha)})^2 q_j$$

Note that this is the same value as above in the case without missing data, and hence the function will have the same dependence on α as in that case. Hence, the function is overall decreasing with α.

Proof: By lemmas 13 and 14, $P[s(a,b) - s(a,c) \geq t]$ is an increasing function of α. Thus, the minimum value of α results in the minimum value of $P[s(a,b) - s(a,c) \geq t]$. This value occurs at $\alpha = \ell^*$, showing lemma 10.

Proof of lemma 11: First, we will show that condition i) will hold with probability $1-\zeta$ if:

$$\frac{ke^{\lambda d}\lambda(\ell^*\delta(d))^2}{32(\ell^*+(1-e^{-\lambda})q)} \geq 3\log(n)+\log 1/\zeta$$

To see this, first note that $dist(LCA(a,b,c),LCA(b,c)) \geq \ell^*$. By lemma 10, we can WLOG assume that $dist(LCA(a,b,c),LCA(b,c)) = \ell^*$ because that is the worst case, i.e. the case where $P[s(a,b)-s(b,c) \geq t]$ is minimized. Any condition sufficient for this case will be sufficient overall. Let $Y = s_c(a,b)$ and $X = s_a(b,c)$. Since $E(Y)-E(X) \geq k\lambda\ell^*\delta(d) \geq k\lambda\ell^*\delta^* = 2t$, in order to ensure that $Y - X \geq t$, it suffices to have $Y - X \geq k\lambda\ell^*\delta(d)/2$ which holds when $Y \geq E(Y) - k\lambda\ell^*\delta(d)/4$ and $X \leq E(X) + k\lambda\ell^*\delta(d)/4$. Thus, we have

$$P[Y-X<t] \leq P[Y < E(Y) - k\lambda\ell^*\delta(d)/4] + P[X > E(X) + k\lambda\ell^*\delta(d)/4]$$

To bound the probability of both events using the above versions of Hoeffding's inequality, we have:

$$\begin{aligned}
P[Y < E(Y) - k\lambda\ell^*\delta(d)/4] &= Pr[Y \leq E(Y)(1 - \frac{k\lambda\ell^*\delta(d)}{4E(Y)})] \\
&\leq exp(-\frac{k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(Y)}) \\
&\leq exp(-\frac{k^2\lambda^2\ell^{*2}\delta(d)^2}{32ke^{-\lambda d}(\lambda\ell^* + \lambda(1-e^{-\lambda})q)}) \\
P[X > E(X) + k\lambda\ell^*\delta(d)/4] &= Pr[X \geq E(X)(1 + \frac{k\lambda\ell^*\delta(d)}{4E(X)})] \\
&\leq exp(-\frac{k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(X) + 4k\lambda\ell^*\delta(d)}) \\
&\leq exp(-\frac{k^2\lambda^2\ell^{*2}\delta(d)^2}{32ke^{-\lambda d}(\lambda\ell^* + \lambda(1-e^{-\lambda})q)})
\end{aligned}$$

The last line follows from the fact that $\delta(d) \leq e^{-\lambda d}$ and $E(X) \leq ke^{-\lambda d}(1-e^{-\lambda})^2 q \leq ke^{-\lambda d}\lambda(1-e^{-\lambda})q$. Since $e^{\lambda d}\delta(d) = 1-q+qe^{-2\lambda(1-d)} \geq 1-q+qe^{-2\lambda}$, in order to ensure that both bad events have probability at most ζn^{-3}, it suffices to take

$$\frac{k\lambda(1-q+qe^{-2\lambda})\delta(d)\ell^{*2}}{32(\ell^*+(1-e^{-\lambda})q)} \geq 3\log(n)+\log 1/\zeta$$

Since $\delta^* \leq \delta(d)$, the above condition holds as long as

$$k \geq \frac{(96\log n + 32\log 1/\zeta)(\ell^*+(1-e^{-\lambda})q)}{\lambda\ell^{*2}\delta^*(1-q+qe^{-2\lambda})}$$

Applying the same argument to $s_c(a,b) - s_b(a,c)$ and combining both results gives $P[s_c(a,b) - \max(s_b(a,c), s_a(b,c)) < t] \leq 4\zeta n^{-3}$. Taking a union bound over all $\binom{n}{3} = O(n^3)$

triplets, we see that the probability of condition i) failing for any triplet is at most ζ.
To get guarantees on the second condition, note that condition i) implies that condition ii) holds for all triplets separated by an edge of length at least ℓ^*. Thus we can focus on the triplets that are not covered by condition i). Let $(a, b|c)$ be an arbitrary triplet such that $dist(LCA(a, b, c), LCA(a, b)) < \ell^*$ and again let $Y = s_c(a, b)$, $X = s_a(b, c)$ and d be the depth of $LCA(a, b, c)$ (note that we are focusing WLOG on $s_a(b, c)$ since it has the same distribution as $s_b(a, c)$).

We want to show that with high probability, $X - Y < t$. Again, it suffices to upper bound $P[Y \leq E(Y) - t/2]$ and $P[X \geq E(X) \geq t/2]$ because $E(Y) \geq E(X)$. Note that we have already bounded the second quantity. To bound the first quantity, note that the worst case scenario is that $dist(LCA(a, b, c), LCA(a, b))$ is as small as possible. Since in this lemma we make no assumption on the minimal edge length, this quantity can be arbitrarily small and in the worst case, $dist(LCA(a, b, c), LCA(a, b)) = 0$, which means Y has the same distribution as X. Note that this case technically cannot happen as it would imply that $\mathcal{T}$ has a trifurcating branch but it is possible to get arbitrarily close to this case with no restrictions on edge lengths. This gives:

$$\begin{aligned} Pr[Y \leq E(Y) - k\lambda\ell^*\delta^*/4] &\leq Pr[X \leq E(X) - k\lambda\ell^*\delta^*/4] \\ &= Pr[X \leq E(X)(1 - \frac{k\lambda\ell^*\delta^*}{4E(X)})] \\ &\leq exp(-\frac{k^2\lambda^2\ell^{*2}\delta^{*2}}{32ke^{-\lambda d}\lambda^2(1-d)^2q}) \\ &\leq exp(-\frac{k\ell^{*2}\delta^{*2}}{32q}) \end{aligned}$$

Thus, if we take:

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)q}{\ell^{*2}\delta^{*2}}, \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1-e^{-\lambda})q)}{\lambda\ell^{*2}\delta^*(1-q+qe^{-2\lambda})}\Big)$$

then we have $P[X - Y \geq t] < \zeta n^{-3}$. By symmetry, this means $P[s_a(b, c) - s_c(a, b) \geq t \bigcup s_b(a, c) - s_c(a, b) \geq t] \leq 2\zeta n^{-3}$ Since we can union bound over one bad event of probability at most $2\zeta n^{-3}$ for each of the $\binom{n}{3}$ triplets, we have that conditions i) and ii) both hold with probability at least $1 - \zeta$.

Proof of lemma 12: Given a triplet $(a, b|c)$, with LCA at depth at most d^*, we defined ϵ to be the probability that no dropout occurs in a particular character of all three cells. First we will justify the assumption that $\epsilon \geq (1 - p_d)^3$, which is to say the dropout events are positively correlated. Let p_h be the probability that heritable dropout occurs on the path from r to a. Let p_b be the probability that a heritable dropout occurs on the path from

$LCA(a,b)$ to b given that no dropout has occurred on the path from r to $LCA(a,b)$ and define p_c similarly. Note that $p_b \leq p_c \leq p_h$ since the probability a dropout occurs along a path increases with the length of the path. Let p_s be the probability that a stochastic dropout occurs at given character on a leaf given that no heritable dropout has occurred yet on that character. Then we have $\epsilon = (1-p_h)(1-p_b)(1-p_a)(1-p_s)^3 \geq ((1-p_h)(1-p_s))^3 = (1-p_d)^3$. Since at least one of the cells in the triplet needs to not incur dropout at a character in order for all three of them to have no dropout, we also have $\epsilon \leq 1-p_d$.
To prove the bounds on k we will proceed as in the proof of lemma 11 and assume that $dist(LCA(a,b,c), LCA(a,b)) = \ell^*$, noting that lemma 14 extends the result of lemma 10 to the general case with missing data. Let $Y = s_c(a,b)$ and $X = s_a(b,c)$. Thus, we have:

$$\begin{aligned} Pr[Y \leq E(Y) - k\epsilon\lambda\ell^*\delta(d)/4] &= Pr[Y \leq E(Y)(1 - \frac{k\epsilon\lambda\ell^*\delta(d)}{4E(Y)})] \\ &\leq exp(-\frac{\epsilon^2k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(Y)}) \\ Pr[X \geq E(X) + k\epsilon\lambda\ell^*\delta(d)/4] &= Pr[X \geq E(X)(1 + \frac{k\epsilon\lambda\ell^*\delta(d)}{4E(X)})] \\ &\leq exp(-\frac{\epsilon^2k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(X) + 4k\lambda\ell^*\delta(d)}) \end{aligned}$$

Since $E(Y) \leq k\epsilon e^{-\lambda d}\lambda\ell^*$ and $E(X) \leq k\epsilon e^{-\lambda d}\lambda^2 q$, we see that both probabilities are at most $n^{-3}\zeta$ when

$$\frac{k(1-p_d)^3\lambda(1-q+qe^{-2\lambda})\delta(d)\ell^{*2}}{32(\ell^* + (1-e^{-\lambda})q)} \geq \frac{k\epsilon\lambda(1-q+qe^{-2\lambda})\delta(d)\ell^{*2}}{32(\ell^* + (1-e^{-\lambda})q)} \geq 3\log(n) + \log 1/\zeta$$

If both bad events don't happen, then we have $Y - X \geq k\epsilon\lambda\ell^*\delta^*/2 \geq k(1-p_d)^3\lambda\ell^*\delta^*/2 = t$. This gives the necessary bound for condition i) to hold.
To get guarantees on the second condition, let $X = s_b(a,c)$ for any triplet a,b,c whose LCA has depth at most d^* and where c is the outgroup. we have that

$$\begin{aligned} Pr[X \leq E(X) - k(1-p_d)^3\lambda\ell^*\delta^*/4] &= Pr[X \leq E(X)(1 - \frac{k(1-p_d)^3\lambda\ell^*\delta^*}{4E(X)})] \\ &\leq exp(-\frac{k(1-p_d)^6\ell^{*2}\delta^{*2}}{32\epsilon q}) \\ &\leq exp(-\frac{k(1-p_d)^5\ell^{*2}\delta^{*2}}{32q}) \end{aligned}$$

To ensure that this probability is at most $n^{-3}\zeta$, it suffices to take

$$k \geq \frac{(96 \log n + 32 \log 1/\zeta)q}{\ell^{*2}\delta^{*2}(1-p_d)^5}$$

The rest of the argument is exactly the same as in the proof of lemma 11.

Proof of Theorem 15 To prove Theorem 15, we must first bound the effects of incorrectly inferred mutations at internal nodes. Assume that r' is an internal node generated in line 5 of the algorithm. Further assume that the tree rooted by r' is correct (i.e., it is a sub-graph of $\mathcal{T}$). The process of assigning a mutation state to r' (lines 6–12) may incorrectly include mutations that emerged independently at its two child trees. To account for this, we define $P_{i,j}(r')$ to be the probability that every leaf beneath r' has character i mutated to state j, given that character i is not mutated at r'. This probability is bounded according to the following lemma (proof shown later).

Lemma 15. *Given the constraints on λ, ℓ and q in Theorem 3, it follows that $P_{i,j}(r') \leq 2\lambda^2 c\ell q_j^2$ for any internal node $r' \in \mathcal{T}$, character i and state j.*

To guarantee that the algorithm has a low probability of making mistakes in forming cherries (line 13), we focus on a pair of "active" nodes u, w in a given iteration of the algorithm (i.e., nodes that are roots of trees in the current set S; defined in lines 2 and 14). We assume that up until this point, the algorithm did not form any incorrect cherries (i.e., all trees in S are sub-graphs of $\mathcal{T}$). It suffices to show that with high probability, if u and w do not form a cherry in $\mathcal{T}$, there must be some internal node, v on the path from $r' = LCA(u, w)$ to u such that $s(u, v) > s(u, w)$ (note that r' is LCA in $\mathcal{T}$). If this holds, then if u, w are the first incorrect pair to be merged, then there must be some descendent u' of v such that $s(u, u') \geq s(u, v) > s(u, w)$, contradicting the fact that u, w was the most similar pair. Note that u and w can be either leaves or internal nodes.
Let d be the depth of r' and let $\alpha = dist(r', v)$. Since the maximum edge length is $\sqrt{c\ell}$, we have WLOG that $dist(r', u), dist(r', w) \leq \alpha + \sqrt{c\ell}$ (if $dist(r', w)$ is greater than this quantity we can simply switch the roles of u and w and redefined α to be the distance from r' to the parent of w). Let Z be the number of mutations that occurred on the path from r' to v. This gives:

$$E(Z) = ke^{-\lambda d}(1 - e^{-\lambda\alpha})$$

Note that due to irreversibility the mutations tallied in Z will be present in all nodes under v, including u and u'. Let X be the number of character assignments that are shared between u and w (assigned by the algorithm per lines 7–11) and that are not present in r' (per the ground truth). There are two ways in which a character can be assigned state j at u (or w). Firstly, in the ground truth tree, a character can mutate to state j on the paths from r' to u (or w), with probability at most $(1 - e^{-\lambda(\alpha+\sqrt{c\ell})})q_j$. In that case, the mutation will be present

in all downstream leaves, and thus assigned by the algorithm to u (or to w). Secondly, if it did not mutate on that path, it could instead have mutated in enough places in the sub-tree rooted beneath u (or w) such that every leaf in that sub-tree has the mutation at that character. By lemma 15, we see that the probability of this occurring is at most $2\lambda^2 c\ell q_j^2$. Requiring that in both u and w we have an occurrence of at least one of these scenarios for a given state j and summing over all states, we get:

$$\begin{aligned}
E(X) &\leq ke^{-\lambda d}\sum_j((1-e^{-\lambda(\alpha+\sqrt{c\ell})})q_j + e^{-\lambda(\alpha+\sqrt{c\ell})}2\lambda^2 c\ell q_j^2)^2 \\
&\leq ke^{-\lambda d}\sum_j((1-e^{-\lambda(\alpha+\sqrt{c\ell})})q_j + 2\lambda^2 c\ell q_j^2)^2 \\
&= ke^{-\lambda d}\sum_j(1-e^{-\lambda(\alpha+\sqrt{c\ell})} + 2\lambda^2 c\ell q_j)^2 q_j^2 \\
&\leq ke^{-\lambda d}(1-e^{-\lambda(\alpha+\sqrt{c\ell})} + 2\lambda^2 c\ell \max_j q_j)^2 q
\end{aligned}$$

Let $q_{\max} = \max_j q_j$. Assume $E(Z) > E(X)$ and let $\Delta = E(Z) - E(X)$. Again by the above versions of Hoeffding's inequality, we have the following concentration inequalities:

$$\begin{aligned}
Pr[Z < E[Z] - \Delta/2] &\leq exp(-\frac{\Delta^2}{8E(Z)}) \\
&\leq exp(-\frac{\Delta}{10E(Z)}) \\
Pr[X \geq E(X) + \Delta/2] &\leq exp(-\frac{\Delta^2}{8E(X)+2\Delta}) \\
&\leq exp(-\frac{\Delta^2}{8E(X)+2(E(Z)-E(X))}) \\
&\leq exp(-\frac{\Delta}{10E(Z)})
\end{aligned}$$

Suppose $\lambda q \leq \frac{\beta}{\max(1,c)}$. Let $\gamma = 2\sqrt{c\ell}$. Then, we have:

$$\begin{aligned}
\Delta &\geq ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda(\alpha+\sqrt{c\ell})} + 2\lambda^2 c\ell q_{\max})^2 q) \\
&= ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda\alpha}e^{-\lambda\sqrt{c\ell}} + 2\lambda^2 c\ell q_{\max})^2 q) \\
&\geq ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda\alpha}(1 - \lambda\sqrt{c\ell}) + 2\lambda^2 c\ell q_{\max})^2 q) \\
&= ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda\alpha} + e^{-\lambda\alpha}\lambda\sqrt{c\ell} + 2\lambda^2 c\ell q_{\max})^2 q) \\
&= ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda\alpha} + \lambda\sqrt{c\ell}(e^{-\lambda\alpha} + \lambda\gamma q_{\max}))^2 q) \\
&= ke^{-\lambda d}(1 - e^{-\lambda\alpha} - (1 - e^{-\lambda\alpha})^2 q - 2\lambda(1 - e^{-\lambda\alpha})(e^{-\lambda\alpha} + \lambda\gamma q_{\max})\sqrt{c\ell}q - \lambda^2(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2 c\ell q) \\
&\geq ke^{-\lambda d}((1 - e^{-\lambda\alpha})(1 - (1 - e^{-\lambda\alpha})q - \beta\gamma(e^{-\lambda\alpha} + \lambda\gamma q_{\max})) - \beta\lambda(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2\ell) \\
&\geq ke^{-\lambda d}((1 - e^{-\lambda\alpha})(1 - \lambda\alpha q - \beta\gamma(e^{-\lambda\alpha} + \lambda\gamma q_{\max})) - \beta\lambda(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2\ell) \\
&\geq ke^{-\lambda d}((1 - e^{-\lambda\alpha})(1 - \beta(\alpha + \gamma e^{-\lambda\alpha} + \lambda\gamma^2 q_{\max})) - \beta\lambda(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2\ell) \\
&\geq ke^{-\lambda d}((1 - e^{-\lambda\alpha})(1 - \beta(1 + \gamma e^{-\lambda} + \lambda\gamma^2 q_{\max})) - \beta\lambda(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2\ell)
\end{aligned}$$

Where the last line follows from the fact that the maximum of the function $\alpha + \gamma e^{-\lambda\alpha}$ occurs at $\alpha = 1$.

Now, it remains to find a bound on β so that $\Delta > 0$ and $\frac{\Delta^2}{E[Z]}$ is lower bounded. Let $C = \gamma e^{-\lambda} + \lambda\gamma^2 q_{\max}$. To ensure that $\Delta > 0$, we see that the last line from the previous block needs to be > 0. Taking this inequality and rearranging terms, it suffices to show that:

$$1 > \beta(1 + C + \frac{\lambda\ell(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2}{1 - e^{-\lambda\alpha}})$$

Note that

$$\frac{\lambda\ell(e^{-\lambda\alpha} + \lambda\gamma q_{\max})^2}{1 - e^{-\lambda\alpha}} \leq \frac{\lambda\ell(e^{-\lambda\ell} + \lambda\gamma q_{\max})^2}{1 - e^{-\lambda\ell}} \approx (e^{-\lambda\ell} + \lambda\gamma q_{\max})^2$$

so our sufficient condition can be written as

$$1 > \beta(1 + C + (e^{-\lambda\ell} + \lambda\gamma q_{\max})^2)$$

We lower bound $\frac{\Delta^2}{E[Z]}$ by the following:

$$\begin{aligned}
&\frac{\Delta^2}{E[Z]} \geq \min_{\alpha\in[\ell,1]} \frac{k^2e^{-2\lambda d}((1-e^{-\lambda\alpha})(1-\beta(1+C))-\beta\lambda(e^{-\lambda\alpha}+\lambda\gamma q_{\max})^2\ell)^2}{ke^{-\lambda d}(1-e^{-\lambda\alpha})}\\
&= \frac{k^2e^{-2\lambda d}((1-e^{-\lambda\ell})(1-\beta(1+C))-\beta\lambda(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2\ell)^2}{ke^{-\lambda d}(1-e^{-\lambda\ell})}\\
&\geq ke^{-\lambda d}(\frac{(1-e^{-\lambda\ell})^2(1-\beta(1+C))^2-2(1-e^{-\lambda\ell})(1-\beta(1+C))\beta\lambda\ell(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2}{(1-e^{-\lambda\ell})}\\
&+\frac{(\beta\lambda\ell(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)^2}{(1-e^{-\lambda\ell})})\\
&\geq ke^{-\lambda d}((1-e^{-\lambda\ell})(1-\beta(1+C))^2-2(1-\beta(1+C))\beta\lambda\ell(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)\\
&\approx ke^{-\lambda d}\lambda\ell((1-\beta(1+C))^2-2(1-\beta(1+C))\beta(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)\\
&= ke^{-\lambda d}\lambda\ell(1-\beta(1+C))(1-\beta(1+C)-2\beta(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)
\end{aligned}$$

In order for this bound to be positive, we need

$$1 > \beta(1+C+2(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)$$

Note that this condition satisfies the above condition on β and thus would imply $E[Z] > E[X]$. Thus, if

$$\beta < \frac{1}{1+C+2(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2} = \frac{1}{1+2\sqrt{c\ell}e^{-\lambda}+4\lambda c\ell q_{\max}+2(e^{-\lambda\ell}+2\lambda\sqrt{c\ell}q_{\max})^2}$$

both Δ and $\frac{\Delta^2}{E[Z]}$ are strictly positive.
To bound the probability that $Z < E[Z]-\Delta/2$ and $X \geq E(X)+\Delta/2)$ is at most $n^{-2}\zeta$, we need:

$$exp(-\frac{\Delta^2}{10E[Z]}) \leq n^{-2}\zeta$$

This is satisfied if k satisfies the following:

$$k \geq \frac{20\log n+10\log(1/\zeta)}{\lambda e^{-\lambda}\ell(1-\beta(1+C))(1-\beta(1+C)-2\beta(e^{-\lambda\ell}+\lambda\gamma q_{\max})^2)}$$

In other words, for any pair of vertices u, w that are not children of the same node, there will be a vertex u' that such that $LCA(u,u')$ is a descendent of $LCA(u,w)$ and $P[s(u,u') \leq s(u,w)] \leq 2n^{-2}\zeta$. If $s(u,u') > s(u,w)$, then (u,w) cannot be the first pair of incorrectly joined vertices. Taking a union bound over at most $n^2/2$ pairs of vertices, we see that with with probability at least $1-\zeta$, there is no first pair of incorrectly joined vertices, which means the algorithm is correct.

Proof of lemma 15: For this proof, we use the following results:

Lemma 16. *Let ρ be the maximum edge length in $\mathcal{T}$. For any character, state pair (i, j), if there exists a number $p > 0$ which satisfies $((1 - e^{-\lambda\rho})q_j + p)^2 \leq p$, then p is an upper bound on $P_{i,j}(v)$ for any node $v \in \mathcal{T}$.*

Proof: We will proceed by induction on $\mathcal{T}$. Suppose v is a leaf. Then $P_{i,j}(v) = 0$, since if the i^{th} character does not mutate, it cannot take on state j. Now let v be an arbitrary non-leaf vertex, with children u and w. By our inductive hypothesis, $P_{i,j}(w) \leq p$ and $P_{i,j}(u) \leq p$. Since the length of the edge from v to either of it's children is at most ρ, the probability that the character mutates to state j on either edge is at most $(1 - e^{-\lambda\rho})q_j$. Thus, if we condition on the fact that χ_i is not mutated on v, we have:

$$\begin{aligned} P_{i,j}(v) &\leq ((1 - e^{-\lambda\rho})q_j + e^{-\lambda\rho}P_{i,j}(u))((1 - e^{-\lambda\rho})q_j + e^{-\lambda\rho}P_{i,j}(w)) \\ &\leq ((1 - e^{-\lambda\rho})q_j + P_{i,j}(u))((1 - e^{-\lambda\rho})q_j + P_{i,j}(w)) \\ &\leq ((1 - e^{-\lambda\rho})q_j + p)^2 \\ &\leq p \end{aligned}$$

Where the last inequality follows from our assumption on p. Given the above lemma, we can simply solve for p to find an upper bound on all $P_{i,j}(v)$ in $\mathcal{T}$.

Lemma 17. *Suppose the maximum edge length satisfies:*

$$\rho \leq -\frac{1}{\lambda}\ln[(1 - \frac{3}{16\max_j(q_j)})]$$

Then we have $P_{i,j}(v) \leq 2\lambda^2\rho^2q_j^2$ for any node $v \in \mathcal{T}$

Proof: Let $y = (1 - e^{-\lambda\rho})q_j$. Note that by our above assumption, $y \leq 3/16$. By our above assumption, we have:

$$\begin{aligned} \rho \leq -\frac{1}{\lambda}\ln[(1 - \frac{3}{16\max_j(q_j)})] \\ (1 - e^{-\lambda\rho})\max_j(q_j) \leq \frac{3}{16} \\ (1 - e^{-\lambda\rho})q_j \leq \frac{3}{16} \\ y \leq \frac{3}{16} \end{aligned}$$

Next, by lemma 16, we know that if there is a $p > 0$ such that $(y + p)^2 \leq p$, then such a p would be an upper bound on all $P_{i,j}(v)$. We can find such a p by setting the inequality to an equality and finding the smallest root of the resulting polynomial.

$$\begin{aligned} y^2 + 2yp + p^2 = p \\ y^2 + (2y - 1)p + p^2 = 0 \end{aligned}$$

Our initial assumption of ρ guarantees that $4y < 1$, which means the smallest root of the polynomial above can be given as

$$\begin{aligned} p &= \frac{1}{2}(1 - 2y - \sqrt{((2y-1)^2 - 4y^2)}) \\ &= \frac{1}{2}(1 - 2y - \sqrt{(4y^2 - 4y + 1 - 4y^2)}) \\ &= \frac{1}{2}(\sqrt{1 - 4y + 4y^2} - \sqrt{(1-4y)}) \\ &= \frac{1}{2}\frac{4y^2}{\sqrt{1 - 4y + 4y^2} + \sqrt{(1-4y)}} \end{aligned}$$

Where the last line follows by multiplying the numerator and denominator by $\sqrt{1 - 4y + 4y^2}+\sqrt{(1-4y}$. To upper bound p, we have

$$\begin{aligned} p &\leq \frac{1}{2}\frac{4y^2}{2\sqrt{1-4y}} \\ &= \frac{y^2}{\sqrt{1-4y}} \\ &\leq 2y^2 \end{aligned}$$

Where the last inequality follows from the fact $y \leq 3/16$, which means the denominator is at least $1/2$. Finally, we have $2y^2 = 2(1 - e^{-\lambda\rho})^2 q_j^2 \leq 2\lambda^2\rho^2 q_j^2$.
Now simply take $\rho = \sqrt{c\ell^*}$ and apply lemma 17 to show lemma 15.

Additional Analyses

Alternative Strategy for Regime with only Stochastic Missing Data Now we assume that missing data only occurs due to technical difficulties in reading out mutation sequences in cells, i.e. that only stochastic missing data occurs. Specifically, for any given cell and any given Cas9 recording site, we assume that the sequence of this site is missing from our data with probability p_s and that this omission occurs independently from all other cells or mutation sites. With this definition, we consider a slightly modified oracle by defining $s(a, b)$ as the number of mutations shared by a and b in characters that did not suffer dropout in either sample. Note that this definition of $s(a, b)$ is independent of a third cell c unlike in the general case. Accordingly we revise conditions i) and ii) by setting the decision threshold t to be $k(1 - p_s)^2\lambda\ell\delta^*$. These modified definitions lead to a more relaxed dependency between the mutation rate and the extent of missing data:

Lemma 18. *In the presence of stochastic missing data at a rate of p_s, condition i) holds with probability at least $1-\zeta$ if we we have the following guarantees on the parameters q, λ, ℓ^*, d^*, and k:*

$$k \geq \frac{(96\log n + 32\log 1/\zeta)(\ell^* + (1 - e^{-\lambda})q + e^{\lambda d^*}d^*)}{(1-p_s)^2\lambda{\ell^*}^2\delta^*(1 - q + qe^{-2\lambda})}$$

Both conditions i and ii hold with probability at least $1-\zeta$ if we have:

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)(\ell^* + \lambda q + e^{\lambda d^*}d^*)}{(1-p_s)^2\lambda\ell^{*2}\delta^*(1-q+qe^{-2\lambda})}, \frac{(96\log n + 32\log 1/\zeta)(d^* + (1-e^{-\lambda})q)}{(1-p_s)^2\lambda\ell^{*2}\delta^{*2}}\Big)$$

An empirical demonstration and validation of the tightness of lemma 18 using simulations is provided in supplemental Figure 3.38.

Proof: For a triplet $(a,b|c)$ in the case without dropout, any mutation that occurred on the path from the root to $LCA(a,b,c)$ is inherited by each member of the triplet, and thus $s(a,b) - s(b,c) = s_c(a,b) - s_a(b,c)$. But in the case of missing data, this is no longer true as mutations that occurred before $LCA(a,b,c)$ may be obscured by dropout and therefore not present in the character information of a, b, or c. We must now account for these early mutations in our calculations.

Let p_s be the stochastic missing data rate and let $s(a,b)$ be the number of mutations shared between a and b, ignoring characters that have dropout in either a or b. The number of mutations shared by a,b after their divergence is now Binomial(k,p) where p is

$$\geq (1-p_s)^2(1-e^{-\lambda d} + e^{-\lambda d}((1-e^{-\lambda\ell^*}) + e^{-\lambda\ell^*}(1-e^{-\lambda(1-\ell^*-d)})^2 q)$$

Here $(1-p_s)^2$ is the probability that this character does not acquire dropout in neither a nor b, $1-e^{-\lambda d}$ is the probability that a given mutation occurred before d, and of the remaining terms the left term is the probability the mutation occurred on the path from $LCA(a,b,c)$ to $LCA(a,b)$, and the right term is the probability a given mutation is shared by a,b due to convergent evolution.
Thus:

$$E[s(a,b)] \geq k(1-p_s)^2(1-e^{-\lambda d} + e^{-\lambda d}((1-e^{-\lambda\ell^*}) + e^{-\lambda\ell^*}(1-e^{-\lambda(1-\ell^*-d)})^2 q)$$

Similarly:

$$E[s(b,c)] \leq k(1-p_s)^2(1-e^{-\lambda d} + e^{-\lambda d}(1-e^{-\lambda(1-d)})^2 q)$$

Thus, we have that:

$$\begin{aligned}
E[s(a,b)-s(b,c)] &\geq (1-p_s)^2 k(1-e^{-\lambda d} + e^{-\lambda d}((1-e^{-\lambda\ell^*}) + e^{-\lambda\ell^*}(1-e^{-\lambda(1-\ell^*-d)})^2 q - (1-e^{-\lambda d} + e^{-\lambda d}(1-e^{-\lambda(1-d)})^2 q)) \\
&= (1\text{-}p_s)^2 k e^{-\lambda d}(1-e^{-\lambda\ell^*} + q(e^{-\lambda\ell^*} - 1 + e^{-2\lambda(1-d)+\lambda\ell^*} - e^{-2\lambda(1-d)})) \\
&= (1\text{-}p_s)^2 k e^{-\lambda d}((1-e^{-\lambda\ell^*})(1-q) + qe^{-2\lambda(1-d)}(e^{\lambda\ell^*}-1)) \\
&\geq (1-p_s)^2 k e^{-\lambda d}((1-e^{-\lambda\ell^*})(1-q) + qe^{-2\lambda(1-d)}\lambda\ell^*) \\
&\approx (1-p_s)^2 k(e^{-\lambda d}\lambda\ell^*(1-q) + qe^{-\lambda(2-d)}\lambda\ell^*) \\
&= (1\text{-}p_s)^2 k\lambda\ell^*(e^{-\lambda d}(1-q) + qe^{-\lambda(2-d)})
\end{aligned}$$

Again taking $\delta(d) = e^{-\lambda d}(1-q) + qe^{-\lambda(2-d)}$. We then have that for any triplet $(a, b|c)$, where $depth(LCA(a,b,c)) = d$ and $dist(LCA(a,b,c),\ LCA(a,b)) \geq \ell^*$

$$E[s(a,b) - s(b,c)] \geq (1-p_s)^2 k\lambda\ell^*\delta(d)$$

First, we will show that condition i) will hold with probability $1-\zeta$ if:

$$\frac{(1-p_s)^2 k e^{\lambda d}\lambda(\ell^*\delta(d))^2}{32(\ell^* + (1-e^{-\lambda})q + e^{\lambda d}d)} \geq 3\log(n) + \log 1/\zeta$$

To see this, let $(a, b|c)$ be any triplet at depth at most d^* and the distance between their LCAs be at least ℓ^*. By lemma 10, we can WLOG assume that $dist(LCA(a,b,c), LCA(b,c)) = \ell^*$ because that is the worst case, i.e. the case where $P[s(a,b) - s(b,c) \geq t]$ is minimized. Note that lemma 14 extends the result of lemma 10 to the case with only stochastic missing data. Any condition sufficient for this case will be sufficient overall. Let $Y = s(a,b)$ and $X = s(b,c)$. Since $E(Y) - E(X) \geq (1-p_s)^2 k\lambda\ell^*\delta^* = 2t$, in order to ensure that $Y - X \geq t$, it suffices to have $Y > E(Y) = t/2$ and $X < E(X) = t/2$. To show that both occur with high probability, we have:

$$\begin{aligned}
Pr[Y \leq E(Y) - (1-p_s)^2 k\lambda\ell^*\delta(d)/4] &= Pr[Y \leq E(Y)(1 - \frac{(1-p_s)^2 k\lambda\ell^*\delta(d)}{4E(Y)})] \\
&\leq exp(-\frac{(1-p_s)^4 k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(Y)}) \\
&\leq exp(-\frac{(1-p_s)^4 k^2\lambda^2\ell^{*2}\delta(d)^2}{32(1-p_s)^2 k\lambda(e^{-\lambda d}\ell^* + e^{-\lambda d}(1-e^{-\lambda})q + d)}) \\
Pr[X \geq E(X) - (1-p_s)^2 k\lambda\ell^*\delta(d)/4] &= Pr[X \geq E(X)(1 - \frac{(1-p_s)^2 k\lambda\ell^*\delta(d)}{4E(X)})] \\
&\leq exp(-\frac{(1-p_s)^4 k^2\lambda^2\ell^{*2}\delta(d)^2}{32E(X) + 4(1-p_s)^2 k\lambda\ell^*\delta(d)}) \\
&\leq exp(-\frac{(1-p_s)^4 k^2\lambda^2\ell^{*2}\delta(d)^2}{32(1-p_s)^2 k\lambda(e^{-\lambda d}\ell^* + e^{-\lambda d}(1-e^{-\lambda})q + d)})
\end{aligned}$$

The last line follows from the fact that $\delta(d) \leq e^{-\lambda d}$ and $E(X) \leq (1-p_s)^2 k e^{-\lambda d}(1-e^{-\lambda})^2 q + \lambda d \leq (1-p_s)^2 k e^{-\lambda d}(1-e^{-\lambda})\lambda q + \lambda d$. Since $e^{\lambda d}\delta(d) = 1 - q + qe^{-2\lambda(1-d)} \geq 1 - q + qe^{-2\lambda}$ and any $d < d^*$, in order to ensure that both bad events have probability at most ζn^{-3}, it suffices to take

$$\frac{(1-p_s)^2 k\lambda(1-q+qe^{-2\lambda})\delta(d)\ell^{*2}}{32(\ell^* + (1-e^{-\lambda})q + e^{\lambda d^*}d^*)} \geq 3\log(n) + \log 1/\zeta$$

Applying the same argument to $s(a,b) - s(a,c)$ and combining both results gives $P[s(a,b) - \max(s(a,c), s(b,c)) < t] \leq 4\zeta n^{-3}$. Taking a union bound over all $\binom{n}{3} = O(n^3)$ triplets, we

see that the probability of condition i) failing for any triplet is at most ζ.
To get guarantees on the second condition, note that condition i) implies that condition ii) holds for all triplets separated by an edge of length at least ℓ^*. Thus we can focus on the triplets that are not covered by condition i). Let $(a, b|c)$ be an arbitrary triplet such that $dist(LCA(a, b, c), LCA(a, b)) < \ell^*$ and again let $Y = s(a, b)$, $X = s(b, c)$ and d be the depth of $LCA(a, b, c)$ (note that we are focusing WLOG on $s(b, c)$ since it has the same distribution as $s(a, c)$). We want to show that with high probability, $X - Y < t$. Again, it suffices to upper bound $P[Y \leq E(Y) - t/2]$ and $P[X \geq E(X) \geq t/2]$ because $E(Y) \geq E(X)$. Note that we have already bounded the second quantity. To bound the first quantity, note that the worst case scenario is that $dist(LCA(a, b, c), LCA(a, b))$ is as small as possible, but since this quantity can be arbitrarily small, we can assume that in the worst case, $dist(LCA(a, b, c), LCA(a, b)) = 0$, which means Y has the same distribution as X. Note that this case technically cannot happen as it would imply that $\mathcal{T}$ has a trifurcating branch but it is possible to get arbitrarily close to this case with no restrictions on edge lengths. This gives:

$$\begin{aligned} Pr[X \leq E(X) - (1-p_s)^2 k\lambda\ell^*\delta^*/4] &= Pr[X \leq E(X)(1 - \frac{(1-p_s)^2 k\lambda\ell^*\delta^*}{4E(X)})] \\ &\leq exp(-\frac{(1-p_s)^4 k^2\lambda^2\ell^{*2}\delta^{*2}}{32(1-p_s)^2 k\lambda(d + e^{-\lambda d}(1-e^{-\lambda})(1-d)^2 q)}) \\ &\leq exp(-\frac{(1-p_s)^2 k\lambda\ell^{*2}\delta^{*2}}{32(d + (1-e^{-\lambda})q)}) \end{aligned}$$

Thus, if we take:

$$k \geq \max\Big(\frac{(96\log n + 32\log 1/\zeta)(\ell^* + \lambda q + e^{\lambda d^*} d^*)}{(1-p_s)^2\lambda\ell^{*2}\delta^*(1 - q + qe^{-2\lambda})}, \frac{(96\log n + 32\log 1/\zeta)(d^* + (1-e^{-\lambda})q)}{(1-p_s)^2\lambda\ell^{*2}\delta^{*2}}\Big)$$

then we have $P[X - Y \geq t] < \zeta n^{-3}$. By symmetry, this means $P[s(b, c) - s(a, b) \geq t \bigcup s(a, c) - s(a, b) \geq t] \leq 2\zeta n^{-3}$ Since we can union bound over one bad event of probability at most $2\zeta n^{-3}$ for each of the $\binom{n}{3}$ triplets, we have that conditions i) and ii) both hold with probability at least $1 - \zeta$.

Alternative Analysis of the Bottom-up Algorithm

Theorem 16. *The constraint on λ and q in Theorem 15 can be replaced with*

$$\lambda < \min(\frac{1 - \lambda\ell/2}{2q\gamma_2}, -\ln(1 + 2\gamma_1 + \frac{\gamma_1^2}{1 - e^{-\lambda}} - \frac{1}{2q}))$$

Where $\gamma_1 = e^{-\lambda}\sqrt{c\ell} + 2\lambda^2 c\ell q_{\max}$ and $\gamma_2 = (\sqrt{\ell} + \sqrt{c} + 2\lambda c\sqrt{\ell} q_{\max})^2$. In that case, the Bottom-Up Algorithm returns the correct tree with probability at least $1 - \zeta$ if

$$k \geq \frac{e^{\lambda}(20\log n + 10\log(1/\zeta))}{\min(\lambda\ell(1 - \lambda\ell/2 - 2\lambda q\gamma_2),\ (1 - e^{-\lambda})(1 - 2(1 - e^{-\lambda} + 2\gamma_1 + \frac{\gamma_1^2}{1 - e^{-\lambda}})q))}$$

Note this means that as $\ell \to 0$, our constraint on λ and q becomes $\lambda q < \frac{1}{2c}$ and our bound on k approaches

$$\frac{e^{\lambda}(20\log n + 10\log(1/\zeta))}{\lambda\ell(1-2c\lambda q)} = O(\frac{\log n}{\ell})$$

Proof of Theorem 16: To upper bound the probabilities of the bad events, it suffices to ensure that $E(Z) > E(X)$ and bound the quantity $\frac{(E(Z)-E(X))^2}{E(Z)} = \frac{\Delta^2}{E(Z)}$. Since this quantity depends on α, we will first derive a lower bound on this quantity and determine where the minimum occurs for $\alpha \in [\ell, 1]$ as follows:

$$\begin{aligned}\frac{\Delta^2}{E[Z]} &\geq E[Z] - 2E[X] \\ &\geq ke^{-\lambda d}(1 - e^{-\lambda\alpha} - 2(1 - e^{-\lambda(\alpha+\sqrt{c\ell})} + 2\lambda^2 c\ell q_{\max})^2 q)\end{aligned}$$

Now let $f(\alpha) = 1 - e^{-\lambda\alpha} - 2(1 - e^{-\lambda(\alpha+\sqrt{c\ell})} + 2\lambda^2 c\ell q_{\max})^2 q)$. Next we will show that for any interval $[a, b]$, the minimum of f on $[a, b]$ occurs at either a or b. Let $y(\alpha) = e^{-\lambda\alpha}$, and let $g(y) = 1 - y - 2(1 - ye^{-\lambda\sqrt{(c\ell)}} + 2\lambda^2 c\ell q_{\max})^2 q$. Then $f = g \circ y$. Note that g' is linear with negative slope, as

$$g'(y) = -1 + 4(1 - ye^{-\lambda\sqrt{c\ell}} + 2\lambda^2 c\ell q_{\max})qe^{-\lambda\sqrt{c\ell}}$$

Since $f'(x) = -g'(e^{-\lambda x})e^{-\lambda x}$, $f'(x) = 0$ only when $g'(e^{-\lambda x}) = 0$. Since $e^{-\lambda x}$ is an increasing function, there can be at most one point where f' is 0. On the other hand, one can verify that $\lim_{x\to-\infty} f'(x) = \infty$, which means that if there is any x where $f'(x) = 0$, then f' is positive on (∞, x) and negative on (x, ∞), which means x must be a local maximum. Thus, any minimum of f on an interval $[a, b]$ can only occur on the boundaries.

$\alpha = 1$ **case:** In the case where $\alpha = 1$, our lower bound can be written as

$$\begin{aligned}f(\alpha) &= (1 - e^{-\lambda} - 2(1 - e^{-\lambda}e^{-\lambda\sqrt{c\ell}} + 2\lambda^2 c\ell q_{\max})^2 q) \\ &\geq (1 - e^{-\lambda} - 2(1 - e^{-\lambda}(1 - \sqrt{c\ell}) + 2\lambda^2 c\ell q_{\max})^2 q) \\ &= (1 - e^{-\lambda} - 2(1 - e^{-\lambda} + e^{-\lambda}\sqrt{c\ell} + 2\lambda^2 c\ell q_{\max})^2 q)\end{aligned}$$

To make the notation easier to follow, let $\gamma_1 = e^{-\lambda}\sqrt{c\ell} + 2\lambda^2 c\ell q_{\max}$. Then we have

$$\begin{aligned}\frac{\Delta^2}{E[Z]} &\geq (1 - e^{-\lambda} - 2(1 - e^{-\lambda} + \gamma_1)^2 q) \\ &= (1 - e^{-\lambda} - 2((1 - e^{-\lambda})^2 + 2(1 - e^{-\lambda})\gamma_1 + \gamma_1^2)q) \\ &= (1 - e^{-\lambda})(1 - 2(1 - e^{-\lambda} + 2\gamma_1 + \frac{\gamma_1^2}{1 - e^{-\lambda}})q)\end{aligned}$$

Thus, in order for the bound to be non-trivial, we need $2(1-e^{-\lambda}+2\gamma_1+\frac{\gamma_1^2}{1-e^{-\lambda}})q<1$, which means

$$e^{-\lambda} > 1+2\gamma_1+\frac{\gamma_1^2}{1-e^{-\lambda}}-\frac{1}{2q}$$
$$\lambda < -\ln(1+2\gamma_1+\frac{\gamma_1^2}{1-e^{-\lambda}}-\frac{1}{2q})$$

Note that as $\ell \to 0$, the RHS approaches $\lambda < \ln(1-\frac{1}{2q})$ which is at least $\frac{1}{2q}$, and in general, when λ satisfies the constraint above, $f(1)$ is lower bounded by a constant independent of ℓ. Also, note that the bound is bound is trival if the term inside the ln is negative.

$\alpha = \ell$ **case:** In this case, our lower bound becomes as follows:

$$\begin{aligned} f(\alpha) &= (1-e^{-\lambda\ell}-2(1-e^{-\lambda(\ell+\sqrt{c\ell})}+2\lambda^2 c\ell q_{\max})^2 q) \\ &\geq (\lambda\ell-\frac{\lambda^2\ell^2}{2}-2\lambda^2(\ell+\sqrt{c\ell}+2\lambda c\ell q_{\max})^2 q) \\ &= \lambda\ell(1-\frac{\lambda\ell}{2}-2\lambda q(\sqrt{\ell}+\sqrt{c}+2\lambda c\sqrt{\ell}q_{\max})^2) \end{aligned}$$

Now let $\gamma_2 = (\sqrt{\ell}+\sqrt{c}+2\lambda c\sqrt{\ell}q_{\max})^2$. Then in order for the bound to be non-trivial, we need $\frac{\lambda\ell}{2}+2\lambda q\gamma_2 < 1$, which means

$$\lambda < \frac{1-\lambda\ell/2}{2q(\sqrt{\ell}+\sqrt{c}+2\lambda c\sqrt{\ell}q_{\max})^2} = \frac{1-\lambda\ell/2}{2q\gamma_2}$$

Note that as $\ell \to 0$, the RHS becomes $\frac{1}{2qc}$. Since $\frac{\Delta^2}{E[Z]} \geq ke^{-\lambda d}f(\alpha)$, when the constraints:

$$\lambda < \min(\frac{1-\lambda\ell/2}{2q\gamma_2}, -\ln(1+2\gamma_1+\frac{\gamma_1^2}{1-e^{-\lambda}}-\frac{1}{2q}))$$

are satisfied, and if we take

$$k \geq \frac{e^{\lambda}(20\log n+10\log(1/\zeta))}{\min(\lambda\ell(1-\lambda\ell/2-2\lambda q\gamma_2),\ (1-e^{-\lambda})(1-2(1-e^{-\lambda}+2\gamma_1+\frac{\gamma_1^2}{1-e^{-\lambda}})q))}$$

the probability that either of the above events occur is at most $n^{-2}\zeta$. In other words, for any pair of vertices u, w that are not children of the same node, there will be a vertex u' that such that $LCA(u,u')$ is a descendent of $LCA(u,w)$ and $P[s(u,u') \leq s(u,w)] \leq 2n^{-2}\zeta$. If $s(u,u') > s(u,w)$, then (u,w) cannot be the first pair of incorrectly joined vertices. Taking a union bound over at most $n^2/2$ pairs of vertices, we see that with with probability at least $1-\zeta$, there is no first pair of incorrectly joined vertices, which means the algorithm is correct.

Since the bound on $f(\ell)$ depends on ℓ, it is general much smaller than the bound on $f(1)$, which is lower bounded by a constant. Thus, asymptotically, the bound on k is

$$\frac{e^{\lambda}(20\log n + 10\log(1/\zeta))}{\lambda\ell(1-2c\lambda q)} = O(\frac{\log n}{\ell})$$

Additional Simulations

Visualization of the $(\ell^*, d^*)-$Oracle: The $(\ell^*, d^*)-$Oracle presented above attempts to determine the outgroup of a triplet using the difference in the number of shared mutations between its members. In Figure 3.36 we visualize how well the decision rule holds for correctly and incorrectly resolved triplets. We plot $s(a,b) - \max(s(a,c), s(b,c))$ against $dist(LCA(a,b), LCA(a,b,c))$ for triplets (a,b,c) on simulated trees. The blue points show the difference between the mutations shared by the ingroup versus the mutations shared by the outgroup, and the orange points show the difference between the mutations shared by the outgroup and the ingroup. We see that 10% of triplets are such that $s(a,b) - \max(s(a,c), s(b,c)) \leq t$ with (a,b) as the ingroup and thus violate condition i), and that 2.3% are such that $s(a,b) - \max(s(a,c), s(b,c)) \leq t$ with (a,b) as the outgroup. Note that as $s(a,b) - \max(s(a,c), s(b,c)) \leq s(a,b) - s(a,c)$, requiring that $s(a,b) - \max(s(a,c), s(b,c)) < t$ is a slightly weaker condition than condition ii) and thus at most 2.3% of triplets violate it. For the set of parameters used here, t is chosen such that the probability that a triplet is indeterminable is low and the probability that the outgroup is given incorrectly is low, showing that the oracle is relatively accurate on this regime for the low number of characters ($k = 10$). For for exact reconstruction of a tree though, we require that all triplets on that tree be separated by the threshold t.
As $dist(LCA(a,b), LCA(a,b,c))$ increases, this difference in the number of shared mutations between the ingroup and the non-ingroup pairs grows, making the triplets more separable. This gives us the V-shape in the figure. As the distance increases, the number of triplets that cross the threshold and thus violate either condition decreases, and after a certain point no triplets cross the threshold. This illustrates that in the result in Theorem 14 that, as we are only interested in triplets where $dist(LCA(a,b), LCA(a,b,c)) > \ell^*$ for larger values of ℓ^*, them the bound on the number of characters k needed to separate these triplets decreases.

Dependence on n: To compare the asymptotic dependence of k on n in the theoretical bounds with the dependence of the necessary k in the empirical case, we simulate varying trees of size. In Figure 3.37 we plot the values of k in simulation against the bounds for both algorithms, in a regime where $\ell = 0.5$, $q = 0.05$. From this figure it can be seen that the bounds are tight (within a constant factor) against the empirical values and share the same shape in the dependence of k on n. These results offer empirical validation of the asymptotic dependence of k in the bounds for both algorithms.

Missing Data: Here we present simulations for the minimum k to give 0.9 probability of exact reconstruction for the case of stochastic missing data only (lemma 18). The simulations are performed with uniform edge lengths and use $p_s = 0.1$ proportion of stochastic missing data. Visualizing the bounds for lemma 18 show that indeed higher values of k are necessary to overcome the lost information (Figure 3.38A), consistent with the additional $(1-p_d)^2$ term in the denominator and gap term of $e^{\lambda d^*} d^*$ when compared to the bounds of Theorem 13. Notably, the reconstruction now becomes increasingly intractable for high values of λ, due to

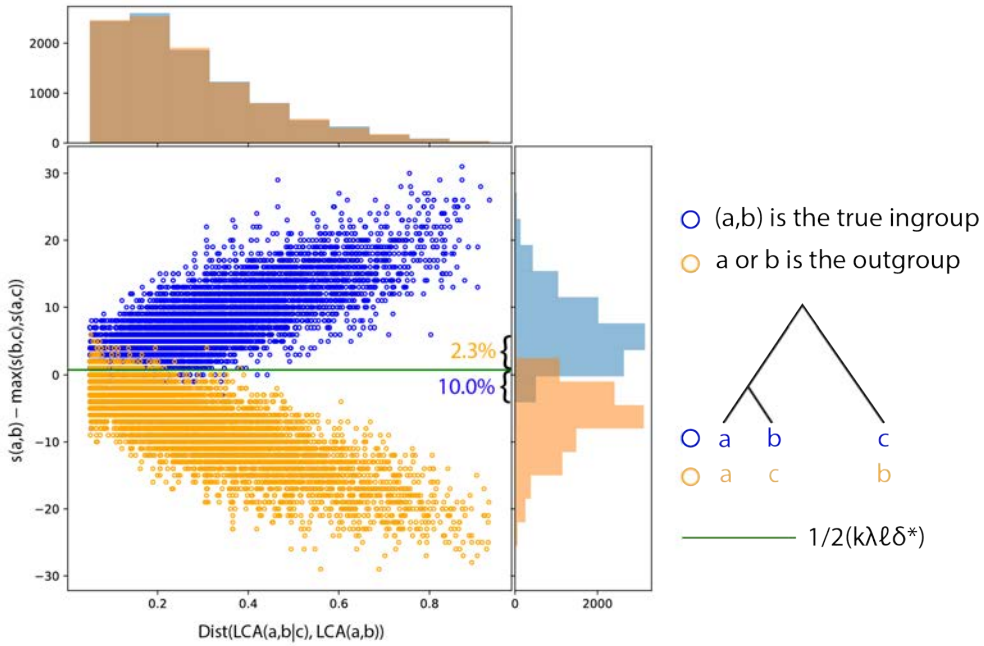


Figure 3.36: **Visualizing the $(\ell^*, d^*)-$Oracle Decision Rule in Simulation**. We sample 100 triplets uniformly for each of 100 trees simulated under the Asynchronous simulation framework described in supplemental with $k = 10$, $\lambda = 0.5$, $q = 0.05$ and $n \approx 256$. For each triplet, we plot $s(a, b) - \max(s(a, c), s(b, c))$ for two cases: when (a, b) notates the true ingroup of the triplet (blue) and 2) when (a, b) notates one member of the ingroup and the outgroup (orange). The histograms show the density of each case for each triplet along the axes.

the gap term being exponential in λ. In simulation (Figure 3.38B) we see that the necessary k is higher across the board, especially for values with high λ, validating the theoretical trends.

Surprisingly, unlike in lemma 11, the bound on k in lemma 18 has no asymptotic dependence on q. Taking q to be arbitrarily small (or even $q < \ell/\lambda$) causes the bound in lemma 11 to become $O(\frac{\log n}{\ell})$, yet an arbitrarily small q causes the bound in lemma 18 to remain in $O(\frac{\log n}{\ell^2})$. Examining the difference in the bounds between lemma 18 and lemma 11 (Figure 3.38C (Top)), we see that the difference grows larger in regions where $q < \ell/\lambda$, indicating that the bound changes asymptotically in these regions. This same pattern is reflected in the empirical results, validating this change in dependence on q (Figure 3.38C (Bottom)).

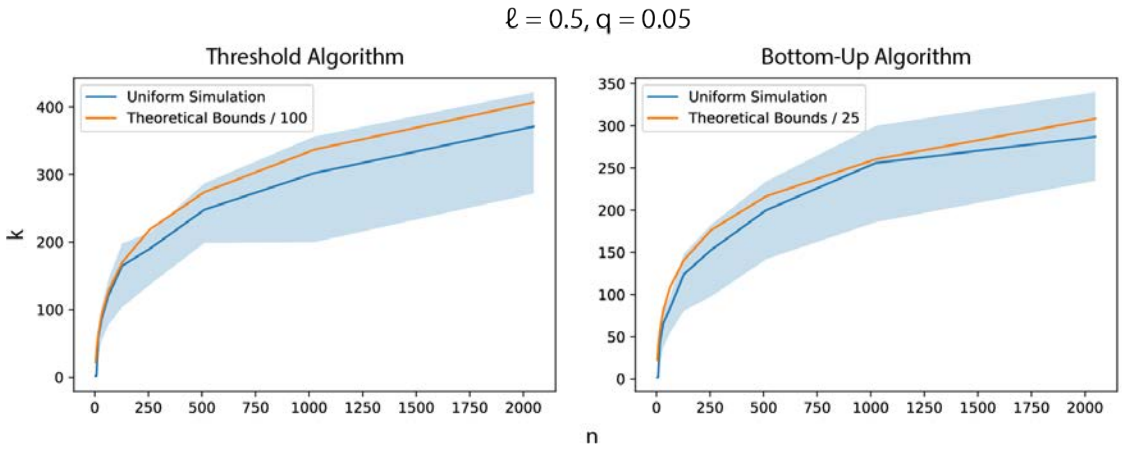


Figure 3.37: **Comparing the Dependence of k on n in Theory and Simulation**. The minimum necessary k given n of the Threshold Algorithm (left) and the Bottom-Up Algorithm (right) with the theoretical bounds for each case (90% of full reconstruction). Simulations performed with the uniform edge length regime for trees of size $2^2, 2^3, ..., 2^{11}$ leaves. For each value of n, edge lengths were re-scaled to be $\frac{1}{log_2(n)+1}$ to maintain uniform edge lengths. The bounds are rescaled by a constant factor, 100 for the Threshold Algorithm and 25 for the Bottom-Up Algorithm. Point-wise 95% confidence intervals are generated from the regression coefficients using the delta method, see supplemental.

Triplets Correct: Previous benchmarking works do not use exact reconstruction as a metric for the accuracy of phylogenetic reconstructions. A common, more relaxed metric is the triplets correct metric (or the closely related triples distance), which measures the proportion of sampled leaf triplets that are correctly (incorrectly in the case of triplets distance) inferred by the reconstructed tree [98, 39, 69]. We present the minimum k necessary in simulation for high probability of a high ($\geq$ 95%) triplets correct score on uniformly sampled triplets, showing the necessary k when exact reconstruction is not required (Figure 3.39). We see that the empirical necessary k decreases substantially overall compared to the case of exact reconstruction for both algorithms, showing that these algorithms can perform well in practice with a low number of characters according to traditional standards of accuracy.

Regarding the Threshold Algorithm, the reduction in the necessary k is potentially due to the fact that if triplets are sampled uniformly, most of them will have an LCA close to the root. We saw from the partial reconstruction results that the necessary k decreases across the board as d^* (the depth up to which triplets must be correct) decreases. We see also that large values of λ coincide with lower relative k in the triplets case than in the case of full reconstruction, just as in the case of $d^* << 1$. Thus, we can treat the case of uniformly sampling triplets as similar to setting a low d^*. We see that both of these effects, the lower overall necessary k and the lower k for high values of λ, also hold true of the Bottom-Up Algorithm in the case of uniformly sampling triplets. This indicates that perhaps reconstruction of triplets that diverge at the top of the tree is easier and less affected by mutation saturation in the case of the Bottom-Up Algorithm as well.

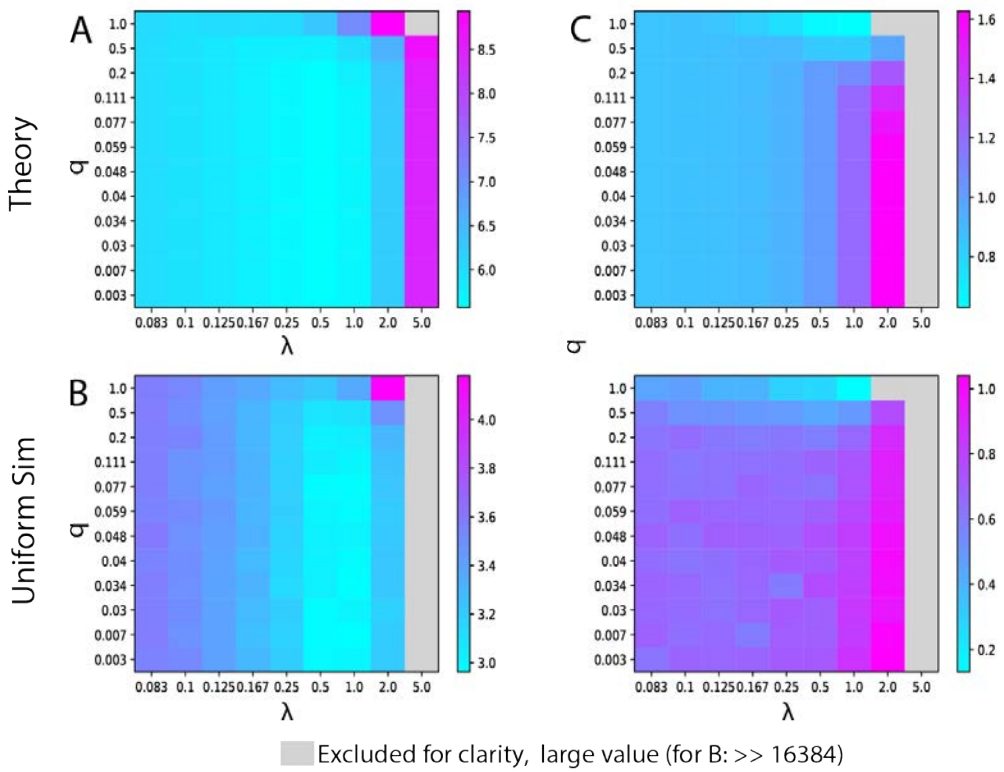


Figure 3.38: **Comparing the Threshold Algorithm in the Case of Stochastic Missing Data in Theory and Simulation**. The results of the Threshold Algorithm using the missing data strategy outlined in lemma 18 are shown. Simulated trees with 256 leaves, $n = 256$. Entries are log_{10} scaled. (A) Theoretical lower bound on k required for 0.9 probability of perfect tree reconstruction for varying values of q and λ in the case of missing data, with $\ell = 1/9$, $d^* = 1$, and $p_s = 0.1$. (B) Minimum k required for 0.9 probability of perfect reconstruction in simulation with a cell division topology with uniform branch lengths, $\ell = 1/9$, $d^* = 1$, and $p_s = 0.1$. (C) Scalar difference in the values of k in the case with and without missing data. Top: Difference in theoretical bounds for k for 0.9 probability of perfect reconstruction, $\ell = 1/9$, $d^* = 1$, and $p_s = 0.1$. Bottom: Difference in minimum k required for 0.9 probability of perfect reconstruction in simulation with a cell division topology with uniform branch lengths, $\ell = 1/9$, $d^* = 1$, and $p_s = 0.1$.

Additional Implementation Details:

Threshold Algorithm: Due to ties in the number of shared characters, sometimes the edge-removal procedure produces more than two connected components on the sample graph G. If this occurs, we enforce a bifurcation in the tree by merging the components $C_1, C_2, ..., C_n$ into two groups. We expand the previous pseudocode as follows:

Here, we use a naive parsimony approach. We infer the character states of the LCA of each component as the most parsimonious states given the states of the leaves in that component (a mutation appears in the LCA of a component only if it shared by all samples in that component, discounting samples with missing data at that character). Then, the mu-

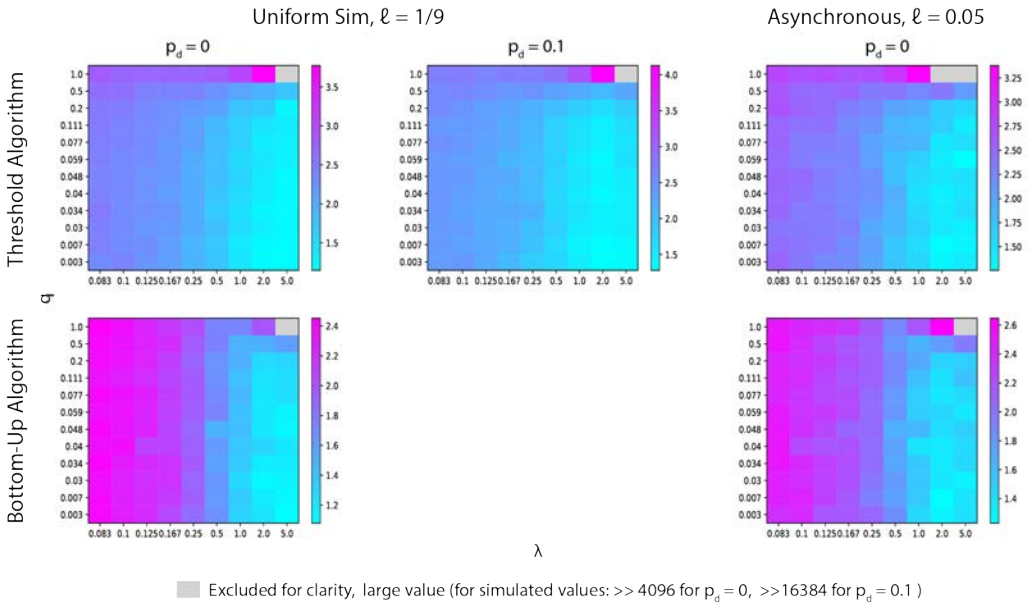


Figure 3.39: **Triplets Corrects Analysis for Both Algorithms**. Simulated trees with 256 leaves, $n = 256$. Entries are log_{10} scaled. Minimum k required for 0.9 probability of ≤ 0.95 proportion of 500 uniformly sampled triplets correctly reconstructed in simulation. Top row: Results for the Threshold Algorithm in the case of (From left to right) uniform edge length topology without missing data ($\ell = 1/9$, $p_s = 0.1$), uniform edge length topology with missing data ($\ell = 1/9$, $p_s = 0.1$), asynchronous topology ($\ell = 0.05$). Bottom row: Results for the Bottom-Up Algorithm in the case of (From left to right) uniform edge length topology ($\ell = 1/9$) and asynchronous topology ($\ell = 0.05$).

tation shared by the most LCAs is found. Heuristically, this mutation occurred early in the phylogeny and thus components sharing this mutation are more closely related. Thus, we separate components into two groups based on whether or not the LCA has this mutation.
This algorithm is implemented as the "PercolationSolver" class in the "solver" module of the Cassiopeia codebase. Here, the default arguments are used, with "joining_solver" specified as an instance of the "VanillaGreedySolver" class.

Bottom-Up Algorithm: The implementation of the Bottom-Up Algorithm follows the description in the main text. If a tie occurs in the number of shared mutations between nodes, then an arbitrary pair is chosen to be merged first.
This algorithm is implemented as the "SharedMutationSolver" class in the "solver" module of the Cassiopeia codebase. Here, the default arguments are used.

Simulating Cell-Division Topology We present the pseudocode used for this simulation here:
The topology simulation framework is implemented in the Cassiopeia codebase as the "BirthDeathFitnessSimulator" class in the "simulator" module. Here, "birth_waiting_distribution" is set to a lambda function that takes a rate and returns a random waiting time from an exponential distribution with that rate, shifted by 0.05. "initial_birth_scale" is set to ≈ 23.70.

```
1: procedure SplitSamples(V)
2:     G ← Complete graph over V
3:     while G is connected do
4:         (u*, v*) = argmin_{(u,v)∈E} s(u, v)
5:         Delete (u*, v*) from G
6:     C_1, C_2, ..., C_n ← connected components of G
7:     if length(C_1, C_2, ..., C_n) > 2 then
8:         C_1, C_2 ← MergeComponents(C_1, C_2, ..., C_n)
9:     T_1, T_2 ← SplitSamples(C_1), SplitSamples(C_2)
10:    Return binary tree with T_1 and T_2 as children of the root.
11:
12: procedure MergeComponents(C_1, C_2, ..., C_n)
13:    LCA_1, LCA_2, ..., LCA_n ← InferParsimoniousStates(C_1, C_2, ..., C_n)
14:    m ← FindMostFrequentMutation(LCA_1, LCA_2, ..., LCA_n)
15:    for i in n do
16:        if m in LCA_i then
17:            Add C_i to GroupA
18:        else
19:            Add C_i to GroupB
20:    Return GroupA, GroupB
```

"death_waiting_distribution" is set to a lambda function that takes no arguments and returns a random waiting time from an exponential distribution with that rate $1/ \approx 23.70 + 0.05$. "experiment_time" is set to 1. The other arguments are set to their defaults.

Simulating CRISPR-Cas9 Lineage Tracing Data The lineage tracing simulation framework is implemented in the Cassiopeia codebase as the "Cas9LineageTracingDataSimulator" class in the "simulator" module. Here, "number_of_cassettes" is set to the respective k, "size_of_cassette" is set to 1, "mutation_rate" is set to the respective λ, "state_priors" is set to a dictionary representing the q distribution, "heritable_silencing_rate" is set to 0, and "stochastic_silencing_rate" is set to 0 and 0.1, depending on whether the particular simulation has missing data. All other arguments are set to default.

Scoring Criterion

Full reconstruction: We say the reconstructed tree achieves perfect reconstruction if it has a Robinson-Foulds Distance of 0, meaning the trees are isomorphic with regard to their labels.
Robinson-Foulds Distance is implemented in the codebase as the "robinson_foulds" method in the "critique" module. This method makes use of the the Ete3 package [94].

Algorithm 4

procedure FORWARD SIMULATION($\lambda_{birth}, \lambda_{death}$)
    $Lineages \leftarrow Queue()$;
    $T \leftarrow$ new empty tree;
    $Leaves \leftarrow \{\}$
    $Lineages.push((root, None, 0))$
    $SampleLineageEvent(node, parent, time, T, Leaves, Lineages, \lambda_{birth}, \lambda_{death})$
    while $Lineages$ is not empty **do**
        $node, parent, time \leftarrow Lineages.pop()$
        $SampleLineageEvent(node, parent, time, T, Leaves, Lineages, \lambda_{birth}, \lambda_{death})$
    Remove all nodes in T that do not have a descendant in $Leaves$
    Return T

procedure SAMPLELINEAGEEVENT($node, parent, time, T, Leaves, Lineages, \lambda_{birth}, \lambda_{death}$))
    $t_{birth}, t_{death} \leftarrow Exp(\lambda_{birth}) + a, Exp(\lambda_{death})$;
    if $time + min(t_{birth}, t_{death}) > 1$ **then**
        $T \leftarrow T \cup \{(parent, node), weight = 1 - time\}$;
        $Leaves \leftarrow Leaves \cup \{node\}$;
    else if $t_{birth} < t_{death}$ **then**
        $T \leftarrow T \cup \{(parent, node), weight = t_{birth}\}$;
        $Lineages.push(Node(), node, time + t_{birth})$;

Partial Reconstruction: To determine the sufficient k needed for exact partial reconstruction for triplets whose LCA is up to depth d, we use the same framework as in the case of full reconstruction but we change the scoring criterion. We can no longer compare ground truth and reconstructed trees by Robinson-Foulds Distance, which compares the entire tree. We instead present and show the correctness of an algorithm to determine if all triplets in a tree up to depth d are resolved correctly in the reconstructed tree. The algorithm is as follows. Let $\mathcal{T}$ be the ground truth tree, $\mathcal{T}'$ be the reconstructed tree, and d be the depth: Next we prove its correctness:

Claim: All triplets whose LCA is at depth $< d$ in $\mathcal{T}$ are resolved correctly in $\mathcal{T}'$ iff for every node $n < d$ in $\mathcal{T}$ there exists a node n' in $\mathcal{T}'$ whose daughter clades partition the set of leaf descendants of that node in the same way.

Proof: if: For a triplet $(a, b|c)$ whose LCA is node n, a, b must be in the clade of one daughter of n and c on the other. If there exists a node n' in $\mathcal{T}'$ that partitions the leaf nodes into the same two clades, then for each triplet whose LCA is at n, a, b will be grouped together in the same clade with c on other side, hence every triplet will be resolved correctly. If there exists n' for each $n < d$, then all triplets with LCA $< d$ will be resolved correctly.

only if: If there is a node $n < d$ in $\mathcal{T}$ with no node n' in $\mathcal{T}'$ with an analogous partition, this implies that there is some partition of the leaves in $\mathcal{T}'$ starting from the root at or

Algorithm 5

procedure CHECKPARTIALRECONSTRUCTION($\mathcal{T}, \mathcal{T}', d$)
    CheckSplit(root($\mathcal{T}$), root($\mathcal{T}'$))
    Return TRUE

procedure CHECKSPLIT(n, n', d)
    if depth(n) $< d$ **then**
        return
    l, r = children(n)
    l', r' = children(n')
    if leaves_beneath(l) == leaves_beneath(l') and leaves_beneath(r) == leaves_beneath(r') **then**
        $CheckSplit(l, l', d)$, $CheckSplit(r, r', d)$
    else
        Return FALSE

above n that does not match the partition in $\mathcal{T}$, as if all partitions were correct then n' must exist. If there is a non-matching partition, there is some partition $\{a_1, ..., a_m\}|\{b_1, ..., b_n\}$ in $\mathcal{T}$ where in $\mathcal{T}'$ there is at least one incorrect member in one of the partitioned sets: $\{a_1, ..., a'_m, b_1, ..., b'_n\}|\{a'_{m+1}, ..., a_m, b_{n'+1}, ..., b_n\}$. Then in $\mathcal{T}'$, WLOG b_1 is closer to some a_i than some b_i, and all triplets involving b_1 and a_i are incorrect in $\mathcal{T}'$. As the partition with the non-analogous partition is at depth $< d$ in $\mathcal{T}$, then some triplets whose LCA is at depth $< d$ in are incorrectly resolved in $\mathcal{T}'$.
The algorithm will find n' for every $n < d$ by matching it to a node in $\mathcal{T}'$ that has the same partition. If the algorithm does not find n' for a certain $n < d$, then it does not exist as if all partitions checked by the algorithm match up to and including n in $\mathcal{T}'$ then it must exist, and the given partition cannot be formed later as leaf descendant sets cannot add members down the tree.

Triplets Correct: Additionally, we report the necessary k needed for 95% triplets correct in simulation. The triplets correct score is determined by sampling 500 triplets uniformly from the ground truth tree and counting the proportion of triplets resolved accurately in the reconstructed tree.

Chapter 4

Inferring Network Activity - Th17 Tissue Specific Metabolism

4.1 Collaborators and Contributions

The last section of this dissertation describes a work in progress between the Yosef Lab and the Kuchroo Lab (Harvard University). While experimental validations are currently ongoing, the computational results look promising and are summarized below.

The primary collaborators of this work include: Alex Khodaverdian, Allon Wagner (Berkeley), Linglin Huang (Harvard), Alexandra Schnell (Harvard), Martina Spiljar (Harvard), Nir Yosef (Principal Investigator, Berkeley), and Vijay Kuchroo (Principal Investigator, Harvard).

4.2 Introduction

$CD4^+$ T-cells, and in particular Th17 cells play an important role in tissue homeostasis and pathology [81, 14, 183, 117, 106, 112, 22]. In fact not all Th17 cells are identical, and they can often be stratified into subpopulations such as Effector-like, ISG (interferon stimulated gene)high, Treg-like, amongst others, with different subpopulations playing varying roles in the function and response of Th17 cells [60, 83, 64, 159, 200]. Differentiation of Th17 cells is influenced by various factors such as the presence of SFB and Candida albicans as previously shown by Litteman and others. However, not only are Th17 cells regulated for phenotype by tissue or microenvironment, but abundance as well. For example, in the murine gut Th17 cells are found to be most abundant within the duodenum, and least abundant within the colon, with the inverse relationship being true for T-regs [43]. In addition, the microenvironment has a significant impact on T cell activation and subsequent differentiation. In the case of ulcerative colitis (UC), intestinal stromal subtypes associated with cytokine signaling and T cell activation, were observed to be present in UC and scarce in

healthy controls [111]. Thus its understood that Th17 subpopulations and their abundance vary between tissues and microenvironments [96].

Another key component of $CD4^+$ T cell function emerges from cellular metabolism. In particular, different cell types may express different metabolic pathways and thus may rely on different metabolites for homeostasis. For example, effector-like T cell subsets all show an increase in the glycolysis pathway post activation [141, 132, 162]. In contrast, Tregs have been observed to express genes involved in fatty acid oxidation, while effector-like cells down-regulate such pathways [132]. Interestingly, inhibition of nutrients or pathways may also lead to a loss of phenotype [181]. For example, within effector T cells, loss of L-glutamine results in the inability for such cells to proliferate [23, 197]. More specifically, within Th17 cells, the polyamine pathway was more recently discovered as a critical regulator of Th17 effector-like function, whose inhibition led Th17 cells towards a Treg-like state [178].

Tissue microenvironments were shown to have diverse metabolic programs that influence immune cell function. For example within tumor microenvironments proliferative cancer cells are often supported by aerobic glycolysis. In such an environment nutrients are limited, and there is competition between cancer cells and immune cells. This leads to a metabolic switch in the immune cells from aerobic glycolysis to fatty acid oxidation which causes immuno-suppression and evasion of cancer cells from the immune system [19, 192, 121, 150, 24]. Within the gut, microbiota play an immense role in immunometabolism via metabolites such as short-chain fatty acids, bile acids, and tryptophan metabolites. For example, in $CD8^+$ T cells microbiota-derived short-chain fatty acids like butyrate help promote effector function by stimulating OXPHOS and mitochondrial mass, as well as glycolytic activity [133].

Therefore, given the observation of metabolic differences in tissues, and metabolism directly affecting Th17 function, it's crucial to understand how the metabolic programs directly link to Th17 function. This knowledge can benefit in designing therapeutics to treat various pathologies.

To this end, we turn our attention to single cell RNA Sequencing (scRNA-seq), where we may transcriptionally profile immune cells within various tissues. As mentioned above, this profiling goes beyond broad categorizations, and in the case of Th17 cells, we find heterogeneity within Th17 cell subpopulations with vastly different phenotypic signatures. However, we can actually take a step further, and explore these cells at a metabolic level. For this, we use Compass, an in silico based approach, which uses cell transcriptomic data and models network wide metabolic flux using a Flux Balance Analysis (FBA) [178]. The goal thus becomes to categorize and explore the heterogeneity of Th17 cells within various tissues at a metabolic difference, with the goal of identifying metabolic programs or metabolites of interest.

To tackle this goal, we turn our attention towards gut and lymphoid tissues. In particular,

Th17 cells within the gut are known for their role in tissue homeostasis and their influence in extra-intestinal autoimmune diseases. We utilize current and ex Th17 cells collected from mice across 5 tissues: colon, small intestine, peyer's patches, mesenteric lymph nodes, and spleen. We consider ex Th17 cells to identify instances of Th17 plasticity. We apply a combination of transcriptomic analysis, both at a tissue level and phenotypic level, to identify metabolic genes and programs of interest at a tissue specific level. We further this analysis by utilizing Compass to explore these cells at a network wide metabolic level. Our work identifies a gut-specific metabolic target creatine kinase, which we are validating experimentally in-vitro, responsible for regulating effector-like function and gut homeostasis.

4.3 Results

4.3.1 Single Cell Sequencing of Th17 Cells from multiple tissues

To characterize the metabolic differences across Th17 cells, Th17 cell data procured by Schnell et al [159] via scRNA-seq in mice Th17 was used. In particular, in order to collect both cells which both currently expressed IL17A as well as once expressed IL17A, Schnell et al began by crossing active IL17A-GFP reporter mice ($IL17A^{GFP}$) with IL-17A fate-reporter mice ($IL17A^{Cre}Rosa26^{tdTomato}$), generating $IL17A^{GFP}$ x $IL17A^{Cre}Rosa26^{tdTomato}$ mice.

From there $CD4^+$ $tdTomato^+$ cells were collected from 9 mice, spanning a subset of the following 5 tissues per mice: spleen (SPL), mesenteric lymph node (MLN), Peyer's patches (PP), small intestine (SI), and colon (COL). From there, from the cells collected via scRNA-seq, IL17A-GFP^+ $tdTomato^+$ cells were labeled as current Th17 cells, whereas IL17A-GFP^- $tdTomato^+$ cells were labeled as ex Th17 cells.

4.3.2 Differential expression analysis reveals metabolic differences between Th17 cells in different tissues

Although the heterogeneity of the whole transcriptional profiles in Th17 cells across different tissues has been characterized by Schnell et al, it remains unclear if the metabolic genes also show tissue-specific expression patterns. Differential expression analysis (Methods) identified a host of metabolic genes that are upregulated in at least one of the tissues (Figure 4.1B; FDR < 0.05, fold change > 1.5).

The UMAP visualization based on the metabolic latent space also showed clear separation of tissues (Figure 4.1C and Figure 4.3A), indicating that the tissues are distinct metabolically.

Lastly, we noted that the colonic Th17 cells had the biggest number of tissue-specific metabolic genes (Figure 4.1B) and are the most distinct from the splenic Th17 cells in the metabolic latent space (Methods; Figure 4.1C and Figure 4.1B), suggesting that Th17 cells

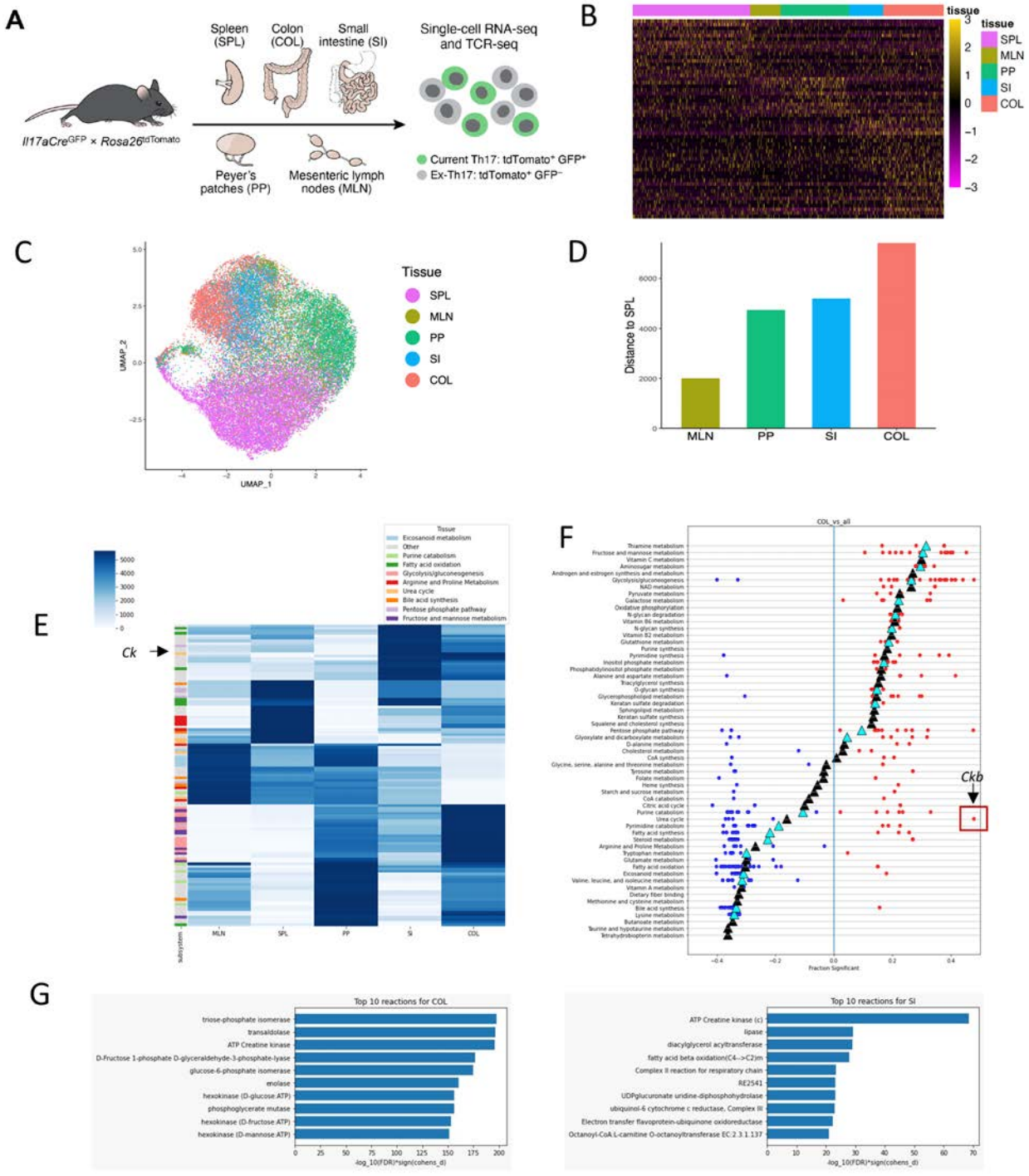


Figure 4.1: **Tissue Th17 cells are metabolically distinct** (a) Mouse model (b) Normalized expression of differentially expressed metabolic genes (rows) in a random sample of 1000 cells (columns) from all tissues. (DESeq2 was applied on pseudo-bulk data to compare each tissue to the average of all tissues; lowly expressed genes (expressed in $< 10\%$ of cells in every tissue) were excluded from the DE analysis; FDR < 0.05, fold change > 1.5) (c) Distance of tissues to SPL (measured as the inverse of the proportion of cells in each tissue being k-nearest neighbors of any SPL cells; $k = 20$) (d) UMAP based on metabolic gene expression only; colored by tissue. (e) Heatmap of top 25 metareactions per tissue in Compass. Each cell entry corresponds to the relative rank of that reaction relative to all other reactions of interest in a one versus all comparison. (f) Summary of all differentially expressed metareactions in a COL vs all comparison, summarized by subsystem (rows). Blue dots represent reactions lower in the colon, red dots represent reactions higher in the colon. Triangles represent mean cohen's d, with cyan triangles representing overrepresentation of a subsystem calculated via a hypergeometric test (g) Top 10 metareactions in the COL vs all comparison and SI vs all comparison. Reactions are ranked by $-log(FDR) * sign(cohens_d)$

may have adjusted to the complex metabolic environment in the colon and display differential and active metabolic characteristics.

4.3.3 Creatine kinase as a gut specific upregulated metabolic gene

Given the first hurdle, which is metabolic differences amongst tissues, has been cleared, the question now becomes what the specific differences between tissues are. To this end, we proceed by considering broad pathways upregulated per tissue, and from there proceed towards a gene level analysis.

Running a basic enrichment analysis amongst tissue specific differentially expressed genes, we find that within the spleen Urea Cycle and Arginine Biosynthesis are both upregulated. Elevating arginine levels in the spleen has been linked to a shift from glycolysis to oxidative phosphorylation in activated T cells, and promotes the generation of central memory-like T-cells with higher survival capacity [67]. Lastly, Arginine biosynthesis and Urea Cycle have been associated with both pro-Th17p and pro-Th17n cells [178].

Within the Peyer's Patches we find the upregulation of Purine metabolism, Pyrimidine metabolism, and Glycolysis. Lastly within the gut (Colon and Small Intestine) we find upregulation of Glutamine metabolism and Arginine metabolism. This result is consistent with the fact that arginine metabolism has been associated with increased pathogenicity in Th17 cells [178]. Unlike the spleen however, the Urea cycle is not found to be upregulated. Within the colon, glutamine metabolism has been associated with differentiation of Th17 cells, while restraining Th1 cells [97].

Moving onto specific genes, we find the following: within the Spleen we find increased expression of ASS1, a gene known to play a functional role in T cells [172], and a gene often knocked out in cancer cells [5, 42, 13]. In addition, we find DGKa upregulated, which alongside DGKgamma, controls Th17 cell differentiation [195].

Within the Peyer's patches, we find PLCG1, a gene associated with promoting Th17 trafficking to inflamed organs in the case of Lupus-prone mice, as well as a common finding in tumoral Cutaneous T-cell lymphomas [176, 142]. In addition, we find expression of NT5E (CD73), a marker typically associated with inflammatory Th17 cells, and a marker present in EAE model (Although whose KO doesn't result in EAE severity differences) [80]. Lastly, CD73 expressed on Treg exosomes contribute to Treg suppressive activity .

Within the gut we find upregulation of ODC1, a cell typically found to be associated with pro inflammatory Th17 cell function, consistent with the belief that Th17 cells within the gut would be more active than in immune tissues such as the spleen [178]. In addition

in the gut, we find increased expression of DGAT1, a protein found to inhibit Treg function, consistent with the view of increased effector activity in the gut [70]. We also find both GCLC and GOT1 have been associated with increased autoimmune response, the KO of both leading to reductions in autoimmunity [122, 191].

Interestingly, although we did not see any broad Urea Cycle upregulation in the gut, we found the gene CKB upregulated in both the Colon and Small Intestine. From an immunological standpoint, creatine has been broadly associated with increased antitumor activity in CD8 cells [45]. However, on the flip side, there have been studies which suggest that CKB silencing can lead to pro-inflammatory responses for example in the case of white adipocytes [123]. In the gut specifically, creatine has been found to maintain intestinal homeostasis and protect against colitis [175]. However, there have been no studies on CKB and Th17 cells within the gut specifically.

4.3.4 In sillico Flux Balance Analysis with Compass

Given the broad metabolic differences observed in the single cell gene expression data, as well as the metabolic differences in both pathways and at a gene level, we surmised that it would be of interest to further characterize our cells on the immunometabolism level. To this end we applied Compass, an Insilco Flux Balance Analysis based approach, characterizing and spanning the entire metabolic model provided (mouse), and providing confidence scores per reaction in the network (Methods).

We first begin by verifying once again, is there a metabolic difference between tissues at a reaction level? To answer this we note the heterogeneity observed between tissues, as seen in Figure 4.1E (rank normalized reactions per tissue, the higher meaning more active relative to all other reactions). Amongst this heterogeneity we make some interesting observations, in no particular order. We first note the upregulation of glycolysis/gluconeogenesis in the COL. In contrast, we notice an upregulation of Fatty Acid Oxidation, Urea Cycle, and Arginine and Proline Metabolism amongst the Spleen. These findings were mostly present in the gene expression analysis.

When zooming in on the Colon (Figure 4.1F), we note some of the top subsystems: Thiamine metabolism, Fructose and Mannose metabolism (a subsystem of Glycolysis), Aminosugar Metabolism, and Glycolysis, subsystems typically associated with energy hungry cells, i.e. Effector-like Th17 cells. In contrast some of the most downregulated subsystems include Bile Acid Synthesis and Lysine Metabolism.

Zooming in on the gut in particular, we can further focus ourselves on the top 10 reactions per tissue (Figure 4.1G). We note that amongst the many reactions, most are related to glycolysis (or a corresponding subsystem). However, we note of interest the reaction “ATP

Creatine Kinase". This reaction seems to be associated with the gene of interest discussed earlier (CKB). In addition, this reaction is upregulated in the Urea Cycle, despite being a part of a subsystem which is downregulated in the Colon (Figure 4.1F).

4.3.5 Cell state axis

Thus far, we have looked at every tissue in aggregate against one another. However, we know that various tissues can exhibit different distributions of subpopulations amongst their respective Th17 populations. Given this fact, we decided to further characterize each tissue by subpopulation similar to Schnell et al. Every tissue was clustered individually via Leiden clustering (Figure 4.2A, top row), and cluster-specific upregulated genes were identified within each tissue. Next, we computed the distance between pairs of tissue clusters based on the Spearman dissimilarity of average expression of the union of all cluster-specific upregulated genes. Then hierarchical clustering was performed to identify clusters amongst different tissues with the same characteristic phenotype (Figure 4.2B). In particular, we identified 7 "superclusters" that correspond to naive, proliferating, Treg-like, effector-like, Tfh-like, migratory and ISG-high (Figure 4.2A, bottom row), each were annotated based on the top cluster-specific upregulated genes (Figure2C).

Given these subpopulations, our next steps were to focus on the gut (COL and SI). In particular, we ran a one versus all analysis per subpopulation intra-tissue (ex proliferating ex Treg). Given the presence of CKB largely upregulated in the COL and SI, the question became - is there a particular subpopulation where CKB is upregulated, or is it just broadly upregulated in the Colon. We note that the ATP Creatine Kinase observed in both the COL and SI was specifically up in the Effector-like populations in both tissues, and was in fact the top hit in both subpopulations (Highlighted in orange in Figure 4.2D).

4.3.6 An experimental target emergences in creatine kinase

Given the goals of this project, and the positive results observed for creatine kinase (CKB) across the various axes of analysis, our group has chosen to further explore this target experimentally. In particular, we were able to consider our problem from both the lens of transcriptomics and at a reaction level, giving us a working hypothesis of a possible regulator of gut specific Th17 cells. During this analysis, we also took careful consideration of the fact that different tissues can exhibit different subpopulations of Th17 cells, and even under such a lens, found creatine kinase to be upregulated. Given further time and analysis, I'm confident we would have been able to consider other metabolic targets of interest given the depth of our dataset.

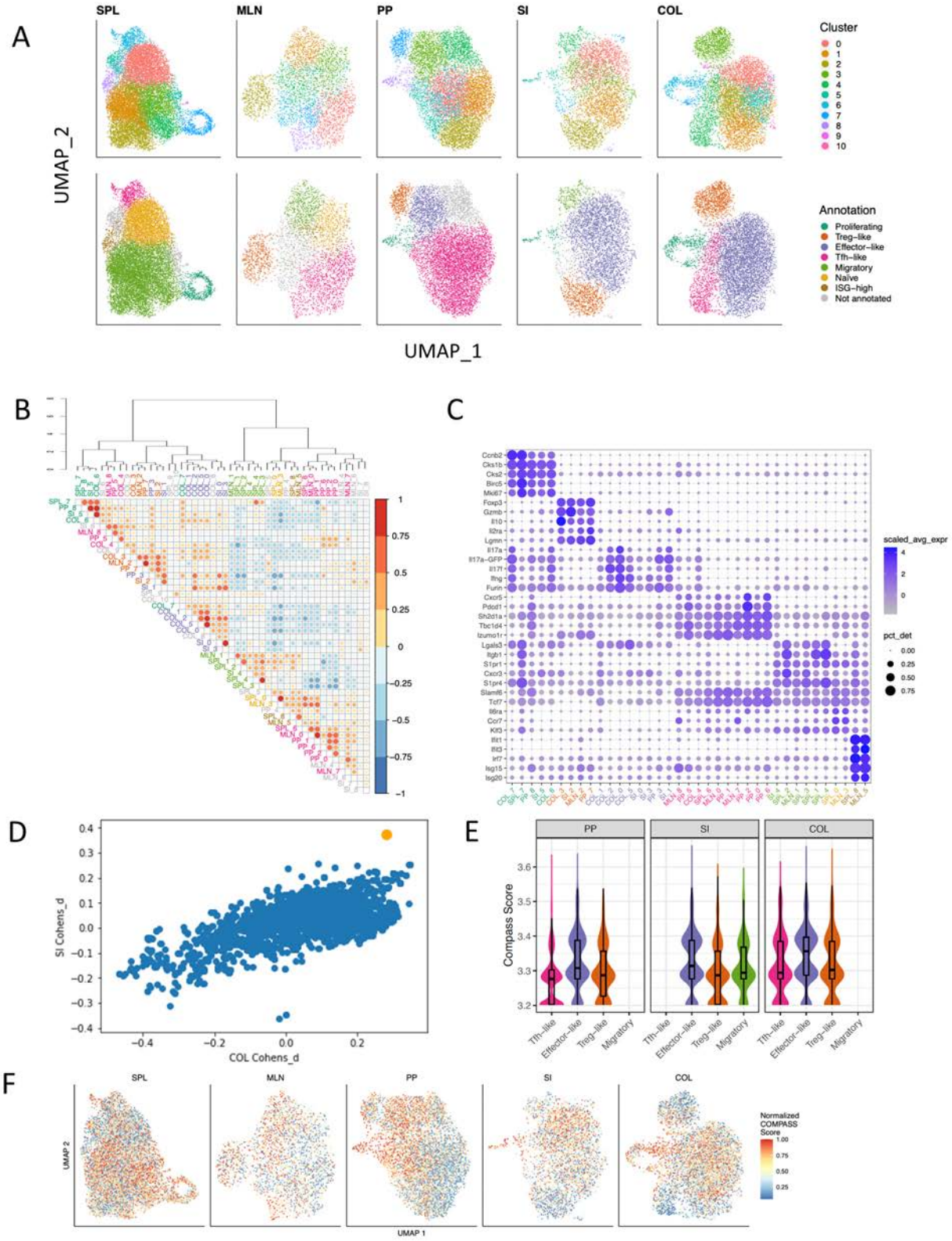


Figure 4.2: **Common metabolic features of intra-tissue variation in Th17 cells across tissues** (a) Annotation of superclusters (b) Correlation of intratissue clusters in metabolic space (c) Genes used for supercluster annotation. (d) - (e) Average COMPASS scores of CK_{pos} reaction. Scores were rank-normalized within each tissue (higher score means more "active"). (f) COMPASS scores of CK_{pos} reaction. Scores were rank-normalized within each tissue (higher score means more "active").

4.4 Methods

4.4.1 Single-cell RNA-seq data analysis

Preprocessed UMI count matrices were obtained from Schnell et al, where low quality or non-Th17 cells have been excluded. Given the cursory filtered single cell gene expression data collected originated from 9 different mice, our next step was to preprocess the data to correct for batch effects, and to generate an accurate latent space void of such covariates. To this end, we employed scvi-tools, a package designed for probabilistic modeling and analysis of single-cell omics data (Gayoso et al. 2022). Scvi-tools employs a variational autoencoder to train over gene expression data, with the ability to deal with natural problems related to single cell data such as dropout, imputation, and batch effects. To this end, we ran scvi-tools over the metabolic gene expression space generating a metabolic latent space of 10 dimensions, corrected for batch effects (Supplemental Fig 4.3).

Differential expression analyses were performed with edgeR [ref] on pseudo-bulk data. The tissue comparison in Figure 4.1B models the mean expression as a linear combination of tissue (categorical, summation coded) and batch (categorical, dummy coded) using a negative binomial regression (glmFit function). The effect of each tissue was tested against the average of all tissues using moderated likelihood ratio tests (glmLRTest function). Lowly detected genes were excluded prior to fitting the model. Results were intersected with the metabolic gene list. Tissue vs. spleen comparison in Figure S1A was computed using the same fitted model as in Figure 4.1B, but with different contrast vectors such that each tissue was compared to the spleen rather than the tissue averages.

Tissue distances in Figure 4.1D were measured by the non-overlapping of tissue cells' nearest neighbors in the scVI latent space. Firstly, k-nearest neighbors (k = 30) were identified for every cell in the scVI latent space (top 10 dimensions; Euclidean distance). Next, for each pair of tissues, compute the number of times a cell in one tissue being a k-nearest neighbor of any other cells in the other tissue, then divide by the product of the number of cells in the two tissues. While the proportion is a measure of the tissue similarity, taking its inverse gives the tissue distance.

4.4.2 Metabolic Network Analysis via Compass

Compass takes in a single cell gene expression counts matrix alongside a metabolic network model (mouse). The output is a cell by reaction matrix M, spanning every reaction in the metabolic network, with a likelihood score per reaction.

Once we have our matrix M, our interest becomes to identify reactions and/or metabolites of interest which are upregulated in each tissue.

Given the inherent metabolic differences per tissue, if we were to run a simple one versus all analysis per tissue, the Colon would come up as metabolically more active for almost all reactions. As a result, our goal became to understand which metabolic programs are relatively more active in each tissue over another. To this end, for each cell in our matrix M, we begin by Z-normalizing across all reactions. From there, for each tissue and for each reaction, we run a one versus all analysis as follows: For a particular reaction we run a Wilcoxon ranksum test between all cells in a tissue versus all cells in all other tissues to extract a p-value. For effect size, we take the cohen's D between the two groups. Given we have many reactions, we adjust our p-values using the Benjamini-Hochberg procedure. At the end of this process, we're left with 5 comparisons of interest, one per tissue, for all 5 tissues of interest. Each comparison contains an FDR, and cohen's D for every reaction present in the metabolic network.

4.5 Supplementary Figures

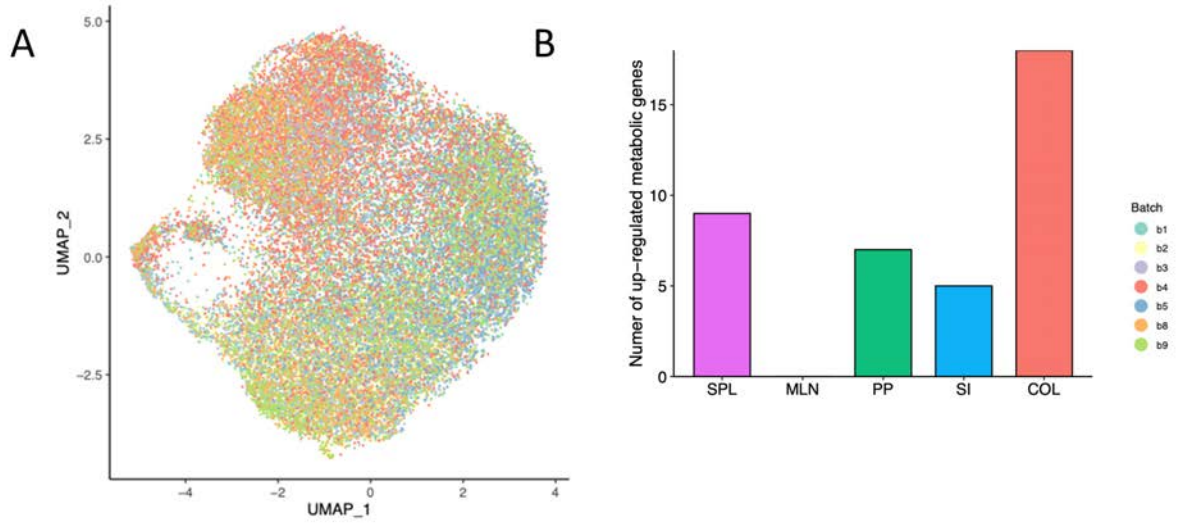


Figure 4.3: (a) Number of metabolic genes that are upregulated in each tissue (same DE method and criteria as Fig 1B) (b) UMAP based on metabolic gene expression only; colored by batch.

Chapter 5

Conclusion

In this dissertation I have explored various abstractions of biological systems as networks, over which we have defined network optimization algorithms, with analysis of theoretical guarantees and applications to measurements of molecular abundance (phosphoproteomics, transcriptomics). Within each subsection, we discuss future directions. Here I summarize a few of the most interesting future directions, as well as other directions not considered in the past chapters.

When it comes towards subnetwork optimization in our Condition settings, an open question that has always remained on my mind is whether the generalized monotonic Directed Condition Steiner Network has an approximation algorithm that admits better approximation bounds than the trivial bound. Recall instances are *monotonic*: if an edge e exists in some graph G_c, then it exists in all the subsequent graphs $G_{c'}, c' \geq c$. In Chapter 2 of this dissertation we successfully showed via approximation preserving reduction to Directed Steiner Tree that Monotonic Single-Source Directed Condition Steiner Network has an $O(k^\epsilon)$-approximation algorithm for every $\epsilon > 0$. Attempts to apply similar reduction techniques to the generalized monotonic Directed Condition Steiner Network have proved to be unsuccessful, but I believe this problem deserves some further thought.

Crispr/Cas9 lineage tracing systems, coupled with methodologies such as Cassiopeia have already seen use in a variety of biological studies. We've already seen these methods applied towards the unprecedented exploration of zebrafish [129, 147, 165, 180] and mouse development [103, 28] in regards to embryogenesis. More recent work by Yang et al considered the phylodynamics, plasticity and paths of tumor evolution. I believe we will continue to see applications of Crispr/Cas9 technology towards the study and analysis of naturally phylogenetic biological processes, and further improvements in methodology with time.

Lastly, I believe the applications of Compass and analysis of metabolic systems within cells is very early. Thus far, Compass has been primarily utilized for the study of Th17 cells, be it by Wagner et al, or by the work presented in Chapter 4 of this dissertation. I can

envision in the near future applications towards other cell types and other disease states, and to provide a view of cellular metabolism previously nebulous to when considering transcriptomics.

Overall, I'd like to say that I found this entire PhD program quite interesting and fulfilling. I really enjoyed approaching my research from the perspective of "What biological problem are we trying to solve? How do we model this problem as a network?". From this perspective, I was able to abstract our problem, analyze from the lens of a Computer Scientist, and then turn around and reapply those abstractions towards actual biological data. I'm grateful for the time spent at Berkeley and thankful for all the incredible people I was able to work with and learn from throughout this program.

Bibliography

[1] Nandini Acharya et al. "Endogenous glucocorticoid signaling regulates CD8+ T cell differentiation and development of dysfunction in the tumor microenvironment". en. In: *Immunity* 53.3 (Sept. 2020), 658–671.e6.

[2] Britt Adamson et al. "A Multiplexed Single-Cell CRISPR Screening Platform Enables Systematic Dissection of the Unfolded Protein Response". In: *Cell* 167.7 (2016), 1867–1882.e21. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2016.11.048. URL: http://www.sciencedirect.com/science/article/pii/S0092867416316609.

[3] Ajit Agrawal, Philip Klein, and R Ravi. "When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks". In: *SIAM Journal on Computing* 24.3 (1995), pp. 440–456.

[4] Anna Alemany et al. "Whole-organism clone tracing using single-cell sequencing". In: *Nature* 556.7699 (2018), pp. 108–112.

[5] Constantinos Alexandrou et al. "Sensitivity of colorectal cancer to arginine deprivation therapy is shaped by differential expression of urea cycle enzymes". en. In: *Sci. Rep.* 8.1 (Aug. 2018), p. 12096.

[6] Felicity Allen et al. "Predicting the mutations generated by repair of Cas9-induced double-strand breaks". In: *Nature Biotechnology* 37 (Nov. 2018). URL: https://doi.org/10.1038/nbt.4317.

[7] Andrew V Anzalone, Luke W Koblan, and David R Liu. "Genome editing with CRISPR–Cas nucleases, base editors, transposases and prime editors". In: *Nature biotechnology* 38.7 (2020), pp. 824–844.

[8] Andrew V Anzalone et al. "Search-and-replace genome editing without double-strand breaks or donor DNA". In: *Nature* 576.7785 (2019), pp. 149–157.

[9] Aaron Archer et al. "Improved Approximation Algorithms for Prize-Collecting Steiner Tree and TSP". In: *SIAM journal on computing* 40.2 (2011), pp. 309–332.

[10] Sanjeev Arora et al. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM (JACM)* 45.3 (1998), pp. 501–555.

[11] Sanjeev Arora et al. "The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations". In: *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*. IEEE. 1993, pp. 724–733.

[12] Kevin Atteson. "The performance of neighbor-joining methods of phylogenetic reconstruction". In: *Algorithmica* 25.2 (1999), pp. 251–278.

[13] Leslie A. Bateman et al. "Argininosuccinate Synthase 1 is a Metabolic Regulator of Colorectal Cancer Pathogenicity". In: *ACS Chemical Biology* 12.4 (Apr. 2017), pp. 905–911. ISSN: 1554-8929. DOI: 10.1021/acschembio.6b01158. URL: https://doi.org/10.1021/acschembio.6b01158.

[14] Simone Kennedy Bedoya et al. "Th17 cells in immunity and autoimmunity". en. In: *Clin. Dev. Immunol.* 2013 (Dec. 2013), p. 986789.

[15] Taly Ben-Shitrit et al. "Systematic identification of gene annotation errors in the widely used yeast mutation collections". In: *Nat Meth* 9.4 (Apr. 2012), pp. 373–378.

[16] Sean C. Bendall et al. "Single-Cell Mass Cytometry of Differential Immune and Drug Responses Across a Human Hematopoietic Continuum". In: *Science* 332.6030 (May 2011). 21551058[pmid], pp. 687–696. ISSN: 0036-8075. DOI: 10.1126/science.1198704. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3273988/.

[17] Michele Berlingerio, Fabio Pinelli, and Francesco Calabrese. "ABACUS: frequent pAttern mining-BAsed Community discovery in mUltidimensional networkS". In: *Data Mining and Knowledge Discovery* 27 (Mar. 2013). DOI: 10.1007/s10618-013-0331-0.

[18] Hans L. Bodlaender, Mike R. Fellows, and Tandy J. Warnow. "Two strikes against perfect phylogeny". In: *Automata, Languages and Programming*. Ed. by W. Kuich. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 273–283. ISBN: 978-3-540-47278-0.

[19] Michael D Buck et al. "Metabolic instruction of immunity". en. In: *Cell* 169.4 (May 2017), pp. 570–586.

[20] Jaroslaw Byrka et al. "An Improved LP-based Approximation for Steiner Tree". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM. 2010, pp. 583–592.

[21] Joseph H. Camin and Robert R. Sokal. "A Method for Deducing Branching Sequences in Phylogeny". In: *Evolution* 19.3 (1965), pp. 311–326. ISSN: 00143820, 15585646. URL: http://www.jstor.org/stable/2406441.

[22] Anthony T Cao et al. "Th17 cells upregulate polymeric Ig receptor and intestinal IgA and contribute to intestinal homeostasis". en. In: *J. Immunol.* 189.9 (Nov. 2012), pp. 4666–4673.

[23] Erikka L Carr et al. "Glutamine uptake and metabolism are coordinately regulated by ERK/MAPK during T lymphocyte activation". en. In: *J. Immunol.* 185.2 (July 2010), pp. 1037–1044.

[24] Tina Cascone et al. "Increased tumor glycolysis characterizes immune resistance to adoptive T cell therapy". en. In: *Cell Metab.* 27.5 (May 2018), 977–987.e4.

[25] L. L. Cavalli-Sforza and A. W. F. Edwards. "Phylogenetic Analysis: Models and Estimation Procedures". In: *Evolution* 21.3 (1967), pp. 550–570.

[26] James A. Cavender. "Taxonomy with confidence". In: *Mathematical Biosciences* 40.3 (1978), pp. 271–280. ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(78)90089-5. URL: https://www.sciencedirect.com/science/article/pii/0025556478900895.

[27] Michelle M Chan et al. "Molecular recording of mammalian embryogenesis". In: *Nature* 570.7759 (2019), pp. 77–82.

[28] Michelle M. Chan et al. "Molecular recording of mammalian embryogenesis". In: *Nature* 570.7759 (2019), pp. 77–82. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1184-5.

[29] Moses Charikar, Joseph Naor, and Baruch Schieber. "Resource optimization in QoS multicast routing of real-time multimedia". In: *IEEE/ACM Transactions on Networking* 12.2 (2004), pp. 340–348.

[30] Moses Charikar et al. "Approximation Algorithms for Directed Steiner Problems". In: *Journal of Algorithms* 33.1 (1999), pp. 73–91.

[31] Andrew Chatr-aryamontri et al. "The BioGRID interaction database: 2015 update". In: *Nucleic Acids Research* 43.Database issue (Jan. 2015), pp. D470–D478. DOI: 10.1093/nar/gku1204.

[32] Chandra Chekuri et al. "Set Connectivity Problems in Undirected Graphs and the Directed Steiner Network Problem". In: *ACM Transactions on Algorithms (TALG)* 7.2 (2011), p. 18.

[33] Peter J Chen et al. "Enhanced prime editing systems by manipulating cellular determinants of editing outcomes". In: *Cell* (2021).

[34] Wei Chen et al. "Massively parallel profiling and predictive modeling of the outcomes of CRISPR/Cas9-mediated double-strand break repair". In: *bioRxiv* (2018). DOI: 10.1101/481069. eprint: https://www.biorxiv.org/content/early/2018/11/28/481069.full.pdf.

[35] Miroslav Chlebık and Janka Chlebıková. "The Steiner tree problem on graphs: Inapproximability results". In: *Theoretical Computer Science* 406.3 (2008), pp. 207–214.

[36] Junhong Choi et al. "Precise genomic deletions using paired prime editing". In: *bioRxiv* (2021), pp. 2020–12.

[37] Julia Chuzhoy et al. "On the Approximability of Some Network Design Problems". In: *ACM Transactions on Algorithms (TALG)* 4.2 (2008), p. 23.

[38] Francesca D. Ciccarelli et al. "Toward Automatic Reconstruction of a Highly Resolved Tree of Life". In: *Science* 311.5765 (2006), pp. 1283–1287. ISSN: 0036-8075. DOI: 10.1126/science.1123061. eprint: https://science.sciencemag.org/content/311/5765/1283.full.pdf. URL: https://science.sciencemag.org/content/311/5765/1283.

[39] Douglas E. Critchlow, Dennis K. Pearl, and Chunlin Qian. "The Triples Distance for Rooted Bifurcating Phylogenetic Trees". In: *Systematic Biology* 45.3 (1996), pp. 323–334. ISSN: 10635157, 1076836X. URL: http://www.jstor.org/stable/2413567.

[40] Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. "Optimal Phylogenetic Reconstruction". In: STOC '06 (2006), pp. 159–168. DOI: 10.1145/1132516.1132540. URL: https://doi.org/10.1145/1132516.1132540.

[41] Constantinos Daskalakis, Elchanan Mossel, and Sebastien Roch. "Phylogenies without Branch Bounds: Contracting the Short, Pruning the Deep". In: *Research in Computational Molecular Biology*. Ed. by Serafim Batzoglou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 451–465. ISBN: 978-3-642-02008-7.

[42] Barbara Delage et al. "Arginine deprivation and argininosuccinate synthetase expression in the treatment of cancer". en. In: *Int. J. Cancer* 126.12 (June 2010), pp. 2762–2772.

[43] Timothy L Denning et al. "Functional specializations of intestinal dendritic cell and macrophage subsets that control Th17 and regulatory T cell responses are dependent on the T cell/APC ratio, source of mouse strain, and regional localization". en. In: *J. Immunol.* 187.2 (July 2011), pp. 733–747.

[44] U Deppe et al. "Cell lineages of the embryo of the nematode Caenorhabditis elegans". In: 75.1 (1978), pp. 376–380. DOI: 10.1073/pnas.75.1.376.

[45] Stefano Di Biase et al. "Creatine uptake regulates CD8 T cell antitumor immunity". en. In: *J. Exp. Med.* 216.12 (Dec. 2019), pp. 2869–2882.

[46] Irit Dinur and Pasin Manurangsi. "ETH-Hardness of Approximating 2-CSPs and Directed Steiner Network". In: *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.

[47] Yevgeniy Dodis and Sanjeev Khanna. "Designing Networks with Bounded Pairwise Distance". In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM. 1999, pp. 750–759.

[48] James S. Farris. "A Probability Model for Inferring Evolutionary Trees". In: *Systematic Zoology* 22.3 (1973), pp. 250–256. ISSN: 00397989. URL: http://www.jstor.org/stable/2412305.

[49] James S. Farris. "Methods for Computing Wagner Trees". In: *Systematic Zoology* 19.1 (1970).

[50] Uriel Feige. "A Threshold of Ln N for Approximating Set Cover". In: *Journal of the ACM (JACM)* 45.4 (1998), pp. 634–652.

[51] Jon Feldman and Matthias Ruhl. "The Directed Steiner Network problem is tractable for a constant number of terminals". In: *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE. 1999, pp. 299–308.

[52] Moran Feldman, Guy Kortsarz, and Zeev Nutov. "Improved Approximating Algorithms for Directed Steiner Forest". In: *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2009, pp. 922–931.

[53] J Felsenstein. "PHYLIP (Phylogeny Inference Package)". In: *Distributed by the author. Department of Genome Sciences, University of Washington, Seattle* ().

[54] Joseph Felsenstein. "Evolutionary trees from DNA sequences: A maximum likelihood approach". In: *Journal of Molecular Evolution* 17.6 (Nov. 1981), pp. 368–376. ISSN: 1432-1432. DOI: 10.1007/BF01734359. URL: https://doi.org/10.1007/BF01734359.

[55] Jean Feng et al. "Estimation of cell lineage trees by maximum-likelihood phylogenetics". In: *bioRxiv* (2019). DOI: 10.1101/595215. eprint: https://www.biorxiv.org/content/early/2019/03/31/595215.full.pdf. URL: https://www.biorxiv.org/content/early/2019/03/31/595215.

[56] Walter Fitch. "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology". In: *Systematic Zoology* 20.4 (1971).

[57] Walter M. Fitch and Emanuel Margoliash. "Construction of Phylogenetic Trees". In: *Science* 155.3760 (1967), pp. 279–284. ISSN: 0036-8075. DOI: 10.1126/science.155.3760.279. eprint: https://science.sciencemag.org/content/155/3760/279.full.pdf. URL: https://science.sciencemag.org/content/155/3760/279.

[58] Denise C Fitzgerald et al. "Interferon regulatory factor (IRF) 3 is critical for the development of experimental autoimmune encephalomyelitis". en. In: *J. Neuroinflammation* 11.1 (July 2014), p. 130.

[59] Doreen M Floss et al. "Defining the functional binding sites of interleukin 12 receptor β1 and interleukin 23 receptor to Janus kinases". en. In: *Mol. Biol. Cell* 27.14 (July 2016), pp. 2301–2316.

[60] Nicola Gagliani et al. "Th17 cells transdifferentiate into regulatory T cells during resolution of inflammation". en. In: *Nature* 523.7559 (July 2015), pp. 221–225.

[61] Farshid S. Garmaroudi et al. "Pairwise network mechanisms in the host signaling response to coxsackievirus B3 infection". In: *Proceedings of the National Academy of Sciences* 107.39 (2010), pp. 17053–17058. ISSN: 0027-8424. DOI: 10.1073/pnas.1006478107. eprint: https://www.pnas.org/content/107/39/17053.full.pdf. URL: https://www.pnas.org/content/107/39/17053.

[62] O Gascuel and M Steel. "Neighbor-Joining Revealed". In: *Molecular Biology and Evolution* 23.11 (2006), pp. 1997–2000. DOI: 10.1093/molbev/msl072.

[63] Olivier Gascuel and Mike Steel. "Neighbor-Joining Revealed". In: *Molecular Biology and Evolution* 23.11 (July 2006), pp. 1997–2000. ISSN: 0737-4038. DOI: 10.1093/molbev/msl072. eprint: https://academic.oup.com/mbe/article-pdf/23/11/1997/4146478/msl072.pdf. URL: https://doi.org/10.1093/molbev/msl072.

[64] Jellert T Gaublomme et al. "Single-cell genomics unveils critical regulators of Th17 cell pathogenicity". en. In: *Cell* 163.6 (Dec. 2015), pp. 1400–1412.

[65] Nicole M. Gaudelli et al. "Programmable base editing of A*T to G*C in genomic DNA without DNA cleavage". In: *Nature* 551 (Oct. 2017). Article. URL: https://doi.org/10.1038/nature24644.

[66] Jason M. Gehrke et al. "An APOBEC3A-Cas9 base editor with minimized bystander and off-target activities". In: *Nature Biotechnology* 36 (July 2018). URL: https://doi.org/10.1038/nbt.4199.

[67] Roger Geiger et al. "L-arginine modulates T cell metabolism and enhances survival and anti-tumor activity". en. In: *Cell* 167.3 (Oct. 2016), 829–842.e13.

[68] Luke A. Gilbert et al. "Genome-Scale CRISPR-Mediated Control of Gene Repression and Activation". In: *Cell* 159.3 (2014), pp. 647–661. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2014.09.029. URL: http://www.sciencedirect.com/science/article/pii/S0092867414011787.

[69] Wuming Gong et al. "Benchmarked approaches for reconstruction of in vitro cell lineages and in silico models of C. elegans and M. musculus developmental trees". In: *Cell Systems* (2021).

[70] Kareem L. Graham et al. "DGAT1 inhibits retinol-dependent regulatory T cell formation and mediates autoimmune encephalomyelitis". In: *Proceedings of the National Academy of Sciences* 116.8 (2019), pp. 3126–3135. DOI: 10.1073/pnas.1817669116. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1817669116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1817669116.

[71] Ilan Gronau, Shlomo Moran, and Sagi Snir. "Fast and Reliable Reconstruction of Phylogenetic Trees with Very Short Edges". In: SODA '08 (2008), pp. 379–388.

[72] M Grotschel, A Martin, and R Weismanel. "The Steiner Tree packing problem in VLSI design". In: *Mathematical Programming* 78 (1997), pp. 265–281.

[73] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018. URL: http://www.gurobi.com.

[74] Dan Gusfield. "Efficient algorithms for inferring evolutionary trees". In: *Networks* 21.1 (1991), pp. 19–28. DOI: 10.1002/net.3230210104. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230210104.

[75] Dan Gusfield. "The Multi-State Perfect Phylogeny Problem with Missing and Removable Data: Solutions via Integer-Programming and Chordal Graph Theory". In: *Journal of Computational Biology* 17.3 (2010), pp. 383–399. DOI: 10.1089/cmb.2009.0200.

[76] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[77] Oskar Hagen and Tanja Stadler. "TreeSim GM: Simulating phylogenetic trees under general Bellman–Harris models with lineage-specific shifts of speciation and extinction in R". In: *Methods in ecology and evolution* 9.3 (2018), pp. 754–760.

[78] Eran Halperin and Robert Krauthgamer. "Polylogarithmic Inapproximability". In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM. 2003, pp. 585–594.

[79] Christopher S Helvig, Gabriel Robins, and Alexander Zelikovsky. "An Improved Approximation Scheme for the Group Steiner Problem". In: *Networks* 37.1 (2001), pp. 8–20.

[80] Gerard Hernandez-Mir and Mandy J McGeachy. "CD73 is expressed by inflammatory Th17 cells in experimental autoimmune encephalomyelitis but does not limit differentiation or pathogenesis". en. In: *PLoS One* 12.3 (Mar. 2017), e0173655.

[81] Nydiaris Hernández-Santos and Sarah L. Gaffen. "Th17 Cells in Immunity to ¡em¿Candida albicans¡/em¿". In: *Cell Host & Microbe* 11.5 (May 2012), pp. 425–435. ISSN: 1931-3128. DOI: 10.1016/j.chom.2012.04.008. URL: https://doi.org/10.1016/j.chom.2012.04.008.

[82] Gaelen T. Hess et al. "Directed evolution using dCas9-targeted somatic hypermutation in mammalian cells". In: *Nature Methods* 13 (Oct. 2016). Article. URL: https://doi.org/10.1038/nmeth.4038.

[83] Keiji Hirota et al. "Fate mapping of IL-17-producing T cells in inflammatory responses". en. In: *Nat. Immunol.* 12.3 (Mar. 2011), pp. 255–263.

[84] Petter Holme. "Modern temporal network theory: A colloquium". In: *The European Physical Journal B* 88 (Aug. 2015). DOI: 10.1140/epjb/e2015-60657-4.

[85] Petter Holme and Jari Saramäki. "Temporal networks". In: *Physics Reports* 519.3 (2012). Temporal Networks, pp. 97–125. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2012.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0370157312000841.

[86] Peter V. Hornbeck et al. "PhosphoSitePlus, 2014: mutations, PTMs and recalibrations". In: *Nucleic Acids Res* 43.Database issue (2015), pp. D512–D520. ISSN: 0305-1048. DOI: 10.1093/nar/gku1267. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4383998/.

[87] Patrick D Hsu et al. "DNA targeting specificity of RNA-guided Cas9 nucleases". In: *Nature biotechnology* 31.9 (2013), pp. 827–832.

[88] S. S. Huang and E. Fraenkel. "Integrating proteomic, transcriptional, and interactome data reveals hidden components of signaling and regulatory networks". In: *Sci Signal* 2.81 (July 2009), ra40.

[89] Shao-shan Carol Huang and Ernest Fraenkel. "Integrating Proteomic, Transcriptional, and Interactome Data Reveals Hidden Components of Signaling and Regulatory Networks". In: *Science Signaling* 2.81 (2009), ra40–ra40. ISSN: 1945-0877. DOI: 10.1126/scisignal.2000350. eprint: http://stke.sciencemag.org/content/2/81/ra40.full.pdf. URL: http://stke.sciencemag.org/content/2/81/ra40.

[90] Silu Huang, Ada Wai-Chee Fu, and Ruifeng Liu. "Minimum Spanning Trees in Temporal Graphs". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 419–430. ISBN: 9781450327589. DOI: 10.1145/2723372.2723717. URL: https://doi.org/10.1145/2723372.2723717.

[91] Wolfgang Hueber et al. "Secukinumab, a human anti-IL-17A monoclonal antibody, for moderate to severe Crohn's disease: unexpected results of a randomised, double-blind placebo-controlled trial". en. In: *Gut* 61.12 (Dec. 2012), pp. 1693–1700.

[92] John P. Huelsenbeck and Fredrik Ronquist. "MRBAYES: Bayesian inference of phylogenetic trees". In: *Bioinformatics* 17.8 (2001), pp. 754–755. DOI: 10.1093/bioinformatics/17.8.754.

[93] John P. Huelsenbeck et al. "Bayesian Inference of Phylogeny and Its Impact on Evolutionary Biology". In: *Science* 294.5550 (2001), pp. 2310–2314. ISSN: 0036-8075. DOI: 10.1126/science.1065889. eprint: http://science.sciencemag.org/content/294/5550/2310.full.pdf. URL: http://science.sciencemag.org/content/294/5550/2310.

[94] Jaime Huerta-Cepas, François Serra, and Peer Bork. "ETE 3: reconstruction, analysis, and visualization of phylogenomic data". In: *Molecular biology and evolution* 33.6 (2016), pp. 1635–1638.

[95] Takuto Ikuta and Takuya Akiba. "Integer Programming Approach for Directed Minimum Spanning Tree Problem on Temporal Graphs". In: *Proceedings of the 1st ACM SIGMOD Workshop on Network Data Analytics*. NDA '16. San Francisco, California: Association for Computing Machinery, 2016. ISBN: 9781450345132. DOI: 10.1145/2980523.2980528. URL: https://doi.org/10.1145/2980523.2980528.

[96] Ivaylo I Ivanov et al. "Induction of intestinal Th17 cells by segmented filamentous bacteria". en. In: *Cell* 139.3 (Oct. 2009), pp. 485–498.

[97] Marc O Johnson et al. "Distinct regulation of Th17 and Th1 cell differentiation by glutaminase-dependent metabolism". en. In: *Cell* 175.7 (Dec. 2018), 1780–1795.e19.

[98] Matthew G. Jones et al. "Inference of single-cell phylogenies from lineage tracing data using Cassiopeia". In: *Genome Biology* 21.1 (Apr. 2020), p. 92. ISSN: 1474-760X. DOI: 10.1186/s13059-020-02000-8. URL: https://doi.org/10.1186/s13059-020-02000-8.

[99] Matthew G. Jones et al. *Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. All next generation sequencing datasets used in this study: RNA-seq libraries*. URL: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146712.

[100] Marco Jost et al. "Combined CRISPRi/a-Based Chemical Genetic Screens Reveal that Rigosertib Is a Microtubule-Destabilizing Agent". In: *Molecular Cell* 68.1 (2017), 210–223.e6. ISSN: 1097-2765. DOI: https://doi.org/10.1016/j.molcel.2017.09.012. URL: http://www.sciencedirect.com/science/article/pii/S1097276517306603.

[101] Marco Jost et al. "Titrating gene expression with series of systematically compromised CRISPR guide RNAs". In: *bioRxiv* (2019). DOI: 10.1101/717389. eprint: https://www.biorxiv.org/content/early/2019/07/28/717389.full.pdf.

[102] Reza Kalhor, Prashant Mali, and George M. Church. "Rapidly evolving homing CRISPR barcodes". In: *Nature Methods* 14 (Dec. 2016). Article.

[103] Reza Kalhor et al. "Developmental barcoding of whole mouse via homing CRISPR". In: *Science* 361.6405 (2018). ISSN: 0036-8075. DOI: 10.1126/science.aat9804. eprint: http://science.sciencemag.org/content/361/6405/eaat9804.full.pdf. URL: http://science.sciencemag.org/content/361/6405/eaat9804.

[104] Kumaran Kandasamy et al. "NetPath: a public resource of curated signal transduction pathways". In: *Genome Biology* 11.1 (2010), R3. ISSN: 1474-760X. DOI: 10.1186/gb-2010-11-1-r3. URL: https://doi.org/10.1186/gb-2010-11-1-r3.

[105] Evgeny Kanshin et al. "A Cell-Signaling Network Temporally Resolves Specific versus Promiscuous Phosphorylation". In: *Cell Reports* 10.7 (2015), pp. 1202–1214. ISSN: 2211-1247. DOI: http://dx.doi.org/10.1016/j.celrep.2015.01.052.

[106] Hania Kebir et al. "Preferential recruitment of interferon-gamma-expressing TH17 cells in multiple sclerosis". en. In: *Ann. Neurol.* 66.3 (Sept. 2009), pp. 390–402.

[107] T. S. Keshava Prasad et al. "Human Protein Reference Database—2009 update". In: *Nucleic Acids Research* 37.suppl_1 (2009), pp. D767–D772. DOI: 10.1093/nar/gkn892. URL: +%20http://dx.doi.org/10.1093/nar/gkn892.

[108] Lennart Kester and Alexander van Oudenaarden. "Single-Cell Transcriptomics Meets Lineage Tracing". In: *Cell Stem Cell* 23.2 (2018), pp. 166–179. ISSN: 1934-5909. DOI: https://doi.org/10.1016/j.stem.2018.04.014. URL: http://www.sciencedirect.com/science/article/pii/S1934590918301760.

[109] Hui Kwon Kim et al. "Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity". In: *Nature Biotechnology* 36 (Jan. 2018). URL: https://doi.org/10.1038/nbt.4061.

[110] Motoo Kimura. "The Number of Heterozygous Nucleotide Sites Maintained in a Finite Population Due to Steady Flux of Mutations". In: *Genetics* 61.4 (), pp. 893–903.

[111] James Kinchen et al. "Structural remodeling of the human colonic mesenchyme in inflammatory bowel disease". en. In: *Cell* 175.2 (Oct. 2018), 372–386.e17.

[112] T Kinugasa et al. "Claudins regulate the intestinal barrier in response to immune mediators". en. In: *Gastroenterology* 118.6 (June 2000), pp. 1001–1011.

[113] Mikko Kivelä et al. "Multilayer networks". In: *Journal of Complex Networks* 2.3 (July 2014), pp. 203–271. ISSN: 2051-1310. DOI: 10.1093/comnet/cnu016. eprint: http://oup.prod.sis.lan/comnet/article-pdf/2/3/203/9130906/cnu016.pdf. URL: https://dx.doi.org/10.1093/comnet/cnu016.

[114] Sanne E Klompe et al. "Transposon-encoded CRISPR–Cas systems direct RNA-guided DNA integration". In: *Nature* 571.7764 (2019), pp. 219–225.

[115] Bryan Kolaczkowski and Joseph W. Thornton. "Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous". In: *Nature* 431.7011 (2004), pp. 980–984. ISSN: 1476-4687. DOI: 10.1038/nature02917. URL: https://doi.org/10.1038/nature02917.

[116] Alexis C. Komor et al. "Programmable editing of a target base in genomic DNA without double-stranded DNA cleavage". In: *Nature* 533 (Apr. 2016). URL: https://doi.org/10.1038/nbt.4199.

[117] Thomas Korn et al. "IL-17 and Th17 cells". en. In: *Annu. Rev. Immunol.* 27.1 (2009), pp. 485–517.

[118] F. Lemoine et al. "Renewing Felsenstein's phylogenetic bootstrap in the era of big data". In: *Nature* 556.7702 (2018), pp. 452–456. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0043-0. URL: https://doi.org/10.1038/s41586-018-0043-0.

[119] Taibo Li et al. "A scored human protein-protein interaction network to catalyze genomic interpretation". In: *Nature Methods* 14 (2016). URL: https://doi.org/10.1038/nmeth.4083.

[120] Chin Lung Lu, Chuan Yi Tang, and Richard Chia-Tung Lee. "The full Steiner tree problem". In: *Theoretical Computer Science* 306.1 (2003), pp. 55–67. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(03)00209-3. URL: http://www.sciencedirect.com/science/article/pii/S0304397503002093.

[121] Alba Luengo, Dan Y Gui, and Matthew G Vander Heiden. "Targeting metabolism for cancer therapy". en. In: *Cell Chem. Biol.* 24.9 (Sept. 2017), pp. 1161–1180.

[122] Tak W Mak et al. "Glutathione primes T cell metabolism for inflammation". en. In: *Immunity* 46.4 (Apr. 2017), pp. 675–689.

[123] Salwan Maqdasy et al. "Impaired phosphocreatine metabolism in white adipocytes promotes inflammation". In: *Nature Metabolism* 4.2 (Feb. 2022), pp. 190–202. ISSN: 2522-5812. DOI: 10.1038/s42255-022-00525-9. URL: https://doi.org/10.1038/s42255-022-00525-9.

[124] Naoki Masuda and Renaud Lambiotte. *A Guide to Temporal Networks*. Oct. 2016. ISBN: 978-1-78634-114-3. DOI: 10.1142/q0268.

[125] Arnon Mazza et al. "A Minimum-Labeling Approach for Reconstructing Protein Networks across Multiple Conditions". In: *Algorithms for Molecular Biology* 9.1 (2014), p. 1.

[126] Christopher S. McGinnis, Lyndsay M. Murrow, and Zev J. Gartner. "DoubletFinder: Doublet Detection in Single-Cell RNA Sequencing Data Using Artificial Nearest Neighbors". In: *Cell Systems* 8.4 (Apr. 2019), 329–337.e4. ISSN: 2405-4712. DOI: 10.1016/j.cels.2019.03.003. URL: https://doi.org/10.1016/j.cels.2019.03.003.

[127] Aaron McKenna and James A. Gagnon. "Recording development with single cell dynamic lineage tracing". In: *Development* 146.12 (2019). ISSN: 0950-1991. DOI: 10.1242/dev.169730.

[128] Aaron McKenna and James A. Gagnon. "Recording development with single cell dynamic lineage tracing". In: *Development* 146.12 (2019). ISSN: 0950-1991. DOI: 10.1242/dev.169730. eprint: https://dev.biologists.org/content/146/12/dev169730.full.pdf. URL: https://dev.biologists.org/content/146/12/dev169730.

[129] Aaron McKenna et al. "Whole organism lineage tracing by combinatorial and cumulative genome editing". In: *Science* (2016). ISSN: 0036-8075. DOI: 10.1126/science.aaf7907.

[130] Aaron McKenna et al. "Whole-organism lineage tracing by combinatorial and cumulative genome editing". In: *Science* 353.6298 (2016). ISSN: 0036-8075. DOI: 10.1126/science.aaf7907. eprint: https://science.sciencemag.org/content/353/6298/aaf7907.full.pdf. URL: https://science.sciencemag.org/content/353/6298/aaf7907.

[131] Philipp Mertins et al. "An Integrative Framework Reveals Signaling-to-Transcription Events in Toll-like Receptor Signaling". In: *Cell Reports* 19.13 (), pp. 2853–2866. ISSN: 2211-1247. DOI: 10.1016/j.celrep.2017.06.016. URL: http://dx.doi.org/10.1016/j.celrep.2017.06.016.

[132] Ryan D Michalek et al. "Cutting edge: distinct glycolytic and lipid oxidative metabolic programs are essential for effector and regulatory CD4+ T cell subsets". en. In: *J. Immunol.* 186.6 (Mar. 2011), pp. 3299–3303.

[133] Chloé Michaudel and Harry Sokol. "The gut Microbiota at the service of immunometabolism". en. In: *Cell Metab.* 32.4 (Oct. 2020), pp. 514–523.

[134] R Mihaescu, D Levy, and L Pachter. "Why Neighbor-Joining Works". In: *arXiv* (2006). DOI: arXiv:cs/0602041v3.

[135] Radu Mihaescu, C. Hill, and S. Rao. "Fast Phylogeny Reconstruction Through Learning of Ancestral Sequences". In: *Algorithmica* 66 (2012), pp. 419–449.

[136] Radu Mihaescu, Dan Levy, and Lior Pachter. "Why neighbor-joining works". In: *CoRR* abs/cs/0602041 (2006). arXiv: cs/0602041. URL: http://arxiv.org/abs/cs/0602041.

[137] E. Mossel. "Distorted Metrics on Trees and Phylogenetic Forests". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4.1 (2007), pp. 108–116. DOI: 10.1109/TCBB.2007.1010.

[138] Vincent Moulton, Andreas Spillner, and Taoyang Wu. "UPGMA and the normalized equidistant minimum evolution problem". In: *Theoretical Computer Science* 721 (Apr. 2017). DOI: 10.1016/j.tcs.2018.01.022.

[139] Vincenzo Nicosia et al. "Components in time-varying graphs". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.2 (2012), p. 023101. DOI: 10.1063/1.3697996. eprint: https://doi.org/10.1063/1.3697996. URL: https://doi.org/10.1063/1.3697996.

[140] Artem S Novozhilov, Georgy P Karev, and Eugene V Koonin. "Biological applications of the theory of birth-and-death processes". In: *Briefings in bioinformatics* 7.1 (2006), pp. 70–85.

[141] Luke A J O'Neill, Rigel J Kishton, and Jeff Rathmell. "A guide to immunometabolism for immunologists". en. In: *Nat. Rev. Immunol.* 16.9 (Sept. 2016), pp. 553–565.

[142] Amanda Poissonnier et al. "CD95-mediated calcium signaling promotes T helper 17 trafficking to inflamed organs in lupus-prone mice". en. In: *Immunity* 45.1 (July 2016), pp. 209–223.

[143] Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin. "FastTree: Computing Large Minimum Evolution Trees with Profiles instead of a Distance Matrix". In: *Molecular Biology and Evolution* 26.7 (2009), pp. 1641–1650. DOI: 10.1093/molbev/msp077.

[144] Teresa M. Przytycka, Mona Singh, and Donna K. Slonim. "Toward the dynamic interactome: it's about time". In: *Briefings in Bioinformatics* 11.1 (2010), pp. 15–29. DOI: 10.1093/bib/bbp057. URL: http://dx.doi.org/10.1093/bib/bbp057.

[145] Jeffrey J Quinn et al. "Single-cell lineages reveal the rates, routes, and drivers of metastasis in cancer xenografts". In: *Science* 371.6532 (2021).

[146] Bushra Raj, James A Gagnon, and Alexander F Schier. "Large-scale reconstruction of cell lineages using single-cell readout of transcriptomes and CRISPR–Cas9 barcodes by scGESTALT". In: *Nature protocols* 13.11 (2018), pp. 2685–2713.

[147] Bushra Raj et al. "Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain". In: *Nature Biotechnology* 36 (Mar. 2018).

[148] Rossana Rauti et al. "Effect of SARS-CoV-2 proteins on vascular permeability". In: *eLife* 10 (Oct. 2021). Ed. by Arduino A Mangoni et al., e69314. ISSN: 2050-084X. DOI: 10.7554/eLife.69314. URL: https://doi.org/10.7554/eLife.69314.

[149] Ran Raz. "A Parallel Repetition Theorem". In: *SIAM Journal on Computing* 27.3 (1998), pp. 763–803.

[150] Miguel Reina-Campos, Jorge Moscat, and Maria Diaz-Meco. "Metabolism shapes the tumor microenvironment". en. In: *Curr. Opin. Cell Biol.* 48 (Oct. 2017), pp. 47–53.

[151] D.F. Robinson and L.R. Foulds. "Comparison of phylogenetic trees". In: *Mathematical Biosciences* 53.1 (1981), pp. 131–147. ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(81)90043-2. URL: http://www.sciencedirect.com/science/article/pii/0025556481900432.

[152] Polina Rozenshtein et al. "Reconstructing an Epidemic Over Time". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1835–1844. ISBN: 9781450342322. DOI: 10.1145/2939672.2939865. URL: https://doi.org/10.1145/2939672.2939865.

[153] N Saitou and M Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular Biology and Evolution* 4.4 (1987), pp. 406–425. DOI: 10.1093/oxfordjournals.molbev.a040454.

[154] N. Saitou and M. Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular biology and evolution* 4 4 (1987), pp. 406–25.

[155] Irepan Salvador-Martınez et al. "Is it possible to reconstruct an accurate cell lineage using CRISPR recorders?" In: *Elife* 8 (2019), e40292.

[156] William J Sandborn et al. "Ustekinumab induction and maintenance therapy in refractory Crohn's disease". en. In: *N. Engl. J. Med.* 367.16 (Oct. 2012), pp. 1519–1528.

[157] Bruce E Sands, Stefan Schreiber, and Richard A Lirio. "Vedolizumab versus adalimumab for moderate-to-severe ulcerative colitis. Reply". en. In: *N. Engl. J. Med.* 382.1 (Jan. 2020), pp. 93–94.

[158] Geoffrey Schiebinger et al. "Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming". In: *Cell* 176.4 (2019), 928–943.e22. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2019.01.006. URL: https://www.sciencedirect.com/science/article/pii/S009286741930039X.

[159] Alexandra Schnell et al. "Stem-like intestinal Th17 cells give rise to pathogenic effector T cells during autoimmunity". en. In: *Cell* 184.26 (Dec. 2021), 6281–6298.e23.

[160] Michelle S Scott et al. "Identifying Regulatory Subnetworks for a Set of Genes". In: *Molecular & Cellular Proteomics* 4.5 (2005), pp. 683–692.

[161] Gur Sevillya, Zeev Frenkel, and Sagi Snir. "Triplet MaxCut: a new toolkit for rooted supertree". In: *Methods in Ecology and Evolution* 7.11 (2016), pp. 1359–1365. DOI: 10.1111/2041-210X.12606. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.12606. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.12606.

[162] Lewis Z Shi et al. "HIF1alpha-dependent glycolytic pathway orchestrates a metabolic checkpoint for the differentiation of TH17 and Treg cells". en. In: *J. Exp. Med.* 208.7 (July 2011), pp. 1367–1376.

[163] L. F. Signorini, T. Almozlino, and R. Sharan. "ANAT 3.0: a framework for elucidating functional protein subnetworks using graph-theoretic and machine learning approaches". In: *BMC Bioinformatics* 22.1 (Oct. 2021), p. 526. ISSN: 1471-2105. DOI: 10.1186/s12859-021-04449-1. URL: https://doi.org/10.1186/s12859-021-04449-1.

[164] Robert R Sokal. "A statistical method for evaluating systematic relationships." In: *Univ. Kansas, Sci. Bull.* 38 (1958), pp. 1409–1438.

[165] Bastiaan Spanjaard et al. "Simultaneous lineage tracing and cell-type identification using CRISPR-Cas9-induced genetic scars". In: *Nature Biotechnology* 36 (Apr. 2018). URL: http://dx.doi.org/10.1038/nbt.4124.

[166] Michael Steel. "The complexity of reconstructing trees from qualitative characters and subtrees". In: *Journal of Classification* 9.1 (Jan. 1992), pp. 91–116. ISSN: 1432-1343. DOI: 10.1007/BF02618470. URL: https://doi.org/10.1007/BF02618470.

[167] Marlon Stoeckius et al. "Cell Hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics". In: *Genome Biology* 19.1 (2018), p. 224. ISSN: 1474-760X. DOI: 10.1186/s13059-018-1603-1. URL: https://doi.org/10.1186/s13059-018-1603-1.

[168] Evgeny B Stukalin et al. "Age-dependent stochastic models for understanding population fluctuations in continuously cultured cells". In: *Journal of the royal society interface* 10.85 (2013), p. 20130325.

[169] Ken Sugino and Tzumin Lee. "Robust Reconstruction of CRISPR and Tumor Lineage Using Depth Metrics". In: *bioRxiv* (2019). DOI: 10.1101/609107. eprint: https://www.biorxiv.org/content/early/2019/04/15/609107.full.pdf. URL: https://www.biorxiv.org/content/early/2019/04/15/609107.

[170] J.E. Sulston et al. "The embryonic cell lineage of the nematode Caenorhabditis elegans". In: *Developmental Biology* 100.1 (1983), pp. 64–119. ISSN: 0012-1606. DOI: https://doi.org/10.1016/0012-1606(83)90201-4. URL: http://www.sciencedirect.com/science/article/pii/0012160683902014.

[171] Fumio Tajima. "Infinite-allele model and infinite-site model in population genetics". In: *Journal of Genetics* 75.1 (Apr. 1996), p. 27. DOI: 10.1007/BF02931749. URL: https://doi.org/10.1007/BF02931749.

[172] Tatyana N Tarasenko, Julio Gomez-Rodriguez, and Peter J McGuire. "Impaired T cell function in argininosuccinate synthetase deficiency". en. In: *J. Leukoc. Biol.* 97.2 (Feb. 2015), pp. 273–278.

[173] Jeffrey P. Townsend. "Profiling Phylogenetic Informativeness". In: *Systematic Biology* 56.2 (2007), pp. 222–231. DOI: 10.1080/10635150701311362.

[174] Nurcan Tuncbag et al. "Network-Based Interpretation of Diverse High-Throughput Datasets through the Omics Integrator Software Package". In: *PLOS Computational Biology* 12.4 (Apr. 2016), pp. 1–18. DOI: 10.1371/journal.pcbi.1004879. URL: https://doi.org/10.1371/journal.pcbi.1004879.

[175] Emre Turer et al. "Creatine maintains intestinal homeostasis and protects against colitis". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 114.7 (Feb. 2017), E1273–E1281.

[176] José P Vaqué et al. "PLCG1 mutations in cutaneous T-cell lymphomas". en. In: *Blood* 123.13 (Mar. 2014), pp. 2034–2043.

[177] Michel Verleysen and Damien François. "The Curse of Dimensionality in Data Mining and Time Series Prediction". In: (2005). Ed. by Joan Cabestany, Alberto Prieto, and Francisco Sandoval, pp. 758–770.

[178] Allon Wagner et al. "Metabolic modeling of single Th17 cells reveals regulators of autoimmunity". en. In: *Cell* 184.16 (Aug. 2021), 4168–4185.e21.

[179] Daniel E Wagner and Allon M Klein. "Lineage tracing meets single-cell omics: opportunities and challenges". In: *Nature Reviews Genetics* 21.7 (2020), pp. 410–427.

[180] Daniel E. Wagner et al. "Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo". In: *Science* 360.6392 (2018), pp. 981–987. ISSN: 0036-8075. DOI: 10.1126/science.aar4362. eprint: http://science.sciencemag.org/content/360/6392/981.full.pdf. URL: http://science.sciencemag.org/content/360/6392/981.

[181] Ruoning Wang et al. "The transcription factor Myc controls metabolic reprogramming upon T lymphocyte activation". en. In: *Immunity* 35.6 (Dec. 2011), pp. 871–882.

[182] Sibo Wang et al. "Efficient Route Planning on Public Transportation Networks: A Labelling Approach". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 967–982. ISBN: 9781450327589. DOI: 10.1145/2723372.2749456. URL: https://doi.org/10.1145/2723372.2749456.

[183] Casey T Weaver et al. "The Th17 pathway and inflammatory diseases of the intestines, lungs, and skin". en. In: *Annu. Rev. Pathol.* 8 (Jan. 2013), pp. 477–512.

[184] J. F. Weng, I. Mareels, and D. A. Thomas. "Probability Steiner trees and maximum parsimony in phylogenetic analysis". In: *Journal of Mathematical Biology* 64.7 (June 2012), pp. 1225–1251. ISSN: 1432-1416. DOI: 10.1007/s00285-011-0442-4. URL: https://doi.org/10.1007/s00285-011-0442-4.

[185] Samuel L. Wolock, Romain Lopez, and Allon M. Klein. "Scrublet: Computational Identification of Cell Doublets in Single-Cell Transcriptomic Data". In: *Cell Systems* 8.4 (2019), 281–291.e9. ISSN: 2405-4712. DOI: https://doi.org/10.1016/j.cels.2018.11.005. URL: http://www.sciencedirect.com/science/article/pii/S2405471218304745.

[186] Chuan Wu et al. "Induction of pathogenic TH17 cells by inducible salt-sensing kinase SGK1". In: *Nature* 496.7446 (Apr. 2013), pp. 513–517.

[187] Huanhuan Wu et al. "Path Problems in Temporal Graphs". In: *Proc. VLDB Endow.* 7.9 (May 2014), pp. 721–732. ISSN: 2150-8097. DOI: 10.14778/2732939.2732945. URL: https://doi.org/10.14778/2732939.2732945.

[188] Huanhuan Wu et al. "Reachability and time-based path queries in temporal graphs". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. 2016, pp. 145–156. DOI: 10.1109/ICDE.2016.7498236.

[189] Jimmy Wu et al. "Connectivity problems on heterogeneous graphs". In: *Algorithms for Molecular Biology* 14.1 (Mar. 2019), p. 5. ISSN: 1748-7188. DOI: 10.1186/s13015-019-0141-z. URL: https://doi.org/10.1186/s13015-019-0141-z.

[190] Zhenxiang Xi, Liang Liu, and Charles C. Davis. "The Impact of Missing Data on Species Tree Estimation". In: *Molecular Biology and Evolution* 33.3 (Nov. 2015), pp. 838–860. ISSN: 0737-4038. DOI: 10.1093/molbev/msv266. eprint: http://oup.prod.sis.lan/mbe/article-pdf/33/3/838/17471628/msv266.pdf. URL: https://doi.org/10.1093/molbev/msv266.

[191] Tao Xu et al. "Metabolic control of TH17 and induced Treg cell balance by an epigenetic mechanism". en. In: *Nature* 548.7666 (Aug. 2017), pp. 228–233.

[192] Yaolin Xu et al. "Metabolic reprogramming in the tumor microenvironment with immunocytes and immune checkpoints". en. In: *Front. Oncol.* 11 (Nov. 2021), p. 759015.

[193] Dian Yang et al. "Lineage tracing reveals the phylodynamics, plasticity, and paths of tumor evolution". In: *Cell* 185.11 (May 2022), 1905–1923.e25. ISSN: 0092-8674. DOI: 10.1016/j.cell.2022.04.015. URL: https://doi.org/10.1016/j.cell.2022.04.015.

[194] Hui Yang et al. "Base editing generates substantial off-target single nucleotide variants". In: *bioRxiv* (2018). DOI: 10.1101/480145. eprint: https://www.biorxiv.org/content/early/2018/11/27/480145.full.pdf. URL: https://www.biorxiv.org/content/early/2018/11/27/480145.

[195] Jialong Yang et al. "DGK α and ζ activities control TH1 and TH17 cell differentiation". en. In: *Front. Immunol.* 10 (2019), p. 3048.

[196] Ziheng Yang and Bruce Rannala. "Molecular phylogenetics: principles and practice". In: *Nature Reviews Genetics* 13 (Mar. 2012). Review Article. URL: https://doi.org/10.1038/nrg3186.

[197] P Yaqoob and P C Calder. "Glutamine requirement of proliferating T lymphocytes". en. In: *Nutrition* 13.7-8 (July 1997), pp. 646–651.

[198] Nir Yosef et al. "Toward accurate reconstruction of functional protein networks". In: *Molecular systems biology* 5.1 (2009), p. 248.

[199] Hamim Zafar, Chieh Lin, and Ziv Bar-Joseph. "Single-cell lineage tracing by integrating CRISPR-Cas9 mutations with transcriptomic data". In: *Nature Communications* 11.1 (June 2020), p. 3055. ISSN: 2041-1723. DOI: 10.1038/s41467-020-16821-5. URL: https://doi.org/10.1038/s41467-020-16821-5.

[200] Grace X. Y. Zheng et al. "Massively parallel digital transcriptional profiling of single cells". In: *Nature Communications* 8.1 (Jan. 2017), p. 14049. ISSN: 2041-1723. DOI: 10.1038/ncomms14049. URL: https://doi.org/10.1038/ncomms14049.

[201] Leonid Zosin and Samir Khuller. "On Directed Steiner Trees". In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '02. San Francisco, California: Society for Industrial and Applied Mathematics, 2002, pp. 59–63. ISBN: 0-89871-513-X. URL: http://dl.acm.org/citation.cfm?id=545381.545388.