Bioinformatics, 36, 2020, i169–i176
doi: 10.1093/bioinformatics/btaa464
ISMB 2020

OXFORD

PhISCS-BnB: a fast branch and bound algorithm for the perfect tumor phylogeny reconstruction problem

Erfan Sadeqi Azer[1,†], Farid Rashidi Mehrabadi[1,2,†], Salem Malikić[1], Xuan Cindy Li[2,3], Osnat Bartok[4], Kevin Litchfield[5,6], Ronen Levy[4], Yardena Samuels[4], Alejandro A. Schäffer[2], E. Michael Gertz[2], Chi-Ping Day[7], Eva Pérez-Guijarro[7], Kerrie Marie[7], Maxwell P. Lee[7], Glenn Merlino[7], Funda Ergun[1] and S. Cenk Sahinalp[2,*]

[1]Department of Computer Science, Indiana University, Bloomington, IN 47408, USA, [2]Cancer Data Science Laboratory, Center for Cancer Research, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA, [3]Program in Computational Biology, Bioinformatics and Genomics, University of Maryland, College Park, MD 20742, USA, [4]Department of Molecular Cell Biology, Weizmann Institute of Science, Rehovot, Israel, [5]Cancer Evolution and Genome Instability Laboratory, Francis Crick Institute, London NW1 1AT, UK, [6]Cancer Research UK Lung Cancer Centre of Excellence London, University College London Cancer Institute, London WC1E 6DD, UK and [7]Laboratory of Cancer Biology and Genetics, Center for Cancer Research, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Abstract

Motivation: Recent advances in single-cell sequencing (SCS) offer an unprecedented insight into tumor emergence and evolution. Principled approaches to tumor phylogeny reconstruction via SCS data are typically based on general computational methods for solving an integer linear program, or a constraint satisfaction program, which, although guaranteeing convergence to the most likely solution, are very slow. Others based on Monte Carlo Markov Chain or alternative heuristics not only offer no such guarantee, but also are not faster in practice. As a result, novel methods that can scale up to handle the size and noise characteristics of emerging SCS data are highly desirable to fully utilize this technology.

Results: We introduce PhISCS-BnB (phylogeny inference using SCS via branch and bound), a branch and bound algorithm to compute the most likely perfect phylogeny on an input genotype matrix extracted from an SCS dataset. PhISCS-BnB not only offers an optimality guarantee, but is also 10–100 times faster than the best available methods on simulated tumor SCS data. We also applied PhISCS-BnB on a recently published large melanoma dataset derived from the sublineages of a cell line involving 20 clones with 2367 mutations, which returned the optimal tumor phylogeny in <4 h. The resulting phylogeny agrees with and extends the published results by providing a more detailed picture on the clonal evolution of the tumor.

Availability and implementation: https://github.com/algo-cancer/PhISCS-BnB.

Contact: cenk.sahinalp@nih.gov

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Cancer is a highly dynamic, evolutionary disease. Constantly shaped by mutation and selection, cancer progression often results in the emergence of distinct tumor cell populations with varying sets of somatic mutations, commonly known as (sub)clones. The diverse pool of subclones may harbor treatment-resistant mutations. When favorably selected for in the tumor environment by treatment exposure, treatment-resistant subclones may gain dominance over others and eventually contribute to treatment failure (Alizadeh *et al.*, 2015). The challenges in developing effective cancer therapies under the heterogeneous tumor landscape thus motivate the following question: can we reconstruct the tumor phylogeny and unravel spatial and temporal intra-tumor heterogeneity (ITH) to enlighten cancer treatment strategies?

In recent years, several computational tools for analyzing ITH and evolution from bulk sequencing data of tumor samples have been developed (Deshwar *et al.*, 2015; Donmez *et al.*, 2017; El-Kebir *et al.*, 2015, 2016; Hajirasouliha *et al.*, 2014; Jiao *et al.*, 2014; Malikic *et al.*, 2015; Marass *et al.*, 2016; Popic *et al.*, 2015; Satas *et al.*, 2017; Strino *et al.*, 2013). However, bulk sequencing data provides only an aggregate signal over large number of cells and, due to its limited resolution, all of these methods have several limitations in unambiguously inferring trees of tumor evolution. Most

Published by Oxford University Press 2020. This work is written by US Government employees and is in the public domain in the US.

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

notably, they typically rely on clustering of mutations of similar cellular prevalence. Consequently, if two sets of mutations evolving on different branches of phylogenetic tree have similar cellular prevalence values, they get clustered together. Furthermore, even in cases where the cellular prevalence values of clusters are different, methods based on bulk sequencing data are frequently unable to distinguish between multiple trees that describe the observed data equally well (Kuipers *et al.*, 2017; Malikic *et al.*, 2019a).

The rise of single-cell sequencing (SCS) has enabled exploration of ITH at a higher, cellular resolution. Unfortunately, even SCS cannot trivially provide a comprehensive understanding of ITH. Among the lingering caveats with SCS, the most prominent is the prevailing presence of sequencing noise (Zafar *et al.*, 2018). We are particularly interested in three types of noise in SCS datasets: (i) false positive mutation calls, potentially from sources like read errors, (ii) false negative mutation calls, potentially from sources like variance in sequence coverage or allele dropout and (iii) missing values for mutations from sites affected by DNA amplification failure. [A final noise source is the *doublets*, the technical artifacts of two (or rarely more) cells with heterogeneous mutation profiles treated and sequenced as a single cell. Since there are a number of preprocessing techniques such as (Roth *et al.*, 2016) to detect and eliminate doublets fairly well we will not focus on doublets as a source of noise in this paper.] The multi-faceted and high levels of sequencing noise have prompted the development of novel computational approaches that need to infer a tumor evolutionary model while compensating for all three sources of noise.

The first principled approaches for studying ITH by the use of SCS data were all based on probabilistic formulations that aim to infer the *most likely perfect phylogeny* (PP) of a tumor. SCITE (Jahn *et al.*, 2016) and OncoNEM (Ross *et al.*, 2016) are among these methods that primarily aim to build a PP (i.e. an evolutionary tree where no mutation can appear more than once and is never lost).[Another tool, SiFit (Zafar *et al.*, 2017) aims to allow for deletion events and loss of heterozygosity.] Following up on this, SPhyR (El-Kebir, 2018) formulates the tumor phylogeny reconstruction problem as an integer linear program (ILP) under the constraints imposed by the k-Dollo parsimony model where a gained mutation can only be lost k times. SCIΦ (Singer *et al.*, 2018) simultaneously performs mutation calling and the tumor phylogeny inference taking read counts data as the input, rather than the more commonly used genotype matrix with inferred mutations, represented by columns, in distinct cells, represented in by rows.

As datasets with matching SCS and bulk sequencing data become publicly available, methods to infer tumor phylogeny through integrative use of these two data types are becoming available. B-SCITE (Malikic *et al.*, 2019a) e.g. combines CITUP, which is designed for bulk sequencing data, with SCITE through a Monte Carlo Markov Chain (MCMC) strategy. More recently PhISCS (phylogeny inference using SCS) (Malikic *et al.*, 2019b), offers the option of formulating integrative reconstruction of the most likely tumor phylogeny either as an ILP or as a Boolean constraint satisfaction program (CSP), while allowing for a fixed number of PP violating mutations. The CSP version of PhISCS, when employing state-of-the-art CSP (more specifically weighted max-SAT) solvers such as RC2 (Ignatiev *et al.*, 2019) and Open-WBO (Martins *et al.*, 2014) turn out to be the fastest among all available techniques even when only SCS data are available. Nevertheless, none of the available techniques can scale up to handle emerging datasets that involve thousands of cells (Laks *et al.*, 2019); even moderate size SCS data involving a few hundred mutations and cells turn out to be problematic especially with the current 'standard' false negative rate of 15–20%. Other recent techniques such as scVILP (Edrisi *et al.*, 2019) and SiCloneFit (Zafar *et al.*, 2019) focus on adding new features to (or relax constraints for) the tumor phylogeny reconstruction problem and are (typically) not faster. [One exception is ScisTree (Wu, 2019) which is reported to be faster but is a heuristic approach with no optimality guarantees.] Finally, even though new sequencing techniques such as single-cell cloning (SCC, i.e. bulk sequencing of homogeneous cell populations, each derived from a single cell) offer much lower false negative rates, the scale of the data they produce—involving thousands of mutations, require much faster solutions to the tumor phylogeny reconstruction problem.

In this paper, we present a branch and bound (BnB) algorithm and its implementation (called PhISCS-BnB, phylogeny inference using single-cell sequencing via branch and bound) to optimally reconstruct a tumor phylogeny very efficiently. Generally speaking, our BnB approach clusters entries of the input genotype matrix and processes them together, enabling faster execution (Section 3.1 and Lemma 3.1). We introduce a number of *bounding* algorithms, some faster but offering limited pruning and others slower but with better pruning efficiency. Among them, a novel bounding algorithm with a 2-SAT formulation is a key technical contribution of our paper: through its use, PhISCS-BnB improves the running time of the fastest available methods for tumor phylogeny reconstruction by a factor of up to 100 (Section 4).

2 PP reconstruction problem

Given a binary genotype matrix I, we would like to reconstruct the most likely phylogeny by discovering how to flip the smallest number of entries of I so it can provide a PP.

Preliminaries. Our input is a binary (genotype) matrix $I \in \{0,\ 1\}^{n \times m}$. The n rows represent genotypes of single cells observed in an SCS experiment and the m columns represent a set of considered somatic mutations. (Germ line is typically available.) $I(i,j) = 1$ indicates that mutation j is present in cell i; $I(i,j) = 0$ indicates that it is not.

The *three-gametes rule* stipulates that a binary matrix $X \in \{0,\ 1\}^{n \times m}$ should not have three rows and two columns (in any order) with the corresponding six entries displaying the configuration (1, 0), (0, 1) and (1, 1). If the forbidden configuration is present, we say that there is a *violation*, referenced by the three rows and the two columns containing it. It was shown in Gusfield (1991) that satisfaction of the three-gametes rule by I is necessary and sufficient for the existence of a PP corresponding to I.

Given input matrix I, we call a binary matrix X a *descendant* of I if all entries of X are identical to those of I except some that have been *flipped* from 0 to 1. For a matrix X, $F_{0\to1}(I, X)$ is defined as the number of entries that are 0 in I and 1 in X. We sometimes refer to this value as the number of flips to get to X from I.

Our problem. Given a genotype matrix I, we would like to obtain a minimum-cardinality set of *bit flips* (from 0 to 1, indicating a correction for a false negative—since false positive rates are several orders of magnitude lower) (In general both false negative and false positives (respectively, 1 read as 0 and 0 read as 1) happen with distinct probabilities. The qualitative difference in these probabilities is due to the sequencing technology in use and thresholding rules employed in establishing I. As is well known, the false positive rate is three orders of magnitude or more lower than the false negative rate. In fact, in emerging data, e.g. from SCC experiments, the false positive rate approaches zero and thus can be ignored. As a result, we focus only on false negatives and our proposed algorithm and its subroutines make use of this assumption.) that removes all three-gametes rule violations in I and thus transforms I into a matrix Y that provides a PP.

3 BnB method

In order to discover the smallest number of 0 to 1 flips that will remove all violations in input matrix I, we use a BnB technique. In what follows, we give an overview of the building blocks of our BnB approach. Then we put all of them together in Algorithm PhISCS-BnB.

Our BnB algorithm forms a search tree where each node contains a matrix, with input matrix I at the root—for simplicity we might refer to a node with its label as well as its matrix. In this tree, a matrix X at node v is a *descendant* (as described in the preliminaries) of the matrix Y at v's parent node; all matrices in the tree are thus descendants of I. The tree terminates in leaf nodes that are PP; non-

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

PP nodes will have two child nodes as the tree grows unless they have been pruned due to detected non-optimality.

When a node v with matrix X is formed, v is assigned a *priority score* equal to the number of bit flips needed to get from I to X plus a lower bound on the number of flips necessary to remove all the violations in X. All nodes are kept in a priority queue and are explored in ascending order of their priority scores, unless they have been removed from consideration (pruned) by the bounding mechanism. When the whole tree has been explored or pruned, one of the PP nodes with the smallest number of flips away from I yields the answer.

For matrix X, we let $R^X_{a,b}(p,q)$ denote the set of rows with a in column p and b in column q, i.e. $R^X_{a,b}(p,q) = \{i \mid X(i,p) = a \wedge X(i,q) = b\}$. We drop the superscript, when the matrix X is clear from the context, and only write $R_{a,b}(p,q)$.

3.1 Branching

Let X be the matrix at the node being explored. If X has no violation, it is considered a leaf. Otherwise, let (p, q) be the pair of columns for one particular violation that was found (If there are multiple pairs of columns involved in violations we impose an ordering on them and pick a pair according to this order; thus, we are always considering a single column pair.), i.e. $|R_{a,b}(p,q)| > 0$ for all $(a,b) \in \{(0,1),(1,0),(1,1)\}$. We have two options for fixing the violation. As a violation involving columns p, q contains both a (1, 0) and a (0, 1) in different rows, we have the option of converting either one to a (1, 1) to remove the violation. To reflect this, we construct two child nodes from the current node, one for each option. As an added optimization, once we decide to fix a (0, 1) [resp. (1, 0)] on columns p, q, we fix all (0, 1) [resp. (1, 0)] on these two columns, by changing them to (1, 1). In particular, in the left child, all entries whose row is in $R_{0,1}(p,q)$ and whose column is p are flipped from 0 to 1. Similarly, in the right child entries whose row is in $R_{1,0}(p,q)$ and whose column is q are flipped.

In some cases, the above branching rule, which can flip multiple 0 s at a time, shrinks the height of the search tree compared to the algorithm in Chen *et al.* (2006) and Cai (1996), which flips a single 0 in a child node. The following lemma formally expresses why we flip several entries in a column together at Lines 8 and 11 of the pseudocode: if a (0, 1) in a violation involving columns p, q of matrix X is a (1, 1) in a PP descendant X' of X, all other (0, 1) on (p, q) are (1, 1) as well. An analogous statement holds for (1, 0).

Lemma 3.1 *For any $X \in \{0,\ 1\}^{n\times m}$ with a violation involving columns p, q, let X' be any PP descendant of X. Then, at least one of the following holds:*

- $\forall r_1 \in R_{0,1}(p,q), X'(r_1,p) = 1$, *or*
- $\forall r_2 \in R_{1,0}(p,q), X'(r_2,q) = 1$.

Proof. Assume that the lemma is false; i.e. there is an X' such that $\exists r_1 \in R_{0,1}(p,q),\ X'(r_1,p) = 0 \wedge \exists r_2 \in R_{1,0}(p,q),\ X'(r_2,q) = 0$.

Since (p, q) corresponds to a violation and all the 1 entries in X have to remain 1 in X', there should be a row r_3 such that $X'(r_3,p) = X'(r_3,q) = 1$. This implies that the pair of columns (p, q) and the triplet of rows (r_1, r_2, r_3) corresponds to a violation. This contradicts the assumption that X' is PP. □

3.2 Bounding mechanisms

A *bounding* algorithm is a method that computes a lower bound for the number of flips needed to transform matrix X at a node v to a PP matrix. It thus helps the BnB algorithm to prune the nodes that are provably worse than the currently maintained best node, i.e. the variable Best_Node in Algorithm PhISCS-BnB.

We reuse the calculated lower bound as the estimate of how many flips a matrix X will require to transform into a PP matrix, and then add the number of flips needed to transfer I to X, in order to set the priority score of the node containing X. Recall that all the introduced nodes are pushed to a priority queue, and in each iteration the node with the lowest priority score is chosen to be explored. So, the node X we pick will represent the lowest number of total flips from I to a PP node going through X.

Algorithm 1 PhISCS-BnB

Input: $I \in \{0,\ 1\}^{n\times m}$: Original input to the algorithm
Output: $Y \in \{0,\ 1\}^{n\times m}$ such that
$Y = \mathsf{argmin}_{PP_{(y)}} F_{0\to1}(I, Y)$.
1: $\mathsf{Best_{Node}} \leftarrow$ A simple PP solution (see Section 3.3)
2: $Q \leftarrow$ An empty priority queue
3: Push I in Q
4: **while** Q is not empty **do**
5: $C \leftarrow$ pop next node from Q with the lowest priority score
6: $\mathsf{New_{node_1}} \leftarrow C$
7: **for** $r \in R_{0,1}(p,q)$ **do** ▷ See Lemma 3.1
8: $\mathsf{New_{node_1}}(r,p) \leftarrow 1$ ▷ $F_{0\to1}(I, \mathsf{New_{node_1}})$ is also updated here
9: $\mathsf{New_{node_2}} \leftarrow C$
10: **for** $r \in R_{1,0}(p,q)$ **do**
11: $\mathsf{New_{node_2}}(r,q) \leftarrow 1$
12: **for** $i \in \{1,2\}$ **do**
13: **if** $\mathsf{New_{node_i}}$ is a PP **then** ▷ i.e. $\mathsf{New_{node_i}}$ is a leaf
14: **if** $F_{0\to1}(I, \mathsf{New_{node}}i) < F_{0\to1}(I, \mathsf{Best_{Node}})$ **then**
15: $\mathsf{Best_{Node}} \leftarrow \mathsf{New_{node_i}}$
16: **else** ▷ Use any bounding algorithm proposed in Section 3.2
17: $\mathsf{lb} \leftarrow$ A lower bound for the number of flips to get to a PP matrix from $\mathsf{New_{node}}i$
18: **if** $\mathsf{lb} + F_{0\to1}(I, \mathsf{New_{node}}i) < F_{0\to1}(I, \mathsf{Best_{Node}})$ **then**
19: Push $\mathsf{New_{node}}i$ in Q with priority score set to $\mathsf{lb} + F_{0\to1}(I, \mathsf{New_{node}}i)$
20: Return $\mathsf{Best_{Node}}$

One observation that leads to a lower bound is as follows. Consider a pair of columns that have at least one row with pattern (1, 1), then the number of flips, involving columns p and q, is at least $\min(|R_{0,1}(p,q)|, |R_{1,0}(p,q)|)$. Therefore, for an arbitrary partitioning of the set of columns to pairs, we can aggregate these bounds to achieve a lower bound for the whole matrix.

In the above, one would expect the choice of the partition to have an impact on the quality of the lower bound estimate. To explore this, in what follows, we present three bounding algorithms that progressively add more sophistication to the above idea. These three bounding algorithms offer a tradeoff between the per-node running time and the accuracy of the bound. For some inputs, the fast (and possibly not-so accurate) bounding results in a faster execution, but for other inputs a different tradeoff is better. It is worth mentioning that for biologically plausible inputs, our experiments show that the higher accuracy of bounding is much more important to total time than the per-node running time. We present first two bounding algorithms in Section 3.2.1. The third and the most sophisticated one is presented in Section 3.2.2 and is used in our experiments to compare with previous tools in the literature.

3.2.1 Random partition versus maximum weighted matching

As our first bounding method, we partition the columns of the matrix into pairs uniformly at random. The technique is simple, but more sophisticated techniques might give tighter bounds.

As our second method, we describe a method based on maximum weighted matching (MWM). Construct a weighted undirected graph $G = (V, E, w)$ where the vertices are the columns of I, each column representing a mutation: $V = \{c_1, \ldots c_m\}$ (c_i corresponds to

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

i-th mutation) and the edges are column pairs that display a (1, 1): $E = \{\{c_i, c_j\} \,|\, |R_{1,1}(c_i, c_j)| > 0\}$. In the following, we calculate a weight corresponding to an edge $e = \{c_i, c_j\} \in E$, to be a lower bound on the number of entries in columns c_i and c_j that have to be flipped to make I a PP matrix. Formally, for each edge $e = \{c_i, c_j\} \in E$, $w(e) = \min(|R_{0,1}(c_i, c_j)|, |R_{1,0}(c_i, c_j)|)$. The process of constructing G takes $\Theta(nm^2)$ time. In this graph theoretic formulation, each partitioning corresponds to a matching G. Thus, we take advantage of the algorithm described in Galil (1986) to find MWM with $O(m^3)$ running time.

In both bounding algorithms, one can maintain the bounds dynamically by processing only small changes from one node to another node near it (in some cases, to its sibling). The details of such dynamic maintenance are out of the scope of this work.

3.2.2 2-SAT

For this approach, we present a novel constraint satisfaction formulation that describes a set, containing but not necessarily equal to, all the valid flips-set corresponding to I. Let $z_{i,j}$ denote a binary variable corresponding to the (i, j)-th entry. We define variables for only zero entries. Consider a pair of columns (p, q) with $|R_{1,1}(p, q)| > 0$. For any row $r_1 \in R_{0,1}(p, q)$ and any row $r_2 \in R_{1,0}(p, q)$ add $z_{r_1,p} \vee z_{r_2,q}$. The intuition is that for any violation sextuplet, either one of zeros should be flipped. Satisfying these constraints is necessary to achieve a PP matrix, but not sufficient.

Let MWS (short for minimum weighted SAT) denote an arbitrary off-the-shelf tool that, given a satisfiable Boolean formula, outputs a satisfying assignment with the minimum number of variables assigned to true. Then the number of variables with true value in an optimal assignment, satisfying all these constraints, is a lower bound for the optimal number of flips resulting in a PP matrix. Formally, the lower bound is equal to

$$\mathsf{MWS}\left(\bigwedge_{\substack{p,q\in[m]:\ p<q\\ r_1,r_2:\ r_1\in R_{0,1}(p,q)\ \wedge\ r_2\in R_{1,0}(p,q)}} z_{r_1,p}\vee z_{r_2,q}\right). \tag{1}$$

After achieving a minimum weight satisfying assignment, we flip those zero entries that correspond to z variables with value 1.

Compact formulation. The formulation in Equation (1) can be expressed in fewer constraints by introducing a new set of variables and following the case distinction in Lemma 3.1: for each pair of columns p, q define a corresponding binary variable $B_{p,q}$. The weight of this new set of variables is set to zero in MWS formulation. If this variable is set to zero (by an MWS routine) then all variables $z_{r_1,p},\ r_1 \in R_{0,1}(p, q)$, take value 1. Similarly, if the variable is set to one, then all $z_{r_2,q},\ r_2 \in R_{1,0}(p, q)$ take value 1. Formally, the formulation changes to $\mathsf{MWS}(H_1 \wedge H_2)$, where

$$\begin{aligned} H_1 &= \bigwedge_{\substack{p,q\in[m]:\ p<q\\ r_1:\ r_1\in R_{0,1}(p,q)}} (B_{p,q}\vee z_{r_1,p})\\ H_2 &= \bigwedge_{\substack{p,q\in[m]:\ p<q\\ r_2:\ r_2\in R_{1,0}(p,q)}} (\overline{B_{p,q}}\vee z_{r_2,q}). \end{aligned} \tag{2}$$

The number of constraints corresponding to the column pair (p, q) will decrease from $|R_{0,1}(p, q)| \cdot |R_{1,0}(p, q)|$ in Equation (1) to $|R_{0,1}(p, q)| + |R_{1,0}(p, q)|$ in Equation (2). There are two advantages to the compact formulation: (i) the time spent on forming the set of constraints is shorter, (ii) for some set of inputs, heuristic sat-solvers run more efficiently on the formulation given in Equation (2) than on Equation (1), even though they are logically equivalent. In each experiment, we use only one of these formulations and it is specified in the corresponding description of the experiment.

Extra constraints. As another version of our lower bound, we add a set of new constraints. These constraints improve the lower bound to be a closer estimate of the optimal number of flips for some inputs. The tighter bound helps the BnB framework explore fewer nodes, even though the time to compute the bound per node increases. This advantage comes with a dip in the running time of the bounding calculation within each node.

The idea for the new set of constraints is to preclude some solutions that satisfy all constraints in Equation (1) but still do not remove all violations. In particular, when, for a specific pair of columns (p, q), $R_{1,1}(p, q)$ is empty, there is no constraint involving pair (p, q) in Equation (1). As an example, assume columns 1, 2 contain both (1, 0) and (0, 1) rows, but no (1, 1). Columns 2, 3 contain a violation, which MWS removes by flipping 0 s in column 2. This might create a (1, 1) in columns 1, 2, and create a violation that was not there in the beginning. Since, Equation (1) does not contain any constraints for columns 1, 2, this new violation is not removed.

In order to avoid such an outcome, we add new constraints to our formulation. Now, if some flip introduces a new violation, the extra constraints will enforce at least one additional flip to remove the newly created violation(s). Formally, the proposed set of constraints to add to Equation (1) is $E_1 \wedge E_2$, where

$$\begin{aligned} E_1 &= \left(\bigwedge_{\substack{p,q\in[m]:\ p<q\\ r_1,r_2,r_3:\\ r_1,r_3\in R_{0,1}(p,q)\ \wedge\ r_1\neq r_3\\ \wedge\ r_2\in R_{1,0}(p,q)}} z_{r_1,p}\vee z_{r_2,q}\vee\overline{z_{r_3,p}}\right)\\ E_2 &= \left(\bigwedge_{\substack{p,q\in[m]:\ p<q\\ r_1,r_2,r_3:\ r_1\in R_{0,1}(p,q)\\ \wedge\ r_2,r_3\in R_{1,0}(p,q)\ \wedge\ r_2\neq r_3}} z_{r_1,p}\vee z_{r_2,q}\vee\overline{z_{r_3,q}}\right). \end{aligned} \tag{3}$$

The number of constraints corresponding to columns (p, q) is $|R_{0,1}(p, q)| \cdot |R_{1,0}(p, q)| \cdot (|R_{0,1}(p, q)| + |R_{1,0}(p, q)|)$. This is higher than the number of constraints in both Equations (1) and (2). However, for some matrices (e.g. the one processed in Section 4.1) the resulting tighter bound improves the running time of overall BnB algorithm tremendously.

3.3 Initial solution

For the above bounding mechanism to start pruning, a feasible solution is required to initialize the variable $\mathsf{Best}_{\mathsf{Node}}$ at pseudocode line 1. When using random partition or MWM as a bounding algorithm, find an initial value as follows. We first find a pair of columns corresponding to a violation and flip one of the zero entries involved in the violation. We repeat this until no violation is left. On the other hand, when 2-SAT bounding is used, we solve the corresponding formulation from Equations (1), (2) or (3). We then apply the chosen entries to flip and repeat this process until we obtain a PP matrix. In each iteration, at least one flip will be performed and there are finitely many zero entries to flip. Therefore, this process always terminates and results in a PP matrix.

3.4 Analysis

Correctness. In the search tree explored by the BnB algorithm, we are guaranteed to find the optimum path from I to a PP matrix. This is because throughout the execution (a lower bound on) the projected number of flips that a node needs to reach PP is compared against the currently best-known way of reaching PP. If the node has no chance of beating the current best, it and all of its descendants are pruned. Consequently, PhISCS-BnB does not prune any nodes on the path to an optimum solution before reaching an optimum solution for the first time. Therefore, as long as our lower bounding techniques work correctly, our algorithm will discover an optimum PP.

In both the random partition and MWM techniques for obtaining a lower bound, we consider a partitioning of columns to pairs and calculate the minimum number of flips within each pair. Since the absence of violations within these pairs is necessary (but possibly not sufficient) for the removal of all violations, this estimate is a lower bound for the whole matrix.

In the 2-SAT approach, we form a set of constraints that must be satisfied in order to reach any PP descendant of a given node. That means the set of potential solutions obtained by satisfying all the constraints in the 2-SAT formulation is a superset of all PP matrices. That is why the optimum solution satisfying these conditions requires the same number of or fewer flips than any PP matrix.

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

Running time. The worst-case running time of Algorithm PhISCS-BnB is $O(2^{c_{opt}} \cdot T)$ where c_{opt} is the minimum number of flips needed to turn I to a PP matrix, and T is the running time of the bounding algorithm. The term $2^{c_{opt}}$ is an upper bound on the number of explored nodes, since PhISCS-BnB never explores a node with a priority score higher than c_{opt}. A naive bounding algorithm, as in Chen *et al.* (2006) and Cai (1996), incurs a running time of $T = O(mn)$. This is asymptotically the same as the running time for our random partition technique. Similarly, MWM runs in time $T = O(nm^2 + m^3)$: the first term is for forming the weighted graph, and the second term is for solving MWM. Solving the 2-SAT formulation takes super-polynomial time in the worst case as the weighted 2-SAT problem is NP-hard. While one might expect infeasibly long running times due to the hardness of 2-SAT, our experiments generally seemed to avoid worst-case behavior as the running times stayed mostly reasonable even for large matrices.

4 Experimental results

In this section, we discuss our experimental results on real data and simulations.

4.1 SCC data from mouse melanoma model

We first demonstrate the utility of PhISCS-BnB in studying evolutionary history on a real dataset harboring a large number of somatic mutations. This dataset was recently published in Wolf *et al.* (2019), where the effect of the extent of ITH on the immune response in melanoma was studied. In total, 20 single-cell clones (SCCs), denoted as C3, C4,…, C22, were derived from parental mouse melanoma B2905 cell line and bulk whole exome sequenced at the mean depth of 100×. [See Wolf *et al.* (2019) for additional details.]

After read alignment, calling and filtering the variants in all 20 SCCs, 2367 distinct SNVs were identified to be in at least one SCC. Importantly, the distribution of mutational variant allele frequencies revealed that many SCCs in the original dataset were comprised a mixture of one major and one or more minor subclonal populations. In order to obtain genotypes reflecting true single-cell data to be used as an input to PhISCS-BnB, we considered only those variants whose SCC specific overall read coverage is at least 20× and whose variant allele frequency is at least 40% as *present* in that SCC. All other entries in the genotype of the SCC were set to 0. Even though a stringent threshold on minimum variant allele frequency used in this genotype calling strategy can result in a number of false negative mutation calls, PhISCS-BnB was observed to successfully correct for this type of noise. (The empirical false negative rate reported by PhISCS-BnB was ∼8% due to stringent thresholding we applied on read coverage and variant allele frequency.)

The tree inferred in the original study (shown in the right part of Fig. 1) was derived by the use of non-private mutations only (i.e. mutations present in at least two SCCs) through clustering by the SciClone tool (Miller *et al.*, 2014), followed by manual post-processing and filtering of some of the reported clusters and finally phylogenetic tree was inferred by the use of ClonEvol (Dang *et al.*, 2017), a tool that treats each SCC as a mixture of cell populations and aims to heuristically build a single joint tumor phylogeny.

In order to facilitate a fair and natural comparison of the results by PhISCS-BnB to those in the original study, we first excluded all private mutations. The phylogeny reported by PhISCS-BnB on the remaining set of 1225 non-private SNVs is shown in the left part of Figure 1. As can be seen PhISCS-BnB inferred phylogeny generally agrees with that of the original study but is much more detailed with specific positions for mutations and cells in the tree topology. Furthermore, PhISCS-BnB suggests subtle but important differences, e.g. with respect to the placement of (i) clones C8 and C11 with respect to C4 and C16, (ii) clone C13 with respect to C21 and C22 as well as C12 and C18 and (iii) clones C9 and C17 on the trunk of tree. A more detailed depiction of the tumor phylogeny with mutation assignments for the chromosomes of each clone (marked blue) as well as PhISCS-BnB corrected false negative mutations (marked red) can be found in Supplementary Figure S5. As can be seen, the mutations in C9 form a subset of those in C17, which in turn is a subset of the mutations of their inferred descendants C7 and C15. Neither C9 nor C17 required many false negative corrections to be placed in the trunk. Similar observations can be made for other modifications suggested by PhISCS-BnB increasing our confidence with respect to the resulting tree topology.

In a second step, we built a more detailed phylogeny by including all 2367 mutations—as can be seen in Supplementary Figure S4. As expected, the tree constructed by PhISCS-BnB over the complete set of mutations is topologically equivalent to that presented in Figure 1 for the non-private mutations.

On NCI's Biowulf cluster PhISCS-BnB was able to compute the phylogeny for the complete set of mutations in <4 h and that for the non-private mutations in <2 h (specifics of the Biowulf cluster can be found in Supplementary Section B2). For performance comparison purposes, we also ran (to the best of our knowledge) the fastest available algorithmic tool, PhISCS-B (Malikic *et al.*, 2019b) on this dataset; PhISCS-B required >24 h to compute the same tree.

4.2 Comparison of PhISCS-BnB against PhISCS-B and PhISCS-I on simulated data

Next, we compared the running time of PhISCS-BnB on simulated data against the fastest available algorithmic tool, PhISCS (Malikic *et al.*, 2019b), which offers two variants: PhISCS-B is based on a CSP whereas PhISCS-I is based on ILP. Both were compared to PhISCS-BnB on simulated SCS data with 100–300 cells and 100–300 mutations, with false negative error rates ranging from 5% to 20%. In each case, 10 distinct trees of tumor evolution were simulated, each with 10 subclones. We allowed all three programs to run up to 8 h on each simulated dataset (details of the computational platform we used can be found in Supplementary Section B1). Figure 2 clearly shows that PhISCS-BnB is faster than the best available alternative (i.e. PhISCS-B) by a factor of 10–100. (Note that we

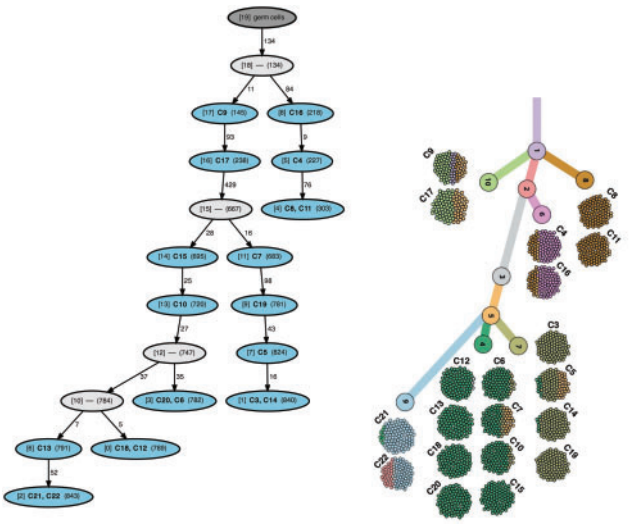

Fig. 1. The clustering and implied tree on the right is from Wolf *et al.* (2019) and was obtained by ClonEvol (Dang *et al.*, 2017) after the set of non-private mutations in 20 clonal sublineages of the B2905 cell line, derived from a mouse melanoma model, was manually filtered and clustered. The phylogeny on the left was obtained by PhISCS-BnB on the same set of mutations without manual processing. In each node of the PhISCS-BnB derived tree, the number inside the square brackets denotes the node id whereas the number inside the parentheses denotes the total number of mutations occurring on the path from the germline (root) to that node (i.e. the total number of mutations harbored by the node). Each edge is labeled with the number of mutations occurring between the associated parent and child nodes. The complete list of mutations occurring at each edge can be found at here. The nodes that are colored blue correspond to actual clonal sublineages whose labels are given in bold letters

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

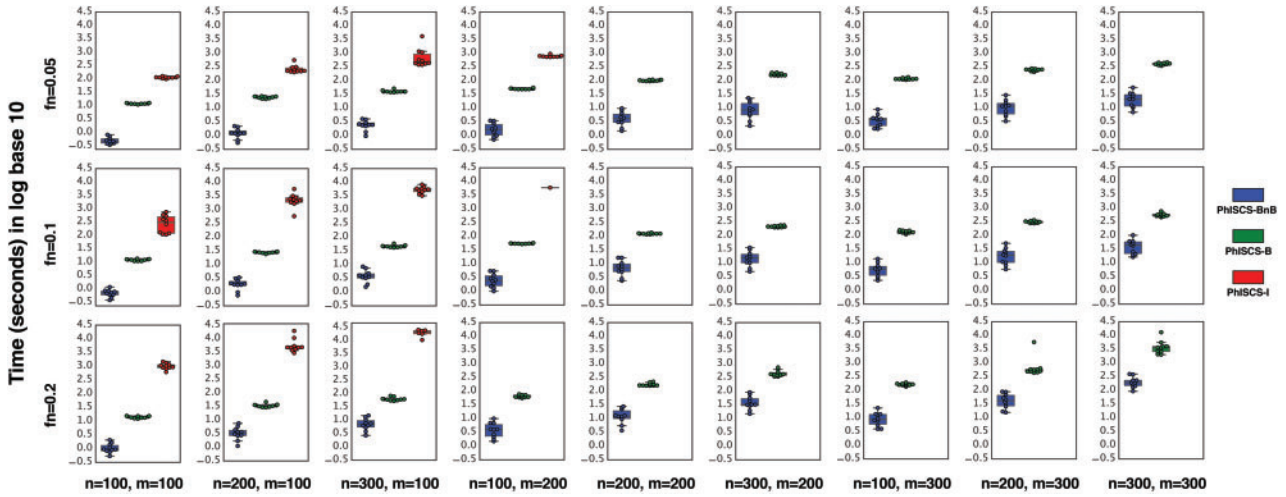

Fig. 2. Comparison of PhISCS-BnB with PhISCS-B and PhISCS-I in terms of running time (seconds) in log base 10. In each case, 10 distinct trees of tumor evolution were generated, each with 10 subclones. A time limit of 8 h was used for running each tool (those cases that exceed the time limit are not represented here). In the plot, *n*, *m* and *fn*, respectively, denote the number of cells, the number of mutations and the false negative error rate

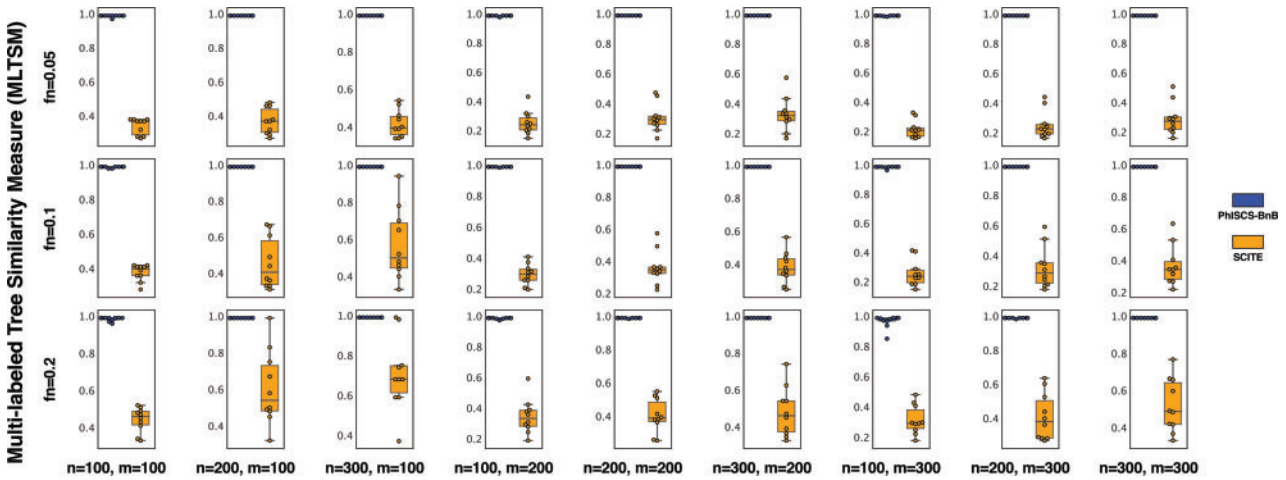

Fig. 3. Comparison of PhISCS-BnB with SCITE with respect to the MLTSM (Karpov *et al.*, 2019). For each panel, 10 distinct trees of tumor evolution were generated, each with 10 subclones. In the plot, *n*, *m* and *fn*, respectively, denote the number of cells, the number of mutations and the false negative error rate

used the compact formulation that is mentioned in Section 3.2.2 to run PhISCS-BnB on the simulated data but not on real data.)

4.3 Comparison of PhISCS-BnB against SCITE on simulated data

In a final experiment, we compared PhISCS-BnB against one of the best-known tools for tumor phylogeny reconstruction, SCITE (Jahn *et al.*, 2016), this time with respect to accuracy. As mentioned earlier, SCITE is based on MCMC and as such requires the user to specify the number of iterations, thus indirectly its running time. (The approximation of the time that SCITE takes per iteration for a given input matrix was calculated by running it 10 times, each with 20 000 iterations with 1 restart and then taking the average running time per iteration in this set of runs.) As input data, we simulated tumor phylogenies, each with 100–300 cells and 100–300 mutations, with false negative error rates ranging from 5% to 20%. In each case, 10 distinct trees of tumor evolution were generated, each with 10 subclones. We allowed SCITE to run with three restarts, each with a running time (the number of iterations allowed was calculated by dividing this time with the average time per iteration we calculate) 10 times that of PhISCS-BnB on the same input (again, details of the computational platform we used can be found in Supplementary Section B1) giving a significant advantage to SCITE over PhISCS-BnB.

For computing the *accuracy* of the inferred tumor phylogenies in comparison to the ground truth, we first used the multi-labeled tree similarity measure (MLTSM) (Karpov *et al.*, 2019) introduced recently. Since MLTSM is a normalized similarity measure, the closer its value to 1.0 implies a higher level of similarity between the inferred tree and the ground truth. The MTLSM between the ground truth trees and the inferred trees is presented in Figure 3. As can be expected, since PhISCS-BnB constructs the optimal tree, the similarity of its output to the ground truth is ∼1.0 for all datasets. On the other hand, even though SCITE was given significantly more running time than required by PhISCS-BnB, its output has a relatively low similarity to the ground truth (in the range of $[0.2, 0.6]$).

We have additionally compared the trees obtained by SCITE to those by PhISCS-BnB with respect to a number of commonly used measures of similarity to the ground truth—definitions given in e.g. Malikic *et al.* (2019a); see Supplementary Section D1 for detailed results. As can be seen, with respect to every measure and in every setting PhISCS-BnB offers on-par or superior performance in the given time limit. Note that we have also let SCITE to run for (approximately) 24 h to evaluate its full capability when the time limits are relaxed. As can be seen in Supplementary Section D4, SCITE can

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

then produce trees similar to the ground truth as per PhISCS-BnB. Finally, we compared PhISCS-BnB and SCITE when it was allowed to restart 3 times in two separate experiments; in the first one, we upper bounded the running time with a 1 h time limit—in the second one, we bounded the number of iterations to 1 million. The results can be found in Supplementary Sections D2 and D3, respectively. As can be seen SCITE does not perform well on larger data on any of the above settings.

As a final step, we have assessed the robustness of PhISCS-BnB on simulated data with non-zero false positive rates. Details of these simulations and the resulting performance of PhISCS-BnB can be found in Supplementary Section D5.

5 Conclusions

We presented new algorithms and based on them a software package, PhISCS-BnB, to solve the PP problem on noisy single-cell (mutation) sequencing data from tumors. On both simulated data and real data from mouse melanoma cell lines, we showed that PhISCS-BnB is one to two orders of magnitude faster than the best available methods, and can solve large instances of practical importance to optimality.

PhISCS-BnB is a BnB method, employing a variety of bounding techniques that use either state-of-the-art solvers for classical NP-hard problems such as max-SAT or polynomial time algorithms for 2-SAT and MWM, to prune efficiently and effectively the search tree of solutions. In theoretical computer science, different NP-complete problems are presented as equivalent and reducible to one another (in polynomial time) (Cormen *et al.*, 2009). However, the disproportionate practical importance of a few NP-complete problems, such as max-SAT and the Traveling Salesperson Problem (TSP) (Applegate *et al.*, 2006), has led to high-quality software that can solve instances of these NP-complete problems efficiently and to optimality. Therefore, efficient reduction of an NP-complete problem (such as PP) to max-SAT or TSP can leverage existing software to solve the problem much more efficiently. Even if the reduction does not preserve solutions (as per our reductions to 2-SAT or MWM problems), but only gives a bound on solution costs, the reduction can be used within a BnB framework, as we have done in PhISCS-BnB. This paradigm is widely applicable in bioinformatics in which domain-specific NP-complete problems abound (Gusfield, 2019).

Better understanding of SCS data may lead to better treatments or better strategies for drug development. In current treatment strategies, mutation sequencing data are presented to tumor boards to decide the course of treatment (Mueller *et al.*, 2019). In that clinical context, the rapid availability of phylogenetic trees to identify the tumor subclones could inform treatment. Hence, improving the efficiency of phylogenetic analysis of tumor data, as we have done in PhISCS-BnB, could have a direct impact on clinical treatment decisions.

Funding

This work was supported in part by the Intramural Research Program of the National Cancer Institute, National Institutes of Health, and utilized the computational resources of the NIH Biowulf high performance computing cluster (http://hpc.nih.gov) and Gurobi (http://www.gurobi.com) to solve some optimization problems. This work was also supported in part by the Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute (Stewart *et al.*, 2017). E.S.A., F.E. and S.C.S. were supported in part by the NSF [AF-1619081]. F.R.M. was supported in part by the Indiana U. Grand Challenges Precision Health Initiative. Y.S. was supported by the ERC [CoG-770854], European Union's Horizon 2020 research and innovation programme [#754282], H2020 European Research Council [#712977], Israel Science Foundation [#696/17] and MRA [#622106].

Conflict of Interest: none declared.

References

Alizadeh,A.A. *et al.* (2015) Toward understanding and exploiting tumor heterogeneity. *Nat. Med.*, **21**, 846–853.

Applegate,D.L. *et al.* (2006) *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA.

Cai,L. (1996) Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, **58**, 171–176.

Chen,D. *et al.* (2006) Minimum-flip supertrees: complexity and algorithms. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **3**, 165–173.

Cormen,T.H. *et al.* (2009) *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA.

Dang,H.X. *et al.* (2017) Clonevol: clonal ordering and visualization in cancer sequencing. *Ann. Oncol.*, **28**, 3076–3082.

Deshwar,A.G. *et al.* (2015) PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome Biol.*, **16**, 35.

Donmez,N. *et al.* (2017) Clonality inference from single tumor samples using low-coverage sequence data. *J. Comput. Biol.*, **24**, 515–523.

Edrisi,M. *et al.* (2019) A combinatorial approach for single-cell variant detection via phylogenetic inference. In: Huber,K.T. and Gusfield,D. (eds.), *19th International Workshop on Algorithms in Bioinformatics (WABI 2019). Leibniz International Proceedings in Informatics (LIPIcs)*, Vol. 143. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. **22**: 1–22:13.

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

El-Kebir,M. *et al.* (2015) Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics*, **31**, i62–i70.

El-Kebir,M. *et al.* (2016) Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell Syst.*, **3**, 43–53.

Galil,Z. (1986) Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, **18**, 23–38.

Gusfield,D. (1991) Efficient algorithms for inferring evolutionary trees. *Networks*, **21**, 19–28.

Gusfield,D. (2019) *Integer Linear Programming in Computational and Systems Biology: An Entry-Level Text and Course*. Cambridge University Press, Cambridge, UK.

Hajirasouliha,I. *et al.* (2014) A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data. *Bioinformatics*, **30**, i78–i86.

Ignatiev,A. *et al.* (2019) Rc2: an efficient MaxSAT solver. *J. Stat. Boolean Model. Comput.*, **11**, 53–64.

Jahn,K. *et al.* (2016) Tree inference for single-cell data. *Genome Biol.*, **17**, 86.

Jiao,W. *et al.* (2014) Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC Bioinform.*, **15**, 35.

Karpov,N. *et al.* (2019) A multi-labeled tree dissimilarity measure for comparing "clonal trees" of tumor progression. *Algor. Mol. Biol.*, **14**, 17.

Kuipers,J. *et al.* (2017) Advances in understanding tumour evolution through single-cell sequencing. *Biochim. Biophys. Acta*, **1867**, 127–138.

Laks,E., *et al.* (2019) Clonal decomposition and DNA replication states defined by scaled single-cell genome sequencing. *Cell*, **179**, 1207–1221.

Malikic,S. *et al.* (2015) Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics*, **31**, 1349–1356.

Malikic,S. *et al.* (2019a) Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. *Nat. Commun.*, **10**, 2750.

Malikic,S. *et al.* (2019b) PhISCS: a combinatorial approach for subperfect tumor phylogeny reconstruction via integrative use of single-cell and bulk sequencing data. *Genome Res.*, **29**, 1860–1877.

Marass,F. *et al.* (2016) A phylogenetic latent feature model for clonal deconvolution. *Ann. Appl. Stat.*, **10**, 2377–2404.

Martins,R. *et al.* (2014) Open-WBO: a modular MaxSAT solver. In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, Cham, Switzerland, pp. 438–445.

Miller,C.A. *et al.* (2014) Sciclone: inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution. *PLoS Comput. Biol.*, **10**, e1003665.

Mueller,S. *et al.* (2019) A pilot precision medicine trial for children with diffuse intrinsic pontine glioma—pnoc003: a report from the pacific pediatric neuro-oncology consortium. *Int. J. Cancer*, **145**, 1889–1901.

Popic,V. *et al.* (2015) Fast and scalable inference of multi-sample cancer lineages. *Genome Biol.*, **16**, 91.

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024

Ross,E.M. *et al*. (2016) OncoNEM: inferring tumor evolution from single-cell sequencing data. *Genome Biol*., **17**, 69.
Roth,A. *et al*. (2016) Clonal genotype and population structure inference from single-cell tumor sequencing. *Nat. Methods*, **13**, 573–576.
Satas,G. *et al*. (2017) Tumor phylogeny inference using tree-constrained importance sampling. *Bioinformatics*, **33**, i152–i160.
Singer,J. *et al*. (2018) Single-cell mutation identification via phylogenetic inference. *Nat. Commun*., **9**, 5144.
Stewart,C.A. *et al*. (2017) Indiana University Pervasive Technology Institute. Bloomington, IN.
Strino,F. *et al*. (2013) Trap: a tree approach for fingerprinting subclonal tumor composition. *Nucleic Acids Res*., **41**, e165.
Wolf,Y. *et al*. (2019) Uvb-induced tumor heterogeneity diminishes immune response in melanoma. *Cell*, **179**, 219–235.
Wu,Y. (2019) Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. *Bioinformatics*, **36**, 742–750.
Zafar,H. *et al*. (2017) SiFit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome Biol*., **18**, 178.
Zafar,H. *et al*. (2018) Computational approaches for inferring tumor evolution from single-cell genomic data. *Curr. Opin. Syst. Biol*., **7**, 16–25.
Zafar,H. *et al*. (2019) SiCloneFit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. *Genome Res*., **29**, 1847–1859.

Downloaded from https://academic.oup.com/bioinformatics/article/36/Supplement_1/i169/5870466 by guest on 23 April 2024