Lineage-Based Subclonal Reconstruction of Cancer Samples

Ph. D. Thesis

submitted to the
Faculty of Technology,
Bielefeld University, Germany
for the degree of Dr. rer. nat.

by

Linda Katharina Sundermann

January, 2019

Lineage-Based Subclonal Reconstruction of Cancer Samples

Ph. D. Thesis

submitted to the
Faculty of Technology,
Bielefeld University, Germany
for the degree of Dr. rer. nat.

by

Linda Katharina Sundermann

January, 2019

Supervisors:

Prof. Dr. Jens Stoye, Bielefeld University
Prof. Dr. Gunnar Rätsch, ETH Zurich
Prof. Quaid Morris, Ph. D., University of Toronto

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706.
Printed on non-aging paper according to DIN-ISO 9706.

Zusammenfassung

Krebs wird durch die Anhäufung von Mutationen verursacht und kann dadurch zu genetisch heterogenen Zellpopulationen führen. Es ist notwenig, eine Krebsprobe in Bezug auf ihre subklonare Rekonstruktion zu untersuchen. Die subklonare Rekonstruktion gibt Aufschluss darüber, welche Mutationen in welcher Population zusammen auftreten, welcher Zellanteil zu welcher Population gehört und in welcher Verwandtschaftsbeziehung die Populationen zueinander stehen. Mutationen, die typischerweise zur Inferenz einer subklonaren Rekonstruktion benutzt werden, sind einfache somatische Mutationen (englisch: *simple somatic mutations* (SSMs)) und Kopienanzahlaberrationen (englisch: *copy number aberrations* (CNAs)).

Methoden, die zur Bildung einer subklonaren Rekonstruktion ausschließlich SSMs verwenden, benutzen das Konzept von *Lineages* anstatt Populationen. Im Unterschied zu einer Population, die alle Zellen des gleichen Genotyps enthält, besteht eine Lineage aus allen Zellen, die aus der gleichen Ursprungszelle hervorgegangen sind. In einer Lineage-basierten subklonaren Rekonstruktion werden Mutationen den Lineages zugeordnet, in denen sie entstanden sind. Die Lineagehäufigkeit gibt den Anteil an Zellen an, in denen Mutationen, die der Lineage zugeordnet sind, auftreten.

Methoden, die eine subklonare Rekonstruktion mit CNAs erstellen, basieren auf Populationen. Im Unterschied zum Lineage-basierten Ansatz werden Mutationen allen Populationen zugeordnet, in denen sie auftreten. Um die Häufigkeit von Mutationen berechnen zu können, müssen die Beziehungen zwischen allen Populationen inferiert werden. Deshalb sind mehrere subklonare Rekonstruktionen nötig, um mehrdeutige Populationsbeziehungen darstellen zu können.

Zwei Populations-basierte subklonare Rekonstruktionsmethoden, die mit SSMs and CNAs arbeiten, sind PhyloWGS and Canopy. Im Vergleich zu Canopy inferiert PhyloWGS die CNAs nicht selber, sondern benötigt sie als Eingabe.

In dieser Doktorarbeit wird die erste Lineage-basierte Methode vorgestellt, die subklonare Rekonstruktionen mithilfe von SSMs und CNAs von massensequenzierten Tumorproben erstellt. Durch die Modellierung von CNAs als relative Kopienanzahl, also Kopienanzahlveränderungen, anstatt absolute Kopienanzahl wird es möglich sie Lineages zuzuordnen. Ein anderes Merkmal unserer Methode ist die Inferenz von vorhandenen oder fehlenden Lineagebeziehungen nur in Fällen, in denen Beziehungen in den Eingabedaten beobachtet werden können. Anderenfalls werden die Beziehungen als mehrdeutig modelliert. Dies erlaubt die Kombination von mehreren mehrdeutigen subklonaren Rekonstruktionen zu einer einzigen subklonaren Rekonstruktion.

Als Eingabe nutzt unsere Methode SSM-Allelehäufigkeiten und die durchschnittliche Allel-spezifische Kopienanzahl von Genomsegmenten. Darüberhinaus wird die Anzahl an Lineages als Eingabe benötigt. Die gemeinsame Wahrscheinlichkeitsfunktion für SSMs und CNAs wird präsentiert und die lineare Relaxation unseres Models als gemischt ganzzahliges lineares Programm veranschaulicht. Um die subklonare Rekonstruktion mit der besten Anzahl an Lineages aus einer Menge von subklonaren Rekonstruktionen zu bestimmen, die auf dem gleichen Datensatz mit unterschiedlicher Lineageanzahl inferiert wurden, wird das Prinzip der minimalen Beschreibungslänge benutzt. Eine ausführliche Analyse der ausgewählten subklonaren Rekonstruktion erlaubt die Klassifizierung von Beziehungen zwischen jedem Paar von Lineages als entweder vorhanden, fehlend oder mehrdeutig.

Unsere Methode ist in einer Software namens Onctopus implementiert. Onctopus wird ausführlich auf simulierten Daten evaluiert. Dabei wird die Laufzeit und der Speicherplatzverbrauch untersucht sowie auch die Performenz, wenn die mathematisch optimale Lösung nicht in gegebener Zeit und gegebenem Speicherplatz bewiesen werden kann. Es werden unterschiedliche Ansätze vorgestellt, um die Performenz von Onctopus zu verbessern. Diese umfassen das Clustern von Mutationen, das Fixieren von CNAs wie auch das Fixieren von Lineagehäufigkeiten.

Abschließend wird die Performenz von Onctopus gegen die Performenz von PhyloWGS and Canopy auf simulierten Datensätzen und einem tief sequenzierten Brustkrebsdatensatz verglichen. Auf dem simulierten Datensatz werden verschiedene Aspekte der inferierten subklonaren Rekonstruktionen verglichen. Es zeigt sich, dass Onctopus im Inferieren der Lineageanzahl und der Lineagebeziehungen überlegen ist. Für den Brustkrebsdatensatz wird einer Analyse von Deshwar *et al.* gefolgt, bei der die Zuordnung der inferierten Mutationen mit einer Goldstandardzuordnung verglichen wird. Onctopus und PhyloWGS erreichen hier eine vergleichbare Performanz.

Abstract

Cancer is caused by the accumulation of mutations, leading to genetically heterogeneous cell populations. The characterization of a cancer sample in terms of a *subclonal reconstruction* is essential. The subclonal reconstruction informs about the co-occurrence of mutations per population, as well as the proportion of cells belonging to each population, and the ancestral relationships among populations. Typical mutations used to infer a subclonal reconstruction are simple somatic mutations (SSMs) and copy number aberrations (CNAs).

Methods building subclonal reconstructions only with SSMs use the concept of lineages instead of populations. In contrast to a population, which comprises only cells with the same genotype, a lineage comprises all cells that are descendant from the same founder cell. In a lineage-based subclonal reconstruction, mutations are assigned to the lineage in which they arose. The lineage frequency indicates the proportion of cells in which mutations assigned to this lineage can be found.

Methods building subclonal reconstructions with CNAs are population-based. In contrast to the lineage-based approach, mutations are assigned to all populations in which they occur, not just to the one in which they arose. In order to calculate the mutation frequencies, the ancestor-descendant relationships between all populations have to be inferred. Hence, multiple subclonal reconstructions are needed to model ambiguous population relationships.

Two population-based subclonal reconstruction methods working with SSMs and CNAs are PhyloWGS and Canopy. In contrast to Canopy, PhyloWGS does not infer CNAs but needs them as input.

In this thesis, we present the first lineage-based model that builds subclonal reconstructions from SSMs and CNAs of bulk-sequenced tumor samples. Modeling CNAs as relative copy numbers, so copy number changes, instead of absolute copy num-

bers allows us to assign them to lineages. Another special feature of our method is that we infer present or absent ancestor-descendant relationships between lineages only if they can be observed in the data, modeling them as ambiguous relationships otherwise. This enables us to combine multiple ambiguous subclonal reconstructions within a single subclonal reconstruction.

As input, our method uses the variant allele frequencies of SSMs, as well as the average allele-specific major and minor copy numbers of genome segments where the genome is segmented in a way that consecutive regions with the same copy number profile belong to the same segment. Furthermore, the number of lineages needs to be given as input. We present a joint likelihood function for SSMs and CNAs and show a linear relaxation of our model as a mixed integer linear program that can be solved with state-of-the-art solvers. Given subclonal reconstructions of the same dataset inferred with different lineage numbers, we use the minimum description length principle to choose the subclonal reconstruction with the best lineage number. An extensive analysis of the chosen subclonal reconstruction allows us to classify the ancestor-descendant relationships between each pair of lineages as either present, absent or ambiguous.

We implemented our method in a software called Onctopus. We evaluate Onctopus extensively on simulated data, analyzing its run time and memory usage as well as its performance when the mathematically optimal solution cannot be proved in the given time and space. We present different approaches to improve Onctopus' performance, such as by clustering mutations, fixing CNAs or fixing lineage frequencies.

Finally, we compare the performance of Onctopus against the performance of PhyloWGS and Canopy on simulated datasets and a deep sequenced breast cancer dataset. On the simulated datasets, we evaluate different aspects of the inferred subclonal reconstructions and show that Onctopus is superior in inferring the number of lineages and the lineage relationships. For the breast cancer dataset, we follow an analysis by Deshwar *et al.*, comparing the inferred mutation assignment to a gold standard assignment. Here, Onctopus and PhyloWGS reach a comparable performance.

Acknowledgments

Now after finishing the last chapter of this thesis, it's time for me to thank those people who made this work possible.

First, I want to thank Jens Stoye for giving me the chance to join your group after finishing my Bachelor studies and for giving me the time I needed to finish my Master studies. Thank you for giving me time to learn for myself and never telling me what to do but always being there when I had questions. Second, I want to thank Gunnar Rätsch for letting me visit your group at Memorial Sloan Kettering Cancer Center and keeping me attached to your group afterwards. Thank you for inviting me to join your group at ETH Zurich for half a year and for always asking detailed questions about my research, challenging me with further improvements. Third, I want to thank Quaid Morris for having the idea that led to this PhD thesis. Thank you for having time in the last half of a year to meet once a week and to discuss my progress, as well as for always being encouraging and excited.

Also, I want to thank the whole Genome Informatics group and all the students of the DiDy graduate school. Thank you for making this time on U-10 a wonderful time and thanks for all the cakes. I also want to thank Roland Wittler for always having five minutes and good answers, and for always asking critical questions. Thank you Dany Dörr, Elói Soares de Araújo and Tizian Schulz for being great office mates, never complaining about me shutting the door and banning the distracting life outside from the office, especially in the last half of a year. And thank you Nina Luhmann, Guillaume Holley, Jan Kölling, Tina Zekić and Markus Lux, without you it wouldn't have been half as much fun. Furthermore, I want to thank the whole Rätsch lab for "adopting" me twice, making me feel like a member and not just a visitor. Especially, I want to thank André Kahles and Kjong Lehmann for providing me with critical feedback and always good advice.

I also want to thank Nina Luhmann, Panos Papavasileiou, Markus Lux, Kai Stadermann and Roland Wittler for proof reading parts and looong parts of this thesis.

I want to thank my whole family for always being supportive, especially my parents and my tall little brother for always believing in me. My biggest thanks go to Panos, without you this work would not have been possible at all. Thank you for patiently listening to all my talking about optimizations and ambiguity, for cooking for me when I was too busy leaving my desk, and for looking as much as me forward to the exiting time starting now. But most of all, thank you for making me smile, for making me laugh and for making me happy every day.

Finally, I thank the German Academic Exchange Service (DAAD) for providing me with a "FIT weltweit" scholarship, enabling me to visit Memorial Sloan Kettering Cancer Center. I also thank the German Research Foundation (DFG) for partially funding me via the International DFG Research Training Group GRK 1906, and the Young Researchers' Fund of Bielefeld University for funding six months of my final year.

Contents

List of Abbreviations

AIC	Akaike information criterion
AUPRC	area under the precision-recall curve
BIC	Bayesian information criterion
CNA	copy number aberration
CNV	copy number variation
ILP	integer linear program
LOH	loss of heterozygosity
LP	linear program
MCMC	Markov Chain Monte Carlo
MDL	minimum description length
MILP	mixed integer linear program
NGS	next-generation sequencing
SNP	single nucleotide polymorphism
SNV	single nucleotide variant
SOS2	special ordered set of type 2
SSM	single somatic mutation
VAF	variant allele frequency

Notation Tables

General Notation

θ	parameter set
Beta-Bin	probability mass function of the beta binomial distribution
Beta-Bin′	logarithmic probability mass function of the beta binomial distribution
$\mathcal{C}$	a code function, encodes the input into a binary string
D	data
$\mathcal{L}$	a likelihood function
$\mathcal{L}'$	a log-likelihood function
$L_{\mathcal{C}}$	description or code length function
$\mathcal{M}$	a model
m	sample size of data
$\mathcal{N}$	probability density function of the normal distribution
$\mathcal{N}'$	simplified logarithmic probability density function of the normal distribution
$\mathcal{P}$	a probability distribution
x^m	input data, observations
y^m	output data, target values

Numbers and Indices for Subclonal Reconstructions

α	allele index, $\alpha \in \{A, B\}$
I	number of genome segments
I'	number of segments without CNAs
I''	number of segments with CNAs

Numbers and Indices for Subclonal Reconstructions

I°	number of CNAs
$I^\bullet$	number of CNAs that are assigned to a segment that already contains at least one CNA
I^*	number of CNAs and CNA-free segments
i	segment index, $i \in \{0, \ldots, I-1\}$
J	number of SSMs
J'	number of SSMs on segments with CNAs
J''	number of SSMs that are assigned to lineage segments which also contain copy number duplications of the same phases than the SSMs
j	SSM index with $0 \leq j < J$
K	number of lineages or populations
k	lineage or population index, $k \in \{0, \ldots, K-1\}$
M	number of mutations
N	number of tumor samples
n	tumor sample index, $n \in \{0, \ldots, N-1\}$
r	a subclonal reconstruction

Variables for Subclonal Reconstructions

$\phi_{k,n}$	frequency of lineage k in sample n
$\eta_{k,n}$	frequency of population k in sample n
χ_k	set of children of lineage or population k
$\mathcal{A}_k$	set of ancestors of lineage or population k
c_num_max	maximal number of copy number changes per segment i
$c_{i,n}$	average copy number of segment i in sample n
$\hat{c}_{i,n}$	inferred average copy number of segment i in sample n
$c_{\alpha_{i,n}}$	average allele-specific copy number of allele α of segment i in sample n
$\hat{c}_{\alpha_{i,n}}$	inferred average allele-specific copy number of allele α of segment i in sample n
C_k	total integer copy number of population k
$C_{\alpha_{i,n}}$	total integer allele-specific copy number of allele α of segment i in sample n
$\Delta C_{\alpha_{i,k}}$	inferred copy number change on allele α in segment i of lineage k

Variables for Subclonal Reconstructions

$\Delta C^{gain}_{\alpha_{i,k}}$	whether a copy number gain is assigned to allele α in segment i of lineage k
$\Delta C^{loss}_{\alpha_{i,k}}$	whether allele α in segment i of lineage k gets lost
$\Delta C\_\mathcal{A}^{loss}_{\alpha_{i,k,k'}}$	whether lineage k is an ancestor of lineage k' and whether a copy number loss is assigned to allele α of lineage k
$\Delta C\_\mathcal{D}^{gain}_{\alpha_{i,k,k'}}$	whether lineage k' is a descendant of lineage k and whether a copy number gain is assigned to allele α in segment i of lineage $'k$
$\Delta C\_\mathcal{D}^{loss}_{\alpha_{i,k,k'}}$	whether lineage k' is a descendant of lineage k and whether a copy number loss is assigned to allele α in segment i of lineage $'k$
$\Delta C\_freq^{gain}_{\alpha_{i,k,n}}$	$\phi_{k,n}$ if a copy number gain is assigned to allele α in segment i of lineage k, 0 otherwise
$\Delta C\_freq^{loss}_{\alpha_{i,k,n}}$	$\phi_{k,n}$ if a copy number loss is assigned to allele α in segment i of lineage k, 0 otherwise
$child\_freq_{n,k,k'}$	$\phi_{k',n}$ if lineage k' is a child of lineage k, otherwise 0
$\mathcal{D}_k$	set of descendants of lineage or population k
$D_{j,n}$	total read count of SSM j in sample n
$\Delta F_{\alpha_{j,k}}$	whether SSM j is assigned to allele α of lineage k and whether its average copy number $\hat{s}_{j,n}$ is influenced by a copy number gain assigned to α in the same segment of k
n_{knots}	number of knots in an interval to compute a piecewise linear function
$p_{j,n}$	variant allele frequency of SSM j in sample n
$\hat{p}_{j,n}$	inferred variant allele frequency of SSM j in sample n
$\tilde{p_{j,n}}$	approximated inferred variant allele frequency of SSM j in sample n
$R_{j,n}$	reference read count of SSM j in sample n
$\mathcal{S}_k$	set of mutations belonging to lineage k
$\mathcal{S}'_k$	set of mutations belonging to population k
$s_{j,n}$	average copy number of SSM j in sample n
$\hat{s}_{j,n}$	inferred average copy number of SSM j in sample n
$\Delta S_{j,k}$	whether SSM j is assigned unphased to lineage k
$\Delta S_{\alpha_{j,k}}$	whether SSM j is assigned to allele α of lineage k
$\Delta S\_freq_{j,k,n}$	$\phi_{k,n}$ if SSM j is assigned to lineage k, 0 otherwise

Variables for Subclonal Reconstructions

$\Delta S\_freq_{\alpha_{j,k,n}}$	$\phi_{k,n}$ if SSM j is assigned to allele α of lineage k and if its average copy number $\hat{s}_{j,n}$ is influenced by a copy number gain assigned to α in the same segment of k, 0 otherwise
$\Delta S\_freq^{gain}_{\alpha_{j,k,k',n}}$	$\phi_{k',n}$ if lineage k' is a descendant of lineage k, if SSM j is assigned to allele α of k and if in the same segment a copy number gain is assigned to allele α of k'
$\Delta S\_freq^{loss}_{\alpha_{j,k,k',n}}$	$\phi_{k',n}$ if lineage k' is a descendant of lineage k, if SSM j is assigned to allele α of k and if in the same segment a copy number loss is assigned to allele α of k'
$V_{j,n}$	variant read count of SSM j in sample n
$w_{j,n,j'}$	weight of knot j' in interval of piecewise linear function of VAF of SSM j in sample n
$w_{\alpha_{i,n,i'}}$	weight for knot i' in interval of piecewise linear function of average copy number $c_{\alpha_{i,n}}$
$Z_{k,k'}$	ancestor-descendant relationship between lineages k and k'
$Z\_tree\_1_{k,k',k''}$	whether lineage k is an ancestor of lineage k' and whether lineage k' is an ancestor of lineage k''
$Z\_tree\_2_{k,k',k''}$	whether lineage k is an ancestor of lineage k'' and whether lineage k' is an ancestor of lineage k''

Chapter 1

Introduction

Worldwide, cancer is one of the leading causes of death [5]. It is caused by the accumulation of genetic mutations [84], which leads to heterogeneous cell populations [74]. This intratumor heterogeneity can cause cancer therapies to fail, e. g. by positively selecting cells that harbor resistances against the drug applied [35]. Thus, a characterization of the tumor in terms of a *subclonal reconstruction* is essential.

The subclonal reconstruction describes the co-occurrence of mutations per population, the proportion of cells belonging to each population, and the ancestral relationships among them. The current standard practice for this analysis of cancer samples is based on bulk-sequencing data. Working with single-cell sequencing data is also possible but not well established yet, among other reasons due to its high noise levels [31].

Mutations typically used to infer a subclonal reconstruction are simple somatic mutations (SSMs) and copy number aberrations (CNAs). SSMs comprise substitutions of single DNA base pairs and insertions and deletions of a couple of DNA base pairs. CNAs are structural variations that change the copy number of a genome segment.

Methods working only with SSMs infer subclonal reconstructions in terms of lineages. In contrast to a population, which comprises all cells of the same genotype, a lineage comprises all cells that descend from the same founder cell. In a lineage-based subclonal reconstruction, mutations are assigned to the lineage in which they occur first. The lineage frequency indicates the proportion of cells which contain the mutations that are assigned to this lineage.

Methods working with CNAs assign mutations to all populations containing the mutation. All ancestral relationships need to be inferred in order to compute the frequency of a mutation. Thus, ambiguity caused by relationships between populations can be detected only through different subclonal reconstructions.

Subclonal reconstruction methods working only with SSMs [24,64] are restricted to copy number neutral regions of the genome. Other methods utilize CNA information as well [47,78] but make the simplified assumption that all cells containing an SSM are either influenced by a copy number change or not. Only recent methods model cells with SSMs to be differently influenced by copy number changes [21,25,46]. However, since these methods model the subclonal reconstruction in terms of populations, they need to sample multiple subclonal reconstructions to be able to capture ambiguity in the input data.

In this thesis, we present the first method utilizing SSMs and CNAs to build a lineage-based subclonal reconstruction. Our method is based on a probabilistic model with a joint likelihood function for SSMs and CNAs. We formulate the linear relaxation of our model as a mixed integer linear program (MILP) that can be solved with state-of-the-art solvers. Thanks to an extensive analysis of all lineage relationships of a subclonal reconstruction, we can classify all relationships either as present, absent or ambiguous ancestor-descendant relationships. This allows us to combine ambiguous subclonal reconstructions algorithmically within a single subclonal reconstruction without having to sample over the whole solution space. Our method is implemented in a software called Onctopus.

After giving a short introduction into the topic of this thesis and the current state of the field, we provide more information about probabilistic models and optimization, biological and technical details, and subclonal reconstructions of cancer samples in Chapter 2. In Chapter 3, we present our new lineage-based subclonal reconstruction model and explain how we deal with ambiguity in lineage relationships in Chapter 4. We analyze Onctopus' performance on simulated data and show how we improve it in Chapter 5 before evaluating Onctopus against the two methods PhyloWGS [21] and Canopy [46] on simulated data as well as on a deep sequenced breast cancer dataset in Chapter 6. At the end of this thesis in Chapter 7, we summarize our findings and give an outlook over interesting extensions of Onctopus.

Chapter 2

Background

In this chapter, we give information about probabilistic models and optimization in Section 2.1, about the biological and technical background of cancer, sequencing and mutation calling in Section 2.2, and about subclonal reconstructions of cancer samples in Section 2.3.

2.1. Probabilistic Models and Optimization

Given some data $D = (x^m, y^m)$, a probabilistic model $\mathcal{M} = \mathcal{P}(y^m \mid \theta, x^m)$ defines a probability $\mathcal{P}$ that the observations x^m are mapped to the target values y^m. The used parameter set θ is usually unknown at the time of data observation and needs to be estimated. A popular method to estimate θ is the *maximum likelihood principle* [7], which finds an estimate $\hat{\theta}$ that maximizes $P_\theta(y^m \mid x^m)$ for $\theta \in \Theta$, the parameter space. For some model classes, the maximum likelihood estimator $\hat{\theta}$ can be derived with a closed formula. For other model classes, global optimization methods or sampling approaches are needed in order to find or approximate $\hat{\theta}$.

In Subsection 2.1.1, we explain the global optimization method mixed integer linear programming, which can be used to derive a maximum likelihood estimator and with which we work in our developed subclonal reconstruction method. Afterwards in Subsection 2.1.2, the sampling approach Markov Chain Monte Carlo (MCMC) is described briefly because our method is compared against two other methods that use MCMCs. Finally, in Subsection 2.1.3, we introduce different concepts of model selection, emphasizing the minimum description length principle since it is applied in our method.

2.1.1. Mixed Integer Linear Programming

To understand mixed integer linear programming, we will first introduce linear programming and integer linear programming based on the book of Cormen *et al.* [17]. Linear programming is a class of convex optimization with a linear objective function and linear equations and non-strict inequalities as constraints. The following is an example of a simple linear program (LP):

$$\begin{aligned} \max \quad & \sum_{j=1}^{M-1} c_j \cdot x_j \\ \text{s.t.} \quad & \sum_{j=0}^{M-1} a_{i,j} \cdot x_j \leq b_i \quad \text{for } i = 1, \ldots, N-1, \\ & x_j \geq 0 \quad \text{for } j = 1, \ldots, M-1, \end{aligned}$$

where x_j are the real-valued variables, and c_j, $a_{i,j}$ and b_i are real numbers as well.

Canonical forms to express LPs are the standard and the slack form, both use a maximization. To describe constraints, the standard form uses linear inequalities and the slack form linear equalities. Each LP can be converted to both forms. [1]

A setting of the variables $x_0, \ldots, x_{M-1}$ that satisfies all constraints is called a *feasible* solution. In contrast, the variable setting of an *infeasible* solution does not satisfy all constraints. If no feasible solution exists for a linear program, it is infeasible. The goal of linear programming is to find the optimal solution, that is the feasible solution that maximizes (or minimizes) the objective function.

A classical linear programming algorithm is the simplex algorithm, which runs fast in practice but exponentially in the worst-case. Polynomial time algorithms are the ellipsoid algorithm as well as interior-point methods.

When all variables of an LP are required to take integer values, we have an integer linear program (ILP). When only some variables are required to take integer values, we have a mixed integer linear program (MILP). Finding a feasible solution for an ILP or MILP is NP-hard.

A commonly used approach to solve ILPs and MILPs are branch-and-cut algorithms. These exact algorithms use an enumeration tree to check variable settings for feasibility and to find the optimal solution. The following explanation of the general concept of branch-and-cut algorithms is based on the description of Johnson *et al.* [48].

[1] In this thesis, we will not use a canonical form but will present our linear constraints in what ever form is most intuitive in the given setting.

Branch-and-Cut Algorithm. The root node of the enumeration tree is the original ILP or MILP formulation[2]. While searching and proving the optimal solution, more nodes are added to the tree, which is called branching. Here, children to the current node are created, which extend the MILP formulation of the current node by adding constraints that explore different values of the integer variables. For example, when a variable is restricted to binary values, the constraint in one child would fix it to 0 and in the other child to 1.

The best feasible solution found so far, when processing some node in the tree, is f_M^* and provides a lower bound for the MILP. Now for each node ν in the tree we define $P_M(\nu)$ as the MILP present at ν and $f_M(\nu)$ as its optimal solution that we want to find. The linear programming relaxation of $P_M(\nu)$, which removes all integer constraints, is defined as $P_L(\nu)$. Its optimal solution is defined as $f_L(\nu)$ and can be found in polynomial time. The following relations between $f_M(\nu)$ and $f_L(\nu)$, as well as between $P_M(\nu)$ and $P_L(\nu)$ hold:

- If $f_L(\nu)$ fulfills all integer constraints of $P_M(\nu)$, then $f_M(\nu) = f_L(\nu)$.
- If $f_L(\nu)$ does not fulfill all integer constraints of $P_M(\nu)$, it is an upper bound of $f_M(\nu)$.
- If $P_L(\nu)$ is infeasible, then $P_M(\nu)$ is infeasible as well.

Now for each node ν in the tree, we solve $P_L(\nu)$. If $P_L(\nu)$ is infeasible, we can discard ν. If $P_L(\nu)$ is feasible but $f_L(\nu)$ does not fulfill all integer constraints, the way of proceeding depends on the value of $f_L(\nu)$. If $f_L(\nu) \leq f_M^*$, we can discard ν. Otherwise, if $f_L(\nu) > f_M^*$, we have to branch the node and investigate whether integer solutions exists on this branch. If $f_L(\nu) = f_M(\nu)$ and if $f_M(\nu) > f_M^*$, we update f_M^* and discard ν. If $f_M(\nu) \leq f_M^*$, we discard ν directly.

To avoid frequent branching, *cuts* can be applied to the node. Cuts are special constraints that permit only solutions to $P_L(\nu)$ that fulfill all integer constraints. Also, a high value for f_M^* can prevent further branching. To find a high value of f_M^* at the beginning, fast heuristics can be used. Finally, the bounds provided by the linear programming relaxation depend on the concrete formulation of the MILP. Useful bounds are the results of a good initial formulation.

Optimality of f_M^* is only proved when the complete enumeration tree is processed. However, the optimal value of f_M^* can be found before the complete tree is processed.

[2]For simplicity, we will only talk about MILPs in the context of the branch-and-cut algorithm from now on.

Linear programming only allows the use of linear constraints. There are, however, some tricks how non-linear constraints or functions can be modeled. In the following, we present two such tricks.

Multiplying a Real-Valued and a Binary Variable. In order to model the multiplication of a real-valued variable x_r, with $0 \leq x_r \leq 1$, and a binary variable x_b, we introduce a real-valued variable x_a that fulfills the following three constraints:

$$x_a \leq x_b \tag{2.1}$$

$$x_a \leq x_r \tag{2.2}$$

$$x_a \geq x_r - (1 - x_b) \tag{2.3}$$

The above equations can also be used to model the following statement based on two conditions:

$$x_a = \begin{cases} 1 & \text{if } x_b = 1 \text{ and } x_r = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Approximating Non-Linear Functions. A concave non-linear function f can be approximated with a piecewise linear function f'. Therefore, in an interval $[a, b]$, the piecewise linear function f' is interpolated through n points $P = (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$, where $a \leq x_i \leq b$ for $0 \leq i < n$ and where $y_i = f(x_i)$. The positions x_i within the interval, which are used for the interpolation, are called *knots*. Now, for each position x in the interval, the approximated value $y = f'(x)$ can be derived as follows:

$$\begin{aligned} x &= \sum_i w_i \cdot x_i \\ y &= \sum_i w_i \cdot y_i \\ \sum_i w_i &= 1, \end{aligned} \tag{2.5}$$

for $i = 0, \ldots, n-1$ and w_i is a weight associated with point (x_i, y_i). All weights $\vec{w} = (w_0, w_1, \ldots, w_{n-1})$ are between 0 and 1 and are in a special ordered set of type 2 (SOS2). This means that at most two weights can have non-zero values and when two weights have non-zero values, these weights have to be adjacent.

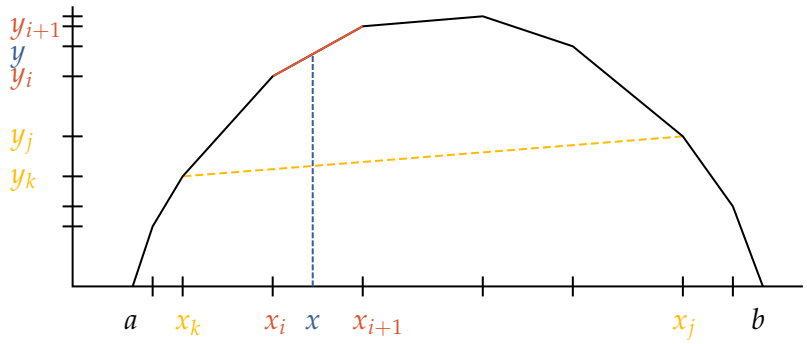


Figure 2.1.: Strictly concave piecewise linear function f' in the interval $[a, b]$. Knots are shown by ticks on the x-axis, their corresponding function values are shown with ticks on the y-axis. The position x lies between the two knots x_i and x_{i+1}. We want to find its maximal value y that lies in the convex hull of f'.

If f' is a strictly concave piecewise linear function and f is a part of the objective function that is maximized, the weights $\vec{w}$ are automatically in an SOS2 and no additional SOS2 constraints are necessary. The reason for this can easily be seen in Figure 2.1. Given x, which lies between the two knots x_i and x_{i+1}, the only way to derive at a maximal value of y is to set the weights w_i and w_{i+1} to a non-zero value and all other weights to 0. Like this, y lies on the line between (x_i, y_i) and (x_{i+1}, y_{i+1}). All other convex combinations of $\vec{w}$ lie within the convex hull of the piecewise linear function f' but not on f' itself, thus they will result in a lower value of y. If $x = x_i$, it is obvious that $w_i = 1$ to maximize y. Thus, the weights $\vec{w}$ are in a SOS2.

To solve LPs, ILPs and MILPs different academic and commercial software packages exist, such as the GNU Linear Programming Kit (GLPK) [33], lp_solve [62], Gurobi [37] and IBM ILOG CPLEX Optimization Studio (CPLEX) [18]. Since we use CPLEX to implement our MILP, the software package is briefly introduced.

CPLEX. CPLEX provides libraries for solving linear programming and variations of it, such as mixed integer linear programming. To solve MILPs, CPLEX uses a branch-and-cut algorithm with preprocessing and different heuristics. It terminates when the optimal solution is proved or when user-specified run time or memory limits are reached. When it terminates because these limits are reached, it reports the best variable setting that was found up to this point.

2.1.2. Markov Chain Monte Carlo

The following explanation of MCMC methods is based on the book by Bishop [7]. MCMCs are a type of numerical sampling methods designed to draw samples from a posterior distribution $\mathcal{P}$. Therefore, L samples, $z_0, z_1, \ldots, z_{L-1}$, are drawn from a proposal distribution $\mathcal{P}'$ based on a Markov chain, which converges towards $\mathcal{P}$ for $L \to \infty$.

A specific class of MCMC algorithms are the Metropolis [68] and the Metropolis-Hastings algorithms [43]. In each iteration of these algorithms, a candidate sample z^* is drawn from the proposal distribution $\mathcal{P}'$, with $\mathcal{P}'(z^* \mid z_t)$. A specific criterion defines the rate with which z^* is accepted. If z^* is accepted, it becomes the next sample in the sample set, $z_{t+1} = z^*$, and drawing the next sample depends on z^*. If z^* is rejected, the previous sample z_t is used again for z_{t+1}.

Since samples are drawn with a Markov chain, successive samples are highly correlated and not independent as required by numerical sampling methods. To acquire an independent set of samples, it is sufficient to keep only every M^{th} sample and discard the others.

In practice, only a finite number of iterations can be made. Thus, instead of reaching the stationary distribution $\mathcal{P}$, it is possible that the MCMC gets stuck in a local optimum. To prevent this, multiple chains from independent starting points can be used.

Because samples drawn at the beginning do not have to reflect the stationary distribution $\mathcal{P}$, a common approach is to discard the first samples. This process is called *burn-in* [42].

2.1.3. Model Selection

Given a model class $\mathcal{M}^*$ and some data $z^m = (x^m, y^m)$, the problem of model selection is to choose the best model $\mathcal{M} \in \mathcal{M}^*$. Generally, a model $\mathcal{M}'$ that uses more parameters than a model $\mathcal{M}''$ will have a higher likelihood $\mathcal{L}_{\mathcal{M}'}$ than $\mathcal{M}''$ because it provides a better fit for the data. However, simply choosing $\mathcal{M}'$ over $\mathcal{M}''$ often results in overfitting, which has to be avoided. To choose a model that balances between data fit and model complexity, different methods such as the Akaike information criterion, the Bayesian information criterion, or the minimum description length principle exist.

Akaike Information Criterion. The Akaike information criterion (AIC) [2] penalizes models relatively to the number of parameters $|\theta|$ they use. It states to choose the model $\mathcal{M}$ which minimizes

$$-2\log(\mathcal{L}_{\mathcal{M}}) + 2|\theta_{\mathcal{M}}|.$$

Hence, it weights the log likelihood and model complexity by equal parts.

Bayesian Information Criterion. Similarly to AIC, the Bayesian information criterion (BIC) [81] uses a penalty term to penalize complex models, containing a high number of parameters. It states to choose the model $\mathcal{M}$ which maximizes

$$\log(\mathcal{L}_{\mathcal{M}}) - \frac{1}{2} \cdot |\theta_{\mathcal{M}}| \cdot \log(m),$$

where $|\theta_{\mathcal{M}}|$ is the number of parameters of $\mathcal{M}$ and m is the sample size. In addition to AIC, BIC weights the model complexity with the data size.

Minimum Description Length Principle. The minimum description length (MDL) principle [76] is based upon the theory of Kolmogorov complexity, which defines the length of a sequence as the length of the shortest computer program that outputs this sequence [14, 53, 83]. The motivation of the MDL principle comes from data compression: If a model describes some data containing regularities well, it can be compressed and hence, its description length is small [42].

The following content about the MDL principle is based on the book of Grünwald *et al.* [36]. In order to compute the description length of a model and some data, we need a code $\mathcal{C}$ that encodes an input x into a binary string b, $\mathcal{C}(x) = b$. The code length or description length of x is $L_{\mathcal{C}}(x) = |b|$, which corresponds to the length of the binary string b, i. e. the number of bits in b. The code $\mathcal{C}$ needs to fulfill two properties: First, it has to be lossless, meaning that x can always be decoded from b and that there is only one unique x for which holds $\mathcal{C}(x) = b$. Second, it has to be a prefix code, meaning that no code word is a prefix of another code word. Given a code $\mathcal{C}$ which fulfills these properties, it is now possible to encode x and send it to a receiver who can then decode it with the help of $\mathcal{C}$.

For building a prefix code, Huffman coding [44] can be used. Alternatively, it is possible to use a fixed-length code, in which all code words have the same fixed length, which is known to the receiver. To send an integer i to the receiver, $\lceil \log(i) \rceil$ bits are necessary. If, however, the receiver does not know the number of bits because it is intrinsic to the input x, the *simple standard code for the integers* can be used to encode i.

Here, the number of bits that is needed to encode i is communicated to the receiver by sending as many 0s, followed by a 1. Afterwards, the actual number i is encoded with $\lceil \log(i) \rceil$ bits.

When we work with the MDL principle, we are only interested in the code length, not in the actual code itself. Allowing non-integer code lengths, the simple standard code for the integers needs $2\log(i) + 1$ bits. Given a probability distribution $\mathcal{P}$, there always exists a code $\mathcal{C}$ with $L_{\mathcal{C}}(\mathcal{P}(x))$ bits $= -\log(\mathcal{P}(x))$ bits.

The earliest version of the MDL principle is the two-part version. Given multiple models, the MDL principle states to chose the model $\mathcal{M}$ which minimizes the total description length

$$L_{\mathcal{C}_1}(y^m \mid \theta, \mathcal{M}, x^m) + L_{\mathcal{C}_2}(\theta \mid \mathcal{M}) \tag{2.6}$$

The first part of the term is simply $-\log(\mathcal{P}(y^m|\theta, x^m))$. For the definition of the second part no clear rules exist. However, arbitrary codes are dangerous as they will perform differently well with different models. Thus, it is important to choose a code $L_{\mathcal{C}_2}$ that is identical for all concrete models with the same input sample size n. Still, it is difficult to design codes that work well on small sample sizes. A reasonable method to build $L_{\mathcal{C}_2}$ is to exploit the natural structures of the models under consideration by using a fixed-length code that encodes the parameters θ. Different extensions and modifications of the two-part version of the MDL principle exist that lead to a non-arbitrary model selection criterion.

2.2. Biological and Technical Background

This section gives information about the biological and technical background of cancer and genetic mutations in Subsection 2.2.1, about next-generation sequencing techniques in Subsection 2.2.2, and about the subclonal reconstruction of cancer samples in Section 2.3.

2.2.1. Cancer and Genetic Mutations

Cancer is a set of genetic diseases that are caused by the accumulation of genetic mutations [84]. It is characterized by uncontrolled proliferating cells, which eventually gain the ability to metastasize and invade other tissues [58]. Malignant tumors are masses of cancerous cells, whereas benign tumors harbor non-cancerous and non-invasive cells. More than 100 different tissues can form cancers, each of which has a typical appearance and biological behavior.

Mutations arise due to different internal and external mutagenic factors. If not being repaired or destroyed by cell internal mechanisms, a mutation can lead to the formation of cancer if it activates an oncogene, which drives the growth and division of cells, or if it inactivates a tumor suppressor gene, which inhibits cell growth and division. Such a mutation is called a *driver* mutation. Mutations that do not bring growth or division advantages are called *passenger* mutations. It has been estimated that to lead to the formation of a cancer, four to six driver mutations are necessary [58].

Heterozygous mutations are present on one copy of a chromosome pair, whereas *homozygous* mutations are present on both copies. Whether mutations are heterozygous or homozygous plays an important role in the inactivation of tumor suppressor genes [58]. The process of determining whether heterozygous mutations within one chromosome belong to the same or different copies is called *phasing* [9]. When genetic information of the parents of an individual is present, it is possible to phase the individual's mutations to the maternal and paternal chromosome copy. When genetic information of many unrelated individuals is present, it is possible to phase mutations to different haplotypes or alleles, i. e. stretches of alternative forms of genetic loci. A single individual has two alleles at a heterozygous locus. Without knowing which allele is inherited from the mother and which from the father, we refer to alleles A and B in order to differentiate them.

Mutations that are inherited from the parents of an individual are called *germline* mutations and are present in all cells of the individual. Mutations that arise during the lifetime of an individual are called *somatic* mutations and are present only in a subset of cells. They are responsible for the development of a tumor. However, germline mutations can promote this process as well [51].

Based on their size, mutations can be divided into different classes. Single nucleotide variations (SNVs), the smallest type of mutations, substitute a single base pair of the DNA. Germline SNVs that can be found in at least 1% of a population are called single nucleotide polymorphisms (SNPs) [82]. In this thesis, we summarize somatic SNVs and small somatic insertions and deletions (indels) of a couple of base pairs as simple somatic mutations (SSMs). Mutations influencing DNA segments larger than 1 kb are called structural variations [29]. They include sequence inversions, translocations, insertions and copy number changes. Copy number variations (CNVs) are germline copy number changes, and copy number aberrations (CNAs) are somatic copy number changes. Different kinds of copy number changes exist. A copy number gain duplicates a segment of a chromosome copy, thus increases the copy number of the affected genome segment. A copy number loss deletes a segment of a chromo-

some copy, thus decreases the copy number. A loss of heterozygosity (LOH) event duplicates a segment of one chromosome copy and deletes this segment on the other copy of the chromosome pair [58]. Thus, an LOH event changes only the allele-specific copy numbers; globally, it is copy number neutral.

2.2.2. Next-Generation Sequencing Techniques

Around 2005, the field of DNA sequencing was revolutionized by the introduction of next-generation sequencing (NGS) sequencing machines [34]. In comparison to Sanger sequencing platforms, these techniques, also called second-generation sequencing, enabled genome sequencing at large scale with high throughput, low costs and a decent error rate. While the produced DNA sequencing reads were only 35 to 50 bases long in the beginning [60], nowadays it is possible to sequence genome fragments of up to 250 and 400 bases length [34]. Recently, sequencing long DNA fragments of 8 to 20 kb and even up to 200 kb has become available with the advent of third-generation sequencing machines.

During sequencing, substitution and indel errors can occur. Also, certain regions, such as GC- or AT-rich regions, can be underrepresented. Different sequencing platforms are prone to different error types [34].

Bulk-Sequencing. In bulk-sequencing, all cells of a sample are processed together, thus their DNA fragments, that are to be sequenced, mix. Given the sequence of two reads, we do not know whether they were derived from the same cell or not. This is not problematic when genetic differences between cells are not of interest, e. g. when the goal is to create a reference genome of a specific species [23]. However, when genetic differences between cells are important, e. g. when investigating intratumor heterogeneity in cancer, bioinformatic approaches are needed to decompose the genetic information.

Single-Cell Sequencing. In contrast to bulk-sequencing, single-cell sequencing allows to sequence the genome of a single cell. Thus, it is a useful approach to investigate intratumor heterogeneity in cancer but can also be applied in other contexts where the difference between cells of a sample is of interest, e. g. in metagenomics [65]. Single-cell sequencing is a young technique, with the first report of sequencing single human cancer cells being published in 2011 [70, 71]. The technique still suffers from significant challenges [31], such as the unbiased isolation of single cells. Another chal-

lenge poses the amplification of DNA, where common errors include loss of coverage or non-uniform coverage, biases towards one allele as well as sequence errors by incorporating wrong bases during amplification. An additional disadvantage of single-cell sequencing compared to bulk-sequencing is its high cost.

2.2.3. Detecting Somatic Mutations

To differentiate somatic from germline mutations of an individual, a cancer sample as well as a normal sample, which contains healthy cells, need to be sequenced and compared. For comparison and detection, the reads of both samples are mapped onto a reference genome and analyzed by bioinformatic software. In the following, we give a short overview how CNAs and SSMs can be identified.

Detecting Copy Number Aberrations. Methods detecting CNAs consist of the three main steps raw copy number inference, segmentation and copy number classification [4]. In the first step, the raw copy numbers are inferred, either based on the read counts or the coverage ratio between the mapped reads of the tumor and the normal sample. To account for sequencing and mappability biases, that could influence the copy number estimation, some methods perform GC-content normalization as well as mappability bias correction [8, 40]. Other methods rely on the assumption that both the tumor and the normal sample are influenced equally by such biases and thus their direct comparison corrects for these biases [52, 86]. In the second step, adjacent loci of similar raw copy number are combined into segments and the final copy number for each segment is calculated. Finally, segments are classified into copy number gains or losses.

When relying only on the coverage, copy number neutral LOH events cannot be detected. To detect them, the allele frequencies of heterozygous SNPs are used. Two methods that infer allele-specific copy numbers on NGS data are Sequenza [28] and Falcon [15]. For each segment, they calculate the major and the minor copy number.

Detecting CNAs in cancer samples is complicated by the presence of normal cells as well as cancer cells of other lineages that do not contain CNAs in a certain segment. Methods that consider intratumor heterogeneity are presented in Subsection 2.3.3 on page 24.

Detecting Simple Somatic Mutations. The general procedure to detect SSMs from matched tumor-normal samples consist of the four steps read processing, read map-

ping, variant calling and post-filtering [87]. First, low quality reads from the sequencing are removed as they often contain errors, which could later lead to false positives. Then the reads are mapped onto a reference genome. Based on the mapping, the number of reference and variant reads, R_j and V_j, can be derived for each site j. The variant allele frequency (VAF) p_j can then be computed as follows:

$$p_j = \frac{V_j}{R_j + V_j}. \tag{2.7}$$

Theoretically, given enough reads, each SSM can be observed via its VAF. In practice, however, low variant mutations, that are present only in a subset of cells, are hard to detect. Distinguishing them from artifacts caused by sequencing noise or mapping errors is difficult. Methods, such as VarScan 2 [52], SAMtools [59], Strelka [80], MuTect [16], decide based on statistical tests, simple decision rules, or probabilistic models whether a variant is present at a site or whether the changed VAF is an artifact. In the last step of SSM detection, the found variants between tumor and normal sample are compared and only the variants unique to the tumor sample are kept.

2.3. Subclonal Reconstruction of Cancer Samples

This section provides important information about subclonal reconstructions of cancer samples. In Subsection 2.3.1, the clonal evolution theory and research in the field of intratumor heterogeneity are presented. In Subsection 2.3.2, we formalize the problem description of lineage-based and population-based subclonal reconstructions. At the end, in Subsection 2.3.3, important subclonal reconstruction concepts and different methods are introduced.

2.3.1. Clonal Evolution Theory and Intratumor Heterogeneity

In 1976, Nowell [74] presented one of the first models for the evolution of tumor cells originating from a single cell of origin (see Figure 2.2). According to the model, a tumor is initiated when a normal, healthy cell acquires mutations that give growth advantages over adjacent cells, the neoplastic proliferation begins. Because the neoplastic cells are genetically instable, more mutations arise and accumulate with time. This leads to new genetic variants of which most do not survive because of fitness disadvantages. However some cells are favored by clonal selection and can become the

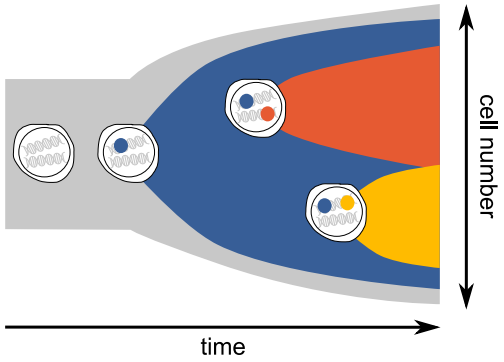


Figure 2.2.: Clonal evolution over time. Before tumor initiation, all cells are healthy and their DNA does not have mutations (gray areas and first shown cell). Then one cell acquires a mutation (blue dot) that gives growth advantages over healthy cells. With time, more mutations accumulate of which some (red and yellow dots) are favored by clonal selection, leading to new lineages and intratumor heterogeneity.

ancestors of new lineages, leading to tumor progression and resulting in intratumor heterogeneity.

In 1999, Cahill *et al.* [10] described tumor evolution as a Darwinian process, similar to Nowell. They emphasized the role of genetic instability as a driving force of tumor progression. When cells have no or only little genetic instability, they are not able to adapt to a changing environment. In contrast, when the genetic instability is too high, damaging mutations accumulate, resulting in cell death. However, the "just right" level of genetic instability leads to a variety of mutated cells of which some are able to adapt to selective pressures, giving adapted proliferating cells the ability to expand in clonal waves.

Mutations in the DNA of tumor cells can be classified based on their appearance in tumor evolution. Mutations that occur early before subclonal diversification are present in all tumor cells and are called *clonal* or *trunk* mutations. Mutations that appear after subclonal diversification are only present in a subset of tumor cells and are called *subclonal* or *branch* mutations. Subclonal mutations can be shared by different subclonal populations (subpopulations) or can be unique to a single one. Based on their appearance in tumor evolution, lineages and populations can be clonal or subclonal as well.

Advances in sequencing technologies enabled researchers to investigate genomic intratumor heterogeneity in many different cancer types and with different approaches, using bulk-sequencing or single-cell sequencing data, investigating a single sample or temporarily and/or spatially distinct samples of the same patient [66]. By studying

different tumor samples, Campell *et al.* [12] discovered parallel and convergent evolution in different pancreatic cancer metastases and genetic heterogeneity of metastasis-initiating cells. They also observed that the amount of clonal and subclonal mutations was considerably different across patients. Ding *et al.* [22] investigated tumor evolution over time in acute myeloid leukemia of eight patients by comparing mutations of primary tumors and relapse samples. They found two major relapse patterns of clonal evolution: One in which the clonal population of the primary tumor accumulated more mutations and became the population responsible for relapse, and a second one in which a subpopulation survived the initial therapy and became the dominant population after relapse. Nik-Zainal *et al.* [72] analyzed the clonal evolution of breast cancer based on single samples of 21 patients and found that all tumors harbored a dominant subclonal lineage which contains more than 50 % of the tumor cells. Another finding was that the last common ancestor of all cancer cells appeared early in molecular time and that much time is spent on diversification of subclonal lineages. One of the first studies to investigate tumor heterogeneity by means of single-cell sequencing was performed in 2011 by Navin *et al.* [70] and observed punctuated clonal expansions of two breast cancers by comparing multiple samples.

Intratumor heterogeneity of tumors cannot always be explained by positive selection and clonal expansion in waves. For some tumors, growth patterns of neutral evolution instead of positive selection were observed [19,66]. Also, mutations do not accumulate sequentially over time in all tumors. In some tumors, mutations arise in form of punctuated bursts [19,66,70]. Within the lifetime of a tumor, it is possible that different evolutionary models are active: Studies on breast cancers have shown that CNAs are punctually gained at the beginning of the neoplastic proliferation while SSMs accumulate sequentially during tumor progression [19].

Intratumor heterogeneity challenges the success of cancer therapies. Subclones that harbor drug resistances are positively selected when the drug is applied to the patient and can expand in clonal waves to become dominant and result in a failure of therapy [35]. To prevent this, different drugs can be combined [3]. Knowledge about the composition of subclones and their mutations can help to identify drugs that target relevant clonal and/or subclonal mutations and thus to develop a personalized treatment [19]. Further, intratumor heterogeneity can be used as a predictive biomarker informing about therapy outcome, since it was shown in different studies that a high level of intratumor heterogeneity correlates with poor outcome of therapy [19].

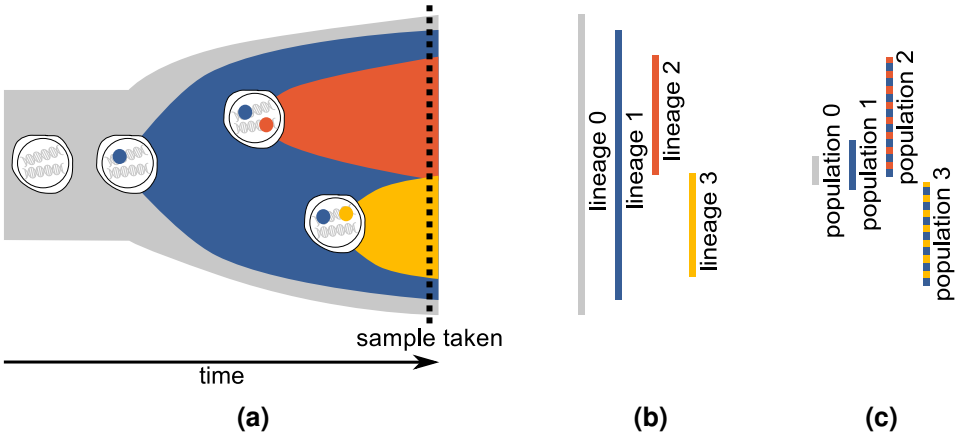


Figure 2.3.: Clonal evolution over time showing lineages and populations. (a) Clonal evolution over time with three mutations. At a certain time point, a sample is taken. (b)/(c) Four lineages and four populations are present in the sample. The colorful bars show the amount and genotype of cells that belong to the lineage or population. (b) Lineage 0 is the normal lineage without any mutations to which all cells in the sample belong. The blue, the red and the yellow mutations lead to the formation of lineages 1, 2 and 3. (c) Each population has a unique genotype that is distinct from the genotype of the other populations. Population 0 is the normal population, which does not contain any mutations.

2.3.2. Formalized Problem Description

While we have used the terms *lineage*, *population* and *subclonal reconstruction* before, we now formally define them in this subsection as well as the *subclonal reconstruction problem*.

Definition 1. *A* lineage *k comprises all cells that descended from the same founder cell. It is defined by a set $\mathcal{S}_k$ of (possibly phased) mutations, a frequency $\phi_{k,n}$ and the relationships to other lineages. Mutations in $\mathcal{S}_k$ arose in lineage k and either lie on the maternal or paternal chromosome copy. The lineage frequency $\phi_{k,n}$ is the frequency of cells belonging to lineage k at the time point a sample n was taken. Lineage k can be ancestor or descendant of another lineage k' or it can be in no ancestor-descendant relationship to it.*

A lineage can also be described as a subtree in a phylogenetic tree and is sometimes referred to as a *clade*. When a lineage k' is a descendant of a lineage k, it is also part of lineage k.

All cancerous cells can be traced back to a healthy cell. The healthy or *normal* lineage is ancestor of all other lineages, which we call *non-normal* or *cancerous* lineages. It has a frequency of 1 across all samples and does not contain somatic mutations.

Figure 2.4.: Lineage-based and population-based subclonal reconstructions. Circles show lineages or populations. Arrows show relationships between lineages and populations, pointing from the ancestor to the descendant. (a) Lineage-based subclonal reconstruction with four lineages and inferred frequencies $\phi_0, \ldots, \phi_3$. Unphased mutations are assigned to the lineage in which they arose. The genotype of the lineages is not shown. (b) Population-based subclonal reconstruction with four populations and inferred frequencies $\eta_0, \ldots, \eta_3$. The genotype based on the mutation assignment is shown for each population.

Figures 2.3a and 2.3b show an example of clonal evolution over time with lineages of a taken sample.

Given M mutations from N bulk-sequencing tumor samples of the same patient and the number of lineages K, we define a *lineage-based subclonal reconstruction* as follows:

Definition 2. *A* lineage-based subclonal reconstruction *contains M (possibly phased) mutations that are assigned to K lineages, inferred frequencies ϕ of the K lineages across N samples and inferred relationships between the lineages.*

The inferred relationships between the lineages describe a phylogenetic tree. Mutations that are assigned to a lineage k are inherited by its descendants. The normal lineage is included in the reconstruction. A lineage-based subclonal reconstruction is shown in Figure 2.4a.

Definition 3. *A* population *k comprises all cells that have the same genotype. It is defined by a set $\mathcal{S}'_k$ of (possibly phased) mutations, a frequency $\eta_{k,n}$ and the relationships to other populations. All mutations that can be found in the cells of population k are in the set $\mathcal{S}'_k$ and lie either on the maternal or on the paternal chromosome copy. The population frequency $\eta_{k,n}$ is the frequency of these cells at the time point a sample n is taken. Population k can be ancestor or descendant of another population k', or it can be in no ancestor-descendant relationship to it.*

The normal population, which does not contain any mutations, is ancestor of all cancerous. A *vestigial* population is a population with a frequency of 0 [21]. Such a population is not present anymore at the time point of tumor sampling but gave rise to mutations which were inherited by other populations that are still present. Figure 2.3c shows populations of a tumor sample.

Definition 4. *A* population-based subclonal reconstruction *contains M (possibly phased) mutations assigned to K populations, inferred frequencies η of K populations across N samples and inferred relationships between the populations.*

The inferred population frequencies describe a phylogenetic tree. Mutations that are assigned to a population k are inherited by its descendants. Population frequencies sum up to 1 per sample. The normal population is included in the reconstruction. A population-based subclonal reconstruction is shown in Figure 2.4b.

Lineage-based subclonal reconstructions with K lineages can be converted to population-based subclonal reconstructions with K populations and vice versa. Here, we explain only the conversion from a lineage-based subclonal reconstruction to a population-based one. Relationships that are inferred for lineage k, with $k \in \{0, \ldots, K-1\}$, are used for population k. Mutations that are assigned to lineage k are assigned to population k and its descendants. To compute the population frequency $\eta_{k,n}$ for all $n \in \{0, \ldots, N-1\}$, the following relation between lineage and population frequencies is used:

$$\phi_{k,n} = \eta_{k,n} + \sum_{k' \in \mathcal{D}_k} \eta_{k',n},$$

where $\mathcal{D}_k$ comprises all descendants of population k.

Mutations that are often used for subclonal reconstructions are SSMs and CNAs [54]. SSMs can immediately be assigned to the lineages. CNAs can either be given directly and immediately be assigned to the lineages as well, or they can be given indirectly via copy number information of genome segments and have to be inferred before lineage assignment. When no CNAs are used for the subclonal reconstructions, all mutations are unphased.

For a subclonal reconstruction r, a likelihood $\mathcal{L}$ can be computed.

Problem 1. *Given M mutations and N bulk-sequencing tumor samples of the same patient, the* lineage-based subclonal reconstruction problem *is to find the correct number of lineages K and the lineage-based subclonal reconstruction with K lineages that maximizes the likelihood $\mathcal{L}$.*

Problem 2. *Given I genome segments with copy number information, J SSMs and N bulk-sequencing tumor samples of the same patient, the* lineage-based subclonal reconstruction problem with copy number inference *is to infer CNAs and solve the lineage-based subclonal reconstruction problem.*

The *population-based subclonal reconstruction problem* and the *population-based subclonal reconstruction problem with copy number inference* are defined analogously.

2.3.3. Subclonal Reconstruction Concepts and Methods

Given K lineages, it is possible to build K^{K-1} different rooted trees[3], in which each lineage is represented by a node. However, not all of these trees represent valid subclonal reconstructions. When the lineage frequencies across N samples are inferred, trees can be excluded from the solution space of subclonal reconstructions with the lineage precedence rule [73], the sum rule [47] and the crossing rule [47].

Lineage Precedence Rule. The lineage precedence rule imposes a partial ordering on the lineages given their frequencies. It states that a lineage k that is an ancestor of a lineage k' has a greater or equal frequency than k' across all samples. Thus, a lineage k'' with a lower frequency than lineage k in at least one sample cannot be the ancestor of k.

Sum Rule. The sum rule relates the frequency of a lineage k with the frequencies of its direct descendants or children in each sample. It states that the frequency of lineage k is higher or equal than the sum over the lineage frequencies of its children. If the frequency of lineage k is equal to the frequency sum of its children, the corresponding population k is a vestigial population. For a single sample, a linear phylogeny is always consistent with the sum rule. The sum rule is also known as children sum to parents condition [41], sum condition [24], lineage divergence rule [73] or described as following the pigeon hole principle [72].

Crossing Rule. The crossing rule can exclude ancestor-descendant relationships between lineages when multiple samples are used. When lineage k has a higher frequency than lineage k' in one sample n and lineage k' has a higher frequency than lineage k in another sample n', an ancestor-descendant relationship between the two is not possible. Using more samples increases the chance of observing a "crossing" of lineage frequencies and thus the evidence to rule out an ancestor-descendant relationship between lineages that lie on different branches of the true underlying phylogenetic tree. The crossing rule follows directly from the lineage precedence rule. It is also called *fork rule* [54].

[3]The number of labeled trees with K nodes is K^{K-2} [13]. Thus, the number of rooted trees is K^{K-1}.

In most cases, applying the lineage precedence, the sum and the crossing rule will not result in a single possible tree. To further reduce the tree solution space, the strong parsimony assumption [21] is made by [41, 85].

Strong Parsimony Assumption. The strong parsimony assumption leads to reconstructions with a maximal number of vestigial populations. In these reconstructions, only a small number of populations is inferred to be present at the time point where the tumor samples are taken. Hence, subclonal reconstruction methods using the strong parsimony assumption favor branching trees over linear trees.

When M mutations should be assigned to $K-1$ cancerous lineages and each mutation can be assigned to as many lineages as given, $(2^{K-1}-1)^M$ different assignments exist. To reduce this number of possible assignments, the weak parsimony assumption [21] and the infinite sites assumption are commonly made.

Weak Parsimony Assumption. The weak parsimony assumption expects two SSMs to have similar VAFs when they arise in the same lineage. Thus, SSMs with similar VAFs are assigned to the same lineage. However, it is possible that two or more distinct lineages have similar frequencies and hence, following the weak parsimony assumption, their SSMs are falsely assigned to the same lineage. Using multiple samples can help to reduce this problem because the lineages can have different frequencies in the separate samples, which leads to different VAFs across samples. It is also possible to not combine SSMs based on their VAFs as done in Strino *et al.* [85]. However, this approach increases the complexity of the subclonal reconstruction problem and is only feasible with a small number of SSMs.

Infinite Sites Assumption. The infinite sites assumption is used to reduce the number of possible mutation assignments to $(K-1)^M$. It was originally presented in 1969 by Kimura [50] for Mendelian populations. It assumes that the genome consists of many (practically infinite) sites, so that each site is hit by a mutation at most once, and once it is hit, the mutation will not revert back to the original state. Applied to cancer genomics, the infinite sites assumption implies that no position in the genome is influenced by more than one mutation. Each mutation arises only once during tumor evolution and does not get lost, it is inherited to all descendant lineages. When building a subclonal reconstruction, a mutation can only be assigned to one lineage.

Using the infinite sites assumption for SSMs seems reasonable: The human genome consists of 3 billion nucleotides and cancers can contain tens of thousands of SSMs [49]. The chance that one nucleotide is hit by a mutation is rather small and the chance that a mutated site is hit by a mutation a second time is even smaller. Thus, the infinite sites assumption is used by many subclonal reconstruction methods [21,24,46,47,64].

However, the validity of the infinite sites assumption for tumor sequencing data was not investigated until 2017, when Kuipers *et al.* [55] developed a statistical framework for single-cell sequencing data. The authors found violations of the infinite sites assumption in eleven of twelve datasets, which included parallel mutations of the same site, back mutations to the original state and losses of mutation containing alleles. They conclude that the infinite sites assumption has to be used with caution.

A modified version of the infinite sites assumption has been developed that allows the loss of SSMs when they lie on alleles that get deleted [21,46,67]. In the supplementary material to their paper, McPherson *et al.* [67] write that their model can be understood as "a simplified version of the stochastic Dollo process" and Kuipers *et al.* [55] describe this approach as "Dollo parsimony with loss". This concept of Dollo parsimony states that the evolution of a trait is a rare event that happens only once during evolution and is based on an interpretation of Le Quesne [27,56]. In contrast, the loss of such a trait is not as rare and can happen multiple times on different branches of the phylogenetic tree.

For CNAs, the infinite sites assumption is too restrictive. Unlike for SSMs where only the two states "mutation present" and "mutation absent" exist, multiple copy number states exist for CNAs. Moreover, CNAs can influence up to 85 Mb [6], thereby increasing the chance of a CNA overlap, and thus of nucleotides that are influenced multiple times. To overcome the multi-state problem, El-Kebir *et al.* [25] use the infinite alleles assumption. In this assumption, a site is allowed to mutate to multiple states but each mutation to a state is allowed only once. This leads to the construction of a multi-state perfect phylogeny [38]. In contrast, Jiang *et al.* [46] use a different interpretation of CNA events when they apply the infinite sites assumption: If CNAs have the same copy number state and the exact breakpoints across all samples, they are treated as a single event, that can be present or absent. Overlapping CNAs for which this is not the case, are treated as separate events that cannot conflict with the infinite sites assumption.

The first automated subclonal reconstruction methods infer the genotype and the proportions of populations or lineages but not their relationships. They work either

Table 2.1.: Methods Table. The input and output of different subclonal reconstruction methods are shown, as well as the rules they use.
N: no, Y: yes, n/a: not applicable
mult.: multiple, freqs.: frequencies, lin. prec. rule: lineage precedence rule, pars.: parsimony, ISA: infinite sites assumption, IAA: infinite alleles assumption

	SSMs	CNA info	mult. samples	infers CNAs	infers freqs.	builds trees	lin. prec. rule	sum rule	crossing rule	strong pars.	weak pars.	ISA
THetA [75]	N	Y	N	Y	Y	N	n/a	n/a	n/a	n/a	Y	n/a
TITAN [39]	N	Y	N	Y	Y	N	n/a	n/a	n/a	n/a	Y	n/a
Clomial [88]	Y	N	Y	N	Y	N	n/a	n/a	n/a	n/a	Y	N
SciClone [69]	Y	Y	Y	N	Y	N	n/a	n/a	n/a	n/a	Y	Y
PyClone [78]	Y	Y	Y	N	Y	N	n/a	n/a	n/a	n/a	Y	Y
CloneHD [30]	Y	Y	Y	Y	Y	N	n/a	n/a	n/a	n/a	Y	N
rec-BTP [41]	Y	N	N	N	N	Y	Y	Y	n/a	Y	n/a	Y
SCHISM [73]	Y	Y	Y	N	N	Y	Y	Y	Y	N	n/a	Y
TrAp [85]	Y	Y	N	N	N	Y	Y	Y	n/a	Y	no	N
PyDollo [67]	Y	Y	Y	Y[1]	N	Y	n/a[2]	n/a[2]	n/a[2]	N	n/a[3]	Y[4]
AncesTree [24]	Y	N	Y	N	Y	Y	Y	Y	Y	N	Y	Y
CITUP [64]	Y	N	Y	N	Y	Y	Y	Y	Y	N	Y	Y
PhyloSub [47]	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y
PhyloWGS [21]	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y[4]
Spruce [25]	Y	Y	Y	N	Y	Y	Y	Y	Y	N	N	IAA
Canopy [46]	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y[4]

[1]: CNAs are inferred only at the inner nodes of the tree, [2]: PyDollo does not work with frequencies, [3]: clusters are already given, [4]: Dollo parsimony with loss

only with CNAs [39,75], or only with SSMs [88], or with both [30,69,78]. With the availability of population and lineage genotypes and proportions, methods were realized that infer phylogenetic trees. These methods are either based on SSMs alone [41], or combine information of CNAs and SSMs [67,73,85]. Recently, complete subclonal reconstruction methods were established that jointly infer the genotype and the proportions of populations or lineages and arrange them in a phylogenetic tree. These methods use either only SSMs [24,64] or SSMs and CNAs [21,25,46,47]. An overview of all these methods can be found in Table 2.1.

In the following, we will explain how population and lineage frequencies can be inferred from CNA and SSM information.

Inferring Population Frequencies From CNA Information. Current methods that work with CNA information infer non-negative integer copy numbers and assign them to populations. The observed CNA information, e. g. the total average copy number c, results from the combination of a) the integer copy numbers of populations *without* the CNA and their frequency, and b) the integer copy numbers of populations *with* the CNA and their frequency. A CNA that arises in population k is inherited to its descendants. Thus, all population relationships need to be known in order to compute the frequency of all populations that contain the CNA, which equals the frequency ϕ_k of lineage k. Given only the observed CNA information, the two unknown variables ϕ_k and C_k, which is the copy number of population k and its descendants, have to be inferred simultaneously:

$$c = 2 \cdot (1 - \phi_k) + C_k \cdot \phi_k \,. \tag{2.8}$$

When CNAs are inferred, three different observations can be used as input. First, a comparison between the coverage of the tumor and the normal genome, or a comparison between the total and the haploid coverage of the tumor genome allow to compute the total average copy number c. This information allows the detection only of CNAs that change the copy number of the genome. To find copy number neutral LOH events, the allele frequencies of heterozygous SNP loci can be used as second input together with the coverage differences [39]. A third observation, which can be used as input, are average allele-specific copy numbers of genome segments, which allow the detection of copy number neutral LOH events as well [46].

Using a probabilistic model, the coverage can be modeled with a Poisson or negative binomial distribution and the allele frequencies with a binomial or beta-binomial distribution [30]. The average allele-specific copy numbers can be modeled with a normal distribution [46].

Inferring Lineage Frequencies From SSMs. Current methods that work with SSMs assign them to lineages and infer the frequencies of these lineages. As input information, the variant and reference counts, V and R, of the SSMs are given, which allow to compute their VAFs p (Equation 2.7). When an SSM j lies within a copy number

neutral genome segment i, the frequency ϕ_k of the lineage k in which j arose, can be computed as

$$p_j = \frac{s_j}{c_i} = \frac{\phi_k}{2},$$

where s_j is the copy number of the SSM and c_i the total average copy number of segment i, which equals 2.

Since SSM data contain noise, the observed VAFs cluster around the true VAFs. The uncertainty in the read counts can be expressed with a probabilistic model, by modeling the read counts with a binomial distribution [30,47]. To allow more variance in read count, some methods use a beta-binomial distribution [30,78].

When an SSM j does not lie in a copy number neutral segment, the CNA in the segment i of the SSM influences the VAF p_j. The CNA changes the copy number c_i of the segment and potentially also the copy number s_j of the SSM. Thus, in presence of CNAs, computing the lineage frequencies requires knowing the influence of CNAs on the SSMs. This is why some methods exclude regions with CNAs [24,41]. Other methods make the simplified assumption that all cells containing the SSM are equally influenced by CNAs [47,69,78]. Only recent methods model cells with SSMs to be differently influenced by CNAs [21,25,46].

Influence of CNAs on SSMs. When an SSM j lies in a genome segment i with a CNA, this CNA influences its VAF p_j by changing the copy number c_i (Equation 2.8) and potentially also the copy number s_j of the SSM. The influence on s_j depends on the ancestral relationship between the CNA and the SSM and their phase. Only when the CNA is descendant to the SSM and belongs to the same chromosome copy, it changes s_j. Assuming a) that the SSM j arises in lineage k^* and belongs to phase A, b) that the CNA arises in population k, leading to a copy number of C_{A_i} and a copy number C_{B_i}, c) that all cells containing the CNA belong to lineage k, and that d) lineage k^* is an ancestor of lineage k, the copy number s_j can be computed as

$$s_j = (\phi_{k^*} - \phi_k) + C_{A_i} \cdot \phi_k \,. \tag{2.9}$$

If the CNA is not phased to allele A, C_{A_i} is 1, and s_j is simply ϕ_{k^*}, so it is not influenced. Equation 2.9 can be extended to allow the influence of multiple CNAs on s_j. When CNAs are ancestral to SSMs or appear on different branches of the phylogeny, they cannot influence their copy numbers.

Incorporating the influence of CNAs on SSMs in a probabilistic model allows to detect ancestor-descendant relationships between populations in cases, in which the likelihood can be increased by modeling an ancestor-descendant relationship between populations and allowing a CNA to change the copy number of an SSM.

Now, we will introduce PhyloWGS and Canopy, two methods that we will compare Onctopus to.

PhyloWGS. PhyloWGS [21] was the first fully automated method to build a complete subclonal reconstruction based on SSMs and CNAs, without making the unrealistic assumption that all cells containing an SSM are equally influenced by CNAs. As input, it requires the variant and total read counts of SSMs, as well as major and minor copy numbers of CNAs with their population frequencies. Thus, it does not infer CNAs.

PhyloWGS uses an MCMC procedure that samples phylogenetic trees from the model posterior. It is parameter free and runs in one MCMC chain. The first 1000 iterations are burned-in. Then, the resulting subclonal reconstructions with different numbers of populations of the next 2500 iterations are reported. If a single subclonal reconstruction is needed, the authors suggest using the reconstruction that maximizes the likelihood.

Canopy. Canopy [46] is the first subclonal reconstruction method that works with SSMs and infers CNAs. As input, it needs the variant and total read counts of SSMs, as well as the average allele-specific copy numbers of genome segments. Canopy applies an MCMC procedure and computes the posterior distribution over configurations. A configuration summarizes all subclonal reconstructions that have the same relationships between populations and the same mutation assignment. The population frequencies can be different. The number of chains of the MCMC, the number of burn-ins and total iterations can be adjusted via parameters.

Canopy constructs binary trees and assigns mutations to the inner nodes and leaves. Not all nodes have to receive mutations. Nodes with mutations represent populations, nodes without mutations do not represent populations and can be collapsed to their parent node. An exception is the leftmost leaf, which represents the normal population and thus does not contain any mutations. Populations at the inner nodes to which no leaf is collapsed are vestigial populations. The number of non-vestigial populations, i. e. the number of leaves, is specified via a parameter. Since the MCMC sampling can

lead to nodes that do not represent populations, the total number of populations can differ for the same number of non-vestigial populations. To determine the underlying number of non-vestigial populations, Canopy compares reconstructions with different population numbers using BIC.

Chapter 3

A New Lineage-Based Subclonal Reconstruction Model

In this chapter, we present our probabilistic model that solves the lineage-based subclonal reconstruction problem with copy number inference.

As input, our model takes average allele-specific copy numbers of I genome segments and the variant read counts as well as the reference read counts of J SSMs in N bulk-sequenced tumor samples of the same patient. Furthermore, the number of lineages K is given as input.

We apply the lineage precedence rule, the sum rule and the crossing rule to exclude invalid subclonal reconstructions from the solution space of subclonal reconstructions (compare Subsection 2.3.3). To reduce the number of possible mutation assignments, we apply a version of the infinite sites assumption that permits loss of SSMs through copy number losses. We do not use the infinite sites assumption for CNAs. Our probabilistic model makes implicit use of the weak parsimony assumption but we also offer the possibility of using the weak parsimony assumption explicitly by clustering SSMs based on their VAFs.

Our model solves the lineage-based subclonal reconstruction problem with copy number inference and thus outputs inferred CNAs, that are phased to allele A or B and assigned to the given lineages. SSMs are phased relatively to CNAs and are assigned to the lineages as well. Also, the lineage frequencies across all samples and the lineage relationships are inferred.

Unique features of our model compared to other models that solve the subclonal reconstruction problem are the modeling of CNAs and the way we deal with ambiguity in the input data. Instead of modeling *absolute* copy numbers of populations, we

model *relative* copy numbers, so copy number changes, of lineages. This enables us to combine CNAs and SSMs in a lineage-based model and improves the handling of ambiguity: Instead of inferring the relationships between all lineages and having to decide for all ancestor-descendant relationships whether they are present or absent, we infer only the relationships that can be observed in the data. The other relationships are modeled as ambiguous relationships.

In Section 3.1, we present the likelihood function of our model. Then, in Section 3.2, we explain the different model components, the lineage frequencies, the lineage relationships, and the assignment of CNAs and SSMs. We also detail the rules that define our model. In Section 3.3, we present an MILP formulation of our model. Afterwards, in Section 3.4, we analyze the complexity of the optimization. Finally, in Section 3.5, we show how we determine the underlying lineage number of the input data.

If not stated otherwise, the following indices have the following ranges and meanings:

- $i \in \{0, \ldots, I-1\}$ indexes segments,
- $j \in \{0, \ldots, J-1\}$ indexes SSMs,
- $n \in \{0, \ldots, N-1\}$ indexes tumor samples,
- $\alpha \in \{A, B\}$ indexes alleles.

3.1. The Likelihood Function

Our likelihood function $\mathcal{L}$ for a subclonal reconstruction r is composed of two main parts that consider the likelihood values for CNAs and SSMs.

As done in Canopy, we assume that average allele-specific copy numbers follow a normal distribution with mean $c_{\alpha_{i,n}}$ and standard deviation $\sigma_{\alpha_{i,n}}$ for allele α of segment i in sample n. Thereby, $c_{\alpha_{i,n}}$ is the observed average allele-specific copy number and $\sigma_{\alpha_{i,n}}$ is the standard error, which is either given by the copy number detection method or calculated as shown in Section 5.2 on page 94. Our model infers the average allele-specific copy number $\hat{c}_{\alpha_{i,n}}$ from the input data (later shown in Equation 3.18) with which we compute the likelihood of observing $c_{\alpha_{i,n}}$ with $\sigma_{\alpha_{i,n}}$ as

$$\mathcal{N}(\hat{c}_{\alpha_{i,n}} \mid c_{\alpha_{i,n}}, \sigma_{\alpha_{i,n}}) = \frac{1}{\sqrt{2\pi\sigma^2_{\alpha_{i,n}}}} e^{-\frac{(\hat{c}_{\alpha_{i,n}} - c_{\alpha_{i,n}})^2}{2\sigma^2_{\alpha_{i,n}}}}. \tag{3.1}$$

As done in PyClone and CloneHD, we assume that the variant read counts of SSMs follow a beta-binomial distribution with $D_{j,n}$ trials and $V_{j,n}$ successes, and parameters $\hat{\alpha}_{j,n}$ and $\hat{\beta}_{j,n}$, as defined in the following, for SSM j in sample n. Thereby, $V_{j,n}$ is the observed variant and $R_{j,n}$ the observed reference read count, with $D_{j,n} = V_{j,n} + R_{j,n}$. Our model infers the VAF $\hat{p}_{j,n}$ from the input data (later shown in Equation 3.26), from which we compute

$$\hat{\alpha}_{j,n} = \hat{p}_{j,n} \cdot s_{\text{SSM}} \tag{3.2}$$

and

$$\hat{\beta}_{j,n} = (1 - \hat{p}_{j,n}) \cdot s_{\text{SSM}}, \tag{3.3}$$

where s_{SSM} is the overdispersion parameter of the beta-binomial distribution. Given the inferred VAF $\hat{p}_{j,n}$, we compute the likelihood of observing $V_{j,n}$ variant reads with $D_{j,n}$ total reads as

$$\text{Beta-Bin}(\hat{\alpha}_{j,n}, \hat{\beta}_{j,n} \mid D_{j,n}, V_{j,n}) = \binom{D_{j,n}}{V_{j,n}} \frac{Beta(V_{j,n} + \hat{\alpha}_{j,n}, D_{j,n} - V_{j,n} + \hat{\beta}_{j,n})}{Beta(\hat{\alpha}_{j,n}, \hat{\beta}_{j,n})}, \tag{3.4}$$

where *Beta* is the beta function.

We assume that CNAs and SSMs are *independent*, conditioned on the true lineage frequencies. Thus, we compute the combined log-likelihood $\mathcal{L}'$ for a subclonal reconstruction r as

$$\begin{aligned}\mathcal{L}'(r) = \mathcal{L}'(\hat{c}_A, \hat{c}_B, \hat{\alpha}, \hat{\beta} \mid c_A, c_B, \sigma_A, \sigma_B, D, V) = \quad & \sum_n \sum_i \log(\mathcal{N}(\hat{c}_{A_{i,n}} \mid c_{A_{i,n}}, \sigma^2_{A_{i,n}})) \\ & + \sum_n \sum_i \log(\mathcal{N}(\hat{c}_{B_{i,n}} \mid c_{B_{i,n}}, \sigma^2_{B_{i,n}})) \\ & + \sum_n \sum_j \log(\text{Beta-Bin}(\hat{\alpha}_{j,n}, \hat{\beta}_{j,n} \mid D_{j,n}, V_{j,n})).\end{aligned} \tag{3.5}$$

3.2. Model Components and Rules

In this section, we explain the different model components. These are the inferred lineage frequencies (see Subsection 3.2.1), the inferred lineage relationships (see Subsection 3.2.2), and the mutation assignments of CNAs (see Subsection 3.2.3) and SSMs (see Subsection 3.2.4). For the variables of each model component, we specify their allowed values and indicate which rules ensure a valid reconstruction.

3.2.1. Inferred Lineage Frequencies

Given K lineages and N samples, we store their lineage frequencies in the matrix $\phi \in \mathbb{R}^{K \times N}$, with $0 \leq \phi_{k,n} \leq 1$. The first row of ϕ contains the frequencies of the normal lineage:

$$\phi_{0,n} = 1 \; \forall \; n, 0 \leq n < N. \tag{3.6}$$

The first column of ϕ is sorted in decreasing order of frequencies, such that

$$\phi_{k,0} \geq \phi_{k',0} \; \forall \; 0 \leq k < k' < K. \tag{3.7}$$

Thus, the first column defines an order of the lineages, such that the lineages can then be indexed by row indices. The frequencies in the other columns are not forced to be sorted.

Sum Rule. We model the sum rule as

$$\sum_{k' \in \chi_k} \phi_{k',n} \leq \phi_{k,n}\,, \tag{3.8}$$

where χ_k is the set of children of lineage k and later formally defined in Equation 3.11.

3.2.2. Inferred Lineage Relationships

The inferred relationships between the K lineages are stored in the lineage relationship matrix $Z \in \{1, 0, ?\}^{K \times K}$. $Z_{k,k'} = 1$ indicates that lineage k is an ancestor of lineage k', hence an ancestor-descendant relationship between the two lineages is present. If $Z_{k,k'} = 0$, lineage k is not an ancestor of lineage k', thus the ancestor-descendant relationship between the lineages is absent. $Z_{k,k'} = ?$ shows that lineage k could be an ancestor of lineage k', consequently the ancestor-descendant relationship is ambiguous.

With the help of the lineage relationship matrix Z, we define the set of all ancestors $\mathcal{A}_k$ of lineage k as

$$\mathcal{A}_k = \{k^* \mid Z_{k^*,k} = 1, k^* < k\}. \tag{3.9}$$

We define $\mathcal{D}_k$ as the set of all descendants of lineage k:

$$\mathcal{D}_k = \{k' \mid Z_{k,k'} = 1, k' > k\}. \tag{3.10}$$

Further, we define the set of all children χ_k of lineage k as

$$\chi_k = \{k' \mid Z_{k,k'} = 1, Z_{k^\circ,k'} \in \{0,?\} \; \forall \; k^\circ \in \mathcal{D}_k \, , k^\circ < k'\}. \tag{3.11}$$

A trivial property of the lineage relationship matrix Z is that a lineage cannot be its own ancestor or descendant:

$$Z_{k,k} = 0. \tag{3.12}$$

Following the properties of subclonal reconstructions, the normal lineage 0 is the ancestor of all other lineages:

$$Z_{0,k} = 1 \; \forall \; k, 1 \leq k < K. \tag{3.13}$$

Lineage Precedence Rule and Crossing Rule. We model the lineage precedence rule and the crossing rule jointly. As the lineages are ordered by their frequencies in the first sample, we know that in the first sample, lineage k' cannot have a higher frequency than lineage k, with $k < k'$. Thus, we do not allow lineage k' to be an ancestor of lineage k, leading to the lower left triangle of Z being filled with 0s:

$$Z_{k',k} = 0 \; \forall \; k < k'. \tag{3.14}$$

Further, we do not allow lineage k to be an ancestor of lineage k' if lineage k' has a higher frequency than lineage k in at least one sample n:

$$Z_{k,k'} \in \begin{cases} \{0\} & \text{if } \phi_{k',n} > \phi_{k,n} \text{ for some } n, 1 \leq n < N, \\ \{0,1,?\} & \text{otherwise.} \end{cases} \tag{3.15}$$

Phylogenetic Tree Rules. The lineage relationships describe an underlying phylogenetic tree, from which the following two rules for lineages k, k' and k'', $0 \leq k < k' < k'' < K$, follow directly:

1. If lineage k is an ancestor of lineage k' and lineage k' is an ancestor of lineage k'', then lineage k is also an ancestor of lineage k'':

$$Z_{k,k''} \in \begin{cases} \{1\} & \text{if } Z_{k,k'} = 1 \wedge Z_{k',k''} = 1, \\ \{0,1,?\} & \text{otherwise.} \end{cases} \tag{3.16}$$

2. If both lineages k and k' are ancestors of lineage k'', then lineage k is also an ancestor of lineage k':

$$Z_{k,k'} \in \begin{cases} \{1\} & \text{if } Z_{k,k''} = 1 \wedge Z_{k',k''} = 1, \\ \{0,1,?\} & \text{otherwise.} \end{cases} \tag{3.17}$$

We can summarize that a 0 is placed in the lineage relationship matrix Z to not allow that a lineage is its own ancestor or descendant. Also, a 0 is placed when otherwise the lineage precedence rule or the crossing rule would be violated. A 1 is placed to make all cancerous lineages descendants of the normal lineage and to fulfill the phylogenetic tree rules. For all entries $Z_{k,k'}$ for which none of the above applies, the three values 0, 1 and ? are possible. We will explain later in Subsection 3.2.4 on page 37 how these possible values can influence the likelihood of our model. Also, we will show in Chapter 4 how to deal with ambiguity and how to build a single reconstruction that represents ambiguity of the input data.

3.2.3. Copy Number Aberration Assignment

We model CNAs as *relative* copy numbers, i. e. allele-specific copy number changes per lineage. These changes are stored in the two matrices ΔC_A and $\Delta C_B \in \mathbb{Z}^{I \times K}$. The values in $\Delta C_{\alpha_{i,k}}$ have the following meaning: $\Delta C_{\alpha_{i,k}} = 0$ indicates that *no* copy number change is assigned to allele α in segment i of lineage k. When a copy number *gain* is assigned to allele α in segment i of lineage k, then $\Delta C_{\alpha_{i,k}} > 0$. A copy number *loss* assigned to allele α in segment i of lineage k is indicated by $\Delta C_{\alpha_{i,k}} < 0$.

The inferred average allele-specific copy number $\hat{c}_{\alpha_{i,n}}$ is computed by adding the normal copy number of the allele, which is 1, to the average copy number change of the allele over all lineages:

$$\hat{c}_{\alpha_{i,n}} = 1 + \sum_k \phi_{k,n} \cdot \Delta C_{\alpha_{i,k}} . \tag{3.18}$$

We can then compute the inferred average copy number $\hat{c}_{i,n}$ of both alleles as

$$\hat{c}_{i,n} = \hat{c}_{A_{i,n}} + \hat{c}_{B_{i,n}} . \tag{3.19}$$

We assign each major copy number to allele A and each minor copy number to allele B. Thus, we also ensure in our model that $\hat{c}_{A_{i,n}}$ is larger than or equal to $\hat{c}_{B_{i,n}}$:

$$\hat{c}_{A_{i,n}} \geq \hat{c}_{B_{i,n}}. \tag{3.20}$$

While modeling CNAs, we do not allow copy number changes to be assigned to the normal lineage:

$$\Delta C_{\alpha_{i,0}} = 0. \tag{3.21}$$

Allowed Copy Number Changes. We do not use the infinite sites assumption for CNAs, which means that we allow multiple copy number changes in different lineages in the same segment i. However, if multiple copy number changes appear in segment i, we restrict them to be either a) all copy number gains, b) all copy number losses, or c) an LOH event in a single lineage:

$$\Delta C_{A_{i,\cdot}} \cup \Delta C_{B_{i,\cdot}} \in \left\{ \binom{\mathbb{Z}_0^+}{2K}, \binom{\mathbb{Z}_0^-}{2K}, \right. \\ \left. \left\{ \{\Delta C_{A_{i,k}}, \Delta C_{B_{i,k}}, 0\} \mid \Delta C_{A_{i,k}} \cdot \Delta C_{B_{i,k}} = -1, k \in \{1, \ldots, K-1\} \right\} \right\}. \tag{3.22}$$

As a consequence of these copy number change restrictions and because each allele can be lost only once on a branch of the phylogeny, the minimum value for $\Delta C_{\alpha_{i,k}}$ is -1.

If multiple copy number losses appear in a segment i, we do not allow an allele to get lost multiple times on the same branch of the underlying phylogenetic tree:

$$\Delta C_{\alpha_{i,k}} \in \begin{cases} \{0\} & \text{if for any } k^* \in \mathcal{A}_k : \Delta C_{\alpha_{i,k^*}} < 0, \\ \mathbb{Z} & \text{otherwise.} \end{cases} \tag{3.23}$$

Also, we ensure that if we observe a copy number gain for allele α, the model does not infer copy number losses, and if we observe a copy number loss, the model does not infer copy number gains:

$$\Delta C_{\alpha_{i,\cdot}} \in \begin{cases} \begin{pmatrix} \mathbb{Z}_0^+ \\ K \end{pmatrix} & \text{if } c_{\alpha_{i,n^*}} > 1 \text{ for some } n^* \in \{0,\dots,N-1\} \wedge \\ & \quad c_{\alpha_{i,n'}} \geq 1 \;\forall\; n' \in \{0,\dots,N-1\}, n' \neq n^*, \\ \begin{pmatrix} \mathbb{Z}_0^- \\ K \end{pmatrix} & \text{if } c_{\alpha_{i,n^*}} < 1 \text{ for some } n^* \in \{0,\dots,N-1\} \wedge \\ & \quad c_{\alpha_{i,n'}} \leq 1 \;\forall\; n' \in \{0,\dots,N-1\}, n' \neq n^*, \\ \begin{pmatrix} \mathbb{Z} \\ K \end{pmatrix} & \text{otherwise.} \end{cases} \tag{3.24}$$

Unobservable Lineage. When a lineage k has an inferred frequency of ≈ 0 across all samples, no copy number change is allowed to be assigned to this lineage:

$$\Delta C_{\alpha_{\cdot,k}} \in \begin{cases} \{0\} & \text{if } \sum_n \phi_{k,n} \leq \phi_\epsilon, \\ \mathbb{Z} & \text{otherwise.} \end{cases} \tag{3.25}$$

As default, we set ϕ_ϵ to be 0.00001.

3.2.4. Simple Somatic Mutation Assignment

We model the VAF $\hat{p}_{j,n}$ of SSM j in sample n as

$$\hat{p}_{j,n} = \frac{\hat{s}_{j,n}}{\hat{c}_{i,n}}, \tag{3.26}$$

where $\hat{s}_{j,n}$ is the inferred average copy number of SSM j in sample n (see Equation 3.27) and $\hat{c}_{i,n}$ the average inferred copy number of segment i (see Equation 3.19) in which SSM j appears. The average copy number $\hat{s}_{j,n}$ of SSM j depends on a) the lineage and phase it is assigned to and whether it is influenced by copy number changes in b) its own lineage or c) descendant lineages:

$$\begin{aligned}\hat{s}_{j,n} = &\sum_k \left(\left(\Delta S_{j,k} + \Delta S_{A_{j,k}} + \Delta S_{B_{j,k}} \right) \cdot \phi_{k,n} \right) && \text{(a)} \\ &+ \sum_k \left(\left(\Delta F_{A_{j,k}} + \Delta F_{B_{j,k}} \right) \cdot \phi_{k,n} \right) && \text{(b)} \\ &+ \sum_k \left(\sum_{k' \in \mathcal{D}_k} \left(\Delta S_{A_{j,k}} \cdot \Delta C_{A_{i,k'}} + \Delta S_{B_{j,k}} \cdot \Delta C_{B_{i,k'}} \right) \cdot \phi_{k',n} \right). && \text{(c)} \end{aligned} \tag{3.27}$$

The three matrices ΔS, ΔS_A and $\Delta S_B \in \{0,1\}^{J \times K}$ model the lineage and phase assignments of SSMs, and the two matrices ΔF_A and $\Delta F_B \in \{0,1\}^{J \times K}$ model whether SSMs are influenced by copy number changes in their own lineage.

If SSM j is assigned to lineage k and if a copy number change in the same segment and on the same allele as j is assigned to lineage k', with $k < k'$, the average copy number $\hat{s}_{j,n}$ of j depends on the ancestor-descendant relationship between lineages k and k'. If a present relationship increases the likelihood compared to an absent or ambiguous relationship, we say that we can observe the presence of the relationship in the input data. Note that the same likelihood is computed for $Z_{k,k'} = 0$ and $Z_{k,k'} = ?$ because both values lead to the same the set of descendants $\mathcal{D}_k$ (see Equation 3.10).

ΔS_A models the assignment to allele A, ΔS_B the assignment to allele B and ΔS the unphased assignment. An entry at position (j,k) in any of the three matrices is 1 if SSM j is assigned to lineage k with the corresponding phase, otherwise the entry is 0.

ΔF_A models the influence of a copy number change on allele A, ΔF_B on allele B, respectively. A copy number change that arises in the same lineage as the SSM is able to influence $\hat{s}_{n,j}$ only if it is a gain. If it was a loss, an SSM phased to the same allele would also be lost and thus could not have been detected. $\Delta F_{\alpha_{j,k}}$ is 1 if SSM j is assigned to allele α of lineage k and if $\hat{s}_{j,n}$ is influenced by a gain of allele α in lineage k, otherwise the entry is 0. We model $\Delta F_{\alpha_{j,k}}$ as follows:

$$\Delta F_{\alpha_{j,k}} \in \begin{cases} \{0\} & \text{if } \Delta S_{\alpha_{j,k}} = 0 \text{ or } \Delta C_{\alpha_{i,k}} \leq 0, \\ \{0,1\} & \text{otherwise.} \end{cases} \tag{3.28}$$

While modeling the lineage and phase assignments of SSMs, we do not allow an SSM j to be assigned to the normal lineage:

$$\Delta S_{j,0} = 0, \tag{3.29}$$

$$\Delta S_{A_{j,0}} = 0, \tag{3.30}$$

$$\Delta S_{B_{j,0}} = 0. \tag{3.31}$$

Infinite Sites Assumption. We use the infinite sites assumption to model SSM assignments, hence each SSM j can be assigned only to a single lineage and phase:

$$\sum_k \left(\Delta S_{j,k} + \Delta S_{A_{j,k}} + \Delta S_{B_{j,k}} \right) = 1. \tag{3.32}$$

Unobservable Lineage. When a lineage k has an inferred frequency of ≈ 0 across all samples, no SSM is allowed to be assigned to this lineage:

$$\Delta S_{\cdot,k} \cup \Delta S_{A_{\cdot,k}} \cup \Delta S_{B_{\cdot,k}} \in \begin{cases} \{0\} & \text{if } \sum_n \phi_{k,n} \leq \phi_\epsilon, \\ \{0,1\} & \text{otherwise,} \end{cases} \tag{3.33}$$

with ϕ_ϵ being the default minimal frequency as presented in Subsection 3.2.3 on page 36.

Removing Phasing Ambiguity. When the assigned phase of an SSM j does not have an influence on its average copy number $\hat{s}_{j,n}$, different phase assignments of j lead to the same likelihood, and therefore ambiguous subclonal reconstructions exist (see Section 4.1). To remove this phasing ambiguity, we apply Equations 3.34 – 3.36.

An SSM j in segment i of lineage k is modeled as unphased if its average copy number $\hat{s}_{j,n}$ is not influenced by copy number changes. This means that a) no copy number changes are assigned to descendant lineages, that no copy number losses are

assigned to b) ancestral lineages and c) k itself, and that d) j is not influenced by copy number gains of k:

$$\begin{aligned}\Delta S_{A_{j,k}} + \Delta S_{B_{j,k}} \leq & \sum_{k' \in \mathcal{D}_k} \left(|\Delta C_{A_{i,k'}}| + |\Delta C_{B_{i,k'}}| \right) && \text{(a)} \\ & + \sum_{k^* \in \mathcal{A}_k} \left(\Delta C^{loss}_{A_{i,k^*}} + \Delta C^{loss}_{B_{i,k^*}} \right) && \text{(b)} \\ & + \Delta C^{loss}_{A_{i,k}} + \Delta C^{loss}_{B_{i,k}} && \text{(c)} \\ & + \Delta F_{A_{j,k}} + \Delta F_{B_{j,k}}\,, && \text{(d)} \end{aligned} \tag{3.34}$$

with

$$\Delta C^{loss}_{\alpha_{i,k}} = \begin{cases} 1 & \text{if } \Delta C_{\alpha_{i,k}} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

An SSM j in segment i of lineage k is modeled as phased if a copy number change happens in a descendant lineage:

$$\Delta S_{j,k} \in \begin{cases} \{0\} & \text{if for any } \alpha \in \{A, B\} : \sum_{k' \in \mathcal{D}_k} \Delta C_{\alpha_{i,k'}} \neq 0, \\ \{0, 1\} & \text{otherwise.} \end{cases} \tag{3.35}$$

An SSM j in segment i of lineage k is not allowed to be phased to allele α or to be unphased if α got lost in an ancestral lineage or k itself:

$$\Delta S_{j,k} + \Delta S_{\alpha_{j,k}} \leq - \sum_{k^* \in \mathcal{A}_k} \Delta C^{loss}_{\alpha_{i,k^*}} - \Delta C^{loss}_{\alpha_{i,k}} + 1. \tag{3.36}$$

Clustering SSMs: Weak Parsimony Assumption. We explicitly use the weak parsimony assumption by offering the possibility to cluster SSMs based on their VAFs. In a preprocessing step, the SSMs of each segment are clustered with a clustering algorithm within their segments. Then, for each cluster κ, we receive a list Λ_κ of indices of the SSMs that are assigned to that cluster. We implemented two versions of how to proceed from here. In the first version, we assign all SSMs of the same cluster to the same lineage. In the second version, we combine SSMs of the same cluster to so called *superSSMs* and perform the optimization on these superSSMs. We evaluate the performance of different clustering algorithms and compare the two versions in Section 5.5.

Clustering SSMs: Version 1. In version 1, we assign all SSMs of the same cluster κ to the same lineage and phase:

$$\Delta S_{\alpha_{j^*,k}} = \Delta S_{\alpha_{j',k}}, \text{ for } j^* = \Lambda_\kappa[0], \forall\, j' \in \Lambda_\kappa, j' \neq j^*. \tag{3.37}$$

Also, we ensure that SSMs of the same cluster are equally influenced by copy number gains in their lineage:

$$\Delta F_{\alpha_{j^*,k}} = \Delta F_{\alpha_{j',k}}, \text{ for } j^* = \Lambda_\kappa[0], \forall\, j' \in \Lambda_\kappa, j' \neq j^*. \tag{3.38}$$

Clustering SSMs: Version 2. In version 2, we reduce the number of SSMs by combining all SSMs of one cluster κ to a superSSM j_κ with:

$$V_{j_\kappa,n} = \sum_{j' \in \Lambda_\kappa} V_{j',n}$$

and

$$R_{j_\kappa,n} = \sum_{j' \in \Lambda_\kappa} R_{j',n}\,.$$

3.3. Optimization with Mixed Integer Linear Programming

Since we could not find a closed formula from which we could derive the maximum likelihood estimator $\hat{\theta}$ of our model and since we are not aware that a closed formula exists for any related subclonal reconstruction problem, we decided to develop a MILP to find $\hat{\theta}$. This MILP is explained in this section.

In Subsection 3.3.1, we show how we build the objective function and how a basic version of our MILP looks like. Afterwards, in Subsections 3.3.2 – 3.3.5, we present all variables needed to create the model components and we explain how we translate the equations of Section 3.2, that define the rules of our model, into linear constraints. At the end, we show in Subsection 3.3.6 how the number of variables and constraints of our MILP can be reduced.

3.3.1. Objective Function and Basic Mixed Integer Linear Program

An objective function for an MILP needs to be linear, however, the single terms (see Equations 3.1 and 3.4) of our combined log-likelihood function $\mathcal{L}'$ are not linear, thus $\mathcal{L}'$ is not linear either. Hence, we approximate the non-linear terms with piecewise

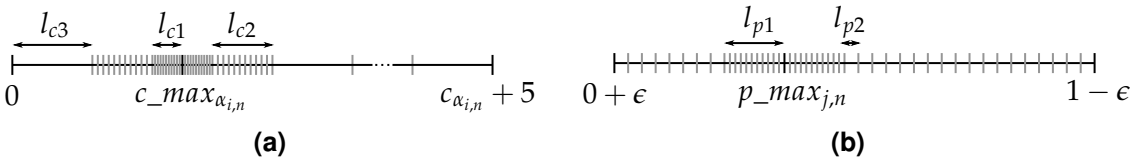


Figure 3.1.: Placement of knots in (a) copy number and (b) VAF intervals. (a) On each side of the knot at position $c\_max_{\alpha_{i,n}}$, n_{knots} knots with equal distance are inserted in an interval of length l_{c1}. Afterwards, n_{knots} knots with equal distance are placed in an interval around that with length l_{c2} to both sides. Finally, knots with distance l_{c3} are inserted in the remaining interval $[0, c_{\alpha_{i,n}} + 5]$, starting from the beginning and from the end. (b) On each side of the knot at position $p\_max_{j,n}$, n_{knots} knots with equal distance are inserted in an interval of length l_{p1}. The remaining interval $[0 + \epsilon, 1 - \epsilon]$ is filled with knots having distance l_{p2} to adjacent knots.

linear functions in order to build the objective function as explained in Subsection 2.1.1 on page 6. For both observed average allele-specific copy numbers $c_{A_{i,n}}$ and $c_{B_{i,n}}$ of each segment i and for the observed VAF $p_{j,n}$ of each SSM j in each sample n, we choose an interval that is large enough for the optimization. The size of the interval is crucial since a too small interval leads to an infeasible solution.

In each of the intervals, we place knots at specific positions, compute their corresponding log-likelihood and interpolate a piecewise linear function through these points. Our goal is to approximate each term of the log-likelihood function $\mathcal{L}'$ as well as possible, especially around the maximum. The more points are used to interpolate the piecewise linear function through, the better the approximation. However, the run time of the optimization increases with increasing number of points. Thus, we choose a placement of knots that has the highest density around the maximum and a lower density elsewhere.

Piecewise Linear Functions for Average Allele-Specific Copy Numbers. For the observed average allele-specific copy number $c_{\alpha_{i,n}}$, we build the piecewise linear function in the interval $[0, c_{\alpha_{i,n}} + 5]$ of possibly inferred copy numbers. Here, 0 is the natural start of the interval since the copy number cannot be smaller than 0. We chose $c_{\alpha_{i,n}} + 5$ as the right end of the interval because we assume that the inferred copy number is close to the observed one and that by allowing an offset of 5, a feasible solution can be found.

At the beginning, we compute the position $c\_max_{\alpha_{i,n}}$ that maximizes the simplified log-likelihood $\mathcal{N}'$ (see Equation 3.39). Then, we insert knots with different distances into the interval (see Figure 3.1a). The first knot is inserted at position $c\_max_{\alpha_{i,n}}$. After-

wards, n_{knots} knots with equal distance are inserted left and right of position $c\_max_{\alpha_{i,n}}$ in an area of length l_{c1}. Then, we insert n_{knots} knots with equal distance to the left of position $c\_max_{\alpha_{i,n}} - l_{c1}$ and to the right of position $c\_max_{\alpha_{i,n}} + l_{c1}$ in an area of length l_{c2}. We do not insert knots to the left of position $c\_max_{\alpha_{i,n}}$ that exceed the interval begin of 0. At the end, we insert knots with distance of l_{c3} at the beginning of the interval until they reach the position $c\_max_{\alpha_{i,n}} - (l_{c1} + l_{c2})$ and at the end of the interval until they reach the position $c\_max_{\alpha_{i,n}} + l_{c1} + l_{c2}$. Per default, we set $l_{c1} = 0.1$, $l_{c2} = 0.9$, $l_{c3} = 1$, and $n_{\text{knots}} = 50$. Like this, we achieve a high density of knots around the maximum and a lower density elsewhere.

After inserting all knots into the interval, we compute the simplified log-likelihood

$$\mathcal{N}'(c_{\alpha_{i,n,i'}} \mid c_{\alpha_{i,n}}, \sigma^2_{\alpha_{i,n}}) = -\frac{(c_{\alpha_{i,n,i'}} - c_{\alpha_{i,n}})^2}{2 \cdot \sigma^2_{\alpha_{i,n}}} \tag{3.39}$$

for each knot i' in the interval and interpolate a piecewise linear function through all points. We can neglect the first factor, $\frac{1}{\sqrt{2\pi\sigma^2_{\alpha_{i,n}}}}$, of the probability density function of the normal distribution since it does not depend on $c_{\alpha_{i,n,i'}}$ and becomes a constant summand, that does not influence the optimization when we apply the logarithm.

Piecewise Linear Functions for SSMs. For the observed VAF of SSM j in sample n, we construct the piecewise linear function in the interval $[0 + \epsilon, 1 - \epsilon]$ of possibly inferred VAFs. We need to shift the borders of the interval by $\epsilon > 0$ because the logarithm of the beta-binomial probability mass function that we use for the VAFs is undefined for $\hat{p}_{j,n} = 0$ and $\hat{p}_{j,n} = 1$.

At the beginning, we compute the position $p\_max_{j,n}$ that maximizes the log-likelihood Beta-Bin$'$ (see Equation 3.40). Afterwards, we insert knots with different distances into the interval (see Figure 3.1b). The first knot is inserted at position $p\_max_{j,n}$. Then, n_{knots} knots with equal distance are inserted left and right of position $p\_max_{j,n}$ in an area of length l_{p1}. Next, we insert knots at the beginning of the interval in equal distance of l_{p2} until the position $p\_max_{j,n} - l_{p1}$ is reached and at the end of the interval until the position $p\_max_{j,n} + l_{p1}$ is reached. Per default, we set $l_{p1} = 0.1$, $l_{p2} = 0.02$, and $\epsilon = 0.0000001$.

After we inserted all knots, we compute the log-likelihood

$$\text{Beta-Bin}'(\alpha_{j,n,j'}, \beta_{j,n,j'} \mid D_{j,n}, V_{j,n}) = \log(\text{Beta-Bin}(\alpha_{j,n,j'}, \beta_{j,n,j'} \mid D_{j,n}, V_{j,n})), \tag{3.40}$$

where $\alpha_{j,n,j'}$ and $\beta_{j,n,j'}$ are derived from $p_{j,n,j'}$ (see Equations 3.2 and 3.3), for each knot j' in the interval. Then, we interpolate a piecewise linear function through all points.

Note that for a future improvement, we plan to insert the knots not with predefined distances but for different percentiles of the log-likelihood functions. This way, the coverage of knots adapts to the shape of the log-likelihood functions.

After computing all piecewise linear functions, we can build the MILP. A basic version of it is shown here:

$$
\begin{aligned}
\max \quad & \sum_n \sum_i \sum_{i'} w_{A_{i,n,i'}} \cdot \mathcal{N}'(c_{A_{i,n,i'}} \mid c_{A_{i,n}}, \sigma^2_{A_{i,n}}) \\
& + \sum_n \sum_i \sum_{i'} w_{B_{i,n,i'}} \cdot \mathcal{N}'(c_{B_{i,n,i'}} \mid c_{B_{i,n}}, \sigma^2_{B_{i,n}}) \\
& + \sum_n \sum_j \sum_{j'} w_{j,n,j'} \cdot \text{Beta-Bin}'(\alpha_{j,n,j'}, \beta_{j,n,j'} \mid D_{j,n}, V_{j,n}) \\
\text{s. t.} \quad & \sum_{i'} w_{A_{i,n,i'}} \cdot c_{A_{i,n,i'}} = \hat{c}_{A_{i,n}} && \forall\, i, n \\
& \sum_{i'} w_{B_{i,n,i'}} \cdot c_{A_{i,n,i'}} = \hat{c}_{B_{i,n}} && \forall\, i, n \\
& \sum_{j'} w_{j,n,j'} \cdot p_{j,n,j'} = \tilde{p_{j,n}} && \forall\, j, n \\
& \sum_{i'} w_{A_{i,n,i'}} = 1 && \forall\, i, n \\
& \sum_{i'} w_{B_{i,n,i'}} = 1 && \forall\, i, n \\
& \sum_{j'} w_{j,n,j'} = 1 && \forall\, j, n
\end{aligned}
$$

The indices i' and j' iterate over all knots in the intervals of the corresponding piecewise linear functions. The variables $w_{A_{i,n,i'}}$, $w_{B_{i,n,i'}}$ and $w_{j,n,j'}$ are real-valued between 0 and 1. The variables $\hat{c}_{A_{i,n}}$, $\hat{c}_{B_{i,n}}$ and $\tilde{p_{j,n}}$ are real-valued as well and defined in the following subsections by Constraints 3.43 and 3.53.

3.3.2. Variables and Constraints for Lineage Frequencies

We represent the lineage frequencies with $K \cdot N$ real-valued variables $\phi_{k,n}$, $0 \leq k < K$, $0 \leq n < N$, with $0 \leq \phi_{k,n} \leq 1$. Equation 3.6 for the frequency of the normal lineage and Equation 3.7 for sorting the lineages according to their frequencies in the first sample are already linear constraints.

Sum Rule. To model the sum rule (see Equation 3.8), we construct $N \cdot K^2$ auxiliary real-valued variables $child\_freq_{n,k,k'}$, $0 \leq n < N$, $0 \leq k < K$, $0 \leq k' < K$, with $0 \leq child\_freq_{n,k,k'} \leq 1$. If lineage k' is a child of lineage k, $child\_freq_{n,k,k'}$ equals $\phi_{k',n}$, otherwise it is 0. We create the variables with the trick to multiply real-valued and binary variables shown in Subsection 2.1.1 on page 6. Now, we can model the sum rule as linear constraint:

$$\sum_{k'|k'<k} child\_freq_{n,k,k'} \leq \phi_{k,n} .$$

Fixing Lineage Frequencies. When the lineage frequencies are known, e. g. from previous experiments, they can be fixed for the optimization:

$$\phi_{k,n} = \Phi_{k,n} ,$$

where $\Phi_{k,n}$ is the known frequency. Fixing the lineage frequencies removes a lot of ambiguity (see Section 4.1). The benefits of fixing the lineage frequencies are shown in Subsection 5.7.3.

3.3.3. Variables and Constraints for Lineage Relationships

As $Z_{k,k'} = ?$ leads to the same likelihood as $Z_{k,k'} = 0$ (see Subsection 3.2.4, page 37), and thus to the same objective value, we remove ambiguity in the optimization by not using the value ?. After the optimization, we analyze which of the lineage relationships inferred as absent are actually ambiguous (see Section 4.2). Not using the value ? allows us to model the lineage relationships in the MILP with Z^2 binary variables $Z_{k,k'}$, $0 \leq k < K$, $0 \leq k' < K$.

We do not need to create variables for the set of ancestors $\mathcal{A}_k$ of lineage k (Equation 3.9), the set of descendants $\mathcal{D}_k$ (Equation 3.10) and the set of children χ_k (Equation 3.11) since we will model the membership in these sets with individual variables as done with the variables $child\_freq_{n,k,k'}$ in Subsection 3.3.2.

The trivial property that a lineage cannot be its own ancestor or descendant (Equation 3.12), and the condition that the normal lineage 0 is ancestor of all other lineages (Equation 3.13) are already linear constraints.

Lineage Precedence Rule and Crossing Rule. Equation 3.14, that fills the lower left triangle of the lineage relationship matrix Z with 0s, is already a linear constraint. We model the linear constraint that lineage k cannot be an ancestor of lineage k' if

lineage k' has a higher frequency $\phi_{k',n}$ than lineage k in at least one sample n (Equation 3.15) as

$$\phi_{k,n} - \phi_{k',n} \geq Z_{k,k'} - 1.$$

Phylogenetic Tree Rules. To model the two tree rules of Equations 3.16 and 3.17, we create $\mathcal{O}(K^3)$ auxiliary binary variables $Z\_tree\_1_{k,k',k''}$ and $Z\_tree\_2_{k,k',k''}$, with $1 \leq k < k' < k'' < K$. Variable $Z\_tree\_1_{k,k',k''}$ is 1 if $Z_{k,k'} = 1$ and $Z_{k',k''} = 1$, otherwise it is 0. Variable $Z\_tree\_2_{k,k',k''}$ is 1 if $Z_{k,k''} = 1$ and $Z_{k',k''} = 1$, otherwise it is 0. We construct the variables as show in Section 2.1.1 on page 6. Given these auxiliary variables, we create linear constraints that model Equations 3.16 and 3.17 as follows:

$$Z_{k,k''} \geq Z\_tree\_1_{k,k',k''} \; \forall \; k', k < k' < k'', \tag{3.41}$$

and

$$Z_{k,k'} \geq Z\_tree\_2_{k,k',k''} \; \forall \; k'', k' < k'' < K. \tag{3.42}$$

In Subsection 3.3.6, we show how we can model Equations 3.16 and 3.17 without having to create $\mathcal{O}(K^3)$ auxiliary binary variables.

Fixing Lineage Relationships. When lineage relationships are known, e. g. from previous experiments, they can be fixed for the optimization:

$$Z_{k,k'} = \mathcal{Z}_{k,k'},$$

where $\mathcal{Z}_{k,k'}$ is the known relationship.

3.3.4. Variables and Constraints for Copy Number Aberrations

In order to model the multiplication in Equation 3.18 to compute inferred average allele-specific copy numbers, we first need to model the matrices ΔC_A and ΔC_B as binary matrices. To differentiate between different copy number changes, we use the matrices ΔC_A^{gain} and ΔC_B^{gain} to model copy number gains and the matrices ΔC_A^{loss} and ΔC_B^{loss} to model copy number losses. Because our current implementation allows copy number changes of only $+1$ or -1, differentiating between gains and losses is sufficient. Thus, we create $\mathcal{O}(I \cdot K)$ binary variables to model copy number changes per allele, segment and lineage.

With the help of the binary matrices ΔC_A^{gain}, ΔC_B^{gain}, ΔC_A^{loss} and ΔC_B^{loss} and the trick to multiply real-valued and binary variables shown in Subsection 2.1.1 on page 6, we

create $\mathcal{O}(I \cdot K \cdot N)$ real-valued variables $\Delta C\_freq^{gain}_{A_{i,k,n}}$, $\Delta C\_freq^{gain}_{B_{i,k,n}}$, $\Delta C\_freq^{loss}_{A_{i,k,n}}$ and $\Delta C\_freq^{loss}_{B_{i,k,n}}$ with a lower bound of 0 and an upper bound of 1. An entry (i,k,n) equals the lineage frequency $\phi_{k,n}$ if in segment i the specific copy number change is assigned to lineage k. Now we can formulate Equation 3.18 to compute inferred average allele-specific copy numbers as the following linear constraint:

$$\hat{c}_{\alpha_{i,n}} = 1 + \sum_k \Delta C\_freq^{gain}_{\alpha_{i,k,n}} - \sum_k \Delta C\_freq^{loss}_{\alpha_{i,k,n}} . \tag{3.43}$$

Equation 3.20, which ensures that $\hat{c}_{A_{i,n}}$ is larger than or equal to $\hat{c}_{B_{i,n}}$, can be modeled as a straight-forward linear constraint with the help of Constraint 3.43.

Equation 3.21, which does not allow the normal lineage to have copy number changes, is already a linear constraint.

Allowed Copy Number Changes. To model Equation 3.22, that ensures that each segment i contains only copy number gains, losses or one LOH event, we need to apply the following four linear constraints for all lineages k and k', with $1 \leq k < K$, $1 \leq k' < K$, and $k \neq k'$:

$$\Delta C^{gain}_{A_{i,k}} + \Delta C^{loss}_{A_{i,k'}} \leq 1, \tag{3.44}$$

$$\Delta C^{gain}_{A_{i,k}} + \Delta C^{loss}_{B_{i,k'}} \leq 1, \tag{3.45}$$

$$\Delta C^{gain}_{B_{i,k}} + \Delta C^{loss}_{A_{i,k'}} \leq 1, \tag{3.46}$$

$$\Delta C^{gain}_{B_{i,k}} + \Delta C^{loss}_{B_{i,k'}} \leq 1. \tag{3.47}$$

Like this, gains and losses are allowed to appear together in a segment only if they are assigned to the same lineage. Further, we need to exclude the case that the same allele is duplicated *and* lost in the same lineage:

$$\Delta C^{gain}_{\alpha_{i,k}} + \Delta C^{loss}_{\alpha_{i,k}} \leq 1. \tag{3.48}$$

To model Equation 3.23, that does not allow alleles of a segment to get lost multiple times on the same branch of the phylogeny, we first need to create $\mathcal{O}(I \cdot K^2)$ binary variables $\Delta C\_\mathcal{A}^{loss}_{A_{i,k,k'}}$ and $\Delta C\_\mathcal{A}^{loss}_{B_{i,k,k'}}$, with $1 \leq k < k' < K$. A variable $\Delta C\_\mathcal{A}^{loss}_{\alpha_{i,k,k'}}$ is 1 if lineage k is an ancestor of lineage k' and if k loses allele α in segment i, otherwise it

is 0. We create the variables with the trick to multiply real-valued and binary variables (see Subsection 2.1.1 on page 6). Now we can formulate the following constraint:

$$\sum_{k|k<k'} \left(\Delta C\_\mathcal{A}^{loss}_{\alpha_{i,k,k'}} + \Delta C^{loss}_{\alpha_{i,k'}} + \Delta C^{gain}_{\alpha_{i,k'}} \right) \leq 1, \tag{3.49}$$

for $0 < k < k' < K$ and $2 \leq k'$. In addition to not allowing the loss of an already lost allele, this constraint does not allow the gain of an already lost allele. We discuss in Subsection 3.3.6, how we can model Equations 3.22 and 3.23 with less constraints in the MILP.

To model Equation 3.24, we use Constraints 3.50 and 3.51. Constraint 3.50 does not allow copy number losses and is used if at least one copy number gain and no copy number loss is observed for segment i in all samples. Constraint 3.51 is applied if at least one copy number loss and no copy number gain is observed for segment i in all samples and does not allow copy number gains:

$$\Delta C^{loss}_{\alpha_{i,k}} = 0 \tag{3.50}$$

and

$$\Delta C^{gain}_{\alpha_{i,k}} = 0. \tag{3.51}$$

Unobservable Lineage. To not allow the assignments of copy number changes to lineages with an inferred frequency ≈ 0 (see Equation 3.25), we use the following linear constraints:

$$\sum_n \phi_{k,n} - \Delta C^{gain}_{\alpha_{i,k}} \geq \phi_\epsilon - 1 \; \forall \; i \in \{0, \ldots, I-1\}, \forall \; \alpha \in \{A, B\}$$

and

$$\sum_n \phi_{k,n} - \Delta C^{loss}_{\alpha_{i,k}} \geq \phi_\epsilon - 1 \; \forall \; i \in \{0, \ldots, I-1\}, \forall \; \alpha \in \{A, B\}.$$

Maximal Number of Copy Number Changes. The maximal number of copy number changes per segment is $2 \cdot (K-1)$. Because enumerating over all these possible copy number change assignments in the branch-and-cut algorithm is time-consuming, we offer the possibility to restrict the number of allowed copy number changes per segment to *c_num_max*:

$$\sum_k \left(\Delta C^{gain}_{A_{i,k}} + \Delta C^{gain}_{B_{i,k}} + \Delta C^{loss}_{A_{i,k}} + \Delta C^{loss}_{B_{i,k}} \right) \leq c\_num\_max. \tag{3.52}$$

Fixing CNA Assignments. When copy number changes are known, e. g. from previous experiments or because we do not want to infer copy number changes in CNA-free segments with neutral allele-specific copy numbers, we can fix them for the optimization:

$$\begin{aligned}
\Delta C^{gain}_{A_{i,k}} &= fixed_C^{gain}_{A_{i,k}}, \\
\Delta C^{loss}_{A_{i,k}} &= fixed_C^{loss}_{A_{i,k}}, \\
\Delta C^{gain}_{B_{i,k}} &= fixed_C^{gain}_{B_{i,k}}, \\
\Delta C^{loss}_{B_{i,k}} &= fixed_C^{loss}_{B_{i,k}},
\end{aligned}$$

where $fixed\_C^{gain}_{A_{i,k}}$, $fixed\_C^{loss}_{A_{i,k}}$, $fixed\_C^{gain}_{B_{i,k}}$ and $fixed\_C^{loss}_{B_{i,k}} \in \{0, 1\}$ are the known values.

Combining CNA-Free Segments. To reduce the number of segments and thus the number of binary variables, we offer the possibility to combine all CNA-free segments to one *super segment*, which we fix to have no copy number changes. We treat all SSMs of these CNA-free segments as belonging to this super segment. When using superSSMs as well, this has the clear advantage that SSMs of multiple segments can be summarized in one superSSM which reduces the number of SSM variables as well.

3.3.5. Variables and Constraints for Simple Somatic Mutations

We create $\mathcal{O}(J \cdot K)$ binary variables for the two binary matrices ΔS_A and ΔS_B, that model the lineage and phase assignments of SSMs, and the two binary matrices ΔF_A and ΔF_B, that model whether SSMs are influenced by copy number changes in their own lineage. Note that, in our MILP, we do not represent the binary matrix ΔS, that models whether SSMs are unphased, because we can determine after the optimization whether an SSM is unphased without changing the likelihood. Hence, we need less binary variables, as well as less constraints to remove phasing ambiguity (see page 50).

In a MILP, we cannot model the inferred VAF $\hat{p}_{j,n}$ of SSM j in sample n with Equation 3.26 because the division of the inferred average copy number $\hat{s}_{j,n}$ by the inferred average copy number $\hat{c}_{i,n}$ of segment i cannot be formulated as linear constraint. Thus, we approximate $\hat{p}_{j,n}$ by using the observed average copy number $c_{i,n} = c_{A_{i,n}} + c_{B_{i,n}}$.

Since $c_{i,n}$ is not a variable but a coefficient of our MILP, we can build the linear constraint:

$$\tilde{p_{j,n}} = \frac{\hat{s}_{j,n}}{c_{i,n}}. \tag{3.53}$$

In Section 5.8, we evaluate the impact of this approximation. Note that we use the approximation only in our MILP; when we compute the likelihood of a reconstruction r, we use the exact Equation 3.26.

To model the average copy number $\hat{s}_{j,n}$ of SSM j (see Equation 3.27) as linear constraint, we first need to create auxiliary real-valued variables to model the multiplication of (a) lineage and phase assignments with lineage frequencies, (b) copy number change influences in the same lineage with lineage frequencies and (c) copy number change influences in descendant lineages with lineage frequencies. We apply the trick shown in Subsection 2.1.1 on page 6 to create the following real-valued variables with lower bound 0 and upper bound 1:

a) $\Delta S\_freq_{j,k,n}$,

b) $\Delta S\_freq_{\alpha_{j,k,n}}$,

c) $\Delta S\_freq^{gain}_{\alpha_{j,k,k',n}}$, $\Delta S\_freq^{loss}_{\alpha_{j,k,k',n}}$,

for $0 \leq j < J$, $0 \leq k < K$, $0 \leq k' < K$ and $0 \leq n < N$.

Given the real-valued variables, we can model $\hat{s}_{j,n}$ as follows:

$$\begin{aligned}
\hat{s}_{j,n} = \quad & \sum_k \Delta S_freq_{j,k,n} && \text{(a)} \\
& + \sum_k \left(\Delta S_freq_{A_{j,k,n}} + \Delta S_freq_{B_{j,k,n}} \right) && \text{(b)} \\
& + \sum_k \sum_{k'} \left(\Delta S_freq^{gain}_{A_{j,k,k',n}} + \Delta S_freq^{gain}_{B_{j,k,k',n}} + \Delta S_freq^{loss}_{A_{j,k,k',n}} + \Delta S_freq^{loss}_{B_{j,k,k',n}} \right). && \text{(c)}
\end{aligned} \tag{3.54}$$

We model Equation 3.28, that defines the values of $\Delta F_{\alpha_{j,k}}$, with the two following constraints:

$$\Delta F_{\alpha_{j,k}} \leq \Delta S_{\alpha_{j,k}}$$

and

$$\Delta F_{\alpha_{j,k}} \leq \Delta C_{\alpha_{i,k}}\,.$$

Equations 3.30 and 3.31, that do not allow to assign an SSM to the normal lineage, are already linear constraints. Equation 3.29 that considers the unphased assignment is not modeled in our MILP.

Infinite Sites Assumption. Equation 3.32, that models the infinite sites assumption, is already a linear constraint.

Unobservable Lineage. To not allow the assignments of SSMs to lineages with an inferred frequency ≈ 0 (see Equation 3.33), we use the following linear constraint:

$$\sum_n \phi_{k,n} - \Delta S_{\alpha_{j,k}} \geq \phi_\epsilon - 1 \; \forall \; j \in \{0, \ldots, J-1\}, \forall \; \alpha \in \{A, B\}.$$

Removing Phasing Ambiguity. Equation 3.34 models that an SSM j is unphased if its average copy number $\hat{s}_{j,n}$ is not influenced by copy number changes. Since we do not use a variable for unphased SSMs in our MILP, we simply assign these SSMs to allele A:

$$\begin{aligned}
\Delta S_{B_{j,k}} \leq \quad & \sum_{k' | k' > k} \left(\Delta C_\mathcal{D}^{gain}_{A_{i,k,k'}} + \Delta C_\mathcal{D}^{gain}_{B_{i,k,k'}} + \Delta C_\mathcal{D}^{loss}_{A_{i,k,k'}} + \Delta C_\mathcal{D}^{loss}_{B_{i,k,k'}} \right) \\
& + \sum_{k^* | k^* < k} \left(\Delta C_\mathcal{A}^{loss}_{A_{i,k^*,k}} + \Delta C_\mathcal{A}^{loss}_{B_{i,k^*,k}} \right) \\
& + \Delta C^{loss}_{A_{i,k}} + \Delta C^{loss}_{B_{i,k}} \\
& + \Delta F_{A_{j,k}} + \Delta F_{B_{j,k}} \,,
\end{aligned}$$

where $\Delta C\_\mathcal{D}^{gain}_{A_{i,k,k'}}$, $\Delta C\_\mathcal{D}^{gain}_{B_{i,k,k'}}$, $\Delta C\_\mathcal{D}^{loss}_{A_{i,k,k'}}$ and $\Delta C\_\mathcal{D}^{loss}_{B_{i,k,k'}}$, with $1 \leq k < k' < K$, are $\mathcal{O}(I \cdot K^2)$ binary variables created with the trick shown in Subsection 2.1.1 on page 6. They equal 1 if the corresponding copy number change is assigned to lineage k' and if k' is a descendant of lineage k. Otherwise they are 0.

After the optimization, we check for all SSMs whether they are influenced by copy number changes or not. If they are not influenced, we represent them as unphased SSMs.

We do not need to model Equation 3.35 in our MILP because during the optimization, we model all SSMs as phased.

We build the linear constraint for Equation 3.36, that ensures that an SSM cannot be assigned to an allele that got already lost in an ancestral lineage, as:

$$\Delta S_{\alpha_{j,k}} \leq - \sum_{k^*|k^*<k} \Delta C\_\mathcal{A}^{loss}_{\alpha_{i,k^*,k}} - \Delta C^{loss}_{\alpha_{i,k}} + 1.$$

Clustering SSMs: Version 1. Equations 3.37 and 3.38, that ensure that all SSMs belonging to one cluster are assigned to the same lineage and phase and are equally influenced by copy number gains in their lineage, are already linear constraints.

Clustering SSMs: Version 2. Optimizing with superSSMs does not need additional constraints. However, since using superSSMs can lead to slightly differently inferred lineage frequencies, we perform a second optimization with the original SSMs. Here, the mutation assignments and the lineage relationships are fixed to the solution of the first optimization and only the lineage frequencies get inferred.

Fixing SSM assignments. When lineage and phase assignments of SSMs are known, e. g. from clustering with superSSMs, we can fix the SSMs for the next optimization:

$$\Delta S_{A_{j,k}} = fixed\_S_{A_{j,k}},$$

$$\Delta S_{B_{j,k}} = fixed\_S_{B_{j,k}},$$

where $fixed\_S_{A_{j,k}}$ and $fixed\_S_{B_{j,k}} \in \{0,1\}$ are the known values.

3.3.6. Reducing the Number of Variables and Constraints

In the current version of our MILP, we create variables that are always fixed to the same values in each possible reconstruction These variables include all variables that concern the normal lineage 0: the frequency of the normal lineage $\phi_{0,n}$, each relationship $Z_{0,k}$ of the normal lineage to the cancerous lineages, $0 < k < K-1$, and all variables that model the assignment of CNAs and SSMs to the normal lineage, their influence and the frequency of the normal lineage (ΔC and ΔS variables). Also, each lineage relationship $Z_{k,k}$ and $Z_{k',k}$, $k < k'$, is fixed, and as a consequence also each variable $child\_freq_{n,k',k}$. Thus, we could reduce the number of variables by sparing these variables, which always have the same value, and adapt the constraints accordingly. However, as we remove only variables that concern the normal lineage and the

variables of the lower left triangle of the lineage relationship matrix Z, the asymptotic number of variables stays the same.

Less Variables: Phylogenetic Tree Rules. We can reduce the asymptotic number of variables by removing the binary variables $Z\_tree\_1_{k,k',k''}$ and $Z\_tree\_2_{k,k',k''}$ and modeling the phylogenetic tree rules (Equation 3.16 and 3.17) directly with the variables of the lineage relationship matrix Z.

For Equation 3.16, we need the following three constraints for all k' with $k < k' < k''$:

$$Z_{k,k''} \leq Z_{k,k'},$$

$$Z_{k,k''} \leq Z_{k',k''}$$

and

$$Z_{k,k''} \geq Z_{k,k'} + Z_{k',k''} - 1.$$

For Equation 3.17, we formulate the following three constraints for all k'' with $k' < k'' < K$:

$$Z_{k,k'} \leq Z_{k,k''},$$

$$Z_{k,k'} \leq Z_{k',k''}$$

and

$$Z_{k,k'} \geq Z_{k,k''} + Z_{k',k''} - 1.$$

Thus, we can remove $\mathcal{O}(K^3)$ binary variables from our MILP.

Less Constraints: Allowed Copy Number Changes. We use the linear constraints 3.44 – 3.51 to model allowed copy number changes. Thereby, Constraints 3.48 and 3.49 both do not allow that a copy number gain and loss are assigned to the same allele of the same lineage. We can decrease the number of used constraints by using Constraint 3.48 only for $k = 1$ since this case is not covered by Constraint 3.49.

Whether decreasing the number of constraints decreases the run time and memory requirements of the optimization is uncertain because CPLEX [18], which we use in our implementation, preprocesses the MILP before starting the optimization and might remove the redundant constraints. Anyhow, reducing the constraints leads to a clearer and simpler MILP.

3.4. Optimization Complexity

Optimizing a MILP is NP-hard. Given n_b binary variables, the branch-and-cut algorithm has to process 2^{n_b} nodes in the worst case. Thus, the number of binary variables has a high influence on the run time and memory requirements of the optimization.

Given I segments, J SSMs, N samples, K lineages, and $\mathcal{O}(n_{knots})$ knots to approximate each piecewise linear function of our objective function, our MILP consists of

$$\mathcal{O}\left((I \cdot N \cdot n_{knots}) + (J \cdot N \cdot n_{knots}) + (N \cdot K^2) + (I \cdot K \cdot N) + (J \cdot K^2 \cdot N)\right)$$

real-valued variables and

$$\mathcal{O}\left(K^3 + (I \cdot K^2) + (J \cdot K)\right)$$

binary variables. In Subsection 3.3.6, we show how these numbers can be reduced. However, all reductions, except the one where we can reduce $\mathcal{O}(K^3)$ to $\mathcal{O}(K^2)$ by removing the variables $Z\_tree\_1_{k,k',k''}$ and $Z\_tree\_2_{k,k',k''}$, do not change the asymptotic number of variables.

Given the number of variables, we make three main observations:

1. Only real-valued variables depend on the number of samples N, not the binary ones.

2. The number of real-valued and binary variables grows linearly with the number of segments I and SSMs J.

3. The number of real-valued variables grows quadratically and the number of binary variables grows cubicly (after improvements quadratically) with the number of lineages K.

Thus, the number of lineages K is the most critical parameter when it comes to finding an optimal solution in reasonable time and space, as we show in Section 5.4. Note that K is the only parameter that we have to set externally for the optimization. The other parameters are intrinsic to the input data and can be reduced only if CNA-free segments are combined to a super segment or if SSMs are combined to superSSMs through clustering.

To find an optimal or good solution in reasonable time and space, exploiting the structure of our model can help as we explain in the following.

We classify the variables into two classes: main and auxiliary variables. Main variables are the variables that directly influence the likelihood of the reconstruction.

These are the lineage frequencies $\phi_{k,n}$, the lineage relationships $Z_{k,k'}$, the CNA assignments $\Delta C_{A_{i,k}}$ and $\Delta C_{B_{i,k}}$, and the SSM assignments $\Delta S_{A_{j,k}}$ and $\Delta S_{B_{j,k}}$ with copy number influence $\Delta F_{A_{j,k}}$ and $\Delta F_{B_{j,k}}$ in the same lineage. All other variables are auxiliary variables. They are either needed to ensure the correct calculation of the piecewise linear functions or to model the multiplication of real-valued with binary variables. We will focus only on the main variables as the auxiliary variables follow from these.

Of the main variables, the lineage frequencies $\phi_{k,n}$ and relationships $Z_{k,k'}$ are global variables, the CNA and SSM assignments $\Delta C_{A_{i,k}}$, $\Delta C_{B_{i,k}}$, $\Delta S_{A_{j,k}}$ and $\Delta S_{B_{j,k}}$, and copy number influences $\Delta F_{A_{j,k}}$ and $\Delta F_{B_{j,k}}$ in the same lineages are local variables. This means that the lineage frequencies and relationships influence the calculations of each inferred average allele-specific copy number $\hat{c}_{\alpha_{i,n}}$ (see Equation 3.18) and each inferred average copy number $\hat{s}_{j,n}$ (see Equation 3.27), independent of to which segment they belong. In contrast, local variables of different segments influence each other only indirectly via the global variables.

If the global variables were known, they could be fixed for the optimization. Thus, the large optimization problem would decompose to I independent smaller problems and each segment with its local variables could be solved on its own. A good estimate for the lineage frequencies can be derived by optimizing input data of CNA-free segments first (see Subsection 5.7.2). However, inferring representative lineage relationships without the complete input data is not possible and iterating over all possible structures of the lineage relationship matrix Z is unreasonable for $K >= 5$ since too many different structures exist. But even fixing only the lineage frequencies decreases the practical run time and memory requirements considerably, as shown in Subsection 5.7.3.

3.5. Determining the Number of Lineages

When we want to compute the lineage-based subclonal reconstruction based on the allele-specific copy numbers of I genome segments and J SSMs of N samples, we also need to provide the number of lineages K. To determine the underlying lineage number of the input data, our method needs to be invoked multiple times with different values of K. The resulting subclonal reconstructions are then compared using the two-part version of the MDL principle (see Subsection 2.1.3). The lineage number of the subclonal reconstruction r that minimizes Equation 2.6 is then chosen as underlying lineage number and r as final reconstruction.

The description length of the first part of Equation 2.6, $L_{\mathcal{C}_1}(y^m \mid \theta, \mathcal{M}, x^m)$, is simply the computed log-likelihood $\mathcal{L}'(r)$. The description length of the second part, $L_{\mathcal{C}_2}(\theta \mid \mathcal{M})$, is the description length of our subclonal reconstruction r. We use a fixed-length code, which exploits the natural structure of the subclonal reconstruction, to send this length to the receiver. First, we need to encode the number of lineages K (1). Afterwards, we encode the complete mutation assignment. Because we infer CNAs in terms of copy number changes, a segment without CNA is the result of inferring no copy number change. Thus, in addition to encoding CNAs in the subclonal reconstruction, we also encode which segments are CNA-free. Thus, we need to encode the number of CNAs and CNA-free segments (2). Then, we encode CNA lineage assignment (3), mapping to segment indices (4), and the specific copy number change of CNAs (5) and their phase (6). To encode SSMs in the subclonal reconstruction, we encode their lineage assignment (7), phase (8) and whether they are influenced by a copy number gain in the same lineage segment (9). Finally, we need to encode the lineage frequencies (10) and the lineage relationships (11). The number of genome segments I, SSMs J and samples N are known to the receiver.

(1) We only need to send the number of cancerous lineages $K - 1$ because the receiver knows that the normal lineage is always present in our subclonal reconstructions. To encode this number, we use the simple standard code for the integers (see Subsection 2.1.3):

$$2 \cdot \log(K - 1) + 1 \text{ bits.}$$

(2) To send the number I^* of CNAs and CNA-free segments, we first define the following: Let I° be the number of CNAs and I' be the number of CNA-free segments, with $I^\circ + I' = I^*$. Also, let I'' be the number of segments with CNAs, with $I' + I'' = I$ being the number of segments. Then $I^\bullet = I^\circ - I''$ is the number of CNAs that are assigned to a segment that already contains at least one CNA. Because $I^* = I + I^\bullet$ and because the receiver knows the number of segments I, it is sufficient to send $I^\bullet$ with the simple standard code for integers:

$$2 \cdot \log(I^\bullet) + 1 \text{ bits.}$$

(3) We use the lineage assignment to differentiate between CNAs and CNA-free segments. CNAs are assigned to a cancerous lineage $k \in \{1, \ldots, K-1\}$, whereas CNA-free segments are assigned to the normal lineage 0:

$$I^* \cdot \log(K) \text{ bits.}$$

(4) Since a segment can contain multiple CNAs, we need to send the segment index i for each CNA and CNA-free segment. We encode this information in a binary string, where a 1 indicates using the next segment index $i+1$ for the current CNA or CNA-free segment and a 0 indicates using the current segment index i again:

$$I^* \cdot \log(2) \text{ bits.}$$

(5) The current implementation of our model allows copy number gains of $+1$ and copy number losses of -1. These two changes need to be encoded for all I° CNAs. The receiver calculates the value of I° from the lineage assignment (3) which was already sent:

$$I^\circ \cdot \log(2) \text{ bits.}$$

(6) For each CNA, we indicate whether it is phased to allele A or B by encoding the information in a binary string:

$$I^\circ \cdot \log(2) \text{ bits.}$$

(7) Each SSM is assigned to a cancerous lineage $k \in 1, \ldots, K-1$, which we encode in

$$J \cdot \log(K-1) \text{ bits.}$$

(8) Only the J' SSMs that lie in segments with CNAs can be phased to allele A or B or can be unphased, the other SSMs are always unphased. The information which J' SSMs lie in segments with CNAs can be derived from the segment assignment of SSMs given with the input data and the sent segment assignment of CNAs (4). Thus, we have to send only

$$J' \cdot \log(3) \text{ bits.}$$

(9) The J'' SSMs that are assigned to lineage segments that also contain copy number gains of the same phases as the SSMs can be influenced by these gains. Which J''

SSMs could be influenced by gains can be derived from the segment assignment of SSMs, their lineage assignment (7) and their phase (8), and the segment assignment of CNAs (4), their copy number change (5) and their phase (6). Hence, we encode this information in a binary string with

$$J'' \cdot \log(2) \text{ bits.}$$

(10) We encode the frequency of each cancerous lineage with a precision of $\frac{1}{1000}$, starting with all cancerous lineages of the first sample and then continuing with the other samples:

$$(K-1) \cdot N \cdot \log(1000) \text{ bits.}$$

(11) To send the lineage relationships, it is sufficient to encode only the parent of each lineage. Since the receiver knows that the lineage relationships describe a phylogenetic tree, all other present or absent ancestor-descendant relationships can be derived from this parental information. The information which absent ancestor-descendant relationships are ambiguous is computed after subclonal reconstruction (see Section 4.2) and does not need to be sent. Because the normal lineage has no parent and the parent of lineage 1 is always the normal lineage, we need to send the parental information only for lineages $2, \ldots, K-1$. Also, since the receiver knows that only a lineage k^*, with $k^* < k$, can be the parent of a lineage k, we encode the parent information only for lineages with index lower than k:

$$\sum_{k=2}^{K-1} \log(k) \text{ bits.}$$

Summarized, for a subclonal reconstruction r, we compute the description length $L_{\mathcal{C}_2}(\theta \mid \mathcal{M}) = L_{\mathcal{C}_2}(r)$ as

$$\begin{aligned} L_{\mathcal{C}_2}(r) \quad = \quad & 2 \cdot \log(K-1) + 1 + 2 \cdot \log(I^\bullet) + 1 + I^* \cdot \log(K) + I^* \cdot \log(2) + I^\circ \cdot \log(2) \\ & + I^\circ \cdot \log(2) + J \cdot \log(K-1) + J' \cdot \log(3) + J'' \cdot \log(2) \\ & + (K-1) \cdot N \cdot \log(1000) + \sum_{k=2}^{K-1} \log(k) \text{ bits.} \end{aligned} \tag{3.55}$$

Because we compute $L_{\mathcal{C}_1}(y^m \mid \theta, \mathcal{M}, x^m) = \mathcal{L}'(r)$ with the natural logarithm, we also use the natural logarithm to compute the description length $L_{\mathcal{C}_2}(r)$.

Chapter 4

Dealing with Ambiguity

Given a dataset, we are interested in the subclonal reconstruction r with the highest likelihood. However, it is possible that r is ambiguous, which means that there are other subclonal reconstructions that have the same likelihood as r. To not draw wrong conclusions from a single subclonal reconstruction, we are interested in other subclonal reconstructions with the same likelihood.

In Section 4.1, we define ambiguity in the context of subclonal reconstructions and explain how it arises. In Section 4.2, we explain how we are able to combine multiple subclonal reconstructions with the same likelihood in a single subclonal reconstruction. Finally in Section 4.3, we argue why our lineage-based subclonal reconstructions can deal better with ambiguity caused by lineage relationship inference than population-based subclonal reconstruction methods.

4.1. Defining Ambiguity

In this section, we define ambiguity for subclonal reconstructions and show different reasons for it.

Definition 5 (Ambiguous Subclonal Reconstructions)**.** *A subclonal reconstruction r is* ambiguous *if another subclonal reconstruction r' on the same dataset exists that has the same likelihood as r.*

Observation 1. *Two subclonal reconstructions r and r' on the same dataset have the same likelihood if the same average allele-specific copy numbers $\hat{c}_{A_{i,n}}$ and $\hat{c}_{B_{i,n}}$ as well as VAFs $\hat{p}_{j,n}$ are inferred for each segment i and each SSM j in each sample n for both r and r'.*

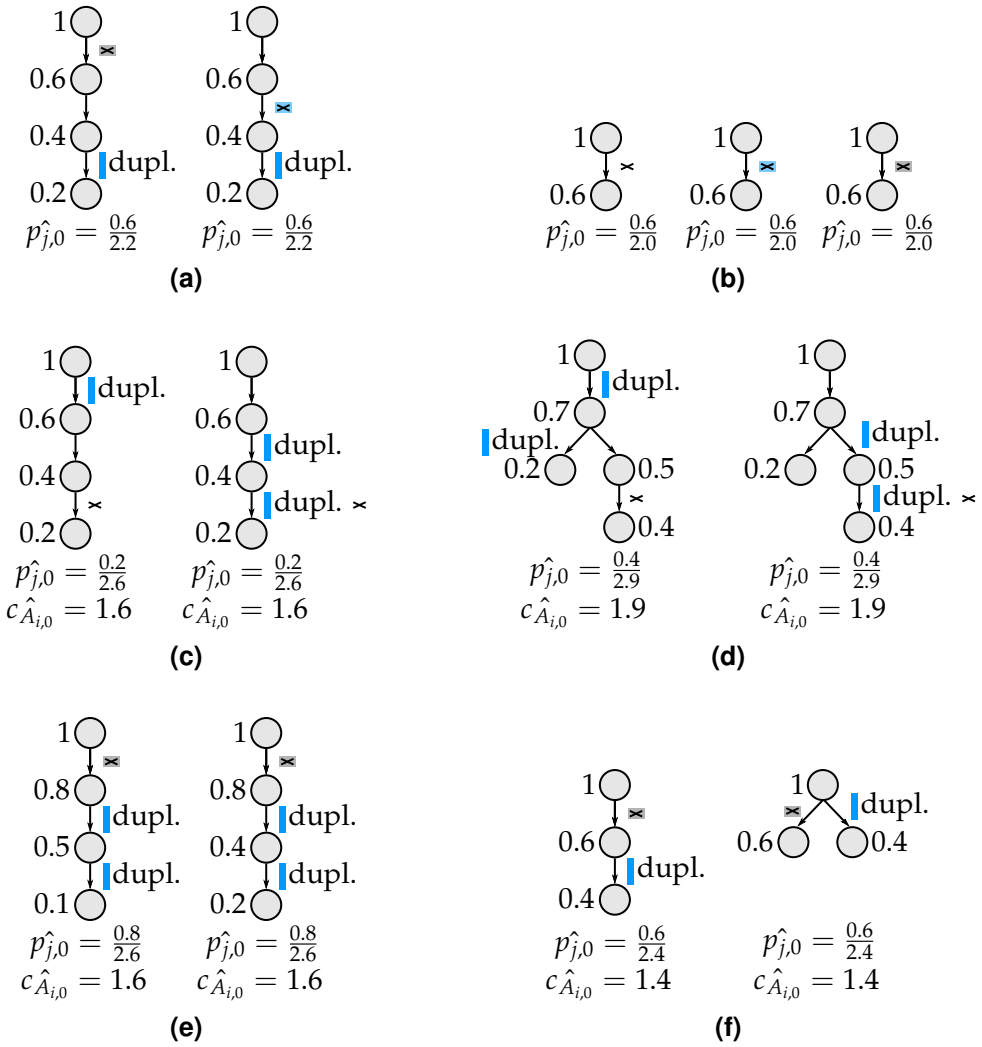


Figure 4.1.: Different ambiguous subclonal reconstructions.
Ambiguous subclonal reconstructions are shown for (a) SSM assignment, (b) SSM phasing, (c) different numbers of CNAs, (d) CNA assignment, (e) lineage frequency inference, and (f) lineage relationship inference.
Phylogenetic trees with lineage frequencies are shown. Cross indicates SSM j, assigned to lineage below, background color indicates phasing: no color: no phasing, blue: phased to allele A, gray: phased to allele B. Copy number gain in segment i, assigned to allele A of following lineage is shown as duplication of allele A with a blue bar.
dupl.: duplication

Observation 1 follows directly from Equation 3.5.
Ambiguity can be caused by all components of a subclonal reconstruction:

1. SSM assignment and phasing:
   An SSM j can be assigned to different lineages (see Figure 4.1a) or can have different phases (see Figure 4.1b) without changing the VAF $\hat{p}_{j,n}$.

2. CNA inference and assignment:
   In a segment i different numbers of CNAs can be inferred (see Figure 4.1c) or the CNAs can be assigned to different lineages (see Figure 4.1d) without changing the average allele-specific copy numbers $\hat{c}_{A_{i,n}}$ and $\hat{c}_{B_{i,n}}$, and without changing the VAF $\hat{p}'$ of all SSMs in segment i.

3. Lineage frequency inference:
   Lineages can have different frequencies without changing the average allele-specific copy numbers $\hat{c}_A$ and $\hat{c}_B$ and VAFs $\hat{p}$ (see Figure 4.1e).

4. Lineage relationship inference:
   Lineage relationships can be different without changing the average allele-specific copy numbers $\hat{c}_A$ and $\hat{c}_B$ and VAFs $\hat{p}$ (see Figure 4.1f).

4.2. Handling Ambiguity

To reduce the number of ambiguous subclonal reconstructions, we remove phasing ambiguity (see Subsection 3.2.4, page 38). Also, if an ancestor-descendant relationship between two lineages has no influence on the likelihood and is not crucial for a subclonal reconstruction to be valid, we model it as ambiguous relationship. This allows us to combine ambiguous subclonal reconstructions in a single one.

In our MILP, we work only with present and absent ancestor-descendant relationships but not with ambiguous ones (see Subsection 3.3.3). Thus, after the optimization, we need to identify which present and absent relationships are really necessary and which are actually ambiguous.

Definition 6 (Necessary Present and Absent Ancestor-Descendant Relationships)**.** *Present or absent relationships are* necessary *if they either have an influence on the likelihood or if they are crucial in order for a subclonal reconstruction to be valid.*

Definition 7 (Present Ancestor-Descendant Relationships Influencing the Likelihood)**.** *A present ancestor-descendant relationship between two lineages k and k', with $k < k'$, has an*

influence on the likelihood if a copy number change of lineage k' influences the average copy number of an SSM of lineage k in the same segment.

A copy number change of lineage k' influences the average copy number of an SSM of lineage k in the same segment if the copy number change and the SSM are phased to the same allele.

Definition 8 (Present Ancestor-Descendant Relationships Crucial for Subclonal Reconstruction)**.** *A present ancestor-descendant relationship $Z_{k,k'}$ between two lineages k and k' is crucial for the validity of a subclonal reconstruction if*

a) k is the normal lineage (see Equation 3.13),

b) the phylogenetic tree rules (see Equations 3.16 and 3.17) were violated if the relationship was not present, or

c) the sum rule (see Equation 3.8) was violated if the relationship was not present.

Definition 9 (Absent Ancestor-Descendant Relationships Influencing the Likelihood)**.** *An absent ancestor-descendant relationship between two lineages k and k', with $k < k'$, has an influence on the likelihood if a copy number change of lineage k' changed the average copy number of an SSM of lineage k in the same segment if lineage k was an ancestor of lineage k'.*

If an SSM of lineage k is phased, a copy number change of lineage k' can have an influence on its average copy number only if it is phased to the same allele (see Figure 4.2a). If an SSM is unphased, copy number changes on both alleles can influence its average copy number as well. This is because the SSM is actually present on one of the two alleles, we just cannot say on which one. Thus, if lineage k' has copy number changes on both alleles, one copy number change can influence the average copy number of the unphased SSM (see Figure 4.2b). If lineage k' has only one copy number change on allele α but is in an ancestor-descendant relationship with another lineage $k^\bullet$, with $k < k^\bullet$, that has a copy number change on allele β, with $\alpha \neq \beta$, in the same segment, the average copy number of the unphased SSM can also be influenced (see Figure 4.2c). Even if $k^\bullet < k$, the average copy number of the SSM can be influenced if the copy number changes are copy number losses. The reason is that if lineage k' was a descendant of lineage k, lineage k would be a descendant of lineage $k^\bullet$, following Equation 3.17. Thus, the unphased SSM would have to be phased to allele α, following Equation 3.36. Hence, its average copy number would be changed by the copy number loss in lineage k' (see Figure 4.2d). Note that if in this case the copy number changes

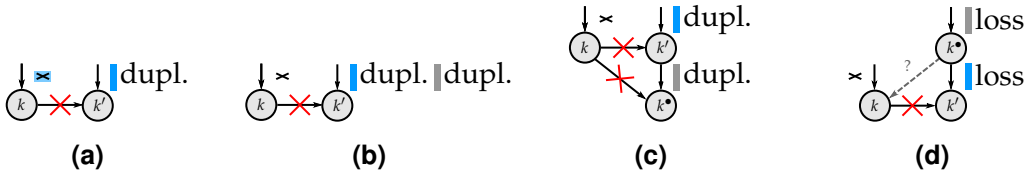


Figure 4.2.: Different scenarios with absent ancestor-descendant relationships influencing the likelihood.
Mutation assignments within a single segment are shown for lineages k, k' and $k^\bullet$, with $k < k'$. Absent ancestor-descendant relationships are shown by black arrows with red cross. Arrows without circle on top indicate ancestor-descendant relationships to lineages not shown in the figure. (a) Lineage k contains an SSM that is phased to the same allele as the copy number change of lineage k'. (b) Lineage k contains an unphased SSM whose average copy number can be influenced by the two copy number changes on both alleles of lineage k'. (c) The average copy number of the unphased SSM of lineage k can be influenced by the copy number changes on different alleles of lineages k' and $k^\bullet$. As long as $k < k^\bullet$, it does not matter whether $k' < k^\bullet$ or $k' > k^\bullet$. (d) If $k^\bullet < k < k'$, copy number changes on different alleles of lineages $k^\bullet$ and k' can influence an unphased SSM in lineage k if they are copy number losses. Note that the ancestor-descendant relationship between lineages $k^\bullet$ and k alone does not change the average copy number of the SSM.
dupl.: duplication

are copy number gains, the average copy number of the SSM cannot be influenced since a gain in an ancestral lineage does not influence the phasing of an SSM.

Definition 10 (Absent Ancestor-Descendant Relationships Crucial for Subclonal Reconstruction)**.** *An absent ancestor-descendant relationship Z_{k,k° between two lineages k and k° is crucial for the validity of a subclonal reconstruction if*

a) $k = k^\circ$ (see Equation 3.12),

b) $k^\circ < k$ (see Equation 3.14),

c) the sum rule (see Equation 3.8) was violated if lineage k was the parent of lineage k° and lineage k does not have any other descendant that are allowed to be a parent of lineage k°,

d) the crossing rule (see Equation 3.15) was violated if the relationship was present,

e) the same allele in the same segment is deleted in lineages k and k° (see Equation 3.23), or

f) an SSM of lineage k° would be assigned to an allele already lost in lineage k (see Equation 3.36) if the relationship was present.

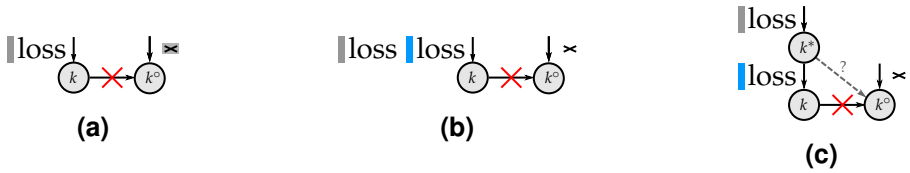


Figure 4.3.: Different scenarios with absent ancestor-descendant relationships crucial for subclonal reconstructions.
Mutation assignments within a single segment is shown for lineages k^*, k and k°, with $k^* < k < k^\circ$. (a) The SSM of lineage k° is assigned to the same phase as the copy number loss of lineage k. (b) Since lineage k loses both alleles, the unphased SSM of lineage k° could not be assigned to any allele if lineage k° was a descendant if lineage k. (c) If lineage k° was an ancestor of lineage k, its SSM could not be assigned to any allele since both allele got lost. Note that lineage k^* is allowed to be an ancestor of lineage k°. Then the SSM would be phased to the present allele.

Considering case f) of Definition 10, whether an SSM of lineage k° would be assigned to an allele already lost in lineage k if the relationship was present depends on the phasing of the SSM, the copy number change assignment of lineage k and the present ancestor-descendant relationships lineage k has to other lineages. If the SSM is phased to allele α, α has to get lost in lineage k (see Figure 4.3a). If the SSM is unphased, either both alleles have to get lost in lineage k (see Figure 4.3b), or one allele has to get lost in lineage k and the other in an ancestral lineage k^*, with $k^* < k < k^\circ$ (see Figure 4.3c). Note that if the SSM is unphased and lineage k does not have an ancestor with a copy number loss in the same segment assigned to the different allele β, lineage k could be an ancestor of lineage k°. Then, the SSM would simply be phased to allele β following Equation 3.36.

Note that there are cases in which a present ancestor-descendant relationship between lineages k and k' is necessary to fulfill the sum rule, and cases in which an absent ancestor-descendant relationship is necessary to fulfill the sum rule. Whether one of the two relationships is necessary depends on whether lineage k is currently the parent of lineage k' and which other lineages can be parents of lineage k' as well. If lineage k is currently the parent of lineage k', and lineage k' cannot be the child of any other lineage k°, with $k^\circ < k$, the ancestor-descendant relationship between lineages k and k' needs to be present (see Figure 4.4a). If lineage k' is already the child of other lineage k°, if the sum rule was violated if lineage k was a parent of lineage k', and lineage k does not have any descendants that are allowed to be parents of lineage k', the ancestor-descendant relationship between lineages k and k' needs to be absent (see Figure 4.4b). If lineage k has descendants that are allowed to be parents

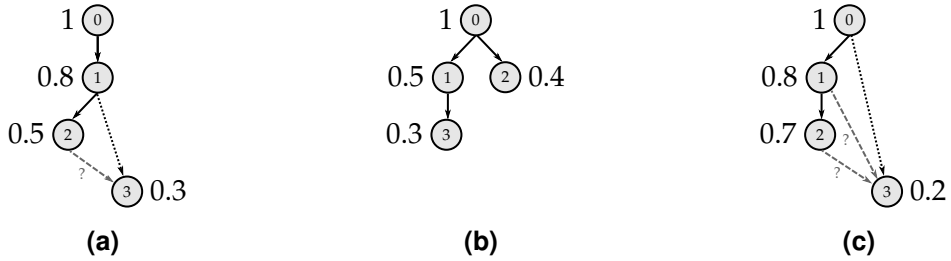


Figure 4.4.: Different scenarios in which the ancestor-descendant relationship between two lineages needs to be (a) present or (b) absent, or (c) can be ambiguous in order to fulfill the sum rule.
Lineages are shown together with indices and frequencies. Solid black arrows between lineages indicate parent-child relationships, dotted black arrows indicate ancestor-descendant relationships, gray dashed arrows with a question mark indicate ambiguous relationships. If two lineages are not connected via an arrow or path of arrows, they are not in an ancestor-descendant relationship. (a) The ancestor-descendant relationship between lineages 1 and 3 needs to be present because the sum rule was violated if lineage 3 was a child of lineage 0. (b) The ancestor-descendant relationship between lineages 1 and 2 needs to be absent since the sum rule was violated if lineage 2 was a child of lineage 1 and it is not possible that lineage 2 becomes a child of lineage 3. (c) The ancestor-descendant relationship between lineages 1 and 3 can be ambiguous. Although lineage 3 is not allowed to be a child of lineage 1 as it would violate the sum rule, lineage 3 can become a child of lineage 2, thus making lineage 1 one of its ancestors.

of lineage k', lineage k can be an ancestor of lineage k'. Thus, the two lineages can be in an ambiguous ancestor-descendant relationship (see Figure 4.4c).

Finding Ambiguous Ancestor-Descendant Relationships. We developed an algorithm that finds ambiguous ancestor-descendant relationships after the optimization (called *ambiguity algorithm* from now on). It investigates each relationship $Z_{k,k'}$, for $0 < k < k' < K$, since these relationships can be ambiguous. The ambiguity algorithm consists of seven main steps:

1. Present ancestor-descendant relationships between cancerous lineages that do not have an influence on the likelihood are transformed into ambiguous relationships.

2. Absent ancestor-descendant relationships are transformed into ambiguous relationships.

3. All relationships of each combination of three lineages k, k' and k'', with $0 \leq k < k' < k'' < K$, are checked to fulfill the phylogenetic tree rules and updated if required.

4. SSMs that do not need to be phased are unphased.

5. Absent ancestor-descendant relationships that are necessary because of the crossing rule or mutation assignments are identified and the corresponding ambiguous relationships are transformed back into absent ones.

6. Present ancestor-descendant relationships that are necessary because of the sum rule are identified and the corresponding ambiguous relationships are transformed into present ones.

7. Absent ancestor-descendant relationships that are necessary because of the sum rule are identified and the corresponding ambiguous relationships are transformed into absent ones.

After step 7, the ambiguous ancestor-descendant relationships left are true ambiguous relationships that can be updated to present or absent relationships without changing the likelihood or leading to an invalid subclonal reconstruction.

Step 2 is implemented straight-forward. The other steps are more complex and are presented and explained in detail in Subsections 4.2.1 to 4.2.6.

4.2.1. Finding Present Ancestor-Descendant Relationships Necessary because of Likelihood Influence

All ancestor-descendant relationships that are necessary because they have an influence on the likelihood (see Definition 7) are found with Algorithm 1. First, it is checked for all present pairwise ancestor-descendant relationships between cancerous lineages whether the descendant lineage contains a copy number change in any segment i that is phased to the same allele as any SSM in segment i of the ancestral lineage (lines 1 – 7). If this is the case, the present ancestor-descendant relationship is kept, otherwise it is transformed into an ambiguous relationship (lines 7 – 13). Ancestor-descendant relationships that need to be present because they are crutial for the subclonal reconstruction are found in later steps of the ambiguity algorithm.

Algorithm 1 Finding all necessary ancestor-descendant relationships influencing the likelihood

Input: subclonal reconstruction r with lineage number K, lineage relationship matrix Z, CNA assignment with phasing, and SSM assignment with phasing

Output: subclonal reconstruction r, containing present ancestor-descendant relationships only if they influence the likelihood

```
for k ← 1, ..., K − 2 do
  for k′ ← k + 1, ..., K − 1 do
    if Z_{k,k′} = 1 then
      relationship_present ← false
      for each segment i that has at least one copy number change in lineage k′ do
        for each copy number change i′ of lineage k′ in segment i do
          if lineage k has SSMs in segment i that are phased to the same allele as copy number change i′ then
            relationship_present ← true
            break
        if relationship_present is true then
          break
      if relationship_present is false then
        Z_{k,k′} = ?
return r
```

4.2.2. Updating Lineage Relationships

To describe a proper phylogenetic tree, the ancestor-descendant relationships of all combinations of three lineages k, k' and k'' have to be consistent with Equations 3.16 and 3.17. Given that an entry in the lineage relationship matrix Z can take three possible values, there are $3^3 = 27$ relationship combinations. In Table 4.1, we show all 27 relationship combinations and indicate, whether they are consistent with Equations 3.16 and 3.17 and whether they lead to an update of ambiguous ancestor-descendant relationships into present or absent ones.

Table 4.1.: All 27 lineage relationship combinations for three lineages. Relationships for lineages k, k' and k'', with $0 \leq k < k' < k'' < K$, are shown in an excerpt of the lineage relationship matrix Z and with a graphical representation. For each combination it is indicated whether it is consistent with the phylogenetic tree rules.
Eq.: Equations

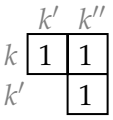
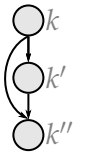
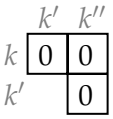
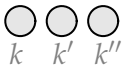
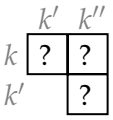
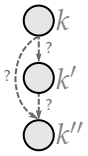
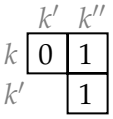
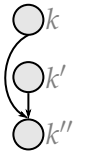
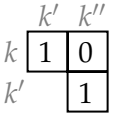
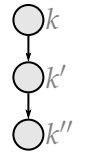
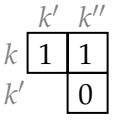
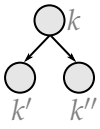
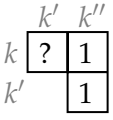
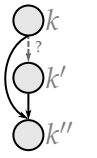
	lineage relationships	graphical representation	consistent with Eq. 3.16 and 3.17
1.			yes
2.			yes
3.			yes
4.			no, violates Equation 3.17
5.			no, violates Equation 3.16
6.			yes
7.			yes $Z_{k,k'}$ can be updated to 1 because $Z_{k,k'} = 0$ would violate Equation 3.17

Table 4.1.: All 27 lineage relationship combinations for three lineages. Continued.

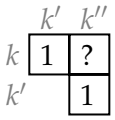
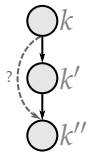
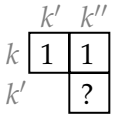
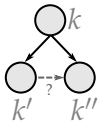
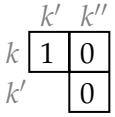
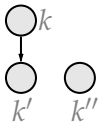
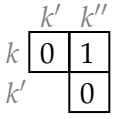
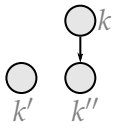
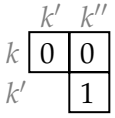
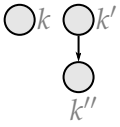
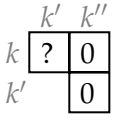
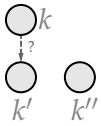
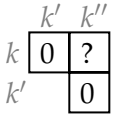
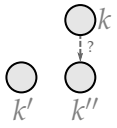
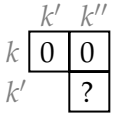
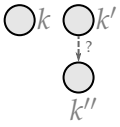
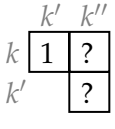
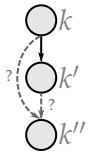
	lineage relationships	graphical representation	consistent with Eq. 3.16 and 3.17
8.			yes $Z_{k,k''}$ can be updated to 1 because $Z_{k,k''} = 0$ would violate Equation 3.16
9.			yes
10.			yes
11.			yes
12.			yes
13.			yes
14.			yes
15.			yes
16.			yes

Table 4.1.: All 27 lineage relationship combinations for three lineages. Continued.

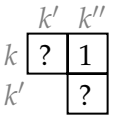
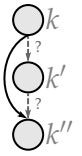
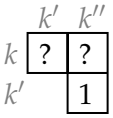
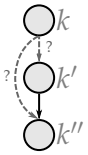
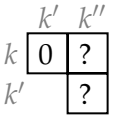
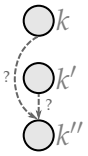
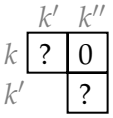
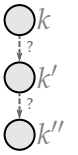
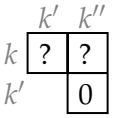
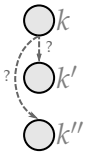
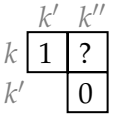
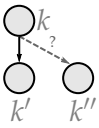
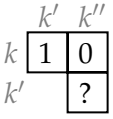
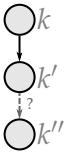
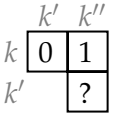
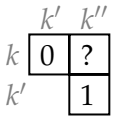
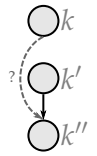
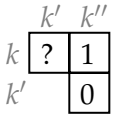
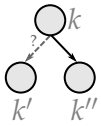
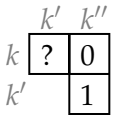
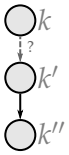
	lineage relationships	graphical representation	consistent with Eq. 3.16 and 3.17
17.			yes
18.			yes
19.			yes
20.			yes
21.			yes
22.			yes
23.			yes $Z_{k',k''}$ can be updated to 0 because $Z_{k',k''} = 1$ would violate Equation 3.16
24.			yes $Z_{k',k''}$ can be updated to 0 because $Z_{k',k''} = 1$ would violate Equation 3.17

Table 4.1.: All 27 lineage relationship combinations for three lineages. Continued.

	lineage relationships	graphical representation	consistent with Eq. 3.16 and 3.17
25.			yes $Z_{k,k''}$ can be updated to 0 because $Z_{k,k''} = 1$ would violate Equation 3.17
26.			yes
27.			yes $Z_{k,k''}$ can be updated to 0 because $Z_{k,k''} = 1$ would violate Equation 3.16

In step 1 of the ambiguity algorithm, present ancestor-descendant relationships that do not have an influence on the likelihood are transformed into ambiguous relationships (see Subsection 4.2.1). In step 2, absent ancestor-descendant relationships are transformed into ambiguous relationships as well. Thus, at the beginning of step 3, the upper right triangle of the lineage relationship matrix Z consists only of 1s and ?s. Now in step 3, it is investigated for each combination of three lineages in this triangle whether they are updated by the phylogenetic tree rules, meaning that an ambiguous relationship is transformed back into a present one. Whenever relationships are updated, it is checked whether this update has an impact on already processed relationships. At the end of step 3, the subclonal reconstruction fulfills the phylogenetic tree rules again.

Note that in step 3, only ambiguous ancestor-descendant relationships that were inferred as present are updated to present relationships. Ambiguous relationships that were inferred as absent cannot be updated to absent relationships at this stage since for this update one of the three lineage relationships would have to be absent already (see combinations 23, 24, 25 and 27). Also, it is not possible here that a relationship that was inferred as absent is updated to a present relationship. This is because if a relationship inferred as absent has to be a present relationship to fulfill the phylogenetic tree rules,

it would have been already inferred as present during the optimization. Otherwise, the found subclonal reconstruction would have violated the phylogenetic tree rules.

Since in step 3 only ambiguous relationships are updated to present relationships that were present before, no update can lead to incompatibilities with the phylogenetic tree rules and also the phasing of SSMs is correct. However, if ancestor-descendant relationships should be updated at later stages of the ambiguity algorithm, it is possible that such an update is not possible or that the SSM phasing has to be changed. Algorithm 2 shows how the relationship between two lineages k and k' is updated to v, with $v \in \{0, 1, ?\}$. If v equals the current relationship of lineages k and k', or if v is the ambiguous relationship, nothing needs to be updated (lines 1 – 2). If lineages k and k' are already in a non-ambiguous relationship that is different from v, the update is not compatible with the subclonal reconstruction and hence not possible (lines 3 – 4). If the relationship between lineages k and k' should be updated to a present relationship but the SSM phasing does not allow this, the update is not possible as well (lines 5 – 9). If the current phasing of SSMs allows the update to a present relationship, unphased SSMs of lineages k and k' are phased in all segments in which the other lineage contains copy number changes that would influence the SSMs if they were phased to the other allele (lines 12 – 15). Afterwards, the relationship can be updated (line 18). All other ancestor-descendant relationships that are affected by the relationship change are tried to be updated as well (lines 19 – 20). If all updates were successful, the updated subclonal reconstruction is returned (line 21).

Note that Algorithm 2 allows in theory an update to a present relationship if one lineage has unphased SSMs in a segment in which the other lineage has copy number changes on both alleles. Thus, an ambiguous relationship that is required to be an absent one could be updated to a present one. In practice, however, this case does not arise since we identify all necessary absent ancestor-descendant relationships before updating to present relationships which were not inferred as present during the optimization.

4.2.3. Unphasing Simple Somatic Mutations

In our optimization, SSMs get phased only if in the same segment a copy number change happens in a descendant lineage (see Equation 3.35), a copy number loss is assigned to an ancestral lineage or the lineage with the SSMs itself (see Equation 3.36), or the SSMs are influenced by copy number gains of their own lineage (compare Equation 3.34). Since present ancestor-descendant relationships between lineages got re-

Algorithm 2 Updating lineage relationship matrix and SSM phasing

Input: subclonal reconstruction r with lineage number K, lineage relationship matrix Z, CNA assignment with phasing, and SSM assignment with phasing,
lineage indices k, k', with $k < k'$,
relationship value $v \in \{0, 1, ?\}$
Output: subclonal reconstruction r with $Z_{k,k'} = v$

if $v = ?$ **or** $Z_{k,k'} = v$ **then**
  return r
else if $Z_{k,k'} \neq v$ **and** $Z_{k,k'} \neq ?$ **then**
  raise update not possible
else if $v = 1$ **then**
  for each segment i that has at least one copy number change in lineages k or k' **do**
    for each copy number change i' in segment i of lineage k°, with $k^\circ \in \{k, k'\}$, **do**
      if $k^\circ = k'$ **and** lineage k has SSMs in segment i phased to allele of i' **then**
        raise update not possible ▷ *see Definition 9*
      if $k^\circ = k$ **and** i' is copy number loss **and** lineage k' has SSMs in segment i phased to allele of i' **then**
        raise update not possible ▷ *see Definition 10*
  for each segment i that has at least one copy number change in lineages k or k' **do**
    for each copy number change i' in segment i of lineage k°, with $k^\circ \in \{k, k'\}$, **do**
      if $k^\circ = k'$ **and** lineage k has unphased SSMs in segment i **then**
        move unphased SSMs to allele not affected by i'
      if $k^\circ = k$ **and** i' is copy number loss **and** lineage k' has unphased SSMs in segment i **then**
        move unphased SSMs to allele not affected by i'
$Z_{k,k'} = v$
for each entry $Z_{i,i'}$, with either i or $i' \in \{k, k'\}$, that should get updated to $v' \in \{0, 1\}$ because of $Z_{k,k'} = v$ ▷ *see Table 4.1* **do**
  update Z and SSM phasing for r, i, i', and v' ▷ *see Algorithm 2*
return r

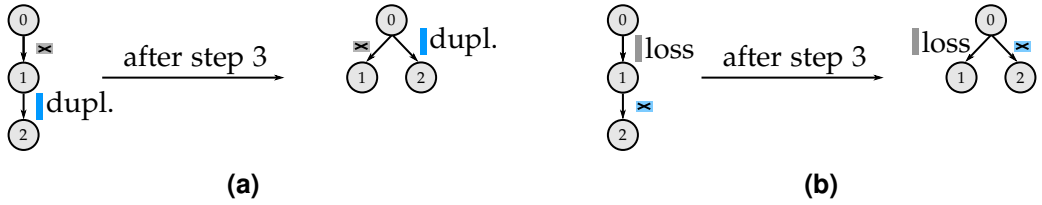


Figure 4.5.: Subclonal reconstructions with unnecessary SSM phasing. Left side of subfigures shows subclonal reconstruction after optimization, right side after step 3 of ambiguity algorithm. (a) SSM phasing is necessary if lineage with at least one SSM is ancestor of lineage with copy number change in the same segment. The phasing is not necessary if the two lineages are not in an ancestor-descendant relationship. (Note that if the SSM was phased to the same allele as the copy number change, the ancestor-descendant relationship would not have been removed in step 1 of the algorithm.) (b) SSM phasing is necessary if lineage with at least one SSM is descendant of lineage with copy number loss in the same segment. The phasing is not necessary if the two lineages are not in an ancestor-descendant relationship.
dupl.: duplication

moved in step 1 of the ambiguity algorithm, it is possible that SSMs are phased after step 3 although relationships that caused the phasing are absent (see Figure 4.5).

To provide a subclonal reconstruction with unambiguous SSM phasing, SSMs that are not required to be phased anymore get unphased in step 4, shown in Algorithm 3. For each segment with phased SSMs, separately for all cancerous lineages, the phasing is analyzed (lines 1 – 4). If SSMs are not influenced by copy number changes (lines 7 – 17), they are unphased (lines 18 – 21).

4.2.4. Identifying Absent Ancestor-Descendant Relationships Necessary because of Crossing Rule and Mutation Assignment

After step 4 of the ambiguity algorithm, ancestor-descendant relationships between lineages of the subclonal reconstruction r are either present or ambiguous. Now, in step 5, absent ancestor-descendant relationships that are necessary because of the crossing rule or mutation assignment are found with Algorithm 6. First, it is checked for each ambiguous lineage relationship $Z_{k,k'}$, with $0 < k < k' < K$, whether the crossing rule was violated if lineages k and k' were in a present ancestor-descendant relationship (lines 1 – 7). If it was violated, the relationship is transformed to an absent relationship. If the change of the relationship has an influence on other ambiguous relationships (see Table 4.1), they are updated as well. Now the algorithm iterates over each segment i with copy number changes and each lineage k, having at least one copy

Algorithm 3 Unphasing SSMs that do not need to be phased

Input: subclonal reconstruction r with lineage number K, lineage relationship matrix Z, CNA assignment with phasing, and SSM assignment with phasing
Output: subclonal reconstruction r whose SSMs are phased only if needed

for $k \leftarrow 1, \ldots, K-1$ **do**
  $\mathcal{A}_k \leftarrow$ indices of ancestors of lineage k according to Z ▷ *see Algorithm 4*
  $\mathcal{D}_k \leftarrow$ indices of descendants of lineage k according to Z ▷ *see Algorithm 5*
  for each segment i that has phased SSMs assigned to lineage k **do**
    keep_phased_A $\leftarrow$ **false**
    keep_phased_B $\leftarrow$ **false**
    if any lineage k°, with $k^\circ \in \mathcal{A}_k$, or lineage k have at least one copy number loss in segment i **then**
      keep_phased_A $\leftarrow$ **true**
      keep_phased_B $\leftarrow$ **true**
    else if any lineage k', with $k' \in \mathcal{D}_k$, has at least one copy number change in segment i **then**
      keep_phased_A $\leftarrow$ **true**
      keep_phased_B $\leftarrow$ **true**
    else if lineage k has at least one copy number gain in segment i **then**
      if one copy number gain is phased to allele A **and** at least one SSM in lineage k is influenced by it **then**
        keep_phased_A $\leftarrow$ **true**
      if one copy number gain is phased to allele B **and** at least one SSM in lineage k is influenced by it **then**
        keep_phased_B $\leftarrow$ **true**
    if *keep_phased_A* is **false then**
      unphase SSMs of lineage k in segment i that are phased to allele A
    if *keep_phased_B* is **false then**
      unphase SSMs of lineage k in segment i that are phased to allele B
return r

Algorithm 4 Getting all ancestors

Input: lineage relationship matrix Z,
  lineage index k
Output: stack A which contains indices of all ancestors of lineage k

create empty stack A
for $k^\circ \leftarrow 0, \ldots, k-1$ **do**
  if $Z_{k^\circ,k} = 1$ **then**
    A.push(k°)
return A

Algorithm 5 Getting all descendants

Input: lineage relationship matrix Z,
lineage index k,
number of lineages K
Output: stack D which contains indices of all descendants of lineage k

create empty stack D
for $k' \leftarrow k+1, \ldots, K-1$ **do**
  if $Z_{k,k'} = 1$ **then**
    D.push(k')
return D

Algorithm 6 Finding all necessary absent ancestor-descendant relationships

Input: subclonal reconstruction r with lineage number K, lineage relationship matrix Z, CNA assignment with phasing, and SSM assignment with phasing
Output: subclonal reconstruction r with necessary absent ancestor-descendant relationships

for $k \leftarrow 1, \ldots, K-2$ **do**
  for $k' \leftarrow k+1, \ldots, K-1$ **do**
    if $Z_{k,k'} = ?$ **then**
      if crossing rule was violated for $Z_{k,k'} = 1$ **then**
        $Z_{k,k'} = 0$
        for each entry $Z_{i,i'}$, with either i or $i' \in \{k, k'\}$, that should get updated to $v' = 0$ because of $Z_{k,k'} = 0$ ▷ *see Table 4.1* **do**
          update Z and SSM phasing for r, i, i', and v' ▷ *see Algorithm 2*
for each segment i with at least one copy number change **do**
  for each lineage k containing at least one copy number change in segment i **do**
    for $k^* \leftarrow 1, \ldots, k-1$ **do**
      if $Z_{k^*,k} = ?$ **then**
        if ancestor-descendant relationship between lineages k^* and k influences the likelihood ▷ *see Definition 9* **then**
          $Z_{k^*,k} = 0$
          for each entry $Z_{i,i'}$, with either i or $i' \in \{k^*, k\}$, that should get updated to $v' = 0$ because of $Z_{k^*,k} = 0$ ▷ *see Table 4.1* **do**
            update Z and SSM phasing for r, i, i', and v' ▷ *see Algorithm 2*
    for $k' \leftarrow k+1, \ldots, K-1$ **do**
      if $Z_{k,k'} = ?$ **then**
        if absent ancestor-descendant relationship between lineages k and k' is crucial for validity of subclonal reconstruction ▷ *see Definition 10* **then**
          $Z_{k,k'} = 0$
          for each entry $Z_{i,i'}$, with either i or $i' \in \{k, k'\}$, that should get updated to $v' = 0$ because of $Z_{k,k'} = 0$ ▷ *see Table 4.1* **do**
            update Z and SSM phasing for r, i, i', and v' ▷ *see Algorithm 2*
return r

number change in segment i. Necessary absent ancestor-descendant relationships between lineage k and lineages with lower and higher index, with which lineage k is in an ambiguous relationship, are found by checking the criteria of Definitions 9 and 10 (lines 8 – 21). If an absent ancestor-descendant relationship is found, the relationship is updated and if needed, other relationships are updated as well as. After all segments are processed, r is returned (line 22).

Note that relationships Z_{k,k° that need to be absent because $k = k^\circ$ or $k^\circ < k$ do not have to be found because they were not transformed into ambiguous relationships in step 2 of the ambiguity algorithm. Relationships that need to be absent because of the sum rule are found later in step 7 (see Subsection 4.2.6), after relationships that need to be present because of the sum rule were found in step 6 (see Subsection 4.2.5).

If after Algorithm 6 ambiguous ancestor-descendant relationships are still present, we check whether the lineage relationship matrix Z contains *necessary unfolding* absent relationships.

Definition 11 (Necessary Unfolding Absent Ancestor-Descendant Relationships)**.** Necessary unfolding absent *ancestor-descendant relationships are ambiguous relationships between three lineages k, k' and k'', with $0 < k < k' < k'' < K$, that are not allowed to be present at the same time because either the likelihood would change (see Definition 9) or the subclonal reconstruction would be invalid since an SSM would be assigned to an allele already deleted (see case f) in Definition 10).*

Observation 2. *Necessary unfolding absent ancestor-descendant relationships arise from three cases:*

1. *Lineage k contains an unphased SSM in the same segment in which lineages k' and k'' have copy number changes on different alleles. Either, all relationships are ambiguous, or the relationship between lineages k' and k'' is absent and the other two are ambiguous (see Figure 4.6a).*

2. *Lineage k' contains an unphased SSM in the same segment, in which lineages k and k'' have copy number losses on different alleles. All relationships are ambiguous (see Figure 4.6b).*

3. *Lineage k'' contains an unphased SSM in the same segment, in which lineages k and k' have copy number losses on different alleles. All relationships are ambiguous (see Figure 4.6c).*

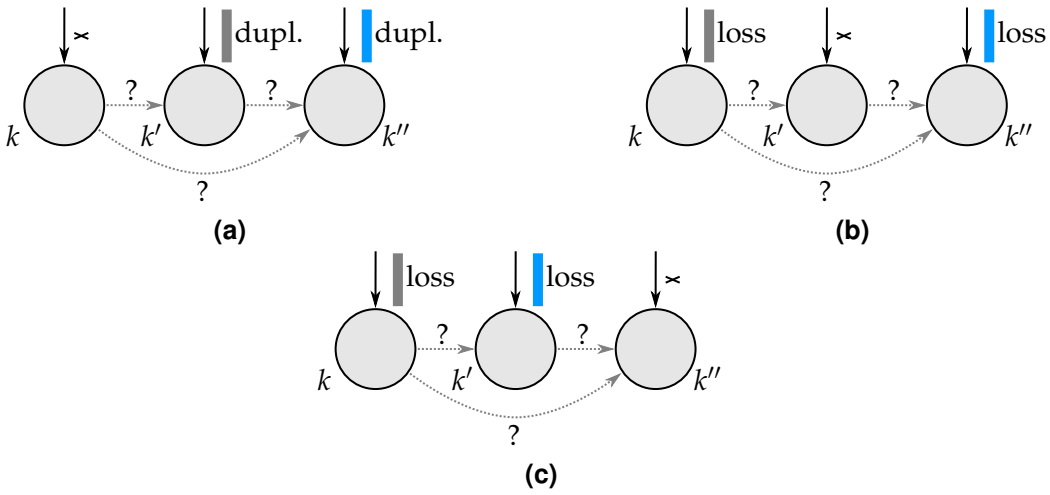


Figure 4.6.: Three different cases from which necessary unfolding absent ancestor-descendant relationships arise.
Lineages k, k' and k'', with $0 < k < k' < k'' < K$, are shown with different copy number change and unphased SSM assignment. (a) A scenario with copy number losses leads to necessary unfolding absent ancestor-descendant relationships as well. The relationship between lineages k' and k'' can also be absent. (b,c) In these scenarios, copy number gains do not lead to necessary unfolding absent ancestor-descendant relationships since copy number gains of ancestral lineages do not influence the phasing of an SSM.
dupl.: duplication

The ambiguous relationships between the lineages are not allowed to be present at the same time. The reasons are the following for the three cases:

1. If lineage k was an ancestor of both lineages k' and k'', the SSM had to be phased. Thus, its average copy number would be influenced by one copy number change and the likelihood would change. It does not matter whether lineages k' and k'' are in an ancestor-descendant relationship or not.

2. If lineage k was an ancestor of lineage k' and if lineage k'' was a descendant of lineage k', the SSM had to be phased to the allele not deleted in lineage k. Hence, its average copy number would be influenced by the copy number loss of lineage k'' and the likelihood would change.

3. If lineages k and k' were ancestors of lineage k'', both alleles were deleted. However, the SSM is not allowed to be assigned to a lost allele.

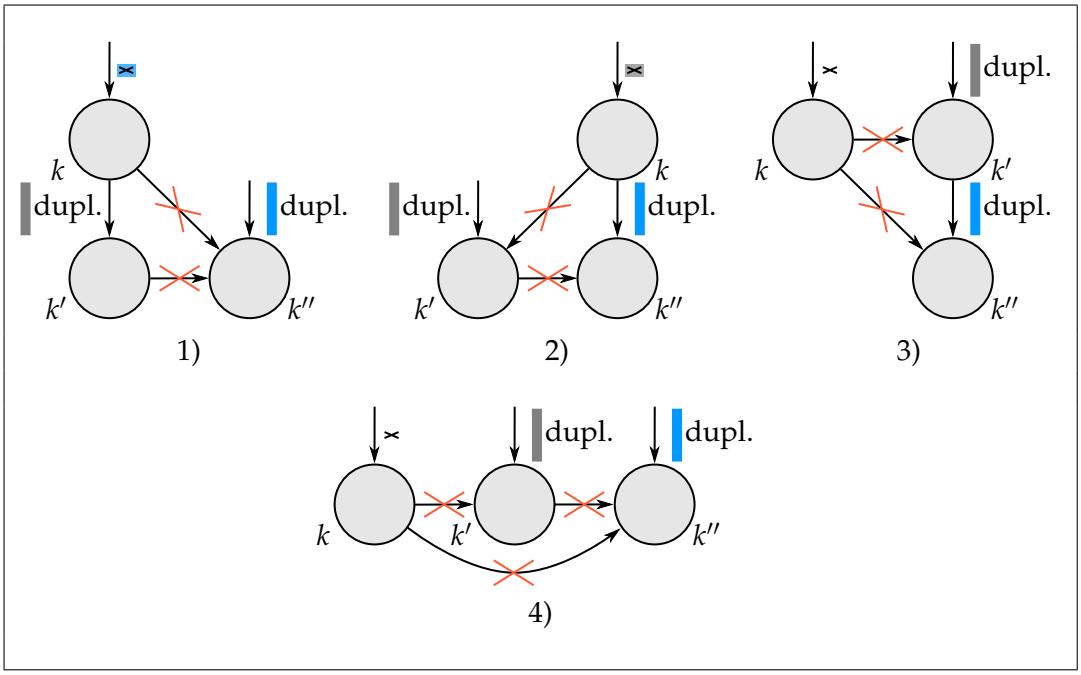
The three cases arise only if the SSM is unphased and if all three relationships are ambiguous, with exception of case 1 where the relationship between lineages k' and k'' can also be absent. If the SSM was phased, the phasing would have been considered in the likelihood and in the subclonal reconstruction during the optimization. All absent ancestor-descendant relationships necessary because of mutation assignment would have been found already with Algorithm 6. If the two lineages with copy number changes were in a present ancestor-descendant relationship before Algorithm 6, necessary absent ancestor-descendant relationships to the lineage with the unphased SSM would have been already found (compare Figures 4.2c, 4.2d and 4.3c). If the lineage with the SSM was in a present ancestor-descendant relationship with one of the lineages with copy number changes before, the SSM would have been phased already. If one relationship was absent, except the one of case 1, the SSM could only be influenced by the copy number change of one lineage, and hence can be phased to the allele not affected. This would not change the likelihood nor would it lead to an invalid subclonal reconstruction.

To prevent that all necessary unfolding absent ancestor-descendant relationships between three lineages are transformed to present relationships in step 6 of the ambiguity algorithm (see Subsection 4.2.5), we fork the lineage relationship matrix Z to multiple matrices. Each new matrix contains one valid relationship setting between the three lineages.

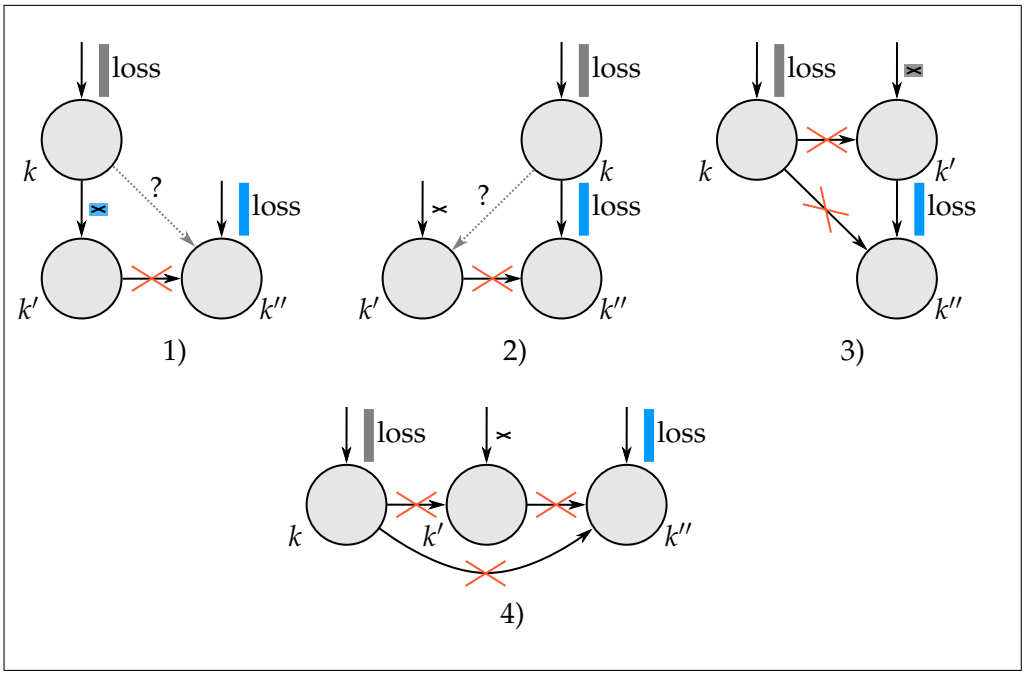
Definition 12 (Relationship Setting)**.** *A* relationship setting f *contains three relationship values* $f[0]$, $f[1]$, $f[2] \in \{0, 1, ?\}$ *to which the ancestor-descendant relationships* $Z_{k,k'}$, $Z_{k,k''}$ *and* $Z_{k',k''}$ *of the three lineages* k, k' *and* k'' *should get updated.*

We derive the different settings by separately transforming each ambiguous relationship to a present relationship, while transforming only the other ambiguous relationships to absent ones that would change the likelihood or would lead to an invalid subclonal reconstruction if they were present. The last possible valid setting is the one in which all ambiguous relationships are transformed to absent relationships (see Figure 4.7).

Algorithm 7 shows how we fork the lineage relationship matrix Z to multiple matrices if necessary unfolding absent ancestor-descendant relationships exist. First, an empty stack R is created into which a duplicate of the subclonal reconstruction r, which is going to be investigated, is inserted (lines 1 – 2). Like that, all following operations are performed on the duplicate and the original subclonal reconstruction including the lineage relationship matrix Z are not changed. Necessary unfolding

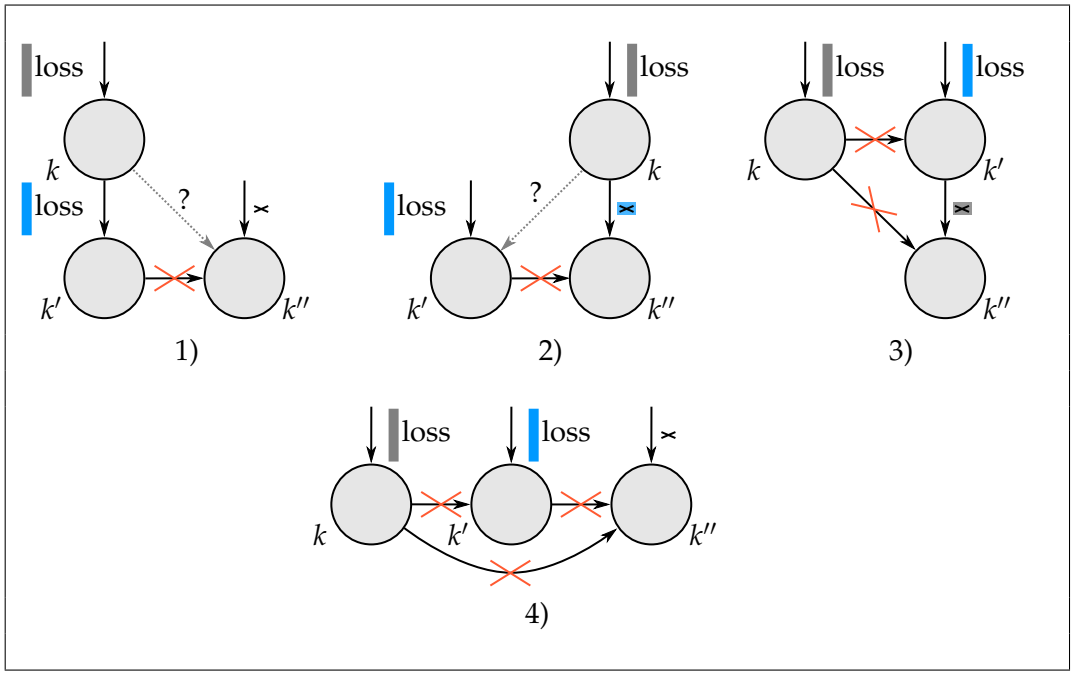


(a)



(b)

Figure 4.7.: Different relationship settings for necessary unfolding ancestor-descendant relationships.
Full caption see next page.



(c)

Figure 4.7.: Different relationship settings for necessary unfolding ancestor-descendant relationships.
Lineages k, k' and k'', with $0 < k < k' < k'' < K$, are shown with different copy number change and SSM assignment. The four relationship settings are shown for the three different cases of necessary unfolding absent ancestor-descendant relationships of Observation 2: (a) case 1, (b) case 2, and (c) case 3.
dupl.: duplication

absent ancestor-descendant relationships that exist in r are processed segment-wise (lines 3 – 4). All necessary unfolding absent ancestor-descendant relationships are found because we are looking for them in the lineage relationship matrix Z of r which stays unchanged during the algorithm. For each occurrence of necessary unfolding absent ancestor-descendant relationships between three lineages k, k' and k'', a new stack R_{new} is created (line 5), which will receive all forked lineage relationship matrices with their corresponding subclonal reconstructions. Each subclonal reconstruction r' in the stack R is processed and its lineage relationship matrix is forked with all valid relationship settings (lines 6 – 15). Each update with a relationship setting f is done on a duplicate r'' of r' with lineage relationship matrix Z''. The three entries $Z''_{k,k'}$, $Z''_{k,k''}$ and $Z''_{k',k''}$ are updated to the values $f[0]$, $f[1]$ and $f[2]$ with Algorithm 2. Since the lineage relationship matrices are forked iteratively, it is possible that different forking scenar-

Algorithm 7 Transforming necessary unfolding absent ancestor-descendant relationships

Input: subclonal reconstruction r with lineage number K, lineage relationship matrix Z, CNA assignment with phasing, and SSM assignment with phasing
Output: stack R of subclonal reconstructions without necessary unfolding absent ancestor-descendant relationships

```
1: create empty stack R
2: R.push(duplicate of r)
3: for each segment i with at least two copy number changes on different alleles in different lineages do
4:   for all lineages k, k' and k'', with 0 < k < k' < k'' < K, with necessary unfolding absent ancestor-descendant relationships in segment i according to Z ▷ see Observation 2 do
5:     create empty stack R_new
6:     while R is not empty do
7:       r' ← R.pop()
8:       F ← all relationship settings for lineages k, k' and k'' according to Z
9:       for each relationship setting f ∈ F do
10:        r'' ← duplicate of r'
11:        Z'' ← lineage relationship matrix of r''
12:        try
13:          update Z'' and SSM phasing for r'', k, k', and f[0] ▷ see Algorithm 2
14:          update Z'' and SSM phasing for r'', k, k'', and f[1] ▷ see Algorithm 2
15:          update Z'' and SSM phasing for r'', k', k'', and f[2] ▷ see Algorithm 2
16:          if r'' is not contained in R_new
17:            R_new.push(r'')
18:        catch update not possible
19:          pass
20:    R ← R_new
21: return R
```

ios of different lineage relationship matrices lead to the same subclonal reconstructions. Thus, if updating Z'' was successful, the updated subclonal reconstruction r'' is added to the stack R_{new} only if it is not contained already (lines 16 – 17). If updating Z'' was not possible, r'' is not used further (lines 18 – 19). Reasons why updating the lineage relationship matrix with a relationship setting f can fail, are explained below. After processing all subclonal reconstructions in the stack R, R_{new} becomes R for the next round (line 20) and further forking is performed on the updated subclonal reconstructions in R. At the end, the stack R is returned (line 21), which now contains all possible

subclonal reconstructions without necessary unfolding absent ancestor-descendant relationships.

Updating the lineage relationship matrix Z'' with a relationship setting fails if an entry $Z''_{k^\circ,k^\bullet}$ that should be updated to $v \neq ?$ is not ambiguous anymore and unequal v, or if SSMs phased to alleles A or B of lineages k° or $k^\bullet$ in r'' do not allow to update the relationship to a present one (see Algorithm 2). Although necessary unfolding absent ancestor-descendant relationships arise between three lineages with ambiguous ancestor-descendant relationships and unphased SSMs, it is possible that two relationship settings are not compatible with each other. This is because applying Algorithm 7 combines multiple relationship settings, which update relationships and SSM phasing.

Definition 13 (Relationship Setting Compatibility)**.** *Two relationship settings f_1 and f_2 are* compatible *if one of the following is true:*

1. *They are equal.*
2. *The lineages involved in relationship updates because of f_1 are different from the ones because of f_2.*
3. *Lineage relationships updated by both f_1 and f_2 are either equal or ambiguous in either f_1 or f_2, and all SSMs that are phased in the same segments because of f_1 and f_2 are phased to the same alleles.*

From Definition 13 we directly see that if two relationship settings are compatible, the order in which they are applied does not change the result. Also, if two relationship settings are not compatible, they will not become compatible if their order is reverted. Thus, the order in which the relationship matrices are forked in Algorithm 7, does not change the subclonal reconstructions of the final result.

If an incompatible relationship setting is applied, the forked subclonal reconstruction will not be used further. One subclonal reconstruction that is always a valid result is the one in which all necessary unfolding absent ancestor-descendant relationships are updated to absent relationships. Hence, the stack of final subclonal reconstructions is never empty.

4.2.5. Identifying Present Ancestor-Descendant Relationships Necessary because of Sum Rule

After step 5 of the ambiguity algorithm, we have a stack of subclonal reconstructions. Since they were created from a single subclonal reconstruction r, all have the same

lineage frequencies, CNA assignment and phasing, and SSM assignment. As their lineage relationship matrices were forked from the original matrix Z of r with Algorithm 7, the lineage relationship matrices as well as the SSM phasing differ. However, all lineage relationship matrices have in common that they contain all necessary absent and present ancestor-descendant relationships, except the ones necessary because of the sum rule.

Algorithm 8 processes each subclonal reconstruction of the stack R and transforms ambiguous ancestor-descendant relationships that need to be present because of the sum rule into present ones. The algorithm works in a top-down approach and iteratively checks for any lineage k, with $0 \leq k < K-2$, whether the sum rule holds (line 1). Lineages $K-1$ and $K-2$ do not have to be checked as they have at most one child and the sum rule can be violated only if a lineage has at least two children. For each processed lineage k, a new stack R_{new} is created that will receive the subclonal reconstructions that fulfill the sum rule for lineage k (line 2). Now each subclonal reconstruction r with lineage relationship matrix Z and children set χ_k is removed from R and processed (lines 3 – 6). If lineage k has at most one child, the sum rule is fulfilled. Since updating ambiguous ancestor-descendant relationships of different subclonal reconstructions can lead to equal subclonal reconstructions, we add r to the stack R_{new} only if it is not already contained in it (lines 7 – 9). If r has at least two children and the sum rule is fulfilled for each sample, r is added to R_{new} if it is not already contained in it (lines 10 – 12). Otherwise, the sum rule is tried to be fulfilled by removing children of lineage k and making them its grandchildren. Therefore, all possible pairwise children combinations are received (line 14). Each combination consists of two lineage indices k' and k'', with lineages k' and k'' both being children of lineage k and being in an ambiguous ancestor-descendant relationship. If no such combination exists, the current subclonal reconstruction r is not able to fulfill the sum rule. If combinations exist, the subclonal reconstruction r is duplicated to r' with lineage relationship matrix Z' for each combination l (lines 15 – 18), so that following updates do not change r. Now lineage k'' is made a child of lineage k' by updating Z' with Algorithm 2 (lines 19 – 21). Note that this update always works since absent ancestor-descendant relationships for the original lineage relationship matrix Z that are necessary because of the crossing rule or mutation assignment are already found. Afterwards, it is checked whether the sum rule is fulfilled for lineage k that is not a parent of lineage k'' anymore. If this is the case and the new subclonal reconstruction r' is not yet in the stack R_{new}, it is added to it (lines 22 – 24). Otherwise, lineage k has to be processed again, which is achieved by adding r' to the stack R if it is not already contained in it (lines 25 – 26).

Algorithm 8 Finding ancestor-descendant relationships that need to be present because of the sum rule

Input: stack R with subclonal reconstructions with lineage number K, lineage relationship matrix, CNA assignment with phasing, SSM assignment with phasing, and lineage frequencies ϕ
sample number N
Output: stack R with subclonal reconstructions that fulfill sum rule

for $k \leftarrow 0, \ldots, K-3$ **do**
  create empty stack R_{new}
  while R is not empty **do**
    $r \leftarrow R$.pop()
    $Z \leftarrow$ lineage relationship matrix of r
    $\chi_k \leftarrow$ indices of children of lineage k according to Z ▷ *see Algorithm 9*
    if $|\chi_k| \leq 1$ **then**
      if r is not contained in R_{new} **then**
        R_{new}.push(r)
    else if $\phi_{k,n} \geq \sum_{k' \in \chi_k} \phi_{k',n}$ **for each** $n \in \{0, \ldots, N-1\}$ **then**
      if r is not contained in R_{new} **then**
        R_{new}.push(r)
    else
      $L \leftarrow$ all possible combinations of χ_k according to Z ▷ *see Algorithm 10*
      if L is not empty **then**
        for each l in L **do**
          $r' \leftarrow$ duplicate of r
          $Z' \leftarrow$ lineage relationship matrix of r'
          $k' \leftarrow l[0]$
          $k'' \leftarrow l[1]$
          update Z' and SSM phasing for r', k', k'', and 1 ▷ *see Algorithm 2*
          if $\phi_{k,n} \geq \sum_{k' \in \chi_k} \phi_{k',n} - \phi_{k'',n}$ **for each** $n \in \{0, \ldots, N-1\}$ **then**
            if r' is not contained in R_{new} **then**
              R_{new}.push(r')
          else if r' is not contained in R **then**
            R.push(r')
  $R \leftarrow R_{new}$
return R

Algorithm 9 Getting all children

Input: lineage relationship matrix Z,
lineage index k,
number of lineages K
Output: stack χ which contains indices of all children of lineage k

create empty stack χ
for $k'' \leftarrow k+1, \ldots, K-1$ **do**
  if $Z_{k,k''} = 1$ **then**
    $parent =$ **true**
    for $k' \leftarrow k+1, \ldots, k''-1$ **do**
      if $Z_{k',k''} = 1$ **then**
        $parent =$ **false**
        break
    if $parent =$ **true then**
      χ.push(k'')
return χ

Algorithm 10 Getting all possible children combinations

Input: lineage relationship matrix Z,
stack χ_k which contains ordered indices of all children of a lineage k,
number of children n_χ
Output: stack L which contains all possible pairwise children combinations

create empty stack L
for $c \leftarrow 0, \ldots, n_\chi - 2$ **do**
  $k' \leftarrow \chi_k[c]$
  for $c' \leftarrow c+1, \ldots, n_\chi - 1$ **do**
    $k'' \leftarrow \chi_k[c']$
    if $Z_{k',k''} =$? **then**
      L.push((k', k''))
return L

After all subclonal reconstructions of stack R are processed, R receives the subclonal reconstructions of R_{new}, which are now going to be checked for lineage $k+1$ (line 27). After all lineages for all subclonal reconstructions are checked, the final stack, which contains all present ancestor-descendant relationships that are necessary because of the sum rule, is returned (line 28).

We chose a top-down approach for Algorithm 8 because in order to fulfill a violated sum rule for lineage k, we have to make some of its children its grandchildren. Thus, we move some lineages to its descendants with indices higher than k. Following the top-down approach, these lineages will be checked later to fulfill the sum rule. Hence, if assigning more children to them leads to a violation of the sum rule, the violation is either solved by making some children to grandchildren again, or if solving the violation is not possible, by not working with the subclonal reconstruction any longer.

Note that Algorithm 8 is greedy in a way that it may create too many subclonal reconstructions. It is possible that the lineage relationship matrix Z of one subclonal reconstruction r can be transformed to the lineage relationship matrix Z' of another subclonal reconstruction r' by updating ambiguous relationships. Thus, Z' actually contains present or absent ancestor-descendant relationships that are not necessary. Developing a bottom-up algorithm that removes subclonal reconstructions as r' is part of future work.

4.2.6. Identifying Absent Ancestor-Descendant Relationships Necessary because of Sum Rule

After step 6 of the ambiguity algorithm, a subclonal reconstruction r could still contain ambiguous ancestor-descendant relationships that need to be absent because of the sum rule. These necessary absent relationships are found with Algorithm 11. For each subclonal reconstruction r in the stack with subclonal reconstructions, each ambiguous relationship $Z_{k,k'}$, with $0 \leq k < k' < K-1$ is checked (lines 1 – 5). If the sum rule was violated if lineage k was a parent of lineage k' (lines 6 – 8), it is checked whether a subclonal reconstruction exists in which lineage k is an ancestor of lineage k' and the sum rule is fulfilled (lines 9 – 15). This is done by duplicating the subclonal reconstruction r to r' and making lineage k a parent of lineage k'. Afterwards, Algorithm 8, which finds ancestor-descendant relationships that need to be present because of the sum rule, is applied onto r', started by trying to solve the sum rule conflict for lineage k. If such a valid subclonal reconstruction can be found, the relationship between lineages k and k' stays ambiguous. Otherwise, it is transformed into an absent relation-

ship and the lineage relationship matrix Z is updated (lines 16 – 19). Each ambiguous ancestor-descendant relationship in the subclonal reconstruction of the returned stack (line 20) is truly ambiguous and can be updated to present or absent relationships without changing the likelihood or leading to an invalid subclonal reconstruction.

Algorithm 11 Finding absent ancestor-descendant relationships necessary because of sum rule

Input: stack R with subclonal reconstructions with lineage number K, lineage relationship matrix, CNA assignment with phasing, SSM assignment with phasing, and lineage frequencies ϕ
sample number N

Output: stack R with subclonal reconstructions whose ambiguous ancestor-descendant relationships are truly ambiguous

1: **for each** $r \in R$ **do**
2: $Z \leftarrow$ lineage relationship matrix of r
3: **for** $k \leftarrow 0, \ldots, K-3$ **do**
4: **for** $k' \leftarrow k+1, \ldots, K-2$ **do**
5: **if** $Z_{k,k'} = ?$ **then**
6: $\mathcal{D}_k \leftarrow$ indices of descendants of lineage k according to Z ▷ *see Algorithm 5*
7: $\chi_k \leftarrow$ indices of children of lineage k according to Z ▷ *see Algorithm 9*
8: **if** $\phi_{k,n} < \sum_{k^\circ \in \chi_k} \phi_{k^\circ,n} + \phi_{k',n}$ **for any** $n \in \{0, \ldots, N-1\}$ **then**
9: $r' \leftarrow$ duplicate of r
10: $Z' \leftarrow$lineage relationship matrix of r'
11: $Z'_{k,k'} = 1$
12: update Z' and SSM phasing for r', k, k', and 1 ▷ *see Algorithm 2*
13: create empty stack R'
14: R'.push(r')
15: find all ancestor-descendant relationships for r' in R', starting with k, that need to be present because of the sum rule ▷ *compare Algorithm 8*
16: **if** R' is empty **then**
17: $Z_{k,k'} = 0$
18: **for each** entry $Z_{i,i'}$, with either i or $i' \in \{k, k'\}$, that should get updated to $v' = 0$ because of $Z_{k,k'} = 0$ ▷ *see Table 4.1* **do**
19: update Z and SSM phasing for r, i, i', and v' ▷ *see Algorithm 2*
20: **return** R

Note that given the number of ambiguous ancestor-descendant relationships n_{amb} of a subclonal reconstruction r, $2^{n_{amb}}$ is the upper bound of valid lineage relationship matrices for r and not necessarily the actual number. The reason is that if an ambiguous relationship is transformed into a present or absent one, other ambiguous relationships can get updated as well.

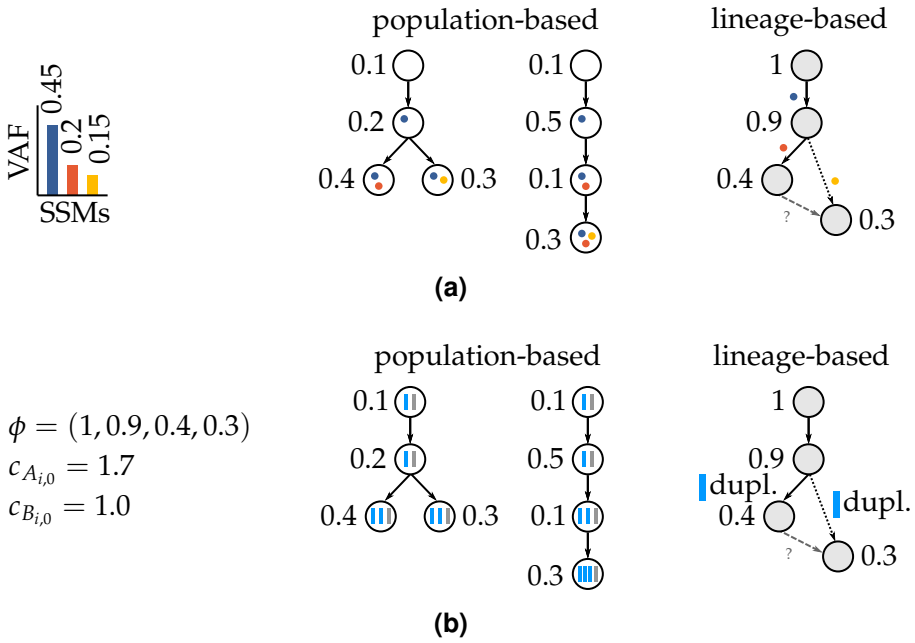


Figure 4.8.: Population-based and lineage-based subclonal reconstructions based on (a) SSM and (b) copy number information.
White circles with numbers show populations with population frequencies, gray circles with numbers show lineages with lineage frequencies. Colored dots indicate different SSMs, assigned to populations or lineages. Blue and gray bars represent alleles A and B, which can be duplicated.
VAF: variant allele frequency, SSMs: simple somatic mutations, dupl.: duplication

4.3. Lineage-Based versus Population-Based Subclonal Reconstruction

Since we model CNAs as copy number changes per lineage, we can combine CNAs and SSMs in a lineage-based model which allows us to handle ambiguity better than population-based models. The reason is that population-based models have to infer the relationships between all populations to model SSMs and CNAs, while we can work with ambiguous lineage relationships. Hence, we can model ambiguity caused by lineage relationships in a single subclonal reconstruction and its forked lineage relationship matrices.

An example how ambiguous lineage relationships can be represented in a single lineage-based subclonal reconstruction is shown in Figure 4.8a. Given three SSMs with their measured VAFs, two population-based subclonal reconstructions with different frequencies and relationships exist that explain the VAFs equally well. It is

not possible to decide which of the two subclonal reconstructions is better. However, using a lineage-based subclonal reconstruction with ambiguous relationships allows to explain the VAFs in a single subclonal reconstruction without having to decide for all relationships whether they are present or absent. A similar example for CNAs is shown in Figure 4.8b. Here, average allele-specific copy numbers of a segment i and lineage frequencies ϕ are given. Two population-based subclonal reconstructions with different CNAs explain the input data equally well, as does also a single lineage-based subclonal reconstruction.

Thus, lineage-based subclonal reconstructions with ambiguous lineage relationships allow us to model uncertainty within a single subclonal reconstruction and its forked lineage relationship matrices.

Chapter 5

Analyzing Onctopus' Performance

In this chapter, we describe how we implemented our subclonal reconstruction method in the software Onctopus, analyze its performance and investigate how the performance can be improved.

In Section 5.1, we briefly explain the implementation. Afterwards, in Section 5.2, we describe our data simulation with which we created all simulated datasets in this thesis. In Sections 5.3, we shortly explain different metrics used in the chapter, before analyzing the optimality, run time and memory usage of Onctopus in Section 5.4. In Section 5.5, 5.6 and 5.7, we show how Onctopus' performance can be improved by clustering SSMs, fixing CNAs and fixing lineage frequencies. Finally, in Section 5.8, we investigate how approximating the VAFs in the MILP influences the performance of Onctopus.

5.1. Implementation

Onctopus is implemented in Python 2[1]. The MILP is solved with CPLEX 12.6.1 [18], which is used through its Python API.

Currently, Onctopus takes average allele-specific copy numbers of I genome segments and the variant and reference read counts of J SSMs of a single tumor sample as input. Future work will include extending the implementation to work with copy number and SSM information of multiple tumor samples of the same patient.

In its standard setting, Onctopus optimizes the variables of a subclonal reconstruction, which are the lineage frequencies, the lineage relationships, the CNA assignments

[1] https://www.python.org/

including phasing, and the SSM assignments with phasing and with copy number influence in the same lineage. It is also possible to fix the values of some of these variables for the optimization. Another possibility is to use specific values as start values of the optimization.

Input parameters of Onctopus, together with a short explanation and default values, are shown in Table A.1. The source code of Onctopus is available at https://github.com/ratschlab/onctopus.

5.2. Data Simulation

In this section, we describe our approach to simulate input data for Onctopus, namely average allele-specific copy numbers of I genome segments and the variant as well as the reference read counts of J SSMs of one heterogeneous tumor sample.

The lineage frequencies ϕ as well as the ancestor-descendant relationships between K lineages are given as input to the simulation. Then, CNAs and SSMs are assigned to the lineages. Afterwards, we compute the length of all genome segments, their copy number standard errors and the copy numbers themselves, the read coverage of all segments and the variant and reference read counts of all SSMs.

CNA Assignment. Each CNA is assigned to one allele of a segment of a cancerous lineage, where each allele contains at most one CNA.

The CNA assignment can either be given as input to the simulation or it can be sampled. If the assignment should get sampled, the number of all segments I and the number of segments with CNAs I'' need to be given as input. CNAs are assigned to the first I'' segments, the cancerous lineages to which CNAs are assigned are chosen following a uniform distribution. For each of the I'' segments, one of six copy number change possibilities is sampled:

1. copy number gain assignment to allele A,
2. copy number gain assignment to alleles A and B,
3. copy number loss assignment to allele B,
4. copy number loss assignment to alleles A and B,
5. LOH, copy number gain assignment to allele A and copy number loss assignment to allele B,

6. copy number gain or loss assignment to allele A or B of cancerous lineage k, same copy number change assignment to allele A or B of cancerous lineage k', $k \neq k'$, where gain or loss, as well as both alleles and lineage k' are sampled from a uniform distribution. Copy number change assignments to an allele that got already lost in an ancestral lineage are not possible.

The probabilities of the six copy number change possibilities can be specified.

We do not allow copy number change assignments in which both alleles of all segments of lineage 1 get lost since if lineage 1 is the clonal lineage, no alleles are left to assign SSMs to.

SSM Assignment. Each SSM is assigned to one allele of a segment of a cancerous lineage, where we do not allow the assignment to an allele that got already lost.

We assign SSMs following one of three strategies. First, the SSM assignment can be given as input to the simulation. Second, the same number of SSMs can be assigned to all alleles of all segments of all cancerous lineages where the alleles are not lost. Third, the SSM assignment can be sampled.

If the SSM assignment is sampled, the lineages of the assignments are chosen according to one of two possibilities. Either a cancerous lineage is sampled from a uniform distribution, or a lineage is chosen depending on the overall lineage frequencies. For the second possibility, we first construct an array $freq\_tab$ of length $K-1$ with the following properties:

$$freq\_tab[0] = \phi_{1,0}$$

and

$$freq\_tab[k] = freq\_tab[k-1] + \phi_{k,0} \text{ for } 1 < k < K.$$

Then, we uniformly draw a number x between 0 and $freq\_tab[K-2]$, and choose the lineage $k+1$ for which holds $freq\_tab[k-1] < x \leq freq\_tab[k]$ if $1 < k < K$ or $freq\_tab[k] \leq x$ if $k = 0$. The segment and allele to which the SSM is assigned are chosen using a uniform distribution.

Segment Length Computation. We compute the length of each segment i depending on the number of SSMs that are assigned to it:

$$|\text{segment } i| = \max(seg\_min\_length, \text{number of SSMs on segment } i \cdot 1000 + 1000),$$

where seg_min_length is the minimal segment length. Per default, it is $1,000,000$.

Copy Number Standard Error Computation. In order to compute the copy number standard error $\sigma_{\alpha_{i,0}}$ of allele α in segment i, we simulate heterogeneous SNP counts of a tumor and a matched-normal sample and calculate the resulting mean copy number $\bar{c}_{\alpha_{i,0}}$.

First, we compute the number of heterogeneous SNPs n_{SNPs_i} in segment i by assuming that seven heterogeneous SNPs appear within 10.000 bp[2] of a segment:

$$n_{SNPs_i} = \max\left(1, \text{round}\left(\frac{7}{10,000} \cdot |\text{segment } i|\right)\right).$$

Then, we draw n_{SNPs_i} read counts for the simulated tumor and the simulated matched-normal sample using a negative binomial distribution. To create the two parameters $f\_tumor_{\alpha_{i,0}}$ and $p\_tumor_{\alpha_{i,0}}$ of the negative binomial distribution, we need the coverage overdispersion parameter s_{COV}, and the allele-specific coverage of segment i $\text{coverage}_{\alpha_{i,0}} = \text{haploid coverage} \cdot c_{\alpha_{i,0}}$, where both s_{COV} and the haploid coverage are given as input to the simulation:

$$p\_tumor_{\alpha_{i,0}} = \frac{s_{\text{COV}}}{s_{\text{COV}} + \text{ coverage}_{\alpha_{i,0}}},$$

$$f\_tumor_{\alpha_{i,0}} = \frac{\text{coverage}_{\alpha_{i,0}} \cdot p\_tumor_{\alpha_{i,0}}}{1 - p\_tumor_{\alpha_{i,0}}}.$$

Now we can draw each tumor read count $rc\_tumor_{\alpha_{i,0,i'}}$, with $0 \leq i' < n_{SNPs_i}$, as:

$$rc\_tumor_{\alpha_{i,0,i'}} \sim \text{Negative Binomial}(f\_tumor_{\alpha_{i,0}}, p\_tumor_{\alpha_{i,0}}).$$

To draw read counts for the matched-normal sample, we create the two parameters $f\_normal_{\alpha_i}$ and $p\_normal_{\alpha_i}$ as

$$p\_normal_{\alpha_i} = \frac{s_{\text{COV}}}{s_{\text{COV}} + \text{ haploid coverage}}$$

and

$$f\_normal_{\alpha_i} = \frac{\text{haploid coverage} \cdot p\_normal_{\alpha_i}}{1 - p\_normal_{\alpha_i}}.$$

[2]This number is derived from the 1000 Genomes Project [1], stating that a typical genome contains 4.1 million to 5 million differences to the human reference genome, with more than 99.9% of them being SNPs.

Then, each read count $rc\_normal_{\alpha_{i,i'}}$ is drawn as

$$rc\_normal_{\alpha_{i,i'}} \sim \text{Negative Binomial}(f\_normal_{\alpha_i}, p\_normal_{\alpha_i}).$$

After drawing all read counts, we calculate the mean copy number $\bar{c}_{\alpha_{i,0}}$ as

$$\bar{c}_{\alpha_{i,0}} = \frac{\sum_{i'} \frac{rc\_tumor_{\alpha_{i,0,i'}}}{rc\_normal_{\alpha_{i,i'}}}}{n_{SNPs_i}}.$$

Finally, we calculate the standard error $\sigma_{\alpha_{i,0}}$ as

$$\sigma_{\alpha_{i,0}} = \sqrt{\frac{\sum_{i'} \left(\frac{rc\_tumor_{\alpha_{i,0,i'}}}{rc\_normal_{\alpha_{i,i'}}} \right)^2 - n_{SNPs_i} \cdot \bar{c}_{\alpha_{i,0}}^{\,2}}{n_{SNPs_i} \cdot (n_{SNPs_i} - 1)}}.$$

Copy Number Computation. For each allele α of each segment i, we compute the average allele-specific copy number $c_{\alpha_{i,0}}$ based on the assignment of CNAs:

$$c_{\alpha_{i,0}} = 1 + \sum_k \phi_{k,0} \cdot d_{\alpha_{i,k}},$$

where $d_{\alpha_{i,k}}$ is the copy number change of allele α in segment i of lineage k:

$$d_{\alpha_{i,k}} = \begin{cases} 1 & \text{if copy number gain is assigned,} \\ -1 & \text{if copy number loss is assigned,} \\ 0 & \text{if no CNA is assigned.} \end{cases}$$

If noise should be added to the true copy number $c_{\alpha_{i,0}}$, we draw the new allele-specific copy number $c'_{\alpha_{i,0}}$ from a normal distribution with mean $c_{\alpha_{i,0}}$ and standard deviation $\sigma_{\alpha_{i,0}}$. Since the copy number cannot be negative, we restrict $c'_{\alpha_{i,0}}$ to be at least 0:

$$c'_{\alpha_{i,0}} = \max(0, \mathcal{N}(c_{\alpha_{i,0}}, \sigma^2_{\alpha_{i,0}})).$$

Coverage and Read Count Computation. For each segment i, its read coverage is computed based on its copy number and the haploid coverage:

$$\text{coverage}_{i,0} = (c_{A_{i,0}} + c_{B_{i,0}}) \cdot \text{haploid coverage}.$$

To compute the variant and reference read counts of an SSM j, which is assigned to allele α of lineage k, we first need to compute its average copy number $s_{j,0}$. If a copy number gain is assigned to allele α in the segment i of the lineage k as well, j happens before the copy number change with a user defined probability that is given as input to the simulation. If j is sampled to happen before the copy number change, $f_{\alpha_{j,k}}$ is 1, otherwise 0. We then compute the average copy number $s_{j,0}$ as

$$s_{j,0} = \phi_k + f_{\alpha_{j,k}} \cdot \phi_k + \sum_{k' \in \mathcal{D}_k} d_{\alpha_{i,k}} \cdot \phi_{k'}.$$

Given the average copy number $s_{j,0}$ and the copy number $c_{i,0} = c_{A_{i,0}} + c_{B_{i,0}}$ of segment i, we compute the VAF $p_{j,0}$ as

$$p_{j,0} = \frac{s_{j,0}}{c_{i,0}}.$$

We draw the total read count $D_{j,0}$ from a negative binomial distribution:

$$D_{j,0} \sim \text{Negative Binomial}(f\_tumor_{i,0}, p\_tumor_{i,0}),$$

where

$$p\_tumor_{i,0} = \frac{s_{\text{COV}}}{s_{\text{COV}} + \text{ coverage}_{i,0}}$$

and

$$f\_tumor_{i,0} = \frac{\text{coverage}_{i,0} \cdot p\_tumor_{i,0}}{1 - p\_tumor_{i,0}}.$$

We use $D_{j,0}$ to draw the variant count $V_{j,0}$ from a beta-binomial distribution:

$$V_{j,0} \sim \text{Beta-Binomial}(D_{j,0}, \alpha_{j,0}, \beta_{j,0}),$$

with $\alpha_{j,0} = p_{j,0} \cdot s_{\text{SSM}}$ and $\beta_{j,0} = (1 - p_{j,0}) \cdot s_{\text{SSM}}$, where s_{SSM} is the beta-binomial overdispersion parameter which is provided as input. Finally, we compute the reference read count $R_{j,0}$ as

$$R_{j,0} = \max(0, D_{j,0} - V_{j,0}).$$

Important parameters of the simulation, their explanation and default values can be found in Table B.1.

5.3. Evaluation Metrics

In the following analyses, we use the SMC-Het 1C score and the area under the precision-recall curve (AUPRC) of SSM co-clustering to measure and compare performances of different subclonal reconstructions.

SMC-Het 1C Score. The SMC-Het 1C score was developed for the crowd-sourced benchmarking challenge *ICGC-TCGA DREAM Somatic Mutation Calling Tumour Heterogeneity* (SMC-Het) [79]. The motivation of this challenge is to provide benchmarking datasets and evaluation metrics for subclonal reconstruction methods. Thus, the challenge organizers simulated realistic tumor genomes and developed mathematically sound quantitative metrics that evaluate different aspects of subclonal reconstructions.

The SMC-Het 1C score evaluates the inference of lineage frequencies and the number of SSMs assigned to each lineage, while being independent of the number of lineages. The score ranges from 0 and 1, where 0 indicates a poor and 1 a perfect performance, in which the lineage frequencies are inferred correctly and the correct number of SSMs is assigned to each lineage.

Area Under the Precision-Recall Curve of SSM Co-Clustering. The co-clustering of SSMs indicates which SSMs are assigned to the same lineage [21]. The AUPRC of SSM co-clustering is a measure of how accurate the SSMs that belong to the same lineage in the ground truth dataset are assigned to the same lineage in the inferred subclonal reconstruction. It is also possible to compute the average SSM co-clustering of multiple sampled subclonal reconstructions and compare it against the ground truth co-clustering with the AUPRC.

5.4. Optimality, Run Time and Memory Usage

For some datasets, the branch-and-cut algorithm finds and proves the optimal subclonal reconstruction in reasonable time and space. For other datasets, this is not possible. The reason for this is that the number of binary variables of the MILP is too high because the datasets consist of too many genome segments or SSMs, or are reconstructed with too many lineages. Still, it is possible that a good solution can be found even when the optimality of this solution cannot be proved.

Table 5.1.: Parameters for simulating datasets of general experiment to analyze optimality, run time and memory usage of Onctopus runs.
Twenty different parameter sets are created by iterating over the different parameter values. At least two parameters have to have the values shown in bold.

parameter	parameter values							
lineage number	2	**3**	4	5	6	7		
SSM number	**50**	51	52	53	54	64	74	84
segment number	**1**	2	3	4	5	15	25	35

In this subsection, we analyze the optimality, the run time and the memory usage of Onctopus runs on different datasets. Also, we investigate the accuracy of found subclonal reconstructions.

In Subsection 5.4.1, we present a general experiment over a large number of datasets with different parameter settings. In Subsection 5.4.2, we analyze the effect of increasing the run time of the optimization. At the end, in Subsection 5.4.3, we summarize our findings.

5.4.1. General Experiment

We simulated 1000 different datasets to evaluate the optimality status, run time and memory usage of Onctopus runs. We used different numbers of lineages, SSMs and segments, where each segment contains one or two copy number changes in only one lineage. We created 20 different parameter combinations, starting with three lineages, 50 SSMs and one segment, and then changing only one of the parameters (see Table 5.1). Then, for each lineage number, we created five different phylogenetic trees with different lineage frequencies (see Figure B.1). For each parameter combination and each phylogenetic tree, we simulated ten datasets, resulting in 50 datasets per parameter combination and 1000 datasets in total. The simulation setup can be found in Section B.2.

We ran Onctopus on each of the simulated datasets with a single thread, a maximal run time of ten hours and without memory restrictions. As lineage numbers, we used only the ground truth numbers. The other parameters had default values.

Optimization Statuses. For 705 of the 1000 datasets, the optimal result could be proved. For the other 295 datasets, the time limit of ten hours was reached.

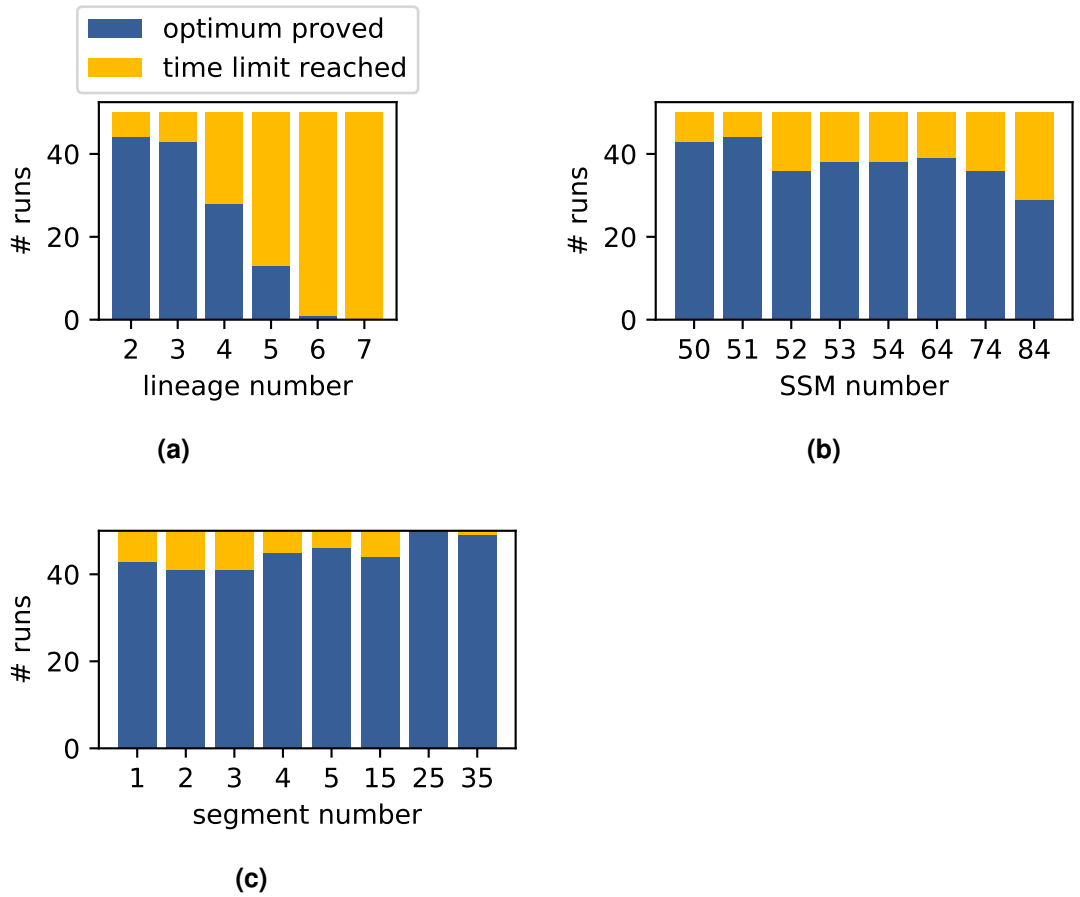


Figure 5.1.: Optimization statuses of Onctopus runs of general experiment to analyze optimality, run time and memory usage.
(a) Results for datasets with 50 SSMs, one segment and increasing lineage number. (b) Results for datasets with three lineages, one segment and increasing SSM number. (c) Results for datasets with three lineages, 50 SSMs and increasing segment number.

With increasing lineage number, more runs reach the run time limit before the optimal solution can be proved (see Figure 5.1a). For seven lineages, all 50 runs reach the run time limit. In general, for an increasing SSM number, the number of runs that reach the run time limit increases as well (see Figure 5.1b). However, the increase is not monotone. For 64 SSMs, for example, less runs reach the time limit than for 52, 53 and 54 SSMs. So although the MILPs for the datasets with 64 SSMs contain more variables, the branch-and-cut algorithm is able to prove the optimal solution in more cases. This

could be because a good solution is found earlier and the search tree does not have to be branched so often. Interestingly, for an increasing segment number from 1 to 35, the optimal solution can be proved for more runs (see Figure 5.1c). For 25 segments, the optimal solution can even be proved for all 50 runs. The reason could be that the copy number changes were simulated without noise and thus could compensate the noise in the VAFs of the SSMs, which could allow the branch-and-cut algorithm to find the optimal solution more quickly. To observe that an increasing number of SSMs and segments makes the finding and proving of an optimal solution more difficult, higher numbers with larger differences to the previous datasets have to be used.

Run Time Until Optimal Solution Is Proved. We investigate the run time behavior for Onctopus runs in which the optimality of the solution could be proved.

For increasing lineage numbers from two to five lineages, the run time increases faster than linear (see Figure 5.2a). For six lineages, the optimal solution could only be proved for one run and for seven lineages, no optimal solution could be proved. Thus, the run times on the datasets with six and seven lineages cannot be compared properly with the other ones. For an increasing SSM number, the run time increases (see Figure 5.2b) but not as quickly as for an increasing lineage number. With an increasing number of segments the overall run time increases as well, however the median run times show more fluctuation than for an increasing SSM number (see Figure 5.2c).

Memory Usage. The memory usage of the Onctopus runs increases with the run time of the optimization (see Figure 5.3). The highest variation for runs is present for runs for which the solution could not be proved to be optimal and which were terminated due to the run time restriction of ten hours. Here, the smallest memory usage is 0.25 GB and the largest 141 GB.

Accuracy of Found Subclonal Reconstructions. We measure the accuracy of the found subclonal reconstructions with the SMC-Het 1C score. Runs for which the optimal solution was proved have a higher SMC-Het 1C score than runs for which the optimality of the solution could not be proved (p-value$= 2.75 \cdot 10^{-30}$, Mann-Whitney U test; see Figure 5.4). Nearly all reconstructions that could be proved to be optimal reach a SMC-Het 1C higher than 0.9. Some reconstructions with proved optimality, however, have an SMC-Het 1C score lower than 0.5. The reason is that due to the ambiguity in the datasets, there might exist different subclonal reconstructions with

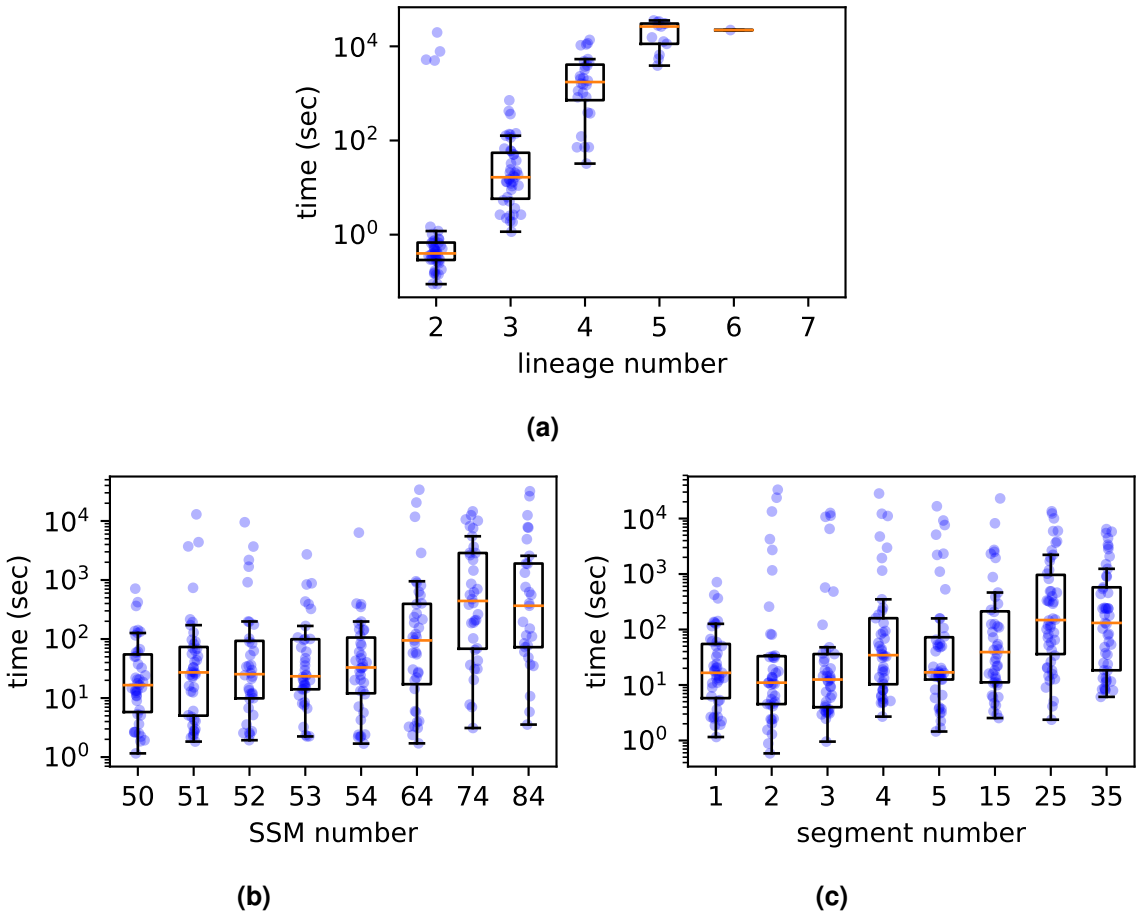


Figure 5.2.: Run time of Onctopus runs, for which optimal solution was proved, of general experiment to analyze optimality, run time and memory usage.
(a) Results for datasets with 50 SSMs, one segment and increasing lineage number.
(b) Results for datasets with three lineages, one segment and increasing SSM number.
(c) Results for datasets with three lineages, 50 SSMs and increasing segment number.

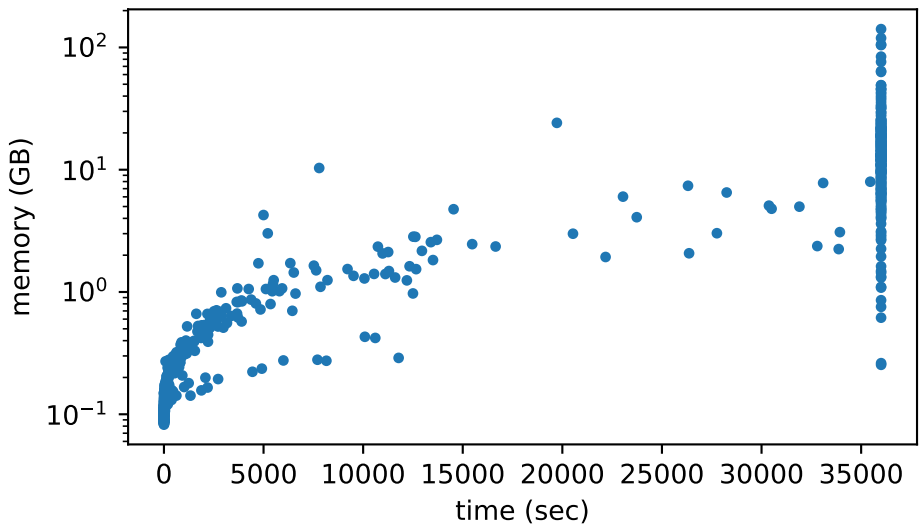


Figure 5.3.: Memory usage of Onctopus runs of general experiment to analyze optimality, run time and memory usage. Runs are sorted according to their run time.

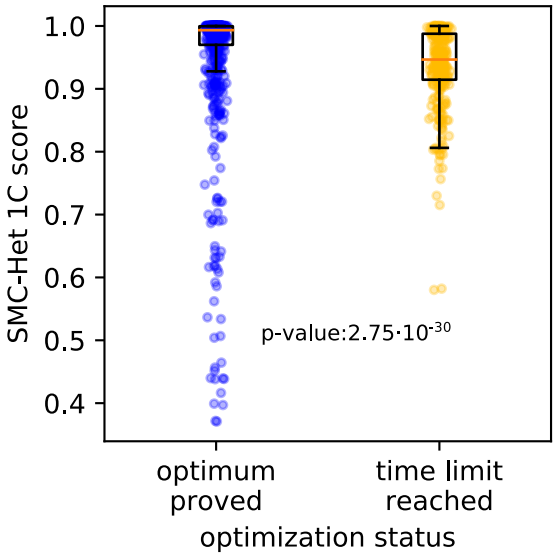


Figure 5.4.: SMC-Het 1C scores for different optimization statuses of Onctopus runs of general experiment to analyze optimality, run time and memory usage. The p-value was computed with a Mann-Whitney U test.

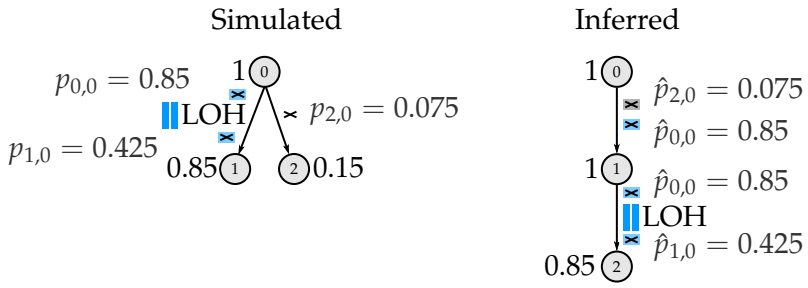


Figure 5.5.: Example of a simulated and its inferred subclonal reconstruction that are ambiguous.
The underlying dataset consists of three lineages, one segment and 84 SSMs, of which only a subset is shown. The VAFs presented in the figure are shown without noise to easily visualize the ambiguity in the subclonal reconstructions.
In the simulated subclonal reconstruction, an LOH event is assigned to lineage 1, so that allele A is duplicated and allele B is lost. Thus, SSMs arising on allele A before the duplication have a higher VAF as SSMs arising afterwards. SSMs assigned to lineage 2 are not directly affected by the copy number changes. In the subclonal reconstruction inferred by Onctopus, an LOH event is assigned to lineage 2. Because it has a similar frequency as lineage 1 in the simulated subclonal reconstruction, the average allele-specific copy numbers are similar as well. Since lineage 2 is a child of lineage 1, the LOH event influences the average copy numbers of the SSMs assigned to lineage 1 and similar VAFs as in the simulated subclonal reconstruction are inferred. SSMs with a VAF of 0.85 can be assigned to lineage 1 if they are phased to allele A or to lineage 2 if they are phased to allele A and arise before the copy number changes. The inferred subclonal reconstruction of Onctopus has a slightly higher log-likelihood of -235.65 compared to -237.71 of the simulated subclonal reconstruction. The achieved SMC-Het 1C score is only 0.37.
LOH: loss of heterozygosity

equal or higher likelihood than the simulated ground truth subclonal reconstruction. If Onctopus finds a subclonal reconstruction that differs strongly in lineage frequencies and SSM assignment, the SMC-Het 1C is low (see Figure 5.5 for an example).

5.4.2. Increasing Run Time

We now investigate the effect of the run time on the accuracy of subclonal reconstructions that could not be proved to be optimal. For this purpose, we simulated 50 datasets with four lineages, where each ten datasets were simulated based on the five phylogenetic trees also used for the previous experiment (see Figure B.1). Each dataset contains 35 segments, where each segment has one or two copy number changes in only one lineage, and 2940 SSMs across all segments. The simulation setup can be found in Section B.2.

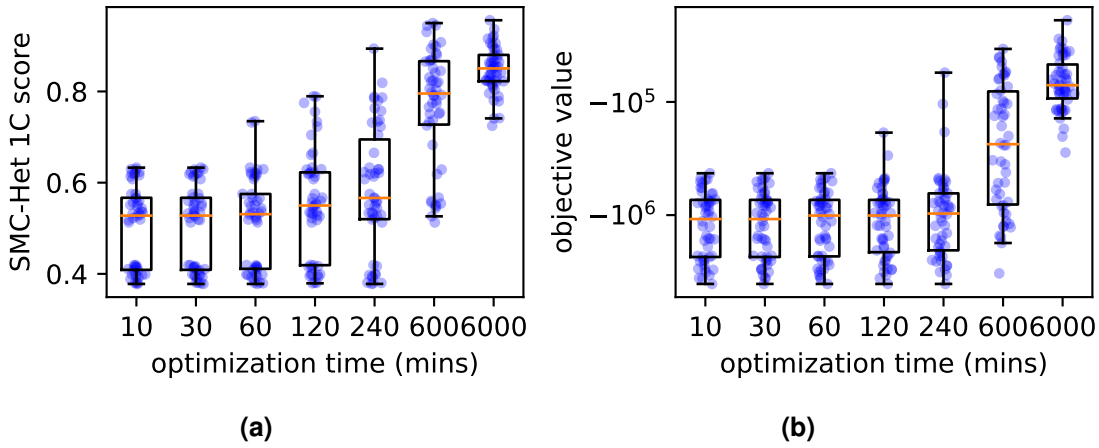


Figure 5.6.: (a) SMC-Het 1C scores and (b) objective values on 50 datasets for increasing optimization time.

We ran Onctopus on the simulated datasets with the following run time restrictions:

$$10, 30, 60, 120, 240, 600, 6000 \text{ minutes.}$$

All runs but the ones with 6000 minutes were run with a single thread, the ones with 6000 minutes were run with ten threads for ten hours. We used no memory restrictions and standard values for the other parameters. We set the number of lineages to be four.

Optimization Statuses. For none of the 50 Onctopus runs the optimal solution could be proved.

Improvements of Subclonal Reconstructions with Increasing Run Time. The SMC-Het 1C score of the 50 subclonal reconstructions improves with increasing run time (see Figure 5.6a). Also, the objective value of the subclonal reconstructions increases in the same pattern as the SMC-Het 1C score (see Figure 5.6b). Thus, we can conclude that our MILP models a subclonal reconstruction correctly and that the objective function works well but that the optimization itself is very hard. Hence, a good solution might not be found within a low run time. A higher run time can help to find a better subclonal reconstruction even though its optimality might still not be proved.

5.4.3. Conclusion

The run time to prove the optimality of a solution increases with the number of used lineages, the number of SSMs and in general also with the number of segments. The larger the dataset, the more time is needed to prove the optimality of the subclonal reconstruction. A high run time can help finding a better solution even though it is possible that its optimality still cannot be proved. The memory usage increases with the run time.

A trivial approach to improve the accuracy of a subclonal reconstruction is to use more run time for the optimization. However, a practical run time limit can be reached quickly. Thus, in the next subsections, we present other approaches to improve the solution that save run time and thus also memory usage of the optimization.

5.5. Clustering Simple Somatic Mutations

In this section, we investigate the effect on the accuracy of found subclonal reconstructions when built with clustered SSMs according to the weak parsimony assumption. First, in Subsection 5.5.1, we analyze the ability of different clustering algorithms and numbers of clusters to cluster SSMs based on their VAFs. Afterwards, in Subsection 5.5.2, we present the performance of Onctopus on simulated data when executed without SSM clustering and with clustered SSMs of version 1 and 2 (see Subsection 3.2.4 on page 39).

5.5.1. Clustering Algorithms and Cluster Numbers

To investigate the ability of different clustering algorithms and numbers of clusters to cluster SSMs based on their VAFs, we created 480 datasets. The datasets were simulated with one segment and with different numbers of lineages, copy number changes and SSMs. The SSMs were assigned following two different strategies (see Table 5.2). According to the first strategy, the same number of SSMs is assigned to each allele, which is not deleted because of a copy number loss, of each cancerous lineage. Hence, all cancerous lineages that do not contain copy number losses get the same number of SSMs assigned. Following the second strategy, the SSM assignment is sampled and we chose cancerous lineages depending on the overall lineage frequencies. Thus, different lineages do contain different numbers of SSMs. For each lineage number, we created one phylogenetic tree and for each copy number change number, we created a

Table 5.2.: Parameters for simulating datasets to analyze ability of different clustering algorithms and numbers of clusters to cluster SSMs based on their VAFs.

parameter	parameter values			
lineage number	4	6		
segment number	1			
copy number change number	0	1	2	3

SSM assignment strategy	SSM number		
same number per non-lost allele of each cancerous lineage	5	500	500
sampled assignment, average number per lineage	10	100	1000

specific copy number change assignment per tree (see Figure B.2). For each combination of lineage, copy number change and SSM number, and for each of the two SSM assignment strategies, we simulated ten datasets. The simulation setup can be found in Subsection B.3.1.

As clustering algorithms we used k-means [61] and agglomerative clustering, which are implemented in the python package *scikit learn*[3]. For agglomerative clustering, we use three different linkage options, defining which distance is minimized when merging clusters. The first option *ward* linkage minimizes the variance of each cluster. The second option *complete* linkage minimizes the largest distance between all items of two clusters. The third option *average* linkage minimizes the average distance between clusters. We call these three approaches agglomerative-ward, agglomerative-complete and agglomerative-average clustering from now on. We used different cluster numbers depending on the number of lineages K of a dataset:

$$n_c \cdot (K-1) \; \forall \; n_c \in \{1,2,3,4\}.$$

We then applied each clustering algorithm with each number of clusters on the VAFs of the SSMs of the 480 simulated datasets.

Using clustered SSMs for our optimization makes sense only if clusters contain mostly SSMs of the same lineage that are equally influenced by copy number changes. It is okay for us if SSMs of the same lineage with equal copy number change influence

[3]https://scikit-learn.org

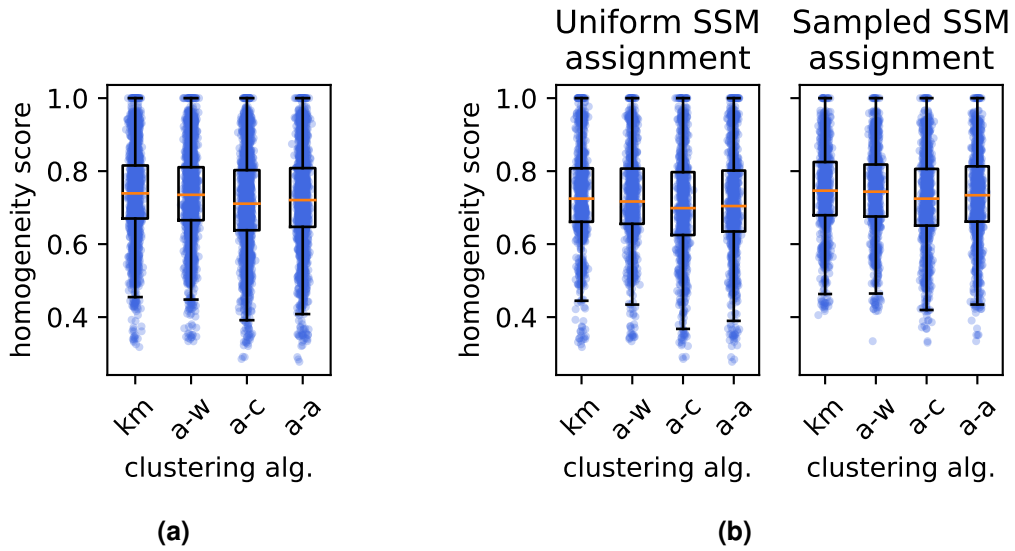


Figure 5.7.: Homogeneity score of the four clustering algorithms.
Results are shown for all cluster numbers for (a) all datasets and (b) datasets separated by SSM assignment strategy. 'Uniform SSMs' refers to first strategy, in which the same SSM number is assigned to all alleles of all cancerous lineages and 'uneven SSMs' refers to the second strategy, in which the SSM assignment is sampled.
alg.: algorithm, km: k-means, a-w: agglomerative-ward, a-c: agglomerative-complete, a-a: agglomerative-average

are distributed over several clusters, as long as each of these clusters is homogeneous. To evaluate this property, we use the homogeneity score [77].

Overall Performance. The four clustering algorithms achieve a similar performance with homogeneity scores ranging between 0.3 and 1 on the simulated datasets with different numbers of clusters (see Figure 5.7a). The k-means and the agglomerative-ward clustering algorithm achieve a higher median score than the agglomerative-complete and agglomerative-average clustering algorithms.

Performance Based on SSM Assignment Strategy. The homogeneity scores of the four clustering algorithms are higher on the datasets in which the SSM assignment was sampled than on the datasets in which the same number of SSMs was assigned to each allele of each cancerous lineage (see Figure 5.7b). For both SSM assignment strategies, the median homogeneity scores of the k-means and the agglomerative-ward clustering

algorithm are again slightly higher than the ones of the agglomerative-complete and agglomerative-average clustering algorithm.

Performance Based on Number of Clusters. Increasing the number of clusters increases the homogeneity scores for all of the four clustering algorithms (see Figure 5.8a). The largest improvement happens from $(K-1)$ clusters to $(K-1)\cdot 2$ clusters. For datasets without copy number changes, the homogeneity scores increase with increasing clustering number in the same way as for all datasets (see Figure 5.8b). This is interesting because without copy number changes, SSMs should easily cluster to $K-1$ clusters. However, due to noise in the VAFs, $K-1$ clusters are not enough to separate SSMs of different lineages. Thus, increasing the number of clusters improves the clustering performance.

A homogeneity score of 1 could be achieved for all datasets if we set the number of clusters to be equal to the number of SSMs. However, then we would not cluster the SSMs anymore.

Since the *k*-means and the agglomerative-ward algorithm showed the best performance, we chose them for further testing. The performance of the algorithms improves with increasing number of clusters. However, the higher the cluster number, the more complex does the optimization get again and we do not gain an advantage by clustering SSMs. Thus, we chose $2\cdot(K-1)$ and $3\cdot(K-1)$ clusters for further tests.

5.5.2. Building Subclonal Reconstructions with Clustered Simple Somatic Mutations

Now we present Onctopus' performance when run on simulated data without SSM clustering and with clustered SSMs of versions 1 and 2.

We created 120 datasets with four lineages, and different numbers of SSMs and segments, where half of the segments contain one or two copy number changes in one lineage (see Table 5.3). As in the previous experiment, we used two strategies to assign SSMs. Either the same number of SSMs is assigned to each allele, that is not lost, in each segment of each cancerous lineage, or the SSM assignment is sampled and SSMs are assigned to cancerous lineages according to the overall lineage frequencies. We created one phylogenetic tree (see Figure B.3) and simulated ten datasets for each combination of segment and SSM numbers and SSM assignment strategies (see Subsection B.3.2).

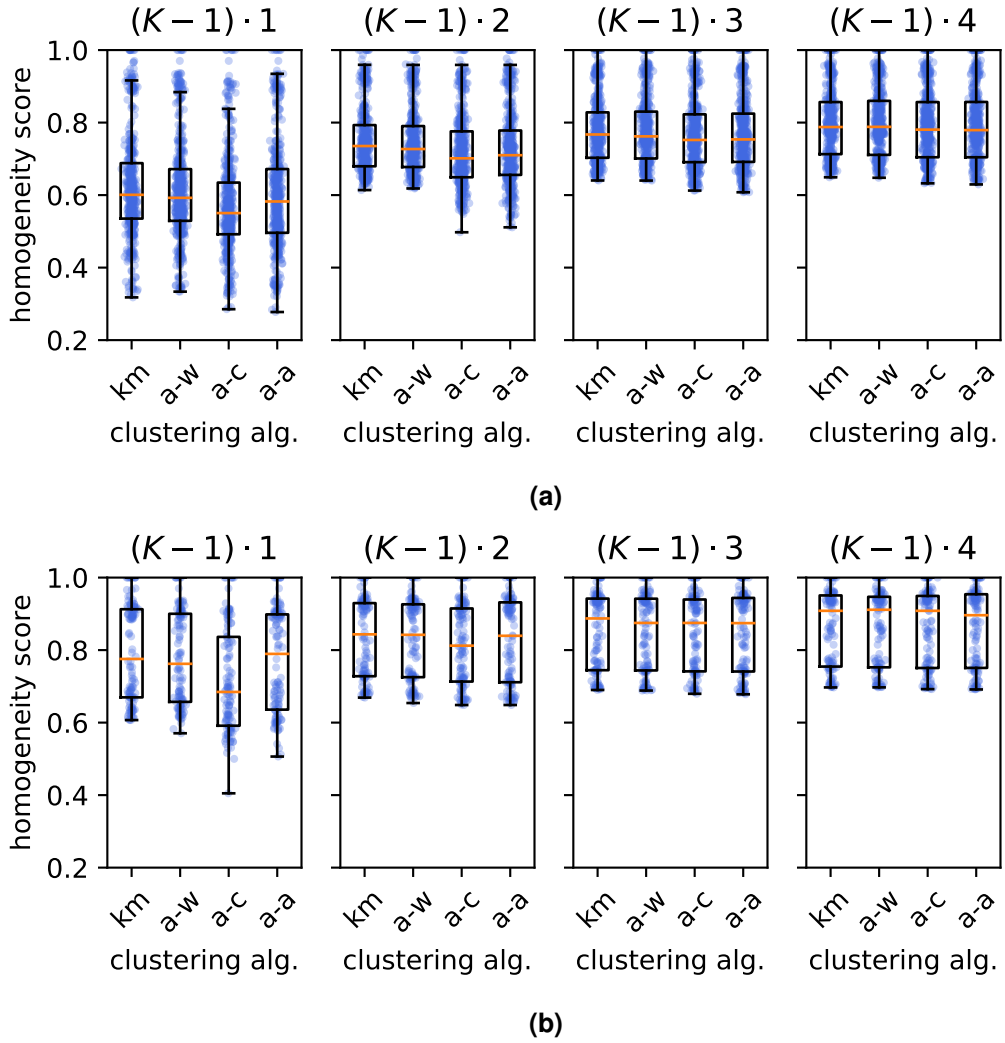


Figure 5.8.: Homogeneity score of the four clustering algorithms.
Results are shown for increasing number of clusters. (a) Results for all datasets are shown, separated by cluster number. (b) Only results of datasets without copy number changes are shown, separated by cluster number.
K: number of lineages, alg.: algorithm, km: k-means, a-w: agglomerative-ward, a-c: agglomerative-complete, a-a: agglomerative-average

Table 5.3.: Parameters for data simulation to analyze Onctopus' performance with and without clustered SSMs.

parameter	parameter values		
lineage number	4		
segment number I	2	10	
segments with copy number changes	$0.5 \cdot I$		
SSM assignment strategy	**SSM number**		
same number per non-lost allele of each segment of each cancerous lineage	5	500	5000
sampled assignment, average number per cancerous lineage per segment	10	100	1000

Table 5.4.: Clustering parameters of Onctopus.

parameter	parameter values	
clustering algorithm	k-means	agglomerative-ward
cluster number	$2 \cdot (K-1)$	$3 \cdot (K-1)$
using superSSMs	yes	no

K: number of lineages

We ran Onctopus on the datasets without SSM clustering and with eight combinations of different clustering algorithms, number of clusters and clustering versions (see Table 5.4). For all runs, we used a single thread, a maximal run time of two hours, no memory restrictions, four lineages and standard values for the other parameters.

Optimization Statuses. Without SSM clustering, for only 18 of the 120 runs the optimal solution could be proved (see Figure 5.9a). For 138 of the 480 runs that clustered the SSMs according to version 1, without using superSSMs, no solution could be found at all within the two hours of run time. This is caused by the additional clustering constraints, allowing only solutions in which the SSMs that belong to the same cluster are assigned to the same lineage and the same phase. When the run time restrictions are too tight, no solution can be found that fulfills these constraints. Of the runs in which a solution was found, optimality could be proved for 25 runs for each of the four clustering combinations. When clustering version 2, with superSSMs, was used, optimality could be proved for nearly all runs with $2 \cdot (K-1)$ clusters per segment and for more than half of the runs with $3 \cdot (K-1)$ clusters per segment. The reason for this is that less variables exist in the MILP, thus the optimization is less complex.

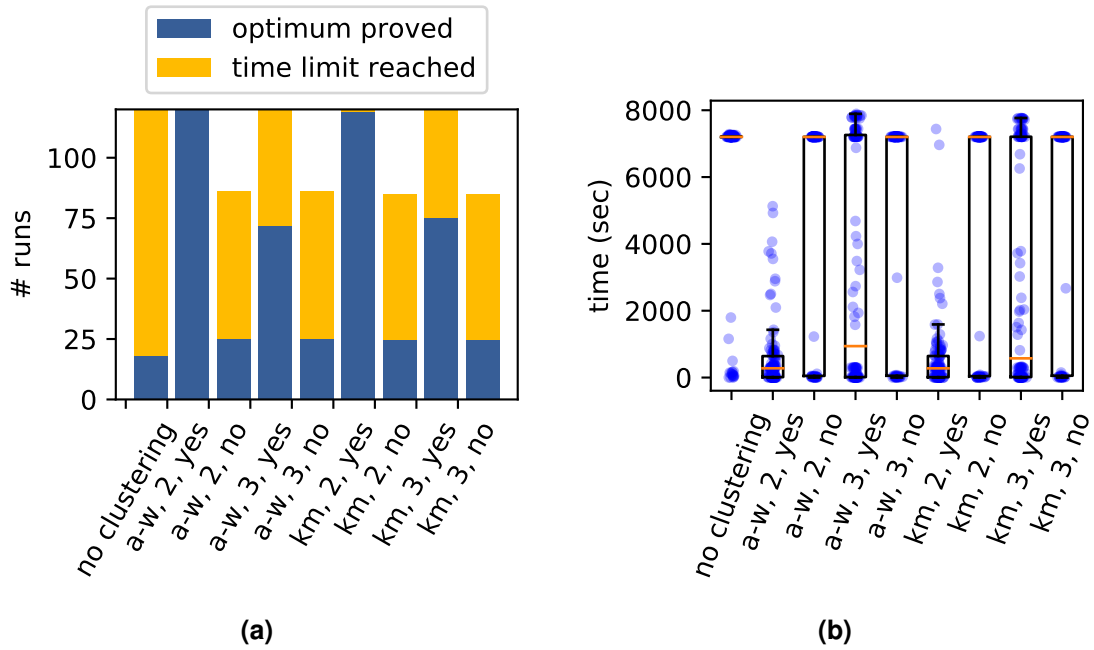


Figure 5.9.: (a) Optimization statuses and (b) run time of experiment to analyze Onctopus' performance with and without clustered SSMs.
For 138 runs clustering without superSSMs, no subclonal reconstruction could be built in the given time. No results are shown for these runs in the corresponding four columns of optimization statuses and run time.
clustering parameters: clustering algorithm, cluster number variable (with number of clusters = cluster number variable $\cdot(K-1)$), whether superSSMs were used
km: k-means, a-w: agglomerative-ward, K: number of lineages

Run Time. We executed Onctopus with a maximal optimization time of two hours. The complete run time of Onctopus, however, which includes creating variables and constraints for the second optimization when using superSSMs, can exceed this run time restriction.

The run time for runs without SSM clustering and with SSM clustering according to version 1 is very similar, with most of the runs needing the maximum optimization time of two hours (see Figure 5.9b). The run times of runs using superSSMs and $2 \cdot (K-1)$ clusters are the fastest, their medians are at around 270 sec. When superSSMs and $3 \cdot (K-1)$ clusters are used, the run times are higher but still significantly faster than without superSSMs.

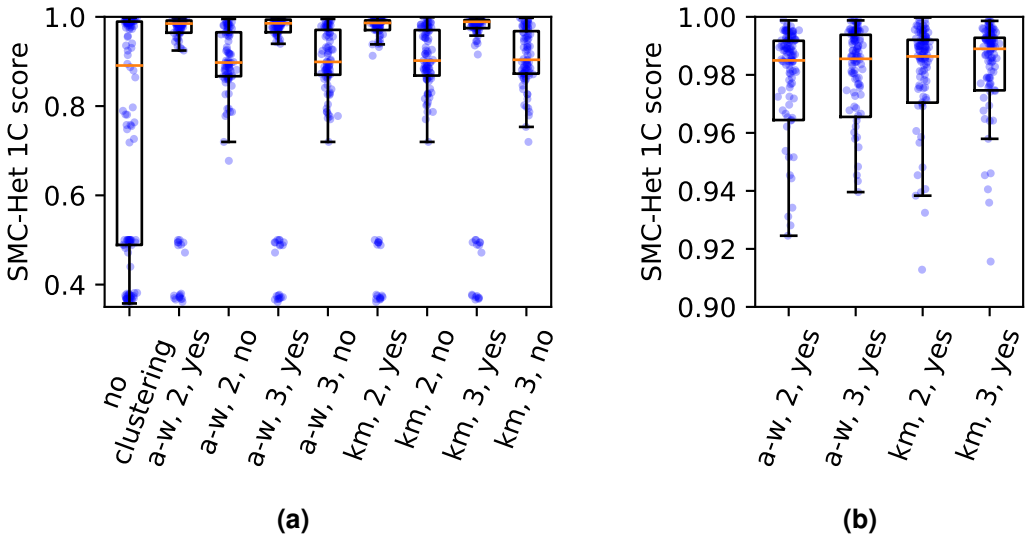


Figure 5.10.: SMC-Het 1C scores of experiment to analyze Onctopus' performance with and without clustered SSMs.
(a) All optimizations and all datasets. No results are shown for the optimizations with clustering without superSSMs on the datasets on which no solution could be found within time. Note that these are the datasets that led to the worst performance in the other optimizations. (b) Only optimizations with clustering using superSSMs, zoomed in view.
clustering parameters: clustering algorithm, cluster number variable (with number of clusters = cluster number variable $\cdot(K-1)$), whether superSSMs were used
a-w: agglomerative-ward, km: k-means, K: number of lineages

Comparison of Found Subclonal Reconstructions. We use the SMC-Het 1C score to compare the accuracy of found subclonal reconstructions of the nine different Onctopus clustering settings.

Interestingly, runs with SSM clustering of version 1 do not perform better in terms of SMC-Het 1C score than the runs without SSM clustering (see Figure 5.10a). Using superSSMs in the optimization leads to significantly better SMC-Het 1C scores.

To investigate which clustering setting of the Onctopus runs with superSSMs achieved the best performance, we took a closer look at the SMC-Het 1C scores higher than 0.9 (see Figure 5.10b). The scores of runs in which k-means clustering and $3 \cdot (K-1)$ clusters were used are the highest, while the scores of runs with agglomerative-ward clustering and $2 \cdot (K-1)$ are the lowest. Significant differences can be found only for these two settings (see Figure 5.11), not for the other two.

In this subsection, we compared the performance of Onctopus when run on the same datasets with different clustering settings. Using clustering without superSSMs

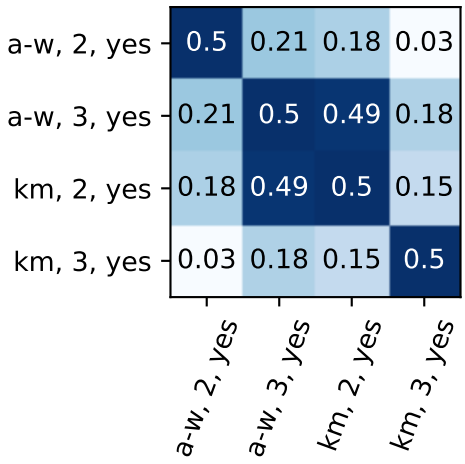


Figure 5.11.: P-values of Mann-Whitney-U test of SMC-Het 1C scores of runs with superSSMs.
clustering parameters: cluster algorithm, number of clusters $\cdot(K-1)$, whether superSSMs were used
a-w: agglomerative-ward, km: k-means, K: number of lineages

did not improve the performance compared to the setting without clustering. In some cases, this clustering option even prevented Onctopus to find any subclonal reconstruction within the given run time restriction. This is due to the additional constraints being used. When reducing the number of variables by using clustering with superSSMs, the performance increases significantly. We tested four different settings with superSSMs. Clustering with k-means and $3 \cdot (K-1)$ clusters led to the highest SMC-Het 1C scores, whereas these scores were only significantly different to one of the other three clustering settings.

We did not test the performance for more than $3 \cdot (K-1)$ clusters. It is possible that the performance for superSSMs would further increase. However, at some point the higher number of clusters leads to so many variables that the positive effect of needing less variables when using superSSMs is diminished and no better result can be found within a restricted run time compared to not clustering SSMs.

Here, we tested the performance of clustering with only k-means and agglomerative clustering. Other clustering algorithms, such as binomial mixture clustering, which is applied by Canopy, could be tested as well.

Table 5.5.: Parameters for data simulation to investigate effect of CNA fixation.

parameter	parameter values		
lineage number	4		
segment number I	10	50	100
segments with copy number changes	0	$0.5 \cdot I$	
SSM number	$40 \cdot I$		

5.6. Fixing Copy Number Aberrations

In this section, we investigate the effect on the optimization when segments without copy number changes are restricted to contain zero copy number changes (see Equation 3.52). For this purpose, we created 300 datasets. We simulated datasets with four lineages, different numbers of segments, different numbers of segments with copy number changes, and on average 40 SSMs per segment (see Table 5.5). We created five different phylogenetic trees with different lineage frequencies (see Figure B.4) and simulated ten different datasets for each tree, segment number and number of segments with copy number changes combination. Other parameters of the simulation can be found in Section B.4.

We ran Onctopus on the datasets with a maximal optimization time of four hours on a single thread, a memory restriction of 100 GB and four lineages. We used three different copy number fixation settings. First, we did not use any copy number fixation. Then, we restricted all CNA-free segments to contain zero copy number changes. Finally, as third setting, we combined all CNA-free segments to one *super segment* and restricted it to contain zero copy number changes. Other parameters used have default values.

Optimization Statuses. For none of the 300 datasets with the three different copy number fixation settings, the optimal solution can be proved. All runs are terminated due to the optimization time restriction.

Comparison of Found Subclonal Reconstructions. To compare the accuracy of found subclonal reconstructions of the three different copy number fixation settings, we use the SMC-Het 1C score. In general, we observe that optimizations in which the CNA-free segments are restricted to contain zero copy numbers perform better than optimizations in which no copy number change fixation is applied (see Figure 5.12). This effect is strongest for 50 segments, and for 100 segments when no copy number

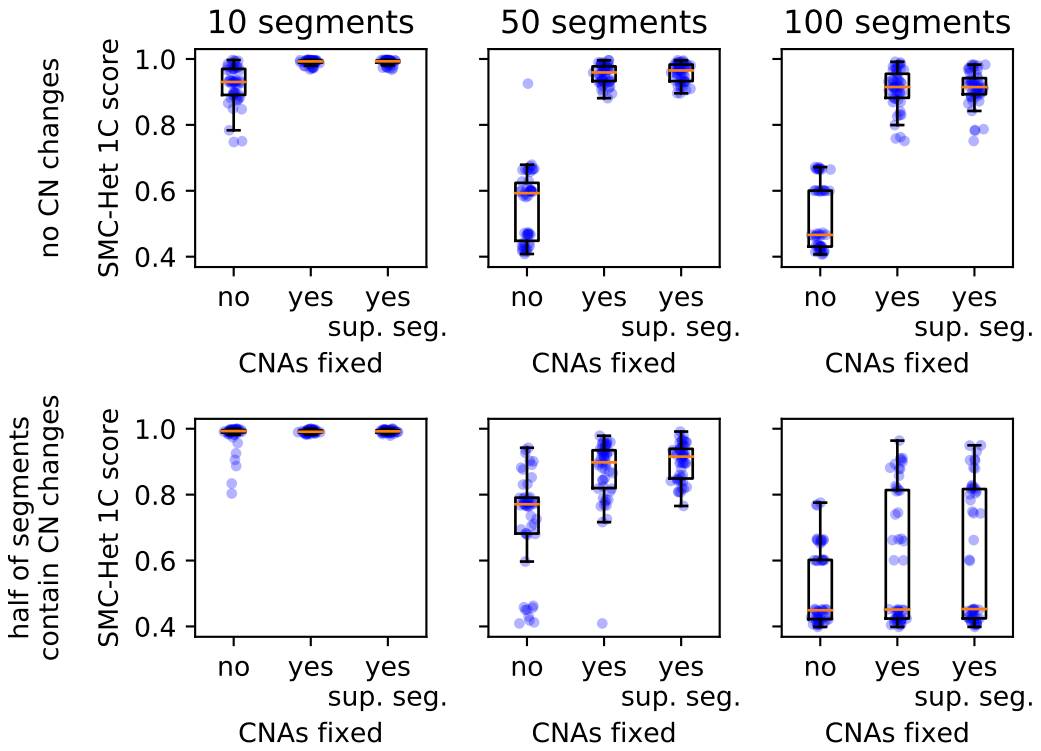


Figure 5.12.: SMC-Het 1C scores of experiment to analyze effect of CNA fixation.
Datasets are separated by number of segments and number of segments without CNAs.
CN: copy number, sup. seg.: super segment

changes exist. For ten segments and no copy number changes, the SMC-Het 1C scores of the Onctopus runs without fixation have a median score of 0.93, thus cannot be improved a lot. Interestingly, for ten segments and copy number changes in five segments, the SMC-Het 1C scores without fixation are nearly as good as the ones with fixation. For 100 segments and copy number changes in 50 segments, restricting CNA-free segments to contain zero copy number changes improves the scores of only a few runs. The optimization is so difficult that fixing CNAs in only half of the segments does not facilitate the problem in the given time.

Between the optimizations with individually fixed CNA-free segments and fixed *super segment*, no clear differences exist.

We have shown in this section that restricting CNA-free segments to contain zero copy number changes can improve the performance of Onctopus. The improvement

depends on the number of segments in total and on the number of segments that are not fixed. If the total number is low, using fixed copy number changes does not bring a high performance increase. If the number of segments without fixation is still high, the optimization can still be very hard. As restricting CNA-free segments to contain zero copy number changes does not worsen the performance, we recommend to always use this option. Furthermore, the information which segments are CNA-free is given by the copy number calling tools that provide the average allele-specific copy numbers, which are needed as input.

In this experiment, we did not observe an advantage of optimizing with a fixed *super segment*. This is because we did not use SSM clustering with superSSMs, which would allow to combine SSMs of different CNA-free segments to one superSSMs, and thus further reduce the number of MILP variables.

5.7. Fixing Lineage Frequencies

Different possible lineage frequencies lead to ambiguous subclonal reconstructions (see Section 4.1) which makes it harder for the optimization to find and prove an optimal solution. If the lineage frequencies were given, less ambiguous subclonal reconstructions existed, thus the optimization got easier.

In this section, we investigate the effect of fixed lineage frequencies on the performance of Onctopus. First, we analyze in Subsection 5.7.1 how the run time and performance improves if the correct lineages frequencies are given. Then, in Subsection 5.7.2, we evaluate how many SSMs are necessary to infer the lineage frequencies sufficiently. Finally, in Subsection 5.7.3, we study how run time and performance of Onctopus behave if lineage frequencies are fixed to inferred values.

5.7.1. Performance with Correct Lineage Frequencies

To investigate Onctopus' performance when the lineage frequencies are fixed to the correct values, we used the 250 simulated datasets of the general experiment to analyze optimality, run time and memory usage (see Subsection 5.4.1) with three to seven lineages, one segment and 50 SSMs.

We ran Onctopus with a run time restriction of one hour on a single thread with no memory restriction and correct lineage numbers. In one set of runs we fixed the lineage frequencies ϕ to the correct values, in the other set we did not fix lineage frequencies. The other parameters are set to standard values.

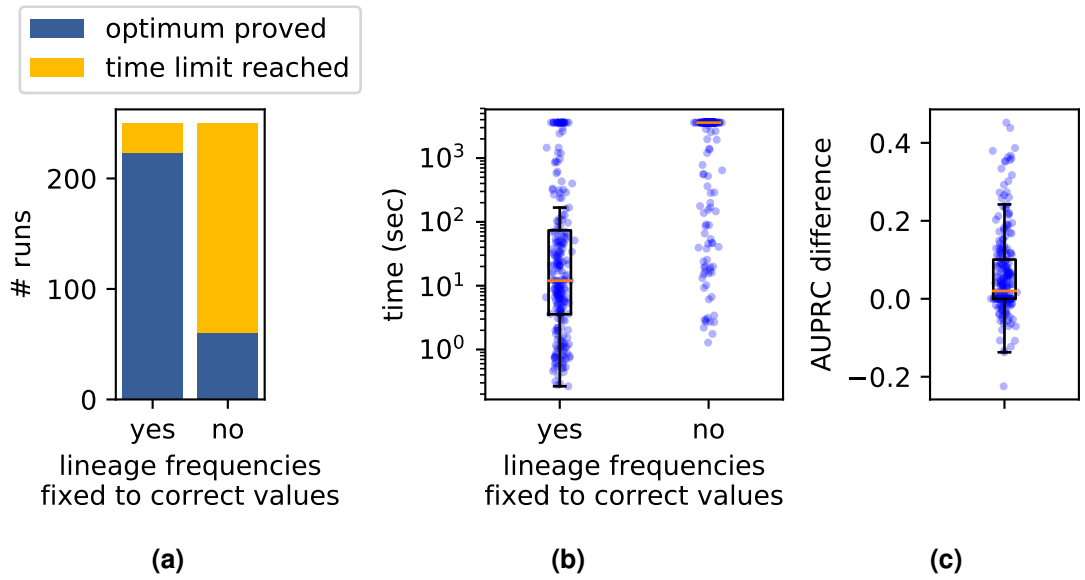


Figure 5.13.: (a) Optimization statuses, (b) optimization time and (c) difference in AUPRC of SSM co-clustering for first clustering experiment.
(a),(b) Values shown for optimizations where lineage frequencies were fixed to the correct values and without fixing the lineage frequencies. (c) AUPRC difference is calculated as taking the AUPRC of the SSM co-clustering of the optimizations where the lineage frequencies were fixed to the correct values and subtracting the AUPRC of the optimizations without lineage frequency fixation. When the difference is larger than 0, the AUPRC of the optimization with fixed lineage frequencies is better.

Optimization Statuses. When lineage frequencies were fixed to correct values, the optimal solution could be proved in four times more runs within the given time than when the lineage frequencies were not fixed (see Figure 5.13a).

Optimization Time. When the lineage frequencies were fixed to correct values, the optimization was done in less than 100 seconds for three quarters of the datasets (see Figure 5.13b). When the lineage frequencies were not fixed, the optimizations were terminated by the run time restriction of one hour for more than three quarters of the datasets.

Comparison of Found Subclonal Reconstructions. We compare accuracy of the subclonal reconstructions of the two lineage frequency fixation settings with the AUPRC of SSM co-clustering. When the lineage frequencies were fixed to correct values, more than half of the runs achieve a better AUPRC. For about a quarter of runs, fixing the lineage frequency to their correct values does not change the AUPRC.

Table 5.6.: Parameters for data simulation for second lineage frequency fixation experiment.

parameter	parameter values				
lineage number K	3	4	5	6	7
SSM number $\cdot(K-1)$	10	50	100	500	1000
segment number			1		

That about one quarter of runs reach a lower AUPRC when the lineage frequencies are fixed to the correct values can be explained through VAF noise and potentially different CNA inference.

We saw in this subsection that the optimization is faster and leads to better results when the lineage frequencies are fixed to the correct values.

In the next subsection, we will investigate how we can infer lineage frequencies which we can use as fixation values in the optimization.

5.7.2. Inference of Lineage Frequencies Depending on the Number of Simple Somatic Mutations

Ambiguous subclonal reconstructions can arise through the interplay of lineage frequencies and CNAs. In CNA-free regions, lineage frequency ambiguity plays a smaller role. Thus, our idea is to build subclonal reconstructions only with SSMs of CNA-free segments and use the inferred lineage frequencies as fixation values of a second optimization on the complete input data.

In this subsection, we investigate how many SSMs are needed to infer the lineage frequencies sufficiently.

We created 1250 datasets that consist of a single segment without copy number changes, different numbers of lineages and different numbers of SSMs per lineage (see Table 5.6). For each lineage number, we used five phylogenetic trees with different lineage frequencies and simulated ten datasets for each combination of tree, lineage and SSM number. The simulation setup can be found in Subsection B.5.1.

We ran Onctopus with an optimization time restriction of two hours on one thread without memory restriction and the correct lineage number K. The single segment was fixed to contain no copy number change, and SSMs were combined to superSSMs with k-means clustering and $3 \cdot (K-1)$ clusters. The other parameters of Onctopus have standard values.

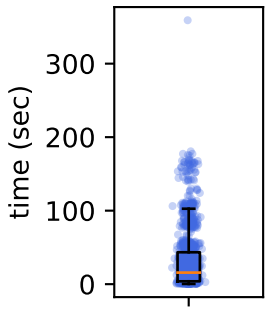


Figure 5.14.: Optimization time for all runs of second lineage frequency fixation experiment.

Optimization Statuses. For all runs the optimal solution could be proved in the given time.

Run Time. The optimal solution for three quarters of the runs is proved in less than 100 seconds (see Figure 5.14). All other runs except one are finished in less than 200 seconds.

Influence of Increasing SSM Number. We use the SMC-Het 1C score to investigate the influence of an increasing SSM number on the accuracy of lineage frequency inference. In general, we observe that the SMC-Het 1C score is higher than 0.96 for more than half of all datasets (see Figure 5.15). The lowest achieved SMC-Het 1C score is still higher than 0.93. For three lineages, the SMC-Het 1C scores improve with an increasing number of SSMs per lineage. For four and five lineages, the SMC-Het 1C scores increase from ten to 50 SSMs and then do not improve much. For six and seven lineages, the number of SSMs per lineage does not seem to have an influence on the SMC-Het 1C scores. This could be the case because with six or more lineages, the average frequency distance between lineages k and $k+1$ is smaller than 0.167. Thus, VAF clusters overlap on average more than for a smaller number of lineages, raising the chance that SSMs are assigned to the wrong lineages. Increasing the SSM number also increases the number of SSMs that are assigned to wrong lineages, which does not improve the lineage frequency inference.

Note that the SMC-Het 1C scores between datasets with the same lineage number but different SSM numbers cannot be compared directly because we simulated 50 datasets for each SSM number. To enable a direct comparison, we should have simulated 50 datasets for the highest number of SSMs and then create datasets with fewer

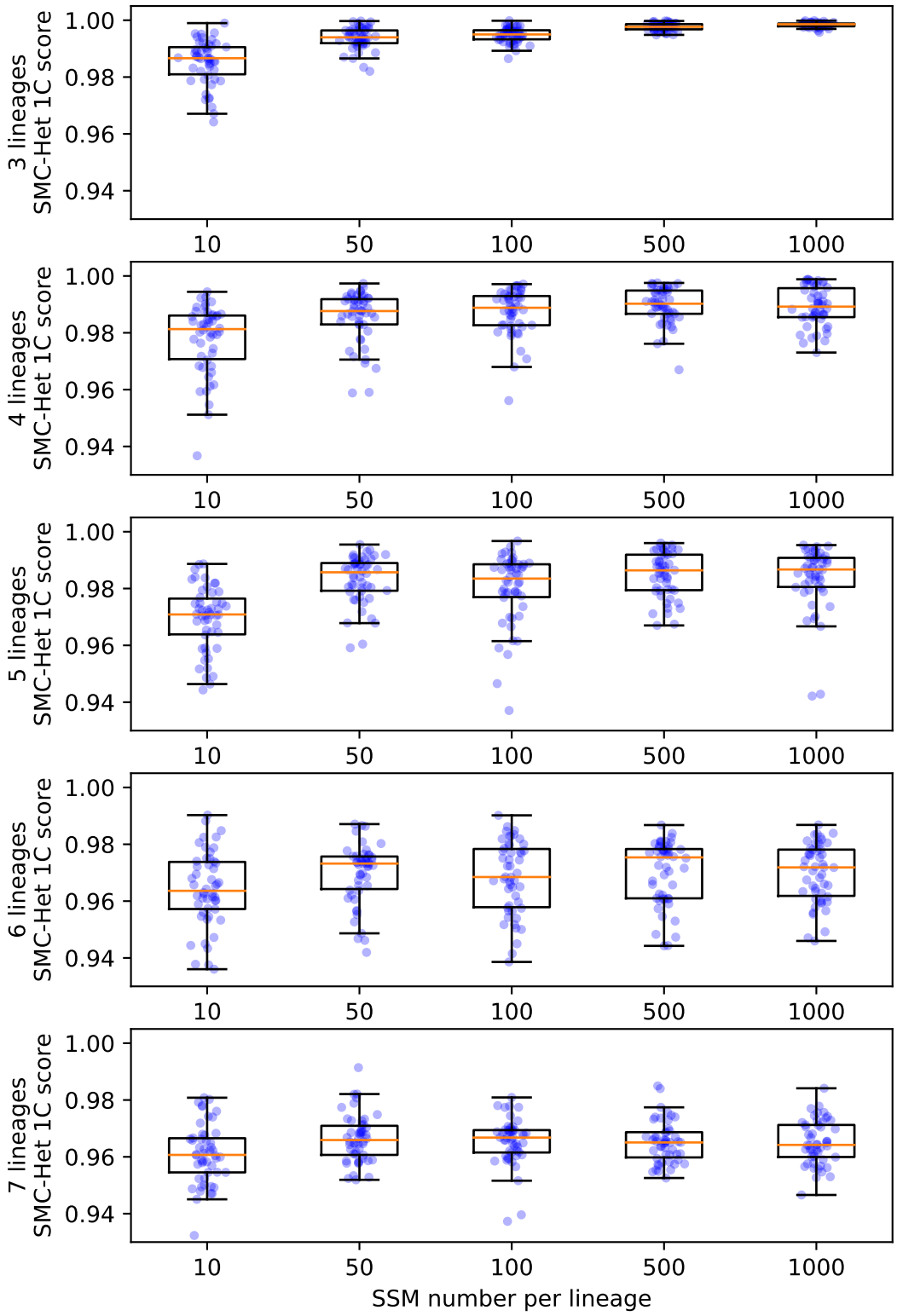


Figure 5.15.: SMC-Het 1C scores for second lineage frequency fixation experiment. SMC-Het 1C scores are shown for different lineage numbers for increasing number of SSMs.

Table 5.7.: Parameters for data simulation for third lineage frequency fixation experiment.

parameter	parameter values		
lineage number K	4	6	
segment number I	10	50	100
SSM number	$10 \cdot (K-1) \cdot I$		
segments with copy number changes	$0.5 \cdot I$		

SSMs by downsampling. However, since we did not sample CNAs, the VAFs between datasets with different SSM numbers should not differ too much, still enabling a valid comparison.

In this subsection, we showed that by combining SSMs in CNA-free regions to superSSMs, the optimization can find and prove the optimal solution in a short time. The SMC-Het 1C scores, which reflect the correctness of the inferred lineage frequencies, are higher than 0.93 for all datasets. This leads to the conclusion that the inferred lineage frequencies are close to the correct ones if at least ten SSMs per lineage are present in the dataset. We did not test whether this is also the case for fewer SSMs. However, with more SSMs, the inference of lineage frequencies can be improved. Thus, if the input data consists of multiple CNA-free segments, we recommend combining these segments to one *super segment*, so that SSMs across these segments can be clustered together.

5.7.3. Performance with Inferred Lineage Frequencies

In this subsection, we analyze how Onctopus performs when the lineage frequencies are fixed to values that were inferred from a previous optimization only with SSMs of CNA-free segments.

We created 300 simulated datasets with different numbers of lineages and segments (see Table 5.7). The number of sampled SSMs depends on the number of lineages and segments. We sampled one or two copy number changes in one lineage for half of the segments. We used five different phylogenetic trees with different lineage frequencies for each lineage number and simulated ten datasets for each combination of tree, lineage and segment number. The simulation setup can be found in Subsection B.5.2.

To infer lineage frequencies, which can be fixed in a second optimization, we built subclonal reconstructions based on all SSMs on CNA-free segments. We combined these SSMs to superSSMs using k-means clustering with $3 \cdot (K-1)$ clusters. The CNA-

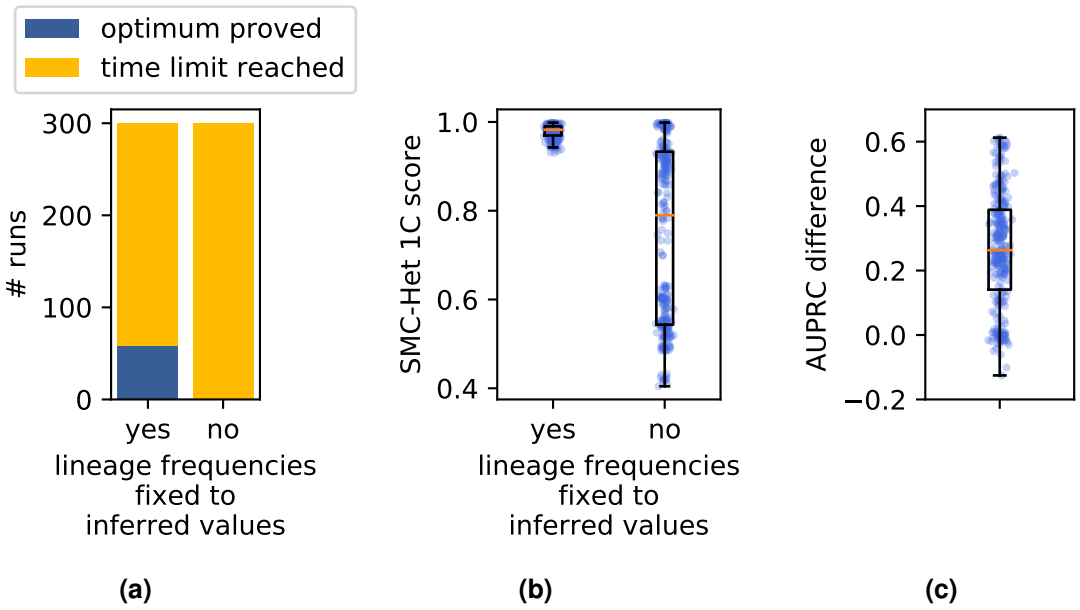


Figure 5.16.: (a) Optimization statuses, (b) SMC-Het 1C scores and (c) AUPRCs of SSM co-clustering for third lineage frequency fixation experiment.
(c) AUPRC difference is calculated as taking the AUPRC of the SSM co-clustering of the optimizations where the lineage frequencies were fixed to the correct values and subtracting the AUPRC of the optimizations without lineage frequency fixation.

free, combined *super segment* was restricted to have zero copy number changes. We ran Onctopus with an optimization time restriction of ten minutes on one thread and without memory restriction.

For the optimization on the complete datasets, we used a run time restriction of two hours on a single thread and no memory restriction. Segments without copy number changes were restricted to contain zero copy number changes. We ran Onctopus once without lineage frequency fixation and once with lineage frequencies fixed to the inferred values.

Optimization Statuses. When the lineage frequencies were fixed to the inferred values, the optimal solution could be proved for 57 of the 300 runs (see Figure 5.16a). When the lineage frequencies were not fixed, no optimal solution could be proved.

Comparison of Found Subclonal Reconstructions. To compare Onctopus' performance when run without and with lineage frequency fixation, we use the SMC-Het 1C score and the AUPRC of the SSM co-clustering. The SMC-Het 1C scores of the sub-

clonal reconstructions built with fixed lineage frequencies mainly reflect the accuracy of the lineage frequencies, which we inferred in the first optimization. The AUPRC also allows an evaluation of Onctopus' performance in the second optimization with the complete input data.

The SMC-Het 1C scores of the optimization with fixed lineage frequencies are significantly higher than the ones of the optimizations without lineage frequency fixation (p-value $< 2.2 \cdot 10^{-16}$, Mann-Whitney U test, see Figure 5.16b).

When the lineage frequencies are fixed to the inferred values, the AUPRC of the SSM co-clustering can be significantly improved for nearly all optimizations (p-value $< 2.2 \cdot 10^{-16}$, Mann-Whitney U test, see Figure 5.16c).

In this subsection, we showed on the given datasets that the performance of Onctopus can be significantly improved if the lineage frequencies are fixed to values that were inferred with SSMs on CNA-free segments. Thus, fixing the lineage frequencies in this way can be a good approach to faster arrive at a better solution.

5.8. Approximating Variant Allele Frequencies in Mixed Integer Linear Program

To compute the VAF $\hat{p}_{j,n}$ (see Equation 3.26) of an SSM j in sample n with our MILP, we have to approximate $\hat{p}_{j,n}$ with $\tilde{p_{j,n}}$ (see Equation 3.53 on page 49) by substituting the inferred average copy number $\hat{c}_{i,n}$ with the observed average copy number $c_{i,n}$. If $\hat{c}_{i,n} \neq c_{i,n}$ for a segment i in sample n, the optimal found subclonal reconstruction r does not maximize the log-likelihood $\mathcal{L}'$ defined by our model (see Equation 3.5) but is an approximation of it. Assume that $\hat{p}^*_{j,n}$ is the VAF that maximizes $\mathcal{L}'$, with $\hat{p}^*_{j,n} = \frac{\hat{s}^*_{j,n}}{\hat{c}_{i,n}}$. If $c_{i,n} < \hat{c}_{i,n}$, then the inferred average copy number $\hat{s}_{j,n}$ of SSM j needs to be smaller than $\hat{s}^*_{j,n}$, in order to compute $\hat{p}^*_{j,n}$. If $c_{i,n} > \hat{c}_{i,n}$, then $\hat{s}_{j,n}$ needs to be larger than $\hat{s}^*_{j,n}$. Thus, the lineage frequencies could be inferred differently or SSMs could be assigned to a different lineage than in the correct subclonal reconstruction.

In this section, we investigate the impact of approximating VAFs on the performance of Onctopus. Therefore, we analyze the difference between the inferred average copy number $\hat{c}_{i,n}$ and the observed average copy number $c_{i,n}$. We evaluate differences in Onctopus' performance when the VAFs are not approximated.

If a solution is not proved to be optimal, it can change when the optimization is done for a longer time. Since we wanted to analyze the effect of approximating VAFs on

Table 5.8.: Parameters for data simulation for investigating impact of approximating VAFs.

parameter	parameter values	
lineage number K	3	4
segment number I	2	3
segments with copy number changes	1	2
SSM number per non-lost allele of each segment of each cancerous lineage	$(K-1) \cdot I$	$2 \cdot (K-1) \cdot I$

optimal subclonal reconstructions, we created datasets for which the optimal solution could be proved.

We created 320 datasets with different numbers of lineages, segments, segments with copy number changes and SSMs (see Table 5.8). We sampled one or two copy number changes in one lineage in each segment that should contain copy number changes. The average copy numbers of the segments were simulated with noise. Each allele, which was not deleted by a copy number loss, in each segment of each cancerous lineage, got the same number of SSMs assigned. We created two phylogenetic trees with different lineage frequencies for each lineage number (see Figure B.5) and simulated ten datasets for each combination of tree, lineage, segment, segments with copy number changes and SSM number. The simulation setup can be found in Section B.6.

We ran Onctopus with an optimization time restriction of five hours on a single thread without memory restriction with standard parameters and for three different fixation settings. In the first fixation setting, we did not fix anything. In the second fixation setting, we fixed each average allele-specific copy number $\hat{c}_{\alpha_{i,n}}$ to the true average allele-specific copy number $c^*_{\alpha_{i,n}}$, while at the same time changing Equation 3.53 to divide by the true average copy number $c^*_{i,n}$, with $c^*_{i,n} = c^*_{A_{i,n}} + c^*_{B_{i,n}}$. Thus, the MILP computes the VAFs exactly and does not approximate it. In the third fixation setting, we fixed $\hat{c}_{\alpha_{i,n}}$ to the inferred value $\hat{c}'_{\alpha_{i,n}}$ of the optimization without fixation, while changing Equation 3.53 to divide by the previously inferred average copy number $\hat{c}'_{i,n}$, with $\hat{c}'_{i,n} = \hat{c}'_{A_{i,n}} + \hat{c}'_{B_{i,n}}$. In this setting the VAFs are also computed exactly. For all runs in all settings, we used the correct lineage numbers.

Optimization Statuses. The optimal solution could be proved for all 320 runs in all three fixation settings.

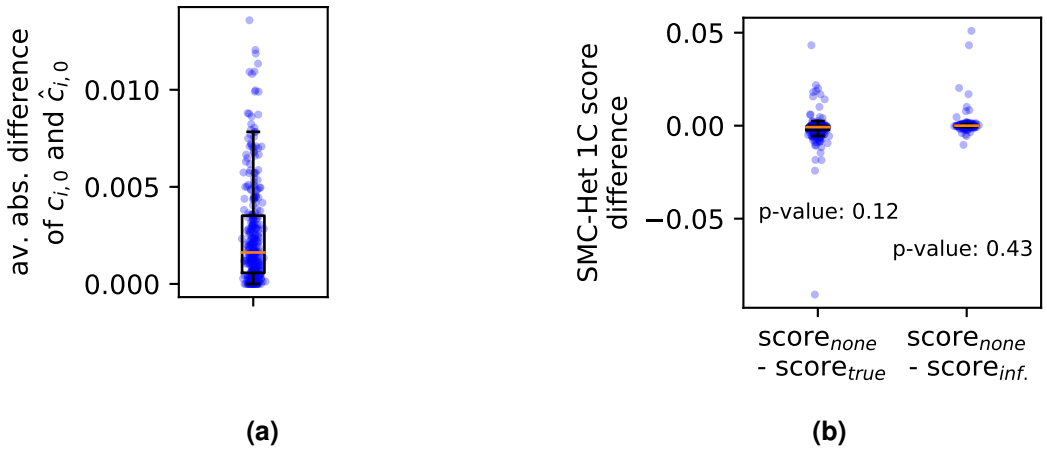


Figure 5.17.: (a) Copy number differences and (b) performance differences of experiment to investigate VAF approximation.
(a) Average absolute copy number differences between the observed and the inferred copy numbers of all segments are computed for the subclonal reconstructions of the optimization without fixation. (b) Performance differences between the SMC-Het 1C scores of the subclonal reconstructions without and with fixation in the optimization are shown. Differences are computed by taking the SMC-Het 1C scores of the subclonal reconstructions without fixation in the optimization (score$_{none}$) and subtracting the scores of the subclonal reconstructions with fixation to true copy numbers (score$_{true}$), or previously inferred copy numbers (score$_{inf.}$), respectively. The p-values were computed with a Mann-Whitney U test.
av. abs.: average absolute, inf.: inferred

Copy Number Difference. For each subclonal reconstruction found in the optimization without fixation, we computed the average absolute difference of the observed copy number $c_{i,0}$ and the inferred copy number $\hat{c}_{i,0}$ over all I segments. For more than three quarters of the subclonal reconstructions, the difference is smaller than 0.005 (see Figure 5.17a).

Performance Differences with Exact VAF Computation. Differences between the SMC-Het 1C scores of the subclonal reconstructions of the two optimizations without fixation and with fixation to the true copy numbers are very small and not significant (p-value $= 0.12$, Mann-Whitney U test; see Figure 5.17b). Between the SMC-Het 1C scores of the subclonal reconstructions of the two optimizations without fixation and with fixation to the previously inferred copy numbers, the differences are even smaller and less significant (p-value $= 0.43$, Mann-Whitney U test).

For the given datasets, the differences between the inferred and observed copy numbers were small, and thus, approximating the VAFs did not influence the optimization significantly. Differences in inferred and observed copy numbers could be analyzed for larger datasets, for which the optimal solution could not be proven. It is possible that approximating the VAFs has an effect on the performance in these cases.

Chapter 6

Results and Evaluation

In this chapter, we compare the performance of Onctopus against the performance of PhyloWGS [21] and Canopy [46]. To our knowledge, these are the only methods reconstructing consistent phylogenies along the genome with all provided CNA and SSM information from bulk-sequencing data. Compared to Onctopus and Canopy, PhyloWGS does not infer CNAs but need them as input.

We start in Section 6.1 by introducing three metrics that we use in this chapter. Afterwards, we evaluate the general performance of Onctopus, PhyloWGS and Canopy in simulated datasets on Section 6.2. At the end in Section 6.3, we present results on a breast cancer dataset.

6.1. Evaluation Metrics

Next to the SMC-Het 1C score, which we used in Chapter 5 as a measure of general performance, we use the SMC-Het 2B and the SMC-Het 3B score in this chapter to evaluate different aspects of the inferred subclonal reconstructions. Also, we evaluate the ability to infer CNAs.

SMC-Het 2B Score. Like the SMC-Het 1C score, the SMC-Het 2B score was developed for the SMC-Het challenge. It evaluates the lineage assignment of SSMs based on their co-clustering. When multiple subclonal reconstructions are given, the evaluation is based on the average co-clustering. Unlike simply calculating the AUPRC between the true and the inferred SSM co-clustering (see Section 5.3), the SMC-Het 2B score calculates the mean of the three normalized correlation measures Pearson correlation

coefficient, Matthews correlation coefficient and average Jensen-Shannon divergence. A score of 0 shows the worst performance and a score of 1 the best performance.

SMC-Het 3B Score. The SMC-Het 3B score was also developed for the SMC-Het challenge. It assesses the inferred lineage relationships between lineages and is based on the lineage assignment of SSMs. Given one or multiple subclonal reconstructions, it processes the probabilities of the four relationships in which two lineages, to which SSM j and j' are assigned, can be: The lineages can be identical, one lineage can be an ancestor or a descendant of the other lineage, or both lineages can lie on different branches of the phylogeny. The poorest possible performance is indicated by a score of 0, while 1 shows the best performance of a subclonal reconstruction, in which all relationships between lineages containing SSMs are inferred as in the ground truth reconstruction.

Given a subclonal reconstruction built by Onctopus with two lineages being in an ambiguous relationship, we weight both probabilities that one lineage is an ancestor of the other and that both lineages are on different branches of the phylogeny to be 0.5.

Evaluation of Copy Number Aberration Inference. We evaluate the CNA inference by comparing the inferred copy number gains and losses with the correct ones of the ground truth dataset. For all segments of a dataset, we count the exact copy number change of gains and losses that got over- or underestimated and normalize these values by the number of segments.

6.2. Results on Simulated Data

To compare the general performance of Onctopus, PhyloWGS and Canopy, we simulated 300 datasets with different parameters which we describe in Subsection 6.2.1. Afterwards, in Subsection 6.2.2, we explain the setting the three methods were run with. In Subsection 6.2.3, we present results on the number of inferred lineages, the performance in terms of SMC-Het scores and the inferrence of CNAs, which we then discuss in Subsection 6.2.4.

Table 6.1.: Parameters for data simulation. Bold values represent the parameter set that was used to generate reference datasets. Additional nine parameter sets were created by changing one parameter at a time to one of the neighboring values.

parameter	parameter values		
lineage number	2	**4**	6
segment number I	2	**20**	200
SSM number	$10 \cdot I$	$\mathbf{100 \cdot I}$	$1000 \cdot I$
coverage	50	**100**	500
number of lineages with copy number changes per segment		**1**	2

6.2.1. Data Simulation

We simulated 300 datasets with different parameters. The reference datasets consist of four lineages, 20 segments and 2000 SSMs. For half of the segments, one or two copy number changes were simulated and assigned to one lineage per segment. We simulated the reference datasets with a coverage of 100. Then, we created further datasets by changing only one parameter (see Table 6.1), resulting in ten different parameter combinations. For each lineage number, we created three different phylogenetic trees with lineage frequencies and simulated ten datasets for each tree and parameter combination. The simulation set up can be found in Section B.7.

6.2.2. Inferring Subclonal Reconstructions

In this subsection, we describe the settings we used to run the three methods Onctopus, PhyloWGS and Canopy.

Onctopus. We ran Onctopus on all datasets for two to nine lineages. We used three main optimizations, where the memory usage of each optimization was restricted to 50 GB. First, we combined all CNA-free segments of a dataset to one *super segment* and clustered SSMs in this segment to superSSMs using k-means and $3 \cdot (K-1)$ clusters, with K being the number of lineages. We ran Onctopus with an optimization time of ten minutes with five threads. Afterwards, we fixed the lineage frequencies $\phi_{k,0}$ of the second optimization to the inferred lineage frequencies of the first optimization, and clustered SSMs to superSSMs using again k-means and $3 \cdot (K-1)$ clusters per segment. We used an optimization time of four hours on five threads. Then, we used the inferred lineage frequencies $\phi_{k,0}$, the inferred lineage relationships $Z_{k,k'}$, the inferred CNA assignments $\Delta C_{A_{i,k}}$ and $\Delta C_{B_{i,k}}$, and the inferred SSM assignments $\Delta S_{A_{j,k}}$ and $\Delta S_{B_{j,k}}$ as

start values for the third optimization. Again, we optimized for four hours on five threads. All optimization parameters not mentioned have standard values.

At the end, we determined the lineage number (see Section 3.5) for each dataset and used the corresponding subclonal reconstruction for the following evaluation.

PhyloWGS. For each dataset, we ran the parameter-free PhyloWGS three times and selected the run with the best overall log-likelihood.

PhyloWGS runs only on a single thread and does not allow to cluster SSMs. Thus, it can take considerable of run time when applied to a large dataset. On the 60 datasets with 20,000 SSMs, we let it run between 14.5 to 16.5 days. Afterwards, the highest reached sampling iteration number after burn-in was only 1130 instead of 2500. The lowest reached iteration number was 90, and the median 410.

To evaluate the number of lineages, we chose the subclonal reconstruction with the highest log-likelihood. The measuring of the other metrics was done over all 2500 (or less) sampled subclonal reconstructions and with the subclonal reconstruction having the highest log-likelihood of the chosen run for each dataset.

Canopy. For all datasets but the ones with 20,000 SSMs, we ran Canopy with three chains, 5000 burn-in iterations and 50,000 iterations in total per chain. For the 60 datasets with 20,000 SSMs, we only used two chains and 25,000 iterations to save run time and memory. We ran all datasets with SSM clustering and with two to seven non-vestigial populations. We used BIC implemented in Canopy to infer the population number for each dataset and used the corresponding subclonal reconstructions for the following evaluation.

For 15 of the 300 datasets, Canopy was not able to build a subclonal reconstruction and terminated with an unexplained error message. Of these datasets, 13 were created with the parameter set with two segments and two with the one with two lineages. We will not present results of Canopy on these datasets.

For evaluating the number of lineages, we chose the configuration with the highest posterior probability. The measuring of the other metrics is done over all subclonal reconstructions of the chosen population number.

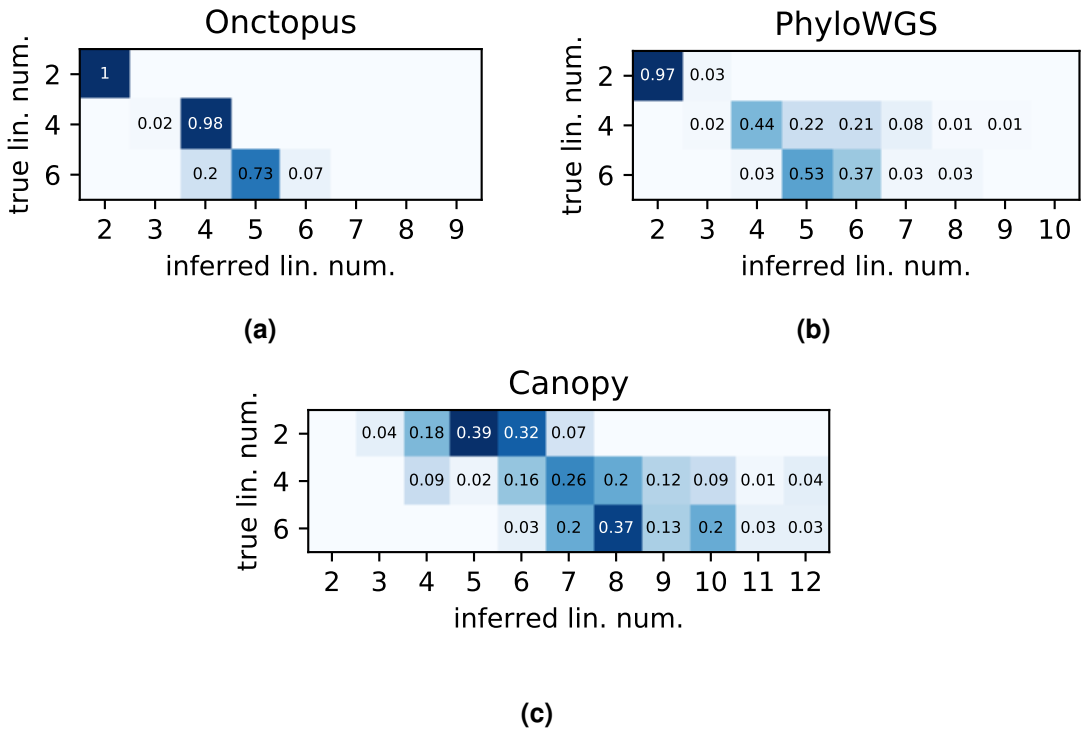


Figure 6.1.: Inferred lineage numbers of (a) Onctopus, (b) PhyloWGS and (c) Canopy.
Numbers of inferred lineages are normalized by numbers of datasets with true lineage numbers. There are 30 datasets with a true lineage number of two and six, and 240 datasets with a true lineage number of four. For Canopy, there are only 28 datasets with a true lineage number of two and 227 datasets with a true lineage number of four since Canopy could not build subclonal reconstructions for all datasets.
lin. num.: lineage number

6.2.3. Results

We evaluate the performance of Onctopus, PhyloWGS and Canopy by comparing the lineage numbers they inferred on the simulated datasets, their SMC-Het scores and their CNA inference.

Lineage Number Inference. Onctopus perfectly infers the lineage numbers of the 30 datasets with a true lineage number of two (see Figure 6.1a), and nearly perfectly infers the lineage numbers of the 240 datasets with a true lineage number of four. For 93% of the 30 datasets with a true lineage number of six, Onctopus infers a too low lineage number of four or five. PhyloWGS nearly perfectly infers the lineage num-

bers for the datasets with two (see Figure 6.1b) lineages but overestimates the lineage numbers for the datasets with four lineages. The lineage numbers of the datasets with six lineages are mostly underestimated. Canopy highly overestimates the number of lineages for all datasets (see Figure 6.1c), with in some cases even eight more lineages. Hence, Onctopus infers the lineage numbers best, while Canopy infers them worst.

Evaluating Subclonal Reconstructions. The performances of Onctopus, PhyloWGS and Canopy in terms of the SMC-Het 1C, SMC-Het 2B and SMC-Het 3B scores can be found in Figure 6.2. On all datasets and with all metrics, Canopy has the worst performance of the three methods. Depending on the datasets and the metric, Onctopus performs better, equally well or worse than PhyloWGS.

For the SMC-Het 1C score, Onctopus performs significantly better than PhyloWGS averaged over all sampled subclonal reconstructions on the datasets with ten SSMs on average per segment, two segments, two lineages, a coverage of 500 and two lineages with copy number changes in one segment (see Figure 6.2a). For these datasets with ten SSMs and two segments, Onctopus' performance does not differ significantly from PhyloWGS' when evaluated on the single best subclonal reconstruction. PhyloWGS outperforms Onctopus on the reference datasets and the datasets with 200 segments and a coverage of 50. On the datasets with 1000 SSMs on average per segment, PhyloWGS outperforms Onctopus only when evaluated on the single best subclonal reconstruction. PhyloWGS averaged over all sampled subclonal reconstructions has significantly higher SMC-Het 2B scores on all datasets but the ones with two segments, two lineages, a coverage of 500 and two lineages with copy number changes in one segment, where PhyloWGS and Onctopus perform equally well (see Figure 6.2b). However, compared to the single best subclonal reconstruction of PhyloWGS, Onctopus shows equally well or even significantly better performance over all datasets. The SMC-Het 3B scores of Onctopus are significantly better than the ones of PhyloWGS averaged over all sampled subclonal reconstructions for all datasets except for the ones with six lineages, where PhyloWGS performs significantly better than Onctopus, and the ones with two lineages and a coverage of 50, where no significant difference between the two methods can be measured (see Figure 6.2c). In comparison to PhyloWGS' single best subclonal reconstruction, Onctopus performs also significantly better on the datasets with six lineages and a coverage of 50.

CNA Inference. Since PhyloWGS gets the correct CNAs as input, it infers them perfectly on all datasets. Onctopus overestimates the number of copy number gains in

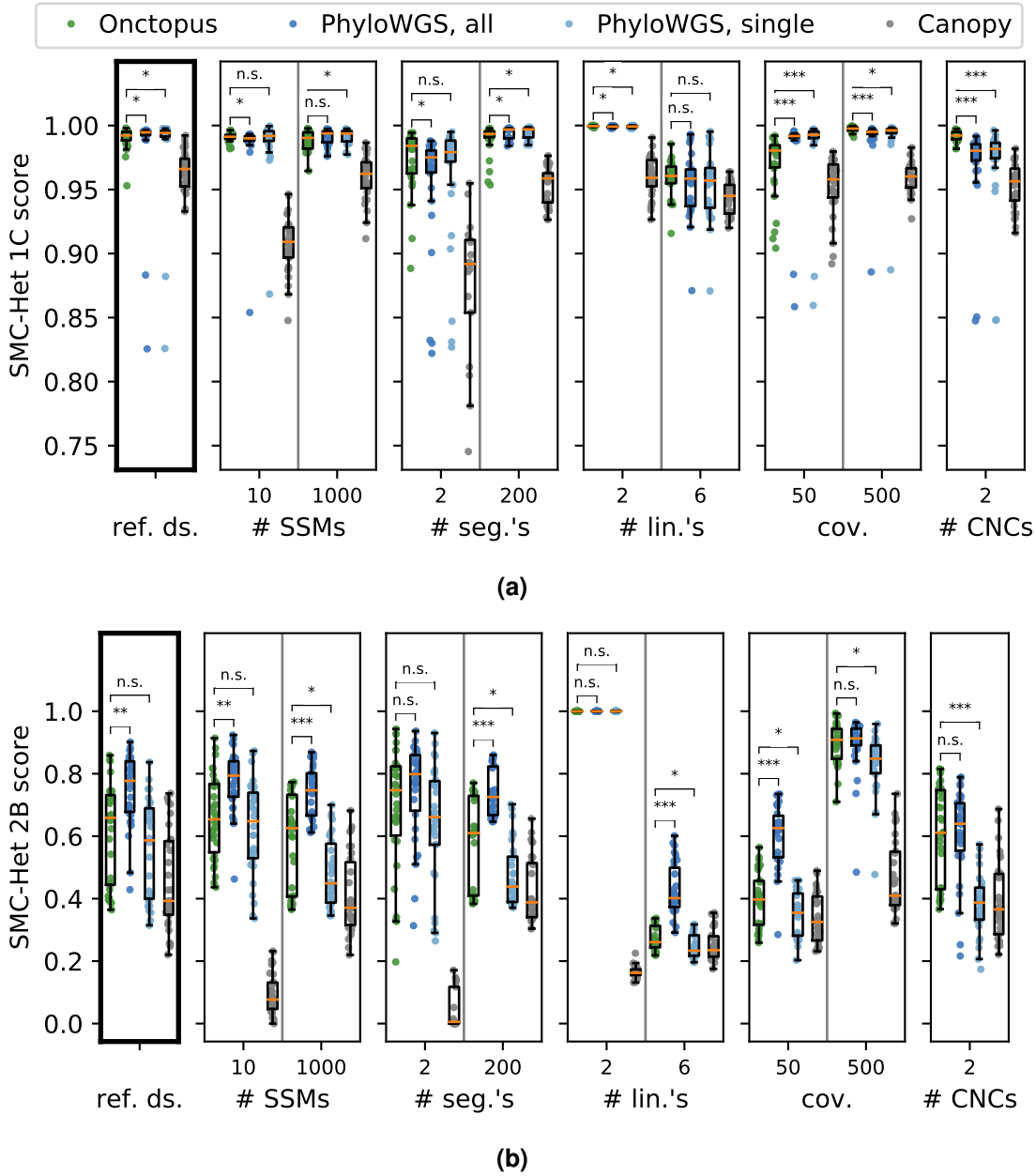


Figure 6.2.: (a) SMC-Het 1C, (b) SMC-Het 2B and (c) SMC-Het 3B scores for Onctopus, PhyloWGS and Canopy.
Full caption see next page.

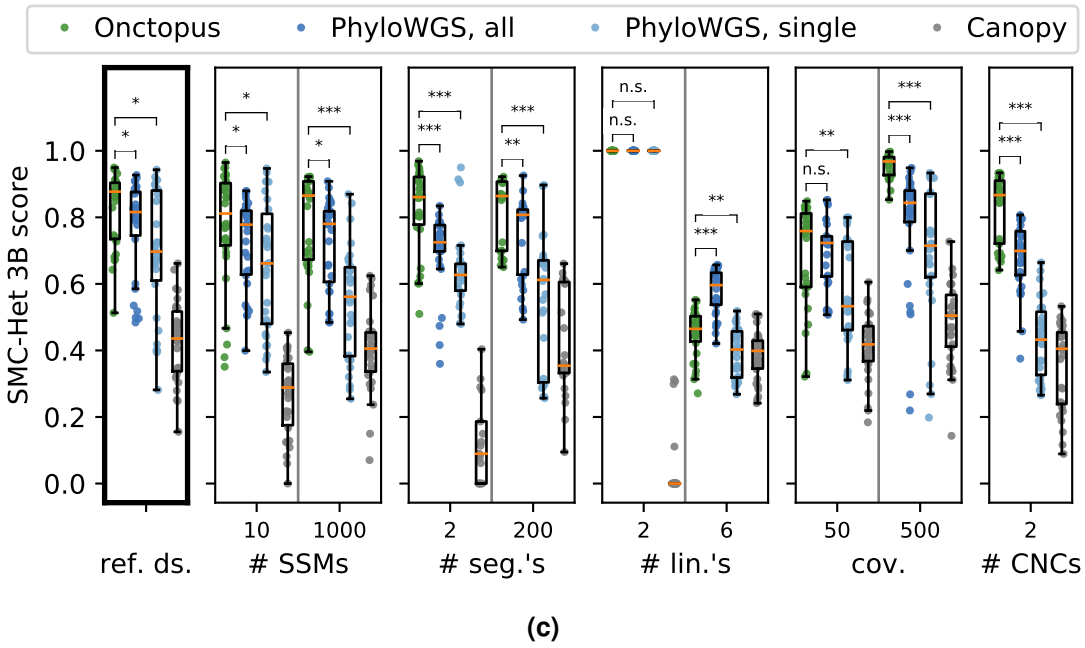


Figure 6.2.: (a) SMC-Het 1C, (b) SMC-Het 2B and (c) SMC-Het 3B scores for Onctopus, PhyloWGS and Canopy.
Leftmost frame shows performance on reference datasets with on average 100 SSMs per segment, 20 segments, four lineages, a coverage of 100 and one lineage with copy number changes in one segment. Other frames show performance on datasets with changed parameters. Between Onctopus and PhyloWGS p-values are computed with a Mann-Whitney U test: n.s.: p-value ≥ 0.05, *: $0.05 <$ p-value ≤ 0.001, **: $0.001 <$ p-value < 0.0001, ***: p-value ≤ 0.0001. No results of Canopy are shown for the datasets on which it could not build subclonal reconstructions.
PhyloWGS, all: evaluation over all sampled subclonal reconstructions; PhyloWGS, single: evaluation of subclonal reconstruction with highest log-likelihood
n. s.: not significant, ref. ds.: reference datasets, seg.'s: segments, lin.'s: lineages, cov.: coverage, CNCs: lineages with copy number changes in one segment

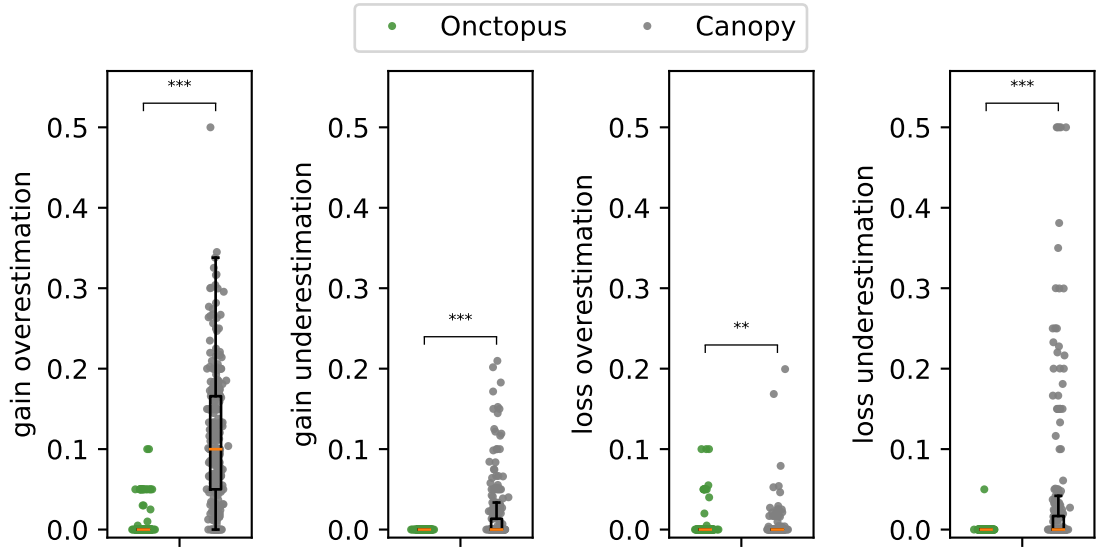


Figure 6.3.: CNA inference of Onctopus and Canopy.
The p-values are computed with a Mann-Whitney U test: **: $0.001 <$ p-value < 0.0001, ***: p-value ≤ 0.0001. No results of Canopy are shown for the datasets on which it could not build subclonal reconstructions.

24 and the number of losses in 14 of 300 cases (see Figure 6.3) but infers them significantly better than Canopy on average. Canopy overestimates the number of copy number gains for all but 26 datasets and it underestimates the number of copy number gains and losses for half of the datasets. Note that the over- and underestimation of copy number gains or losses on the same dataset is possible since we measure the normalized over- and underestimation per segment.

6.2.4. Discussion

In this section, we compared Onctopus' performance against PhyloWGS' and Canopy's in terms of lineage number inference, SMC-Het scores and CNA inference.

Onctopus underestimates the number of lineages in datasets with a true lineage number of six. This could be explained by the description length $L_{\mathcal{C}_2}(r)$ being too large. One way to decrease the description length is to not encode the SSM assignment (parts (7), (8) and in (9) in Section 3.5). Neglecting it can be justified because the computation of the SSM assignment that maximizes the likelihood is straight-forward when all other parts of the subclonal reconstruction r are given. It needs to be tested

whether this change in the description length leads to a better inference of the lineage number or to an overestimation of the number of inferred lineages in the datasets with less than six lineages.

For PhyloWGS and Canopy, we evaluated only the lineage number inference of the best subclonal reconstruction. The lineage numbers of the other sampled subclonal reconstructions remain to be evaluated in the future.

Canopy has the worst performance of all three used methods. One reason for the bad performance could be that Canopy highly overestimates the number of lineages and thus assigns SSMs that belong to a single lineage to multiple lineages, which leads to low scores. On the other hand, this does not explain Canopy's poor performance on the SMC-Het 1C score since it is independent of the number of lineages. Furthermore, Canopy also has problems inferring CNAs. This leads to the conclusion that Canopy's poor performance is not only the result of a lineage number overestimation but rather of Canopy's general difficulty to build subclonal reconstructions from our simulated datasets. A potential problem for Canopy could be the providing of a single sample instead of multiple ones that could have helped in resolving ambiguity.

The SMC-Het 2B score considers the average SSM assignment over the given subclonal reconstructions, which reflects ambiguity. When evaluated over all sampled subclonal reconstructions, PhyloWGS obtains significantly higher SMC-Het 2B scores than Onctopus on nearly all datasets. A reason could be that the VAFs of the SSMs are noisy and that VAFs of SSMs assigned to different lineages overlap, especially when influenced by copy number changes. Thus, the SSM assignment is highly ambiguous. On the datasets with two lineages, where only one lineage assignment of SSMs exists, or on the ones with a coverage of 500, where the VAFs do not overlap as much, Onctopus performs as well as PhyloWGS. On the datasets with six lineages or a coverage of 50 where the VAFs overlap potentially highly, PhyloWGS outperforms Onctopus. However, when evaluating only the single best subclonal reconstruction inferred by PhyloWGS, Onctopus assigns the SSMs better. Thus, one of PhyloWGS' advantages comes from averaging over multiple subclonal reconstructions that together represent the ambiguity in SSM assignments better than a single subclonal reconstruction. Another advantage of PhyloWGS can be explained by the fact that PhyloWGS does not have to infer the CNAs but receives them as input. On the evaluated datasets it even receives the true underlying CNAs, thus has an advantage over Onctopus. To compare PhyloWGS and Onctopus performance without this CNA advantage, the two methods should be compared on datasets where PhyloWGS does not receive the true

underlying CNAs but the copy number calls of a method that also provides the input for Onctopus. However, we did not simulate our data in a way that copy number calling tools can be applied to it, thus another data simulation would have to be used for this purpose. Another idea to compare the two methods without PhyloWGS' CNA advantage is to start the Onctopus runs with the lineage frequencies, relationships and CNA assignments from a subclonal reconstruction built by PhyloWGS.

In terms of inferring ancestor-descendant relationships between lineages, Onctopus outperforms PhyloWGS, measured over all sampled subclonal reconstructions and on the single best subclonal reconstruction, significantly on nearly all datasets. Thus, our approach of inferring necessary present and absent lineage relationships as well as ambiguous relationships in a single subclonal reconstruction works better on the presented datasets than trying to capture ambiguity by averaging over multiple subclonal reconstructions as done by PhyloWGS.

6.3. Results on a Breast Cancer Dataset

In this section, we evaluate Onctopus, PhyloWGS and Canopy on the whole genome sequenced breast cancer dataset PD4120a by Nik-Zainal *et al.* [72] by following the analysis of Deshwar *et al.* [21]. In Subsection 6.3.1, we briefly describe the dataset, followed by the settings we used to run the three methods in Subsection 6.3.2. Afterwards, in Subsection 6.3.3, we present and discuss the results.

6.3.1. Data Description

In their paper, Deshwar *et al.* [21] show that PhyloWGS outperforms PyClone and SciClone when building subclonal reconstructions on a subset of the whole genome sequenced breast cancer dataset PD4120a by Nik-Zainal *et al.* [72]. As subset, Deshwar *et al.* chose genomic regions for which the same CNAs were called in the original analysis by Nik-Zainal *et al.* and in a later analysis by Oesper *et al.* [75]. These regions are the full length chromosomes 3, 4q, 5, 10, 13, 16p, 17, 19 and 20, with a total of 24,109 SSMs. Chromosomes 3, 5, 10, 16p, 17, 19 and 20 are CNA-free, whereas chromosomes 4q and 13 are affected by a copy number loss.

For their paper, Nik-Zainal *et al.* performed a semi-manual SSM clustering on the PD4120a dataset, which results in three main clusters. Deshwar *et al.* used this clustering as a gold standard to which they compared the co-clustering of the SSMs in their analysis in terms of AUPRC. In this work, we use the SMC-Het 2B score to evaluate

the SSM co-clustering on an updated SSM clustering (unpublished data provided by David Wedge, used in [20]) which contains the cluster assignment confidence for each SSM.

To get the average allele-specific copy numbers of the two genome segments affected by the heterozygous copy number losses, we used the purity values provided by Nik-Zainal *et al.* We computed the standard error of all segments as described in Section 5.2 on page 94, where we do not let the number of heterogeneous SNPs n_{SNPs_i} in segment i exceed the larger of the two values number of SSMs in segment i or 3000. As haploid coverage, we used half of the median SSM count and standard values for all other parameters.

6.3.2. Inferring Subclonal Reconstructions

We describe in this subsection the settings we used to run the three methods Onctopus, PhyloWGS and Canopy.

Onctopus. We ran Onctopus with three main optimizations on the chosen chromosomes of the PD4120a dataset with two to nine lineages and a memory restriction of 50 GB. First, we combined all CNA-free segments to one *super segment* and clustered the SSMs in this segment with k-means and $3 \cdot (K-1)$ clusters, where K is the lineage number. We optimized for ten minutes on five threads to get a first estimate of the lineage frequencies. Again, we combined all CNA-free segments to one *super segment* and clustered all SSMs with k-means and $3 \cdot (K-1)$ clusters. We then fixed the lineage frequencies in a second optimization and optimized for two hours on ten threads. We used the results of the second optimization as start values for the third optimization where we restricted the run time again to two hours on ten threads. All optimization parameters not mentioned have standard values. The subclonal reconstruction with four lineages is the one that minimizes our MDL criterion and which we use for further evaluation.

PhyloWGS. We ran PhyloWGS three times on the chosen chromosomes of the PD4120a dataset and chose the run with the overall highest log-likelihood.

Canopy. We ran Canopy with three chains, a total of 15,000 iterations per chain of which 5000 iterations were burned-in. We used SSM clustering and ran Canopy with three to six non-vestigial populations. For the evaluation, we use the subclonal

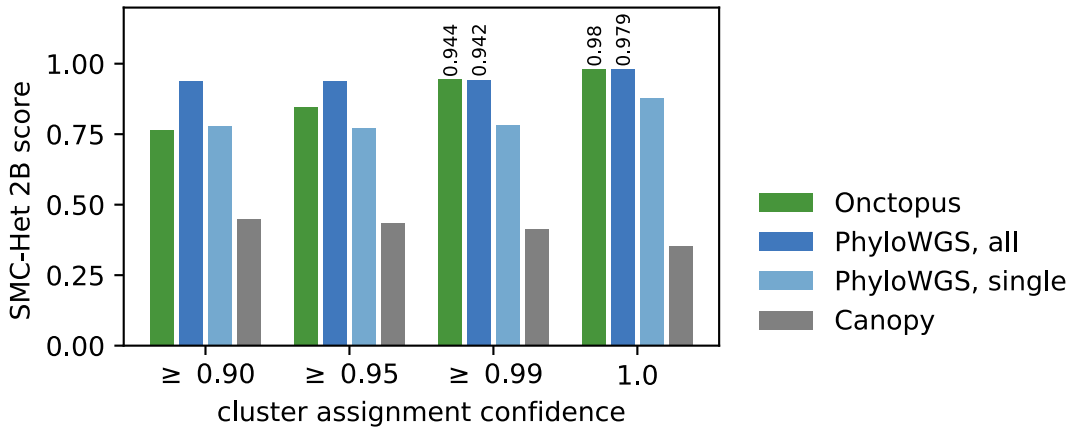


Figure 6.4.: SMC-Het 2B scores for Onctopus, PhyloWGS and Canopy.
Scores are shown for decreasing number of SSMs with increasing confidence to be assigned to one of the three main clusters identified by Nik-Zainal *et al.*
PhyloWGS, all: evaluation over all 2500 sampled subclonal reconstructions; PhyloWGS, single: evaluation of subclonal reconstruction with highest likelihood

reconstructions with six non-vestigial populations as they were chosen by Canopy's provided BIC criterion.

6.3.3. Results and Discussion

We evaluate the performance of Onctopus, PhyloWGS and Canopy against the SSM clustering of Nik-Zainal *et al.* in terms of the SMC-Het 2B score (see Figure 6.4). PhyloWGS, when averaging over all 2500 sampled subclonal reconstructions, already achieves an SMC-Het 2B score of 0.94 on SSMs with a cluster assignment confidence of at least 0.9. The SMC-Het 2B score slowly increases to 0.98 when only SSMs are considered that are assigned to one of the main clusters with a confidence of 1.0. Onctopus starts with a lower SMC-Het 2B score of 0.77 that reaches PhyloWGS' score when considering only SSMs with a cluster assignment of at least 0.99 or 1.0. Like in Subsection 6.2.3, where we evaluated the methods on simulated datasets, we see that PhyloWGS, when averaging over all subclonal reconstructions, clusters SSMs with ambiguous assignments better than Onctopus. However, the less ambiguous the cluster assignment of SSMs gets, the smaller the difference between the two methods gets until Onctopus performs equally well with a single subclonal reconstruction. When evaluating only the best subclonal reconstruction built by PhyloWGS, Onctopus already performs better for SSMs with a cluster assignment confidence of at least 0.95.

Compared to Onctopus and PhyloWGS, the SMC-Het 2B score for Canopy decreases for increasing cluster assigment confidence of SSMs. The built subclonal reconstructions of Canopy seem to not represent the clustering by Nik-Zainal *et al.*

Chapter 7

Conclusion and Outlook

In this thesis, we presented the first lineage-based subclonal reconstruction method working with SSM and CNA information of bulk-sequenced tumor samples. Modeling SSMs and CNAs in a lineage-based approach as well as an extensive analysis of lineage relationships in a subclonal reconstruction allow us to combine multiple subclonal reconstructions with ambiguous lineage relationships within a single subclonal reconstruction.

After giving background information in Chapter 2, we introduced our new lineage-based subclonal reconstruction model in Chapter 3. We presented a joint likelihood function for SSMs and CNAs and explained the different model components and rules that apply to them. By modeling CNAs not as absolute copy numbers but as copy number changes, we are able to assign them to lineages. Another key feature of our method is that we model ancestor-descendant relationships between lineages as either present, absent or ambiguous. We showed how we derive a linear relaxation of our model and explained the variables and constraints of our MILP. After analyzing the complexity of our MILP, we described how we determine the number of lineages with the two-part version of the MDL principle.

In Chapter 4, we defined the concept of ambiguity and presented an algorithm that finds ambiguous ancestor-descendant relationships after the optimization. The algorithm consists of seven main steps in which necessary present and absent ancestor-descendant relationships that either influence the likelihood or are crucial for the validity of the subclonal reconstruction are found. We finished Chapter 4 by showing that our lineage-based method can model ambiguous subclonal reconstructions within a single subclonal reconstruction while population-based methods cannot.

In Chapter 5, we briefly talked about the implementation of Onctopus and explained how we simulated datasets. We extensively evaluated Onctopus on simulated datasets and showed that a good subclonal reconstruction can be found even if its optimality cannot be proved in the given time and space. We showed that Onctopus' performance can be improved by clustering mutations, fixing CNAs or fixing lineage frequencies.

We compared the performance of Onctopus against the performance of PhyloWGS [21] and Canopy [46] in Chapter 6. First, we compared the three methods on simulated datasets on which we could show that Onctopus is superior in inferring the underlying lineage numbers as well as the lineage relationships. PhyloWGS when evaluated over all sampled subclonal reconstructions outperformed Onctopus in terms of mutation assignment. However, compared to PhyloWGS' subclonal reconstruction with the highest log-likelihood, Onctopus' mutation assignments were better. Both methods inferred lineage frequencies equally well, with either both methods performing comparably, or one method outperforming the other, or vice versa on different groups of datasets. On all datasets, Canopy had the worst performance. Second, we evaluated the three methods on a deep sequenced breast cancer dataset following an analysis by Deshwar *et al.* [21]. Again, Canopy showed the worst performance. Onctopus achieved a comparable performance when PhyloWGS' results were evaluated over all sampled subclonal reconstructions, and it even had a better performance when it was compared to the single best subclonal reconstruction of PhyloWGS.

With our lineage-based subclonal reconstruction method implemented in Onctopus, we introduced a new valuable subclonal reconstruction method. In the following, we present different ideas which datasets are further interesting to apply Onctopus on and how Onctopus can be extended to work with multiple samples, capture ambiguity in SSM assignments and be used with single-cell sequencing data.

Comparison on Different Datasets. We compared Onctopus, PhyloWGS and Canopy on datasets we simulated ourselves and on the deep sequenced breast cancer dataset PD4120a [72]. Working with simulated datasets, one has to ask the question how realistic the datasets are. Working with real datasets poses the question to what extend the gold standard is correct.

Our presented data simulation is a straight-forward one. We did not simulate genome sequences from which mutations can be called but directly created the mutation information Onctopus needs as input. A drawback of this strategy is that we had to provide PhyloWGS with true underlying CNAs which gave it a further advantage over Onctopus. This advantage would diminish if we had simulated data from

which the mutations had to be called first, thus both methods were provided with noisy input. An interesting software to simulate such data would be a version of BAMSurgeon [26, 57] that was extended for the SMC-Het challenge with the attempt to simulate realistic BAM files and to allow CNA errors through calling pipelines. However, it is not available yet.

As for the correctness of the gold standard of the PD4120a dataset, we focused in our analysis only on mutations that were assigned to a cluster with confidence of at least 90%, hoping to receive an accurate ground truth dataset.

Another interesting dataset to evaluate Onctopus on is the Pan-Cancer Analysis of Whole Genomes (PCAWG) [11] dataset which comprises more than 2600 cancer genomes across 39 different cancer types. For these cancer genomes, subclonal reconstructions have already been inferred [32] that we could use to compare Onctopus to.

Extension to Multiple Samples. Our presented model and the ambiguity algorithm can already work with multiple samples of the same patient. Just the implementation of Onctopus needs to be adapted which will be our first step as a future extension.

An interesting question that has to be answered when dealing with multiple samples is how to treat overlapping CNAs of different samples. A straight-forward solution would be to start a new segment at each copy number change breakpoint of any sample and use weights in the objective function to reward neighboring segments of the same sample that have equal copy number states.

Currently, we improve Onctopus' performance by fixing the lineage frequencies to values previously inferred from CNA-free segments. Inferring the lineage frequencies from CNA-free segments of multiple samples can become more difficult since a linear phylogeny does not have to be consistent with the data but a special branching phylogeny might be needed. Thus, it needs to be evaluated to what extent this approach is applicable.

Improving SSM Assignment through a Probabilistic Framework. We could show on simulated datasets that PhyloWGS when evaluated over all sampled subclonal reconstructions infers the SSM assignment better than Onctopus. By averaging over different SSM assignments, PhyloWGS is able to model this ambiguity or uncertainty in the input data. To capture the ambiguity in SSM assignment, Onctopus could be extended with a probabilistic framework.

A straight-forward approach to derive probabilities with which an SSM is assigned to different lineages and phases is to treat the inferred lineage relationships and frequencies, as well as the copy number change assignments with phasing as given and calculate the likelihood of the SSM being assigned to each lineage and phase. Once the global variables of the lineage relationships and frequencies and the local variables of the copy number change assignments are fixed, each SSM can be assigned individually to a lineage and phase without influencing the probabilities of the other SSMs. By normalizing the different assignment likelihoods of an SSM, we derive its probabilities of being assigned to a specific lineage and phase. Like this, we can capture ambiguity in SSM assignment.

Sampling SSM assignments from the assignment probabilities could give us a broader picture of subclonal reconstructions with different likelihoods. Still, these subclonal reconstructions have the same lineage frequencies and copy number change assignment as the subclonal reconstruction from which we calculated the SSM assignment probabilities. Applying a broader sampling approach that starts from the probabilities of our optimized subclonal reconstruction is another interesting extension for Onctopus and open for further research.

Using the SSM assignment probabilities can be useful not only to build a probabilistic framework for Onctopus, but also to improve subclonal reconstructions that are not proved to be optimal. Given the assignment probabilities, each SSM should be assigned to the lineage and phase leading to the highest probability. Afterwards, the lineage frequencies can be optimized based on the new SSM assignments and the subclonal reconstruction rules. An iterative approach is possible that finishes when each SSM is assigned to the best lineage and the lineage frequencies do not change anymore.

Combining Bulk and Single-Cell Sequencing Data. The current standard practice to build subclonal reconstructions of cancer samples is based on bulk-sequencing data where only the frequency of mutations is observed. Thus, by using the weak parsimony assumption, mutations with similar frequencies get clustered together in one lineage although they can belong to distinct lineages with similar frequencies. Furthermore, the detection of low frequency variants is challenging to distinguish from noise [16]. Using multiple samples and increasing the coverage can help to prevent these problems. Another solution is to use single-cell sequencing data of cancer samples since mutation profiles are observed per cell. However, single-cell data contain

high noise levels [31]. Hence, approaches combining bulk and single-cell sequencing data are promising.

The method B-SCITE [63] combines bulk and single-cell sequencing data to build full subclonal reconstructions. B-SCITE is a probabilistic method and works with a mutation tree, a special form of a phylogenetic tree where each mutation is represented by its own node. As input data, it takes the variant and total read counts of SSMs, as well as the mutation profiles of single cells. It computes a joint score of a mutation tree measured for bulk and single-cell sequencing data.

Given a tree T, which already includes the mutation assignment, the bulk data based tree score is computed by maximizing the log-likelihood of T by inferring the cellular prevalence of the mutations, which equal Onctopus' lineage frequencies. For the maximization, a quadratic program based on CITUP [64] (see Table 2.1) is applied. Given the tree T and an error profile of probabilities of observing false positive and false negative mutations, the single-cell based tree score is computed as log-likelihood of observing the mutation profiles. An MCMC variant introduced for the subclonal reconstruction method SCITE [45], which works with single-cell sequencing data, is used to sample new mutation trees and error profiles.

Since the computation of B-SCITE's bulk data based tree score is based on the optimization of CITUP, which works only with SSMs, B-SCITE is restricted to copy number neutral regions of the genome. Using an optimization based on Onctopus would allow to model regions with copy number changes as well. The copy number changes can be placed in the tree T like SSMs. We propose restricting the number of copy number changes per segment and allele with CNA to one or two in order to decrease the combinatorial space of placing copy number changes in the tree T. Because the tree T, which also contains the mutation assignment, is given, we can fix the lineage relationships in the optimization. Furthermore, we can fix the mutation assignments to lineages. However, since the phasing still needs to be computed, we have to extend our implementation to enable this fixation. Then, the optimization is done only over the lineage frequencies and the phasing of mutations. It needs to be tested whether using a MILP in each step of the MCMC where the tree T is updated is fast enough. Setting a run time restriction for the optimization is probably necessary. Since the number of lineages corresponds to the number of mutations, we do not have to choose the lineage number by comparing the MDL of different subclonal reconstructions. Thus, the optimization needs to be done only once per MCMC step, saving valuable time. We believe that extending B-SCITE to include regions with copy number changes by using an optimization based on Onctopus is a valuable improvement.

The characterization of a tumor through a subclonal reconstruction is essential to understand the intratumor heterogeneity that influences the outcome of cancer therapies. We believe that with the introduction of our new lineage-based model and our ambiguity algorithm implemented in Onctopus, a new subclonal reconstruction method was created that can provide valuable insides needed to understand this intratumor heterogeneity.

Bibliography

[1] 1000 Genomes Project Consortium and others. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[3] B. Al-Lazikani, U. Banerji, and P. Workman. Combinatorial drug therapy for cancer in the post-genomic era. *Nature Biotechnology*, 30(7):679–692, 2012.

[4] A. Alkodsi, R. Louhimo, and S. Hautaniemi. Comparative analysis of methods for identifying somatic copy number alterations from deep sequencing data. *Briefings in Bioinformatics*, 16(2):242–254, 2014.

[5] American Cancer Society. Global cancer facts & figures 2nd edition. *Atlanta: American Cancer Society*, 1–57, 2011.

[6] R. Beroukhim, C. H. Mermel, D. Porter, G. Wei, S. Raychaudhuri, J. Donovan, J. Barretina, J. S. Boehm, J. Dobson, *et al.* The landscape of somatic copy-number alteration across human cancers. *Nature*, 463(7283):899–905, 2010.

[7] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[8] V. Boeva, T. Popova, K. Bleakley, P. Chiche, J. Cappo, G. Schleiermacher, I. Janoueix-Lerosey, O. Delattre, and E. Barillot. Control-FREEC: a tool for assessing copy number and allelic content using next-generation sequencing data. *Bioinformatics*, 28(3):423–425, 2011.

[9] S. R. Browning and B. L. Browning. Haplotype phasing: Existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[10] D. P. Cahill, K. W. Kinzler, B. Vogelstein, and C. Lengauer. Genetic instability and darwinian selection in tumours. *Trends in Cell Biology*, 9(12):M57–M60, 1999.

[11] P. J. Campbell, G. Getz, J. M. Stuart, J. O. Korbel, L. D. Stein, and ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Net. Pan-cancer analysis of whole genomes. *BioRxiv*, 2017.

[12] P. J. Campbell, S. Yachida, L. J. Mudie, P. J. Stephens, E. D. Pleasance, L. A. Stebbings, L. A. Morsberger, C. Latimer, S. McLaren, *et al.* The patterns and dynamics of genomic instability in metastatic pancreatic cancer. *Nature*, 467(7319):1109, 2010.

[13] A. Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, 13:26–28, 1897.

[14] G. J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16(1):145–159, 1969.

[15] H. Chen, J. M. Bell, N. A. Zavala, H. P. Ji, and N. R. Zhang. Allele-specific copy number profiling by next-generation DNA sequencing. *Nucleic acids research*, 43(4):e23, 2014.

[16] K. Cibulskis, M. S. Lawrence, S. L. Carter, A. Sivachenko, D. Jaffe, C. Sougnez, S. Gabriel, M. Meyerson, E. S. Lander, and G. Getz. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature Biotechnology*, 31(3):213–219, 2013.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[18] IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio?mhq=cplex&mhsrc=ibmsearch_a. Accessed: 2018-11-11.

[19] A. Davis, R. Gao, and N. Navin. Tumor evolution: Linear, branching, neutral or punctuated? *Biochimica et Biophysica Acta – Reviews on Cancer*, 1867(2):151–161, 2017.

[20] A. G. Deshwar. *Reconstructing the evolutionary history of tumours*. Ph.D. thesis, University of Toronto, 2017.

[21] A. G. Deshwar, S. Vembu, C. K. Yung, G. H. Jang, L. Stein, and Q. Morris. PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome Biology*, 16(1):35–55, 2015.

[22] L. Ding, T. J. Ley, D. E. Larson, C. A. Miller, D. C. Koboldt, J. S. Welch, J. K. Ritchey, M. A. Young, T. Lamprecht, *et al.* Clonal evolution in relapsed acute myeloid leukaemia revealed by whole-genome sequencing. *Nature*, 481(7382):506, 2012.

[23] J. C. Dohm, A. E. Minoche, D. Holtgräwe, S. Capella-Gutiérrez, F. Zakrzewski, H. Tafer, O. Rupp, T. R. Sörensen, R. Stracke, *et al.* The genome of the recently domesticated crop plant sugar beet (beta vulgaris). *Nature*, 505(7484):546–549, 2014.

[24] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics*, 31(12):i62–i70, 2015.

[25] M. El-Kebir, G. Satas, L. Oesper, and B. J. Raphael. Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell systems*, 3(1):43–53, 2016.

[26] A. D. Ewing, K. E. Houlahan, Y. Hu, K. Ellrott, C. Caloian, T. N. Yamaguchi, J. C. Bare, C. P'ng, D. Waggott, *et al.* Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection. *Nature Methods*, 12(7):623–630, 2015.

[27] J. S. Farris. Phylogenetic analysis under dollo's law. *Systematic Biology*, 26(1):77–88, 1977.

[28] F. Favero, T. Joshi, A. M. Marquard, N. J. Birkbak, M. Krzystanek, Q. Li, Z. Szallasi, and A. C. Eklund. Sequenza: Allele-specific copy number and mutation profiles from tumor sequencing data. *Annals of Oncology*, 26(1):64–70, 2014.

[29] L. Feuk, A. R. Carson, and S. W. Scherer. Structural variation in the human genome. *Nature Reviews Genetics*, 7(2):85–97, 2006.

[30] A. Fischer, I. Vázquez-García, C. J. Illingworth, and V. Mustonen. High-definition reconstruction of clonal composition in cancer. *Cell Reports*, 7(5):1740–1752, 2014.

[31] C. Gawad, W. Koh, and S. R. Quake. Single-cell genome sequencing: Current state of the science. *Nature Reviews Genetics*, 17(3):175–188, 2016.

[32] M. Gerstung, C. Jolly, I. Leshchiner, S. C. Dentro, S. Gonzalez, T. J. Mitchell, Y. Rubanova, P. Anur, D. Rosebrock, *et al.* The evolutionary history of 2,658 cancers. *bioRxiv*, 2017.

[33] GLPK (GNU linear programming kit). https://www.gnu.org/software/glpk/, 2012. Accessed: 2018-11-11.

[34] S. Goodwin, J. D. McPherson, and W. R. McCombie. Coming of age: Ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016.

[35] M. Greaves. Evolutionary determinants of cancer. *Cancer discovery*, 5(8):806–820, 2015.

[36] P. D. Grünwald, I. J. Myung, and M. A. Pitt. *Advances in minimum description length: Theory and applications*. MIT Press, 2005.

[37] Gurobi optimization. http://www.gurobi.com/. Accessed: 2018-11-11.

[38] D. Gusfield. *ReCombinatorics: The algorithmics of ancestral recombination graphs and explicit phylogenetic networks*. MIT Press, 2014.

[39] G. Ha, A. Roth, J. Khattra, J. Ho, D. Yap, L. M. Prentice, N. Melnyk, A. McPherson, A. Bashashati, *et al.* TITAN: Inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data. *Genome Research*, 24:1881–1893, 2014.

[40] G. Ha, A. Roth, D. Lai, A. Bashashati, J. Ding, R. Goya, R. Giuliany, J. Rosner, A. Oloumi, *et al.* Integrative analysis of genome-wide loss of heterozygosity and monoallelic expression at nucleotide resolution reveals disrupted pathways in triple-negative breast cancer. *Genome Research*, 2012.

[41] I. Hajirasouliha, A. Mahmoody, and B. J. Raphael. A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data. *Bioinformatics*, 30(12):i78–i86, 2014.

[42] T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, vol. 2. Springer Series in Statistics, 2008.

[43] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[44] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[45] K. Jahn, J. Kuipers, and N. Beerenwinkel. Tree inference for single-cell data. *Genome Biology*, 17(1):86, 2016.

[46] Y. Jiang, Y. Qiu, A. J. Minn, and N. R. Zhang. Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing. *Proceedings of the National Academy of Sciences of the U. S. A*, 113(37):E5528–E5537, 2016.

[47] W. Jiao, S. Vembu, A. G. Deshwar, L. Stein, and Q. Morris. Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC Bioinformatics*, 15(1):35–51, 2014.

[48] E. L. Johnson, G. L. Nemhauser, and M. W. Savelsbergh. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing*, 12(1):2–23, 2000.

[49] C. Kandoth, M. D. McLellan, F. Vandin, K. Ye, B. Niu, C. Lu, M. Xie, Q. Zhang, J. F. McMichael, *et al.* Mutational landscape and significance across 12 major cancer types. *Nature*, 502(7471):333–339, 2013.

[50] M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893–903, 1969.

[51] A. G. Knudson. Cancer genetics. *American Journal of Medical Genetics*, 111(1):96–102, 2002.

[52] D. C. Koboldt, Q. Zhang, D. E. Larson, D. Shen, M. D. McLellan, L. Lin, C. A. Miller, E. R. Mardis, L. Ding, and R. K. Wilson. VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Research*, 2012.

[53] A. N. Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of Information Transmission*, 1(1):1–7, 1965.

[54] J. Kuipers, K. Jahn, and N. Beerenwinkel. Advances in understanding tumour evolution through single-cell sequencing. *Biochimica et Biophysica Acta – Reviews on Cancer*, 1867(2):127–138, 2017.

[55] J. Kuipers, K. Jahn, B. J. Raphael, and N. Beerenwinkel. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome Research*, 27(1):1885–1894, 2017.

[56] W. J. Le Quesne. The uniquely evolved character concept and its cladistic application. *Systematic Biology*, 23(4):513–517, 1974.

[57] A. Y. Lee, A. D. Ewing, K. Ellrott, Y. Hu, K. E. Houlahan, J. C. Bare, S. M. G. Espiritu, V. Huang, K. Dang, *et al.* Combining accurate tumor genome simulation with crowdsourcing to benchmark somatic structural variant detection. *Genome Biology*, 19(1):188, 2018.

[58] B. Lewin, L. Cassimeris, V. R. Lingappa, and G. Plopper. *Cells*. Jones and Bartlett, 2007.

[59] H. Li. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993, 2011.

[60] L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology*, 2012, 2012.

[61] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[62] Introduction to lp_solve 5.5.2.5. http://lpsolve.sourceforge.net/5.5/. Accessed: 2018-11-11.

[63] S. Malikic, K. Jahn, J. Kuipers, C. Sahinalp, and N. Beerenwinkel. Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. *bioRxiv*, 2017.

[64] S. Malikic, A. W. McPherson, N. Donmez, and C. S. Sahinalp. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics*, 31(9):1349–1356, 2015.

[65] Y. Marcy, C. Ouverney, E. M. Bik, T. Lösekann, N. Ivanova, H. G. Martin, E. Szeto, D. Platt, P. Hugenholtz, *et al.* Dissecting biological "dark matter" with single-cell genetic analysis of rare and uncultivated TM7 microbes from the human mouth. *Proceedings of the National Academy of Sciences of the U. S. A*, 104(29):11 889–11 894, 2007.

[66] N. McGranahan and C. Swanton. Clonal heterogeneity and tumor evolution: past, present, and the future. *Cell*, 168(4):613–628, 2017.

[67] A. McPherson, A. Roth, E. Laks, T. Masud, A. Bashashati, A. W. Zhang, G. Ha, J. Biele, D. Yap, *et al.* Divergent modes of clonal spread and intraperitoneal mixing in high-grade serous ovarian cancer. *Nature Genetics*, 48(7):758–769, 2016.

[68] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[69] C. A. Miller, B. S. White, N. D. Dees, M. Griffith, J. S. Welch, O. L. Griffith, R. Vij, M. H. Tomasson, T. A. Graubert, *et al.* SciClone: Inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution. *PLOS Computational Biology*, 10(8):e1003 665, 2014.

[70] N. Navin, J. Kendall, J. Troge, P. Andrews, L. Rodgers, J. McIndoo, K. Cook, A. Stepansky, D. Levy, *et al.* Tumour evolution inferred by single-cell sequencing. *Nature*, 472(7341):90, 2011.

[71] N. E. Navin. The first five years of single-cell cancer genomics and beyond. *Genome research*, 25(10):1499–1507, 2015.

[72] S. Nik-Zainal, P. Van Loo, D. C. Wedge, L. B. Alexandrov, C. D. Greenman, K. W. Lau, K. Raine, D. Jones, J. Marshall, *et al.* The life history of 21 breast cancers. *Cell*, 149(5):994–1007, 2012.

[73] N. Niknafs, V. Beleva-Guthrie, D. Q. Naiman, and R. Karchin. Subclonal hierarchy inference from somatic mutations: Automatic reconstruction of cancer evolutionary trees from multi-region next generation sequencing. *PLOS Computational Biology*, 11(10):e1004 416, 2015.

[74] P. C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194(4260):23–28, 1976.

[75] L. Oesper, A. Mahmoody, and B. J. Raphael. THetA: Inferring intra-tumor heterogeneity from high-throughput DNA sequencing data. *Genome Biology*, 14(7):R80, 2013.

[76] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[77] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007.

[78] A. Roth, J. Khattra, D. Yap, A. Wan, E. Laks, J. Biele, G. Ha, S. Aparicio, A. Bouchard-Côté, and S. P. Shah. PyClone: Statistical inference of clonal population structure in cancer. *Nature Methods*, 11(4):396–398, 2014.

[79] A. Salcedo, M. Tarabichi, S. M. G. Espiritu, A. G. Deshwar, M. David, N. M. Wilson, S. Dentro, J. A. Wintersinger, L. Y. Liu, *et al.* Creating standards for evaluating tumour subclonal reconstruction. *BioRxiv*, 2018.

[80] C. T. Saunders, W. S. Wong, S. Swamy, J. Becq, L. J. Murray, and R. K. Cheetham. Strelka: Accurate somatic small-variant calling from sequenced tumor-normal sample pairs. *Bioinformatics*, 28(14):1811–1817, 2012.

[81] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[82] Scitable by Nature Education. SNP. https://www.nature.com/scitable/definition/single-nucleotide-polymorphism-snp-295, 2014. Accessed: 2018-11-11.

[83] R. J. Solomonoff. A formal theory of inductive inference, part 1 and part 2. *Information and Control*, 7:1–22, 224–254, 1964.

[84] M. R. Stratton, P. J. Campbell, and P. A. Futreal. The cancer genome. *Nature*, 458(7239):719–724, 2009.

[85] F. Strino, F. Parisi, M. Micsinai, and Y. Kluger. TrAp: a tree approach for fingerprinting subclonal tumor composition. *Nucleic Acids Research*, 41(17):e165, 2013.

[86] R. Xi, A. G. Hadjipanayis, L. J. Luquette, T.-M. Kim, E. Lee, J. Zhang, M. D. Johnson, D. M. Muzny, D. A. Wheeler, *et al.* Copy number variation detection in

whole-genome sequencing data using the bayesian information criterion. *Proceedings of the National Academy of Sciences of the U. S. A*, 108(46):E1128–E1136, 2011.

[87] C. Xu. A review of somatic single nucleotide variant calling algorithms for next-generation sequencing data. *Computational and Structural Biotechnology Journal*, 16:15–24, 2018.

[88] H. Zare, J. Wang, A. Hu, K. Weber, J. Smith, D. Nickerson, C. Song, D. Witten, C. A. Blau, and W. S. Noble. Inferring clonal composition from multiple sections of a breast cancer. *PLOS Computational Biology*, 10(7):e1003 703, 2014.

Appendix A

Onctopus Software

Input parameters of Onctopus, together with a short explanation and default values are shown in Table A.1.

Table A.1.: Input parameters of Onctopus.

parameter name	explanation	default value
`lin_num`	lineage number K	-
`segments`	file with average allele-specific copy numbers with standard errors of I genome segments	-
`ssms`	file with variant and reference counts of J SSMs	-
`time`	maximal optimization time, in seconds	10^{75}
`threads`	number of threads used for the optimization	1
`treememory`	maximal memory for the optimization, in MB	10^{75}
`fixed_variables`	files with values of variables, which should be fixed	-
`start_variables`	files with values of variables, which should be used as optimization start	-

Table A.1.: Input parameters of Onctopus. Continued.

parameter name	explanation	default value
`spline_points`	number of knots n_{knots} used to approximate piecewise linear functions of objective functions	50
`c_num_max`	maximal number of copy number changes per segment	2
`cluster_SSM`	whether SSMs should get clustered	False
`super_SSMs`	whether clustering version 2 should be used	False
`cluster_algo`	clustering algorithm that should be used	-
`cluster_num_param`	number of clusters, gets multiplied with lineage number	-
`normal_seg_ind`	indices of CNA-free segments, segments get restricted to contain zero copy number changes	-
`combine_segments`	whether indicated CNA-free segments should be combined to a *super segment*	False
`overdispersion`	overdispersion parameter for beta-binomial distribution	1000

Appendix B

Data Simulation

B.1. Data Simulation

Table B.1 shows important parameters of the simulation, their explanation and default values.

B.2. Simulated Datasets for Analyzing Optimality, Run Time and Memory Usage

We created five different trees for each lineage number, linear and branching ones (see Figure B.1).

Parameters used for the data simulation, others than the number of lineages, SSMs and segments, can be found in Table B.2. Parameters that are not shown have default values as indicated in Table B.1.

B.3. Simulated Datasets for Simple Somatic Mutation Clustering Analysis

B.3.1. Clustering Algorithms and Cluster Numbers

We created one phylogenetic tree for each lineage number and for each copy number change number, we created a specific copy number change assignment per tree (see Figure B.2).

Table B.1.: Simulation parameters.

parameter name	explanation	default value
`coverage_overdispersion` (s_{COV})	used to create the two needed parameters for the negative binomial distribution from the coverage	1000
`frequency_overdispersion` (s_{SSM})	used to create the two needed parameters for the beta-binomial distribution from the VAF	1000
`p1_A_prop`	probability with which a copy number gain is assigned to allele A	0.2
`p1_A_B_prop`	probability with which a copy number gain is assigned to allele A and allele B	0.2
`m1_B_prop`	probability with which a copy number loss is assigned to allele B	0.2
`m1_A_B_prop`	probability with which a copy number loss is assigned to allele A and allele B	0.2
`p1_m1_prop`	probability with which a copy number gain is assigned to allele A and a loss to allele B	0.2
`CNAs_mult_lin_prop`	probability with which two copy number changes are samples within a segment and assigned to two lineages	0.0
`SSM_before_CNV_LH`	probability with which an SSM appears before a copy number gain that is assigned to the same allele, segment and lineage than the SSM	0.5
`seg_min_length`	minimal length of a segment	$1,000,000$

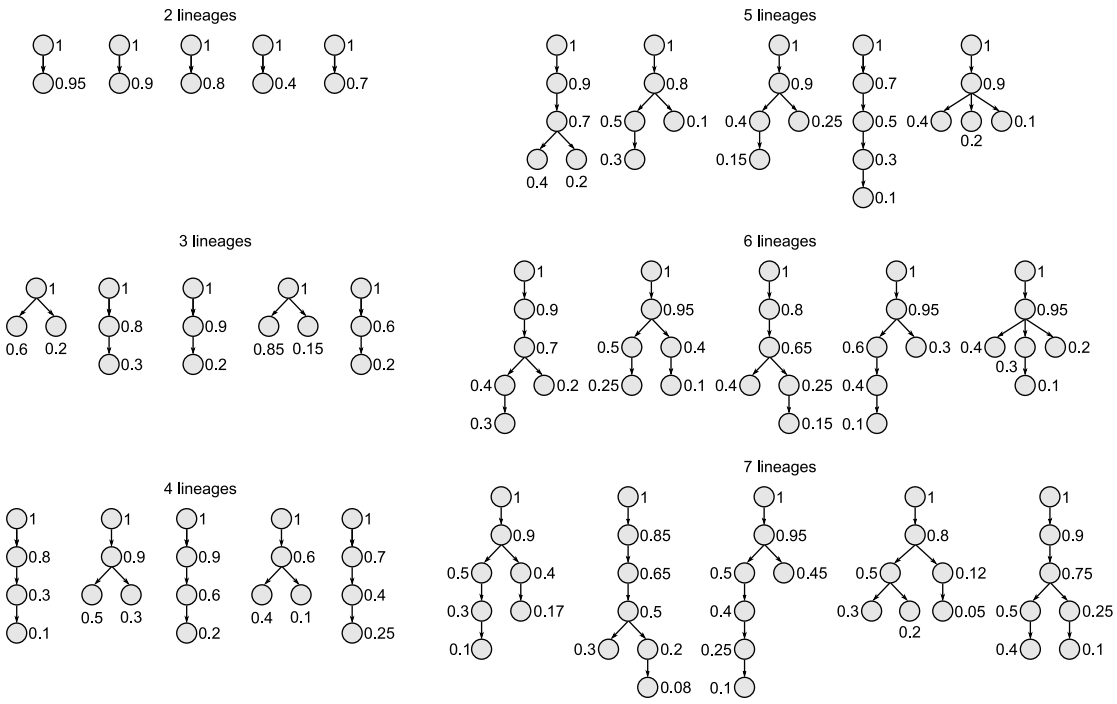


Figure B.1.: Phylogenetic trees with lineage frequencies for simulating datasets to analyze optimality, run time and memory usage of Onctopus runs.

Table B.2.: Parameters for simulating datasets to analyze optimality, run time and memory usage of Onctopus runs.
K: lineage number

parameter	parameter value
haploid coverage	100
CNA noise	no
SSM lineage assignment probabilities	all equal with $\frac{1}{K-1}$
SSM noise	yes

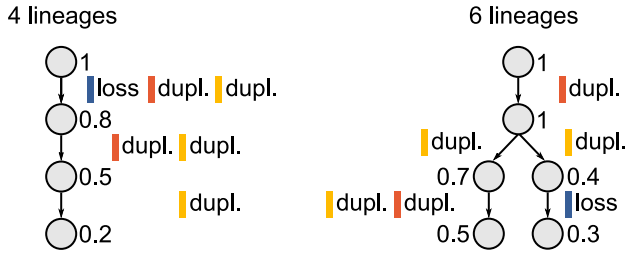


Figure B.2.: Phylogenetic trees with lineage frequencies and with copy number change assignments for data simulation of experiment to investigate the ability of different clustering algorithms and numbers of clusters.
Three different copy number change assignments per phylogeny are shown, indicated by the different colors of the alleles.
dupl.: duplication

Table B.3.: Parameters for data simulation of experiment to investigate the ability of different clustering algorithms and numbers of clusters to cluster SSMs based on their VAFs.

parameter	parameter value
haploid coverage	100
CNA noise	no
SSM noise	yes

Parameters used for the data simulation, others than the number of lineages, copy number changes and SSMs and different SSM assignment strategies, are shown in Table B.3. Parameters not present in the table have default values as indicated in Table B.1.

B.3.2. Building Subclonal Reconstructions with Clustered Simple Somatic Mutations

We created one phylogenetic tree for four lineages (see Figure B.3).

Used parameters for the simulation, others than the numbers of lineages, segments, copy number changes and SSMs, as well as different SSM assignment strategies, can be found in Table B.4. Parameters not shown have default values as indicated in Table B.1.

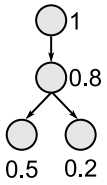


Figure B.3.: Phylogenetic tree with lineage frequencies for data simulation to analyze Onctopus' performance with and without clustered SSMs.

Table B.4.: Parameters for dataset simulation to analyze Onctopus' performance with and without clustered SSMs.

parameter	parameter value
haploid coverage	100
CNA noise	no
SSM noise	yes

B.4. Simulated Datasets for Fixing Copy Number Aberration Analysis

We created five phylogenetic trees for four lineages (see Figure B.4).

Used parameters for the data simulation, others than numbers of lineages, segments, segments with copy number changes and SSMs, can be found in Table B.5. Parameters not shown have default values as indicated in Table B.1.

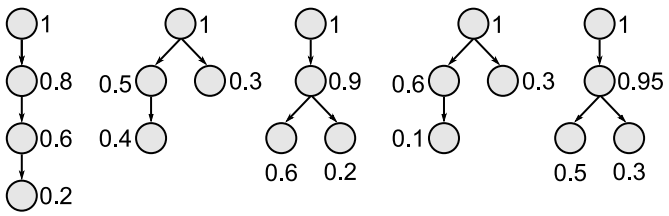


Figure B.4.: Phylogenetic trees with lineage frequencies for data simulation for CNA fixation experiment.

Table B.5.: Parameters for dataset simulation for CNA fixation experiment.
K: lineage number

parameter	parameter value
haploid coverage	100
CNA noise	no
SSM lineage assignment probabilities	all equal with $\frac{1}{K-1}$
SSM noise	yes

Table B.6.: Further parameters for data simulation for inferring lineage frequencies depending on the number of lineages.
K: lineage number

parameter	parameter value
haploid coverage	100
SSM lineage assignment probabilities	all equal with $\frac{1}{K-1}$
SSM noise	yes

B.5. Simulated Datasets for Fixing Lineage Frequencies Analysis

B.5.1. Simulated Datasets for Inference of Lineage Frequencies Depending on the Number of Simple Somatic Mutations

We used the phylogenetic trees of the general experiment to investigate optimality, run time and memory usage (see Figure B.1) for this experiment.

Parameters for the data simulation, others than numbers of lineages, segments and SSMs, are shown in Table B.6. Parameters not present have default values as indicated in Table B.1.

B.5.2. Simulated Datasets for Analysis of Performance with Inferred Lineage Frequencies

We used the phylogenetic trees for four and six lineages of the general experiment to investigate optimality, run time and memory usage (see Figure B.1) for this experiment.

The parameters of the data simulation, others than the numbers of lineages, segments, SSMs and copy number changes, are presented in Table B.7. Parameters not shown have default values as indicated in Table B.1.

Table B.7.: Further parameters for data simulation for analysis of performance with inferred lineage frequencies.
K: lineage number

parameter	parameter value
haploid coverage	100
CNA noise	no
SSM lineage assignment probabilities	all equal with $\frac{1}{K-1}$
SSM noise	yes

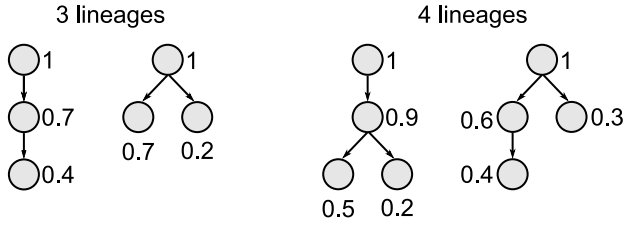


Figure B.5.: Phylogenetic trees with lineage frequencies for data simulation for investigating impact of approximating VAFs..

B.6. Simulated Datasets for Analysis of Approximating Variant Allele Frequencies in Mixed Integer Linear Program

We created two phylogenetic trees for three and four lineages (see Figure B.4).

The parameters of the data simulation, beyond those described in Section 5.8, are presented in Table B.8. Parameters not shown have default values as indicated in Table B.1.

Table B.8.: Further parameters for data simulation for analyzing impact of approximating VAFs.

parameter	parameter value
haploid coverage	100
SSM noise	yes

Table B.9.: CNA and SSM sampling parameters for data simulation for comparison between Onctopus, PhyloWGS and Canopy.

parameter	parameter value
CNA noise	yes
SSM lineage assignment probabilities	all equal with $\frac{1}{K-1}$
SSM noise	yes

B.7. Simulated Datasets for Comparison between Onctopus, PhyloWGS and Canopy

To simulate the datasets, we used the first three phylogenetic trees for two, four and six lineages of the general experiment where we investigated optimality, run time and memory usage (see Figure B.1).

Parameters of the CNA and SSM sampling are shown in Table B.9. Parameters not described in this section and Subsection 6.2.1 have default values as indicated in Table B.1.