Reconstructing Rearrangement Phylogenies of Natural Genomes

Leonard Bohnenkämper
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Jens Stoye
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Daniel Dörr
Department for Endocrinology and Diabetology, Medical Faculty and University Hospital Düsseldorf, Heinrich Heine University Düsseldorf, Germany
German Diabetes Center (DDZ), Leibniz Institute for Diabetes Research Germany, and Center for Digital Medicine, Heinrich Heine University Düsseldorf, Germany

Abstract

We study the classical problem of inferring ancestral genomes from a set of extant genomes under a given phylogeny, known as the *Small Parsimony Problem* (SPP). Genomes are represented as sequences of oriented markers, organized in one or more linear or circular chromosomes. Any marker may appear in several copies, without restriction on orientation or genomic location, known as the *natural genomes* model. Evolutionary events along the branches of the phylogeny encompass large scale rearrangements, including segmental inversions, translocations, gain and loss (DCJ-indel model).

Even under simpler rearrangement models, such as the classical breakpoint model without duplicates, the SPP is computationally intractable. Nevertheless, the SPP for natural genomes under the DCJ-indel model has been studied recently, with limited success. Here, we improve on that earlier work, giving a highly optimized ILP that is able to solve the SPP for sufficiently small phylogenies and gene families. A notable improvement w.r.t. the previous result is an optimized way of handling both circular and linear chromosomes. This is especially relevant to the SPP, since the chromosomal structure of ancestral genomes is unknown and the solution space for this chromosomal structure is typically large.

We benchmark our method on simulated and real data. On simulated phylogenies we observe a considerable performance improvement on problems that include linear chromosomes. And even when the ground truth contains only one circular chromosome per genome, our method outperforms its predecessor due to its optimized handling of the solution space. The practical advantage becomes also visible in an analysis of seven *Anopheles* taxa.

2012 ACM Subject Classification Applied computing → Bioinformatics; Theory of computation → Integer programming

Keywords and phrases genome rearrangement, ancestral reconstruction, small parsimony, integer linear programming, double-cut-and-join

Digital Object Identifier 10.4230/LIPIcs.WABI.2024.12

Supplementary Material *Software (Source Code)*: `https://github.com/marschall-lab/spp_dcj_v2` [4], archived at `swh:1:dir:9f96ced9254d812c0c0cd34376094007bc578a63`

Acknowledgements LB thanks Luca Parmigiani for helping with some C++ issues at a critical moment. DD thanks Cedric Chauve for providing the Anopheles dataset.

1 Introduction

The *Small Parsimony Problem* (SPP) is a general optimization problem in phylogenetics that aims at annotating the internal vertices of a given phylogenetic tree $T = (V, E)$ whose leaves are already annotated, such that the total *tree distance* $d_T = \sum_{(A,B)\in E} d(A, B)$ is minimized. Here, $d(A, B)$ is a function returning the distance between the annotations of any two vertices A and B of the phylogenetic tree. Traditional tree annotations may be DNA or protein sequences, while more recently, with the emergence of phylogenomic studies, also complete genomes, often in form of so-called *marker sequences* may be used.

Distance functions for marker sequences usually consider rearrangements and content-modifying operations on the elements of the sequences. A useful general-purpose distance in genome rearrangement is based on the *DCJ-indel* model. Conceived by Braga et al. [5] as an extension of the Double-Cut-and-Join model by Yancopoulos et al. [14], operations in the DCJ-indel model are either genomic rearrangements, modeled by a double cut and subsequent joining of the so created ends (*DCJ*), or segmental gains and losses of arbitrary length (*indels*).

When each marker occurs not more than once per genome, calculating the DCJ-indel distance between two genomes is polynomial [5]. However, on genomes with unrestricted distributions of markers, also called *natural genomes*, calculating the DCJ-indel distance is NP-hard. Nonetheless, efficient ILP solutions exist, such as *ding* [3].

The first attempt to generalize this method from the pairwise genomic distance to the phylogenomic SPP under the DCJ-indel model was an ILP by Doerr and Chauve [8], called *SPP-DCJ*. They did so by solving a generalized problem, in which – as a result of some pre-processing – adjacencies in ancestral genomes could be absent or present, and in the latter case they may be assigned a weight that would be taken into consideration during optimization. One major issue in this generalization was ding's use of *caps*, telomeric markers that need to be matched during optimization and for which the solution space is superexponential [12]. Doerr and Chauve went to great lengths to limit the effect of this additional solution space, but were ultimately not able to completely remove it from their solution.

The ILP solution presented in this manuscript combines a recent reformulation of the DCJ-indel model that allows one to forego the matching of caps [2] with the basic modeling of SPP pioneered by SPP-DCJ. We additionally resolve another issue described in [8], which is the dependence of SPP-DCJ on previously known candidates for circular singletons, for each of which SPP-DCJ creates a number of constraints and variables. Since the number of circular singleton candidates is exponential, the size of SPP-DCJ is exponential as well. While this problem may be less relevant when given few, refined candidate adjacencies for ancestors, our ILP is the first to solve the SPP for natural genomes under the DCJ-indel model while remaining of polynomial size w.r.t. any input data.

The remainder of the manuscript is organized as follows. In Section 2, we give basic definitions and previous results needed to derive our algorithm. In Section 3, we explain the fundamental features of our method (Subsections 3.1 and 3.2) before presenting the ILP in Subsection 3.3. We evaluate the performance of our method in Section 4 and discuss our overall findings in Section 5.

2 Preliminaries

For the purposes of this work, we use the abstraction to describe genomes as sequences of oriented markers. A *(genomic) marker* $g = (g^{\mathrm{t}}, g^{\mathrm{h}})$ is a universally unique entity consisting of *marker extremities* tail of g, denoted by g^{t}, and head of g, denoted by g^{h}.

The structure of a genome can be described via its adjacencies. An adjacency $\{f^x, g^y\}$ (with $x, y \in \{\mathrm{t}, \mathrm{h}\}$) describes that markers f and g are neighbors on the same chromosome and oriented, such that extremities f^x and g^y are adjacent. For ease of notation we also write $f^x g^y$ for an adjacency. Note that adjacencies can be read in either direction, i.e. $g^y f^x$ is the same as $f^x g^y$.

For the sake of a simpler formulation of the theory, we aim for each extremity to be part of some adjacency. In order to accomplish this, we use additional extremities modeling the ends of linear chromosomes, called *telomeres*. A *telomere* t° is a universally unique entity encompassing a single telomeric extremity denoted by "$\circ$". A genome can then be described as a graph as follows.

▶ **Definition 1.** *A* genome $\mathbb{A}$ *is a graph with vertices* $\mathcal{E}(\mathbb{A}) \cup \mathcal{T}(\mathbb{A})$*, namely its marker extremities* $\mathcal{E}(\mathbb{A})$ *and telomeric extremities* $\mathcal{T}(\mathbb{A})$*. The set of edges is* $\mathcal{M}(\mathbb{A}) \cup \mathcal{A}(\mathbb{A})$*, namely its marker edges* $\mathcal{M}(\mathbb{A})$ *and adjacency edges* $\mathcal{A}(\mathbb{A})$*. This graph fulfills the following properties:*
1. $\mathcal{M}(\mathbb{A})$ *is a perfect matching on* $\mathcal{E}(\mathbb{A})$ *with* $\mathcal{M}(\mathbb{A}) = \{\{m^t, m^h\} \mid \forall m^t, m^h \in \mathcal{E}(\mathbb{A})\}$*, and*
2. $\mathcal{A}(\mathbb{A})$ *is a perfect matching on* $\mathcal{E}(\mathbb{A}) \cup \mathcal{T}(\mathbb{A})$*.*

An example of a genome is given in Figure 1.

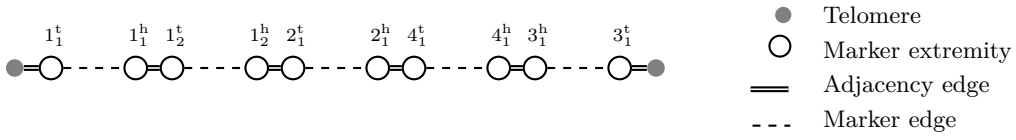

Figure 1 A genome of five markers 1_1, 1_2, 2_1, 3_1, 4_1 on a single linear chromosome.

Because each marker is universally unique, in order to compare genomes we need to establish which markers are homologous. We model homology as an equivalence relation ($\equiv$), that is $m_a \equiv m_b$ for some markers $m_a \in \mathcal{M}(\mathbb{A})$, $m_b \in \mathcal{M}(\mathbb{B})$ and genomes $\mathbb{A}, \mathbb{B}$. Note that this includes the case $\mathbb{A} = \mathbb{B}$, i.e. there can be homologous markers in the same genome (in-paralogs). The equivalence class of a marker m, denoted by $[m]$, is called its *family*. If a marker m exists in $\mathbb{A}$, but has no equivalent in $\mathbb{B}$ or vice versa, we refer to m as *singular* w.r.t. $\mathbb{A}, \mathbb{B}$.

Given the equivalence relation on markers, one can easily derive an equivalence relation on extremities, namely $m_a^{\mathrm{t}} \equiv m_b^{\mathrm{t}}$ and $m_a^{\mathrm{h}} \equiv m_b^{\mathrm{h}}$ if and only if $m_a \equiv m_b$. For this derived equivalence we have $m_a^{\mathrm{h}} \not\equiv m_b^{\mathrm{t}}$ for all m_a, m_b. We call extremities *singular* if and only if their corresponding marker is singular. One can visualize such an equivalence relation for two genomes $\mathbb{A}, \mathbb{B}$ using the Capping-Free Multi-Relational Diagram:

▶ **Definition 2.** *Given two genomes* $\mathbb{A}, \mathbb{B}$ *and a homology* ($\equiv$)*, the* Capping-Free Multi-Relational Diagram (CFMRD) *is a graph* $\mathcal{CFMRD}(\mathbb{A}, \mathbb{B}, \equiv) = (\mathcal{E} \cup \mathcal{T}, E_{adj} \cup E_{self} \cup E_{ext})$ *with* $\mathcal{E} = \mathcal{E}(\mathbb{A}) \cup \mathcal{E}(\mathbb{B})$*,* $\mathcal{T} = \mathcal{T}(\mathbb{A}) \cup \mathcal{T}(\mathbb{B})$*, adjacency edges* $E_{adj} = \mathcal{A}(\mathbb{A}) \cup \mathcal{A}(\mathbb{B})$*, self edges* $E_{self} = \{m \in \mathcal{M}(\mathbb{A}) \cup \mathcal{M}(\mathbb{B}) \mid m$ *singular w.r.t.* $\mathbb{A}, \mathbb{B}\}$ *and extremity edges* $E_{ext} = \{\{u, v\} \mid u \in \mathcal{E}(\mathbb{A}), v \in \mathcal{E}(\mathbb{B}), u \equiv v\}$*.*

An example of a CFMRD is given in Figure 2.

An established way to compare two genomes on a structural level is the *rearrangement distance*. The rearrangement distance of two genomes $\mathbb{A}, \mathbb{B}$ is defined as the minimum number of operations needed to transform $\mathbb{A}$ into $\mathbb{B}$ with operations restricted to a certain model (such as DCJ-indel). When ($\equiv$) maps each marker of genome $\mathbb{A}$ to at most one marker of

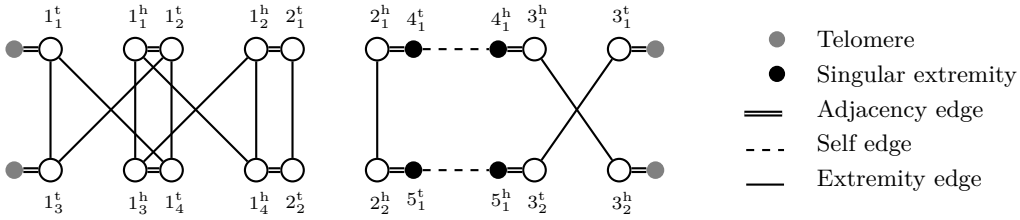

Figure 2 Capping-Free Multi-Relational Diagram for two genomes on an unresolved homology ($\equiv_1$) with families $\{1_1, 1_2, 1_3, 1_4\}$, $\{2_1, 2_2\}, \{3_1, 3_2\}, \{4_1\}, \{5_1\}$.

genome $\mathbb{B}$, calculating the rearrangement distance between $\mathbb{A}$ and $\mathbb{B}$ is typically easy. We refer to such a homology as *resolved*. More formally, a homology is resolved if for each genome $\mathbb{A}$ and marker $m \in \mathcal{M}(\mathbb{A})$ the family of m contains only itself, i.e. $[m] \cap \mathcal{M}(\mathbb{A}) = \{m\}$. On these homologies, $\mathcal{CFMRD}(\mathbb{A}, \mathbb{B}, \equiv)$ consists only of simple cycles and simple paths. An example of a CFMRD on a resolved homology is shown in Figure 3.

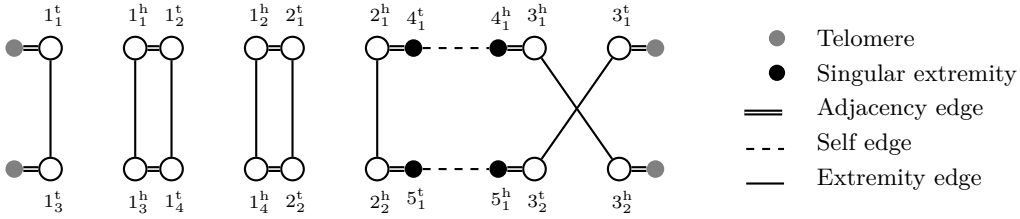

Figure 3 CFMRD for the two genomes of Figure 2 on a resolved homology ($\equiv_2$) with families $\{1_1, 1_3\}$, $\{1_2, 1_4\}$, $\{2_1, 2_2\}$, $\{3_1, 3_2\}$, $\{4_1\}$, $\{5_1\}$. Note that ($\equiv_2$) is a matching on ($\equiv_1$).

With a resolved homology, the DCJ-indel distance can be calculated easily by just counting different types of components in the CFMRD. For the purpose of this counting, we ignore self edges. We write c for the number of cycles and p_{ab} (resp. p_{aa}, resp. p_{bb}) for the number of paths that start in $\mathbb{A}$ and end in $\mathbb{B}$ (resp. start in $\mathbb{A}$ and end in $\mathbb{A}$, resp. start in $\mathbb{B}$ and end in $\mathbb{B}$). Since the graph is undirected, we canonize their labels by reading paths from $\mathbb{A}$ to $\mathbb{B}$. When the vertex the path starts or ends in is a telomere of $\mathbb{A}$ (resp. $\mathbb{B}$), we write A (resp. B) in uppercase. When the path ends because the only way to continue it would be a self edge (note that we ignore self edges here), we write a (resp. b) in lowercase. When a path starts and ends in the same genome, we read it from telomere to singular extremity (note that in all other cases, the label is symmetric).

For example, the CFMRD of Figure 3 has $c = 2$, $p_{AB} = 1$ (path $t^\circ, 1_1^{\mathrm{t}}, 1_3^{\mathrm{t}}, t^\circ$), $p_{ab} = 1$ (path $4_1^{\mathrm{t}}, 2_1^{\mathrm{h}}, 2_2^{\mathrm{h}}, 5_1^{\mathrm{t}}$), $p_{aB} = 1$ (path $4_1^{\mathrm{h}}, 3_1^{\mathrm{h}}, 3_2^{\mathrm{h}}, t^\circ$) and $p_{Ab} = 1$ (path $t^\circ, 3_1^{\mathrm{t}}, 3_2^{\mathrm{t}}, 5_1^{\mathrm{h}}$).

There is one case, in which we need to consider self edges, namely *circular singletons*. Circular singletons are cycles that consist only of adjacency and self edges. We denote their number by s. For a more in-depth explanation of these terms, the interested reader is referred to [2]. Using these terms, the following formula can be used.

▶ **Theorem 3** (adapted from [2]). *For two genomes $\mathbb{A}, \mathbb{B}$ and a resolved homology ($\overset{\star}{\equiv}$), the DCJ-indel distance is*

$$\bar{d}_{\mathrm{DCJ-ID}}(\mathbb{A}, \mathbb{B}, \overset{\star}{\equiv}) = n - c + \left\lceil \frac{p_{ab} + \max(p_{Aa}, p_{aB}) + \max(p_{Ab}, p_{Bb}) - p_{AB}}{2} \right\rceil + s$$

with n the number of matched markers, $n = |\{(m_a, m_b) \in \mathcal{M}(\mathbb{A}) \times \mathcal{M}(\mathbb{B}) \mid m_a \overset{\star}{\equiv} m_b\}|$.

This formula holds because it is equivalent to previously proven distance formulas under the DCJ-indel model, however it can also be derived independently. Details are explained in [2]. To paraphrase the results there, it is shown that two genomes are equal if and only if their CFMRD consists of only c cycles and p_{AB} paths between telomeres of both genomes with $n = c + \frac{p_{AB}}{2}$. Additionally, for each DCJ or indel operation the formula of Theorem 3 changes by at most 1. These two facts combined yield the formula as a lower bound. Additionally [2] contains an algorithm transforming $\mathbb{A}$ into $\mathbb{B}$ using DCJ and indel operations that is able to reach this lower bound, proving it is a formula for the rearrangement distance under the DCJ-indel model.

When the homology is not resolved, we need to refine the homology to be resolved. We call such a refinement a *matching*. More formally, a matching $(\overset{\star}{\equiv})$ on $(\equiv)$ is a resolved homology, such that $m_a \overset{\star}{\equiv} m_b \implies m_a \equiv m_b$.

Since allowing for arbitrary matchings can lead to an excess of indels in the sorting scenario, we restrict ourselves to the maximum matching model. A matching $(\overset{+}{\equiv})$ is *maximum* w.r.t. $\mathbb{A}, \mathbb{B}$ if a maximum amount of markers in $\mathbb{A}$ has a homolog in $\mathbb{B}$ and vice versa.

▶ **Definition 4.** *Given homology $(\equiv)$, the DCJ-indel distance between $\mathbb{A}$ and $\mathbb{B}$ under the maximum matching model is*

$$d_{\mathrm{DCJ-ID}}(\mathbb{A}, \mathbb{B}, \equiv) = \min_{(\overset{+}{\equiv}) \text{ maximum matching on } (\equiv)} \bar{d}_{\mathrm{DCJ-ID}}(\mathbb{A}, \mathbb{B}, \overset{+}{\equiv}).$$

When reconstructing a phylogeny, only extant genomes are known, that is, there is no definitive information about the adjacencies at the inner nodes. In order to capture this uncertainty, a typical approach is to generate a large set of candidate adjacencies at each inner node that very likely will include the correct ones. Such a set can be viewed as a *degenerate genome*, which however may contain multiple conflicting adjacencies, such as ab and ac with $b \neq c$. (In a normal genome this cannot occur, as the matching requirement ensures that there is only one adjacency that involves a.) More formally, a degenerate genome $\mathbb{D}$ is a graph $(\mathcal{E}(\mathbb{D}) \cup \mathcal{T}(\mathbb{D}), \mathcal{M}(\mathbb{D}) \cup \mathcal{A}(\mathbb{D}))$ that fulfills only Property 1 of Definition 1.

All possible ancestors at a certain node in the phylogeny are then built from disambiguations of these conflicting adjacencies. We call these possible ancestors *linearizations*. A linearization of a degenerate genome $\mathbb{D}$ is a genome $\mathbb{A}$, such that $\mathcal{E}(\mathbb{A}) = \mathcal{E}(\mathbb{D})$, $\mathcal{T}(\mathbb{A}) \subseteq \mathcal{T}(\mathbb{D})$, $\mathcal{M}(\mathbb{A}) = \mathcal{M}(\mathbb{D})$ and $\mathcal{A}(\mathbb{A}) \subseteq \mathcal{A}(\mathbb{D})$. If such a linearization exists, we call $\mathbb{D}$ *linearizable*. We give an example of a linearizable degenerate genome and one of its linearizations in Figure 4. Note that each genome is also a degenerate genome with precisely one linearization, namely itself.

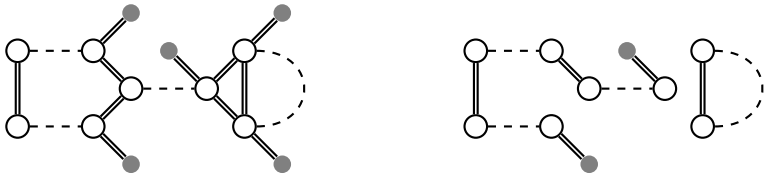

Figure 4 Left: A degenerate genome. Right: A linearization of it.

We can then formulate the problem we are considering in this paper as finding linearizations of all (degenerate) genomes in the phylogeny, such that the sum of all DCJ-indel distances in the tree is minimized. Optionally, we also allow to put weights on the adjacencies and take these into account during the minimization.

▶ **Problem 5** (Weighted Small Parsimony Linearization Problem). *Given a phylogeny $T = (V, E)$, a homology ($\equiv$), a weighting function w for adjacencies, and a parameter $\alpha \in [0, 1]$, find a linearization $\mathbb{L}_i$ for each (degenerate) genome $\mathbb{D}_i$ in T, such that*

$$\sum_{(\mathbb{D}_i,\mathbb{D}_k)\in E} \left(\alpha\, d_{\mathrm{DCJ-ID}}(\mathbb{L}_i, \mathbb{L}_k, \equiv) \;+\; (\alpha - 1) \sum_{ab \in \mathcal{A}(\mathbb{L}_i)\cup\mathcal{A}(\mathbb{L}_k)} w(ab) \right) \tag{1}$$

is minimized.

Because the pairwise comparison of (non-degenerate) natural genomes is already NP-hard, the Weighted Small Parsimony Linearization Problem is NP-hard as well. Doerr and Chauve's algorithm SPP-DCJ, which solves Problem 5, is therefore formulated as an ILP. Thus, we formulate our improved algorithm in Section 3.3 as an ILP as well.

3 A new method

3.1 Capping-free model

The previous solution by Doerr and Chauve [8] was based on a different graph structure, namely the Capped Multi-Relational Diagram (CMRD). The CMRD differs from the CFMRD in the way it treats telomeres. In the CMRD of two genomes $\mathbb{A}$ and $\mathbb{B}$ there exist additional extremity edges between each telomere of $\mathbb{A}$ and each telomere of $\mathbb{B}$, leading to additional $|\mathcal{T}(\mathbb{A})| \cdot |\mathcal{T}(\mathbb{B})|$ extremity edges.

When calculating the DCJ-indel distance using the CMRD, one must not only determine the resolved homology, but also a matching between telomeres, that is, on $\mathcal{T}(\mathbb{A}) \times \mathcal{T}(\mathbb{B})$. As identified in [12], this leads to a superexponential increase of the solution space. As our new method is based on the CFMRD, we can use the formula of Theorem 3 and thus avoid such an increase in the solution space.

3.2 Ensuring linearizability

It is vital for our method that the degenerate genomes in the phylogeny are linearizable (see Problem 5). However, not all degenerate genomes are linearizable (see Figure 5). Moreover, not all methods used to infer candidate adjacencies for ancestors guarantee this requirement. In particular DeCoSTAR [9], a method for inferring ancestral genomes that is integrated together with SPP-DCJ into a larger reconstruction workflow detailed in [6], generates conflicting ancestral adjacencies. Yet, testing for linearizability is a computationally challenging problem of unknown complexity.

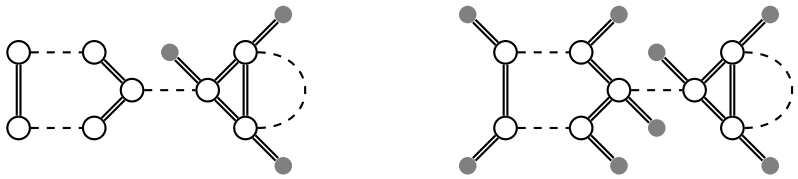

Figure 5 Left: This degenerate genome is not linearizable because of missing telomeres. Right: The genome becomes linearizable when adding telomeres. One linearization is that of Figure 4.

While we cannot test for linearizability, we can modify the given degenerate genomes by adding telomeres, such that they become linearizable. We offer two different modes, which we detail in the following subsections.

3.2.1 Local guarantees

We cannot generally decide, whether a degenerate genome is linearizable. Still, some edge cases are simple to solve:

▶ **Lemma 6.** *A perfect matching $M \subseteq \mathcal{A}(\mathbb{D})$ in a degenerate genome $\mathbb{D} = (\mathcal{E}(\mathbb{D}) \cup \mathcal{T}(\mathbb{D}), \mathcal{M}(\mathbb{D}) \cup \mathcal{A}(\mathbb{D}))$ corresponds to a linearization of $\mathbb{D}$.*

Proof. Observe that in the M-induced degenerate genome $\mathbb{D}' = (\mathcal{E}(\mathbb{D}) \cup \mathcal{T}(\mathbb{D}), \mathcal{M}(\mathbb{D}) \cup M)$ each node is incident to exactly one adjacency edge. Further each connected component corresponds to a linear component where both degree-one nodes correspond to telomeres, or a circular component where each node corresponds to a marker extremity. ◀

However, the converse is not true: Since not all telomeric extremities must be covered, $\mathbb{D}$ may still be linearizable even if no perfect matching may be derived from $\mathbb{D}$.

In the earlier version of SPP-DCJ [8], a simple approach was introduced that complements each degenerate genome $\mathbb{D}$ with additional telomeres and telomeric adjacencies to ensure linearizability. To this end, $\mathbb{D}$ is decomposed into connected components that are independently tested. If the size of a component, i.e., the number of its vertices, is even, and it is either linear, circular, or fully connected, then it is considered as *locally linearizable*. Otherwise, each extremity e of the component is complemented with a telomere t_e, and a telomeric adjacency $\{e, t_e\}$ is added to the degenerate genome, ensuring that it is linearizable as a whole.

3.2.2 Allowing each extremity to be connected to a telomere

Given the uncertainty about inferred ancestral adjacencies, even when a component is locally linearizable, individual adjacencies of that component might still be wrongly inferred by the preprocessing and thus might be erroneously included in the linearization, simply because otherwise a linearization might not be possible.

In order to prevent this behavior, we offer a mode in which *each* extremity is connected to an (artificially introduced) telomere to reflect this uncertainty. In contrast to the method described above, we do this even in components with local guarantees.

This approach was previously practically unsound because of inefficient handling of telomeres. Now it may become the standard mode of operation, as it allows to find reasonable solutions in case of noisy input data, while the computational overhead introduced by the addition of the artificial telomeres remains moderate. We refer to this mode as the *safer linearization mode* in subsequent sections.

3.3 A new ILP formulation

Algorithm 1 gives an overview of our method with additional tables detailing some of the parts of the ILP.

In principle, our algorithm solves Problem 5 in the same way as SPP-DCJ [8], namely it determines linearizations while simultaneously computing the distances between nodes in the phylogeny with the objective of minimizing the total distance. However, for ease of readability, we separate the linearization and distance computation into two different subsections.

On the *global level*, the linearizations $\mathbb{L}_i$ are derived for each (degenerate) genome $\mathbb{D}_i$. On the *local level*, the resulting linearizations are compared to each other along the branches of the phylogeny. Each branch gives rise to a pairwise comparison by means of the CFMRD. In doing so, the selection of adjacencies of a derived genome is propagated from across CFMRDs, thus ensuring global consistency.

The main differences between our algorithm and that in [8] are found in the local level, as this is where the CFMRD plays a role.

3.3.1 Global level

The global level deals with the setting of adjacencies or telomeres of (ancestral) genomes. For each (marker or telomeric) extremity v, we determine its presence or absence with a binary variable $\mathtt{g}_v$. We require each marker extremity to occur – but not each telomere, that may or may not occur (see Constraint `C.01`). Each extremity is required to be part of exactly one (possibly telomeric) adjacency (`C.02`), which ensures a properly linearized genome.

3.3.2 Local level

The local level deals with each edge of the tree separately, making use of the CFMRD of the corresponding genome pair. Since this part is entirely local to the edge in question, we presume that each vertex v_i of the CFMRD has a unique identifier among all other CFMRDs, making all its variables globally unique. In order to limit the range of the general variable y_{v_i}, we also assign each vertex a rank i that is local and unique only within the specific CFMRD. We map each extremity to its identifier for the global level by the function γ.

In order to compute decompositions of CFMRDs, we make use of a capping-free formulation for the computation of the pairwise DCJ indel distance derived in [2]. This formulation is based on the distance formula found in Theorem 3.

The formulation counts cycles $\mathtt{c}_E$ as well as the six different types of paths relevant to Theorem 3, namely $\mathtt{p}^{ab}, \mathtt{p}^{Aa}, \mathtt{p}^{aB}, \mathtt{p}^{Ab}, \mathtt{p}^{Bb}, \mathtt{p}^{AB}$. Each counting variable $\mathtt{p}^X$ is set by summing up binary report variables $\mathtt{r}_v^X$ that are set to 1 once per component on a specific vertex v (see Constraints `C.08` to `C.12` and `C.17`). These counters are then combined to the terms of the formula in Constraints `C.13` to `C.16` and `C.03` to `C.07`. The constraints for ensuring the reporting variables being set correctly can be found in Tables 4, 5 and 6. For a complete description of this part of the ILP the interested reader is referred to [2].

We make only few major changes in our local section w.r.t. the ILP described in [2]. Firstly, we determine whether an adjacency edge e is set ($\mathtt{x}_e = 1$) by "inheriting" this value from the linearization generated in the global section (see `C.20`) of the corresponding adjacency. Secondly, we allow only vertices that are part of the linearized genome ($\mathtt{g}_v = 1$) to contribute to the count of components that decrease the formula ($\mathtt{z}_v = 1$), see `C.21`.

Due to the fact that ancestral genomes may be degenerate, the number of possible circular singletons can be as large as the number of possible circular chromosomes. Listing all candidates, such as is done in [2] and in SPP-DCJ [8], leads to a combinatorial explosion on certain input data. Particularly, when all possible adjacencies are present in the degenerate genome, any non-empty subset of singular markers can form a circular singleton. A lower bound on the number of candidates is therefore $\sum_{i=1}^{|E_{\text{self}}|} \binom{|E_{\text{self}}|}{i} = 2^{|E_{\text{self}}|} - 1$. To avoid an exponential worst case size of our ILP, we use a new technique for counting circular singletons without listing all candidates when the number of candidates is larger than a given (polynomial) threshold, which we arbitrarily set at twice the number of self edges. The constraints for this technique are listed in Table 1 and described in the following.

A circular singleton manifests in the graph as a cycle of alternating adjacency and indel edges. The idea of the technique is to have a general integer variable `w` that is required to increase at each adjacency edge in a walk of the cycle. There must then be one point in the walk in which it decreases again. Detecting this, one can then report a circular singleton. For this to work, the walk needs a direction. This is accomplished by annotating the vertices

Algorithm 1 Capping-free Small Parsimony.

Objective

`Minimize`

$$\sum_{E \in E(T)} (\alpha\, \mathtt{f}_E + (\alpha - 1)\, \mathtt{w}_E)$$

Global level

For each genome $\mathbb{A} = (\mathcal{E} \cup \mathcal{T}, \mathcal{M} \cup \mathcal{A})$ of phylogeny T:

(C.01)	$\mathtt{g}_v = 1$	$v \in \mathcal{E}$
(C.02)	$\sum_{uv \in \mathcal{A}} \mathtt{a}_{uv} = \mathtt{g}_v$	$\forall v \in \mathcal{E} \cup \mathcal{T}$

Local level

For each edge $E = (\mathbb{A}, \mathbb{B}) \in E(T)$ with $\mathcal{CFMRD}(\mathbb{A}, \mathbb{B}) = (\mathcal{E} \cup \mathcal{T}, E_{\text{adj}} \cup E_{\text{ext}} \cup E_{\text{self}})$:

(C.03)	$\mathtt{w}_E = \sum_{e \in E_{\text{adj}}} \mathbf{w}(e) x_e$	
(C.04)	$\mathtt{f}_E = \mathtt{n}_E - \mathtt{c}_E + \mathtt{q}_E + \mathtt{s}_E$	
(C.05)	$\mathtt{n}_E = \frac{1}{2} \sum_{e \in E_{\text{ext}}} \mathtt{x}_e$	
(C.06)	$\mathtt{c}_E = \sum_{v \in \mathcal{E}} \mathtt{r}_v^c$	
(C.07)	$2\mathtt{q}_E \geq \mathtt{p}_E^{ab} + \mathtt{p}_E^{\max a} + \mathtt{p}_E^{\max b} - \mathtt{p}_E^{AB}$	
(C.08)	$\mathtt{p}_E^{ab} = \sum_{v \in \mathcal{E}} \mathtt{r}_v^{ab}$	
(C.09)	$\mathtt{p}_E^{Aa} = \sum_{v \in \mathcal{T}^A} \mathtt{r}_v^{Aa}$	
(C.10)	$\mathtt{p}_E^{aB} = \sum_{v \in \mathcal{T}^B} \mathtt{r}_v^{aB}$	
(C.11)	$\mathtt{p}_E^{Ab} = \sum_{v \in \mathcal{T}^A} \mathtt{r}_v^{Ab}$	
(C.12)	$\mathtt{p}_E^{Bb} = \sum_{v \in \mathcal{T}^B} \mathtt{r}_v^{Bb}$	
(C.13)	$\mathtt{p}_E^{\max a} \geq p_E^{Aa}$	
(C.14)	$\mathtt{p}_E^{\max a} \geq p_E^{aB}$	
(C.15)	$\mathtt{p}_E^{\max b} \geq p_E^{Ab}$	
(C.16)	$\mathtt{p}_E^{\max b} \geq p_E^{Bb}$	
(C.17)	$\mathtt{p}_E^{AB} = \sum_{v \in \mathcal{T}^A} \mathtt{r}_v^{AB}$	
(C.18)	$\mathtt{s}_E = \sum_{v \in \mathcal{E}} \mathtt{r}_v^s$	
(C.19)	$\sum_{uv \in E_{\text{ext}} \cup E_{\text{self}}} \mathtt{x}_{uv} = \mathtt{g}_{\gamma(u)}$	$\forall u \in \mathcal{E}$
(C.20)	$\mathtt{a}_{\gamma(u)\gamma(v)} = \mathtt{x}_{uv}$	$\forall uv \in E_{\text{adj}}$
(C.21)	$\mathtt{z}_v \leq \mathtt{g}_{\gamma(v)}$	$\forall v \in \mathcal{E}$
(C.22) to (C.24)	Reporting circular singletons	– see Table 1
(C.25) to (C.27)	Shao-Lin-Moret [13] constraints	– see Table 4 in Appendix A
(C.28) to (C.31)	Reporting for regular vertices	– see Table 5 in Appendix A
(C.32) to (C.37)	Reporting for pseudo-caps	– see Table 6 in Appendix A
(D.01) to (D.13)	Domains	– see Tables 2 and 3

with a binary variable $\mathtt{d}_v$ that "flips" across each pair of connected vertices (see C.22). We then require $\mathtt{w}$ to be the same for vertices connected by an indel edge (see C.23) and for it to increase by 1 in the direction of the vertex that has $\mathtt{d}_v = 1$ (see C.24). We require this except when vertices are not connected ($1 - \mathtt{x}_{uv} = 0$) or when reporting a circular singleton ($\mathtt{r}_u^s = 1$ or $\mathtt{r}_v^s = 1$). In this case, the constraint is automatically fulfilled by adding the maximum length of circular singletons K to the left hand side of the inequation.

Table 1 Reporting circular singletons.

(C.22)	$\mathtt{d}_u + \mathtt{d}_v + \mathtt{x}_{uv} \leq 2$	$\forall uv \in E_{\text{adj}} \cup E_{\text{self}}$
	$\mathtt{d}_u + \mathtt{d}_v - \mathtt{x}_{uv} \geq 0$	$\forall uv \in E_{\text{adj}} \cup E_{\text{self}}$
(C.23)	$\mathtt{w}_u = \mathtt{w}_v$	$\forall uv \in E_{\text{self}}$
(C.24)	$K(1 - \mathtt{x}_{uv} + \mathtt{r}_u^s + \mathtt{r}_v^s) + \mathtt{w}_v \geq \mathtt{w}_u + \mathtt{d}_v - \mathtt{d}_u$	$\forall uv \in E_{\text{adj}}$

Table 2 Domains – global level.

(D.01)	$\mathtt{g}_v \in \{0,1\}$	for each genome $\mathbb{X}$, $\forall v \in \mathcal{E}(\mathbb{X}) \cup \mathcal{T}(\mathbb{X})$
(D.02)	$\mathtt{f}_E, \mathtt{n}_E, \mathtt{c}_E, \mathtt{s}_E \in \mathbb{N}_0$	$\forall E \in E(T)$
(D.03)	$\mathtt{p}_E^{xy}, p_E^{\max a}, p_E^{\max b} \in \mathbb{N}_0$	$\forall E \in E(T)\, \forall x, y \in \{A, B, a, b\}, x \neq y$
(D.04)	$q_E \in \mathbb{Z}$	$\forall E \in E(T)$
(D.05)	$\mathtt{w}_E \in \mathbb{R}$	$\forall E \in E(T)$

Table 3 Domains – local level. For each edge $(\mathbb{A}, \mathbb{B}) \in E(T)$ with $\mathcal{CFMRD}(\mathbb{A}, \mathbb{B}) = (\mathcal{E} \cup \mathcal{T}, E_{\text{all}})$ with $E_{\text{all}} = E_{\text{adj}} \cup E_{\text{ext}} \cup E_{\text{self}}$.

(D.06)	$\mathtt{x}_e \in \{0,1\}$	$\forall e \in E_{\text{all}}$	(D.10)	$\mathtt{w}_v \in \mathbb{N}_0$	$v \in \mathcal{E}$
(D.07)	$\mathtt{y}_{v_i} \in \{0, ..., i\}$	$v_i \in \mathcal{E} \cup \mathcal{T}$	(D.11)	$r_v^{ab} \in \{0,1\}$	$\forall v \in \mathcal{E}(\mathbb{A})$
(D.08)	$\mathtt{z}_v, l_v \in \{0,1\}$	$v \in \mathcal{E} \cup \mathcal{T}$	(D.12)	$r_v^{Aa}, r_v^{Ab}, r_v^{AB} \in \{0,1\}$	$\forall v \in \mathcal{T}(\mathbb{A})$
(D.09)	$\mathtt{d}_v \in \{0,1\}$	$v \in \mathcal{E}$	(D.13)	$r_v^{aB}, r_v^{Bb} \in \{0,1\}$	$\forall v \in \mathcal{T}(\mathbb{B})$

3.3.3 Size of the ILP

For each CFMRD, the local level of the ILP assigns a constant number of variables to each vertex and edge (see Table 3). Additionally there is a constant number of constraints associated with each vertex and edge (see Tables 1, 4, 5, 6). For each edge in the phylogeny, there is a constant number of constraints and variables in the global level (see C.01 to C.02 and Table 2 respectively). The size of the ILP is thus linear with respect to the total size of all CFMRDs of the tree.

4 Evaluation

We implemented Algorithm 1 and made it publicly available[1]. We refer to this algorithm as *SPP-DCJ-v2* in the following. We performed a number of different experiments evaluating the solving time under different conditions as compared to SPP-DCJ as well as precision and recall for the safer linearization mode.

While solving the same problem, SPP-DCJ adds another parameter β to the optimization which gives further negative weight to telomeres. In short, the optimization function of SPP-DCJ is equivalent to the form

Minimize

$$\alpha' \sum_{E \in E(T)} \mathtt{f}_E + \beta' \sum_{E \in E(T)} \#\text{telomeres in decompositions of } E - (1 - \alpha' - \beta') \sum_{E \in E(T)} \mathtt{w}_E$$

We can simulate this behavior in our ILP by decreasing the assigned weight of telomeric adjacencies and by using a re-scaled α.

[1] https://github.com/marschall-lab/spp_dcj_v2

When comparing to SPP-DCJ, we thus used default settings for SPP-DCJ with $\alpha' = \frac{1}{2}$, $\beta' = \frac{1}{4}$. This corresponds in our ILP to $\alpha = \frac{2}{3}$ and reducing the weight of each telomeric adjacency by 1, so we used these parameters for SPP-DCJ-v2 when comparing to SPP-DCJ.

In all instances, we used gurobi version 11.0.0 on a single thread and with a time limit of 1 hour (3600 seconds) to solve the ILPs.

4.1 Performance on linear genomes

In order to compare the behavior of SPP-DCJ and SPP-DCJ-v2 in the presence of multiple linear chromosomes, we used the simulator ffs-dcj introduced in [2]. The simulator performs a number of DCJs, indels and duplications with fixed rates for a given tree topology. In this experiment, we used a fixed balanced tree topology, namely $(((A : 1.0, B : 1.0)F : 1.0), ((C : 1.0, D : 1.0)G : 1.0))Root;$. We simulated 30 operations per branch on genomes of size 100 markers. More detailed settings (such as rates of duplications and indels) can be found in Table 7 (Appendix B). The experiment was run for $2, 4, 6, 8, 10, 12, 14$ and 16 linear chromosomes at the root of the tree with 10 replicates for each step. We then proceeded to introduce 30 adjacencies of adversarial noise for each sample at the inner nodes utilizing a script provided by the SPP-DCJ repository.

We then ran SPP-DCJ and SPP-DCJ-v2 on degenerate genomes consisting of the true and noise adjacencies. The results in solving time are shown in Figure 6.

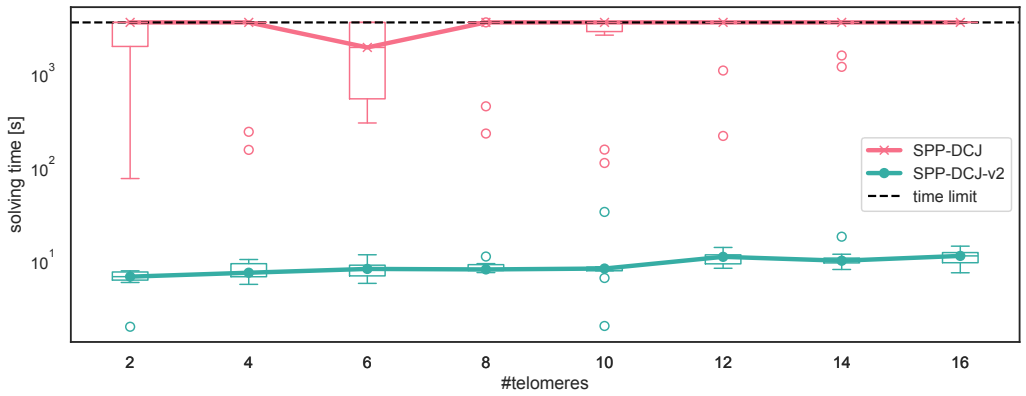

Figure 6 Solving times for SPP-DCJ and SPP-DCJ-v2 on simulated genomes with increasing numbers of telomeres. Solid lines represent corresponding median values.

We see that SPP-DCJ-v2 consistently needed one or two orders of magnitude less solving time than SPP-DCJ. A majority of SPP-DCJ runs did not complete within the time limit. The performance of SPP-DCJ also dramatically worsens with increasing numbers of linear chromosomes, such that for 16 linear chromosomes no run was completed within the time limit.

SPP-DCJ-v2 in turn was also affected by the rising numbers of linear chromosomes, but the effect is less drastic. In fact, the solving time for SPP-DCJ-v2 is well below a minute for all samples.

4.2 Performance on circular genomes

As we have seen in Section 3.2, even when in the ground truth all linearizations of chromosomes are circular, additional telomeres might still be necessary to ensure that all degenerate genomes are linearizable.

In order to examine this effect, we used the same pipeline as in [8] to simulate trees and genomes of 100 markers for each tree using ZOMBI [7] with tree scales ranging from 5 to 15 with 10 samples per step (for all parameter settings see Table 8). We then inferred degenerate genomes using DeCoSTAR [9] and solved the resulting SPP instances using SPP-DCJ and SPP-DCJ-v2. We visualize the resulting solving times in Figure 7.

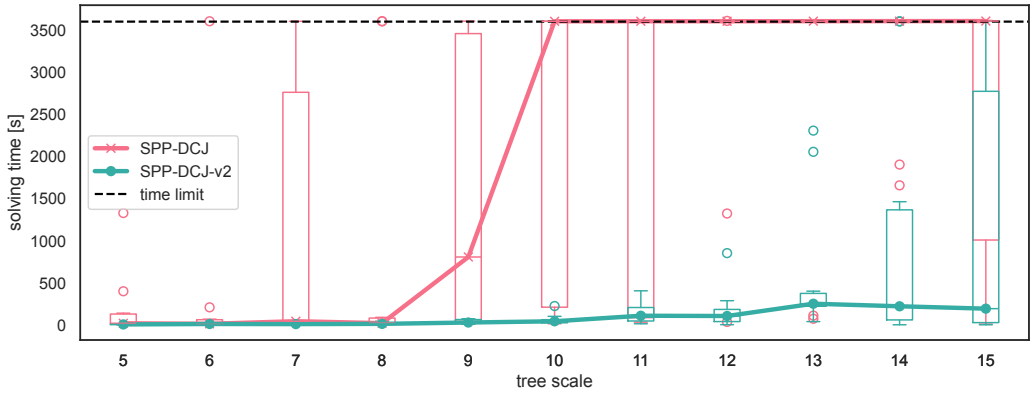

Figure 7 Solving times for SPP-SCJ and SPP-DCJ-v2 on genomes generated by ZOMBI on a range of trees with increasing branch lengths with ancestral adjacencies inferred by DecoSTAR. Solid lines represent corresponding median values.

Genomes generated by ZOMBI are circular, so one might assume that there is only negligible difference in runtime between SPP-DCJ and SPP-DCJ-v2. However, the results indicate that the improved handling of the solution space by SPP-DCJ-v2 allows it to solve problem instances with up to twice the tree scale as SPP-DCJ with comparable solving times.

Additionally, solving times by SPP-DCJ-v2 are much more predictable, increasing steadily with the tree scale while the runtimes of SPP-DCJ vary widely with some tree scales for which almost all runs did not terminate before the time limit. At present, we do not have a definitive explanation for this behavior. However, we hypothesize that certain tree scales might lead to sub-structures that have a more complicated structure with regard to the added telomeres, which then in turn negatively influence the solving time.

4.3 Evaluation of the safer linearization mode

We used the same pipeline to simulate genomes of 1000 markers with ZOMBI, inferring degenerate ancestral genomes with DecoSTAR over a range of tree scales with five samples per step. All other parameters are the same as in Table 8. This time, however, we used SPP-DCJ-v2 with both the default and the safer linearization modes and examined the precision and recall of recovered adjacencies. In this experiment, we used $\alpha = 0.5$ with weight 0 for the telomeric adjacencies added to ensure linearizability (see Section 3.2).

The results, illustrated in Figure 8, indicate that while our method displays very high precision and recall rates in both modes, the safer linearization mode has a minor, but consistent advantage over the default setting, especially considering precision. The trend in the data shows that this gap could widen further on more noisy data.

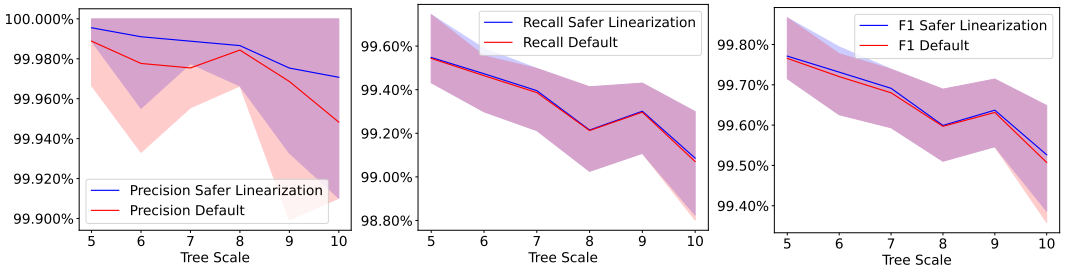

Figure 8 Mean precision, recall and F1 score for default and safer linearization mode for varying tree scales. Transparent ranges indicate minimum to maximum range of the five tested samples per step. Note that the diagrams have different y-axis scaling.

4.4 Reconstructing the ancestral X chromosomes of seven mosquitos

We further evaluated our method on biological data from seven *Anopheles* species whose inferred phylogeny is depicted in Figure 9. Gene annotations from protein coding genes of the X chromosome of present-day mosquitos were obtained from VectorBase [1]. Chromosome sizes fluctuated at around 600 genes. We then used the *ancestral gene order* (AGO) pipeline [6] to obtain candidate ancestral adjacencies. Using AGO, multiple sequence alignments were computed with MACSE [11], based upon which gene trees were inferred and reconciled with the species tree with IQ-TREE [10]. Finally, candidate ancestral adjacencies were computed with DeCoSTAR.

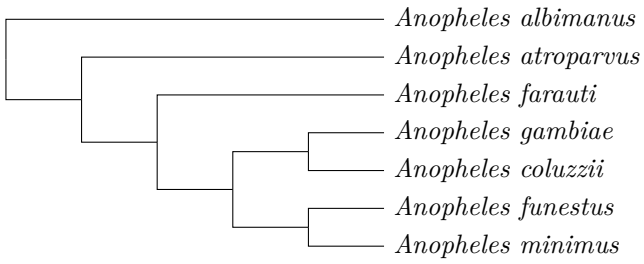

Figure 9 Cladogram for seven *Anopheles* taxa.

We ran both SPP-DCJ and SPP-DCJ-v2 to generate corresponding ILPs. These were then input to gurobi, which ran on 10 threads with a time limit of 12 hours on the same machine for both ILPs. Based on the SPP-DCJ ILP, gurobi did not find any solution within the time limit whereas using the improved ILP formulation with SPP-DCJ-v2, gurobi found an approximate solution with a 0.79% gap. Initial approximate solutions with a gap of 10% were found with SPP-DCJ-v2 already within the first minute of solving time.

5 Discussion

We presented SPP-DCJ-v2, the first ILP of polynomial size to solve the Small Parsimony Problem for natural genomes under the DCJ-indel model. Using a more efficient representation of the solution space, the Capping-Free Multi-Relational Diagram, we were able to significantly improve upon the performance of its predecessor, SPP-DCJ. Additionally, we introduced a new method of ensuring linearizability that is more robust when applied to (potentially noisy) real data because linearization is not the main constraint any more. Finally, we demonstrated that our approach is efficient enough to derive good solutions for SPP on real phylogenies.

References

1 Beatrice Amos, Cristina Aurrecoechea, Matthieu Barba, Ana Barreto, Evelina Y. Basenko, Wojciech Bażant, Robert Belnap, Ann S. Blevins, Ulrike Böhme, John Brestelli, Brian P. Brunk, Mark Caddick, Danielle Callan, Lahcen Campbell, Mikkel B. Christensen, George K. Christophides, Kathryn Crouch, Kristina Davis, Jeremy DeBarry, Ryan Doherty, Yikun Duan, Michael Dunn, Dave Falke, Steve Fisher, Paul Flicek, Brett Fox, Bindu Gajria, Gloria I. Giraldo-Calderón, Omar S. Harb, Elizabeth Harper, Christiane Hertz-Fowler, Mark J. Hickman, Connor Howington, Sufen Hu, Jay Humphrey, John Iodice, Andrew Jones, John Judkins, Sarah A. Kelly, Jessica C. Kissinger, Dae Kun Kwon, Kristopher Lamoureux, Daniel Lawson, Wei Li, Kallie Lies, Disha Lodha, Jamie Long, Robert M. MacCallum, Gareth Maslen, Mary Ann McDowell, Jaroslaw Nabrzyski, David S. Roos, Samuel S. C. Rund, Stephanie Wever Schulman, Achchuthan Shanmugasundram, Vasily Sitnik, Drew Spruill, David Starns, Christian J. Stoeckert, Jr., Sheena Shah Tomko, Haiming Wang, Susanne Warrenfeltz, Robert Wieck, Paul A. Wilkinson, Lin Xu, and Jie Zheng. VEuPathDB: the eukaryotic pathogen, vector and host bioinformatics resource center. *Nucleic Acids Research*, 50(D1):D898–D911, 2021. doi:10.1093/nar/gkab929.

2 Leonard Bohnenkämper. Recombinations, chains and caps: resolving problems with the DCJ-indel model. *Algorithms for Molecular Biology*, 19:8, 2024. doi:10.1186/s13015-024-00253-7.

3 Leonard Bohnenkämper, Marilia D. V. Braga, Daniel Doerr, and Jens Stoye. Computing the rearrangement distance of natural genomes. *Journal of Computational Biology*, 28(4):410–431, 2021. doi:10.1089/cmb.2020.0434.

4 Leonard Bohnenkämper and Daniel Dörr. SPP-DCJ. Software, swhId: swh:1:dir:9f96ced9254d812c0c0cd34376094007bc578a63 (visited on 2024-08-16). URL: https://github.com/marschall-lab/spp_dcj_v2.

5 Marilia D. V. Braga, Eyla Willing, and Jens Stoye. Double cut and join with insertions and deletions. *Journal of Computational Biology*, 18(9):1167–1184, 2011. doi:10.1089/cmb.2011.0118.

6 Evan P. Cribbie, Daniel Doerr, and Cedric Chauve. AGO, a framework for the reconstruction of ancestral syntenies and gene orders. In João C. Setubal, Jens Stoye, and Peter F. Stadler, editors, *Comparative Genomics, vol. 2*, volume 2802 of *Methods Molecular Biology*, pages 247–265. Springer, 2024. doi:10.1007/978-1-0716-3838-5_10.

7 Adrián A. Davín, Théo Tricou, Eric Tannier, Damien M. de Vienne, and Gergely J. Szöllősi. Zombi: a phylogenetic simulator of trees, genomes and sequences that accounts for dead linages. *Bioinformatics*, 36(4):1286–1288, 2019. doi:10.1093/bioinformatics/btz710.

8 Daniel Doerr and Cedric Chauve. Small parsimony for natural genomes in the DCJ-indel model. *Journal of Bioinformatics and Computational Biology*, 19(06):2140009, 2021. doi:10.1142/S0219720021400096.

9 Wandrille Duchemin, Yoann Anselmetti, Murray Patterson, Yann Ponty, Sèverine Bérard, Cedric Chauve, Celine Scornavacca, Vincent Daubin, and Eric Tannier. DeCoSTAR: Reconstructing the ancestral organization of genes or genomes using reconciled phylogenies. *Genome Biology and Evolution*, 9(5):1312–1319, 2017. doi:10.1093/gbe/evx069.

10 Bui Quang Minh, Heiko A. Schmidt, Olga Chernomor, Dominik Schrempf, Michael D. Woodhams, Arndt von Haeseler, and Robert Lanfear. IQ-TREE 2: New models and efficient methods for phylogenetic inference in the genomic era. *Molecular Biology and Evolution*, 37(5):1530–1534, 2020. doi:10.1093/molbev/msaa015.

11 Vincent Ranwez, Emmanuel J. P. Douzery, Cédric Cambon, Nathalie Chantret, and Frédéric Delsuc. MACSE v2: Toolkit for the alignment of coding sequences accounting for frameshifts and stop codons. *Molecular Biology and Evolution*, 35(10):2582–2584, 2018. doi:10.1093/molbev/msy159.

12 Diego P. Rubert and Marilia D. V. Braga. Efficient gene orthology inference via large-scale rearrangements. *Algorithms for Molecular Biology*, 18:14, 2023. doi:10.1186/s13015-023-00238-y.

13 Mingfu Shao, Yu Lin, and Bernard M. E. Moret. An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. *Journal of Computational Biology*, 22(5):425–435, 2015. doi:10.1089/cmb.2014.0096.

14 Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005. doi:10.1093/bioinformatics/bti535.

A Additional constraint tables

Table 4 Shao-Lin-Moret constraints.

(C.25)	$\mathtt{x}_e = \mathtt{x}_d$	for all sibling edges e, d
(C.26)	$\mathtt{y}_{v_i} + j(1 - \mathtt{x}_{u_j v_i}) \geq \mathtt{y}_{u_j}$	$\forall u_j v_i \in E_{\text{adj}} \cup E_{\text{ext}}$
	$j(1 - \mathtt{x}_{u_j v_i}) \geq \mathtt{y}_{u_j}$	$\forall u_j v_i \in E_{\text{self}}$
(C.27)	$i\mathtt{z}_{v_i} \leq \mathtt{y}_{v_i}$	$\forall v \in \mathcal{E} \cup \mathcal{T}$

Table 5 Reporting for regular vertices.

(C.28)	$\mathtt{l}_v \leq 1 - \mathtt{x}_{uv}$	$\forall uv \in E_{\text{self}}, u \in \mathcal{E}(\mathbb{A})$
	$\mathtt{l}_v \geq \mathtt{x}_{uv}$	$\forall uv \in E_{\text{self}}, u \in \mathcal{E}(\mathbb{B})$
(C.29)	$\mathtt{l}_v \leq \mathtt{l}_u + (1 - \mathtt{x}_{uv})$	$\forall uv \in E_{\text{ext}}$
	$\mathtt{l}_u \leq \mathtt{l}_v + \mathtt{r}_{uv}^{ab} + (1 - \mathtt{x}_{uv})$	$\forall uv \in E_{adj}, u \in \mathcal{E}(\mathbb{A})$
	$\mathtt{l}_u \leq \mathtt{l}_v + (1 - \mathtt{x}_{uv})$	$\forall uv \in E_{adj}, u \in \mathcal{E}(\mathbb{B})$
(C.30)	$\mathtt{r}_v^c \leq \mathtt{z}_v$	$\forall v \in \mathcal{E}(\mathbb{A})$
(C.31)	$\mathtt{r}_u^{ab} \leq \mathtt{x}_{uv}$	$\forall uv \in E_{\text{self}}, u \in \mathcal{E}(\mathbb{A})$

Table 6 Reporting for telomeres.

(C.32)	$\mathtt{l}_v = 0$	$\forall v \in \mathcal{T}(\mathbb{A})$
	$\mathtt{l}_v = 1$	$\forall v \in \mathcal{T}(\mathbb{B})$
(C.33)	$\mathtt{l}_u \leq \mathtt{l}_v + \mathtt{r}_v^{AB} + \mathtt{r}_v^{Ab} + (1 - \mathtt{x}_{uv})$	$\forall uv \in E_{\text{adj}}, v \in \mathcal{T}(\mathbb{A})$
	$\mathtt{l}_u \leq \mathtt{l}_v + \mathtt{r}_u^{aB} + (1 - \mathtt{x}_{uv})$	$\forall uv \in E_{\text{adj}}, u \in \mathcal{T}(\mathbb{B})$
(C.34)	$\mathtt{r}_v^{AB} \leq \mathtt{z}_v$	$\forall v \in \mathcal{T}(\mathbb{A})$
(C.35)	$1 - \mathtt{y}_v \leq \mathtt{r}_v^{Ab} + \mathtt{r}_v^{Aa}$	$v \in \mathcal{T}(\mathbb{A})$
	$1 - \mathtt{y}_v \leq \mathtt{r}_v^{aB} + \mathtt{r}_v^{Bb}$	$v \in \mathcal{T}(\mathbb{B})$
(C.36)	$\mathtt{y}_{v_i} \leq i(1 - \mathtt{r}_v^R)$	$v \in \mathcal{T}(\mathbb{A}), R \in \{Ab, Aa\}$
	$\mathtt{y}_{v_i} \leq i(1 - \mathtt{r}_v^R)$	$v \in \mathcal{T}(\mathbb{B}), R \in \{aB, Bb\}$
(C.37)	$\mathtt{r}_v^{AB} \leq \mathtt{l}_u + (1 - x_{uv})$	$\forall uv \in E_{\text{adj}}, v \in \mathcal{T}(\mathbb{A})$
	$\mathtt{r}_v^{Ab} \leq \mathtt{l}_u + (1 - x_{uv})$	$\forall uv \in E_{\text{adj}}, v \in \mathcal{T}(\mathbb{A})$
	$\mathtt{r}_v^{aB} \leq 1 - \mathtt{l}_u + (1 - x_{uv})$	$\forall uv \in E_{\text{adj}}, v \in \mathcal{T}(\mathbb{B})$

B Experiment parameter tables

Table 7 Parameters for ffs-DCJ for the linear chromosome experiment.

Duplication rate	0.4
Zipf parameter duplication	6.0
Deletion Rate	0.2
Insertion Rate	0.1
Zipf parameter indel	4.0

Table 8 Parameter settings for ZOMBI and DeCoSTAR for the tree scale and precision experiments. For the sake of benchmarking SPP-DCJ-v2, ZOMBI parameters for genome evolution were chosen to represent an elevated degree of genome evolution, both in terms of gene content innovation (duplication+loss) and rearrangement (inversion+transposition).

ZOMBI	
`DUPLICATION`	f:2
`INITIAL_GENOME_SIZE`	100
`LOSS`	f:2
`LOSS_EXTENSION`	g:0.8
`ORIGINATION`	f:0
`INVERSION`	f:2
`INVERSION_EXTENSION`	g:0.5
`TRANSPOSITION`	f:2
`TRANSPOSITION_EXTENSION`	g:0.5

DeCoSTAR	
`use.boltzmann`	1
`boltzmann.temperature`	1.0
`nb.sample`	1000