Towards a Unified Exact Solution of Rearrangement Small Parsimony for Natural Genomes

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

Daria Frolova
European Bioinformatics Institute, Hinxton, Cambridge, UK
Department of Computational Biology, University of Lausanne, Switzerland

Abstract

Phylogenetic reconstruction is a fundamental problem in comparative genomics. As a theoretical problem in rearrangement studies, this has been modelled as the Small Parsimony Problem (SPP), in which ancestral genome structures have to be determined minimizing the number of rearrangement events occurring throughout the phylogeny. This problem is of significant interest in microbial and cancer genomics, due to the prevalence and clinical importance of rearrangement events.

Genome structures in this problem are expressed as sequences of *markers*, which are themselves oriented sequence features (such as genes) that abstract from non-structural variations. Recent research has focused on the problem under the natural genomes model, in which arbitrary variations in copy number of markers are allowed. Natural genomes are often studied under the DCJ-indel model, a model which has already been successfully applied to plasmid data. There also exist ILP solutions to a variant of the Small Parsimony Problem under the DCJ-indel model. However, these solutions are limited in their applicability, as they make some critical simplifications for tractability purposes: ancestral marker frequencies and precomputed putative ancestral adjancencies, with their predicted likelihoods, are assumed as input.

This creates multiple problems from both a theoretical and practical perspective. Firstly, this simplification means that not the full state space is searched for a solution, but rather only the subset of genomes with the precomputed putative adjacencies, meaning an optimal solution to the exact SPP is not guaranteed. Secondly, marker frequencies are given externally, without any theoretical guarantees. Thirdly, the method used to precompute adjacencies relies on gene trees, which requires the use of genes as markers, when gene annotation is often unreliable, especially in regions with a lot of rearrangement. Additionally, this restricts the applicability of the approach to sets of genomes that are both divergent and large enough to be able to produce informative gene trees. This is, for example, rarely the case for plasmids, where nucleotide mutations are rarer than rearrangements and genomes are small.

Hence, we revisit the problem to solve the exact SPP by introducing a cost to indel operations, which allows us to compute ranges of marker frequencies and derive theoretical results, that allow us to reduce the solution space that the ILP searches without sacrificing optimality. We show that this makes the problem tractable for the case of small and recently related genomes, first on simulated genomes, and then on a set of pathogenic plasmids which represent a realistic use case for the method.

2012 ACM Subject Classification Applied computing → Computational genomics

Keywords and phrases Rearrangement, Small Parsimony Problem, Natural Genomes, DCJ-indel

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.15

Related Version *Preprint*: `https://doi.org/10.64898/2026.06.23.733974`

Supplementary Material *Software*: `https://github.com/gi-bielefeld/spp_dcj_exact` [4]
archived at `swh:1:dir:9d2d2213b28d627478afc8a5f365b31b0d0979f1`

Acknowledgements LB thanks Roland Wittler for pointing us towards helpful literature concerning Wagner Parsimony as well as Niccolò Gugliette for a helpful discussion.

1 Introduction

The problem of parsimonious ancestral reconstruction with a given tree is known as the *Small Parsimony Problem (SPP)*. In general, SPP assumes a given tree with known states at the leaves and a cost function penalising changes between states. It then asks to find ancestral states minimising the total cost in the tree. SPP is solved by Fitch's method for nucleotide substitutions under unit cost [18]. However, genomic change is not exclusively characterised by substitutions, and structural changes like gene rearrangements and insertions/deletions (indels) of sequence regions play a significant role in evolution – notable examples include microbial plasmids [24], bacteria [11] and fungi [27], both chromosomal and extrachromosomal DNA in cancer [14, 23], and eukaryotic mitochondrial DNA [2, 32].

To quantify such structural changes various combinatorial rearrangement models have been suggested, in which genomes are treated as sequences or oriented markers, where markers correspond to some sequence features (e.g. genes or synteny blocks). We distinguish here between *singular genomes*, in which each marker occurs at most once and *natural genomes*, for which no restrictions pertaining to copy number is made. In the context of rearrangement models, SPP then asks to reconstruct ancestral marker orders which minimise the total number of rearrangements under a given model.

As a fundamental reconstruction problem, the small parsimony problem has been studied under various rearrangement models. Notably, even for singular genomes, the problem is already NP-hard for only three leaves (median of 3 problem) under complex models, such as the Double-Cut-And-Join model [29] or the Reversal model [9]. In fact, the Problem is only known to be polynomially solvable for the simple Single-Cut-or-Join model [17], which however is not able to handle arbitrary natural genomes.

One rearrangement model that has been successfully extended to handle natural genomes is the *Double-Cut-And-Join and indel (DCJ-indel) model* [8, 5, 3]. The rearrangement operations in this model are Double-Cut-And-Join (DCJ) – a general rearrangement operation able to mimic reversals, translocations and transpositions – as well as insertions and deletions of one or multiple markers at once. As determining the minimum number of DCJs and indels between a pair of natural genomes is already NP-hard [5], the solutions to this problem [5, 3] are based on an Integer Linear Program (ILP) [26].

The first attempts at a solution to DCJ-indel SPP for natural genomes were undertaken by Doerr and Chauve [13] and Bohnenkämper *et al* [7]. However, owing to the computational complexity of the problem, these solutions rely on several simplifications. Firstly, the copy number of each marker at ancestral nodes is assumed as input. Secondly, a collection of putative ancestral adjacencies is precomputed, and the ILP searches only within the space of these previously inferred adjacencies, rather than the full space of adjacencies. Thirdly, these precomputed adjacencies are given a weight, and this weight is incorporated into the objective function of the ILP. Although these simplifications help make the problem tractable, it introduces several critical limitations. Since only a part of the space of adjacencies is searched, it does not truly solve the SPP, and an optimal solution on the full space is not guaranteed. Additionally, the current method used to precompute ancestral adjacencies and their weights relies on gene trees [15]. Gene annotation is often unreliable, especially in regions with structural variation (e.g. even two identical genomes can be annotated differently due to genes being missed), making genes difficult to use as markers in practice [19, 30]. Moreover, grouping genes into homologous groups is also error prone and for recently related or very small genomes there is little SNP information to build gene trees from, making them insufficiently informative for the problem, and therefore limiting its applicability to between-species comparisons.

Naively, one can still achieve a solution to the exact DCJ-indel SPP, and avoid reliance on gene trees, by choosing weights of zero and inputting the entire space of adjacencies. However, this requires knowledge of the correct copy number for each internal node, for which so far no theoretical result exists. Moreover, in practice, such large input data cannot be handled by ILP solvers efficiently, as we demonstrate in Section 4.

In this work, we make the first step to resolve these problems, establishing a way to infer copy numbers using a modified version of the DCJ-indel distance and using properties of this distance measure – notably a restored triangle inequality – to restrict the solution space from all adjacencies to a set of candidates that guarantees at least one optimal solution remains.

The remainder of this manuscript is organised as follows. In Section 2, we give basic definitions and detail previous work. In Section 3.1, we discuss how to parsimoniously infer marker frequencies using a Fitch-like approach. In Sections 3.2 and 3.3, we introduce a way to restrict the solution space, such that at least one optimal solution under the DCJ-indel model is preserved. In Section 3.4, we introduce a way to use strategies like weighting adjacencies as a runtime heuristic as opposed to a correctness heuristic in an ILP context. We evaluate our solution in Section 4 and discuss our findings in Section 5.

2 Background

In rearrangement studies, one often abstracts from the concrete sequence content to the higher level of synteny blocks or *markers*. Formally, a marker m is a symbol from a (large) alphabet. To represent strandedness, markers are assigned an *orientation*, i.e. $+m$ or $-m$. We then speak of an *oriented marker*. A *chromosome* is then modeled as a circular or linear string over the alphabet of oriented markers and a *genome* as a collection of chromosomes.

In our notation, we write a circular string in round and a linear string in square brackets. For example, $E = \{[+1 - 2 - 1], (+3), (+3)\}$ is a genome with a linear and two circular chromosomes. Note that we do not restrict the number of occurrences of a chromosome or marker in our definitions. Also note that inversions of chromosomes are considered equivalent, i.e. $[+1 + 2 + 3] = [-3 - 2 - 1]$ as well as rotations of circular chromosomes, i.e. $(+1 - 2) = (-2 + 1)$.

We define the *universe of natural genomes* $\mathcal{U}$ as the set of all possible genomes over a given marker alphabet. Observe that $\mathcal{U}$ is infinite even though the underlying marker alphabet considered is finite.

The *Small Parsimony Problem* is a reconstruction problem, in which ancestral states are reconstructed for a known tree and known extant genomes in such a manner that the minimum number of evolutionary events is needed to "explain" the resulting phylogeny.

Formally, a binary tree T is a connected acyclic graph, such that each node has at most three neighbors. A node is called a *leaf* if it has degree 1, otherwise it is an *internal node*. Sometimes it is helpful to examine only a part of the tree by considering a *subtree* of T. To that end, removing some edge $(u, r) \in E(T)$ splits the tree into two components. Formally, the *subtree* with *root* r is then the component containing r. In the Small Parsimony Problem, each leaf represents an extant genome whereas each internal node represents the most recent common ancestor of all leaves in one of the subtrees of which it is the root.

Given a rearrangement distance measure d, the Small Parsimony Problem in the context of natural genomes can then be simply defined as follows.

▶ **Problem 1** (Small Parsimony Problem for Natural Genomes)**.** *Given a binary tree T with edges $E(T)$ and genome labels $L_u \in \mathcal{U}$ for each leaf u, find labels $L_v \in \mathcal{U}$ for each internal node v, such that*

$$\sum_{(v',u')\in E(T)} d(L_{v'}, L_{u'})$$

is minimised *under distance d.*

2.1 Previous Work

In this subsection, we summarise previous work, which is necessary to derive the results of our contribution. We begin by discussing the DCJ-indel distance in Section 2.1.1 and the previous approach by Doerr and Chauve to solve a restricted version of SPP in Section 2.1.2.

2.1.1 DCJ-indel distance for natural genomes

Here we summarise previous results on an important part of the distance model d that we will later use in Problem 1. Rearrangement distances between two genomes $A, B \in \mathcal{U}$ are often defined as the minimum number of rearrangement operations under a chosen model needed to transform A into B. The problem of determining this minimum number of rearrangements between two genomes is known as the *distance problem*. To that end, it is often useful to disambiguate between different occurrences of the same marker. We thus presume that there is some arbitrary labelling A^l for each genome A that adds a unique sub-indentifier i to each occurrence of a marker m, obtaining m_i. We call m_i a *marker occurrence* and A^l a *labelled genome*. For example, $E^l = \{[+1_1 - 2_1 - 1_2], (+3_1), (+3_2)\}$ would be one of multiple labellings for the genome E introduced previously. From now on, we presume each genome is a labelled genome.

Each (labelled) genome can be equivalently described by its *adjacencies*. For each marker occurrence m_i in A, we denote its beginning by m_i^t (tail) and its end by m_i^h (head), i.e. $+m_i = (m_i^t, m_i^h)$ and $-m_i = (m_i^h, m_i^t)$. We call m_i^h, m_i^t *extremities*. An *adjacency* is then a pair of extremities $\{m_i^x, n_j^y\}$, $x, y \in \{t, h\}$, such that the corresponding marker occurrences m_i and n_j are neighboring on a chromosome and m_i^x and n_j^y are adjacent. For example, in E^l we have the adjacencies $\{1_1^h, 2_1^h\}, \{2_1^t, 1_2^h\}, \{3_1^h, 3_1^t\}, \{3_2^h, 3_2^t\}$. In linear chromosomes, the first extremity of the first marker and second extremity of the last marker are not adjacent to any other extremity. For mathematical convenience we thus define a *telomeric adjacency* $\{o_k^z, \varnothing(o_k^z)\}$ for such an extremity o_k^z for a unique artificial extremity $\varnothing(o_k^z)$. $\varnothing(o_k^z)$ is called a *pseudo-cap*. For example, in E^l we have the telomeric adjacencies $\{1_1^t, \varnothing(1_1^t)\}$ and $\{1_2^t, \varnothing(1_2^t)\}$. When the concrete extremities are irrelevant, we use the shorthand notation $ab = \{m_i^x, n_i^y\}$ to denote the adjacency as an arbitrary pair of extremities a and b. Note that the order of a and b is still arbitrary, i.e. $ab = ba$.

A relatively simple but powerful rearrangement model is the DCJ-indel model [8]. It was first introduced on *singular genomes*, a finite subset $S \subset \mathcal{U}$ of all natural genomes. In a singular genome each marker may occur at most once. To this end, we define the *copy number* $\phi(A, m)$ of marker m as the number of times marker m occurs in genome A as some occurrence m_i. For example, earlier in E^l we have $\phi(E^l, 1) = \phi(E^l, 3) = 2, \phi(E^l, 2) = 1$. We can then define the set of singular genomes as $S \coloneqq \{A \in \mathcal{U} \mid \forall m\, \phi(A, m) \leq 1\}$. Moreover, we say a marker m with $\phi(A, m) = 1$ and its extremities are *singular* in A.

Within the DCJ-indel model, the following operations are permitted:

- A *Double-Cut-And-Join (DCJ)* transforms the adjacencies of a genome in one of the following ways:
  - $ab, cd \to ac, bd$
  - $a\varnothing(a), b\varnothing(b) \to ab$
  - $ab, c\varnothing(c) \to ac, b\varnothing(b)$
  - $ab \to a\varnothing(a), b\varnothing(b)$
- An *insertion or deletion* (indel), in which a segment of one or multiple markers is inserted or deleted at any position in the genome.

In order to prevent "free lunch" scenarios, in which chromosomes are simply deleted as a whole, the original formulation in [8] allows only to insert or delete non-shared markers between the two genomes. The DCJ-indel distance $d_{\mathrm{DCJ}}^{\mathrm{id}}(A, B)$ for two singular genomes $A, B \in S$ is then the minimum number of DCJ and indel operations needed to transform one genome into the other.

This distance can be calculated in linear time using the *relational diagram*. Given two singular genomes A and B, the *Relational Diagram (RD)* $R(A, B)$ is a graph (V, E). Its set of vertices is $V \coloneqq V(A) \cup V(B)$, where $V(A)$ has a vertex for each extremity of each marker occurrence of genome A, and $V(B)$ has a vertex for each extremity of each marker occurrence of genome B. For any adjacency ab in A, an *adjacency edge* is added between vertices a and b, and similarly for any adjacency in B. Additionally, for every shared marker m, any m_i^h in $V(A)$ is connected to any m_j^h in $V(B)$ by an *extremity edge*, and similarly any m_i^t in $V(A)$ to any m_j^t in $V(B)$. An example for a RD is given in Figure 1.

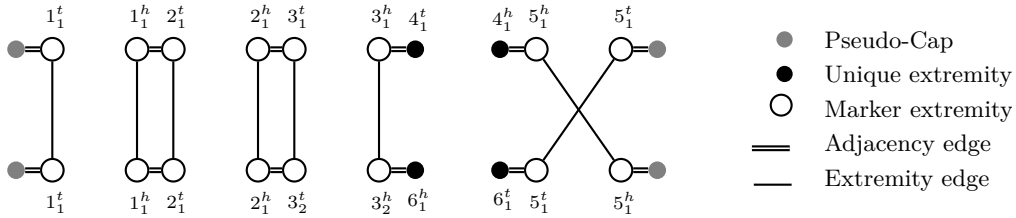

Figure 1 Relational Diagram (RD) for singular genomes $A = \{[+1_1 + 2_1 + 3_1 + 4_1 - 5_1]\}$ and $B = \{[+1_1 + 2_1 + 3_1 - 6_1 + 5_1]\}$.

Notably, since each marker occurs at most once in each genome, the RD consists only of simple cycles and paths. Paths can either end in a pseudo-cap, in which case we denote the path end by the genome name to which the pseudo-cap belongs in capital letters (A or B) or in an extremity unique to one of the genomes, in which case we denote the path end by the genome name in lowercase letters (a or b). All combinations of these path ends yield 10 different path types, the counts of which in a given Relational Diagram we denote accordingly by $p_{AA}, p_{AB}, p_{BB}, p_{Aa}, p_{Ab}, p_{Ba}, p_{Bb}, p_{aa}, p_{ab}, p_{bb}$. We write the number of cycles as c. For example, in Figure 1, we have $c = 2, p_{AB} = 1, p_{ab} = 1, p_{Ab} = 1, p_{Ba} = 1$ with all other counts being 0.

Circular singletons are structures that are not represented in this graph (see [3] for details), but still necessary to calculate the DCJ-indel distance. A circular singleton is a circular chromosome consisting only of markers unique to one genome. We denote the number of circular singletons by s. In [3], the following result is derived.

▶ **Theorem 1** (adapted from [3]). *For two singular genomes A, B, the DCJ-indel distance is*

$$d_{DCJ}^{id}(A, B) = n - c + \left\lceil \frac{p_{ab} + \max(p_{Aa}, p_{Ba}) + \max(p_{Ab}, p_{Bb}) - p_{AB}}{2} \right\rceil + s$$

with n the number of shared markers.

The distance problem of natural genomes can be reduced to the distance problem of singular genomes by employing a *matching*, which informs for a marker m which occurrence m_i in A should be considered the same as m_j in B. A matching between genomes A and B is defined as a set of pairs $M = \{(m_i, m_j) \mid m \text{ shared marker}, m_i \text{ occurrence in } A, m_j \text{ occurrence in } B\}$, such that each marker occurrence appears at most once in M.

Given a matching for each marker in A and B, we can compare them as singular genomes. For example, given $(m_i, m_j) \in M$ can then rewrite both m_i in A and m_j in B to the same unique character u^{ij}, and rewrite any occurrence not in M to a character that does not occur in the other genome. We call the resulting genomes A^M, B^M the *singularisations* of A and B. The distance between two natural genomes can then be defined as the minimum distance between any two singularisations obtained by a matching.

However, not all matchings are meaningful. For example, choosing the empty matching would lead to all chromosomes being inserted or deleted, which under the DCJ-indel distance may actually be optimal, since it requires only one (indel) operation per chromosome. One way to resolve this issue is to require the matching to be *maximal*, i.e. such that it cannot be extended. In this case, this is equivalent to the matching having maximum cardinality. Formally, we write $\mathbb{M}(A, B)$ as the set of all possible matchings between two genomes A, B and $\overline{\mathbb{M}}(A, B) \coloneqq \{M \in \mathbb{M}(A, B) \mid \nexists M' \in \mathbb{M}(A, B), |M'| > |M|\}$ as the set of *maximal matchings*.

In [5, 13, 7], the distance measure $\overline{d^{\text{id}}_{\text{DCJ}}}$ for two natural genomes A, B is defined as the minimum DCJ-indel distance between any singularisations obtained by a maximal matching:

$$\overline{d^{\text{id}}_{\text{DCJ}}}(A, B) = \min_{M \in \overline{\mathbb{M}}(A,B)} d^{\text{id}}_{\text{DCJ}}(A^M, B^M). \tag{1}$$

We call this distance model the *restricted* DCJ-indel distance.

2.1.2 Linearising degenerate genomes

The first attempt to solve the Small Parsimony Problem under the DCJ-indel model for natural genomes was undertaken by Doerr and Chauve in 2021 [13]. In their formulation, the authors restrict Problem 1 in two critical ways: (I) It is assumed that the copy number $\phi(L_v, m)$ of the genome label $L_v \in \mathcal{U}$ internal node v is given for each marker m. This restricts the infinite universe $\mathcal{U}$ to a finite subset of candidate genomes. (II) It is assumed that even within that restricted subset, only some adjacencies are permitted, further restricting the space of possibilities.

The first assumption about marker copy numbers was relaxed in [7], allowing for a range of given copy numbers $l(v, m) \leq \phi(L_v, m) \leq h(v, m)$ for the genome label L_v at internal node v. Nonetheless, this solution still relies on external heuristics to derive copy number counts and candidate adjacencies, which inform the finite candidate set that is explored. The advantage of this formulation is that the solution space with given input copy numbers and adjacencies can be encoded as a *degenerate genome*.

▶ **Definition 2** (adapted from [13]). *A* degenerate genome *is a set of adjacencies D that satisfies the following conditions: (i) $\forall g_i^t \in \bigcup D$, there exists also extremity $g_i^h \in \bigcup A$ and vice versa, and (ii) each telomeric extremity $\varnothing(g_i^x)$ is used only in the adjacency $\{g_i^x, \varnothing(g_i^x)\}$.*

The solution space spanned by a degenerate genome D is then each genome that has only adjacencies from D. Such a genome is called a *linearisation of D*. Formally, a *linearisation* of degenerate genome D is a genome A, such that each adjacency $\{m_i^x n_j^y\}$ of A is part of D, i.e. $\{m_i^x n_j^y\} \in D$.

In the original formulation [13], each extremity in D must be used in the linearisation A. However, since the ILP solution in [7] allows for variant copy numbers, we choose a slight reformulation that explicitly encodes the upper and lower bounds for each marker. The variant of SPP solved in [7] can then be formulated as follows.

▶ **Problem 2** (adapted from [7]; Weighted Small Parsimony Linearisation Problem)**.** *Given a binary tree T with edges $E(T)$, genome labels $L_u \in \mathcal{U}$ for each leaf u, degenerate genomes D_v for each internal node v, a weighting function w_v for adjacencies of each degenerate genome D_v, lower and higher bounds $l(v,m), h(v,m)$ for each marker m at each internal node v and a parameter $\alpha \in [0,1]$, find a linearisation L_v for each degenerate genome D_v, such that $l(v,m) \leq \phi(L_v,m) \leq h(v,m)$, minimising*

$$\sum_{(u,v)\in E(T)} \left(\alpha\, \overline{d^{id}_{DCJ}}(L_u, L_v) \;+\; (\alpha - 1) \left(\sum_{\substack{ab \\ adj.\ of\ L_v}} w_v(ab) \;+\; \sum_{\substack{ab \\ adj.\ of\ L_u}} w_u(ab) \right) \right).$$

Observe that what is optimised in this problem formulation is not the restricted DCJ-indel distance, but, depending on α, also a weight bias w_v toward the input adjacencies at each node v. These weights are part of the input and result from confidence scores of the adjacency prediction for ancestral genomes. The consequence of this formulation is that the set of degenerate genomes and weights associated with each of their adjacencies are assumed to be given, and must be precomputed before solving Problem 2. Since these are derived externally and without any theoretical guarantees, this formulation does not solve the SPP for natural genomes as formulated in Problem 1.

Note however, that if one could guarantee that the optimal solution is encoded in the degenerate genomes D_v and set $\alpha = 1$, Problem 2 could be used to solve to Problem 1. In the following, we will develop the theoretical background that allows us to do exactly this. We will show how to restrict the copy number of each marker to guarantee all optimal solutions are still represented and derive results that help to restrict the space of possible adjacencies while guaranteeing at least one optimal solution can be found.

2.2 Problem Definition

Besides restricting the infinite solution space $\mathcal{U}$ to the finite solution space expressible by degenerate genomes, the reason why marker copy numbers ϕ are assumed to be given in [13] and [5], is that using $d = \overline{d^{\text{id}}_{\text{DCJ}}}$ in Problem 1 would inherently cause problems due to some adverse properties of the restricted DCJ-indel distance. Importantly, this distance is not a metric, because it violates the triangle inequality [8].

While this distance superficially resembles a transformation distance and therefore would automatically fulfil the triangle inequality, it is not, because not all intermediate genomes are permitted when transforming one genome into the other.

The most striking example for this is choosing any two unichromosomal genomes $A = \{[S_A]\}, B = \{[S_B]\}$ for any oriented strings S_A and S_B over the marker alphabet that have distance $\overline{d^{\text{id}}_{\text{DCJ}}}(A,B) > 2$. If we now regard the empty genome $C = \{\}$, we observe that $\overline{d^{\text{id}}_{\text{DCJ}}}(A,C) + \overline{d^{\text{id}}_{\text{DCJ}}}(C,B) = 1 + 1 < \overline{d^{\text{id}}_{\text{DCJ}}}(A,B)$ as only a single segmental deletion or insertion is necessary to transform A into C or C into B, but when transforming A into B, C is not permitted as an intermediate genome. The empty genome is thus almost always among the genomes with the lowest total restricted DCJ-indel distance to any set of genomes. Thus, solving Problem 1 simply setting $d = \overline{d^{\text{id}}_{\text{DCJ}}}$ will result with high likelihood in empty or near-empty ancestral genomes.

However, instead of restricting marker multiplicities externally, we opt to instead restore the triangle inequality for the DCJ-indel distance by permitting all intermediate genomes, but penalising excessive gains and losses. To that end, we adapt the solution suggested by Braga, Willing and Stoye in the original publication of the DCJ-indel model [8]. There the authors propose to use a surcharge δ for each inserted or deleted marker between two singular genomes. Essentially, an indel, instead of having fixed cost 1, instead has cost $1 + \delta l$ where l is the length of the inserted or deleted segment. For singular genomes, this cost can then be calculated after determining the default DCJ-indel distance by simply examining the non-shared markers. Defining the non-shared markers as $A \triangle B \coloneqq \{m \mid \phi(A, m) = 1, \phi(B, m) = 0\} \cup \{m \mid \phi(A, m) = 0, \phi(B, m) = 1\}$, this distance is then $d^s(A, B) = d^{\text{id}}_{\text{DCJ}}(A, B) + \delta|A \triangle B|$.

Note that this result is restricted to singular genomes S. In order to apply it to all natural genomes $\mathcal{U}$, we can generalise as follows.

$$d(A, B) \coloneqq \min_{M \in \mathbb{M}(A,B)} d^{\text{id}}_{\text{DCJ}}(A^M, B^M) + \delta|A^M \triangle B^M| \tag{2}$$

Importantly, we allow any matching $M \in \mathbb{M}(A, B)$ here, not just maximal matchings, permitting any intermediate genome. Observe that then d is equivalent to the minimum cost of transforming A into B when each DCJ has cost 1 and each indel of length l costs $1 + \delta l$, but with no restrictions on which markers can be inserted or deleted – even shared markers can be part of indel operations. Thus, d is a transformation distance. We then arrive at the following.

▶ **Theorem 3.** *d is a metric on $\mathcal{U}$ for any $\delta \in \mathbb{R}, \delta \geq 0$.*

Proof. Identity and symmetry follow directly from the identity and symmetry of the DCJ-indel distance and identity and symmetry of $|A^M \triangle B^M|$. What remains to be shown is the triangle inequality, which follows directly from d being a transformation distance. Let thus $A, B, C \in \mathcal{U}$. Observe that if transforming A into C has cost $d(A, C)$ and C into B has cost $d(C, B)$, then there is a way to transform A into B with cost $d(A, C) + d(C, B)$. Thus, $d(A, C) + d(C, B) \geq d(A, B)$. ◀

In principle, we can thus choose any δ for our formulation, but choosing a small δ will lead to the same problems as the pure DCJ-indel distance. In previous work on natural genomes, these problems are circumvented by allowing only maximal matchings [5, 13, 7].

Note maximising the matching size $|M|$ is equivalent to minimising $|A^M \triangle B^M|$. In the context of our distance model d, this is then equivalent to regarding $\delta \to \infty$, so that any difference in $\delta|A^M \triangle B^M|$ is greater than any difference in $d^{\text{id}}_{\text{DCJ}}$, enforcing as many markers as possible are matched. This means for our formulation we can first minimise $\delta|A^M \triangle B^M|$ and then minimise $d^{\text{id}}_{\text{DCJ}}$ under the maximal matching model.

3 Methods

Our method rests on reducing Problem 1 to Problem 2, that is, we find degenerate genomes that guarantee to encode the optimal linearisations from the universe $\mathcal{U}$. To that end, knowing the copy number for each marker is essential. Since we presume $\delta|A^M \triangle B^M|$ is the dominant term in d and can therefore be optimised in advance, we first derive marker copy numbers along the tree based on this cost in Section 3.1. Once we know the copy number bounds $l(v, m), h(v, m)$ for each marker m at each internal node v, we can then create the degenerate genome D_v consisting of *all possible adjacencies* for the given extremities. We call

such a genome *fully degenerate*. Concretely, the extremities given the copy number bounds are $X = \{m_i^t \mid \forall m, 1 \leq i \leq h(v,m)\} \cup \{m_i^h \mid \forall m, 1 \leq i \leq h(v,m)\}$, and the fully degenerate genome is $D_v = \{\{x,y\} \mid x \in X, y \in X\} \cup \{(x, \varnothing(x)) \mid x \in X\}$.

In principle, this then allows us to solve SPP by solving a variant of Problem 2. We do this by setting $\alpha = 1$ and enforcing in the ILP of [7] that the total cost contributed by $\sum_{(u,v)\in E(T)} \delta |L_u^M \triangle L_v^M|$ in the tree is exactly the minimum calculated beforehand.

However, in practice, it makes sense to further restrict the space of possible adjacencies. To accomplish this in a manner that guarantees at least one optimal solution remains, further theoretical results are necessary. We derive these results in Section 3.2.

Moreover, when we consider the full space of adjacencies, this opens up a way to restrict the space of possible matchings, a technique that could not be used in previous solutions to the problem. We describe the relevant theoretical results in Section 3.3.

Finally, we develop in Section 3.4 how we can use aspects of correctness heuristics like the weights used in Problem 2 as part of our runtime heuristic to obtain good solutions early in the solving process.

3.1 Finding Optimal Marker Frequencies

Since under our distance function a cost of δ is needed for each marker that is not matched to another, changes in copy number of marker m between two linearisations L_u, L_v that are adjacent in the phylogeny ($(u,v) \in T$) imply a cost of $\delta|\phi(L_u, m) - \phi(L_v, m)|$. We can thus infer marker copy numbers under this model, minimising this total cost across the phylogeny. Formally, this is equivalent to a character-state small parsimony problem, where each marker is a character and states are copy numbers, drawn from the alphabet $\mathbb{N}_0$ and the cost function is $c(x,y) = \delta|x-y|$. Since any cost in this model is scaled by δ, we can instead equivalently consider the cost function $c'(x,y) = |x-y|$.

This is then a Wagner Parsimony Problem, for which ancestral copy numbers are easily computed using the Fitch-like Farris optimisation [16]. An efficient, polynomial algorithm to construct all possible character state ranges is given by Swofford and Maddison in [28].

3.2 Optimally Filtering Adjacencies

With the maximum copy number per marker known, we can now build a fully degenerate genome for each internal node that is guaranteed to have all optimal genome labels from $\mathcal{U}$ for that node as linearisations. However, this solution space is much larger than needed for many problem instances. Particularly, since we are interested in phylogenies with a recent history, distances in subtrees close to the leaves are expected to be small. The following lemma allows us to use this fact to restrict the solution space for adjacencies in these cases. To that end, notice that the adjacencies of a genome A are themselves a degenerate genome that permits exactly one linearisation, namely A. Therefore, any Lemma that applies to degenerate genomes also applies to genomes in general.

▶ **Lemma 4.** *Given three (degenerate) genomes D_v, D_u, D_w, at distinct nodes u, v, w, such that $(u,w), (v,w) \in E(T)$ and the maximum distance between any linearisations of D_v and D_u is $d_{u,v}^{\max}$ and $D_u \subset D_w$. For the optimal linearisations L_w^* of D_w and L_v^* of D_v holds $d(L_v^*, L_w^*) \leq d_{u,v}^{\max}$.*

Proof. For the sake of contradiction assume $d(L_v^*, L_w^*) > d_{u,v}^{\max}$. Let $(x,w) \in E(T)$ with $x \neq u, x \neq v$, i.e. x is the third node adjacent to w in the tree with degenerate genome D_x and optimal linearisation L_x^*. The total cost contributed to the tree at node w by L_w^* is then

$d(L_v^*, L_w^*) + d(L_u^*, L_w^*) + d(L_w^*, L_x^*) > d_{u,v}^{\max} + d(L_u^*, L_w^*) + d(L_w^*, L_x^*) \geq d_{u,v}^{\max} + d(L_u^*, L_x^*) = d(L_u^*, L_u^*) + d(L_u^*, L_v^*) + d(L_u^*, L_x^*)$. Therefore, replacing L_w^* with L_u^* as a linearisation of D_w would yield a smaller total cost. L_w^* would thus not be optimal, a contradiction. ◀

Most importantly, we can use this result for two adjacent leaves ("cherries") that have a distance of at most 1.

▶ **Corollary 5.** *For two leaves u, v incident at internal node w, $(u, w), (v, w) \in E(T)$, such that $d(L_u, L_v) \leq 1$. Then the optimal linearisation L_w^* of D_w is L_u or L_v.*

In fact, we can iterate this argument: Since L_w^* is then either L_u or L_v, we can restrict the degenerate genome D_w to only permit L_u or L_v as linearisations. If we then consider two other distinct nodes x, w' with $(x, w), (x, w') \in E(T)$, such that all linearisations of $D_w, D_{w'}$ have distance of at most one, we can restrict the candidate set D_x as well. As long as the distance between genomes is at most 1, we can remove entire subtrees from the problem without sacrificing optimality.

▶ **Corollary 6.** *The optimal linearisation L_r^* of D_r of any subtree E' with root r and leaves L', such that for all $u, v \in L'$ $d(L_v, L_u) \leq 1$ is a genome from $\{L_v \mid v \in L\}$.*

We can then remove all nodes of T' except r from the tree. Moreover, the optimal linearisation for all other nodes can be determined using the Fitch algorithm: In a bottom up pass, linearisation candidates at each internal node are determined and after the optimal linearisation L_r^* is found, a top down pass decides which linearisation candidate is used.

These results help to keep the solution space small when distances are small. However, even when distances are larger, individual adjacencies often stay preserved within a subtree. The next observation allows us to exploit this fact.

▶ **Lemma 7.** *Given (degenerate) genomes D_u, D_v, D_w, D_x for distinct nodes u, v, w, x, such that $(u, w), (v, w), (w, x) \in E$, and such that any optimal linearisations L_u, L_v of D_u and D_v both contain adjacency ab where a, b are singular extremities and D_w is fully degenerate, but any optimal linearisation contains a, b as singular extremities. Then for any optimal linearisation L_w^* of D_w, there is a linearisation L'_w that contains ab and is co-optimal.*

Proof. Regard optimal linearisations $L_u^*, L_v^*, L_w^*, L_x^*$ of D_u, D_w, D_w, D_x. Assume L_w^* does not contain ab, but instead aa' and bb' with $a' \neq b, b' \neq a$. Regard L'_w, which differs from L_w^* by instead containing adjacencies ab and $a'b'$, all other adjacencies being identical to L_w^*. Observe that $d(L'_w, L_x^*) \leq d(L_w^*, L_x^*) + 1$, since L'_w differs from L_w^* by a single DCJ. Moreover, the components in the Relational diagrams between L'_w, L_u^* and L'_w, L_v^* differ from those of L_w^*, L_u^* and L_w^*, L_v^* by only an extra cycle (see Figure 2). This means the term c (see Theorem 1) is increased by one. Since no other component types are changed, all terms based on the graph remain the same. The number of common markers n remains also the same. The only term that could have changed otherwise is s, the number of circular singletons. However observe that if a' and b' are part of a circular singletons, it means their markers incur a cost of δ upon being deleted when transforming L_w^* into L_u^* or L_v^*. Since $\delta|A^M \triangle B^M|$ is the dominant term in d, L_w^* could not have been optimal if they are circular singletons with both L_u^* and L_v^*. Let thus w.l.o.g. $a'b'$ not be a circular singleton for L'_w, L_u^*. Then, $d(L'_w, L_u^*) = d(L_w^*, L_u^*) - 1$ and $d(L'_w, L_v^*) \leq d(L'_w, L_v^*)$. With $d(L'_w, L_x^*) \leq d(L_w^*, L_x^*) + 1$, L'_w is thus at least co-optimal to L_w^*. ◀

Again, this restriction can be propagated through the tree. As long as an adjacency consists of singular extremities and is preserved in the subtree, no alternatives have to be considered. Once duplicates are involved, the solution space is better constrained on the matchings than the adjacencies as we delineate in the next subsection.

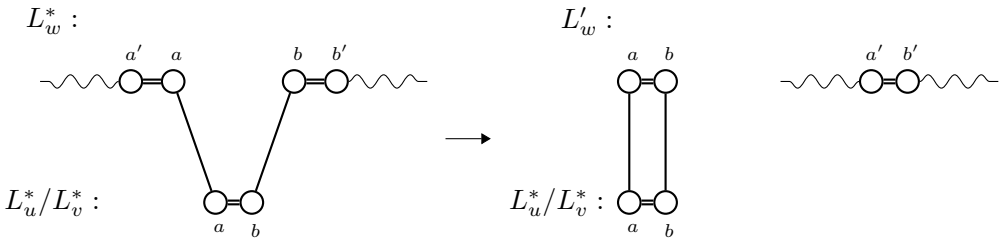

Figure 2 Difference for the Relational Diagrams of L_w^* (top left) and L'_w (top right) with L_v^* or L_u^* (both bottom). Squiggled lines represent arbitrary simple paths. L'_w has an extra cycle and otherwise the same graph component types. Particularly, the type of component that a', b' are in is unchanged.

3.3 Optimally restricting the Matching Space

Note that with duplicates, we are not able to meaningfully restrict the adjacency space unless distances are small (Corollary 6). However, because we are using the full adjacency space, we are able to restrict the number of different matchings. To see how this is possible observe the following pair of labeled genomes: $E_2 = \{[+1_1 - 2_1 + 2_2]\}, E_3 = \{[+1_1 - 2_2 + 2_1]\}$. Both are distinct linearisations of the same fully degenerate genome. However, both linearisations represent the same structure – the distance to any other genome D is the same, $d(E_2, D) = d(E_3, D)$. Only the matching to achieve this distance will be different, i.e. if the matching between E_2 and D contains $(2_1, 2_i)$, the corresponding matching for E_3 and D contains $(2_2, 2_i)$. We can exploit this redundancy to restrict the matching space as follows.

▶ **Lemma 8.** *Let D_u and D_v be degenerate genomes. Let marker m occur k times in D_u and $n \geq k$ times in D_v, such that each occurrence in D_v has the same adjacencies, i.e. $\{m_i^x, g_j^y\} \in D_v \implies \{m_{i'}^x, g_j^y\} \in D_v \forall 1 \leq i' \leq n$. Then for any linearisations L_u, L_v with optimal matching M, there are linearisations L'_u, L'_v, such that their optimal matching M' contains only pairs of the form (m_i, m_i) and $d^s(L_u^M, L_v^M) = d^s(L_u'^{M'}, L_v'^{M'})$.*

Proof. By induction over $x(M) := |\{(m_i, m_j) \in M \mid i \neq j\}|$.

Induction start: Let $x(M) = 0$. Then M is already a matching that conforms to the lemma, i.e. $L'_u = L_u$, $L'_v = L_v$ and $M' = M$.

Induction step: Let $x(M) > 0$. Let $(m_i, m_j) \in M$ with $i \neq j$. Then switch positions of m_i and m_j in L_v to obtain L''_v. Then replace (m_i, m_j) in M by (m_i, m_i) and if $(m_k, m_i) \in M$, replace it by (m_k, m_j), obtaining matching M''. The distance is then the same, $d^s(L_u''^{M''}, L_v''^{M''}) = d^s(L_u^M, l_v^M)$, and $x(M'') \leq x(M) - 1$. ◀

For any internal node u with degenerate genome D_u and n copies of marker m, we can thus find a neighbour $(u, v) \in E(T)$ that has $k \leq n$ copies of m and allow only matchings of the type (m_i, m_i) in the ILP solution presented in [7]. Note that we can do this only once per marker per internal node, that is restricting the matching space for more than one neighbour would lose optimality. For maximum effect, we chose the neighbour with the highest number of matching candidates.

3.4 Solving Strategy

While the weighting for Problem 2 modulates between an optimisation based on weights and an optimisation based on the distance and thus sacrifices optimality for minimising the overall distance in the tree, the problem tends to be easier to solve for $\alpha \to 0$. In fact, in [7], it is shown that finding a linearisation, can be reduced to a maximum weight matching problem, making it polynomial. We can use this situation to find solutions early.

Intuitively, adjacencies that are particularly frequent in a subtree – while not necessarily optimal – are probably not bad candidates as ancestral adjacencies either. We can use these frequencies as weights for Problem 2. Concretely, we use the weighted average of the number of occurrences for an adjacency at an internal node as its weight. We then first optimise versions of Problem 2 with $\alpha < 1$ before finally optimising with $\alpha = 1$. This ensures on hard instances of the problem that good solutions are found early.

A known heuristic strategy for SPP is to iteratively improve a solution by solving median problems at internal nodes [25]. We adapt this strategy to solving subtrees of the original tree first. In this step, the heuristic solution found in the weighted step is projected onto a subtree, used as an initial solution for the DCJ-SPP restricted to this subtree, and the DCJ-SPP for the subtree is solved. This gives us local solutions, and the lower bound as well as "variable hints" for the ILP solver from this subtree can be carried over to the problem on the full tree.

4 Evaluation

We implemented the Swofford-Maddison algorithm to determine marker numbers, the filtering of the solution space and the solving strategies as described in Section 3, integrating the implementation with the ILP developed in [7]. Importantly, we enforce the minimisation of $\delta|A^M \triangle B^M|$ not by including it in the objective function, but by requiring the cost for deleting markers is exactly the minimum determined previously by the Swofford-Maddison algorithm by a series of linear constraints.

We made the software available at `https://github.com/gi-bielefeld/spp_dcj_exact`. We call this software *SPP-DCJ-exact* in the following.

4.1 Simulated Data

We simulated data using Zombi [12]. Since our solution is intended to fit smaller phylogenies with a recent rearrangement history, we used scaled Zombi's default rearrangement parameters by a scale of 0.2, obtaining rates of 0.2 for duplications, 0.6 for gene losses, and 0.4 for inversions, transpositions and gene originations. By default, we simulated 10 lineages with the root genome containing 100 markers, unless specified otherwise. In the following, we will examine the impact of these parameters on SPP-DCJ-exact.

4.1.1 Performance

Comparison with SPP-DCJ. Recall that given the correct marker counts the naive solution to Small Parsimony for Natural Genomes (Problem 1) inputs the entire adjacency space into a linearisation problem optimizing for the total DCJ-indel distance. The latest solution for this problem is *SPP-DCJ-v2*, which consistently outperforms the original SPP-DCJ [7]. Since the space of adjacencies grows primarily with the number of markers, we varied the number of markers at the root genome of the phylogeny for this experiment. To that end, we simulated phylogenies with root genomes of 5 to 30 markers in increments of 5 with 10 samples per step and examined the solving time by gurobi 13.0 set to a time limit of 1 hour. In case no exact solution was found, we instead examined the *relative optimality gap*, that is the ratio $(b - l)/b$ between the current best solution b and the tightest lower bound l found by the solver. Recall that what we optimise for is the minimum number of rearrangements in the tree and thus $l \geq 0$ and a trivial solution is easily found. Therefore, we clamp the optimality gap to be between 0 and 100%. In this experiment, we did not use the solving strategies outlined in Section 3.4, neither for SPP-DCJ-v2 nor for SPP-DCJ-exact. The results are shown in Figure 3.

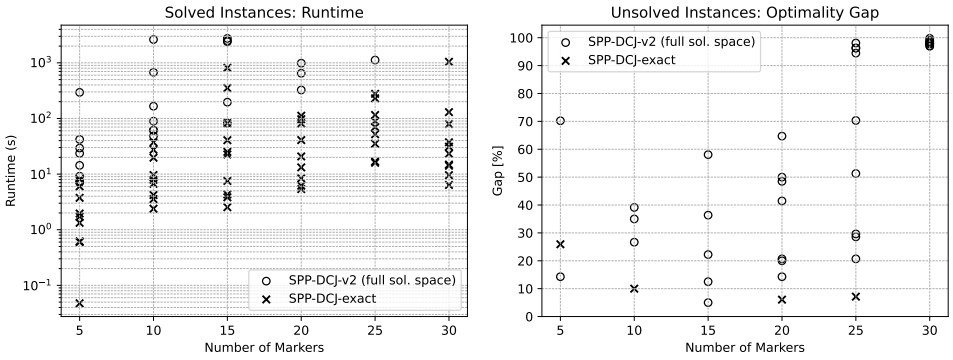

Figure 3 Number of markers vs runtime or optimality gap of SPP-DCJ-v2 and SPP-DCJ-exact as solved by gurobi 13.0 with a time limit of 1 hour (= 3600s). Solved instances are shown on the left, giving their runtime. Instances that exceeded the time limit are shown on the right, instead giving the relative optimality gap reached within 1 hour.

We see that while both ILPs solving times rise with the number of markers, SPP-DCJ-exact outperforms SPP-DCJ-v2, finding exact solutions faster and if no exact solution is found, finding much closer approximations. Notably, almost all SPP-DCJ-v2 instances become unsolvable within the time limit once going above 20 markers, reaching over 90% relative optimality gaps for 30 markers, while the majority of instances of SPP-DCJ-exact remain solvable within 1 hour.

Limiting Behaviour. We examined the limits of which problem instances remain solvable using the strategies as detailed in Section 3.4. We evaluated three different settings: *Default*, in which we optimise using only gurobi with a set time limit; *Weights*, in which we allocate 30% of the time limit to solving the problem for $\alpha = 0.0, 0.1, 0.5$ before a final optimisation with the remaining time for $\alpha = 1$; *Weights+Subtrees*, in which we first perform the same weight based solves for 30% of the time limit and then switch to solve subtrees for 30% of the time limit, after which we run the final optimisation.

We performed three experiments, increasing the number of markers, lineages and rearrangements simulated by Zombi. These experiments are discussed in Appendix A.

To summarise the results, we find a majority of ILPs is solved within an hour up until about 600 markers, 15 lineages and up to 0.3 of the default rearrangement rate in Zombi. We also observe that using the "Weights" or "Weights+Subtree" strategies only starts to positively influence the solution for difficult problem instances and even has an adverse effect for simple instances. Moreover, the "Weights" strategy by itself is typically enough to obtain beneficial effects on optimality and through more time spent on the final optimisation may even outperform the "Weights+Subtrees" strategy.

One possible direction for future research is thus to examine whether these strategies can be improved by using more targeted approach, for example using a specific α or solving specific subtrees as well as allocating different ratios of solving time.

4.1.2 Reconstruction Quality

We simulated genomes using Zombi setting rearrangement rates to $0.2, 0.3, 0.4$ and 0.5 of the default, generating 20 samples per step and ran SPP-DCJ-exact on a single thread with a time limit of 1 hour on each sample. For this experiment, we used the "Weights" strategy to guarantee a solution.

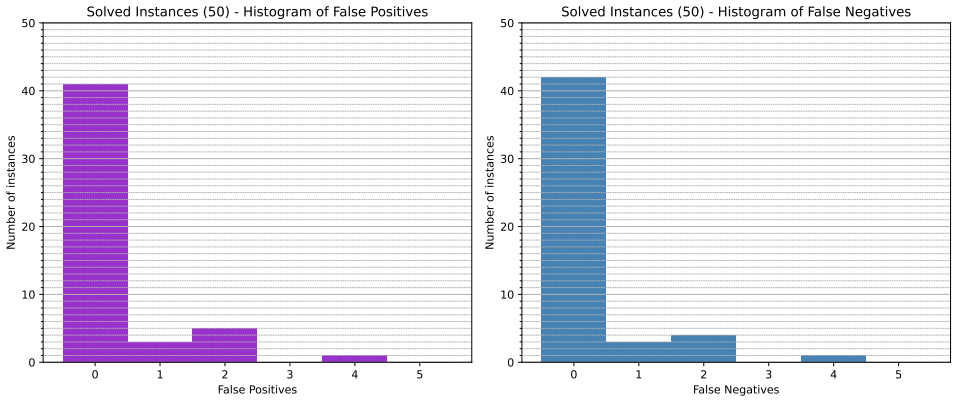

Figure 4 Distribution of errors in reconstructed adjacencies. The histograms show the number of false positives (left) and false negatives (right) on the x-axis giving the number of solved instances with a given number of false positives or false negatives on the y-axis.

We then examined adjacencies at each internal node separately. Since duplicates complicate the comparison of adjacency sets, we limited ourselves in this analysis to adjacencies involving only non-duplicated markers. On average, each ground truth sample contained 780 such adjacencies. We then proceeded to evaluate *false positives* (FP), that is such adjacencies that are found in the reconstruction of SPP-DCJ-exact at a given node, but not in the ground truth simulated by Zombi and *false negatives* (FN), such adjacencies that are found in the ground truth, but not in the reconstruction.

The results for all solved instances are shown in Figure 4. We see that the reconstruction for the majority of samples is perfect – the adjacencies reconstructed by SPP-DCJ-exact correspond directly to adjacencies in the ground truth. In fact, all 41 samples without false positives also had no false negatives. Even when there are false reconstructions, the number of incorrectly reconstructed adjacencies observed in this experiment was at most 4 in a single instance.

For unsolved instances, we instead examined the relation between the relative optimality gap and the reconstruction quality. To that end, we calculated the *true positives* (TP), i.e. the adjacencies that are found in both the ground truth and the reconstruction at a given node. We then calculated *precision* as $TP/(TP + FP)$, *recall* as $TP/(TP + FN)$ and the *F1-score* as $2TP/(2TP + FP + FN)$. The results are visualised in Figure 5.

We see that, as expected, solution quality degenerates with a higher gap. However, overall, the quality of solutions remains high with precision and recall for all samples above 0.95, even when approaching a 100% gap. This suggests that SPP-DCJ-exact, at least when the input data has low to moderate numbers of rearrangements, may find optimal or near-optimal solutions, but that the lower bound does not always tighten fast enough to prove optimality.

4.2 Real Data

We applied our method to a set of 7 pathogenic plasmids from *Klebsiella pneumoniae*, previously characterised by Biggel *et al* in an epidemiological study of plasmids associated with bovine mastisis [1]. Sampling in this study was constrained to diseased Swiss cows over the course of two years, meaning the plasmid lineages within are recently related and relatively small. We selected the lineage "subcommunity 3" from this study, due to it being the most structurally diverse of the plasmid lineages.

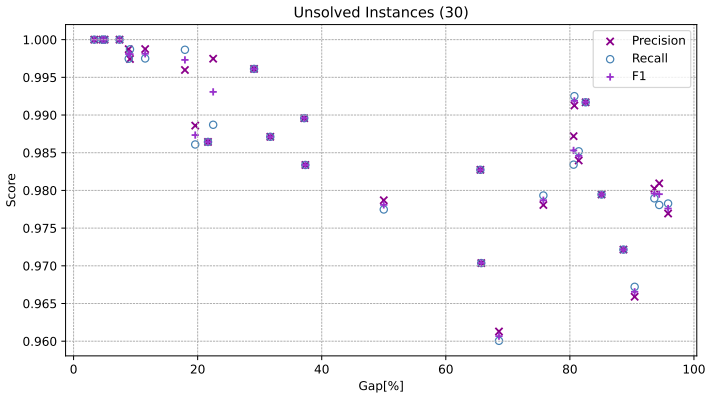

Figure 5 Precision, recall and F1-score in adjacency reconstruction for unsolved instances by relative optimality gap after 1 hour of solving time.

We used MICE [6] to identify synteny blocks to use as markers, using a de Bruijn graph constructed using ggcat [10] with k-mer size 31 as input, and filtering out any synteny blocks below 200 bases. Statistics regarding the input genomes generated in this manner are found in Table 1 of Appendix B. We used the tree topology from Figure 2 in [1] as the input tree. We then applied SPP-DCJ-exact using both weighted and subtree strategies to determine ancestral sequences, using gurobi v11, 8 threads, and setting a time limit of a 12 hours. The best solution encompassed 28 DCJ-indel operations across the tree and was found after 70 minutes. Once the time limit was reached, a lower bound of 21 had been achieved, resulting in a relative optimality gap of 25%. The solution is visualised in Figure 6 with the help of gggenomes package in R [20].

All ancestral sequences are circular and unichromosomal, same as the extant plasmids. Their overall size is similar to that of the extant plasmids and, for cherry (n5(pK-40150-A,pK-75504-A)), the parental sequence n5 is larger than both its children. This demonstrates that in practice SPP-DCJ-exact does not "empty out" ancestral genomes, which is a direct result of restoring the triangle inequality. Additionally, there are multiple duplicate markers throughout the tree, including ones nested in rearrangement events spanning multiple markers, but nonetheless the solution parsimoniously matches these markers. This is particularly important for plasmids, as rearrangements are often associated with transposable elements flanked by duplicate genes [21]. From this solution, it is also possible to see that it is the same regions that are undergoing repeated rearrangements, e.g. the markers at the end of all the sequences are inverted multiple times across the whole tree. This is consistent with observations from previous studies of epidemiologically significant plasmids, which usually rely on manual reconstruction of rearrangement events [22, 31]. Overall, we can see in this example that our approach yields biologically plausible ancestral sequences, which can provide insights into the evolutionary history of a plasmid lineage.

5 Discussion

We introduced a unified framework for solving the Small Parsimony Problem for natural genomes that requires only extant genomes and the underlying tree as input, while still leveraging the efficient encoding of the solution space using degenerate genomes. In addition, we derived initial theoretical results that substantially restrict the solution space, thereby increasing the number of instances that can be solved exactly or approximated with strong guarantees.

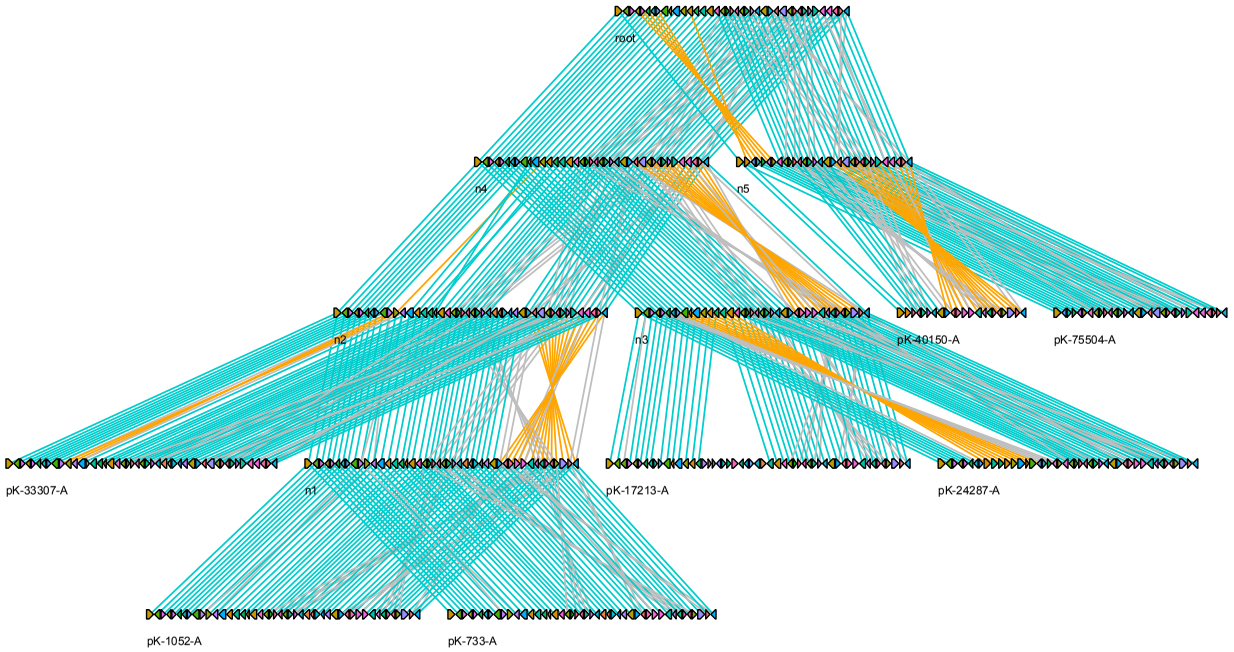

Figure 6 Extant and reconstructed plasmid sequences, arranged according to their tree topology. The same markers across different sequences have the same colour, and marker orientation is shown by arrow direction. Teal links between markers mean they were matched and are oriented the same, orange links between markers mean they were matched and have opposite orientation, while grey lines show links between duplicate markers that were not matched in the final solution. Note that the selection of the root sequence is arbitrary; any intermediary genome between n4 and n5 could be the root, but here we simply choose n4.

When applied to both simulated data and real data sets with a recent evolutionary history, our approach consistently produces high-quality solutions within reasonable running times, even in cases where optimality was not conclusively proven.

The framework presented here also opens several promising directions for future research. So far, we have investigated only a limited number of simple, though effective, methods to constrain the solution space. A more detailed analysis of the underlying distance measure may allow for an even tighter pruning of degenerate genomes. Moreover, it would be interesting to study scenarios in which differences in marker content are not treated as the dominant term in the distance but can instead be traded off with rearrangement operations (i.e. "$\delta < \infty$"). Such an extension could naturally connect our framework to Duplication–Transfer–Loss models of gene tree parsimony, replacing the uniform insertion/deletion cost δ with operation-specific costs that distinguish between duplication, transfer, and loss events. Exploring these connections may further broaden the applicability of our approach.

References

1 Michael Biggel, Magdalena Nüesch-Inderbinen, Mitsuko Logean, Sabrina Corti, Lucien Kelbert, and Roger Stephan. Genomic characterization of klebsiella spp. from bovine mastitis: dissemination of a conserved, highly transmissible lacacq+ fec+ plasmid drives burden of disease. *Applied and Environmental Microbiology*, 91(12):e01162–25, 2025. doi:10.1128/aem.01162-25.

2 Mathieu Blanchette, Takashi Kunisawa, and David Sankoff. Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution*, 49(2):193–203, 1999. doi:10.1007/pl00006542.

3 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.

4 Leonard Bohnenkämper and Daria Frolova. gi-bielefeld/spp_dcj_exact. Software, swhId: swh:1:dir:9d2d2213b28d627478afc8a5f365b31b0d0979f1 (visited on 2026-08-15). URL: https://github.com/gi-bielefeld/spp_dcj_exact, doi:10.4230/artifacts.27684.

5 Leonard Bohnenkämper, Marília 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.

6 Leonard Bohnenkämper, Luca Parmigiani, Cedric Chauve, and Jens Stoye. On deriving synteny blocks by compacting elements. *bioRxiv*, page 2025.12.15.694404, 2026. doi:10.64898/2025.12.15.694404.

7 Leonard Bohnenkämper, Jens Stoye, and Daniel Doerr. Reconstructing rearrangement phylogenies of natural genomes. *Algorithms for Molecular Biology*, 20(1):10, 2025. doi:10.1186/s13015-025-00279-5.

8 Marília 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.

9 Alberto Caprara. Formulations and hardness of multiple sorting by reversals. In *Proceedings of the third annual international conference on Computational molecular biology*, pages 84–93, 1999. doi:10.1145/299432.299461.

10 Andrea Cracco and Alexandru I Tomescu. Extremely fast construction and querying of compacted and colored de bruijn graphs with ggcat. *Genome Research*, 33(7):1198–1207, 2023.

11 Elise Darmon and David R. F. Leach. Bacterial genome instability. *Microbiology and Molecular Biology Reviews*, 78(1):1–39, 2014. doi:10.1128/mmbr.00035-13.

12 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.

13 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.

14 Yucheng Dong, Qi He, Xinyu Chen, Fan Yang, Li He, and Yongchang Zheng. Extrachromosomal DNA (ecDNA) in cancer: mechanisms, functions, and clinical implications. *Frontiers in Oncology*, 13:1194405, 2023. doi:10.3389/fonc.2023.1194405.

15 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.

16 James S Farris. Methods for computing Wagner trees. *Systematic zoology*, pages 83–92, 1970.

17 Pedro Feijao and Joao Meidanis. Scj: A breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1318–1329, 2011. doi:10.1109/TCBB.2011.34.

18 Walter M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971. doi:10.1093/sysbio/20.4.406.

19 Daria Frolova. *Studying plasmid relatedness and evolution through structural variation*. PhD thesis, Apollo - University of Cambridge Repository, 2025. doi:10.17863/CAM.123913.

20 Thomas Hackl, Markus Ankenbrand, Bart van Adrichem, David Wilkins, and Kristina Haslinger. gggenomes: effective and versatile visualizations for comparative genomics. *arXiv*, 2024. doi:10.48550/arxiv.2411.13556.

21 Christopher J. Harmer, Robert A. Moran, and Ruth M. Hall. Movement of IS26-associated antibiotic resistance genes occurs via a translocatable unit that includes a single IS26 and preferentially inserts adjacent to another IS26. *mBio*, 5(5):10.1128/mbio.01801–14, 2014. doi:10.1128/mbio.01801-14.

22 Timothy J. Johnson, Jessica L. Danzeisen, Bonnie Youmans, Kyle Case, Katharine Llop, Jeannette Munoz-Aguayo, Cristian Flores-Figueroa, Maliha Aziz, Nicole Stoesser, Evgeni Sokurenko, Lance B. Price, and James R. Johnson. Separate f-type plasmids have shaped the evolution of the h30 subclone of escherichia coli sequence type 131. *mSphere*, 1(4):10.1128/msphere.00121–16, 2016. doi:10.1128/msphere.00121-16.

23 Ankit Malhotra, Michael Lindberg, Gregory G. Faust, Mitchell L. Leibowitz, Royden A. Clark, Ryan M. Layer, Aaron R. Quinlan, and Ira M. Hall. Breakpoint profiling of 64 cancer genomes reveals numerous complex rearrangements spawned by homology-independent mechanisms. *Genome Research*, 23(5):762–776, 2013. doi:10.1101/gr.143677.112.

24 William Matlock, Liam P Shaw, Samuel K Sheppard, and Edward Feil. Towards quantifying plasmid similarity. *Microbial Genomics*, 10(9), 2024. doi:10.1099/mgen.0.001290.

25 David Sankoff and Mathieu Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *Journal of computational biology*, 5(3):555–570, 1998. doi:10.1089/CMB.1998.5.555.

26 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. PMID: 25517208. doi:10.1089/cmb.2014.0096.

27 Eva H. Stukenbrock and Daniel Croll. The evolving fungal genome. *Fungal Biology Reviews*, 28(1):1–12, 2014. doi:10.1016/j.fbr.2014.02.001.

28 David L. Swofford and Wayne P. Maddison. Reconstructing ancestral character states under Wagner parsimony. *Mathematical Biosciences*, 87(2):199–229, 1987. doi:10.1016/0025-5564(87)90074-5.

29 Eric Tannier, Chunfang Zheng, and David Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, 10(1):120, April 2009. doi:10.1186/1471-2105-10-120.

30 Gerry Tonkin-Hill, Jukka Corander, and Julian Parkhill. Challenges in prokaryote pangenomics. *Microbial Genomics*, 9(5):mgen001021, 2023. doi:10.1099/mgen.0.001021.

31 Ethan R. Wyrsch, Rhys N. Bushell, Marc S. Marenda, Glenn F. Browning, and Steven P. Djordjevic. Global phylogeny and f virulence plasmid carriage in pandemic escherichia coli ST1193. *Microbiology Spectrum*, 10(6):e02554–22, 2022. doi:10.1128/spectrum.02554-22.

32 Wei Xu, Daniel Jameson, Bin Tang, and Paul G. Higgs. The relationship between the rate of molecular evolution and the rate of genome rearrangement in animal mitochondrial genomes. *Journal of Molecular Evolution*, 63(3):375–392, 2006. doi:10.1007/s00239-005-0246-5.

A Evaluating Limiting Behaviour of Solving Strategies

To assess the performance of the three strategies introduced in Section 4.1.1 (*Default*, *Weights*, and *Weights+Subtrees*), we evaluated their relative optimality gap under a time limit of one hour. We considered three parameters: (i) the number of markers in the root genome, (ii) the number of lineages, and (iii) the number of rearrangements. For each parameter configuration, we generated 10 samples using Zombi. We solved each problem instance with all three strategies, each running on a single thread.

In the first experiment, we increased the number of markers in the root genome from 200 to 1000 in increments of 100. As shown in Figure 7 (A), the optimality gap increases with the number of markers for all three strategies. For small genomes (200–500 markers), most instances are solved optimally within the time limit. Somewhat unexpectedly, the *Default* strategy performed best in this setting, outperforming both *Weights* and *Weights+Subtrees*. A possible explanation is that, for larger genomes, the weighted and subtree-based approaches generate subproblems that themselves become computationally expensive. The additional overhead needed to solve these subproblems may thus "absorb" solving time that would be better used solving the original problem.

In the second experiment, we increased the number of lineages from 11 to 20 in increments of 1. The results are visualised in Figure 7 (B). Across all strategies, the variance in the observed optimality gaps is high. Up to 15 lineages, the majority of instances are solved optimally within the time limit. For 19 and 20 lineages onwards, the *Weights* strategy begins to show an improvement over *Default*. The *Weights+Subtrees* strategy surpasses *Default* only for some instances with a high number of lineages. Notably, however, it exhibits a considerably smaller variance in optimality gaps than both other strategies at these higher values, indicating more stable performance on particularly difficult instances.

In the third experiment, we varied the number of rearrangements. To that end, we scaled duplication rates by x, gene loss rates by $3x$, and other rearrangement rates by $2s$, increasing s from 0.2 to 1.0 in increments of 0.1. We call s *rearrangement scale* in the following. As shown in Figure 7 (C), the optimality gap increased sharply with growing rearrangement rates. Most problem instances are solved optimally for $s \leq 0.4$. From $s = 0.7$ onwards, the *Default* strategy almost exclusively exhibits gaps of 100%, indicating that either no solution was found and/or the lower bound could not be tightened further. In the intermediate range $s = 0.6$ to 0.8, *Weights+Subtrees* outperforms *Weights*, but only on some instances. For higher rates of rearrangements, however, the relative optimality gap of all strategies approaches 100%, suggesting the overall time limit of 1 hour is not enough to obtain any significant information about a given problem instance.

Overall, these results indicate that *Weights* and *Weights+Subtrees* are primarily beneficial strategies for solving difficult problem instances. For simple problem instances, the additional preprocessing and decomposition effort appears unnecessary and may even reduce performance. This suggests that further fine-tuning is required, in particular mechanisms for identifying in advance whether solving particular subtrees or applying a weighted optimisation first is likely to improve performance.

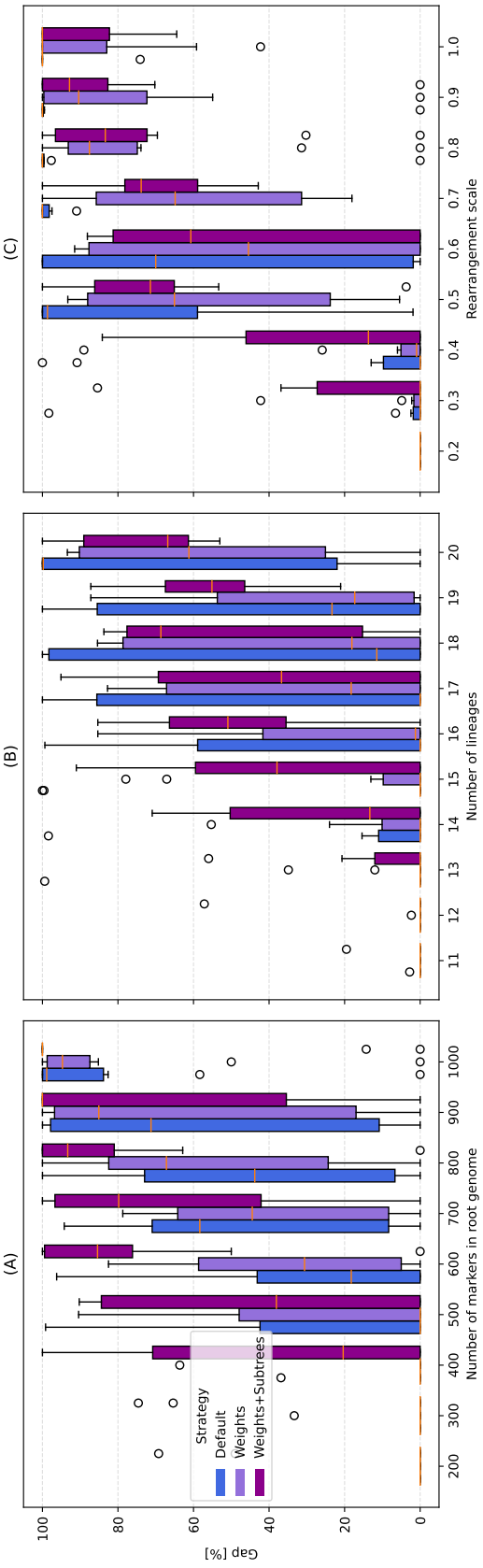

Figure 7 Distribution of optimality gaps for three solving strategies (Default, Weights and Weights+Subtrees) after 1 hour of solving time using gurobi 13.0 in three experiments simulated with Zombi. Relative optimality gaps are visualised as a box plot. (A) The number of markers in the root genome is increased from 200 to 1000 in steps of 100. (B) The number of simulated lineages is increased from 11 to 20. (C) The default rearrangement rate is scaled from 0.2 to 1.0 in increments of 0.1. For each step in each of the experiments 10 samples were generated.

B Plasmid Marker Statistics

Table 1 Summarising statistics of the MICE output used for marker definition. The first column is the plasmid sequence ID, followed by the number of marker occurrences, followed by number of singular markers, and finally the number of duplicate markers.

seq_id	length	# singular markers	# duplicate markers
pK-75504-A	31	15	8
pK-1052-A	47	29	9
pK-33307-A	47	29	9
pK-40150-A	22	8	7
pK-24287-A	45	19	13
pK-733-A	46	28	9
pK-17213-A	53	31	11