Pairwise Rearrangement is Fixed-Parameter Tractable in the Single Cut-and-Join Model

Lora Bailey
Department of Mathematics, Grand Valley State University, Allendale, MI, USA

Heather Smith Blake
Department of Mathematics and Computer Science, Davidson College, NC, USA

Garner Cochran
Department of Mathematics and Computer Science, Berry College, Mount Berry, GA, USA

Nathan Fox
Department of Quantitative Sciences, Canisius University, Buffalo, NY, USA

Michael Levet[1]
Department of Computer Science, College of Charleston, SC, USA

Reem Mahmoud
Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

Inne Singgih
Department of Mathematical Sciences, University of Cincinnati, OH, USA

Grace Stadnyk
Department of Mathematics, Furman University, Greenville, SC, USA

Alexander Wiedemann
Department of Mathematics, Randolph–Macon College, Ashland, VA, USA

Abstract

Genome rearrangement is a common model for molecular evolution. In this paper, we consider the Pairwise Rearrangement problem, which takes as input two genomes and asks for the number of minimum-length sequences of permissible operations transforming the first genome into the second. In the Single Cut-and-Join model (Bergeron, Medvedev, & Stoye, *J. Comput. Biol.* 2010), Pairwise Rearrangement is #P-complete (Bailey, et. al., COCOON 2023), which implies that exact sampling is intractable. In order to cope with this intractability, we investigate the parameterized complexity of this problem. We exhibit a fixed-parameter tractable algorithm with respect to the number of components in the *adjacency graph* that are not cycles of length 2 or paths of length 1. As a consequence, we obtain that Pairwise Rearrangement in the Single Cut-and-Join model is fixed-parameter tractable by distance. Our results suggest that the number of nontrivial components in the adjacency graph serves as the key obstacle for efficient sampling.

2012 ACM Subject Classification Theory of computation → Complexity classes; Mathematics of computing → Graph theory

Keywords and phrases Genome Rearrangement, Phylogenetics, Single Cut-and-Join, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.3

Related Version *Full Version*: `https://arxiv.org/abs/2402.01942`

Acknowledgements We wish to thank the American Mathematical Society for organizing the Mathematics Research Community workshop where this work began. This material is based upon work supported by the National Science Foundation under Grant Number DMS 1641020.

[1] Corresponding author

1 Introduction

With the natural occurrence of mutations in genomes and the wide range of effects this can incite, scientists seek to understand the evolutionary relationship between species. Several discrete mathematical models have been proposed to model these mutations based on biological observations. Genome rearrangement models consider situations in which large-scale mutations alter the order of the genes within the genome. Sturtevant [17, 18] observed the biological phenomenon of genome rearrangement in the study of strains of *Drosophila* (fruit flies), only a few years after he produced the first genetic map [16]. Palmer & Herbon [15] observed similar phenomenon in plants. McClintock [10] also found experimental evidence of genes rearranging themselves, or "transposing" themselves, within chromosomes.

Subsequent to his work on *Drosophila*, Sturtevant together with Novitski [19] introduced one of the first genome rearrangement problems, seeking a minimum length sequence of operations, called a *scenario*, that would transform one genome into another. In investigating these questions, it is of key importance to balance biological relevance with computational tractability. One central issue is that of combinatorial explosion: the number of scenarios even between small genomes may be too large to handle, making it difficult to test hypotheses on all possible scenarios. Thus, we desire a polynomial time algorithm to uniformly sample from the rearrangement scenarios. Since uniform sampling is no harder than exact enumeration [8], we investigate the computational complexity of the Pairwise Rearrangement problem (Definition 5) which asks for the number of minimum-length scenarios transforming one genome into another.

The Pairwise Rearrangement problem has received significant attention in several genome rearrangement models. Pairwise Rearrangement is known to be in FP for the Single Cut or Join model [11], but is conjectured to be #P-complete for the Double Cut-and-Join model [14]. The Single Cut-and-Join model sits between these two models. Recently, Bailey, et al. [1], showed that Pairwise Rearrangement is #P-complete in the Single Cut-and-Join model. However, in practice, the key structures that serve as obstacles to efficient sampling may not necessarily appear. In particular, the relevant obstacles for efficient sampling in the Single Cut-and-Join model remain opaque.

Main Results. In this paper, we investigate the Pairwise Rearrangement problem in the Single Cut-and-Join model through the lens of parameterized complexity. Our main result is the following.

▶ **Theorem 1.** *In the Single Cut-and-Join model, Pairwise Rearrangement is fixed-parameter tractable with respect to the number of components in the adjacency graph (see Definition 6) that are not trivial (cycles of length* 2 *or paths of length* 1*).*

Our parameterization in Theorem 1 has biological significance. Indeed, chromoanagenesis is a carcinogenic mechanism that involves massive chromosomal rearrangements, which may lead to fewer components in the adjacency graph [7].

The *adjacency graph* (see Definition 6) is a bipartite multigraph illustrating where two genomes differ. Bergeron, Medvedev, and Stoye [2] established a precise relationship between the adjacency graph and the distance between two genomes in the Single Cut-and-Join model. The operations in this model induce structural changes on the adjacency graph [1, Observation 2.7]. We leverage this crucially to establish Theorem 1. Our precise technique involves developing a dynamic programming algorithm that, when the number of nontrivial components is bounded, the corresponding lookup table only has a polynomial number of entries. This establishes our claim of polynomial-time computation. We stress that arriving at the recurrence relations for the dynamic programming algorithm and proving their correctness is technical and nontrivial. Indeed this is not surprising, as Pairwise Rearrangement is #P-complete [1]. While our work is theoretical in nature, the algorithm underlying Theorem 1 in fact yields an efficient implementation (see GitHub).

We also note that if the distance (Equation (1)) between the two genomes is bounded [2], then so is the number of nontrivial components. As a consequence, we obtain the following corollary:

▶ **Corollary 2.** *In the Single Cut-and-Join model, Pairwise Rearrangement is fixed parameter tractable with respect to the distance between two genomes.*

To the best of our knowledge, parameterized complexity has received minimal attention in the genome rearrangement literature. For instance, a fixed-parameter tractable algorithm (parameterized by the number of components in the adjacency graph) for Pairwise Rearrangement in the Double Cut-and-Join model can easily be deduced from the work of [3], though the authors do not explicitly investigate the parameterized complexity of this problem. Thus, beyond providing a means of coping with the intractability of Pairwise Rearrangement in the Single Cut-and-Join model, Theorem 1 (together with the results of [3]) makes precise that the number of components in the adjacency graph serves as a key obstacle towards efficient sampling, across multiple models of genome rearrangement.

In contrast, there has been significant algorithmic work on approximation and sampling (see, for instance, [11, 12, 4, 5, 13, 9]), to cope with the intractability of enumeration. To the best of our knowledge, such approaches have not been fruitful against the Reversal model, for which the complexity of Pairwise Rearrangement is a longstanding open problem. We are not aware of any work on approximate counting or sampling for Pairwise Rearrangement in the Single Cut-and-Join model.

2 Preliminaries

We recall preliminaries regarding genome rearrangement.

▶ **Definition 3.** *A* genome *is an edge-labeled directed graph in which each label is unique and the total degree of each vertex is 1 or 2 (in-degree and out-degree combined). In particular, a genome consists of disjoint paths and cycles. The weak components of a genome we call* chromosomes*. Each edge begins at its* tail *and ends at its* head*, collectively referred to as its* extremities*. Degree 2 vertices are called* adjacencies*, and degree 1 vertices are called* telomeres*. See Figure 1.*

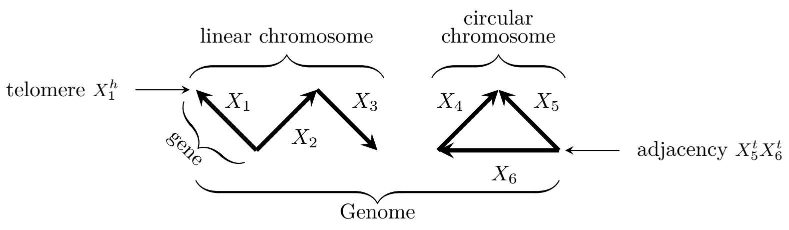

Figure 1 An edge-labeled genome [1, Fig. 1].

Adjacencies can be viewed as sets of two extremities, and telomeres as sets containing exactly one extremity. For simplicity, we write adjacency $\{a, b\}$ as ab or ba and telomere $\{c\}$ as c. For example, the adjacency $X_5^tX_6^t$ in Figure 1 denotes that the tail of the edge X_5 and the tail of the edge X_6 meet, and the telomere X_1^h is where the edge X_1 ends. Each genome is then uniquely defined by its set of adjacencies and telomeres.

Consider the following operations on a given genome:

(i) *Cut*: an adjacency ab is separated into two telomeres, a and b,

(ii) *Join*: two telomeres a and b become one adjacency, ab,

(iii) *Cut-join*: adjacency ab and telomere c are replaced with adjacency ac and telomere b, and

(iv) *Double-cut-join*: adjacencies ab and cd are replaced with adjacencies ac and bd.

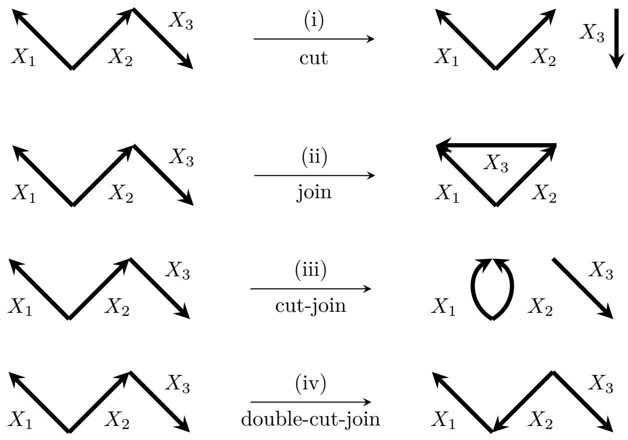

Figure 2 (i) Adjacency $X_2^hX_3^t$ is cut. (ii) Telomeres X_1^h and X_3^h are joined. (iii) Adjacency $X_2^hX_3^t$ is cut, and resulting telomere X_2^h is joined with X_1^h. (iv) Adjacencies $X_1^tX_2^t$ and $X_2^hX_3^t$ are replaced with $X_1^tX_2^h$ and $X_2^tX_3^t$.

Note that a cut-join operation combines one cut and one join into a single operation, and a double-cut-join operation performs two cuts and two joins in one operation. See Figure 2 [1] for an illustration of these operations.

Several key models are based on these operations. The *Double Cut-and-Join (DCJ)* model was initially introduced by Yancopoulos, Attie, & Friedberg [20] and permits all four operations. Later, Feijao & Meidanis [6] introduced the *Single Cut-or-Join (SCoJ)* model, which only allows operations (i) and (ii). Alternatively, the *Single Cut-and-Join (SCaJ)* model [2] allows operations (i)-(iii), but not operation (iv). In this paper, we consider the Single Cut-and-Join model.

▶ **Definition 4.** *For any two genomes G_1 and G_2 with the same set of edge labels, there is a sequences of Single Cut-and-Join operations that transforms G_1 into G_2. Such a sequence is called a* scenario*. The minimum possible length of such a scenario is called the* distance *and is denoted $d(G_1, G_2)$. An operation on a genome G_1 that (strictly) decreases the distance to genome G_2 is called a* sorting operation *for G_1 and G_2. A scenario requiring $d(G_1, G_2)$ operations to transform G_1 into G_2 is called a* most parsimonious scenario *or* sorting scenario*. When G_2 is understood, we refer to the action of transforming G_1 into G_2 using the minimum number of operations as* sorting G_1*. The number of most parsimonious scenarios transforming G_1 into G_2 is denoted $\#MPS(G_1, G_2)$.*

We now turn to defining the key algorithmic problem that we will consider in this paper.

▶ **Definition 5.** *Let G_1 and G_2 be genomes. The* Distance *problem asks to compute $d(G_1, G_2)$. The* Pairwise Rearrangement *problem asks to compute $\#MPS(G_1, G_2)$.*

To investigate Pairwise Rearrangement, we begin by introducing the adjacency graph.

▶ **Definition 6.** *Given two genomes G_1 and G_2 with the same set of edge labels, the* adjacency graph *$A(G_1, G_2)$ is a bipartite undirected multigraph $(V_1 \dot\cup V_2, E)$ where the vertices in V_i are the adjacencies and telomeres in G_i and for any $X \in V_1$ and $Y \in V_2$, the number of edges between X and Y is $|X \cap Y|$.*

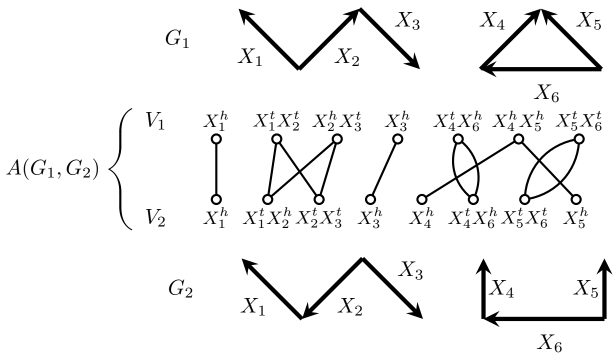

Figure 3 (Taken from [1].) The adjacency graph $A(G_1, G_2)$ is shown in the middle, for genomes G_1 and G_2 shown above and below, respectively.

Note that each vertex in an adjacency graph $A(G_1, G_2)$ must have either degree 1 or 2 (corresponding, respectively, to telomeres and adjacencies in the original genome), and so $A(G_1, G_2)$ is composed entirely of disjoint cycles and paths. Note also that every operation on G_1 corresponds to an operation on V_1 in $A(G_1, G_2)$. For example, in Figure 3 the cut operation on G_1 which separates adjacency $X_4^h X_5^h$ into telomeres X_4^h and X_5^h equates to separating the corresponding vertex $X_4^h X_5^h$ in V_1 into two vertices X_4^h and X_5^h, thus splitting the path of length 2 in $A(G_1, G_2)$ into two disjoint paths of length 1. In a similar fashion, a join operation on G_1 corresponds to combining two vertices a, b in V_1 into a single vertex ab, and a cut-join operation on G_1 corresponds to replacing vertices ab, c in V_1 with vertices ac, b. Whether or not an operation on $A(G_1, G_2)$ corresponds to a sorting operation on G_1 – that is, whether it decreases the distance to G_2 or not – depends highly on the structure of the components acted on. To better describe such sorting operations, we adopt the following classification of components in $A(G_1, G_2)$ and notion of their size:

▶ **Definition 7.** *The possible connected components of $A(G_1, G_2)$ are classified as follows:*
- *A W-shaped component is an even path with its two endpoints in V_1.*
- *An M-shaped component is an even path with its two endpoints in V_2.*
- *An N-shaped component is an odd path.*
- *A* crown *is an even cycle.*

▶ **Definition 8.** *The* size *of a component B in $A(G_1, G_2)$ is defined to be $\lfloor |E(B)|/2 \rfloor$. We refer to an N-shaped component of size 0 (a single edge) as a* trivial path*, and a crown of size 1 (a 2-cycle) as a* trivial crown.

The language "trivial" is motivated by the fact that such components indicate where G_1 and G_2 already agree, and hence no sorting operations are required on vertices belonging to trivial components (see, e.g., the trivial components in Figure 3). Indeed, a sorting scenario can be viewed as a minimal length sequence of operations which produces an adjacency graph consisting of only trivial components.

▶ **Observation 9** ([1, Observation 2.7]). *In the SCaJ model, a case analysis of all operations yields precisely these as the only sorting operations on $A(G_1, G_2)$:*

(a) *A cut-join operation on a nontrivial N-shaped component, producing an N-shaped component and a trivial crown*

(b) *A cut-join operation on a W-shaped component of size at least 2, producing a trivial crown and a W-shaped component*

(c) *A join operation on a W-shaped component of size 1, producing a trivial crown*

(d) *A cut operation on an M-shaped component, producing two N-shaped components*

(e) *A cut operation on a nontrivial crown, producing a W-shaped component*

(f) *A cut-join operation on an M-shaped component and a W-shaped component, where an adjacency in the M-shaped component is cut and joined to a telomere in the W-shaped component, producing two N-shaped components*

(g) *A cut-join operation on a nontrivial crown and an N-shaped component, where an adjacency in the crown is cut and joined to the telomere in the N-shaped component, producing an N-shaped component*

(h) *A cut-join operation on a nontrivial crown and a W-shaped component, where an adjacency from the crown is cut and joined with a telomere from the W-shaped component, producing a W-shaped component*

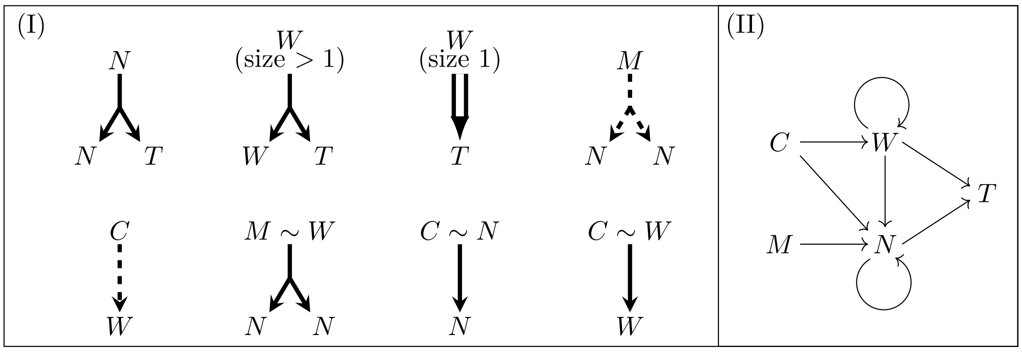

Figure 4 This figure depicts the operations described in Observation 9, where the arrows point from the original component type(s) to the component type(s) produced by the three operations allowed in the SCaJ model: cut, join, and cut-join. The eight diagrams in (I) show each of the sorting operations (a)-(h) where bold single arrows represent cut-join operations, dashed arrows represent cut operations, and the double arrow represents the join operation. Diagram (II) summarizes which components can be produced. Note that all operations will only result in W-shaped components, N-shaped components, and/or trivial crowns. Here, T denotes a trivial crown and C denotes a non-trivial crown.

Note that (a) - (e) are sorting operations on G_1 that operate on only one component in the adjacency graph, though they may produce two different components. On the other hand, (f) - (h) are sorting operations on G_1 that operate on two separate components in the adjacency graph. See Figure 4 for a visualization of each sorting operation.

Using these sorting operations on $A(G_1, G_2)$, the distance between two genomes G_1 and G_2 for the SCaJ model is given by

$$d(G_1, G_2) = n - \frac{\#N}{2} - \#T + \#C \qquad (1)$$

where n is the number of edges in G_1 (equivalently, one half of the number of edges in $A(G_1, G_2)$), $\#N$ is the number of N-shaped components, $\#T$ is the number of trivial crowns, and $\#C$ is the number of nontrivial crowns [2].

Let $\mathcal{B}$ be the set of all components of $A(G_1, G_2)$ and let $\mathcal{B}'$ be a subset of $\mathcal{B}$. Define

$$d(\mathcal{B}') := \left(\sum_{B\in\mathcal{B}'} \text{size}(B)\right) - \#T_{\mathcal{B}'} + \#C_{\mathcal{B}'} \tag{2}$$

where $\#T_{\mathcal{B}'}$ and $\#C_{\mathcal{B}'}$ are the number of trivial crowns and nontrivial crowns in $\mathcal{B}'$, respectively. The quantity $d(\mathcal{B}')$ is the minimum number of operations needed to transform all components of $\mathcal{B}'$ into trivial components, with no operation acting on a component not belonging to $\mathcal{B}'$. Note that $d(\mathcal{B}) = d(G_1, G_2)$, as the $\frac{\#N}{2}$ term is absorbed into the summation of the sizes of all components.

▶ **Definition 10** ([1, Definition 2.8]). *Let A and B be components of an adjacency graph, and consider a particular sorting scenario. We say $A \sim B$ if either $A = B$ or there is a cut-join operation in the scenario where an extremity a from A and an extremity b from B are joined into an adjacency. The transitive closure of $\sim$ is an equivalence relation which we call* sort together*. We will be particularly interested in subsets of the equivalence classes of "sort together." We abuse terminology by referring to such a subset as a set that* sorts together*.*

Note that if two components A, B in $A(G_1, G_2)$ satisfy $A \sim B$, the cut-join operation does not need to occur immediately. For example, two nontrivial crowns C_1 and C_2 can satisfy $C_1 \sim C_2$ by first cutting C_1 to produce a W-shaped component, then operation (b) can be applied multiple times before operation (h) sorts C_2 and the remaining W-shaped component together; see Figure 4.

We will now recall additional notation from [1] that we will use in this paper. For a subset $\mathcal{B}'$ of $\mathcal{B}$, define $\#MPS(\mathcal{B}')$ as the number of sequences with $d(\mathcal{B}')$ operations in which the components of $\mathcal{B}'$ are transformed into trivial components with no operation acting on a component not belonging to $\mathcal{B}'$. Note that $\#\text{MPS}(\mathcal{B})$ is the number of most parsimonious scenarios transforming G_1 into G_2. Let $\#\text{ST}(\mathcal{B}')$ denote the number of sequences with $d(\mathcal{B}')$ operations in which the components of $\mathcal{B}'$ sort together and are transformed into trivial components with no operation acting on a component not belonging to $\mathcal{B}'$.

Note that if $\mathcal{B}'$ and $\mathcal{B}''$ are two subsets of $\mathcal{B}$ that have all the same component types with all the same sizes, then $\#MPS(\mathcal{B}') = \#MPS(\mathcal{B}'')$ and $\#\text{ST}(\mathcal{B}') = \#\text{ST}(\mathcal{B}'')$. Going forward, we will often care about values of $\#\text{MPS}$ and $\#\text{ST}$ only in the context of their component types and sizes. Suppose we are given multisets $\mathcal{C}$, $\mathcal{M}$, $\mathcal{W}$, and $\mathcal{N}$ of nonnegative integers with every element of $\mathcal{C}$ at least 2 and every element of $\mathcal{M}$ and $\mathcal{W}$ at least 1. We define $\#\text{MPS}(\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N})$ to equal $\#\text{MPS}(\mathcal{B}')$ for any set of components $\mathcal{B}'$ with $|\mathcal{C}|$ nontrivial crowns of sizes in $\mathcal{C}$, $|\mathcal{M}|$ M-shaped components of sizes in $\mathcal{M}$, $|\mathcal{W}|$ W-shaped components of sizes in $\mathcal{W}$, and $|\mathcal{N}|$ N-shaped components of sizes in $\mathcal{N}$. We define $\#\text{ST}(\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N})$ similarly.

3 Combinatorics of Genome Rearrangement

In this section, we will prove Theorem 1. Our strategy will be to build a lookup table for dynamic programming. In particular, our technique relies crucially on the following lemma.

▶ **Lemma 11** ([1, Lemma A.4]). *Let $\mathcal{B}'$ be a subset of components of an adjacency graph, and let $\Pi(\mathcal{B}')$ denote the set of all partitions of $\mathcal{B}'$. We have*

$$\#MPS(\mathcal{B}') = \sum_{\pi\in\Pi(\mathcal{B}')} \binom{d(\mathcal{B}')}{d(\pi_1), d(\pi_2), \dots, d(\pi_{p(\pi)})} \prod_{i=1}^{p(\pi)} \#ST(\pi_i),$$

where $\pi = \{\pi_1, \pi_2, \dots, \pi_{p(\pi)}\}$.

We will utilize Lemma 11 in the following manner. Fix an entry in the lookup table, and let $\mathcal{B}'$ denote the set of components being considered at said entry. Now fix a partition $\pi \in \Pi(\mathcal{B}')$. In order to compute $\#\mathrm{MPS}(\mathcal{B}')$, we will proceed as follows. For each $i \in [p(\pi)]$, we first check if $\#\mathrm{ST}(\pi_i) = 0$. This step is computable in polynomial-time by checking whether there exists a permissible sorting operation (see Observation 9). If $\#\mathrm{ST}(\pi_i) \neq 0$, then we access the entry in the lookup table for $\#\mathrm{ST}(\pi_i)$. This will allow us to compute

$$\binom{d(\mathcal{B}')}{d(\pi_1), d(\pi_2), \ldots, d(\pi_{p(\pi)})} \prod_{i=1}^{p(\pi)} \#\mathrm{ST}(\pi_i).$$

We will show later that as the number of nontrivial crowns in the adjacency graph is bounded (by assumption), there are only a polynomial number of partitions $\pi = (\pi_1, \ldots, \pi_{p(\pi)}) \in \Pi(\mathcal{B}')$ such that $\#\mathrm{ST}(\pi_i) \neq 0$ for all $i \in [p(\pi)]$.

We will now investigate how to compute $\#\mathrm{ST}(\pi_i)$, which requires studying which sets of components can and cannot sort together.

▶ **Proposition 12.** *Given a set $\mathcal{B}'$ of components of some adjacency graph G, $\#\mathrm{ST}(\mathcal{B}') = 0$ if $\mathcal{B}'$ has any of the following properties:*

1. *$\mathcal{B}'$ contains at least two components, at least one of which is a trivial crown.*
2. *$\mathcal{B}'$ contains more than one W-shaped component.*
3. *$\mathcal{B}'$ contains more than one M-shaped component.*
4. *$\mathcal{B}'$ contains more than one path and at least one N-shaped component.*

Proof. We refer to Observation 9 and Figure 4 for the permissible sorting operations, from which the proof essentially follows. We provide full details in the proof of Proposition 3.2 in the full version. ◀

If $\mathcal{B}'$ consists of only one trivial crown, then $\#\mathrm{ST}(\mathcal{B}') = 1$. Otherwise there are five possibilities for when $\#\mathrm{ST}(\mathcal{B}') \neq 0$. Below we list each case and define a function along with each that we will use to simplify $\#\mathrm{ST}(\mathcal{C}, M, W, N)$. In what follows, $\mathcal{Z}_{\geq 2}$ denotes the set of finite multisets of integers in which each integer is at least 2. Also, for this paper we take $\mathbb{N}$ to include 0. We will now list our cases:

- The components are a single N-shaped component and zero or more nontrivial crowns. Define $\#\mathrm{ST}_N : \mathcal{Z}_{\geq 2} \times \mathbb{N} \to \mathbb{N}$ as $\#\mathrm{ST}_N(\mathcal{C}, \eta) = \#\mathrm{ST}(\mathcal{C}, \emptyset, \emptyset, \{\eta\})$.
- The components are a single W-shaped component and zero or more nontrivial crowns. Define $\#\mathrm{ST}_W : \mathcal{Z}_{\geq 2} \times \mathbb{Z}^+ \to \mathbb{N}$ as $\#\mathrm{ST}_W(\mathcal{C}, w) = \#\mathrm{ST}(\mathcal{C}, \emptyset, \{w\}, \emptyset)$.
- The components are one or more nontrivial crowns. Define $\#\mathrm{ST}_C : \mathcal{Z}_{\geq 2} - \{\emptyset\} \to \mathbb{N}$ as $\#\mathrm{ST}_C(\mathcal{C}) = \#\mathrm{ST}(\mathcal{C}, \emptyset, \emptyset, \emptyset)$.
- The components are a single M-shaped component and zero or more nontrivial crowns. Define $\#\mathrm{ST}_M : \mathcal{Z}_{\geq 2} \times \mathbb{Z}^+ \to \mathbb{N}$ as $\#\mathrm{ST}_M(\mathcal{C}, m) = \#\mathrm{ST}(\mathcal{C}, \{m\}, \emptyset, \emptyset)$.
- The components are a single M-shaped component, a single W-shaped component, and zero or more nontrivial crowns. Define $\#\mathrm{ST}_{MW} : \mathcal{Z}_{\geq 2} \times \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{N}$ as $\#\mathrm{ST}_{MW}(\mathcal{C}, m, w) = \#\mathrm{ST}(\mathcal{C}, \{m\}, \{w\}, \emptyset)$.

We recall the following lemma, which allows us to compute the number of sorting scenarios for a single component.

▶ **Lemma 13** ([1, Lemma 2.9]).

- *For all $\eta \in \mathbb{N}$, $\#\mathrm{ST}_N(\emptyset, \eta) = 1$.*
- *For all $w \in \mathbb{Z}^+$, $\#\mathrm{ST}_W(\emptyset, w) = 2^{w-1}$.*
- *For all $m \in \mathbb{Z}^+$, $\#\mathrm{ST}_M(\emptyset, m) = 2^{m-1}$.*
- *For all $c \in \mathbb{Z}_{\geq 2}$, $\#\mathrm{ST}_C(\{c\}) = c \cdot 2^{c-1}$.*

Our goal now is to enumerate the number of sorting scenarios in each of these cases. We first provide a recurrence relation for $\#\mathrm{ST}_N(\mathcal{C},\eta)$.

▶ **Proposition 14.** *We have the following recurrence relations for* $\#\mathrm{ST}_N$*:*

$$\#\mathrm{ST}_N(\mathcal{C},\eta) = \begin{cases} 1 & : \mathcal{C} = \emptyset, \eta = 0, \\ \sum_{c\in\mathcal{C}} \left(2c \cdot \#\mathrm{ST}_N(\mathcal{C}-\{c\},c)\right) & : \mathcal{C} \neq \emptyset, \eta = 0, \\ \#\mathrm{ST}_N(\mathcal{C},\eta-1) + \sum_{c\in\mathcal{C}} \left(2c \cdot \#\mathrm{ST}_N(\mathcal{C}-\{c\},c+\eta)\right) & : \textit{otherwise.} \end{cases}$$

Proof. First, $\#\mathrm{ST}_N(\emptyset,0)$ is the case of a single path of size 0 (a single edge). This represents that the given telomere has already been sorted, so there is only one way to sort this component (do nothing). On the other hand, when sorting a collection of nontrivial crowns together with an N-shaped component, the first operation either consists of applying a cut-join on the N-shaped component alone if it has size greater than 0 (1 way to do this), or applying a cut-join of a nontrivial crown to the N-shaped component. If the crown being operated on has size c, there are c possible places to cut it, and there are 2 possible ways to join it to to the N-shaped component (either telomere of the newly cut crown). The result of this operation is one fewer crown and an increase in the size of the N-shaped component by c. These considerations lead to the recursive formulas for $\#\mathrm{ST}_N(\mathcal{C},\eta)$. ◀

We next provide a recurrence relation for $\#\mathrm{ST}_W(\mathcal{C},w)$.

▶ **Proposition 15.** *We have the following recurrence relations for* $\#\mathrm{ST}_W$*:*

$$\#\mathrm{ST}_W(\mathcal{C},w) = \begin{cases} 1 & : \mathcal{C} = \emptyset, w = 1, \\ \sum_{c\in\mathcal{C}} \left(4c \cdot \#\mathrm{ST}_W(\mathcal{C}-\{c\},c+1)\right) & : \mathcal{C} \neq \emptyset, w = 1, \\ 2 \cdot \#\mathrm{ST}_W(\mathcal{C},w-1) + \sum_{c\in\mathcal{C}} \left(4c \cdot \#\mathrm{ST}_W(\mathcal{C}-\{c\},c+w)\right) & : \textit{otherwise.} \end{cases}$$

Proof. First, $\#\mathrm{ST}_W(\emptyset,1)$ is the case of a single W-shaped component of size 1. To sort such a path, join the telomeres in the top genome. So, there is only one way to sort this component. On the other hand, when sorting a collection of nontrivial crowns together with a W-shaped component, the first operation either consists of a cut-join of one end of the W-shaped component if it has size greater than 1 (2 ways to do this) or a cut-join of a nontrivial crown to the W-shaped component. If the crown being operated on has size c, there are c possible places to cut it, and there are 4 possible ways to join it to to the W-shaped component (either telomere of the newly cut crown could join with either telomere of the W-shaped component). The result of this operation is one fewer nontrivial crown and an increase in the size of the W-shaped component by c. These considerations lead to the recursive formulas for $\#\mathrm{ST}_W(\mathcal{C},w)$. ◀

We next provide an expression for $\#\mathrm{ST}_C(\mathcal{C})$.

▶ **Proposition 16.** *We have the following expression for* $\#\mathrm{ST}_C$*:*

$$\#\mathrm{ST}_C(\mathcal{C}) = \sum_{c\in\mathcal{C}} \left(c \cdot \#\mathrm{ST}_W(\mathcal{C}-\{c\},c)\right).$$

Note that $\mathcal{C} = \emptyset$ is not in the domain of $\#\mathrm{ST}_C$, and so this expression is well-defined.

Proof. When a collection of crowns sorts together, the first operation must be a cut. If the crown being operated on has size c, there are c possible places to cut it. The result of this operation is one fewer crown and a W-shaped component of size c. These considerations lead to the expression for $\#\mathrm{ST}_C(\mathcal{C})$. ◀

▶ Remark 17. A closed-form expression for $\#\mathrm{ST}_C(\mathcal{C})$ was previously established in [1, Corollary 3.2].

We have already discussed $\#\mathrm{ST}_W(\mathcal{C}, w)$. Below we establish tools to show $\#\mathrm{ST}_M(\mathcal{C}, w)$ is the same as $\#\mathrm{ST}_W(\mathcal{C}, w)$. We have thus far considered $A(G_1, G_2)$, where operations act on V_1. We can also consider $A(G_2, G_1)$, which is the same as $A(G_1, G_2)$, but we now operate on V_2. Every component of $A(G_1, G_2)$ corresponds to a component in $A(G_2, G_1)$ (on the same vertices), though component types are not necessarily the same. For example, an M-shaped component A_{12} in $A(G_1, G_2)$ corresponds to a W-shaped component A_{21} in $A(G_2, G_1)$.

▶ **Definition 18.** *For an operation α, define the reverse operation α^{rev} as follows. If α is a cut at adjacency ab, then α^{rev} is a join of a and b. Similarly, the reverse of a join is a cut. Further, if α is a cut-join ab, c to ac, b, then α^{rev} is a cut-join ac, b to ab, c. For a sequence σ of operations $\sigma_1, \ldots, \sigma_k$, let the reverse sequence σ^{rev} be $\sigma_k^{rev}, \ldots, \sigma_1^{rev}$.*

Observe that reversing an operation does not change the extremities operated on. We now show that reversal preserves the sort together relation.

▶ **Proposition 19.** *For a sorting scenario σ and components A_{12} and B_{12} in $A(G_1, G_2)$, suppose we have that $A_{12} \sim B_{12}$. Then, for σ^{rev} and corresponding components A_{21} and B_{21} in $A(G_2, G_1)$, we have $A_{21} \sim B_{21}$.*

Proof. Let σ be a sorting scenario for $A(G_1, G_2)$ that consists of a sequence of operations $\sigma_1, \ldots, \sigma_k$. Suppose further that σ sorts two components A_{12} and B_{12} together such that $A_{12} \sim B_{12}$. Since σ transforms G_1 into G_2, the sequence σ^{rev} transforms G_2 into G_1. Thus, σ^{rev} is a sorting scenario for $A(G_2, G_1)$.

Let A_{21} and B_{21} denote the two components in $A(G_2, G_1)$ that correspond to A_{12} and B_{12}. We will show that $A_{21} \sim B_{21}$ with respect to σ^{rev}. Since $A_{12} \sim B_{12}$ with respect to σ, there is some i such that operation σ_i is a cut-join operation which joins an extremity a of A_{12} with an extremity b of B_{12}. Moreover, σ_i^{rev} cuts the adjacency ab. Since a and b come from different components, they must have been joined through operation σ_j^{rev} for some $j > i$ to create adjacency ab (which is then cut by σ_i^{rev}). Thus, $A_{12} \sim B_{12}$ in $A(G_2, G_1)$ with respect to σ^{rev}, as required. ◀

▶ **Corollary 20.** *Let $\mathcal{B}'_{12}$ be a subset of the components of $A(G_1, G_2)$, and let $\mathcal{B}'_{21}$ be the corresponding subset of the components of $A(G_2, G_1)$. Then $\#\mathrm{ST}(\mathcal{B}'_{12}) = \#\mathrm{ST}(\mathcal{B}'_{21})$. Consequently, $\#\mathrm{ST}_M(\mathcal{C}, m) = \#\mathrm{ST}_W(\mathcal{C}, m)$, and $\#\mathrm{ST}_{MW}(\mathcal{C}, m, w) = \#\mathrm{ST}_{MW}(\mathcal{C}, w, m)$.*

Proof. Given a subset $\mathcal{B}'_{12}$ of components of $A(G_1, G_2)$, let $\mathcal{B}'_{21}$ be the set of corresponding components in $A(G_2, G_1)$. Now, let Σ_{12} be the set of scenarios that sort the components of $\mathcal{B}'_{12}$ together, and let Σ_{21} be the set of scenarios that sort the components of $\mathcal{B}'_{21}$ together. By definition, $|\Sigma_{12}| = \#\mathrm{ST}(\mathcal{B}'_{12})$ and $|\Sigma_{21}| = \#\mathrm{ST}(\mathcal{B}'_{21})$. Next, let Σ_{12}^{rev} and Σ_{21}^{rev} be the sets of reversals of the scenarios in Σ_{12} and Σ_{21} respectively. By Proposition 19, $\Sigma_{12}^{rev} \subseteq \Sigma_{21}$. Since reversal is an involution, we also obtain that $\Sigma_{21}^{rev} \subseteq \Sigma_{12}$. Since $|\Sigma_{12}| = |\Sigma_{12}^{rev}|$ and $|\Sigma_{21}| = |\Sigma_{21}^{rev}|$, this implies that $|\Sigma_{12}| \leq |\Sigma_{21}|$ and $|\Sigma_{21}| \leq |\Sigma_{12}|$. Together, these imply that $|\Sigma_{12}| = |\Sigma_{21}|$, meaning $\#\mathrm{ST}(\mathcal{B}'_{12}) = \#\mathrm{ST}(\mathcal{B}'_{21})$, as required.

We note that an M-shaped component in $A(G_1, G_2)$ is a W-shaped component in $A(G_2, G_1)$. Thus, we obtain that $\#\mathrm{ST}_M(\mathcal{C}, m) = \#\mathrm{ST}_W(\mathcal{C}, m)$ and $\#\mathrm{ST}_{MW}(\mathcal{C}, m, w) = \#\mathrm{ST}_{MW}(\mathcal{C}, w, m)$, as desired. ◀

By Corollary 20, $\#\mathrm{ST}_M(\mathcal{C}, m) = \#\mathrm{ST}_W(\mathcal{C}, m)$. But, here we present a different expression for $\#\mathrm{ST}_M(\mathcal{C}, m)$, which will be useful later when we examine $\#\mathrm{ST}_{MW}$. To simplify notation, we introduce a new definition. Denote:

$$\mathrm{sum}(\mathcal{C}) := \sum_{c \in C} c.$$

▶ **Lemma 21.** *We have the following expression for* $\#\mathrm{ST}_M$*:*

$$\#\mathrm{ST}_M(\mathcal{C}, m) = \sum_{\eta=0}^{m-1} \sum_{\mathcal{C}' \subseteq \mathcal{C}} \binom{\mathrm{sum}(\mathcal{C}) + |\mathcal{C}| + m - 1}{\mathrm{sum}(\mathcal{C}') + |\mathcal{C}'| + \eta} \#\mathrm{ST}_N(\mathcal{C}', \eta) \cdot \#\mathrm{ST}_N(\mathcal{C} - \mathcal{C}', m - 1 - \eta)$$
$$+ \sum_{c \in \mathcal{C}} (c \cdot \#\mathrm{ST}_{MW}(\mathcal{C} - \{c\}, m, c)) .$$

Proof. See the proof of Lemma 3.11 in the full version. ◀

It now remains to define a recurrence relation for $\#\mathrm{ST}_{MW}(\mathcal{C}, m, w)$. We begin with the following lemma.

▶ **Lemma 22.** *We have the following recurrence relations for* $\#\mathrm{ST}_{MW}$*:*

$$\#\mathrm{ST}_{MW}(\mathcal{C}, m, w) = \begin{cases} f(\mathcal{C}, m, w) & : w = 1, \\ 2 \cdot \#\mathrm{ST}_{MW}(\mathcal{C}, m, w-1) + f(\mathcal{C}, m, w) & : w > 1. \end{cases}$$

where

$$f(\mathcal{C}, m, w) = \sum_{c \in \mathcal{C}} (4c \cdot \#\mathrm{ST}_{MW}(\mathcal{C} - \{c\}, m, c + w))$$
$$+ \sum_{\eta=0}^{m-1} 4 \cdot \sum_{\mathcal{C}' \subseteq \mathcal{C}} \binom{\mathrm{sum}(\mathcal{C}) + |\mathcal{C}| + m + w - 1}{\mathrm{sum}(\mathcal{C}') + |\mathcal{C}'| + \eta} \#\mathrm{ST}_N(\mathcal{C}', \eta) \cdot \#\mathrm{ST}_N(\mathcal{C} - \mathcal{C}', m + w - \eta - 1).$$

Proof. See the proof of Lemma 3.12 in the full version. ◀

A priori, to use Lemma 22, we have to track the case when an M-shaped component sorting with a W-shaped component results in two N-shaped components. This requires $O(2^{|\mathcal{C}|})$ steps to compute the double-summation in $f(\mathcal{C}, m, w)$ from Lemma 22. The next proposition allows us to both avoid these steps and simplify our algorithm.

▶ **Proposition 23.** *We have the following recurrence relation for* $\#\mathrm{ST}_{MW}$*:*

$$\#\mathrm{ST}_{MW}(\mathcal{C}, m, w) = \begin{cases} g(\mathcal{C}, m, w) & : m = 1,\ w = 1, \\ \#\mathrm{ST}_{MW}(\mathcal{C}, m-1, w) + g(\mathcal{C}, m, w) & : m > 1,\ w = 1, \\ \#\mathrm{ST}_{MW}(\mathcal{C}, m, w-1) + g(\mathcal{C}, m, w) & : m = 1,\ w > 1, \\ \#\mathrm{ST}_{MW}(\mathcal{C}, m-1, w) & : \textit{otherwise} \\ \quad + \#\mathrm{ST}_{MW}(\mathcal{C}, m, w-1) + g(\mathcal{C}, m, w). & \end{cases}$$

where

$$g(\mathcal{C}, m, w) = 2 \cdot \#\mathrm{ST}_W(\mathcal{C}, m + w) + \sum_{c \in \mathcal{C}} 2c \Big(\#\mathrm{ST}_{MW}(\mathcal{C} - \{c\}, m + c, w)$$
$$+ \#\mathrm{ST}_{MW}(\mathcal{C} - \{c\}, m, w + c) - \#\mathrm{ST}_{MW}(\mathcal{C} - \{c\}, m + w, c) \Big).$$

Proof. See the proof of Proposition 3.13 in the full version. ◀

4 Fixed-Parameter Tractability of Pairwise Rearrangement

In this section, we will use the recurrences we have found to establish our main result.

▶ **Theorem 1.** *In the Single Cut-and-Join model,* Pairwise Rearrangement *is fixed-parameter tractable, with respect to the number of nontrivial components in the adjacency graph.*

Proof. Given two genomes with n edges each, the adjacency graph is easily constructed in polynomial time. Let $\mathcal{B}$ be the set of components for this adjacency graph, and let k denote the number of nontrivial components in $\mathcal{B}$. It suffices to show that the number of scenarios $\#\text{MPS}(\mathcal{B})$ can be determined in time $O(k \cdot 2^k \cdot B_k + k \cdot 2^k \cdot n^2)$, where B_k is the k^{th} Bell number.

Let $\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N}$ be multisets of nonnegative integers, denoting the sizes of the nontrivial crowns, M-shaped components, W-shaped components, and N-shaped components, respectively, of $\mathcal{B}$. Thus, we assume that every element of $\mathcal{C}$ is at least 2, and every element of $\mathcal{M}$ and $\mathcal{W}$ is at least 1.

We will proceed in two stages. In the first stage, we will build up a table of values for $\#\text{ST}_N(\mathcal{A}, \eta)$, $\#\text{ST}_W(\mathcal{A}, w)$, $\#\text{ST}_C(\mathcal{A})$, and $\#\text{ST}_{MW}(\mathcal{A}, m, w)$ with $\mathcal{A}$ any sub-multiset of $\mathcal{C}$ and $\eta \geq 0$, $w \geq 1$, and $m \geq 1$. Then in the second stage, we will use this final table of values along with Lemma 11 to compute $\#\text{MPS}(\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N})$.

Stage 1. When $\eta \geq 0$ we have by Lemma 13 that $\#\text{ST}_N(\emptyset, \eta) = 1$. Similarly, for $w \geq 1$, we have that $\#\text{ST}_W(\emptyset, w) = 2^{w-1}$, which is computable in time $O(n)$.

Now fix $m \in [n], w \in [n]$ with $m + w = i$, and suppose that for each $1 \leq m' \leq m$ and each $1 \leq w' \leq w$ with $m' + w' < i$, that we have computed $\#\text{ST}_{MW}(\emptyset, m', w')$. Using these values, together with the fact that $\#\text{ST}_W(\emptyset, w) = 2^{w-1}$, we may apply Proposition 23 to compute $\#\text{ST}_{MW}(\emptyset, m, w)$. There are at most n^2 pairs $(m', w') \in [n] \times [n]$, and so this step takes time $O(n^2)$.

Let u be an integer $0 < u \leq |\mathcal{C}| \leq k$. First, let $0 < x \leq n$. Suppose we have a table of values of $\#\text{ST}_N(\mathcal{A}, \eta)$ and $\#\text{ST}_W(\mathcal{A}, w)$ for all sub-multisets $\mathcal{A}$ of $\mathcal{C}$ with $|\mathcal{A}| < u$ and $\eta, w \leq n$ and also for all sub-multisets $\mathcal{A}$ of $\mathcal{C}$ with $|\mathcal{A}| = u$ and $\eta, w < x$. Let $\mathcal{C}'$ be a sub-multiset of $\mathcal{C}$ with $|\mathcal{C}'| = u$. We now proceed to fill in some of our lookup table from the bottom-up, using the recurrences from the previous section.

- We may compute $\#\text{ST}_N(\mathcal{C}', x)$ using Proposition 14. There are at most kn cells in our lookup table, and so this step takes time $O(kn)$.
- We may compute $\#\text{ST}_W(\mathcal{C}', x)$ using Proposition 15. There are at most kn cells in our lookup table, and so this step takes time $O(kn)$.
- We now turn to computing $\#\text{ST}_C(\mathcal{C}')$. To do so, we apply Proposition 16 using the previously computed values of $\#\text{ST}_W(\mathcal{A}, w)$ for all $\mathcal{A}$ of size less than u and all $1 \leq w \leq n$. This step takes time $O(k)$.

Now, let x and y be nonnegative integers with $x + y \leq n$. Suppose we have a table of values of $\#\text{ST}_{MW}(\mathcal{A}, m, w)$ for all sub-multisets $\mathcal{A}$ of $\mathcal{C}$ satisfying one of the following:

- $|\mathcal{A}| < u$ for any nonnegative integers m and w with $m + w \leq n$, or
- $|\mathcal{A}| = u$ with $m + w < x + y$.

We now turn to computing $\#\text{ST}_{MW}(\mathcal{C}', x, y)$. To do so, we apply Proposition 23 using the previously computed values of $\#\text{ST}_W(\mathcal{A}, w)$ for all $1 \leq w \leq n$ and of $\#\text{ST}_{MW}(\mathcal{A}, m, w)$ for all sub-multisets $\mathcal{A}$ of $\mathcal{C}$ satisfying one of the conditions above. There are at most kn^2 many such cells in our lookup table, and so filling in these cells takes time $O(kn^2)$.

To summarize, our first stage involves building up a table of values for $\#\mathrm{ST}_N(\mathcal{A},\eta)$, $\#\mathrm{ST}_W(\mathcal{A},w)$, $\#\mathrm{ST}_C(\mathcal{A})$, and $\#\mathrm{ST}_{MW}(\mathcal{A},m,w)$ with $\mathcal{A}$ any sub-multiset of $\mathcal{C}$ and η, w, and m having values as above. This first stage is computable in time $O(k\cdot 2^k\cdot n^2)$ by iterating through all sub-multisets of $\mathcal{C}$ in non-decreasing order of cardinality.

Stage 2. We will use the table of values computed in Stage 1, along with Lemma 11, to compute $\#\mathrm{MPS}(\mathcal{C},\mathcal{M},\mathcal{W},\mathcal{N})$ as follows. First, let $\mathcal{B}'\subseteq\mathcal{B}$ denote the set of all nontrivial components in $\mathcal{B}$, and let t denote the number of trivial N-shaped components in $\mathcal{B}$. As per Lemma 11, we consider each partition π of $\mathcal{B}$, and examine $\#\mathrm{ST}(\pi_i)$ for each part π_i of π. By Proposition 12, we can determine in time $O(k)$ if $\#\mathrm{ST}(\pi_i)=0$. In such a case, the term of $\#\mathrm{MPS}(\mathcal{B})$ corresponding to π contributes 0, and so we may discard π. Call a partition π *permissible* if $\#\mathrm{ST}(\pi_i)>0$ for all parts. Recall the parts π_i from Proposition 12 for which $\#\mathrm{ST}(\pi_i)=0$. Note that the only part π_i with a combination of trivial and nontrivial components for which $\#\mathrm{ST}(\pi_i)>0$ is the case in which π_i contains precisely nontrivial crowns and a single trivial N-shaped component. So the permissible partitions of $\mathcal{B}$ are precisely the partitions generated by the following procedure:

- Start with a permissible partition π' of $\mathcal{B}'$.
- For at most t parts π'_i of π' that each consist of only nontrivial crowns, add one of the t trivial N-shaped components from $\mathcal{B}$ to π'_i.
- Each of the remaining trivial N-shaped components and each trivial crown from $\mathcal{B}$ become a singleton part.

Let $\pi'=(\pi'_1,\dots,\pi'_{p(\pi')})$ be a permissible partition of $\mathcal{B}'$. We say that a permissible partition $\pi=(\pi_1,\dots,\pi_{p(\pi)})$ of $\mathcal{B}$ *extends* π' if there exists an injective function $f:[p(\pi')]\to[p(\pi)]$ such that $\pi'_i\subseteq\pi_{f(i)}$ for all $i\in[p(\pi')]$. Note that by Proposition 12, π'_i and $\pi_{f(i)}$ may only be different if π'_i contains only nontrivial crowns, and $\pi_{f(i)}$ contains a nontrivial N-shaped component.

Note that while $\mathcal{B}'$ has at most k components, $\mathcal{B}$ may in general not have bounded size. Fix a permissible partition π' of $\mathcal{B}'$. Our goal is to compute:

$$\sum_{\substack{\pi\in\Pi(\mathcal{B})\\ \pi\text{ extends }\pi'}}\binom{d(\mathcal{B})}{d(\pi_1),d(\pi_2),\dots,d(\pi_{p(\pi)})}\prod_{i=1}^{p(\pi)}\#\mathrm{ST}(\pi_i).$$

As above, let π' be a permissible partition of $\mathcal{B}'$, and let π be an extension of π'. Recall, if π_i consists of only one trivial component, then $\#\mathrm{ST}(\pi_i)=1$. Let π_i be a part of π that contains only nontrivial crowns and a single trivial N-shaped component. Let π'_i be obtained from π_i by removing the trivial N-shaped component. We have by equation (2) that $d(\pi'_i)=d(\pi_i)$. Thus, if π extends π', we have that:

$$\binom{d(\mathcal{B})}{d(\pi_1),d(\pi_2),\dots,d(\pi_{p(\pi)})}=\binom{d(\mathcal{B}')}{d(\pi'_1),d(\pi'_2),\dots,d\left(\pi'_{p(\pi')}\right)}.$$

Proposition 12 and the above procedure for constructing permissible partitions yields precisely the following cases:

- **Case 1:** Suppose that π'_i consists of an N-shaped component and zero or more nontrivial crowns. In this case, we can use a known value of $\#\mathrm{ST}_N$ to count the number of ways to sort this part.
- **Case 2:** Suppose that π'_i consists of a W-shaped component and zero or more nontrivial crowns. In this case, we can use a known value of $\#\mathrm{ST}_W$ to count the number of ways to sort this part.

- **Case 3:** Suppose that π'_i consists of an M-shaped component and zero or more nontrivial crowns. In this case, we can also use a known value of $\#\text{ST}_M$ to count the number of ways to sort this part.
- **Case 4:** Suppose that π'_i consists of an M-shaped component, a W-shaped component, and zero or more nontrivial crowns. In this case, we can use a known value of $\#\text{ST}_{MW}$ to count the number of ways to sort this part.
- **Case 5:** If none of Cases 1-4 hold, then we necessarily have that π'_i consists of only nontrivial crowns. Since the presence of such parts implies that π' may not necessarily extend uniquely to a permissible partition of $\mathcal{B}$, we consider all such parts π'_i simultaneously in order to handle all such permissible extensions of π'. Let $\pi'_{i_1}, \dots, \pi'_{i_\ell}$ be the parts of π' that contain only nontrivial crowns. By Proposition 12, we have that for $j \in [\ell]$, we can only add at most one trivial N-shaped component to π'_{i_j}. As the trivial N-shaped components and the parts of π' are both distinguishable, there are $P(t, v) = t!/(t-v)!$ ways to assign $v \leq t$ trivial N-shaped components to v parts drawn from $\pi'_{i_1}, \dots, \pi'_{i_\ell}$. The remaining trivial N-shaped components each form their own singleton component in the corresponding partition π of $\mathcal{B}$ extending π'. If part π'_{i_j} $(j \in [\ell])$ has a trivial N-shaped component added, we can use a known value of $\#\text{ST}_N$ to count the number of ways to sort this part. If not, we can use a known value of $\#\text{ST}_C$ to count the number of ways to sort this part. The number of ways to sort each part is independent of the number of ways to assign the trivial N-shaped components. Thus, multiply the value returned from the lookup table by $P(t, v)$. Finally, using Lemma 11, we multiply together each of the $\#\text{ST}_N$ and $\#\text{ST}_C$ values that we accessed from the lookup table. This takes time $O(k)$ since there are $O(k)$ parts.
  We now iterate over all integers $0 \leq v \leq \min(t, \ell)$, where we consider all possible assignments of v trivial N-shaped components to the parts of π' (where the assignment is as described in the preceding paragraph). As each π'_{i_j} can only receive at most one trivial N-shaped component, we iterate over all v-subsets of $[\ell]$ to fully enumerate each case. As $\ell \leq k$, there are at most 2^k total subsets to consider across all values of v, and so computing the count in this case takes time $O(k \cdot 2^k)$.

In the first four cases, we only look at a single part at a time. Since there are $O(k)$ parts, Lemma 11 gives a complexity of $O(k)$ for each of these cases. In Case 5 we already account for Lemma 11, so we have a complexity of $O(k \cdot 2^k)$. Thus for all cases the runtime is at most $O(k \cdot 2^k)$ using the existing values in the lookup table to compute $\#\text{ST}(\pi_i)$. As we are summing over all possible partitions, we are evaluating at most the number of partitions of a set of cardinality k, which is the k^{th} Bell number B_k. This gives a complexity of $O(k \cdot 2^k \cdot B_k)$ for computing $\#\text{MPS}(\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N})$ from the pre-established lookup tables.

So, overall, we have a complexity of $O(k \cdot 2^k \cdot n^2)$ for our first stage and a complexity of $O(k \cdot 2^k \cdot B_k)$ for our second stage. Thus, $\#\text{MPS}(\mathcal{C}, \mathcal{M}, \mathcal{W}, \mathcal{N})$ is computable in time $O(k \cdot 2^k \cdot B_k + k \cdot 2^k \cdot n^2)$, as required. The result now follows. ◀

5 Conclusion

We investigated the computational complexity of the Pairwise Rearrangement problem in the Single Cut-and-Join model. In particular, while Pairwise Rearrangement was previously shown to be #P-complete under polynomial-time Turing reductions [1], we proved it is fixed-parameter tractable when parameterized by the number of components in the adjacency graph that are not trivial crowns (Theorem 1). In particular, our results show that the number of nontrivial components serves as a key barrier towards efficiently enumerating and sampling minimum length sorting scenarios. We conclude with some open questions.

▶ **Question 24.** In the Single Cut-and-Join model, does PAIRWISE REARRANGEMENT belong to FPRAS?

▶ **Question 25.** In the Double Cut-and-Join model, is PAIRWISE REARRANGEMENT #P-complete?

The work of Braga and Stoye [3] immediately yields a fixed-parameter tractable algorithm for PAIRWISE REARRANGEMENT in the Double Cut-and-Join model. Their algorithm is considerably simpler than our work in this paper.

The computational complexity of the PAIRWISE REARRANGEMENT problem in the Reversal model is a long-standing open question. In particular, it is believed that this problem is #P-complete. Furthermore, PAIRWISE REARRANGEMENT has been resistant to efficient sampling; membership in FPRAS remains open. Thus, perhaps the lens of parameterized complexity might shed new light on this problem. The following question is natural, though might be hard.

▶ **Question 26.** In the Reversal model, is PAIRWISE REARRANGEMENT fixed-parameter tractable by the number of components in the adjacency graph?

References

1 Lora Bailey, Heather Smith Blake, Garner Cochran, Nathan Fox, Michael Levet, Reem Mahmoud, Elizabeth Bailey Matson, Inne Singgih, Grace Stadnyk, Xinyi Wang, and Alexander Wiedemann. Complexity and enumeration in models of genome rearrangement. In Weili Wu and Guangmo Tong, editors, *Computing and Combinatorics*, pages 3–14, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-49190-0_1.

2 Anne Bergeron, Paul Medvedev, and Jens Stoye. Rearrangement models and single-cut operations. *Journal of computational biology : a journal of computational molecular cell biology*, 17:1213–25, September 2010. doi:10.1089/cmb.2010.0091.

3 Marília D. V. Braga and Jens Stoye. Counting all DCJ sorting scenarios. In Francesca D. Ciccarelli and István Miklós, editors, *Comparative Genomics*, pages 36–47, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-04744-2_4.

4 Aaron Darling, István Miklós, and Mark Ragan. Dynamics of genome rearrangement in bacterial populations. *PLoS genetics*, 4:e1000128, July 2008. doi:10.1371/journal.pgen.1000128.

5 Rick Durrett, Rasmus Nielsen, and Thomas York. Bayesian estimation of genomic distance. *Genetics*, 166:621–9, February 2004. doi:10.1534/genetics.166.1.621.

6 Pedro Feijão and Joao Meidanis. SCJ: A breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM transactions on computational biology and bioinformatics*, 8:1318–29, February 2011. doi:10.1109/TCBB.2011.34.

7 Andrew Holland and Don Cleveland. Chromoanagenesis and cancer: Mechanisms and consequences of localized, complex chromosomal rearrangements. *Nature medicine*, 18:1630–8, November 2012. doi:10.1038/nm.2988.

8 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.

9 Bret Larget, Donald L. Simon, Joseph B. Kadane, and Deborah Sweet. A Bayesian Analysis of Metazoan Mitochondrial Genome Arrangements. *Molecular Biology and Evolution*, 22(3):486–495, November 2004. doi:10.1093/molbev/msi032.

10 Barbara McClintock. Chromosome organization and genic expression. In *Cold Spring Harbor symposia on quantitative biology*, volume 16, pages 13–47. Cold Spring Harbor Laboratory Press, 1951. doi:10.1101/sqb.1951.016.01.004.

11 István Miklós, Sándor Z. Kiss, and Eric Tannier. Counting and sampling SCJ small parsimony solutions. *Theor. Comput. Sci.*, 552:83–98, 2014. doi:10.1016/j.tcs.2014.07.027.

12 István Miklós and Heather Smith. Sampling and counting genome rearrangement scenarios. *BMC Bioinformatics*, 16:S6, October 2015. doi:10.1186/1471-2105-16-S14-S6.

13 István Miklós and Eric Tannier. Bayesian sampling of genomic rearrangement scenarios via double cut and join. *Bioinformatics*, 26(24):3012–3019, October 2010. doi:10.1093/bioinformatics/btq574.

14 István Miklós and Eric Tannier. Approximating the number of double cut-and-join scenarios. *Theoretical Computer Science*, 439:30–40, 2012. doi:10.1016/j.tcs.2012.03.006.

15 J. D. Palmer and L. A. Herbon. Plant mitochondrial dna evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28:87–97, December 1988. doi:10.1007/BF02143500.

16 A. H. Sturtevant. The linear arrangement of six sex-linked factors in drosophila, as shown by their mode of association. *Journal of Experimental Zoology*, 14(1):43–59, 1913. doi:10.1002/jez.1400140104.

17 A. H. Sturtevant. Genetic factors affecting the strength of linkage in drosophila. *Proceedings of the National Academy of Sciences of the United States of America*, 3(9):555–558, 1917. URL: http://www.jstor.org/stable/83776.

18 A. H. Sturtevant. Known and probably inverted sections of the autosomes of Drosophila melanogaster. *Carnegie Institution of Washington Publisher*, 421:1–27, 1931.

19 A H Sturtevant and E Novitski. The Homologies of the Chromosome Elements in the genus Drosophila. *Genetics*, 26(5):517–541, September 1941. doi:10.1093/genetics/26.5.517.

20 Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics (Oxford, England)*, 21:3340–6, September 2005. doi:10.1093/bioinformatics/bti535.