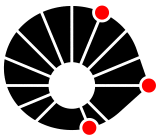
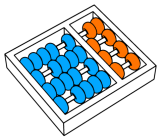




Counting sorting scenarios and intermediate genomes for the rank distance

J. P. P. Zanetti *J. Meidanis*

Technical Report - IC-18-07 - Relatório Técnico
April - 2018 - Abril

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Counting sorting scenarios and intermediate genomes for the rank distance

João Paulo Pereira Zanetti* João Meidanis†

Abstract

An important problem in genome comparison is the genome sorting problem, that is to find a sequence of basic operations that transforms one genome into another and corresponds to the distance between them. These sequences are called optimal sorting scenarios. However, there is usually a large number of such scenarios, and a naive algorithm is very likely to be biased towards a specific type of scenario, impairing its usefulness in real-world applications. One way to go beyond the traditional sorting algorithms is to explore all possible solutions, looking at all the optimal sorting scenarios instead of just an arbitrary one. Another approach in the same direction is to analyze all the intermediate genomes, that is, all genomes that are part of an optimal sorting scenario. In this paper, we show how to count the number of optimal sorting scenarios and the number of intermediate genomes between any two given genomes, under the rank distance.

1 Introduction

At a higher level of abstraction, genomes can be represented as a list of blocks (that can be genes, markers, or other synthenic regions), and their evolution can be modeled by large-scale mutation events we call *genome rearrangements*. Different genome rearrangement models define different events and costs for them, and their most basic application is to determine the lowest cost to transform one genome into another. This is the *genome distance problem*.

Another fundamental problem that elaborates on this is to find sequences of rearrangement operations with minimum cost. We call this the *genome sorting problem*. However, a single arbitrary sequence of events among many optimal ones is hardly representative of the evolutionary process, especially considering that sorting algorithms might be biased towards certain kinds of sorting sequences. On the other hand, listing all possible optimal scenarios is not practical, because their number is simply too large.

A first step towards exploring the solution space of the genome sorting problem as a whole is to count the number of optimal sorting scenarios between two given genomes. This can reveal patterns in the optimal solutions and suggest strategies for real data applications.

To the best of our knowledge, the studies of the solution space of genome sorting for multi-chromosomal genomes have been limited to the DCJ distance [19]. Braga and Stoye showed how to count optimal DCJ scenarios between two genomes, with and without recombination [3]. Ouangraoua and Bergeron also count optimal scenarios without recombination, and establish bijections between these scenarios and well-known combinatorial objects [14]. Feijão took these results one

*Institute of Computing, UNICAMP, 13083-852 Campinas, SP. joao.zanetti@ic.unicamp.br. FAPESP grant 2017/02748-2.

†Institute of Computing, UNICAMP, 13083-852 Campinas, SP. meidanis@ic.unicamp.br

step further, and showed how to count the intermediate genomes between two genomes and how to use those to reconstruct ancestral genomes [5]. Here, we seek to retrace their steps, but with respect to the rank distance model [11] instead of DCJ.

The rank distance, formalized by Meidanis, Biller, and Zanetti [11] is based on a representation of genomes using matrices. The rank distance is twice the algebraic distance [6], and very close to the DCJ distance [19, 6]. This allows for both graphical and algebraic manipulation of genomes. The rank distance is equivalent to the DCJ for genomes with the same free ends. One advantage of the rank distance is that it ensures that there is no recombination while sorting. On the other hand, since the basic operations have different weights, the scenarios have variable lengths, and this raises a challenge when counting all possible scenarios. Fortunately, we were able to overcome this difficulty by adding an extra parameter in the recurrence.

The rest of this paper is organized as follows. Section 2 presents the rank distance concepts we use. In Section 3 we present the multi-genome breakpoint graph used in this analysis and how it relates to the rank distance. In Section 4 we show how to count the number of optimal scenarios between two genomes under the rank distance. In Section 5 we count the number of intermediate genomes. In Section 6 we relate our experiments with real genomic data. Finally, Section 7 summarizes our conclusions.

2 Genomes as matrices and the rank distance

In this section we present our representation of genomes as matrices, the rank distance, and its basic operations that will be useful later in the paper. For a more complete source on the rank distance, we refer the reader to the tech report by Meidanis, Biller, and Zanetti [11]

One way to see genomes is as a collection of chromosomes, each of them a linear of circular sequence of genes. Each gene is represented as a linear segment, with two *extremities* through which genes are connected, by *adjacencies*. At this level of abstraction, a genome can be fully represented by its set of extremities and the adjacencies between them. See the example in Figure 1a.

We can encode this representation of genomes in matrices [20, 11]. For a genome G involving n genes (and therefore $2n$ gene extremities), we can choose an ordering for the extremities, assigning them to the columns (and rows) of a $2n \times 2n$ matrix, and then define the corresponding *genome matrix* as follows:

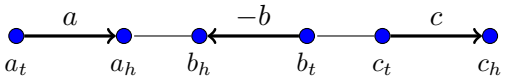
$$G_{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } i \text{ and } j \text{ are adjacent in } G, \text{ or} \\ & \text{if } i = j \text{ and } i \text{ is a free end in } G \\ 0 & \text{if } i \neq j \text{ and } i \text{ and } j \text{ are not adjacent in } G, \text{ or} \\ & \text{if } i = j \text{ and } i \text{ is not a free end in } G \end{cases}$$

An example of a genome matrix can be seen in Figure 1b. These matrices have some interesting properties: they are symmetric, that is, $A^t = A$; orthogonal, that is, $A^t = A^{-1}$; and involutions, that is $A^2 = I$. Also, since they are square, binary, and orthogonal matrices, they are permutation matrices. It is also worthy to note that these matrices are sparse, and amenable to memory-efficient storage.

If A and B are genomes over the same genes, the *distance* $d(A, B)$ between A and B is defined as

$$d(A, B) = r(B - A),$$

where r is the rank of a matrix.



(a) Genome with only one chromosome, linear, and adjacencies $\{a_h, b_h\}$ and $\{b_t, c_c\}$. The extremities a_t and c_h are free ends.

$$\begin{array}{c} \\ a_t \\ a_h \\ b_t \\ b_h \\ c_t \\ c_h \end{array} \begin{array}{c} \begin{array}{cccccc} a_t & a_h & b_t & b_h & c_t & c_h \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

(b) Matrix representation of the genome in Figure 1a.

Figure 1: Example of a genome and its matrix representation.

Although the rank distance is defined in terms of matrices, it is also possible to characterize it as the weight of an optimal series of operations transforming A into B. We say a matrix X is *applicable* to a genome A when $A + X$ is also a genome. A matrix that is applicable to at least one genome is called an *operation*.

An operation is *basic* if it is a *cut* of an adjacency, a *join* of two free ends, or a *double swap* of two adjacencies into two new ones using the same four extremities. Any other rearrangement operation can be decomposed as a sum of these three kinds of operations [11]. This way, we are able to slim down the cast of operations to just three, without loss of generality.

The matrices corresponding to the basic operations are characterized in detail in [11, Chap. 2]. For this work, the most important information about them is that cuts and joins are rank 1 matrices, while double swaps have rank 2.

3 Breakpoint graph

Genomes, as we describe here, are matchings over the set of gene extremities. We can graphically represent two genomes A and B to compute their distance as matchings, using one color (or line style) for A and another for B. See Figure 2.

The graph we use, called *breakpoint graph* [18] is based on the original breakpoint graph introduced by Hannenhalli and Pevzner in 1995 [7]. The difference is that we do not use caps at chromosome ends. Our free ends are simply extremities that are not adjacent to any other. We believe this makes representation simpler and clearer, without adding any extra difficulties to the results. It is important to note that the breakpoint graph is also the line graph of the adjacency graph [18], another graph used in the analysis of genome distances [1].

Given two genomes A and B of equal gene content, we build the two-genome breakpoint graph $BG(A, B)$ as follows. Its vertices are the extremities of the genomes, and there are two sets of edges: A-edges, that connect two extremities that are adjacent in A, and B-edges, that connect every pair of extremities adjacent in B.

While sorting from A to B, we call A the *source genome*, and B the *target genome*. In the graph $BG(A, B)$, we color the edges from the source genome A gray, and the edges from the target genome B black.

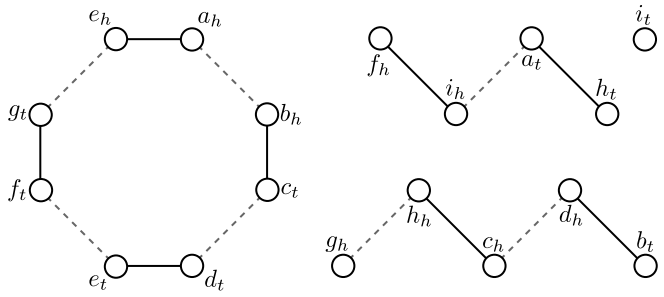


Figure 2: Example breakpoint graph $BG(A, B)$. Genome A has adjacencies $a_h b_h$, $c_t d_t$, $e_t f_t$, $e_h g_t$, $a_t i_h$, $g_h h_h$, and $c_h d_h$, drawn as gray dashed edges, and free ends b_t, f_h, h_t, and i_t. Genome B has adjacencies $a_h e_h$, $b_h c_t$, $d_t e_t$, $f_t g_t$, $f_h i_h$, $a_t h_t$, $c_h h_h$, and $b_t d_h$, drawn as black solid edges, and free ends g_h and i_t.

Note that when the genome is already sorted, that is, when both genomes are equal, the edges of both colors coincide, leaving only two types of components, cycles with two edges — shared adjacencies —, and isolated vertices — shared free ends.

In general, there are three types of components in the breakpoint graph $BG(A, B)$. Cycles always have an even number of edges. On the other hand, the number of edges in a path can be even or odd. Paths with an even number of edges are called *balanced*, because they have the same number of edges of each color. Odd paths are called *unbalanced*; they begin and end with edges from the same genome. Unbalanced edges can be further classified into two types, *AA-paths*, or *gray paths*, and *BB-paths*, or *black paths*, according to which genome accounts for more edges. Therefore, considering $BG(A, B)$, sorting A into B can be seen as the process of applying basic operations to the A-edges until they are all coincident with the B-edges.

A cut either transforms a cycle into a path, or splits one path in two. A join does the reverse: either transforms a path into a cycle, or joins two paths into one. A double swap can extract a cycle from any type of component or reverse a part of a component. When acting on two separate components, a double swap can also insert a circular component into another one (linear or circular) or swap end segments of two paths.

We call *big components* cycles with four or more edges and paths with at least one edge. These are components that need to be worked on in order for the genomes to be sorted. Cycles with two edges and paths of length zero (isolated vertices) will be called *small components*. The next result shows how to compute the rank distance using the breakpoint graph.

Theorem 1. *Given two genomes A and B over the same set of extremities, the rank distance between them is given by*

$$d(A, B) = 2n - 2c - p,$$

where n is the number of genes, and c and p are respectively the number of cycles and paths in $BG(A, B)$

Proof. Feijão and Meidanis in 2013 showed that the algebraic distance $d_{alg}(A, B)$ between two genomes A and B is given by $d_{alg}(A, B) = n - n_C - \frac{n_P}{2}$, where n is the number of genes, and n_C and n_P are respectively the number of cycles and paths in the adjacency graph of A and B [6, Theorem 3.11]. Since the breakpoint graph is the line graph of the adjacency graph, we have $c = n_C$ and $p = n_P$. Later, Zanetti, Biller and Meidanis showed that $d(A, B) = 2d_{alg}(A, B)$ [20], and, therefore, $d(A, B) = 2n - 2c - p$. □

4 Counting the number of scenarios

An optimal solution for sorting genome A into B is a sequence of genomes separated by basic operations, going from A to B with minimum cost. Some authors call such sequence a *geodesic* between A and B, or a *geodesic patch* when the basic operations have different weights [8]. Braga and Stoye present a similiar definition for *sorting scenarios* in their work, as a sequence of the operations involved in sorting A into B. Meidanis, Biller, and Zanetti [11], define sorting scenarios as sequences of genomes. However, for our purposes here, it is more convenient to define a corresponding *operation scenario* from A to B as a list of matrices $\mathcal{L} = [X_1, \ldots, X_\ell]$ such that $A + X_1 + X_2 + \ldots + X_\ell = B$, and each matrix X_i is one of the basic operations and applicable to $A + X_1 + X_2 + \ldots + X_{i-1}$. The *total weight* of a scenario, denoted by $w(\mathcal{L})$, is the sum of the weights of all its operations. Such scenario is *optimal* if and only if $w(\mathcal{L}) = d(A, B)$. In the context of the rank distance, scenarios are geodesic patches, because basic operations can have weight 1 or 2.

In this section, we show how to count the number of optimal rank sorting scenarios between two genomes, with the aid of the breakpoint graph. First, we show that no optimal operation acts on different components of the breakpoint graph. Therefore, we can solve each component separately. Then we recall from the work of Braga and Stoye [3] a formula to count DCJ sorting scenarios which also applies to rank sorting scenarios for cycles, and a recurrence to count rank sorting scenarios for paths. Lastly, we show how to get a count for the whole graph altogether.

4.1 Recombination

As a first step to count the number of sorting scenarios, we want to prove a useful property, namely, that no optimal rearrangement recombines the components of the breakpoint graph. In other words, we show here that there is no optimal operation that acts on extremities of more than one component at the same time.

Lemma 1. *No optimal operation in sorting from A to B involves extremities in different components of $BG(A, B)$.*

Proof. To prove this, we list every possible operation recombinating two components C_1 and C_2 of $BG(A, B)$ and show that they do not change the graph in a way that reduces the distance.

A double swap can be between two cycles, generating one cycle, between a cycle and a path, generating a path, or between two paths, resulting in two paths. All of these moves reduce the number of components or keep their number unchanged and therefore are not optimal.

The other option of operation on two separate components is to join two paths, resulting in a single path, reducing the number of components, also non optimal.

A cut is not considered here, because it only acts on two connected extremities. Therefore, it never affects more than one component. □

In the paper counting DCJ scenarios, the only operation that recombines different components of the graph is a swap between two unbalanced paths resulting in two balanced ones [3]. With regard to the breakpoint graph, the difference between the DCJ and the rank distances is that, under the rank distance, every path counts towards reducing the distance, not just balanced ones. Because of this difference, recombinating two unbalanced paths into two balanced ones is not an optimal move in a rank sorting scenario.

With this result, we conclude that it is possible to count the optimal scenarios for each component in the breakpoint graph independently. Now we determine how to obtain sorting scenarios for the whole graph from the separate solutions for each component.

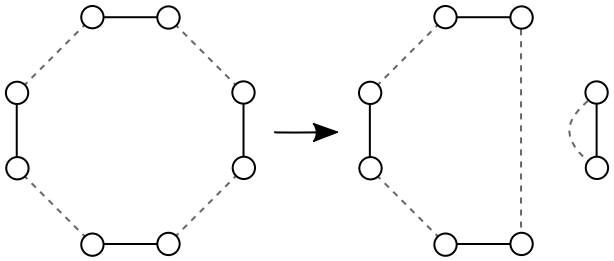


Figure 3: One example of optimal rearrangement for a cycle. A double swap splits the cycle into two smaller ones. In this case, one 8-cycle is decomposed into one 6-cycle (that will require two extra swaps to complete the sorting), and a 2-cycle, that is already sorted.

Let s_1 and s_2 be scenarios for the components C_1 and C_2 respectively, with respective lengths ℓ_1 and ℓ_2. The number of scenarios resulting in the combination of s_1 and s_2 is the number of sequences that have both as subsequences, and this is the shuffle product of s_1 and s_2, whose size is given by the binomial coefficient $\binom{\ell_1+\ell_2}{\ell_1,\ell_2} = \frac{(\ell_1+\ell_2)!}{\ell_1!\ell_2!}$. In general, the number of sequences obtained by shuffling k subsequences is given by the multinomial coefficient $\binom{\ell_1+\ell_2+\ldots+\ell_k}{\ell_1,\ell_2,\ldots,\ell_k} = \frac{(\ell_1+\ell_2+\ldots+\ell_k)!}{\ell_1!\ell_2!\ldots\ell_k!}$, where ℓ_i is the length of the ith subsequence.

4.2 Cycles

Given a cycle in $BG(A, B)$, the only optimal operation that can be applied to it is a double swap that splits the cycle into two smaller ones, as illustrated in Figure 3. This is a rank-two operation that increases the number of cycles by one, therefore decreasing the distance by two.

In this case, the sorting moves are equivalent to the ones for the DCJ distance. We can use the same formula for DCJ to compute the number $S_c(2\ell + 2)$ of scenarios to solve a cycle of length $2\ell + 2$ [3, Theorem 3]:

$$S_c(2\ell + 2) = (\ell + 1)^{(\ell-1)}$$

Each of these $S_c(2\ell + 2)$ scenarios has length ℓ, and total weight 2ℓ. When A and B are co-tailed genomes (that is, A and B have the same free ends), we have that in $BG(A, B)$ the only big components are cycles, and we can compute the total number S_{ct} of optimal sorting scenarios from A to B [3, Theorem 4] as follows:

$$S_{ct} = \frac{(\ell_1 + \ell_2 + \ldots + \ell_c)!}{\ell_1!\ell_2!\ldots\ell_c!} \prod_{i=1}^{c} (\ell_i + 1)^{\ell_i - 1},$$

where c is the number of big cycles in $BG(A, B)$.

4.3 Paths

We have shown a simple formula to compute the number of scenarios for the cyclic components in $BG(A, B)$. For paths the computation is less straightforward. The obstacle that arises when sorting paths is that, because cuts and joins have rank 1, while double swaps have rank 2, scenarios have variable length, and information on the length of the sub-solutions is necessary for the shuffling. Thus, we need a recurrence with two variables: the length of the path, and the length of the scenario.

For a path, there are three possible optimal operations. First, a cut of any A-edge, provided there is at least one A-edge in the path. Such a cut results in two smaller paths that are solved

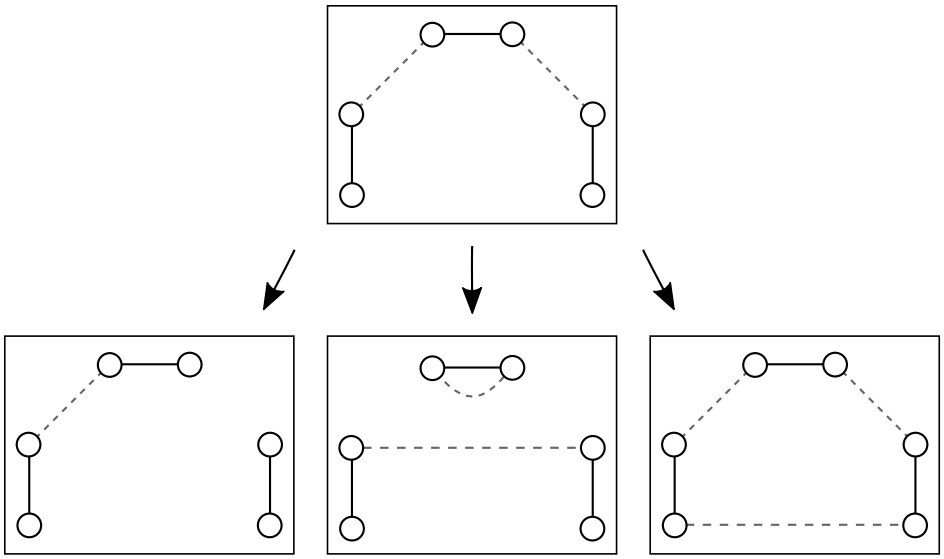


Figure 4: Examples of the three options of optimal moves from a path with black ends. The first one, from the top, is a cut, splitting the 5-path into two paths with 3 and 1 edges, respectively. The second is a double swap extracting a cycle from the path. The last option, only possible in this kind of path, is to join the ends, forming a cycle.

separately. A second option is to execute a double swap on any two A-edges, resulting in a cycle and a path, provided there are at least two A-edges. Here again, both new components have independent scenarios that are then shuffled. A last possible operation is to join the ends of a path, if both ends are incident to B-edges. This leads to a single cycle, that we already know how to process. These possibilities are illustrated in Figure 4.

Given a path with length k, the sizes of the optimal scenarios fall in a limited range. The longest operation scenarios for a k-path are the ones with only cuts and joins, making up a total of k operations. Since the double swaps have twice the weight of a cut or join, scenarios with more swaps are shorter. Their minimum length is defined in the following lemma.

Lemma 2. *Let A and B be two genomes over the same genes such that the sole big component of $BG(A, B)$ is a k-path P. The minimum length of a scenario that sorts A into B is $\left\lfloor \frac{k}{2} \right\rfloor + 1$.*

Proof. We will show this by induction on k. Since P is a big path, we know that $k \geq 1$. A path of length 1 is always sorted with one operation, either a cut or a join. A path of length 2 is always sorted with two operations, a cut and a join.

Let us now treat the case $k \geq 3$. Assuming that any path with length $k' < k$ has a minimum length of $\lfloor k'/2 \rfloor + 1$ for its scenarios, we are going to compute the minimum possible length ℓ for the scenarios of a k-path.

There are three different types of operations that can be applied to P: double swaps, cuts, and joins. A double swap produces a cycle of length $2x < k$ and a path of length $k - 2x$, where $x \geq 1$. The length of any scenario for the cycle is $x - 1$, and by the induction hypothesis, the length of a

solution for the path is at least $\left\lfloor \frac{k-2x}{2} \right\rfloor + 1$. Therefore, we have

$$\begin{aligned}
\ell &\geq 1 + x - 1 + \left\lfloor \frac{k-2x}{2} \right\rfloor + 1 \\
&= 1 + x + \left\lfloor \frac{k}{2} \right\rfloor - x \\
&= \left\lfloor \frac{k}{2} \right\rfloor + 1.
\end{aligned}$$

If the operation applied is a cut, the result is two paths, with lengths $x < k$ and $k - x - 1$, where $x \geq 0$. Here we will make use of a property of the floor function: $\lfloor a + b \rfloor \leq \lfloor a \rfloor + \lfloor b \rfloor + 1$ [12, Theorem 4.1], so that we have

$$\begin{aligned}
\ell &\geq 1 + \left\lfloor \frac{x}{2} \right\rfloor + 1 + \left\lfloor \frac{k-x-1}{2} \right\rfloor + 1 \\
&\geq \left\lfloor \frac{k-1}{2} \right\rfloor + 2 \\
&\geq \left\lfloor \frac{k}{2} \right\rfloor + 1.
\end{aligned}$$

The last option is a join. In this case, the resulting component is a cycle with $k+1$ edges, and it requires a scenario with $\frac{k+1}{2} - 1$ operations. Noting that k is odd in this case, we have

$$\begin{aligned}
\ell &= 1 + \frac{k+1}{2} - 1 \\
&= \left\lfloor \frac{k}{2} \right\rfloor + 1.
\end{aligned}$$

For all three types of operations, ℓ has the lower bound we are looking for of $\left\lfloor \frac{k}{2} \right\rfloor + 1$, proving the lemma. □

With this set of optimal operations, and the range for the length of a scenario, we arrive at three recurrences, one for each type of path. Since the cuts and swaps are only applied to A-edges, the indices are different according to the type of the path, and the paths obtained after splitting also have different types.

First, for balanced paths. Let's assume, without loss of generality, that the first edge (zero-indexed), is from A. That means that the A-edges are the ones with even indexes. When one of them is cut, one of the sub-paths is balanced, while the other begins and ends with B-edges. When a double swap is applied, the remaining path is always the same type of the original, is this case, balanced. This way, we write the following recurrence $S_{AB}(k, \ell)$ for the number of scenarios of length ℓ that solve a balanced path with k edges. The base case is when the path is just an isolated vertex ($k = 0$), that does not need any operation ($\ell = 0$).

$$
\begin{aligned}
S_{AB}(0,0) =& 1 \\
S_{AB}(k,\ell) =& \sum_{i=0}^{\frac{k}{2}-1} \sum_{j=i+1}^{2i} S_{AB}(2i,j) S_{BB}(k-2i-1,\ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=0}^{\frac{k}{2}-1} \sum_{z=1}^{\frac{k}{2}-x-1} z^{(z-2)} S_{AB}(k-2z,\ell-z) \binom{\ell-1}{z-1}
\end{aligned}
$$

The AA-paths are processed similarly. Their A-edges are even indexed, but both sub-paths after a cut are balanced. We then get the recurrence $S_{AA}(k,\ell)$ for the number of scenarios of length ℓ that solve a gray path with k edges. The base case, not necessary here but left for clarity, is a single gray edge ($k=1$), that allows a cut ($\ell=1$).

$$
\begin{aligned}
S_{AA}(1,1) =& 1 \\
S_{AA}(k,\ell) =& \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \sum_{j=i+1}^{2i} S_{AB}(2i,j) S_{AB}(k-2i-1,\ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=0}^{\lfloor \frac{k}{2} \rfloor} \sum_{z=1}^{\lfloor \frac{k}{2} \rfloor - x} z^{(z-2)} S_{AA}(k-2z,\ell-z) \binom{\ell-1}{z-1}
\end{aligned}
$$

On the other hand, BB-paths have odd numbered A-edges, and a cut always results in two BB-paths. Furthermore, as we saw before, this type of path has the extra option of joining the ends and making it a $(k+1)$-cycle, adding another term to the sum. So, the third recurrence $S_{BB}(k,\ell)$ counts the number of scenarios of length ℓ that solve a BB-path with k edges. The base case is a single B-edge ($k=1$), that only has one optimal sorting, joining its ends ($\ell=1$).

$$
\begin{aligned}
S_{BB}(1,1) =& 1 \\
S_{BB}(k,\ell) =& \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor - 1} \sum_{j=i+1}^{2i+1} S_{BB}(2i+1,j) S_{BB}(k-2i,\ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=1}^{\lfloor \frac{k}{2} \rfloor} \sum_{z=1}^{\lfloor \frac{k}{2} \rfloor - x} z^{(z-2)} S_{BB}(k-2z,\ell-z) \binom{\ell-1}{z-1} + \\
& \left(\frac{k+1}{2} \right)^{\left(\frac{k+1}{2}-2\right)}
\end{aligned}
$$

Notice how the recurrences have similar structures. We now show how it is possible to use a single recurrence for paths in general. First, we prove that $S_{AA}(k,\ell) = S_{BB}(k,\ell)$, for every k and ℓ.

Lemma 3. *Given two genomes A and B, if $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ is an optimal operation scenario from A to B, then $\mathcal{L}' = [-X_\ell, -X_{\ell-1}, \ldots, -X_1]$ is an optimal operation scenario from B to A.*

Proof. Let X_i be an arbitrary operation in $\mathcal{L}$. It is a basic operation, either a cut, a join, or a double swap. In $\mathcal{L}'$, we have the same operations as in $\mathcal{L}$, but negated, and in reverse order.

All we have to prove is that $B - X_\ell - X_{\ell-1} - \ldots - X_i$ is a genome, for every i such that $1 \leq i \leq \ell$. But $B - X_\ell - X_{\ell-1} - \ldots - X_i = A + X_1 + X_2 + \ldots + X_{i-1}$, which is a genome by hypothesis.

From this, we conclude that the operation $-X_i$ is also a basic operation with the same rank as X_i, and it can be applied to $B - X_\ell - X_{\ell-1} - \ldots - X_{i+1}$. Applying the same reasoning for every operation in $\mathcal{L}$ (and $\mathcal{L}'$), we conclude that $\mathcal{L}'$ is an operation scenario, with $w(\mathcal{L}') = w(\mathcal{L}) = d(A, B) = d(B, A)$. □

Lemma 4. *The number of optimal operation scenarios with ℓ operations for a gray path is the same as the number of solutions for a black path of same number of edges.*

Proof. Let A and B be two genomes such that, to sort A into B, when we build the graph $BG(A, B)$ we get a gray path with $2k+1$ vertices, that starts and ends with edges from the source genome A.

Now let us reverse the direction of the sorting. To sort B into A, we get a graph $BG(B, A)$ that is also a path with $2k + 1$ vertices, that starts and ends with edges from the target genome, A, so now it is a black path. From Lemma 3, we know that for every optimal operation scenario $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ from A to B, there is an optimal scenario $\mathcal{L}' = [X_\ell, X_{\ell-1}, \ldots, X_1]$ from B to A. Therefore, for every scenario sorting the gray $2k + 1$-path in $BG(A, B)$ with ℓ operations, there is a corresponding scenario sorting the black $2k + 1$-path in $BG(B, A)$, also with ℓ operations. □

Because of Lemma 4, we know that $S_{AA}(k, \ell) = S_{BB}(k, \ell)$, for every k and ℓ. That is, we can compute the recurrence for unbalanced (odd) paths, independent of the genome most represented in them. As we now have a recurrence for odd paths only, and another exclusively for even ones, we can further simplify them into a single recurrence relation for the number $S_p(k, \ell)$ of optimal sorting scenarios of length ℓ for a path with k edges.

$$
\begin{aligned}
S_p(0,0) =& 1 \\
S_p(k,\ell) =& \sum_{i=0}^{\lceil k/2 \rceil - 1} \sum_{j=i+1}^{2i} S_p(2i, j) S_p(k - 2i - 1, \ell - j - 1) \binom{\ell - 1}{j} + \\
& \sum_{x=0}^{\lceil k/2 \rceil - 1} \sum_{z=1}^{\lceil k/2 \rceil - x - 1} z^{(z-2)} S_p(k - 2z, \ell - z) \binom{\ell - 1}{z - 1}
\end{aligned}
$$

With $S_c(k)$ and $S_p(k, \ell)$, we can count the total number of scenarios between any two genomes. If the graph $BG(A, B)$ has p big cycles with lengths $2\ell_1 + 2, \ldots, 2\ell_p + 2$, and q big paths with lengths $k_1, \ldots, k_q$, the total number of optimal sorting scenarios from A to B is given by:

$$
\sum_{\ell'_1 = \lfloor k_1/2 \rfloor + 1}^{k_1} \cdots \sum_{\ell'_q = \lfloor k_q/2 \rfloor + 1}^{k_q} \frac{(\ell_1 + \ldots + \ell_p + \ell'_1 + \ldots + \ell'_q)!}{\ell_1! \ldots \ell_p! \ell'_1! \ldots \ell'_q!} \prod_{i=1}^{p} S_c(2\ell_i + 2) \prod_{j=1}^{q} S_p(k_j, \ell'_j).
$$

5 Intermediate genomes

We say a genome B is an intermediate genome between genomes A and C when

$$d(A, C) = d(A, B) + d(B, C).$$

We have already shown that there is never recombination of different components in the rank distance. Therefore, we can get all possible indermediates by looking separately at each component of $BG(A, C)$. If the graph $BG(A, C)$ has p big cycles with lengths $k_1, k_2, \ldots, k_p$, and q big paths with lengths $k'_1, k'_2, \ldots, k'_q$, the total number $I(A, C)$ of intermediate genomes between A and C is

$$I(A, C) = \prod_{i=1}^{p} I_c(k_i) \prod_{j=1}^{q} I_p(k'_j),$$

where $I_c(k)$ is the number of intermediates for a k-cycle, and $I_p(k)$ is the number of intermediates for a k-path.

For cycles of length $2k$, with $k \geq 1$, rank optimal operations are the same as DCJ optimal operations, so we can use the analogous result for the DCJ distance [5]:

$$I_c(2k) = \frac{1}{k+1}\binom{2k}{k}.$$

For paths, again we have more options than when sorting by DCJ. In order to prove the needed results for paths in $BG(A, C)$, we will label their nodes. Given an alternating path P with m vertices in $BG(A, C)$, we will label its vertices $v_1^P, v_2^P, \ldots, v_m^P$, or just $v_1, v_2, \ldots, v_m$ when the path P is made clear by the context. We say an edge $\{v_i, v_{i+x}\}$ has *span* x with respect to P. If v_1 or v_m are incident on a black (gray) edge, they are called a *black (gray) end* of the path.

Lemma 5. *Let A and C be two genomes over the same genes such that the big components of $BG(A, C)$ consists solely of one path $v_1, \ldots, v_m$. If B is an intermediate genome such that $BG(B, C)$ has a cycle containing the edge $\{v_i, v_{i+2k}\}$, this cycle has at least one other edge of even span.*

Proof. To form an alternating cycle of length $2l$ in $BG(B, C)$, there are l C-edges and l B-edges. Since B is an intermediate between A and C, the cycle cannot contain extremities that are part of other components in $BG(A, C)$. The C-edges are fixed, disjoint, and each of them incides on one even-indexed extremity and one odd-indexed extremity. Therefore, in the cycle, there are l even vertices and l odd vertices.

To complete the cycle, the l B-edges have to cover all $2l$ vertices. The edge $\{v_i, v_{i+2k}\}$ incides on two vertices of the same parity. The remaining $l-1$ B-edges have to cover $l-2$ vertices of one parity, and l of the other. Therefore, by the pidgeonhole principle, at least one B-edge incides on two vertices of the opposite parity from i. This edge has even span. □

Lemma 6. *If A and C are two genomes over the same genes and the sole big component of $BG(A, C)$ is a path, then no intermediate genome B between A and C has an adjacency with even span.*

Proof. Let $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ be a scenario where the genome $B_j = A + X_1 + X_2 + \ldots + X_j$ has the adjacency $\{v_i, v_{i+2k}\}$. Assume without loss of generality that B_j is the first genome generated by $\mathcal{L}$ that has an even span adjacency. The operation X_j is either a join of v_i and v_{i+2k} or a double swap creating $\{v_i, v_{i+2k}\}$ and another adjacency.

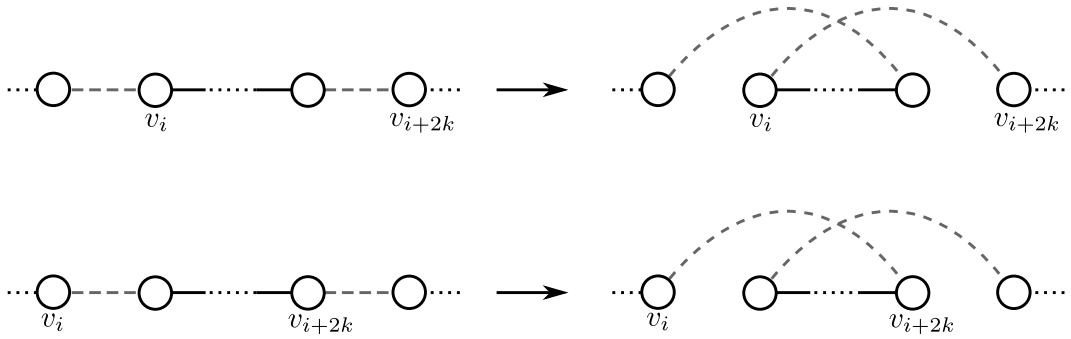


Figure 5: Two possible configurations for a double swap that creates the adjacency $v_i v_{i+2k}$. On the left it is shown the breakpoint graph before the swap, and on the right is the breakpoint graph after the swap. The two gray (dashed) edges are replaced by $v_i v_{i+2k}$ and another adjacency. In both cases, it is possible to notice that the number of cycles and paths is not affected by the swap, it merely reverses a part of a component.

If X_j is a double swap, there are two possibilities for it, illustrated in Figure 5. For both of these options, the swap merely reverses a part of the component, and the number of paths and cycles in $BG(B_{j-1}, C)$, where $B_{j-1} = B_j - X_j$, is the same as in $BG(B_j, C)$. Thus, X_j is not a sorting operation, a contradiction.

If X_j is a join, it joins the ends of a CC-path P in $BG(B_{j-1}, C)$, closing a cycle with the edge $\{v_i, v_{i+2k}\}$. According to Lemma 5, this cycle has another gray edge of even span. This edge was already present in P in $BG(B_{j-1}, C)$, which contradicts the hypothesis that B_j is the first genome generated by $\mathcal{L}$ that has an even distance adjacency.

Therefore, intermediate genomes cannot have even-span edges. □

With the help of Lemma 6, we characterize the intermediate genomes of a path in $BG(A, C)$.

Theorem 2. *If A and C are two genomes over the same genes and $BG(A, C)$ consists solely of one k-path $v_1, v_2, \ldots, v_{k+1}$ and small components, a matching B is an intermediary genome between A and C if and only if:*

- *B has the same edges of A and C outside $\{v_1, \ldots, v_{k+1}\}$;*
- *all edges in B involving vertices $v_1, \ldots, v_{k+1}$ have odd span;*
- *the edges in B do not cross one another, that is, there are no two edges $\{v_i, v_j\}$ and $\{v_{i'}, v_{j'}\}$ such that $i < i' < j < j'$;*
- *for every edge $\{v_i, v_j\}$ in B, with $i < j$, all vertices $v_{i+1}, \ldots, v_{j-1}$ are in an adjacency.*

Proof. Let's begin by showing that an intermediate genome satisfies the conditions. The first condition ensures that all small components of $BG(A, C)$ are present in both $BG(A, B)$ and $BG(B, C)$. From Lemma 6, we know that only edges with odd span are possible in an intermediate.

At any point during sorting, if an operation X would add an edge that crosses another, this would be an operation that recombines two separate components, and hence cannot be optimal. So, there are no crossing edges.

As for the fourth and last condition, if an edge $\{v_i, v_j\}$ has span greater than 1, it means that $\{v_i, v_j\}$ was created by a double swap or a join, and all vertices $v_{i+1}, \ldots, v_{j-1}$ are part of a cycle. As we saw in Section 4.2, all vertices in a cycle have to be part of an adjacency in all intermediates.

Now, it is necessary to show that any matching B that fulfills all four conditions is an intermediate between A and C. We will do so using induction on the number of edges of B.

The base case is when B has no edges other than the common edges of A and C. The matching B is an intermediate between A and C, because with a path it is always possible to start at A, reach B by cutting all A-edges incident to any of $v_1, \ldots, v_m$, and then join at all C-edges incident to $v_1, \ldots, v_m$, optimally arriving at C.

In the general case, B has one or more edges incident to $v_1, \ldots, v_m$. Cut an edge of B with maximum span, obtaining a genome B', with $d(B, B') = 1$. Genome B' satisfies all four conditions and, therefore, by the induction hypothesis, is an intermediate genome between A and C. Note that $d(B, A)$ is either $d(B', A) + 1$ or $d(B', A) - 1$, and the same applies to $d(B, C)$. However, B cannot be closer than B' to both A and C, since this would contradict the triangle inequality. We just have to show that B is not farther than B' from both A and C.

Let $\{v_i, v_j\}$ be the edge of B cut to generate B'. We know that this edge has odd span, and therefore, the edges $\{v_i, v_{i+1}\}$ and $\{v_{j+1}, v_j\}$ are either both A-edges or both C-edges. Suppose they are A-edges. In this case, the cut from B to B' cut a cycle in $BG(B, A)$, making $d(B, A) < d(B', A)$, or $d(B, A) = d(B', A) - 1$. On the other hand, $\{v_i, v_j\}$ is part of a path in $BG(B, C)$, and the cut splits this path, making $d(B, C) = d(B', C) + 1$.

In the case where $\{v_i, v_{i+1}\}$ and $\{v_{j+1}, v_j\}$ are C-edges, the same reasoning applies, and we have $d(B, A) = d(B', A) + 1$, and $d(B, C) = d(B', C) - 1$. In both cases, $d(B, A) + d(B, C) = d(B', A) + d(B', C) = d(A, C)$, and that makes B an intermediate. □

The characterization of path intermediates given by Theorem 2 allows us to enumerate all possible intermediates using a recursion. Given a path with length k formed by the vertices $v_1, v_2, \ldots, v_{k+1}$, consider all possible adjacencies incident to v_1. They are $\{v_1, v_2\}, \{v_1, v_4\}$, and so on (pairs involving v_1 and an even-indexed vertex).

There are four different cases. The first case is when v_1 has no adjacency. In this case, the number of intermediates is the number of intermediates for the path that goes from v_2 to v_{k+1}. Therefore, there are $I_p(k-1)$ intermediates.

Then, we consider the adjacency with the smallest span $\{v_1, v_2\}$. There is no vertex "under" the adjacency that needs to be considered, that is, no vertex between v_1 and v_2, so the number of intermediates is $I_p(k-2)$.

Another case is when considering the adjacency with the largest possible span, $\{v_1, v_{2\lceil k/2 \rceil}\}$. The vertices $v_2, \ldots, v_{2\lceil k/2 \rceil - 1}$ are "under" the adjacency, and, according to Theorem 2, the number of intermediates for them is equivalent to the number of intermediates of a cycle that covers $2\lceil k/2 \rceil - 2$ vertices, $I_c(2\lceil k/2 \rceil - 2)$. There is at most one vertex after $v_{2\lceil k/2 \rceil}$, so no adjacency can exist "outside" $\{v_1, v_{2\lceil k/2 \rceil}\}$ and contribute to the number of intermediates. Thus, the total number of intermediates for this case is $I_c(2\lceil k/2 \rceil - 2)$.

Finally, for each possible adjacency $\{v_1, v_{2i}\}$, with $1 < i < \lceil k/2 \rceil$, the number of intermediates containing this adjacency is the number of intermediates for $v_2, \ldots, v_{2i-1}$ times the number of intermediates for $v_{2i+1}, \ldots, v_{k+1}$. According to Theorem 2, the first factor is equivalent to the number of intermediates of a cycle that covers $v_2, \ldots, v_{2i-1}$. The second factor equals the number of intermediates of the remaining path $v_{2i+1}, \ldots, v_{k+1}$. Therefore, for every adjacency $\{v_1, v_{2i}\}$, $1 < i < \lceil k/2 \rceil$, there are $I_c(2i-2)I_p(k-2i)$ intermediates.

The base cases are paths with lengths 0, 1, or 2. A 0-path (a single vertex) means both genomes are equal, so there is only one intermediate. A path with one edge has two intermediates, with and without that adjacency. A path with two edges can have either of the adjacencies, or none of them, adding up to three intermediates.

From this construction, we arrive at the following recurrence $I_p(k)$ for the number of intermediates of a path with length k:

$$
\begin{aligned}
&I_p(0) = 1\\
&I_p(1) = 2\\
&I_p(2) = 3\\
&I_p(k) = I_p(k-1) + I_p(k-2) + I_c(2\lceil k/2\rceil - 2) + \sum_{i=2}^{\lceil k/2\rceil - 1} I_c(2i-2) I_p(k-2i)
\end{aligned}
$$

With the help of the OEIS, we were able to relate I_p to the sequence A001405 [13] in a way that suggests the closed formula for $I_p(k)$ with $k \geq 0$:

$$I_p(k) = \binom{k+1}{\lfloor (k+1)/2 \rfloor}.$$

We will now show that this formula satisfies the recurrence for I_p.

Lemma 7. *For $m \geq 1$,*

$$\sum_{i=1}^{m-1} \frac{2(m-2i-1)}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} = -2\binom{2m-2}{m}.$$

Proof. First we want to write the terms of the sum as a difference that will allow us to cancel inner terms.

$$
\begin{aligned}
&\binom{2i+2}{i+1}\binom{2m-2i-2}{m-i-1} - \binom{2i}{i}\binom{2m-2i}{m-i} =\\
&= \frac{2i+2}{i+1}\binom{2i+1}{i}\frac{m-i}{2m-2i-1}\binom{2m-2i-1}{m-i} - \binom{2i}{i}\binom{2m-2i}{m-i}\\
&= 2\frac{2i+1}{i+1}\binom{2i}{i}\frac{m-i}{2m-2i-1}\frac{m-i}{2m-2i}\binom{2m-2i}{m-i} - \binom{2i}{i}\binom{2m-2i}{m-i}\\
&= \frac{(2i+1)(m-i) - (i+1)(2m-2i-1)}{(i+1)(2m-2i-1)}\binom{2i}{i}\binom{2m-2i}{m-i}\\
&= \frac{2mi - 2i^2 + m - i - 2mi + 2i^2 + i - 2m + 2i + 1}{(i+1)(2m-2i-1)}\binom{2i}{i}\binom{2m-2i}{m-i}\\
&= -\frac{m-2i-1}{(i+1)(2m-2i-1)}\binom{2i}{i}\binom{2m-2i}{m-i}\\
&= -\frac{m-2i-1}{i(2m-2i-1)}\binom{2i}{i-1}\binom{2m-2i}{m-i}\\
&= -\frac{m-2i-1}{i(2m-2i-1)}\binom{2i}{i-1}2\binom{2m-2i-1}{m-i-1}\\
&= -\frac{2(m-2i-1)}{i(2m-2i-1)}\binom{2i}{i-1}\frac{2m-2i-1}{m-i}\binom{2m-2i-2}{m-i-1}\\
&= -\frac{2(2m-2i-1)}{i(m-i)}\binom{2i}{i-1}\binom{2m-2i-2}{m-i-1}
\end{aligned}
$$

We will now substitute the expression inside the sum.

$$\begin{aligned}
&\sum_{i=1}^{m-1} \frac{2(m-2i-1)}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} = \\
&= -\sum_{i=1}^{m-1} \left[\binom{2i+2}{i+1} \binom{2m-2i-2}{m-i-1} - \binom{2i}{i} \binom{2m-2i}{m-i} \right] \\
&= -\binom{2m}{m} \binom{0}{0} + \binom{2}{1} \binom{2m-2}{m-1} \\
&= 2\binom{2m-2}{m-1} - 2\binom{2m-1}{m-1} \\
&= 2\binom{2m-2}{m-1} - \frac{2(2m-1)}{m} \binom{2m-2}{m-1} \\
&= 2\binom{2m-2}{m-1} \left[1 - \frac{2m-1}{m} \right] \\
&= 2\binom{2m-2}{m-1} \frac{m-2m+1}{m} \\
&= -2\frac{m-1}{m} \binom{2m-2}{m-1} \\
&= -2\binom{2m-2}{m}
\end{aligned}$$

□

Lemma 8. *For $m \geq 2$,*

$$\sum_{i=1}^{m-1} \frac{1}{i} \binom{2i}{i-1} \binom{2m-2i}{m-i} = 2\binom{2m-1}{m-2}.$$

Proof. Let

$$f(m) = \sum_{i=1}^{m-1} \frac{1}{i} \binom{2i}{i-1} \binom{2m-2i}{m-i},$$

for $m \geq 2$. We will use induction on m to show that $f(m) = 2\binom{2m-1}{m-2}$.

For $m = 2$, we have

$$f(2) = \sum_{i=1}^{1} \frac{1}{i} \binom{2i}{i-1} \binom{4-2i}{2-i} = \binom{2}{0} \binom{2}{1} = 2 = 2\binom{3}{0}.$$

Assume that $f(m') = 2\binom{2m'-1}{m'-2}$ for $2 \leq m' < m$.

$$\begin{aligned} f(m) &= \sum_{i=1}^{m-1} \frac{1}{i} \binom{2i}{i-1} \binom{2m-2i}{m-i} \\ &= \sum_{i=1}^{m-1} \frac{1}{i} \binom{2i}{i-1} \frac{2(2m-2i-1)}{m-i} \binom{2m-2i-2}{m-i-1} \\ &= \sum_{i=1}^{m-1} \frac{2(2m-2i-1)}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} \end{aligned}$$

We can now separate this summation in two terms, as follows:

$$\begin{aligned} f(m) = &\sum_{i=1}^{m-1} \frac{2m}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} \\ &+ \sum_{i=1}^{m-1} \frac{2(m-2i-1)}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} \end{aligned}$$

Lemma 7 gives us the sum of the second term. Then:

$$f(m) = \sum_{i=1}^{m-1} \frac{2m}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} - 2\binom{2m-2}{m}$$

Let

$$g(m) = \sum_{i=1}^{m-1} \frac{2}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1}.$$

$$f(m) = mg(m) - 2\binom{2m-2}{m} \tag{1}$$

But $f(m)$ can also be separated in another way:

$$\begin{aligned} f(m) &= \sum_{i=1}^{m-1} \frac{2}{i} \left(2 - \frac{1}{m-i}\right) \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} \\ &= 4 \sum_{i=1}^{m-1} \frac{1}{i} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} - \sum_{i=1}^{m-1} \frac{2}{i(m-i)} \binom{2i}{i-1} \binom{2m-2i-2}{m-i-1} \\ &= \frac{4}{m-1} \binom{2m-2}{m-2} + 4f(m-1) - g(m) \end{aligned} \tag{2}$$

Equating the formulas for $f(m)$ given by Equations (1) and (2) and using the induction hypothesis, we can now get to a value for $g(m)$:

$$
\begin{aligned}
mg(m) - 2\binom{2m-2}{m} &= \frac{4}{m-1}\binom{2m-2}{m-2} + 4f(m-1) - g(m) \\
(m+1)g(m) &= \frac{4}{m-1}\binom{2m-2}{m-2} + \frac{8}{m-1}\binom{2m-3}{m-3} + 2\binom{2m-2}{m} \\
(m+1)g(m) &= \frac{4}{m-1}\binom{2m-2}{m-2} + \frac{8(m-2)}{2m-2}\binom{2m-2}{m-2} + 2\binom{2m-2}{m-2} \\
(m+1)g(m) &= \left(\frac{4}{m-1} + \frac{4(m-2)}{m-1} + 2\right)\binom{2m-2}{m-2} \\
(m+1)g(m) &= \frac{4+4m-8+2m-2}{m-1}\binom{2m-2}{m-2} \\
(m+1)g(m) &= \frac{6m-6}{m-1}\binom{2m-2}{m-2} \\
g(m) &= \frac{6}{m+1}\binom{2m-2}{m-2}
\end{aligned}
$$

Now we can finally get a value for $f(m)$:

$$
\begin{aligned}
f(m) &= mg(m) - 2\binom{2m-2}{m} \\
f(m) &= \frac{6m}{m+1}\binom{2m-2}{m-2} - 2\binom{2m-2}{m} \\
f(m) &= \frac{6m-2m-2}{m+1}\binom{2m-2}{m-2} \\
f(m) &= \frac{4m-2}{m+1}\binom{2m-2}{m-2} \\
f(m) &= \frac{2(2m-1)}{m+1}\frac{(2m-2)!}{(m-2)!m!} \\
f(m) &= 2\frac{(2m-1)!}{(m-2)!(m+1)!} \\
f(m) &= 2\binom{2m-1}{m-2}
\end{aligned}
$$

□

Theorem 3. *For $k \geq 0$,*

$$I_p(k) = \binom{k+1}{\lfloor (k+1)/2 \rfloor}.$$

Proof. We want to show it by induction on k. For $k = 0, 1, 2$ we have that

$$
\begin{aligned}
I_p(0) &= 1 = \binom{1}{0} \\
I_p(1) &= 2 = \binom{2}{1} \\
I_p(2) &= 3 = \binom{3}{1}.
\end{aligned}
$$

Now, assume that for $k' < k$, the equation holds.
For the even case, $k = 2m$, and $m \geq 2$:

$$\begin{aligned} I_p(2m) &= I_p(2m-1) + I_p(2m-2) + I_c(2m-2) + \sum_{i=2}^{m-1} I_c(2i-2) I_p(2m-2i) \\ &= I_p(2m-1) + I_p(2m-2) + I_c(2m-2) + \sum_{i=1}^{m-2} I_c(2i) I_p(2m-2i-2) \end{aligned}$$

Applying the induction hypothesis, and substituting the values of I_c, we get

$$\begin{aligned} I_p(2m) &= \binom{2m}{m} + \binom{2m-1}{m-1} + \frac{1}{m}\binom{2m-2}{m-1} + \sum_{i=1}^{m-2} \frac{1}{i+1}\binom{2i}{i}\binom{2m-2i-1}{m-i-1} \\ &= \binom{2m}{m} + \binom{2m-1}{m-1} + \sum_{i=1}^{m-1} \frac{1}{i+1}\binom{2i}{i}\binom{2m-2i-1}{m-i-1} \\ &= \binom{2m}{m} + \binom{2m-1}{m-1} + \frac{1}{2}\sum_{i=1}^{m-1} \frac{1}{i+1}\binom{2i}{i}\binom{2m-2i}{m-i}. \end{aligned}$$

Applying Lemma 8, we get

$$I_p(2m) = \binom{2m}{m} + \binom{2m-1}{m-1} + \binom{2m-1}{m-2}$$

Finally, applying Pascal's rule

$$\begin{aligned} I_p(2m) &= \binom{2m}{m} + \binom{2m}{m-1} \\ I_p(2m) &= \binom{2m+1}{m} \end{aligned}$$

The odd case, $k = 2m + 1$, with $m \geq 1$:

$$
\begin{aligned}
I_p(2m+1) &= I_p(2m) + I_p(2m-1) + I_c(2m) + \sum_{i=2}^{m} I_c(2i-2) I_p(2m-2i+1) \\
&= I_p(2m) + I_p(2m-1) + I_c(2m) + \sum_{i=1}^{m-1} I_c(2i) I_p(2m-2i-1) \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \frac{1}{m+1}\binom{2m}{m} + \sum_{i=1}^{m-1} \frac{1}{i+1}\binom{2i}{i}\binom{2m-2i}{m-i} \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \frac{1}{m+1}\binom{2m}{m} + 2\binom{2m-1}{m-2} \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \frac{2m}{m(m-1)}\binom{2m-1}{m-2} + 2\binom{2m-1}{m-2} \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \binom{2m-1}{m-2}\left[\frac{2m}{m(m-1)} + 2\right] \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \binom{2m-1}{m-2}\frac{2m}{m-1} \\
&= \binom{2m+1}{m} + \binom{2m}{m} + \binom{2m}{m-1} \\
&= \binom{2m+1}{m} + \binom{2m+1}{m} \\
&= 2\binom{2m+1}{m} \\
&= \frac{2m+2}{m+1}\binom{2m+1}{m} \\
&= \binom{2m+2}{m+1}
\end{aligned}
$$

□

The number $I_p(k)$ is equivalent to the number of $\lfloor (k+1)/2 \rfloor$-element subsets of a set with $(k+1)$ elements. According to Sperner's Theorem [17, 10], this is the maximal number of subsets of a set with $k+1$ elements where no set contains another. Such family of sets is called a *Sperner family*. Building on this idea, we suggest a bijection between the intermediates of a k-path, and the Sperner family of all $\lfloor (k+1)/2 \rfloor$-element subsets of a $k+1$-set.

Algorithm 1: Algorithm to transform a set with $\lfloor (k+1)/2 \rfloor$ elements into an intermediate of a k-length path.

Data: Set S.
Result: The set E of adjacencies of the corresponding intermediate.
$T \leftarrow$ empty stack
$E \leftarrow \{\}$
for i *from* 0 *to* $(n-1)$ **do**
  if $i \in S$ **then**
    T.push(i)
  else
    if T *is not empty* **then**
      $x \leftarrow T$.pop()
      $E \leftarrow E \cup \{\{x, i\}\}$
return E

Algorithm 2: Algorithm to encode an intermediate of a k-length path into a set with $\lfloor (k+1)/2 \rfloor$ elements.

Data: The sets E of adjacencies and T of telomeres of an intermediate.
Result: The corresponding set.
$S \leftarrow \{\}$
for $\{x, y\} \in E$ **do**
  $S \leftarrow S \cup \{\min(x, y)\}$
$T' \leftarrow \lfloor |T|/2 \rfloor$ largest telomeres of T
return $S \cup T'$

6 Experiments

We implemented the formulas given here and tested our methods counting the number of scenarios and intermediates between pairs of a number of genomes found in the literature.

6.1 Data sets

We used four data sets from different sources. The first and simplest data set is from the work of Palmer and Hebron on plants of the Brassica genus [15]. It consists of two pairs of circular mitochondrial DNA, comparing *Brassica campestris* against *B. oleracea* and *B. napus*. Both instances have 5 synteny blocks. Other comparison in this work have insertions or deletions and were not considered.

Another data set is the human and mouse X chromosome, from Pevzner and Tesler [16]. This pair of linear, single-chromosome, inputs has 11 synteny blocks.

The third data set is composed of 12 chloroplast genomes from the Campanulaceae family, plus tobacco, with 105 synteny blocks, taken from the work of Cosner, Raubeson and Jansen [4], and also used by Bourque and Pevzner [2]. These were processed pairwise, making it a total of 78 instances of the sorting problem.

The fourth and largest data set contains genomes used by Kim et al. to test their reconstruction algorithm DESCHRAMBLER [9]. It consists of 20 instances comparing the human genome against the genome from other animals, namely, 18 eutherean mammals, plus opossum and chicken as outgroups. These pairs have between 101 and 621 synteny blocks.

Table 1: Distances, number of scenarios, and number of intermediates between the human genome and the genomes of four primates.

Genome	Distance	Scenarios	Intermediates
Chimpanzee	27	6.54×10^{11}	2.46×10^{4}
Orangutan	53	6.03×10^{38}	1.29×10^{10}
Rhesus	150	1.21×10^{138}	1.45×10^{28}
Marmoset	204	3.99×10^{250}	3.13×10^{43}

6.2 Code

The code to run our experiments was implemented in Python, and executed in a virtual machine using a single 2.3GHz processor core and 2GB of memory. Building the breakpoint graph of our tests instances, with up to 1242 extremities, is not a computational intensive task, and neither is computing the number of intermediates, a product of binomial coefficients.

The most demanding task is to compute the number of scenarios, especially when the breakpoint graph has a large number of paths. This is a consequence of the fact that each path multiplies the number of products to be computed. As a consequence, special attention must be dedicated to the generation of the lengths of the scenarios for each path. Considering more lengths than necessary can be very costly, and storing all values to multiply them later demands too much memory.

6.3 Results

The two Brassica instances have the same rank distance of 6 (in theses cases, three double swaps), and the same results: 9 scenarios and 10 intermediate genomes.

For the pair of X chromosomes, with a rank distance of 14, we get 237440 scenarios and 560 intermediates.

With the Campanulaceae pairs, we get varying results. Some pairs, like *Trachelium* and *Campanula* are only one swap apart, and therefore have only one optimal scenario and two intermediates (the input genomes). The pair of genomes that are farthest apart is *Merciera* and *Platycodon*, with a distance of 48. They have 1.4×10^{32} optimal scenarios between them, and a total of 4.9×10^{12} intermediate genomes.

With the eutherean data set, we reached a limit for our code to count the number of optimal scenarios. Out of the 20 instances, we only reached a result for 4 of them, in Table 1. On the other hand, computing the number of intermediates proved easy even for the farthest pairs of genomes. In Table 2 we list the number of intermediates for all the remaining instances.

6.4 Discussion

Comparing the larger instances of the Campanulaceae data set with the ones from the human, cat and mouse genomes, we note that linear and multichromosomal genomes tend to have more scenarios than circular unichromosomal ones. That is due to the fact that multiple paths in the breakpoint graph quickly add more factors to the scenario formula. The number of intermediates, on the other hand, seems to be less variable between instances with the same distance.

Table 2: Distances, and number of intermediates between the human genome and the genomes of 16 animals.

Genome	Distance	Intermediates
Cattle	383	7.39×10^{83}
Marmoset	204	3.13×10^{43}
Dog	304	6.37×10^{70}
Goat	393	5.30×10^{85}
Guinea pig	640	5.85×10^{166}
White rhinoceros	328	1.36×10^{84}
Tenrec	407	1.71×10^{105}
Horse	225	1.31×10^{51}
Chicken	736	6.38×10^{193}
Elephant	336	8.56×10^{86}
Mouse	509	2.44×10^{131}
Opossum	778	4.01×10^{204}
Pika	385	2.89×10^{98}
Chimpanzee	27	2.46×10^{4}
Orangutan	53	1.29×10^{10}
Rhesus	150	1.45×10^{28}
Rat	788	2.30×10^{189}
Shrew	487	1.02×10^{128}
Pig	318	1.61×10^{73}
Manatee	519	3.12×10^{142}

7 Conclusion

We opened the doors for the exploration of the solution space of the rank distance problem. We demonstrated that there is no recombination between components of the breakpoint graph in any optimal rank sorting scenario. We then showed a formula to compute the exact number of optimal sorting scenarios for co-tailed genomes, and a recurrence for the general case.

We also presented a formula for the number of intermediate genomes between any two genomes. Furthermore, we suggested a bijection that works as a very simple way to uniformly sample intermediates. Being able to sample intermediate genomes is the next step in this study of the solution space and can be very useful in future applications.

References

[1] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. *Algorithms in Bioinformatics*, pages 163–173, 2006.

[2] G. Bourque and P. A. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome research*, 12(1):26–36, 2002.

[3] M. D. Braga and J. Stoye. The solution space of sorting by DCJ. *Journal of Computational Biology*, 17(9):1145–1165, 2010.

[4] M. E. Cosner, L. A. Raubeson, and R. K. Jansen. Chloroplast DNA rearrangements in Campanulaceae: phylogenetic utility of highly rearranged genomes. *BMC evolutionary biology*, 4(1):1–17, 2004.

[5] P. Feijão. Reconstruction of ancestral gene orders using intermediate genomes. *BMC Bioinformatics*, 16(14):S3, Oct 2015.

[6] P. Feijão and J. Meidanis. Extending the algebraic formalism for genome rearrangements to include linear chromosomes. *IEEE/ACM transactions on computational biology and bioinformatics*, 10(4):819–831, 2013.

[7] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 581–592. IEEE, 1995.

[8] A. Jamshidpey, A. Jamshidpey, and D. Sankoff. Sets of medians in the non-geodesic pseudometric space of unsigned genomes with breakpoints. *BMC Genomics*, 15(6):S3, 2014.

[9] J. Kim, M. Farré, L. Auvil, B. Capitanu, D. M. Larkin, J. Ma, and H. A. Lewin. Reconstruction and evolutionary history of eutherian chromosomes. *Proceedings of the National Academy of Sciences*, 114(27):E5379–E5388, 2017.

[10] D. Lubell. A short proof of Sperner's lemma. *Journal of Combinatorial Theory*, 1(2):299, 1966.

[11] J. Meidanis, P. Biller, and J. P. P. Zanetti. A Matrix-Based Theory for Genome Rearrangements. Technical Report IC-17-11, Institute of Computing, University of Campinas, August 2017. In English, 45 pages.

[12] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An introduction to the theory of numbers*. John Wiley & Sons, 2008.

[13] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. `http://oeis.org/A001405`.

[14] A. Ouangraoua and A. Bergeron. Combinatorial structure of genome rearrangements scenarios. *Journal of Computational Biology*, 17(9):1129–1144, 2010.

[15] J. D. Palmer and L. A. Herbon. Plant mitochondrial DNA evolved rapidly in structure, but slowly in sequence. *Journal of Molecular evolution*, 28(1):87–97, 1988.

[16] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome research*, 13(1):37–45, 2003.

[17] E. Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, Dec 1928.

[18] E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC bioinformatics*, 10(1):120, 2009.

[19] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

[20] J. P. P. Zanetti, P. Biller, and J. Meidanis. Median approximations for genomes modeled as matrices. *Bulletin of Mathematical Biology*, 78(4):786–814, Apr 2016.