Counting All DCJ Sorting Scenarios

Marília D. V. Braga and Jens Stoye

Technische Fakultät, Universität Bielefeld, Germany.
mbraga@cebitec.uni-bielefeld.de, stoye@techfak.uni-bielefeld.de

Abstract. In genome rearrangements, the double cut and join (DCJ) operation, introduced by Yancopoulos *et al.*, allows to represent most rearrangement events that could happen in multichromosomal genomes, such as inversions, translocations, fusions and fissions. No restriction on the genome structure considering linear and circular chromosomes is imposed. An advantage of this general model is that it leads to considerable algorithmic simplifications. Recently several works concerning the DCJ operation have been published, and in particular an algorithm was proposed to find an optimal DCJ sequence for sorting one genome into another one. Here we study the solution space of this problem and give an easy to compute formula that corresponds to the exact number of optimal DCJ sorting sequences to a particular subset of instances of the problem. In addition, this formula is also a lower bound to the number of sorting sequences to any instance of the problem.

1 Introduction

Genome rearrangements provide the opportunity for tracking evolutionary events at a structural, whole-genome level. A typical approach is the determination of the minimum number of rearrangement operations that are necessary to transform one genome into another one [7]. The corresponding computational problem is called the *genomic distance problem* [5]. A bit more detailed is the task when in addition to the numeric distance also one or more scenarios of rearrangement operations are to be determined, the so-called *genomic sorting problem*.

Most algorithms that solve the genomic sorting problem will report just one out of a possibly very high number of rearrangement scenarios, and studies of such a particular scenario are not well suited for drawing general conclusions on properties of the relationship between the two genomes under study. Moreover, there are normally too many sorting scenarios in order to enumerate them all [8]. Consequently, people have started characterizing the space of all possible genome rearrangement scenarios without explicit enumeration [2, 4]. This space exhibits a nice sub-structure that allows efficient enumeration of substantially different rearrangement scenarios, for example. This may be a good basis for further studies based on statistical approaches or sampling strategies.

Based on the type of genomes and the organism under study, various genome rearrangement operations have been considered. Most results are known for unichromosomal linear genomes, where the only operation is an inversion of

a piece of the chromosome. In this model, the space of all sorting scenarios has been well characterized, allowing to group sorting scenarios into classes of equivalence [2]. The number of classes of equivalence, that can be directly enumerated [4], is much smaller than the total number of scenarios.

In this paper, we study the space of all sorting scenarios under a more general rearrangement operation, called *double-cut and join* (DCJ). This operation was introduced by Yancopoulos *et al.* [9] and further studied in [3]. It acts on multichromosomal linear and/or circular genomes and subsumes all traditionally studied rearrangement operations like inversions, translocations, fusions and fissions. We characterise the sorting sequences and show how to commute operations in order to obtain a sorting sequence from another. In addtion we give a closed formula for the number of DCJ sorting scenarios that is exact for a certain class of instances of the problem, and a lower bound for the general case.

We are aware that recently, and independently of our work, similar results to the ones presented here were obtained by Ouangraoua and Bergeron [6]. However, we believe that our approach is more direct and also more complete, since it is not restricted to *co-tailed* genomes as the approach presented in [6].

2 Genomes, adjacency graph and sorting by DCJ

A multichromosomal genome Π, over a set of markers $\mathcal{G}$, is a collection of linear and/or circular chromosomes in which each marker in $\mathcal{G}$ occurs exactly once in Π. Each marker g in $\mathcal{G}$ is a DNA fragment and has an orientation, therefore, to each $g \in \mathcal{G}$, we define its extremities g^t (tail) and g^h (head) and we can represent an adjacency between two markers a and b in Π by an unordered pair containing the extremities of a and b that are actually adjacent. For example, if the head of marker a is adjacent to the head of marker b in one chromosome of Π, then we have the adjacency $\{a^h, b^h\}$, that we represent as $a^h b^h$ or $b^h a^h$ for simplicity. When the extremity of a marker is adjacent to the telomere of a linear chromosome in Π, we have a singleton instead of a pair. For example, if the extremity b^t is adjacent to a telomere of a linear chromosome, then we have the singleton $\{b^t\}$, that we represent simply as b^t. A genome Π can be represented by the set $V(\Pi)$ containing its adjacencies and singletons [3].

A *double-cut and join* or DCJ operation over a genome Π is the operation that cuts two elements (adjacencies or singletons) of $V(\Pi)$ and joins the separated extremities in a different way, creating two new elements [3]. For example, a DCJ acting on two adjacencies pq and rs would create either the new adjacencies pr and qs, or ps and qr (this could correspond to an inversion, a reciprocal translocation between two linear chromosomes, a fusion of two circular chromosomes, or an excision of a circular chromosome). In the same way, a DCJ acting on one adjacency pq and a singleton r would create either pr and q, or p and qr (in this case, the operation could correspond to an inversion, or a translocation, or a fusion of a circular and a linear chromosomes). At last, a DCJ acting on two singletons p and q would create the new adjacency pq (that could represent a circularization of one or a fusion of two linear chromosomes). Conversely, a

DCJ can act on only one adjacency pq and create the two singletons p and q (representing a linearization of a circular, or a fission of a linear chromosome).

Definition 1 ([3]). *Given two genomes Π_1 and Π_2 over the same set of markers $\mathcal{G}$ (with the same content) and without duplications, the* adjacency graph *$AG(\Pi_1, \Pi_2)$ is the graph in which:*

1. *The set of vertices is $V = V(\Pi_1) \cup V(\Pi_2)$.*
2. *For each $g \in \mathcal{G}$, we have an edge connecting the vertices in $V(\Pi_1)$ and $V(\Pi_2)$ that contain g^h and an edge connecting the vertices in $V(\Pi_1)$ and $V(\Pi_2)$ that contain g^t.*

We know that $AG(\Pi_1, \Pi_2)$ is bipartite with maximum degree equal to two (each extremity of a marker appears in at most one adjacency or singleton in Π_1 and in at most one adjacency or singleton in Π_2). Consequently, $AG(\Pi_1, \Pi_2)$ is a collection of cycles of even length and paths, alternating vertices in $V(\Pi_1)$ and $V(\Pi_2)$. A path is said to be *balanced* when it contains the same number of vertices in $V(\Pi_1)$ and in $V(\Pi_2)$, that is, when it contains an even number of vertices (and an odd number of edges). Otherwise the path contains an odd number of vertices (and an even number of edges) and is said to be *unbalanced*.

Observe that both $V(\Pi_1)$ and $V(\Pi_2)$ have an even number of singleton vertices, thus the number of balanced paths in $AG(\Pi_1, \Pi_2)$ is even [3]. An example of an adjacency graph is given in Figure 1.

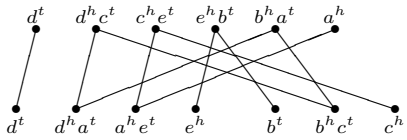

Fig. 1. The adjacency graph for the linear genomes Π_1 and Π_2, defined by the correponding sets of adjacencies $V(\Pi_1) = \{d^t, d^h c^t, c^h e^t, e^h b^t, b^h a^t, a^h\}$ and $V(\Pi_2) = \{d^t, d^h a^t, a^h e^t, e^h, b^t, b^h c^t, c^h\}$, contains one cycle, one unbalanced, and two balanced paths.

Given two genomes Π_1 and Π_2 over the same set of markers $\mathcal{G}$ and without duplications, the problem of sorting Π_1 into Π_2 by DCJ operations has been studied by Bergeron *et al.* [3]. The authors proposed a formula for the DCJ distance, based on the adjacency graph $AG(\Pi_1, \Pi_2)$:

$$d(\Pi_1, \Pi_2) = n - c - \frac{b}{2} \tag{1}$$

where n is the number of markers in $\mathcal{G}$, c and b are respectively the number of cycles and balanced paths in $AG(\Pi_1, \Pi_2)$.

Bergeron *et al.* [3] also observed that an optimal DCJ operation either increases the number of cycles by one or the number of balanced paths by two and proposed a simple algorithm to find one optimal sequence of DCJ operations to sort Π_1 into Π_2.

3 The solution space of sorting by DCJ

Although an optimal DCJ sequence sorting a genome Π_1 into a genome Π_2 can be easily obtained [3], there are several different optimal sorting sequences, and in this study we approach the problem of characterizing and counting these optimal solutions. We want to analyse the space of solutions sorting Π_1 into Π_2, thus we consider only operations acting on genome Π_1, or, in other words, acting on vertices of $V(\Pi_1)$.

3.1 The commutation of two DCJ operations

Here we represent a DCJ operation ρ by the two pairs of adjacencies concerned, $\rho = (adj_0, adj_1)$, that is, the original pair of adjacencies $adj_0 = \{pq, rs\}$ and the new pair of adjacencies, say $adj_1 = \{pr, qs\}$. (In order to generalize this notation to singletons, any extremity among p, q, r, s could be equal to $\circ$ – a telomere.) We say that the adjacencies adj_0 and adj_1 and the extremities p, q, r and s are *affected* by ρ. Two DCJ operations are said to be *independent* when the set of marker extremities (excluding telomeres) affected by one is independent of the set affected by the other.

Proposition 1. *In any optimal DCJ sorting sequence $\ldots\rho\theta\ldots$, two consecutive operations ρ and θ are independent or share one adjacency, that is, one adjacency created by ρ is used (broken) by θ.*

Proof. The second operation cannot use both adjacencies created by the first, otherwise they would not be part of an optimal sequence. Thus, the second operation either uses one adjacency created by the first operation or only adjacencies that were not affected by the first. □

Proposition 2. *In any optimal DCJ sorting sequence $\ldots\rho\theta\ldots$, the operations ρ and θ could be commuted to construct another optimal sorting sequence. If ρ and θ are independent, the commutation is direct, that is, we may simply replace $\ldots\rho\theta\ldots$ by $\ldots\theta\rho\ldots$. Otherwise, the commutation can be done in two ways, with adjustments as follows: Suppose $\rho = (\{pv, qr\}, \{pq, rv\})$ and $\theta = (\{rv, su\}, \{rs, uv\})$. Then we could replace ρ and θ by $\rho' = (\{pv, su\}, \{ps, uv\})$ and $\theta' = (\{ps, qr\}, \{pq, rs\})$ or alternatively by $\rho'' = (\{qr, su\}, \{qu, rs\})$ and $\theta'' = (\{pv, qu\}, \{pq, uv\})$.*

Proof. Observe that either $\rho\theta$, or $\rho'\theta'$, or $\rho''\theta''$ transform the adjacencies pv, qr and su into the adjacencies pq, rs and uv, thus the adjacencies that exist before and after $\rho\theta$, $\rho'\theta'$ and $\rho''\theta''$ are the same. Consequentely the remaining operations are still valid. □

The commutation with adjustments is shown in Figure 2.

Theorem 1. *Any optimal DCJ sequence can be obtained from any other optimal DCJ sequence by successive commutations.*

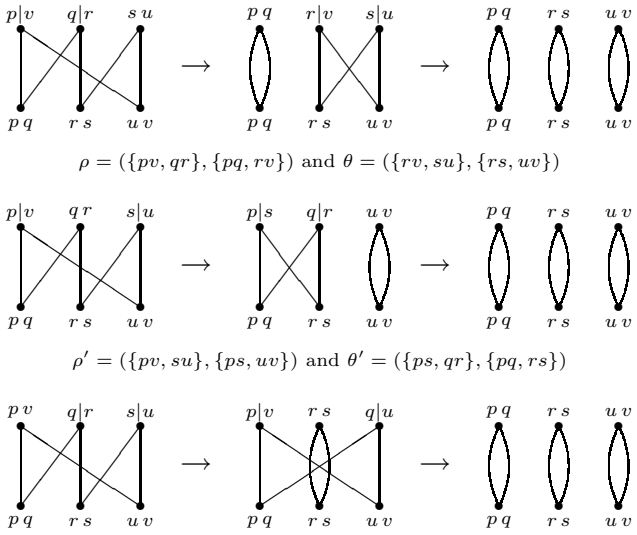

Fig. 2. Example of commutation with adjustments. Observe that the three pairs of consecutive DCJ operations $\rho\theta$, $\rho'\theta'$ and $\rho''\theta''$ transform the adjacencies pv, qr and su into pq, rs and uv.

Proof. Any optimal DCJ sequence transforms the same initial set of adjacencies into the same final set of adjacencies. Suppose the final adjacencies are $adj_1, adj_2, \ldots, adj_k$ and consider two optimal DCJ sorting sequences s and t. We can transform s into t with the following procedure: for each adj_i $(1 \leq i \leq k)$, search the operation in s that produces adj_i and move it to the position it occupies in t by commutations. □

One critical aspect of the commutations is that they can change the actual nature of the operations over the genomes, as we can see in Figure 3.

3.2 Counting DCJ sequences sorting components separately

Proposition 3. *Any DCJ operation acting on vertices of $V(\Pi_1)$ belonging to the same component of $AG(\Pi_1, \Pi_2)$ is optimal.*

Proof. Consider Formula 1 for computing the DCJ distance. We know that an optimal DCJ operation either creates one cycle or two balanced paths [3]. Enumerating the effects of any operation acting on vertices of $V(\Pi_1)$ internal to a component, we have:

- If the vertices are in a cycle, it is split into two cycles, that is, the number of cycles increases by one.
- If the vertices are in a balanced path, it is split into one cycle and one balanced path, that is, the number of cycles increases by one.

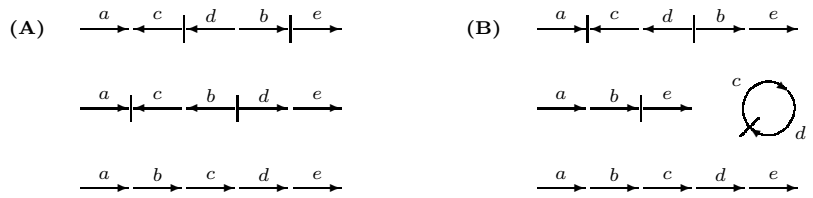

Fig. 3. In this graphic representation of the genomes, each arrow represents a marker (from $\mathcal{G} = \{a, b, c, d, e\}$) with its corresponding orientation. The commutation of two independent DCJ operations sorting $\{a^t, a^h c^h, c^t d^h, d^t b^t, b^h e^t, e^h\}$ into $\{a^t, a^h b^t, b^h c^t, c^h d^t, d^h e^t, e^h\}$ changes the nature of the actual rearrangements. In **(A)** we have $(\{c^t d^h, b^h e^t\}, \{b^h c^t, d^h e^t\})$ followed by $(\{a^h c^h, b^t d^t\}, \{a^h b^t, c^h d^t\})$, a sequence of two inversions, while in **(B)** we have $(\{a^h c^h, b^t d^t\}, \{a^h b^t, c^h d^t\})$ followed by $(\{c^t d^h, b^h e^t\}, \{b^h c^t, d^h e^t\})$, a circular excision followed by a circular integration.

– If the vertices are in an unbalanced path, either it is split into one cycle and one unbalanced path, increasing the number of cycles by one; or it is closed into a cycle (when the path has more vertices in $V(\Pi_1)$), also increasing the number of cycles; or it is split into two balanced paths (when the path has more vertices in $V(\Pi_2)$), increasing the number of balanced paths by two.

Thus, any operation acting on vertices of $V(\Pi_1)$ belonging to the same component of $AG(\Pi_1, \Pi_2)$ is optimal. □

Proposition 4. *Given two genomes Π_1 and Π_2, any component of $AG(\Pi_1, \Pi_2)$ can be sorted independently.*

Proof. This is a direct consequence of Proposition 3. □

Let $d(C)$ be the DCJ distance of a component C in $AG(\Pi_1, \Pi_2)$ and $seq(C)$ be the number of DCJ sequences sorting C. Moreover, let EC_{i+1} be an even cycle with $i+1$ edges in $AG(\Pi_1, \Pi_2)$ and let BP_i be a balanced and UP_{i-1} be an unbalanced path with respectively i and $i-1$ edges (observe that i is odd).

Proposition 5. *For any integer $i \in \{3, 5, 7, \ldots\}$, we have $d(UP_{i-1}) = d(BP_i) = d(EC_{i+1}) = \frac{i-1}{2}$ and $seq(UP_{i-1}) = seq(BP_i) = seq(EC_{i+1})$.*

Proof. A balanced path with i edges could be transformed into a cycle with $i+1$ edges by connecting the two ends of the path (this respects the alternation between vertices of $V(\Pi_1)$ and $V(\Pi_2)$). Analogously, an unbalanced path with $i-1$ edges could be transformed into a cycle with $i+1$ edgess by the insertion of a double "telomere" vertex on the genome that is under-represented in the path. The two ends of the unbalanced path should then be connected to the new vertex (this also respects the alternation between vertices of $V(\Pi_1)$ and $V(\Pi_2)$) and the new vertex can also be used in optimal DCJ operations within the unbalanced path. An example for the case EC_4, BP_3 and UP_2 is given in

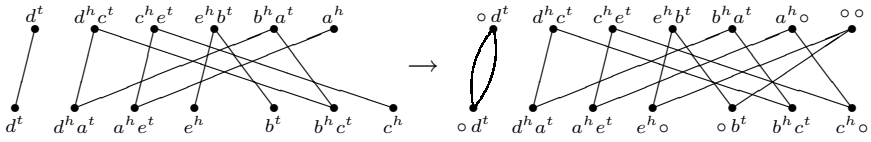

Fig. 4. Adding telomeres to the paths of the adjacency graph. We can see here that the structures of an EC_4, a BP_3 and an UP_2 are identical, therefore these components have the same DCJ distance and the same number of sorting sequences.

Figure 4. It is easy to see that we need $\frac{i-1}{2}$ operations to sort an unbalanced path with $i-1$ edges or a balanced path with i edges or cycle with $i+1$ edges. □

We denote by $C_i \otimes C_j$ the multiplication of two independent sets of sequences sorting two components C_i and C_j, defined as the set of all sequences that are the result of all possible combinations of each sequence sorting C_i with each sequence sorting C_j. Observe that the operation $\otimes$ is symmetric, that is, $C_i \otimes C_j = C_j \otimes C_i$, and associative, that is, $(C_i \otimes C_j) \otimes C_k = C_i \otimes (C_j \otimes C_k)$. We denote by $||C_i||$ the number of sequences sorting C_i, and by d_i the length of each sequence sorting C_i (the DCJ distance of C_i). Then the number of sequences in $C_i \otimes C_j$ corresponds to $||C_i|| \times ||C_j|| \times M(d_i, d_j)$, where $M(d_i, d_j)$ is the number of possible ways to merge a sequence of length d_i with a sequence of length d_j, such that the merged sequences are subsequences of all resulting sequences.

For example, if $s_1 = \rho_1\rho_2$ and $s_2 = \theta_1\theta_2$, then all possible ways of merging s_1 and s_2 are the 6 sequences $\rho_1\rho_2\theta_1\theta_2$, $\rho_1\theta_1\rho_2\theta_2$, $\rho_1\theta_1\theta_2\rho_2$, $\theta_1\rho_1\rho_2\theta_2$, $\theta_1\rho_1\theta_2\rho_2$, and $\theta_1\theta_2\rho_1\rho_2$, thus $M(2,2) = 6$. The number $M(d_1, d_2, \ldots, d_k)$ corresponds to the multinomial coefficient and can be computed by the following formula:

$$M(d_1, d_2, \ldots, d_k) = \binom{d_1 + d_2 + \ldots + d_k}{d_1, d_2, \ldots, d_k} = \frac{(d_1 + d_2 + \ldots + d_k)!}{d_1! d_2! \ldots d_k!}.$$

Proposition 6. *The number of operations sorting a component C whose distance is d is given by $||C|| = (d+1)^{d-1}$.*

Proof. A cycle with n vertices and n edges has $v = n/2$ vertices in $V(\Pi_1)$, DCJ distance $d = v - 1$, and can be broken with one DCJ operation, resulting in two cycles as follows (each pair of cycles can be obtained in v different ways):

- one of size 2 ($v' = 1; d' = 0$) and one of size $n-2$ ($v' = v-1; d' = d-1$);
- one of size 4 ($v' = 2; d' = 1$) and one of size $n-4$ ($v' = v-2; d' = d-2$);
- one of size 6 ($v' = 3; d' = 2$) and one of size $n-6$ ($v' = v-3; d' = d-3$);
- and so on ...

Thus, the computation of the number of sorting sequences is given by the following recurrence formula on v:

$$T(1) = 1$$
$$T(v) = \frac{v}{2} \sum_{i=1}^{v-1} T(v-i) \otimes T(i).$$

We know that $T(v-i) \otimes T(i) = T(v-i) \times T(i) \times \binom{v-2}{i-1}$. Thus, we have
$T(v) = \frac{v}{2} \sum_{i=1}^{v-1} \binom{v-2}{i-1} \times T(v-i) \times T(i)$; or, alternatively
$T(v) = \frac{v}{2} \sum_{k=0}^{v-2} \binom{v-2}{k} \times T(v-k-1) \times T(k+1)$, which is also equivalent to

$$T(v) = \sum_{k=0}^{v-2} \binom{v-2}{k} \times (v-k-1) \times T(v-k-1) \times T(k+1).$$

This last recurrence formula is identical to the recurrence formula presented in [10] for counting labeled trees and results in v^{v-2}. Since we have $d = v-1$, we get $T(v) = (d+1)^{d-1}$. □

We call *small components* the paths and cycles of $AG(\Pi_1, \Pi_2)$ with two vertices (BP_1 and EC_2) whose distance is zero; the other components are *big components* (observe that any unbalanced path is a big component). The DCJ distance and number of sequences sorting big components is shown in Table 1.

unbalanced paths	**balanced paths**	**even cycles**	**sequence length**	**number of sequences**
UP_2	BP_3	EC_4	1	1
UP_4	BP_5	EC_6	2	3
UP_6	BP_7	EC_8	3	16
UP_8	BP_9	EC_{10}	4	125
UP_{10}	BP_{11}	EC_{12}	5	1296
UP_{12}	BP_{13}	EC_{14}	6	16807
⋮	⋮	⋮	⋮	⋮
UP_{2d}	BP_{2d+1}	EC_{2d+2}	d	$(d+1)^{d-1}$

Table 1. The number of DCJ sequences sorting each type of component independently.

Proposition 7. *The number of solutions sorting $AG(\Pi_1, \Pi_2)$ obtained by sorting each component independently is*

$$C_1 \otimes C_2 \otimes \ldots \otimes C_k = \frac{(d_1 + d_2 + \ldots + d_k)!}{d_1! d_2! \ldots d_k!} \times \prod_{i=1}^{k} (d_i + 1)^{d_i - 1}$$

where $C_1, C_2, \ldots, C_k$ are the big *components of $AG(\Pi_1, \Pi_2)$ and $d_1, d_2, \ldots, d_k$ are their respective DCJ distances.*

Proof. We know that the number of solutions obtained by merging the sequences sorting the components independently is given by the multinomial coefficient multiplied by the number of sequences sorting each component, the latter being given by the formula in Proposition 6. □

3.3 Towards the general case

The formula given by Proposition 7 does not correspond to the total number of solutions for a general instance of the problem, due to the recombination of unbalanced paths. A pair of *alternate unbalanced paths* is composed by one unbalanced path with more vertices in $V(\Pi_1)$ and one unbalanced path with more vertices in $V(\Pi_2)$ and can be recombined at any time and in several different ways into two balanced paths. See an example in Figure 5. This can increase considerably the number of optimal solutions, specially when the number of the two types of unbalanced paths in the graph is big.

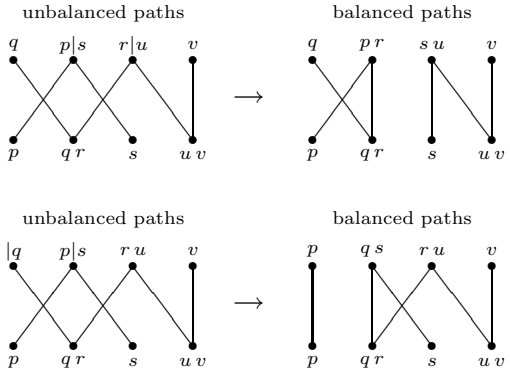

Fig. 5. Here we represent two of many ways of recombining a pair of alternate unbalanced paths into a pair of balanced paths.

Proposition 8. *A DCJ operation acting on vertices of $V(\Pi_1)$ belonging to two different components of $AG(\Pi_1, \Pi_2)$ is optimal, if and only if they are alternate unbalanced paths.*

Proof. Recall that an optimal DCJ operation either increases the number of cycles by one or the number of balanced paths by two [3]. We need to examine all possible DCJ operations acting on two different components of $AG(\Pi_1, \Pi_2)$. If one component is a cycle, and the other component is of type X (X can be either a cycle, or a balanced, or an unbalanced path), then the result is one single component that is also of type X. Thus, the number of cycles in the graph is reduced. If the two components are balanced paths, then the result is either one unbalanced path, or two unbalanced, or two balanced paths. In the first and the second case the number of balanced paths is reduced by two and in the third case it remains unchanged. If one is a balanced and the other is an unbalanced path, then the result is either one balanced path, or one balanced and one unbalanced path. In both cases the number of balanced paths remains unchanged. And the same happens if the two components are unbalanced paths, that do not form an alternate pair. In this case, the result is either one, or two unbalanced paths. Note

that all operations enumerated so far can not be optimal. The last possibility is when the components are a pair of alternate unbalanced paths. In this case, any operation acting on two vertices of $V(\Pi_1)$, each one belonging to each one of the unalanced paths, results in a pair of balanced paths (see Figure 5) and is an optimal DCJ operation. □

Proposition 9. *If $AG(\Pi_1, \Pi_2)$ does not contain any pair of alternate unbalanced paths, the components of $AG(\Pi_1, \Pi_2)$ can only be sorted independently.*

Corollary 1. *The formula given in Proposition 7 is a lower bound to the number of DCJ sorting scenarios for any instance of the problem.*

To give an idea of the increase caused by unbalanced paths, we consider the smallest example, that is, $AG(\Pi_1, \Pi_2)$ has only one pair of alternate unbalanced paths with three vertices each. Since the DCJ distance of each one of these unbalanced paths is one, no DCJ operation can be performed on them before the recombination, and they can be recombined into balanced paths in only four different ways, resulting in a small balanced path with two vertices and a big balanced path with four vertices (see Figure 6). Thus, either these two unbalanced paths are sorted independently or they are recombined into balanced paths in one of these four ways. Let $AG(\Pi_1, \Pi_2)$ have k big balanced components (cycles and balanced paths), numbered from C_1 to C_k. We know that the number of solutions sorting the components of $AG(\Pi_1, \Pi_2)$ independently is $\frac{(d_1+d_2+\ldots+d_k+2)!}{d_1!d_2!\ldots d_k!} \times \prod_{i=1}^{k}(d_i+1)^{d_i-1}$. Analogously, the number of solutions sorting the components of $AG(\Pi_1, \Pi_2)$ that contain one of the four ways of recombining the pair of unbalanced paths is $\frac{(d_1+d_2+\ldots+d_k+2)!}{d_1!d_2!\ldots d_k!2!} \times \prod_{i=1}^{k}(d_i+1)^{d_i-1}$. In consequence, the total number of solutions is:

$$\frac{3 \times (d_1 + d_2 + \ldots + d_k + 2)!}{d_1!d_2!\ldots d_k!} \times \prod_{i=1}^{k}(d_i+1)^{d_i-1}.$$

This means that a single shortest pair of alternate unbalanced paths triplicates the number of solutions with respect to the solutions obtained sorting the components individually.

With this small example, one can figure out the complexity of integrating the recombination of alternate unbalanced paths to the counting formula. The different pairs of alternate unbalanced paths can be recombined separately or simultaneously in many different ways, and not only as the first step sorting the paths to be recombined, but at any time (when these paths have DCJ distance greater than one). However, we conjecture that the formula given in Proposition 7 gives a tight lower bound to the order of magnitude of the number of solutions.

4 Comparing human, chimpanzee and rhesus monkey

From [1] we obtained a database with the synteny blocks of the genomes of human (*Homo sapiens*), chimpanzee (*Pan troglodytes*) and rhesus monkey (*Macaca*

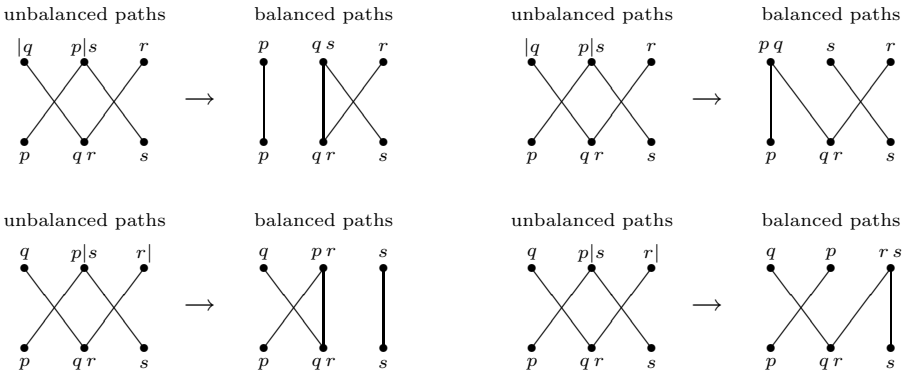

Fig. 6. A pair of shortest alternate unbalanced paths can be recombined into a pair of balanced paths in only four different ways.

mulatta) and used the formula given in Proposition 7 to compute the number of DCJ scenarios for the pairwise comparison of these genomes. The results are shown in Table 2. We observed that the number of paths, and more particularly the number of unbalanced paths in the corresponding adjacency graphs, is usually small. We know that big paths occur when some extremities of linear chromosomes are different in the two analysed genomes, thus our results suggest that this is unlikely to happen when the genomes are closely related. Comparing the human and chimpanzee genomes, for instance, we have only one unbalanced path, so it is not possible to recombine unbalanced into balanced paths. Consequently we are able to give the exact number of solutions for this instance of the problem.

5 Final remarks

In this work we studied the solution space of the sorting by DCJ problem. We were able to characterize the solutions, showing how to transform an optimal sequence into another, and proposed a formula that gives a lower bound to the number of all optimal DCJ sequences sorting one genome into another. This formula can be easily and quickly computed and corresponds to the exact number of sorting sequences for a particular subset of instances of the problem. Although we could identify the structures of the compared genomes that cause the increase of the number of solutions with respect to the given lower bound, finding a general formula to the total number of DCJ sorting scenarios remains an open question.

References

1. Alekseyev, M. A. and Pevzner, P. A.: Breakpoint graphs and ancestral genome reconstructions. *Genome Res.* 19, 943–957, 2009.

genomes	# big cycles	# big bal. paths	# unbal. paths	DCJ distance	# DCJ scenarios
human *vs.* chimpanzee	18	1	$1+0$	22	$\simeq 2.53 \times 10^{21}$
human *vs.* monkey	59	7	$2+4$	106	$\simeq 1.23 \times 10^{177}$
chimpanzee *vs.* monkey	68	8	$1+4$	114	$\simeq 1.53 \times 10^{193}$

Table 2. Counting DCJ sorting sequences between human, chimpanzee and rhesus monkey genomes (data obtained from [1]). For each pairwise comparison, the number of sorting sequences is very large and is thus presented approximately (although it can be computed exactly). Observe that the number of paths is usually much smaller than the number of cycles in all pairwise comparisons. Looking at the human *vs.* chimpanzee comparison in particular, we notice that it results in only one unbalanced path, thus none of its sorting sequences can be obtained by recombining unbalanced into balanced paths. This means that the lower bound given by the formula of Proposition 7 is tight in this case.

2. Bergeron A., Chauve C., Hartmann T. and St-Onge K.: On the properties of sequences of reversals that sort a signed permutation. In: *Proceedings of JOBIM 2002*, 99–108, 2002.
3. Bergeron, A., Mixtacki, J. and Stoye, J.: A unifying view of genome rearrangements. In: *Proceedings of WABI 2006*, LNCS 4175, 163–173, 2006.
4. Braga M. D. V., Sagot M.-F., Scornavacca C. and Tannier E.: Exploring the solution space of sorting by reversals with experiments and an application to evolution. *IEEE/ACM Trans. Comput. Biol. Bioinf.* 5(3), 348–356, 2008. (Preliminary version in *Proceedings of ISBRA 2007*, LNBI 4463, 293–304, 2007).
5. Hannenhalli, S. and Pevzner, P.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: *Proceedings of FOCS 1995*, 581–592, 1995.
6. Ouangraoua, A. and Bergeron A.: Parking functions, labeled trees and DCJ sorting scenarios. In *Proceedings of RECOMB-CG 2009*, LNBI 5817, 2009.
7. Sankoff, D.: Edit Distance for Genome Comparison Based on Non-Local Operations. In *Proceedings of CPM 1992*, LNCS 644, 121–135, 1992.
8. Siepel A.: An algorithm to enumerate sorting reversals for signed permutations. *J. Comput. Biol.* 10, 575–597, 2003.
9. Yancopoulos, S., Attie, O. Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346, 2005.
10. Zeilberger, D.: Yet another proof of Cayley's formula for the number of labelled trees. Available in http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimPDF/cayley.pdf.