Restricted DCJ-indel model: sorting linear genomes with DCJ and indels

Poly H da Silva1,2*, Raphael Machado2, Simone Dantas1, Marília DV Braga2

From Tenth Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics
Niterói, Brazil. 17-19 October 2012

Abstract

Background: The double-cut-and-join (DCJ) is a model that is able to efficiently sort a genome into another, generalizing the typical mutations (inversions, fusions, fissions, translocations) to which genomes are subject, but allowing the existence of circular chromosomes at the intermediate steps. In the general model many circular chromosomes can coexist in some intermediate steps. However, when the compared genomes are linear, it is more plausible to use the so-called restricted DCJ model, in which we proceed the reincorporation of a circular chromosome immediately after its creation. These two consecutive DCJ operations, which create and reincorporate a circular chromosome, mimic a transposition or a block-interchange. When the compared genomes have the same content, it is known that the genomic distance for the restricted DCJ model is the same as the distance for the general model. If the genomes have unequal contents, in addition to DCJ it is necessary to consider indels, which are insertions and deletions of DNA segments. Linear time algorithms were proposed to compute the distance and to find a sorting scenario in a general, unrestricted DCJ-indel model that considers DCJ and indels.

Results: In the present work we consider the restricted DCJ-indel model for sorting linear genomes with unequal contents. We allow DCJ operations and indels with the following constraint: if a circular chromosome is created by a DCJ, it has to be reincorporated in the next step (no other DCJ or indel can be applied between the creation and the reincorporation of a circular chromosome). We then develop a sorting algorithm and give a tight upper bound for the restricted DCJ-indel distance.

Conclusions: We have given a tight upper bound for the restricted DCJ-indel distance. The question whether this bound can be reduced so that both the general and the restricted DCJ-indel distances are equal remains open.

Background

The distance between two genomes is often computed using only the common content, which occurs in both genomes. Such distance takes into consideration only organizational operations, which change the organization of the genome, that is, the positions and orientations of DNA segments, number and types of chromosomes. Inversions, translocations, fusions and fissions are some of these operations [1]. All these rearrangements can be generically represented as double-cut-and-join (DCJ) operations [2]. The DCJ model has simple linear algorithms to compute the distance and to find an optimal sorting sequence [3]. However, while sorting a genome into another by DCJ, circular chromosomes can appear in the intermediate species [3]. In the general model many circular chromosomes can coexist in some intermediate species. Due to this fact, when the compared genomes are linear, it is desirable to consider the so-called restricted model, in which we proceed the reincorporation of a circular chromosome immediately after its creation [2,4]. These two consecutive DCJ operations, which create and reincorporate a circular chromosome, mimic a transposition or a block-interchange. In other words, in the restricted model most of the classical organizational operations (reversals, translocations, fusions...
and fissions) cost one DCJ, while transpositions and block-interchanges cost two DCJs.

When comparing genomes with the same content and without duplicated DNA segments, it is already known that the genomic distance for the restricted DCJ model is the same as the distance for the general model and can be computed in linear time [2,3]. In contrast, while the genomes can be sorted also in linear time in the general model [3], the best sorting algorithm in the restricted model up to now takes $O(n \log n)$ [4]. Figure 1 shows an example of a general and a restricted sorting sequence.

If the genomes have unequal contents, in addition to DCJ operations it is necessary to consider insertions and deletions of DNA segments. Insertions and deletions are jointly called indels. In this context, linear algorithms were proposed to compute the distance and to find a sorting scenario in a general, unrestricted model that handles genomes without duplicated DNA segments, considering DCJ and indel operations [5,6]. During the evolution of many organisms, indel operations are said to occur more often than organizational operations and, consequently, should be assigned to a lower cost. Examples are bacteria that are obligate intracellular parasites, such as Rickettsia [7]. The genomes of such intracellular parasites are observed to have a reductive evolution, that is, the process by which genomes shrink and undergo extreme levels of gene degradation and loss.

The general DCJ-indel model has the flexibility of assigning different positive costs to DCJ and indel operations [5,6]. But, again, many circular chromosomes may coexist in intermediate stages of the sorting process. Thus, while sorting linear genomes, it would be more plausible to consider a restricted DCJ-indel model, in which a circular chromosome must be reincorporated immediately after its creation. Figure 2 shows an example of a general and a restricted sorting sequence with DCJs and indels. In this case, no algorithm was provided up to now and even the question whether the distance is the same for both the general and the restricted DCJ-indel models remains open. Here we address this issue and give a sorting algorithm and a tight upper bound for the restricted DCJ-indel distance, also allowing the assignment of distinct costs to indel and DCJ operations and with the restriction that the indel cost is upper bounded by the DCJ cost.

This paper is organized as follows. In the remainder of this section we recall some key concepts of the DCJ-indel model with distinct operation costs [6], which is based on the DCJ-model [2,3]. We then develop a restricted DCJ-indel sorting algorithm, that gives an upper bound for the restricted DCJ-indel distance. Finally we conclude our work with some final remarks.

The DCJ model

A linear genome is composed of linear chromosomes and can be represented by a set of strings as follows. For each chromosome $C$ of each genome, we build a string obtained by the concatenation of all markers in $C$. Each marker $g$ is a DNA fragment and is represented by the symbol $g$, if it is read in direct orientation, or by the symbol $\bar{g}$, if it is read in reverse orientation. Each one of the two extremities of a linear chromosome is called a telomere, represented by the symbol $\circ$.

Given two linear genomes $A$ and $B$, possibly with unequal content, let $G$, $A$ and $B$ be three disjoint sets,
such that the set $\mathcal{G}$ is the set of markers which occur in $A$ and in $B$, the set $A$ is the set of markers which occur only in $A$ and the set $B$ is the set of markers which occur only in $B$. The markers in $A$ and in $B$ are also called unique markers. As an example, consider the genomes $A = \{\text{obscaud\textsuperscript{bc}}, \text{de}\}$ and $B = \{\text{caud\textsuperscript{bc}}, \text{ydez\textsuperscript{bc}}\}$. Here we have $\mathcal{G} = \{a, b, c, d, e\}$, $A = \{s, u, v\}$ and $B = \{w, x, y, z\}$.

Given two genomes $A$ and $B$, we denote the two extremities of each $g \in \mathcal{G}$ by $g^\ell$ (tail) and $g^w$ (head). A $G$-adjacency or simply adjacency [5] in genome $A$ (respectively in genome $B$) is a string $v = \gamma_1 \ell_1 \gamma_2 \equiv \gamma_2 \ell \gamma_1$, such that each $\gamma_i$ is a telomere or an extremity of a marker from $\mathcal{G}$ and $\ell$ is a substring composed of the markers which are between $\gamma_1$ and $\gamma_2$ in $A$ (respectively in $B$) and contains no marker which also belongs to $\mathcal{G}$. The substring $\ell$ is the label of $v$. If $\ell$ is empty, the adjacency is said to be clean, otherwise it is said to be labeled. If a linear chromosome is composed only of markers which are not in $\mathcal{G}$, it is represented by an adjacency $\circ \ell \circ$.

**DCJ operations**

A cut performed on a genome $A$ separates two adjacent markers of $A$. A cut affects a single adjacency $v$ in $A$: it is done between two symbols of $v$, creating two open ends. A double-cut and join or DCJ applied on a genome $A$ is the operation that performs cuts in two different adjacencies in $A$, creating four open ends, and joins these open ends in a different way. In other words, a DCJ rearranges two adjacencies in $A$, transforming them into two new adjacencies.

Consider a DCJ $\rho$ applied to adjacencies $v_1 = \gamma_1 \ell_1 \ell_4 \gamma_4$ and $v_2 = \gamma_3 \ell_3 \ell_2 \gamma_2$, which creates $x_1 = \gamma_1 \ell_1 \ell_2 \gamma_2$ and $x_2 = \gamma_3 \ell_3 \ell_4 \gamma_4$. We represent such an operation as $\rho = \{(\gamma_1 \ell_1 | \ell_2 \gamma_2, \gamma_3 \ell_3 | \ell_4 \gamma_4) \rightarrow (\gamma_1 \ell_1 | \ell_2 \gamma_2, \gamma_3 \ell_3 | \ell_4 \gamma_4)\}$. The two adjacencies $v_1$ and $v_2$ are called the sources, while the two adjacencies $x_1$ and $x_2$ are called the resultants of $\rho$ [8]. One or more labels among $\ell_1$, $\ell_2$, $\ell_3$ and $\ell_4$ can be equal to $\epsilon$ (the empty string), as well as one or more extremities among $\gamma_1$, $\gamma_2$, $\gamma_3$ and $\gamma_4$ can be equal to $\circ$ (a telomere). A DCJ operation can correspond to several rearrangement events, such as an inversion, a translocation, a fusion or a fission [2].

**Adjacency graph and the DCJ distance**

Given two genomes $A$ and $B$, the adjacency graph $AG(A, B)$ [3] is the bipartite multigraph whose vertices are the adjacencies of $A$ and of $B$ and that has one edge for each common extremity of a pair of vertices. The graph $AG(A, B)$ is composed of connected components that alternate vertices in genome $A$ and in genome $B$. Each component can be either a cycle, or an $AB$-path (which has one endpoint in genome $A$ and the other in $B$), or an $AA$-path (which has both endpoints in genome $A$), or a $BB$-path (which has both endpoints in $B$). A special case of an $AA$ or a $BB$-path is a linear singleton, that is a linear chromosome represented by an adjacency of type $\circ \ell \circ$. In Figure 3 we show the example of an adjacency graph.
Components with 3 or more vertices need to be reduced - by applying DCJ operations - to components with only 2 vertices, that can be cycles or AB-paths [8]. This procedure is called DCJ-sorting of A into B. The number of AB-paths in AG(A, B) is always even and a DCJ operation can be of three types [5]: it can either increase the number of cycles by one, or the number of AB-paths by two (optimal DCJ); or it does not affect the number of cycles and AB-paths (neutral DCJ); or it can either decrease the number of cycles by one, or the number of AB-paths by two (counter-optimal DCJ). We assign the same cost to any DCJ operation. For simplicity, we consider the DCJ cost equal to one. Then, the DCJ distance of A and B, denoted by \( d_{\text{DCJ}}(A, B) \), corresponds to the minimum number of steps required to do a DCJ-sorting of A into B and is given by the following theorem.

**Theorem 1** ([3]) Given two genomes A and B without duplicated markers, we have \( d_{\text{DCJ}}(A, B) = |G| - c - \frac{b}{2} \), where \( G \) is the set of common markers and \( c \) and \( b \) are, respectively, the number of cycles and of AB-paths in AG(A, B).

**The DCJ-indel model with distinct costs**

Although the DCJ-model was defined in the previous sections for genomes with unequal contents, only the common markers were handled. In this section we explain how to deal with unique markers, which are markers that occur only in genome A and markers which occur only in genome B.

**Indel operations**

In order to deal with unique markers, we need operations that change the content of a genome. These operations can be an insertion or a deletion of a block of contiguous markers. Insertions and deletions can be jointly called indel operations. We consider a model in which an indel only affects the label of one single adjacency, by deleting or inserting contiguous markers in this label, with the restriction that an insertion cannot produce duplicated markers [5]. In other words, while sorting A into B, the indel operations are the steps in which the markers in A are deleted and the markers in B are inserted.

Given \( \ell_3 \neq \emptyset \), the deletion of \( \ell_3 \) from the adjacency \( \gamma_1 \ell_1 \ell_3 \epsilon_2 \gamma_2 \) is represented as \( (\gamma_1 \ell_1 | \epsilon_3 | \ell_2 \gamma_2 \rightarrow \gamma_1 \ell_1 | \ell_2 \gamma_2) \), while the insertion of \( \ell_3 \) in the adjacency \( \gamma_1 \ell_1 \ell_2 \gamma_2 \) is represented as \( (\gamma_1 \ell_1 | \ell_2 \gamma_2 \rightarrow \gamma_1 \ell_1 | \ell_3 | \ell_2 \gamma_2) \). One or both extremities among \( \gamma_1 \) and \( \gamma_2 \) can be equal to \( \emptyset \) (a telomere), as well as one or both labels among \( \ell_1 \) and \( \ell_3 \), can be equal to \( \emptyset \) (the empty string). Observe that at most one chromosome can be entirely deleted or inserted at once. Moreover, since duplications are not allowed, an insertion of a marker that already exists is not allowed. Consequently, in this model, it is not possible to apply insertions and/or deletions involving the markers in \( G \).

Given two genomes A and B, the DCJ-indel distance of A and B, denoted by \( d_{\text{DCJ}}(A, B) \), is the minimum cost of a DCJ-indel sequence of operations which sorts A into B, assigning the cost of 1 to each DCJ and a positive cost \( w \leq 1 \) to each indel operation. If \( w = 1 \), the DCJ-indel distance corresponds exactly to the minimum number of steps required to sort A into B [5].

**Runs, indel-potential and the DCJ-indel distance**

Let us recall the concept of run, introduced by Braga et al. [5]. Given two genomes A and B and a component C of AG(A, B), a run is a maximal subpath of C, in which the first and the last vertices are labeled and all labeled vertices belong to the same genome (or partition). A run is then a subpath of a component and can be represented by its list of vertices. A vertex v that corresponds to an entire run is called a compact-run. If a run is not compact, it is a long-run. An example of a component with 3 runs is given in Figure 4. A run in genome A is also called an A -run, and a run in genome B is called a B -run. We denote by \( \Lambda(C) \) the number of runs in a component C. While a path can have 0 or any positive number or runs, a cycle has either 0, 1, or an even number of runs.

A set of labels of one genome can be accumulated with DCJs. In particular, when we apply optimal DCJs on only one component of the adjacency graph, we can...
accumulate an entire run into a single adjacency [5]. It is possible to do a separate DCJ-sorting using only optimal DCJs in any component C of AG(A, B) [8]. We denote by $d_{\text{DCJ}}(C)$ the number of optimal DCJ operations used for DCJ-sorting C separately ($d_{\text{DCJ}}(C)$ depends only on the number of vertices or, equivalently, the number of edges of C [8]). The DCJ distance can also be re-written as $d_{\text{DCJ}}(A, B) = \sum_{C \in AG(A,B)} d_{\text{DCJ}}(C)$.

Runs can be merged by DCJ operations. Consequently, during the optimal DCJ-sorting of a component C, we can reduce its number of runs. The indel-potential of C, denoted by $\lambda(C)$, is defined by Braga et al. [5] as the minimum number of runs that we can obtain doing a separate DCJ-sorting in C with optimal DCJ operations. An example is given in Figure 5.

The indel-potential of a component depends only on its number of runs:

**Proposition 1** ([5]) Given two genomes A and B and a component C of AG(A, B), the indel-potential of C is given by $\lambda(C) = \left\lceil \frac{\Lambda(C)+1}{2} \right\rceil$, if $\Lambda(C) \geq 1$. Otherwise, if $\Lambda(C) = 0$, then $\lambda(C) = 0$.

Let $\lambda_0$ and $\lambda_1$ be, respectively, the sum of the indel-potentials for the components of the adjacency graph before and after a DCJ operation $r$, and let $\Delta\lambda(r) = \lambda_1 - \lambda_0$. If $r$ is an optimal DCJ acting on two adjacencies of a single component of the graph, the definition of indel-potential implies $\Delta\lambda(r) \geq 0$. We also know that $\Delta\lambda(r) \geq 0$, if $r$ is counter-optimal, and $\Delta\lambda(r) \geq -1$, if $r$ is neutral [5]. This allows us to exactly compute $d_{\text{DCJ}}^d(C)$, that is the DCJ-indel distance of a component C of AG(A, B): $d_{\text{DCJ}}^d(C) = d_{\text{DCJ}}(C) + w\lambda(C)$ [6]. We can then derive the following upper bound for the DCJ-indel distance:

**Lemma 1** ([6]) Given two genomes A and B without duplicated markers and a positive indel cost $w \leq 1$, we have $d_{\text{DCJ}}^d(A, B) \leq d_{\text{DCJ}}(A, B) + w \sum_{C \in AG(A,B)} \lambda(C)$.

**Recombinations** Until this point, we have explored the possible effects of any DCJ that is applied to two adjacencies belonging to a single component of the graph. However, there is another type of DCJ that must be considered. A DCJ operation $r$ applied to adjacencies belonging to two different components is called a recombination and can have $\Delta\lambda(r) < 0$ [5]. Thus, depending on the value of $w$ and on whether the recombination is an optimal, a neutral or a counter-optimal DCJ, a recombination with $\Delta\lambda(r) < 0$ can lead to a sorting sequence with lower cost. As an example, a neutral recombination with $\Delta\lambda = -2$ is represented in Figure 6.

Although many different recombinations can occur, it is possible to explore the space of recombinations in linear time and compute the maximum deduction that we can obtain with respect to the upper bound of Lemma 1 [6].

**Results** In this section we develop a restricted DCJ-indel sorting algorithm, from which we can derive an upper bound for the restricted DCJ-indel distance.

**Chained operations** Let us generalize to the DCJ-indel model a concept introduced in [8]. Let $s = r_1r_2 ... r_{n-1}r_n$ be a DCJ-indel
sequence of operations sorting genome $A$ into genome $B$. Two consecutive operations $\rho_i$ and $\rho_{i+1}$ of $s$ are said to be independent when no source of $\rho_{i+1}$ is a resultant of $\rho_i$. Otherwise, $\rho_{i+1}$ use as a source a resultant from $\rho_i$. In this case, the operations $\rho_i$ and $\rho_{i+1}$ are said to be chained.

**Bi-directional approach**

Although in general a sorting algorithm is conceived to follow a single direction, in which all operations are applied on the initial genome, here we design a bi-directional algorithm, in which some operations are applied on genome $A$ and the others are applied on genome $B$. Running a bi-directional algorithm we actually transform genomes $A$ and $B$ into an intermediate genome $I$.

However, with the operations that transform $A$ and $B$ into $I$, we can derive an optimal sequence of operations simply sorting genome $A$ into $B$. Given any DCJ or indel operation $\rho = (X \rightarrow Y)$, the inverse of $\rho$ is $\rho^{-1} = (Y \rightarrow X)$ [5]. This notation can also be extended to a sequence of operations: given a sequence $s = \rho_1\rho_2 \ldots \rho_n$, we have $s^{-1} = \rho_n^{-1}\rho_{n-1}^{-1}\ldots\rho_2^{-1}\rho_1^{-1}$. Observe that the inverse of a deletion is an insertion, and vice-versa.

**Proposition 2** ([5]) Given two genomes $A$ and $B$, and a pair of sequences $s_1$ and $s_2$ composed of DCJ and indel operations applied respectively on genomes $A$ and $B$, transforming both $A$ and $B$ into an intermediate genome $I$, such that $|s_1| + |s_2| = d_{DCJ}(A, B)$, then $s_1s_2^{-1}$ is an optimal sequence of DCJ and indel operations that transforms $A$ into $B$.

Figure 7 illustrates the generation of a sequence of operations sorting $A$ into $B$ from a bi-directional sequence of operations.

**Accumulating x splitting labels**

A DCJ that accumulates labels is always applied to two labeled adjacencies and results into a clean adjacency and an adjacency containing the concatenation of the labels of the original adjacencies. In general, we can represent such an accumulating DCJ $\rho$ by $((\gamma_1\ell_1|\gamma_4, \gamma_3|\ell_2\gamma_5) \rightarrow \gamma_1\ell_6|\gamma_2|\gamma_3|\gamma_4)$. If $\rho$ accumulate labels of an $A$-run, it is denoted by $A^\lambda_\rho > B$. Similarly, if $\rho$ accumulates labels of a $B$-run, it is denoted by $B^\mu_\rho > A$.

The inverse of an accumulating DCJ $\rho$ is a splitting DCJ $\rho^{-1} = (\gamma_1\ell_1|\ell_2\gamma_5|\gamma_3|\gamma_4) \rightarrow (\gamma_1\ell_6|\gamma_2\gamma_3|\gamma_1|\gamma_4)$. Observe that, if $\rho$ is applied on $A$, $\rho^{-1}$ is applied on $B$ and split a label of an $A$-run. In other words, the inverse of an $A^\lambda_\rho > B$ is a

**Figure 5** Two optimal sequences for DCJ-sorting an $AB$-path with $\lambda = 3$ (the cuts of each DCJ in each sequence are represented by $|\rangle$). In (i) the overall number of runs in the resulting components is three, while in (ii) the resulting components have only two runs. Indeed, in this case, the best we can have is the indel-potential $\lambda = 2$.

**Figure 6** This recombination is a neutral DCJ that has $\Delta \lambda = -2$ (we represent only the labels of the adjacencies, the cuts of the recombination are represented by $|\rangle$ and $\langle|$).
DCJ applied on $B$ that separates vertices belonging to the same $\mathcal{A}$-run in two different cycles, denoted by $B \prec A \mathcal{A}$.

Similarly, the inverse of a $B \succ A \mathcal{B}$ is a DCJ applied on $A$ that separates vertices belonging to the same $\mathcal{B}$-run, denoted by $A \prec B \mathcal{B}$. An $A \prec B \mathcal{B}$ or a $B \prec A \mathcal{A}$ is called an inverted-split. In Table 1 we summarize the operations described above.

**Accumulation-deletion x insertion-split**

Let $n$ be a positive integer, such that $n \geq 2$ and let $r = v_1x_1v_2x_2 \ldots v_jx_j \ldots v_{n-1}x_{n-1}$ be a long-run, in which $v_1$ and $v_n$ are labeled, each $x_k$ ($2 \leq k \leq n - 1$) can also be labeled and all $x_k$ ($1 \leq k \leq n - 1$) are clean. We say that two vertices $v_i$ and $v_j$ ($1 \leq i < j \leq n$) in $r_1$ are *partners* if $v_i$ and $v_j$ are labeled and all vertices between $v_i$ and $v_j$ in $r_1$ are clean. We can apply an accumulating DCJ on the two partners $v_i$ and $v_j$ accumulating their labels into a new vertex $v_{i,j}$ reducing $r_1$ to $r_2 = v_1x_1v_2x_2 \ldots v_{i-1}x_{i-1}v_{i,j}v_{j+1}x_{j+1} \ldots v_{n-1}x_{n-1}v_n$. The subsequent step of accumulation then occurs between two partners of $r_2$, reducing $r_2$ to $r_3$, and so on. Assuming that the initial $r_1$ has $m \leq n$ labeled vertices, we need to apply $m - 1$ accumulating operations. In the end of the process, we obtain the compact-run $r_m$, that corresponds to a single vertex whose label is the accumulation of all labels of $r_1$. Observe that all labeled vertices will be used in some accumulating DCJ, until the compact-run $r_m$ is obtained.

As an example, take $v_1 = \gamma_1\ell_1\gamma_2$, $x_1 = \gamma_2\gamma_3$, $v_2 = \gamma_3\ell_2\gamma_4$, $x_2 = \gamma_4\gamma_5$, $v_3 = \gamma_5\ell_3\gamma_6$, $x_3 = \gamma_6\gamma_7$, $v_4 = \gamma_7\ell_4\gamma_8$, with all $\ell_k \neq \epsilon$ and let $r_1 = v_1x_1v_2x_2v_3x_3v_4$ be a $\mathcal{B}$-run. We can start the accumulation with a DCJ of type $B^{B^B}_B > A$ on partners $v_2$ and $v_3$, creating $v_{2,3} = \gamma_3\ell_4\ell_3\gamma_4$ and $\gamma_6\gamma_5$, reducing $r_1$ to $r_2 = v_1x_1v_2x_2v_3x_3v_4$. We then apply another DCJ of type $B^{B^B}_B > A$ on partners $v_1$ and $v_2$, creating $v_{1,2} = \gamma_1\ell_1\ell_4\ell_3\gamma_4$ and $\gamma_6\gamma_5$, reducing $r_2$ to $r_3 = v_1x_1v_2x_2v_3x_3v_4$. Finally, we apply a DCJ of type $B^{B^B}_B > A$ on partners $v_1$ and $v_3$, creating $v_{1,3} = \gamma_1\ell_1\ell_3\ell_4\gamma_4$, and $\gamma_6\gamma_5$, reducing $r_3$ to $r_4 = v_1v_2v_3v_4$. If we follow the accumulation of a run, considering only the labeled vertices, we obtain a rooted tree that is built from the leafs to the root (see Figure 8). The root of the tree indicates the possible positions of a deletion.

The inversion of the run accumulation described in the example above is the inverted-split of the label of the compact-run $r_4 = v_1v_2v_3v_4$ into the labeled vertices $v_1 = \gamma_1\ell_1\gamma_2$, $v_2 = \gamma_2\gamma_3$, $v_3 = \gamma_3\ell_2\gamma_4$ and $v_4 = \gamma_4\ell_4\gamma_8$. We start by applying a $A \prec B \mathcal{A}$ DCJ on $v_1v_2v_3v_4 = \gamma_1\ell_1\ell_3\ell_4\gamma_4$ and $\gamma_6\gamma_5$, obtaining $v_{1,2,3} = \gamma_1\ell_1\ell_3\ell_4\gamma_4$ and $v_4 = \gamma_4\ell_4\gamma_8$. We then apply a $B \prec A \mathcal{B}$ on $v_1v_2v_3v_4$, obtaining $v_2 = \gamma_1\ell_1\gamma_2$ and $v_3 = \gamma_3\ell_2\gamma_4$. Finally we apply a $B \prec A \mathcal{B}$ on $v_1v_2v_3v_4$, considering only the labeled vertices, we obtain a rooted tree that is built from the root to the leafs (see Figure 8 again). In this case, the first inverted-split defines the root. Then, each one of the subsequent inverted-splits must be chained with a DCJ in

**Table 1 Accumulating and splitting DCJ operations**

| Operation | Direction | Effect | Inverse |
|-----------|-----------|--------|---------|
| $A \prec B \mathcal{A}$ | $A \rightarrow B$ | Accumulate labels of an $A$-run | $(A \prec B \mathcal{A})^{-1} = B \prec A \mathcal{A}$ |
| $B \prec A \mathcal{B}$ | $B \rightarrow A$ | Inversely split label of a $B$-run | $(B \prec A \mathcal{B})^{-1} = A \prec B \mathcal{B}$ |
| $B \prec A \mathcal{B}$ | $B \rightarrow A$ | Accumulate labels of a $B$-run | $(B \prec A \mathcal{B})^{-1} = A \prec B \mathcal{B}$ |
| $B \prec A \mathcal{B}$ | $B \rightarrow A$ | Inversely split label of an $A$-run | $(B \prec A \mathcal{B})^{-1} = A \prec B \mathcal{B}$ |
Figure 8 The rooted tree of an accumulation of a $B^r$-run is built from the leaves to the root (bottom to up). Inversely, the rooted tree of an inverted-split of a $B^r$-run is built from the root to the leaves (top to down).

this tree. The root of the tree indicates the possible positions of an insertion.

An indel does not have to occur while a circular chromosome exists

We now show that an indel must not be applied while a circular chromosome exists.

Proposition 3 shows that an insertion can always be “moved up” in a DCJ-indel sorting sequence.

**Proposition 3** Let $s = \rho_1\rho_2 \ldots \rho_{n-1}\rho_n$ be a DCJ-indel sequence sorting genome $A$ into genome $B$, such that, for an integer $1 \leq i < n$, $\rho_i$ is a DCJ operation and $\rho_{i+1}$ is an insertion. Then $\rho_i \rho_{i+1}$ can be replaced by $\theta_1 \theta_2$, such that $\theta_1$ is an insertion and $\theta_2$ is a DCJ and $s' = \rho_1\rho_2 \ldots \rho_i \theta_1 \theta_2 \rho_{i+2} \ldots \rho_{n-1}\rho_n$ is also a DCJ-indel sequence sorting genome $A$ into genome $B$.

**Proof:** Observe that, if $\rho_i$ and $\rho_{i+1}$ are independent, it is easy to see that they can be simply switched, that is: $\theta_1 = \rho_{i+1}$ and $\theta_2 = \rho_i$. We still need to examine the case in which $\rho_i$ and $\rho_{i+1}$ are chained.

Observe that a DCJ in any optimal sorting scenario either accumulates or does not change the composition of runs. Take $\rho_i = (\gamma_1 | \gamma_2 | \gamma_3 | \gamma_4)$ followed by $\theta_2 = (\gamma_1 | \gamma_4 | \gamma_3 | \gamma_2) \rightarrow (\gamma_1 | \gamma_2 | \gamma_3 | \gamma_4)$). Furthermore, since an insertion in any optimal sequence is performed without breaking any existing label, without loss of generality, take $\rho_{i+1} = (\gamma_1 | \gamma_2 | \gamma_3 | \gamma_4) \rightarrow (\gamma_1 | \gamma_2 | \gamma_3 | \gamma_4)$. Then $\rho_i \rho_{i+1}$ could be replaced by: $\theta_1 = (\gamma_2 | \gamma_3 \rightarrow \gamma_2 | \gamma_3 | \gamma_4 \rightarrow \gamma_2 | \gamma_3 | \gamma_4 | \gamma_5)$.

Similarly, a deletion can always be “moved down” in a DCJ-indel sorting sequence.

**Proposition 4** Let $s = \rho_1\rho_2 \ldots \rho_{n-1}\rho_n$ be a DCJ-indel sequence sorting genome $A$ into genome $B$, such that, for an integer $1 \leq i < n$, $\rho_i$ is a deletion and $\rho_{i+1}$ is a DCJ operation. Then $\rho_i \rho_{i+1}$ can be replaced by $\theta_1 \theta_2$, such that $\theta_1$ is a DCJ operation and $\theta_2$ is a deletion and $s' = \rho_1\rho_2 \ldots \rho_{i-1} \rho_{i+1} \theta_1 \theta_2 \rho_{i+2} \ldots \rho_{n-1}\rho_n$ is also a DCJ-indel sequence sorting genome $A$ into genome $B$.

**Proof:** Analogous to the proof of Proposition 3. □

From the previous propositions we observe that finding a position to perform an indel imposes no difficulties to design a restricted DCJ-indel sorting algorithm. The trick is how to determine the DCJ part of the sorting sequence, so that we reincorporate each circular chromosome after its creation and achieve the indel-potential per component.

**Restricted DCJ-indel sorting**

Basically, our approach disregards recombinations and sorts the components of the graph separately, using optimal DCJ operations to achieve the minimum number of indels per component, that is given by the indel-potential. In this way, we achieve the distance given by the upper bound of Lemma 1, as we will see in the remainder of this section.
Capping
Disregarding recombinations, we can first perform the genome capping, a technique that helps us to avoid difficulties and special cases produced by telomeres: we adjoin new markers (caps) to the ends of the chromosomes (and new chromosomes composed of caps only, if necessary) so that we do not change the distance and we do not have to worry about telomeres [4]. After the capping, the two genomes have the same number of chromosomes and the corresponding adjacency graph contains only clean paths of size 1 and cycles. Recall that, since $AG(A, B)$ is bipartite, all cycles have even length and can have 0, 1, or an even number of runs. Capped genomes can be then sorted with translocations (which mimic also fusions and fissions), inversions, circular chromosome excisions and reincorporations.

Merging runs in cycles
An important step of the DCJ-indel sorting is to merge runs in cycles with at least 4 runs, so that the indel-potential for each cycle is achieved.

Proposition 5 The indel-potential of a cycle $C$ with at least 4 vertices and 2 or more runs can be achieved by extracting from $C$ a cycle with a single run.

Proof: For any positive integer $i$ let $\lambda(i) = \lceil \frac{i+1}{2} \rceil$. If $\Lambda(C) = 2$, we can split $C$ into two cycles containing a single run each, and the indel-potential is preserved. For any cycle $C$ with 4 or more runs, since the number of runs in this case is always even, we have $\lambda(i) = \frac{i}{2} + 1$. We then denote by $\lambda'$ the alternative potential, obtained by extracting cycles with a single run from $C$. Observe that, for any $i = 4, 6, 8, ...$, $\lambda'(i) = \lambda(i - 2) + 1$. It is easy to check the base case, that is $\lambda'(4) = \lambda(2) + 1 = 2 + 1 = 3 = \frac{4}{2} + 1 = \lambda(4)$. By induction, for $i = 6, 8, 10, ...$, we have $\lambda'(i) = \lambda(i - 2) + 1 = \frac{i-2}{2} + 1 + 1 = \frac{i}{2} + 1 = \lambda(i)$. □

Chromosome reincorporation
In the restricted sorting of linear genomes a circular chromosome has to be immediately reincorporated after its excision - these two consecutive operations mimic either a transposition or a block-interchange [2,4]. As we have seen before, the general DCJ-indel sorting is bi-directional - the operations can be applied on genome A or B, depending on whether we accumulate runs in A or in B. However, when a DCJ creates a circular chromosome, we need to apply the subsequent DCJ on the same genome, and it is not easy to see how this interferes with the indel-potential of $AG(A, B)$.

Suppose that a DCJ performed an excision of a circular chromosome. Let $(v_1, v_2)$ be a pair of vertices, such that $v_1$ and $v_2$ are in the same genome and belong to the same cycle in $AG(A, B)$, $v_1$ is an adjacency at the circular chromosome and $v_2$ is an adjacency at a linear chromosome. The pair $(v_1, v_2)$ is called a link. Since $v_1$ and $v_2$ are in the same cycle, a chromosome reincorporation can always be done by applying a DCJ on the two vertices $v_1$ and $v_2$ [8].

The cycle to which a link $(v_1, v_2)$ belongs is called a connection cycle. Let $C$ be a connection cycle of $AG(A, B)$ with $2k \geq 4$ vertices. Since $C$ has $k$ vertices in each genome, there are at least $k - 1$ and at most $\left\lceil \frac{k}{2} \right\rceil \cdot \left\lfloor \frac{k}{2} \right\rfloor$ distinct links in $C$.

The two vertices $v_1$ and $v_2$ of a link in a connection cycle $C$ are connected by two distinct subpaths of $C$. The distance between $v_1$ and $v_2$ is given by the number of edges in the shortest path connecting them. Since both $v_1$ and $v_2$ are in the same genome, this distance is always even and positive. If the distance between $v_1$ and $v_2$ is 2, $v_1$ and $v_2$ have a common neighbor, and $(v_1, v_2)$ is called a short-link.

Proposition 6 After the excision of a circular chromosome by a DCJ, there is at least one short-link in $AG(A, B)$.

Proof: Suppose that the circular chromosome is in genome A. If $AG(A, B)$ contained no connection cycle, genome B would also have a circular chromosome, which would be a contradiction. Let $C = v_1 x_1 v_2 x_2 ... v_n x_n$ be a connection cycle in $AG(A, B)$, in which the vertices $v_1$, ..., $v_n$ are in A and the vertices $x_1$, ..., $x_n$ are in B, and let $(v_i, v_j)$ be a link in $C$ such that $v_i$ is in the circular chromosome and $v_j$ is in a linear chromosome of A. Consider without loss of generality that $i < j$. Then take the vertex $v_k$, $i < k < j$, such that $k$ is the largest index of a vertex between $v_i$ and $v_j$ belonging to the circular chromosome. Then $(v_k, v_{k+1})$ is a short-link. □

In order to find out whether the indel-potential of the connection cycle $C$ can be preserved after applying a DCJ on a certain link $(v_1, v_2)$, basically we need to analyze how the connection cycle $C$ is split, by analyzing the vertices that are between $v_1$ and $v_2$ in $C$.

We focus on the short-links only. Let $(v_1, v_2)$ be a short-link in a connection cycle $C$, such that $v_1 = \gamma_1 \ell_1 \gamma_2$ and $v_2 = \gamma_2 \ell_2 \gamma_3$. Without loss of generality, let $z = \gamma_2 \ell_2 \gamma_3$ be the common neighbor of $v_1$ and $v_2$ ($\ell_3$ can also be equal to $z$). We then define the optimal DCJ $\rho(v_1, v_2) = ([v_1, v_2] \rightarrow [x_1, x_2])$, such that $x_1 = \gamma_2 \gamma_3$ and $x_2 = \gamma_1 \ell_1 \ell_2 \gamma_4$. Observe that $\rho(v_1, v_2)$ always extracts $z$ together with a new clean vertex $x_1$ into a cycle, and accumulates the labels of $v_1$ and $v_2$ into a new vertex $x_2$, which is extracted into a cycle with the remaining vertices of $C$. There are three different cases:

1. Gaps: If the two vertices of a short-link have a clean common neighbor, it is called a gap. A DCJ applied to a gap of a connection cycle $C$ splits $C$ into a clean cycle $C'$ and a cycle $C''$ that has the same indel-potential of $C$.

2. Compact-runs: Let $(v_1, v_2)$ be a short-link in $AG(A, B)$, such that the common neighbor $z$ of $v_1$ and
$v_2$ is a compact-run. An optimal DCJ $\rho(v_1, v_2)$ extracts the compact-run $z$ and a new clean vertex into a new cycle. According to Proposition 5, $\rho(v_1, v_2)$ preserves the indel-potential of $AG(A, B)$.

3. Inverted-splits: If a short-link $(v_1, v_2)$ is not a gap nor is separated by a compact-run, only one possibility remains: the common neighbor $z$ of $v_1$ and $v_2$ is labeled and belongs to a long-run $r$. Observe that an optimal DCJ $\rho(v_1, v_2)$ splits $C$ into a cycle $C'$ containing a new clean vertex and $z$ ($\Lambda(C) = 1$) and a cycle $C''$ containing all remaining runs of $C$ and the remaining vertices of $r$, that is, we have $\Lambda(C'') = \Lambda(C)$.

Since each vertex of $C$ in $A$ is part of a link, we can choose a link that contains $v$ and, consequently, is chained with $\rho$. □

The sorting algorithm and an upper bound for the restricted DCJ-indel distance

We put everything together in Algorithm 1 (Additional file 1) and describe the sorting of capped genomes for the restricted model, in which each circular chromosome is reincorporated immediately after its creation. Applying this procedure we can find a sequence of optimal DCJs that sort $A$ into $B$ while preserving the indel-potential. In other words, this algorithm results in a sorting sequence in the restricted model that has exactly the same cost given by the upper bound of Lemma 1.

Conclusions

In this work we have presented a method to compute a restricted DCJ-indel sequence of operations that sort a linear genome into another linear genome. This method leads to a tight upper bound for the restricted DCJ-indel distance. The general DCJ-indel distance can be computed exactly and is a lower bound for the restricted DCJ-indel distance. However, the question whether these bounds are equal, meaning that both distances are equal, remains open.

Algorithm 1 Restricted sorting of genome $A$ into $B$ with optimal DCJs and indels

Input: Two linear genomes $A$ and $B$
Output: A restricted sequence of DCJ and indel operations sorting $A$ into $B$

1. cap genomes $A$ and $B$;
2. [MERGING:] $r \leftarrow \text{null}$;
3. if there is a cycle $C \in AG(A, B)$ with at least 4 vertices and at least 2 runs then $r \leftarrow \text{run from } C$;
4. while $r \neq \text{null}$ do
5. extract $r$ into a cycle; [this preserves the indel-potential of $AG(A, B)$ according to Proposition 5] $r \leftarrow \text{null}$;
6. if a circular chromosome was created then find a short-link $(v_1, v_2)$; [Proposition 6] if $(v_1, v_2)$ is a gap or a compact-run then apply the optimal DCJ $\rho(v_1, v_2)$; else let $r_1$ be the run that would be inversely split by $\rho(v_1, v_2)$; if $\rho(v_1, v_2)$ is the first inverted-split of $r_1$ then apply the optimal DCJ $\rho(v_1, v_2)$; let $r_2$ be the residual part of $r_1$; if $r_2$ is in a cycle with more runs then $r \leftarrow r_2$; [extract $r_2$ from its cycle in the next step] else
\[ r_1 \text{ was inversely split before and is separated alone in cycle} \]

find a link \((x_1, x_2)\) such that \(x_1\) is a vertex created by a previous inverted-split of \(r_1\); [Proposition 7]

apply the optimal DCJ \(p(x_1, x_2)\);

if \(r = \text{null}\) and there is a cycle \(C \in AG(A,B)\) with at least 4 vertices and at least 2 runs then

\( r \leftarrow \text{run from } C; \)

[ACCUMULATING: (each cycle with 4 or more vertices has at most one run)]

while there is a long-run \(r\) in \(AG(A,B)\) do

apply an optimal DCJ accumulating the labels of two partners of \(r\)

if a circular chromosome was created then

find a short-link \((v_1, v_2)\); [Proposition 6]

\( r_1 \) be the run that would be inversely split by \(p(v_1, v_2)\);

if \(p(v_1, v_2)\) is the first inverted-split of \(r_1\) then

apply the optimal DCJ \(p(v_1, v_2)\);

else

let \(r_1\) be the run that would be inversely split by \(p(v_1, v_2)\);

apply the optimal DCJ \(p(v_1, v_2)\);

[\(r_1\) was inversely split before and is separated alone in cycle]

find a link \((x_1, x_2)\) such that \(x_1\) is a vertex created by a previous inverted-split of \(r_1\); [Proposition 7]

apply the optimal DCJ \(p(x_1, x_2)\);

[DCJ-SORTING: (each remaining cycle with 4 or more vertices has at most one compact-run)]

while there is cycle \(C \in AG(A,B)\) with at least 4 vertices do

extract a cycle from \(C\), with an optimal DCJ applied on genome \(A\);

if a circular chromosome was created then

find a short-link \((v_1, v_2)\); [Proposition 6]

at this stage this short-link is a gap or a compact-run

apply the optimal DCJ \(p(v_1, v_2)\);

invert all DCJs applied on genome \(B\);

insert each \(B\)-run \(r\) before the first inverted-split of \(r\);

move up insertions that occur in circular chromosomes;

delete all \(A\)-runs from the DCJ-sorted components;

\( [r_1 \text{ was inversely split before and is separated alone in cycle}] \)

Additional material

Additional file 1: