Genomic Distance with DCJ and Indels

Marília D. V. Braga, Eyla Willing, and Jens Stoye

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

Abstract. The double cut and join (DCJ) operation, introduced by Yancopoulos, Attie and Friedberg in 2005, allows one to represent most rearrangement events in genomes. However, a DCJ cannot perform an insertion or a deletion and most approaches under this model consider only genomes with the same content and without duplications, including the linear time algorithms to compute the DCJ distance and to find an optimal DCJ sorting sequence. In this work, we compare two genomes with unequal content, but still without duplications, and present a new linear time algorithm to compute the genomic distance, considering DCJ and indel operations. With this method we find preliminary evidence of the occurrence of clusters of deletions in the *Rickettsia* bacterium.

1 Introduction

The double cut and join (DCJ) is an abstract rearrangement operation, introduced by Yancopoulos *et al.* in 2005 [5], that allows to represent most large scale mutation events, such as inversions, translocations, fusions and fissions, that can occur in genomes. 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. However, a DCJ cannot perform an insertion or a deletion and most studies concerning DCJ consider genomes with the same content and without duplications. With these restrictions, linear time algorithms have been proposed to compute the DCJ distance and to find an optimal DCJ sorting sequence [1].

In 2008, Yancopoulos and Friedberg [4] proposed an extension of the DCJ paradigm, to include operations performing insertions and deletions, in order to deal with genomes having unequal content. The authors introduced some concepts, but left open the design of an algorithm to handle this problem, which is the subject of this study. We propose an approach in which the cost of an insertion or deletion is the same as that of a DCJ, where several consecutive markers can be inserted or deleted in a single event. Therefore, we generalize the adjacency graph, introduced by Bergeron *et al.* [1], by incorporating the representation of the markers that occur in only one of the two genomes. We then design a linear time algorithm to compute the distance between two genomes with unequal content, but still without duplications, taking into consideration DCJ operations, insertions and deletions. We used this method to do an interesting analysis of a group of bacterial genomes, as described in the last section.

2 DCJ, Adjacency Graph and Indels

In this work, duplications are not allowed. Thus, given a genome A over a set of markers $\mathcal{G}_A$, each g in $\mathcal{G}_A$ occurs exactly once in A. Furthermore, each marker g is a DNA fragment and can be either read in direct orientation and represented by the symbol g, or read in reverse orientation and represented by the symbol $\overline{g}$. We have $\overline{\overline{g}} = g$ and, for any set $\mathcal{F}$, we define $\widehat{\mathcal{F}} = \mathcal{F} \cup \overline{\mathcal{F}}$, where $\overline{\mathcal{F}} = \{\overline{f} \mid f \in \mathcal{F}\}$.

Let A be a genome, possibly composed of linear and circular chromosomes. From each chromosome $\mathcal{C}$ of A we can build a string s over $\widehat{\mathcal{G}_A}$, obtained by the concatenation of all symbols in $\mathcal{C}$, read in any of the two directions. Each end of a linear chromosome is called a *telomere*, represented by the symbol $\circ$. Thus, if $\mathcal{C}$ is linear, it is represented by $\circ s \circ$. If $\mathcal{C}$ is circular, it is simply represented by s (we can start to build s in any symbol of $\mathcal{C}$). A genome A with k chromosomes can be represented by a set of k strings and an example is given in Fig. 1.

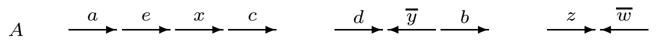

Fig. 1. In this graphic representation of genome $A = \{\circ aexc\circ, \circ d\overline{y}b\circ, \circ z\overline{w}\circ\}$, composed of three linear chromosomes, each arrow represents a marker and its orientation.

In the following we generalize definitions introduced by Bergeron *et al.* [1].

Given a genome A over $\mathcal{G}_A$ and a subset $\mathcal{G} \subseteq \mathcal{G}_A$, for each $g \in \mathcal{G}$ we denote its two extremities by g^t (tail) and g^h (head). A *$\mathcal{G}$-adjacency* is in general a linear string $v = \gamma_1 \ell \gamma_2$, such that γ_1 and γ_2 are telomeres or extremities of markers in $\mathcal{G}$ and ℓ, the substring composed of the markers that are between γ_1 and γ_2 in A, contains no marker that also belongs to $\mathcal{G}$. The substring ℓ is said to be the *label* of v, and the extremities γ_1 and γ_2 are said to be *$\mathcal{G}$-adjacent.* If ℓ is a non-empty string, v is said to be *labeled*, otherwise v is said to be *clean.* Observe that a $\mathcal{G}$-adjacency $\gamma_1 \ell \gamma_2$ can also be represented by $\gamma_2 \overline{\ell} \gamma_1$. Moreover, a labeled $\mathcal{G}$-adjacency $u = \circ \ell \circ$ indicates that A contains a linear chromosome composed only of markers that are not in $\mathcal{G}$, that is, u corresponds to a whole linear chromosome. In the same way, if s is a circular chromosome in A composed only of markers that are not in $\mathcal{G}$, then s is also a $\mathcal{G}$-adjacency. This is the only special case of $\mathcal{G}$-adjacency in which we have a circular instead of a linear string.

A genome A can be then represented by the set $V_{\mathcal{G}}(A)$ containing its $\mathcal{G}$-adjacencies. For example, if $\mathcal{G} = \{a, b, c, d, e\}$, the genome in Fig. 1 has the representation $V_{\mathcal{G}}(A) = \{\circ a^t, a^h e^t, e^h x c^t, c^h \circ, \circ d^t, d^h \overline{y} b^t, b^h \circ, \circ z \overline{w} \circ\}$. However, for $\mathcal{G}' = \mathcal{G}_A = \{a, b, c, d, e, x, y, z, w\}$, we have only clean adjacencies in $V_{\mathcal{G}'}(A) = \{\circ a^t, a^h e^t, e^h x^t, x^h c^t, c^h \circ, \circ d^t, d^h y^h, y^t b^t, b^h \circ, \circ z^t, z^h w^h, w^t \circ\}$.

A *cut* performed on a genome A separates two adjacent markers of A. A cut affects a $\mathcal{G}$-adjacency v of $V_{\mathcal{G}}(A)$ as follows: if v is linear, the cut is done between two symbols of v, creating two open ends in two separate linear strings; if v is circular, the cut creates two open ends in one linear string. A *double-cut and join* or DCJ applied on a genome A is the operation that performs two cuts

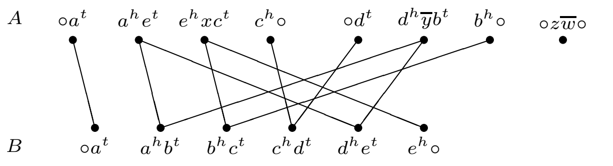

Fig. 2. For genomes $A = \{\circ aexc\circ, \circ d\overline{y}b\circ, \circ z\overline{w}\circ\}$ and $B = \{\circ abcde\circ\}$, the adjacency graph contains one cycle, two AA-paths (one is a linear singleton) and two AB-paths.

in $V_{\mathcal{G}}(A)$, creating four open ends, and joins these open ends in a different way. As an example, considering the genome A from Fig. 1 and $\mathcal{G} = \{a, b, c, d, e\}$, if we apply a DCJ on $a^h e^t$ and $d^h \overline{y} b^t$ of $V_{\mathcal{G}}(A)$ we can create $a^h b^t$ and $d^h \overline{y} e^t$.

Observe that, if the two original $\mathcal{G}$-adjacencies are $\gamma_1 \ell_1 \circ$ and $\gamma_2 \ell_2 \circ$, we can create $\gamma_1 \ell_1 \overline{\ell_2} \gamma_2$ and $\circ\circ$. Conversely, a DCJ operation can be applied to $\circ\circ$ and $\gamma_1 \ell \gamma_2$, creating, for example, $\gamma_1 \ell \circ$ and $\gamma_2 \circ$. These are special cases of the DCJ operation, and the clean $\mathcal{G}$-adjacency $\circ\circ$ is also called *null linear chromosome* [5]. There are also two special cases of a DCJ that change only labels and circular $\mathcal{G}$-adjacencies: when one of the two cuts is on a circular $\mathcal{G}$-adjacency s, the result will be a single $\mathcal{G}$-adjacency v and s will be integrated into the label of v. Conversely, when both cuts of a DCJ are applied to the same $\mathcal{G}$-adjacency, we would have either an inversion in its label or an excision of a circular $\mathcal{G}$-adjacency. With respect to the structure of the involved chromosomes, a DCJ operation can correspond to several events, such as an inversion, a translocation, a fusion, or a fission. In addition a DCJ can also correspond to an excision or an integration of a circular chromosome [1].

Given a genome A over $\mathcal{G}_A$ and a genome B over $\mathcal{G}_B$ with $\mathcal{G} = \mathcal{G}_A \cap \mathcal{G}_B$, the *adjacency graph* $AG(A, B)$ is the graph that has a vertex for each $\mathcal{G}$-adjacency in $V_{\mathcal{G}}(A)$ and a vertex for each $\mathcal{G}$-adjacency in $V_{\mathcal{G}}(B)$. Then, for each $g \in \mathcal{G}$, we have one edge connecting the vertex in $V_{\mathcal{G}}(A)$ and the vertex in $V_{\mathcal{G}}(B)$ that contain g^h and one edge connecting the vertex in $V_{\mathcal{G}}(A)$ and the vertex in $V_{\mathcal{G}}(B)$ that contain g^t. Due to the 1-to-1 correspondence between the vertices of $AG(A, B)$ and the $\mathcal{G}$-adjacencies in $V_{\mathcal{G}}(A)$ and $V_{\mathcal{G}}(B)$, we can identify each adjacency with its corresponding vertex.

We know that $AG(A, B)$ is composed of two types of connected components, cycles and paths, alternating vertices in $V_{\mathcal{G}}(A)$ and $V_{\mathcal{G}}(B)$ [1]. A path that has one endpoint in $V_{\mathcal{G}}(A)$ and the other in $V_{\mathcal{G}}(B)$ is called an *AB-path.* In the same way, both endpoints of an *AA-path* are in $V_{\mathcal{G}}(A)$, as well as both endpoints of a *BB-path* are in $V_{\mathcal{G}}(B)$. Furthermore, the adjacency graph can have two extra types of components: each $\mathcal{G}$-adjacency that corresponds to a linear (respect. circular) chromosome is a *linear* (respect. *circular*) *singleton.* Observe that linear singletons are particular cases of AA-paths and BB-paths. If $\mathcal{G}_A = \mathcal{G}_B = \mathcal{G}$, the adjacency graph is composed only of clean $\mathcal{G}$-adjacencies, has no singletons and is said to be *clean.* An example of an adjacency graph is given in Fig. 2.

Singletons, AB-paths composed of one single edge, and cycles composed of two edges are said to be *DCJ-sorted.* Longer paths and cycles are said to be

DCJ-unsorted. We call *DCJ-sorting* of A into B the procedure of using DCJ operations to turn $AG(A, B)$ into DCJ-sorted components. The *DCJ distance* of A and B, denoted by $d_{DCJ}(A, B)$, corresponds to the minimum number of steps required to do a DCJ-sorting of A into B and can be easily obtained:

Theorem 1 ([1]). *Given a genome A over $\mathcal{G}_A$ and a genome B over $\mathcal{G}_B$, we have $d_{DCJ}(A, B) = n - c - \frac{b}{2}$, where n is the number of markers in $\mathcal{G} = \mathcal{G}_A \cap \mathcal{G}_B$, and c and b are, respectively, the number of cycles and AB-paths in $AG(A, B)$.*

Bergeron *et al.* [1] observed that the number of AB-paths in $AG(A, B)$ is even and that an *optimal* DCJ operation either increases the number of cycles by one, or the number of AB-paths by two (decreasing the DCJ distance by one). In the same way, a *neutral* operation does not affect the number of cycles and AB-paths in the graph, while a *counter-optimal* operation either decreases the number of cycles by one, or the number of AB-paths by two. The problem of finding an optimal sequence of operations that do a DCJ-sorting of A into B can be solved with a simple greedy linear time algorithm [1].

Now let $\mathcal{A}$ be the set of markers that occur only in genome A and let $\mathcal{B}$ be the set of markers that occur only in genome B, that is, $\mathcal{A} = \mathcal{G}_A \setminus \mathcal{G}_B$ and $\mathcal{B} = \mathcal{G}_B \setminus \mathcal{G}_A$. The markers in $\mathcal{A}$ and $\mathcal{B}$ are represented in $AG(A, B)$ as labels and singletons, but they are simply ignored by the approaches to compute the DCJ distance and sorting sequence, mentioned above. However, in order to completely sort A into B, the markers in $\mathcal{A}$ have to be deleted, while the markers in $\mathcal{B}$ have to be inserted. No DCJ operation is actually able to do an insertion or a deletion. Moreover, no operation is able to delete and insert at the same time (such an event would be a replacement, which is not accepted in the model we consider). Thus, for the purpose of this study, an operation is either a DCJ operation, or an *insertion*, or a *deletion*. We will refer to insertions and deletions as *indel* operations. A DCJ and an indel operation have the same cost and we define the *DCJ-indel distance* of A and B, denoted by $d_{DCJ}^{id}(A, B)$, as the minimum number of DCJ and indel operations required to transform A into B.

We can then establish a first simple upper bound for the DCJ-indel distance:

Observation 1 *Given a genome A over $\mathcal{G}_A$ and a genome B over $\mathcal{G}_B$, we have $d_{DCJ}^{id}(A, B) \leq d_{DCJ}(A, B) + |\mathcal{A}| + |\mathcal{B}|$, where $\mathcal{A} = \mathcal{G}_A \setminus \mathcal{G}_B$ and $\mathcal{B} = \mathcal{G}_B \setminus \mathcal{G}_A$.*

3 Accumulating Runs with Optimal DCJ Operations

Observe that a $\mathcal{G}$-adjacency with a non-empty label ℓ can be cut in at least two different positions, either before or after ℓ. Since the position of the cut does not change the effect of the DCJ operation on $d_{DCJ}(A, B)$, we can choose to cut at positions that allow the concatenation of the labels of the original $\mathcal{G}$-adjacencies. As a consequence, a set of labels in $\mathcal{G}$-adjacencies of genome A can be first *accumulated* with DCJ operations and later deleted at once. In the same way, a set of labels in $\mathcal{G}$-adjacencies of genome B can be first inserted at once as a *cluster* and later split with DCJ operations, as we can see in Fig. 3.

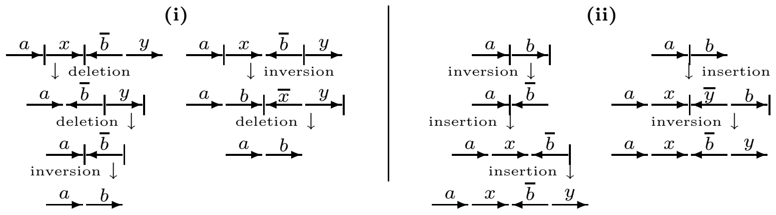

Fig. 3. (i) Two different scenarios sorting $\{\circ a x \overline{b} y \circ\}$ into $\{\circ a b \circ\}$. In the left we have two separate deletions and an optimal DCJ (inversion). In the right, we first perform the inversion, accumulating x and y, so that they can be deleted at once, saving one step. (ii) Conversely, while sorting $\{\circ a b \circ\}$ into $\{\circ a x \overline{b} y \circ\}$, instead of two insertions (left), we can insert a cluster at once and later split it with an inversion (right), saving one step.

Due to the following observation, without loss of generality, we allow operations on both genomes A and B, in order to be able to concatenate labels in $\mathcal{G}$-adjacencies of both genomes. Regarding the operations applied on genome B, this approach can be seen as a backtracing to find the best moment to do a cluster insertion in genome A. An algorithm sorting genome A into B can be derived from this approach.

Observation 2 *Given two genomes A and B, any pair of sequences s_1 and s_2 composed of DCJ and indel operations acting on both genomes A and B, transforming respectively A and B into an intermediate genome I, has a corresponding sequence acting only on genome A, that is, transforming A into B, with length $|s_1| + |s_2|$.*

Given a component C of $AG(A, B)$, we can obtain a string $\ell(C)$ by the concatenation of the labels of the $\mathcal{G}$-adjacencies of C in the order in which they appear. Cycles, AA-paths and BB-paths can be read in any direction, but AB-paths should always be read from A to B. If C is a cycle and has labels in both genomes A and B, we should start to read in a labeled $\mathcal{G}$-adjacency v of genome A, such that the first labeled vertex before v is a $\mathcal{G}$-adjacency in genome B; otherwise C has labels in at most one genome and we can start anywhere. Each maximal substring of $\ell(C)$ in $\widehat{\mathcal{A}}^+$ (respectively in $\widehat{\mathcal{B}}^+$) is called an *$\mathcal{A}$-run* (respectively a *$\mathcal{B}$-run*). Each $\mathcal{A}$-run or $\mathcal{B}$-run can be simply called a *run*. A component composed only of clean $\mathcal{G}$-adjacencies has no run and is said to be *clean*, otherwise the component is *labeled*. We denote by $\Lambda(C)$ the number of runs in a component C. A path can have any number of runs, while a cycle has zero, one, or an even number of runs. Fig. 4 shows a BB-path with 4 runs.

Proposition 1. *If $\gamma_1\gamma_2$ is a clean $\mathcal{G}$-adjacency in a DCJ-unsorted component C of $AG(A, B)$, such that neither γ_1 nor γ_2 are telomeres, then it is always possible to extract a clean cycle from C with an optimal DCJ operation.*

Proof. If $\gamma_1\gamma_2$ is in $V_G(B)$, we apply a DCJ on the two vertices $\gamma_1\ell_1\gamma_3$ and $\gamma_2\ell_2\gamma_4$ of $V_G(A)$ that are neighbors of $\gamma_1\gamma_2$, creating the two new vertices $\gamma_3\overline{\ell_1}\ell_2\gamma_4$ and

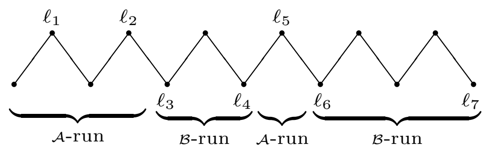

Fig. 4. A BB-path with 4 runs. Only the labels of the $\mathcal{G}$-adjacencies are represented.

$\gamma_1\gamma_2$. Observe that the vertex $\gamma_1\gamma_2$ in $V_{\mathcal{G}}(B)$ and the new vertex $\gamma_1\gamma_2$ in $V_{\mathcal{G}}(A)$ are extracted into a clean cycle. Analogously, if $\gamma_1\gamma_2$ is in $V_{\mathcal{G}}(A)$, we do the same procedure using the two vertices of $V_{\mathcal{G}}(B)$ that are neighbors of $\gamma_1\gamma_2$. □

Proposition 2. *A run can be entirely accumulated in the label of one single $\mathcal{G}$-adjacency with optimal DCJ operations.*

Proof. A run that is not yet accumulated is distributed over two or more $\mathcal{G}$-adjacencies in one genome. The $\mathcal{G}$-adjacencies in the other genome within the run are clean. We can thus apply optimal DCJs that extract clean cycles (Proposition 1) and accumulate the entire run in the label of one $\mathcal{G}$-adjacency. □

Proposition 2 immediately gives a tighter upper bound for the distance:

Lemma 1. *Given two genomes A and B without duplications, we have*

$$d_{DCJ}^{id}(A,B) \leq d_{DCJ}(A,B) + \sum_{C \in AG(A,B)} \Lambda(C).$$

4 Merging Runs in One Component

For some instances of A and B, the upper bound of Lemma 1 gives the exact number of steps required to sort A into B. However, since two runs can be *merged* together with a DCJ operation, the DCJ-indel distance is often smaller than this upper bound. Given a DCJ operation ρ, let Λ_0 and Λ_1 be, respectively, the number of runs in $AG(A,B)$ before and after ρ. We define $\Delta\Lambda(\rho) = \Lambda_1 - \Lambda_0$.

Proposition 3. *Given any DCJ operation ρ, we have $\Delta\Lambda(\rho) \geq -2$.*

Proof. If ρ cuts between an $\mathcal{A}$-run r_1 and a $\mathcal{B}$-run r_2 and between an $\mathcal{A}$-run r_3 and a $\mathcal{B}$-run r_4, with $r_1 \neq r_3$ and $r_2 \neq r_4$, and joins r_1 with r_3 and r_2 with r_4, then $\Delta\Lambda(\rho) = -2$. As a DCJ has at most two cuts and two joins, it is not possible to do better, that is $\Delta\Lambda(\rho) \geq -2$. □

In order to obtain the exact formula for the DCJ-indel distance, we will first analyze the components of the adjacency graph separately. Given two genomes A and B and a component $C \in AG(A,B)$, we denote by $d_{DCJ}(C)$ the minimum number of DCJ operations required to do a separate DCJ-sorting in C, applying DCJs only on vertices of C (or vertices that result from DCJs applied on vertices that were in C). From [3], we know that it is possible to do a separate DCJ-sorting using only optimal DCJs in any component of $AG(A, B)$, or, in other words, $d_{DCJ}(A, B) = \sum_{C \in AG(A,B)} d_{DCJ}(C)$. Moreover, we denote by $\lambda(C)$ the minimum number of runs that we can obtain doing a separate DCJ-sorting in C with optimal DCJ operations. We then have:

Proposition 4. *Given a component C in $AG(A, B)$, we have $\lambda(C) = \lceil \frac{\Lambda(C)+1}{2} \rceil$, if $\Lambda(C) \geq 1$. Otherwise $\lambda(C) = 0$.*

Proof. The proof is by induction on $i = \Lambda(C)$ and the hypothesis is $T(i) = \lceil \frac{i+1}{2} \rceil$. A labeled DCJ-sorted component can have one or two runs, thus we need two base cases, $T(1) = 1$ and $T(2) = 2$. These cases can be easily verified. More intricate is the inductive step, for $i \geq 3$.

When $i \geq 3$ is odd, we can merge the first and the last runs with an optimal DCJ, obtaining a cycle with $i - 1$ runs. This gives $T(i) = T(i-1) = \frac{i-1+2}{2} = \lceil \frac{i+1}{2} \rceil$. If $i \geq 5$, we can also do an optimal DCJ that has $\Delta\Lambda = -2$, extracting a cycle with even $i' \geq 2$ runs and leaving the path with odd $i'' \geq 1$ runs, such that $i = i' + i'' + 2$ and $T(i) = T(i') + T(i'') = \frac{i'+2}{2} + \frac{i''+1}{2} = \frac{i+1}{2} = \lceil \frac{i+1}{2} \rceil$.

When $i \geq 4$ is even, any optimal DCJ merging runs would extract a cycle with even $i' \geq 2$ runs and leave the other component with $i'' \geq 1$ runs. One way is to do a DCJ that has $\Delta\Lambda = -1$, such that $i = i' + i'' + 1$ and $i'' \geq 1$ is odd. This gives $T(i) = \frac{i'+2}{2} + \frac{i''+1}{2} = \frac{i+2}{2} = \lceil \frac{i+1}{2} \rceil$. If $i \geq 6$, it is also possible to do a DCJ that has $\Delta\Lambda = -2$, such that $i = i' + i'' + 2$ and $i'' \geq 2$ is even. We then have $T(i) = \frac{i'+2}{2} + \frac{i''+2}{2} = \frac{i+2}{2} = \lceil \frac{i+1}{2} \rceil$. (All other optimal DCJs applied between runs of components with $\Lambda \geq 3$ would lead to greater values of λ.) $\square$

If λ_0 and λ_1 are, respectively, the sum of the number λ for the components of the adjacency graph before and after ρ, we define $\Delta\lambda(\rho) = \lambda_1 - \lambda_0$. By the definition of λ, any optimal DCJ ρ acting on a single component has $\Delta\lambda(\rho) \geq 0$. However, considering the case in which only one component is affected by ρ, we still need to investigate $\Delta\lambda(\rho)$ when ρ is counter-optimal or neutral.

Proposition 5. *Given a DCJ operation ρ acting on a single component, we have $\Delta\lambda(\rho) \geq 0$, if ρ is counter-optimal, or $\Delta\lambda(\rho) \geq -1$, if ρ is neutral.*

Proof. The linearization of a cycle is the only counter-optimal DCJ that acts on a single component. This can decrease neither Λ, nor λ. Moreover, when $\Lambda \leq 2$, it is not possible to decrease the number λ with any DCJ. When the component has $\Lambda = 3$, the best we can get with a neutral ρ is $\Delta\Lambda(\rho) = -1$. This gives $\lambda_1 = \lceil \frac{(3-1)+1}{2} \rceil = \lceil \frac{3}{2} \rceil = \lceil \frac{3+1}{2} \rceil = \lambda_0$, that is, $\Delta\lambda(\rho) = 0$. And when the component has $\Lambda \geq 4$, we can get $\Delta\Lambda(\rho) = -2$ with a neutral ρ, resulting in $\lambda_1 = \lceil \frac{(\Lambda(C)-2)+1}{2} \rceil = \lceil \frac{\Lambda(C)+1}{2} \rceil - 1 = \lambda_0 - 1$, that is, $\Delta\lambda(\rho) = -1$. $\square$

We denote by $d_{DCJ}^{id}(C)$ the minimum number of DCJ and indel operations required to sort separately a component C of $AG(A, B)$.

Proposition 6. *If C is a component of $AG(A, B)$, then we have $d_{DCJ}^{id}(C) = d_{DCJ}(C) + \lambda(C)$.*

Proof. By the definition of λ, the best we can do with optimal DCJs is $d_{DCJ}(C)+\lambda(C)$. From Proposition 5, we know that $\Delta\lambda(\rho) \geq 0$ if ρ is a counter-optimal DCJ, thus we can only get longer sorting scenarios if we use such operations. We also know that $\Delta\lambda(\rho) \geq -1$ if ρ is neutral, and, since this kind of operation increases the sorting scenario by one with respect to the scenario with only optimal DCJs, this gives at least $d_{DCJ}(C) + \lambda(C)$. □

Proposition 6 gives a new upper bound for the DCJ-indel distance:

Lemma 2. *Given two genomes A and B without duplications, we have*

$$d^{id}_{DCJ}(A,B) \leq d_{DCJ}(A,B) + \sum_{C \in AG(A,B)} \lambda(C).$$

Proof. We can sort the components separately with $\sum_{C\in AG(A,B)} d^{id}_{DCJ}(C)$ steps, which corresponds exactly to $d_{DCJ}(A,B) + \sum_{C\in AG(A,B)} \lambda(C)$. □

Since $\lambda(C) \leq \Lambda(C)$, the upper bound given by Lemma 2 is tighter than the one given by Lemma 1, but can still be improved. Observe that a parsimonious scenario may not simply consist of optimal DCJ operations, insertions and deletions. Sometimes a neutral DCJ can lead to a shorter sequence of operations sorting one genome into another, as we can see in Fig. 5.

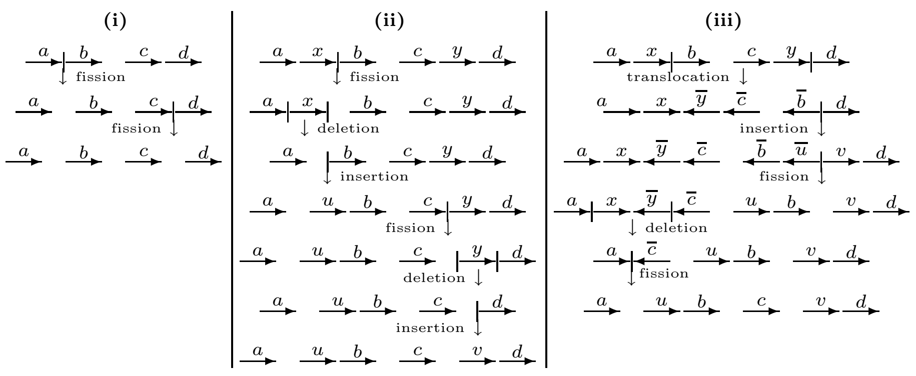

Fig. 5. An optimal scenario sorting $\{\circ ab\circ, \circ cd\circ\}$ into $\{\circ a\circ, \circ b\circ, \circ c\circ, \circ d\circ\}$ (i) and two different scenarios sorting $\{\circ axb\circ, \circ cyd\circ\}$ into $\{\circ a\circ, \circ ub\circ, \circ c\circ, \circ vd\circ\}$. In (ii) in addition to the two optimal DCJ operations (fissions) from (i) we have two insertions and two deletions, using six steps. In (iii) we first use a neutral DCJ operation (translocation) that allows us to do only one deletion and one insertion, achieving a total of five steps.

5 Recombinations and the DCJ-indel Distance

A DCJ operation ρ that acts on two components is called *recombination* and can have $\Delta\lambda(\rho) = -2$. The two components on which the cuts are applied are called

Table 1. Path recombinations that have $\varDelta d \leq -1$ and allow the best reuse of the resultants. Optimal recombinations are in the left, neutral recombinations in the right.

sources	**resultants**	$\boldsymbol{\varDelta\lambda}$	$\boldsymbol{\varDelta_{dcj}}$	$\boldsymbol{\varDelta d}$
$AA_{\mathcal{AB}} + BB_{\mathcal{AB}}$	$AB_{\bullet} + AB_{\bullet}$	-2	0	-2
$AA_{\mathcal{A}} + BB_{\mathcal{AB}}$	$AB_{\bullet} + AB_{\mathcal{AB}}$	-1	0	-1
$BB_{\mathcal{A}} + AA_{\mathcal{AB}}$	$AB_{\bullet} + AB_{\mathcal{BA}}$	-1	0	-1
$AA_{\mathcal{B}} + BB_{\mathcal{AB}}$	$AB_{\bullet} + AB_{\mathcal{BA}}$	-1	0	-1
$BB_{\mathcal{B}} + AA_{\mathcal{AB}}$	$AB_{\bullet} + AB_{\mathcal{AB}}$	-1	0	-1
$AA_{\mathcal{A}} + BB_{\mathcal{A}}$	$AB_{\bullet} + AB_{\bullet}$	-1	0	-1
$AA_{\mathcal{B}} + BB_{\mathcal{B}}$	$AB_{\bullet} + AB_{\bullet}$	-1	0	-1

sources	**resultants**	$\boldsymbol{\varDelta\lambda}$	$\boldsymbol{\varDelta_{dcj}}$	$\boldsymbol{\varDelta d}$
$AA_{\mathcal{AB}} + AA_{\mathcal{AB}}$	$AA_{\mathcal{A}} + AA_{\mathcal{B}}$	-2	$+1$	-1
$BB_{\mathcal{AB}} + BB_{\mathcal{AB}}$	$BB_{\mathcal{A}} + BB_{\mathcal{B}}$	-2	$+1$	-1
$AA_{\mathcal{AB}} + AB_{\mathcal{AB}}$	$AB_{\bullet} + AA_{\mathcal{A}}$	-2	$+1$	-1
$AA_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$AB_{\bullet} + AA_{\mathcal{B}}$	-2	$+1$	-1
$BB_{\mathcal{AB}} + AB_{\mathcal{AB}}$	$AB_{\bullet} + BB_{\mathcal{B}}$	-2	$+1$	-1
$BB_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$AB_{\bullet} + BB_{\mathcal{A}}$	-2	$+1$	-1
$AB_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$AB_{\bullet} + AB_{\bullet}$	-2	$+1$	-1

sources and the components obtained after the joinings are called *resultants* of the recombination.

Proposition 7. *Given any recombination ρ, we have $\varDelta\lambda(\rho) \geq -2$.*

Proof. Only the recombinations that decrease or do not change the number of runs ($\varDelta\Lambda \leq 0$) have to be analyzed (we can not have $\varDelta\lambda \leq -1$ if the number of runs increases). First consider the recombination of two paths with i and j runs, respectively, that result in two new paths with i' and j' runs. Observe that the best we can have is when i and j are even, i' and j' are odd and $\varDelta\Lambda = -2$, that gives: $\lambda_1 = \lceil \frac{i'+1}{2} \rceil + \lceil \frac{j'+1}{2} \rceil = \frac{i'+j'+2}{2} = \frac{i+j}{2} = \frac{i}{2} + \frac{j}{2} = \lceil \frac{i+1}{2} \rceil - 1 + \lceil \frac{j+1}{2} \rceil - 1 = \lambda_0 - 2$. The analysis of recombinations involving cycles is analogous. $\square$

Given a recombination ρ, let $\varDelta_{dcj}(\rho)$ be respectively 0, +1 and +2 depending whether ρ is optimal, neutral or counter-optimal. Any recombination applied to a vertex of an AA-path and a vertex of a BB-path is optimal [3]. A recombination applied to vertices of two different AB-paths can be either neutral, when the result is also a pair of AB-paths, or counter-optimal, when the result is a pair composed of an AA-path and a BB-path. All other types of path recombinations are neutral. In addition, all recombinations involving at least one cycle are counter-optimal. We define $\varDelta d(\rho) = \varDelta_{dcj}(\rho) + \varDelta\lambda(\rho)$. Any counter-optimal recombination has $\varDelta d \geq 0$, thus only path recombinations can have $\varDelta d \leq -1$.

Let $\mathcal{A} = \widehat{\mathcal{A}}^{+}(\widehat{\mathcal{B}}^{+}\widehat{\mathcal{A}}^{+})^{*}$ (respect. $\mathcal{B} = \widehat{\mathcal{B}}^{+}(\widehat{\mathcal{A}}^{+}\widehat{\mathcal{B}}^{+})^{*}$) be a sequence with an odd (≥ 1) number of runs, starting and ending with a run over $\widehat{\mathcal{A}}$ (respect. over $\widehat{\mathcal{B}}$). We can then make any combination of $\mathcal{A}$ and $\mathcal{B}$, such as $\mathcal{AB} = \widehat{\mathcal{A}}^{+}(\widehat{\mathcal{B}}^{+}\widehat{\mathcal{A}}^{+})^{*}\widehat{\mathcal{B}}^{+}$, that is a sequence with an even (≥ 2) number of runs, starting with a run over $\widehat{\mathcal{A}}$ and ending with a run over $\widehat{\mathcal{B}}$. An empty sequence (with no run) is represented by ε. Then each one of the notations AA_{ε}, $AA_{\mathcal{A}}$, $AA_{\mathcal{B}}$, $AA_{\mathcal{AB}}$, BB_{ε}, $BB_{\mathcal{A}}$, $BB_{\mathcal{B}}$, $BB_{\mathcal{AB}}$, AB_{ε}, $AB_{\mathcal{A}}$, $AB_{\mathcal{B}}$, $AB_{\mathcal{AB}}$ and $AB_{\mathcal{BA}}$ represents a particular type of path (AA, BB or AB) with a particular structure of runs (ε, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{AB}$ or $\mathcal{BA}$). The complete set of path recombinations with $\varDelta d \leq -1$ is given in Table 1. In Table 2 we also list recombinations with $\varDelta d = 0$ that create at least one source of recombinations of Table 1. We denote by $AB_{\bullet}$ an AB-path that can not be a source of a recombination in Tables 1 and 2, such as AB_{ε}, $AB_{\mathcal{A}}$ and $AB_{\mathcal{B}}$.

Table 2. Recombinations that have $\varDelta d = 0$ and create resultants that can be used in recombinations with $\varDelta d \leq -1$.

sources	**resultants**	$\boldsymbol{\Delta\lambda}$	$\boldsymbol{\Delta_{dcj}}$	$\boldsymbol{\Delta d}$
$AA_{\mathcal{A}} + AB_{\mathcal{BA}}$	$AB_{\bullet} + AA_{\mathcal{AB}}$	-1	$+1$	0
$AA_{\mathcal{B}} + AB_{\mathcal{AB}}$	$AB_{\bullet} + AA_{\mathcal{AB}}$	-1	$+1$	0
$BB_{\mathcal{A}} + AB_{\mathcal{AB}}$	$AB_{\bullet} + BB_{\mathcal{AB}}$	-1	$+1$	0
$BB_{\mathcal{B}} + AB_{\mathcal{BA}}$	$AB_{\bullet} + BB_{\mathcal{AB}}$	-1	$+1$	0

sources	**resultants**	$\boldsymbol{\Delta\lambda}$	$\boldsymbol{\Delta_{dcj}}$	$\boldsymbol{\Delta d}$
$AA_{\mathcal{A}} + BB_{\mathcal{B}}$	$AB_{\bullet} + AB_{\mathcal{AB}}$	0	0	0
$AA_{\mathcal{B}} + BB_{\mathcal{A}}$	$AB_{\bullet} + AB_{\mathcal{BA}}$	0	0	0
$AB_{\mathcal{AB}} + AB_{\mathcal{AB}}$	$AA_{\mathcal{A}} + BB_{\mathcal{B}}$	-2	$+2$	0
$AB_{\mathcal{BA}} + AB_{\mathcal{BA}}$	$AA_{\mathcal{B}} + BB_{\mathcal{A}}$	-2	$+2$	0

Proposition 8. *The recombinations with $\varDelta d = 0$ involving cycles or circular singletons cannot create new components that can be used as sources of recombinations listed in Tables 1 and 2.*

Proof. A recombination ρ with $\varDelta d = 0$ involving a cycle or a circular singleton C would integrate C to another component C' without changing the type or the structure of runs in C'. Thus, if C' is a source of a recombination in these tables after ρ, C' was already the same type of source before ρ. And if C' was not a source before ρ, C' cannot become a source after ρ. $\square$

With Proposition 8 we already have an exact formula to d_{DCJ}^{id} for a particular set of instances. Given a $\mathcal{G}$-adjacency $\gamma\ell\circ$ of a genome A such that $\gamma \neq \circ$, then γ is said to be a *tail* of a linear chromosome in A. Two genomes are *co-tailed* if their sets of tails are equal (this includes two genomes composed only of circular chromosomes).

Theorem 2. *Given two co-tailed genomes A and B without duplications, we have $d_{DCJ}^{id}(A,B) = d_{DCJ}(A,B) + \sum_{C \in AG(A,B)} \lambda(C)$.*

Proof. The graph $AG(A,B)$ for co-tailed genomes A and B can have only singletons (that could be $AA_{\mathcal{A}}$ and $BB_{\mathcal{B}}$), cycles and AB-paths of one edge. These AB-paths could be $AB_{\mathcal{AB}}$, but never $AB_{\mathcal{BA}}$, thus no recombination listed in Tables 1 and 2 is possible. $\square$

Now we continue the analysis for the general case. The two sources of a recombination can also be called *partners*. Looking at Table 1 we observe that all partners of $AB_{\mathcal{AB}}$ and $AB_{\mathcal{BA}}$ paths are also partners of $AA_{\mathcal{AB}}$ and $BB_{\mathcal{AB}}$ paths, all partners of $AA_{\mathcal{A}}$ and $AA_{\mathcal{B}}$ paths are also partners of $AA_{\mathcal{AB}}$ paths and all partners of $BB_{\mathcal{A}}$ and $BB_{\mathcal{B}}$ paths are also partners of $BB_{\mathcal{AB}}$ paths. Moreover, some resultants of recombinations in Tables 1 and 2 can be used in other recombinations. These observations allow the identification of groups, as listed in Tables 3 and 4.

The deductions shown in Tables 3 and 4 can be computed with an approach that greedily maximizes the number of recombinations in P, Q, T, S, M and N in this order. The P part contains only one operation and is thus very simple. The same happens with Q, since the two groups in this part are exclusive after applying P. The only part that requires more attention is T, in which some combinations of operations can happen at the same time and the order can be relevant. The part S is only the application of all possible remaining operations

Table 3. All recombination groups obtained from Table 1. Observe that the last four groups in T are subsets of groups in Q and the last ten groups in S are subsets of groups in Q and T. The column **scr** indicates the contribution of each path in the distance decrease (the table is sorted in descending order with respect to this column).

	sources	**resultants**	$\boldsymbol{\Delta d}$	**scr**
P	$AA_{\mathcal{AB}} + BB_{\mathcal{AB}}$	$2AB_\bullet$	-2	-1
Q	$2AA_{\mathcal{AB}} + BB_{\mathcal{A}} + BB_{\mathcal{B}}$	$4AB_\bullet$	-3	$-3/4$
	$2BB_{\mathcal{AB}} + AA_{\mathcal{A}} + AA_{\mathcal{B}}$	$4AB_\bullet$	-3	$-3/4$
T	$AA_{\mathcal{AB}} + BB_{\mathcal{A}} + AB_{\mathcal{AB}}$	$3AB_\bullet$	-2	$-2/3$
	$AA_{\mathcal{AB}} + BB_{\mathcal{B}} + AB_{\mathcal{BA}}$	$3AB_\bullet$	-2	$-2/3$
	$BB_{\mathcal{AB}} + AA_{\mathcal{A}} + AB_{\mathcal{BA}}$	$3AB_\bullet$	-2	$-2/3$
	$BB_{\mathcal{AB}} + AA_{\mathcal{B}} + AB_{\mathcal{AB}}$	$3AB_\bullet$	-2	$-2/3$
	$2AA_{\mathcal{AB}} + BB_{\mathcal{A}}$	$2AB_\bullet + AA_{\mathcal{B}}$	-2	$-2/3$
	$2AA_{\mathcal{AB}} + BB_{\mathcal{B}}$	$2AB_\bullet + AA_{\mathcal{A}}$	-2	$-2/3$
	$2BB_{\mathcal{AB}} + AA_{\mathcal{A}}$	$2AB_\bullet + BB_{\mathcal{B}}$	-2	$-2/3$
	$2BB_{\mathcal{AB}} + AA_{\mathcal{B}}$	$2AB_\bullet + BB_{\mathcal{A}}$	-2	$-2/3$
S	$AA_{\mathcal{A}} + BB_{\mathcal{A}}$	$2AB_\bullet$	-1	$-1/2$
	$AA_{\mathcal{B}} + BB_{\mathcal{B}}$	$2AB_\bullet$	-1	$-1/2$
	$AB_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$2AB_\bullet$	-1	$-1/2$
	$BB_{\mathcal{AB}} + AA_{\mathcal{A}}$	$AB_\bullet + AB_{\mathcal{AB}}$	-1	$-1/2$
	$AA_{\mathcal{AB}} + BB_{\mathcal{A}}$	$AB_\bullet + AB_{\mathcal{BA}}$	-1	$-1/2$
	$BB_{\mathcal{AB}} + AA_{\mathcal{B}}$	$AB_\bullet + AB_{\mathcal{BA}}$	-1	$-1/2$
	$AA_{\mathcal{AB}} + BB_{\mathcal{B}}$	$AB_\bullet + AB_{\mathcal{AB}}$	-1	$-1/2$
	$AA_{\mathcal{AB}} + AB_{\mathcal{AB}}$	$AB_\bullet + AA_{\mathcal{A}}$	-1	$-1/2$
	$AA_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$AB_\bullet + AA_{\mathcal{B}}$	-1	$-1/2$
	$BB_{\mathcal{AB}} + AB_{\mathcal{AB}}$	$AB_\bullet + BB_{\mathcal{B}}$	-1	$-1/2$
	$BB_{\mathcal{AB}} + AB_{\mathcal{BA}}$	$AB_\bullet + BB_{\mathcal{A}}$	-1	$-1/2$
	$AA_{\mathcal{AB}} + AA_{\mathcal{AB}}$	$AA_{\mathcal{B}} + AA_{\mathcal{A}}$	-1	$-1/2$
	$BB_{\mathcal{AB}} + BB_{\mathcal{AB}}$	$BB_{\mathcal{B}} + BB_{\mathcal{A}}$	-1	$-1/2$

Table 4. All recombination groups that contain operations from Tables 1 and 2. The groups in N are subsets of the groups in M. The table is sorted in descending order with respect to the contribution of each path in the distance decrease (column **scr**).

	sources	**resultants**	$\boldsymbol{\Delta d}$	**scr**
M	$2AB_{\mathcal{AB}} + AA_{\mathcal{B}} + BB_{\mathcal{A}}$	$4AB_\bullet$	-2	$-1/2$
	$2AB_{\mathcal{BA}} + AA_{\mathcal{A}} + BB_{\mathcal{B}}$	$4AB_\bullet$	-2	$-1/2$
N	$AB_{\mathcal{AB}} + AA_{\mathcal{B}} + BB_{\mathcal{A}}$	$3AB_\bullet$	-1	$-1/3$
	$AB_{\mathcal{BA}} + AA_{\mathcal{A}} + BB_{\mathcal{B}}$	$3AB_\bullet$	-1	$-1/3$
N	$2AB_{\mathcal{AB}} + AA_{\mathcal{B}}$	$2AB_\bullet + AA_{\mathcal{A}}$	-1	$-1/3$
	$2AB_{\mathcal{AB}} + BB_{\mathcal{A}}$	$2AB_\bullet + BB_{\mathcal{B}}$	-1	$-1/3$
	$2AB_{\mathcal{BA}} + AA_{\mathcal{A}}$	$2AB_\bullet + AA_{\mathcal{B}}$	-1	$-1/3$
	$2AB_{\mathcal{BA}} + BB_{\mathcal{B}}$	$2AB_\bullet + BB_{\mathcal{A}}$	-1	$-1/3$

with $\Delta d = -1$. After S, the two groups in M are exclusive and then the same happens to the six groups in N.

The results presented in this section give rise to the following theorem, that gives the exact formula for the DCJ-indel distance:

Theorem 3. *Given two genomes A and B without duplications, we have*

$$d_{DCJ}^{id}(A,B) = d_{DCJ}(A,B) + \sum_{C \in AG(A,B)} \lambda(C) - 2P - 3Q - 2T - S - 2M - N,$$

where P, Q, T, S, M and N are computed as described above.

Both $AG(A,B)$ and $d_{DCJ}(A,B)$ can be computed in linear time [1]. The runs can be obtained by a single walk through each component of $AG(A,B)$, which is also linear. The algorithm to compute P, Q, T, S, M and N is a finite sequence of **if** and **else** statements, that depends only on the number of each type of labeled path in $AG(A,B)$, thus the whole procedure takes linear time.

6 Experiments and Discussion

We used our method to analyze the evolution of *Rickettsia*, a group of obligate intracellular parasites that are carried by many vectors (frequently hematophagous arthropods) and occasionally transmitted from the vector to mammalians (including humans), causing several diseases (typhus, spotted fever, etc.) [2]. 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. There are several completely sequenced *Rickettsia* genomes, and most of them are closely related [2]. The exception is *R. bellii*, which shows a high level of rearrangement with respect to the others. We compared *R. bellii* with six other species of *Rickettsia*, observing in all pairwise analyses a considerable reduction of the indels (see Table 5), when they are grouped into runs (column $\Sigma\Lambda$) and into merged runs (column $\Sigma\lambda$). Although these are preliminary results, they could suggest that each cluster is composed of genes that have been lost together during the evolution of *Rickettsia*.

Table 5. Comparing *R. bellii* (1.52 Mbp) with six other species of *Rickettsia*.

species	Mbp	$\lvert\mathcal{A}\rvert + \lvert\mathcal{B}\rvert$	$\Sigma\Lambda$	$\Sigma\lambda$	d_{DCJ}	d_{DCJ}^{id}	species	Mbp	$\lvert\mathcal{A}\rvert + \lvert\mathcal{B}\rvert$	$\Sigma\Lambda$	$\Sigma\lambda$	d_{DCJ}	d_{DCJ}^{id}
R. felis	1.55	333	241	181	312	493	*R. conorii*	1.27	277	192	153	261	414
R. massiliae	1.36	302	218	172	276	448	*R. prowazekii*	1.11	241	130	117	197	314
R. africanae	1.28	290	212	166	260	426	*R. typhi*	1.11	239	126	114	195	309

Discussion. We propose the first linear time algorithm to compute the distance between two genomes with unequal content, but without duplications, taking into consideration DCJ and indel operations. With this method we analyze a group of bacteria, obtaining interesting results. Due to the lack of available data, we could not yet perform analyses on linear genomes, which would let us test the impact of path recombinations on the distance.

This work opens some perspectives. One is the development of a sorting algorithm that can be derived from the results presented here. Another issue that could be addressed next is the incorporation of replacements in the model, when an insertion and a deletion occur at the same position of the genome.

References

1. Bergeron, A., Mixtacki, J. and Stoye, J.: A unifying view of genome rearrangements. In *Proc. of WABI*, LNCS 4175, 163–173, 2006.
2. Blanc G. *et al.*: Reductive genome evolution from the mother of Rickettsia. *PLoS Genetics*, 3(1): e14, 2007.
3. Braga M. D. V. and Stoye J.: The solution space of sorting by DCJ. To appear in *Journal of Computational Biology*, 2010.
4. Yancopoulos, S. and Friedberg, R.: Sorting Genomes with Insertions, Deletions and Duplications by DCJ. In *Proc. of RECOMB-CG*, LNBI 5267, 170–183, 2008.
5. Yancopoulos, S., Attie, O. and Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346, 2005.