Computing the rearrangement distance of natural genomes

Leonard Bohnenkämper[0000−0003−4508−0078], Marília D. V. Braga[0000−0003−3558−6059], Daniel Doerr[0000−0002−3720−6227], and Jens Stoye[0000−0002−4656−7155]

Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany

Abstract. The computation of genomic distances has been a very active field of computational comparative genomics over the last 25 years. Substantial results include the polynomial-time computability of the inversion distance by Hannenhalli and Pevzner in 1995 and the introduction of the double-cut and join (DCJ) distance by Yancopoulos, Attie and Friedberg in 2005. Both results, however, rely on the assumption that the genomes under comparison contain the same set of unique *markers* (syntenic genomic regions, sometimes also referred to as *genes*). In 2015, Shao, Lin and Moret relax this condition by allowing for duplicate markers in the analysis. This generalized version of the genomic distance problem is NP-hard, and they give an ILP solution that is efficient enough to be applied to real-world datasets. A restriction of their approach is that it can be applied only to *balanced* genomes, that have equal numbers of duplicates of any marker. Therefore it still needs a delicate preprocessing of the input data in which excessive copies of unbalanced markers have to be removed.

In this paper we present an algorithm solving the genomic distance problem for *natural* genomes, in which any marker may occur an arbitrary number of times. Our method is based on a new graph data structure, the *multi-relational diagram*, that allows an elegant extension of the ILP by Shao, Lin and Moret to count *runs* of markers that are under- or over-represented in one genome with respect to the other and need to be inserted or deleted, respectively. With this extension, previous restrictions on the genome configurations are lifted, for the first time enabling an uncompromising rearrangement analysis. Any marker sequence can directly be used for the distance calculation.

The evaluation of our approach shows that it can be used to analyze genomes with up to a few ten thousand markers, which we demonstrate on simulated and real data.

Keywords: Comparative genomics · Genome rearrangement · DCJ-indel distance.

1 Introduction

The study of genome rearrangements has a long tradition in comparative genomics. A central question is how many (and what kind of) mutations have

occurred between the genomic sequences of two individual genomes. In order to avoid disturbances due to minor local effects, often the basic units in such comparisons are syntenic regions identified between the genomes under study, much larger than the individual DNA bases. We refer to such regions as *genomic markers*, or simply *markers*, although often one also finds the term *genes*.

Following the initial statement as an edit distance problem [15], a comprehensive trail of literature has addressed the problem of computing the number of rearrangements between two genomes in the past 25 years. In a seminal paper in 1995, Hannenhalli and Pevzner [12] introduced the first polynomial time algorithm for the computation of the inversion distance of transforming one chromosome into another one by means of segmental inversions. Later, the same authors generalized their results to the HP model [11] which is capable of handling multi-chromosomal genomes and accounts for additional genome rearrangements. Another breakthrough was the introduction of the double cut and join (DCJ) model [2, 18], that is able to capture many genome rearrangements and whose genomic distance is computable in linear time. The model is based on a simple operation in which the genome sequence is cut twice between two consecutive markers and re-assembled by joining the resulting four loose cut-ends in a different combination.

A prerequisite for applying the DCJ model in practice to study rearrangements in genomes of two related species is that their genomic marker sets must be identical and that any marker occurs exactly once in each genome. This severely limits its applicability in practice. Linear time extensions of the DCJ model allow markers to occur in only one of the two genomes, computing a genomic distance that minimizes the sum of DCJ and insertion/deletion (indel) events [5, 9]. Still, markers are required to be *singleton*, i.e., no duplicates can occur. When duplicates are allowed, the problem is more intrincate and all approaches proposed so far are NP-hard, see for instance [1, 6, 7, 14, 16, 17]. From the practical side, more recently, Shao *et al.* [17] presented an integer linear programming (ILP) formulation for computing the DCJ distance in presence of duplicates, but restricted to *balanced genomes*, where both genomes have equal numbers of duplicates. A generalization to unbalanced genomes was presented by Lyubetsky *et al.* [13], but their approach does not seem to be applicable to real data sets, see Section 4.1 for details.

In this paper we present the first feasible exact algorithm for solving the NP-hard problem of computing the distance under a general genome model where any marker may occur an arbitrary number of times in any of the two genomes, called *natural genomes*. Specifically, we adopt the *maximal matches* model where only markers appearing more often in one genome than in the other can be deleted or inserted. Our ILP formulation is based on the one from Shao *et al.* [17], but with an efficient extension that allows to count *runs* of markers that are under- or over-represented in one genome with respect to the other, so that the pre-existing model of minimizing the distance allowing DCJ and indel operations [5] can be adapted to our problem. With this extension, once

we have the genome markers, no other restriction on the genome configurations is imposed.

The evaluation of our approach shows that it can be used to analyze genomes with up to a few ten thousand markers, provided the number of duplicates is not too large.

An extended version of this paper containing omitted proofs and additional results appeared as an arxiv preprint [3]. The complete source code of our ILP implementation and the simulation software used for generating the benchmarking data in Section 4.2 are available from `https://gitlab.ub.uni-bielefeld.de/gi/ding`.

2 Preliminaries

A *genome* is a set of *chromosomes* and each chromosome can be linear or circular. Each *marker* in a chromosome is an oriented DNA fragment. The representation of a marker m in a chromosome can be the symbol m itself, if it is read in direct orientation, or the symbol $\overline{m}$, if it is read in reverse orientation. We represent a chromosome S of a genome A by a string s, obtained by the concatenation of all symbols in S, read in any of the two directions. If S is circular, we can start to read it at any marker and the string s is flanked by parentheses.

Given two genomes A and B, let $\mathcal{U}$ be the set of all markers that occur in both genomes. For each marker $m \in \mathcal{U}$, let $\Phi_A(m)$ be the number of occurrences of m in genome A and $\Phi_B(m)$ be the number of occurrences of m in genome B. We can then define $\Delta\Phi(m) = \Phi_A(m) - \Phi_B(m)$. If both $\Phi_A(m) > 0$ and $\Phi_B(m) > 0$, m is called a *common marker*. We denote by $\mathcal{G} \subseteq \mathcal{U}$ the set of common markers of A and B. The markers in $\mathcal{U} \backslash \mathcal{G}$ are called *exclusive markers*. For example, if we have two unichromosomal linear genomes $A = \{\mathtt{132\overline{5}\overline{4}354}\}$ and $B = \{\mathtt{1623173413}\}$, then $\mathcal{U} = \{\mathtt{1},\mathtt{2},\mathtt{3},\mathtt{4},\mathtt{5},\mathtt{6},\mathtt{7}\}$ and $\mathcal{G} = \{\mathtt{1},\mathtt{2},\mathtt{3},\mathtt{4}\}$. Furthermore, $\Delta\Phi(\mathtt{1}) = 1-3 = -2$, $\Delta\Phi(\mathtt{2}) = 1-1 = 0$, $\Delta\Phi(\mathtt{3}) = 2-3 = -1$, $\Delta\Phi(\mathtt{4}) = 2-1 = 1$, $\Delta\Phi(\mathtt{5}) = 2$, and $\Delta\Phi(\mathtt{6}) = \Delta\Phi(\mathtt{7}) = -1$.

2.1 The DCJ-indel model

A genome can be transformed or *sorted* into another genome with the following types of mutations:

- A *double-cut-and-join* (DCJ) is the operation that cuts a genome at two different positions (possibly in two different chromosomes), creating four open ends, and joins these open ends in a different way. This can represent many different rearrangements, such as inversions, translocations, fusions and fissions. For example, a DCJ can cut linear chromosome $\mathtt{12\overline{4}\overline{3}56}$ before and after $\mathtt{\overline{4}\overline{3}}$, creating the segments $\mathtt{12}\bullet$, $\bullet\mathtt{\overline{4}\overline{3}}\bullet$ and $\bullet\mathtt{56}$, where the symbol $\bullet$ represents the open ends. By joining the first with the third and the second with the fourth open end, we invert $\mathtt{\overline{4}\overline{3}}$ and obtain $\mathtt{123456}$.

– Since the genomes can have distinct multiplicity of markers, we also need to consider *insertions* and *deletions* of segments of contiguous markers [5, 9, 19]. We refer to insertions and deletions collectively as *indels*. For example, the deletion of segment `5262` from linear chromosome `12352624` results in `1234`. Indels have two restrictions: (i) only markers that have positive $\Delta\Phi$ can be deleted; and (ii) only markers that have negative $\Delta\Phi$ can be inserted.

In this paper, we are interested in computing the *DCJ-indel distance* between two genomes A and B, that is denoted by $d_{DCJ}^{id}(A, B)$ and corresponds to the minimum number of DCJs and indels required to sort A into B. We separate the instances of the problem in three types:

1. *Singular genomes*: the genomes contain no duplicate markers, that is, each common marker[1] is singular in each genome. Formally, we have that, for each $m \in \mathcal{G}$, $\Phi_A(m) = \Phi_B(m) = 1$. The distance between singular genomes can be easily computed in linear time [2, 5, 9].
2. *Balanced genomes*: the genomes contain no exclusive markers, but can have duplicates, and the number of duplicates in each genome is the same. Formally, we have $\mathcal{U} = \mathcal{G}$ and, for each $m \in \mathcal{U}$, $\Phi_A(m) = \Phi_B(m)$. Computing the distance for this set of instances is NP-hard, and an ILP formulation was given in [17].
3. *Natural genomes*: these genomes can have exclusive markers and duplicates, with no restrictions on the number of copies. Since these are generalizations of balanced genomes, computing the distance for this set of instances is also NP-hard. In the present work we present an efficient ILP formulation for computing the distance in this case.

2.2 DCJ-indel distance of singular genomes

First we recall the problem when common duplicates do not occur, that is, when we have singular genomes. We will summarize the linear time approach to compute the DCJ-indel distance in this case that was presented in [5], already adapted to the notation required for presenting the new results of this paper.

Relational diagram. For computing a genomic distance it is useful to represent the relation between two genomes in some graph structure [2,4,5,10,11]. Here we adopt a variation of this structure, defined as follows. For each marker m, denote its two extremities by m^t (tail) and m^h (head). Given two singular genomes A and B, the *relational diagram* $R(A, B)$ has a set of vertices $V = V(A) \cup V(B)$, where $V(A)$ has a vertex for each extremity of each marker of genome A and $V(B)$ has a vertex for each extremity of each marker of genome B. Due to the 1-to-1 correspondence between the vertices of $R(A, B)$ and the occurrences of marker extremities in A and B, we can identify each extremity with its corresponding vertex. It is convenient to represent vertices in $V(A)$ in an upper line,

[1] The exclusive markers are not restricted to be singular, because it is mathematically trivial to transform them into singular markers when they occur in multiple copies.

respecting the order in which they appear in each chromosome of A, and the vertices in $V(B)$ in a lower line, respecting the order in which they appear in each chromosome of B.

If the marker extremities γ_1 and γ_2 are adjacent in a chromosome of A, we have an *adjacency edge* connecting them. Similarly, if the marker extremities γ_1' and γ_2' are adjacent in a chromosome of B, we have an adjacency edge connecting them. Marker extremities located at chromosome ends are called *telomeres* and are not connected to any adjacency edge. In contrast, each extremity that is not a telomere is connected to exactly one adjacency edge. Denote by E_{adj}^A and by E_{adj}^B the adjacency edges in A and in B, respectively. In addition, for each common marker $m \in \mathcal{G}$, we have two *extremity edges*, one connecting the vertex m^h from $V(A)$ to the vertex m^h from $V(B)$ and the other connecting the vertex m^t from $V(A)$ to the vertex m^t from $V(B)$. Denote by E_γ the set of extremity edges. Finally, for each occurrence of an exclusive marker in $\mathcal{U}\backslash\mathcal{G}$, we have an *indel edge* connecting the vertices representing its two extremities. Denote by E_{id}^A and by E_{id}^B the indel edges in A and in B. Each vertex is then connected either to an extremity edge or to an indel edge.

All vertices have degree one or two, therefore $R(A, B)$ is a simple collection of cycles and paths. A path that has one endpoint in genome A and the other in genome B is called an *AB-path*. In the same way, both endpoints of an *AA-path* are in A and both endpoints of a *BB-path* are in B. A cycle contains either zero or an even number of extremity edges. When a cycle has at least two extremity edges, it is called an *AB-cycle*. Moreover, a path (respectively cycle) of $R(A, B)$ composed exclusively of indel and adjacency edges in one of the two genomes corresponds to a whole linear (respectively circular) chromosome and is called a *linear* (respectively *circular*) *singleton* in that genome. Actually, linear singletons are particular cases of *AA*-paths or *BB*-paths. An example of a relational diagram is given in Fig. 1.

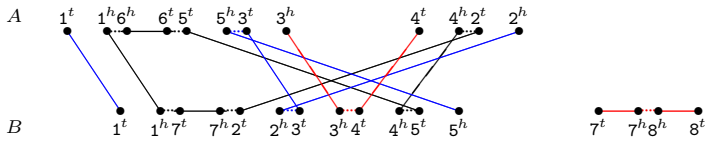

Fig. 1. For genomes $A = \{1\overline{6}53, 42\}$ and $B = \{172345, 7\overline{8}\}$, the relational diagram contains one cycle, two *AB*-paths (represented in blue), one *AA*-path and one *BB*-path (both represented in red). Short dotted horizontal edges are adjacency edges, long horizontal edges are indel edges, top-down edges are extremity edges.

The numbers of telomeres and of *AB*-paths in $R(A, B)$ are even. The *DCJ-cost* [5] of a DCJ operation ρ, denoted by $\|\rho\|$, is defined as follows. If it either increases the number of *AB*-cycles by one, or the number of *AB*-paths by two, ρ is *optimal* and has $\|\rho\| = 0$. If it does not affect the number of *AB*-cycles and *AB*-paths in the diagram, ρ is *neutral* and has $\|\rho\| = 1$. If it either decreases the

number of AB-cycles by one, or the number of AB-paths by two, ρ is *counter-optimal* and has $\|\rho\| = 2$.

Runs and indel-potential. The approach that uses DCJ operations to group exclusive markers for minimizing indels depends on the following concepts.

Given two genomes A and B and a component C of $R(A, B)$, a *run* [5] is a maximal subpath of C, in which the first and the last edges are indel edges, and all indel edges belong to the same genome. It can be an $\mathcal{A}$-run when its indel edges are in genome A, or a $\mathcal{B}$-run when its indel edges are in genome B. We denote by $\Lambda(C)$ the number of runs in component C. If $\Lambda(C) \geq 1$ the component C is said to be *indel-enclosing*, otherwise $\Lambda(C) = 0$ and C is said to be *indel-free*.

While sorting components separately with optimal DCJs only, runs can be *merged* (when two runs become a single one), and also *accumulated* together (when all its indel edges alternate with adjacency edges only and the run can be inserted or deleted at once) [5]. The *indel-potential* of a component C, denoted by $\lambda(C)$, is the minimum number of indels derived from C after this process and can be directly computed from $\Lambda(C)$:

$$\lambda(C) = \begin{cases} 0\,, & \text{if } \Lambda(C) = 0 \quad (C \text{ is indel-free}); \\ \left\lceil \frac{\Lambda(C)+1}{2} \right\rceil\,, & \text{if } \Lambda(C) \geq 1 \quad (C \text{ is indel-enclosing}). \end{cases}$$

Let λ_0 and λ_1 be, respectively, the sum of the indel-potentials for the components of the relational diagram before and after a DCJ ρ. The *indel-cost* of ρ is then $\Delta\lambda(\rho) = \lambda_1 - \lambda_0$, and the DCJ-indel cost of ρ is defined as $\Delta d(\rho) = \|\rho\| + \Delta\lambda(\rho)$. While sorting components separately, it has been shown that by using neutral or counter-optimal DCJs one can never achieve $\Delta d < 0$ [5]. This gives the following result:

Lemma 1 (from [2, 5]). *Given two singular genomes A and B, whose relational diagram $R(A, B)$ has c AB-cycles and i AB-paths, we have*

$$d_{DCJ}^{id}(A, B) \;\leq\; |\mathcal{G}| - c - \frac{i}{2} \;+ \sum_{C \in R(A,B)} \lambda(C)\,.$$

Distance of circular genomes. For singular circular genomes, the graph $R(A, B)$ is composed of cycles only. In this case the upper bound given by Lemma 1 is tight and leads to a simplified formula [5]:

$$d_{DCJ}^{id}(A, B) \;=\; |\mathcal{G}| - c \;+ \sum_{C \in R(A,B)} \lambda(C)\,.$$

Recombinations and linear genomes. For singular linear genomes, the upper bound given by Lemma 1 is achieved when the components of $R(A, B)$ are sorted separately. However, there are optimal or neutral DCJ operations, called *recombinations*, that act on two paths and have $\Delta d < 0$. Such path recombinations

are said to be *deducting*. The total number of types of deducting recombinations is relatively small. By exhaustively exploring the space of recombination types, it is possible to identify groups of chained recombinations (listed in Table 3 of the extended version of this manuscript [3]), so that the sources of each group are the original paths of the graph. In other words, a path that is a resultant of a group is never a source of another group. This results in a greedy approach (detailed in [3,5]) that optimally finds the value $\delta \geq 0$ to be deducted.

Theorem 1 (adapted from [5]). *Given two singular linear genomes A and B, whose relational diagram $R(A,B)$ has c AB-cycles and i AB-paths, and letting δ be the value obtained by maximizing deductions of path recombinations, we have*

$$d_{DCJ}^{id}(A,B) \;=\; |\mathcal{G}| - c - \frac{i}{2} \;+ \sum_{C \in R(A,B)} \lambda(C) - \delta\,.$$

3 DCJ-indel distance of natural genomes

In this work we are interested in comparing two natural genomes A and B. First we note that it is possible to transform A and B into *matched* singular genomes $A^\ddagger$ and $B^\ddagger$ as follows. For each common marker $m \in \mathcal{G}$, if $\Phi_A \leq \Phi_B$, we should determine which occurrence of m in B matches each occurrence of m in A, or if $\Phi_B < \Phi_A$, which occurrence of m in A matches each occurrence of m in B. The matched occurrences receive the same identifier (for example, by adding the same *index*) in $A^\ddagger$ and in $B^\ddagger$. Examples are given in Fig. 2 (top and center). Observe that, after this procedure, the number of common markers between any pair of matched genomes $A^\ddagger$ and $B^\ddagger$ is

$$n_* = \sum_{m \in \mathcal{G}} \min\{\Phi_A(m), \Phi_B(m)\}\,.$$

Let $\mathbb{M}$ be the set of all possible pairs of matched singular genomes obtained from natural genomes A and B. The DCJ-indel distance of A and B is then defined as

$$d_{DCJ}^{id}(A,B) = \min_{(A^\ddagger, B^\ddagger) \in \mathbb{M}} \{d_{DCJ}^{id}(A^\ddagger, B^\ddagger)\}\,.$$

3.1 Multi-relational diagram

While the original relational diagram clearly depends on the singularity of common markers, when they appear in multiple copies we can obtain a data structure that integrates the properties of all possible relational diagrams of matched genomes. The *multi-relational diagram* $MR(A,B)$ of two natural genomes A and B also has a set $V(A)$ with a vertex for each of the two extremities of each marker occurrence of genome A and a set $V(B)$ with a vertex for each of the two extremities of each marker occurrence of genome B.

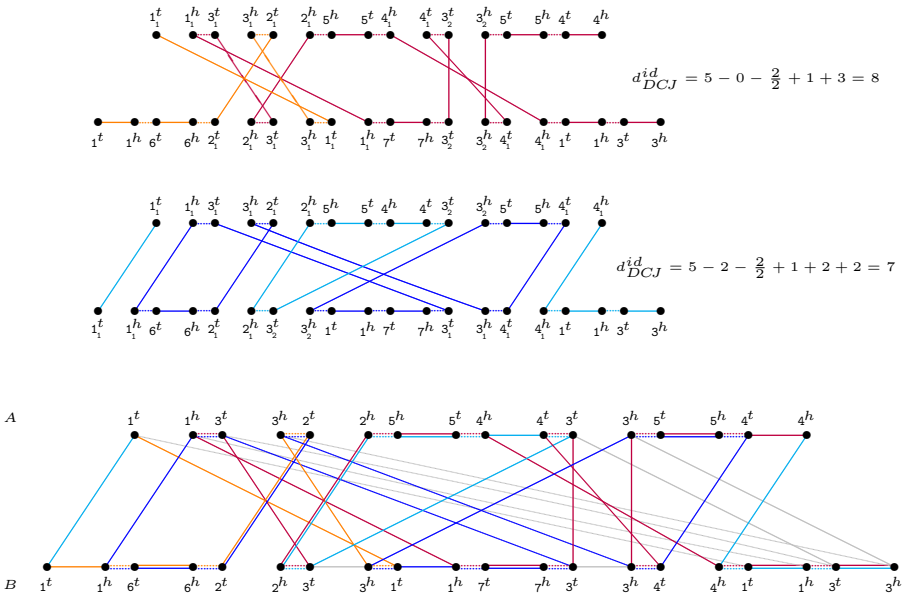

Fig. 2. Natural genomes $A = \texttt{132}\overline{\texttt{5}}\overline{\texttt{4}}\texttt{354}$ and $B = \texttt{1623173413}$ can give rise to many distinct pairs of matched singular genomes. The relational diagrams of two of these pairs are represented here, in the top and center. In the bottom we show the multi-relational diagram $MR(A, B)$. The decomposition that gives the diagram in the top is represented in red/orange. Similarly, the decomposition that gives the diagram in the center is represented in blue/cyan. Edges that are in both decompositions have two colors.

Again, sets E^A_{adj} and E^B_{adj} contain adjacency edges connecting adjacent extremities of markers in A and in B. But here the set E_γ contains, for each marker $m \in \mathcal{G}$, an extremity edge connecting each vertex in $V(A)$ that represents an occurrence of m^t to each vertex in $V(B)$ that represents an occurrence of m^t, and an extremity edge connecting each vertex in $V(A)$ that represents an occurrence of m^h to each vertex in $V(B)$ that represents an occurrence of m^h. Furthermore, for each marker $m \in \mathcal{U}$ with $\Phi_A(m) > \Phi_B(m)$, the set E^A_{id} contains one indel edge connecting the vertices representing the two extremities of the same occurrence of m in A. Similarly, for each marker $m' \in \mathcal{U}$ with $\Phi_B(m') > \Phi_A(m')$, the set E^B_{id} contains one indel edge connecting the vertices representing the two extremities of the same occurrence of m' in B. An example of a multi-relational diagram is given in Fig. 2 (bottom).

Consistent decompositions. Note that if A and B are singular genomes, $MR(A, B)$ reduces to the ordinary $R(A, B)$. On the other hand, in the presence of duplicate common markers, $MR(A, B)$ may contain vertices of degree larger than two. A *decomposition* is a collection of vertex-disjoint *components*, that can be cycles and/or paths, covering all vertices of $MR(A, B)$. There can be multiple

ways of selecting a decomposition, and we need to find one that allows to match occurrences of a marker in genome A with occurrences of the same marker in genome B.

Let $m_{(A)}$ and $m_{(B)}$ be, respectively, occurrences of the same marker m in genomes A and B. The extremity edge that connects $m^h_{(A)}$ to $m^h_{(B)}$ and the extremity edge that connects $m^t_{(A)}$ to $m^t_{(B)}$ are called *siblings*. A set $E_D \subseteq E_\gamma$ is a *sibling-set* if it is exclusively composed of pairs of siblings and does not contain any pair of incident edges. Thus, a *maximal* sibling-set E_D corresponds to a maximal matching of occurrences of common markers in both genomes.

The set of edges D *induced* by a maximal sibling-set E_D is said to be a *consistent decomposition* of $MR(A, B)$ and can be obtained as follows. In the beginning, D is the union of E_D with the sets of adjacency edges E^A_{adj} and E^B_{adj}. Then, for each indel edge e, if its two endpoints have degree one or zero in D, then e is added to D. Note that the consistent decomposition D covers all vertices of $MR(A, B)$ and is composed of cycles and paths, allowing us to compute the value

$$d^{id}_{DCJ}(D) = n_* - c_D - \frac{i_D}{2} + \sum_{C \in D} \lambda(C) - \delta_D \,,$$

where c_D and i_D are the numbers of AB-cycles and AB-paths in D, respectively, and δ_D is the optimal deduction of recombinations of paths from D. Since n_* is constant for any consistent decomposition, we can separate the part of the formula that depends on D, called *weight* of D:

$$w(D) = c_D + \frac{i_D}{2} - \sum_{C \in D} \lambda(C) + \delta_D \,.$$

Theorem 2. *Given two natural genomes A and B, the DCJ-indel distance of A and B can be computed by the following equation:*

$$d^{id}_{DCJ}(A, B) = \min_{D \in \mathbb{D}} \{d^{id}_{DCJ}(D)\} = n_* - \max_{D \in \mathbb{D}} \{w(D)\} \,,$$

where $\mathbb{D}$ is the set of all consistent decompositions of $MR(A, B)$.

Proof. In the extended version of this manuscript.

A consistent decomposition D such that $d^{id}_{DCJ}(D) = d^{id}_{DCJ}(A, B)$ is said to be *optimal*. Computing the DCJ-indel distance between two natural genomes A and B, or, equivalently, finding an optimal consistent decomposition of $MR(A, B)$ is an NP-hard problem. In Section 4 we will describe an efficient ILP formulation to solve it. Before that, we need to introduce a transformation of $MR(A, B)$ that is necessary for our ILP.

3.2 Capping

The ends of linear chromosomes produce some difficulties for the decomposition. Fortunately there is an elegant technique to overcome this problem, called

capping [11]. It consists of modifying the genomes by adding *artificial* singular common markers, also called *caps*, that circularize all linear chromosomes, so that their relational diagram is composed of cycles only, but, if the capping is optimal, the genomic distance is preserved.

Singular genomes. An optimal capping that transforms singular genomes A and B into singular circular genomes $A_\circ$ and $B_\circ$ can be obtained after greedily identifying the recombination groups following a top-down order of Table 3 of the extended version of this manuscript [3]. The optimal Δd for each recombination group is achieved by linking the groups as indicated in Table 5 of the extended version, where we also prove the following theorem.

Theorem 3. *Let κ_A and κ_B be, respectively, the total numbers of linear chromosomes in singular genomes A and B. We can obtain an optimal capping of A and B with exactly*

$$p_* = \max\{\kappa_A, \kappa_B\}$$

caps and $a_* = |\kappa_A - \kappa_B|$ artificial adjacencies between caps.

Capped multi-relational diagram. We can transform $MR(A, B)$ into the *capped multi-relational diagram* $MR_\circ(A, B)$ as follows. First we need to create $4p_*$ new vertices, named $\circ_A^1, \circ_A^2, \ldots, \circ_A^{2p_*}$ and $\circ_B^1, \circ_B^2, \ldots, \circ_B^{2p_*}$, each one representing a *cap extremity*. Each of the $2\kappa_A$ telomeres of A is connected by an adjacency edge to a distinct cap extremity among $\circ_A^1, \circ_A^2, \ldots, \circ_A^{2\kappa_A}$. Similarly, each of the $2\kappa_B$ telomeres of B is connected by an adjacency edge to a distinct cap extremity among $\circ_B^1, \circ_B^2, \ldots, \circ_B^{2\kappa_B}$. Moreover, if $\kappa_A < \kappa_B$, for $i = 2\kappa_A+1, 2\kappa_A+3, \ldots, 2\kappa_B-1$, connect $\circ_A^i$ to $\circ_A^{i+1}$ by an *artificial adjacency edge*. Otherwise, if $\kappa_B < \kappa_A$, for $j = 2\kappa_B + 1, 2\kappa_B + 3, \ldots, 2\kappa_A - 1$, connect $\circ_B^j$ to $\circ_B^{j+1}$ by an artificial adjacency edge. All these new adjacency edges and artificial adjacency edges are added to E_{adj}^A and E_{adj}^B, respectively.

We also connect each $\circ_A^i$, $1 \leq i \leq 2p_*$, by a *cap extremity edge* to each $\circ_B^j$, $1 \leq j \leq 2p_*$, and denote by $E_\circ$ the set of cap extremity edges. A set $E'_D \subseteq E_\circ$ is a *capping-set* if it does not contain any pair of incident edges. A consistent decomposition D of $MR_\circ(A, B)$ is induced by a maximal sibling-set $E_D \subseteq E_\gamma$ and a maximal capping-set $E'_D \subseteq E_\circ$ and is composed of vertex disjoint cycles covering all vertices of $MR_\circ(A, B)$. We then have $d_{DCJ}^{id}(D) = n_* + p_* - w(D)$, where the weight of D can be computed by the simpler formula

$$w(D) = c_D - \sum_{C \in D} \lambda(C)\,.$$

Finally, let $\mathbb{D}_\circ$ be the set of all consistent decompositions of $MR_\circ(A, B)$. Then

$$d_{DCJ}^{id}(A, B) = n_* + p_* - \max_{D \in \mathbb{D}_\circ}\{w(D)\}\,.$$

Note that the $2p_*$ cap extremities added to each genome correspond to p_* implicit caps. Furthermore, the number of artificial adjacency edges added to

the genome with less linear chromosomes is $a_* = |\kappa_A - \kappa_B|$. Since each pair of matched singular genomes $(A^\ddagger, B^\ddagger) \in \mathbb{M}$ can be optimally capped with this number of caps and artificial adjacencies, it is clear that at least one optimal capping of each $(A^\ddagger, B^\ddagger)$ corresponds to a consistent decomposition $D \in \mathbb{D}_\circ$.

4 An algorithm to compute the DCJ-indel distance of natural genomes

An ILP formulation for computing the distance of two balanced genomes A and B was given by Shao *et al.* in [17]. In this section we describe an extension of that formulation for computing the DCJ-indel distance of natural genomes A and B, based on consistent cycle decompositions of $MR_\circ(A, B)$. The main difference is that here we need to address the challenge of computing the indel-potential $\lambda(C)$ for each cycle C of each decomposition. Note that a cycle C of $R(A, B)$ has either 0, or 1, or an even number of runs, therefore its indel-potential can be computed as follows:

$$\lambda(C) = \begin{cases} \Lambda(C)\,, & \text{if } \Lambda(C) \leq 1; \\ \frac{\Lambda(C)}{2} + 1\,, & \text{if } \Lambda(C) \geq 2. \end{cases}$$

The formula above can be redesigned to a simpler one, that is easier to implement in the ILP. First, let a *transition* in a cycle C be an indel-free segment of C that is between a run in one genome and a run in the other genome and denote by $\aleph(C)$ the number of transitions in C. Observe that, if C is indel-free, then obviously $\aleph(C) = 0$. If C has a single run, then we also have $\aleph(C) = 0$. On the other hand, if C has at least 2 runs, then $\aleph(C) = \Lambda(C)$. Our new formula is then split into a part that simply tests whether C is indel-enclosing and a part that depends on the number of transitions $\aleph(C)$.

Proposition 1. *Given the function $r(C)$ defined as $r(C) = 1$ if $\Lambda(C) \geq 1$, otherwise $r(C) = 0$, the indel-potential $\lambda(C)$ can be computed from the number of transitions $\aleph(C)$ with the formula*

$$\lambda(C) = \frac{\aleph(C)}{2} + r(C)\,.$$

Note that $\sum_{C \in D} r(C) = c_D^r + s_D$, where c_D^r and s_D are the number of indel-enclosing *AB*-cycles and the number of circular singletons in D, respectively. Now, we need to find a consistent decomposition D of $MR_\circ(A, B)$ maximizing its weight

$$w(D) = c_D - \sum_{C \in D} \lambda(C) = c_D - \left(c_D^r + s_D + \sum_{C \in D} \frac{\aleph(C)}{2} \right) = c_D^{\tilde{r}} - s_D - \sum_{C \in D} \frac{\aleph(C)}{2}\,,$$

where $c_D^{\tilde{r}} = c_D - c_D^r$ is the number of indel-free *AB*-cycles in D.

4.1 ILP formulation

Our formulation (shown in Algorithm 1) searches for an optimal consistent cycle decomposition of $MR_\circ(A, B) = (V, E)$, where the set of edges E is the union of all disjoint sets of the distinct types of edges, $E = E_\gamma \cup E_\circ \cup E^A_{adj} \cup E^B_{adj} \cup E^A_{id} \cup E^B_{id}$.

In the first part we use the same strategy as Shao *et al.* [17]. A binary variable x_e (`D.01`) is introduced for every edge e, indicating whether e is part of the computed decomposition. Constraint `C.01` ensures that adjacency edges are in all decompositions, Constraint `C.02` ensures that each vertex of each decomposition has degree 2, and Constraint `C.03` ensures that an extremity edge is selected only together with its sibling. Counting the number of cycles in each decomposition is achieved by assigning a unique identifier i to each vertex v_i that is then used to label each cycle with the numerically smallest identifier of any contained vertex (see Constraint `C.04`, Domain `D.02`). A vertex v_i is then marked by variable z_i (`D.03`) as representative of a cycle if its cycle label y_i is equal to i (`C.06`). However, unlike Shao *et al.*, we permit each variable y_i to take on value 0 which, by Constraint `C.05`, will be enforced whenever the corresponding cycle is indel-enclosing. Since the smallest label of any vertex is 1 (cf. `D.02`), any cycle with label 0 will not be counted.

The second part is our extension for counting transitions. We introduce binary variables r_v (`D.04`) to label runs. To this end, Constraint `C.07` ensures that each vertex v is labeled 0 if v is part of an $\mathcal{A}$-run and otherwise it is labeled 1 indicating its participation in a $\mathcal{B}$-run. Transitions between $\mathcal{A}$- and $\mathcal{B}$-runs in a cycle are then recorded by binary variable t_e (`D.05`). If a transition occurs between any neighboring pair of vertices $u, v \in V$ of a cycle, Constraint `C.08` causes transition variable $t_{\{u,v\}}$ to be set to 1. We avoid an excess of co-optimal solutions by canonizing the locations in which such transitions may take place. More specifically, Constraint `C.09` prohibits label changes in adjacencies not directly connected to an indel and Constraint `C.10` in edges other than adjacencies of genome A, resulting in all A-runs containing as few vertices as possible.

In the third part we add a new constraint and a new domain to our ILP, so that we can count the number of circular singletons. Let K be the circular chromosomes in both genomes and E^k_{id} be the set of indel edges of a circular chromosome $k \in K$. For each circular chromosome we introduce a decision variable s_k (`D.06`), that is 1 if k is a circular singleton and 0 otherwise. A circular chromosome is then a singleton if all its indel edges are set (see Constraint `C.11`).

The objective of our ILP is to maximize the weight of a consistent decompositon, that is equivalent to maximizing the number of indel-free cycles, counted by the sum over variables z_i, while simultaneously minimizing the number of transitions in indel-enclosing AB-cycles, calculated by half the sum over variables t_e, and the number of circular singletons, calculated by the sum over variables s_k.

Implementation. We implemented the construction of the ILP as a python application, available at `https://gitlab.ub.uni-bielefeld.de/gi/ding`.

Comparison to the approach by Lyubetsky *et al.* As mentioned in the Introduction, another ILP for the comparison of genomes with unequal content

Algorithm 1 ILP for the computation of the DCJ-indel distance of natural genomes

Objective:

$$\texttt{Maximize} \sum_{1 \leq i \leq |V|} z_i - \frac{1}{2} \sum_{e \in E} t_e - \sum_{k \in K} s_k$$

Constraints:

(C.01)	$x_e = 1$	$\forall\ e \in E^A_{adj} \cup E^B_{adj}$
(C.02)	$\sum_{\{u,v\} \in E} x_{\{u,v\}} = 2$	$\forall\ u \in V$
(C.03)	$x_e = x_d$	$\forall\ e, d \in E_\gamma$ such that e and d are siblings
(C.04)	$y_i \leq y_j + i(1 - x_{\{v_i,v_j\}})$	$\forall\ \{v_i, v_j\} \in E$,
(C.05)	$y_i \leq i(1 - x_{\{v_i,v_j\}})$	$\forall\ \{v_i, v_j\} \in E^A_{id} \cup E^B_{id}$
(C.06)	$i \cdot z_i \leq y_i$	$\forall\ 1 \leq i \leq |V|$
(C.07)	$r_v \leq 1 - x_{\{u,v\}}$	$\forall\ \{u, v\} \in E^A_{id}$,
	$r_{v'} \geq x_{\{u',v'\}}$	$\forall\ \{u', v'\} \in E^B_{id}$
(C.08)	$t_{\{u,v\}} \geq r_v - r_u - (1 - x_{\{u,v\}})$	$\forall\ \{u, v\} \in E$
(C.09)	$\sum_{\{v,w\} \in E^A_{id}} x_{\{v,w\}} - t_{\{u,v\}} \geq 0$	$\forall\ \{u, v\} \in E^A_{adj}$
(C.10)	$t_e = 0$	$\forall\ e \in E \setminus E^A_{adj}$
(C.11)	$\sum_{e \in E^k_{id}} x_e - |k| \leq s_k$	$\forall k \in K$

Domains:

(D.01)	$x_e \in \{0, 1\}$	$\forall\ e \in E$
(D.02)	$0 \leq y_i \leq i$	$\forall\ 1 \leq i \leq |V|$
(D.03)	$z_i \in \{0, 1\}$	$\forall\ 1 \leq i \leq |V|$
(D.04)	$r_v \in \{0, 1\}$	$\forall\ v \in V$
(D.05)	$t_e \in \{0, 1\}$	$\forall\ e \in E$
(D.06)	$s_k \in \{0, 1\}$	$\forall\ k \in K$

and paralogs was presented by Lyubetsky *et al.* [13]. In order to compare our method to theirs, we ran our ILP using CPLEX on a single thread with the two small artificial examples given in that paper on page 8. The results show that both ILPs give the same correct distances and our ILP runs much faster, as shown in Table 1.

Table 1. Comparison of running times and memory usage to the ILP in [13].

dataset	#markers	#marker occurrences	running time as reported in [13]	our running time	our peak memory
Example 1	5/5	9/9	"about 1.5h"	.16s	13200kb
Example 2	10/10	11/11	"about 3h"	.05s	13960kb

4.2 Performance benchmark

For benchmarking purposes, we used gurobi 9.0 as solver. In all our experiments, we ran gurobi on a single thread. Details on how the simulated data is generated are given in the extended version of this manuscript.

In order to evaluate the impact of the number of duplicate occurrences on the running time, we keep the number of simulated DCJ events fixed to 10,000 and vary parameters that affect the number of duplicate occurrences.

Our ILP solves the decomposition problem efficiently for real-sized genomes under small to moderate numbers of duplicate occurrences: the solving times for genome pairs with less than 10,000 duplicate occurrences ($\sim 50\%$ of the genome size) shown in Figure 3 are with few exceptions below 5 minutes and exhibit a linear increase, but the solving time is expected to boost dramatically with higher numbers of duplicate occurrences. To further exploit the conditions under which the ILP is no longer solvable with reasonable compute resources we continued the experiment with even higher amounts of duplicate occurrences and instructed gurobi to terminate within 1 hour of computation. We then partitioned the simulated data set into 8 intervals of length 500 according to the observed number of duplicate occurrences. For each interval, we determined the average as well as the maximal multiplicity of any duplicate marker and examined the average *optimality gap*, i.e., the difference in percentage between the best primal and the best dual solution computed within the time limit. The results are shown in Table 2 and emphasize the impact of duplicate occurrences in the solving time: below 14,000 duplicate occurrences, the optimality gap remains small and sometimes even the exact solution is computed, whereas above that threshold the gap widens very quickly.

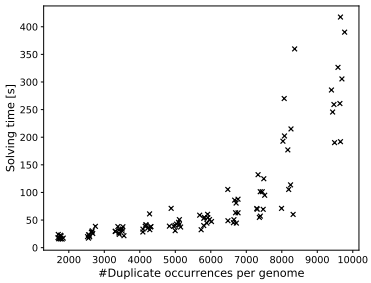

Fig. 3. Solving times for genomes with varying number of duplicate occurrences, totaling 20,000 marker occurrences per genome.

#Dupl. occurrences	avg. mult. of dupl. markers	max. multiplicity	avg. opt. gap (%)
11500..11999	2.206	8	0.000
12000..12499	2.219	8	0.031
12500..12999	2.217	7	0.025
13000..13499	2.233	9	0.108
13500..13999	2.247	8	0.812
14000..14499	2.260	8	1.177
14500..14999	2.274	8	81.865
15000..15499	2.276	9	33.102

Table 2. Average optimality gap for simulated genome pairs grouped by number of duplicate occurrences after 1h of running time.

Additionally, we ran three experiments, in each varying one of the following parameters while keeping the others fixed: (i) genome size, (ii) number of simulated DCJs and indels, and (iii) number of linear chromosomes. The results, given in the extended version of this manuscript, indicate that the number of linear chromosomes also has a considerable impact in the running time, while the other two have minor effect.

4.3 Real data analysis

Recently, the first three high-resolution haplotype-resolved human genomes have been published [8]. The study reports an average number of 156 inversions per genome, of which 121 are characterized as simple and 35 as copy-variable inversions. Here, we demonstrate the applicability of our approach to the study of

real data by calculating the DCJ-indel distance between one of these haplotypes (`HG00514.h0`) and the human reference sequence (`GRCh38`). After the construction of a genomic marker set, we represented each chromosome of both genomes as marker sequence, with the largest chromosome (chr. 1) comprising close to 18,000 markers. We then ran our ILP for the computation of the DCJ-indel distance on each pair of chromosomes independently. We were able to obtain exact solutions for 17 chromosomes within few minutes and two more within a few days. However, the remaining four comparisons did not complete within a timelimit of 3 days. Still, after that time, their optimality gaps were below 0.1%. The calculated DCJ-indel distances ranged between 1.3% and 7.7% of the length of the marker sequences, with the number of runs accounting for at least 48.7% of the distance. Further details on the data set, the construction of the genomic markers, and the calculated DCJ-indel distances are described in Appendix A of the extended version of this paper.

5 Conclusion

By extending the DCJ-indel model to allow for duplicate markers, we introduced a rearrangement model that is capable of handling *natural genomes*, i.e., genomes that contain shared, individual, and duplicated markers. In other words, under this model genomes require no further processing nor manipulation once genomic markers and their homologies are inferred. The DCJ-indel distance of natural genomes being NP-hard, we presented a fast method for its calculation in form of an integer linear program. Our program is capable of handling real-sized genomes, as evidenced in simulation and real data experiments. It can be applied universally in comparative genomics and enables uncompromising analyses of genome rearrangements. We hope that such analyses will provide further insights into the underlying mutational mechanisms. Conversely, we expect the here presented model to be extended and specialized in future to reflect the insights gained by these analyses.

References

1. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. **13**(1), 19–53 (2009), (A preliminary version appeared in Proc. of WALCOM 2008.)
2. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proceedings of the 6th International Conference on Algorithms in Bioinformatics (WABI 2006). LNBI, vol. 4175, pp. 163–173. Springer Verlag (2006)
3. Bohnenkämper, L., Braga, M.D.V., Doerr, D., Stoye, J.: Computing the rearrangement distance of natural genomes. arXiv 2001.02139 (2020)
4. Braga, M.D.V.: An overview of genomic distances modeled with indels. In: Proceedings of the Conference on Computability in Europe (CiE 2013). LNCS, vol. 7921, pp. 22–31. Springer Verlag (2013)
5. Braga, M.D.V., Willing, E., Stoye, J.: Double cut and join with insertions and deletions. Journal of Computational Biology **18**(9), 1167–1184 (2011), (A preliminary version appeared in Proc. of WABI 2010.)

6. Bryant, D.: The complexity of calculating exemplar distances. In: Sankoff, D., Nadeau, J.H. (eds.) Comparative Genomics, pp. 207–211. Kluwer Academic Publishers (2000)
7. Bulteau, L., Jiang, M.: Inapproximability of (1,2)-exemplar distance. IEEE/ACM Trans. Comput. Biol. Bioinf. **10**(6), 1384–1390 (2013), (A preliminary version appeared in Proc. of ISBRA 2012.)
8. Chaisson, M.J.P. *et al.*: Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nature Communications **10**(1), 1–16 (Apr 2019)
9. Compeau, P.E.C.: DCJ-indel sorting revisited. Algorithms for Molecular Biology **8**, 6 (2013), (A preliminary version appeared in Proc. of WABI 2012.)
10. Friedberg, R., Darling, A.E., Yancopoulos, S.: Genome rearrangement by the double cut and join operation. In: Keith, J.M. (ed.) Bioinformatics, Volume I: Data, Sequence Analysis, and Evolution, Methods in Molecular Biology, vol. 452, pp. 385–416. Humana Press (2008)
11. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proceedings of the 36th Annual Symposium of the Foundations of Computer Science (FOCS 1995). pp. 581–592. IEEE Press (1995)
12. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM **46**(1), 1–27 (1999), (A preliminary version appeared in Proc. of STOC 1995.)
13. Lyubetsky, V., Gershgorin, R., Gorbunov, K.: Chromosome structures: reduction of certain problems with unequal gene contemnt and gene paralogs to integer linear programming. BMC Bioinformatics **18**, 537 (2017)
14. Martinez, F.V., Feijão, P., Braga, M.D.V., Stoye, J.: On the family-free DCJ distance and similarity. Algorithms for Molecular Biology **10**, 13 (2015), (A preliminary version appeared in Proc. of WABI 2014.)
15. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching, CPM 1992. LNCS, vol. 644, pp. 121–135. Springer Verlag, Berlin (1992)
16. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics **15**(11), 909–917 (1999)
17. Shao, M., Lin, Y., Moret, B.M.E.: An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. J. Comp. Biol. **22**(5), 425–435 (2015), (A preliminary version appeared in Proc. of RECOMB 2014.)
18. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)
19. Yancopoulos, S., Friedberg, R.: DCJ path formulation for genome transformations which include insertions, deletions, and duplications. Journal of Computational Biology **16**(10), 1311–1338 (2009), (A preliminary version appeared in Proc. of RECOMB-CG 2008.)