

Rearrangements in genomes with centromeres

part I: translocations

Michal Ozery-Flato and Ron Shamir

School of Computer Science, Tel-Aviv University, Tel Aviv 69978, Israel
{ozery, rshamir}@post.tau.ac.il

Abstract. A *centromere* is a special region in the chromosome that plays a vital role during cell division. Every new chromosome created by a genome rearrangement event must have a centromere in order to survive. This constraint has been ignored in the computational modeling and analysis of genome rearrangements to date. Unlike genes, the different centromeres are indistinguishable, they have no orientation, and only their location is known. A prevalent rearrangement event in the evolution of multi-chromosomal species is translocation, i.e., the exchange of tails between two chromosomes. A translocation may create a chromosome with no centromere in it. In this paper we study for the first time centromeres-aware genome rearrangements. We present a polynomial time algorithm for computing a shortest sequence of translocations transforming one genome into the other, where all of the intermediate chromosomes must contain centromeres. We view this as a first step towards analysis of more general genome rearrangement models that take centromeres into consideration.

1 Introduction

Genomes of related species tend to have similar genes that are, however, ordered differently. The distinct orderings of the genes are the result of genome rearrangements. Inferring the sequence of genome rearrangements that took place during the course of evolution is an important question in comparative genomics. The genomes of higher organisms, such as plants and animals, are partitioned into continuous units called *chromosomes*. Every chromosome contains a special region called a *centromere*, which plays a vital role during cell division. An *acentric* chromosome, i.e. one that lacks a centromere, is likely to be lost during subsequent cell divisions [9]. Thus a rearrangement scenario that preserves a centromere in each chromosome is more biologically realistic than a one that does not. The computational studies on genome rearrangements to date have ignored the existence and role of centromeres. Hence, the rearrangement scenarios for multi-chromosomal genomes produced by current algorithms may include genomes with non-viable chromosomes. In this study we begin to address the centromeres in the computational analysis of genome rearrangements.

Since sequencing a centromere is almost impossible due to the repeated sequences it contains, the only information we have on a centromere is its location in the genome. Therefore, in the model we define, centromeres appear as anonymous and orientationless elements. We say that a genome is *legal* if each of its chromosomes contains a single centromere. A *legal* rearrangement operation results in a legal genome (Fig. 1).

The *legal rearrangement sorting problem* is defined as follows: given two legal genomes A and B , find a shortest sequence of legal rearrangement operations that transforms A into B . The length of this sequence is the *legal distance* between A and B .

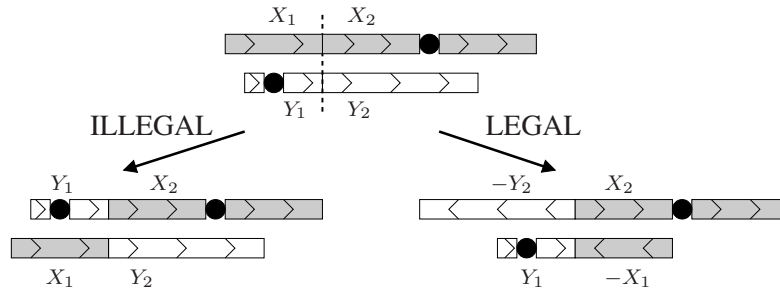


Fig. 1: An example of legal and illegal translocations for a certain cut of two chromosomes. The black circles denote the location of the centromeres. The broken line indicates the positions where the two chromosomes were cut.

A *reciprocal translocation* is a rearrangement in which two chromosomes exchange non-empty ends. A reciprocal translocation results in an illegal genome if exactly one of the exchanged ends contains a centromere. In this paper we focus on the problem of legal sorting by reciprocal translocations, abbreviated hereafter LSRT. This problem is a refinement of the "sorting by reciprocal translocations" problem (hereafter SRT), which ignores centromeres. SRT was studied in [3, 2, 5, 6] and is solvable in polynomial time. Clearly a solution to SRT may not be a solution to LSRT since 50% of the possible reciprocal translocations are illegal (Fig. 1). Indeed, in many cases more rearrangements are needed in order to legally sort a genome.

In this study we present a polynomial time algorithm for LSRT. The basic idea is to transform LSRT into SRT, by replacing pairs of centromeres in the two genomes by new unique oriented elements. Our algorithm is based on finding a mapping between the centromeres of the two given genomes such that the solution to the resulting SRT instance is minimum. We show that an optimal mapping can be found in polynomial time. To the best of our knowledge, this is the first rearrangement algorithm that considers centromeres. While a model that permits only reciprocal translocations is admittedly quite remote from the biological reality, we hope that the principles and structure revealed here will be instrumental for analyzing more realistic models in the future. One additional advantage of centromere-aware models is that they restrict drastically the allowed sequences of operations, and therefore are less likely to suffer from high multiplicity of optimal sequences.

The paper is organized as follows. Section 2 gives the necessary preliminaries. In Section 3 we model LSRT and present some elementary properties of it. Section 4 describes an exponential algorithm for LSRT, which searches for an optimal mapping between the centromeres of A and B , i. e., one that leads to a minimum SRT solution. In Section 5 we take a first step towards a polynomial time algorithm for LSRT by proving a bound that is at most two translocations away from the legal translocation distance. In Section 6 we present a theorem leading to a polynomial time algorithm for

computing the legal translocation distance and solving LSRT. For lack of space most proofs are omitted and the final polynomiality result is only sketched.

2 Preliminaries

This section provides the needed background for SRT. The definitions follow previous literature on translocations [3, 2, 5, 6]. In the model we consider, a *genome* is a set of chromosomes. A *chromosome* is a sequence of genes. A *gene* is identified by a positive integer. All genes in the genome are distinct. When it appears in a genome, a gene is assigned a sign of plus or minus. The following is an example of a genome with two chromosomes and six genes: $\{(1, -5), (-4, -3, -2, 6)\}$.

The *reverse* of a sequence of genes $I = (x_1, \dots, x_l)$ is $-I = (-x_l, \dots, -x_1)$. Two chromosomes, X and Y , are called *identical* if either $X = Y$ or $X = -Y$. Therefore, *flipping* chromosome X into $-X$ does not affect the chromosome it represents.

Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be two chromosomes, where X_1, X_2, Y_1, Y_2 are sequences of genes. A *translocation* cuts X into X_1 and X_2 and Y into Y_1 and Y_2 and exchanges segments between the chromosomes. It is called *reciprocal* if X_1, X_2, Y_1 and Y_2 are all non-empty. There are two types of translocations on X and Y . A *prefix-suffix* translocation switches X_1 with Y_2 :

$$(\underline{X}_1, X_2), (Y_1, \underline{Y}_2) \Rightarrow (-Y_2, X_2), (Y_1, -X_1).$$

A *prefix-prefix* translocation switches X_1 with Y_1 :

$$(\underline{X}_1, X_2), (\underline{Y}_1, Y_2) \Rightarrow (Y_1, X_2), (X_1, Y_2).$$

Note that we can mimic one type of translocation by a flip of one of the chromosomes followed by a translocation of the other type.

For a chromosome $X = (x_1, \dots, x_k)$ define $Tails(X) = \{x_1, -x_k\}$. Note that flipping X does not change $Tails(X)$. For a genome A define $Tails(A) = \bigcup_{X \in A} Tails(X)$. For example: $Tails(\{(1, -3, -2, 4, -7, 8), (6, 5)\}) = \{1, -8, 6, -5\}$. Two genomes A_1 and A_2 are *co-tailed* if $Tails(A_1) = Tails(A_2)$. In particular, two co-tailed genomes have the same number of chromosomes. Note that if A_2 was obtained from A_1 by performing a reciprocal translocation then $Tails(A_2) = Tails(A_1)$. Therefore, SRT is solvable only for genomes that are co-tailed. For the rest of this paper the word ‘‘translocation’’ refers to a reciprocal translocation and we assume that the given genomes, A and B , are co-tailed. Denote the set of tails of A and B by $Tails$.

2.1 The Cycle Graph

Let n and N be the number of genes and chromosomes in A (equivalently, B) respectively. We shall always assume that both A and B consist of the genes $\{1, \dots, n\}$. The *cycle graph* of A and B , denoted $G(A, B)$, is defined as follows. The set of vertices is $\bigcup_{i=1}^n \{i^0, i^1\}$. The vertices i^0 and i^1 are called the two *ends* of gene i (think of them as ends of a small arrow directed from i^0 to i^1). For every two genes, i and j , where j immediately follows i in some chromosome of A (respectively, B) add a black (respectively, grey) edge $(i, j) \equiv (out(i), in(j))$, where $out(i) = i^1$ if i has a positive sign in A (respectively, B) and otherwise $out(i) = i^0$, and $in(j) = j^0$ if j has a positive

sign in A (respectively, B) and otherwise $in(j) = j^1$. An example is given in Fig. 2(a). There are $n - N$ black edges and $n - N$ grey edges in $G(A, B)$. A grey edge (i, j) is *external* if the genes i and j belong to different chromosomes of A , otherwise it is *internal*. A cycle is *external* if it contains an external edge, otherwise it is *internal*.

Every vertex in $G(A, B)$ has degree 2 or 0, where vertices of degree 0 (isolated vertices) belong to *Tails*. Therefore, $G(A, B)$ is uniquely decomposed into cycles with alternating grey and black edges. An *adjacency* is a cycle with two edges. A *breakpoint* is a black edge that is not part of an adjacency.

2.2 The Overlap Graph with Chromosomes

A *signed permutation* $\pi = (\pi_1, \dots, \pi_n)$ is a permutation on the integers $\{1, \dots, n\}$, where a sign of plus or minus is assigned to each number. If A is a genome with the set of genes $\{1, \dots, n\}$ then any concatenation π_A of the chromosomes of A is a signed permutation of size n .

Place the vertices of $G(A, B)$ along a straight line according to their order in π_A . Now, every grey edge and every chromosome is associated with an interval of vertices in $G(A, B)$. Two intervals *overlap* if their intersection is not empty but none contains the other. The *overlap graph with chromosomes* of A and B w.r.t. π_A , denoted $OVCH(A, B, \pi_A)$, is defined as follows. The set of nodes is the set of grey edges and chromosomes in $G(A, B)$. Two nodes are connected if their corresponding intervals overlap. An example is given in Fig. 2(b). This graph is an extension of the overlap graph of a signed permutation defined in [4]. Let $OV(A, B, \pi_A)$ be the subgraph of $OVCH(A, B, \pi_A)$ induced by the set of nodes that correspond to grey edges (i.e. excluding the chromosomes' nodes). We shall use the word "component" for a connected component of $OV(A, B, \pi_A)$.

In order to prevent confusion, we will refer to nodes that correspond to chromosomes as "chromosomes" and reserve the word "vertex" for nodes that correspond to grey edges. A vertex is *external* (resp. *internal*) if it corresponds to an external (resp. internal) grey edge. Obviously a vertex is external iff it is connected to a chromosome. A component is *external* if it contains an external vertex, otherwise it is *internal*. A component is *trivial* if it is composed of one (internal) vertex. A trivial component corresponds to an adjacency. Note that the internal/external state of a vertex in $OVCH(A, B, \pi_A)$ does not depend on π_A . Therefore, the set of internal components in $OVCH(A, B, \pi_A)$ is independent of π_A . The *span* of a component M is the minimal interval of genes $I(M) = [i, j] \subset \pi_A$ that contains the interval of every vertex in M . Clearly, $I(M)$ is independent of π_A iff M is internal. The following lemma follows from A and B being co-tailed and [4, Corollary 2.2]:

Lemma 1 *Every internal component corresponds to the set of grey edges of a union of cycles in $G(A, B)$.*

The set of internal components can be computed in linear time using an algorithm in [1].

2.3 The Forest of Internal Components

(M_1, \dots, M_t) is a *chain* of components if $I(M_j)$ and $I(M_{j+1})$ overlap in exactly one gene for $j = 1, \dots, t - 1$. The *forest of internal components* [2], denoted $F(A, B)$, is

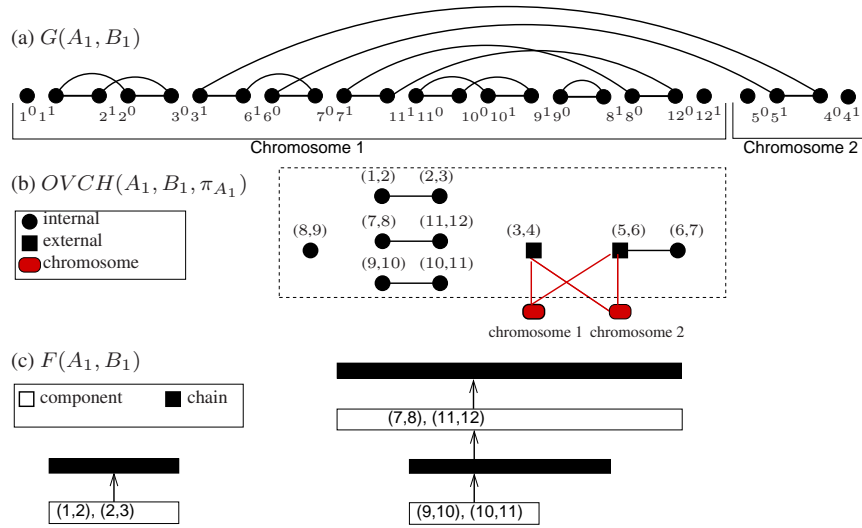


Fig. 2: Auxiliary graphs for $A_1 = \{(1, -2, 3, -6, 7, -11, 10, -9, -8, 12), (5, 4)\}$, $B_1 = \{(1, \dots, 4), (5, \dots, 12)\}$ ($\pi_{A_1} = (1, -2, 3, -6, 7, -11, 10, -9, -8, 12, 5, 4)$). (a) The cycle graph. Black edges are horizontal, grey edges are curved. (b) The overlap graph with chromosomes. The graph induced by the vertices within the dashed rectangle is $OV(A_1, B_1, \pi_{A_1})$. (c) The forest of internal components.

defined as follows. The vertices of $F(A, B)$ are (i) the non-trivial internal components and (ii) every maximal chain of internal components that contains at least one non-trivial component. Let M and C be two vertices in $F(A, B)$ where M corresponds to a component and C to a chain. $M \rightarrow C$ is an edge of $F(A, B)$ if $M \in C$. $C \rightarrow M$ is an edge of $F(A, B)$ if $I(C) \subset I(M)$ and $I(M)$ is minimal. See Fig. 2(c) for an example. We will refer to a component that is a leaf in $F(A, B)$ as simply a *leaf*.

2.4 The Reciprocal Translocation Distance

The *reciprocal translocation distance* between A and B is the length of a shortest sequence of reciprocal translocations that transforms A into B . Let $c(A, B)$ denote the number of cycles in $G(A, B)$. Let $|F(A, B)|$ and $l(A, B)$ denote the number of trees and leaves in $F(A, B)$ respectively. Obviously $|F(A, B)| \leq l(A, B)$. Define

$$\delta(A, B) \equiv \delta(F(A, B)) = \begin{cases} 2 & \text{if } |F(A, B)| = 1 \text{ and } l(A, B) \text{ is even} \\ 1 & \text{if } l(A, B) \text{ is odd} \\ 0 & \text{otherwise } (|F(A, B)| \neq 1 \text{ and } l(A, B) \text{ is even}) \end{cases}$$

Theorem 1 [2, 3] *The reciprocal translocation distance between A and B is $n - N - c(A, B) + l(A, B) + \delta(A, B)$*

Let Δc denote the change in the number of cycles after performing a translocation on A . Then $\Delta c \in \{-1, 0, 1\}$ [3]. A translocation is *proper* if $\Delta c = 1$, *improper* if $\Delta c = 0$ and *bad* if $\Delta c = -1$.

Corollary 1 *Every translocation in a shortest sequence of translocations transforming A into B is either proper or bad.*

Proof. An improper translocation cannot decrease the translocation distance since it does not affect any parameter in its formula. \square

3 Incorporating Centromeres into a Genome

We extend the model described above by adding the requirement that every genome is legal (i.e. every chromosome contains exactly one centromere). We denote the location of a centromere in a chromosome by the element \bullet . The element \bullet is unsigned and thus does not change under chromosome flips. The following is an example of a legal genome: $\{(1, 2, 3, \bullet, 4), (\bullet, 5, 6)\}$. The set of tails is defined for regular elements, thus $Tails(\bullet, 5, 6) = \{5, -6\}$. We assume that a cut of a chromosome does not split a centromere. Clearly, for every cut of two chromosomes one translocation is legal while the other is not (see Fig. 1).

3.1 A New Precondition

We present here a simple condition for the solvability of LSRT. If this condition is not satisfied then A cannot be transformed into B by legal translocations. For chromosome $X = (x_1, \dots, x_i, \bullet, x_{i+1}, \dots, x_k)$ define $Elements(X) = \{x_1, \dots, x_i, -x_{i+1}, \dots, -x_k\}$. Note that $Elements(X) = Elements(-X)$. For genome A we define $Elements(A) = \bigcup_{X \in A} Elements(X)$. For example:

$$Elements(\{(1, 2, \bullet, 3, 4), (\bullet, 5, 6)\}) = \{1, 2, -3, -4, -5, -6\}.$$

Observation 1 *Let A and B be two legal genomes. If A can be transformed into B by a sequence of legal translocations then $Elements(A) = Elements(B)$.*

We will see later that this condition is also sufficient. Thus, for the rest of this paper we assume that the input to LSRT is co-tailed genomes A and B satisfying $Elements(A) = Elements(B) = Elements$. The cycle graph of A and B , $G(A, B)$, ignores the \bullet elements.

3.2 On the Gap Between the Legal Distance and the “Old” Distance

Let $d(A, B)$ denote the legal translocation distance between A and B . Let $d_{old}(A, B)$ denote the translocation distance between A and B when the \bullet elements are ignored. Obviously $d(A, B) \geq d_{old}(A, B)$. Consider the genomes A_2 and B_2 in Fig. 3. It can be easily verified that $d_{old}(A_2, B_2) = 3$ and $d(A_2, B_2) = 4$. This example is easily extendable to two genomes A_{2k} and B_{2k} , with $2k$ chromosomes each, such that $d_{old}(A_{2k}, B_{2k}) = 3k$ and $d(A_{2k}, B_{2k}) = 4k$.

3.3 Telocentric Chromosomes

A chromosome is *telocentric* if its centromere is located at one of its endpoints. For example the chromosome $(\bullet, 5, 6)$ is telocentric.

Lemma 2 Let A and B be co-tailed genomes satisfying $Elements(A) = Elements(B)$. Then A and B have the same number of telocentric chromosomes. Moreover, the set of genes adjacent to the centromeres in the telocentric chromosomes is the same.

Let η denote the number of non-telocentric chromosomes in A and B . We shall show later how mapping between centromeres in non-telocentric chromosomes in A and B can help us to solve LSRT.

3.4 Pericentric and Paracentric Edges

A grey (respectively, black) edge in $G(A, B)$ is said to be *pericentric* if the two genes it connects flank a centromere in genome B (respectively, A). Otherwise it is called *paracentric*. See Fig. 3(a). For a gene i we define:

$$cent(i^0) = \begin{cases} -1 & \text{if } i \text{ has a positive sign in } Elements, \\ 1 & \text{otherwise.} \end{cases} \quad cent(i^1) = -cent(i^0)$$

In other words, the sign of the end closer to the centromere (in both A and B) is positive, and the sign of the remote end is negative. The legality precondition (Section 3.1) implies the following key property:

Lemma 3 Let (u, v) be an edge in $G(A, B)$. If (u, v) is pericentric then $cent(u) = cent(v) = 1$. Otherwise $cent(u)cent(v) = -1$.

3.5 Peri-Cycles

Let C be a cycle in $G(A, B)$. The *peri-cycle* of C , C^P , is defined as follows. The vertices of C^P are the pericentric edges in C . A vertex in C^P is colored grey (respectively, black) if the corresponding edge in C is grey (respectively, black). A path between two consecutive pericentric edges in C is translated to an edge between the two corresponding vertices in C^P . See Fig. 3. Note that if C contains no pericentric edges then its peri-cycle is a null cycle (i.e. a cycle with no vertices).

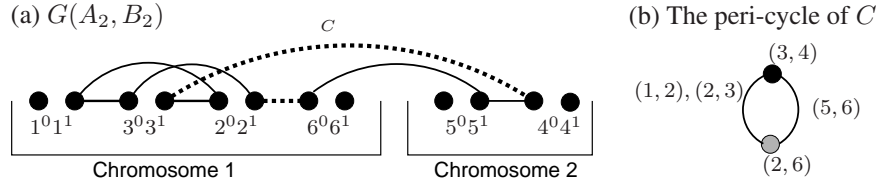


Fig. 3: Pericentric edges and peri-cycles. $A_2 = \{(1, 3, 2, \bullet, 6), (\bullet, 5, 4)\}$, $B_2 = \{(1, 2, 3, \bullet, 4), (\bullet, 5, 6)\}$. (a) The cycle graph $G(A_2, B_2)$. Pericentric edges are denoted by dotted lines. (b) The peri-cycle of the single cycle in $G(A_2, B_2)$. The labels of the edges denote the set of grey edges in the corresponding paths.

Lemma 4 Every peri-cycle has an even length and its node colors alternate along the cycle.

Proof. Let C be a cycle that contains a black pericentric edge (u_1, v_1) . Suppose $u_1, v_1, \dots, u_k, v_k$ is a path between two consecutive black pericentric edges in C . In other words, (u_k, v_k) is a black pericentric edge (possibly $u_1 = u_k$ and $v_1 = v_k$) and there are no other black pericentric edges in this path. Then according to Lemma 3 $cent(v_1) = cent(u_k) = 1$. There is an odd number of edges in the path between v_1 and u_k and thus there must be an odd number of pericentric edges between v_1 and u_k (Lemma 3). It follows that there must exist at least one grey pericentric edge between any two consecutive black pericentric edges. The same argument for a pair of consecutive grey pericentric edges implies that between two such edges there must be at least one black pericentric edge. \square

It follows that every vertex / edge in a peri-cycle has an *opposite* vertex / edge. Removing two opposite vertices / edges from a peri-cycle results in two paths of equal length. We define the *degree* of a cycle as the number of grey (equivalently, black) vertices in its peri-cycle. For example, the single cycle in Fig. 3 is of degree 1.

4 Mapping the Centromeres

This section demonstrates how mapping between the centromeres of A and B can be used to solve LSRT. We shall first see that trying all possible mappings and then solving the resulting SRT gives an exact exponential algorithm for LSRT. Later we shall show how to get an optimal mapping in polynomial time. Let $CEN = \{n+1, \dots, n+\eta\}$. For a genome A let \dot{A} be the set of all possible genomes obtained by the replacement of each \bullet element in the non-telocentric chromosomes by a distinct element from CEN . Each $i \in CEN$ can be added with either positive or negative sign. Thus $|\dot{A}| = \eta!2^\eta$. For example, if $A_1 = \{(1, 2, \bullet, 3, 4), (\bullet, 5, 6)\}$ then \dot{A}_1 consists of the genomes $\{(1, 2, 7, 3, 4), (\bullet, 5, 6)\}$ and $\{(1, 2, -7, 3, 4), (\bullet, 5, 6)\}$. Note that every $\dot{A} \in \dot{A}$ satisfies $Tails(\dot{A}) = Tails$. For each $i \in CEN$ we define $cent(i^0) = cent(i^1) = -1$. A pair $\dot{A} \in \dot{A}$ and $\dot{B} \in \dot{B}$ defines a mapping between the centromeres in non-telocentric chromosomes of A and B .

Observation 2 *Let $\dot{A} \in \dot{A}$ and $\dot{B} \in \dot{B}$. Then every edge (u, v) in $G(\dot{A}, \dot{B})$ is paracentric and satisfies $cent(u)cent(v) = -1$.*

The notion of legality is easily generalized to partially mapped genomes: a genome is *legal* if each of its chromosomes contains either a single \bullet element or a single, distinct element from CEN (but not both). Since A and $\dot{A} \in \dot{A}$ differ only in their centromeres, there is a trivial bijection between the set of translocations on \dot{A} and the set of translocations on A . This bijection also preserves legality: a legal translocation on \dot{A} is bijected to a legal translocation on A .

Lemma 5 *Let $\dot{A} \in \dot{A}$ and $\dot{B} \in \dot{B}$. Then every proper translocation on \dot{A} is legal and $d(\dot{A}, \dot{B}) = d_{old}(\dot{A}, \dot{B})$.*

Proof. Let $k = d_{old}(\dot{A}, \dot{B})$. If $k = 0$ then $\dot{A} = \dot{B}$ and hence $d(\dot{A}, \dot{B}) = 0$. Suppose $k > 0$. Let ρ be a translocation on \dot{A} satisfying $d_{old}(\dot{A} \cdot \rho, \dot{B}) = k - 1$. According to Corollary 1, ρ is either proper or bad. Suppose ρ is bad. Then there is another bad translocation ρ' that cuts the exact positions as ρ , thus satisfying $d_{old}(\dot{A} \cdot \rho', \dot{B}) = k - 1$,

and either ρ or ρ' is legal. Suppose ρ is proper. We shall prove that each of the new chromosomes contains a centromere and hence ρ is legal. Let X be a new chromosome resulting from the translocation ρ and let (u, v) be the new black edge in it. Since ρ is proper, $G(\dot{A} \cdot \rho, \dot{B})$ contains a path between u and v where all the edges existed in $G(\dot{A}, \dot{B})$. This path contains an odd number of edges. Following Observation 2 for $G(\dot{A}, \dot{B})$, $\text{cent}(u)\text{cent}(v) = -1$. X is composed of two old segments, X_u and X_v , that contain u and v respectively. If $\text{cent}(u) = -1$ then X_u contains an element from CEN , otherwise X_v contains one. In either case X contains an element from CEN . \square

Theorem 2 Let $\dot{A} \in \dot{\mathbb{A}}$. Then $d(A, B) = \min\{d_{\text{old}}(\dot{A}, \dot{B}) \mid \dot{B} \in \dot{\mathbb{B}}\}$.

Proof. By Lemma 5, $d(\dot{A}, \dot{B}) = d_{\text{old}}(\dot{A}, \dot{B})$ for every $\dot{A} \in \dot{\mathbb{A}}$ and $\dot{B} \in \dot{\mathbb{B}}$. Obviously a legal sorting of \dot{A} into any $\dot{B} \in \dot{\mathbb{B}}$ induces a legal sorting sequence of the same length, of A to B . Thus, $\min\{d_{\text{old}}(\dot{A}, \dot{B}) \mid \dot{B} \in \dot{\mathbb{B}}\} \geq d(A, B)$. On the other hand, every sequence of legal translocations that sorts A into B induces a legal sorting of \dot{A} into some $\dot{B} \in \dot{\mathbb{B}}$, thus $\min\{d_{\text{old}}(\dot{A}, \dot{B}) \mid \dot{B} \in \dot{\mathbb{B}}\} \leq d(A, B)$. \square

A pair of genomes, $\dot{A} \in \dot{\mathbb{A}}$ and $\dot{B} \in \dot{\mathbb{B}}$, define an *optimal* mapping between the centromeres of A and B if $d(A, B) = d_{\text{old}}(\dot{A}, \dot{B})$. Theorem 2 and Lemma 5 imply the following algorithm for LSRT:

Algorithm *Legal_Sorting*

1. Choose $\dot{A} \in \dot{\mathbb{A}}$ arbitrarily.
2. Compute $\dot{B} = \arg \min\{d_{\text{old}}(\dot{A}, \dot{B}) \mid \dot{B} \in \dot{\mathbb{B}}\}$.
3. Solve SRT on \dot{A} and \dot{B} - making sure that every bad translocation in the sorting sequence is legal.

It can be shown, by a minor modification of the algorithm in [5], that solving *SRT* with the additional condition that every bad translocation is legal can be done in $O(n^{3/2}\sqrt{\log(n)})$. Step 2 can be performed by enumerating all possible mappings and computing the SRT distance for each. This implies:

Lemma 6 *LSRT can be solved in $O(\eta!2^\eta n + n^{3/2}\sqrt{\log(n)})$.*

Our goal in the rest of this paper is to improve this result by speeding up Step 2, i.e., finding efficiently an optimal mapping between the centromeres of A and B .

5 Cent-Mappings

Our general strategy will be to iteratively map between two centromeres in A and B and replace them with a regular element until all centromeres in non-telocentric chromosomes are mapped. The resulting instance can be solved using SRT, but the increase in the number of elements may have also increased the solution value. The main effort henceforth will be to guarantee that the overall increase is minimal. For this we need

to study in detail the effect of each mapping step on the the cycle graph $G(A, B)$. Our analysis uses the SRT distance formula (Theorem 1). We shall ignore for now the parameter δ , and focus on the change in the simplified formula $n - c + l$ (N is not changed by mapping operations).

A mapping between two centromeres affects their corresponding black and grey pericentric edges. Let (i, i') and (j, j') be pericentric black and grey edges in $G(A, B)$ respectively. Suppose $cen \in CEN$ is added between i and i' in \dot{A} and between j and j' in \dot{B} . In this case (i, i') and (j, j') in $G(A, B)$ are replaced by the four (paracentric) edges (i, cen) , (cen, i') , (j, cen) and (cen, j') in $G(\dot{A}, \dot{B})$. (The first two edges are black, the latter are grey.) We refer to the addition of $cen \in CEN$ between (i, i') and (j, j') as a *cent-mapping* since it maps between two centromeres. Note that for each pair of centromeres in A and B there are two possible cent-mappings (corresponding to the relative signs of the added elements). Given $\dot{A} \in \dot{\mathbb{A}}$, every $\dot{B} \in \dot{\mathbb{B}}$ defines η disjoint cent-mappings and vice versa. Obviously every cent-mapping increases the number of genes by one ($\Delta n = +1$).

Lemma 7 *Every cent-mapping satisfies $\Delta c \in \{-1, 0, 1\}$.*

In the rest of the paper we will analyze the effect of a cent-mapping using pericycles. A peri-cycle can be viewed as a compact representation of a cycle focused on pericentric edges, which are the only edges affected by cent-mappings. A cent-mapping is called *proper*, *improper*, *bad* if $\Delta c = 1, 0, -1$ respectively. See Fig. 4 for illustrations of the three types of cent-mappings. We say that a cent-mapping *operates* on a cycle C if C contains at least one of the mapped pericentric edges. A proper / improper cent-mapping always operates on one cycle in $G(A, B)$; A bad cent-mapping always operates on two different cycles in $G(A, B)$.

Observation 3 *Every proper cent-mapping satisfies $\Delta l \in \{0, 1\}$. An improper cent-mapping satisfies $\Delta l = 0$. A bad cent-mapping satisfies $\Delta l \in \{0, -1, -2\}$.*

It follows that a proper cent-mapping satisfies $\Delta(n - c + l) = 0$ iff $\Delta l = 0$; An improper cent-mapping satisfies $\Delta(n - c + l) = 1$; A bad cent-mapping satisfies $\Delta(n - c + l) = 0$ iff $\Delta l = -2$. A proper cent-mapping is *safe* if it satisfies $\Delta l = 0$. In the following sections we present two classes of cycles, “annoying” and “evil” for which any set of proper cent-mappings that eliminates all their pericentric edges is unsafe.

5.1 Annoying Cycles

In this section we focus on cycles in leaves. The degree of every cycle in a leaf is at most 1 (otherwise it must be external). Moreover, a leaf can contain at most one cycle of degree 1 (for the same reason). A cycle is called *annoying* if: (i) it is contained in a leaf, (ii) its degree is 1, and (iii) a proper cent-mapping on its two pericentric edges satisfies $\Delta l = 1$ (i.e. one leaf is split into two leaves). See Fig. 5(a). Thus a proper cent-mapping on an annoying cycle satisfies $\Delta(n - c + l) = 1$. On the other hand, any bad cent-mapping on a cycle contained in the span of a leaf (annoying or not) results in the elimination of that leaf. Thus, a cent-mapping on two cycles in leaves satisfies $\Delta(n - c + l) = 1 + 1 - 2 = 0$. Let \mathbb{C}_{ann} denote the set of annoying cycles and let $\text{ann} = |\mathbb{C}_{\text{ann}}|$. Let \mathbb{C}_{nona} be the set of non-annoying cycles of degree 1 that are contained in the span of a leaf. See Fig. 5(b). Let $\text{nona} = |\mathbb{C}_{\text{nona}}|$.

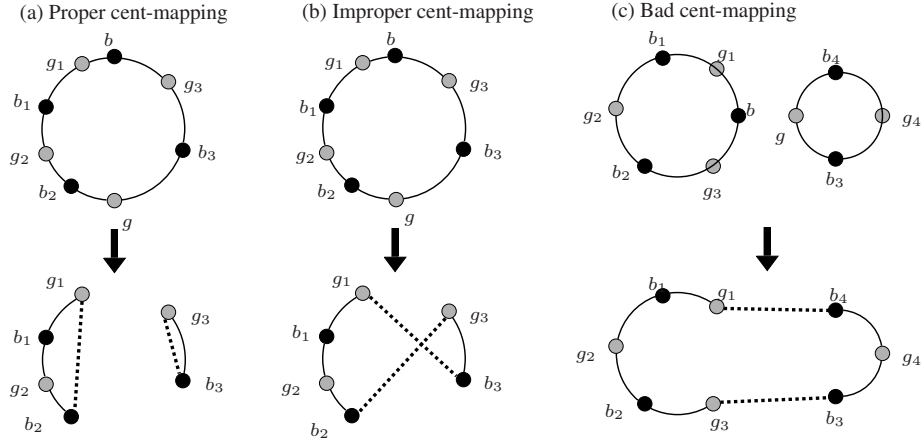


Fig. 4: The effect of a cent-mapping on peri-cycles. Each of the cycles is a peri-cycle with black and grey nodes corresponding to centromeres (pericentric edges) in A and B , respectively. In all cases a cent-mapping on b and g in the top peri-cycles is performed, and the bottom peri-cycles are the result. Dotted lines denote new edges. (a) and (b) show the two alternative cent-mappings of a pair of pericentric edges in the same cycle. In case (c) each of the two alternatives generate a single cycle.

5.2 Evil Cycles

In this section we focus on cycles that are not in leaves. Let C be a cycle of degree 1 that is not in a leaf and let C^P be its peri-cycle. Let (b, g) be an edge in C^P . Denote by $V(b, g)$ the set of grey edges in the corresponding path between b and g in C . The edge (b, g) is *bad* if after a proper cent-mapping on b and g the edges in $V(b, g)$ belong to a leaf, otherwise it is *good*. For example, in Fig. 3 the edge (b, g) where $V(b, g) = \{(1, 2), (2, 3)\}$ is bad.

Lemma 8 *The “badness” of edge (b, g) in a peri-cycle is unchanged by other cent-mappings not involving b and g .*

Lemma 9 *Let C be a cycle satisfying: (i) $\deg(C) > 0$, and (ii) C contains a new grey edge, g_{new} , that was created by a cent-mapping. Let (b, g) be an edge in the peri-cycle of C such that $V(b, g)$ contains g_{new} . Then (b, g) is good.*

A path in a peri-cycle is *bad* if all the edges in it are bad. For a path P , let $\text{len}(P)$ denote the number of vertices in P . A cycle C is called *evil* if its peri-cycle contains a bad path P such that $\text{len}(P) > \deg(C)$. For example, the single cycle in Fig. 3 is evil since it contains a bad edge, which is a bad path of length 2, and its degree is 1. An example of an evil cycle with only bad edges in its peri-cycle is presented in Fig. 5. Let \mathbb{C}_{evil} denote the set of all evil cycles that are not in leaves. Define $\text{evil} = |\mathbb{C}_{\text{evil}}|$.

Lemma 10 *Let C be a cycle that does not belong to a leaf. There is a set of safe proper cent-mappings of all the pericentric edges in C iff C is not evil.*

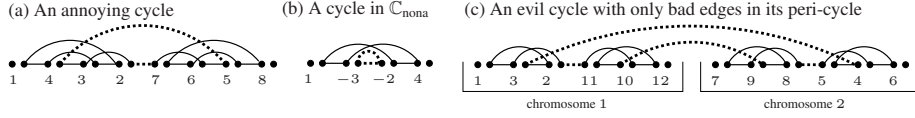


Fig. 5: Examples of cycles in \mathbb{C}_{ann} , \mathbb{C}_{nona} and \mathbb{C}_{evil} . In all the figures the target genome B is a fragmented identity permutation (i.e., every grey edge is of the form $(i, i + 1)$), pericentric edges are denoted by dotted lines.

Proof. Let C^P be the peri-cycle of C and let $k = \deg(C)$. Suppose C is evil. Then PC contains a bad path P with $k + 1$ vertices. There are $2k$ vertices in C^P , thus any proper matching of all the pericentric edges in C must match two vertices from P . It follows that there must be a proper cent-mapping on the two ends of an edge in P . Hence, by definition this cent-mapping is unsafe.

Suppose C is not evil. If $k = 1$ then the two edges in C^P are good and the proper cent-mapping of the two pericentric edges in C is safe. Suppose $k > 1$. Let $C^P = P_1, P_2$ where P_1 is a longest bad path in C^P . Let u be the first vertex in P_1 and let v be the last vertex in P_2 . Then (u, v) is a good edge in C^P . Let C_1 and C_2 be the two cycles created by the proper cent-mapping on u and v , where C_1 contains $V(u, v)$. Obviously this proper cent-mapping is safe, $\deg(C_1) = 0$ and $\deg(C_2) = k - 1$. It suffices to prove that C_2 is not evil. Let C_2^P be the peri-cycle of C_2 . Then $C_2^P = P'_1 P'_2$ where $\text{len}(P'_1) = \text{len}(P_1) - 1$, $\text{len}(P'_2) = \text{len}(P_2) - 1$, and P'_1 and P'_2 are connected by good edges (Lemma 9). Let p be the length of the longest bad path in C_2^P . Then (i) $p \leq \text{len}(P_1) \leq k$ (since P_1 is a longest bad path in C), (ii) $p \leq \max(\text{len}(P'_1), \text{len}(P'_2)) = \text{len}(P'_2)$, and (iii) $\text{len}(P_1) + \text{len}(P_2) = 2k$. It follows that $p \leq k - 1 = \deg(C_2)$. Thus by definition C_2 is not evil. \square

Corollary 2 Every cent-mapping satisfies $\Delta(n - c + l + \text{evil}) \geq 0$.

We partition \mathbb{C}_{evil} into three classes:

- $\mathbb{C}_{\text{evil}}^1$: Cycles of even degree and only bad edges in their peri-cycle.
- $\mathbb{C}_{\text{evil}}^2$: Cycles of odd degree and only bad edges in their peri-cycle.
- $\mathbb{C}_{\text{evil}}^3$: Cycles with at least one good edge in their peri-cycle.

If $C \in \mathbb{C}_{\text{evil}}$ is of degree 1 then $C \in \mathbb{C}_{\text{evil}}^3$ (since otherwise it would be in a leaf). Every new evil cycle (i.e. an evil cycle created by a cent-mapping) contains a good edge (Lemma 9) and hence belong to $\mathbb{C}_{\text{evil}}^3$. Let $C \in \mathbb{C}_{\text{evil}}^3$ and let (b, g) be an edge opposite to a good edge in the peri-cycle of C . A proper cent-mapping on b and g satisfies $\Delta l = 1$, $\Delta \text{evil} = -1$ and hence $\Delta(n - c + l + \text{evil}) = 0$. Such a cent-mapping can be viewed as a replacement of an evil cycle with a leaf. On the other hand, every proper cent-mapping on a cycle in $\mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{evil}}^2$ satisfies $\Delta(n - c + l + \text{evil}) = \Delta(l + \text{evil}) = 1$.

Lemma 11 Let $C \in \mathbb{C}_{\text{evil}}$. There exists an improper cent-mapping on C for which $\Delta \text{evil} = -1$ iff $C \notin \mathbb{C}_{\text{evil}}^1$.

In other words: for every cycle in $\mathbb{C}_{\text{evil}}^2 \cup \mathbb{C}_{\text{evil}}^3$ there exists an improper cent-mapping satisfying $\Delta(n - c + l + \text{evil}) = 0$; Every improper cent-mapping on a cycle in $\mathbb{C}_{\text{evil}}^1$

satisfies $\Delta(n - c + l + evil) = 1$. It follows that a cent-mapping on $C \in \mathbb{C}_{evil}^1 \cup \mathbb{C}_{ann}$ satisfies $\Delta(n - c + l + evil) = 0$ only if it is bad.

Lemma 12 *Let $C_1, C_2 \in \mathbb{C}_{evil} \cup \mathbb{C}_{ann}$, where $deg(C_1) \leq deg(C_2)$. If $deg(C_1) = deg(C_2)$ then every bad cent-mapping on C_1 and C_2 satisfies $\Delta(l + evil) = -2$. If $deg(C_1) < deg(C_2)$ there exists a bad cent-mapping on C_1 and C_2 satisfying $\Delta(l + evil) = -2$ iff $C_2 \in \mathbb{C}_{evil}^3$.*

5.3 Sorting by $d + 2$ Legal Translocations in Polynomial Time

In this section we present upper and lower bounds for the legal translocation distance. These bounds provide an intuition for the rather complicated formula for the legal translocation distance presented in the next section. The proof of the upper bound implies an approximation algorithm that sorts A into B using at most $d(A, B) + 2$ legal translocations. For a set of cycles \mathbb{C} let $subset(\mathbb{C}, i) = \{C \in \mathbb{C} : deg(C) = i\}$. For example, $subset(\mathbb{C}_{evil}^1 \cup \mathbb{C}_{ann}, 1) = \mathbb{C}_{ann}$. Define

$$DEG = \{i : |subset(\mathbb{C}_{evil}^1 \cup \mathbb{C}_{ann}, i)| \text{ is odd}\} \quad CYC = \mathbb{C}_{evil}^3 \cup \mathbb{C}_{nona}$$

For example, if the degrees of the cycles in $\mathbb{C}_{evil}^1 \cup \mathbb{C}_{ann}$ are $\{1, 2, 2, 2, 4, 4, 6, 8\}$ then $DEG = \{1, 2, 6, 8\}$. The *bad cent-mappings* graph, BCM , is defined as follows. It is a bipartite graph whose two parts are DEG and CYC . Vertices $i \in DEG$ and $C \in CYC$ are connected by an edge if $deg(C) \geq i$. See Fig. 6 for an example. Thus an edge (i, C) represents a bad cent-mapping operating on C and $C' \in subset(\mathbb{C}_{evil}^1 \cup \mathbb{C}_{ann}, i)$ for which $\Delta(n - c + l + evil) = 0$ and $\Delta|DEG| = -1$.

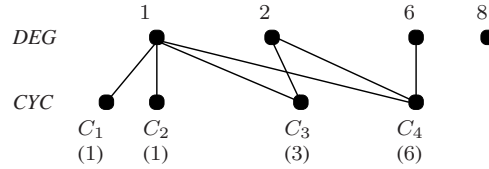


Fig. 6: An example for a bad cent-mappings (BCM) graph. $DEG = \{1, 2, 6, 8\}$, $CYC = \{C_1, C_2, C_3, C_4\}$. The degree of each cycle in CYC appears in brackets below the cycle.

A *matching*, M , is a subgraph of BCM where every vertex is adjacent to exactly one edge. The size of a matching M , denoted $|M|$, is the number of edges in it. Finding a maximal matching in BCM is an easy task that can be completed in linear time. Define $fbad = |DEG| - |M|$, where M is a maximal matching. For a matching M let F_M be the forest of internal components after performing a bad cent-mapping on every $C \in \mathbb{C}_{ann} \cup M$. In other words, F_M is obtained from F by the deletion of every component containing in its span a cycle from either \mathbb{C}_{ann} or $\mathbb{C}_{nona} \cap M$. Below we describe Algorithm *Get_Mapping_1* for finding a mapping between the centromeres of A and B .

Algorithm *Get_Mapping_1*

1. Let M be any maximal matching in BCM
2. Perform a bad cent-mapping on every $C_1, C_2 \in \mathbb{C}_{\text{evil}}^1$, where $\text{deg}(C_1) = \text{deg}(C_2)$.
3. Perform a bad cent-mapping on every pair of cycles in \mathbb{C}_{ann} .
/* Now $|\mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{ann}}| = |\text{DEG}|$ */
4. For every $(i, C) \in M$ perform a bad cent-mapping on C and $C' \in \text{subset}(\mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{ann}}, i)$, such that $\Delta(l + \text{evil}) = -2$ (Lemma 12).
5. While $|\text{DEG}| \geq 3$: For $i = 1, 2, 3$ let $C_i \in \mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{ann}}$, and $\text{deg}(C_1)$ is minimal. Perform a bad cent-mapping on C_2 and C_3 and let C_4 be the new evil cycle. Perform a bad cent-mapping on C_1 and C_4 such that $\Delta(l + \text{evil}) = -2$ (Lemma 12).
6. If $|\text{DEG}| = 2$: Perform a bad cent-mapping on $C, C' \in \mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{ann}}$.
If $|\text{DEG}| = 1$: Perform an improper cent-mapping on $C \in \mathbb{C}_{\text{evil}}^1 \cup \mathbb{C}_{\text{ann}}$.
/* Now $|\mathbb{C}_{\text{evil}}^1| = \text{ann} = 0$ */
7. Perform an improper cent-mapping on every $C \in \mathbb{C}_{\text{evil}}$ such that $\Delta \text{evil} = -1$ (Lemma 11).
/* Now $\text{evil} = 0$ */
8. Perform safe proper cent-mappings on every cycle of degree at least 1 (Lemma 10).
9. Perform a proper cent-mapping on every $C \in \mathbb{C}_{\text{nona}}$.

Theorem 3 Let $d = d(A, B)$ and let $f = n - N - c + l + \text{evil} + \lceil \text{fbad}/3 \rceil$. Then $d \in [f, f + 2]$. In particular, Algorithm *Get_Mapping_1* produces $\dot{A} \in \dot{\mathbb{A}}$ and $\dot{B} \in \dot{\mathbb{B}}$ for which $d(\dot{A}, \dot{B}) \leq d + 2$.

6 The Legal Translocation Distance

Define $\text{mbad} = \text{fbad} \bmod 3$. For a matching M define $\text{fgood}(M) = |\mathbb{C}_{\text{evil}}^3 \setminus M|$. Define $\delta' \in \{0, 1, 2\}$ as follows. $\delta' = 2$ iff the following conditions are satisfied: (i) $\mathbb{C}_{\text{evil}}^2 = \mathbb{C}_{\text{evil}}^3 = \text{DEG} = \emptyset$, (ii) $|F_\emptyset| = 1$, (iii) l and ann are even, and (iv) If $\text{ann} > 0$ then $\text{nona} = 0$. Let $\delta' = 0$ iff at least one of the following is satisfied:

- ($\gamma 1$) There exists a maximal matching M satisfying l_M is even and $|F_M| \neq 1$.
- ($\gamma 2$) $\text{mbad} = 1$, $\text{DEG} = \{1\}$ and either (i) $|F| > 1$, or (ii) l_\emptyset is odd and $|\mathbb{C}_{\text{evil}}^2| > 0$
- ($\gamma 3$) $\text{mbad} = 1$ and $\text{DEG} \neq \{1\}$.
- ($\gamma 4$) $\text{mbad} = 2$ and there exists a maximal matching M for which either (i) l_M is odd or (ii) $\text{fgood}(M) \geq 1$.
- ($\gamma 5$) There exists a maximal matching M for which $\text{fgood}(M) \geq 2$
- ($\gamma 6$) There exists a maximal matching M for which $\text{fgood}(M) \geq 1$, l_M is odd and $C \in \mathbb{C}_{\text{evil}}^3 \setminus M$ can be replaced by a leaf such that $|F_M| > 1$.

Theorem 4 The legal translocation distance between A and B is $d(A, B) = n - N - c(A, B) + l(A, B) + \text{evil}(A, B) + \lceil \text{fbad}(A, B)/3 \rceil + \delta'(A, B)$.

The proof of Theorem 4 is by a case analysis of the change in each the parameters $n - c$, l , evil , fbad and δ' for each cent-mapping and hence is quite involved. It leads to a polynomial time algorithm for finding an optimal mapping between the centromeres of A and B . This algorithm can be viewed as an extension of Algorithm *Get_Mapping_1* that includes a constant number of additional operations that consider δ' .

Theorem 5 *LSRT can be solved in $O(\eta n + n^{3/2}\sqrt{\log(n)})$ time.*

Proof. Finding an optimal mapping between the centromeres of A and B can be done in $O(\eta n)$ in the following manner. The set of peri-cycles can be computed in $O(n)$. For every edge in a peri-cycle we compute its “badness” in $O(n)$ by simply performing the corresponding proper cent-mapping. Computing the badness of all the edges thus takes $O(\eta n)$. Computing $\mathbb{C}_{\text{evil}}^1$, $\mathbb{C}_{\text{evil}}^2$, $\mathbb{C}_{\text{evil}}^3$, \mathbb{C}_{ann} and \mathbb{C}_{nona} and DEG requires a simple traversal of all the edges in every peri-cycle. Hence it can be done in $O(\eta)$. Overall the algorithm performs $O(\eta)$ operations where each can be implemented in $O(n)$ time. \square

7 Summary and Future Work

Computational studies in genome rearrangements have overlooked centromeres to date. In this study we presented a new model for genomes that accounts for centromeres. Using this model we defined the problem of legal sorting by reciprocal translocations (LSRT) and proved that it can be solved in polynomial time. Unfortunately, the legal translocation distance formula appears to be quite complex and it is an interesting open problem whether it or its proof can be simplified.

A solvable LSRT instance requires the two input genomes to be co-tailed and with the same set of elements (see Section 3.1). This requirement is a rather strong and unrealistic. Allowing for reversals, non-reciprocal translocations, fissions and fusions will cancel these restrictions. Under a centromere-aware model, fissions and fusions are legal if they are centric [7, 8]. In future work we intend to study an extension of LSRT that allows for reversals, (centric) fusions and fissions. We expect an exact algorithm for this extended problem to bring us nearer to realistic rearrangement scenarios than can be done today.

References

1. D.A. Bader, B. M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.
2. A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Journal of Computational Biology*, 13(2):567–578, 2006.
3. S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71:137–151, 1996.
4. H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal of Computing*, 29(3):880–892, 2000.
5. M. Ozery-Flato and R. Shamir. An $O(n^{3/2}\sqrt{\log(n)})$ algorithm for sorting by reciprocal translocations. In *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching (CPM 2006)*, volume 4009 of *LNCS*, pages 258–269. Springer, 2006.
6. M. Ozery-Flato and R. Shamir. Sorting by translocations via reversals theory. In *Proceedings of the 4th RECOMB Satellite Workshop on Comparative Genomics*, volume 4205 of *LNCS*, pages 87–98. Springer, 2006.
7. J. Perry, H.R. Slater, and K.H. Andy Choo. Centric fission—simple and complex mechanisms. *Chromosome Research*, 12(6):627–640, 2004.
8. J.B. Searle. Speciation, chromosomes, and genomes. *Genome Research*, 8(1):1–3, 1998.
9. B.A. Sullivan, M.D. Blower, and G.H. Karpen. Determining centromere identity: Cyclical stories and forking paths. *Nature Reviews Genetics*, 2(8):584–596, 2001.