ARTICLE IN PRESS JDA:373

Journal of Discrete Algorithms ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

An $O(n^{3/2}\sqrt{\log(n)})$ algorithm for sorting by reciprocal translocations

Michal Ozery-Flato, Ron Shamir *

The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Translocations
Reversals
Genome rearrangements

ABSTRACT

We prove that sorting by reciprocal translocations can be done in $O(n^{3/2}\sqrt{\log(n)})$ for an n-gene genome. Our algorithm is an adaptation of the algorithm of Tannier, Bergeron and Sagot for sorting by reversals. This improves over the $O(n^3)$ algorithm for sorting by reciprocal translocations given by Bergeron, Mixtacki and Stoye (2006) [4].

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

In this paper we study the problem of sorting by reciprocal translocations (abbreviated SRT). *Reciprocal translocations* exchange non-empty ends between two chromosomes. Given two multi-chromosomal genomes A and B, the problem of SRT is to find a shortest sequence of reciprocal translocations that transforms A into B. SRT was first introduced by Kececioglu and Ravi [11] and was given a polynomial time algorithm by Hannenhalli [6]. Bergeron, Mixtacki and Stoye [4] pointed to an error in Hannenhalli's proof of the reciprocal translocation distance formula and consequently in Hannenhalli's algorithm. They presented a new $O(n^3)$ algorithm, which to the best of our knowledge, is the only extant correct algorithm for SRT.[1]

Reversals (or inversions) reverse the order and the direction of transcription of the genes in a segment inside a chromosome. Given two uni-chromosomal genomes π_1 and π_2, the problem of sorting by reversals (abbreviated SBR) is to find a shortest sequence of reversals that transforms π_1 into π_2. This problem has been intensively studied [8,5,9,1,2,15]. Tannier, Bergeron and Sagot [15] presented an elegant algorithm for SBR that can be implemented in $O(n^{3/2}\sqrt{\log(n)})$ using a clever data structure by Kaplan and Verbin [10]. This is currently the fastest algorithm for SBR.

In this paper we prove that SRT can be solved in $O(n^{3/2}\sqrt{\log(n)})$ for an n-gene genome. Our algorithm for SRT is similar to the algorithm by Tannier, Bergeron and Sagot [15] for SBR. The key idea is to recast translocations as reversals, and then exploit the novel theoretical improvements in SBR theory to obtain faster SRT algorithms. (It should be noted that Hannenhalli and Pevzner have already established and exploited the basic connection between translocations and reversals, in the context of sorting a genome by reversals and translocations [7].) Our approach builds on generalizing the overlap graph. Most studies of SBR to date relied explicitly or implicitly on the combinatorial structure of the overlap graph for representing the relations between two permutations. Since translocations involve multiple chromosomes, we generalize the notion of (uni-chromosomal) overlap graph to include chromosomal information, and show that the same conceptual algorithmic framework developed for SBR applies to SRT, via this generalized overlap graph. While our final algorithm is very similar to that of Tannier et al., the proofs had to be completely redone. Another contribution of this study is in

* Corresponding author.
E-mail address: rshamir@tau.ac.il (R. Shamir).

[1] Li et al. [12] gave a linear time algorithm for computing the reciprocal translocation distance (without producing a shortest sequence). Wang et al. [16] presented an $O(n^2)$ algorithm for SRT. However, the algorithms in [12,16] rely on an erroneous theorem of Hannenhalli and hence provide incorrect results in certain cases.

1570-8667/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2011.04.003

showing that the general SRT problem can be reduced in linear time to a special case, and thus time complexity analysis can be done for such special cases only.

The paper is organized as follows. The necessary preliminaries are given in Section 2. In Section 3 we give a linear time reduction from SRT to a simpler restricted subproblem. In Section 4 we prove the main theorem and present the algorithm for the restricted subproblem. In Section 5 we describe an $O(n^{3/2}\sqrt{\log(n)})$ implementation of the algorithm. A preliminary version of this study was published in the proceedings of CPM 2006 [13].

2. Preliminaries

This section provides a basic background for the analysis of SRT. It follows to a large extent the nomenclature and notation of [6,9,4]. 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. For example, the following genome consists of 8 genes in two chromosomes:

$$A_1 = \{(1, -3, -2, 4, -7, 8), (6, 5)\}$$

The *reverse* of a sequence of genes $I = (x_1, \ldots, x_l)$ is $-I = (-x_l, \ldots, -x_1)$. A *reversal* reverses a segment of genes inside a chromosome. Two chromosomes, X and Y, are *identical* if either $X = Y$ or $X = -Y$. Therefore, *flipping* chromosome X into $-X$ does not affect the chromosome it represents. For example, the following are two equivalent representations of the same genome

$$\{\underline{(1, -3, -2, 4, -7, 8)}, (6, 5)\} \equiv \{\underline{(-8, 7, -4, 2, 3, -1)}, (6, 5)\}$$

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 ways to perform a translocation on X and Y. A *prefix–suffix* translocation switches X_1 with Y_2 resulting in:

$$(\underline{X_1}, X_2), (Y_1, \underline{Y_2}) \quad \Rightarrow \quad (\underline{-Y_2}, X_2), (Y_1, \underline{-X_1})$$

A *prefix–prefix* translocation switches X_1 with Y_1 resulting in:

$$(\underline{X_1}, X_2), (\underline{Y_1}, Y_2) \quad \Rightarrow \quad (\underline{Y_1}, X_2), (\underline{X_1}, Y_2)$$

The following is an example of prefix–prefix and prefix–suffix translocations that cut the genome in the same place:

$$\{(\underline{1, -3, -2, 4}, -7, 8), (\underline{6}, 5)\} \quad \Rightarrow \quad \{(\underline{6}, -7, 8), (\underline{1, -3, -2, 4}, 5)\}$$
$$\{(\underline{1, -3, -2, 4}, -7, 8), (6, \underline{5})\} \quad \Rightarrow \quad \{(\underline{-5}, -7, 8), (6, \underline{-4, 2, 3, -1})\}$$

Recall that chromosome flips do not affect the genome, but rather move between different representations of the same genome. Thus 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 $X = (x_1, \ldots, x_k)$ define $\mathit{Tails}(X) = \{x_1, -x_k\}$. Note that flipping X does not change $\mathit{Tails}(X)$. For a genome A define $\mathit{Tails}(A) = \bigcup_{X \in A} \mathit{Tails}(X)$. For example:

$$\mathit{Tails}(A_1) = \mathit{Tails}(\{(1, -3, -2, 4, -7, 8), (6, 5)\}) = \{1, -8, 6, -5\}$$

Two genomes A' and A'' are *co-tailed* if $\mathit{Tails}(A') = \mathit{Tails}(A'')$. In particular, two co-tailed genomes have the same number of chromosomes (recall that all genes in a genome are unique). Note that if A'' was obtained from A' by performing a reciprocal translocation then $\mathit{Tails}(A'') = \mathit{Tails}(A')$. Therefore, SRT is defined 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.

2.1. The cycle graph

In this section we present the cycle graph of genomes A and B, which was first defined in [6]. Let N be the number of chromosomes in A (equivalently, B). We shall always assume that both A and B contain the genes $\{1, \ldots, n\}$. The *cycle graph* of A and B, denoted $G(A, B)$, is an undirected graph 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 the ends of a small arrow directed from i^0 to i^1). For every pair of genes, i and j, where j immediately follows i in some chromosome of A (respectively, B) add a black (respectively, gray) (undirected) edge

$$(i, j) \equiv (\mathit{out}(i), \mathit{in}(j))$$

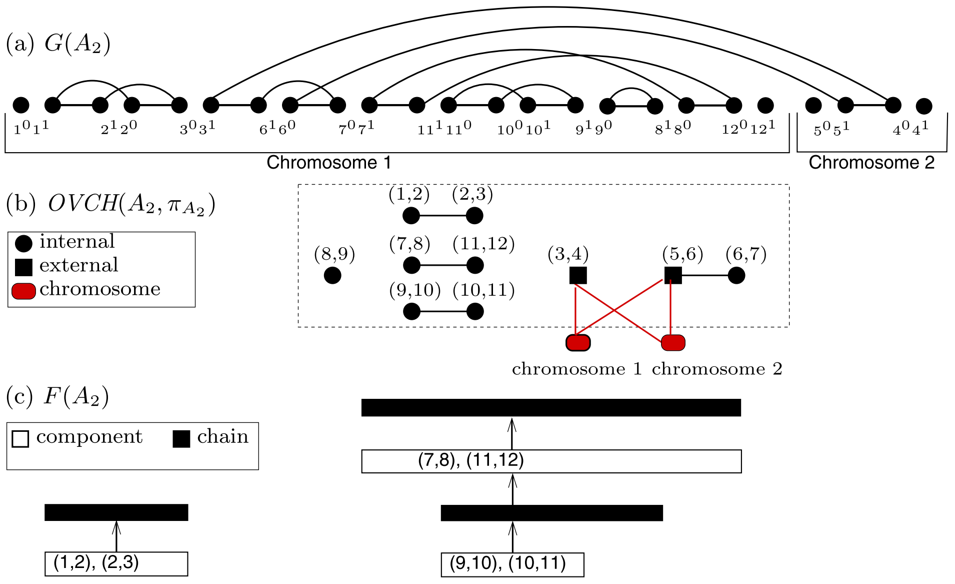

Fig. 1. Auxiliary graphs for $A_2 = \{(1, -2, 3, -6, 7, -11, 10, -9, -8, 12), (5, 4)\}$, $B_2 = \{(1, \ldots, 4), (5, \ldots, 12)\}$, $\pi_{A_2} = (1, -2, 3, -6, 7, -11, 10, -9, -8, 12, 5, 4)$. (a) The cycle graph. Black edges are horizontal; gray edges are curved. (b) The overlap graph with chromosomes. The graph induced by the vertices within the dashed rectangle is $OV(A_2, \pi_{A_2})$, the same graph without the chromosome vertices. (c) The forest of internal components.

where

$$out(i) = \begin{cases} i^1 & \text{if } i \text{ has a positive sign in } A \text{ (respectively, } B\text{)} \\ i^0 & \text{otherwise} \end{cases}$$

and

$$in(j) = \begin{cases} j^0 & \text{if } j \text{ has a positive sign in } A \text{ (respectively, } B\text{)} \\ j^1 & \text{otherwise} \end{cases}$$

An example is given in Fig. 1(a). There are $n - N$ black edges and $n - N$ gray edges in $G(A, B)$. Since genomes A and B are co-tailed, every vertex in $G(A, B)$ has degree 2 or 0, where vertices of degree 0 (isolated vertices) belong to *Tails*(A) (equivalently, *Tails*(B)). Therefore, $G(A, B)$ is uniquely decomposed into cycles with alternating gray and black edges.

In the following we assume, without loss of generality, that each chromosome of B is an increasing sequence of consecutive positive numbers. For example, $B_1 = \{(1, 2, 3, 4, 5), (6, 7, 8)\}$. Thus every gray edge in $G(A, B)$ is of the form $(out(i), in(i + 1)) \equiv (i^1, (i + 1)^0) \equiv (i, i + 1)$. As genomes B and A are co-tailed, once genome A is given, genome B is fixed. Thus we can define $G(A) \equiv G(A, B)$.

Let $c(A)$ denote the number of cycles in $G(A)$. Note that if $A = B$ then $c(A) = n - N$ is maximal. We denote by $A \cdot \phi$ the genome obtained after the translocation ϕ is applied to A. For any parameter ψ, let $\Delta\psi$ be the increase in ψ after applying ϕ, i.e., $\Delta\psi = \psi(A \cdot \phi) - \psi(A)$. The following lemma describes how c is affected by a translocation.

Lemma 1. *(See [11].) Let ϕ be a translocation. If ϕ cuts two black edges in different cycles then the two cycles are merged into one cycle and $\Delta c = -1$. If ϕ acts on black edges belonging two the same cycle then either the cycle is split into two cycles and $\Delta c = 1$, or there is no change in the number of cycles (i.e. $\Delta c = 0$).*

A translocation is *proper* if $\Delta c = 1$ (i.e. one cycle splits into two). A gray edge $(i, i + 1)$ is *external* if i and $i + 1$ belong to two different chromosomes, otherwise it is *internal*. For example, in Fig. 1(a), $(5, 6)$ is external, while $(11, 12)$ is internal. An *adjacency* is a cycle with two edges. Thus, every adjacency corresponds to a pair of genes $i, i + 1$, where either $(i, i + 1)$ or $(-i + 1, -i)$ is contained in one of the chromosomes of A.

Observation 1. *Every external edge $(i, i + 1)$ defines a (proper) translocation that creates the adjacency $(i, i + 1)$.*

2.2. The overlap graph with chromosomes

The overlap graph of a signed permutation was introduced in [9]. In this section we present an extension of this graph for genome A.

A *signed permutation* $\pi = (\pi_1, \ldots, \pi_n)$ is a permutation on the integers $\{1, \ldots, n\}$, where a sign of plus or minus is assigned to each number. Let A be a genome with the set of genes $\{1, \ldots, n\}$. Let π_A be an arbitrary concatenation of the chromosomes in A, in arbitrary order and orientation. Then π_A is a signed permutation of size n.

Place the vertices of $G(A)$ along a straight line according to their order in π_A. Now, every gray edge and every chromosome is associated with an interval of vertices in $G(A)$. Two intervals *overlap* if their intersection is not empty but none contains the other. The *overlap graph with chromosomes* of genome A w.r.t. π_A, denoted $OVCH(A, \pi_A)$, is defined as follows. The set of nodes is the set of chromosomes in A and gray edges in $G(A)$. Two nodes are connected if their corresponding intervals in $G(A)$ overlap. An example is given in Fig. 1(b). 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 gray edges.

Let $OV(A, \pi_A)$ be the subgraph of $OVCH(A, \pi_A)$ induced by the set of nodes that correspond to gray edges (i.e., excluding the chromosomes' nodes). This graph is an extension of the overlap graph of a signed permutation defined in [9]. We shall use the word “component” for a connected component of $OV(A, \pi_A)$. For example, in Fig. 1(b), $OV(A_2, \pi_{A_2})$ contains six components: $\{(8, 9)\}$, $\{(1, 2), (2, 3)\}$, $\{(7, 8), (11, 12)\}$, $\{(9, 10), (10, 11)\}$, $\{(3, 4)\}$, and $\{(5, 6), (6, 7)\}$.

A vertex in $OVCH(A, \pi_A)$ is *external* if its corresponding edge in $G(A)$ is external, otherwise it is *internal*. For example, in Fig. 1(b), the vertex $(5, 6)$ is external while the vertex $(6, 7)$ is internal. Obviously a vertex is external iff it is connected to a chromosome.

A component is *external* if at least one of the vertices in it is external, otherwise it is *internal*. A component is *trivial* if it is composed of one internal vertex, which corresponds to an adjacency. For example, in Fig. 1, $\{(8, 9)\}$ is a trivial component, $\{(7, 8), (11, 12)\}$ is an internal non-trivial component, and $\{(3, 4)\}$ is an external component. Note that if $A = B$ then all the components are trivial. As we shall see later, a genome without non-trivial internal components can be sorted by a sequence of proper translocations. In case a genome does have non-trivial internal components, these components can become external after some non-proper translocations are applied.

The permutation π_A matches to every vertex v of $OV(A, \pi_A)$ an interval of genes, $I(v) \subset \pi_A$. For example, in Fig. 1(b) the vertex $(7, 8)$ is associated with the interval $(7, -11, 10, -9, -8)$. The interval associated with a component M, $I(M) \subset \pi_A$, is the minimal interval of genes for which $I(v) \subset I(M)$, for every vertex $v \in M$. For example, consider the components of $OV(A_2, \pi_{A_2})$, shown in Fig. 1(b). Then $I(\{(7, 8), (11, 12)\}) = (7, -11, 10, -9, -8, 12)$ and $I(\{(5, 6), (6, 7)\}) = (-6, 7, -11, 10, -9, -8, 12, 5)$. Observe that the interval of the former component is contained within a chromosome, while the interval of the latter extends over two chromosomes.

Observation 2. *Let M be a component. Then M is internal iff $I(M)$ is contained in one chromosome.*

Observation 3. *The set of internal components is independent of the specific concatenation π_A. In other words, the set of internal components remains unchanged with all the concatenations of A.*

In [4] the term “component” is defined in a different manner. However, as we show below, the two definitions are equivalent when the components are internal. Note that the terms “internal” and “external” correspond to the terms “intrachromosomal” and “interchromosomal” in [4]. To make a distinction, we refer to the term “component” defined in [4] as “BMS-component”. We now define this term and prove the equivalence.

For a signed permutation π, we denote by $P(\pi)$ the signed permutation obtained from π by adding the first element 0 and the last element $n + 1$. For example, for the permutation in Fig. 1:

$$P(\pi_{A_2}) = (\mathbf{0}, 1, -2, 3, -6, 7, -11, 10, -9, -8, 12, 5, 4, \mathbf{13})$$

We refer to $P(\pi)$ as a *padded* signed permutation.

A *BMS-component* is an interval of $P(\pi)$, from i to $i + j$ or from $-(i + j)$ to $-i$, where $j > 0$, whose set of (unsigned) elements is $\{i, \ldots, i + j\}$, and that is not the union of smaller such intervals. For example, $P(\pi_{A_2})$ contains five BMS-components: $(1, -2, 3)$, $(3, \ldots, 13)$, $(7, \ldots, 12)$, $(-11, 10, -9)$, and $(-9, -8)$. The interval $(-11, 10, -9, -8)$ is not a BMS-component as it is the union of $(-11, 10, -9)$ and $(-9, -8)$.

The overlap graph of a signed permutation was originally defined for a padded permutation [9]. The connected components of this graph play a major role in the analysis of SBR. The analysis for SBR was revised in [3] and an alternative definition was given for the components of the overlap graph, namely BMS-components. It is implied in [3] that there is a bijective mapping between the set of BMS-components of $P(\pi_A)$ and the set of components in $OV(P(\pi_A))$, the overlap graph of $P(\pi_A)$. More specifically, I is a BMS-component of $P(\pi_A)$ iff $I = I(M)$ for some component M in $OV(P(\pi_A))$. A BMS-component I is *internal* if I is contained in one of the chromosomes of A.

Observation 4. *Let $I \subset \pi_A$. Then I is an internal BMS-component iff $I = I(M)$ for some internal component M.*

Proof. Let A' be a uni-chromosomal genome whose single chromosome equals $P(\pi_A)$, i.e., $A' = \{P(\pi_A)\}$. The implied target genome is $\{(0, 1, \ldots, n + 1)\}$. Following [9], $H' = OV(P(\pi_A)) \equiv OV(A', P(\pi_A))$. Thus $H = OV(A, \pi_A)$ is a subgraph of H', where the vertices in $H' \setminus H$ correspond to element pairs $(i, i + 1)$ that are not adjacent in B. (In the example of Fig. 1, those will be the pairs $(0, 1)$, $(4, 5)$ and $(12, 13)$.) Recall that for every BMS-component I there exists a component M in

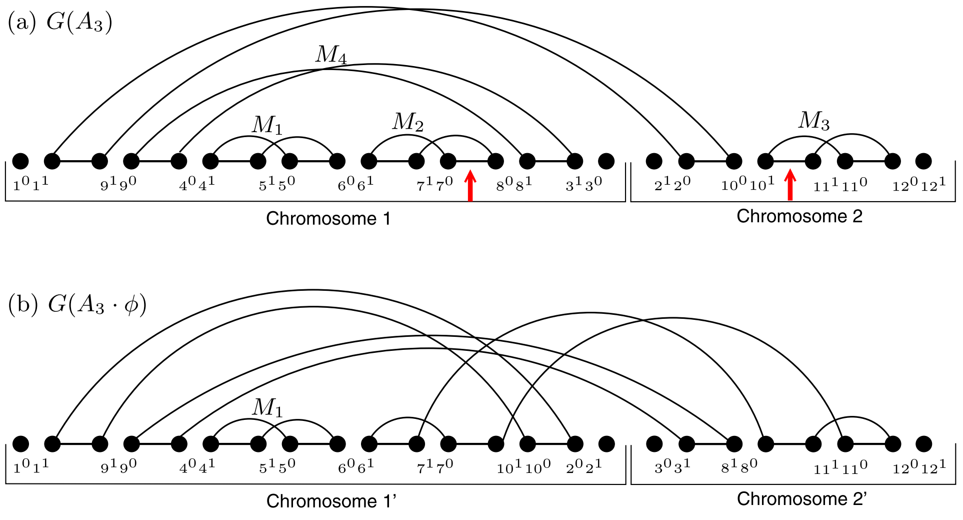

Fig. 2. An example of a bad translocation that eliminates two leaves. (a) The cycle graph $G(A_3) \equiv G(A_3, B_3)$ where $A_3 = \{(1, -9, 4, -5, 6, -7, 8, -3), (-2, 10, -11, 12)\}$ and $B_3 = \{(1, 2), (3, 4, \dots, 12)\}$. The four internal components are designated by $M_1, \dots, M_4$. (b) The cycle graph $G(A_3 \cdot \phi)$, where ϕ is a prefix–suffix translocation cutting the two black edges pointed by the vertical arrows in (a). In $A_3 \cdot \phi$ only one internal component exists, namely M_1. The other internal components, M_2, M_3, and M_4, were eliminated by ϕ.

H' for which $I(M) = I$. Clearly if I is internal then all the vertices in M are internal too, and M is necessarily an internal component in H.

Observe that the vertices that are in $H' \setminus H$ cannot be adjacent to internal vertices in H, since in $G(A')$ the corresponding gray edges are adjacent to black edges bridging across chromosome ends. Therefore, if M is an internal component in H then M is also a component of H' and hence $I(M)$ is an internal BMS-component. □

2.3. The forest of internal components

In this section we present the forest of internal components, originally defined in [4]. Let M' and M'' be two internal components. Then, as discussed in [4], $I(M')$ and $I(M'')$ are either disjoint, nested with different endpoints, or overlapping on one element. We define a *chain* as a sequence of internal components $(M_1, \dots, M_t)$ in which $I(M_j)$ and $I(M_{j+1})$ overlap in exactly one gene for $j = 1, \dots, t-1$. For example, in Fig. 1 let $M' = \{(9, 10), (10, 11)\}$ and $M'' = \{(8, 9)\}$. Then (M', M'') is a chain, as $I(M')$ and $I(M'')$ overlap in one element, which is 9.

For a chain $C = (M_1, \dots, M_t)$ define its associated interval as $I(C) = \bigcup_{j=1}^{t} I(M_j)$. A chain that cannot be extended to the left or right is called *maximal*. The *forest of internal components*, denoted $F(A)$, is defined by the following:

1. The vertices of $F(A)$ are: (i) the non-trivial internal components and (ii) maximal chains that contain at least one non-trivial component.
2. The children of a chain vertex are the non-trivial (internal) components it contains.
3. A chain vertex C is a child of the non-trivial internal component M with the smallest interval $I(M)$ satisfying $I(C) \subset I(M)$. If no such component exists then C is a root of its tree.

See Fig. 1(c) for an example. Observe that each tree in $F(A)$ is contained within one chromosome. For example, the two trees in Fig. 1(c) are contained in chromosome 1. We will refer to a component that is a leaf in $F(A)$ as simply a *leaf*. For example, there are two leaves in Fig. 1(c) corresponding to the intervals $(1, 2, 3)$ and $(-11, 10, -9)$.

Note that if $A = B$ then all the components are trivial and hence $F(A)$ is empty. In addition, $F(A)$ is empty if no non-trivial internal component exists. We say that a non-trivial internal component M is *eliminated* by a translocation ϕ if after ϕ is applied the vertices in M belong to external components. A translocation is called *bad* if $\Delta c = -1$ (i.e. two cycles are merged into one). The following observation describes how non-trivial internal components can be eliminated by bad translocations.

Observation 5. *(See [6,4].) A leaf M is eliminated by performing a translocation that cuts one black edge incident to a gray edge in M and one black edge in another chromosome of A. This translocation is necessarily bad. In addition, all the ancestor components of M in $F(A)$ are eliminated as well.*

An example of a translocation that eliminates two leaf components, with their ancestors, is shown in Fig. 2.

2.4. The translocation distance

Let $T(A)$ and $L(A)$ denote the number of trees and leaves in $F(A)$, respectively. Obviously $T(A) \leqslant L(A)$. Define

$$f(A) = \begin{cases} 2 & \text{if } T(A) = 1 \text{ and } L(A) \text{ is even} \\ 1 & \text{if } L(A) \text{ is odd} \\ 0 & \text{otherwise } (T(A) \neq 1 \text{ and } L(A) \text{ is even}) \end{cases}$$

Theorem 2. *(See [6,4].[2]) The translocation distance between A and B is $d(A) = n - N - c(A) + L(A) + f(A)$.*

An optimal move, i.e., a move that is part of a solution to SRT, is called *valid*.

Lemma 3. *(See [6,4].) $\Delta d = \Delta(-c + L + f) \geqslant -1$. A translocation ϕ is valid iff $\Delta d = -1$.*

A proper translocations is *safe* if it does not create new leaves. The analysis in [6,4] implies that valid translocations are either: (i) bad, or (ii) proper and safe. Bad translocations are valid if $\Delta(L + f) = -2$. As was demonstrated by Bergeron et al. [4] a safe proper translocation may be invalid. However, if there are no leaves, which means that there are no non-trivial internal components, then a safe proper translocation is necessarily valid.

2.5. Analogy to SBR

For the readers familiar with the theory of SBR we now point to the analogy with the SRT theory. The minimum number of reversals needed to sort a signed permutation π (i.e., transform π into the identity permutation) depends on the number of cycles in the cycle graph $G(\pi)$, and on the "unoriented" components in $OV(\pi)$ [8,9]. Unoriented components with minimal intervals are called "hurdles". The sorting of π requires the elimination of all hurdles by *bad reversals*, which decrease the number of cycles by one. If there are no hurdles, then π can be sorted by *proper reversals*, which increase the number of cycles by one. Thus there exists an analogy between the two distance formulas, of SBR and SRT. In particular, the parameter L, which indicates the number of leaves, is analogous to the parameter h, which indicates the number of hurdles.

The elimination of all hurdle components can be done linear time [9,1], and is commonly performed at the beginning of the sorting algorithm. Thus SBR is linearly reduced to a simpler variant, "SBR-no hurdles". Most algorithms for SBR focus on solving this reduced form of SBR.

In the following we show that SRT can be reduced to "SRT-no leaves" in a similar manner, by eliminating all leaves in linear time. In addition, the algorithm we present in Section 4 for "SRT-no leaves" is an adaptation of an algorithm for "SBR-no hurdles". In [14] we show that two additional algorithms for "SBR-no hurdles" can be adapted to solve the "SRT-no leaves".

3. A linear reduction of SRT to SRTNL

A large part of the difficulty in analyzing the translocation distance (Theorem 2) is due to leaves: when there are no leaves $f(A) = L(A) = 0$ and the distance formula is much simpler. Motivated by this observation, we define SRTNL ("SRT-no leaves") as a special case of SRT when there are no leaves (i.e. $L(A) = T(A) = 0$). In this section we present a generic algorithm for solving SRT, using an algorithm for SRTNL. This algorithm, apart from two calls for solving an SRTNL instance, can be implemented in linear time.

Let $L(X)$ denote the number of leaves in chromosome X. Let $N^{\mathrm{L}}(A)$ denote the number of chromosomes of A containing at least one leaf. Equivalently, $N^{\mathrm{L}}(A)$ is the number of chromosomes for which $L(X) > 0$. The sorting of genome A into B requires the elimination of all leaves. The following lemmas describe how to eliminate leaves by valid (bad) translocations.

Lemma 4. *Suppose $N^{\mathrm{L}}(A) \geqslant 2$. Then there exists a valid bad translocation ϕ satisfying:*

(i) *$\Delta L = -2$, and*
(ii) *if $L(A \cdot \phi) \geqslant 2$ then $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$.*

Proof. Assume $N^{\mathrm{L}}(A) \geqslant 2$. First, we prove that any bad translocation ϕ satisfying (i) and (ii) is necessarily valid. The parity of L is the same in A and in $A \cdot \phi$ and hence $\Delta f = 0$ ($f = 1$ if L is odd, and $f = 0$ otherwise). Therefore $\Delta d = \Delta(-c + L + f) = 1 - 2 + 0 = -1$ and ϕ is valid.

We shall now prove that there exists such a bad translocation. Choose $X_1, X_2 \in A$ such that $L(X_1) + L(X_2)$ is maximal. Suppose $L(X_1) \geqslant L(X_2)$.

[2] The formulas in [4] and [6] are equivalent: a leaf component is equivalent to a "minimal subpermutation" (minSP in short); the parameter s in [6], which denotes the number of minSPs, is equivalent to L; the term $(o + 2i)$ in [6] is equivalent to f.

Case 1. $L(X_1) \geqslant 2$ and $L(X_2) \geqslant 2$. Let ϕ be a (bad) prefix–prefix translocation that eliminates the second leaf from the left in X_1 and X_2 (Observation 5). Then each of the new chromosomes in $A \cdot \phi$ contains at least one leaf and hence $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$.

Case 2. $L(X_1) \geqslant 2$ and $L(X_2) = 1$. Let ϕ be a (bad) prefix–prefix translocation that eliminates the second leaf from the left in X_1 and the leaf in X_2. Then at least one of the new chromosomes in $A \cdot \phi$ contains exactly one leaf. If $L(A \cdot \phi) \geqslant 2$ then there must be another chromosome in $A \cdot \phi$ that contains at least one leaf and hence $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$.

Case 3. $L(X_1) = L(X_2) = 1$. Let ϕ be a (bad) translocation that eliminates the two leaves in X_1 and X_2. Clearly in $A \cdot \phi$ every chromosome contains at most one leaf. Hence, if $L(A \cdot \phi) \geqslant 2$ then $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$. □

The following lemma follows from the proof of Theorem 13 in [6], and is proven here for completion.

Lemma 5. *Suppose $N^{\mathrm{L}}(A) = 1$, $L(A) \geqslant 2$, and $f(A) > 0$. Let ϕ be a (prefix–prefix) translocation that eliminates the second leaf from the left in A. Then ϕ is valid. In addition, if $L(A \cdot \phi) \geqslant 2$ then $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$.*

Proof. Clearly $\Delta(-c + L) = 1 - 1 = 0$. If $L(A \cdot \phi) = 1$ then $L(A) = 2$ and $T(A) = 1$ and thus $\Delta f = -1$ and ϕ is valid.

Suppose $L(A \cdot \phi) \geqslant 2$. Let X' be the chromosome containing all the leaves in A, and let X'' be the second chromosome on which ϕ acts. Then in genome $A \cdot \phi$: $L(X'') = 1$ and $L(X') > 0$, thus $N^{\mathrm{L}}(A \cdot \phi) \geqslant 2$. In particular $T(A \cdot \phi) > 1$ and $L(A \cdot \phi) = L(A) - 1$, so $\Delta f = -1$ and ϕ is valid. □

Suppose there are several trees that are all located in one chromosome, i.e., $N^{\mathrm{L}}(A) = 1$, but $T(A) > 1$. To be able to eliminate a pair of leaves by one (bad) translocation, we first need to perform a sequence of (valid) proper translocations that "separates" the trees (and hence the leaves) into two different chromosomes. In the following we describe how to find such a sequence. We say that a sequence of translocations *sorts* a component M, if after performing the sequence every gray edge in M becomes an adjacency.

Lemma 6. *There is a sequence of safe proper translocations that sorts all external components* (*internal components are unchanged*).

Proof. For an interval of genes $I = (i_1, \ldots, i_k)$ let $IN(I) = \{i_2, \ldots, i_{k-1}\}$. Let $S = \{i \mid i \in IN(I)$, where I is an interval corresponding to a tree$\}$. For example, in Fig. 1, $S = \{2, 8, 9, 10, 11\}$. Define A' and B' as the genomes obtained from A and B respectively after the deletion of the genes in S. Note that after a gene is deleted from a genome, its two neighbors become adjacent. Thus any interval corresponding to a tree of A is replaced in A' by a pair of genes forming an adjacency. Therefore $G(A')$ contains no leaves. Thus there is a sequence of safe proper translocations that sorts A' into B' (Theorem 2). This sequence induces a sequence of safe proper translocations on A that sorts all the external components in $G(A)$. □

We call a translocation ϕ *separating* if $N^{\mathrm{L}}(A) = 1$ and $N^{\mathrm{L}}(A \cdot \phi) = 2$. The following lemma shows how to find a sequence of valid proper translocations, whose last translocation is separating.

Lemma 7. *Suppose $N^{\mathrm{L}}(A) = 1$ and $T(A) > 1$. Let $S = (\phi_1, \ldots, \phi_k)$ be a sequence of safe proper translocations that sorts all the external components in $G(A)$. Then S contains a separating translocation ϕ_l, $l \in \{1, \ldots, k\}$. Moreover, $S_l = (\phi_1, \ldots, \phi_l)$ is a sequence of valid translocations.*

Proof. Apply the translocations in S by their order. Let $A_0 = A$ and let A_i be the genome obtained after applying $(\phi_1, \ldots, \phi_i)$ to A. Suppose that S does not contain a separating translocation. Thus, by our assumption $N^{\mathrm{L}}(A_i) = 1$ for $i = 1, \ldots, k$. Observe that a chromosome that contains two trees necessarily contains the endpoint of an external edge. Thus $T(A_k) = 1$, since in A_k there are no external edges and all the leaves belong to one chromosome. Since $T(A) > 1$, there exists $\phi_t \in S$ such that $T(A_{t-1}) > 1$ and $T(A_t) = 1$. Now, ϕ_t is a safe proper translocation and hence does not eliminate any internal component, thus A_{t-1} must contain two trees in two different chromosomes. Therefore $N^{\mathrm{L}}(A_{t-1}) > 1$, a contradiction.

Thus there exists i for which $N^{\mathrm{L}}(A_i) > 1$. Let l be the first index for which $N^{\mathrm{L}}(A_l) > 1$. Then ϕ_l is a separating translocation. As S_l contains only safe proper translocations $L(A_l) = L(A)$ and thus $f(A_l) = f(A)$. Hence $d(A_l) - d(A) = l$ and thus every translocation in S_l is valid. □

Lemmas 4–7 motivate Algorithm 1 for SRT. This algorithm focuses on the efficient and optimal elimination of all leaf components. If all the leaves belong to one chromosome, then we either use Lemma 5 or Lemma 7 to separate the leaves into two chromosomes. Then we use Lemma 4 to eliminate pairs of leaves. At the end, either all leaves have been eliminated, or we are left with a single leaf, which is eliminated by one (valid) bad translocation.

Lemma 8. *Algorithm 1, excluding the two calls to a SRTNL algorithm, can be implemented in linear time.*

Algorithm 1 An algorithm for solving SRT using an algorithm for SRTNL.

```
1: if $N^L = 1$ and $L \geqslant 2$ then
2:   if $f > 0$ then
3:     Eliminate the second leaf from the left by a prefix–prefix translocation /*Lemma 5*/
4:   else
5:     Compute a sequence $S$ of safe proper translocations that sorts all external components /*using an algorithm for SRTNL, Lemma 6*/
6:     Iteratively perform the translocations in $S$ until $N^L > 1$ /*Lemma 7*/
7:   end if
8: end if
9: Let $Q_1$ be the list of chromosomes containing exactly one leaf
10: Let $Q_2$ be the list of chromosomes containing at least two leaves
11: while $L > 0$ do
12:   if $L = 1$ then
13:     Eliminate the single leaf by a prefix–prefix translocation
14:   else
15:     for $i = 1, 2$ do
16:       if $Q_2 \neq \emptyset$ then
17:         $X_i \leftarrow$ an element from $Q_2$. Remove $X_i$ from $Q_2$
18:         $l_i \leftarrow$ the second leaf from the left in chromosome $X_i$
19:       else
20:         $X_i \leftarrow$ an element from $Q_1$. Remove $X_i$ from $Q_1$
21:         $l_i \leftarrow$ the single leaf in $X_i$
22:       end if
23:     end for
24:     Eliminate $l_1$ and $l_2$ by a prefix–prefix translocation /*Lemma 4*/
25:     for $i = 1, 2$ do
26:       if $L(X_i) \geqslant 2$ then
27:         add $X_i$ to $Q_2$
28:       else if $L(X_i) = 1$ then
29:         add $X_i$ to $Q_1$
30:       end if
31:     end for
32:   end if
33: end while /*Invariant: $N^L \geqslant 2$ or $L = 1$ */
34: Solve SRTNL on $A$
```

Proof. The computation of all the parameters can be done in linear time, in a similar manner to the computation of the translocation distance [4].

Steps 5 and 6 are implemented by calling a procedure for SRTNL. However, we need to stop this procedure when a separating translocation is applied. We can locate this separating procedure in linear time by acting as follows. Suppose that $N^L = 1$, $T > 1$ and $S = (\phi_1, \ldots, \phi_k)$ is a sequence of safe proper translocations that sorts all the external components. By Lemma 7 there exists a separating translocation ϕ_l in S. Let I be the minimum interval of genes that contains the intervals of all the leaves. We say that a translocation ϕ *cuts* I if one of the black edges it cuts is contained in I. Note that since I is contained in a single chromosome, a translocation cuts at most one black edge in I. Clearly ϕ_l cuts I. On the other hand, the first translocation that cuts I is necessarily separating. For every translocation ϕ_i in S we can test in $O(1)$ time whether it cuts I.

We implement Steps 11–33 in linear time, as follows. For each chromosome we maintain its genes and the leaves it contains in two ordered linked lists. We use only prefix–prefix (bad) translocations that do not change the signs of the translocated genes. Thus the update of the genes and leaves lists of the chromosomes after a translocation is done in $O(1)$. □

Lemma 8 immediately implies:

Theorem 9. *SRT is linearly reducible to SRTNL.*

4. An algorithm for SRTNL

In this section we present an algorithm for SRTNL. We first describe how the overlap graph is changed after performing a chromosome flip or a proper translocation defined by an external vertex.

As was demonstrated by Hannenhalli and Pevzner [7], a reversal on π_A simulates a translocation on A:

$$(\ldots, X_1, \underline{X_2, \ldots, Y_1}, Y_2, \ldots) \quad \Rightarrow \quad (\ldots, X_1, \underline{-Y_1, \ldots, -X_2}, Y_2, \ldots).$$

The type of translocation depends on the relative orientation of X and Y in π_A (and not on their order): if the orientation is the same, then the translocation is prefix–suffix, otherwise it is prefix–prefix. The segment between X_2 and Y_1 may contain additional chromosomes that are flipped and thus unaffected.

4.1. Updating OVCH for chromosome flips and proper translocations

Suppose $H_1 = OVCH(A, \pi_1)$ and $H_2 = OVCH(A, \pi_2)$, where π_1 and π_2 are two different concatenations and orientations of the chromosomes in A. In this case we refer to H_1 and H_2 as *equivalent*.

Let $H = OVCH(A, \pi_A)$. Let $IN(H)$ denote the set of vertices that are in non-trivial internal components. Thus two equivalent graphs, H_1 and H_2, satisfy $IN(H_1) = IN(H_2)$ (Observation 3).

Let v be any vertex in H. Denote by $CH(v) \equiv CH(v, H)$ the set of chromosomes that are neighbors of v in H. Hence if v is external then $|CH(v)| = 2$, otherwise $CH(v) = \emptyset$ (compare Fig. 1(b)). For a chromosome X, let $\phi(X)$ denote a flip of chromosome X in π_A. Let $H \cdot \phi(X) = OVCH(A, \pi_A \cdot \phi(X))$. Hence, in particular $H \cdot \phi(X)$ and H are equivalent.

Lemma 10. *(See [14].) $H \cdot \phi(X)$ is obtained from H by complementing the subgraph induced by the set $\{u\colon\ X \in CH(u)\}$ and flipping the orientation of every vertex in it.*

Let v be an external vertex in H. Denote by $\phi(v)$ the proper translocation that the corresponding gray edge defines on A (recall Observation 1). Two external vertices v_1 and v_2 in H are *equivalent* if they define the same translocation, i.e. $\phi(v_1) \equiv \phi(v_2)$.

A vertex in the overlap graph is *oriented* if its corresponding edge connects two genes with different signs in π_A, otherwise it is *unoriented*. If v is an oriented external vertex then $\phi(v)$ can be mimicked by a reversal, $\hat{\phi}(v)$, on π_A.

For an external vertex v we define $H \cdot \phi(v)$ in the following way. If v is oriented then $H \cdot \phi(v) = OVCH(A \cdot \phi(v), \pi_A \cdot \hat{\phi}(v))$. Otherwise, suppose $CH(v) = \{X, Y\}$ and that Y appears after X in π_A. Then v is an oriented external vertex in $H' = H \cdot \phi(X)$ and thus we define $H \cdot \phi(v) = H' \cdot \phi(v)$.

Denote by $N(v) \equiv N(v, H)$ the set of vertices that are neighbors of v, including v itself (but not including chromosome neighbors). Given two sets S_1 and S_2 define $S_1 \oplus S_2 = (S_1 \cup S_2) \setminus (S_1 \cap S_2)$. Finally, two chromosomes in $OVCH(A, \pi_A)$ are called *consecutive* if they are consecutive in π_A.

Lemma 11. *(See [14].) Let v be an oriented external vertex in H and suppose the chromosomes in $CH(v)$ are consecutive. Then $H \cdot \phi(v)$ is obtained from H by the following operations*:

(i) *Complement the subgraph induced by $N(v)$ and flip the orientation of every vertex in $N(v)$.*
(ii) *For every vertex $u \in N(v)$ update the edges between u and $CH(u) \cup CH(v)$ such that $CH(u) = CH(u) \oplus CH(v)$. In particular, the external/internal state of a vertex $u \in N(v)$ is flipped iff u is internal or $CH(u) = CH(v)$.*

Lemmas 10 and 11 describe the change in $OVCH(A, \pi_A)$ after performing operations that can be mapped to reversals on π_A. Therefore, the described change in $OVCH(A, \pi_A)$ is similar to the change in $OV(\pi)$ after performing a reversal [9, Observation 4.1].

4.2. The main theorem and algorithm

We now describe the main theorem and algorithm. Our algorithm is formally very similar to the algorithm for SBR presented in [15]. Instead of performing reversals on oriented edges in [15], we perform translocations on external edges. Despite of the great similarity between the algorithms our validity proof is completely new. We analyze an overlap graph with chromosomes of a multi-chromosomal genome, while [15] analyze the overlap graph of a uni-chromosomal genome. Like [15], we perform operations defined by oriented vertices (i.e. translocations). However, in our case these vertices must also be external. If an external vertex is unoriented, we can turn it into an oriented vertex by a flip of a chromosome. Hence, we consider two types of operations in our analysis.

A sequence of vertices $S = (v_1, \ldots, v_k)$ from H is *legal* if v_j is external in $H \cdot \phi(v_1) \cdots \phi(v_{j-1})$ for $j = 1, \ldots, k$. For a legal sequence S define $\phi(S) = \phi(v_1) \cdots \phi(v_k)$. A legal sequence S is *total* if $H \cdot \phi(S)$ contains only trivial components. For an overlap graph with chromosomes H_1, let $EXT(H_1)$ denote the set of vertices that are in external components. If S is a maximal legal sequence of vertices in H then $EXT(H \cdot \phi(S)) = \emptyset$. If in addition S is not total then $IN(H \cdot \phi(S)) \neq \emptyset$.

Theorem 12. *Let $S = (v_1, \ldots, v_k)$ be a maximal legal but not total sequence of vertices in H. Let $IN = IN(H \cdot \phi(S))$. Let v_l be the first vertex in S satisfying $IN(H \cdot \phi(v_1, \ldots, v_l)) = IN$, i.e. $\phi(v_l)$ is the last unsafe translocation in $\phi(S)$. Let $S_1 = (v_1, \ldots, v_{l-1})$ and $S_2 = (v_l, \ldots, v_k)$. Then every maximal sequence of vertices $S' = (w_1, \ldots, w_m)$ in IN that satisfies*

(i) *(S_1, S') is legal, and*
(ii) *v_l is not an adjacency in $H \cdot \phi(S_1, S')$ also satisfies:*
(iii) *S' is not empty, and*
(iv) *(S_1, S', S_2) is a maximal legal sequence.*

Moreover, all the translocations in $\phi(S_2)$ are safe.

Proof. Let $v = v_l$, $H_0 = H \cdot \phi(S_1)$ and $IN_0 = EXT(H_0) \cap IN$. Then $IN_0 \neq \emptyset$ and none of the vertices in IN_0 is equivalent to v in H_0 (otherwise it would be an adjacency in $H \cdot \phi(S)$ and hence not in IN). Hence S' is not empty. Let $A_0 = A \cdot \phi(S_1)$ and $CH(v) = \{X, Y\}$. We choose π_0 to be a concatenation of the chromosomes in A_0 in which X and Y are the first two chromosomes. We can assume w.l.o.g. that $H = OVCH(A, \pi_0)$, hence $H_0 = OVCH(A_0, \pi_0)$. For $j = 1, \ldots, m$ let $H_j = H_0 \cdot \phi(w_1, \ldots, w_j)$. Let $IN_j = EXT(H_j) \cap IN$. Then for $j = 1, \ldots, m$: (i) $w_j \in IN_{j-1}$ and (ii) w_j is not equivalent to v in H_{j-1}. Let $EXT = EXT(H_0 \cdot \phi(v))$. The following conditions hold for H_j when $j = 0$ (see Fig. 3(a)):

(1) The subgraphs of $H_j \cdot \phi(v)$ and $H_0 \cdot \phi(v)$ that are induced by EXT are equivalent.
(2) Every $w \in IN_j$ satisfies: $CH(w) = CH(v) = \{X, Y\}$.
(3) If v is oriented then $N(v) \cap IN = IN_j$.
(4) All the possible edges exist between $N(v) \cap EXT$ and IN_j.
(5) There are no edges between $IN \setminus IN_j$ and vertices outside IN.
(6) There are no edges between $EXT \setminus N(v)$ and vertices outside EXT.

We shall prove below that in H_m v is external and that all the above conditions are satisfied. The first condition ensures that (S_1, S', S_2) is legal. The rest of the conditions ensure that $H_m \cdot \phi(v)$ satisfies: (i) there are no external vertices in IN and (ii) there are no edges between EXT and vertices outside EXT. Hence (S_1, S', S_2) is maximal and every translocation in $\phi(v_{l+1}, \ldots, v_k)$ is safe. $\phi(v_l)$ is safe in H_m since S' is maximal. Therefore, all the translocations in $\phi(S_2)$ are safe.

Assume that v is external in H_j and that all the above conditions hold for a certain j. Since these conditions are true for every graph that is equivalent to H_j we can assume that v is oriented. We now prove, using induction on j, that these conditions are satisfied for every H_i, $i \in \{1, \ldots, m\}$, in which v is external, and that v is external in H_m.

Case 1. w_{j+1} is oriented in H_j. Let $H_{j+1} = H_j \cdot \phi(w_{j+1})$ (see Fig. 3(b)). Then $IN_{j+1} = N(v, H_j) \oplus N(w_{j+1}, H_j)$. $IN_{j+1} \neq \emptyset$, otherwise v is an isolated internal vertex in H_{j+1} and hence equivalent to w_{j+1} in H_j. Hence $m \geqslant j + 2$.

Case 1(a). w_{j+2} is oriented in H_{j+1}. Let $H_{j+2} = H_{j+1} \cdot \phi(w_{j+2})$ (see Fig. 3(c)). Clearly, v is external in H_{j+2}. Let $M = N(v, H_j) \cap EXT$. Then $N(w_{j+2}, H_{j+1}) \cap EXT = N(w_{j+1}, H_j) \cap EXT = M$. Hence the subgraphs of H_{j+2} and H_j that are induced by M are identical and the first condition is satisfied in H_{j+2}.

Case 1(b). w_{j+2} is unoriented in H_{j+1}. Let $H'_{j+1} = H_{j+1} \cdot \phi(X)$ (H'_{j+1} and H_{j+1} are equivalent) (see Fig. 3(d)). Hence w_{j+2} is oriented in H'_{j+1}. Note that v is an internal vertex in H'_j. Let $M' = N(w_{j+1}, H'_{j+1}) \cap EXT$. Let $H_{j+2} = H'_{j+1} \cdot \phi(w_{j+2})$ (see Fig. 3(e)). v is an oriented external vertex in H_{j+2} and $N(v, H_{j+2}) \cap EXT = M'$. Therefore, the two subgraphs of $H_{j+2} \cdot \phi(v)$ (see Fig. 3(f)) and H'_{j+1} (see Fig. 3(d)) that are induced by EXT are identical. The subgraphs of H_{j+1} and $H_j \cdot \phi(v)$ that are induced by EXT are also identical. Hence, the first condition is satisfied.

Looking at Figs. 3(c) and 3(e) it is easy to verify that the rest of the conditions are also satisfied for H_{j+2}.

Case 2. w_{j+1} is unoriented in H_j. We define the three subsets of vertices $M_1, M_2, M_3 \subset EXT$ in H_j as follows:

(1) M_1 is the set of neighbors of w_{j+1} (equivalently, v) that are either internal or external but does not overlap chromosome X.
(2) M_2 is the set of neighbors of w_{j+1} (equivalently, v) that overlap chromosome X. Hence $M_1 \cup M_2 = N(v, H_j) \cap EXT$.
(3) M_3 is the set of vertices that overlap chromosome X but are not neighbors of w_{j+1} (equivalently, v).

For an illustration of H_j see Fig. 3(g). Let $H'_j = H_j \cdot \phi(X)$ (see Fig. 3(h)). In H'_j: w_{j+1} is an oriented external vertex and is not a neighbor of v. Let $H_{j+1} = H'_j \cdot \phi(w_{j+1})$ (see Fig. 3(i)). Obviously, v remains intact in H_{j+1}. Let $H'_{j+1} = H_{j+1} \cdot \phi(X)$ (see Fig. 3(j)). Then, the subgraphs of $H'_{j+1} \cdot \phi(v)$ (see Fig. 3(k)) and $H_j \cdot \phi(v)$ that are induced by M_1, M_2 and M_3 are equivalent (compare the subgraph induced by EXT in H_j in Fig. 3(g) with the subgraph induced by EXT in $H'_{j+1} \cdot \phi(v) \cdot \phi(X)$ in Fig. 3(l)). Hence the first condition is satisfied. Looking at Fig. 3(i), it is easy to verify that conditions (2)–(6) hold for H_{j+1}. □

Algorithm 2 builds a sequence of gray edges in $G(A)$, (S_1, S_2), that corresponds to a total legal sequence of vertices from H. The sequence (S_1, S_2) is built by a repeated application of Theorem 12. It greedily removes external edges in $G(A)$ from an allowed subset and performs the corresponding translocations (step 2(a)). When the allowed subset contains only internal gray edges, the algorithm repeats the last translocations in a reverse order (thereby canceling them) until another vertex in the allowed subset becomes external (step 2(b)). Fig. 4 describes an example of a run of the algorithm. Every translocation in the algorithm is applied at most twice and so the algorithm performs at most $2n$ translocations.

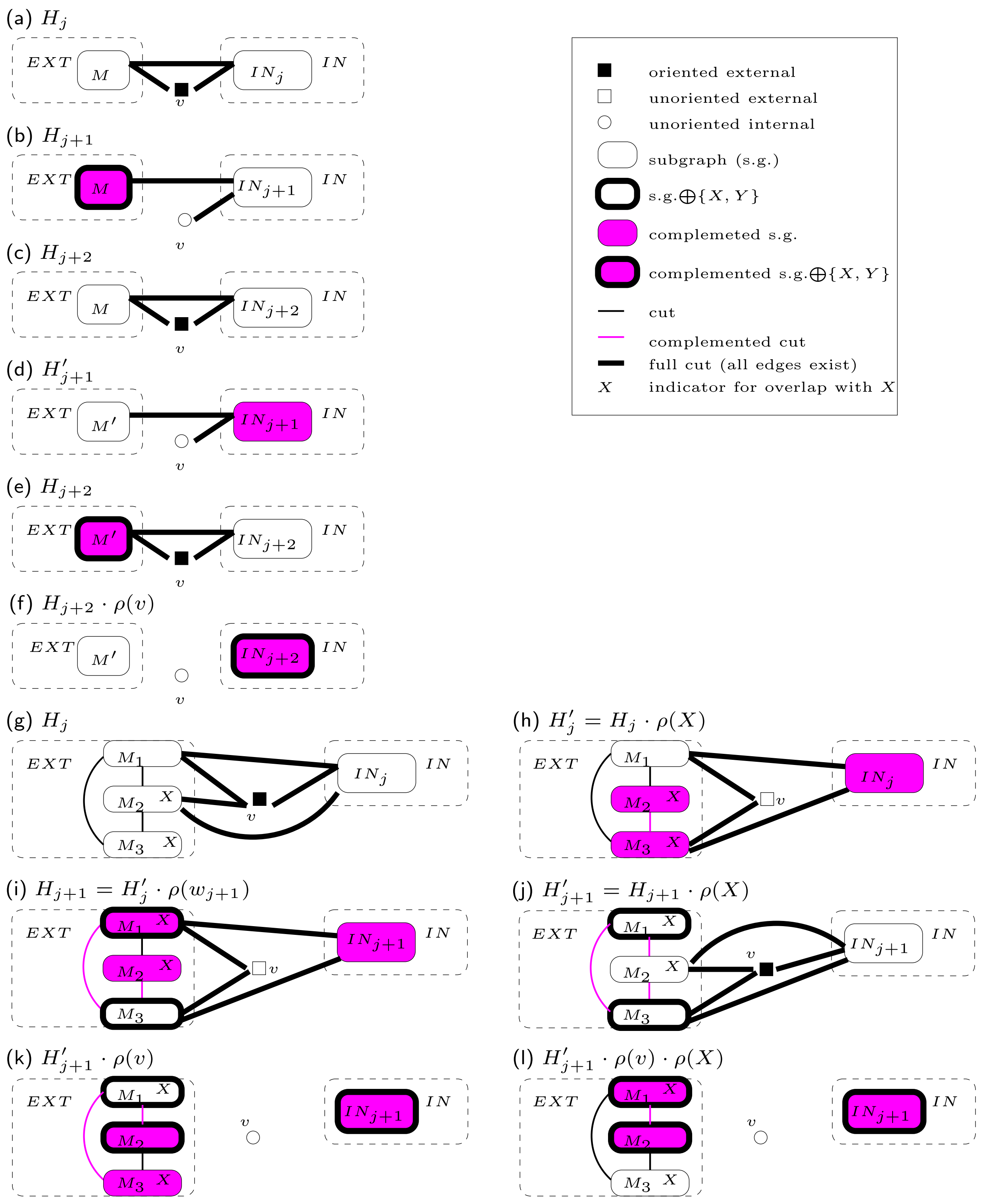
(a) H_j
(b) H_{j+1}
(c) H_{j+2}
(d) H'_{j+1}
(e) H_{j+2}
(f) $H_{j+2} \cdot \rho(v)$
(g) H_j
(h) $H'_j = H_j \cdot \rho(X)$
(i) $H_{j+1} = H'_j \cdot \rho(w_{j+1})$
(j) $H'_{j+1} = H_{j+1} \cdot \rho(X)$
(k) $H'_{j+1} \cdot \rho(v)$
(l) $H'_{j+1} \cdot \rho(v) \cdot \rho(X)$
EXT
IN
M
M'
M_1
M_2
M_3
X
IN_j
IN_{j+1}
IN_{j+2}
v
oriented external
unoriented external
unoriented internal
subgraph (s.g.)
s.g.$\bigoplus\{X, Y\}$
complemeted s.g.
complemented s.g.$\bigoplus\{X, Y\}$
cut
complemented cut
full cut (all edges exist)
X indicator for overlap with X

Fig. 3. Illustrations for the proof of Theorem 12.

5. An $O(n^{3/2}\sqrt{\log(n)})$ time implementation of the algorithm

Algorithm 2 can be implemented in $O(n^2)$ time in a relatively simple manner. We provide below an $O(n^{3/2}\sqrt{\log(n)})$ algorithm. The implementation follows closely the ideas of [10] and [15].

Algorithm 2 An algorithm for solving SRTNL.

```
1: Let $V$ be the set of gray edges in $G(A)$ that are in non-trivial components
2: $S_1 = S_2 = \emptyset$
3: $\Phi = \emptyset$
4: while $V \neq \emptyset$ do
5:   while there exists an external gray edge $v \in V$ in $G(A)$ do
6:     Remove $v$ from $V$
7:     if $v$ is not equivalent to the first element in $S_2$ then
8:       Append $v$ to $S_1$
9:       Append $\phi(v)$ to $\Phi$
10:      $A \leftarrow A \cdot \phi(v)$
11:    end if
12:  end while
13:  if $V = \emptyset$ then
14:    return $\phi(S_1, S_2)$
15:  end if
16:  while all the gray edges in $V$ are internal in $G(A)$ do
17:    Let $v$ be the last gray edge in $S_1$. Remove $v$ from $S_1$
18:    Prepend $v$ to $S_2$
19:    Let $\phi$ be the last translocation in $\Phi$. Remove $\phi$ from $\Phi$
20:    $A \leftarrow A \cdot \phi$
21:  end while
22: end while
```

Genome A	S_1	S_2	V
$(-8, -2, \underline{7, 3}), (\underline{1}, 6, 5, -4)$	$\emptyset$	$\emptyset$	$1, 2, 4, 5, 6, 7$
$(\underline{-8}, -2, -1), (\underline{-3}, -7, 6, 5, -4)$	1	$\emptyset$	$2, 4, 5, 6, 7$
$(\underline{-3}, -2, -1), (\underline{-8}, -7, 6, 5, -4)$	1, 2	$\emptyset$	4, 5, 6, 7
$(-8, -2, -1), (-3, -7, 6, 5, -4)$	1	2	4, 5, 6, 7
$(-8, -2, \underline{-1}), (\underline{-3, -7}, 6, 5, -4)$	1	2	4, 5, 6
$(\underline{-8, -2}, 7, 3), (\underline{1, 6}, 5, -4)$	$\emptyset$	1, 2	4, 5, 6
$(\underline{1}, 6, 7, 3), (\underline{-8, -2, 5}, -4)$	6	1, 2	4, 5
$(-8, -2, 5, 6, 7, 3), (1, -4)$	6, 5	1, 2	4
$(-8, -2, \underline{5, 6, 7, 3}), (\underline{1}, -4)$	6, 5	1, 2	$\emptyset$
$(\underline{-8}, -2, -1), (\underline{-3}, -7, -6, -5, -4)$			
$(-3, -2, -1), (-8, -7, -6, -5, -4)$			

Fig. 4. An example for a run of the algorithm on genomes $A = \{(-8, -2, 7, 3), (1, 6, 5, -4)\}$ and $B = \{(1, 2, 3), (4, \ldots, 8)\}$. A gray edge $(i, i+1)$ (vertex of H) is represented by i. The underlined segments denote a translocation the algorithm chose. The algorithm ends when $V = \emptyset$. The top 9 lines describe the steps of the algorithm. The two bottom lines show the application of $\phi(S_2) = \phi(1, 2)$ on the final genome produced by the algorithm, producing B.

We identify a gray edge $(i, i+1)$ by i and refer to $(i+1)$ as the *remote end* of i. The data structure we use for maintaining the genome A is as follows:

(1) A doubly linked list of $O(\sqrt{\frac{n}{\log(n)}})$ blocks. We partition π_A into continuous blocks such that the size of every block is at least $\frac{1}{2}\sqrt{n\log(n)}$ and at most $2\sqrt{n\log(n)}$.
(2) A balanced search tree for every block. The tree contains the edges in the block ordered by the positions of their remote ends. We use balanced trees that support split and concatenate operations in logarithmic time, such as red–black trees or 2–4 trees. We use $T[v]$ to denote the subtree rooted at v and containing all its descendants.
(3) An n-array of block pointers. The ith entry in the array points to the block containing i.

We add the following fields to the above data structure:

(1) For each edge we keep an external-bit. If the external-bit is *on* then the edge is external, otherwise it is internal.
(2) For each block we keep the following fields: (i) a counter of external edges in V, (ii) a counter of chromosomes' left tails, and (iii) a reverse-flag. If the reverse-flag of a block is *on* then the order and signs of the elements in the block are reversed.
(3) For every subtree $T[v]$ of each block's search tree we keep the following fields in its root v: (i) counters of external and internal edges in V, (ii) a direction-flip-flag, and (iii) an external-flip-flag. If the external-flip-flag of a vertex v is *on* then in $T[v]$ the external-bits of all the elements are flipped and the counters of internal and external elements from V exchange their values. If the direction-flip-flag of a vertex v is *on* then in $T[v]$ the order of the elements is reversed.

Table 1
The subtrees for which the external-flip-flag is flipped as a function of translocation type and block type.

Block	X_1	X_2	Y_1	Y_2
Prefix–prefix	X_2, Y_2	X_1, Y_1	X_2, Y_2	X_1, Y_1
Prefix–suffix	X_2, Y_1	X_1, Y_2	X_1, Y_2	X_2, Y_1

We can clear the direction-flip-flag of a node by reversing the order of its children and flipping the direction-flip-flag in each of them. We can clear the external-flip-flag in a node by exchanging the values of the counters of external and internal edges in V, flipping the external-flip-flag in each of its children and flipping the external-bit of the element residing at the node. One can view this procedure as "pushing down" the flags. An direction-flip-flag and an external-flip-flag that are *on* are "pushed down" whenever $T[v]$ is searched.

We implement the algorithm using the above data structures. A search for an external edge in V is done as follows. We traverse the list of blocks until we reach a block that contains external edges from V. We then search the tree of the block for an external edge i. We locate element $i+1$ (the remote end of edge i) using the n-array and a search of its block.

Let ϕ be a translocation on A operating on the chromosomes $X=(X_1,X_2)$ and $Y=(Y_1,Y_2)$. Then ϕ is performed in $O(\sqrt{n\log(n)})$ time as follows:

(1) Split at most six blocks so that each of the four segments X_1, X_2, Y_1 and Y_2 corresponds to a union of blocks. If ϕ is a prefix–prefix translocation exchange the blocks of X_1 and Y_1. Otherwise, reverse the order and flip the reverse-flags of the blocks of X_2 and Y_1 and then exchange the blocks of X_2 and Y_1.
(2) We now have to modify the trees of each block to reflect the order and direction changes. This is done as follows. Traverse all the blocks and for each block:
  (a) Let T be the balanced search tree of the block. If ϕ is a translocation on an edge i in V and i is contained in the block: decrease by 1 the counters of external edges in V of the block and of every node in T that contains i in its subtree.
  (b) Split T into at most seven subtrees such that each of the segments X_1, X_2, Y_1 and Y_2 has a corresponding subtree.
  (c) If the block corresponds to a segment of X_1, X_2, Y_1 and Y_2 flip the external-flip-flag at the roots of two subtrees according to Table 1.
  (d) If ϕ is a prefix–prefix translocation, exchange the subtrees of X_1 and Y_1. Otherwise, exchange the subtrees of X_2 and Y_1 and flip the direction-flip-flags of both.
  (e) Concatenate the seven subtrees into T.
(3) If necessary, concatenate small blocks and split large blocks such that the size of each block is at least $\frac{1}{2}\sqrt{n\log(n)}$ and at most $2\sqrt{n\log(n)}$.

Theorem 13. *SRTNL can be solved in $O(n^{3/2}\sqrt{\log(n)})$.*

Acknowledgements

This study was supported in part by the Raymond and Beverly Sackler chair in Bioinformatics and by the Israel Science Foundation (grant No. 802/08).

References

[1] D.A. Bader, B.M.E. Moret, M. Yan, A linear-time algorithm for computing inversion distance between signed permutations with an experimental study, Journal of Computational Biology 8 (5) (2001) 483–491.
[2] A. Bergeron, A very elementary presentation of the Hannenhalli–Pevzner theory, Discrete Applied Mathematics 146 (2) (2005) 134–145.
[3] A. Bergeron, J. Mixtacki, J. Stoye, Reversal distance without hurdles and fortresses, in: Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM), in: LNCS, vol. 3109, Springer, 2004, pp. 388–399.
[4] A. Bergeron, J. Mixtacki, J. Stoye, On sorting by translocations, Journal of Computational Biology 13 (2) (2006) 567–578.
[5] P. Berman, S. Hannenhalli, Fast sorting by reversal, in: Proceedings of the 7th Annual Symposium Combinatorial Pattern Matching (CPM), in: LNCS, vol. 1075, Springer, 1996, pp. 168–185.
[6] S. Hannenhalli, Polynomial algorithm for computing translocation distance between genomes, Discrete Applied Mathematics 71 (1996) 137–151.
[7] S. Hannenhalli, P. Pevzner, Transforming men into mice (polynomial algorithm for genomic distance problems), in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, 1995, pp. 581–592.
[8] S. Hannenhalli, P. Pevzner, Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals, Journal of the ACM 46 (1999) 1–27.
[9] H. Kaplan, R. Shamir, R.E. Tarjan, Faster and simpler algorithm for sorting signed permutations by reversals, SIAM Journal of Computing 29 (3) (2000) 880–892.
[10] H. Kaplan, E. Verbin, Sorting signed permutations by reversals, revisited, Journal of Computer and System Sciences 70 (3) (2005) 321–341.
[11] J.D. Kececioglu, R. Ravi, Of mice and men: Algorithms for evolutionary distances between genomes with translocation, in: Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), ACM Press, 1995, pp. 604–613.
[12] G. Li, X. Qi, X. Wang, B. Zhu, A linear-time algorithm for computing translocation distance between signed genomes, in: Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM), in: LNCS, vol. 3109, Springer, 2004, pp. 323–332.

[13] M. Ozery-Flato, 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), in: LNCS, vol. 4009, Springer, 2006.
[14] M. Ozery-Flato, R. Shamir, Sorting by translocations via reversals theory, Journal of Computational Biology 14 (4) (2007) 408–422.
[15] E. Tannier, A. Bergeron, M. Sagot, Advances on sorting by reversals, Discrete Applied Mathematics 155 (6–7) 881–888.
[16] L. Wang, D. Zhu, X. Liu, S. Ma, An $o(n^2)$ algorithm for signed translocation, Journal of Computer and System Sciences 70 (3) (2005) 284–299.