JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 4, 2007

Pp. 408–422
DOI: 10.1089/cmb.2007.A003

Sorting by Reciprocal Translocations via Reversals Theory

MICHAL OZERY-FLATO and RON SHAMIR

ABSTRACT

The understanding of genome rearrangements is an important endeavor in comparative genomics. A major computational problem in this field is finding a shortest sequence of genome rearrangements that transforms, or sorts, one genome into another. In this paper we focus on sorting a multi-chromosomal genome by translocations. We reveal new relationships between this problem and the well studied problem of sorting by reversals. Based on these relationships, we develop two new algorithms for sorting by reciprocal translocations, which mimic known algorithms for sorting by reversals: a score-based method building on Bergeron's algorithm, and a recursive procedure similar to the Berman-Hannenhalli method. Though their proofs are more involved, our procedures for reciprocal translocations match the complexities of the original ones for reversals.

Key words: genome rearrangement, sorting by translocations, sorting by reversals.

1. INTRODUCTION

FOR OVER A DECADE NOW, much effort has been put into large-scale genome sequencing projects. Analysis of the sequences that have accumulated so far showed that genome rearrangements play an important role in the evolution of species. A major computational problem in the research of genome rearrangements is finding a most parsimonious sequence of rearrangements that transforms one genome into another. This is called the *genomic sorting problem*, and the corresponding number of rearrangements is called the *rearrangement distance* between the two genomes. Genomic sorting gives rise to a spectrum of fascinating combinatorial problems, each defined by the set of allowed rearrangement operations and by the representation of the genomes.

In this paper we focus on the problem of sorting by translocations. We reveal new similarities between sorting by translocations and the well studied problem of sorting by reversals. The study of the problem of sorting by translocations is essential for the full comprehension of any genomic sorting problem that permits translocations. Below we review the relevant previous studies and summarize our results. Formal definitions are provided on the next section.

School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.

Following the pioneering work by Nadeau and Taylor (1984), reversals and translocations are believed to be very common in the evolution of mammalian species. *Reversals* (or *inversions*) reverse the order and the direction of transcription of the genes in a segment inside a chromosome. *Translocations* exchange tails between two chromosomes. A translocation is *reciprocal* if none of the exchanged tails is empty. The genomic sorting problem where the allowed rearrangement operations are reversals (respectively, reciprocal translocations) is referred to as *sorting by reversals*, hereafter SBR (respectively, *sorting by reciprocal translocations*, hereafter SRT).

Both SBR and SRT use restricted models that allow for a single type of genome rearrangement. Clearly, a model that allows both reversals and translocations is biologically more realistic than each of these two restricted models. Still, the study of sorting by reversals only or by translocations only is of great importance to the understanding of more complex models that allow for several types of genome rearrangements. For example, the problem of sorting by reversals, translocations, fissions, and fusions is reduced to SBR in polynomial time (Hannenhalli and Pevzner, 1995; Ozery-Flato and Shamir, 2003; Tesler, 2002a). In many cases, algorithms for restricted models can be integrated into algorithms for complex models (Ozery-Flato and Shamir, 2006a; Tesler, 2002a).

SBR and SRT were both proven to be polynomial. Hannenhalli and Pevzner (1999) gave the first polynomial algorithm for SBR; since then, other, more efficient algorithms and simplifications of the analysis have been presented. Berman and Hannenhalli (1996) presented a recursive algorithm for SBR. Kaplan et al. (2000) simplified the analysis and gave an $O(n^2)$ algorithm for SBR. Using a linear time algorithm by Bader et al. (2001) for computing the reversal distance, the algorithm of Berman and Hannenhalli can be implemented in $O(n^2)$. A score-based algorithm for SBR was presented by Bergeron (2005). Tannier et al. (2007) presented an elegant algorithm for SBR that can be implemented in $O(n^{3/2}\sqrt{log(n)})$ using a clever data structure due to Kaplan and Verbin (2005).

SRT was first introduced by Kececioglu and Ravi (1995) and was given a polynomial time algorithm by Hannenhalli (1996). Bergeron et al. (2006a) pointed to an error in Hannenhalli's proof of the reciprocal translocation distance formula and consequently in Hannenhalli's algorithm. They presented a new proof and gave an $O(n^3)$ algorithm for SRT. Recently, we (Ozery-Flato and Shamir, 2006a) proved that the algorithm of Tannier et al. (2007) for SBR can be adapted to solve SRT in $O(n^{3/2}\sqrt{log(n)}))$ time.

Can the rich theory on SBR be used to solve SRT? It is well known that a translocation on a multi-chromosomal genome can be simulated by a reversal on a concatenation of the chromosomes (Hannenhalli and Pevzner, 1995). However, different translocations require different concatenations. In addition, intra-chromosomal reversals do not have matching translocations. Last but not least, the formulas of the reversal distance and the reciprocal translocation distance are different. They differ in particular in the parameters that concern difficult structures for SBR/SRT, which are sometimes referred to as "bad components."[1] Thus, from a first glance the similarity between SRT and SBT is rather superficial.

In Ozery-Flato and Shamir (2006a) we introduced a new auxiliary graph for the analysis of SRT (the "overlap graph with chromosomes" of two multi-chromosomal genomes, an extension of the "overlap graph" of two uni-chromosomal genomes) and used it to adapt the fastest extant algorithm for SBR to SRT (Ozery-Flato and Shamir, 2006a; Tannier et al., 2007). In this paper we reveal new relationships between SRT and SBR. Based on these relationships we develop two new algorithms for SRT, which mimic known algorithms for SBR: a score-based method building on Bergeron's algorithm (2005) and a recursive procedure similar to the Berman and Hannenhalli (1996) method. Though the proofs of the algorithms are more involved than those of their counterparts for SBR, our procedures for translocations match the complexities of the original ones for reversals: the score-based algorithm performs $O(n^2)$ operations on $O(n)$-long bit vectors; the recursive algorithm runs in $O(n^2)$ time.

The paper is organized as follows. Section 2 gives the necessary preliminaries. Section 3 presents the score-based algorithm and Section 4 presents the recursive algorithm. Related genomic sorting problems, as well as possible applications of our results and future research problems, are discussed in Section 5.

---

[1] *Hurdles* (Hannenhalli and Pevzner, 1999; Kaplan et al., 2000) for SBR, *leaves* (Bergeron et al., 2006a) (equivalently, *minimal sub-permutations* [Hannenhalli, 1996]), for SRT.

2. PRELIMINARIES

This section provides a basic background for the analysis of SRT. We follow to a large extent the nomenclature and notation of Hannenhalli (1996) and Kaplan et al. (2000). 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.

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. If A is a genome with the set of genes $\{1, \ldots, n\}$ then any concatenation π_A of the chromosomes of A is a signed permutation of size n. In the following, we assume for simplicity and without loss of generality that there is a concatenation π_B of the chromosomes in the target genome B which is the identity permutation. For example,

$$B = \{(1, 2, \ldots, 5), (6, 7, 8)\}.$$

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:

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

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

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

Note that we can mimic a prefix-prefix (respectively, prefix-suffix) translocation by a flip of one of the chromosomes followed by a prefix-suffix (respectively, prefix-prefix) translocation. As was observed by Hannenhalli and Pevzner (1995), a translocation on A can be simulated by a reversal on π_A in the following way:

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

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.

For an interval of genes $I = (i_1, \ldots, i_k)$ define $Tails(I) = \{i_1, -i_k\}$. Note that $Tails(I) = Tails(-I)$. For a genome A_1 define $Tails(A_1) = \cup_{X \in A_1} Tails(X)$. For example:

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

Two genomes A_1 and A_2 are called *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 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.

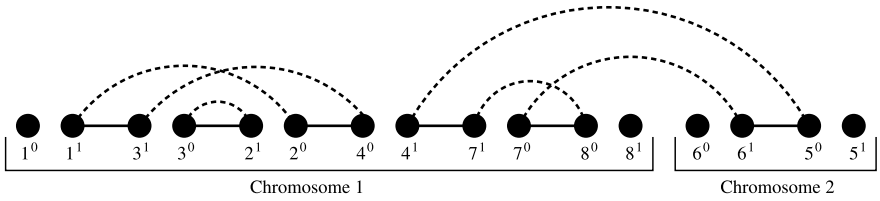

FIG. 1. The cycle graph $G(A_1, B_1)$, where $A_1 = \{(1, -3, -2, 4, -7, 8), (6, 5)\}$ and $B_1 = \{(1, \ldots, 5), (6, 7, 8)\}$. Dotted lines correspond to gray edges. The gray edge $(1, 2)$ is internal, whereas $(4, 5)$ is external. $(2, 3)$ is an adjacency.

2.1. The cycle graph

Let N be the number of chromosomes in A (equivalently, B). We shall always assume that both A and B contain genes $\{1, \ldots, n\}$. The *cycle graph* of A and B, denoted $G(A, B)$, is defined as follows. The set of vertices is $\cup_{i=1}^{n}\{i^0, i^1\}$. For every pair of adjacent genes in B, i and $i + 1$, add a gray edge $(i, i + 1) \equiv (i^1, (i + 1)^0)$. For every pair of adjacent genes in A, i and j, add a black edge $(i, j) \equiv (out(i), in(j))$, where $out(i) = i^1$ if i has a positive sign in A and otherwise $out(i) = i^0$, and $in(j) = j^0$ if j has a positive sign in A and otherwise $in(j) = j^1$. An example is given in Figure 1. There are $n - N$ black edges and $n - N$ gray edges in $G(A, B)$. A gray edge $(i, i + 1)$ is *external* if the genes i and $i + 1$ belong to different chromosomes of A, 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*(A) (equivalently, *Tails*(B)). Therefore, $G(A, B)$ is uniquely decomposable into cycles with alternating gray and black edges. An *adjacency* is a cycle with two edges.

2.2. The overlap graph with chromosomes

Place the vertices of $G(A, B)$ along a straight line according to their order in π_A. Now, every gray edge can be associated with an interval of vertices of $G(A, B)$. Two intervals *overlap* if their intersection is not empty but neither contains the other. The *overlap graph with chromosomes* of A and B w.r.t. π_A, denoted $\Omega(A, B, \pi_A)$, is defined as follows. There are two types of nodes. The first type corresponds to gray edges in $G(A, B)$. The second type corresponds to chromosomes of A. Two nodes are connected if their associated intervals overlap (Fig. 2). For the rest of this paper we will refer to overlap graphs with chromosomes as Ω-graphs.

In order to avoid confusion, we will refer to nodes that correspond to chromosomes as "chromosomes" and reserve the word "vertex" for the nodes that correspond to gray edges of $G(A, B)$. Observe that a vertex in $\Omega(A, B, \pi_A)$ is external iff there is an edge connecting it to a chromosome. Note that the internal/external state of a vertex in $\Omega(A, B, \pi_A)$ does not depend on π_A (the partition of the chromosomes is known from A). A vertex in $\Omega(A, B, \pi_A)$ is *oriented* if its corresponding edge connects two genes with different signs in π_A, otherwise it is *unoriented*.

Let $\mathrm{OV}(A, B, \pi_A)$ be the subgraph of $\Omega(A, B, \pi_A)$ induced by the set of nodes that correspond to gray edges (i.e., excluding the chromosomes' nodes). We shall use the word "component" for a connected component of $\mathrm{OV}(A, B, \pi_A)$. 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. A trivial component

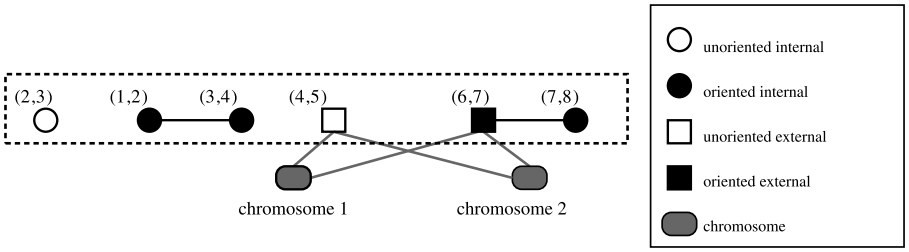

FIG. 2. The overlap graph with chromosomes $\Omega(A_1, B_1, \pi_{A_1})$, where A_1 and B_1 are the genomes from Figure 1 and $\pi_{A_1} = (1, -3, -2, 4, -7, 8, 6, 5)$. The graph induced by the vertices within the dashed rectangle is $\mathrm{OV}(A_1, B_1, \pi_{A_1})$.

corresponds to an adjacency. 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. If the spans of two components intersect then either they overlap by at most gene, or one span contains the other. Clearly, $I(M)$ is independent of π_A iff M is internal. Thus the set of internal components in $\Omega(A, B, \pi_A)$ is independent of π_A. Denote by $\mathcal{IN}(A, B)$ the set of non-trivial internal components in $\Omega(A, B, \pi_A)$. The following lemma follows from the definition of "sub-permutations" in Hannenhalli (1996):

Lemma 1. *Suppose I is the span of an internal component. Then the genes of I form a continuous interval I' in one of the chromosomes of B and Tails(I) = Tails(I').*

2.3. The reciprocal translocation distance

Let $c(A, B)$ denote the number of cycles in $G(A, B)$.

Theorem 1 (Bergeron et al., 2006a; Hannenhalli, 1996). *The reciprocal translocation distance between A and B is $d(A, B) = n - N - c(A, B) + F(A, B)$, where $F(A, B) \geq 0$ and $F(A, B) = 0$ iff $\mathcal{IN}(A, B) = \emptyset$.*

Let Δc denote the change in the number of cycles after performing a translocation on A. Then $\Delta c \in \{-1, 0, 1\}$ (Hannenhalli, 1996). A translocation is *proper* if $\Delta c = 1$. A translocation is *safe* if it does not create any new non-trivial internal component. A translocation ρ is *valid* if $d(A \cdot \rho, B) = d(A, B) - 1$. It follows from Theorem 1 that if $\mathcal{IN}(A, B) = \emptyset$, then every safe proper translocation is necessarily valid.

In a previous study (Ozery-Flato and Shamir, 2006a), we presented a generic algorithm for SRT that uses a sub-procedure for solving SRT when $\mathcal{IN}(A, B) = \emptyset$. The algorithm focuses on the efficient elimination of the non-trivial internal components. We showed that the work performed by this generic algorithm, not including the sub-procedure calls, can be implemented in linear time. This led to the following theorem:

Theorem 2 (Ozery-Flato and Shamir, 2006a). *SRT is linearly reducible to SRT with $\mathcal{IN}(A, B) = \emptyset$.*

By the theorem above, it suffices to solve SRT assuming that $\mathcal{IN}(A, B) = \emptyset$. Both algorithms that we describe below will make this assumption.

2.4. The effect of a translocation on the overlap graph with chromosomes

Let $\pi_{CH} \equiv \pi_{CH}(A, \pi_A)$ be the linear order of the chromosomes in A, as defined by π_A. Slightly abusing terminology, we extend the definition of the Ω-graph to include π_{CH}. In other words, an Ω-graph carries also a permutation of its chromosome nodes defined by π_A. Two chromosomes in $\Omega(A, B, \pi_A)$ are called *consecutive* if they are consecutive in π_{CH}.

Let $H = \Omega(A, B, \pi_A)$ and let v be any vertex in H. Denote by $N(v) \equiv N(v, H)$ the set of vertices that are neighbors of v in H, including v itself (but not including chromosome neighbors). Denote by $CH(v) \equiv CH(v, H)$ the set of chromosomes that are neighbors of v in H. Clearly, if v is external then $|CH(v)| = 2$, otherwise $CH(v) = \emptyset$.

Every external gray edge e defines one proper translocation that cuts the black edges incident to e. (Out of the two possibilities of prefix-prefix or prefix-suffix translocations, exactly one would be proper.) For an external vertex v denote by $\rho(v)$ the proper translocation that the corresponding gray edge defines on A. If v is an oriented external vertex then $\rho(v)$ can be mimicked by a reversal $\hat{\rho}(v)$ on π_A. For an oriented external vertex v define $H \cdot \rho(v) = \Omega(A \cdot \rho(v), B, \pi_A \cdot \hat{\rho}(v))$. The following two lemmas refine claims in Ozery-Flato and Shamir (2006a).

Lemma 2. *Let v be an oriented external vertex in H and suppose the chromosomes in $CH(v)$ are consecutive. Then $H \cdot \rho(v)$ 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 $u \in N(v)$ complement the edges between u and $CH(u) \cup CH(v)$. In particular, the external/internal state of a vertex $u \in N(v)$ is flipped iff u is internal or $CH(u) = CH(v)$.*

Proof. The correctness of *(i)* follows immediately from Observation 4.1 in Kaplan et al. (2000). To prove *(ii)*, let $u \in N(v)$. Since the chromosomes in $CH(v)$ are consecutive, u is either internal or $|CH(u) \cap CH(v)| \in \{1, 2\}$. In each of these cases, $CH(u)$ is complemented w.r.t. $CH(u) \cup CH(v)$ (for illustration, see Fig. 3). Suppose $w \notin N(v)$. Let I_v and I_w be the intervals associated with v and w respectively (see Section 2.2). Then there are three possible cases:

Case 1: $I_w \subset I_v$ and w is internal. Then I_w is contained entirely in one of the exchanged segments. Thus w remains internal and hence $CH(w, H \cdot \rho(v)) = CH(w, H) = \emptyset$.

Case 2: $I_w \subset I_v$ and w is external. Then $CH(w, H) = CH(v, H)$ and the two endpoints of I_w exchange their chromosomes after $\rho(v)$ is performed. Thus $CH(w, H \cdot \rho(v)) = CH(w, H)(= CH(v, H))$.

Case 3: $I_w \cap I_v = \emptyset$ or $I_v \subset I_w$. In these two cases the endpoints of I_w are not affected by $\rho(v)$ and hence $CH(w, H \cdot \rho(v)) = CH(w, H)$. ■

We shall sometimes need to change the chromosome order or flip a chromosome. These operations can be mimicked by reversals on π_A but do not correspond to translocations, and thus are not covered by Lemma 2. For an interval of chromosomes $I \subset \pi_A$, let $\hat{\rho}(I)$ denote the flip, i.e., reversal, of I in π_A. Let $H \cdot \rho(I) = \Omega(A, B, \pi_A \cdot \hat{\rho}(I))$.

Lemma 3. *For an interval of chromosomes $I \subset \pi_A$, $H \cdot \rho(I)$ is obtained from H by the following operations. (i) Reverse the order of the chromosomes in I. (ii) Complement the subgraph induced by the set $\{v$: exactly one of the chromosomes in $CH(v)$ is contained in $I\}$, and flip the orientation of every vertex in it. In particular, if I is a single chromosome of A then $H \cdot \rho(I)$ is obtained by complementing the subgraph induced by the neighbors of I in H, and flipping the orientation of every vertex in it.*

Proof. The vertices affected by $\rho(I)$ are the ones that overlap I. A vertex v overlaps I iff exactly one of its endpoints belong to I (hence it must be external). The rest of the proof follows directly from Observation 4.1 in Kaplan et al. (2000). ■

We refer to two Ω-graphs of the same pair of genomes A and B, irrespective of the concatenation π_A, as *equivalent*. Clearly, we can transform an Ω-graph to any other equivalent graph by a sequence of flips of chromosomes intervals, as defined by Lemma 3.

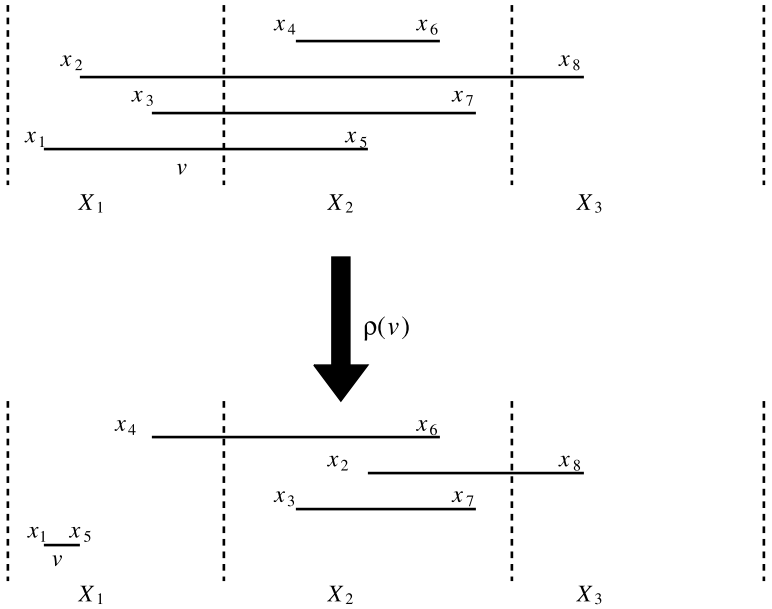

FIG. 3. The effect of performing a translocation, mimicked by a reversal, on overlapping intervals. X_1, X_2, and X_3 are chromosomes, and the dashed lines denote the borders between them in the concatenation (X_1, X_2, X_3). The letters $x_1, \ldots, x_8$ denote the endpoints of the intervals (the endpoints are vertices of the cycle graph). The interval v corresponds to an (external) edge on which a translocation is performed.

Observation 1. *Let H and H' be two equivalent graphs in which v is an oriented external vertex. Then the set of internal components is the same for $H \cdot \rho(v)$ and $H' \cdot \rho(v)$.*

Proof. We can transform $H \cdot \rho(v)$ into $H' \cdot \rho(v)$ by a sequence of flips of chromosomes intervals. By Lemma 3, a flip of an interval of chromosomes does not change the internal/external state of any vertex, and does not affect the neighborhood of any internal vertex. Thus $H \cdot \rho(v)$ and $H' \cdot \rho(v)$ must have the same set of internal components. ■

Let v be an external vertex in H, and let H' be an equivalent graph to H in which v is oriented, possibly $H = H'$ if v is already oriented in H. A key definition that will be crucial throughout the paper is the following: $\Delta\text{IN}(H, v)$ is the set of vertices that belong to external components in H (equivalently, H') but are in non-trivial internal components in $H' \cdot \rho(v)$. By Observation 1, if *(i)* v is an external vertex in H, and *(ii)* H' is equivalent to H, then $\Delta\text{IN}(H, v) = \Delta\text{IN}(H', v)$. It follows that in order to compute $\Delta\text{IN}(H, v)$, we can assume without loss of generality that v is oriented and the chromosomes in $CH(v)$ are consecutive. As we shall see, the additional work required to satisfy this assumption will not change the overall complexity of the algorithms.

3. A SCORE-BASED ALGORITHM

In this section, we present a score-based algorithm for SRT when $\mathcal{IN}(A, B) = \emptyset$. This algorithm is similar to an algorithm by Bergeron (2005) for SBR. Denote by $N_{\text{IN}}(v)$ and $N_{\text{EXT}}(v)$ the neighbors of v that are respectively internal and external. It follows that $N_{\text{IN}}(v) \cup N_{\text{EXT}}(v) \cup \{v\} = N(v)$. For two chromosomes X and Y, let $V_{XY} = \{v : CH(v) = \{X, Y\}\}$.

Lemma 4. *Let X and Y be two consecutive chromosomes in $H = \Omega(A, B, \pi_A)$. Suppose $v \in V_{XY}$ is oriented. Let $w \in N(v)$. If w has no external neighbors in $H \cdot \rho(v)$ then $N_{\text{EXT}}(w) \subseteq N_{\text{EXT}}(v)$ and $N_{\text{IN}}(v) \subseteq N_{\text{IN}}(w)$.*

Proof. It follows from Lemma 2 that if $u \in (N_{\text{EXT}}(w) \setminus N_{\text{EXT}}(v)) \cup (N_{\text{IN}}(v) \setminus N_{\text{IN}}(w))$ then u is an external neighbor of w in $H \cdot \rho(v)$. ■

For each vertex v in $H = \Omega(A, B, \pi_A)$ we define the *score* of v as $|N_{\text{IN}}(v)| - |N_{\text{EXT}}(v)|$. The following lemma lays the basis for the score-based approach and is used by the implementation of the recursive algorithm as well.

Lemma 5. *Let X and Y be two consecutive chromosomes in $H = \Omega(A, B, \pi_A)$ for which $V_{XY} \neq \emptyset$. Let $O \subset V_{XY}$ be a set of oriented (external) vertices and suppose $O \neq \emptyset$. Let $v \in O$ be a vertex with a maximal score in H. Then $O \cap \Delta\text{IN}(H, v) = \emptyset$.*

Proof. Assume $u \in O \cap \Delta\text{IN}(H, v)$. Then $u \in N(v, H)$, and by Lemma 4 $N_{\text{EXT}}(u) \subseteq N_{\text{EXT}}(v)$ and $N_{\text{IN}}(v) \subseteq N_{\text{IN}}(u)$. However, since v has the maximal score in O, we get $N_{\text{EXT}}(u) = N_{\text{EXT}}(v)$ and $N_{\text{IN}}(v) = N_{\text{IN}}(u)$. Therefore, $N(u) = N(v)$, and by Lemma 2 it follows that u is an isolated internal vertex in $H \cdot \rho(v)$, a contradiction to the assumption that $u \in \Delta\text{IN}(H, v)$. ■

Theorem 3. *Let X and Y be two consecutive chromosomes in $H = \Omega(A, B, \pi_A)$. Let O be the set of all the oriented external vertices in V_{XY} and suppose $O \neq \emptyset$. Let $v \in O$ be a vertex that has the maximal score in H. Let $S = S(v)$ be the set of all the vertices w that satisfy the following conditions in H:*

1. *w is a neighbor of v,*
2. *w is an unoriented external vertex and $CH(w) = CH(v)$,*
3. *$N_{\text{EXT}}(w) \subseteq N_{\text{EXT}}(v)$,*
4. *$N_{\text{IN}}(v) \subseteq N_{\text{IN}}(w)$, and*
5. *$O \cap N(v) \subseteq N_{\text{EXT}}(w)$.*

If $S = \emptyset$ then $\rho(v)$ is safe. Otherwise, let $w \in S$ be a vertex that has a maximal score in $H \cdot \rho(X)$, where $X \in CH(v)$. Then $\rho(w)$ is safe.

Proof. Suppose $S = \emptyset$ and assume v is unsafe. Let $w \in \Delta\text{IN}(H, v)$ be a neighbor of v in H. w satisfies conditions 3 and 4 by Lemma 4, it is external and $CH(w) = CH(v)$, by Lemma 2. It follows from Lemma 5 that $O \cap \Delta\text{IN}(H, v) = \emptyset$. Hence w is unoriented in H and the last condition is satisfied (otherwise w has a neighbor from O in $H \cdot \rho(v)$, in contradiction to the choice of $w \in \Delta\text{IN}(H, v)$). It follows that $w \in S$, a contradiction.

Suppose $S \neq \emptyset$. Let $H' = H \cdot \rho(X)$, where $X \in CH(v)$. Let $w \in S$ be a vertex with maximal score in H'. We prove below that if $\Delta\text{IN}(H', w) \neq \emptyset$ then $\Delta\text{IN}(H', w) \cap S \neq \emptyset$, in contradiction to Lemma 5.

Let $O_1 = O \cap N(v)$ in H. Then in H' : *(i)* all the vertices in S are oriented (condition 2), *(ii)* O_1 contains all the unoriented external vertices with $CH = CH(v)$ that are not neighbors of v, and *(iii)* there are no edges between S and $O_1 \cup \{v\}$ (condition 5). It follows that each vertex in $O_1 \cup \{v\}$ remains external after performing a translocation on any vertex in S.

Assume that $\Delta\text{IN}(H', w) \neq \emptyset$. Let $u \in \Delta\text{IN}(H', w)$ be a neighbor of w in H'. We shall prove that $u \in S$. Clearly, u is an external vertex in H' and $CH(u) = CH(w) = CH(v)$. Since all the vertices in $O_1 \cup \{v\}$ are external and there are no edges between them and w in H', $u \notin O_1 \cup \{v\}$ and there are no edges between u and $O_1 \cup \{v\}$ in H' (Lemma 4). Since all the unoriented vertices that are not neighbors of v belong to O_1, u must be oriented. It follows that in H, u satisfies conditions 1, 2 and 5. We now prove that u satisfies conditions 3 and 4 in H as well, thus $u \in S$—a contradiction to Lemma 5.

Suppose u does not satisfy condition 4 in H. Let $x \in N_{\text{IN}}(v) \setminus N_{\text{IN}}(u)$ in H. Since w satisfies condition 4 in H, $x \in N_{\text{IN}}(w) \setminus N_{\text{IN}}(u)$ in H. Since x is internal, all its edges are the same in H and H'. Hence $x \in N_{\text{IN}}(w) \setminus N_{\text{IN}}(u)$ in H'. It follows from Lemma 4 that u has an external neighbor (x) in $H' \cdot \rho(w)$, a contradiction to $u \in \Delta\text{IN}(H', w)$. Thus u must satisfy condition 4 in H.

Suppose u does not satisfy condition 3 in H. Let $z \in N_{\text{EXT}}(u) \setminus N_{\text{EXT}}(v)$ in H.

Case 1: $X \notin CH(z)$. Since w satisfies condition 3, $z \in N_{\text{EXT}}(u) \setminus N_{\text{EXT}}(w)$ in H. Then in H': $z \in N_{\text{EXT}}(u) \setminus N_{\text{EXT}}(w)$ (Lemma 3). Then according to Lemma 4, u has an external neighbor (z) in $H' \cdot \rho(w)$, a contradiction to $u \in \Delta\text{IN}(H', w)$.

Case 2: $X \in CH(z)$. In H: since w satisfies condition 3 and $z \notin N_{\text{EXT}}(v)$ then $z \notin N_{\text{EXT}}(w)$. Thus in H': $z \notin N(u)$, $z \in N(v) \cap N(w)$ (Lemma 3). Therefore, in $H' \cdot \rho(w)$ the path u, z, v exists (Lemma 2), a contradiction to $u \in \Delta\text{IN}(H', w)$ (since v is external in $H' \cdot \rho(w)$). ■

Theorem 3 immediately implies the following polynomial time algorithm (Algorithm 1) for finding a safe proper translocation using $H = \Omega(A, B, \pi_A)$:

Algorithm 1. *Find_Safe_Translocation_Using_Scores* (H)

---

1. Let X and Y be two chromosomes for which there exists a common adjacent (external) vertex u.
2. Flip chromosomes, if necessary, to make X and Y consecutive and to make u oriented.
3. Let $v \in V_{XY}$ be an oriented (external) vertex with a maximal score.
4. Compute the set of vertices $S(v)$ defined by Theorem 3.
5. If $S(v) = \emptyset$ then return $\rho(v)$.
6. Otherwise,
   a. Flip chromosome X or Y, and recalculate the score of the vertices.
   b. Let $w \in S(v)$ be a vertex with a maximal score.
   c. Return $\rho(w)$.

The above algorithm can be implemented in $O(n^2)$ time using $O(n)$ operations on $O(n)$-long bit vectors, in a similar manner to the implementation of the algorithm of Bergeron (2005) for SBR. The implementation is presented in Figure 4 and uses the following notations. The symbols $\boldsymbol{v}$, $\boldsymbol{X}$, ***ext*** and

FIG. 4. An $O(n^2)$ implementation of Algorithm 1 using $O(n)$-long bit vectors.

$\boldsymbol{o}$ represent bit vectors of size $n - N$. The vector $\boldsymbol{v}$ corresponds to the vertex v, where $\boldsymbol{v}[u] = 1$ iff u is a neighbor of v. The vector $\boldsymbol{X}$ corresponds to chromosome X, where $\boldsymbol{X}[v] = 1$ iff $X \in CH(v)$. The chromosome vectors are ordered according to their order in π_A. The vectors $\boldsymbol{ext}$ and $\boldsymbol{o}$ correspond to the sets of external vertices and oriented vertices respectively. In other words, $\boldsymbol{ext}[u] = 1$ iff u is external, $\boldsymbol{o}[u] = 1$ iff u is oriented. The score of each vertex is stored in an integer vector $\boldsymbol{score}$. The symbols $\wedge$, $\vee$, $\oplus$ and $\neg$ respectively denote the bitwise-AND, bitwise-OR, bitwise-XOR and bitwise-NOT operators. Steps 1–6 in the algorithm in Figure 4 locate a safe proper translocation $\rho(v)$. Steps 7 and 8 perform $\rho(v)$ and update the above vectors.

Corollary 1. *The score-based algorithm solves SRT in $O(n^3)$ time.*

4. A RECURSIVE ALGORITHM

In this section, we present a recursive algorithm for SRT when $\mathcal{IN}(A, B) = \emptyset$. This algorithm is similar to the algorithm of Berman and Hannenhalli (1996) for SBR.

4.1. The algorithm

Denote the number of vertices in a graph H by $|H|$. For two chromosomes, X and Y, let O_{XY} (respectively U_{XY}) be the set of oriented (respectively unoriented) vertices in H for which $CH = \{X, Y\}$. Thus $O_{XY} \cup U_{XY} = V_{XY}$.

Theorem 4. *Let $H = \Omega(A, B, \pi_A)$. If H contains an external vertex then it contains an external vertex v for which $\Delta\mathrm{IN}(H, v) \leq \frac{|H|}{2}$.*

Proof. Let X and Y be two chromosomes for which $V_{XY} \neq \emptyset$. Assume w.l.o.g. that X and Y are consecutive and $O_{XY} \neq \emptyset$. Let $v \in O_{XY}$ be a vertex with maximal score in H. If $\Delta\mathrm{IN}(H, v) = \emptyset$ then we are done since $|\Delta\mathrm{IN}(H, v)| = 0 \leq \frac{|H|}{2}$. Suppose $\Delta\mathrm{IN}(H, v) \neq \emptyset$. By Lemma 5, $\Delta\mathrm{IN}(H, v) \cap O_{XY} = \emptyset$. Thus $\Delta\mathrm{IN}(H, v) \cap U_{XY} \neq \emptyset$. Let $H' = H \cdot \rho(X)$ and let $u \in U_{XY}$ be a vertex with maximal score in H'. Let $M_v = \Delta\mathrm{IN}(H, v)$ and $M_u = \Delta\mathrm{IN}(H', u) = \Delta\mathrm{IN}(H, u)$. We shall prove that $M_u \cap M_v = \emptyset$, and hence $\min\{|M_v|, |M_u|\} \leq \frac{|H|}{2}$. Assume $x \in M_v$ and let $x = x_0, \ldots, x_k, x_{k+1} = v$ be a shortest path from x to v in H. Then by Lemma 2, $CH(x_k) = CH(v)$ and $x_0, \ldots, x_{k-1}$ are internal. Hence the path $x_0, \ldots, x_k$ exists in H'. Moreover, $x_k \notin O_{XY}$ since the path $x_0, \ldots, x_k$ exists in $H \cdot \rho(v)$ and $M_v \cap O_{XY} = \emptyset$. Thus $x_k \in U_{XY}$. If none of the vertices in $\{x_0, \ldots, x_k\}$ is in $N(u, H')$ then the path remains intact in $H' \cdot \rho(u)$. Otherwise, let x_j be the first vertex in $x_0, \ldots, x_k$ that is in $N(u, H')$. Thus the path $x_0, \ldots, x_j$ exists in $H' \cdot \rho(v)$. If $x_j \in \{x_0, \ldots, x_{k-1}\}$ then x_j is external in $H' \cdot \rho(u)$. If $x_j = x_k$ then by Lemma 5 $M_u \cap U_{XY} = \emptyset$ and hence $x_k \notin M_u$. Thus in any case $x = x_0 \notin M_u$. ■

Theorem 5. *Let v be an external vertex in $H = \Omega(A, B, \pi_A)$. Suppose $\Delta\mathrm{IN}(H, v) \neq \emptyset$. Let $w \in \Delta\mathrm{IN}(H, v)$ be an external vertex in H. Then $\Delta\mathrm{IN}(H, w) \subset \Delta\mathrm{IN}(H, v)$.*

Proof. Assume w.l.o.g. that the chromosomes in $CH(v)$ are consecutive and v is an oriented (external) vertex in H. By Lemma 2, w is a neighbor of v in H and $CH(v) = CH(w)$ (otherwise it would remain external in $H \cdot \rho(v)$). Let x be a vertex in H such that $x \notin \Delta\mathrm{IN}(H, v)$. It suffices to prove that $x \notin \Delta\mathrm{IN}(H, w)$. Let $x = x_0, \ldots, x_k = y$ be a shortest path from x to an external vertex in $H \cdot \rho(v)$. Then in H: x_j is neighbor of v iff x_j is a neighbor of w, for $j = 1..k$ (otherwise there is a path in $H \cdot \rho(v)$ from w to the external vertex $x_k = y$).

Case 1: w is oriented in H. Then the subgraphs induced by the vertices $\{x_0, \ldots, x_k\}$ in $H \cdot \rho(w)$ and $H \cdot \rho(v)$ are the same. Hence in $H \cdot \rho(w)$: y is external and the path in $x = x_0, \ldots, x_k = y$ exists.

Case 2: w is unoriented in H. In $H \cdot \rho(v)$ the vertices in $\{x_0, \ldots, x_{k-1}\}$ are internal and $x_k (= y)$ is external. Therefore $x_j \in \{x_0, \ldots, x_{k-1}\}$ satisfies in H: *(i)* x_j is a neighbor of v iff x_j is external and $CH(x_j) = CH(w)$, and *(ii)* x_j is not a neighbor of v iff x_j is internal. Denote by H' the graph obtained from H after flipping one of the chromosomes in $CH(w)$.

Case 2.a: At least one vertex in $\{x_0, ..., x_{k-1}\}$ is a neighbor of v in H. Choose $x_j \in \{x_0, \ldots, x_{k-1}\}$ a neighbor of v in H such that $\{x_0, \ldots, x_{j-1}\}$ are not neighbors of v in H. Then in H the following conditions are satisfied: *(i)* $x_0, \ldots, x_j$ is a path, *(ii)* all the vertices in $\{x_0, \ldots, x_{j-1}\}$ are internal and *(iii)* x_j is external satisfying $CH(x_j) = CH(v)$. Therefore in H' the path $x_0, \ldots, x_j$ still exists and none of the vertices in the path is a neighbor of v (equivalently, w). Hence, the path remains intact in $H' \cdot \rho(w)$.

Case 2.b: None of the vertices in $\{x_0, \ldots, x_{k-1}\}$ is a neighbor of v in H. Then the path $x_0, \ldots, x_k$ exists in H'. v is not a neighbor of w in H' hence v remains external in $H' \cdot \rho(w)$. If x_k is a neighbor of v (and w) in H' then the path $x_0, \ldots, x_k, v$ exists in $H' \cdot \rho(w)$ and hence $x = x_0 \notin \Delta\mathrm{IN}(H, w)$. If x_k is not a neighbor of v and w in H' then x_k is necessarily external in H' (equivalently, H). In this case the path $x = x_0, \ldots, x_k = y$ remains intact in $H' \cdot \rho(w)$ and $x = x_0 \notin \Delta\mathrm{IN}(H, w)$. ■

Corollary 2. *Let v be an external vertex in H. Suppose $M = \Delta\text{IN}(H, v) \neq \emptyset$. Let H_M be the subgraph of H induced by the nodes in $M \cup CH(v)$, and let w be an external vertex in H_M. Then $\Delta\text{IN}(H, w) \subseteq \Delta\text{IN}(H_M, w)$. In particular, if $\Delta\text{IN}(H_M, w) = \emptyset$ then $\Delta\text{IN}(H, w) = \emptyset$.*

Proof. We assume w.l.o.g. that the chromosomes in $CH(w)$ are consecutive and w is oriented in H. Then $H_M \cdot \rho(w)$ is identical to the subgraph induced by $M \cup CH(v)$ in $H \cdot \rho(w)$. It follows that every component in $H \cdot \rho(w)$ contained in M is also a component of $H_M \cdot \rho(w)$. By Theorem 5 every internal component in $H \cdot \rho(w)$ is contained in M. Thus $\Delta\text{IN}(H, w) \subseteq \Delta\text{IN}(H_M, w)$. ■

The two theorems above are correct for any subgraph H' of $\Omega(A, B, \pi_A)$ that is induced by a set of vertices and their adjacent chromosomes. By recursive use of Theorem 4 and Corollary 2 we get the following algorithm for locating a safe proper translocation. Algorithm 2 receives $H = \Omega(A, B, \pi_A)$ as an input.

Algorithm 2. *Find_Safe_Translocation_Recursive* (H)

---

1. Choose v from H satisfying $\Delta\text{IN}(H, v) \leq \frac{|H|}{2}$, according to the proof of Theorem 4.
2. $M \leftarrow \Delta\text{IN}(H, v)$
3. If $M \neq \emptyset$:
   a. $H_M \leftarrow$ the subgraph of H induced by $M \cup CH(v)$
   b. $\rho(v) \leftarrow$ *Find_Safe_Translocation_Recursive*(H_M)
4. Return $\rho(v)$

4.2. A linear time implementation

We shall now prove that Algorithm *Find_Safe_Translocation_Recursive* can be implemented in linear time. We shall use an algorithm of Bader et al. (2001) for the computation of $\Delta\text{IN}(H, v)$. We shall use an algorithm by Kaplan et al. (2000) for locating an external vertex v satisfying $|\Delta\text{IN}(H, v)| \leq \frac{|H|}{2}$. A difficulty in trying to apply these algorithms is that they operate on signed permutations and not on Ω-graphs. To overcome this, the algorithm will be initially called with genomes A and B. Before every recursive call it will build two appropriate co-tailed genomes A_M and B_M and pass them as arguments to the recursive call instead of H_M.

Assume w.l.o.g. that there are no adjacencies in $G(A, B)$ (otherwise, every maximal run of adjacencies can be replaced by one element in both A and B). Thus $G(A, B)$ contains no internal components.

4.2.1. Computing $\Delta\text{IN}(H, v)$ in linear time. We apply the translocation $\rho(v)$ on A, and then compute the set of non-trivial internal components. Suppose we want to compute the set of non-trivial internal components in $\Omega(A, B, \pi_A)$. We compute the set of components in $\text{OV}(\pi_A)$ in linear time, using an algorithm by Bader et al. (2001). The output of this algorithm contains the set of components of $\text{OV}(\pi_A)$ along with the span of each. The graph $\text{OV}(\pi_A)$ contains additional vertices that are not in $\Omega(A, B, \pi_A)$. These additional vertices correspond to edges between tails of B. Since A and B are co-tailed, the neighbors of these vertices in $\text{OV}(\pi_A)$ are all external. Therefore the removal of these additional vertices does not affect the set of internal components in this graph. A component is internal iff the two endpoints of its span belong to the same chromosome of A. An internal component is non-trivial if its span contains more than two elements.

4.2.2. Finding an external vertex v satisfying $|\Delta\text{IN}(H, v)| \leq \frac{|H|}{2}$ in linear time. Let X and Y be two chromosomes that contain the endpoints of an external edge v. Build a concatenation π_A in which X and Y are consecutive. Let $H = \Omega(A, B, \pi_A)$ and let $H' = H \cdot \rho(X)$. If O_{XY} (respectively U_{XY}) does not induce a clique in H (respectively H') then we can use the following lemma:

Lemma 6. *Let $v_1, v_2 \in O_{XY}$. If $v_2 \notin N(v_1)$ then* $\min\{|\Delta\text{IN}(H, v_1)|, |\Delta\text{IN}(H, v_2)|\} \leq \frac{|H|}{2}$.

Proof. It suffices to prove that $\Delta\text{IN}(H, v_1) \cap \Delta\text{IN}(H, v_2) = \emptyset$. Assume $x \in \Delta\text{IN}(H, v_1)$ and let $x = x_0, \ldots, x_k = v_1$ be a shortest path from x to v_1 in H. Since the neighborhood of v_2 remains intact in $H \cdot \rho(v_1)$ there is no edge from v_2 to any vertex in that path. Therefore this path exists in $H \cdot \rho(v_2)$ and hence $u \notin \Delta\text{IN}(H, v_2)$. ■

Align the nodes of $G(A, B)$ according to π_A. For two nodes in $G(A, B)$, p_1 and p_2, denote $p_1 < p_2$ iff p_1 is to the left of p_2. For a vertex v in $H = \Omega(A, B, \pi_A)$, denote by $Left(v)$ and $Right(v)$ the left and right endpoints respectively of its gray edge. Suppose $O_{XY} = \{v_1, \ldots, v_k\}$, where $Left(v_j) < Left(v_{j+1})$ for $j = 1..k-1$. If there exist two consecutive vertices v_j and v_{j+1} such that $Right(v_j) > Right(v_{j+1})$, then we found two edges that do not overlap. Thus $v_{j+1} \notin N(v_j, H)$. By Lemma 6 $\min\{|\Delta\text{IN}(H, v_j)|, |\Delta\text{IN}(H, v_{j+1})|\} \leq \frac{|H|}{2}$. Otherwise, the vertices in O_{XY} form a clique in H. We can find whether U_{XY} induces a clique in H' in a similar manner by aligning the nodes of $G(A, B)$ according to $\pi_A \cdot \rho(X)$.

Suppose O_{XY} induces a clique in H and U_{XY} induces a clique in H' (one of which might be empty). In this case we use the proof of Theorem 4 in order to find a vertex v satisfying $|\Delta\text{IN}(H, v)| \leq \frac{|H|}{2}$. We calculate the score in H for every vertex in O_{XY} and the score in H' for every vertex in U_{XY} in the following way. Let $\{I_1, \ldots, I_k\}$ be a set of intervals forming a clique. Let $U = \{J_1, \ldots, J_l\}$ be another set of intervals. Let $U(j)$ denote the number of intervals in U that overlap with I_j. There is an algorithm by Kaplan et al. (2000) that computes $U(1), \ldots, U(k)$ in $O(k + l)$. We use this algorithm twice to compute $|N_{\text{EXT}}(v_j)|$ and $|N_{\text{IN}}(v_j)|$, for $j = 1..k$.

4.2.3. Performing a recursive call Suppose the external vertex v chosen in the first step of the algorithm satisfies $M = \Delta\text{IN}(H, v) \neq \emptyset$. Let $H = \Omega(A, B, \pi_A)$. Let H_M be the subgraph of H induced by $M \cup CH(v)$. We demonstrate below how to build two co-tailed genomes, A_M and B_M, in linear time, for which there exists an Ω-graph $H'_M = \Omega(A_M, B_M, \pi_{A_M})$ satisfying: *(i)* $H_M \subset H'_M$, *(ii)* $|H'_M| \leq |H_M| + 2$, and *(iii)* Every $u \in H'_M \setminus H_M$ is external and $\rho(u) = \rho(v)$.

Every internal component in $G(A \cdot \rho(v), B)$ contains in its span one of the new black edges created by $\rho(v)$. A component in M is *maximal* if its span is maximal. Since there are two new black edges in $G(A \cdot \rho(v), B)$, there are at most two maximal components in M. Note that for every $v \in M$, its two endpoints belong to the span of a maximal component. Construct genomes A_M and B_M in the following way.

Case 1: There are two maximal components in M. Let I_1 and I_2 be the spans of the two maximal components in M (after applying $\rho(v)$). I_1 and I_2 are disjoint since every maximal component belong to a different chromosome of $A \cdot \rho(v)$. By Lemma 1, there exist two intervals I'_1 and I'_2 in B, where for $i = 1, 2$ I_i and I'_i have the same set of elements and $Tails(I_i) = Tails(I'_i)$. Let $B_M = \{I'_1, I'_2\}$. Let A_M be the result of the translocation on $\{I_1, I_2\}$ that cuts the two new black edges in I_1 and I_2 and recreates the old black edges that were originally cut by $\rho(v)$ (i.e., the translocation inverse to $\rho(v)$).

Case 2: There is exactly one maximal component in M. In this case only one of the chromosomes in $A \cdot \rho(v)$ contains components from M. Let I be the span of the maximal component in M. Again, by Lemma 1 there exists an interval I' in B with the same elements as I, satisfying $Tails(I) = Tails(I')$. Let $B_M = \{I', (i_1, i_2)\}$, where (i_1, i_2) is the new black edge in $A \cdot \rho(v)$ that is not contained in any of the components in M. Let A_M be the result of the translocation on $\{I, (i_1, i_2)\}$ that cuts the new black edge in I_1 and (i_1, i_2) and recreates the old black edges that were originally cut by $\rho(v)$ (i.e., the translocation inverse to $\rho(v)$).

Obviously in both cases A_M and B_M are co-tailed. Each of the two chromosomes in A_M (respectively, B_M) is an interval in A (respectively, B). Moreover, A_M (equivalently, B_M) contains the endpoints of each and every gray edge in M. Let $H'_M = \Omega(A_M, B_M, \pi_{A_M})$ where π_{A_M} is a concatenation of the two chromosomes in A_M in which the elements appear in the same order as in π_A. It is not hard to see that the H_M is an induced subgraph of H'_M. H'_M contains one or two additional vertices that do not belong to H_M. These additional vertices define the same translocation as v (one of which is indeed v) and correspond to isolated vertices (i.e., trivial internal components) in $H'_M \cdot \rho(v)$. Thus, the (one or two) additional vertices in H'_M are external. Since H_M does not contain adjacencies, so does H'_M.

The above described implementation implies:

Lemma 7. *Algorithm Find_Safe_Translocation_Recursive can be implemented in linear time.*

Proof. We have demonstrated how to implement the first two steps of the algorithm in linear time. Let v be the vertex chosen in step 1 of the algorithm. Suppose $M = \Delta\text{IN}(H, v) \neq \emptyset$. In this case we presented a way to construct two co-tailed genomes, A_M and B_M, whose Ω-graph is almost identical to H_M (there are one or two additional external vertices in H'_M that define the same translocation as v). Obviously this construction can be done in linear time. It is only left to prove that the number of elements in the genomes decreases by a constant factor in every call.

Let n and n_M be the number of genes in A and A_M, respectively. In every recursive call, the number of chromosomes involved is 2. Hence $|H| = n - N$ (i.e., gray edges in $G(A, B)$) and $|H'_M| = n_M - 2$. Suppose $|H_M| \leq \frac{|H|}{2}$ (step 1), then $|H_M| \leq \frac{n-N}{2} \leq \frac{n}{2} - 1$. Now $n_M = |H'_M| + 2 \leq |H_M| + 4 \leq \frac{n}{2} + 3$. Thus for $n \geq 18$, $n_M \leq \frac{2n}{3}$. We update the algorithm as follow. At the beginning, we verify that the number of genes is at least 18. In this case a recursive call (if needed) will be made with genomes with at most $\frac{2}{3}$ of the genes in A and B. Otherwise, we simply search for a proper safe translocation by computing $\Delta\text{IN}(H, v)$ for every external vertex v. ■

Corollary 3. *The recursive algorithm solves SRT in $O(n^2)$ time.*

5. DISCUSSION

The fundamental observation of Hannenhalli and Pevzner (1995) that translocations can be mimicked by reversals was made over a decade ago, but until recently the analyses of SRT and SBR had little in common. Here and in Ozery-Flato and Shamir (2006a), we tighten the connection between the two problems, by presenting a new framework for the study of SRT that builds directly on ideas and theory developed for SBR. Using this framework we show here how to transform two central algorithms for SBR, Bergeron's score-based algorithm and the Berman-Hannenhalli's recursive algorithm, into algorithms for SRT. These new algorithms for SRT maintain the time complexity of the original algorithms for SBR. These results improve our understanding of the connection between the two problems. Still, deeper investigation into the relation between SRT and SBR is needed. In particular, providing a reduction from SRT to SBR or vice versa is an interesting open problem.

Algorithms for SRT can only be applied to a pair of genomes having the same set of chromosome ends. This requirement is removed if SRT is extended to allow for non-reciprocal translocations, including fissions and fusions of chromosomes, and the latter can be viewed as translocations involving empty chromosomes (Hannenhalli and Pevzner, 1995). This more general problem of sorting by translocations can be reduced in linear time to SRT, as we intend to prove in a future work.

The problem of sorting by reversals, translocations, fissions, and fusions (SBRT) was studied (Hannenhalli and Pevzner, 1995; Ozery-Flato and Shamir, 2003; Tesler, 2002a) and proven to be polynomial. An algorithm solving SBRT is used by the applications GRIMM (Tesler, 2002b) and MGR (Bourque and Pevzner, 2002), which analyze genome rearrangements in real biological data (Bourque et al., 2004; Murphy et al., 2005; Pevzner and Tesler, 2003). The first step in the current algorithm for SBRT generates two co-tailed genomes, say A and B, with the same distance as the two input genomes (Tesler, 2002a). In the following steps, genome A is sorted into genome B using reciprocal translocations and *internal* reversals that do not alter the set of chromosome tails. In other words, SBRT is solved by a reduction to a more constrained problem that allows only for reciprocal translocations and internal reversals. We refer to this constrained problem as SBRT^{C}. SBRT^{C} is currently solved by a reduction to SBR, where each reversal simulates either a reciprocal translocation or an internal reversal. We believe that an algorithm for SBRT^{C} that explicitly treats translocations and reversals as distinct operations would be more natural and powerful than one that does not. In a future work, we intend to prove that each of the algorithms presented here and in (Ozery-Flato and Shamir, 2006a) can be extended to solve SBRT^{C}, even when reversals are given priority over translocations (i.e., a "good" reversal move have a higher priority than a "good" translocation move).

In an optimal solution to SRT, SBR, and SBRT, every move is safe, i.e., it does not create "bad components." Thus the algorithms for these problems mainly focus on finding safe moves. Finding safe moves is conceptually and algorithmically the hardest part in all these algorithms. In a ground-breaking paper, Yancopoulos et al. (2005) proposed a new formulation that bypasses the need for safe reversals/

translocations by introducing a new genome rearrangement operation called *double-cut-and-join* (DCJ). Translocations, reversals, fissions, and fusions can all be viewed as special cases of the DCJ operation. Unlike all the above operations, a DCJ operation can "loop out" a circular chromosome, which can be later reabsorbed by another operation. Thus the problem of sorting by DCJ operations (SDCJ) allows for the creation of intermediate circular chromosomes. Looping out a circular chromosome followed by its reabsorption can also simulate a *block interchange* of two blocks in the same chromosome. The problem of sorting by block interchanges was studied in Christie (1996) and Lin et al. (2005). The ability of DCJs to create and reabsorb circular chromosomes yields a powerful rearrangement model, for which no "bad components" exist. This makes the analysis, distance formula, and algorithms of SDCJ (Bergeron et al., 2006b; Yancopoulos et al., 2005) much simpler and very elegant, in comparison with SRT, SBR, and SBRT. While circular chromosomes are quite common in prokaryotes cells, they have been found sporadically in eukaryotes cells, and with some rare exceptions, they are usually not inherited (Ishikawa and Naito, 1999). Thus for the evolution of eukaryotes species, it is reasonable to assume a minimal use, if any, of circular chromosomes. In particular, when there are no bad components, any algorithm for SBRT solves SDCJ without creating circular intermediates.

In the future we intend to study SBRT with additional restrictions that will make its solutions more biologically acceptable. An example for an additional constraint is the exclusion of translocations that create acentric chromosomes (i.e., chromosomes that lack a centromere), since these chromosomes are likely to be lost during subsequent cell divisions (Sullivan et al., 2001). As a first step towards solving this problem, we recently provided a polynomial time algorithm for the constrained problem where only reciprocal translocations that do not create acentric chromosomes are allowed (Ozery-Flato and Shamir, 2007). Another interesting variant of SBRT we wish to study considers a model in which one type of operation is preferable over the other. We believe that the study of SRT and its alignment with SBR theory will assist in the study of these variants of SBRT.

ACKNOWLEDGMENTS

We thank the referees for careful and critical comments that helped to improve this paper. This study was supported in part by the Israeli Science Foundation (grant 309/02). A preliminary version of this study appeared in the proceedings of 4th RECOMB Satellite Workshop on Comparative Genomics (Ozery-Flato and Shamir, 2006b).

REFERENCES

Bader, D.A., Moret, B.M.E., and Yan, M. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comput. Biol.* 8, 483–491.

Bergeron, A. 2005. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Appl. Math.* 146, 134–145.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006a. On sorting by translocations. *J. Comput. Biol.* 13, 567–578.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006b. A unifying view of genome rearrangements. *Lect. Notes Comput. Sci.* 4175, 163–173.

Berman, P., and Hannenhalli, S. 1996. Fast sorting by reversal. *Lect. Notes Comput. Sci.* 1075, 168–185.

Bourque, G., and Pevzner, P.A. 2002. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Res.* 12, 26–36.

Bourque, G., Pevzner, P.A., and Tesler, G. 2004. Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse, and rat genomes. *Genome Res.* 14, 507–516.

Christie, D.A. 1996. Sorting permutaions by block interchanges. *Inform. Process. Lett.* 60, 165–169.

Hannenhalli, S. 1996. Polynomial algorithm for computing translocation distance between genomes. *Discrete Appl. Math.* 71, 137–151.

Hannenhalli, S., and Pevzner, P. 1995. Transforming men into mice (polynomial algorithm for genomic distance problems). *36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, 581–592. IEEE Computer Society Press, Los Alamitos, CA.

Hannenhalli, S., and Pevzner, P. 1999. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM* 46, 1–27.

Ishikawa, F., and Naito, T. 1999. Why do we have linear chromosomes? A matter of Adam and Eve. *Mutation Res. DNA Repair* 434, 99–107.

Kaplan, H., Shamir, R., and Tarjan, R.E. 2000. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.* 29, 880–892.

Kaplan, H., and Verbin, E. 2005. Sorting signed permutations by reversals, revisited. *J. Comput. Syst. Sci.* 70, 321–341.

Kececioglu. J.D., and Ravi, R. 1995. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. *Proc. 6th Annual Symposium on Discrete Algorithms*, 604–613. ACM Press, New York.

Lin, Y.C., Lu, C.L., Chang, H.-Y., et al. 2005. An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species. *J. Comput. Biol.* 12, 102–112.

Murphy, W.J., Larkin, D.M., Everts van der Wind, A., et al. 2005. Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science* 309, 613–617.

Nadeau, J.H., and Taylor, B.A. 1984. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl. Acad. Sci. USA* 81, 814–818.

Ozery-Flato, M., and Shamir, R. 2003. Two notes on genome rearrangements. *J. Bioinformatics Comput. Biol.* 1, 71–94.

Ozery-Flato, M., and Shamir, R. 2006a. An $O(n^{3/2}\sqrt{\log(n)})$ algorithm for sorting by reciprocal translocations. *Lect. Notes Comput. Sci.* 4009, 258–269.

Ozery-Flato, M., and Shamir, R. 2006b. Sorting by translocations via reversals theory. *Lect. Notes Comput. Sci.* 4205, 87–98.

Ozery-Flato, M., and Shamir, R. 2007. Rearrangements in genomes with centromeres. Part I: Translocations. In *Proceedings of the 11th Annual International Conference on Computational Molecular Biology (RECOMB 2007)*, vol. 4453 of LNCS, Springer, 339–353.

Pevzner, P.A., and Tesler, G. 2003. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome Res.* 13, 37–45.

Sullivan, B.A., Blower, M.D., and Karpen, G.H. 2001. Determining centromere identity: cyclical stories and forking paths. *Nat. Rev. Genet.* 2, 584–596.

Tannier, E., Bergeron, A., and Sagot, M. 2007. Advances on sorting by reversals. *Discrete Appl. Math.* 155, 881–888.

Tesler, G. 2002a. Efficient algorithms for multichromosomal genome rearrangements. *J. Comp. Sys. Sci.* 65, 587–609.

Tesler, G. 2002b. GRIMM: genome rearrangements web server. *Bioinformatics* 18, 492–493.

Yancopoulos, S., Attie, O., and Friedberg, R. 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346.

Address reprint requests to:
Dr. Michal Ozery-Flato
School of Computer Science
Tel-Aviv University
Tel-Aviv 69978, Israel

E-mail: ozery@post.tau.ac.il