SORTING GENOMES BY TRANSLOCATIONS AND DELETIONS

Xingqin Qi
Department of Applied Mathematics, Shandong University at Weihai, Weihai, 264213, China
School of Mathematics and System Sciences, Shandong University, Jinan, 250100, China
Email: qixingqin@163.com

Guojun Li
School of Mathematics and System Sciences, Shandong University, Jinan, 250100, China
Department of Biochemistry and Molecular Biology, University of Georgia, Athens, Georgia 30602, USA
Email: guojun@csbl.bmb.uga.edu

Shuguang Li
School of Mathematics and System Sciences, Shandong University, Jinan, 250100, China
Department of Mathematics and Information Science, Yantai University, Yantai, 264005, China
Email: sgliytu@hotmail.com

Ying Xu
Department of Biochemistry and Molecular Biology, University of Georgia, Athens, Georgia 30602, USA
Email: xyn@csbl.bmb.uga.edu

Given two signed multi-chromosomal genomes Π and Γ with the same gene set, the problem of *sorting by translocations* (SBT) is to find a shortest sequence of translocations transforming Π to Γ, where the length of the sequence is called *the translocation distance* between Π and Γ. In 1996, Hannenhalli gave the formula of the translocation distance for the first time, based on which an $O(n^3)$ algorithm for SBT was given. In 2005, Anne Bergeron et al. revisited this problem and gave an elementary proof for the formula of the translocation distance which leads to a new $O(n^3)$ algorithm for SBT. In this paper, we show how to extend Anne Bergron's algorithm for SBT to include deletions, which allows us to compare genomes containing different genes. We present an asymptotically optimal algorithm for transforming Π to Γ by translocations and deletions, providing a feasible sequence with length at most $OPT+2$, where OPT is the minimum number of translocations and deletions transforming Π to Γ. Furthermore, this analysis can be used to approximate the minimum number of translocations and insertions transforming one genome to another.

1. INTRODUCTION

A translocation considered here is always *reciprocal* which exchanges *non-empty* tails between two chromosomes. Given two multi-chromosomal genomes Π and Γ, the problem of *sorting by translocations* (abbreviated as SBT) is to find a shortest translocation sequence that transforms Π to Γ. SBT was first introduced by Kececioglue and Ravi [1] and was given a polynomial time algorithm by Hannenhalli [2]. Bergeron, Mixtacki and Stoye [3] pointed out an error in Hannenhalli's sorting strategy and gave a new $O(n^3)$ algorithm for SBT. Li et al. [4] gave a linear time algorithm for computing the translocation distance (without producing a shortest sequence). Wang et al. [5] presented an $O(n^2)$ algorithm for SBT.

Note that all above algorithms assume that the two genomes have the same gene content. Such is of course rarely the case in biological practice. In this paper we consider a more general case: when the gene set of Γ is a subset of the gene set of Π. Clearly, in such case, "deletions" are needed. Write

$\mathcal{A}$ for the set of genes in both Π and Γ, write $\mathcal{A}_\Pi$ for those in Π only. We assume that each gene in $\mathcal{A}$ appears exactly once in each genome. We will try to computer the minimum number of translocations and deletions transforming Π to Γ, which is denoted as $d_{td}(\Pi,\Gamma)$. We present an asymptotically optimal algorithm, which provides a feasible sequence with length at most $d_{td}(\Pi,\Gamma)+2$.

The paper is organized as follows. The necessary preliminaries are given in Section 2 and Section 3. A lower bound on $d_{td}(\Pi,\Gamma)$ is given in Section 4. In Section 5 and Section 6 we give the approximation algorithm and its analysis respectively. Conclusions are given in Section 7.

2. PRELIMINARIES

As usual, we represent a *gene* by a positive integer and an associated sign ("+" or "−") reflecting the *direction* of the gene, and the corresponding element is said to be *positive element* or *negative element.* A chromosome is a sequence of genes and does not have an orientation. A genome is a set of chromosomes.

A chromosome is orientation-less, therefore *flipping* a chromosome $X = x_1, ..., x_k$ into $-X = -x_k, ..., -x_1$ does not affect the chromosome it represents. Hence, a chromosome X is said to be *identical* to a chromosome Y iff either $X = Y$ or $X = -Y$. As a convention we illustrate a chromosome horizontally and read it from left to right. Genomes Π and Γ are said to be *identical* if their sets of chromosomes are the same. 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 *prefix-prefix* translocation switches X_1 with Y_1 resulting in $(Y_1, X_2), (X_1, Y_2)$. A *prefix-suffix* translocation switches X_1 with Y_2 resulting in $(-Y_2, X_2), (Y_1, -X_1)$. The resulting genome after applying a translocation ρ on genome Π is denoted as $\Pi \cdot \rho$. For a chromosome $X = (x_1, ..., x_k)$, the numbers $+x_1$ and $-x_k$ are called *tails* of X. The set of tails of all the chromosomes in Π is denoted by $\mathcal{T}(\Pi)$. Genomes Π and Γ are *co-tailed* if $\mathcal{T}(\Pi) = \mathcal{T}(\Gamma)$. Therefore, SBT is limited to genomes that are co-tailed.

In the following, *w.l.o.g*, we assume that the elements in each chromosome of the target genome Γ are positive and in increasing order. For example, let $\Pi = \{(4,3),(1,2,-7,5),(6,-8,9)\}$ and $\Gamma = \{(1,2,3),(4,5),(6,7,8,9)\}$. Reader(s) are assumed to have a thoughtful understanding of Refs. 2 and 3.

2.1. The Cycle Graph

For a chromosome $X = (x_1, ..., x_k)$, replace every positive element $+x_i$ by ordered pair (x_i^t, x_i^h) of vertices and replace every negative element $-x_i$ by ordered pair (x_i^h, x_i^t) of vertices. Vertices u and v are *neighbors* in X if they are adjacent in the ordered list constructed in aforementioned manner. We say that vertices u and v are *neighbors in a genome* if they are neighbors in some chromosome in this genome. For gene x, vertices x^t and x^h are always neighbors and for simplicity, we exclude them from the definition of "neighbors" in the following discussion.

The bicolored *cycle graph* $\mathcal{G}(\Pi,\Gamma)$ of Π with respect to Γ which have the same gene content is defined as follows. The vertex set V contains the pairs of vertices x^t and x^h for every gene x in Π, *i.e.* V={u: u is either x^t or x^h, where x is a gene in Π}. Vertices u and v are connected by a *black* edge iff they are neighbors in Π. Vertices u and v are connected by a *gray* edge iff they are neighbors in Γ.

A gray edge (u,v) in $\mathcal{G}(\Pi,\Gamma)$ is *interchromosomal* if u, v belong to different chromosomes, otherwise is *intrachromosomal.* Each vertex has degree either 2 or 0, hence the graph can be uniquely decomposed into a number of disjoint cycles. A cycle is *interchromosomal* if it contains at least one interchromosomal gray edge, otherwise is *intrachromosomal.*

2.2. The Sub-permutation

A *segment* is an interval $I = x_i, ..., x_j$ within a chromosome $X = x_1, x_2, ..., x_m$. Let V_I be the set of vertices induced by genes in I, *i.e.*, V_I ={u: u is either x_k^t or x_k^h, $i \le k \le j$}. We refer to left vertex corresponding to x_i and right vertex corresponding to x_j as $Left(I)$ and $Right(I)$ respectively. Define $IN(I)$=$V_I \setminus \{Left(I), Right(I)\}$. An edge (u,v) in $\mathcal{G}(\Pi,\Gamma)$ is said to be *inside* the interval I if $u, v \in IN(I)$.

A *sub-permutation(SP)* is an interval $I = x_i, x_{i+1}, .., x_j$ within a chromosome X of Π such that **(i)** there exists no edge (u,v) with $u \in IN(I), v \notin IN(I)$ and **(ii)** that is not the union of smaller such intervals.

We refer to a *SP* by giving its first and last element such as $(x_i ... x_j)$. A *minimal sub-permutation (MSP)* is a *SP* not containing any other *SP*s. A *SP* is *trivial* if it is of the form $(i ... i+1)$ or $(-(i+1) ... -i)$, otherwise is *non-trivial.*

2.3. The Forest of SPs

Two different *SP*s of a chromosome are either disjoint, nested with different endpoints, or overlapping on one element. When two *SP*s overlap on one element, we say that they are *linked.* Successive linked *SP*s form a *chain.* A chain that can not be extended to the left or right is called *maximal.* The nesting and linking relation of *SP*s on a chromosome can be shown in the following way.

Definition 2.1. Given a chromosome X and its SPs, define the forest F_X by the following construction:

1. Each non-trivial SP is represented by a round node.

2. Each maximal chain that contains a non-trivial SP is represented by a square node whose (ordered) children are the round nodes that represent the non-trivial SPs of this chain.

3. A square node is the child of the smallest SP that contains this chain.

The above definition can be extended to a forest of a genome by combining the forests of all chromosomes:

Definition 2.2. Given a genome Π consisting of chromosomes $\{X_1, X_2, ..., X_N\}$. The forest F_Π is the union of forests $F_{X_1}, ..., F_{X_N}$.

Note that a leaf of F_Π corresponds to a *MSP* of Π. Denote the number of leaves and trees of F_Π by L and T respectively. If $T = 1$ and L is even, genome Π has an *even-isolation.* We will refer to a *MSP* that is a leaf in F_Π as simply a *leaf.*

2.4. The Translocation Distance

Let $\rho(X, Y, i, j)$ be a translocation acting on chromosomes $X = (x_1, ..., x_p)$ and $Y = (y_1, ..., y_q)$, where the cleavages occur in X between x_{i-1} and x_i and in Y between y_{j-1} and y_j. Let $f \in \{x^t_{i-1}, x^h_{i-1}\}$ and $g \in \{x^t_i, x^h_i\}$ such that f and g are neighbors in X. Let $u \in \{y^t_{j-1}, y^h_{j-1}\}$ and $v \in \{y^t_j, y^h_j\}$ such that u and v are neighbors in Y. Then ρ *acts on* black edges (f, g) and (u, v). Let Δc denote the change in the number of cycles after performing a translocation on Π. Then $\Delta c \in \{-1, 0, 1\}$ [2]. A translocation is *proper* if $\Delta c = 1$, *improper* if $\Delta c = 0$ and *bad* if $\Delta c = -1$.

It is easy to see each interchromosomal gray edge (u, v) in $\mathcal{G}(\Pi, \Gamma)$ determines a proper (prefix-prefix or prefix-suffix) translocation ρ of Π by cutting the two black edges incident on u and v respectively. In the following, as in Ref. 2, we only consider proper translocations determined by interchromosomal gray edges.

We say that a translocation *destroys* a *SP* C if C is not a *SP* in the resulting genome. The only way to destroy a *SP* with translocations is to apply a translocation with one cleavage in the *SP*, and one cleavage in another chromosome. Such translocations always merge cycles and thus are always bad. Yet, a translocation may destroy more than one *SP* at the same time. In fact, at most two *MSP*s on two different chromosomes, plus all *SP*s containing these two *MSP*s, can be destroyed by a single translocation. If a translocation destroys two *MSP*s on different chromosomes at the same time, we say the translocation *merges* the two *MSP*s. Anne Bergeron et al. [3] proved that it is also possible to eventually merge two *MSP*s that initially belong to two different trees of the same chromosome.

Lemma 2.1. [3] *If a chromosome* X *of genome* Π *contains more than one tree, and no other chromosome of* Π *contains non-trivial MSP, then the trees can be separated on different chromosomes by proper translocations without modifying* F_Π.

Lemma 2.2. [3] *Given two genomes* Π *and* Γ *with the same gene set, assume there are* N *chromosomes and* n *genes in both* Π *and* Γ. *Let* c *be the number of cycles in* $\mathcal{G}(\Pi, \Gamma)$. *The minimum number of translocations for transforming* Π *to* Γ *is* $d_t(\Pi, \Gamma) = n - N - c + t$, *where*

$$t = \begin{cases} L + 2, \textit{ if } L \textit{ is even and } T{=}1 \\ L + 1, \textit{ if } L \textit{ is odd} \\ L, \qquad \textit{if } L \textit{ is even and } T \neq 1 \end{cases}$$

A translocation ρ is *valid* if $d_t(\Pi, \Gamma) - d_t(\Pi \cdot \rho, \Gamma) = 1$. A translocation is *safe* if it does not create any new non-trivial *SP*s. Based on this formula, A. Bergeron et al. gave an $O(n^3)$ efficient algorithm for SBT (hereafter called Algorithm I). For completeness, we describe it in the following.

```
Algorithm I
1. if $L$ is even and $T = 1$
       destroy one leaf such that $L' = L - 1$
2. if $L$ is odd
       perform a bad translocation such that
       $T' = 0$, or $T' > 1$ and $L' = L - 1$
3. while $\Pi$ is not sorted do
       if there exist MSPs on different chromosomes then
           perform a bad translocation such that
           $T' = 0$, or $T' > 1$ and $L' = L - 2$
       else
           perform a proper translocation such that
           $T$ and $L$ remain unchanged.
   end while
```

For above algorithm, we have two points to remark.

Remark 2.1. Through Algorithm I, we always try to merge a pair of *valid MSP*s on different chromosomes, *i.e.*, merging them will not create an even-isolation in the resulting genome. How to select a pair of valid *MSP*s to merge has been introduced in the proof of Lemma 4 in Ref. 3.

Remark 2.2. Through Algorithm I, we always try to destroy a *valid MSP* on some chromosome, *i.e.*, destroying it will not create an even-isolation in the resulting genome. How to select a valid *MSP* to destroy has been described in the proof of Theorem 2 in Ref. 3.

3. ON SORTING BY TRANSLOCATIONS AND DELETIONS

Returning to the problem at hand, *i.e.*, when the gene set of Γ is a subset of the gene set of Π, to find the minimum of translocations and deletions required to transform genome Π to Γ. Let $\widetilde{\Pi}$ be a genome which is induced from Π by deleting from Π all the genes in $\mathcal{A}_\Pi$. We always assume genomes Π and Γ are *co-tailed*, that implies $\widetilde{\Pi}$ and Γ are co-tailed too. Thus one can use Algorithm I to transform $\widetilde{\Pi}$ to Γ.

3.1. The New Definition for Cycle Graph

Given that the genes of $\mathcal{A}_\Pi$ are destined to be deleted, their identities and signs are irrelevant, and could be replaced with any symbols different from those used in $\mathcal{A}$. For any segment of form $S = u_1, u_2 \cdots u_{p-1}, u_p$, where u_1, u_p correspond to two elements of $\mathcal{A}$, and for all $i, 2 \leq i \leq p-1$, u_i corresponds to a gene of $\mathcal{A}_\Pi$, we replace S by $S' = u_1\delta(u_1)u_p$, where $\delta(u_1)$ is the segment of $\mathcal{A}_\Pi$ between u_1 and u_p. And u_1 and u_p are *separated* by a δ.

Example 3.1. Let $S = +1, -a, +2, -3, +b, -c, +4, -5$ be a segment on a chromosome of Π, then S can be rewritten as $+1, \delta(1^h), +2, -3, \delta(3^t), +4, -5$, where $\delta(1^h) = -a$, $\delta(3^t) = +b, -c$.

We represent genomes Π and Γ by the redefined *cycle graph* $\mathcal{G}(\Pi, \Gamma)$, where V is the set of vertices, B is the set of black edges and D is the set of gray edges. These three sets are defined as follows:

- $V = \{x^s\}_{x \in \mathcal{A}}^{s \in \{h,t\}}$
- The black edges pertain to genome Π. There are two kinds of black edges: *direct black edges* which link two adjacent vertices in Π; *indirect black edges* which link two vertices separated by a δ. For an indirect black edge $e = (a, b)$, then $\delta(a)$, the segment of elements in $\mathcal{A}_\Pi$ between a and b, is called the *label of e*.
- Gray edges link adjacent vertices in Γ.

An *indirect cycle* (or *indirect SP*) is one containing at least one indirect black edge, otherwise is *direct.* Color the leaves corresponding to indirect *MSP*s *red*, and the leaves corresponding to direct *MSP*s *blue.*

An example is given in the following Fig.1, where $\Pi = \{(1, -2, 3, 8, 4, -5, a, b, c, 6), (7, 9, -10, 11, -12, 13, 14, -15, d, e, f, 16)\}$. Indirect black edges are indicated by thick lines.

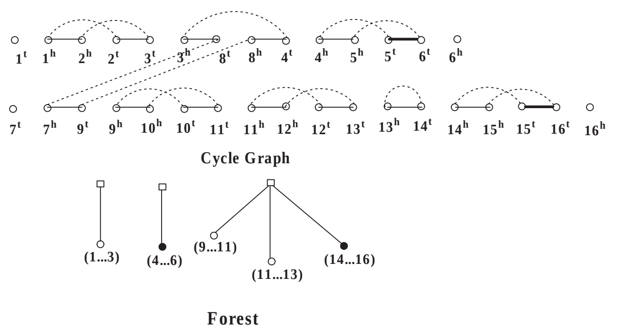
1^t 1^h 2^h 2^t 3^t 3^h 8^t 8^h 4^t 4^h 5^h 5^t 6^t 6^h
7^t 7^h 9^t 9^h 10^h 10^t 11^t 11^h 12^h 12^t 13^t 13^h 14^t 14^h 15^h 15^t 16^t 16^h
Cycle Graph
(1...3)
(4...6)
(9...11)
(11...13)
(14...16)
Forest

Fig. 1. The cycle graph $\mathcal{G}(\Pi, \Gamma)$ and the forest F_Π.

3.2. The New Definition for a Translocation

In $\mathcal{G}(\Pi, \Gamma)$, an indirect black edge determines not an adjacency of genome Π but an interval containing only genes to be deleted. We thus have to redefine what we mean by "the bad translocation acting on two black edges" or "the proper translocation determined by an interchromosomal gray edge". Let $e = (a, b)$ be one indirect edge in $\mathcal{G}(\Pi, \Gamma)$. The segment $[x, \delta(a)]$ designates the interval bounded on the left by x and on the right by the element of $\mathcal{A}_\Pi$ adjacent to b. The segment $[\delta(a), x]$ designates the interval bounded on the left by the element of $\mathcal{A}_\Pi$ adjacent to a and on the right by x. To give Definition 3.1 simply, we define $\delta(a) = \emptyset$ for a direct black edge $e = (a, b)$. Then the segment $[x, \delta(a)]$ designates the interval bounded on the left by x and on the right by a. The segment $[\delta(a), x]$ designates the interval bounded on the left by b and on the right by x.

Definition 3.1. Assume the two black edges $e = (a, b)$ and $f = (c, d)$ are on two different chromosomes $X = x_1, ..., a, \delta(a), b, ..., x_p$ and $Y = y_1, ..., c, \delta(c), d, ..., y_q$, where $x_i (1 \leq i \leq p)$ and y_j $(1 \leq j \leq q)$ are vertices of $\mathcal{G}(\Pi, \Gamma)$.

(1) The translocation determined by $g = (a, c)$ exchanges the segment $[x_1, a]$ of X with the segment $[\delta(c), y_q]$ of Y.

(2) The translocation determined by $g = (b, d)$ exchanges the segment $[x_1, \delta(a)]$ of X with the segment $[d, y_q]$ of Y.

(3) The translocation determined by $g = (a, d)$ exchanges the segment $[x_1, a]$ of X with the segment $[y_1, \delta(c)]$ of Y.

(4) The translocation determined by $g = (b, c)$ exchanges the segment $[x_1, \delta(a)]$ of X with the segment $[y_1, c]$ of Y.

(5)The translocation determined by e and f exchanges the segment $[x_1, \delta(a)]$ of X with the segment $[y_1, c]$ of Y.

4. A LOWER BOUND ON $d_{td}(\Pi, \Gamma)$

Algorithm I can be generalized to graphs containing direct and indirect edges by making use of the new definition of a translocation.

Lemma 4.1. *A proper translocation determined by an interchromosomal gray edge of a cycle C transforms C into two cycles C_1 and C_2, at least one of which is of size* 1*, say C_1. Then the black edge of C_1 is direct.*

Corollary 4.1. *An interchromosomal cycle C of size k is transformed by Algorithm I with $k - 1$ translocations, into k cycles of size* 1*. If C is direct, the k cycles are all direct. If not, only one of these cycles is indirect.*

By Corollary 4.1, for each interchromosomal indirect cycle, Algorithm I gathers all the genes to be deleted into a single segment. At the end, a single deletion is required for each interchromosomal indirect cycle. Now consider the intrachromosomal indirect cycles which are forming *MSP*s. The merging of two *MSP*s is achieved by combining two intrachromosomal cycles C_1 and C_2 one from each of the two *MSP*s. This gets rid of the two *MSP*s and creates an interchromosomal cycle. The destroying of one *MSP* M is achieved by combining one intrachromosomal cycle $C_1 \in M$ with another cycle C_2 which is not in any *MSP*. This gets rid of M and creates an interchromosomal cycle. The resulting interchromosomal cycles can be resolved as described above. For either destroying or merging, if both C_1 and C_2 are indirect, we will save one step of deletion. Thus we get the optimal sorting scheme: *So that there are as few deletions and translocations as possible, we just need to merge as many as possible of pairs of* ***indirect cycles*** *through Algorithm I.*

Given two genome Π and Γ with different genes, denote the number of red leaves in F_Π by $r(\Pi, \Gamma)$ and the number of indirect cycles in $\mathcal{G}(\Pi, \Gamma)$ by $ci(\Pi, \Gamma)$.

Lemma 4.2. *The number of merging of pairs of indirect cycles through Algorithm I is at most* $\lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1$.

Proof. We will prove it in the following subcases.

subcase 1: if L is even and $T = 1$. In such case, two *MSP*s M_1 and M_2 will be destroyed in step 1 and 2 respectively, and the left *MSP*s are paired to merge in step 3. If both M_1 and M_2 are indirect, there are at most $\lfloor \frac{r(\Pi,\Gamma)-2}{2} \rfloor$ mergings of pairs of indirect cycles in step 3, thus there are at most $\lfloor \frac{r(\Pi,\Gamma)-2}{2} \rfloor + 2 = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1$ mergings of pairs of indirect cycles through Algorithm I;[a] if only one of M_1 and M_2 is indirect, then there are at most $\lfloor \frac{r(\Pi,\Gamma)-1}{2} \rfloor$ mergings of pairs of indirect cycles in step 3, thus there are at most $\lfloor \frac{r(\Pi,\Gamma)-1}{2} \rfloor + 1 \leq \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1$ mergings of pairs of indirect cycles through Algorithm I; if neither of M_1 and M_2 is indirect, it is impossible to merge a pair of indirect cycles in steps 1 and 2, thus there are at most $\lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$ mergings of pairs of indirect cycles through Algorithm I.

subcase 2: if L is odd. In such case, step 1 is not executed. One *MSP* M will be destroyed in step 2, and the left *MSP*s are paired to merge in step 3. If M is indirect, there are at most $\lfloor \frac{r(\Pi,\Gamma)-1}{2} \rfloor$ mergings of pairs of indirect cycles in step 3, thus there are at most $\lfloor \frac{r(\Pi,\Gamma)-1}{2} \rfloor + 1 \leq \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1$ mergings of pairs of indirect cycles through Algorithm I; if M is direct, it is impossible to merge a pair of indirect cycles in step 2, so there are at most $\lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$ mergings of pairs of indirect cycles through Algorithm I.

subcase 3: if L is even and $T \neq 1$. In such case, steps 1 and 2 are not executed. So there are at most $\lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$ mergings of pairs of indirect cycles in step 3, *i.e.*, through Algorithm I. □

Theorem 4.1. $d_{td}(\Pi,\Gamma) \geq d_t(\widetilde{\Pi},\Gamma) + ci(\Pi,\Gamma) - \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor - 1$, *where* $\widetilde{\Pi}$ *is the induced genome by deleting from* Π *all the genes in* $\mathcal{A}_{\Pi}$ *and* $d_t(\widetilde{\Pi},\Gamma)$ *is the translocation distance between* $\widetilde{\Pi}$ *and* Γ.

5. DESIGN AN ALGORITHM

We will approximate $d_{td}(\Pi,\Gamma)$ by merging as many as possible of pairs of indirect *MSP*s through Algorithm I. To do this, when some *MSP* must be destroyed, our strategy prefers to destroy a *direct MSP*. Our sorting scheme requires careful consideration of *MSP* choice and cycle (*i.e.*, black edge) choice for each *MSP*. In summary:

***MSP*s merging**

1. Choose a pair of *valid MSP*s M_1 and M_2, having the same color if possible.
2. If both M_1 and M_2 are indirect, choose an indirect black edge e in M_1 and an indirect black edge f in M_2; otherwise, choose any black edge e in M_1 and f in M_2.
3. Apply the (prefix-prefix) translocation determined by e and f by Definition 3.1.

***MSP* destroying**

1. Choose a *valid MSP* M, direct if possible.
2. Choose a black edge e in M, and a black edge f on a different chromosome not contained in any *MSP*.
3. Apply the (prefix-prefix) translocation determined by e and f by Definition 3.1.

5.1. Special Cases and Corresponding Sub-procedures

We always try to merge a pair of "valid" *MSPs* with "the same color", or destroy a "valid" "direct" *MSP*. But in some cases, the two conditions for "merging" or "destroying" are not compatible. We list the following six cases. For a tree with x leaves, we denote it by x-tree. If x is even, it is called an *even-tree*, otherwise, is called an *odd-tree.*

Case 1: $T = 3$, one is an even-tree, the other two are 1-trees. All leaves of the even-tree have color i, the leaves of both the two 1-trees have color j, where $i \neq j$.

Sub-procedure 1: Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi,\Gamma)$.

Case 2: $T = 2$, one is an x-tree, the other is a 1-tree, where x is odd and $x \geq 3$. The leaf of the 1-tree has color i. The rightmost leaf $\mathcal{R}$ of the odd-tree has color k, the leftmost leaf $\mathcal{L}$ of the odd-tree has color l, the other internal leaves of the odd-tree have color j, where $i \neq j$.

Sub-procedure 2:

[a]Since *MSP* destroying may merge a pair of indirect cycles.

1. Apply a sequence of proper translocations without changing F_Π until the two trees are on different chromosomes by Lemma 2.1.
2. If $k = l = i$, then
   - 2.1. merge the leaf of the 1-tree with the middle leaf of the odd-tree.[b]
   - 2.2. merge $\mathcal{R}$ with $\mathcal{L}$.
3. Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi, \Gamma)$.

Case 3: $T = 2$, one is an x-tree, the other is a y-tree, where $x, y \geq 2$ and $x + y$ is even. All leaves of the x-tree have color i, all leaves of the y-tree have color j, $i \neq j$.

Sub-procedure 3:

1. Apply a sequence of proper translocations without changing F_Π until the two trees are on different chromosomes by Lemma 2.1.
2. Merge the middle leaves of the two trees.[c]
3. Flip the chromosome on which the tree of color j is on the right of the tree of color i.
4. Keep merging the pair of leftmost leaves of color j on different chromosomes until at most one leaf of color j is left.[d]
5. Keep merging the pair of rightmost leaves of color i on different chromosomes until at most one leaf of color i is left.
6. Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi, \Gamma)$.

Case 4: $T = 1$, L is odd and $L \geq 3$. The rightmost leaf $\mathcal{R}$ has color i, the leftmost leaf $\mathcal{L}$ has color j, and the other internal leaves all have color red.

Sub-procedure 4:

1. If $i = j =$blue, then
   - 1.1 destroy the middle (red) leaf of the tree.[e]
   - 1.2 merge $\mathcal{R}$ with $\mathcal{L}$.
2. Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi, \Gamma)$.

Case 5: $T = 2$, one is an even-tree, the other is a 1-tree. All leaves of the even-tree have color red, and the leaf of the 1-tree has color blue.

Sub-procedure 5: Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi, \Gamma)$.

Case 6: $T = 1$, L is even and all leaves of F_Π are red.

Sub-procedure 6: Ignore the color of leaves and apply Algorithm I on $\mathcal{G}(\Pi, \Gamma)$.

5.2. Main Lemmas

The following lemmas will be central in providing an invariant for the sorting algorithm.

Lemma 5.1. [3] *If a chromosome X of genome Π contains more than one tree, then there exists a proper translocation involving chromosome X.*

Lemma 5.2. [2] *If there exists a proper translocation in Π, then there exists a proper translocation ρ in Π such that $\Pi \cdot \rho$ does not have any new non-trivial MSP.*

Lemma 5.3. [3] *If a chromosome X of genome Π contains more than one tree, and no other chromosome of Π contains non-trivial MSP, then there exists a proper translocation involving X that does not modify F_Π.*

Lemma 5.4. *In the process of Algorithm I, if some MSP must be destroyed, we can always destroy a "valid" "direct" MSP as long as it is not the Case 4, 5 or 6.*

Lemma 5.5. *Let Π be a genome whose forest F_Π has $L \geq 4$ leaves and $T \geq 2$ trees, where L is even and F_Π is not the Case 1, 2 or 3. If there are same color leaves on different chromosomes, then there exists a bad translocation merging a pair of **same color** leaves such that the forest $F_{\Pi'}$ has $L' = L - 2$ leaves and $T' \neq 1$ trees.*

Proof. We will prove it by discussing on T.

subcase 1: $T = 2$. Assume the two trees are x-tree and y-tree, where $x + y \geq 4$. Clearly, the two trees must be on different chromosomes. When $x \geq 2$

[b] This will make $\mathcal{R}$ and $\mathcal{L}$ be on different chromosomes.
[c] This will create four trees.
[d] It is feasible since we always use a prefix-prefix translocation to merge *MSP*s.
[e] This will make $\mathcal{R}$ and $\mathcal{L}$ be on different chromosomes.

and $y \geq 2$, since F_Π is not Case 3, there must exist a pair of same color leaves between the two trees. Merging this pair of same color leaves will result in a genome Π' with $L' = L - 2$ leaves and $T' \neq 1$ trees. When $x = 1, y \geq 3$ ($y = 1, x \geq 3$, respectively), since F_Π is not the Case 2, there exists an *internal* leaf l_1 of y-tree (x-tree, respectively) such that l_1 has the same color with the only leaf l_2 of x-tree (y-tree, respectively). Merging l_1 and l_2 will result in a genome Π' with $L' = L - 2$ leaves and $T' \neq 1$ trees.

subcase 2: $T = 3$. Since F_Π is not the Case 1, there exist a pair of same color leaves l_1 and l_2 on different chromosomes such that either l_1 or l_2 is a leaf of some x-tree, where $x \geq 2$. Clearly, merging l_1 and l_2 will result in a genome Π' with $L' = L - 2$ leaves and $T' \neq 1$ trees.

subcase 3: $T \geq 4$. In this case, merging any pair of same color leaves on different chromosomes will result in a genome Π' with $L' = L - 2$ leaves and $T' \neq 1$ trees. □

Lemma 5.6. *Let* Π *be a genome whose forest* F_Π *has* $L \geq 4$ *leaves and* $T \geq 2$ *trees, where* L *is even and* F_Π *is not the Case* $1, 2$ *or* 3. *If there does not exist any pair of same color leaves on different chromosomes, then there exists a valid proper translocation of* Π *without changing* L.

Proof. We will prove it by discussing on T.

subcase 1: $T = 2$. The two trees must be on the same chromosome. By Lemma 5.3, there exists a valid proper translocation without changing F_Π.

subcase 2: $T \geq 3$. Since there are only two kind of colors, all trees in F_Π must be on at most two chromosomes.

If the trees are on one chromosome, by Lemma 5.3, there exists a valid proper translocation without modifying F_Π.

Otherwise, assume the two chromosomes are X and Y, and all leaves on X have color i while all leaves on Y have color j, where $i \neq j$. Since $T \geq 3$, either X or Y has more than one tree. Then by Lemmas 5.1 and 5.2, there exists a proper translocation without changing L. The resulting genome must have more than two trees, thus this proper translocation is valid. □

Lemma 5.7. *Let* Π *be a genome having no leaves. If* Π *is not sorted, then there always exists a safe proper translocation on* Π.

Proof. Since Π is not sorted and $F_\Pi = \emptyset$, there must be an interchromosomal gray edge in $\mathcal{G}_{\Pi,\Gamma}$, which determines a proper (prefix-prefix or prefix-suffix) translocation. Then by Lemma 5.2, there exists a safe proper translocation on Π. □

5.3. The Approximation Algorithm

Our extended algorithm for merging as many as possible of pairs of indirect *MSP*s is given in the following Algorithm II.

Algorithm II

```
1. if $L$ is even and $T = 1$
       if it is Case 6, go to sub-procedure 6;
       else, destroy a valid blue leaf by Lemma 5.4.
2. if $L$ is odd
   (a) if $L = 1$, destroy this leaf.
   (b) if it is Case 4 or 5, go to sub-procedure 4 or 5 respectively;
       else, destroy a valid blue leaf by Lemma 5.4.
3. while $L \geq 4$ do
   (a) if it is Case 1,2 or 3, go to sub-procedure $1, 2$ or 3 respectively.
   (b) if there exist same color leaves on different chromosomes then
           perform a valid bad translocation merging a pair of same color leaves by Lemma 5.5.
       else,
           perform a valid proper translocation by Lemma 5.6.
4. if $L = 2$ ( comment: there must be $T = 2$)
   (a) perform proper translocations without changing $F_\Pi$ until the two leaves are on different chromosomes by Lemma 2.1.
   (b) perform a bad translocation merging this two leaves.
5. Perform safe proper translocations on $\Pi$ until $\Pi$ is sorted by Lemma 5.7.
```

Let $\mathcal{G}_1(\Pi,\Gamma)$ be the graph containing only length 1 cycles obtained by applying Algorithm II to the graph $\mathcal{G}(\Pi,\Gamma)$. Now we apply the following Procedure *Deletion* on $\mathcal{G}_1(\Pi,\Gamma)$.

```
Procedure Deletion
For each indirect edge $e = (a, b)$ of $\mathcal{G}_1(\Pi,\Gamma)$
    Delete the gene segment between $a$ and $b$, labeled by $\delta(a)$.
```

We call Algorithm II augmented by Procedure Deletion the *Translocation-Deletion algorithm.* It is clear that the algorithm transforms genome Π to genome Γ. Let $\beta(\Pi,\Gamma)$ be the number of mergings of pairs of *indirect cycles* in Algorithm II.

Theorem 5.1. *The number of translocations and deletions obtained by the Translocation-Deletion algorithm is* $Apx_{td}(\Pi,\Gamma) = d_t(\widetilde{\Pi},\Gamma) + ci(\Pi,\Gamma) - \beta(\Pi,\Gamma)$.

6. ANALYSIS OF TRANSLOCATION-DELETION ALGORITHM

Let $\alpha(\Pi,\Gamma)$ be the number of mergings of pairs of *indirect MSPs* during Algorithm II. Obviously, $\beta(\Pi,\Gamma) \geq \alpha(\Pi,\Gamma)$. Assume that there are $r_{step\ 3}$ red leaves at the beginning of step 3. Assume there are r_{s_i} red leaves when sub-procedure i happens and there will be y_i pairs of red *MSP*s merged in sub-procedure i, where $i = 1, 2, ..., 6$.

Lemma 6.1. *For* $i = 1, 2, 3, 4, 5, 6$, *if* r_{s_i} *is even,* $y_i = \lfloor \frac{r_{s_i}}{2} \rfloor - 1$; *if* r_{s_i} *is odd,* $y_i = \lfloor \frac{r_{s_i}}{2} \rfloor$.

Proof. We discuss the six cases respectively.

Case 1 *appears.* r_{s_1} is even. In sub-procedure 1, two red leaves are used to merge with two blue leaves respectively, the other red leaves if exist are paired to merge, so $y_1 = \lfloor \frac{r_{s_1}-2}{2} \rfloor = \lfloor \frac{r_{s_1}}{2} \rfloor - 1$.

Case 2 *appears.* If $k = l = i$, then r_{s_2} is odd. In sub-procedure 2, one red leaf is used to merge with a blue leaf, the other red leaves if exist are paired to merge, so $y_2 = \lfloor \frac{r_{s_2}-1}{2} \rfloor = \lfloor \frac{r_{s_2}}{2} \rfloor$. If $k = i, l = j$ *or* $k = j, l = i$, then r_{s_2} is even. In sub-procedure 2, two red leaves are used to merge with two blue leaves respectively, the other red leaves if exist are paired to merge, so $y_2 = \lfloor \frac{r_{s_2}-2}{2} \rfloor = \lfloor \frac{r_{s_2}}{2} \rfloor - 1$. If $k = l = j$, then r_{s_2} is odd. In sub-procedure 2, one red leaf is used to merge with a blue leaf, the other red leaves if exist are paired to merge, so $y_2 = \lfloor \frac{r_{s_2}-1}{2} \rfloor = \lfloor \frac{r_{s_2}}{2} \rfloor$.

Case 3 *appears.* If r_{s_3} *is even*, in sub-procedure 3, two red leaves are used to merge with two blue leaves respectively, the other red leaves are paired to merge, so $y_3 = \lfloor \frac{r_{s_3}-2}{2} \rfloor = \lfloor \frac{r_{s_3}}{2} \rfloor - 1$. If r_{s_3} *is odd*, in sub-procedure 3, the middle red leaf is used to merge with the middle blue leaf, the other red leaves are paired to merge, so $y_3 = \lfloor \frac{r_{s_3}-1}{2} \rfloor = \lfloor \frac{r_{s_3}}{2} \rfloor$.

Case 4 *appears.* If $i = j = blue$, then r_{s_4} is odd. In sub-procedure 4, one red leaf is cut, the other red leaves are paired to merge, so $y_4 = \lfloor \frac{r_{s_4}-1}{2} \rfloor = \lfloor \frac{r_{s_4}}{2} \rfloor$. If all leaves of the odd-tree are red, then r_{s_4} is odd. In sub-procedure 4, one red leaf is destroyed, the other red leaves are paired to merge, so $y_4 = \lfloor \frac{r_{s_4}-1}{2} \rfloor = \lfloor \frac{r_{s_4}}{2} \rfloor$. If only one of the rightmost and leftmost leaf is red, then r_{s_4} is even. In sub-procedure 4, a red internal leaf will be destroyed, another red leaf will be used to merge with the only one blue leaf, the other red leaves are paired to merge, so $y_4 = \lfloor \frac{r_{s_4}-2}{2} \rfloor = \lfloor \frac{r_{s_4}}{2} \rfloor - 1$.

Case 5 *appears.* r_{s_5} is even, in sub-procedure 5, one red leaf is destroyed and another red leaf is used to merge with the blue leaf, the other red leaves are paired to merge, so $y_5 = \lfloor \frac{r_{s_5}-2}{2} \rfloor = \lfloor \frac{r_{s_5}}{2} \rfloor - 1$.

Case 6 *appears.* r_{s_6} is even, in sub-procedure 6, two red leaves are destroyed respectively, and the other red leaves are paired to merge, so $y_6 = \lfloor \frac{r_{s_6}-2}{2} \rfloor = \lfloor \frac{r_{s_6}}{2} \rfloor - 1$. □

Since when Cases $i = 4, 5$ or 6 appears in Algorithm II, $r_{s_i} = r(\Pi,\Gamma)$ and $\alpha(\Pi,\Gamma) = y_i$, so we have

Lemma 6.2. *If one of Cases* 4, 5, 6 *appears in Algorithm II, then* $\alpha(\Pi,\Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$ *if* $r(\Pi,\Gamma)$ *is odd, and* $\alpha(\Pi,\Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor - 1$ *if* $r(\Pi,\Gamma)$ *is even.*

Lemma 6.3. *If one of Cases* 1, 2, 3 *appears in Algorithm II, then* $\alpha(\Pi,\Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor - 1$ *if* $r(\Pi,\Gamma)$ *is even, and* $\alpha(\Pi,\Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$ *if* $r(\Pi,\Gamma)$ *is odd.*

Proof. Clearly, $r_{step\ 3} = r(\Pi,\Gamma)$. Note that a red leaf is always merged with another red leaf in step 3 until one of the Cases 1, 2, 3 appears. Assume that there are x_i pairs of indirect *MSP*s which have been merged in step 3 when sub-procedure i happens, $i = 1, 2, 3$. Since $r_{s_i} + 2x_i = r_{step\ 3} = r(\Pi,\Gamma)$,

r_{s_i} and $r(\Pi, \Gamma)$ have the same parity. We have $\alpha(\Pi, \Gamma) = y_i + x_i$. So by Lemma 6.1, if $r(\Pi, \Gamma)$ is even, $\alpha(\Pi, \Gamma) = x_i + \lfloor \frac{r_{s_i}}{2} \rfloor - 1 = \lfloor \frac{r_{s_i} + 2x_i}{2} \rfloor - 1 = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor - 1$; otherwise, $\alpha(\Pi, \Gamma) = x_i + \lfloor \frac{r_{s_i}}{2} \rfloor = \lfloor \frac{r_{s_i} + 2x_i}{2} \rfloor = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$. □

Lemma 6.4. *If none of the six cases appears in Algorithm II,* $\alpha(\Pi, \Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$.

Proof. Since none of the six cases appears in Algorithm II, if $r(\Pi, \Gamma)$ is even, a red leaf is always merged with another red leaf; if $r(\Pi, \Gamma)$ is odd, except one red leaf is merged with a blue leaf, the others are paired to merge, so $\alpha(\Pi, \Gamma) = \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor$. □

By Lemmas 6.2 – 6.4, we have

Lemma 6.5. $\alpha(\Pi, \Gamma) \geq \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor - 1$

Theorem 6.1. $Apx_{td}(\Pi, \Gamma) - d_{td}(\Pi, \Gamma) \leq 2$.

Proof. By Theorems 4.1 and 5.1, $Apx_{td}(\Pi, \Gamma) - d_{td}(\Pi, \Gamma) \leq \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1 - \beta(\Pi, \Gamma) \leq \lfloor \frac{r(\Pi,\Gamma)}{2} \rfloor + 1 - \alpha(\Pi, \Gamma) \leq 2$. □

Example 6.1. See Fig.1, there are two indirect cycles in $\mathcal{G}(\Pi, \Gamma)$ and five non-trivial *MSP*s: $(1...3), (4...6), (9...11), (11...13), (14....16)$, where $(4...6)$ and $(14...16)$ are indirect. $\widetilde{\Pi} = \{(1, -2, 3, 8, 4, -5, 6), (7, 9, -10, 11, -12, 13, 14, -15, 16)\}$ and $d_t(\widetilde{\Pi}, \Gamma) = 16 - 2 - 7 + 6 = 13$. If in step 2 of Algorithm II, (1...3) is chosen to destroy, the resulting forest will be Case 2, then $\beta(\Pi, \Gamma) = 0$, so $Apx_{td}(\Pi, \Gamma) = 13 + 2 - 0 = 15$; if (11...13) is chosen to destroy, (4...6) will merge with (14...16) and (1...3) will merge with (9...11) respectively, then $\beta(\Pi, \Gamma) = 1$, so $Apx_{td}(\Pi, \Gamma) = 13 + 2 - 1 = 14$. Note that 14 is the shortest number of translocations and deletions transforming Π to Γ.

7. CONCLUSIONS

In this paper, we give an asymptotically optimal algorithm when the gene set of Γ is a subset of the gene set of Π. In fact, the problem of transforming Π to Γ with a minimum of translocations and insertions can be approximated by the translocation-deletion analysis, where Π takes the role of Γ, and vice versa. To the best of our knowledge, this is the first time to consider SBT when the genomes have different gene sets.

ACKNOWLEDGMENTS

The work is supported, in part, by National Science Foundation (NSF/DBI-0354771, NSF/ITR-IIS-0407204). Guojun Li's work is also supported by NSFC under Grant No. 60373025.

References

1. J. D. Kececioglu, R. Ravi. Of mice and men: Algorithms for evolutionary distance between genomes with translocation. In: Proc. 6th ACM-SIAM Symposium on Discrete Algorithm. 1995: 604–613.
2. S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics* 1996; **71**: 137–151.
3. A. Bergeron, J. Mixtacki, J. Stoye. On sorting by translocations. In: Proc. 9th Annual International Conference On Research in Computional Molecular Biology (RECOMB'05). Cambridge, MA. 2005: 615–629.
4. G. Li, X. Qi, X. Wang, B. Zhu. A linear time algorithm for computing translocation distance between signed genomes. In: Proc. 15th Annual symposiun on Combinaotrial Pattern Matching (CPM'04). Springer-Verlag. 2004: 323–332.
5. L. Wang, D. Zhu, X. Liu, S. Ma. An $O(n^2)$ algorithm for signed translocation. *Journal of Computer and System Sciences* 2005; **70(3)**: 284–299.
6. N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In: Proc. 11th Annual symposiun on Combinaotrial Pattern Matching (CPM'00). Springer-Verlag. 2000: 222–234.