AN $O(N^2)$ ALGORITHM FOR SIGNED TRANSLOCATION PROBELM [*]

LUSHENG WANG

Department of Computer Science, City University of Hong Kong
Kowloon, Hong Kong, E-mail: cswangl@cityu.edu.hk

DAMING ZHU, XIAOWEN LIU AND SHAOHAN MA

School of Computer Science and Technology
Shandong University, Jinan, Shandong, P. R. China, 250100
E-mail: dmzhu@sdu.edu.cn, liu_xiaowen@mail.sdu.edu.cn, msh@sdu.edu.cn

Genome rearrangement is an important area in computational biology. There are three basic operations, *reversal*, *translocation*, and *transposition*. Here we study the translocation operations. Multichromosomal genomes frequently evolve by *translocation* events that exchange genetic material between two chromosomes. We focus on the signed case, where the direction of each gene is known. The *signed translocation* problem asks to find the minimum number of translocation operations as well as the sequence of translocation operations to transform one genome into the other. A linear-time algorithm that computes the the minimum number of translocation operations was given in Li et al., 2004.[14] However, that algorithm cannot give the optimum sequence of translocation operations. The best known algorithm that can give the optimum sequence of translocation operations for signed translocation problem runs in $O(n^2 \log n)$ time. In this paper, we design an $O(n^2)$ algorithm.

1. Introduction

Genome rearrangement is a new and rapidly developing area in computational biology.[18,19] It contains rich results in terms of both computation and biology. More than sixty years ago, Dobzhansky and Sturtevant published a milestone paper with an evolutionary tree presenting a rearrangement scenario with 17 reversal operations for the species Drosophila pseudoobscura and Miranda.[3] Genome rearrangement is a common mode of molecular evolution in plants, mammals, viral, and bacteria.[1,6,7,9,8,13,11,12,18,19] Although the rearrangement process is very complicated, there are three basic operations, *reversal*, *translocation* and *transposition*. In this paper, we study the translocation operations. Multichromosomal genomes frequently evolve by translocation events that exchange genetic material between two chromosomes. A genome is a set of chromosomes and a chromosome $X = x_1, x_2, ..., x_p$ is a sequence of genes, where x_i is a signed integer representing a gene.

[*]Lusheng Wang and Xiaowen Liu are fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1196/03E]. Daming Zhu is supported by NSFC 60073042, and NSFC 60273032.

Let $X = x_1, \ldots, x_{b-1}, x_b, \ldots, x_p$ and $Y = y_1, \ldots, y_{c-1}, y_c, \ldots, y_q$ be two chromosomes in a signed genome. A translocation swaps the segments in the chromosomes and results two new chromosomes. For a *prefix-prefix* translocation, the new chromosomes are $X' = x_1, ..., x_{b-1}, y_c, ..., y_q$ and $Y' = y_1, ..., y_{c-1}, x_b, ..., x_p$. For a *prefix-suffix* translocation, the new chromosomes are $X' = x_1, ..., x_{b-1}, -y_{c-1}, ..., -y_1$ and $Y' = -x_p, ..., -x_b, y_c, ..., y_q$.

Note that the choices of *prefix-prefix* and *prefix-suffix* translocations implies that one can change the direction of a chromosome without increasing the translocation distance. A chromosome X is *identical* to chromosome Y if either $X = Y$ or $X = -Y$. Genome A is *identical* to genome B if and only if the sets of chromosomes for A and B are the same.

The *translocation distance* between genomes A and B, denoted as $d(A, B)$, is the minimum number of translocations required to transform A into B. Given two genomes, the *signed translocation problem* is to find the minimum number of translocations as well as the sequence of translocation operations to transform one signed genome into the other.

The signed translocation problem was first studied in Kececioglu and Ravi, 1995.[13] Hannenhalli gave the first polynomial time algorithm to solve the problem.[10] The running time is $O(n^3)$, where n is the total number of genes in the genome. An $O(n^2 \log n)$ algorithm was given in Zhu and Ma, 2002.[20] A linear-time algorithm that computes the the minimum number of translocation operations was given in Li et al., 2004.[14] However, that algorithm cannot give the optimal sequence of translocation operations. In this paper, we present an $O(n^2)$ algorithm that can compute the optimum sequence of translocation operations and thus improves upon the best known algorithm.

It seems that it is common to have linear-time algorithms to compute the distance values for various kinds of rearrangement operations. However, it takes more time to give an optimal sequence of operations. For example, for the signed reversal distance, a linear-time algorithm that computes the reversal distance value was given in Bader et al., 2001.[15] However, the best known algorithms to give an optimal sequence of reversal operations still take $O(n^2)$ time.[16,4,5,2] (Tesler, 2002) dealed with minimum number of reversals, translocations, fissions and fusions.[17] The value can be computed in linear-time. However, it still takes $O(n^2)$ time to give the sequence of the four operations.[17] The translocation distance is different from the the distance studied in Tesler, 2002.[17] Our algorithm makes use of some new and non-trival properities and structures.

2. Preliminaries

In this section, we give some basic definitions and describe some previous results that are necessary to present our new algorithm.

2.1. *The cycle graph*

For a genome A, we will construct a graph G_A. For each chromosome $X = x_1, x_2, \ldots, x_p$ in genome A, we have $2p$ vertices in G_A, two vertices x_i^h, x_i^t for each gene x_i in X. The

$2p$ vertices are arranged in a linear order from left to right as

$$l(x_1)r(x_1)l(x_2)r(x_2)\ldots l(x_p)r(x_p), \tag{1}$$

where if x_i is a positive integer, then $l(x_i) = x_i^t$ and $r(x_i) = x_i^h$; and if x_i is a negative integer, then $l(x_i) = x_i^h$ and $r(x_i) = x_i^t$. For each $i \in \{1, 2, \ldots, n-1\}$, there is a black edge $(r(x_i), l(x_{i+1}))$ in G_A. Vertices u and v are *neighbors* in G_A if there is a black edge connecting u and v in G_A.

Given two genomes A and B, we can construct the *cycle graph* G_{AB} from G_A by adding a *grey* edge to every pair of vertices u and v, where u and v are neighbors in G_B. The graph G_{AB} contains two kinds of edges, *black* edge and *grey* edge. Each vertex in G_{AB} (except the first and the last in a chromosome) is incident to two edges, one black and one grey. Thus, each vertex is in a unique cycle in G_{AB}. From the construction, each black edge in the cycle is followed by a grey edge and vice visa. A cycle is *long* if it contains at least two black edges. Otherwise, the cycle is *short*. If $A = B$, then all cycles in G_{AB} are short. $d(A, B)$ is closely related to the number of cycles in G_{AB}.

Let $X = x_1, x_2, ..., x_p$ be a chromosome in A. A *sub permutation* is an interval $x_i, x_{i+1}, \ldots, x_{i+l}$ in X containing at least three genes such that there is another interval of the same length $y_k, y_{k+1}, \ldots, y_{k+l}$ in a chromosome Y of B satisfying $\{|x_i|, |x_{i+1}|, \ldots, |x_{i+l}|\} = \{|y_k|, |y_{k+1}|, \ldots, |y_{k+l}|\}$, $y_k = x_i$, $y_{k+l} = x_{i+l}$, and $, x_{i+1}, ..., x_{i+l-1} \neq y_{k+1}, \ldots, y_{k+l-1}$. x_i and x_{i+l} are called the *ending* genes of the sub permutation.

Let $I = x_i, x_{i+1}, ..., x_j$ be an interval for chromosome X in A. $V(I) = \{x_i^t, x_i^h, x_{i+1}^t, x_{i+1}^h, \ldots, x_j^t, x_j^h\}$ be the set of vertices in G_{AB}. The leftmost vertex and the rightmost vertex in I are referred to as $LEFT(I) = l(x_i)$ and $RIGHT(I) = r(x_j)$. Define $IN(I) = V(I) - \{LEFT(I), RIGHT(I)\}$. An edge (u, v) is *inside* the interval I if both u and v are in $IN(I)$. A sub permutation I can be viewed as a sub graph $G_{AB}(I)$ of G_{AB} containing the vertex set $IN(I)$ such that

(1) there is no edge (u, v) such that $u \in IN(I)$ and $v \notin IN(I)$.

(2) the sub graph corresponding to I has at least one long cycle.

A *minimal sub permutation* ($minSP$ for short) is a sub permutation such that any other interval in the minimal sub permutation is not a sub permutation.

Let u and v be two vertices in (1). u is on the left of v in X. A *segment* $[u, v]$ on chromosome X contains all the vertices in (1) starting at u and ending at v. A segment $[u, v]$ is *inside* a segment $[x, y]$ if both u and v are in $[x, y]$.

The following lemma is used to prove other lemmas, e.g., Lemma 4.3.

Lemma 2.1. [10] *Cutting a $minSP$ into two segments L and R, there must exist a grey edge (u, v) such that $u \in V(L), v \in V(R)$ and $(u, v) \neq (RIGHT(L), LEFT(R))$.*

There exists an *even isolation* in G_{AB} if the following three conditions hold: (1) there are even number of $minSP$'s in G_{AB}, (2) all the $minSP$'s are on a single chromosome of A, and (3) all the $minSP$'s are contained in a single sub permutation. Note that, there is at most one even isolation. Define $i_{AB} = 1$ if there is an even isolation and otherwise, $i_{AB} = 0$.

o_{AB} is defined as $o_{AB} = 1$ if the number of $minSP$ is odd and otherwise $o_{AB} = 0$.

s_{AB} denotes the number of minimal sub permutations in G_{AB} and c_{AB} denotes the number of cycles in G_{AB}. The following theorem gives the value of the translocation distance and is the key to design polynomial time algorithm solving the problem.[10]

Theorem 2.1. [10] *Let n be the number of genes in the genomes and m the number of chromosomes in the genomes. The translocation distance between two signed genomes A and B is*

$$d(A, B) = n - m - c_{AB} + s_{AB} + o_{AB} + 2 \cdot i_{AB}. \tag{2}$$

2.2. *The existing algorithms*

Consider two black edges (u, v) and (f, g) in a long cycle in G_{AB}, where (u, v) is in chromosome X in A and (f, g) is in chromosome Y in A. Consider a translocation ρ acting on X and Y cutting the two black edge (u, v) and (f, g). ρ is a *proper* translocation if the cycle containing (u, v) and (f, g) in G_{AB} becomes two cycles in the new cycle graph. Otherwise, ρ is *improper*. Sometimes, the two black edges that a translocation cuts might be in different cycles in G_{AB}. In that case, a translocation merge the two cycles into one. A *bad* translocation merges two cycles into one.

The formula (2) gives the value of the translocation distance between two genomes. We want to find translocations such that after applying such a translocation, the translocation distance is reduce by one. Define function $\Psi_{AB} = \Psi_{[A,B]} = c_{AB} - s_{AB} - o_{AB} - 2 \cdot i_{AB}$. A translocation ρ is *valid* if $\Delta\Psi_{AB} = \Psi_{[A \cdot \rho, B]} - \Psi_{[A,B]} = 1$, where $A \cdot \rho$ is the new genome after ρ is applied.

It is proved that (1) if there are proper translocations for G_{AB}, there must be a valid proper translocation for G_{AB}; and (2) if there is no proper translocation, there must be a valid bad translocation.[10] The algorithm is given in Figure 1.

```
1.  **while** $A$ is not identical to $B$ **do**
2.    **if** there is a proper translocation for $G_{AB}(V, E)$ **do**
3.      select a valid proper translocation $\rho$
4.    **else** select a valid bad translocation $\rho$
5.      $A \leftarrow A \cdot \rho$
6.    **end** while
```

Figure 1. Algorithm 1: The old algorithm.

Suppose there are n genes in the genomes. $d(A, B)$ is at most $O(n)$. The method in Hannenhalli, 1995 can find a bad valid translocation in $O(n)$ time when no proper valid translocation is available.[10] Thus, the running time depends on the time to find a proper valid translocation. For the best known algorithm, it takes $O(nlogn)$ time to find a valid proper translocation.[20] Thus the total time complexity of the algorithm in Zhu and Ma, 2002[20] is $O(n^2 logn)$. Here we propose a faster algorithm that takes $O(n)$ time to find a proper valid translocation.

3. Computing the translocation distance

We have designed an algorithm that computes all $minSP$'s in $O(n^2)$ time. This algorithm will be used in section 4. Due to space limit, we omit it here.

Theorem 3.1. *There exists an algorithm that computes all $minSP$'s in $O(n^2)$ time.*

After all the $minSP$'s are determined, it is easy to test if there are odd number of $minSP$'s and if there is an even isolation in $O(n)$ time. Thus by now, translocation distance between two signed genomes can be computed in $O(n^2)$ time. The remaining work is to find all the valid translocation operations to transform A into B in time $O(n^2)$.

It is worth to point out that Algorithm 2 is important for the $O(n)$ algorithm that finds a valid proper translocation described in Section 4, where we assume that all the old $minSP$'s have been found by Algorithm 2. Since Algorithm 2 (running in $O(n^2)$ time) is called once in the whole algorithm, solving the signed translocation problem requires $O(n^2)$ time in total.

4. Finding a valid proper translocation in $O(n)$ time

A grey edge is *proper* if its two ends are in different chromosomes. For a proper grey edge (u, v), there are two translocations (prefix-prefix and suffix-prefix) to cut the two black edges adjacent to the grey edge. One of the two translocations breaks a long cycle into two and thus is a proper translocation and the other is improper. From now on, we use a proper grey edge (u, v) to refer to its proper translocation, denoted as $\rho(u, v)$. We use the two terms interchangeably.

Note that some proper translocation may not cut two black edges adjacent to a proper grey edge. However, whenever there is a proper translocation ρ, there must be a proper grey edge in the long cycle that ρ breaks. In our algorithm, we always focus on the proper translocations indicated by proper grey edges.

If a proper grey edge (translocation) does not produce a new $minSP$, then it is valid. Otherwise, it is not valid. The following lemma shows that in this case, we can find a valid proper grey edge inside the new $minSP$.

Lemma 4.1. [20] *If a proper translocation for G_{AB} produces a new $minSP$, say, P, then there must be a proper grey edge inside P that is valid for G_{AB}.*

4.1. *Finding the new $minSP$*

Let $min = \{P_1, P_2, \ldots, P_k\}$ be the set of all $minSP$'s for G_{AB}. Let X_1Y_1 be a new chromosome produced by a proper grey edge in G_{AB}, where X_1 is from chromosome X in genome A and Y_1 is from chromosome Y in A. The black edge

$(RIGHT(X_1), LEFT(Y_1))$ connecting the two parts X_1 and Y_1 is called the *connecting edge* in X_1Y_1. Obviously, a new $minSP$ must contain the connecting edge.

We can find whether a new $minSP$ is produced in X_1Y_1 in $O(n)$ time. The idea of our algorithm is to search the new chromosome X_1Y_1 starting from the two ends of the connecting edge to left and right, respectively. Let l and r be the vertices in X_1 and Y_1 that we are going to check. L denotes the leftmost vertex in X_1 that a new $minSP$ could reach and R denotes the rightmost vertex in Y_1 that a new $minSP$ could reach. $left(u)/right(u)$ denotes the vertex that is on the left/right of vertex u in the cycle graph G_{AB}. See Figure 2.

```
1.  **Initialize** $L = l$ to indicate the rightmost vertex on $X_1$ in a long cycle. **Initialize**
    $R = r$ to indicate the leftmost vertex in $Y_1$ in a long cycle. (**if** there is no long
    cycle in $X_1$ or $Y_1$, **then** returm “no new $minSP$ is found”.)
2.  **Let** $(l, u)$ and $(r, v)$ be the grey edges incident to $l$ and $r$, respectively.
        $(a)$ **if** $v \in V(X_1)$ and $v$ is on the left of $L$ **then** set $L = v$.
        $(b)$ **if** $v \in V(Y_1)$ and $v$ is on the right of $R$ **then** set $R = v$
        $(c)$ **if** $u \in V(X_1)$ and $u$ is on the left of $L$ **then** set $L = u$.
        $(d)$ **if** $u \in V(Y_1)$ and $u$ is on the right of $R$ **then** set $R = u$
        $(e)$ **if** $u$ or $v$ is not in $V(X_1Y_1)$ **then** return “no new $minSP$ is found”.
3.  **If** $l \neq L$ **then** $l = left(l)$. **If** $r \neq R$ **then** $r = right(r)$.
4.  **If** $l \neq L$ or $r \neq R$ goto Step 2.
5.  **If** $[L, R]$ does not contain any $minSP$ in $min$ **then** return $[L, R]$
    **else return** “no new $minSP$ is found”.
```

Figure 2. Algorithm 3: Testing whether a new $minSP$ exists in $O(n)$ time.

In Step 5, we have to test if an old $minSP$ is in $[L, R]$. This can be done in $O(n)$ time by looking at all the old $minSP$'s in min produced by Algorithm 2.

A new sub permutation I in X_1Y_1 containing the connecting edge is a *nested* sub permutation if I does not contain any sub permutation $P' \subset I$ such that $P' \subseteq X_1$ or $P' \subseteq Y_1$.

Theorem 4.1. *Algorithm 3 correctly tests whether X_1Y_1 contains a new $minSP$ and if yes, outputs the new $minSP$. Algorithm 3 runs in $O(n)$ time.*

4.2. *Partition of the new $minSP$*

Let X and Y be two chromosomes of A. Let e be a proper grey edge and b and c the two black edges adjacent to e in G_{AB}. Suppose the proper translocation cutting b and c produces two new chromosomes $X_LX_MY_MY_R$ and Y_LX_R such that $P = X_MY_M$ is a new $minSP$, where X_M is from X and Y_M is from Y. See Figure 3. We use $l(b)$ and $r(b)$ to represent the left and the right ends of edge b. Thus, we have $RIGHT(X_M) = l(b)$ and $LEFT(Y_M) = r(c)$.

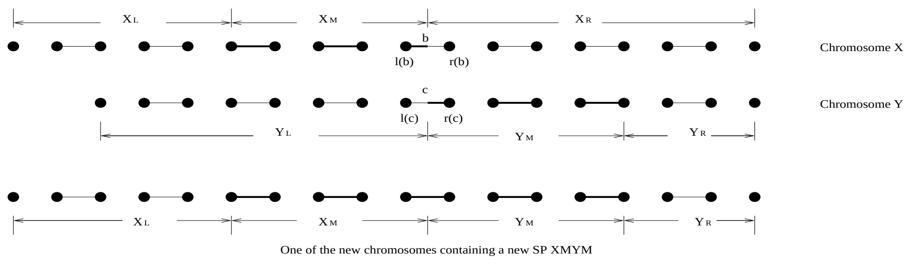

Figure 3. A proper grey edge (translocation) acting on X and Y generates a new $minSP$ in the resulting chromosomes. The bold parts represent segments in the new $minSP$.

From Lemma 4.1, we only have to consider the grey edges inside X_MY_M. However, this grey edge cannot be $(RIGHT(X_M), LEFT(Y_M))$, since if such a grey edge $(RIGHT(X_M),\ LEFT(Y_M))$ exists, then $(RIGHT(X_M), LEFT(Y_M))$ is the grey edge (translocation) resulting in the two new chromosomes $X_LX_MY_MY_R$ and Y_LX_R. See Figure 3. Thus, we want to find a grey edge $(v_R, u_R) \neq (RIGHT(X_M), LEFT(Y_M))$ such that $v_R \in V(X_M), u_R \in V(Y_M)$ and $\rho(u_R, v_R)$ is a valid proper translocation for $G_{AB}(V, E)$.

Lemma 4.2. *Let ρ be a proper translocation acting on chromosomes X and Y that produces the two new chromosomes $X_LX_MY_MY_R$ and Y_LX_R such that $P = X_MY_M$ is a new $minSP$. Let (u, v) be a grey edge inside X_MY_M. $\rho(u, v)$ acting on X and Y produces two new chromosomes $X' = X_1X_VY_VY_1$ and $Y' = X_2X_UY_UY_2$ such that $V(X_VY_V) \neq \emptyset$, $V(X_UY_U) \neq \emptyset$, X_V and X_U form X_M, and Y_V and Y_U form Y_M. If X' or Y' contains a new $minSP$, say, P, then P must be inside X_VY_V or X_UY_U.*

Lemma 4.1 does not tell us how to find such a valid grey edge. We can use $findEdge(X_MY_M)$ to find a proper grey edge that can produce at most one new $minSP$ though it may not be valid.

```
Algorithm 4: $findEdge(X_MY_M)$.
We can start from the right end $l(b)$ of $X_M$, go to the left in $X_M$ and find the first
vertex $v_R$ in $V(X_M)$ satisfying
    (1) $v_R$ is connected to vertex $u_R \in V(Y_M)$ via a grey edge $(u_R, v_R)$ in $G_{AB}$.
    (2) $(v_R, u_R) \neq (l(b), r(c))$. ($RIGHT(X_M) = l(b)$ and $LEFT(Y_M) = r(c)$.)
```

Figure 4. Algorithm 4: Finding a grey edge in X_MY_M such that at least one of the new chromosomes does not contain any $minSP$.

Lemma 4.3. *Let X_MY_M be the new $minSP$. The grey edge (u_R, v_R) is found in $findEdge$ (X_MY_M). (u_R, u) and (v_R, v) denote the two black edges adjacent to the* *grey edge (u_R, v_R) in G_{AB}. $X_N = v_1, v_2, \ldots v_k$, where $v_k = l(b)$ if X_N is not empty, is the segment of vertices (not including v_R) in X_M checked in $findEdge(X_M Y_M)$ before vertex v_R is found in X_M. At most one of the two new chromosomes produced by translocation $\rho(u_R, v_R)$ contains a new $minSP$. In particular, if X_N is not empty, then the new chromosome X' containing the segment X_N does not contain any new $minSP$.*

Corollary 4.1. *Lemma 4.3 still holds if the input $X_M Y_M$ of $findEdge()$ is a nested sub permutation, but not a $minSP$.*

Let $X' = X_1 X_V Y_V Y_1$ be the new chromosome produced by translocation $\rho(u_R, v_R)$ that does not contain any new $minSP$, where $X_1 \cap X_M = \emptyset$, $Y_1 \cap Y_M = \emptyset$, $X_V \subseteq X_M$ and $Y_V \subseteq Y_M$. Let Y' be the other new chromosome produced by $\rho(u_R, v_R)$. According to Lemma 4.3, Y' may contain a new $minSP$, say, P. Lemma 4.1 says that a valid proper translocation can be found in P then. Next, we design a method to repeatedly reduce the size of the new $minSP$ and eventually find the valid proper grey edge.

4.3. *Finding the valid proper grey edge in the new $minSP$*

Let (u_R, v_R) be selected in $findEdge(X_M Y_M)$. One of the two new chromosomes $X' = X_1 X_V Y_V Y_1$ does not contain any new $minSP$. The other chromosome $Y' = X_2 X_U Y_U Y_2$ (call it *crucial* chromosome) that may contain a new $minSP$. Note that the two segments X_U and X_V form X_M and Y_U and Y_V form Y_M (the order may not be fixed). From Lemmas 4.2 and 4.3, the new $minSP$ P in Y' must be inside the segment $X_U Y_U$. Next, we try to reduce the range in $X_U Y_U$ that the new $minSP$ could be. Since $X_U \subseteq X_M$, $Y_U \subseteq Y_M$ and $X_M Y_M$ is a $minSP$ at the very beginning, for any grey edge with one end in $X_U Y_U$, the other end must be in $V(X_M Y_M) = V(X_U) \cup V(X_V) \cup V(Y_U) \cup V(Y_V)$. Thus, it is enough to consider the vertices in $V(X_U) \cup V(X_V) \cup V(Y_U) \cup V(Y_V)$.

A vertex is *ignorable* if it is in $V(X_U Y_U)$, but not in the new $minSP$ in Y'. We need the following lemma to prune segment $X_U Y_U$.

Lemma 4.4. *If there is a grey edge (u_1, v_1) such that $u_1 \in V(X_V Y_V)$ and $v_1 \in V(X_U Y_U)$, then v_1 is ignorable.*

By the definition of $minSP$, the following lemma holds.

Lemma 4.5. *If $u \in V(X_U)$ is ignorable, then any vertex v on the left of u in X_U is ignorable. If $u \in V(Y_U)$ is ignorable, then any v on the right of u in Y_U is ignorable.*

Lemma 4.6. *Let (u, v) be an grey edge inside $X_U Y_U$. If u is ignorable then v is ignorable.*

We can reduce the range of $X_U Y_U$ based on Lemmas 4.4-4.6. Let l and r be the rightmost vertex in X_U and the leftmost vertex in Y_U such that there are grey edges (v_1, l) and (v_2, r) with $v_1 \in V(X_V Y_V)$ and $v_2 \in V(X_V Y_V)$. Let L and R be the vertices in X_U and Y_U that we are going to check (based on Lemma 4.6). Initially, we set $L = right(LEFT(X_U))$ and $R = left(RIGHT(Y_U))$. We can use the algorithm in Figure 5 to prune the segment X_UY_U in Y'. We claim that there always exists a grey edge (u, v) with $u \in V(X_UY_U)$ and $v \in V(X_VY_V)$. The proof is left to interested readers.

```
Algorithm 5: prune($X_U, Y_U, X_V, Y_V$)
1. Set $l = right(LEFT(X_U))$ and $r = left(RIGHT(Y_U))$.
2. Search every vertex $v \in V(X_VY_V)$ and find the the rightmost vertex $l$ in $X_U$
   and the leftmost vertex $r$ in $Y_U$ such that there are grey edges $(v_1, l)$ and $(v_2, r)$
   with $v_1 \in V(X_VY_V)$ and $v_2 \in V(X_VY_V)$.
3. Let $L = right(LEFT(X_U))$ and $R = left(RIGHT(Y_U))$.
4. Consider the grey edges $(L, u)$ and $(R, v)$. if $u \in V(X_U)$ ($v \in V(Y_U)$) and $u$
   ($v$) is on the right of $l$, then $l = u$ ($l = v$). if $u \in V(Y_U)$ ($v \in V(Y_U)$) and $u$
   ($v$) is on the left of $r$, then $r = u$ ($r = v$).
5. if ($l \neq L$) then $L = right(L)$. if ($r \neq R$) then $R = left(R)$. if ($l \neq L$ or
   $r \neq R$) then goto Step 4.
6. $l = right(l)$ and $r = left(r)$.
7. Move $r$ to the left until no short cycle is on the left of $r$ in $Y_U$. Move $l$ to the
   right until no short cycle is on the right of $l$ in $X_U$.
8. output: $[l, r]$.
```

Figure 5. Algorithm 5: Reducing the range of the $minSP$ in the crucial chromosome.

Theorem 4.2. *If algorithm $prune(X_U, Y_U, X_V, Y_V)$ returns l and r as the two ends of the connecting edge in $X_2X_UY_UY_2$, then $\rho(u_R, v_R)$ is valid. If l or r is not the end of the connecting edge, $\rho(u_R, v_R)$ is not valid. In this case, $\rho(u_R, v_R)$ produces a new $minSP$ contained in the interval $[l, r]$. Moreover, $[l, r]$ is a nested sub permutation in this case.*

Now, we can use $findEdge()$ and $prune()$ to find a valid grey edge as in Figure 6.

```
Algorithm 6: findValid($G_{AB}$)
Output   $(u_R, v_R)$.
1. Arbitrarily select a proper grey edge $(u, v)$ in $G_{AB}$ and apply the translocation.
2. Use algorithm 3 to test if any of the two new chromosomes contains a new
   $minSP$. if no new $minSP$ is found then return $(u, v)$ and stop.
3. Let $X_MY_M$ be the new $minSP$ found in Step 2.
4. Call $findEdge(X_MY_M)$ to get $(u_R, v_R)$, and determine $X_U, Y_U, X_V, Y_V$.
5. Call $prune(X_U, Y_U, X_V, Y_V)$ to get $[l, r]$. if $l = RIGHT(X_U)$ and $r =$
   $LEFT(Y_U)$ then return $(u_R, v_R)$ and stop.
6. Update $X_M = [l, x]$ and $Y_M = [y, r]$, where $x$ and $y$ are the two ends of the
   connecting edge and goto Step 4.
```

Figure 6. Algorithm 6: Finding a valid proper grey edge (translocation) in $O(n)$ time.

Theorem 4.3. *Algorithm 6 finds a valid proper grey edge (translocation) in $O(n)$ time.*

References

1. V. Bafna and P. Pevzner. Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of x chromosome. *Molecular Biology Evolution*, 12:239–246, 1995.
2. A. Bergeron. A very elementary presentation of the hannenhalli-pevzner theory. *Proceedings of the Twelfth Annual Symposium on Combinatorial Pattern Matching*, pages 106–117, July 2001.
3. T. Dobzhansky and A.H. Sturtevant. Inversions in the chromosomes of *drosophila pesudoobscura*. *Genetics*, 23:28–64, 1938.
4. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 178–189, 1995.
5. H. Kaplan, R. Shamir, and R. E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 344–351, January 1997.
6. D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B.F. Lang, and R. Cedergen. Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences of the United States of America*, 89:6575–6579, 1992.
7. S. Hannenhalli, C. Chappey, E.V. Koonin, and P. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30:299–311, 1995.
8. S. Hannenhalli and P. Pevzner. Transforming men into mice: Polynomial algorithm for genomic distance problem. *FOCUS'95*, pages 581–592, 1995.
9. S. Hannenhalli and P. Pevzner. Towards a computational theory of genome rearrangement. *Lecture Notes in Computer Science Vol.1000*, pages 184–202, 1995.
10. S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. *Proceedings of the Sixth Annual Symposium on Combinatorial Pattern Matching*, pages 162–176, July 1995.
11. S. Hannenhalli and P. Pevzner. To cut ... or not to cut (applications of comparative physical maps in molecular evolution). *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313, January 1996.
12. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999.
13. J. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, January 1995.
14. G. Li, X. Qi, X. Wang, B, Zhu, A linear-time algorithm for computing translocation distance between signed genomes, *CPM'2004*.
15. D. A. Bader, B. M.E. Moret, and M. Yan, A linear-time algorithm for computing inversion distance between signed permutation, Journal of Computational Biology, vol. 8, pp.483-491, 2001.
16. D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Proceedings of the Seventh International Workshop on Algorithms and Data Structures*, pages 365–376, August 2001.
17. G. Tesler, Efficient algorithms for multichromosomal genome rearrangements, Journal of Computer and System Sciences, vol. 65, pp. 587-609, 2002.
18. D. Sankoff and J.H. Nadeau, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families. Volume* 1 *of Series in Computational Biology*, pages 225–241. Dordrecht, NL. Kluwer Academic Press, 2000.
19. D. Sankoff, N. El-Mabrouk. Genome Rearrangement in T. Jiang, Y. Xu and Q. Zhang editors, *Current Topics in Computational Molecular Biology*, pages 132-155. The MIT, Press, 1992.
20. D.M. Zhu and S.H. Ma. Improved polynomial-time algorithm for computing translocation distance between genomes. *The Chinese Journal of Computers (in Chinese)*, 25(2):189–196, 2002.