Exploring the Solution Space of Sorting by Translocations

Amritanjali\textsuperscript{a}, Shivani Anand\textsuperscript{b}, G.Sahoo\textsuperscript{c}\*  
\textsuperscript{a}Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi-835215, India  
\textsuperscript{b}Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi-835215, India  
\textsuperscript{c}Department of Information Technology, Birla Institute of Technology, Mesra, Ranchi-835215, India

Abstract

Genome sorting is a way to trace back the genomes of two species to their common ancestor. The study of genome rearrangements allows to better understand the process of evolution and is an important tool in comparative genomics. Given two genomes A and B, the goal is to find a shortest sequence of exchanges of non-empty chromosome ends that transform A into B. The length of such a shortest sequence is called the translocation distance between A and B, and the problem of finding the optimal sequence of translocations is called the sorting by translocations problem. For most problem instances, however, several minimum length sequences or optimal sequences of translocation exist, and in the absence of any additional information, no one is of greater value than the others. The problem of finding all optimal solutions for sorting by translocations is thus a natural generalization of sorting by translocations. In this paper, an algorithm is presented to find all solutions for sorting by translocations.

© 2012 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of the Program Committee of CSBio 2012.

Keywords: genome rearrangements, sorting algorithms, translocations

1. Introduction

Every organism has specific set of chromosomes in their cells. Each chromosome has specific gene sequence. Each gene is associated with a sign, which represents its direction of transcription. Different organisms are found to be related if they share similar genes, which were inherited from a common ancestor (orthologous genes). During genes’ evolution point mutations and short insertions and deletions cause local changes, whereas evolution at the level of genome proceeds by large scale operations, such as reversals and

\* Corresponding authors.  
E-mail addresses: amritanjali@bitmesra.ac.in (Amritanjali), shivanianand.bit@gmail.com (Shivani Anand), gsahoo@bitmesra.ac.in (G. Sahoo).
translocations, which rearrange the order and direction of genes along the genome. Genomes of two closely related species have almost same set of genes, but the genes may be arranged in a different sequence or be moved between chromosomes. Such differences in gene orders are the results of rearrangement events that are common in molecular evolution. For example, in unichromosomal genomes, the most common rearrangement events are reversals, in which a contiguous interval of genes is put into the reverse order. For multichromosomal genomes, translocation is the most common rearrangement event, after reversal. Reversals reverse both the order and the direction of the genes in a segment inside a chromosome. Translocations exchange tails between two chromosomes. A translocation considered here is always reciprocal, i.e., none of the two tails is empty. To find a most parsimonious sequence of reciprocal translocations (reversals, respectively) that transforms one genome into another is referred to as the problem of sorting by reciprocal translocations.

We have designed an algorithm to enumerate all possible optimal solutions for sorting genomes using translocation. The existing algorithms give a single solution. However, there can be multiple optimal solutions for sorting by translocations. Giving just one solution may not give the correct picture of the actual scenario in the evolution of the genome. By taking into account different biological constraints biologists can analyze the various rearrangement scenarios given by our method and identify the most plausible one.

2. Notations and Definitions

Genome Rearrangements change the order of segments in a genome, and may also change the DNA strand in which the segment lies. Thus, we identify the gene sequence with the integers 1, 2... n with a plus or minus sign to indicate the strand they lie on.

2.1. Representation of Genome

We consider a unichromosomal genome to be a sequence of n genes. The genes are represented by numbers 1, 2... n. The two orientations of gene i are represented by i and -i. The order and orientation of genes of one species in relation to the other is represented by a signed permutation of size n, i.e. by permutation of a set {1, 2,...,n}, where each number 1,2,…n is associated with a sign ‘+’ or ‘-’.

While comparing the genomes of two different species, one is taken in ascending order (usually the source genome) and our aim is to sort the other also in ascending order (the destination genome).

The permutation with all the genes having ‘+’ sign and sorted in ascending order (i.e. {1,2,…n}) is known as Identity Permutation. A multichromosomal genome consists of n genes spread over m chromosomes. For example, the following genome consists of 12 genes in three chromosomes: A = {(7, -2, 8, 3), (5, 9, -6, -1, 12), (11, 4, 10)}.

2.2. Genome Rearrangements

Translocation is a prevalent rearrangement event in multichromosomal species. In translocation exchange of tails between two chromosomes takes place. A translocation is reciprocal if none of the exchanged tails is empty.

Let X = (x1, . . ., xi-1, xi, . . . , xm ) and Y = (y1, . . . , yj-1, yj, . . . , yn ) be two chromosomes in a signed genome. A translocation swaps the segments in the two chromosomes. There are two ways to perform the translocation. A prefix-prefix translocation ρpp (X, Y, i, j) transforms X and Y into chromosomes (x1, . . . , xi-1, yj, . . . , yn ) and (y1, . . . , yj-1, xi, . . . , xm ). A prefix-suffix translocation ρps (X, Y, i, j) transforms X and Y into chromosomes (-yn, . . . , -yj , xi, . . . , xm) and (y1, . . . , yj-1, -xi-1, . . . , -x1 ).

For example, a prefix-suffix translocation between chromosomes 1 and 3 of the genome {(7, -2, 8, 3), (5, 9, -6, -1, 12), (11, 4, 10)} results in the rearranged genome {(7, -2, 8, -4, -11), (5, 9, -6, -1, 12), (-3, 10)}. The minimum number of rearrangements required to transform one genome into another is a natural
measure of evolutionary distance. If each of the two genomes has exactly one copy of each of \( n \) genes, then the genomes can be represented by permutations of size \( n \), and their evolutionary distance is equal to the minimum number of rearrangements required to transform one permutation into the other.

2.3. Cycle Graph

Let A and B be a pair of co-tailed genomes. We assume that both A and B contain genes \( (1, 2\ldots n) \). Let \( n \) and \( N \) be number of genes and number of chromosomes in A (equivalently, B). The cycle graph of A and B, denoted by \( G(A, B) \) is a bicoloured graph and is defined as follows. For a chromosome \( X = (x_1, x_2, \ldots, x_m) \), replace each gene \( x_i \) by a pair of ordered vertices \((h(x_i), t(x_i))\). If the sign of \( x_i \) is '+' then \( h(x_i) = x_i^0 \) and \( t(x_i) = x_i^1 \). Otherwise, if the sign of \( x_i \) is '-' then \( h(x_i) = x_i^1 \) and \( t(x_i) = x_i^0 \). In this way, each chromosome \( X \) in the genome A (B) is changed to \( X' = (h(x_1) \ t(x_1) \ h(x_2) \ t(x_2) \ldots \ h(x_m) \ t(x_m)) \), an ordered list of vertices corresponding to the genes present in the chromosome. The set of vertices in the cycle graph is \( \{1, 2, \ldots, n\} \). For every two genes \( x_i \) and \( x_j \) such that \( x_j \) immediately follows \( x_i \) in some chromosome of A (respectively, B) add a black (respectively, grey) edge between vertices \( t(x_i) \) and \( h(x_j) \). There are \( (n-N) \) black edges and \( (n-N) \) grey edges in \( G(A, B) \). An example cycle graph is shown in Fig. 1. Every vertex in the \( G(A, B) \) has degree 2 or 0.

The cycle graph \( G(A, B) \) is uniquely decomposed into cycles with alternating grey and black edges. A grey edge \((x_i, x_j)\) is external if the genes \( x_i \) and \( x_j \) belong to different chromosomes of A, otherwise it is internal. An internal grey edge is oriented if it connects two genes with different signs in A, otherwise it is unoriented. The cycle containing edges whose end points are in the same chromosome is called Internal Cycle. The cycle containing edges whose end points belong to different chromosomes is called External Cycle. Clearly, the number of cycles is maximised when A is identical to the target genome B.

![Fig. 1. The cycle graph \( G(A,B) \), where \( A = [(1,-3,2,-6,7);(5,4)] \) and \( B = [(1,2,3,4);(5,6,7)] \). The grey edges are shown with dotted lines.](image)

2.4. Testing Conditions

As we are considering only reciprocal translocation, so number of genes as well as number of chromosomes remains unchanged after the translocation operation. Reciprocal translocation is a type of balanced genome rearrangement. Here, we give the necessary and sufficient conditions that one genome can be transformed into the other.

Considering the source genome A and the destination genome B, A can be transformed into B by translocation if and only if [1]:

1) The two genomes contain the same set of genes;
2) The two genomes contain the same number (must be at least 2) of chromosomes;
3) The two genomes have the same set of ending (either head or tail) genes;
4) For any gene \( g \) that is an ending gene in \( A \), (1) if \( g \)'s sign in \( A \) is different from that in \( B \), then \( g \) must be a head in one genome and a tail in the other; (2) if \( g \) has the same sign in both \( A \) and \( B \), then \( g \) must be either a head in both genomes or a tail in both genomes.

3. Sorting By Translocations and its Solution Space

For a given pair of genomes, the translocation distance is unique, but there are usually several possible translocation scenarios with this distance. Given two multi chromosomal genomes \( A \) and \( B \), our aim is to find a shortest sequence of translocations that transform \( A \) into \( B \). There can be multiple optimal solutions for this. To the best of our knowledge, this is the first attempt to explore the solution space of sorting by translocations. Our work is motivated by the work done by M.D.V. Braga [6] that enumerates all possible sorting sequences of reversals. It will be useful to describe first the available algorithm to understand the basis of sorting by translocations. We have then presented our algorithm.

S.Hannenhalli and P.Pevzner [2] gave the polynomial time algorithm for sorting by translocations problem where a translocation is called valid if it decreases the translocation distance. To the best of our knowledge, this is the first attempt to explore the solution space of sorting by translocations. Our work is motivated by the work done by M.D.V. Braga [6] that enumerates all possible sorting sequences of

Let \( \Delta c \) denote the change in the number of cycles after performing a translocation \( \rho \) on \( A \). Then \( \Delta c = \{-1, 0, +1\} \) [3].

- Proper Translocation: One which increases the number of cycles by 1 i.e., \( \Delta c = 1 \).
- Bad Translocation: One which decreases the number of cycles by 1 i.e., \( \Delta c = -1 \).
- Improper Translocation: One in which the number of cycles remain unchanged i.e., \( \Delta c = 0 \).
- Valid Translocation: One which reduces the translocation distance by 1.

For an external grey edge 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 (non-trivial) cycle into two and thus is a proper translocation and the other one is improper. Every external grey edge \( e \) defines one proper translocation that cuts the black edges adjacent to \( e \). If a proper translocation does not produce new non-trivial internal components then it is valid. Such a translocation turns an external cycle of length \( l \) into an external cycle of length \( l-1 \) and a short cycle.

Lemma 1: \( \Delta c = n-N \), iff \( A \) is identical to the target genome.
After sorting we get (n-N) trivial cycles each consisting of a pair of grey and black edges.

Lemma 2: For a translocation \( \rho \), \( \Delta c = 1 \), iff \( \rho \) is proper, \( \Delta c = 0 \), iff \( \rho \) is improper and \( \Delta c = -1 \), iff \( \rho \) is bad.

Lemmas 1 and 2 imply the following.

Theorem: For an arbitrary genome \( A \), the lower bound on the translocation distance is given by

\[
\text{d (A)} \geq n-N-c. \quad [2]
\]

where

- \( A \)=source genome
- \( c \)= number of disjoint cycles in the cycle graph
ALGORITHM (Hannenhalli’s algorithm, from [2])

1. while A is not identical to the target genome do
2.   if there is a proper translocation in A then
3.     select a valid proper translocation $\rho$
4.   else
5.     select a valid bad translocation $\rho$
6.   end if
7.   $A \leftarrow A \rho$
8. end while

The main assumption behind the algorithm is that if there exists a proper translocation, then there always
equals a valid proper translocation [3]. Also, for every genome there exists a valid translocation. If there is no
proper translocation, there must be a valid bad translocation. A bad translocation cuts two black edges
belonging to different cycles.

The overall running time of this algorithm is $O(n^3)$. This was improved to $O(n^2 \log n)$ algorithm in [4].
In 2004, Li and Zhu et al. [5] gave a linear time algorithm to compute the minimum number of translocation
operations. However, that algorithm cannot give the optimal sequence of translocations.

The existing algorithms give one optimal sequence of translocations to transform the given source genome
to target genome. However, there can be multiple optimal solutions for sorting by translocation. Giving just one
solution may not give the correct picture of the actual scenario in the evolution of the genome. At each step we
may get more than one choice of cut-points. Considering different pairs of cut-points at each step we can get
multiple optimal sequences of translocations. Let us understand the multiple solutions of sorting genomes with
the help of an example.

Source genome (A):

X: 1 3 9
Y: 7 8 4 5 6
Z: 10 2 11 12 13

Target genome (B):

X: 1 2 3 4 5 6
Y: 7 8 9
Z: 10 11 12 13

Cut points are taken on the black edges in different chromosomes involved in a cycle. In the above Fig. 2 we
can see for the cycle involving chromosomes X and Z, in the Z chromosome there are two options for selecting
cut points, one between edges 10 and 20 and the other between 21 and 110. Selecting either will give the same
result for translocation distance. Thus, for this external cycle, there are 2 solutions for sorting by translocations.
For the other external cycle involving X and Y, there is another solution. So, all together for this example
there are 3 solutions for translocations.
3.1 The Algorithm

In this paper we present our algorithm for finding multiple solutions for sorting genomes by translocations. We have divided the algorithm into two parts. The first part (Algorithm 1) finds all possible valid translocations and the second part (Algorithm 2) enumerates the different possible translocation scenarios.

---

**Algorithm 1**

Construct cycle graph for A.

Find all external cycles $E$

$T \leftarrow 0$

For each external cycle in $E$

- Add all valid and proper translocation to $T$

End for

If ($T=0$)

Add a valid bad translocation to $T$

Return $T$

---

Algorithm 2 is an adaptation of the Braga algorithm [6] that enumerates all possible sorting sequences of reversals. In comparative genomics sorting by reversals has been the topic of several literatures [7-8]. In 2002, Siepel [9] proposed $O(n^3)$ algorithm to list all sorting reversals. In [6], the Siepel algorithm is applied repetitively to generate all possible sorting sequences of reversals.
Algorithm 2 Enumerating all optimal sorting sequences for a permutation

D ← translocation distance

T ← set of possible next translocations for A [Returned by algorithm1]

S ← T

For each integer i from 2 to D do

S1 ← 0

For each s in S do

A1 ← A*s [apply the sequence of translocations in s]

T ← set of possible next translocations for A1 [Returned by algorithm 1]

For each translocation t in T do

s1 ← s.t [concatenate with the previous sequence of translocations]

Insert s1 in S1

End for

End for

S ← S1

End for

Return S [complete set of translocations]

As we can get multiple pairs of cut-points at each step, so by taking different pairs of cut-points we generate all possible sorting sequences of translocations. Running Algorithm 2 for the example sequence gives 3 optimal sorting sequences as listed below:

Solution 1.

X:{1;3} ; Z:{2;11}
X:{1;11} ; Z:{10;2}
X:{3;9} ; Y:{8;4}

Solution 2.

X:{1;3} ; Z:{10;2}
X:{2;11} ; Z:{10;3}
X:{3;9} ; Y:{8;4}
Solution 3.

X: 3 9  Y: 8 4
X: 1 3  Z: 10 2
X: 2 11  Z: 10 3

In each row we have shown the position of cut-point in each of the two chromosomes that are involved in a particular translocation. Here we see that there can be multiple solutions for translocation depending upon the choice of the cut points.

For illustration purpose we have used a small example, however, we have tested our method with more complex problems. Consider another example,

Source Genome (A):

|   |   |   |   |   |   |
|---|---|---|---|---|---|
|X:| 1 | 7 | 8 | 2 | 14 | 15 | 16 |
|Y:| 5 | 6 | 9 | 4 |
|Z:| 13 | 3 | 10 | 11 | 12 |

Target Genome (B):

|   |   |   |   |   |   |
|---|---|---|---|---|---|
|X:| 1 | 2 | 3 | 4 |
|Y:| 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|Z:| 13 | 14 | 15 | 16 |

There are 22 optimal sorting sequences of translocations for this example sequence. Here, the translocation distance (d) is 4, the number of genes (n) is 16 and the number of chromosomes (N) is 3. The solution space increases with translocation distance and size of the permutation. In next section, we have analysed the theoretical complexity of the Algorithm 2.

3.2 Theoretical Complexity of Algorithm 2

The complexity of the algorithm depends upon time taken in finding all valid and proper translocation at each step and time taken to perform each translocation. A translocation can be performed in O(\(\sqrt{n \log n}\)) time [10]. Number of translocations that can be performed at each step depends upon the total number of external edges (cut-points) detected at each step. This number is bounded by (n-N), where N is the number of chromosomes. Thus, for d number of iterations, the total number of cut-points that can be generated is given by \(\sum_{k=1}^{d} (n-N)^k\). It takes O(n) to find a valid proper translocation. Therefore, Algorithm 2 has time complexity of O(\(\sqrt{n \log n} \times n \times \sum_{k=1}^{d} (n-N)^k\)). Since, translocation distance d is at most O(n), so the complete set of solutions can be generated in time O(\(\sqrt{n \log n} \times n \times (n-N)^n\)).

4. Conclusion

In this paper we explored the possibility of multiple solutions of sorting by translocations problem. Also we have designed an algorithm for Sorting by Translocations, which find all possible optimal solutions for sorting.
Instead of giving one optimal solution we are presenting all possible optimal solutions. The choice of solution will depend upon the presence of certain biological facts which need to be considered for a given problem domain.

The complexity of the algorithm is depending on the total number of solutions in the solution space. To efficiently handle permutations with large solution space, parallel approach may be appropriate. It is possible to implement our algorithm in parallel. We can explore the solution space using depth first approach for parallelization. It may be taken as future work. We also observed that some of the solutions consist of same pair of cut points; however, their sequences are different. This fact may be used to reduce the solution space of the problem to some extent. We have yet to implement it practically so that we can test with real data that are more complex in nature. Another future scope is to find the complete set of sorting sequence using both reversals and translocations. These results will be then more close to the true evolutionary scenario.

Acknowledgements

We would like to express our sincere gratitude to the Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi. During this work we have collaborated with many colleagues for whom we have great regard, and we wish to extend our warmest thanks to all those who have helped us with our work.

References

[1] Wang L, Zhu D, Liu X, Ma S, “An O (n^2) algorithm for signed translocation”, Journal of Computer and System Sciences 2005; 70: 284-299.
[2] Hannenhalli S and Pevzner P, “Polynomial algorithm for computing translocation distance between genomes”, Discrete Applied Mathematics 1996; 71: 137-151.
[3] Bergeron A, Mixtacki J and Stoye J, “On Sorting by Translocations”, Journal of Computational Biology 2006; 13: 567-578.
[4] Zhu D, Ma S, “Improved polynomial-time algorithm for computing translocation distance between genomes”, Chinese J. Comput. 2002; 25(2): 189-196.
[5] Li G, Qi X, Wang X, Zhu B, “A linear-time algorithm for computing translocation distance between signed genomes”, In: CPM 2004 Proceedings 2004; 109, LNCS, Springer Verlag: 323-332.
[6] Braga M D V, “baobabLUNA: the solution space of sorting by reversals”, Bioinformatics 2009; 25(14): 1833-1835.
[7] Hannenhalli S, Pevzner P, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, J. Assoc. Comput. Mach. 1999; 46: 1–27.
[8] Tannier E, Bergeron A and Sagot M, “Advances on Sorting by Reversals”, Discrete Applied Mathematics 2007: 155: 881-888.
[9] Siepel A, “An Algorithm to Enumerate Sorting Reversals for Signed Permutations”, Journal of Computational Biology 2003; 10: 575-597.
[10] Ozery-Flato M, Shamir R, “An O (njscn3/2 ) algorithm for sorting by reciprocal translocations”, In: Proc. 17th Annual Symposium Combinatorial Pattern Matching (CPM 2006); 258-269.