Multi-Chromosomal Genome Sorting

Pranav Kumar[1]
[1]*Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi-835215, India*

G. Sahoo[2]
[2]*Department of Information Technology, Birla Institute of Technology, Mesra, Ranchi-835215, India*

***Abstract*— Evaluation of genome rearrangements is an important effort in comparative genomics. A major problem in this field is finding a sequence of genome rearrangements that transforms one genome into another. Finding the reversal & translocation distance and finding one optimal sequence of reversals to transform a genome into another are useful tools to analyse real evolutionary scenarios but can be more useful by finding all possible solutions. However, the number of different optimal sorting sequences is usually huge and some solutions should be taken in consideration in order to obtain a more accurate analysis. Reversal and translocation are both common rearrangement events in the evolution of mammalian species. Given two multi-chromosomal genomes A and B, the problem of sorting by reversals and translocations is to find a shortest sequence of reversals and translocations and its multiple solutions that transforms A into B. This problem is currently solved by a reduction to the problem of sorting by reversals and translocations separately, but here both are applicable over the same problem at a time. In this paper, we will present a new algorithm for sorting by reversals and translocations that explicitly treats translocations and reversals as distinct operations and find the multiple solutions which is better theoretical and practical than just a single solution finding method. We give examples where the result is more relevant than a unique optimal solution or the list of all solutions.**

***Keywords*-genome rearrangement; reversal; translocation; algorithm;**

I. INTRODUCTION

Genome is the set of chromosomes. Two genomes may have many genes in common, but the genes may be arranged in a different sequence or be interchanged between chromosomes. Such differences in gene orders are the results of rearrangement events that are common in molecular evolution. For example, in uni-chromosomal genomes, the most common rearrangement events are reversals, in which a adjoining set of genes is put into the reverse order. For multi-chromosomal genomes, the most common rearrangement events are reversals, translocations, fissions, and fusions, there are three basic operations: *reversal*, t*ranslocation* and *transposition*. In this paper, we focus on the reversal and translocation operations. Reversal and translocation are both common rearrangement events in the evolution of species. Reversal, break and invert the order and the direction of transcription of the genes in a segment inside a chromosome and translocation interchange the order in 2 different chromosomes. A reversal is *internal* if it does not include ends of the chromosomes. Translocation exchange tails between two chromosomes.

Comparison of genome sequences through sorting techniques can divulge significant information about their development. Assumption is that genomes share the same set of genes with no duplications; a genome can be represented by a signed permutation. Here we take two genome sequences that have almost identical gene sequences. By applying various rearrangement operations (reversal and translocation) we try to transform one sequence into another.

Given two multi-chromosomal genomes A and B, the problem of *sorting by reversals and translocations* is to find a shortest sequence of reversals and translocations that transforms A into B and then repeat the process for finding more solutions. The minimum steps in which the sequence can be sorted called the rearrangement

distance between A and B, denoted as d(A,B). The genomic sorting problem where the allowed rearrangement operations are reversals (respectively, translocations) is referred to as *sorting by reversals* (respectively, *sorting by translocations*). The first polynomial algorithm for sorting by Translocation was given by Hannenhalli and Pevzner [1]. After many subsequent improvements on the running time, the currently fastest algorithm developed by Tannier whose time complexity is $O(n^{3/2}\_\log(n))$ [7].

A major computational problem in the research of genome rearrangements is finding a most stingy sequence of rearrangements that transforms one genome into another and their corresponding number of rearrangements is called the genomic sorting and rearrangement distance between the two genomes. Genomic sorting gives rise to a range of captivating combinatorial problems, each defined by the set of allowed rearrangement operations and by the representation of the genomes. Sorting by reversal and translocation is currently solved by a reduction to the problem of sorting by reversal and translocation, where each reversal simulates either a reciprocal translocation or an internal reversal [1].

This algorithm is based on linking all the chromosomes into a permutation and after every steps a new permutation is being generated. Sometimes it may be possible that given certain biological constraints only a particular sorting sequence, out of the multiple solutions, satisfies them. In a recent paper [12], Xiao Yin and Daming Zhu has used the combined concept of: "Sorting by reversal and translocation that explicitly treats translocations and reversals as distinct operations and proposing a solution" and multiple solution was proposed by Braga, "baobabLUNA: the solution space of sorting by reversals" [13]. In this paper, we present a new algorithm for sorting by reversal and translocation which explicitly treats translocations and reversals as distinct operations via sorting by translocation and sorting by reversal theory, which gives a descriptive attempt to Xiao and Daming's algorithm[12] that we can provide all possible solutions for sorting by reversals and translocations. According to Siepel[6], there can be multiple sorting reversals. Braga[13] proposed an algorithm to enumerate the solution space of sorting by reversals. Also, Amritanjali [14] proposed an algorithm for multiple solutions of sorting by translocations. So by combining these two approaches we are proposing an algorithm to list all the optimal solutions for sorting by reversals and translocations.

II. DEFINATIONS AND NOTATIONS

Elaboration of following concept, here a gene will be represented by a signed integer, where the sign represents the direction of the gene and in a graph sign is shown by an diverging/converging arrow, a chromosome is a sequence of genes and a genome is a set of chromosomes. Let us assume a genome $A=((a_{11},a_{12},..,a_1n_1),(a_{21},a_{22},..,a_2n_2),\ldots(ak_1,ak_2,.aknk))$,$B=((b_{11},b_{12},..b_1n_1),(b_{21},b_{22},..,b_2n_2)\ldots(bk_1,bk_2.bknk)$), contain the same set of genes and that every gene appears in each genome exactly once means no duplication of genes. For an arbitrary sequence $M=m_1,m_2\ldots.m_k$ of genes, we will denote the reverse ordering of M by -M. i.e., $-M=-m_k,-m_{k-1}$ ….,$-m_1$.A chromosome Y is said to be indistinguishable to a chromosome $X= (x_1,x_2,\ldots.x_n)$ iff either Y=X or Y= -X . Genomes A and B are said to be indistinguishable (A=B) if the sets of chromosomes corresponding to A and B are the same.

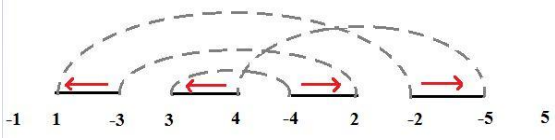
This section involves a basic background for the analysis of sorting by reversal and translocation. It follows to different types of notation of [3], [2], [1], [12]. Here we consider, a genome is a set of chromosomes. A chromosome is a sequence of genes. Each gene is identified by an integer with a sign of '+' or '-' which denotes its direction inside a chromosome. All genes in the genome are distinct. For example, {(1, 2, 3), (4, 5, 6, 7)} is a genome consist of 7 genes in two chromosomes. Genomes *A* and *B* are *identical* if they have the same set of chromosomes. The prefix-prefix and prefix-suffix operations allow us to treat x_1, x_2 ,…, x_n and $–x_n$,$-x_{n-1}$, …, $-x_1$ as the same chromosome. Thus, we can assume that gene 1 in A is always the left ending gene and the sign is always positive. (Otherwise, we can simply reverse the chromosome to fit our assumption.) Suppose that genes 1 to i have been put in the correct positions with positive sign and there exists a chromosome whose genes are all greater than i. If gene i and gene i+1 are not in the same chromosome, we can use one translocation operation to put gene i+1 in the correct position with positive sign and keep genes 1 to i "unchanged". If gene i and gene i+1 are in the same chromosome, then we can put gene i+1 in the correct position with positive sign and keep genes 1 to i "unchanged" by using two translocations. This process can continue until genes 1 to i are in the correct positions with positive sign, gene i and gene i+1 are in the same chromosome, say, Y, and there does not exist a chromosome whose genes are all greater than i.

A. Cycle Graph

A cycle of alternate grey and black edges is called cycle graph, Cycle that contains a pair of diverging black edges is a good cycle otherwise it is a bad cycle. A reversal/translocation is bad iff it cuts two black edges that belong to different cycles. These are the steps involve in constructing cycle graph as follows:

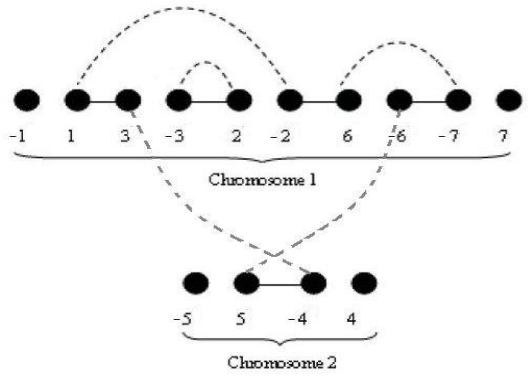
- For each element i of the permutation add two vertices (–i , +i) if (i > 0) otherwise (+i , -i) in that order.
- Leaving behind the leftmost and rightmost vertices in all the chromosomes, add black edges between adjacent vertices.
- Add grey edges (arc) between vertices +i to –(i+1), If cycle formed are within the same chromosome this gives diverging /converging edges which shows the direction by red arrow, where $0 \leq i \leq n$.
- Rearrangement distance (d) =n-N-c+k+f
  Where n= no of gene
  N= no of chromosomes
  c= no of cycle
  k= no of knots
  f=no of fortress

Example: {1, 3, -4, -2, 5}



Above example shows the cycle formed (1, -2, -5, 4, 3, -4, 2, -3) are within the same chromosome, so the diverging/converging edge is shown by red arrow. Grey edge are of 2 type, (1) edge connected within the same chromosomes called internal grey edge, (2) external edge if connected in different chromosomes. Let n_1 and n_2 be the number of genes and chromosomes in A and B. We will always assume that both A and B consist of genes {1,2,…,n}. The cycle graph of A and B, denoted G(A,B), is defined as follows. The set of vertices is $U^{n}_{i=1}\{+i, -i\}$. For every two genes i and j, where j immediately follows i in some chromosome of A(respectively, B), add a black (respectively, grey) edge (i, j) = (left(i), right(j)), where left(i) = -i if i has a positive sign in A (respectively B) and otherwise left(i) = +i, and right(j) = +j if j has a positive sign in A (respectively, B) and otherwise right(j) = -j.

Example: {(1, -3, -2, -6, 7) (5, 4)}



Here 'A' is a genome in which there are two chromosomes 1 and 2. A gray edge (c_1, c_2) between chromosome in cycle graph G(A,B) is *external* if c_1, c_2 belong to different chromosomes of A; otherwise is *internal*. An internal grey edge is *oriented* if it connects two genes with different signs in A; otherwise is *unoriented*. A cycle is *long* if it has more than 2 edges and *short* otherwise. An internal cycle is *oriented* if there exists an oriented grey edge in it and *unoriented* otherwise.

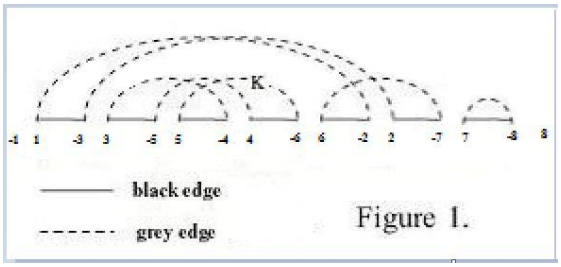


Figure 1.

B. Knots in Graph

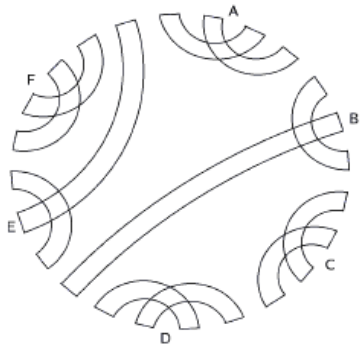
Considering a sequence I = x_i, x_{i+1},...,x_{j-1}, x_j in a chromosome of A. Let V (I) = $U_{i\leq k\leq j}\{-x_k, +x_k\}$, IN(I) = V (I)\{l(x_i), r(x_j)}. The sub graph induced by the vertex set IN(I) is an *internal component* if: (1) there is no edge (u, v) such that u ∈ IN(I), v /∈ IN(I), (2) the sub graph corresponding to I has at least one long cycle. Obviously, each internal component consists of one or more internal cycles. An internal component is *oriented* if it has at least one oriented cycle and *unoriented* otherwise. Short cycles are trivial and we exclude them.

A *minimal knot* is an unoriented internal component not containing any other unoriented internal components. A *greatest knot* is an unoriented internal component K satisfies the following two conditions: (1) K contains all the other unoriented internal components in G(A,B); (2) any vertex in K does not separate any two unoriented internal components. An *knot* is either a minimal knot or a greatest knot. A knot K *protect* a non-knot U if deleting K transforms U from a non-knot into a knot (i.e. U is a knot in G(A,B) \ K). A knot is a *superknot* if it protects a non-knot. A knot is *simple* if it is not a superknot. G(A,B) contains a *fortress-of-knots* if it has an odd number of knots and all these knots are superknots. An example of fortress-of-knots is shown in Figure 1. The rearrangement distance between A and B is d(A,B) = n − N − c(A,B) + k(A,B) + f(A,B). Where c(A,B) be the number of cycles in G(A,B) and k(A,B) be the number of knots in G(A,B). Define f(A,B) = 1 if there exists a fortress-of-knots in G(A,B) and f(A,B) = 0 otherwise. Rearrangement distance is the minimum number of steps required to transform one genome into the other.

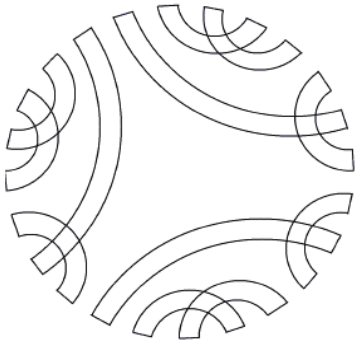
C. Hurdles and Fortresses

Hurdle is a bad component that does not separate any other bad components.A, F, C & D are hurdles and E & B are non-hurdles.A hurdle is called a ***super hurdle*** if it were eliminated, a non-hurdle would emerge as a hurdle; otherwise it is a simple hurdle.

F is a super hurdle and A & D are simple hurdles



A fortress exists iff there is an odd number of hurdles and all are super hurdles. According to the Siepel[6], Every unoriented component can be classi. ed as either a *hurdle* or a *protected nonhurdle*. A hurdle is an unoriented component that does not *separate* other unoriented components, and a protected nonhurdle is one that does. A component is said to separate two other components in a figure if, in a traversal is impossible to pass (in either circular direction) from a vertex belonging to 2nd vertex to a vertex belonging to 3rd without encountering a vertex belonging to 1st. Note that, while separation is primarily used with respect to unoriented components, the definition (as stated here) applies as well to benign components. A hurdle is called a *superhurdle* if, were it eliminated, a protected nonhurdle would emerge as a hurdle; otherwise, it is called a *simple hurdle*.



Smallest possible fortress

III. AIMS AND OBJECTIVES

Our aim is to extend the Yin and Zhu's 2009 algorithm for sorting genomes by reversals and translocations so that we can provide all possible solutions for sorting by reversals and translocations. According to Siepel's 2003, there can be multiple sorting reversals. Braga[13] proposed an algorithm to enumerate the solution space of sorting by reversals. Also, Sahoo and Amritanjali's 2012 proposed an algorithm for multiple solutions of sorting by translocations. So by combining these two approaches we are proposing an algorithm to list all the optimal solutions for sorting by reversals and translocations.

1. Reversal and Translocation are the most common rearrangement events in the evolution of mammalian species.

2. Given two multi-chromosomal genomes A and B, our aim is to find the shortest sequence of reversals and translocations that transform A into B. The length of such a sequence is called the *rearrangement distance* between A and B.

3. This problem is currently being solved by reduction to the problem of sorting by reversals, where each reversal is treated as an internal reversal or a translocation.

4. We have proposed an algorithm that will explicitly treat reversal and translocation as distinct operations and thus would be more similar to the natural scenario.

5. We have extended the algorithm to find all possible optimal solutions for translocation, given a sample permutation.

IV. NEW ALGORITHM FOR SORTING BY REVERSAL AND TRANSLOCATION

Our algorithm can sort a genome A into B using exactly d(A,B) number of operations and we will apply same step for finding all the solutions.

Let Δc, Δk, Δf denote the change in c, k and f after performing a reversal/translocation on A. Then $\Delta c \in \{-1, 0, 1\}$ [3],[2]. A reversal/ translocation is *proper* if $\Delta c = 1$ and *bad* if $\Delta c = -1$. A reversal/translocation ρ is *valid* if $d(A\cdot\rho,B) = d(A,B)-1$, i.e, $\Delta c-\Delta k-\Delta f = 1$. Our algorithm is based on finding d(A,B) valid reversals and translocations this is explained in three parts: eliminating unoriented internal components, eliminating oriented internal components, eliminating external cycles [12].

THE ENUMERATION OF THE SOLUTIONS

Genome of an organism is organized in a set of chromosomes. Each chromosome can be viewed as an ordered sequence of genes. In each gene sign 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**). Point mutations and short insertions and deletions cause local changes during a gene's evolution. In contrast, evolution at the level of the genome proceeds by large scale operations, such as reversals and translocations, which rearrange the order and direction of genes along the genome. Reversals most often reflect the differences between and within species.

The existing algorithms give one optimal sequence of reversal and translocations to transform the given source genome to target genome. However, there can be multiple optimal solutions for sorting by reversal and translocation. As for most problem instances, multiple minimum sequences of sorting by translocations exist, and by obtaining the complete set of solution can be useful in exploring the space of genome rearrangements in a better way. By the absence of any additional information we cannot decide which sequence of reversal and translocation is more closer to the actual scenario.The choice of solution will depend upon the presence of certain biological facts which need to be considered for a given problem domain.

Sometimes it may be possible that given certain biological constraints only a particular sorting sequence, out of the multiple solutions, satisfies them. At each step we may get more than one choice of cut-points as in case of translocation and more no of reversals. 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 2 4 7
Y: 6 5
Target genome (B):
X: 1 2 3 4 5
Y: 6 7

For this example Xiao and Daming[12] has used single solution, which may not be the most feasible solution for the biologist.

A. THE ALGORITHM

In this paper we present our algorithm for finding multiple solutions for sorting genomes by reversals and translocations. We have divided the algorithm into two parts. The first part (Algorithm1) finds all possible valid reversals and translocations and the second part (Algorithm 2) enumerates the different possible reversal and translocation scenarios.

Algorithm 1

Construct cycle graph for A.
Find all external cycles E
T ← 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 [13] that enumerates all possible sorting sequences of reversals and Amritanjali [14] for translocations. In comparative genomics sorting by reversals has been the topic of several literatures. In 2002, the Siepel algorithm [6] is applied repetitively to generate all possible sorting sequences of reversals. Above algorithm shows there can be 2 ways to generate all possible sorting sequences of translocations i-e proper and bad translocation [14] .

Algorithm 2

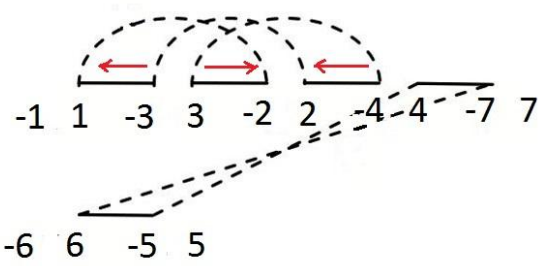
Input: Source genome A and Target genome B
Output: The set of all sequences of reversals and translocations sorting source genome A into target genome B
Assumption: Cycle graph created is free from huddles.
begin
d ← rearrangement distance between A and B
R← {$\rho_{(e1,e2)}$ | $\rho_{(e1,e2)}$ is an optimal 1−sequence of reversals for A composed of a pair {e_1,e_2} where e_1 is the leftmost element and e_2 is the rightmost element in the reversal interval ρ}
[Applying algorithm 1 to each chromosome in A]
T ← {$\rho_{(e1,e2)}$ | $\rho_{(e1,e2)}$ is an optimal 1−sequence of translocations for A composed of a pair {e_1,e_2} where first cut-point is after element e_1 and second cut-point is after element e_2 }
[Applying algorithm 2 to each chromosome in A]
S ← {R} U {T}
for each integer i from 2 to d **do**
S′←0 *[contains the i−sequences]*
for each s in S *[s is an (i−1) −sequence]* **do**
π′ ← π ∘ s *[apply the (i−1) sequences to π ,for each $\rho_{(e1,e2)} \in s$ if e_1 & e_2 are on same chromosome then Apply reversal else apply translocation]*
R← {$\rho_{(e1,e2)}$ | $\rho_{(e1,e2)}$ is an optimal 1−sequence for π′ } *[algo 1]*
T← {$\rho_{(e1,e2)}$ | $\rho_{(e1,e2)}$ is an optimal 1−sequence for π′ } *[algo 2]*
G← {R} U {T}
for each $\rho_{(e1,e2)} \in G$ **do**
s′ ← s. $\rho_{(e1,e2)}$
[concatenate ρ at the end of sequences]
insert s′ in S′ *[s′ is an i−sequence]*
end for
end for
S ←S′
end for
return S *[S is the final set of d−sequences]*
end

V. AN EXAMPLE

Source genome (A):
X: 1 3 2 4 7
Y: 6 5
Target genome (B):
X: 1 2 3 4 5
Y: 6 7

- In this section we present an example to illustrate the process sorting a genome into another, using the algorithm. Rearrangement distance(d) is the minimum number of steps in which the source genome can be converted into target genome.
- Rearrangement distance (d) =n-N-c+k
  where n= no of gene
  N= no of chromosomes
  c= no of cycle
  k= no of knots
  So, d= 7-2-2+1 = 4

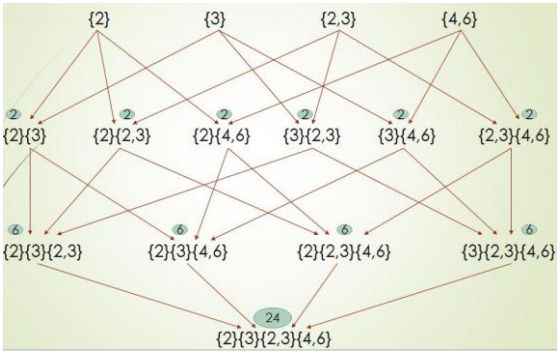
Below figure shows the cycle graph in which red line shows the diverging /converging points. Dotted line shows the grey edge. Plain line shows the black edge.



We have implemented the algorithm for sorting a given multi-chromosomal permutation by using the reversal & translocation operation. We have then extended the algorithm to give multiple optimal solutions (i.e. the solutions which give the sorted sequence in minimum possible number of steps). Multiple solutions are possible for the previous example genome. There can be 24 solutions are possible.
Solutions are:-

{2}{3}{2,3}{4,6}$_T$	{2}{3}{4,6}$_T${2,3}
{2}{2,3}{3}{4,6}$_T$	{2}{2,3}{4,6}$_T${3}
{2}{4,6}$_T${2,3}{3}	{2}{4,6}$_T${3}{2,3}
{3}{2}{2,3}{4,6}$_T$	{3}{2}{4,6}$_T${2,3}
{3}{2,3}{2}{4,6}$_T$	{3}{2,3}{4,6}$_T${2}
{3}{4,6}$_T${2,3}{2}	{3}{4,6}$_T${2}{2,3}
{2,3}{2}{3}{4,6}$_T$	{2,3}{2}{4,6}$_T${3}
{2,3}{3}{2}{4,6}$_T$	{2,3}{3}{4,6}$_T${2}
{2,3}{4,6}$_T${3}{2}	{2,3}{4,6}$_T${2}{3}
{4,6}$_T${2}{3}{2,3}	{4,6}$_T${2}{2,3}{3}
{4,6}$_T${3}{2}{2,3}	{4,6}$_T${3}{2,3}{2}
{4,6}$_T${2,3}{3}{2}	{4,6}$_T${2,3}{2}{3}

Here elements shown with suffix "T" shows translocation otherwise it is reversal.



Above figure shows traces or the path how actually 24 solutions have been achieved.

VI. NEED FOR FINDING MULTIPLE SOLUTIONS

- By the absence of any additional information we cannot decide which sequence of reversal and translocation is more close to the actual scenario.
- The choice of solution will depend upon the presence of certain biological facts which need to be considered for a given problem domain.
- For most problem instances, multiple minimum sequences of sorting by translocations exist, and by obtaining the complete set of solution can be useful in exploring the space of genome rearrangements in a better way.
- Sometimes it may be possible that given certain biological constraints only a particular sorting sequence, out of the multiple solutions, satisfies them.

VII. LIMITATIONS AND FUTURE SCOPE

- Here, we have performed sorting of genomes using reversal & translocation operation only.
- But, there can be some scenarios in which the input gene sequence may not be sorted only by the use of reversal & translocations.
- Reversals may be required to sort the genes within a chromosome.
- Reversal is a uni-chromosomal genome rearrangement operation. In reversals, an interval within a single chromosome is reversed or inverted.
- In case of multi-chromosomal genomes, after applying all translocations, if we don't get the sorted sequence, then reversal needs to be applied for each chromosome.

The real challenge in developing genome rearrangement algorithms is to propose algorithm which involves all the rearrangement operations that take place in the natural scenario.
A future work in this direction can be to use all operations for finding all optimal solutions.

VIII. CONCLUSION

At last we would like to conclude by saying that in the absence of any additional information we cannot decide which sequence of reversal and translocation is more close to the actual scenario.
So, apart from single solution of rearrangement operations we have to take into account multiple solutions for a given set of genome.

ACKNOWLEDGMENT

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] S. Hannenhalli, and P. Pevzner, "Transforming men into mice: Polynomial algorithm for genomic distance problem", *in Proc. 36th Ann. Symp. Foundations of Computer Science (FOCS 1995)*, pp. 581-592, 1995.
[2]S. Hannenhalli, "Polynomial algorithm for computing translocation distance between genomes", *Discrete Applied Mathematics*, vol. 71, pp. 137-151, 1996.
[3] S. Hannenhalli, and P. Pevzner, "Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals", *Journal of the ACM*, vol. 46, pp. 1-27, 1999.
[4] D. M. Zhu, and S. H. Ma, "Improved polynomial time algorithm for computing translocation distance between genomes", *The Chinese Journal of Computers*, vol. 25, pp. 189-196, 2002.
[5]G. Tesler, "Efficient algorithms for multi chromosomal genome rearrangements", *Journal of Computer and System Sciences*, vol. 65, pp. 587-609, 2002.
[6] Siepel A, "An algorithm to enumerate sorting reversals for signed permutations", *Journal of Computational Biology 2003*, vol. 10, pp. 575-597, 2003.
[7] E. Tannier, A. Bergeron, and M. Sagot, "Advances on sorting by reversals", *Discrete Applied Mathematics*, vol. 155, pp. 881-888, 2007.
[8]A. Bergeron, J. Mixtacki, and J. Stoye, "On sorting by translocations", *Journal of Computational Biology*, vol 13, pp. 567-578, 2006.
[9] L. S. Wang. D. M. Zhu, X. W. Liu, and S. H. Ma, "An $O(n^2)$ algorithm for signed translocation", *Journal of Computer and System Sciences*, vol. 70, pp. 284-299, 2005.
[10] M. Ozery-Flato, and R. Shamir, "An $O(n^{3/2} \log(n))$ algorithm for sorting by reciprocal translocations", *in Proc. 17th Ann. Symp. Combinatorial Pattern Matching (CPM 2006)*, pp. 258-169, 2006.
[11] M. Ozery-Flato, and R.Shamir, "Sorting by reciprocal translocations via reversals theory", *Journal of Computational Biology*, vol. 14, pp. 408-422, 2007.
[12] Xiao Yin and Daming Zhu, "Sorting genome by reversals and translocations", *School of Computer Science and Technology, Asia-Pacific Conference on Information Processing*, IEEE paper, 2009.
[13] Braga M. D. V, "baobabLUNA: the solution space of sorting by reversals", *Bioinformatics 2009*, vol. 25(14), pp. 1833-1835, 2009.

[14] Amritanjali, "Exploring the solution space of sorting by translocations", *Procedia Computer Science*, vol. 11, pp. 160-168, 2012.