World Academy of Science, Engineering and Technology
International Journal of Bioengineering and Life Sciences
Vol:2, No:2, 2008

Sorting Primitives and Genome Rearrangement in Bioinformatics: A Unified Perspective

Swapnoneel Roy, Minhazur Rahman, and Ashok Kumar Thakur

***Abstract*—**Bioinformatics and computational biology involve the use of techniques including applied mathematics, informatics, statistics, computer science, artificial intelligence, chemistry, and biochemistry to solve biological problems usually on the molecular level. Research in computational biology often overlaps with systems biology. Major research efforts in the field include sequence alignment, gene finding, genome assembly, protein structure alignment, protein structure prediction, prediction of gene expression and protein-protein interactions, and the modeling of evolution. Various global rearrangements of permutations, such as reversals and transpositions,have recently become of interest because of their applications in computational molecular biology. A reversal is an operation that reverses the order of a substring of a permutation. A transposition is an operation that swaps two adjacent substrings of a permutation. The problem of determining the smallest number of reversals required to transform a given permutation into the identity permutation is called sorting by reversals. Similar problems can be defined for transpositions and other global rearrangements. In this work we perform a study about some genome rearrangement primitives. We show how a genome is modelled by a permutation, introduce some of the existing primitives and the lower and upper bounds on them. We then provide a comparison of the introduced primitives.

***Keywords*—**Sorting Primitives, Genome Rearrangements, Transpositions, Block Interchanges, Strip Exchanges.

I. Introduction

GENOME is the entire DNA of a living organism. Gene is a segment of DNA that is involved e.g. in producing a protein, and its orientation depends on the DNA-strand that it lies on. Genome consists of chromosomes. Chromosomes are linear or circular.

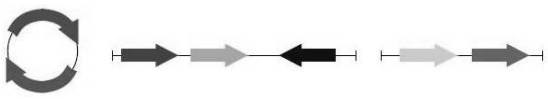

Fig. 1. Circular and Linear Chromosomes [12]

Swapnoneel Roy is affiliated to India Software Labs IBM India Pvt. Ltd, email: swapnoneel.roy@in.ibm.com
Minhazur Rahman is affiliated to Rational India Labs IBM India Pvt. Ltd, email: minhazur_r@in.ibm.com
Ashok Kumar Thakur is affiliated to India Software Labs IBM India Pvt. Ltd, email: ashok.thakur@in.ibm.com

The genome will be represented as a string of genes and will labeled as $1, 2, 3..., n$ for the sake of notational simplicity. One of the two genomic sequences will be treated as a base sequence for comparison with the other sequence. Since we are interested in the order of genes, we label each gene a unique number. This number can be unsigned. If a label of a gene is signed, for instance -5, it means that this gene is the reverse of another gene, which is labeled as 5.

While comparing two genomes, it has been found that these two genomes contain the same set of genes. But the orderings of the genes are different. For example, it was found that both human X chromosome and mouse X chromosome contain eight genes which are identical. They are labeled as 1, 2, ... , 8. In human, the genes are ordered as

$$4\ 6\ 1\ 7\ 2\ 3\ 5\ 8$$

and in mouse, they are ordered as

$$1\ 2\ 3\ 4\ 5\ 6\ 7\ 8.$$

Similarly, it was found that a set of genes in cabbage are ordered as

$$1\ -5\ 4\ -3\ 2$$

and in turnip, they are ordered as

$$1\ 2\ 3\ 4\ 5$$

Two genomes may have many genes in common, 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, the most common rearrangement events are reversals, translocations, fissions, and fusions, which are described later. The pairwise genome rearrangement problem is to find an optimal scenario transforming one genome to another via these rearrangement events.

The comparison of two genomes is significant because it provides us some insight as to how far away genetically these species are. If two genomes are similar to each other, they are genetically close; otherwise they are not. The question is how we measure the similarity of two genomes. Essentially, we measure the similarity of two genomes by measuring how easy it is to transform one genome to another by some primitives. The number of primitive steps needed to transform one genome into another is a measure for the evolutionary distance between two species.

Distance d(A,B): minimum number of primitive operations needed to transform genome A into genome B [12].

Some well known primitives for genome rearrangements are transpositions [5], reversals (aka inversions) [3], [4], strip (block) moves [1], [2], [6], block interchanges [7].

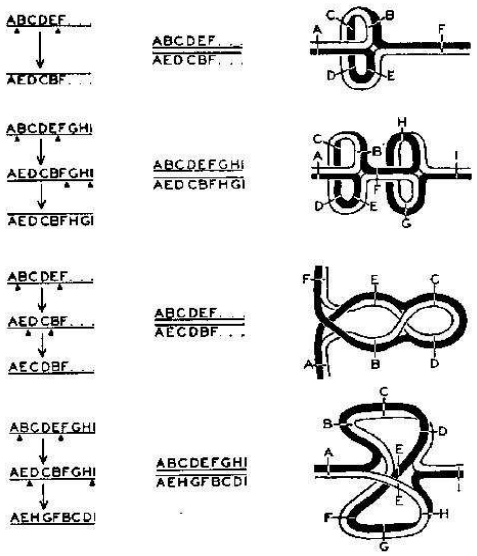

Fig. 2. Genome Rearrangements [12]

The strip move (aka block move) primitive is a special kind of transposition in which a strip (block) in the permuation is moves from one position to another to form a larger strip (block).

Since we are transforming a sequence of numbers into another sequence, without losing generality, we may always assume that the target sequence is 1 2 ... n. This is how genome rearrangement problems are viewed as sorting problems. We try to convert an arbitrary permutation into the identity permutation by using the minimum number of such primitive operations. These sorting problems are combinatorial optimization problems in which the number of steps required to sort an arbitraty permutation is optimized. While the sorting by reversal (aka inversion) problem and the sorting by strip (block) moves have been proven to be NP-Hard, the computational complexity of the sorting by transposition problem still remains open. The sorting by block interchanges problem is one of the very few such problems which has been proven to be polynomially solvable. An $O(n^2)$ algorithm exists for this problem.

The similarity between two sequences will be measured by the minimum number of operations to transform a sequence into another. Since the target sequence is always $1, 2, ..., n$, we view the problem as a sorting problem. But this is not a usual sorting problem which we are familiar with. Our sorting problem is to sort a sequence in such a way that the number of operations is minimized. In other words, we are interested in finding algorithms which always sort a sequence with minimum number of operations. Here we study some of the primitives defined for genome rearrangement. The primitives we choose for study are transpositions, block moves, block interchanges and strip exchanges. The next chapters discuss these primitives one by one. We discuss the lower and upper bounds and the computational complexity of the primitives. Finally we give a comparison as how these primitives perform in case of some arbitrary permutaions and what more could be done on these primitives.

II. Preliminaries

Definition 1 (Permutation)**.** *A permutation π is a bijection defined on the set $\{1, \ldots, n\}$. We use the string view of a permutation, that is, $\pi_1, \ldots, \pi_n$. Here, π_i is the image of i under π. We use id_n to denote the identity (sorted) permutation of n elements; $id_i = i, 1 \leq i \leq n$.*

As mentioned earlier, in our model:

1) A gene is represented by an unique integer.
2) A genome is represented by a permutation of integers
3) The target permutation is assumed to be the identity permutation.

Additionally, for the sake of simplicity, we impose the following constraints for the input permutations:

1) *No Gene Duplications*: The input sequence cannot contain two identical numbers. For instance, it cannot contain two 5s.
2) *No Gene Inversions*: No negative number should appear in the input sequence.
3) *No Gene Additions or Deletions*: If i and j appear in the sequence and $i \leq k \leq j$, k must appear in the sequence. That is, we do not allow the case where 5 and 7 appear, but 6 does not appear.

Definition 2 (Genome rearrangement algorithm)**.** *A genome rearrangement algorithm is modelled as a sorting problem by considering the target permutation as the sorted permutation. The lesser the number of primitives it takes for an algorithm, the better it is. The lower bound of a primitive is called the primitive distance.*

Generally the genomic rearrangement algorithms work in the following way:

Algorithm 1

Input: Two permutations π and id
Output: The approximate or exact primitive distance $d(\pi)$ between π and id_n
$d(\pi) = 0$.
while $(\pi \neq id)$ **do**
  Perform a primitive move
  $d(\pi) = d(\pi) + 1$
end while
Output the distance $d(\pi)$

Now we shall discuss the different primitives in details.

III. Transposition

In this section, we introduce the primitive transpositions and state certain lower bounds pertaining to it.

Definition 3 (Transpositions)**.** *Let π be a permutation on n elements (1,2,....,n), written as a string $\pi_1\pi_2...\pi_n$. The order of genes in a genome is represented by any such permutation. For a permutation π, a $transposition$ $\rho(i, j, k)$ (defined for all $1 \leq i < j \leq n+1$ and all $1 \leq k \leq n+1$ such that $k \notin [i, j]$) "inserts" an interval $[i, j-1]$ of π between π_{k-1} and π_k.*
The transposition primitive might also be viewed in an alternate way: For a permutation π, a $transposition$ $\rho(i, j, k)$ (defined for all $1 \leq i < j \leq n+1$ and all $1 \leq k \leq n+1$ such that $k \notin [i, j]$), "swaps" the substrings $\pi_i\pi_{i+1}...\pi_{j-1}$ and $\pi_j\pi_{j+1}...\pi_{k-1}$ in the permutation.

Given permutations π and σ, the $transposition\ distance\ problem$ is to find a series of transpositions $\rho_1,\rho_2,....,\rho_t$ such that $\pi.\rho_1.\rho_2....\rho_t=\sigma$ and t is minimum. t is called the transposition distance between σ and π. $Sorting\ \pi\ by\ transpositions$ is the problem of finding the transposition distance $d(\pi)$, between π and the identity permutaition id.

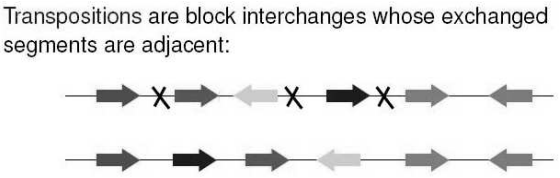

Fig. 3. The Transposition Primitive [12]

1) Lower bounds for the problem:

Definition 4 (Breakpoint)**.** *For all $0 \leq i \leq n$ in a permutation, there is a breakpoint between π_i and π_{i+1} if $\pi_{i+1} \neq \pi_i + 1$. For instance the permutation 0345216 with breakpoints added is $0, 345, 2, 1, 6$. The identity permutation id_n does not contain any breakpoint.*

2) Transposition cycle graphs:

Definition 5 (Cycle graphs)**.** *A cycle graph of a permutation π, denoted by $g(\pi)$, is a directed edge color graph with vertex set $\{0, 1, 2, ..., n, n+1\}$ and edge set defined as follows: for all $1 \leq i \leq n+1$, gray edges are directed from $i-1$ to i and black edges from π to $\pi - 1$ [5].*

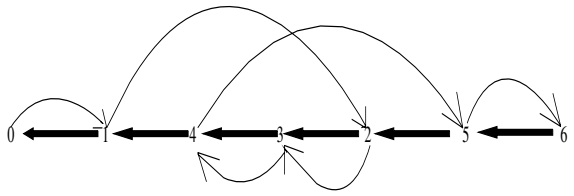

Fig. 4. A cycle graph

As an example we show the cycle graph for the permutation $1\ 4\ 5\ 2\ 3$ in the figure 1.

Definition 6 (Alternating cycles)**.** *An alternating cycle of a cycle graph is a cycle where each pair of adjacent edges are of different colors. the length of an alternating cycle is defined to be the number of black edges in the cycle. We call an alternating cycle with k black edges a $k-cycle$. Also is a cycle contains odd (even) number of black edges, it is termed as an odd (even) cycle.*

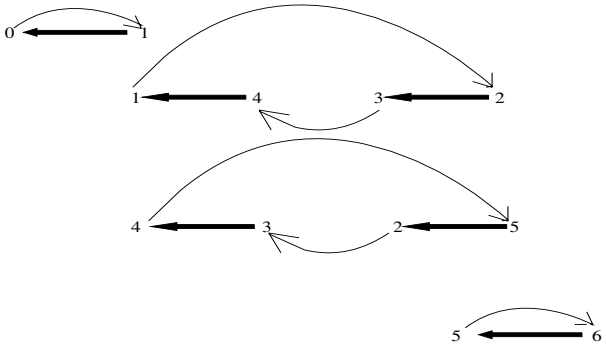

Fig. 5. Alternating cycles

As an example we show the alternating cycles of the graph of figure 1 in the figure 2.

Theorem 7. $t(\pi) = \frac{n+1-C_{odd}(\pi)}{2}$, *where $t(\pi)$ is the transposition distance n is the number of elements in permutation π and $C_{odd}(\pi)$ is the number of odd alternating cycles in the cycle graph of π.*

The above lower bound for sorting by transposition is a tighter lower bound and has been formulated in [5].

The computational complexity of sorting by transposition is still open. The best known algorithm for this problem has an approximation ratio of 1.375.

Definition 8 (Block Interchanges)**.** *The* block interchanges *primitive was designed by D A Christie in [7]. A* block *was defined here to be any substring of the given permutation. The block interchanges primitive interchanges the positions of any two blocks in the permutation. The sorting by block interchanges problem was to find an sequence of block interchanges moves required to sort any permutation.*

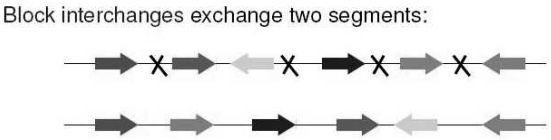

Fig. 6. The Block Interchange Primitive [12]

A. Lower bounds for the problem

Theorem 9 (D A Christie [7])**.** *Block interchanging distance $bi(\pi) = \frac{n+1-C(\pi)}{2}$ where $bi(\pi)$ is the block interchanging distance n is the number of elements in permutation π and $C(\pi)$ is the total number of alternating cycles in the cycle graph of π.*

The primitive transposition is actually a special kind of block interchange where the substrings interchanged are adjacent.
The sorting by block interchanges problem has been proved to be polynomially solvable in [7] and a $O(n^2)$ algorithm has been designed there.

IV. Strip moves

In this section, we introduce the primitive strip moves aka block moves and state certain lower bounds pertaining to it

Definition 10 (Strip)**.** *A strip in π is a maximal substring which is also a substring of the identity permutation id. For example, in the permutation 825639147 of nine elements, there are eight strips, and $[56]$ is the only strip containing more than a single element. A strip has also been referred to as a* block *in [2], [1] and [6].*

Definition 11 (Strip move)**.** *A strip move involves changing the positions of a strip in the permutation π, so that it combines with other strips in π to form larger strips. For example, in the permutation 13524 moving the position of strips 3 is a strip move which results in permutation 15234. Here the strip 2 and 4 have combined with strip 3 to form a larger strip $[234]$.*

Definition 12 (Sorting by strip moves)**.** *The sorting by strip moves (aka block sorting) problem is to sort a given permutation by the minimum number of strip moves. Thus sorting by strip moves is essentially an optimization problem in which the number of strip move required to sort a permutation is minimized.*

Definition 13 (Strip move (block sorting) distance $sm(\pi)$)**.** *The strip move distance $sm(\pi)$ for any permutation π is the minimum number of strip moves required to sort π.*

A. Transpositions and Strip Moves

In [1] and [6] a process of *reducing* a permutation has been given. Any permutation can be reduced by replacing the strips in it by their *ranks* in the permutation. The rank of a strip is decided by its position in the identity permutation. The reduced permutation has been termed as a *kernel permutation* in [2]. As an example the permutation π = 8 [2 3] 9 1 [4 5 6 7] could be reduced to its equivalent kernel permutation $ker(\pi)$ = 4 2 5 1 3. The strip move primitive can be viewed as a nontrivial variantion of the primitive transpositions. We can always convert the given permutation to its equivalent kernel permutation. We observe that in a kernel permutation, all the strips have unit length. Since a strip is also a substring of the given permutation, all strip moves are also transposition moves. But the vice versa is not true, as in transposition, the positions of *any* substrings of a permutation could be changed. In case of a kernel permutation, strip moves reduces into transpositions where only substrings of length 1 are allowed to be moved.

B. Lower bounds for the problem

The sorted or identity permutation $123...n$ contains only 1 strip. Let the initial permutation π contain s strips. So the problem could be viewed as reducing the number of strips from s to 1. To sort the permutation, $s-1$ strips have to be reduced.
It is easy to observe that the number of strips can be reduced at most by 3 by performing a strip move. Since the identity permutation contains only one strip, a lower bound to this problem is $sm(\pi) = (s-1)/3$, where s is the number of strips in the original permutation.
A trivial algorithm A_{tri} for this problem would be to perform a strip move which reduces the number of strips by at least 1. This can always be done. For any element π_i, if $\pi_{i+1} \neq \pi_i + 1$, move $\pi_i + 1$ in the place of π_{i+1}. This algorithm will take atmost $(s-1)$ moves to sort the permutation. Thus the approximation ratio of A_{tri} can be found out as $\frac{A_{tri}}{A_{opt}} = \frac{(s-1)}{(s-1)/3} = 3$.
Hence A_{tri} is a 3-approximation algorithm for sorting by strip-exchanges.

The sorting by strip moves problem has been proved to be NP-Complete. The best known algorithms for the problem have an approximation ratio of 2. This primitive also finds extensive usage in optical character recognition.

V. Strip exchanges

In this section, we introduce the primitive strip exchanges which has been our contribution to genome rearragnement primitives and state certain lower bounds pertaining to it. Then we show its similarities between another primitive the block interchanges.

Definition 14 (Strip exchanging move)**.** *A strip exchanging move involves interchanging the positions of two strips in the permutation π, so that they combine with other strips in π to form larger strips.*
For example, in the permutation 13524 interchanging the position of strips 2 and 3 is a strip exchanging move which results in permutation 12534. Here the strip 2 has combined with strip 1 to form a larger strip $[12]$. Similarly strip 3 has combined with strip 4 to form a larger strip $[34]$.

Definition 15 (Sorting by strip-exchanges)**.** *The sorting by strip-exchanges problem is to sort a given permutation by the minimum number of strip exchanging moves. Thus sorting by strip exchanges is essentially an optimization problem in which the number of strip exchanging move required to sort a permutation is minimized.*

Definition 16 (Strip exchanging distance $se(\pi)$)**.** *The strip exchanging distance $se(\pi)$ for any permutation π is the minimum number of strip exchanging moves required to sort π.*

A. Block Interchanges and Strip Exchanges

The strip exchanges primitive can be viewed as a nontrivial invariant of the primitive block interchanges. We can always convert the given permutation to its equivalent kernel permutation. We observe that in a kernel permutation, all the strips have unit length. Since a strip is also a substring of the given permutation, all strip exchanges moves are also block interchanges move. But the vice versa is not true, as in block interchanges, the positions of *any* two substrings of a permutation could be interchanged. In case of a kernel permutation, strip exchanges reduces into block interchanges where positions of blocks of only length 1 are allowed to be interchanged. The *minimal block interchanges* algorithm to polynomially solve the sorting by block interchanges [7] problem does not work for sorting by strip exchanges. This algorithm interchanges any two substrings of the permutation.

The computational complexity of sorting by Strip Exchanges is still open. The best known algorithm for this problem has an approximation ratio of 2.

VI. A comparison of various sorting primitives

We present a comparison on the present status of the primitives strip moves, transpositions, reversals and strips exchange and block interchanges.

Primitive	Complexity	Approximation Ratio
Reversals	NP-Hard	1.375
Transpositions	Open	1.375
Strip Moves	NP-Hard	2
Block Interchanges	Polynomial	Optimal
Strip Exchanges	Open	2

It is very hard to predict which primitive would be more accurate in a particular case. Now a days people use the combinations of various such primitives for research purpose.

VII. Conclusions

Some genome rearrangement primitives have been discussed in this work. We have also shown some primitives to be the nontrivial variations of the others. There are still quite a lot of questions to be answered here. We list down a few of them:

1) Sorting by strip moves has been proved to be NP-Hard. Its a nontrivial variation of sorting by transpositions; but the complexity of transpositions has been a decade long open problem.
2) Sorting by block interchanges has been proved to be computationally polynomial. Strip exchanges is a nontrivial variation of block interchanges; but the complexity of strip exchanges is another open problem.
3) The hardness of approximation for sorting by strip moves is another interesting open question.

References

[1] M. Mahajan, R.Rama, V. Raman, and S. Vijaykumar. Approximate block sorting. International Journal of Foundation of Computer Science,17(2):337-355, 2006.
[2] M. Mahajan, R. Rama, V. Raman, and S. Vijayakumar. Merging and sorting by block moves. In Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2003, LNCS vol. 2914, pages 314 - 325. Springer-Verlag, Dec 2003.
[3] P. Pevzner. Computational Molecular Biology: An Algorithmic Approach. MIT Press, Cambridge, MA, USA, 2000.
[4] V.Bafna and P.Pevzner. Genome rearrangements and sorting by reversals. SIAM Journal on Computing, 25:272-289, 1996.
[5] V. Bafna and P. Pevzner. Sorting by transpositions. SIAM Journal on Discrete Mathematics 11(2):224-240, may 1998.
[6] W.W. Bein, L.L. Larmore, L. Morales, and I.H. Sudborough. A Polymomial Time 2-Approximation for Block Sorting. In Proc. of the 15th Annual Symposium on Fundamentals of Computing Theory, FCT 2005, August 2005, LNCS 3623, pages 115-124, Springer-Verlag, August 2005.
[7] David Alan Christie. Genome Rearrangement Problems. PhD thesis, The University of Glasgow, Glasgow,UK, August 1998.
[8] G.H Lin and G Xue. Signed genome arrangement by reversals and transpositions: Models and approximations. $Theoretical\ Computer\ Science$, 259:513-531, 2001.
[9] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM, 46:1-27, 1999.
[10] T. Hartman. A simpler 1.5-approximation algorithm for sorting by transpositions. In Combinatorial Pattern Matching (CPM 03), 2676:156169, 2003.
[11] Swapnoneel Roy, Ashok Kumar Thakur, Anupama Pande, Minhazur Rahman, Algorithms and Design for an Autonomous Biological System, icas, p. 35, Third International Conference on Autonomic and Autonomous Systems (ICAS'07), 2007
[12] A. Bergeron, J. Mixtacki, J. Stoye. A unifying view of genome rearrangements. PICB Spring School, Shanghai, March 12-16, 2007
[13] Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Bioinformatics.
[14] Swapnoneel Roy, Amitabh Bhattacharya. Algorithms for Sorting using Genomically Motivated Primitives. In Proc. of National Conference on Methods and Models in Computing (NCM2C-2007), JNU, New Delhi, India.

Swapnoneel Roy Swapnoneel Roy received his M.S in Computer Science and Engineering at the Indian Institute of Technology Madras. He is a Software Engineer at the Department of Information Management, IBM India Software Lab. His research interests include Computational Biology and Bioinformatics, Approximation Algorithms, Database Systems. He is author of a couple of research papers published at different conference proceedings.

Ashok K Thakur Ashok K Thakur received his B.Tech in Computer Science and Engineering at the Indian Institute of Technology Kanpur. He is a Software Engineer at the Department of Information Management, IBM India Software Lab. His research interests include Computational Biology and Bioinformatics, Approximation Algorithms, Database Systems. He is author of a couple of research papers published at different conference proceedings.

Minhazur Rahman Minhazur Rahman received his B.E in Computer Science and Engineering at Jadavpur University, Calcutta. He is a Software Engineer at the Department of Rational Software, IBM India Software Lab. His research interests include Computational Biology and Bioinformatics, Approximation Algorithms, Software Systems. He is author of a couple of research papers published at different conference proceedings.