Parallel Algorithm for Sorting a Signed Permutation by Reversals on MOT Interconnection Network

Amritanjali
Department of Computer Science and Engineering
Birla Institute of Technology
Mesra, Ranchi, India – 835215.

G. Sahoo
Department of Information Technology
Birla Institute of Technology
Mesra, Ranchi, India – 835215.

ABSTRACT
The problem of sorting a signed permutation by reversals is inspired and motivated by comparative genomics. Following the first polynomial time solution of this problem, several improvements have been published on the subject. The currently fastest algorithms is defined by the sequence augmentation sorting algorithm using balanced binary tree with running time $O(n^{3/2}\sqrt{\log n})$. We give a parallel implementation of the sequence augmentation sorting algorithm on the Mesh of trees architecture.

Categories and Subject Descriptors
C.1.2[**Processor Architectures**]: Multiple Data Stream Architectures – *Interconnection Architectures*

G.1.0 [**Numerical Analysis**]: General – *parallel algorithms*

General Terms
Algorithms

Keywords
Genome rearrangements, Sorting by reversals, Interconnection network, Time complexity.

1.INTRODUCTION
With new techniques for sequencing the entire genome of organisms, biology is rapidly moving towards a data-intensive computational science. The great challenge we face now is how to process this huge amount of data and extract from it relevant biological information. One way to structure this information is by comparative genomics, where we analyze data coming from distinct species and learn from the similarities and differences in related genomes. In this area, very large DNA molecules are investigated with respect to the relative order of genes in them. This is motivated from an observation that closely related species mostly have homologous genes but the genes occur in different orders. The order of genes in the genomes of species can change during evolution and can provide information about their phylogenetic relationship. An interesting method to infer the phylogenetic relationship from the gene orders is to use different types of rearrangement operations and to find possible rearrangement scenarios using these operations. One of the most common rearrangement operations is reversals, which reverse the order of a subset of neighbored genes. Reversals are by far the best studied gene order rearrangement operations. In comparative genomics, algorithms that sort permutations by reversals are often used to propose evolutionary scenarios of large-scale genomic mutations between species.

In this context, unichromosomal genomes are usually encoded as signed permutations, in which each element represents a gene and the sign determines the orientation of the gene. The problem of sorting signed permutations by reversals can be stated as: given two chromosomes, represented as sequences of genomic segments, find a parsimonious sequence of reversals that transforms a chromosome into the other one. The first polynomial algorithm was given by Hannenhalli and Pevzner [6], and took $O(n^4)$ time. After many subsequent improvements on the running time currently, the best known algorithm for this problem runs in $O(n^{3/2}\sqrt{\log n})$ time [10].

Sorting a signed permutation π by reversals involves two successive procedures on what is called the overlap graph of π. The first one consists in transforming the overlap graph such that all its connected components become oriented. This can be done in linear time [2]. The computational bottleneck of the sorting by reversals problem occurs in the second procedure: sorting the oriented components. The complexity of sorting the oriented components is determined by the time necessary to detect and choose an oriented arc, and apply the corresponding reversal to the permutation. With a classical data structure, using for instance a vector array to represent the current permutation, this is easily achievable in linear time. As a consequence, the algorithm, as well as any algorithm for sorting by reversals that has to apply a reversal at each step, will run in $O(n^2)$. In [7], Kaplan and Verbin described a clever data structure which allows to choose an oriented arc and apply the corresponding reversal in sublinear time. Eric Tannier, Anne Bergeron, Marie-France Sagot [10] used the same data structure, just adding some flags in order to be able to choose an oriented arc in a specific subset of the arcs, making it possible to apply a reversal and maintain the data structure in time $O(\sqrt{n} \log n)$ resulting in overall $O(n^{3/2}\sqrt{\log n})$ time complexity.

In this paper we propose O(n log n) parallel algorithm to sort a permutation of length n on the MOT (2^m) interconnection network [4] where $2^{m-1} < n \leq 2^m$. This paper is organized as follows: we have formalized the sorting by reversal problem in

section 2. Section 3 gives a brief introduction of the MOT interconnection network. We give details of parallel implementation of sorting by reversal algorithm on MOT interconnection network in section 4. We summarize our results in Section 5.

2.PROBLEM FORMALIZATION

For the purpose of sorting we identify the genes with the integers 1, . . . , n, with a plus or minus sign to indicate their direction. The order and direction of genomic markers will be represented by a signed permutation of {1, . . . , n}.

To simplify exposition, we adopt the usual extension which consists in adding $\pi_0 = 0$, and $\pi_{n+1} = n + 1$ to the permutation. We usually denote a signed permutation by simply writing ($0\ \pi_1 \ldots \pi_n\ n+1$). The identity permutation ($0\ 1 \ldots n+1$) is denoted by I. The inverse permutation π^{-1} of π is the (signed) permutation such that $\pi \cdot \pi^{-1} = I$.

The reversal of the interval [i, j] , $1 \le i, j \le n$, $i \ne j$ is the signed permutation $\rho_{i,j} = (0 \ldots i - 1 - j \ldots - i\ \ j + 1 \ldots n + 1)$. Note that $\pi \cdot \rho_{i,j}$ is the permutation obtained from π by reversing the order and flipping the signs of the elements in the interval [i, j]:

$$\pi.\rho_{i,j} = (\pi_0 \ldots \pi_{i-1} -\pi_j \ldots -\pi_i\ \pi_{j+1} \ldots \pi_{n+1}).$$

If $\rho_1,\ldots,\rho_k$ is a sequence of reversals, we say that it sorts a permutation $\pi.\rho_1 \ldots \rho_k = I$.

To each π_i , $0 \le i \le n+1$, are associated two points, named π_i^- and π_i^+ , except for 0 and n + 1, for which we define only the points 0^+ and $(n + 1)^-$ respectively.

An arc(π_i, π_j) is drawn between the points $\pi_i^+$and π_j^- where $0 \le i,j \le n$ and $|\pi_j| = |\pi_i| + 1$ (see Figure 1). Thus, there are n + 1 arcs and each point is the endpoint of a unique arc. Such arcs will be referred to as the arcs of π. Two arcs are said to overlap if the intervals they span (that is the set of points between the endpoints in the given order) intersect but none is contained in the other. If an arc is not overlapping with any other arc then it is said to be isolated. The arc(π_i, π_j) is said to be oriented if π_i and π_j have different signs, otherwise unoriented.

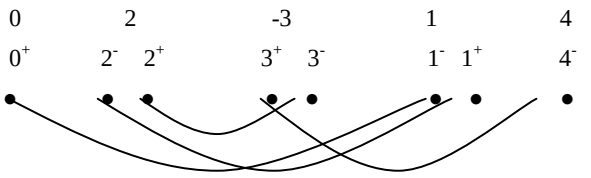

Figure 1. a signed permutation (0 2 -3 1 4) with associated points and arcs

3.MESH OF TREES ARCHITECTURE

The mesh of trees is a hybrid interconnection network based on arrays and trees. It owns two advantages of small diameter and large bisection width and is known as the fastest network when considered solely in terms of speed.

A two-dimensional $2^n \times 2^n$ mesh of trees consists of $2^n \times 2^n$ grid of nodes where there is a 2^n-leaf complete binary tree on each row and each column. For convenience MOT(2^n) to denote a $2^n \times 2^n$ mesh of trees. The trees built on rows and columns are called row trees and column trees, respectively. The leaf nodes are labeled by a pair of integers (i,j); i is the row index and j is the column index. For example MOT(4) is depicted in Figure 3, where each square represents a leaf node and each darkened circle represents a root node. There are $3\times2^{2n}-2^{n+1}$ nodes contained in MOT(2^n). The diameter and bisection width of MOT(2^n) are 4n and 2^n,respectively. T_d^r denotes the row tree with leaf nodes denoted as (d,0),(d,1), …,(d,(2^n-1)) and by T_d^c denotes the column tree with leaf nodes (0,d), (1,d), …, , ((2^n-1),d) where $0 \le d \le 2^n-1$. Each leaf-node is the intersection of a row tree and a column tree. So a leaf node (i,j) will be at the intersection of T_i^r and T_j^c, i.e. it is the j^{th} leaf node of tree T_i^r and i^{th} leaf node of tree T_j^c.

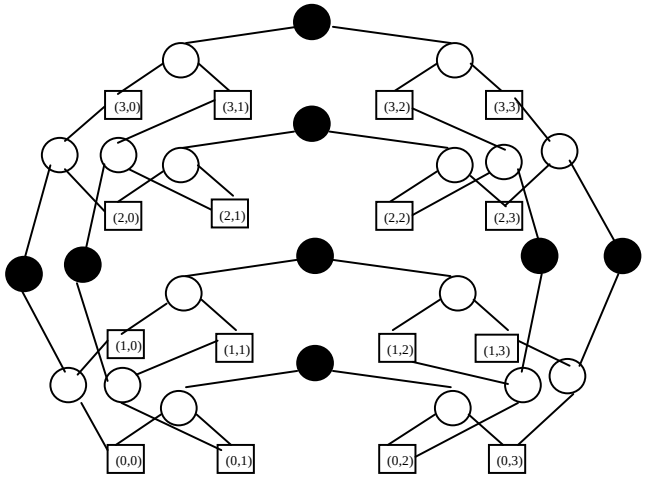

Figure 2: MOT (4)

4.PROPOSED PARALLEL ALGORITHM

Let we have to sort a signed permutation π of extended length n (n is two plus the actual number of elements in the original permutation).The only operation that we have used is the reversal operation. We present our parallel algorithm for sorting the permutation π on MOT(2^m) where $m \ge \log n$. If $m = \log n$ then we have MOT(n) with n row trees and n column trees where each tree is a complete binary tree having n leaf nodes. If $m > \log n$ (when n is not power of two) then only first n leaf nodes of n leftmost column trees and n bottommost row trees will be used.

4.1Initialization

The elements of the permutation π are distributed to the root nodes of all the trees such that root node of i^{th} row/column tree has $(i+1)^{th}$ element of the permutation π, where $0 \le i \le n-1$. Now the value stored in the root node of each tree is broadcasted to all its leaf nodes. Each leaf node (i,j) has two data, R_{ij} and C_{ij}. The value broadcasted by the root node of the i^{th} row tree is stored in R_{ij} and the value broadcasted by the root node of the j^{th} column tree is stored in C_{ij} where $0 \le i, j \le n-1$. Figure 3 shows initialization of the leaf nodes of MOT (4) for the permutation (0 2 -1 3).

4.2Parallel Sorting by Reversal Algorithm

In this we are giving the parallel algorithm that sorts a permutation of extended length n using MOT(n) interconnection network.

4.2.1Main Algorithm

The main sorting algorithm (Algorithm 1) is using two auxiliary algorithms, Algorithm 2 for finding neighbors of each element of the current permutation and Algorithm 3 for carrying out reversals of elements in a given interval.

Algorithm1

Input*:* the elements of the permutation are initially distributed as described above (see subsection 4.1). The lists S1 and S2 in the root node of the column tree T_0 are initialized as null.

Output*:* the sorted elements are present in R_{i0}, $0 \leq i \leq n-1$, in the leaf nodes of the column tree T_0. They are also present in, C_{0j}, $0 \leq j \leq n-1$, in the leaf nodes of the row tree T_0. Root node of the column tree T_0 contains the sequence of reversal intervals in the combined list S1S2.

The above output is obtained by applying the following steps-

Step 1. Find the set of arcs using the neighbor finding algorithm (Algorithm 2 described in subsubsection 4.2.2).

Step 2. For $0 \leq i \leq n-2$ and $0 \leq j \leq n-1$, the leaf node (i,0) of the column tree T_0 sends its arc along with the oriented and isolated flags to its parent node. The parent node sets it oriented flag as false if none of the arcs are oriented. If all the arcs are isolated then it sets isolated flag as true. Otherwise, if there is an oriented arc which is not isolated then it sets oriented flag as true and isolated as false and also saves that arc. However, if all the oriented arcs are isolated then it will set oriented as true and isolated as true (i.e. there is no oriented arc which is not isolated). Now the parent nodes send their flags to their parent nodes. The arc is also sent if oriented=true and isolated=false for that arc. The new parent node compute their flags using the same rule as described above and the process is repeated until the parent node is the root node.

Step 3. The root node of the column tree T_0 checks its oriented and isolated flags.

a. If (oriented = false & isolated = true) then go to step 5.

b. If (oriented = true & isolated = true) then remove the last reversal interval (low, high) from the list S1 and add it to S2.

c. Otherwise, the root node of the column tree T_0 computes the reversal interval for the oriented arc $((i, \pi_i), (j, \pi_j))$, received from one of its child nodes, as follows:

i. If $i<j$ then $l:=i$ and $h:=j$ else $l=j$ and $h:=i$.

ii. If $(\pi_l < 0 \ \& \ |\pi_l| < \pi_h)$ or $(\pi_l \geq 0 \ \& \ \pi_l > |\pi_h|)$ then low = l and high = h-1 else low = l+1 and high = h.

iii. It adds the reversal interval (low, high) to the list s1.

Step 4. Use the parallel reversal algorithm (Algorithm 3 described in subsubsection 4.2.3) to perform the reversal in the interval (low, high). Go to step 1.

Step 5. Root node of the column tree T_0 output the sequence S1S2.

4.2.2Neighbor Finding Algorithm

We say that π_j is neighbor of π_i iff $|\pi_i| +1 = |\pi_j|$, $0 \leq i \leq n-2$, $1 \leq j \leq n-1$. That is there is an arc between π_i & π_j, which we denote as arc $((i, \pi_i), (j, \pi_j))$.The arc $((i, \pi_i), (j, \pi_j))$ is said to be oriented if π_i & π_j have opposite signs otherwise it is unoriented. According to the definition of the isolated arc given in section 2, we can say that arc $((i, \pi_i), (j, \pi_j))$ is isolated if both π_i & π_j are negative and j-i =1or if both π_i & π_j are positive and i-j=1. Each row tree T_i except T_{n-1} uses the neighbor finding algorithm to find its neighbor tree T_j such that $|\pi_i| +1 = |\pi_j|$. We have dropped the superscript for the row as only row trees are participating in this algorithm.

Algorithm 2

Input: R_{ij} contains the element π_i and C_{ij} contains element π_j of the input permutation π in the leaf node (i,j), $0 \leq i, j \leq n-1$.

Output: The leftmost leaf node of each row except T_{n-1} tree has an arc $((i,\pi_i),(j,\pi_j))$ along with isolated and oriented flags to indicate if the arc is isolated or oriented respectively.

The corresponding algorithm is defined as-

For $0 \leq i \leq n-2$, $0 \leq j \leq n-1$ do in parallel

Step 1. Leaf nodes of each row tree T_i (except T_{n-1}) compare contents of R_{ij} and C_{ij}, where and $i \neq j$. If a leaf node finds $| \pi_i | + 1 = | \pi_j |$, then it sends arc $((i, \pi_i), (j, \pi_j))$ to the leftmost leaf node. Obviously, this will be true for only one leaf node of a tree.

Step 2. Each leaf node (i,0) of the column tree T_0 sets the oriented flag as true if π_i & π_j have opposite signs otherwise false. Also isolated flag is set as true if both π_i & π_j are negative and j-i =1or if both π_i & π_j are positive and i-j=1 otherwise false.

4.2.3Parallel Reversal Algorithm

Each row tree in the reversal interval (low, high) compute their partner row tree with which it has to exchange its element after reversing its sign. This change in the permutation is also reflected in the respective columns in the reversal interval.

Algorithm 3

Input: A reversal interval (low, high) in the root node of tree T_{n+2}.

The corresponding algorithm is defined as-

For low ≤ k ≤ high do in parallel

Step 1. Root node of each row tree T_k computes its reversal partner p as p = low + high – k, if low ≠ high. If low = high then k = p = low (or high).

Step 2. Root node of each row tree T_k broadcasts the value stored in C_{kp} to all its leaf nodes after reversing its sign.

Step 3. Root node of each column tree T_k simultaneously broadcasts the value of stored in R_{kk} to all its leaf nodes.

To sort an input permutation (0 2 -1 3) using parallel sorting by reversal algorithm we need MOT(4) interconnection network,as there are 4 elements in the permutation. First the MOT(4) is initialized with the elements of the permutation as shown in figure 3.

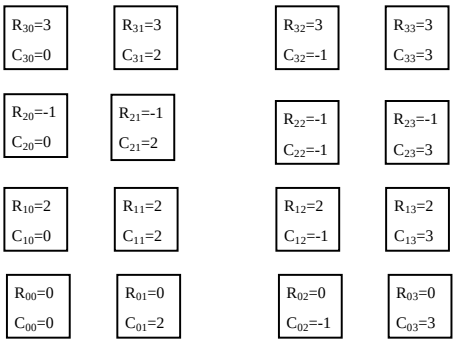

Figure 3. Initializing leaf nodes of MOT (4) for the input permutation (0 2 -1 3)

Then steps 1 - 4 of the main algorithm are repeated until the termination condition is satisfied. Execution of the neighbor finding algorithm (Algorithm 2) is shown in the Figure 4.

Next the root node of the column tree T_0 selects one of the oriented arcs received from its leaf nodes. In this case arc ((2,-1),(1,2)) and arc ((0,0),(2,-1)) are oriented. Let the root node selects arc ((0,0),(2,-1)) then it computes the reversal interval for the arc-

low = (i+1) =1 and

high = (j) = 2

The reversal interval (low,high) is added to list S1 and the parallel reversal algorithm (Algorithm 3) is used to reverse the elements in the reversal interval.

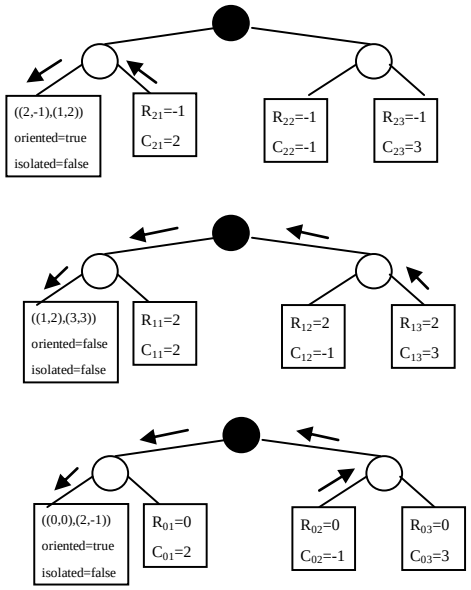

Figure 4. Step 1 of the main algorithm for the input permutation (0 2 -1 3)

Row trees T_1 and T_2 are in the reversal interval. Each of them compute their reversal partner (for k=1, p is 2 and for k=2, p is 1). Now the row tree T_1 broadcasts the value $–C_{12}$ and the row tree T_2 broadcasts the value $–C_{21}$ resulting in:

$R_{10} = R_{11} = R_{12} = R_{13} = -C_{12} = 1$ and $R_{20} = R_{21} = R_{22} = R_{23} = -C_{21} = -2$.

Next the column trees T_1 and T_2 broadcasts the value in R_{11} and in R_{22} respectively, resulting in: $C_{01} = C_{11} = C_{21} = C_{31} = R_{11} = 1$ and $C_{02} = C_{12} = C_{22} = C_{32} = R_{22} = -2$.

The new distribution of the elements is shown in Figure 5.

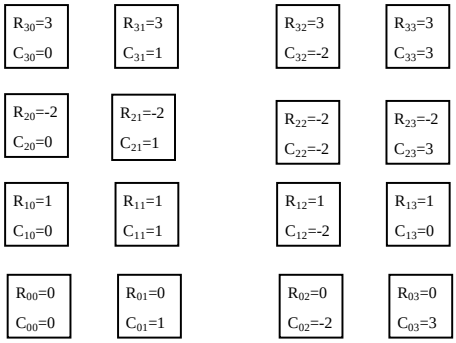

Figure 5. Contents of leaf nodes of MOT(4) after first iteration

In the next iteration the two oriented arcs are ((1,1),(2,-2)) and ((2,-2),(3,3)). For any of these arc reversal interval is low=2 and high=2. So, k=p=low or high=2.

Therefore, the row tree T_2 broadcast the value $–C_{22}$ resulting in: $R_{20} = R_{21} = R_{22} = R_{23} = -C_{22} = 2$. Then the column tree T2 broadcasts the value R_{22} to all its leaf nodes,

C_{02} = C_{12} = C_{22} = C_{32} = R_{22} = 2 (see Figure 6). In step 3 of third iteration it is found that none of the arcs are oriented and all the arcs are isolated. The main algorithm (Algorithm 1) terminates after executing step 5. Now the elements in the R_{i0}, $0 \leq i \leq n-1$, of the leaf nodes of the column tree T_0 (see Figure 6) are the elements of the input permutation sorted in ascending order. These sorted elements also present in the C_{i0}, $0 \leq i \leq n-1$, of the leaf nodes of the row tree T_0. The optimal sequence of reversals is present in the list S1S2.

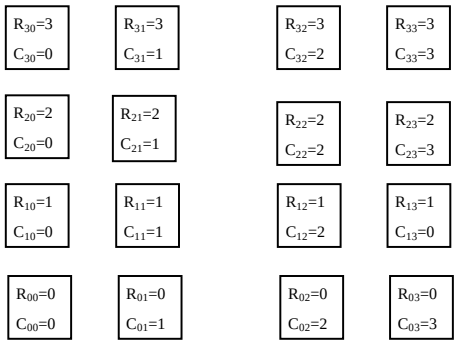

Figure 6. Contents of leaf nodes of MOT (4) after second iteration

5.DISCUSSION AND CONCLUSION

In the parallel sorting by reversal algorithm a reversal is applied only once unless it is an unsafe reversal [10]. If the reversal is unsafe it is reapplied. Hence, the complexity is determined by the time required for finding neighbors, detecting and selecting an oriented arc, and applying the corresponding reversal to the permutation. In the MOT(n) interconnection network data broadcast operation needs O(log n) time [4]. The input permutation π is can be sorted on MOT(2^m) ,where $2^{m-1} < n \leq 2^m$. If m > log n (when n is not power of two) then only first n leaf nodes of n bottommost row-trees and n leftmost column trees will be used. If m = log n then we have MOT(n) with n row trees and n column trees where each tree is a complete binary tree having n leaf nodes. Initialization step takes O(log n) time to distribute the elements of the input permutation to all the leaf nodes as decribed in section 4.1. In step 1 of the main algorithm (Algorithm 1) the neighbor finding algorithm (Algorithm 2) is used to find the set of arcs in the current permutation. It takes O(1) time to compare the two values (R_{ij} and C_{ij}) in the leaf nodes and O(log n) time for sending the arc to the leftmost leaf node. In constant time the leftmost leaf nodes can set their oriented and isolated flags. Hence, the neighbor finding algorithm takes O(log n) time. The second step of Algorithm 1 also takes O(log n) time. Now the root node of the column tree T_0 can check the flags, select an oriented arc and determine the reversal interval in constant time. In step 4 of Algorithm 1, the parallel reversal algorithm (Algorithm 3) is used to carry out the reversal in a given interval. The row trees in the reversal interval determine the index of their exchange partner in O(1) time. The reversal is done using row/column broadcast operation in O(log n) time. Thus, the parallel reversal algorithm takes O(log n) time. Steps 1-4 of Algorithm 1 are repeated until all the arcs are isolated and unoriented. In each iteration (except the last) one reversal operation is performed. After each reversal operation at least one (at most two) arc becomes isolated out of total (n-1) arcs. Therefore, the time complexity of the sorting algorithm to sort a signed permutation of length n by reversal operation is O(n log n) on MOT(n) interconnection network. In the case where $2^{m-1} < n < 2^m$, the time complexity of the proposed parallel algorithm will be O(n + n log n) which is also O(n log n).

6.REFERENCES

[1] Arslan, A. N. 2006. An algorithm for string edit distance allowing substring reversals. Sixth IEEE Symposium on BionInformatics and BioEngineering.

[2] Bader, D. A., Moret, B. M. E., Yan, M. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. In Proceedings of the Workshop on Algorithms and Data Structures, 2001, pp. 365–376.

[3] Bader, M., Abouelhoda, M., Ohlebusch, E. 2008. A fast algorithm for the multiple genome rearrangement problem with weighted reversals and transpositions. BioMed Central Bioinformatics .

[4] Chen, W., Chen, G., Hsu, D. F. 2000. Combinatorial properties of Mesh of Trees. In Proceedings of the International Symposium on Architectures, Algorithms and Networks, 134-139.

[5] Diekmann, Y., Sagot, M., and Tannier, E. 2007. Evolution under Reversals : Parsimony and Conservetion of Common Intervals. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 4, No. 2.

[6] Hannenhalli, S., Pevzner, P. 1999. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). J. Assoc. Comput. Mach. 46 (1999) 1–27.

[7] Kaplan, H., Verbin, E. 2003. Efficient data structures and a new randomized approach for sorting signed per mutations by reversals. In Proceedings of the CPM'03, Lecture Notes in Computer Science, vol. 2676, Springer, Berlin, 170–185.

[8] Li, Z., Wang, L. and Zhang, K. 2006. Algorithmic Approaches for Genome Rearrangement: A Review. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 36.

[9] Roy, S., Rahman, M., and Thakur, A. K. 2008. Sorting Primitives and Genome Rearrangement in Bioinformatics: A Unified Perspective. In Proceedings of World Academy of Science, Engineering and Technology, Vol. 28.

[10] Tannier, E., Bergeron, A., Sagot, M. F. 2007. Advances on sorting by reversals. Applied Discrete Mathematics Elsevier (2007), 881– 888.