Using Permutation Patterns for Content-Based Phylogeny

Md Enamul Karim[1], Laxmi Parida[2], and Arun Lakhotia[1]

[1] Center for Advanced Computer Studies
University of Louisiana at Lafayette, USA
{mek, arun}@cacs.louisiana.edu
[2] Computational Biology Center
IBM T J Watson Research Center
Yorktown Heights, USA
parida@us.ibm.com

Abstract. When the same set of genes appear in different orders on the chromosomes, they form a permutation pattern. Permutation patterns have been used to identify potential haplogroups in mammalian data [8]. They also have been successfully used to detect phylogenetic relationships between computer viruses [9]. In this paper we explore the use of these patterns as a content similarity measure and use this in inferring phylogenies from genome rearrangement data in polynomial time. The method uses a function of the cardinality of the set of common maximal permutation patterns as a proxy for evolutionary "proximity" between genomes. We introduce *Pi-logen*, a phylogeny tool based on this method. We summarize results of feasibility study for this scheme on synthetic data by (1) content verification and (2) ancestor prediction. We also successfully infer phylogenies on series of synthetic data and on chloroplast gene order of Campanulaceae data.

1 Introduction

Genome rearrangements may occur due to events such as inversions, transpositions, fusions, fissions, insertions or deletions. A major challenge in building a phylogeny from genome rearrangement data is estimating the common ancestor, either by reversing the effect of evolutionary events or by some other means.

Early approaches used breakpoint distance [2], [10],[11] to estimate the effect of evolution. Breakpoints are the adjacent genes present in one genome, but not in the other and breakpoint distance is the total number of such breakpoints. Consider two genomes each with five genes as shown below. The two breakpoints are shown by the arrows and the breakpoint distance between G_1 and G_2 is two.

$$\begin{aligned} G_1 &= g_1 \quad g_2\, g_3 \quad g_4\, g_5 \\ G_2 &= g_1 \updownarrow g_3\, g_2 \updownarrow g_4\, g_5 \end{aligned}$$

Thus breakpoints in the genome indicate the operations transposition and inversion. However, one or zero (absent) breakpoint may correspond to multiple such operations. Moreover, computing breakpoint phylogeny is an NP-hard problem [12]. Also, it is unclear how to suitably adapt it for multiple genomes.

Yet another scheme is the use of reversal distance between two genomes as an estimate of the evolutionary distance. This has been extensively studied in literature [1], [14], [15], [16] and the use of signed genes actually renders the problem polynomial time solvable for a pair of genomes [13].

A reversal in a signed permutation is an operation that takes an interval in a permutation, reverses the order of the numbers, and changes their signs. In the following example, G_3 is transformed into G_3' by one reversal of the boxed segment as shown.

$$\begin{array}{l} G_3 = g_5\ g_1 \boxed{\ g_3\ g_2\ -g_9\quad g_7} -g_4\ g_6\ g_8 \\ G_3' = g_5\ g_1 \boxed{-g_7\ g_9\ -g_2\ -g_3} -g_4\ g_6\ g_8 \end{array}$$

The reversal distance between two genomes is the minimum number of reversals required to get from one genome to the other. One reversal can eliminate maximum two breakpoints. Though reversal on signed permutations requires polynomial time for computation, its generalized version is still an NP hard problem [4]. Reversal distance, like the breakpoint, can underestimate the actual number of steps that occurred biologically and prefers all the genomes under study to have same set of genes.

Various other hybrid and heuristic based approaches also have been studied in literature and the reader is directed to [5] for an excellent summary.

In this paper, we propose a content-similarity based measure to handle gene order data based on permutation patterns [6]. The content similarity is based on the nature and location of the permutation patterns that co-occur in the genomes. Through simulations, we observe that ancestral information is substantially preserved through the common permutations of various lengths. Based on this observation we develop a similarity matrix and use this matrix to estimate ancestor information and build the phylogeny. *Pi-logen* is an implementation of this scheme: it can be used for multi-chromosomal genomes and has a polynomial run time. The tool is available at www.cacs.louisiana.edu/∼mek8058/Pi-logen.

2 Permutation Patterns

Consider genomes G_1 and G_2 defined as gene orders:

$G_1{=}g_{11}g_1g_2g_3g_4g_5g_6g_7g_8g_{12}$, $G_2{=}g_2g_4g_1g_3g_5g_9g_7g_6g_{10}$.

Is there anything common between the two genomes? The following clusters or groups of genes appear in the two genomes: $p_1 = \{g_1, g_2, g_3, g_4\}$, $p_2 = \{g_6, g_7\}$. p_1 and p_2 are called permutation patterns.

Let G_1 and G_2, defined on Σ, be of length n each. The number of common permutation patterns in G_1 and G_2 can be no more than $\mathcal{O}(n^2)$, since each pattern can start at location i and end at location $(j > i)$, $1 \le i < j \le n$. In the following example, we show that $\mathcal{O}(n^2)$ can actually be obtained. Consider G_1 and G_2 of size $4n$ each as shown.

$$\begin{array}{l} G_1 = g_{1_a}g_{1_b}g_{1_c}g_{1_d}g_{2_a}g_{2_b}g_{2_c}g_{2_d} \cdots g_{n_a}g_{n_b}g_{n_c}g_{n_d} \\ G_2 = g_{1_c}g_{1_a}g_{1_d}g_{1_b}g_{2_c}g_{2_a}g_{2_d}g_{2_b} \cdots g_{n_c}g_{n_a}g_{n_d}g_{n_b} \end{array}$$

These two sequences have $\mathcal{O}(n^2)$ common permutation patterns given by $p_{ij} = \cup_{k=i}^{j}\{g_{k_a}, g_{k_b}, g_{k_c}, g_{k_d}\}$, $1 \le i < j \le n$. The following lemma is easy to see.

Lemma 1. *Given m sequences of length n each, the number of permutation patterns that appear in at least $K(\geq 2)$ sequences is $\leq mn^2$.*

Given K, $2 \leq K \leq m$ and a collection of m sequences (genomes) G_i, $1 \leq i \leq m$, let $\mathcal{P}$ be the collection of all permutation patterns that appear at least K times. An m-dimensional array F_p corresponding to a permutation pattern $p \in \mathcal{P}$ as follows, can be viewed as G_i's feature vector, where $f(p)$ is some appropriate function of p:

$$F_p[i] = \begin{cases} f(p) & \text{if } p \text{ occurs in } G_i \\ 0 & \text{otherwise} \end{cases}$$

2.1 Dimension Reduction Via Maximality

Choosing the right length of a permutation pattern to extract content information is tricky. One option is to use all possible lengths, however, this gives a $\mathcal{O}(n^2)$-dimension feature space (n is the length of the genome).

We tackle this problem using maximal permutation patterns which reduces the dimension to $\mathcal{O}(n)$. Maximal permutation patterns cover all the permutation patterns of different granularity. We recall the definition of a maximal permutation pattern [6].

Definition 1. *(maximal) Let $\mathcal{P}$ be the collection of all permutation patterns on a given data. Let $p_1, p_2 \in \mathcal{P}$ be such that each occurrence of p_2 in the data is covered by an occurrence of p_1 and each occurrence of p_1 covers an occurrence of p_2, then p_1 is not maximal with respect to p_2. A pattern $p \in \mathcal{P}$ is maximal if there exists no $q \in \mathcal{P}$ such that p is not maximal with respect to q.*

Consider the following example. $G_1 = g_1 g_2 g_3 g_9 g_0 g_4 g_5 g_6$, $G_2 = g_3 g_0 g_2 g_9 g_4 g_5 g_7 g_8$ and $G_3 = g_4 g_9 g_2 g_0 g_3 g_8 g_7 g_5$. Let $\mathcal{P}'$ be the collection of all maximal permutation patterns, then $\mathcal{P}' = \{p_1, p_2, p_3\}$, where

$p_1 = \{g_0, g_2, g_3, g_4, g_5, g_7, g_8, g_9\}$ that occurs in G_2 and G_3;
$p_2 = \{g_0, g_2, g_3, g_4, g_5, g_9\}$ that occurs in G_1 and G_2;
$p_3 = \{g_0, g_2, g_3, g_4, g_9\}$ that occurs in G_1, G_2 and G_3.

Other permutation patterns are not in $\mathcal{P}'$ because they are not maximal w.r.t either of p_1, p_2, p_3. For example, $\{g_4, g_5\}$ is a permutation pattern appearing in G_1 and G_2, however, both of its occurrences are covered by two occurrences of p_2 making is non maximal.

For the above example, by the definition, $F_{p_1} = [0, 1, 1]$, $F_{p_2} = [1, 1, 0]$, $F_{p_3} = [1, 1, 1]$. Now, the following lemma is straightforward to verify.

Lemma 2. *(p is not maximal w.r.t. q) $\Rightarrow$ ($F_p = F_q$).*

The converse of the lemma is not true, since there may be distinct permutation patterns that occur in the same input sequences.

Each maximal permutation pattern may have two kinds of components: (1) sequence-preserving and (2) true permutations. For example, in p_2, $\{g_0, g_2, g_3, g_9\}$ is a true permutation and $\{g_4, g_5\}$ is a sequence preserving component. In fact a maximal permutation pattern has a clean hierarchical structure that is explored in [8]. Lets define $cnt_1(p)$ to be the ratio of sequence-preserving components and $cnt_2(p)$ to be the ratio of true permutation components in p.

2.2 Similarity Measure

We use the common permutations as an estimate of the similarity between genomes. Let P' be the set of maximal permutation patterns and let $P'_i \subseteq P'$ be the collection that occurs in genome i. We define similarity between two genomes i and j, $S(i,j)$ as,

$$S(i,j) = \sum_{\substack{(p \in P'_i \cap P'_j \\ \text{OR} \\ p \notin P'_i \cup P'_j)}} (1-\alpha)\, cnt_1(p) + \alpha\, cnt_2(p) \tag{1}$$

where $0 \le \alpha \le 1$ is a fixed constant (weighting factor) to take care of the effect of the internal structure of p, if there exists any.

This similarity measure rewards for the presence in both genomes i and j or absence in both i and j, and penalizes for being present in one and absent in the other.

3 Method

In this section we describe the method implemented in *Pi-logen*: it uses an *agglomerative hierarchical clustering* method [7]. Given the m genomes, in this scheme every genome is initially considered a cluster. Then the two genomes with the highest similarity are combined into a cluster. This iterative procedure continues until a stopping criterion is fulfilled (a single cluster, say).

Computing pairwise similarity. The pairwise similarity measure for genomes G_i, $1 \le i \le m'$, for a given quorum K ($\le m'$) and a weighting factor $0 \le \alpha \le 1$ is computed in four steps as follows:

(Step 1) Compute $\mathcal{P}'$, the collection of all the maximal permutation patterns that occur in at least K genomes. Let $|\mathcal{P}'|$ be denoted by n. For each $p_i \in \mathcal{P}'$, also compute $cnt_1(p_i)$ and $cnt_2(p_i)$ (see Section 2.1).
(Step 2) Create m feature vectors of n dimension each as the ($m \times n$) feature matrix F:

$$F[i,j] = \begin{cases} 1 & p_j \text{ occurs in } G_i \\ 0 & \text{otherwise} \end{cases}$$

This matrix is required for easy updates during the clustering.
(Step 3) Build a temporary $(m' \times m') \times n$ matrix T:

$$T[i,j,k] = \begin{cases} 1 & \text{if } F[i,k] = F[j,k] \\ 0 & \text{otherwise} \end{cases}$$

This matrix is not explicitly built but is given here for ease of exposition. Note that in the next step, this matrix can be temporarily built as and when required.

(Step 4) Build an $m' \times m'$ similarity matrix S as follows:

$$S[i,j] = \sum_{k=1}^{n} T[i,j,k]\,((1-\alpha)\,cnt_1(p_k) + \alpha\,cnt_2(p_k))$$

This completes the computation of the similarity matrix S.

Hierarchical clustering. The iterative process is applied as follows:

```
$m' \leftarrow m$
Compute $(m' \times m')$ similarity matrix $S$
**Repeat**
  (a) Let $S[p,q]$ have the largest value in $S$, link $p,q$
  (b) Replace row $q$ by $(q \vee p)$ in $F$ and recompute row and column $q$ in $S$ accordingly
  (c) Remove row and column $p$ from $F$ and $S$
  (d) $m' \leftarrow (m' - 1)$
**Until** $(m' = 1)$
```

The hierarchy that is constructed by this process corresponds to the inferred phylogeny tree. In actual implementation we use either the upper or lower triangular of S because S is symmetric and just ignore row and column p instead of actually removing them.

Time complexity. Assume that all m genomes are of length N each. Step 1 takes $\mathcal{O}(N^2 m \log G \log n)$ where G is the number of distinct genes in the data [6], [8]. Step (2) takes $\mathcal{O}(mn)$ time and Step (4) takes $\mathcal{O}(m^2 n)$ time. The algorithm is iterated $\mathcal{O}(m)$ times, each iteration taking $\mathcal{O}(mn)$ time. Thus the algorithm takes $\mathcal{O}(m^2 n + N^2 m \log G \log n))$ time.

4 Feasibility Experiments

The key contributions of our approach is the ancestor content prediction (step (b) in *repeat* loop) based on maximal permutation patterns and the similarity measure (step 3 and 4). The effectiveness of this approach to discover a good phylogeny depends on the feasibility of these two methods. Hence, we setup two experiments using synthetic data to verify their feasibility empirically. Synthetic data, in all the experiments, is produced through simulation of evolution.

For the first experiment we carry out a content verification test. For the second one we check how well the similar species are grouped together under each internal nodes (ancestors). We call this measure "ancestor prediction". The second experiment also involves estimating a good weighting factor, α.

(1) Content verification: The experiment involves taking a genome of length n and applying d "evolution" edit operations on it to obtain a set of m evolved

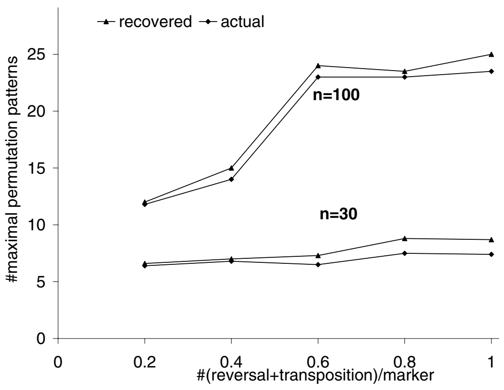

Fig. 1. Content verification: Number of maximal permutation patterns recovered is plotted against the actual number of permutations in the data. See text for details.

genomes. These operations are reversals and transpositions. The nature and location of the operation is picked at random. Then we look for the presence of P', the union of permutation patterns in the m evolved genomes, in the ancestor genome.

The ancestral genome is the identity permutation 0 1 2 3 ... (n-2) (n-1). We describe the results for $n = 30, 100$ and $m = 3$. Three genomes G_1, G_2, G_3 are obtained by d edit operations (i.e., reversals and transpositions) each on the ancestral genome.

Let change ratio r_d be defined as $r_d = md/n$. The simulations are repeated a number of times to obtain an average trend and the result is shown in Figure 1. It plots the number of recovered maximal permutation patterns and the actual number of them in the ancestor sequence against r_d.

Because we use UNION operation to compute ancestral content from descendants, it is likely that we may overestimate ancestral content. We use an evaluation function ω_n that calculates the amount of overestimation in the ancestral content prediction where the ancestral genome is of length n. The experiment is performed for l distinct change ratios. If for a specific change ratio, the predicted number of permutations is o_n and a_n of them are actually present, then

$$\omega_n = \frac{-1}{l}\sum_{i=1}^{l} \frac{a_n - o_n}{a_n}$$

In our experiments, we obtained $\omega_{30} = 0.083$ and $\omega_{100} = 0.052$. In other words, on an average, there is an overestimation of only 8% on genomes of length 30 and an overestimation of only 5% on genomes of length 100.

(2) Ancestor prediction: We created a set of m genomes with n genes each. These m genomes correspond to the m leaves of a reference phylogeny tree T_r. In this tree, each descendant is obtained by at most d edit operations (inversion or transposition). The length of a segment affected by these edit operations is randomly chosen between 1 and 10.

Our suite of experiments uses $m = 16$ and $n = 120$. For each d we produce five such data sets and run the experiments over a series of α. For comparison purposes, every time we generated 16 species, we maintained a fixed reference tree, $T_r = (T_r^1, T_r^2)$ where

$$T_r^1 = (((0,1)(2,3)),((4,5),(6,7)))$$

$$T_r^2 = (((8,9),(10,11)),((12,13),(14,15)))$$

Note that this reference tree is a complete binary tree that has fifteen internal nodes, including the root node. In each experiment, the topology of the reference tree is the same, but the edit (or evolution) operations on the branches of the tree as well as the genomes corresponding to the leaves are different.

Measuring matches of trees. We used the following measure to match different trees that have the same set of leaf node labels. Let an internal node(ancestor) numbered i of tree T_X be denoted as T_X^i and let $D(T_X^i)$ denote the set of leaves reachable from this node. For example, in the reference tree, $D(T_r^1) = \{0,1,2,3,4,5,6,7\}$. If two sets D_1 and D_2 are equal then we have an ancestor match, formally

$$\delta(D_1, D_2) = \begin{cases} 1 \text{ if } (D_1 = D_2) \\ 0 \text{ otherwise} \end{cases}$$

We use a simple measure $Match(T_I, T_r)$ to compare the inferred tree T_I with the reference tree T_r whose values range from values 0 to $(m-1)$ with $(m-1)$ denoting a perfect match and 0 denoting a complete mismatch. Formally,

$$Match(T_I, T_r) = \sum_{i=1}^{(m-1)} \sum_{j=1}^{(m-1)} \delta(D(T_I^j), D(T_r^i))$$

Estimating α, the weighting factor: A control parameter in this scheme is the weighting factor α, in the similarity measure, $S(i,j)$, of genomes i and j as shown in Equation (1). We carry out a series of experiments and the results are summarized in Table (1). We obtain the best values of match for $(1-\alpha) = 0.5$ and 0.6. We

Table 1. $Match(T_I, T_r)$, for d edit operations and weighting factor α

	$(1-\alpha)$											
d	0 .0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Average
1	7.3	8.9	9.1	10.8	12	12.7	12.1	10.9	10.8	10.9	10.8	10.57
2	9.2	9.3	9.7	10.7	11.7	12.2	12.4	11.5	11.3	10.9	11.1	10.91
3	8.4	8.5	9.5	10.3	11.5	11.2	11.5	10.2	9.7	9.4	9.4	9.96
4	10.1	12.2	12.9	13.5	13.8	14.2	14.2	14.1	13.7	13.4	12.7	13.16
5	9.7	10.6	11.5	12.9	13	13.7	13.5	13.4	12.9	12.2	11.3	12.25
6	12.6	12.8	12.8	13.3	13.8	13.9	14.2	13.7	13.7	13.1	13	13.35
7	13.2	12.9	12.9	13.7	13.8	14.3	14.1	13	13.1	12.9	12.7	13.33
8	13.1	13.3	13.4	14.2	14.2	14.3	14.3	13.4	13.1	14.1	14	13.76
Average	10.45	11.06	11.48	12.43	12.98	**13.31**	**13.29**	12.53	12.19	12.11	11.88	12.16

Table 2. $Match(T_I, T_r)$ for 10 trees with $d = 4$, $(1 - \alpha) = 0.5$

14	13	15	14	14	13	12	13	15	13	Average 13.6

verify this with further simulation experiments, whose results are summarized in Table (2).

Effect of d on reconstruction: Further, we observed that the accuracy of the tree reconstruction using the measure $Match(T_r, T_I)$, usually improves with increase in the number of edit operations d during each "evolution" process. The results are shown in Table (1), which is not a surprising observation and is in fact reassuring about our proposed scheme.

The conclusion of the exercise performed in this section is that it is worthwhile to explore the reconstruction of an underlying phylogeny tree using the set of maximal permutation patterns.

5 Experimental Results

Here we discuss our results of using *Pi-logen* on synthetic data and then on chloroplast DNA (cpDNA) of the Campanulaceae family.

5.1 Synthetic Data

We now describe our simulation experiments for inferring phylogeny trees. We fixed the topology of the reference tree to T_r of the last section. We generated 100 cases: in each we generated 16 genomes (corresponding to the leaves of T_r) by using randomly chosen values of the number of edit operations $d = 1, 2, 3, \ldots, 10$, for each evolution step. Given this set of 16 genomes, we then inferred the underlying phylogeny tree with *Pi-logen* using the estimated value of $\alpha = 0.4$ and 0.5 from the previous section and $K = 2$. Figure 2 shows three of the trees inferred by the algorithm for $d = 3$. In one of them (leftmost) the reference tree is predicted exactly.

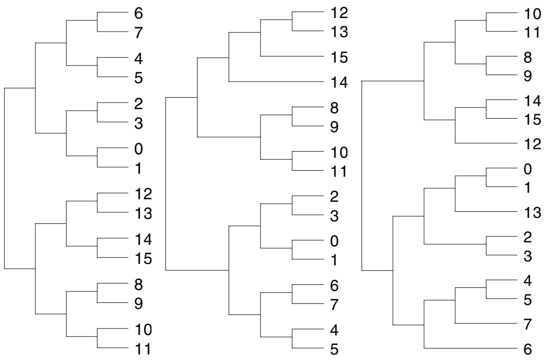

Fig. 2. Three trees inferred by *Pi-logen* for $d = 3$

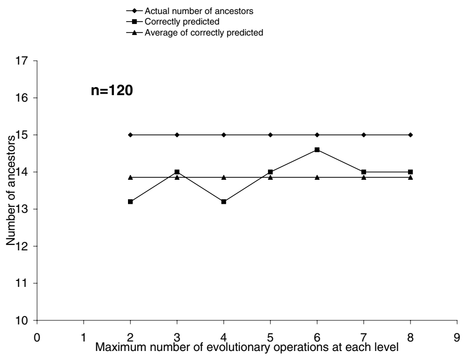

Fig. 3. Number of ancestors correctly predicted plotted against the number of edit operations

Figure 3 shows $Match(T_I, T_r)$ for the inferred trees for different amount of evolutionary changes d. The average $Match(T_I, T_r)$ value for this setup was found to be 13.85. Recall that for this setup the best value of $Match(.,.)$ is 15 (and the worst is 0). The average number of maximal permutation patterns for this setup was 471.4 and the average tree computation required 16.13 seconds on a 2.3 GHz pentium 4 processor.

5.2 Campanulaceae Data

We next use our algorithm on the cpDNA for Campanulaceae data set that has been also used by [3], [5]. This data set has about 105 genes in 13 extant species. We found 167 maximal permutation patterns and it took approximate 11 seconds to generate the phylogeny tree.

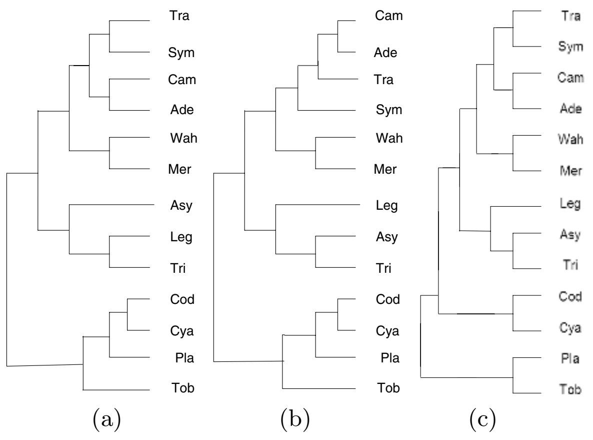

Fig. 4. The phylogeny tree inferred using (a) maximal permutation pattern (b) reversal and (c) breakpoint based methods on the cpDNA of Campanulaceae data set

Figure 4 shows three inferred trees: (a) using maximal permutation patterns (using *Pi-logen*), (b) using reversal based algorithm [3] and (c) using breakpoint based algorithm [5] .

The sub-tree $(((Tra, Sym), (Cam, Ade)), (Wah, Mar))$ in (a) is identical to the one in (c). The sub-tree $(((Cod, Cya), Pla), Tob)$ in (a) is identical to the one in (b). The sub-tree $((Leg, Tri), Asy)$ in (a) is different from the one in (b) and (c). The aligned genomic sequences are shown below:

Leg : 76-56 s_1 **90**-84 s_2 **91**-96 5-8 55-53
Tri : 76-56 s_1 89-84 s_2 90-96 **X** 55-53
Asy : 76-**57** s_1 89-84 s_2 90-96 **X** **X**

The numbers refer to the gene encodings and s_1 and s_2 correspond to common segments (of genes) in the three. One can see that from this alignment, the *correct* choice of a subtree on these three genomes is not apparent.

6 Conclusion

We present *Pi-logen* a content similarity based method for inferring phylogeny in genome arrangement data. This similarity is based on a well studied regularity measure, a permutation pattern, that co-occurs in multiple genomes. We summarize our results of an extensive feasibility study of using this scheme by content verification and ancestor prediction. We also successfully test the scheme on synthetic and cpDNA of Campanulaceae data.

References

1. Kececioglu, J. and Sankoff, D.(1994) Efficient bounds for oriented chromosome inversion distance. *5th Annual Symposium on Combinatorial Pattern Matching* CPM, **807** 307-325
2. Blanchette, M., Bourque, G. and Sankoff, D. (1997) Breakpoint phylogenies. *In Genome Informatics Workshop (GIW 1997)*, (eds. S. Miyano and T. Takagi), pp. 25-34. University Academy Press, Tokyo.
3. Bourque, G. and Pevzner, P. A.(2002) Genome-Scale Evolution: Reconstructing Gene Orders in the Ancestral Species. *Genome Research*, **12**(1): 26-36, Cold Spring Harbor Laboratory Press
4. Caprara, A. (1999) Formulations and complexity of multiple sorting by reversals. *Proceedings of the Third Annual International Conference on Computational Molecular Biology RECOMB*, (eds. S. Istrail et al.), pp. 8493. ACM Press, Lyon, France.
5. Cosner, M. E. et. al. (2000) An Empirical Comparison of Phylogenetic Methods on Chloroplast Gene Order Data in Campanulaceae. *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, D. Sankoff and J. Nadeau, eds., 99-121, Kluwer Academic Publishers
6. Eres R., Landau, G. M. and Parida, L. (2003) Combinatorial approach to automatic discovery of cluster patterns. *Algorithms in Bioinformatics: Third International Workshop, WABI, Budapest, Hungary*, WABI, pp. 139 - 150, Springer-Verlag
7. Kauffman, L. and Rousseeuw, P. (1990) Finding Groups in Data: An Introduction to Cluster Analysis

8. Landau, G. M., Parida, L. and Weimann, O. (2005) Using PQ Trees for Comparative Genomics. *Combinatorial Pattern Matching, CPM, Jeju Island, South Korea*, CPM, pp. 128 - 143, Springer-Verlag
9. Karim, M. E., Walenstein, A., Lakhotia, A. and Parida, L. (2005) Malware phylogeny generation using permutations of code. *European Journal of Computer Virology* **1** 1-11
10. Nadeau, J. and Taylor, B. (1984) Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl.Acad. Sci.* PNAS **81** 814-818
11. Watterson, G., Ewens, W., Hall, T. and Morgan, A. (1982) The chromosome inversion problem. *J. Theor. Biol.* **99** 1-7
12. Peer, I. and Shamir, R. (1998) The median problems for breakpoints are NP-complete, *Electronic Colloquium on Computational Complexity* Technical Report 98-071, http://www.eccc.uni-trier.de/eccc.
13. Hannenhalli, S. and Pevzner, P. (1995) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proc. of the 27th Annual Symposium on Theory of Computing STOC*, 178189
14. Sankoff, D. (1992) Edit distance for genome comparison based on non-local operations. *Proc of the 3rd Annual Symposium on Combinatorial Pattern Matching* CPM, 121-135
15. Berman, P. and Hannenhalli, S.(1996) Fast sorting by reversal. *7th Annual Symposium on Combinatorial Pattern Matching* CPM, **1075** 168-185
16. Kaplan, H., Shamir, R. and Tarjan,R. (1997) Faster and simpler algorithm for sorting signed per- mutations by reversals. *Proc of the 8th Annual ACM-SIAM Symposium onDiscrete Algorithms* SODA, 344-351