Estimating Genome Reversal Distance by Genetic Algorithm

Andy Auyeung
Oklahoma State University
Math Science 219
Stillwater, OK 74078
1 (405) 744–5668
wingha@cs.okstate.edu

Ajith Abraham
Oklahoma State University
North Hall 328
Tulsa, OK 74106
1 (918) 594–8188
ajith.abraham@ieee.org

Abstract- Sorting by reversals is an important problem in inferring the evolutionary relationship between two genomes. The problem of sorting unsigned permutation has been proven to be NP-hard. The best guaranteed error bounded is the 3/2-approximation algorithm. However, the problem of sorting signed permutation can be solved easily. Fast algorithms have been developed both for finding the sorting sequence and finding the reversal distance of signed permutation. In this paper, we present a way to view the problem of sorting unsigned permutation as signed permutation. And the problem can then be seen as searching an optimal signed permutation in all 2^n corresponding signed permutations. We use genetic algorithm to conduct the search. Our experimental result shows that the proposed method outperforms the 3/2-approximation algorithm.

1 Introduction

Genome Rearrangement is a mechanism that happens in mitochondrial genomes (Russell 2002). The genes order in mitochondrial genome is constantly under rearrangement. Therefore, by estimating the rearrangement distance between two genomes, the relationship between them can also be estimated (Pevzner 2001). Reversal is the most commonly seen mechanism that genomes are rearranged. Figure 1 shows the estimated transformation from *Tobacco* to *Lobelia fervens* by reversals (Bafna and Pevzner; 1996). There are two variations of this problem, signed permutation and unsigned permutation. For unsigned permutation, a genome is modeled as a permutation π with order n (i.e. a permutation of $\{1, 2, \ldots, n\}$), where n is the number of gene blocks in the genome. Let the permutation $\pi = \pi[1]\ \pi[2]\ \ldots\ \pi[n]$, the reversal operation $\rho(i,j)$ rearrange π into $\pi[1]\ \ldots\ \pi[i\text{-}1]\ \pi[j\text{-}1]\ \ldots\ \pi[i]\ \pi[j]\ \ldots\ \pi[n]$. For signed permutation π', each $\pi'[k]$ has either a positive or a negative sign. Each reversal operation $\rho(i,j)$ not only rearrange π' but also negate the sign of $\pi'[k]$ for $i \le k < j$. The problem of estimating reversal distance between two genomes is formulated as sorting permutation by reversal operation. That is, given π *(or π')*, we want to find a sorting sequence that uses minimum number of reversal to sort π (or π') into identity permutation (i.e. the permutation, 1 2 … n for unsigned permutation, and +1 +2 … +n for signed permutation). We called the minimum number of reversal the reversal distance.

The problem of sorting signed permutation can be solved in $O(n^2)$ time (Kaplan et al. 1997). The problem of finding the reversal distance of signed permutation can be solved in $O(n)$ time (Bader et al. 2001). However, both sorting and finding the reversal distance of unsigned permutation has been proven to be NP-hard (Caprara 1997). So, error bounded heuristic solutions have been proposed (Bafna and Pevzner 1996; Kececioglu and Sankoff 1995). The lowest guaranteed error bound thus far is the 3/2-approximation algorithm (Christie 1998). The 3/2-approximation algorithm uses the fact that any cycle decomposition of the breakpoint graph that maximize the number of 2-cycles exists a sorting sequence with length at most 3/2 of the optimal sorting sequence.

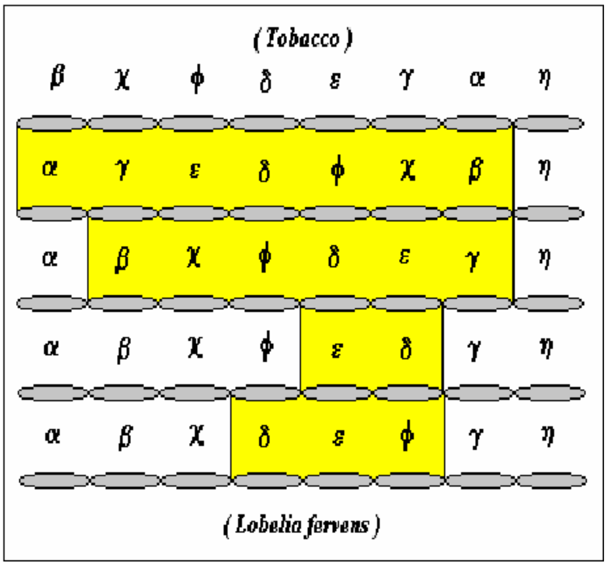

Figure 1. Transformation from *Tabacco* to *Lobelia fervens* by reversals (Bafna and Pevzner; 1996)

In this paper, we propose a genetic algorithm for sorting unsigned permutation by reversal. Our method does not provide guaranteed error bound. However, our experiment shows that it finds better solution than the 3/2-approximation algorithm. Also, so far, all heuristic algorithms for this problem use a constructive manner to find the solution. We would like to show an alternative approach how this problem can be solved in inductive manner.

The rest of the paper is organized as the following. In Section 2, some background materials on sorting permutation by reversal and the concept of genetic algorithm are presented. In Section 3, the proposed method is explained. In Section 4, the experimental setup and results are shown. In Section 5, observations from the experiment are discussed. Finally in Section 6, some concluding remarks are made.

2 Reviews

2.1 Breakpoint Graph

An unsigned permutation can be modeled by a breakpoint graph. For each gene (a number in the permutation), we will create a node for it. The idea of breakpoint graph is to mark the desired and realistic relationship between these nodes in the permutation. For each pair of the nodes we draw a black edge between them if they are adjacent in the permutation, and we draw a red edge between them if they are adjacent in the identity permutation. In order to model the orientation, we expand the unsigned permutation to have a zero at the front and a n+1 at the end. An example is shown in Figure 2. It has been shown that given a cycle decomposition of the breakpoint graph, any reversal can at most change the number of cycles by one. Besides, it has also been shown that given a cycle decomposition of the breakpoint graph, the corresponding shortest sorting sequence can then be found. Thus, the key problem is to find a cycle decomposition that provides the shortest sorting sequence for the unsigned permutation. However, the problem of finding an optimal cycle decomposition is NP-hard.

On the other hand, sorting signed permutation can be solved easily. We first create the breakpoint graph as above. Then for each gene node i, we split it into two nodes 2i-1 and 2i with ascending order if i is positive, descending order otherwise. Also note that node 0 and n+1 are replaced by node 0 and 2n+1 respectively. An example is shown in Figure 3. The advantage is that now each node has exactly one red edge and one black edge associated with it. Thus, there is only one cycle decomposition of this breakpoint graph. Many algorithms have been proposed to find the sorting sequence. The best known time bound is $O(n^2)$ time. And the best known time bound time for finding the reversal distance is $O(n)$.

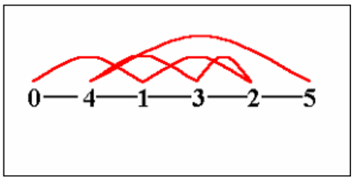

Figure 2. Breakpoint graph for unsigned permutation: 4 1, 3, and 2

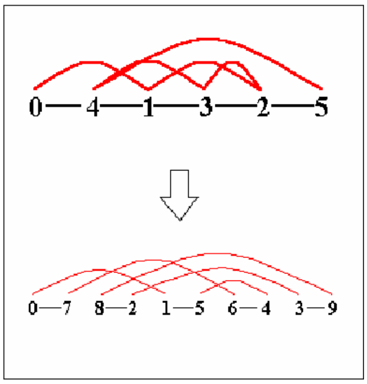

Figure 3. Breakpoint graph for signed permutation: +4, -1, +3 and -2

2.2 Genetic Algorithm

We used the standard genetic algorithm and the management of population/ selection of individuals and reproduction and can be described as the follows.

A population of possible solutions is initially generated. The algorithm is divided into generations. In each generation, if the termination condition has not been met, next population will be first determined by selection and crossover. In selection, individuals in the current population are probabilistically selected according to their fitness to move to the next population. In crossover, pairs of individuals are probabilistically selected according to their fitness to generate a new pair of individuals (offspring) by the crossover operator. Once the next population has been determined, then mutation operation will be probabilistically applied to individuals in the next population. Finally, we re-label the next population to be the current population that is to symbolize the old population has died out. And the fitness of the new individuals will be evaluated. In each generation, only fit individuals can produce offspring and survive. Thus, a population of fit solutions would be expected when the algorithm terminates. And the best individual would be used as the solution to the problem.

3 Proposed Method

The idea of the proposed method is to view all the possible cycle decomposition of the unsigned permutation as the signed permutations that have the same gene order. Except node 0 and n+1, every node in the breakpoint graph has degree 2 for red edges and also for black edges. So, a cycle decomposition is used to define the (color) alternating paths. However, there is a corresponding signed permutation that actually defines the same alternating paths. Therefore, we can now turn our focus on signed permutation instead of cycle decomposition. Define the set $Signed(\pi)$ be the set of signed permutations that have the same gene order as π. For example, when π is 2 1, then $Signed(\pi)$ is { -2 -1, -2 +1, +2 –1, +2 +1}.

Thus, the size of *Signed(π)* is 2^n. The following two observations are required for our method.

Observation 1:

Each $\pi' \in Signed(\pi)$ can deduces a valid sorting sequence for π.

Proof:

Let sorting sequence ρ sort π' into identity (i.e. $\pi'_\rho = id$). Because $|\pi'[i]| = \pi[i]$, then $\pi'_\rho[i] = \pi_\rho[i]$, for all i. Thus, ρ can also sort π (i.e. $\pi_\rho = id$).

Observation 2:

There exist $\pi^* \in Signed(\pi)$ that deduces an optimal sorting sequence for π.

Proof:

Let ρ be a sorting sequence for π that uses minimum number of reversals. For each $\pi[k]$, let count[k] be the number of times that $\pi[k]$ is included in reversals of ρ, i.e. $\rho(i,j)$ $i \leq k < j$. Then π^* is the following, $\pi^*[k]$ has a positive sign if count[k] is even, otherwise it has a negative sign. Because $|\pi'[i]| = \pi[i]$, $\pi_\rho[i] = |\pi'_\rho[i]|$, for all i. However, all $\pi'_\rho[i]$, must be positive by our construct. Thus ρ can also sort π^* and the reversal distance for π^* is equal to the reversal distance for π.

From the two observations, we can see that the problem can be solved in O(2^n n) time. That is to find the reversal distance for all 2^n π'. Let π* be the π' that has minimum reversal distance. Then the sorting sequence of π is the sorting sequence of π*. However, it is not feasible to go through all 2^n π'. Thus, we use genetic algorithm to find π*. There is no guarantee that the genetic algorithm would find π*, however, we could expect the genetic algorithm would find a π' that has low reversal distance.

4 Experiment

We allow the population size to be n^2. The initial population is randomly generated binary strings representing the signs of the genes in the permutation. However, we apply a heuristic on taking all trivial cycles (i.e. cycles that compose of exactly one red edge and one black edge). Thus, sorted substrings would be assigned to the same sign (positive for ascending sorted substring; negative for descending sorted substring). The fitness is evaluated by the reversal distance of this signed permutation. Single point crossover and mutation are used with rate 0.3 and 0.8 respectively. And the genetic algorithm is terminated when the best reversal distance in the population remains unchanged in 20 generations.

We conduct the experiment by randomly generated permutations, where permutations are generated by performing n random swap operations on the identity permutation. Figure 4 shows the comparison between the 3/2-approximation algorithm and the proposed method. (The actual data is shown in Appendix A.) The figure shows the number of reversals required in the sorting sequences found by the two methods. The comparison is on the average solution of ten runs with *n* is between 10 to 150. We can see that the proposed method produce better solution than the 3/2-approximation algorithm.

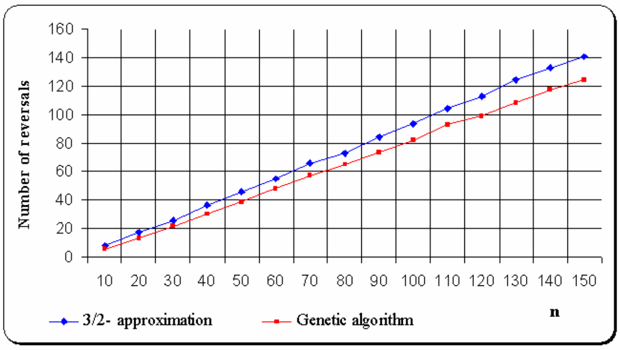

Figure 4. Performance comparison of 3/2-approximation and genetic algorithm.

5 Discussion

Sorting unsigned permutation has a trivial *n*-1 upper bound. That is, simply using one reversal to put one gene block in place. When *n*-1 gene blocks are in place, the *n*-th block is already in place. In fact, this is the reversal distance for the "hard-to-sort" Gollan permutation γ_n (and γ_n^{-1}). From Figure 4, we can see that the solutions found by both the proposed method and the 3/2-approximation are not very far from the *n*-1 upper bound. That observation is consistent with the analysis done by Bafna and Pevzner that says the reversal distance between 2 random permutations is very close to the *n*-1 upper bound. Figure 4 shows that the solutions produced by both the proposed method and the 3/2-approximation are growing linearly with *n*. However, the proposed method consistently outperforms the 3/2-approximation in various problem size (*n*). In fact, the solution produced by the proposed method has an average improvement of 12.8% over the solution produced by the 3/2-approximation. Therefore, the improvement is significant and scale up with *n*.

During our implementation, we found our heuristically initialized population enhances the average performance of the genetic algorithm. In fact, sorted substrings do not always stay together. Notice when we optimally sort the permutation 3 4 1 2, one of the sorted substrings (i.e. 3 4 or 1 2) has to be broken. However interestingly, it has been proven that there exists a sorting sequence that does not break sorted substring with length longer than 2 (Hannenhalli and Pevzner, 1996).

The computational time was not our major concern during the implementation of the proposed method. So, we used the $O(n^2)$ time Java implementation created by Itsik Mantin to find the reversal distance of signed permutations for the fitness evaluation (Mantin and Shamir, 1999). However, $O(n)$ time algorithm is known and can be used. Here we provide a time complexity comparison. The 3/2-approximation requires $O(n^2)$ time. On the other hand, the proposed method requires $O(n^2)$ size of population; each requires $O(n)$ time to evaluate its fitness in each generation; and there can be at most $O(n)$ generations (because the upper bound of reversal distance is n for signed permutations and the genetic algorithm enforces improvement on the best reversal distance in every 20 generations otherwise terminates). Therefore, the proposed method requires $O(n^4)$ time.

6 Conclusion

Sorting unsigned permutation by reversals served an important role in inferring evolutionary history. A 3/2-approximation has been proposed and is the lowest guaranteed error bound thus far. All previous methods use constructive approach to produce the solution. This paper introduces a new inductive approach that uses genetic algorithm to find the solution. Experimental result shows that the proposed method outperforms the 3/2-approximation algorithm, although it does not mathematically guarantee the quality of the solution. Due to the inductive perspective, different searching methods can then be applied to solve the problem. Also, this perspective creates possibilities to estimate the reversal distance by sampling the solution space. Such estimation allows the estimated answer on both higher and lower sides, while the previous methods always produce over estimated answer.

Acknowledgements

We would like to thank David Christie for all his help during our implementation of the 3/2-approximation algorithm. We also would like to thank Itsik Mantin and Ron Shamir make available to public the Java applet and the source code for the sorting signed permutation algorithm.

Bibliography

Bader, D. A., Moret, B. M. E. and Yan, M. (2001) “A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study,” Journal of Computational Biology, 8*(5):483-491.*

Bafna, V. and Pevzner, P. A. (1996) “Genome Rearrangements and Sorting by Reversals,” SIAM Journal on Computing, 25(2):273-289.

Bergeron, A. (2001) “A Very Elementary Presentation of the Hannenhalli-Pevzner Theory,” The Twelfth Annual Symposium on Combinatorial Pattern Matching, 106-117.

Berman, P. and Hannenhalli, S. (1996) “Fast Sorting by Reversal,” The Seventh Annual Symposium on Combinatorial Pattern Matching, 1075:168-185.

Caprara, A. (1997) “Sorting by Reversals is Difficult,” Proceedings of the First International Conference on Computational Molecular Biology, 75-83.

Christie, D. A. (1998) “A 3/2-Approximation Algorithm for Sorting by Reversals,” Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 244-252.

Diestel, R. (2000) “Graph Theory,” Springer Verlag.

Hannenhalli, S. and Pevzner, P. A. (1995) “Transforming Cabbage into Turnip,” Proceedings of the Twenty-seventh Annual ACM Symposium on the Theory of Computing, 178-189.

Hannenhalli, S. and Pevzner, P. A. (1995) “Transforming Men into Mice,” Proceedings of the Thirty-sixth Annual IEEE Symposium on Foundations of Computer Science, 581-592.

Hannenhalli, S. and Pevzner, P. A. (1996) “To Cut … or not to Cut,” The Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 304-313.

Kaplan, H., Shamir, R. and Tarjan, R. E. (1997) “Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals,” Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 344-351.

Kececioglu, J. and Ravi, R. (1995) “Of Mice and Men: Evolutionary Distance between Genomes under Translocation,” Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 604-613.

Kececioglu, J. and Sankoff, D. (1995) “Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement,” Algorithmica, 13:180-210.

Lovasz, L. and Plummer, M. D. (1986) “Matching Theory,” North-Holland.

Mantin, I. and Shamir, R. (1999) “An Algorithm for Sorting Signed Permutations by Reversals,” http://www.math.tau.ac.il/~rshamir/GR/ .

Pevzner, P. A. (2001) “Computational Molecular Biology An Algorithmic Approach,” MIT Press.

Russell, P. J. (2002) “iGenetics,” Benjamin Cummings.

Russell, S. J. and Norvig, P. (2002) “Artificial Intelligence: A Modern Approach,” 2nd ed. Prentice Hall.

Sankoff, D. (1992) “Edit Distance for Genome Comparison based on Non-local Operations,” The Third Annual Symposium on Combinatorial Pattern Matching, 644:121-135.

Sankoff, D., Cedergren, R. and Abel, Y. (1990) “Genomic Divergence through Gene Rearrangement,” Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences, 26:428-438.

Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B. and Cedergren, R. (1992) “Gene Order Comparisons for Phylogenetic Inference: Evolution of the Mitochondrial Genome,” Proceedings of the National Academy of Science USA, 89:6575-6579.

Slinko, A. (2002) “Combinatorics. Tutorial: 1 Permutations,” http://www.matholymp.com/ .

Tao, J., Ying, X. and Zhang, M. Q. (2002) “Current Topics in Computational Molecular Biology,” MIT Press.

Appendix A

Table 1. Permformance comparison of 3/2-approximation and genetic algorithm.

n	3/2-approximation	Genetic algorithm
10	7.4	5.6
20	16.8	13.5
30	25.7	21.3
40	36.3	29.8
50	45.7	38.6
60	54.5	47.7
70	65.4	57
80	73	64.7
90	84.1	73.7
100	93.5	82.3
110	104.5	92.4
120	112.8	98.6
130	124.1	108.5
140	132.9	117.7
150	140.7	124.3