Bioinformatics Advance Access published January 19, 2008

GENESIS: Genome Evolution Scenarios

Simon Gog, Martin Bader, and Enno Ohlebusch

Institute of Theoretical Computer Science, University of Ulm, D-89069 Ulm, Germany

Associate Editor: Dr. Chris Stoeckert

ABSTRACT

Summary: We implemented a software tool called GENESIS for three different genome rearrangement problems: Sorting a unichromosomal genome by weighted reversals and transpositions (SwRT), sorting a multichromosomal genome by reversals, translocations, fusions, and ssio ns (SRTl), and sorting a multichromosomal genome by weighted reversals, translocations, fusions, ssio ns, and transpositions (SwRTTl).

Availability: Source code can be obtained by the authors, or use the web interface http://www.uni-ulm.de/in/theo/research/genesis.html.

Contact: {martin.bader, simon.gog, enno.ohlebusch}@uni-ulm.de

1 INTRODUCTION

During evolution, DNA molecules are subject to local and global mutations. Local mutations (point mutations) consist of the substitution, insertion, or deletion of single nucleotides, while global mutations (genome rearrangements) change the DNA molecules on a large scale. In unichromosomal genomes, the most common rearrangements are inversions (also called reversals in bioinformatics), where - from a mathematical point of view - a section of the genome is excised, reversed in orientation, and re-inserted. Biologically, inversions can be caused by replication errors. But also large-scale duplications, deletions (gene loss), insertions (e.g. horizontal gene transfer), and transpositions play a role. In a transposition, a section of the genome is excised and inserted at a new position in the genome; this may or may not also involve an inversion. In genomes with multiple chromosomes further genome rearrangements are translocations (in a reciprocal translocation, two non-homologous chromosomes break and exchange fragments), fusions (where two chromosomes fuse), and fissions (where a chromosome breaks into two parts).

Since the 1980s, computer scientists have developed several algorithms for reconstructing genome rearrangement scenarios that transform one genome into another genome (or equivalently, to sort a signed permutation into the identity permutation). These algorithms can be categorized according to the genome rearrangement operations they can deal with, and as to whether they take multiple chromosomes into account. For unichromosomal genomes, the following results are known. If only reversals are allowed, Hannenhalli and Pevzner's [7] algorithm yields an exact solution to the problem. The currently best algorithm for transpositions is a 1.375-approximation [4]. For equally weighted reversals and transpositions, Hartman and Sharan [8] devised a 1.5-approximation algorithm. Bader and Ohlebusch [1] extended their algorithm to a 1.5-approximation algorithm for any weight ratio between 1:1 and 1:2 (reversals:transpositions). Another program for the weighted case is DERANGE [3], but the authors of [3] did not provide a guaranteed approximation ratio.

For multichromosomal genomes, less results have been obtained so far. The most realistic solution to the problem was given by Hannenhalli and Pevzner [6]. Their algorithm takes reversals, translocations, fusions, and fissions into account and returns an exact solution. To the best of our knowlegde, the new algorithm presented below is the first that augments their algorithm with transpositions.

We have implemented the following three algorithms:

1. The algorithm for SwRT by Bader and Ohlebusch [1] with quadratic running time. Moreover, the combination of this algorithm with a greedy strategy resulted in a practicable method. The price to be paid for this improvement is a cubic running time.
2. The algorithm for SRTl by Hannenhalli and Pevzner [6] with the improvements of Tesler [11] and of Ozery-Flato and Shamir [10]. The running time is quadratic.
3. An algorithm for SwRTTl, which is a combination of the two algorithms above. The running time of the algorithm is cubic. Our experiments show that it produces good results for biologically reasonable weights. To guarantee an approximation ratio of 2, we further combined the new algorithm with the strategy presented by Yancopoulus et al. [12], albeit for a different set of rearrangement operations.

GENESIS consists of two parts: The programs themselves and the web interface. The web interface is a simple front-end: One can choose the algorithm, set source and target genome, and gets the resulting rearrangement scenario. For the sake of readability, the web interface limits the size of permutations to 80, whereas the main programs can handle permutations of several thousands elements.

2 METHODS

All our algorithms work with the *reality-desire diagram* (also called *breakpoint graph*) as described in [2]. Let $c(\pi)$ denote the number of cycles in the reality-desire diagram, $c_{odd}(\pi)$ the number of cycles with an odd number of reality-edges (called *odd cycles*), and $c_{even}(\pi)$ the number of cycles with an even number of reality-edges (called *even cycles*). For the weighted algorithms, let w_r be the weight of reversals, translocations, fusions, and fissions, and let w_t be the weight of transpositions. We make the assumption that $w_r \leq w_t \leq 2w_r$.

The algorithm for SwRT is an implementation of the algorithm devised in [1]. The *score* of a permutation is defined by $\sigma(\pi) = c_{odd}(\pi) + (2 - 2 \cdot w_r/w_t)c_{even}(\pi)$. In each step, the algorithm

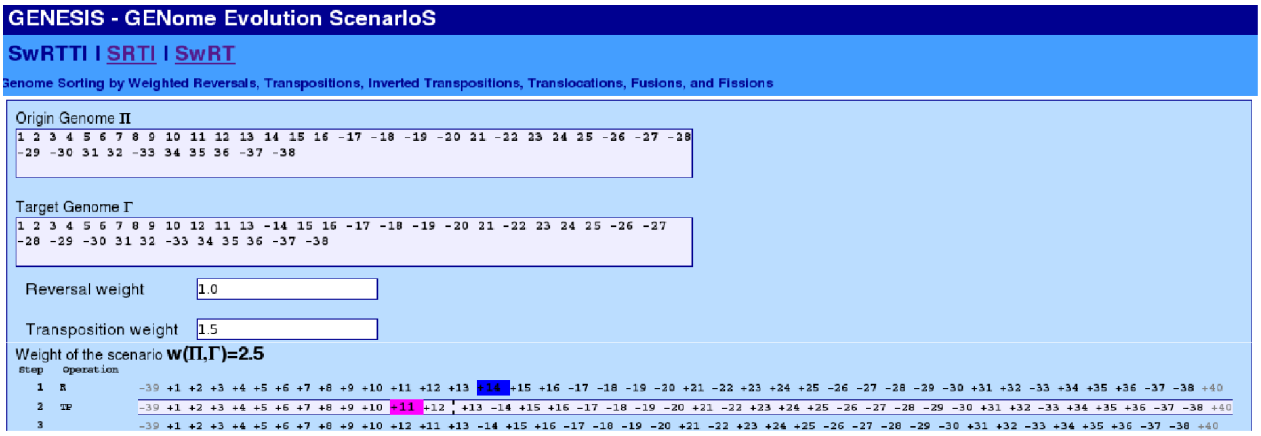

Fig. 1. Rearrangement scenario between the mitochondrial genomes of *Drosophila melanogaster* [5] and *Anopheles quadrimaculatus* [9]. The SwRT/SwRTTl-algorithm has found a scenario consisting of one reversal and one transpositions, whereas the SRTl algorithm calculates a scenario consisting of four reversals.

searches for a small starting sequence $op_1, \ldots, op_k$ with $k \leq 4$ and weight w and an increment $\Delta\sigma = \sigma(op_k(\ldots(op_1(\pi)\ldots))) - \sigma(\pi)$ of the score such that $\Delta\sigma/w \geq 4/(3w_t)$, and applies this starting sequence. This step is repeated until the permutation is sorted. As the score is maximized for the identity permutation and the maximum possible gain in score per weight is $2/w_t$, this yields a 1.5 approximation. We combined the algorithm with a greedy strategy that generates different starting sequences, out of which the one with the best gain in score per weight is applied. A further improvement of the output can be obtained if the greedy strategy is provided with a lookahead, which additionally considers the gain in score per weight of following starting sequences. Our experiments showed that the resulting method with limited lookahead is practicable.

The algorithm for SRTL is an efficient implementation of the algorithm proposed in [6]. First, the reality-desire diagram is built. Elements corresponding to chromosome boundaries are connected such that the distance between the genomes remains the same. Then, in each step the algorithm applies a reversal or translocation that reduces the distance $d(\Pi, \Gamma) = n + m - c + r + \lceil \frac{s' + gr' + fr'}{2} \rceil$ by one. In the formula, n is the number of genes, m the number of chromosomes, and the remaining parameters as well as details are explained in [10].

The algorithm for SwRTTl is a mixture of both algorithms. First, it creates the reality-desire diagram. Elements corresponding to chromosome boundaries are connected such that $\sigma(\pi)$ is maximized. Then, the algorithm generates possible starting sequences as in the greedy strategy of the SwRT-algorithm. However, some of these sequences may contain forbidden transpositions because chromosome boundaries are not allowed in transposed segments. These starting sequences are ignored, and out of the remaining sequences the one with the best increment of score per weight is applied. If all the starting sequences are forbidden, then there must be a translocation that increases the number of cycles by one, and the algorithm applies this translocation. Again, these steps are repeated until the permutation is sorted.

Figure 1 exemplifies the application of our programs to two mitochondrial genomes. A more complex example consists of the whole genomes of man and mouse (data is available on our website). In this case SRTl produces a scenario with 35 operations, while our new algorithm SwRTTl finds a scenario with only 33 operations.

REFERENCES

[1]M. Bader and E. Ohlebusch. Sorting by weighted reversals, transpositions, and inverted transpositions. *Journal of Computational Biology*, 14(5):615–636, 2007.

[2]V. Bafna and P.A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[3]M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene*, 172:GC11–17, 1996.

[4]I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379, 2006.

[5]R. Garesse. Drosophila melanogaster mitochondrial DNA: Gene organization and evolutionary considerations. *Genetics*, 118(4):649–663, 1988.

[6]S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genetic distance problem). In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[7]S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999.

[8]T. Hartman and R. Sharan. A 1.5-approximation algorithm for sorting by transpositions and transreversals. *Journal of Computer and System Sciences*, 70(3):300–320, 2005.

[9]S. Michell, A. Cockburn, and J. Seawright. The mitochondrial genome of Anopheles quadrimaculatus species: A complete nucleotide sequence and gene organization. *Genome*, 36:1058–1073, 1993.

[10]M. Ozery-Flato and R. Shamir. Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology*, 1(1):71–94, 2003.

[11]Glenn Tesler. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences*, 65(3):587–609, 2002.

[12]S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.