ELSEVIER

Gene 172 (1996) GC 11–17

GENE-COMBIS™

Parametric genome rearrangement

Mathieu Blanchette [a,1], Takashi Kunisawa [b,2], David Sankoff [a,*]

[a] *Centre de recherches mathématiques, Université de Montréal, CP 6128 Succursale Centre-ville, Montréal, Québec H3C 3J7, Canada*
[b] *Department of Applied Biological Sciences, Science University of Tokyo, Noda 278, Japan*

Received 6 September 1995; revised 8 December 1995; accepted 14 December 1995

Abstract

Algorithms inspired by comparative genomics calculate an edit distance between two linear orders based on elementary edit operations such as inversion, transposition and reciprocal translocation. All operations are generally assigned the same weight, simply by default, because no systematic empirical studies exist verifying whether algorithmic outputs involve realistic proportion of each. Nor do we have data on how weights should vary with the length of the inverted or transposed segment of the chromosome. In this paper, we present a rapid algorithm that allows each operation to take on a range of weights, producing an relatively tight upper bound on the distance between single-chromosome genomes, by means of a greedy search with look-ahead. The efficiency of this algorithm allows us to test random genomes for each parameter setting, to detect gene order similarity and to infer the parameter values most appropriate to the phylogenetic domain under study. We apply this method to genome segments in which the same gene order is conserved in *Escherichia coli* and *Bacillus subtilis*, as well as to the gene order in human versus *Drosophila* mitochondrial genomes. In both cases, we conclude that it is most appropriate to assign somewhat more than twice the weight to transpositions and inverted transpositions than to inversions. We also explore segment-length weighting for fungal mitochondrial gene orders.

1. Introduction

The algorithmic study of comparative genomics has focused on inferring the most economical explanation for observed differences in gene orders in two or more genomes in terms of a limited number of rearrangement processes. For single-chromosome genomes, this has been formulated as the problem of calculating an edit distance between two linear orders on the same set of objects, representing the ordering of homologous genes in two genomes. In the most realistic version of the problem, a sign (plus or minus) is associated with each object in the linear order, representing the direction of transcription, or strandedness, of the corresponding gene. The elementary edit operations may include one or more of:

(1) Inversion, or reversal, of any number of consecutive terms in the ordered set, which, in the case of signed orders, also reverses the polarity of each term within the scope of the inversion [1,5,4,8–11].

(2) Transposition of any number of consecutive terms from their position in the order to a new position between any other pair of consecutive [2]. This may or may not also involve an inversion.

In addition, for multi-chromosome genomes, a major role is played by: (3) reciprocal translocation [3,7].

Where there is more than one type of edit operation, the tendency has been to consider all types as contributing the same amount (cost or weight) to the edit distance [6,14]. This is hardly justified, since, for example, transpositions are observed much less frequently than inversions in many evolutionary contexts, and hence should cost much more. Some effort has been made to devise algorithms where the operations are weighted differently [13,14], and has led both to technical problems in extending the algorithms, and to questions of what costs or weights to use.

There are as yet no systematic studies, either within or across major phylogenetic groups, of the relative frequencies of inversion, transposition, and other rearrangement processes, based on any type of comparative mapping. Consequently, it is not known what values to assign to the cost parameters for the different types of rearrangement in order to infer an evolutionary history with a biologically

* Corresponding author. E-mail: sankoff@ere.umontreal.ca
[1] E-mail: blanchet@hans.crm.umontreal.ca
[2] E-mail: kunisawa@jipdalph.rb.noda.sut.ac.jp

SSDI 0378-1119(95)00878-0

realistic proportion of each of these types. And we have even less knowledge of the typical lengths of chromosome fragments which participate in the rearrangement processes. A consequence of this lack of data is that we cannot test whether two genomes are closer together than random genomes since we do not have an empirical basis on which to base simulation studies. Eventually, knowledge of this type may be generated through comparative studies, but cost parameter assignment is unlikely to ever be a cut-and-dried process. It is clear, for example, that there is no clock, and hence no universal parameter values, for these types of evolutionary operations; translocations are common in mammals, almost non-existent in *Drosophila*. Inversions are common in animal mitochondrial genomes, rare in fungal mitochondrial genomes.

In this paper, we suggest approaches to these types of problem, focusing on single-chromosome genomes. While there does not seems to be much hope in transferring the algorithmic machinery developed for the uniform cost problem to the parametric problem, where each process can take on a range of weights, a rapid and constructive upper bound algorithm based on a greedy search with limited look-ahead has been shown to be rather precise in many contexts [10], and incorporates weights with no difficulty. The efficiency of this algorithm can be greatly improved by including some branch-and-bound techniques and by excluding some highly unlikely configurations. This allows us to test experimentally characterized genomes against a large number of random genomes for each parameter setting. Under the hypothesis that ''wrong'' values of the parameters introduces noise into the analysis, the ''true'' value of the genomic distance is likely to be close to those found by parameter settings whose output deviates as much as possible from randomness. In this way we can simultaneously test for detectible gene order similarity, and also infer the parameter values most appropriate to the phylogenetic domain under study.

2. Formalization and algorithm

Our discussion applies to circular genomes, though it is easily modified for the linear case. Starting with any gene, label the genes in one genome in order $1,2,\ldots,n$, and their homologues as a permutation $p_1,p_2,\ldots,p_n$, where p_i has negative polarity iff gene i is transcribed in the opposite directions in the two genomes. The problem is to find the most economical sequence of moves (inversions, transpositions and inverted transpositions) that transforms $p_1,p_2,\ldots,p_n$ into the identity permutation $1,2,\ldots,n$, or its inverse $-n,-n+1,\ldots,-1$.

A breakpoint is any adjacent pair (p_i,p_{i+1}), including (p_n,p_1), such that $p_{i+1}-p_i \neq 1$, and $p_{i+1}-p_i \neq 1-n$, for $i=1,\ldots,n$. The identity permutation and its inverse being the only ones having no breakpoint, a solution to our problem requires the successive elimination of breakpoints from the permutation p. Inversions affect two adjacent pairs, so that they change the number of breakpoints by two at most. Transpositions and inverted transpositions involve three pairs and can eliminate up to three breakpoints.

inversion: ...ab...cd... becomes ...a −c...−bd...

transposition: ...ab...cd...ef... becomes ...ad...eb...cf... or (with inversion) ...ad...e −c...−bf...

An exhaustive search for a solution considers all possible moves on p, and for each new permutation thus produced, solves the problem recursively until all possible ways of arriving at the identity permutation in less than n moves [10] have been examined. This is not feasible for even moderate n, since the number of moves possible for each permutation is on the order of n^3, so that the search tree soon becomes unmanageable. Even restricting the search to moves where the two or three adjacent pairs affected are all among the b breakpoints, while in practice reducing the calculations considerably, does not change the fundamental computational complexity of the problem.

We adopt a number of compromises which, although they reduce the computation drastically by sacrificing the guarantee of optimality, are unlikely to produce a solution containing more than one or two extra moves:

(1) We consider only moves that reduce the number of breakpoints. This reduces the search on each permutation to a task quadratic in b.

(2) We use a depth-first search with limited look-ahead and branch-and-bound; with look-ahead $k=1$, the best possible move is executed on p, then the best possible on the resulting permutation and so on, until the identity is reached.

(3) To implement a longer look-ahead, e.g. $k=4$, we do not try all possible combinations to find the four best moves; for the first move we consider all possibilities that remove at least one breakpoint; for the second move, if there are any that remove at least two breakpoints, we consider only them, and for the third move, if there are any transpositions that remove three breakpoints, we consider only them. Empirically, restricting the number of possibilities in this way in order to achieve a longer look-head results in better solutions. Note that once an optimum sequence of k moves is found, we only implement the first of these before beginning our search again.

The exhaustive search (within the constraints just mentioned) in a look-ahead is accelerated by a branch-and-bound technique. Once a first solution for a k-move look-ahead is reached, we backtrack, undoing the most recent move and trying the next-best alternative as long as one exists. If not, we undo the most recent move at the previous level. After each backtrack step, we again proceed until a complete k-move look-ahead is found, and this solution is evaluated to see if it is better than, and thus can replace, the current best look-ahead. For any point r in the

look-ahead search tree, if m moves are required to reach r from the starting point q, and there remain b breakpoints more than in the current best k-move look-ahead from q, where

$$m + \frac{b}{3} > k,$$

then point r, and all its ''descendants'', need not be considered since they cannot possibly lead to an optimal solution, each move removing at most three breakpoints. (This rule is actually implemented in terms of costs – see Section 3 below – and not moves.)

Much work on rearrangement algorithms has been devoted to improving lower bounds, e.g. the $b/3$ in the preceding paragraph, [2,11], and when this is successful, it may allow optimal look-ahead for large k, and eventually, if $k \geq n$, optimal solutions for the full rearrangement problem, but for the generalizations we shall discuss, particularly those in Section 3, this does not seem feasible.

One technique that we found to improve our solution, though it does not further restrict the search space, is ''chunneling''; instead of proceeding from the identity to p, we search for the best k moves from the identity to p and the best k in the other direction, and choose the better of the two. The solution is thus constructed from both ends.

3. Weights

The goal of this research is to explore the effect of weighting each kind of move differently. In analogy to parametric sequence alignment [15], this provides a range of solutions, in which the proportions of the different kinds of moves vary.

For single-chromosome genomes, we not only need to weight transpositions differently from inversions, we also need to account for inverted transpositions. Somewhat arbitrarily, we weight inverted transpositions in the same way as transpositions; once a genome fragment is displaced, it does not seem of overwhelming importance whether it is re-inserted in its original orientation, or on the opposite strand. In phylogenetic domains where transcription is universally from one strand, perhaps this parameter should instead be weighted in the same way as inversions: with high or infinite cost.

For technical reasons, we included in our local cost function a term which depends on how many breakpoints are removed by an operation. At first glance this might seem superfluous, since the sum over all operations in a solution of the number of breakpoints removed is a constant; it does not change from solution to solution. Since our algorithm tries to find an optimal solution by a greedy search strategy, however, it is important to include the breakpoints term, given the reduced search space of the algorithm; it ensures that inversions that remove two breakpoints or transpositions that remove three are not overlooked. The cost for any sequence of moves is then

$$cost = \sum_{move_x \text{ in sequence}} w_x - B_x$$

where w and B indicate the weight of a move and the number of breakpoints it removes, respectively. To restore to this (possibly negative) cost the properties of a metric, we can simply add a constant representing the number of breakpoints.

The inversion weight was fixed at $w_I = 1$, and we varied the transposition weight from somewhat less to somewhat more than $w_T = 2$, the ''phase transition'' point in solution space, where for a typical permutation the best transposition x or inversion y have $w_T - B_x = 0$ or $w_I - B_y = 0$; there is almost always a transposition possible in a permutation that removes two breakpoints and an inversion that removes one, and only rarely in most permutations is there a move that removes an extra breakpoint. Thus for $w_T < 2$, transpositions are favoured, while for $w_T > 2$, inversions are favoured.

We also added an optional cost $\alpha L > 0$ on each move, depending on the length L of the segment affected. This enables us to model the hypothesis that shorter moves may be favoured in some biological contexts.

4. Implementation

The C program DERANGE II (Accession Number **BC00444** [3]) incorporating weights and performing the algorithm in Section 2 is available from the authors. This program was used in the analyses discussed below.

5. The bacterial genome

Associating different weights to each kind of move allows us to influence the proportion in which the different moves will be used in the solution. For each parameter setting, we compare the analysis of the experimental permutation with the same analysis of 1,000 random permutations.

Kunisawa [12] identified some 47 genomic segments containing two or more homologous genes conserved during the evolutionary divergence of *E. coli* and *B. subtilis*. These segments contain a total of 255 genes. As input to DERANGE II, we considered each of these segments as if they constituted a single ''gene'', since conserved segments need never be split up during the analysis.

The results are shown in Fig. 1, which plots the number of moves required to sort a permutation (normalized by n, the length of the permutation), as a function of the weight

[3] ftp://ftp.ebi.ac.uk/pub/software/unix/derange2.tar.Z

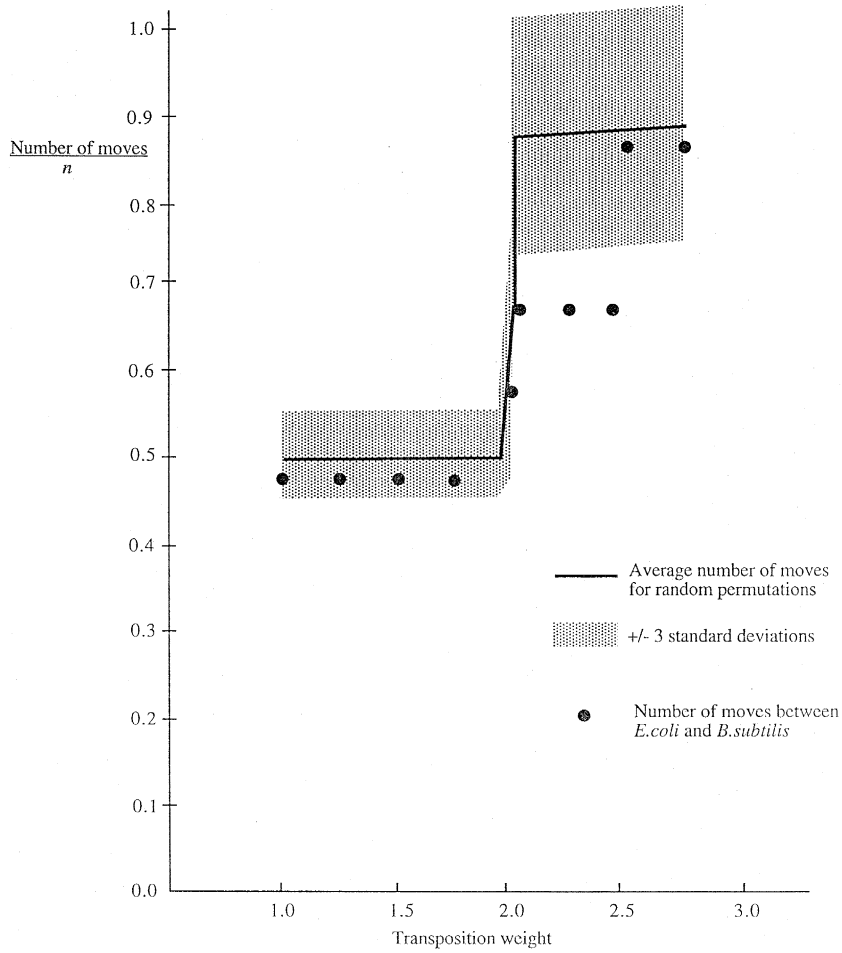

Fig. 1. Number of moves required to sort permutation (normalized by n, the length of the permutation), as a function of the weight associated with a transposition (inversion weight = 1).

associated with a transposition. (Inversion weight = 1.). Most of the data curve falls directly on the random permutation curve. A significant deviation occurs, however, between $2 < w_T < 2.5$, where the experimental permutation requires only 31 moves: 16 inversions and 15 transpositions, about 11 moves less than the random permutations, which have only one or two transpositions each. Both the economical accounting for the observed data, and the inference that many transpositions are retained despite their elevated cost, suggest that this may be a meaningful solution, and that w_T slightly greater than 2 may be an appropriate value.

6. The mitochondrial genome

Gene orders were obtained from F. Lang, drawn from his database on mitochondrial genomes. Testing the 37 homologous genes in human and *Drosophila* shows a great deal of non-randomness. This, however, is already clear from the fact that there are only 21 breakpoints in this comparison, instead of the 35 or so to be found in typical random genomes.

Normalizing the results by the number of breakpoints in the initial permutation, however, equivalent to analyzing permutations of conserved segments as in Section 5 rather than gene orders, gives additional results, seen in Fig. 2. First, for $w_T < 2$, it takes 12 moves to sort the 21-breakpoint *Drosophila*-human permutation, somewhat more than for random permutations. For $w_T > 2$, on the other hand, 16 moves including 13 inversions, or 17 moves including 15 inversions are required, less than for random permutations. In particular, the number of inversions is 30% below what would be expected from the simulations. Some transpositions do not seem to be replaceable by inversions, even with high values of w_T. (Note that the random curve constructed in Section 5 also serves here, since when the number of moves in a solution is normalized by the number of breakpoints, results from random trials do not vary perceptibly over a wide range of n.)

The deviations from randomness of this solution suggest that $2 < w_T < 2.5$ is an appropriate weighting in this

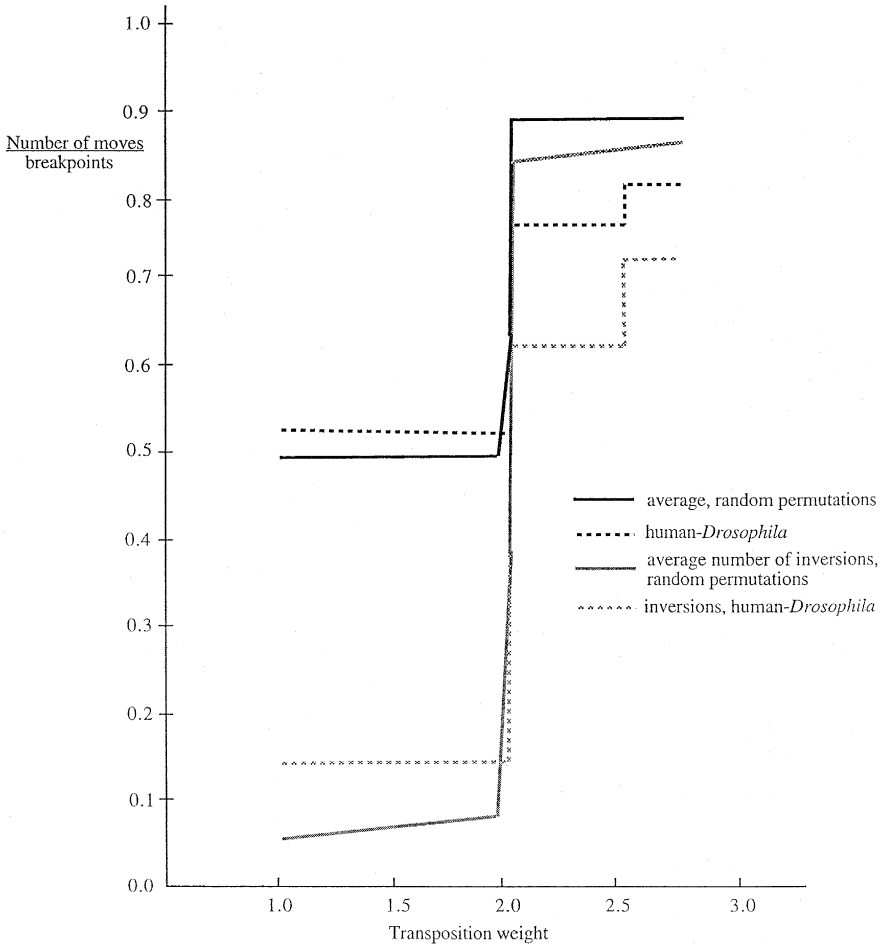

Fig. 2. Total number of moves in optimal sorting of permutation, and number of inversions in this sorting, both normalized by the number of breakpoints in the initial permutation.

context as well, giving a solution with 16 moves, of which 13 are inversions.

7. Weighting segment lengths

In a second series of experiments, we added a linear function of the physical length (in base pairs) of the segment to the cost of transposing or inverting it, lacking empirical grounds for choosing some biologically more realistic convex function. Here again, we plotted curves representing the average total number of moves necessary to solve a random permutation (normalized by the number of breakpoints in the initial permutation) as a function of the weight α, while transposition and inversion weights were held at 1.

In Fig. 3, the random permutation curve (sample size 100 at each point) is shown as background to the curve obtained from the mitochondrial genomes of two fungi, *Neurospora crassa* and *Aspergillus nidulans*. As α increases, the weighted distance is affected less (about 50% less) than the simulation results. This suggests that the ''true solution'' may not involve the movement of many long segments, so that penalizing long moves has a relatively small effect.

The region where the distance is sensitive to α, roughly $10^{-3} < \alpha < 10^{-2.5}$ may represent a meaningful range for this parameter, since it would seem to reflect the increasing cost of some stable solution. The insensitivity of the distance to changes in α at higher values necessarily reflects different solutions for each value of the parameter. On the other hand, lower values of α do not have an appreciable effect on the analysis of sequences of this length. Of course, other genome pairs may give somewhat different results.

8. Conclusions

The number of breakpoints in a genome or, equivalently, the number of conserved segments, contains much

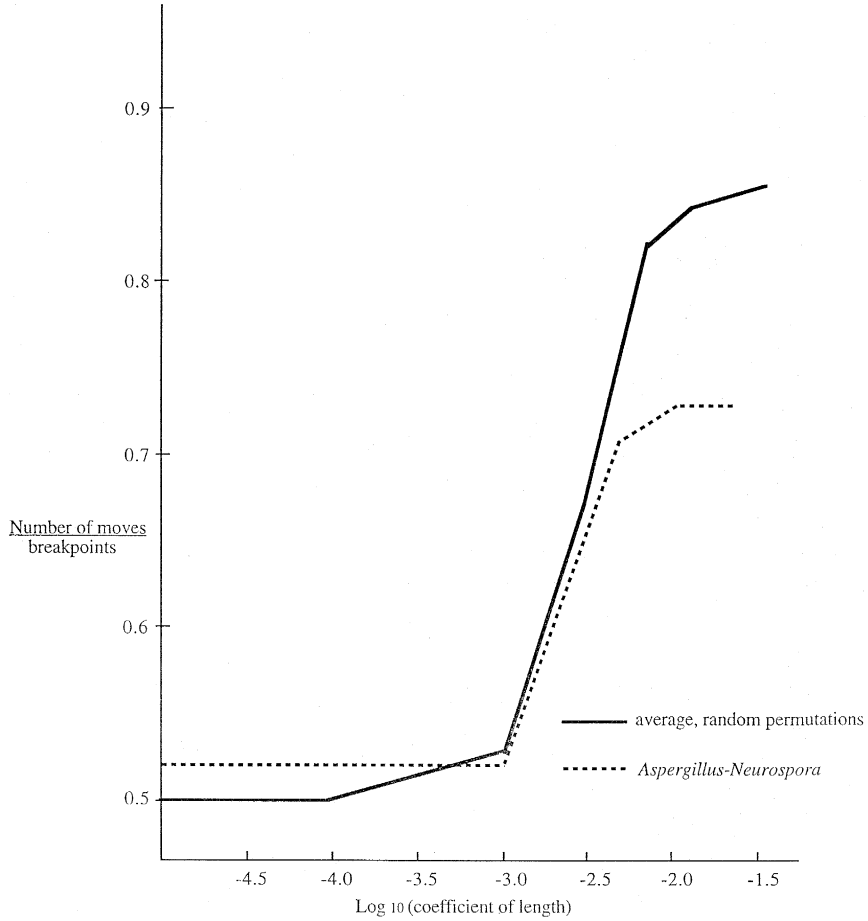

Fig. 3. Normalized number of moves required to sort a permutation as a function oflength coefficient in cost function.

of the information about the amount of rearrangement between two genomes. Both the bacterial genome comparisons and the mitochondrial genome comparisons, however, show that algorithmic inference of rearrangement events can yield additional information. In our framework, the weighting $2w_I < w_T < 2.5w_I$ seems to be the most revealing. This puts into question analyses of genome rearrangement that count each event equally.

Our exploration of segment-length weights points out the sensitivity of some rearrangement analyses to this parameter, and hence to the need for the empirical study of the lengths of inversions and transposed segments.

Acknowledgements

Research supported by grants from the Natural Sciences and Engineering Research Council of Canada and the Canadian Genome Analysis and Technology program. D.S. is a fellow of the Canadian Institute for Advanced Research.

References

[1] Bafna, V. and Pevzner, P.A. (1993) Genome rearrangements and sorting by reversals. 34th Annu. IEEE Symp. Found. Computer Sci., *SIAM J. Comput.*, pp. 148–157.

[2] Bafna, V. and Pevzner, P.A. (1995) Sorting by transpositions. *Proc. 6th Annu. ACM-SIAM Symp. Discrete Algorithms*, pp. 614–623.

[3] Hannenhalli, S. (1995) Polynomial algorithm for computing translocation distance between genomes. Proc. 6th Symp. Combinat. Pattern Matching, *Springer-Verlag Lect. Notes Computer Sci.*, 162–176.

[4] Hannenhalli, S., Chappey, C., Koonin, E.V. and Pevzner, P.A. (1996) Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics*, in press.

[5] Hannenhalli, S. and Pevzner, P.A. (1995) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proc. 27th Ann. ACM-SIAM Symp. Theory Comput.*, pp. 178–189.

[6] Hannenhalli, S. and Pevzner, P.A. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). *Proc. 36th Annu. Symp. Found. Computer Sci.*

[7] Kececioglu, J. and Ravi, R. (1995) Of mice and men. Evolutionary distances between genomes under translocation. *Proc. 6th Annu. ACM-SIAM Symp. Discrete Algorithms*, pp. 604–613.

[8] Kececioglu, J. and Sankoff, D. (1992) Exact and approximation algorithms for sorting by reversals. *Centre Rech. Math. Tech. Rep.*, (July) 1824.

[9] Kececioglu, J. and Sankoff, D. (1993) Exact and approximation algorithms for the inversion distance between two chromosomes. Proc. 4th Symp. Combinat. Pattern Matching, *Springer-Verlag Lect. Notes Computer Sci.* **684**, 87–105.

[10] Kececioglu, J. and Sankoff, D. (1994) Efficient bounds for oriented chromosome inversion distance. Proc. 5th Symp. Combinat. Pattern Matching, *Springer-Verlag Lect. Notes Computer Sci.* **807**, 307–325.

[11] Kececioglu, J. and Sankoff, D. (1995) Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica* **13**, 180–210.

[12] Kunisawa, T. (1994) Identification and chromosomal distribution of DNA segments conserved since divergence of *Escherichia coli* and *Bacillus subtilis. J. Mol. Evol.* **39**.

[13] Sankoff, D. (1992) Edit distance for genome comparison based on non-local operations. *Proc. 3rd Symp. Combinat. Pattern Matching, Springer-Verlag Lect. Notes Computer Sci.* **644**, 121–135. (1992).

[14] Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B.F. and Cedergren, R. (1992) Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. *Proc. Natl. Acad. Sci. USA* **89**, 6575–6579.

[15] Waterman, M.S., Eggert, M. and Lander, E. (1992) Parametric sequence comparison. *Proc. Natl. Acad. Sci. USA* **89**, 6090–6093.