BIOINFORMATICS

Vol. 21 Suppl. 1 2005, pages i502–i508
doi:10.1093/bioinformatics/bti1037

Reversal distance for partially ordered genomes

*Chunfang Zheng, Aleksander Lenert and David Sankoff**

University of Ottawa, 585 King Edward Avenue, Ottawa K1N 6N5, Canada

Received on January 15, 2005; accepted on March 27, 2005

ABSTRACT

Motivation: The total order of the genes or markers on a chromosome inherent in its representation as a signed permutation must often be weakened to a partial order in the case of real data. This is due to lack of resolution (where several genes are mapped to the same chromosomal position) to missing data from some of the datasets used to compile a gene order, and to conflicts between these datasets. The available genome rearrangement algorithms, however, require total orders as input. A more general approach is needed to handle rearrangements of gene partial orders.

Results: We formalize the uncertainty in gene order data by representing a chromosome from each genome as a partial order, summarized by a directed acyclic graph (DAG). The rearrangement problem is then to infer a minimal sequence of reversals for transforming any topological sort of one DAG to any one of the other DAG. Each topological sort represents a possible linearization compatible with all the datasets on the chromosome. The set of all possible topological sorts is embedded in each DAG by appropriately augmenting the edge set, so that it becomes a general directed graph (DG). The DGs representing chromosomes of two genomes are combined to produce a bicoloured graph from which we extract a maximal decomposition into alternating coloured cycles, and from which, in turn, an optimal sequence of reversals can usually be identified. We test this approach on simulated incomplete comparative maps and on cereal chromosomal maps drawn from the Gramene browser.

Contact: sankoff@uottawa.ca

1 INTRODUCTION

Structural rearrangement within a chromosome can be modeled by the transformation of one signed permutation to another by means of reversals (inversions). Analyses that compare gene orders, such as the Hannenhalli–Pevzner algorithm (1999), infer the smallest number of reversals to transform one order to another, i.e. to transform an arbitrary string containing either one positive or negative occurrence of each of $\{1, 2, \ldots, n\}$ to the reference order $1, 2, \ldots, n$, where the allowed operations consist of reversals of any substring (while changing the polarity of each term in the substring).

The total order of a chromosome inherent in its representation as a permutation or string must often be weakened in the case of realistic data, where mapping information only suffices to partially order the set of genes on a chromosome. Unfortunately, the concepts and methods of genome rearrangement pertain only to totally ordered sets of genes, markers or segments, and are meaningless in the context of partial orders.

Here we extend genome rearrangement theory to the more general context where both gene orders are represented by directed acyclic graphs (DAGs) rather than linear strings. The use of DAGs reflects uncertainty of the gene order on chromosomes in the genomes of most advanced organisms. This may be due to lack of resolution (where several genes are mapped to the same chromosomal position) to missing data from some of the datasets used to compile a gene order and/or to conflicts between these datasets.

We construct the DAGs for each species from two or more partially incompatible databases, or a single low-resolution dataset. The lack of order information in each dataset, due to missing genes or missing order information, is converted into parallel subpaths within the DAG in a straightforward manner.

Outright conflicts of order create cycles that must be broken. We suggest a number of reasonable alternative conventions for breaking cycles. This is not the focus of our analysis, however; whatever convention is adopted does not affect our subsequent analysis.

The genome rearrangement problem then is *to infer a sequence of reversals which transform a linearization (topological sort) of the DAG for one genome to a linearization of the DAG for the other genome, minimizing the number of reversals required*. Each topological sort represents one of the possible linearizations of all the partial information in all the datasets on a genome. We embed the set of all possible topological sorts in each DAG by appropriately augmenting the edge set, so that it becomes a general directed graph (DG).

We then combine the edges of the two DGs, representing two genomes, to produce a single large bicoloured graph from which we extract a maximal decomposition into alternating coloured cycles, so that a Hannenhalli–Pevzner type of procedure can then generate an optimal sequence of rearrangements. We focus here on obtaining the cycle decomposition; this is equivalent to optimally linearizing the partial orders, so

*To whom correspondence should be addressed.

that finding the actual rearrangements can be done using the previously available algorithms.

We test our method on simulated incomplete comparative maps and on homologous maize and sorghum chromosomal maps.

2 GENE ORDER DATA

Maps of genes or other markers produced by recombination analysis, physical imaging and other methods—no matter how highly resolved—inevitably are missing some (and usually most) genes or markers and fail to order, or incorrectly order, some pairs of neighboring genes with respect to each other. Even at the ultimate level of resolution, that of genome sequences, the application of different gene-finding protocols often gives maps that differ at the level of gene content.

Moreover, experimental methodologies and statistical mapping procedures inevitably give rise to some small proportion of errors: two neighbouring genes incorrectly ordered, a gene mapped to the wrong chromosome, a gene incorrectly named or annotated. However it is not these errors we focus on in this paper, but the more widespread issues of lack of resolution and genes missing from a map. These should not be considered errors; they are normal and inherent in all ways of constructing a map, except for highly polished genome sequencing with accurate gene identification (something that has not yet been achieved in the higher eukaryotes, even for humans).

A linear map that has several genes or markers at the same position p, because their order has not been resolved, can be considered a partial order, where all the genes before p are ordered before all the genes at p, and all the genes at p are ordered before all the genes following p, but the genes at p are not ordered amongst themselves. We call this procedure **make_po**.

For many genomes, there exist two or more gene maps constructed from different kinds of data or using different methodologies. There is only one meaningful way of combining the order information on two (partially ordered) maps of the same genome containing somewhat different subsets of genes, as long as there are no conflicting order relations ($a < b, b < a$) in the two, namely by taking the union of the partial orders, and extending this set through transitivity. This procedure is **combine_po**.

All the compatible partial order data on a genome can be represented in a minimal DAG whose vertex set is the union of all gene sets in the contributing gene maps, and whose edges correspond to just those order relations that cannot be derived from other order relations by transitivity. The outcome of this construction, **dagger**, is illustrated in Figure 1.

On the other hand, if different maps of the same genome conflict, there are a variety of possible ways of resolving the conflict or, equivalently, of breaking any potential cycles in the construction of the DAG. One way is to delete all order relations that conflict with at least one other order relation.

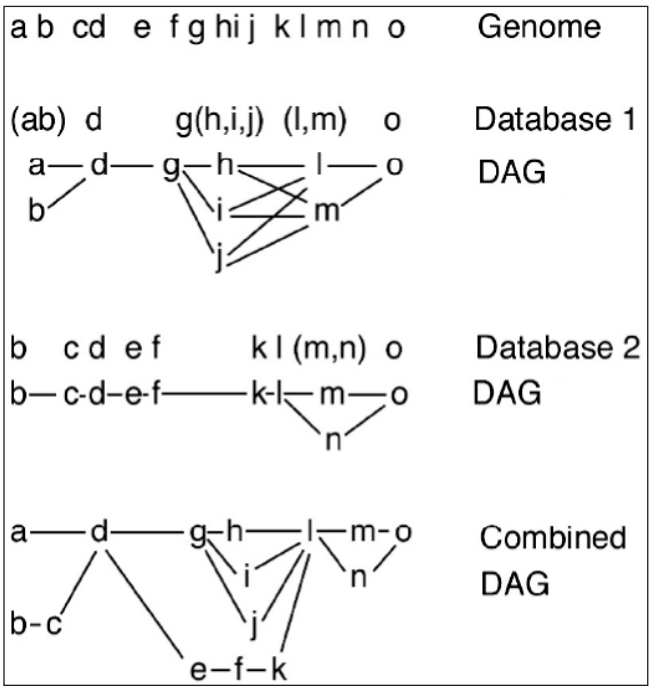

Fig. 1. Construction of DAGs from individual databases each containing partial information on genome, due to missing genes and missing order information, followed by construction of combined DAG representing all known information on the genome. All edges directed from left to right.

Another is to delete a minimal set of order relations so that all conflicts can be resolved. Still another is to ignore a minimum set of genes that will accomplish the same end. Any of these approaches, or others, which we denote by the generic routine name **resolve**, will produce results appropriate for our subsequent analysis.

3 THE DG EMBEDDING OF TOPOLOGICAL SORTS

A DAG can generally be linearized in many different ways, all derivable from a topological sorting routine. All the possible adjacencies in these linear sorts can be represented by the edges of a DG containing all the edges of the DAG plus two edges of opposite directions between all pairs of vertices, that are not ordered by the DAG. This is illustrated in Figure 2. The routine for constructing this graph is **dgger**.

4 THE ALGORITHM

4.1 Background

Before discussing our algorithm for comparing DGs derived from our DAG representations, we review the existing technology for the special case when the DAG and the associated

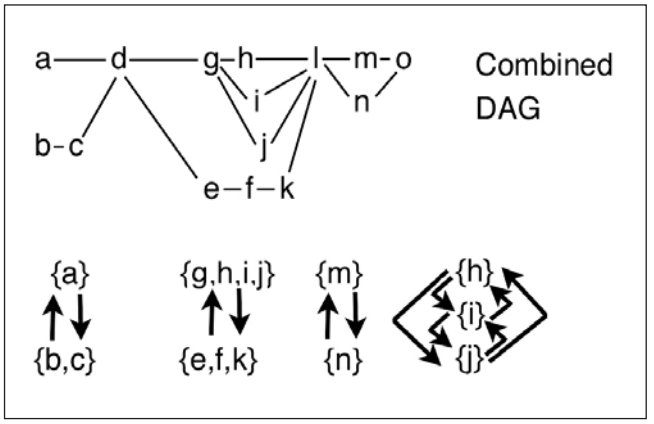

Fig. 2. Edges added to DAG to obtain DG containing all linearizations as paths (though not all paths in the DG are linearizations of the DAG!). Each arrow represents a set of directed edges, one from each element in one set to each element of the other set.

DG represent a total order, which is the traditional subject of computational comparative genomics.

The Hannenhalli–Pevzner construction of a shortest sequence of reversals for transforming one genome R, of length n, into another B begins by combining the signed permutation representations of the two genomes. The following procedure **make_bicoloured** produces a bicoloured graph on $2n + 2$ vertices that decomposes uniquely into a set of alternating-coloured cycles. Each gene or marker x determines two vertices, xt and xh, to which two additional vertices s and f (for start and finish) are added. One colour edge, say red, is determined by the adjacencies in R. If x is the left-hand neighbour of y in R, and both have positive polarity, then xh is connected by a red edge to yt. If they both have negative polarity, it is xt that is joined to yh. If x is positive and y negative, or x is negative and y positive, xh is joined to yh, or xt is joined to yt, respectively. If x is the first gene, then s is joined to xt or xh depending on whether x has positive or negative polarity, respectively. If x is the last gene, then f is joined to xt or xh depending on whether x has negative or positive polarity.

Black edges are added according to the same rules, based on the adjacencies in genome B.

It can be seen that each vertex is incident to exactly one red edge and one black edge and that the bicoloured graph decomposes uniquely into a number of alternating cycles. If the number of cycles is c, then the number of reversals r necessary to convert R into B is given by the Hannenhalli–Pevzner equation:

$$r = n + 1 - c + h + f \quad (1)$$

where h and f are corrections (for 'hurdles and fortresses'), which are usually zero for simulated or empirical data. For simplicity of exposition, we will ignore these corrections here, though an eventual full-scale program will incorporate them with no difficulty.

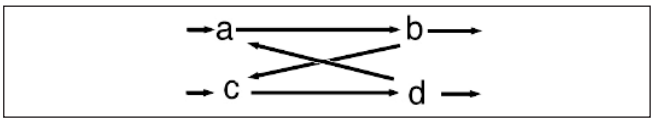

Fig. 3. If ab and cd are DAG edges, then the two non-DAG edges da and bc are mutually exclusive, since together they lead to the wrong order for a and b.

4.2 Generalization to partial orders

The routine **make_bicoloured** can also be applied to the set of edges in the DGs for the two genomes. In the resulting graph, each of the DAG edges and both of the edges connecting each of the unordered pairs in the DG represent potential adjacencies in our eventual linearization of a genome. The n genes or markers determine $2n + 2$ vertices and the potential adjacencies determine the red and black edges, based on the polarity of the genes or markers. Where the construction for the totally ordered genomes contains exactly $n + 1$ edges of each colour, in our construction in the presence of uncertainty there are more than $2(n+1)$ potential edges, but only $2(n+1)$ can be chosen in our construction of the cycle graph, which is equivalent to the simultaneous linearization by topological sorting of each genome.

It is this problem of selecting the right subset of edges that makes the problem difficult (and, we conjecture, NP-hard).

The choice of certain edges generally excludes the choice of certain other ones. This is not just a question of avoiding multiple edges of the same colour at a vertex. There are more subtle conflicts particularly involving the non-DAG edges, as illustrated in Figure 3.

Our approach to this problem is a depth-first branch-and-bound search, **find_cycle_decomp**, in the environment of continually updated partial orders for each genome. The strategy is to build cycles one at a time.

Initially all edges in the DG for each genome are 'eligible' and all vertices are 'unused'. We arbitrarily choose an initiating vertex u in the bicoloured graph as well as an edge ε incident to it leading to another vertex v. All remaining edges of the same colour, incident to either u or v, immediately become ineligible. For certain choices of ε (detailed below), the partial order must be updated through the addition of the order relation represented by ε, plus all others involving one vertex ordered before u and one ordered after v.

At each successive stage of the search we add an eligible edge ε that does not conflict with the current partial order, incident to the most recently included vertex u to extend the current cycle or path to some as yet unused vertex v or, preferably, to complete a cycle.

When an edge ε is added, the partial order for the genosome containing ε is updated, if necessary, including whenever ε is not a DAG edge. If ε is in the DAG, no update is necessary

(since the initial partial orders for the branch and bound are determined by the DAGs), unless u or v is incident to more than one eligible edge of the same colour as ε, in which case additional order is imposed by the choice of ε. All remaining edges of the same colour incident to u or v are made ineligible and u is now 'used'.

When a cycle is completed, the initiating vertex also becomes 'used'. Any unused vertex can then be chosen to initiate a new cycle.

The search is bounded by using the fact that a cycle has at least two edges, and that a complete solution, representing some linearization, optimal or not, always has $2n+2$ edges. Suppose the current best solution has c^* cycles. Suppose, further, the construction now underway is at a point where there are c' cycles, and this has used m edges. This means there are only $2n+2-m$ edges left to choose from. Then the final number of cycles when the current construction is terminated will be no more than $c'+(2n+2-m)/2 = c'+n+1-m/2$. So if

$$c' + n + 1 - m/2 < c^*, \tag{2}$$

this branch of the search is abandoned and backtracking begins.

Backtracking is also invoked if no cycles can be made up of the unused vertices. During backtracking, when an edge is removed, so are the extra partial order relations it induced, as well as any 'eligible' and 'unused' status it annulled in edges and vertices, respectively.

An initial value of c^* can be found using any linearizations of the two DAGs or simply by running the depth-first algorithm until a first complete decomposition of the bicoloured graph is found.

We have implemented and tested our algorithm for moderately sized examples as a proof of principle for the synergy of linearization and reversal inference. We note that, aside from computing the time reduction due to the application of the bound in Equation (2), as the search proceeds deeper into the search tree, large numbers of competing edges are excluded, each time an edge is selected, both because vertices are being used up and because of conflicts involving non-DAG edges, such as that in Figure 3.

4.3 Summary of the analysis

Figure 4 summarizes the steps in our analysis, starting from several sets of incomplete maps for each of two genomes, and outputting two totally ordered maps related by a minimal reversal scenario.

4.4 Complexity issues

The major time and space costs of our method are of course due to the branch and bound procedure in **find_cycle_decomp**. The number of potential edges to be considered for inclusion in the decomposition can grow as $O(n^2O^2)$, where S is the maximum number of parallel paths through the DAGs, but the depth of our search tree remains $O(n)$. The costs at each step are dominated by the necessity of checking and updating a partial order matrix of size $O(n^2)$.

Input: One or more incomplete maps for each of 2 genomes
Remove: Markers missing from all maps for either genome
For each map,
    make_po
For each genome,
    (resolve)
    combine_po
    dagger
    dgger
(Add signs to markers)
make_bicoloured
find_cycle_decomp
Output: Optimal cycles and linearizations

Fig. 4. Summary of steps in simultansous linearization and reversal inference. Steps in parentheses—see Sections 2 and 6—not required for signed maps with no outright intra-genome conflicts.

5 SIMULATED DATA

In Section 6 we will apply our algorithm to real genomes. Here, we simulate DAGs representing varying levels of uncertainty, modeling lack of resolution of gene order, and missing genes in two datasets.

For a totally ordered genome containing n genes, we simulate m datasets based on two parameters p and q, representing the probability that any two adjacent genes are not resolved in a data set (i.e. are mapped to the same position) and the probability that any particular gene is deleted (i.e. does not show up on the map), respectively.

In any particular dataset, the genes that are not deleted are ordered as in the underlying genome, except when two or more genes are mapped to the same position. These are considered to have no order among them.

The second genome is derived from the first by a series of r random reversals. Then m datasets are constructed as for the original genome.

The information contained in all m datasets for each genome is then fed into the analysis of the previous section.

We carried out simulations with $n = 12$ and $r = 3$ reversals. We tested $p = 1/3$ and $p = 2/3$ and the same two values for q. We also tried $m = 2$ and $m = 4$. The goal was to see how much uncertainty could be introduced into the system without losing the ability to infer the 3 reversals.

In five runs with each combination of p, q and m, the program recovered 3 reversals in a majority of runs, except where both p and q are 2/3. Both for $m = 2$ and $m = 4$, enough uncertainty was introduced so that linearizations compatible with only 2 or 1 reversals were generated.

Thus, except with very high rates of missing and superimposed genes, enough information is retained in the incomplete

maps that our method can usually recover the correct reversal history intervening between the two genomes.

6 APPLICATION TO CEREAL GENOMES

To illustrate the application of our method to real data, we choose an example that is non-trivial but that is small enough to verify visually. The Gramene database, http://www.gramene.org/, contains a variety of maps of the rice, maize, oats and other cereal genomes. We examined two datasets each for the relatively closely related corn and sorghum genomes: the 'IBM neighbors' and the 'IBM2 neighbors 2003' maps for chromosome 3 of maize (Polacco and Coe, 2002, http://www.maizegdb.org/ancillary/IBMneighbors.html) and the 'Paterson 2003' and 'Klein 2004' (Bowers *et al*., 2003; Menz *et al*., 2002) maps of the chromosome labeled A and LG-03, respectively, of sorghum. (We chose the two versions of 'IBM neighbors' for illustrative purposes, to obtain a large enough set of markers homologous with sorghum markers, and containing typical kinds of differences between datasets for a single genome, despite their being drawn from successive updates of the same database.)

We extracted all markers indicated as having a maize–sorghum homologous pair involving at least one of the datasets from each species.

The 24 markers, their 'IBM2 neighbors 2003' names and their orders in the four databases are listed in Table 1.

The two DAGs constructed from the maize sets and the soghum sets by the routines **combine_po** and **dagger** are depicted in Figure 5.

Our construction did not include the marker umc103 (our number 16), which, in sorghum, may be orthologous to an occurrence on chromosome 8 of maize rather than the one on chromosome 3.

In addition, our construction of the sorghum DAG reflects the resolution of a conflict between the two datasets involving umc5 (our marker 5) versus rz244 and cdo 920 (markers 4 and 24). The same marker is also in conflict with rz995 (marker 6) in the maize databases and this uncertainty has been incorporated into the maize DAG.

The DAG edges for maize and sorghum combined with the edges representing all unordered pairs of markers (11 for maize and 31 for sorghum) to form DGs were combined to make a bicoloured graph for rearrangement analysis.

The efficient algorithms for genome comparison, starting with the Hannenhalli–Pevzner algorithm, require the orientation, or strandedness, of the genes or markers, and not only their order. More specifically, the information required is which markers are on different strands on the two genomes and which have not changed strand from one genome to the other. The experimental work underlying traditional genetic maps, such as those we have used here, generally cannot specify strandedness, so the usual solution in computational genomics is to assign polarity so as to minimize the rearrangement cost,

Table 1. (Vertical) Orders of markers appearing in at least one maize dataset and one sorghum dataset, numbered according to the 'IBM2 neighbors 2003' dataset

IBM2		IBM	Paterson	Klein
umc121	1			
rz543	2	1		
csu621	3	2	16	
rz244	4	2	2	
umc10	5		3	
rz995	6	(6,4)	6	
cdo1160	7	5		1
sps2	8	7	(23,4)	5
csu776	9	8		4
rz444		9	(5,24)	23
csu351	(10,11)			22
csu690	12	(10,11)	(22,21)	7
bcd738	13	12	20	18
bnl15.20	14		19	17
csu706	15	(14,13)	7	14
umc103		15	15	13
umc17	(16,17,18)	17	12	
bcd828			11	
csu744		(19,20)	10	
csu456	(19,20)		9	
csu397		(21,22)	8	
umc63	(21,22)	23		
cdo920	23	24		
lhcb1	24			

Parentheses group markers mapped to the identical chromosomal position. Horizontal alignment of markers in different datasets is arbitrary and has no significance.

cf. http://www.cse.ucsd.edu/groups/bioinformatics/GRIMM/ and Tesler (2002a).

In our example, this turns out to require a negative polarity on sorghum markers 4–6, and 8–24.

Applying the **find_cycle_decomp** algorithm gives a 19-cycle decomposition, implying five (23 markers $+1 - 19$ cycles $= 5$) reversals are necessary to transform a linearization of the sorghum chromosome into a linearization of the maize chromosome. The two linearizations in the output are for maize and sorghum, respectively:

1 2 3 4 6 5 7 8 9 10 11 12 13 14 15 17 18 19 20 21 22 23 24
1 2 3 6 4 23 24 5 22 21 20 19 7 18 17 15 14 13 12 11 10 9 8

where the negative polarity items in sorghum are underlined. Five reversals that convert the sorghum order into the maize order are indicated by boldface in the following sequence:

1 2 3 **6 4** 23 24 5 22 21 20 19 7 18 17 15 14 13 12 11 10 9 8
1 2 3 4 6 **23 24** 5 22 21 20 19 7 18 17 15 14 13 12 11 10 9 8
1 2 3 4 6 5 **24 23 22 21 20 19 7 18 17 15 14 13 12 11 10 9 8**
1 2 3 4 6 5 **8 9 10 11 12 13 14 15 17 18 7** 19 20 21 22 23 24
1 2 3 4 6 5 7 **18 17 15 14 13 12 11 19 9 8** 19 20 21 22 23 24
1 2 3 4 6 5 7 8 9 10 11 12 13 14 15 17 18 19 20 21 22 23 24

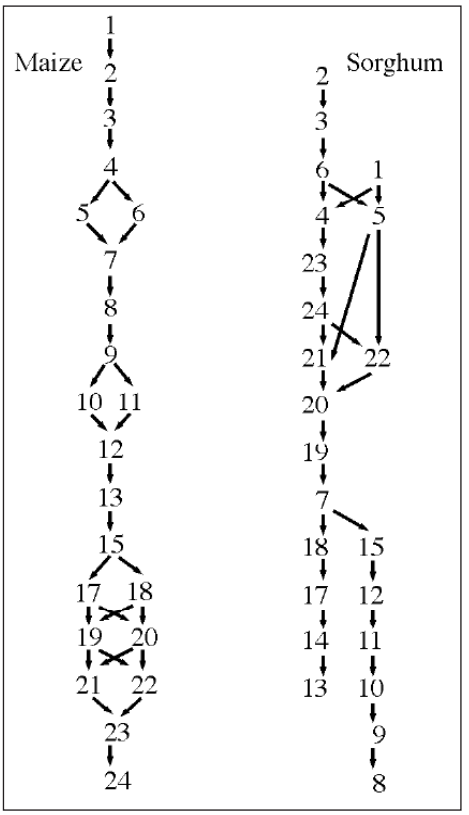

Fig. 5. DAGS constructed from by combining the partial orders of two maize datasets and two sorghum datases presented in Table 1.

Note that two of these reversals are necessary only to account for the apparent movement of marker 7 in the chromosome.

Our analysis has, thus, taken the rather incomplete data in the two datasets for each chromosome, as represented in the DAG, providing a plausible linear order for all the markers in both genomes, while simultaneously inferring the most economical sequence of evolutionary rearrangements to account for the gene order differences.

7 CONCLUSIONS

In this paper we have proposed a more realistic version of the genome comparison problem, where the gene order or marker order in the two genomes is incompletely specified. This is not a special case, but the general case for genomic maps of all higher organisms.

When diverse maps of the same genome are combined to produce a more detailed map, the main sources of uncertainty in the order are generally not the conflicts between the original maps, but simply the different gene or marker content of the original maps and the lack of resolution in them.

We have shown that the appropriate way to combine maps of this sort, so as to retain all the order information in each, and without imposing any additional structure, is through the DAG representation.

We locate the problem of inferring rearrangements for two genomes in DAG form in the selection of edges in the generalized Hannenhalli–Pevzner bicoloured graph defined by these DAGs, so as to obtain a maximal decomposition of the vertices into alternating cycles.

To find the most economical rearrangement scenario, which is the solution to this problem, our algorithm simultaneously finds the optimal topological sorting of both genome DAGs into totally ordered form. This is a rigorous way of using comparative data to help order the genes or markers on a chromosome.

We have implemented our algorithm so that it works for small but non-trivial data, exemplified by a chromosomal comparison between maize and sorghum. However, there are many ways to improve the efficiency of this method so that it will work with much larger n. This is the current priority for work on this project.

At the same time, we are incorporating the full Hannenhalli–Pevzner equation to minimize the number of reversals, rather than just maximizing the number of cycles.

We have extended our analysis (Zheng and Sankoff, 2005) to handle translocations and reversals as in Hannenhalli and Pevzner (1995) and Tesler (2002b). There is little conceptual difference in formulating our problem for the reversals only case and for the reversals and translocations case.

In further research (Sankoff *et al.*, 2005), it will be necessary to implement a method to automatically assign polarity. Our method is most appropriate for genomes with relatively little genomic sequencing, and most frequently with maps that do not specify strandedness.

We point out that in both the reversals context and the translocations plus reversals context, once a method for optimizing the cycle decomposition is perfected, there is no additional work required to find a way to identify the actual rearrangements. Since finding the cycles also gives us linearizations, these linear genomes can be used in existing methods, such as the GRIMM server (Tesler, 2002a).

ACKNOWLEDGEMENTS

Research supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). D.S. holds the Canada Research Chair in Mathematical Genomics and is a Fellow of the Evolutionary Biology Program of the Canadian Institute for Advanced Research.

REFERENCES

Bowers,J.E., Abbey,C., Anderson,S., Chang,C., Draye,X., Hoppe,A.H., Jessup,R., Lemke,C., Lennington,J., Li,Z. *et al.* (2003) A high-density genetic recombination map of sequence tagged sites for sorghum, as a framework for comparative

structural and evolutionary genomics of tropical grains and grasses. *Genetics*, **165**, 367–386.

Hannenhalli,S. and Pevzner,P.A. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, pp. 581–592.

Hannenhalli,S. and Pevzner,P.A. (1999) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *J. ACM*, **48**, 1–27.

Menz,M.A., Klein,R.R., Mullet,J.E., Obert,J.A., Unruh,N.C. and Klein,P.E. (2002) A high-density genetic map of Sorghum bicolor (L.) Moench based on 2926 AFLP, RFLP and SSR markers. *Plant Mol. Biol.*, **48**, 483–499.

Polacco,M.L. and Coe,E., Jr (2002) IBM neighbors: a consensus GeneticMap.

Sankoff,D., Zheng,C. and Lenert,A. (2005) Reversals of fortune. Technical Report, Laboratory for Innovation in Bioinformatics, University of Ottawa.

Tesler,G. (2002a) GRIMM: genome rearrangements web server. *Bioinformatics*, **18**, 492–493.

Tesler,G. (2002b). Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, **65**, 587–609.

Ware,D., Jaiswal,P., Ni,J., Pan,X., Chang,K., Clark,K., Teytelman,L., Schmidt,S., Zhao,W., Cartinhour,S., McCouch,S. and Stein,L. (2002) Gramene: a resource for comparative grass genomics. *Nucleic Acids Res.*, **30**, 103–105.

Zheng,C. and Sankoff,D. (2005) Genome rearrangements with partially ordered chromosomes. In Wang,L. (ed.) *Lecture Notes in Computer Science, COCOON 2005*, Spring, in press.