J Comb Optim (2006) 11: 133–144
DOI 10.1007/s10878-006-7120-2

Genome rearrangements with partially ordered chromosomes

Chunfang Zheng · David Sankoff

Received: 25 September 2005 / Accepted: 25 December 2005

Abstract Genomic maps often do not specify the order within some groups of two or more markers. The synthesis of a master map from several sources introduces additional order ambiguity due to markers missing from some sources. We represent each chromosome as a partial order, summarized by a directed acyclic graph (DAG), to account for poor resolution and missing data. The genome rearrangement problem is then to infer a minimum number of translocations and reversals for transforming a set of linearizations, one for each chromosomal DAG in the genome of one species, to linearizations of the DAGs of another species. We augment each DAG to a directed graph (DG) in which all possible linearizations are embedded. The chromosomal DGs representing two genomes are combined to produce a single bicoloured graph. From this we extract a maximal decomposition into alternating coloured cycles, determining an optimal sequence of rearrangements. We test this approach on simulated partially ordered genomes and on marker data from maize and sorghum chromosomal maps.

1. Introduction

Genome rearrangement algorithms, which require the genes or markers on a chromosome to be totally ordered, cannot apply directly to comparative maps where some order information is lacking. In this article we develop a strategy to adapt the algorithms to such maps.

1.1. Formalizing genomes and their evolutionary mechanisms

The genetic structure of a genome can be modeled as a set $\chi = \{\chi_1, \cdots, \chi_k\}$ of $k \geq 1$ chromosomes, where each chromosome χ_i consists of $n_i > 0$ genes or other genetic markers,

C. Zheng (✉)
Department of Biology
e-mail: czhen033@uottawa.ca

D. Sankoff
Department of Mathematics and Statistics, University of Ottawa, Canada K1N 6N5
e-mail: sankoff@uottawa.ca

signed and totally ordered: e.g., $g_1 < \cdots < g_{n_i}$, also written simply as $g_1 \cdots g_{n_i}$, where each g is a positive or negative integer, and where each g appears only once in all of χ. Without loss of generality, we may assume each integer between 1 and $n = n_1 + \cdots + n_k$ appears exactly once in χ, either with a plus or minus sign. n is the size of the genome. An example of a genome of size 8 is thus $\{\chi_1, \chi_2, \chi_3\}$, where $\chi_1 = 5\ -1\ -4$, $\chi_2 = 7\ 8\ 3$, and $\chi_3 = -2\ 6$. Henceforward we will use the term gene to refer to either genes or genetic markers of any kind. The order relation abstracts the position of the gene on the linear chromosome, and the sign carries information about which of the two DNA strands the gene is located.

Genome evolution can be modeled by the transformation of a genome χ of size n to a genome ψ of the same size, by means of reversals within a chromosome and translocations between chromosomes. A reversal transforms any contiguous part of an order to its reverse order, changing the polarity of all genes in its scope, e.g., $g\ h\ i\ j\ k \rightarrow g - j - i - h\ k$ is a reversal of the "segment" $h\ i\ j$. A (reciprocal) translocation exchanges any prefixes of two chromosomes (or equivalently, any suffixes of two chromosomes) or the reversed prefix of one with the reversed suffix of the other, for example $g\ h\ i, x\ y\ z \rightarrow g\ h\ y\ z, x\ i; g\ h\ i, x\ y\ z \rightarrow g - x, -i - h\ y\ z$. There are two special cases: $g\ h\ i, x\ y\ z \rightarrow g\ h\ i\ x\ y\ z; g\ h\ i\ x\ y\ z \rightarrow g\ h\ i, x\ y\ z$ where a null prefix of one chromosome $x\ y\ z$ is exchanged with the largest prefix of the other $g\ h\ i$ (chromosome fusion) and where a null chromosome translocates with $g\ h\ i\ x\ y\ z$ (chromosome fission), respectively. To make biological sense, since a chromosome does not change its nature when it is moved around in space, a reversal of an entire chromosome is considered to leave it unchanged: $g_1 \cdots g_{n_i} = -g_{n_i} \cdots - g_1$.

The Hannenhalli-Pevzner algorithms (Hannenhalli and Pevzner, 1995, 1999) for comparing genomes, and their improvements (e.g., Bader et al., 2001; Tesler, 2002), infer $d(\chi, \psi)$, the smallest number of reversals and translocations necessary to transform genome χ into ψ, as well as a sequence of such operations that actually achieves this minimum.

1.2. Partially ordered genomes

The representation of a genome as a set of totally ordered chromosomes must often be weakened in the case of real data, where mapping information only suffices to partially order the set of genes on a chromosome. The concepts and methods of genome rearrangement, however, pertain only to totally ordered sets of genes or markers, and are meaningless in the context of partial orders.

Our approach is to extend genome rearrangement theory to the more general context where all the chromosomes are directed acyclic graphs (DAGs) rather than total orders (Zheng et al., 2005). 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 chromosomal DAGs for each species from two or more incomplete data sets, or from a single low-resolution data set. The frequent lack of order information in each data set, due to missing genes or missing order information, is converted into parallel subpaths within each chromosomal DAG in a straightforward manner.

Outright conflicts of order create cycles that must be broken to preserve a DAG structure. 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 rearrangement problem is then TO INFER A TRANSFORMATION SEQUENCE (TRANSLOCATIONS AND/OR REVERSALS) FOR TRANSFORMING A SET OF LINEARIZATIONS (TOPOLOGICAL SORTS), ONE FOR EACH CHROMOSOMAL DAG IN THE GENOME OF ONE SPECIES, TO A SET OF LINEARIZATIONS OF THE CHROMOSOMAL DAGS IN THE GENOME OF ANOTHER SPECIES, MINIMIZING THE NUMBER OF TRANSLOCATIONS AND REVERSALS REQUIRED. To do this, we embed the set of all possible linearizations in each DAG by appropriately augmenting the edge set, so that it becomes a general directed graph (DG). We combine the two sets of chromosomal 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 practically equivalent to optimally linearizing the partial orders, so that finding the rearrangements themselves can be done using the previously available algorithms.[1]

2. Gene order data

2.1. The methodological origins of incomplete maps

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 some pairs of neighbouring genes with respect to each other. Even at the ultimate level of resolution, that of genome sequences, the application of different gene-finding protocols usually gives maps with different 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 of 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).

2.2. Simulating incomplete maps of pairs of two related genomes

How incomplete maps arise may perhaps be best understood through a description of how we simulate them.

[1] To be precise, the optimal linearization should minimize an expression of form $n - c + \theta$, where n is the number of genes, a constant, c is the number of "good" structures in the graph decomposition, including alternating coloured cycles and certain paths, and θ is a generally small correction but one that requires a complicated calculation. Indeed θ is often ignored in practice. Moreover, under the biologically most satisfying version of the genome rearrangement problem (Yancopoulos et al., 2005), which adds generalized transpositions, or movements of segments (possibly cyclically rotated) of chromosomes from one site to another in the genome, to the repertoire of rearrangement operations, where such transpositions count as two rearrangement events, it can be shown that $\theta \equiv 0$.

2.2.1. Simulating the genomes

For a given n, we pick a small integer k, as well as positive $n_1, \cdots, n_k$ with the constraint $n = n_1 + \cdots + n_k$. Then we define

$$\chi = \{m_1 \cdots n_1, m_2 \cdots n_1 + n_2, \cdots, m_k \cdots n\}, \tag{1}$$

where $m_1 = 1$ and the remaining $m_i = m_{i-1} + n_{i-1}$. It is well known that the genes in two genomes being compared through translocation and reversal distance may always be relabeled in so that in one of the genomes they have they all have positive sign and have the form in (1).

To obtain the second genome ψ, we perform r reversals distributed at random among the k chromosomes of χ, reversing randomly chosen segments, and t reciprocal translocations between random pairs of chromosomes, exchanging randomly sized prefixes or suffixes, plus f_1 fusions and f_2 fissions. The operations are performed in random order, each one applied to the transformed genome produced by the preceding operation.

The non-negative parameters $n, k, n_1, \cdots, n_k, r, t, f_1$ and f_2 are specified in advance, as is the random choice procedure, depending on the kind of genomes we wish to model and compare. For $n = 25, k = 4, n_1 = 7, n_2 = 8, n_3 = 5, n_4 = 5, r = 4, t = 4, f_1 = 0$ and $f_2 = 1$, (2) shows genome χ and an example of ψ.

```
    {1  2  3  4  5  6  7,                      {-17 -3  -2  -1  18 -5 -4 -16,
χ =  8  9 10 11 12 13 14 15,            ψ =      8 -13 -12 -11 -22 -21,
    16 17 18 19 20,                             -7  -20  10 -24 -23  25,
    21 22 23 24 25}                              14  15,
                                                  9 -19   6}
```
(2)

2.2.2. Simulating the maps

Once we have obtained simulated genomes χ and ψ, we fix probabilities p_{missing} and p_{group}, and numbers of data sets N_χ and N_ψ.

For each of the two genomes we construct each of the N_χ or N_ψ data sets independently, as follows. For each chromosome, each gene on the chromosome except the last one is submitted to a grouping event with probability p_{group}, which determines whether or not the gene position can be distinguished from that of the next gene. Then each gene is submitted independently to a deletion event with probability p_{missing}, conditioned on the event that the gene cannot be deleted from all the datasets. Note that if a gene g_1 is grouped with the next gene g_2, which is subsequently deleted, and if g_2 was not itself grouped with the next gene g_3, then g_1 is not grouped with g_3 in the data set.

Note that this procedure cannot produce two data sets on the same genome with conflicting order relations ($a < b$ in one, $b < a$ in the other), nor with the same gene on two different chromosomes. Three data sets produced from the genomes in (2) are shown in (3), with boxes around unresolved groups of genes.

3. Constructing the chromosomal DAGs

A linear map of a chromosome that has several genes or markers at the same position π, because their order has not been resolved, can be reformulated as a partial order, where all the genes before π are ordered before all the genes at π and all the genes at π are ordered

$$
\chi = \begin{array}{l}
\{\ 1\ \ 2\ \ \boxed{4\ 5}\ \ 7, \\
\ \ 8\ 10\ 11\ 15, \\
\ 16\ 17\ 20, \\
\ 22\ 23\ 25\}
\end{array}
\qquad
\begin{array}{l}
\{\ \boxed{2\ 3}\ \ 6\ \ 7, \\
\ \ 8\ \ 9\ 12\ 14, \\
\ 17\ 18\ 19, \\
\ 21\ 24\ 25\}
\end{array}
\qquad
\begin{array}{l}
\{\ 2\ \ 4\ \ 6, \\
\ 9\ 11\ \boxed{12\ 13}\ \boxed{14\ 15}, \\
\boxed{16\ 18}\ 20, \\
\ \ \ \boxed{21\ 23}\ 24\}
\end{array}
\tag{3}
$$

$$
\psi = \begin{array}{l}
\{\ -17\ \ -2\ \ 18\ \ -4, \\
\boxed{-13\ -12\ -11}\ \boxed{-22\ -21}, \\
\ \ -7\ \boxed{-20\ \ 10\ \ -24\ -23}\ \ 25, \\
\ \ 14\ \ 15, \\
\ \ \ 9\ -19\}
\end{array}
\qquad
\begin{array}{l}
\{\boxed{-17\ \ -3\ \ -2}\ \ -1\ \ \boxed{-5\ \ -4\ -16}, \\
\ \ \boxed{8\ -13}\ \boxed{-12\ \ -11\ -22}, \\
\ \boxed{-7\ -20\ \ 10}\ \boxed{-24\ \ 25}, \\
\ \ 14\ \ 15, \\
-19\ \ \ 6\}
\end{array}
\qquad
\begin{array}{l}
\{\ \boxed{-2\ \ 18}\ \ -5\ \ -16, \\
\ \ \ 8\ \boxed{-13\ \ -11}\ \ -21, \\
-20\ \ 10\ \ -24\ \ -23, \\
\ \ 14\ \ 15, \\
\ \ \ \boxed{9\ \ 6}\}
\end{array}
$$

before all the genes following π, but the genes at π are not ordered amongst themselves. We call this procedure **make_po**.

For genomes with 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 chromosome containing different subsets of genes. Assuming there are no conflicting order relations ($a < b, b < a$) nor conflicting assignments of genes to chromosomes among the data sets (as in the data sets on our simulated genomes), for each chromosome we simply take the union of the partial orders, and extend this set through transitivity. This procedure is **combine_po**.

All the partial order data on a chromosome can be represented in a minimal DAG whose vertex set is the union of all gene sets on that chromosome in the contributing data sets, 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.

In real applications, different maps of the same genome do occasionally conflict, either because $b < a$ in one data set while $a < b$ in the other or because a gene is assigned to different chromosomes in the two data sets. There are a variety of possible ways of resolving order conflicts or, equivalently, of avoiding any cycles in the construction of the DAG. One way is to delete all order relations that conflict with at least one other order relation. 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. The latter method also resolves conflicts due to gene assignment to different chromosomes. Any

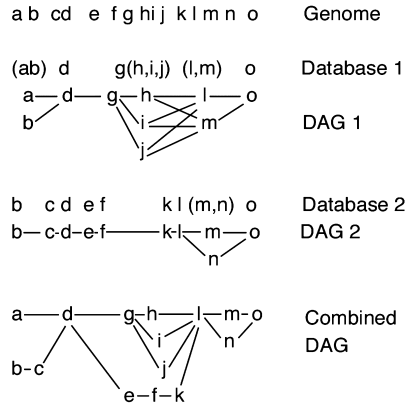

Fig. 1 Construction of DAGs from individual databases each containing partial information on genome, due to missing genes and missing order information, followed by construction of combined DAG representing all known information on the genome. All edges directed from left to right

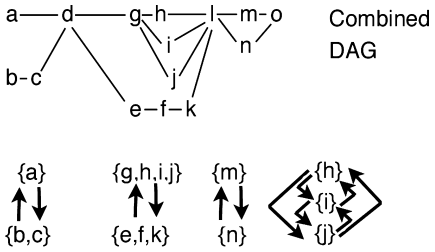

Fig. 2 Edges added to DAG to obtain DG containing all linearization as paths (though not all paths in the DG are linearizations of the DAG!). Each arrow represents a set of directed edges, one from each element in one set to each element of the other set

of these approaches, or others, which we denote by the generic routine name **resolve**, will produce results appropriate for our subsequent analysis.

4. 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 directed graph (DG) containing all the edges of the DAG, plus two edges of opposite directions connecting each pair of vertices not ordered by the DAG. This is illustrated in Figure 2. The routine for constructing this graph is **dgger**.

5. The algorithm

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

Hannenhalli and Pevzner (1995) showed how to find a shortest sequence of reversals and translocations that transform one genome χ with n genes on k chromosomes into another genome ψ of the same size but with h chromosomes, in polynomial time. As described in Tesler (2002), this construction begins by combining the signed order representations of all the chromosomes in the two genomes. The following procedure, **make_bicoloured**, produces a bicoloured graph on $2n + 2k$ vertices that decomposes uniquely into a set of alternating-coloured cycles and set of $h + k$ alternating-colour paths. First, each gene or marker x in χ determines two vertices, x_t and x_h, to which two additional dummy vertices e_{i_1} and e_{i_2} are added to the ends of each chromosome χ_i. Edges of one of the colours, say red, are determined by the adjacencies in χ. If x is the left-hand neighbour of y in χ, and both have positive polarity, then x_h is connected by a red edge to y_t. If they both have negative polarity, it is x_t that is joined to y_h. If x is positive and y negative, or x is negative and y positive, x_h is joined to y_h, or x_t is joined to y_t, respectively. If x is the first gene in χ_i, then e_{i_1} is joined to x_t or x_h depending on whether x has positive or negative polarity, respectively. If x is the last gene, then e_{i_2} is joined to x_t or x_h depending on whether x is negative or positive.

Black edges are added according to the same rules, based on the adjacencies in genome ψ, though no dummy vertices are added in this genome.

It can be seen that each vertex is incident to exactly one red edge and one black edge, except for the dummy vertices in χ, which are each incident to only a red edge, plus the two (non-dummy) vertices at the ends of each chromosome in ψ, which are also each incident only to a red edge. The bicoloured graph decomposes uniquely into a number of alternating cycles plus $h + k$ alternating paths terminating in either the dummy vertices of χ or the end vertices of ψ, or one of each. Suppose the number of these paths that terminate in at least one dummy vertex (good paths) is $j \leq h + k$. If the number of cycles is c, then the minimum number of reversals r and translocations t necessary to convert χ into ψ is given by the Hannenhalli-Pevzner equation:

$$r + t = n + k - j - c + \theta \tag{4}$$

where θ is a correction term that is usually zero for simulated or empirical data. For simplicity of exposition, we ignore this correction here, though an eventual full-scale program will incorporate it with little or no computational cost.

5.2. Generalization to partial orders

The routine **make_bicoloured** can also be applied to the set of edges in the DGs for two partially ordered genomes. In the resulting graph, each of the DAG edges and both of the edges connecting each of the unordered pairs in the DG for each chromosome represent potential adjacencies in our eventual linearization of a genome. The n genes or markers and $2k$ dummies determine $2n + 2k$ 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 $n + k$ red edges and $n - h$ black edges, in our construction in the presence of uncertainty there are more potential edges of each colour, but only $2n + k - h$ can be chosen in our construction of the cycle graph, which equivalent to the simultaneous linearization by topological sorting of each chromosome in 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 incident to a single 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 $h + k$ continually updated partial orders, one for each chromosome in each genome. The strategy is to build cycles and paths one at a time.

Initially all edges in the DG for each chromosome are "eligible" and all vertices are"unused". We choose any initiating vertex u in the bicoloured graph and an edge ϵ connecting it to another vertex v. All remaining edges of the same colour incident to either u or v then become ineligible. For certain choices of ϵ, the partial order associated with that chromosome must be updated through the addition of the order relation

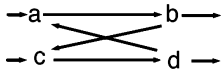

Fig. 3 If ab and cd are DAG edges, then the two non-DAG edges da and bc are mutually exclusive, since using them both leads to the wrong order for a and b

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 close a cycle or complete a cycle or path. A complication is that when the construction reaches a potential end vertex in a chromosome of ψ, it is not always clearly the termination of a path since the DAG may contain several competing end vertices. (This is not a problem with the chromosomes in χ because the dummies e_{i_1} and e_{i_2} are always at the ends of the chromosomes.) In this case, the choice of path termination or not becomes one of the branches to explore in the branch and bound.

When an edge ϵ is added, the partial order for the chromosome 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 path 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". When a path is complete, both the initiating and terminating vertices become "used". Any unused vertex can then be chosen as to initiate a new cycle or path. (In our implementation, we increase the efficiency of the search by choosing a dummy or an end vertex whenever possible, including the very first choice of u.)

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 $2n + k - h$ edges. Suppose the current best solution has c^* cycles and (necessarily) $h + k$ paths of which j^* are "good" as in (4). Suppose further the construction now underway is at a point where there are c' cycles and l paths, with j' good ones, and this has used m edges. This means there are only $2n + k - h - m$ edges left to chose of which at least $h + k - l$ must be in paths. Then the final number of cycles when the current construction is terminated will be no more than $c' + (2n + k - h - m - h - k + l)/2 = c' + n - h - (m - l)/2$. The final number of "good" paths will be no more than $j' + h + k - l$. So if

$$\begin{aligned} c' + n - h - (m - l)/2 + j' + h + k - l &= c' + j' + n + k - (m + l)/2 \\ &< c^* + j^*, \end{aligned} \tag{5}$$

this branch of the search is abandoned and backtracking begins.

Backtracking is also invoked if no cycles or paths can be made up of the unused vertices. During backtracking, when an edge is removed, so are the extra partial order relations it induced. The "eligible" and "unused" status it annulled are restored. An initial value of c^* can be found using any linearizations of the chromosomal DAGs of the two genomes or simply by running the depth-first algorithm until a first complete decomposition of the bicoloured graph is found.

5.2.1. A speed-up for the case of one totally ordered genome

In some comparisons, particularly where the genes in each chromosome in one of the genomes, say χ, are totally ordered, and where the amount of uncertainty is small with respect to n in the other genome ψ, the method can be speeded up considerably by a rapid

preprocessing stage and minor modifications to the algorithm. The idea is to identify the large majority of DAG edges that are necessarily in the linearization of the DG for ψ, and to incorporate them at the outset into a modified version of **find_cycle_decomp**. For example, if uv and vw are the only two edges incident to vertex v, and all other vertices in the DAG are ordered with respect to v, both of uv and vw must be included in the linearization. Moreover we can simplify the appropriate partial order by removing all order relations involving v. The revised algorithm incorporates all such edges as a first step, followed by a branch and bound search for the remaining edges in the cycle decomposition. This not only has the advantage of reducing the number of steps in the search, but it also greatly reduces the costly task of updating a partial order every time certain edges are tentatively included.

In initializing the branch and bound with several edges, not necessarily forming any complete cycles or paths, the algorithm must be modified to take account of several potential cycles and paths at any point in time, instead of just one at a time. After the initialization, grouping all the remaining edges in p classes, each class determined by the "starting" vertex of the edge, we observe that exactly one edge from each class must appear in a solution. Each time we try an edge, we must update the partial order of the chromosome containing its vertices.

In this approach, we use a different upper bound on the number of cycles and good paths that can be constructed in the remaining steps of the search. If edges from p' of the p classes are already in the solution under construction, this bound is the sum of $p - p'$ quantities, one for each remaining class. The p quantities $z_1, \cdots, z_p$ are calculated only once, before the branch and bound part of the construction. Obviously, at each step of the branch and bound, $p - p'$ can be added on to the existing number of cycles and good paths to provide an upper bound, i.e., $z_i = 1$ for each i, since exactly one edge from each class must be in a solution, and this edge can be in at most one cycle or good path. But an exhaustive search for cycles or good paths starting from all trial edges in the i-th class is sometimes feasible (it suffices to consider adding edges only from the 1-st,$\cdots, i-1$-st classes, as well as some of the initializing edges), and if this search terminates without finding one, we can set $z_i = 0$ instead of $z_i = 1$, providing tighter upper bounds.

Note that in contrast with the one-cycle-at-a-time version of the algorithm, when we add a trial edge in the present version, we must check to see if it combines with one or two existing paths, possibly completing a cycle or a good path, or simply inaugurates a new path in the partial solution.

6. Summary of the analysis

The steps in our analysis, starting from several sets of incomplete chromosomal orders for each of two genomes, and outputting two genomes with totally ordered chromosomes, as well as a minimum number of reversals and translocations necessary to convert one to the other, are as follows.

Input: A number of incomplete maps for each genome
Remove: Genes or markers that do not appear in at least one map for each genome
For each chromosome in each map,
    make_po
For each genome,
    resolve
For each chromosome,
    combine_po
    dagger

dgger
make_bicoloured
find_cycle_decomp
Output: Optimal cycles, paths and linearizations

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 $O(n^2)$, 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 $O(n^2/h^2)$, assuming $h = k$, and all chromosomes are about the same size.

In the speed-up, the number of potential edges to be considered for inclusion in the decomposition is only is $O(p^2)$, since all the edges in one of the genomes are obligatorily included. The depth of our search tree is $O(p)$. The costs at each step are dominated by the necessity of checking and updating a partial order matrix of size $O(p^2/h^2)$.

7. Analyzing the simulated incomplete data

We submitted the data in (2) to our analysis. In (6) we compare the results to the original genomes in (2).

```
True genomes:
      {1  2   3  4  5  6  7,                {-17 -3 -2 -1  18 -5 -4 -16,
   χ = 8  9  10 11 12 13 14 15,          ψ =   8 -13 -12 -11 -22 -21,
      16 17 18 19 20,                        -7 -20  10 -24 -23  25,
      21 22 23 24 25}                        14  15,
                                              9 -19   6}
Reconstructed genomes:                                                  (6)
      {1  3   2  5  4  6  7,                {-17 -2 -3  -1  18 -4  -5 -16,
   χ = 8  9 10 11 12 13 14 15,           ψ =   8 -13 -12 -11 -21 -22,
      16 17 18 19 20,                        -7 -20  10 -24 -23  25,
      22 21 23 24 25}                        14  15,
                                              9 -19   6}
```

The only reconstruction errors appear to be the reverse order of genes 2 and 3, 4 and 5, and 21 and 22 in both reconstructed genomes. An inspection of the simulated incomplete data in (3), however, shows that no information is present in any of the data sets for either genome that bears on the ordering within any of these pairs.

In addition, the reconstructed linearized chromosomes of χ in (6) require 4 translocations, 1 fission and four inversions to be transformed into the reconstructed ψ, exactly how ψ was originally obtained from χ in (2).

8. Performance

The experimental version of our program can handle moderate size maps. Tests on simulated 6-chromosome maps with 100 genes, of which 10% were missing, and 20% unresolved from each of three datasets for both of the two genomes being compared ($d(\chi, \psi) = 20$), executed in less than a second on a Macintosh G4. With 20% missing and 40% unresolved, an analysis usually required about 3 minutes. Increasing uncertainty beyond this quickly led to run times of several hours.

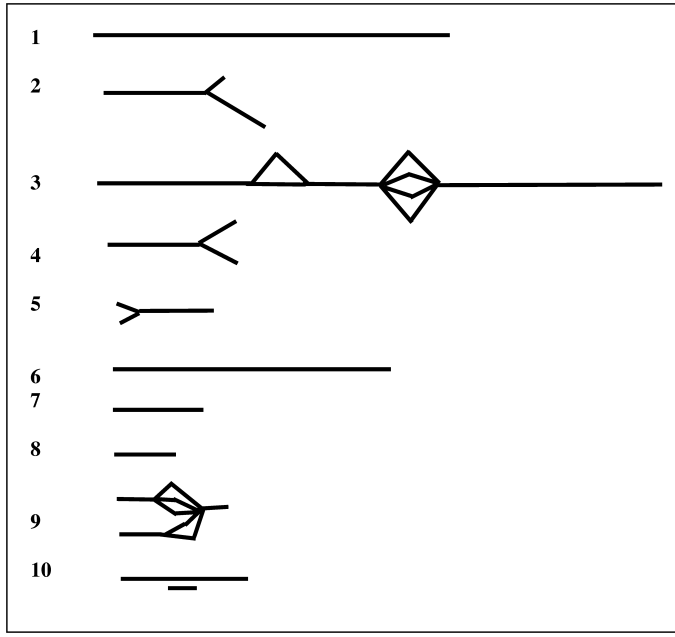

Fig. 4 DAGs for 10 sorghum chromosomes, scaled by number of markers analysed

9. Application to cereal genomes

In Sankoff et al. (2005), we applied the sped-up version of the algorithm to the comparison of the maize and sorghum genomes. We used one set of genomic markers for maize (Polacco and Coe, 2002) and two for sorghum (Menz et al., 2002; Bowers et al., 2003) as accessible in the Gramene database (Ware et al., 2002). We extracted all markers registered as having homologs in maize and at least one of the sorghum datasets. We then discarded likely erroneous markers and all but one copy of any duplicated markers.

This left 191 different markers occurring in both genomes, which could then be input into our exact linearisation algorithm. The DAGs for the sorghum chromosomes are illustrated in Figure 4. The solution involved 6 non-trivial cycles (more than two edges)

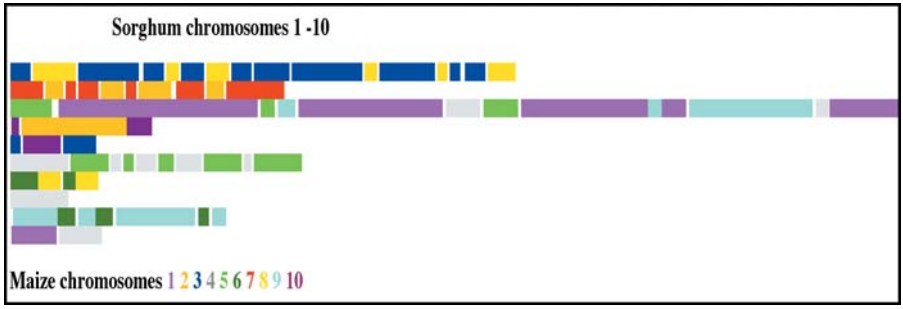

Fig. 5 Conserved segments on sorghum chromosomes, scaled by number of markers analysed

and 20 paths, implying a total of 73 inversions and translocations. Figure 5 portrays the configuration of the conserved segments in the two genomes, disposed on the sorghum chromosomes.

The advantage of the speed-up was clear; from 18 minutes on a Mac G4 for the original algorithm to 10 seconds using the speed-up.

10. Future work

The current implementation of our algorithm is fairly straightforward, and there are a number of promising possibilities for increasing efficiency. Since many comparative maps have only a few hundred genes our method seems quite practical.

As we have shown, the one-sided version of our problem, where the chromosomes of one of the genomes being compared are totally ordered is of great interest. If one genome is known in great detail, we can then resolve many of the uncertainties in less densely mapped species, despite somewhat rearranged genomes, using our technique.

Acknowledgments Research supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). DS 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

Bader DA, Bernard ME, Moret BME, Yan M (2001) A linear-time algorithm for computing inversion distance between signed permutations with an experimental study, Journal of Computational Biology 8: 483–491

Bowers JE, Abbey C, Anderson S, Chang C, Draye X, Hoppe AH, Jessup R, Lemke C, Lennington J, Li Z, Lin YR, Liu SC, Luo L, Marler BS, Ming R, Mitchell SE, Qiang D, Reischmann K, Schulze SR, Skinner DN, Wang YW, Kresovich S, Schertz KF, Paterson AH (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, Pevzner PA (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science pp. 581–592

Hannenhalli S, Pevzner PA (1999) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). Journal of the ACM 48:1–27

Menz MA, Klein RR, Mullet JE, Obert JA, Unruh NC, Klein PE (2002) A high-density genetic map of Sorghum bicolor (L.) Moench based on 2926 AFLP, RFLP and SSR markers. Plant Molecular Biology 48:483–499

Polacco ML, Coe E Jr (2002) IBM Neighbors: A Consensus Genetic Map. (http://www.maizegdb.org/ancillary/IBMneighbors.html)

Sankoff D, Zheng C, Lenert A (2005) Reversals of fortune. Proceedings of the RECOMB 2005 Workshop on Comparative Genomics (A. McLysaght and D. Huson, eds.) Lecture Notes in Bioinformatics 3678, Springer, pp. 131–141

Tesler G (2002) Efficient algorithms for multichromosomal genome rearrangements. Journal of Computer and System Sciences 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, Stein L (2002) Gramene: a resource for comparative grass genomics. Nucleic Acids Research 30:103–105

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

Zheng C, Lenert A, Sankoff D (2005) Reversal distance for partially ordered genomes. Bioinformatics 21:i502–i508.