uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Chunfang Zheng

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Biology)

GRADE / DEGREE

Department of Biology

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Genome Rearrangements Algorithms Applied to Comparative Maps

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. D. Sankoff

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. G. Drouin

Dr. G. Carmody

Dr. S. Aris-Brosou

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Genome Rearrangement Algorithms Applied to Comparative Maps

Chunfang Zheng

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
University of Ottawa
in partial fulfillment of the requirements for the
M. Sc. degree in the

Ottawa-Carleton Institute of biology

Thèse soumise à
Faculté des études supérieures et postdoctorales
Université d'Ottawa
en vue de l'obtention de la maitrise ès sciences

L'Institut de biologie d'Ottawa-Carleton

Your file Votre référence
ISBN: 978-0-494-18487-5
Our file Notre référence
ISBN: 978-0-494-18487-5

Acknowledgements

I would like to thank the following people, without whom this work would not have been possible.

Above all, I thank my advisor Dr. David Sankoff. I am grateful to him for introducing me to the exciting world of genomic rearrangements and providing many opportunities to learn about this world. I very much appreciate his guidance, his teaching and his patience.

I also thank my committee members, Dr. Guy Drouin and Dr. George Carmody for their encouragement and for being available to help me.

Thanks to all the members of the Laboratory for Innovation in Bioinformatics, I have benefited much from their collaboration and helpful discussions. Special thanks to Aleksander Lenert for developing the data sets on maize and sorghum.

Finally, I am grateful to my family for their encouragement, support and patience throughout this work and in my daily life.

Abstract

The Hannenhalli-Pevzner algorithm for computing the evolutionary distance between two genomes is very efficient when the genomes are signed and totally ordered. But in real comparative maps, the data suffer from problems such as coarseness, missing data, no signs, paralogy, order conflicts and mapping noise. In this thesis we have developed a suite of algorithms for genome rearrangement analysis in the presence of noise and incomplete information.

For coarseness and missing data, we represent each chromosome as a partial order, summarized by a directed acyclic graph (DAG). 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. The major contribution of the thesis is an algorithm for extracting a maximal decomposition of some subgraph into alternating coloured cycles, determining an optimal sequence of rearrangements, and hence the genomic distance. Also based on this framework, we have proposed an algorithm to solve all the above problems of comparative maps simultaneously by adding heuristic preprocessing to the exact algorithm approach. We have applied this to the comparison of maize and sorghum genomic maps on the GRAMENE database.

A further contribution treats the inflation of genome distance by high levels of noise due to incorrectly resolved paralogy and error at the mapping, sequencing and alignment levels. We have developed an algorithm to remove the noise by maximizing strips and tested its robustness as noise levels increase.

Résumé

L'algorithme de Hannenhalli-Pevzner pour calculer la distance évolutive entre deux génomes est très efficace quand les génomes sont signés (l'orientation, ou la direction de lecture, des marqueurs est connue) et chacun est muni d'une relation d'ordre total. Cependant, dans les vraies cartes comparatives, il manque parfois de relations d'ordre entre marqueurs (résolution insuffisante), on ne connaît pas toujours l'orientation des marqueurs et d'autres marqueurs sont absents. Il y a aussi de la paralogie, des conflits d'ordre et du bruit statistique. Dans cette thèse, nous développons une suite d'algorithmes pour l'inférence d'ordre des génome en présence de bruit et d'information partielle.

Dans le cas de résolution insuffisante ou de marqueurs absents, nous représentons chaque chromosome comme ordre partiel, récapitulé par un graphe acyclique dirigé (DAG). Nous augmentons chaque DAG afin de créer un graphe dirigé (DG) dans lequel toutes les linéarisations possibles sont incorporées. Les DGs chromosomiques représentant deux génomes sont combinés pour produire un graphe bicolore. La contribution principale de la thèse est un algorithme pour extraire une décomposition maximale d'un certain sous-graphe en cycles où les couleurs alternent, ce qui détermine un ordre optimal des réarrangements, et par conséquent la distance génomique. En nous basant sur cette idée, nous avons proposé un algorithme pour résoudre simultanément tous les problèmes des cartes comparatives mentionnées ci-haut, en ajoutant le prétraitement heuristique à l'approche algorithmique exacte. Nous avons entrepris l'application de notre méthode à la comparaison des cartes génomiques de maïs et de sorgho obtenues dans la banque de données GRAMENE.

Une autre contribution traite de la distorsion de la distance génomique par le bruit dû à la paralogie mal résolue et des erreurs de cartographie aux niveaux de positionnement des marqueurs, de séquençage et d'alignement. Nous avons développé un algorithme pour enlever le bruit en maximisant des bandes de marqueurs consécutifs dans les deux génomes et nous avons examiné sa robustesse en fonction du niveau de bruit.

Keywords

comparative genomics, bioinformatics, comparative maps, gene order, Hannenhalli-Pevzner algorithm, partial order, directed acyclic graph, unsigned genomes, noise

Table of Contents

Table of Figures

List of Tables

Chapter 1 General Introduction

The entire complement of genetic material carried by an individual is called the genome. With the exception of some viruses, each genome contains one or more DNA molecules, one per chromosome. DNA consists of two complementary strands twisted around each other to form a right-handed double helix. A gene is a segment of DNA sequence with a specific function such as coding for a protein or RNA structure. Genes can be ordered by their DNA sequence location. Because genes may be located on both strands of a DNA molecule, a sign (+/-) is usually used to indicate on which strand a gene is located. If two genes in different species are similar due to sharing a common ancestor, these genes are called homologs. There are two kinds of homologs: Orthologs and paralogs. Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Paralogs are genes related by duplication within a genome. Orthologs tend to retain the same function in the course of evolution, whereas some paralogs evolve new functions, often related to the original one.

The genome is structurally specific to each species, and it changes only slowly over time. Therefore genome comparison among different species can provide us with much evidence about evolution. We may distinguish two ways of comparing genomes: one is to look at local changes such as point mutation. Another way is to compare the order of the genes or other markers in two genomes. This order may have changed through genome rearrangement during evolution. Although genome rearrangements are much less frequent than point mutation, they are important because they involve major segments of the genome and are likely to be involved in speciation during evolution.

Comparative maps provide information on the linear ordering of orthologous genes on

the chromosomes of two or more genomes. The first chromosomal map dates from 1913 [34], at the same time the definitive chromosomal theory of heredity [21] was being elaborated. Soon comparative mapping had become an integral part of genetic research, e.g., Fig. 1 in [35], published in 1921. Long before the genomic era, comparative maps existed for *Drosophila* and other insects, mammals, including humans, livestock and rodents, cereals and other eukaryotic and prokaryotic groups.

Despite their immediate availability and the wealth of evidence they contain about evolutionary history, traditional comparative maps were bypassed when genome rearrangement algorithms [15, 16], inspired by analyses of organelle and other small genomes (e.g., [26, 32]), were adapted for direct use on DNA segments derived from whole nuclear genome sequences [27, 4, 6].

In this thesis we will consider rearrangement algorithms for gene order in comparative maps.

1.1 Genome rearrangements

Genome rearrangements are an important aspect of the evolution of species. Even where the gene content of two genomes is almost identical, gene order can be quite different. The accumulation of inter- and intra-chromosomal exchange events over time can result in a patchwork pattern of conserved segments, as shown in Figure 1.1.

Geneticists have studied genome rearrangements almost from the first statements of the chromosome theory of heredity early in the twentieth century [36], but modern mathematical approaches are barely twenty years old [42,22]. One such approach is the attempt to infer the series of rearrangement events responsible for producing a pattern such as that in Figure 1.1.

Several algorithms have been proposed to calculate the genomic distance, i.e., the smallest number of operations or events necessary to transform one genome to another, as well as the actual operations that accomplish this. A chart of the gene-order-related rearrangement operations usually studied is shown in Figure 1.2.

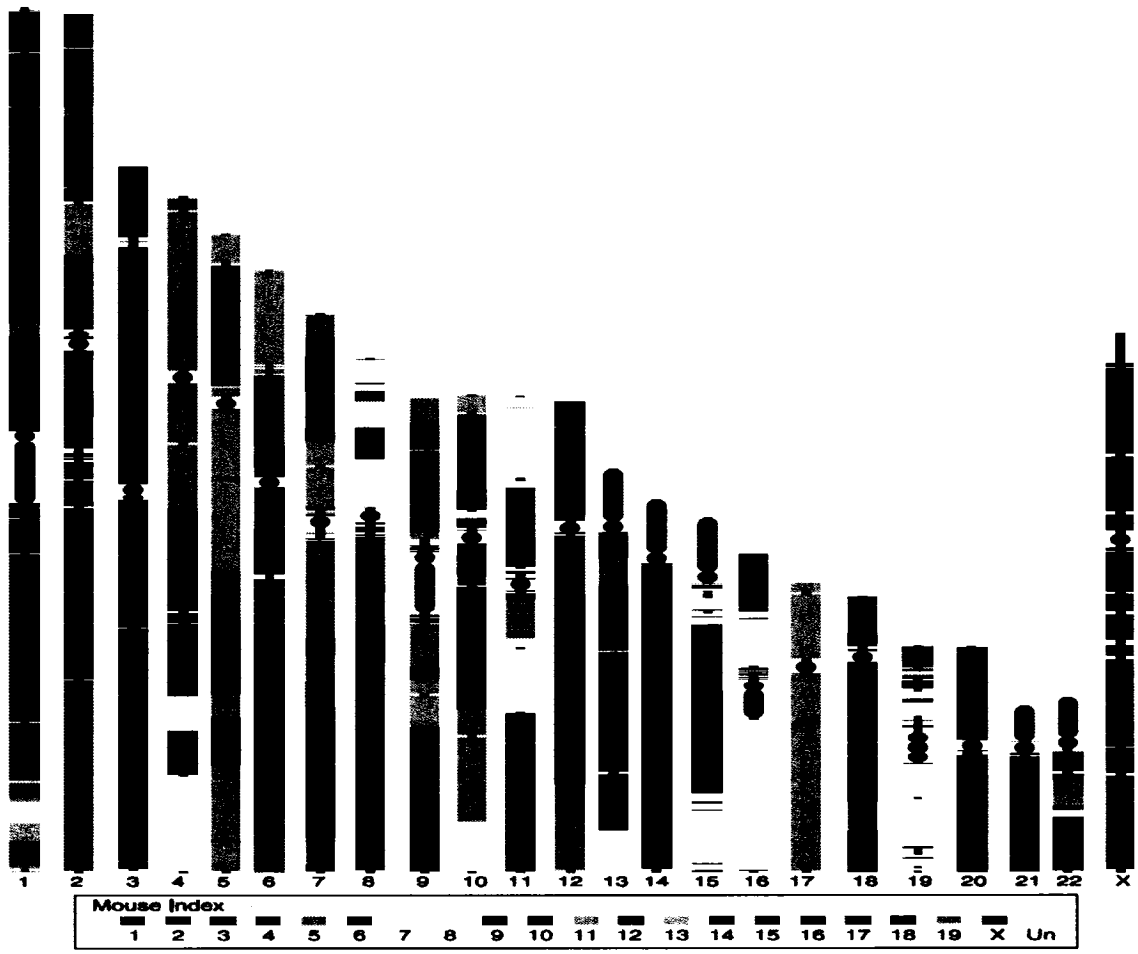


Figure 1.1: Synteny on human and mouse (from Eichler and Sankoff 2003)

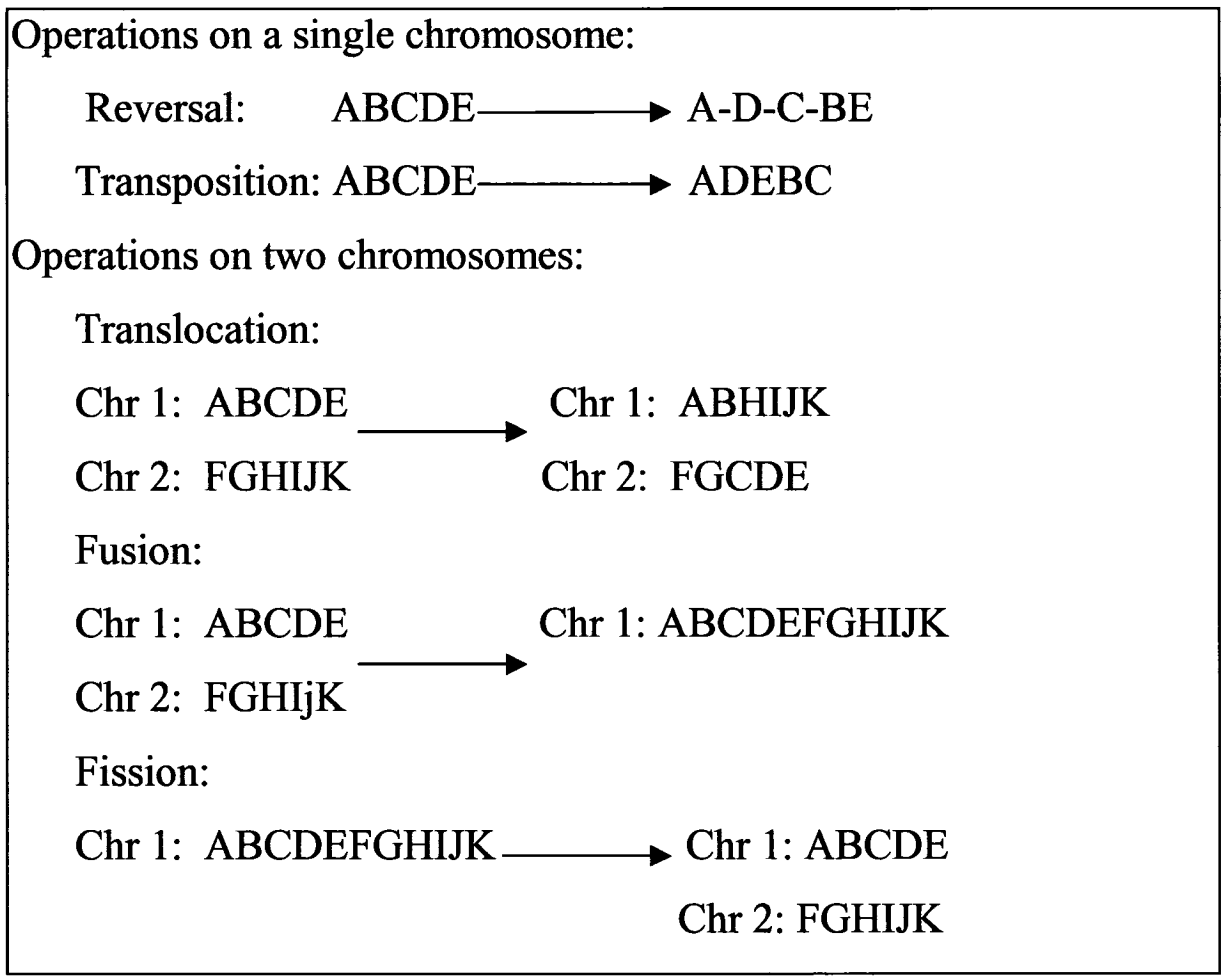


Figure 1.2: Genome rearrangement operations

In 1995, Hannenhalli and Pevzner [15] published a polynomial algorithm for computing reversal distance and generating optimal reversal scenarios in unichromosomal genomes. Shortly thereafter, they published a translocation plus reversal distance algorithm for multichromosomal genomes, using similar techniques [16]. In 2002, Tesler [39] improved the latter algorithm by incorporating the method of Bader, Moret and Yan [2] so that the time cost of computing the distance is $o(n)$ and the time cost of producing translocation and reversal scenarios is $o(n^2)$. In 2005, Yancopoulos et al. [43] introduced a rearrangement algorithm that includes the transposition operation as well as reversal and translocation.

1.2 The bicoloured graph in rearrangement algorithms

Hannenhalli and Pevzner [16] showed how to find a shortest sequence of reversals and translocations that transform one completely specified genome χ with n genes on k chromosomes into another genome ψ of the same size but with h chromosomes, in polynomial time. Completely specified means that each chromosome is totally ordered, the sign of each gene is known, and there is no paralogy.

As described in [39], we construct a bicolored graph on $2n + 2k$ vertices that decomposes uniquely into a set of alternating-colored cycles and $h + k$ alternating-color paths, as shown in Figure 1.3. First, each gene x in χ determines two vertices x_t and x_h. Two dummy vertices e_{i1} and e_{i2} are added to the two ends of chromosome χ_i, $i = 1, \ldots k$. The adjacencies in χ determine red edges. If x is the left neighbor of y in χ, and both have positive polarity, then x_h is connected by a red edge to y_t. If they both are negative, x_t 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_{i1} 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_{i2} 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.

Each vertex is incident to exactly one red and one black edge, except for the dummies in χ and the (non-dummy) vertices at the ends of chromosomes in ψ, which are incident to only a red edge respectively. 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 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:

$$r + t = n + k - j - c + \theta \tag{1.1}$$

where θ is a correction term that is usually zero in the case of simulated or empirical data. For simplicity of exposition, we ignore this correction here. Indeed, in a recent framework [43] allowing p transpositions and more general block interchanges, $\theta \equiv 0$, and we simply have

$$r + t + 2p = n + k - j - c. \tag{1.2}$$

The $n + k - j - c$ actual rearrangement steps for transforming χ into ψ can then be found via certain well-defined operations on the cycles of the bicoloured graph.

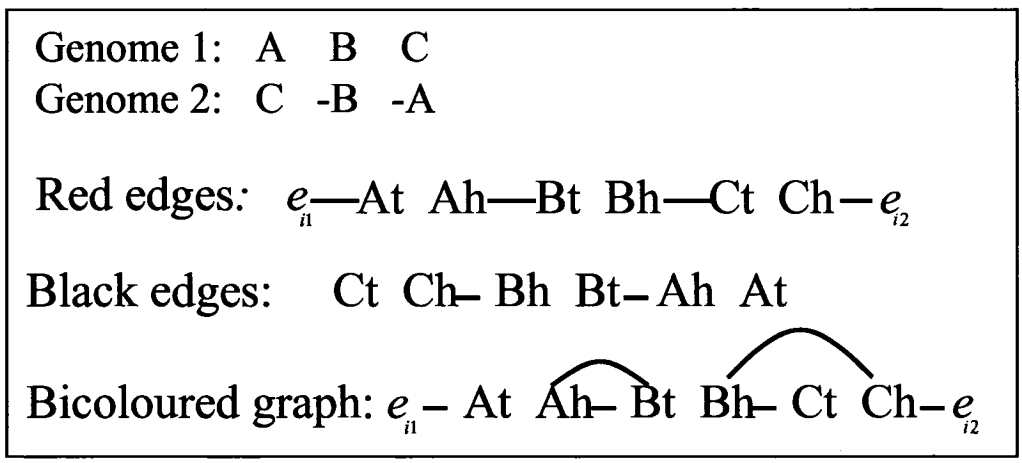


Figure 1.3: Bicoloured graph

1.3 Gene order data in comparative maps

As mentioned in Section 1.2, the Hannenhalli-Pevzner algorithms and their improvements require completely specified genomes as the inputs, and infer the smallest number of operations to transform one genome to another.

However, in traditional comparative maps, such ideal data is not available. Here are the difficulties we usually encounter when we apply the algorithms to the real data:

Noise: Experimental methodologies and statistical mapping procedures inevitably give rise to some small proportion of errors, two neighboring genes incorrectly ordered, a gene mapped to the wrong chromosome, a gene incorrectly named or annotated. As a result the maps may contain much erroneous information. In addition, in many contexts, noise arises because of misinterpreted paralogy caused by gene, segment or genome duplication.

Coarseness and missing data: Maps of genes or other markers produced by recombination analysis, LD mapping, physical imaging and other methods, no matter how highly resolved, inevitably are missing some genes or markers and fail to order some pairs of neighboring genes with respect to each other. Even at the ultimate level of resolution, that of genome sequence, the application of different gene-finding protocols usually gives maps with different gene content.

Uncertain orthologs: Several copies of a gene in a map lead to a one-to-many or many-to-many correspondence between two maps. Different notations used in different genome maps increase the difficulty in deciding which gene in one genome corresponds to which gene in the other genome.

Missing signs: Genetic and physical mapping methods may not give us the information about which DNA strand the gene is on.

Conflicts: When two or more relatively sparse maps of a genome, compiled from different sources, are combined prior to comparison with the map of other genomes, there is often conflict concerning the orders of some of the markers on both maps.

1.4 Overview of the chapters

In this thesis I propose several algorithms focused on the above problems in comparative maps. The general principle is: Reformulate the rearrangement problem without changing the objective function (minimum of rearrangements necessary to transform one genome into another) so that the solution also provides (estimates) the missing biological information.

This thesis is based on my four papers that are published or accepted. Thus it is organized as follows:

- **Chapter 2: Reversal Distance for Partially Ordered Genomes [44]**

  This chapter deals with the reversal distance of two uni-chromosomal genomes that suffer from coarseness and missing data. For each genome, we get as complete as possible gene order information by combining all available datasets for this genome. Then our algorithm calculates the distance based on the combined gene order information and at the same time extracts a total gene order for these two genomes.

- **Chapter 3:Genome Rearrangements with Partially Ordered Chromosomes [45]**

  This chapter extends the result in Chapter 2 from uni-chromosomal to multi-chromosomal genomes. It also introduces an efficient modification for the branch and bound search strategy for the important one-sided case (one totally ordered genome and one partially ordered genome).

- **Chapter 4: Reversals of Fortune [33]**

  The exact algorithms in Chapter 2 and 3 focus on problems such as coarseness and missing data. In this chapter, we broaden our view to other kinds of problems in comparative maps. Many papers propose methods to solve these problems individually. We describe an approach to solve them all simultaneously. Although

this is mainly a heuristic approach, it is an effective and feasible approach to real problems.

- **Chapter 5: Rearrangement of Noisy Genomes [46]**

  In this chapter, we propose a method for genome rearrangement analysis in the presence of noise. The algorithm removes the noise by maximizing the decomposition of the genomes into corresponding ascending or descending strips.

Chapter 2 Reversal Distance for Partially Ordered Genomes

2.1 Introduction

Structural rearrangement within uni-chromosomal organism can be modeled by the transformation of one signed permutation to another by means of reversals. Analyses that compare gene orders, such as the Hannanhalli -Pevzner algorithm [15], 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, ..., ,*n*} to the reference order 12 ... *n*, where the allowed operations consist of reversals of any substring. 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*. 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: $g_1 \ldots g_{ni} = -g_{ni} \ldots -g_1$.

As mentioned in Chapter 1, the total order of a chromosome inherent in its representation as a permutation or string 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. Unfortunately, the concepts and methods of genome rearrangement including the Hannanhalli-Pevzner algorithm pertain only to totally ordered sets of genes, markers or segments, and are meaningless in the context of partial orders.

Here we extend the standard 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 data set. The lack of order information in each data set, 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 must be broken. We suggest a number of reasonable alternatives 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 *to infer a sequence of reversals that 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 possible linearization of all the partial information in all the data sets 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 that finding the rearrangements themselves 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.

2.2 Partially Ordered Gene Data

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 2.1.

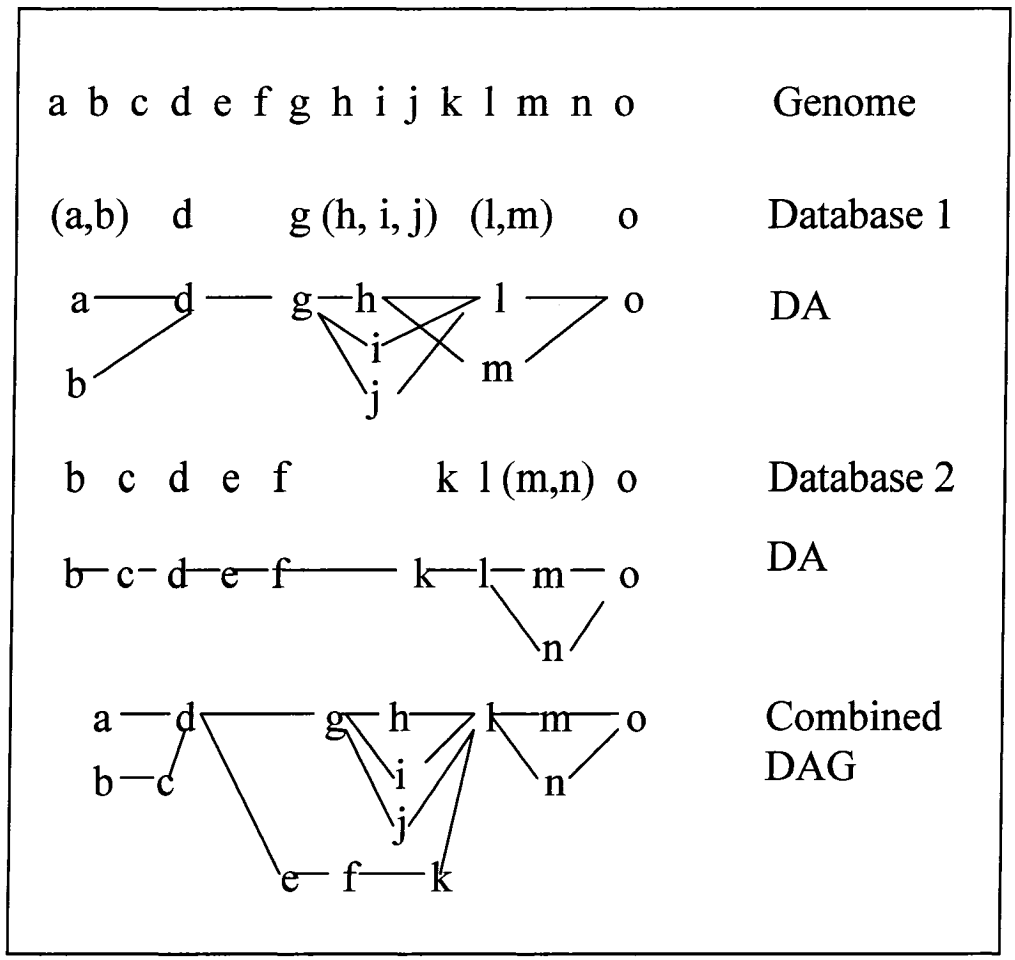


Figure 2.1: Construction of DAGs from individual databases each containing partial information on genome, due to missing genes and missing order information, followed by construction of combined DAG representing all known information on the genome. All edges directed from left to right.

2.3 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 directed graph (DG) containing all the edges of the DAG plus two edges of opposite directions between all pairs of vertices, which are not ordered by the DAG. This is illustrated in Figure 2.2. The routine for constructing this graph is **dgger.**

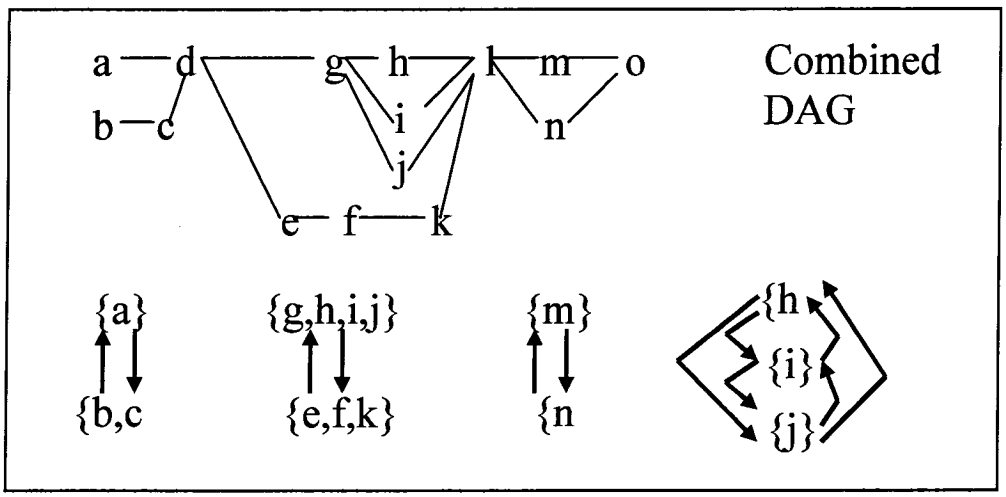


Figure 2.2: Edges added to DAG to obtain DG containing all linearizationas paths (though not all paths in the DG are linearizations of the DAG!). Each arrow represents a set of directed edges, one from each element in one set to each element of the other set.

2.4 The algorithm

The procedure **make_bicoloured** produces the bicoloured graph that we described in Chapter 1. It can be applied not only to totally ordered chromosomes but also to the set of edges in the DGs for the two genomes. Because in this chapter each genome contains only one chromosome, we can simplify the algorithm by adding the same dummy nodes to both genomes instead of just one. In this way, the graph just contains cycles and we do not need to consider the additional complexity represented by paths.

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 color, 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 2.3.

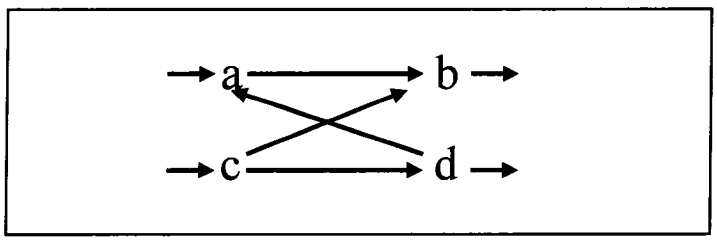


Figure 2.3: If ab and cd are DAG edges, then the two non-DAG edges da and bc are mutually exclusive, since together they lead to the wrong order for a and b.

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 genome 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 color as ε, in which case additional order is imposed by the choice of ε. All remaining edges of the same color 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 as 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 that 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 chose. 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

$$c' + n + 1 - m/2 < c^*, \tag{2.1}$$

this branch of the search is abandoned and backtracking begins.

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 branch and bound 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

the computing time reduction due to the application of the bound in (2.1), 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 2.3.

2.5 Summary of the analysis

Figure 2.4 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.

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,
    combine_po
    dagger
    dgger
(Add signs to markers)
make_bicoloured
find_cycle_decomp
Output: Optimal cycles and linearizations

Figure 2.4: Summary of steps in simultansous linearization and reversal inference. Step in parentheses not required for signed maps.

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 $O(n^2)$.

2.6 Simulated data

In Section 2.7 we will apply our algorithm to real genomes. First, we simulate DAGs representing varying levels of uncertainty, modeling lack of resolution of gene order, and missing genes in two data sets.

For a totally ordered genome containing n genes, we simulate m data sets 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 grouped and mapped to the same position) and the probability that any particular gene is deleted (i.e., does not show up on the map), respectively. Each gene is submitted to a group event with probability p, and a deletion event with probability q conditioned on that the gene cannot be deleted from all the datasets.

In any particular data set, 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 data sets are constructed as for the original genome.

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

We carried out simulations with $n = 12$ genes 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. This trend is indicative, though hardly statistically significant with only five runs.

Thus except with very high rates of missing and superimposed genes, enough information is retained in the incomplete maps that our method can usual recover the correct reversal history intervening between the two genomes.

2.7 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 [27] and the "Paterson 2003" and "Klein 2004" [6, 19] 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 2.1.

The two DAGs constructed from the maize sets and the sorghum sets by the routines **combine_po** and **dagger** are depicted in Figure 2.5.

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 evolving 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 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 strandness, 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 strandness, and thus the usual solution in computational genomics is to assign polarity so as to minimize the rearrangement cost, cf. http://www.cse.ucsd.edu/groups/bioinformatics/GRIMM/ and Tesler [38].

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

Table 2.1:(Vertical) orders of markers appearing in at least one maize dataset and one sorghum dataset, numbered according to the "IBM2 neighbors 2003" dataset. Parentheses group markers mapped to the identical chromosomal position. Horizontal alignment of markers in different datasets is arbitrary and has no significance.

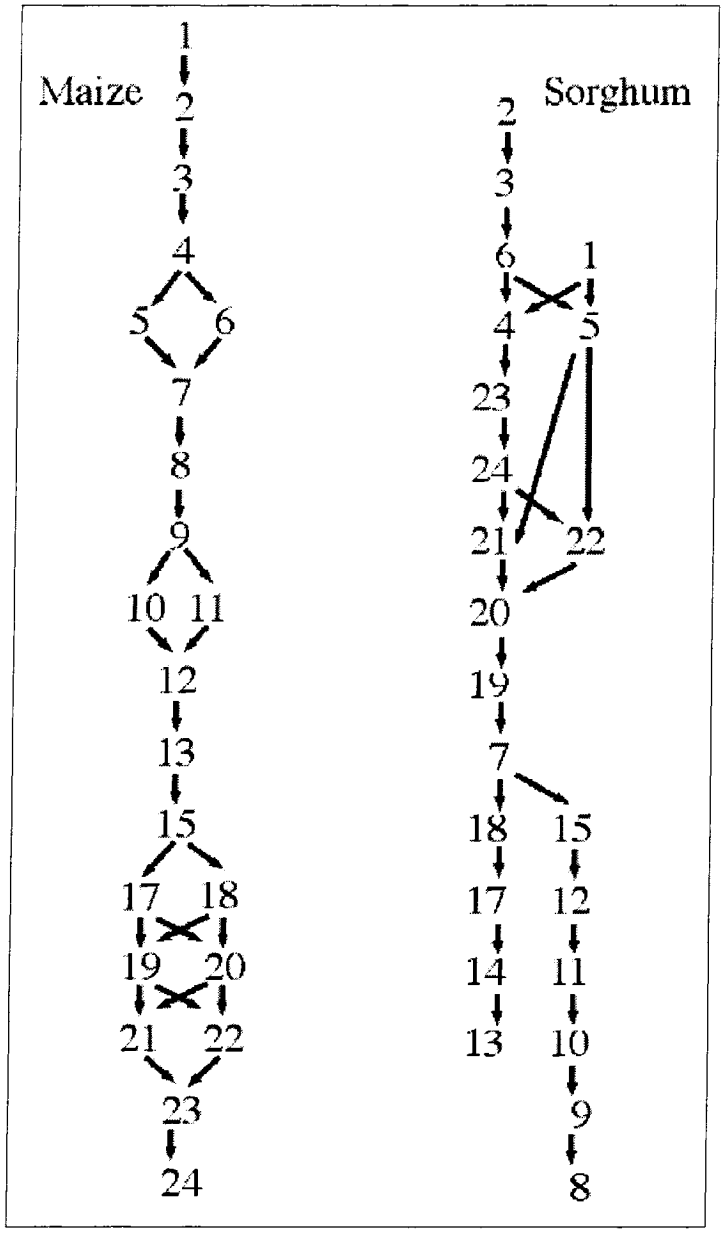


Figure 2.5: DAGS constructed from by combining the partial orders of two maize datasets and two sorghum datasets presented in Table 2.1.

In our example, this turns out to require a negative polarity on sorghum markers 4 to 6, and 8 to 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, respectively for maize and sorghum, are

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 10 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

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 data sets for each chromosome as represented in the DAG, provided 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.

2.8 Conclusion

In this chapter we have proposed a more realistic version of the genome comparison problem, for unichromosomal organism where the gene order or marker order in the two genomes is incompletely specified.

When diverse maps of the same genome are combined to produce a more detailed map,

the main sources of uncertainty in the order is 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 of these maps.

We have shown that the appropriate way to combine maps of this sort, to retain all the order information in each, 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.

In finding the most economical rearrangement scenario, 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 moderate-sized data, although we have only exemplified by a small chromosomal comparison between maize and sorghum. The current implementation will handle a problem with a few hundred markers, as long as there are not too many missing data per map and not too many unresolved marker locations. Moreover, there are many ways to improve the efficiency of this method so that it will work with much larger n. Some of them will be discussed in Chapter 3.

Our method is most appropriate for genomes with relatively little genomic sequencing, where maps of such genomes frequently do not specify strandness. Eventually, it will be necessary to implement an automatic way of assigning polarity such as the one to be discussed in Chapter 5.

We point out that 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 [38].

Chapter 3 Genome Rearrangements with Partially Ordered Chromosomes

3.1 Introduction

In Chapter 2 we developed a strategy to adapt genome rearrangement algorithms to study comparative maps with one chromosome where some order information is lacking. In this chapter we will generalize the algorithm to maps with one or more chromosomes.

The genetic structure of a genome is modeled, in traditional genome rearrangement studies [39], 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, signed and totally ordered: e.g., $g_1 < \cdots < g_{ni}$, also written simply as $g_1 \cdots g_{ni}$, 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 that 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 in number of genes. An example of a genome of size 8 is thus $\{\chi_1, \chi_2, \chi_3\}$, where χ_1 = 5 -1 -4, χ_2 = 7 8 3, and χ_3 = -2 6.

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. We discussed reversals in Chapter 2. 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* → *g h y z, x i; g h i, x y z* → *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.

The Hannenhalli-Pevzner algorithms [15, 16] for comparing genomes, and their improvements [39, 2], infer $d(\chi,\psi)$, a metric distance that counts 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.

3.2 Simulating incomplete maps of pairs of two related 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

$$\chi = \{m_1 \cdots n_1, m_2 \cdots n_1 + n_2, \cdots, m_k \cdots n\}, \tag{3.1}$$

where $m_1 = 1$ and the remaining $m_i = m_{i-1} + n_{i-1}$. The genes in two genomes being compared through translocation and reversal distance may always be relabeled so that in one of the genomes they all have positive sign and have the form in (3.1).

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 are 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 $f_2 = 1$, (3.2) shows genome χ and an example of ψ.

$$\chi = \{1\ 2\ 3\ 4\ 5\ 6\ 7,\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15,\ 16\ 17\ 18\ 19\ 20,\ 21\ 22\ 23\ 24\ 25\} \qquad \psi = \{-17\ -3\ -2\ -1\ 18\ -5\ -4\ -16,\ 8\ -13\ -12\ -11\ -22\ -21,\ -7\ -20\ 10\ -24\ -23\ 25,\ 14\ 15,\ 9\ -19\ 6\} \tag{3.2}$$

Once we have obtained simulated genomes χ and ψ, we can simulate the maps according the method described in Section 2.6. Three data sets produced from the genomes in (3.2) are shown in (3.3), with boxes around unresolved groups of genes, i.e. genes whose relative order in unknown.

$$\chi = \{1\ 2\ [4\ 5]\ 7,\ 8\ 10\ 11\ 15,\ 16\ 17\ 20,\ 22\ 23\ 25\} \quad \{[2\ 3]\ 6\ 7,\ 8\ 9\ 12\ 14,\ 17\ 18\ 19,\ 21\ 24\ 25\} \quad \{2\ 4\ 6,\ 9\ 11\ [12\ 13]\ [14\ 15],\ [16\ 18]\ 20,\ [21\ 23]\ 24\}$$

$$\psi = \{-17\ -2\ 18\ -4,\ [-13\ -12\ -11]\ [-22\ -21],\ -7\ [-20\ 10\ -24\ -23]\ 25,\ 14\ 15,\ 9\ -19\} \quad \{[-17\ -3\ -2]\ -1\ [-5\ -4\ -16],\ [8\ -13]\ [-12\ -11\ -22],\ [-7\ -20\ 10]\ [-24\ 25],\ 14\ 15,\ -19\ 6\} \quad \{[-2\ 18]\ -5\ -16,\ 8\ [-13\ -11]\ -21,\ -20\ 10\ -24\ -23,\ 14\ 15,\ [9\ 6]\}$$

(3.3)

3.3 Generalization from uni-chromosomal to multi-chromosomal genomes

Our strategy here is almost same as that in Chapter 2 except as follows:

1. The version of the genome distance algorithm we use is that developed by Tesler in 2002 [39] for the case of reversals and translocations. We only add dummy nodes for chromosomes in genome χ instead of both of them. The bicoloured graph and distance have already been described in Chapter 1.
2. Compared to uni-chromosomal genomes, 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 Chapter 2 because the dummies 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.

3. 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". 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

$$c' + n - h - (m - l)/2 + j' + h + k - l = c' + j' + n + k - (m + l)/2 < c^* + j^*, \qquad (3.4)$$

this branch of the search is abandoned and backtracking begins.

4. Backtracking is also invoked if no cycles or paths can be made up of the unused vertices. This is caused by the wrong choice of terminal node for a path.

3.4 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.

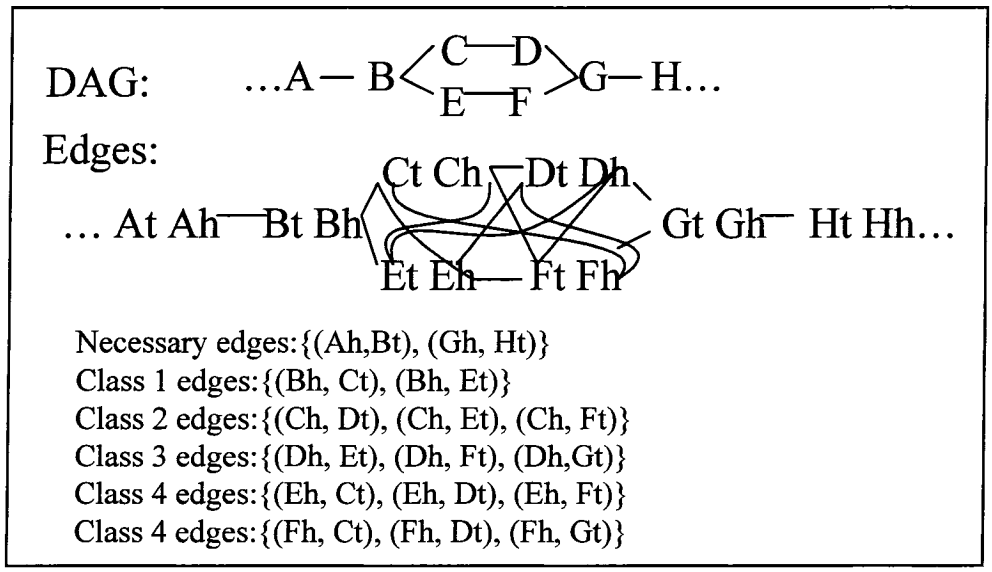


Figure 3.1: The edges can be grouped in 5 classes. A solution must include all necessary edges and exactly one edge from each class

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. This is illustrated in Figure 3.1. 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-th 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.

3.5 Analyzing the simulated incomplete data

We submitted the data in (3.2), with incomplete order information, to our analysis. In (3.6)

we compare the results to the original genomes in (3.2).

True genomes:

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

Reconstructed genomes:

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

(3.6)

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 (3.3), however, shows that the information present in each data set for both genomes does not bear on the ordering within any of these pairs.

In addition, the reconstructed linearized chromosomes of χ in (3.6) require 4 translocations, 1 fission and 4 inversions to be transformed into the reconstructed ψ, exactly how ψ was originally obtained from χ in (3.2).

3.6 Conclusion

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 $O(n)$. And in the speed-up, 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

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

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.

We submitted the data to both versions of the algorithm to compare the efficiency. When the data are suitable for the speed-up version (one genome is totally ordered, and the other with small amount of uncertainty with respect to n), the running time of the speed-up version is much less than the other version, 10 seconds instead of 18 minutes on a Macintosh G4.

Chapter 4 Reversals of Fortune

4.1 Introduction

In this chapter we generalize our approach to the application of rearrangement methods to traditional comparative maps, i.e., maps based on estimates of gene and marker locations in nuclear genome, and not directly on genome sequence. As mentioned in Chapter 1, comparative maps may suffer from the problems listed below:

1. **Coarseness**: Lack of resolution of the maps, i.e., two or more genes mapped to the same position in one of the genomes. Genome rearrangement algorithms require that the input markers be totally ordered along each chromosome.
2. **Missing data**: Each map just contains subset of genes or marks of a genome. Two genes or markers which are not ordered by any of the component maps will often remain unordered in the composite map. Again, rearrangement algorithms require that the input markers be totally ordered along each chromosome.
3. **No signs:** No information about reading direction, i.e., which DNA strand the gene or marker is on. This information is not available from many of the methods used to construct maps. Genome rearrangement algorithms require this "orientation" information for efficient and exact execution.
4. **Uncertain orthology**: This leads to a one-to-many or many-to-many correspondence between the two maps. Genome rearrangement algorithms require one-to-one correspondences as input.
5. **Conflicts**: When two or more relatively sparse maps of a genome, compiled from different sources, are combined prior to comparison with the map of another genomes, there is often conflict concerning the orders of some of the markers on both maps.

With the possible exception of **paralogy**, these difficulties are neatly avoided while complete genome sequences are being compared at the sequence level [27, 4, 6], though of course there are many other technical problems to be solved in that approach.

The difficulties listed above all have in common that we are missing some essential biological information required to carry out genome rearrangement analysis. Moreover, in each case *the genome rearrangement problem may be reformulated in such a way that the solution not only provides a minimal series of reversals and/or translocations necessary to transform one genome into another, but also supplies an optimal estimate of the missing information.* It is the comparative context, together with the rearrangement-minimizing objective function, which "fills in" the gaps in our biological knowledge in the most reasonable way. This unexpected bounty from the rearrangement analysis is what is alluded to in the title of this chapter.

Exact algorithms have been published to take care of **coarseness, missing data, no sign** and **paralogy**, all requiring exponential worst-case computing time. The latter two, the topics of Sections 4.2 and 4.3, respectively, have been proved NP-hard, and we have conjectured as much for the first two as discussed in Chapter 2 and 3.

Solution of a typical comparative map rearrangement problem would require treating at least **coarseness, no sign** and **paralogy** simultaneously, and usually **missing data** and **conflict** as well. We will state the pertinent combinatorial optimization problem, but its exact solution would be feasible only on very particular, small instances. We do, however, give results of applying our exact algorithms allowing for **coarseness** and **missing data** from Chapter 2 and 3, in all generality, applied to data where **no sign, paralogy** and **conflict** are dealt with heuristically during preprocessing, using some of the key ideas in their

respective algorithms.

4.2 Sign assignment

Our first problem is that of adding signs to an unsigned genome so as to achieve a minimal reversal distance to the identity permutation 1, ···, *n*. This is equivalent to the problem of sorting an unsigned permutation, known to be NP-hard [9].

As conjectured in [18] and proved in [17], for all segments of the permutation consisting of three or more consecutive integers (strips) in increasing order, plus signs can be given to all these integers, and for all decreasing strips, minus signs can be given, and this assignment is consistent with a solution. In [18], it is also shown how to give signs to 2-strips. The algorithm these authors develop is exponential only in k, the number of singletons, and is polynomial if k is *O(log n)*. In our experience with comparative maps, however, *k* often seems closer to *O(n)*.

Though there is much recent literature on approximation to unsigned reversal distance, not much more work has been done on exact algorithms. Tesler has extended the approach in [18] to reversal and translocation distance, and implemented it in GRIMM [38].

4.3 Duplicate genes, paralogy, gene families

When there are paralogs, gene orders cannot be modeled as permutations, but only as more general strings. Though sorting strings by reversals can be done in polynomial time, this does not automatically give the reversal distance between strings, in contrast to sorting permutations by reversals, which is equivalent to calculating reversal distance. Indeed, reversal distance for strings is NP-hard [29].

The problem in analyzing genomes containing paralogs is how to decide which paralog in one genome should be identified with which one in the other genome, in a biologically meaningful way. Thus string-based analyses that attempt to match all or as many as possible of the paralogs of a gene in one genome to distinct paralogs in the other are only meaningful under the often assumption that all paralogs were present in the common ancestor genome.

A less ambitious, but biologically more reasonable, approach is to try to match only one paralog of each gene in one genome to one in the other, such that the gene orders of the matched paralogs (the *exemplars*) of each family result in a minimal reversal distance [30]. The hypothesis motivating this is that genomes reduced to contain only these exemplars will better tend to reflect the actual reversal history than reduced genomes made up of any other choice of exemplars, using a parsimony argument.

There is a growing literature on the problem of incorporating paralogy into genome rearrangement theory. This is most meaningfully carried out within the phylogenetic context [31, 5], taking into account that the origin of paralogs in duplication events may occur on earlier or later branches of the evolutionary tree. In addition to work characterizing, approximating or generalizing the exemplar approach [8, 25], there is research on rearrangement in the context of string theory [11, 29], conserved interval/block theory [3, 5] and other a number of other approaches [10, 37]. Virtually all of these are based on the same principle, that matching of paralogs should minimize the rearrangement distance.

4.4 Partial order and conflict

The detailed solution for the partial order problem is described in Chapter 2 and 3. It is an exact algorithm and it works for moderate size data.

One problem we have not dealt with is **conflict**; different maps of the same genome do occasionally conflict, either because marker b < marker 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 is 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. Perhaps the approach that best balances loss of information with ease of application and interpretation is to discard a minimum set of gene occurrences so that all order conflicts are resolved. This method also resolves conflicts due to gene assignment, i.e. different maps assign a gene to different chromosomes. Any gene that is discarded from all the data sets for one genome has, of course, to be discarded from the other.

4.5 Synthesis and application

Given a map comparison that suffers from some combination of **coarseness, missing data, no sign** and **paralogy**, we can ask: simultaneously find the exemplars and sign assignments resulting in a minimum number of translocations and inversions necessary to transform some DAG linearisation of one genome into some DAG linearisation of the other. Since all three component problems are NP-hard, there is scant hope that their combination is tractable. In this section, we describe a practical approach to one problem of this type.

Note that if there is **conflict**, we might want to avoid discarding exemplars in resolving conflict; if that is impossible, then we should at least take into account the sizes of any discarded gene families to assure that a minimum of genes occurrences are discarded. In any

case, this minimum should be established beforehand, and should constrain the exemplar selection, if this is an issue. Under this one constraint, the goal is the minimization of genomic distance over all combinations of exemplar choices, eligible conflict resolutions, sign assignments and DAG linearization.

The particular application we study, using the implementation of the DAG linearization described in Chapter 3, is the comparison of the maize and sorghum genomes. We used one set of genomic markers for maize [28] and two for sorghum [20, 7] as accessible in Gramene [41]. We extracted all markers registered as having homologs in maize and at least one of the sorghum datasets. This gave 463 marker occurrences in maize and 387 in sorghum, based on 296 total different markers.

Partly because the size of this problem is excessive for our implementation, we ignored any pairs of chromosomes, one in maize and one in sorghum, with less than four markers in common. Some threshold, though perhaps not as large as four, is also justified by the facts that occasional syntenies of this sort are often the result of incorrect marker homology assignment or other error, and that especially in the case of singletons, the rearrangement solution simply includes two or three rearrangements solely to account for the position of this marker, and is independent of the rearrangement of the rest of the genome. This step left us with 381 marker occurrences in maize and 301 in sorghum, based on 263 total non-homologous markers. Thus by removing only 33(11%) of the 246 different markers from the original data, we remove 82+86 = 168 (65%) of the 258 excess paralogs, consistent with our suspicion that these paralogs do not represent orthologies.

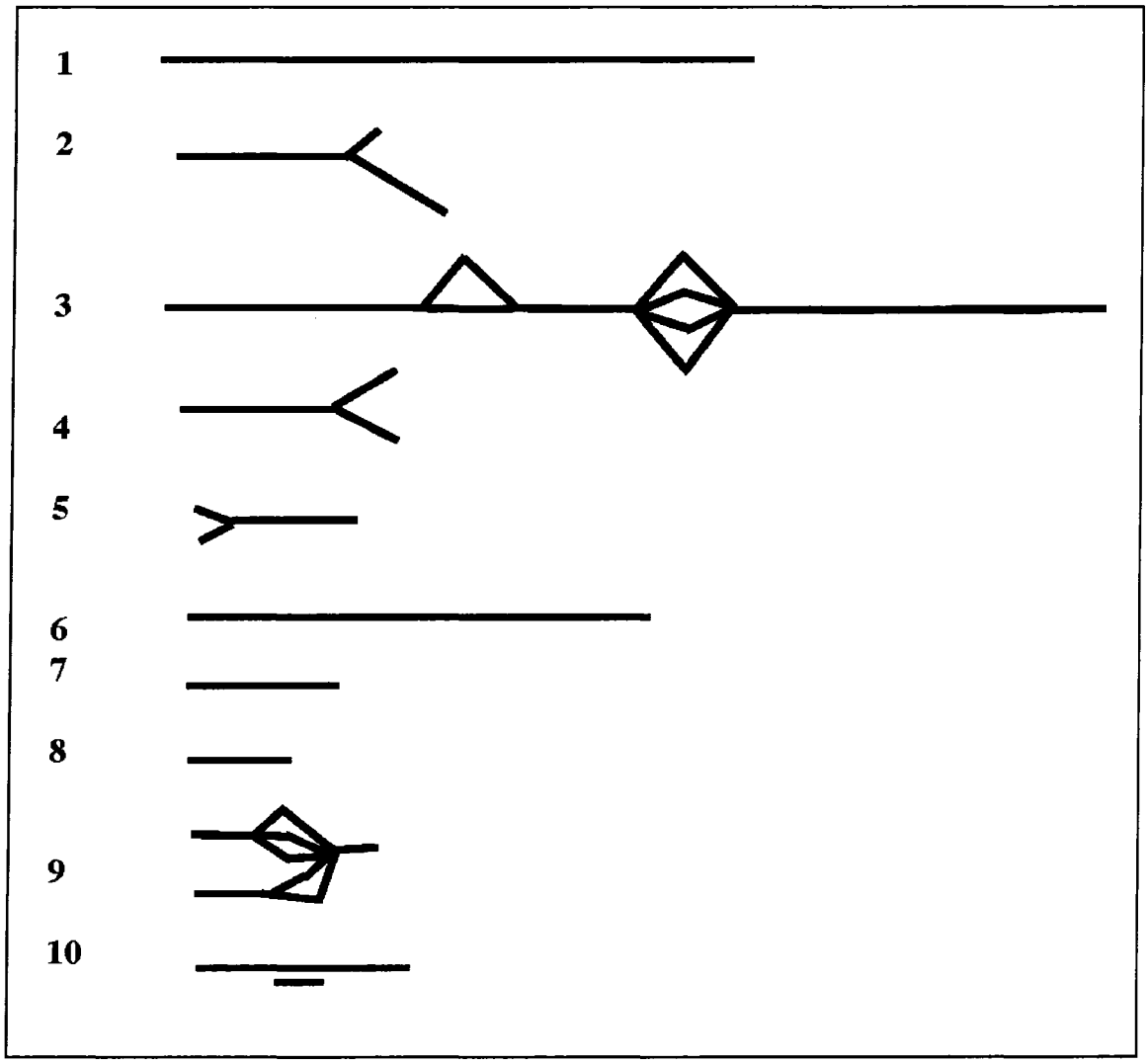


Figure 4.1: DAGs for the 10 sorghum chromosomes, scaled by number of markers.

As a next step, we identified all strips, as this is crucial not only to solving **no signs**, but is also helpful for **paralogy** and **conflict**. To take further advantage of strips, we removed paralogs and markers involved conflicts whenever they interrupted contiguous strips. We then found the exemplars for the remaining paralogies and resolved the remaining conflicts. To further reduce the size of the problem, we discarded a number of other singletons. Many of these were discarded because they interrupted a long strip, others were chosen because their presence or absence would not affect the solution for the remaining genes.

The remaining markers in two sorghum and one maize datasets, representing 191

different markers, constituting 99 strips and singletons, could then be input into our exact linearisation algorithm. The DAGs for the sorghum chromosomes are illustrated in Figure 4.2. The solution involved 6 non-trivial cycles (more than two edges) and 20 paths, implying a total of 73 inversions and translocations. Figure 4.3 portrays the configuration of the conserved segments in the two genomes, disposed on the sorghum chromosomes.

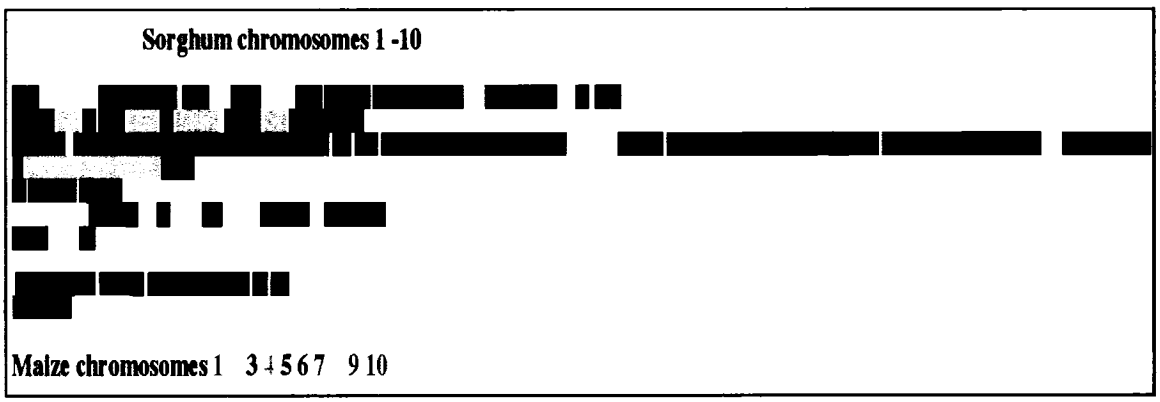


Figure 4.2: Conserved segments on sorghum chromosomes.

4.6 Conclusion

A generally usable algorithm for the simultaneous solution of the linearization and sign assignment problems seems feasible, since both can be handled within the partial order framework, though of course this is still a worst-case hard problem. There are many approaches possible to improve the current bound, to find a better sequence of edges as candidates to add to the current alternating colour cycle, and to incorporate heuristics, such as formalizing our strip maximization/ singleton-minimization procedure for discarding the most likely erroneous markers. Some of this is investigated in the next chapter.

The situation with paralogy and conflict is more complicated, as the strong constraint of acyclicity in the DAG representation of the map data cannot be satisfied. Nevertheless, there is hope for some method drawn from the homology assignment literature we have cited to be

incorporated into the solution of these problems in the comparative map context. The maize genome is known to have originated in a genome doubling event [14]; thus the treatment of duplicates through the exemplar or similar paradigm may be less appropriate than a genome halving analysis [13], which is only of polynomial complexity.

Chapter 5 Rearrangement of noisy genomes

5.1 Introduction

The comparison of genomic maps is susceptible to high levels of noise that threaten the accuracy of the results. This noise is due not only to error at the mapping, sequencing and alignment levels, but also in many contexts, to incorrect assignment of orthology due to gene, segment or genome duplication. In addition, comparison is sometimes hampered, in genetic maps, by lack of gene orientation information and lack of resolution of gene order discussed in the preceding chapter. In this chapter, we suggest an approach to genome rearrangement analysis in the presence of noise, and use simulation to estimate the robustness of the results as noise levels increase.

Our method, designed particularly for unsigned genes (unknown DNA strand) in multichromosomal contexts, is focused on the combinatorial search for the optimal set of compatible "pre-strips". A pre-strip is a set of two or more genes, not necessarily contiguous, on a single chromosome in one unsigned genome matched with genes in the same order (or reversed) on any single chromosome in the other genome, i.e., a non-trivial common subsequence of the two chromosomes (one of them possibly reversed). Any set of compatible pre-strips, when all other genes have been deleted from the genomes, becomes a set of "strips"; a strip [18, 17] is a pre-strip with all terms contiguous on the chromosome.

The compilation of pre-strips for our analysis will proceed by an initial identification of all maximal, or inextensible, pre-strips, i.e., pre-strips not contained in another pre-strip, followed by the addition of certain non-maximal pre-strips contained in each maximal pre-strip. We will then define a compatibility relation among all these pre-strips, namely whether or nor they can logically co-exist in the same chromosome. This procedure will be

followed by the search for a maximum-weight clique (MWC), using the algorithm in [19], where the weights are just the number of genes in the pre-strip. Though of course this algorithm may not execute in polynomial time, it is relatively efficient for dense graphs [1], such as the one representing pre-strips for multichromosomal genomes: where there are many chromosomes, a given pre-strip will not usually involve the same two chromosomes as another pre-strip, and hence will necessarily be compatible with it.

Once the maximal compatible set of pre-strips *cum* strips is output from the MWC routine, we can sign these by the method in [17], and then submit them to standard rearrangement (GR) algorithms [15, 16, 39].

A large proportion of singletons, i.e., genes not in any pre-strip and hence not considered in the MWC analysis, are nonetheless usually compatible with the output of the MWC and we will be reincorporated them into the solution. There are different criteria for doing this, depending on the origins of the original data.

We will test our method on genomes with 100 genes partitioned among four chromosomes, subjecting them to a fixed series of random translocations and inversions before the addition of various levels of noise. Under conditions of moderate noise, the method can assign almost all genes to strips and recover the rearrangements. With increasing noise levels, the distance calculation remains surprisingly stable, though the original rearrangements can no longer be inferred. The distance calculation precedes the inference of the rearrangements.

In the succeeding sections, we will describe each step as follows:

i). Compile set of all maximal (inextensible) pre-strips. Augment set by adding certain non-maximal pre-strips (Section 5.2).

ii). Apply MWC, maximum weighted cliques [19], with weight = number of genes in a strip (Section 5.3).

iii). Assign gene orientation according to strip analysis [18, 17] (Section 5.4).

iv). Input into standard genome rearrangement (GR) algorithm (Section 5.5).

v). Restoration of singletons (Section 5.6).

vi). Simulations (Section 5.7).

As stated, Steps i - iii are exact for the problem of maximum weight strip analysis when singletons are disregarded. Step i is NP-hard, but is tractable when the number of pre-strips is fixed, and is very rapid for realistic data sets in any case. Step ii is NP-hard and is the rate-limiting step in our analysis. Step iii takes polynomial time and is linear if there are no two-term strips in the output of Step ii. Step iv is polynomial. There are three options for what we might like to achieve in Step v, depending on what is known about the origin of the noise, but we present exact solutions for all of them.

5.2 The pre-strips

Because mapping errors and other errors are likely to result in singleton genes in discordant genomic contexts in the two species being compared, our strategy is to rely entirely on strips to decide which data are erroneous and which are meaningful. But we have to start with structures that are more general than strips in two ways. Intervening error singletons may disrupt a strip, destroying contiguity, or a gene that should be in a strip may be itself erroneously mapped out of the strip, leaving its left and right neighbours adjacent in one of the genomes.

The common order (or its reverse) of the terms of a strip in the two unsigned genomes is

the only aspect of its structure we have access to. Thus we search for common subsequences in the two genomes as potential strips – pre-strips – relying on the subsequent analyses to eliminate the singletons and thus reveal the underlying strips. As a first step, then, we identify all maximal pre-strips by finding all inextensible common subsequences (and reversed order common subsequences) of the genes in all pairs of chromosomes, one chromosome from each genome. This is easily programmed to run in polynomial time, as long as the number of maximal pre-strips is a fixed parameter, or even when the number of such pre-strips is allowed to grow polynomially as a function of the number of genes.

Our goal is to find strips, i.e., to reduce the pre-strips to strips by eliminating extraneous genes from the chromosome that interrupt the contiguity of pre-strip genes. As shown in Figure 5.1, two maximal pre-strips may intersect, so that both cannot be present in a solution. However all or part of one and part of the other may be present, so that some non-maximal pre-strips must be considered in our search strategy. It suffices to use all contiguous fragments of a maximal pre-strip, as illustrated in Figure 5.1. (One singleton does not count as a pre-strip, but two singletons, successive terms of a pre-strip, do count.)

We need not consider any other subsequences of maximal pre-strips. To understand this, consider a1 a3 and a6 a9 from maximal pre-strip a1 a3 A1 a6 a9 in Figure 5.1. If anything intervenes to prevent A1 from forming a strip together with a1 a3 and a6 a9, it will necessarily also prevent a1 a3 and a6 a9 from forming a strip together. Our construction thus establishes the following:

Proposition 1: *All possible strips that can be formed by the deletion of genes from two genomes, are contiguous subsequences, containing at least two genes, of inextensible common subsequences (or their reverse) of two chromosomes, one from each genome.*

5.3 Maximum weight cliques

Once we have a set of pre-strips from which to construct our strips, we next need to construct a matrix of compatibilities among these pre-strips. Two pre-strips are incompatible if they share at least one gene or if either one contains a gene that destroys the contiguity of the genes in the other one. This definition entails:

> **Proposition 2:** *Given any set C of pairwise compatible pre-strips. Consider the reduced genomes produced by deleting all genes that are in none of the pre-strips in C. In these genomes all of the genes in each pre-strip in C appear as strip.*

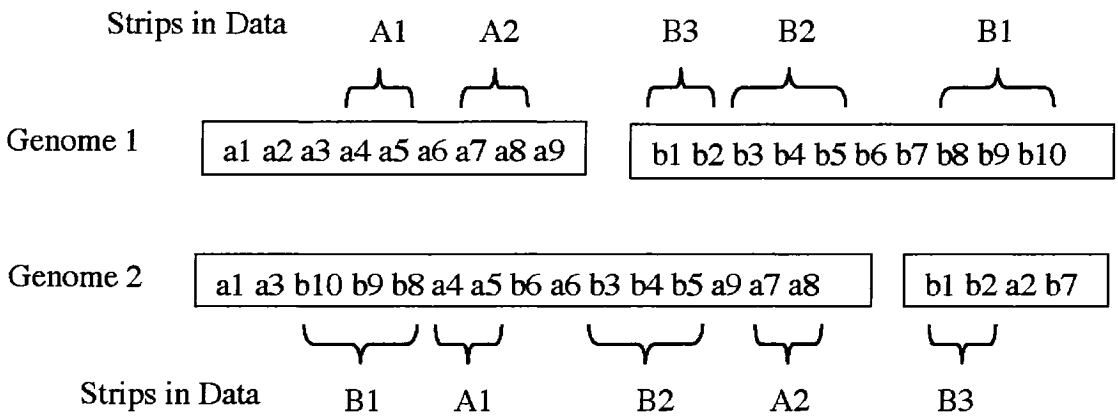
From the two original n-gene genomes, we wish to find a reduction, composed completely of strips, that minimizes the number d of genes deleted. From the compatibility matrix we construct a graph G with the pre-strips as the vertices and with compatible vertices joined by an edge.

> **Proposition 3:** *The solution C of the maximum weighted clique problem on G, where the weight w(i) on a pre-strip i is simply the number of genes it contains, induces a reduction of the original genomes so that they are composed completely of strips and so that the number of genes deleted is minimized.*
>
> *Proof:* Let $W = \sum_{i \in C} w(i)$ Then the statement follows from Propositions 1 and 2 are

$$d = n - W. \qquad (5.1)$$

Kumlander's algorithm [19] is based on a heuristic vertex coloring of sets of independent vertices, followed by two sorts of pruning of the clique search tree, one based on color classes and the other a backtrack search. Empirically, it was demonstrated to work better than competing algorithms on denser graphs. In the case of genome comparison, we would expect G to become dense, in the sense of [1], as the number of chromosomes

increases, as explained in Section 5.1.



Maximal Pre-strips	
a1 a3 A1 a6 a9	B1 b6 b3
a1 a3 A1 a6 A2	B1 b6 b4
a9 a7	B1 b6 b5
a9 a8	B2
	B3 b7

Additional Pre-strips	
a1 a3	B1 b6
a1 a3 A1	b6 b3
a1 a3 A1 a6	b6 b4
a3 A1	b6 b5
a3 A1 a6	B1
a3 A1 a6 a9	a3 A1 a6 A2
A1	A2
A1 a6	B3
A1 a6 a9	A1 a6 A2
a6 a9	a6 A2

Figure 5.1: Example of maximal pre-strips and additional pre-strips to be tested for compatibilty. Genome 1 and Genome 2 both contain two chromosomes. Any of the maximal pre-strips containing A1 intersects with any of those containing B1, and are hence incompatible with them. Any maximal pre-strips containing the same term, e.g., the three containing a9, are incompatible with each other.

In a preliminary implementation of this algorithm, we found that it tends to work well with hundreds of pre-strips but runs into computing time problems with thousands of vertices.

For the example in Figure 5.1, the algorithm gives the solution in Figure 5.2.

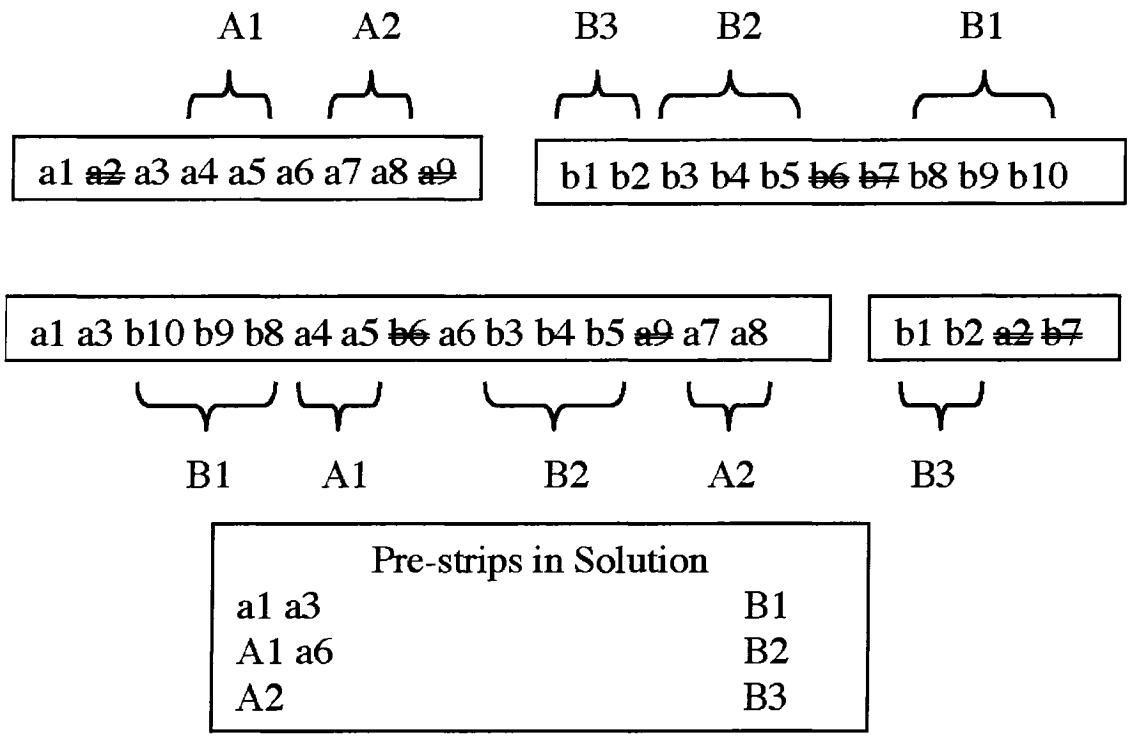


Figure 5.2: MWC solution of problem in Figure 1. Rejected singletons a2, a9, b6 and b7 crossed out. Weight =15 (out of 19 genes), distance d = 4.

5.4 Sorting unsigned genomes

The problem of adding signs to an unsigned genome so as to achieve a minimal reversal distance to the identity permutation 1, .., *n* is equivalent to the problem of sorting an unsigned permutation, known to be NP-hard [9].

However, as conjectured in [18] and proved in [17], for all segments of the permutation consisting of three or more consecutive integers (strips) in increasing order, plus signs can be given to all these integers, and for all decreasing strips, such as B1 in the example of Figures 5.1 and 5.2, minus signs can be given, and this assignment is consistent with a solution. In [17], it is also shown how to give signs to 2-strips in polynomial time. The algorithm these authors develop is exponential only in *s*, the number of singletons, and is

polynomial if *s* is *O(log n)*. In our case, however, s = 0, so sign assignment can be done in polynomial time.

5.5 Genome rearrangement

To infer the same rearrangements we used to generate the original data, we constructed all the cycles in the breakpoint graph of the two genomes [15, 43]. A summary statistic is provided by $r = \frac{2(b-c)}{b}$, the reuse rate, where b is the number of breakpoints in the comparison of the two genomes, and c is the number of cycles in their breakpoint graph. The addition of noise generally causes r to increase by adding proportionately more breakpoints than cycles and increasing the sizes of cycles. In the example of Figure 5.1 and 5.2, $b = 5$, $c = 1$, $d = 4$ and $r = 1.6$.

5.6 Restoration of singletons

There are at least three strategies that can be followed.

1. Restore all singletons that do not conflict with the strips in the MWC. This has the advantages that it is easy to identify these singletons and it includes a maximal amount of the original data in the final analysis. In contexts where the "noise" is due primarily to paralogy or lack of orientation information rather than error, it may, together with other data, such as gene homology measures, help determine relations of orthology and strandedness.

   At the same time, it has the disadvantage that in contexts where the "noise" is due primarily to mapping error, it does not reduce this noise to the full extent possible.,

since some of these errors may not conflict with evidence from correctly mapped genes. Moreover, it requires testing all MWCs, which may be numerous in some examples, and for each such clique to check all 2 s combinations of signs on the s eligible singletons.

With this method, once the singletons are restored, the GR algorithm may return an analysis that differs substantially from that based on the strips only. For example, in Figure 5.2, genes a2 and b6 conflict with the strips in the solution, and hence cannot be restored to the genomes. At the other extreme, the singleton b7 may be restored without materially affecting the rearrangement inference of three inversions and one translocation. The singleton a9 does not conflict with any strips in the solution, but its inclusion in the GR arrangement changes the solution, requiring an additional inversion. This is a disadvantage only if a9 represents erroneous information. If its singleton status is due to genomic processes, lack of information, or simply insufficient data, the GR analysis may give a more accurate result when such singletons are included.

2. Restore all singletons that do not substantially change the analysis based on the strips only, such as b7. This has the advantage that it only includes items that are maximally compatible with the strip-based analysis, so it should be more effective in excluding noise, while helping resolve lack of orientation. It does increase the amount of the original data included in the final analysis, but this may represent only a small proportion of the singletons. A clear difficulty with this approach is that it requires operationalizing the notion that two GR analysis based on two different data sets, one subsumed in the other, are substantially the same. For example, we could

ask that some optimal GR scenario for the larger set of genes, reduced by omitting all genes absent from the smaller set, also be optimal for the smaller set, taking into account rearrangements that disappear, such as inversions containing genes only in the larger set.

3. Restore no singletons. While this reduces noise maximally, it has the clear disadvantage that it takes into account none of the potentially informative singletons, especially when the noise is not simply mapping error. However, for our artificial (simulated) data, this is the most appropriate method. Only experience will tell which of the three methods is appropriate for particular real data sets.

We have implemented a program that permits any of these three options, but we present here simulations that test only the final option, generating data that are randomly noisy, without any attempt to model biological or methodological sources of error. Note that a global solution, treating singletons as if they were strips would require a different weighting system for MWC to avoid a trivial solution, and would also have to confront the NP-hardness of GR in this context.

5.7 Simulations

Prior to our simulations, we divided 100 genes approximately equally among four chromosomes, subjected them to four translocations and 13 inversions, with randomly chosen breakpoints.

For each replication, we add noise by randomly choosing one gene and moving it to a new, random, position in the genome. This models errors such as the mistaken identification of a gene in one genome to be orthologous to one in the other, when it is only a paralog of

the true ortholog. We repeat this process, eventually moving a total of 50 genes in a trial. We do 50 such replications in all.

In each trial, we apply our method after each of the 50 random changes and calculate the number of genes in the MWC solution and the distance between the genomes. In Figure 5.3 we show how the distance remains remarkably stable, despite the loss of over 40 % of the genome to noise. We might conjecture, subject to further investigation that the variance of the inferred distance increases with the amount of noise. More telling, we calculate the reuse rate (2 x distance / number of breakpoints), a measure of how random one genome is with respect to the other, and thus how much doubt should be thrown on the genome rearrangement analysis. Figure 5.4 shows a steady increase in reuse, eventually approaching 2.0, the maximum value, indicative of complete randomness. Although the distance remains more or less constant, the rearrangements inferred between the two genomes changes drastically as noise is added. Thus while our method seems to infer genome distance in a robust way, the details of the rearrangement scenario are obscured by increased noise levels.

5.8 Conclusions

The main contribution of this chapter is the conversion of the maximal weight strip problem to the MWC problem, based on the induced elimination of as few genes as possible from the genomes being compared. Our method recovers the underlying rearrangements while the noise level is low, but is degraded by higher levels. In our simulations, the total genomic distance, though not the actual rearrangements, remain fairly constant despite high levels of noise.

Though our pre-strip computation is not worst-case polynomial-time, in real problems it

is not the bottleneck. Nevertheless, we are now implementing a polynomial time algorithm for producing only 2- and 3- strips. Applying the MWC algorithm to these small strips, followed by piecing all overlapping ones together, gives the same solution as the present method.

The MWC itself is the bottleneck. Since our compatibility graph is dense, methods such as those introduced in [1] might speed up the MWC search.

It might be thought that with the advent of genome sequencing, comparative mapping without strandedness would become obsolete. In fact, the trend towards low-coverage sequencing without finishing, leaving many gene order ambiguities, makes it likely that physical and genetic mapping methods will continue to predominate, aside from very few model organisms.

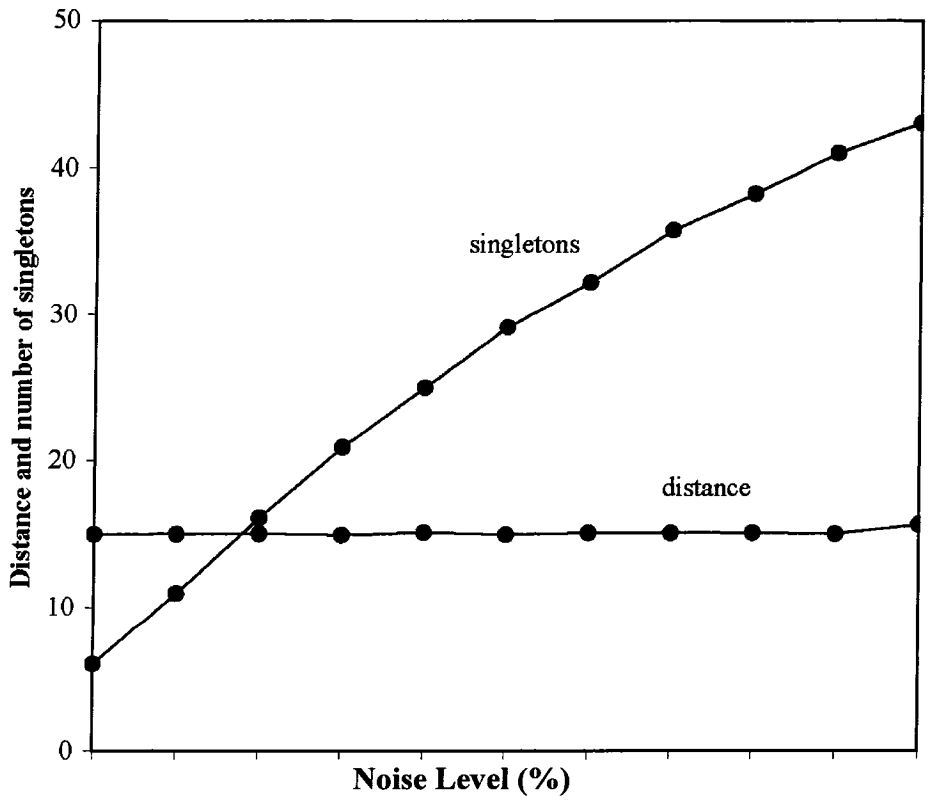


Figure 5.3: Simulation results: stability of genome distance despite the loss of large numbers of singletons.

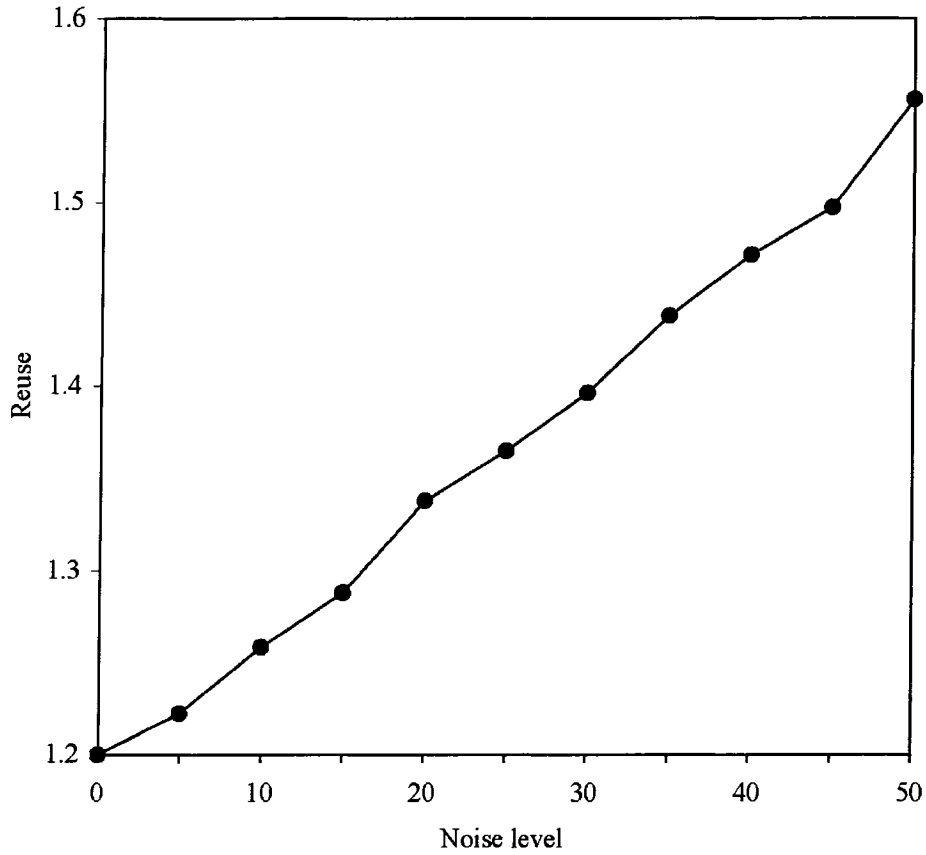


Figure 5.4: Increase in reuse as a result of noise

Chapter 6 Discussion

This thesis has been devoted to methods used to (1) calculate the genome distance for traditional comparative maps, (2) to estimate the missing biological information and (3) to try to pinpoint possible errors in the mapping data. We designed the algorithms as well as implemented them. Each chapter has included a discussion of the topic treated there. Summarizing, although some solutions are given for the problems of coarseness, missing data and mapping error, much improvement is needed. All the sub-problems in the comparative map problem (except for signed, totally ordered genomes) are NP-hard. Our program can only handle moderate amount of data. We need to optimize the branch and bound approaches for exact solutions as well as various heuristics, approximations and their statistical assessment when exact solutions are not feasible We need to improve the theoretical algorithmic results, not only for coarseness, missing data, and mapping error, but also for missing signs and paralogy, the most challenging parts of the problem. We also realize that the problems with comparative maps present themselves at the same time, so the best solution is to solve them simultaneously.

References

[1]. Arora, S., Karger, D. and and Karpinski, M. 1999. Polynomial time approximation schemes for dense instances of NP-hard problems. JCCS 58, pp. 193–210.

[2]. Bader, D. A., Bernard M.E. Moret, B. M. E. and Yan, M. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. Journal of Computational Biology 8, pp. 483–491.

[3]. Blin, G. and Rizzi, R. 2005. Conserved interval distance computation between nontrivial genomes. Proceedings of COCOON '05. LNCS, Springer, pp. 22-31.

[4]. Bourque, G., Pevzner, P.A. and Tesler, G. 2004. Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse, and rat genomes. Genome Research 14, pp. 507–516.

[5]. Bourque, G., Yacef, Y., and El-Mabrouk, N. 2005. Maximizing synteny blocks to identify ancestral homologs. Comparative Genomes 2005, pp. 21–34.

[6]. Bourque, G., Zdobnov, E., Bork, P., Pevzner, P. and Tesler, G. 2005. Comparative architectures of mammalian and chicken genomes reveal highly variable rates of genomic rearrangements across different lineages. Genome Research, 15, pp. 98-110.

[7]. Bowers, J. E., Abbey, C., Anderson, S., Chang, C., Draye, X., Hoppe, A. H., Jessup, R., Lemke, C., Lennington, J., Li, Z., Lin, Y. R., Liu, S. C., Luo, L., Marler, B. S., Ming, R., Mitchell, S.E., Qiang, D., Reischmann, K., Schulze, S. R., Skinner, D. N., Wang, Y. W., Kresovich, S., Schertz, K. F., Paterson, A. H. 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, pp. 367-386.

[8]. Bryant, D. 2000. The complexity of calculating exemplar distances.in D. Sankoff and J. Nadeau (eds), Comparative Genomics. Dordrecht, NL: Kluwer, pp. 207-212.

[9]. Caprara, A. 1997. Sorting by Reversals is Difficult. in S. Istrail, P. Pevzner, M. Waterman (eds.) Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB'97), ACM Press, pp. 75-83.

[10]. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S. and Jiang, T. 2005. Assignment of orthologous genes via genome rearrangement. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2, no. 4, pp. 302-315.

[11]. Christie, D. A. 1996. Sorting permutations by block interchanges. Information Processing Letters 60:165-169.

[12]. Eichler, Evan E. and Sankoff, D. (2003) Structural dynamics of eukaryotics chromosome evolution. *Science* 301, pp. 793-797

[13]. El-Mabrouk, N. and Sankoff, D. 2003. The reconstruction of doubled genomes. SIAM Journal on Computing 32, pp. 754-792.

[14]. Gaut, B. S. and Doebley, J.F. 1997. DNA sequence evidence for the segmental allotetraploid origin of maize. Proc. Natl. Acad. Sci. U S A. 94, pp. 6809-6814.

[15]. Hannenhalli, S. and Pevzner, P. 1995. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In Proc. 27th Annual ACM Symposium on the Theory of Computing, pp. 178-189.

[16]. Hannenhalli, S. and Pevzner, P.A. 1995. Transforming men into mice (polynomial algorithm for genomic distance problem. Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science, pp. 581–92.

[17]. Hannenhalli, S. and Pevzner , P. 1996. To cut or not to cut (applications of comparative physical maps in molecular evolution). in Proceedings of the 7th annual ACM-SIAM symposium on discrete algorithms, Philadelphia: SIAM, pp. 304-313.

[18]. Kececioglu, J. and Sankoff, D. 1993. Exact and approximation algorithms for the inversion distance between two permutations. in Proceedings of 4th Combinatorial Pattern Matching symposium, LNCS 684, Springer, pp. 87-105. (cf. Algorithmica, 13: 180-210, 1995.).

[19]. Kumlander, D. (2005) A new exact algorithm for the maximum-weight clique problem based on a heuristic vertex-coloring and a backtrack search. Fourth European Congress of Mathematics. Poster and manuscript.

[20]. 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 Molecular Biology 48, pp. 483-99.

[21]. Morgan, T. H., Sturtevant, A. H. , Muller, H . J., and Bridges, C.B. 1915. The mechanism of Mendelian heredity. New York: Henry Holt. and Co.

[22]. Nadeau, J.H. and Taylor, B.A. 1984. Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. 81, pp. 814-818.

[23]. NCBI Human Mouse Homology.http://www.ncbi.nlm.nih.gov/Homology/

[24]. Nicholas, F.W., Barendse, W., Collins, A., Darymple, B.P., Edwards, J.H., Gregory, S., Hobbs, M., Khatkar, M.S., Liao, W., Maddox, J.F., Raadsma, H.W. and Zenger K. R. 2004. Integrated maps and Oxford grids: maximising the power of comparative mapping. Poster at International Society of Animal Genetics.

[25]. Nguyen, C.T., Tay, Y.C. and Zhang, L. 2005. Divide-and-conquer approach for the exemplar breakpoint distance. Bioinformatics 21, pp. 2171-2176.

[26]. Palmer, J.D., Osorio, B. and Thompson, W.F. 1988. Evolutionary significance of inversions in legume chloroplast DNAs. Curr. Genet., 14, pp. 65-74.

[27]. Pevzner, P. and Tesler, G. 2003. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. Proc Natl Acad Sci USA 100, pp.7672–7677

[28]. Polacco, M.L.; Coe, E., Jr. 2002. IBM Neighbors: A Consensus GeneticMap. (http://www.maizegdb.org/ancillary/IBMneighbors.html)

[29]. Radcliffe, A.J., Scott, A.D. and Wilmer, R.E. 2005. Reversals and transpositions over finite alphabets. SIAM Journal on Disc. Math 19, pp. 224-244.

[30]. Sankoff, D. 1999. Genome rearrangement with gene families. Bioinformatics, 15, pp. 909-917.

[31]. Sankoff, D. and El-Mabrouk, N. 2000. Duplication, rearrangement and reconciliation. in Sankoff, D. and Nadeau, J. H., (eds) Comparative Genomics. Dordrecht, NL. Kluwer.

[32]. 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, pp. 6575-6579.

[33]. Sankoff, D., Zheng, C. and Lenert, A. 2005. Reversals of fortune. In McLysaght, A. and Huson, D. (ed), RECOMB 2005 Ws on Comparative Genomics, LNBI 3678. Berlin, Heidelberg:Springer Verlag, pp.131-141.

[34]. Sturtevant, A. H. 1913. The linear arrangement of six sex-linked factors in Drosophila, as shown by their mode of association. Jour. Exp.Zol. 14, pp. 43-59.

[35]. Sturtevant, A. H. 1921. Genetic studies on Drosophila simulans. II. Sex-linked group of genes. Genetics 6, pp. 43-64.

[36]. Sturtevant, A.H. (1921) A case of rearrangement of genes in Drosophila. Proc. Nat. Acad. Sci. 7, pp. 235-237.

[37]. Tang, J. and Moret, B.M.E. 2003. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. in WADS '03. LNCS 2748, pp. 37-46.

[38]. Tesler, G. 2002 GRIMM: genome rearrangements web server. Bioinformatics, 18, pp. 492-493.

[39]. Tesler, G. 2002. Efficient algorithms for multichromosomal genome rearrangements. Journal of Computer and System Sciences 65, pp. 587–609.

[40]. UCSC Genome Browser

[41]. 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 Research 30, pp. 103-105.

[42]. Watterson, G.A., Ewens, W.J., Hall, T.E., and Morgan, A. 1982. The chromosome inversion problem. J. Theor. Biol. 99, pp. 1-7.

[43]. Yancopoulos, S., Attie, O. and Friedberg, R. 2005. Effcient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics 21, pp. 3340 - 3346.

[44]. Zheng, C., Lenert, A. and Sankoff, D. 2005. Reversal distance for partially ordered genomes. Bioinformatics 21, pp. i502-i508.

[45]. Zheng, C. and Sankof D. 2005. Genome Rearrangements with partially ordered chromosomes. In L.Wang (ed.) Proceedings of COCOON 2005, LNCS, Springer, pp. 52-62.

[46]. Zheng, C. and Sankof D. 2005. Rearrangement of noisy genomes. Accepted by IWBRA06.