BIOINFORMATICS ORIGINAL PAPER Vol. 26 no. 13 2010, pages 1587–1594
doi:10.1093/bioinformatics/btq255

Genome analysis Advance Access publication May 18, 2010

Pathgroups, a dynamic data structure for genome reconstruction problems

Chunfang Zheng

Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Ancestral gene order reconstruction problems, including the median problem, quartet construction, small phylogeny, guided genome halving and genome aliquoting, are NP hard. Available heuristics dedicated to each of these problems are computationally costly for even small instances.

Results: We present a data structure enabling rapid heuristic solution to all these ancestral genome reconstruction problems. A generic greedy algorithm with look-ahead based on an automatically generated priority system suffices for all the problems using this data structure. The efficiency of the algorithm is due to fast updating of the structure during run time and to the simplicity of the priority scheme. We illustrate with the first rapid algorithm for quartet construction and apply this to a set of yeast genomes to corroborate a recent gene sequence-based phylogeny.

Availability: http://albuquerque.bioinformatics.uottawa.ca/pathgroup/Quartet.html

Contact: chunfang313@gmail.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on March 3, 2010; revised on May 11, 2010; accepted on May 12, 2010

1 INTRODUCTION

Many comparative genomic problems require the reconstruction of unknown genomes, more specifically their gene orders, starting from knowledge of one or more given, contemporary, genomes. These include the inference of ancestral genomes as part of the small phylogeny problem (Sankoff and Blanchette, 1998), and its archetypical cases, the median (Sankoff and Blanchette, 1997) and quartet (Liu *et al.*, 2005) problems, based on three or four given genomes, respectively. These also include the genome halving (El-Mabrouk and Sankoff, 2003) and genome aliquoting (Warren and Sankoff, 2009) problems, based on a single genome where every gene is duplicated or in a gene family of size $m \geq 2$; the idea being to infer the immediate pre-polyploid ancestor. There are various hybrid problems, such as guided genome halving (GGH; Sankoff *et al.*, 2007; Zheng *et al.*, 2006), which combines genome halving with phylogenetic inference, or small phylogeny on unequal genomes (Tang and Moret, 2003), which integrates missing data considerations.

In addition, there are also problems where one or more current, rather than ancestral, genomes, lacking full gene order (Tang and Moret, 2003; Zheng and Sankoff, 2006), gene orientation (Hannenhalli and Pevzner, 1996), or affected by paralogy (Tannier, 2009) or high levels of gene order error, are to be completed or corrected based on comparative evidence. We do not study this latter group of problems here, having previously shown how to leverage the minimization inherent in rearrangement distance definitions in order to optimally complete or correct the genomes (Choi *et al.*, 2007; Sankoff *et al.*, 2005; Zheng *et al.*, 2007).

The computational status of most of the ancestral reconstruction problems have been reviewed in Tannier *et al.* (2009) and in a recent textbook (Fertin *et al.*, 2009). With few exceptions (such as genome halving or multichromosomal breakpoint median), they are NP hard problems, especially the more realistic versions. Nevertheless, with high demand from the phylogenetics community (Murphy *et al.*, 2005; Soltis *et al.*, 2009), a great variety of exact and heuristic algorithms have been developed, some of which are in wide use (Bourque and Pevzner, 2002; GRAPPA, 2004).

At the heart of many of these methods, especially those that find, or at least seek, a most economical solution in terms of genomic distances is the strategy of maximizing the number of cycles in the breakpoint graph (Caprara, 2001; El-Mabrouk and Sankoff, 2003; Siepel, 2001) [or its dual, the adjacency graph (Bergeron *et al.*, 2006)], while reconstructing the unknown genome. In this article, we propose a data structure that is designed entirely for this type of strategy. *Pathgroups* is a compact and flexible way of storing partially completed cycles, so that genome-wide greedy searches (allowing look-ahead strategies and problem-specific constraints) are rapidly executed and the database rapidly updated. *Pathgroups* is readily adapted to virtually all the gene-order reconstruction problems and indeed its virtues can be appreciated by studying the basic steps in such procedures as diverse as genome halving (El-Mabrouk and Sankoff, 2003) and the median problem (Siepel, 2001). The procedure handles constraints on the reconstructed genomes (e.g. exact tetraploidy in genome halving problems) efficiently. A key advantage of the method is that its running time depends only on genome size and not the rearrangement distances among the input genomes, while run time of other reconstruction methods are highly dependent on the distance.

In this article, after formalizing a number of ancestral reconstruction tasks, we present the basic structure of pathgroups, and show how it is adapted to all of the problems. In particular, we use it in a new heuristic to efficiently search for a solution to the quartet problem. As applied to a number of yeast gene orders, we show that gene order data confirm the phylogeny previously obtained from gene sequence data.

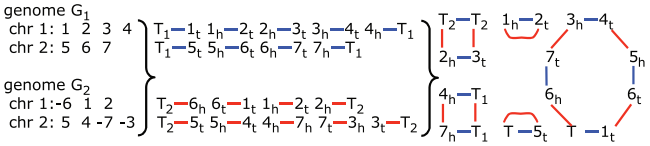

Fig. 1. Construction of the breakpoint graph. Left: genomes G_1 and G_2, with '-' sign indicating negative polarity. Middle: vertices and edges of individual genome graphs. Right: cycles in completed breakpoint graph.

2 THE PROBLEMS

In this section, after establishing basic concepts and definitions, we sketch four ancestral genome reconstruction problems with reference to Figure 2 which depicts them all.

2.1 Preliminaries

2.1.1 Genomes and rearrangement operations A genome can be modeled as a set of chromosomes, each chromosome consisting of a number of linearly ordered elements called genes. The genes are all distinct and each one has positive or negative polarity. Biologically, the polarity indicates on which of the two DNA strands the gene is located or, equivalently, if the gene is transcribed from left to right or from right to left.

Genomes can be rearranged, i.e. the order of the genes and their polarity, which chromosome contains a given gene, the total number of chromosomes, can all change, through the accumulated operation of number of classical processes familiar in classical genetics: inversion, reciprocal translocation, transposition, chromosome fusion and fission. We will not delve into the details of these rearrangement operations; they can all be subsumed under a single operation called double-cut-and-join, which need not be described here. All that is needed for our purposes is a formula due to Yancopoulos *et al.* (2005), stated in Section 2.1.2, that gives the minimum number of rearrangement operations needed to transform one genome into another, in terms of properties of the 'breakpoint graph' determined by the initial and final genomes.

2.1.2 Rearrangement distance The genomic distance $d(G_1,G_2)$ is a metric counting the number of rearrangement operations necessary to transform one multichromosomal gene order G_1 into another G_2, where both contain the same n genes. To calculate D efficiently, we use the breakpoint graph of G_1 and G_2, constructed as illustrated in Figure 1.

For each genome, each gene g with a positive polarity is replaced by two vertices representing its two ends, i.e. by a 'tail' vertex and a 'head' vertex in the order g_t, g_h; for $-g$ we would put g_h, g_t. Each pair of successive genes in the gene order defines an adjacency, namely, the pair of vertices that are adjacent in the vertex order thus induced. For example, if $i, j, -k$ are three neighboring genes on a chromosome then the unordered pairs $\{i_h, j_t\}$ and $\{j_h, k_h\}$ are the two adjacencies they define.

If there are m genes on a chromosome, there are $2m$ vertices at this stage. The first and the last of these vertices are called telomeres. We convert all the telomeres in genome G_1 and G_2 into adjacencies with additional vertices all labeled T_1 or T_2, respectively. The breakpoint graph has a blue edge connecting the vertices in each adjacency in G_1 and a red edge for each adjacency in G_2. We make a cycle of any path ending in two T_1 or two T_2 vertices, connecting them by a red or blue edge, respectively, while for a path ending in a T_1 and T_2, we collapse them to a single vertex denoted 'T'.

Each vertex is now incident to exactly one blue and one red edge. This bicolored graph decomposes uniquely into κ alternating cycles. If n' is the number of blue edges, then (Yancopoulos *et al.*, 2005):

$$d(G_1,G_2) = n' - \kappa. \quad (1)$$

2.2 The median problem

Let G_1, G_2 and G_3 be three genomes on the same set of n genes. *The rearrangement median problem* 'MED' *is to find a genome M such that $d(G_1,M)+d(G_2,M)+d(G_3,M)$ is minimal.* We may also define a median problem for more than three genomes $G_1, G_2, \ldots, G_\nu$ as finding M so that $d(G_1,M)+d(G_2,M)+\cdots+d(G_\nu,M)$ is minimized.

2.3 The quartet problem

Let G_1, G_2, G_3 and G_4 be four genomes on the same set of n genes, with a given topology: genomes G_1 and G_2 are adjacent to the same vertex M_1 and G_3 and G_4 are adjacent to the same vertex M_2. In addition, M_1 is adjacent to M_2. *The quartet problem* 'QRT' *is to reconstruct genomes M_1 and M_2 such that $d(G_1,M_1)+d(G_2,M_1)+d(G_3,M_2)+d(G_4,M_2)+d(M_1,M_2)$ is minimal.*

2.4 GGH

Let T be a genome consisting of ψ chromosomes and $2n$ genes $a_1^{(1)},\ldots,a_n^{(1)};a_1^{(2)},\ldots,a_n^{(2)}$, dispersed in any order on the chromosomes. For each i, we call $a_i^{(1)}$ and $a_i^{(2)}$ 'duplicates', but there is no particular property distinguishing all elements of the set of $a_i^{(1)}$ in common from all those in the set of $a_i^{(2)}$. A potential 'doubled ancestor' of T is written $A' \oplus A''$, and consists of 2χ chromosomes, where some half (χ) of the chromosomes, symbolized by A', contains exactly one of $a_i^{(1)}$ or $a_i^{(2)}$ for each $i=1,\ldots,n$. The remaining χ chromosomes, symbolized by A'', are each identical to one in the first half, in that where $a_i^{(1)}$ appears on a chromosome in the A', $a_i^{(2)}$ appears on the corresponding chromosome in A'', and where $a_i^{(2)}$ appears in A', $a_i^{(1)}$ appears in A''. We define A to be either of the two halves of $A' \oplus A''$, where the superscript (1) or (2) is suppressed from each $a_i^{(1)}$ or $a_i^{(2)}$. These χ chromosomes, and the n genes they contain, $a_1,\ldots,a_n$ constitute a potential 'doubled ancestor' of T. (Note that ascribing $2n$ genes on 2χ chromosomes to a tetraploid is consistent with only n genes on χ chromosomes in a diploid. As far as gene order is concerned, there is no need to distinguish between maternal and paternal chromosomes).

The genome halving problem for T is to find an A for which some $d(A' \oplus A'', T)$ is minimal.

The genome halving problem can be solved in linear time (El-Mabrouk and Sankoff, 2003). Unfortunately, the algorithmic result suffers from severe non-uniqueness. To overcome this problem, a reference genome, or outgroup, is introduced. Let T be a genome consisting of ψ chromosomes and $2n$ genes $a_1^{(1)},\ldots,a_n^{(1)};a_1^{(2)},\cdots,a_n^{(2)}$, dispersed in any order on the chromosomes, where for each i, genes $a_i^{(1)}$ and $a_i^{(2)}$ are duplicates. Any genome R is a reference or outgroup genome for T if it contains the n genes $a_1,\ldots,a_n$.

Let R be a reference genome for T. The GGH *problem with one outgroup is to find a potential ancestral genome A such that some* $d(R,A)+d(A'\oplus A'',T)$ *is minimal. Let* R_1 *and* R_2 *be two reference genomes for T. The GGH problem with two outgroups is to find a potential ancestral genome A and a median genome M such that some* $d(R_1,M)+d(R_2,M)+d(M,A)+d(A'\oplus A'',T)$ *is minimal.*

2.5 Genome aliquoting

The genome aliquoting problem 'ALQ' (Warren and Sankoff, 2009) is a generalization of the genome halving problem. Whereas the genome halving problem reconstructed the 'doubled ancestor' of a genome which contains exactly two copies of each gene, genome aliquoting extends the scope from two copies of duplicates to m copies of paralogs in a genome. Let genome P (for "polyploid") consist of mn $(m>2)$ genes $a_1^{(1)},\ldots,a_n^{(1)};a_1^{(2)},\ldots,a_n^{(2)};\ldots;a_1^{(m)},\ldots,a_n^{(m)}$. *The genome aliquoting problem for P is to find an A for which some* $d(A_1\oplus A_2\oplus\cdots\oplus A_m,P)$ *is minimal.*

3 ALGORITHM

All the problems listed in Section 2 are known to be NP hard, with the possible exception of aliquoting, whose status is not settled. Our method will be shown to run in linear time, so obviously it is not guaranteed to find an exact solution. However, its rapidity makes it an ideal way to obtain reasonably accurate first approximations for large-scale instances and good initializations for iterative methods when there are two or more ancestral genomes to be reconstructed.

3.1 General solution

3.1.1 Paths and fragments We generalize our definition of a path to be any connected subgraph of a breakpoint graph, namely, any connected part of a cycle as previously defined in Section 2.1.2 and Figure 1, where the two endpoints are incident to blue edges. We represent each path by an unordered pair of vertices $(u,v)=(v,u)$ consisting of its current endpoints, though we keep track of all its vertices and edges. Initially, each blue edge in the given genomes is a path.

A *fragment* is any set of genes connected by red edges in a linear order, i.e. there are two gene tails and/or heads not connected to any other vertex. The set of fragments represents the current state of the reconstruction procedure. Initially, the set of fragments contains all the genes, but no red edges, so each gene is a fragment by itself.

3.1.2 Pathgroups The objective functions for each of the ancestral reconstruction problems consists of the sum of a number of genomic distances, e.g. three distances for the median of three genomes, five distances for quartet construction, just one for genome halving or aliquoting and two for GGH. Each of these distances corresponds to a breakpoint graph. A given genome determines blue edges in one breakpoint graph, while the red edges correspond to the ancestral genome being constructed. For each such ancestor, *the red edges are identical in all the breakpoint graphs corresponding to distances to that ancestor.* In problems such as halving or aliquoting, there may be two or more identical copies of each red edge in breakpoint graphs involving a polyploid ancestor.

A pathgroup is a set of $r\geq 3$ paths, one or more from each partial breakpoint graph currently being constructed. For a given problem, all the pathgroups contain the same number r of paths. There is one pathgroup for each non-T vertex and for each ancestor. We do not construct a pathgroup for each T vertex separately, though paths ending in T vertices are found in pathgroups determined by non-T vertices. This approach was chosen for reasons of efficiency and simplicity, as will be explained in Section 3.1.4. In the median of $v\geq 3$ genomes problem, each breakpoint graph contributes one path to each pathgroup, so $r=v$. In the GGH problem, the tetraploid contributes two paths and the reference genome one, so $r=3$. In genome aliquoting based on m-gene families, each pathgroup contains $r=m$ paths, all from the single breakpoint graph constructed in this problem.

In the quartet problem, there are separate pathgroups pertaining to the two ancestral genomes. As with the median problem for three genomes, there are (potentially) $r=3$ paths in the pathgroup. However, one of the paths in the pathgroup determined by a vertex x may be missing, since the blue edge incident to x in the breakpoint graph of the two ancestral genomes may not have been drawn yet. In this case, we simply place x in the pathgroup instead of the missing path. Moreover, paths from this breakpoint graph may begin or end with red edges.

Pathgroups overlap because most paths are in two pathgroups, one associated with its initial vertex and one with its final vertex, unless the latter is a T. With respect to a given path xy, we say the pathgroup determined by vertex x is the *partner* of the pathgroup determined by y. A pathgroup may have up to r distinct partners.

3.1.3 Priorities Our main algorithm presented in Section 3.1.4 aims to construct one or more breakpoint graphs with a maximum aggregate number of cycles. At each step it adds one red edge, the same red edge, to each of the paths in the pathgroup, thus changing one or more breakpoint graphs. This removes two (partner) pathgroups, turning one or more of their paths into cycles and concatenating each of their remaining paths with a path in some other pathgroup. We do not add red edges incident to T vertices.

It is always possible to create one cycle, at least (as long as not all the paths in the pathgroup contain a T) by adding a red edge between the two ends of any one of the paths. The strategy is to create as many cycles as possible. Among alternate choices of steps creating the same number of cycles, to choose the one that sets up the best configuration for the next step.

Thus, the pathgroups are prioritized,

(1) first by the maximum number of cycles that can be created within the group, without giving rise to circular chromosomes; and

(2) second, for those pathgroups allowing equal number of cycles, by considering the maximum number of cycles that could be created in the next iteration of Step 1, in any one pathgroup affected by the current choice.

A pathgroup may receive no priority, if creating any cycle within the pathgroup necessarily creates a circular ancestral chromosome. Note that in adding a red edge xy, this causes not only the disappearance of two partnered pathgroups, but it also changes paths in other pathgroups, which we call *secondary* pathgroups. Furthermore, each secondary pathgroup may itself have partner pathgroups whose paths, though not affected by the addition of xy, may have changed priorities. We call these *tertiary* pathgroups.

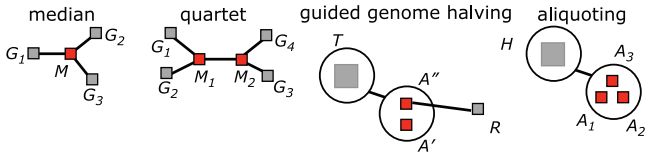

Fig. 2. Representation of four reconstruction problems, including the median M of three genomes G_1, G_2 and G_3, quartet construction on $G_1 \ldots G_4$, including ancestors M_1 and M_2, genome halving of tetraploid T to identical diploids A' and A'' guided by reference diploid genome R, and aliquoting of a hexaploid H to three diploids A_1, A_2 and A_3. Gray squares indicate given genomes, red squares those to be reconstructed. Each line connecting two genomes represents a breakpoint graph and a distance.

For the MED ($\nu = 3$), GGH, QRT and ALQ ($m = 3$) problems, the pathgroups and the priorities are basically the same, as illustrated in Figure 2.

3.1.4 The **makeCycles** *algorithm* By maintaining a list of pathgroups for each priority level, and a list of fragment endpoint pairs (initial and final), together with appropriate pointers, the algorithm **makeCycles** requires $O(n)$ running time.

Algorithm makeCycles
input: pathgroups each consisting of r blue one-edge paths
output: ancestral genome
while: there remain pathgroups with priorities

(1) add red edge to pathgroup of highest priority, creating at least one cycle, thus deleting this pathgroup and its partner.
(2) update the paths in the secondary pathgroups affected by the addition of the red edge, and update the red fragment extended by this edge or created by the joining together of two existing red fragments.
(3) update the priorities of the secondary pathgroups, the tertiary pathgroups and the at most two pathgroups associated with the endpoints of the red fragment extended or created in Step 2.

Note that we are not inferring that all the red fragments output by **makeCycles** are complete chromosomes of the ancestral genome; some of them may just be chromosome fragments, depending on the particular instance.

PROPOSITION. *For pathgroups of size r, adding a red edge xy in a pathgroup creates at most* $2(r-1)$ *secondary pathgroups and at most* $2(r-1)^2$ *tertiary pathgroups.*

PROOF. The paths in the pathgroups for x and y contain no more than $2(r-1)$ endpoints that are not x or y, so there can be no more than $2(r-1)$ secondary pathgroups. A secondary pathgroup determined by vertex z contains no more than $r-1$ paths that are not x, y or z, so that there are at most $2(r-1)^2$ tertiary pathgroups. ■

The proposition, together with the two facts:

(1) the total number of pathgroups decreases by two by each step; and
(2) the calculation or recalculation of the priority of each pathgroup requires constant time;

ensure the $O(n)$ running time of the algorithm. Note that if we had allowed red edges to connect T vertices or if we had allowed pathgroups determined by T vertices, the number of potential secondary and tertiary pathgroups affected by the addition of a red edge would have depended strongly on the number of chromosomes in the genomes.

In the remainder of this section, we will discuss how this algorithm applies to the MED, GGH, QRT and ALQ problems, with reference to Figure 3, which details the different priorities for pathgroups, and the origin of each path in the pathgroup in the appropriate breakpoint graph. An example of how the algorithm works is presented in the Supplementary Material.

3.2 Application to specific problems

3.2.1 Median As illustrated in Figure 3, there are seven priority levels, corresponding to constructing

(1) three cycles,
(2) two cycles setting up (a) three, (b) two or (c) one in the look-ahead or
(3) one cycle setting up (a) three, (b) two or (c) one in the look-ahead.

Note that when a red edge is defined, the pathgroup is emptied, either by the creation of cycles, or by the integration of x as a non-endpoint of some path.

For the median of ν genomes, the number of priority levels is $1+\nu(\nu-1)$.

3.2.2 GGH Recall that the pathgroups for GGH contain three paths, two from the tetraploid and one from the reference diploid. As illustrated in Figure 3, the pathgroup structure assures that as a red edge is added, two identical red fragments are extended in the same way. The reconstructed genome will thus eventually consist of two identical sets of chromosomes, a wholly doubled genome.

We may use the notation x and $\bar{x}$ for the two copies of a gene, although this is not indicated in the general scheme of Figure 3. At the outset, it is inconsequential which copy is labeled x and which is labeled $\bar{x}$. The vertex x may be linked with y or with $\bar{y}$, and this does have consequences for the breakpoint graph, including the number of cycles. The two alternatives must be checked each time a priority is calculated, to see which one gives the better priority.

3.2.3 Genome aliquoting The pathgroups for the genome aliquoting problem with mn genes are similar to those for the median problem for $\nu = m$ genomes. The main difference is the blue edges all come from a single polyploid genome instead of ν separate genomes. Moreover, there is a complication in calculating the priorities. Although it is clear how many cycles are formed when adding a red line to a pathgroup, the look-ahead may have to search among several possibilities of joining paralogs together to find the maximum number of cycles that could be produced.

For example, if $x, \bar{x}$ and $\bar{\bar{x}}$ are the three copies of gene x, consider the case where the path (x, y) is turned into a cycle by the addition of a red edge, but the other two paths in the pathgroup are (x, w) and (x, z). Then (x, w) may be joined by a red edge to $(\bar{y}, s)$ creating path (w, s), while (x, z) may be joined by a red edge to $(\bar{\bar{y}}, t)$ creating path (z, t). But we could also construct (w, t) and (z, s) instead; we must

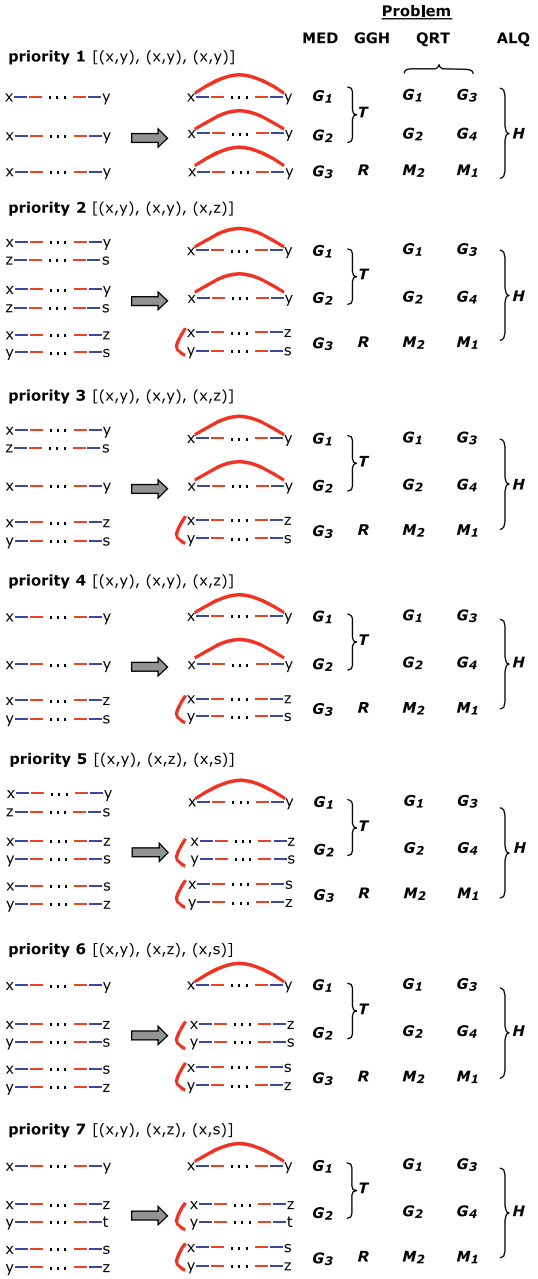

Fig. 3. Priorities of all pathgroups of form $[(x,a),(x,b),(x,c)]$ for inserting red edges, in four reconstruction problems. Includes sketch of three paths in 'x' pathgroup plus other paths involved in calculating priority. For example, completing the pathgroup $[(x,y),(x,y),(x,z)]$ by adding the red edge xy always produces two cycles, but can set up a pathgroup with 3 potential cycles (priority 2), 2 potential cycles (priority 3) or 1 potential cycles (priority 4).

consider both alternatives in calculating the priority of the pathgroup determined by x.

3.2.4 Quartets In the quartet problem, there are 14 pathgroups shown in Figure 3, seven for adding red edges to paths in the median M_1 between G_1, G_2 and M_2 and seven for adding red edges to paths in the median M_2 between G_3, G_4 and M_1. When red edges are added to M_1, the edges in M_1 are considered as blue paths, and vice versa.

As with the median problem for three genomes, there are (potentially) three paths in the pathgroup. However, one of the paths in the pathgroup determined by x may be missing, since the blue edge incident to x in the breakpoint graph of the two ancestral genomes may not have been drawn yet. In this case, we simply place x in the pathgroup instead of the missing path. Moreover, paths from this breakpoint graph may end with red edges. These aspects are very important, as will be detailed in Section 5.1. Note that when a red edge is added in a pathgroup for M_1, this becomes a blue edge in a pathgroup for M_2, and vice versa.

4 IMPLEMENTATION

Versions of the pathgroups algorithms have been implemented before, for the median problem (Muñoz and Sankoff, 2009) and the guided halving problem (Sankoff *et al.*, 2008; Zheng *et al.*, 2008), but without any attempt to achieve the computational efficiencies available with this approach.

In the present work, we implemented **makeCycles** so that it could achieve its worst case linear run time capability. The particular problem we focused on was quartet construction, with two $r=3$ pathgroups, one for each ancestor, for each vertex x in the breakpoint graph, as in the QRT column of Figure 3.

4.1 Efficiency

We carried out a number of simulation experiments to establish the average run time performance.

Each data point in our experiments comes from the average of 100 simulations. All simulations started with 20 chromosomes, and rearrangements were apportioned as 90% inversions and 10% reciprocal translocations. For each rearrangement, chromosomes were chosen at random and breakpoints within the chromosome were chosen at random. The initial genome at M_1 was used to generate G_1, G_2 and M_2 independently with genome size n and d rearrangements on each branch. Then the genome at M_2 was used to generate G_3 and G_4 in the same way.

Figure 4 shows the result of four experiments. One where run time t is plotted against n varying from 1000 to 10 000, while $d=1000$. Another is the same except $d=25000$. The other two experiments set $d=n/2$ and $d=2n$, respectively. Except for the first experiment, the average behavior of t as a function of n appears the same. In other words, except for low values of d the run time of the **makeCycles** routine is independent of d, depending only on n. Moreover, above $n=5000$ the average run time is close to linear.

Figure 5 highlights the fact that run time does not depend on d for moderate and large values of d.

4.2 Questions of accuracy

As mentioned in Section 3, we do not expect to guarantee exact solutions for NP hard problems in linear time. Moreover, because this is a single-pass method, we cannot even expect to find locally optimal solutions. Thus, we must investigate how close is the approximation and what are the prospects for improvement.

In the same experiment as Figure 5, we measured the total distance of the quartet by summing the edge distances using the simulated ancestors constructed by random rearrangement. We then

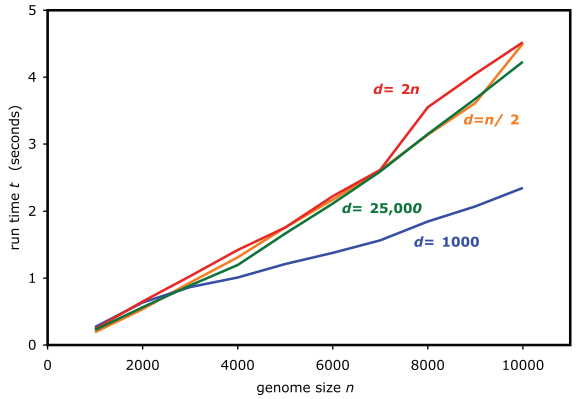

Fig. 4. The near-linear dependance of average run time on the number of genes in a genome. This characteristic is achieved after $n=5000$. Note that with genomic distance $d=1000$, the slope is less than with $d=25000$, but there is little difference among the curves for $n=25000$, $d=n/2$ and $d=2n$.

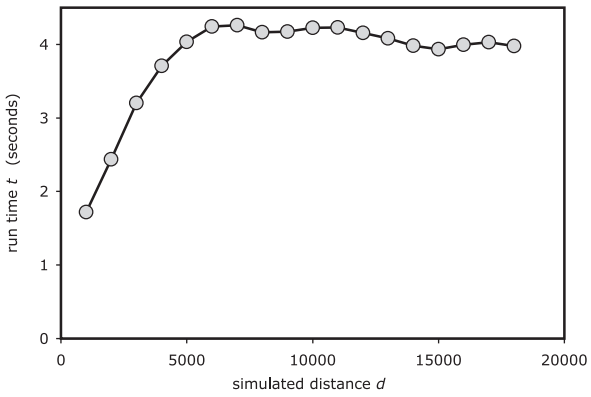

Fig. 5. The lack of dependence of run time on d, for $d>5000$. Here $n=10000$ for all simulations.

compared this with total distance using the ancestors we inferred with **makeCycles**. The results appear in Figure 6.

That both curves are concave is no surprise. By definition, the inferred distance between two genomes is less than or equal than the number of rearrangements used to derive one from the other during simulation. And it is known that this difference increases as the number of rearrangements increases; indeed the inferred distance cannot be greater than n while the number of rearrangements that can be applied is not bounded. Nor is it surprising that the total based on the inferred ancestors, which are chosen to minimize the total is smaller than that based on the simulated ancestors, which are fixed. What is of concern is the interval below $d=5000$, where the distances based on the inferred ancestors is marginally greater than the total based on the fixed ancestors. This clearly reflects some degree of imprecision of this method for this problem.

Whether the tradeoff between speed and precision is worth, it depends on the particular application. And the pathgroups solution may be useful as an initialization of an iterated method where the median algorithm is applied alternatively to M_1 and M_2 until no further improvement is obtained.

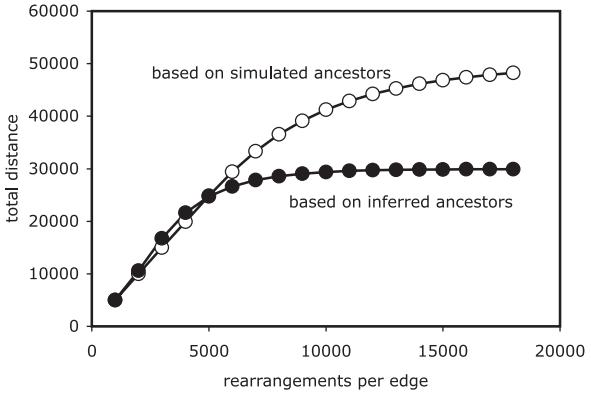

Fig. 6. Total simulated distance and total reconstructed distance.

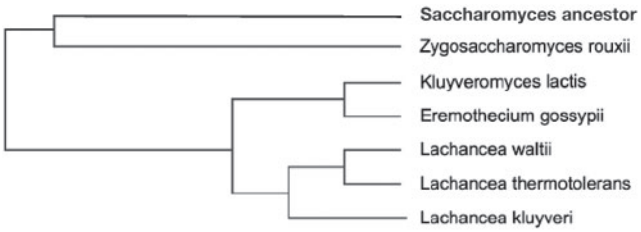

Fig. 7. Phylogeny of seven yeast genomes (Hedtke *et al.*, 2006), including *Saccharaomyces* ancestor before whole-genome duplication.

4.3 Quartets of rapidly and slowly evolving yeasts

Saccharomyces cerevisiae and its closest relatives are descendants of a whole-genome duplication event more than 100 million years ago (Byrne and Wolfe, 2005). In a maximum likelihood phylogeny of 79 yeast species inferred from eight gene sequences (Hedtke *et al.*, 2006), there are six relatives of *S.cerevisiae* whose genomes have been sequenced, but that diverged before the whole gene duplication. These six, plus the manually reconstructed ancestral genome (Gordon *et al.*, 2009) that underwent whole-genome duplication are depicted in Figure 7. Since their divergence these genomes have evolved at very different rates, with *Eremothecium gossypii*, for example, showing a substitution rate over three times as great as *Lachancea Kluyveri*, making phylogenetic inference prone to various biases.

We extracted gene orders involving the 4011 sets of orthologous genes that these genomes all have in common from the Yeast Gene Order Browser (Byrne and Wolfe, 2005). Gene order rearrangement distances between them shows that the evolution rate for gene order varies in much the same way as for gene sequence, with *E.gossypii*, for example, changing gene order much more rapidly than *L.kluyveri*. We applied our quartet construction algorithm to all 35 subsets of four genomes. Each of these subsets may be arranged in three different quartets, for a total of 105 quartets (and 30 s of computing time). Only one of the three can be consistent with a given tree. For the tree in Figure 7, the least total distance quartet was consistent with this tree in 34 out 35 subsets, which represents a very high degree of confirmation.

5 DISCUSSION

5.1 Small phylogeny

Median solving and quartet building are the prototypical small phylogeny problems, and the special nature of the pathgroups for the quartet problem hints at the solution of the small phylogeny problem. Indeed, with appropriate calculation of priorities, the **makeCycles** algorithm requires no extensions or modifications to apply to this general problem.

There should be an entire set of pathgroups for every internal (ancestral) node of a small phylogeny. Consider 'binary' topologies, where each given genome is connected to one ancestral genome, and all ancestral genomes are adjacent to three genomes. Initially, the pathgroups for those ancestral nodes connected to two given genomes should have the same form as those for M_1 and M_2 in the quartet problem—one of the paths missing and replaced by a single vertex of the breakpoint graph. Those ancestral nodes connected to only one given genome will have two missing paths in each pathgroup, both replaced by the vertex. Finally, those ancestral nodes connected only to other internal nodes will have all paths missing in each pathgroup, all replaced by the vertex.

As the algorithm executes, all the pathgroups associated with all the internal nodes are scanned at each step to find the one with the highest priority. The pathgroups connected to two given nodes will tend to be processed first, building up all three paths and combining the pathgroups one by one. Each time a red edge is added to a path, this becomes a blue edge in the corresponding pathgroup for ancestral genome(s) connected to it.

Eventually even the nodes furthest from any given genomes will accumulate enough edges in their pathgroups so that cycles can be formed and so that fragments of the associated genomes begin their reconstruction.

It is clear from the structure of binary trees that the algorithm will continue until all the ancestral genomes are constructed. These considerations carry over to trees where the internal nodes have degree greater or equal to three. Higher degree nodes have more paths in their pathgroups, but the same principle should be used to establish priorities: the greater the number of cycles that can be created, the higher the priority.

5.2 Impact and future work

The main interest of the pathgroups approach is its applicability to a wide range of reconstruction problems. Ultimately all these problems may be considered to be part of the small phylogeny gene order problem, where polyploidization events are allowed.

The virtues of our heuristic are its flexibility, simplicity and speed. Of course, being a one-pass algorithm, where all information flows from the given genomes to the most ancestral ones, with no traceback, it cannot always find solutions that are as good as the common iterative approach originating with (Sankoff and Blanchette, 1997, 1998). Methods like these and others derived from them (Adam and Sankoff, 2003) iteratively traverse the overlapping medians constituting the small phylogeny in the search for improvements.

Nevertheless, it is known that good initialization for the ancestral genomes is the key to finding good solutions, and our method represents a rapid and relatively accurate way of providing such initialization. Moreover, the pathgroups approach is perfectly suited to iterative improvement of overlapping medians or quartets.

The formulation of our problems and the method has been situated in the context of genomes with multiple linear chromosomes, i.e. modeling eukaryotic nuclear genomes. Indeed the halving and aliquoting problems are only really pertinent in this context. This explains our attention to avoiding circular fragments in the algorithm. There would be no additional difficulty in reformulating and solving the median, quartet and small phylogeny problems allowing genomes to consist of one (or more) circular chromosomes.

Finally, enhancing the accuracy of our method could start with more elaborate priority schemes, without sacrificing the linear dependence of run time of genome size or its independence of gene order rearrangement distances. At the same time, we could allow additional pathgroups, those determined by T vertices. This would tend to increase the accuracy by creating more high priority pathgroups, and would not substantially slow the algorithm for a reasonable number of chromosomes.

ACKNOWLEDGEMENTS

I thank David Sankoff for suggesting this research and Nadia El-Mabrouk for her support. I also thank Victor A. Albert and Ann Stapleton for encouraging the development of the software.

Funding: Postdoctoral fellowship from the Natural Sciences and Engineering Research Council of Canada.

Conflict of Interest: none declared.

REFERENCES

Adam,Z. and Sankoff,D. (2003) The ABCs of MGR with DCJ. *Evol. Bioinformatics*, **4**, 69–74.

Bergeron,A. *et al.* (2006) A unifying view of genome rearrangements. In Bücher,P. and Moret,B.M.E. (eds.) *Algorithms in Bioinformatics. Proceedings of WABI 2006*. Vol. 4175 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 163–173.

Bourque,G. and Pevzner,P. (2002) Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, **12**, 26–36.

Byrne,K.P. and Wolfe,K.H. (2005) The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome Res.*, **15**, 1456–1461.

Caprara,A. (2001) On the practical solution of the reversal median problem. In Gascuel,O. and Moret,B.M.E. (eds) *Algorithms in Bioinformatics. Proceedings of WABI 2001*. Vol. 2149 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 238–251.

Choi,V. *et al.* (2007) Algorithms for the extraction of synteny blocks from comparative maps. In Giancarlo,R. and Hannenhalli,S. (eds) *Algorithms in Bioinformatics. Proceedings of WABI 2007*. Vol. 4645 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 277–288.

El-Mabrouk,N. and Sankoff,D. (2003) The reconstruction of doubled genomes. *SIAM J. Comput.*, **32**, 754–92.

Fertin,G. *et al.* (2009) *Combinatorics of Genome Rearrangements.* MIT Press, Cambridge, MA.

Genome rearrangements analysis under parsimony and other phylogenetic algorithms (2004) Available at http://www.cs.unm.edu/∼moret/GRAPPA/ (last accessed date May 28, 2010).

Gordon,J.L. *et al.* (2009) Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern *Saccharomyces cerevisiae* genome. *PLoS Genet.*, **5**, e1000485.

Hannenhalli,S. and Pevzner,P.A. (1996) To cut ... or not to cut: applications of comparative physical maps in molecular evolution. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 96)*, Society for Industrial and Applied Mathematics, Philadlephia, pp. 304–313.

Hedtke, S.M. *et al.* (2006) Resolution of phylogenetic conflict in large data sets by increased taxon sampling. *Syst. Biol.*, **55**, 522–529.

Liu,T. *et al.* (2005) Quartet methods for phylogeny reconstruction from gene orders. In *Computing and Combinatorics (COCOON). Eleventh Annual Conference*. Vol. 3595 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 63–73.

Muñoz,A. and Sankoff,D. (2009) Rearrangement phylogeny of genomes in contig form. In Mandoiu,I. *et al.* (eds) *Bioinformatics Research and Applications, 5th International Symposium (ISBRA)*. Vol. 5542 of *Lecture Notes in Computer Science*, pp. 160–172.

Murphy,W.J. *et al.* (2005) Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science*, **309**, 613–617.

Sankoff,D. and Blanchette,M. (1997) The median problem for breakpoints in comparative genomics. In Jiang,T. and Lee,D.T. (eds) *Computing and Combinatorics (COCOON). Third Annual Conference*. Vol. 1276 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 251–263.

Sankoff,D. and Blanchette,M. (1998) Multiple genome rearrangement and breakpoint phylogeny. *J. Comput. Biol.*, **5**, 555–570.

Sankoff,D. *et al.* (2005) Reversals of fortune. In McLysaght,A. and Huson,D. (eds) *Comparative Genomics (RECOMB CG). Third Annual Workshop*. Vol. 3678 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 131–141.

Sankoff,D. *et al.* (2007) Polyploids, genome halving and phylogeny. *Bioinformatics*, **23**, i433–i439.

Sankoff,D. *et al.* (2008) Internal validation of ancestral gene order reconstruction in angiosperm phylogeny. In Nelson,C.E. and Vialette,S. (eds) *Comparative Genomics (RECOMB CG). Sixth Annual Workshop*. Vol. 5267 of *Lecture Notes in Computer Science*, pp. 252–264.

Siepel,A.C. (2001) Exact algorithms for the reversal median problem. Master's Thesis, University of New Mexico, Albuquerque, NM.

Soltis,D.E. *et al.* (2009) Polyploidy and angiosperm diversification. *Am. J. Bot.*, **96**, 336–348.

Tang,J. and Moret,B.M.E. (2003) Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. In *Proceedings of the 8th Workshop on Algorithms and Data Structures (WADS)*. Vol. 2748 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 37–46.

Tannier,E. (2009) Yeast ancestral genome reconstructions: the possibilities of computational methods. In Ciccarelli,F.D. and Miklós,I. (eds) *Comparative Genomics (RECOMB CG). Seventh Annual Workshop*. Vol. 5817 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 1–12.

Tannier,E. *et al.* (2009) Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, **10**, 120.

Warren,R. and Sankoff,D. (2009) Genome aliquoting with double cut and join. *BMC Bioinformatics*, **10** (Suppl. 1), 1–11.

Yancopoulos,S. *et al.* (2005) Efficient sorting of genomic permutations by translocation, inversion, and block interchange. *Bioinformatics*, **21**, 3340–3346.

Zheng,C. and Sankoff,D. (2006) Genome rearrangements with partially ordered chromosomes. *J. Comb. Optim.*, **11**, 133–144.

Zheng,C. *et al.* (2006) Genome halving with an outgroup. *Evol. Bioinformatics*, **2**, 319–326.

Zheng,C. *et al.* (2007) Removing noise and ambiguities from comparative maps in rearrangement analysis. *Trans. Comput. Biol. Bioinf.*, **4**, 515–522.

Zheng,C. *et al.* (2008) Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes. *Bioinformatics*, **24**, i96–i104.