COMPUTING THE BREAKPOINT DISTANCE BETWEEN PARTIALLY ORDERED GENOMES

ZHENG FU AND TAO JIANG

Computer Science Department, University of California, Riverside

The total order of the genes or markers on a chromosome is crucial for most comparative genomics studies. However, the current gene mapping efforts might only suffice to provide a partial order of the genes on a chromosome. Several different genes or markers might be mapped at the same position due to the low resolution of gene mapping or missing data. Moreover, conflicting datasets might give rise to the ambiguity of gene order. In this paper, we consider the reversal distance and breakpoint distance problems for partially ordered genomes. We first prove that these problems are NP-hard, and then give an efficient heuristic algorithm to compute the breakpoint distance between partially ordered genomes. The algorithm is based on an efficient approximation algorithm for a natural generalization of the well-known feedback vertex set problem, and has been tested on both simulated and real biological datasets. The experimental results demonstrate that our algorithm is quite effective for estimating the breakpoint distance between partially ordered genomes and for inferring the gene (total) order.

1. Introduction

The total order of the genes or markers on a chromosome is very important for most comparative genomics studies. The *breakpoint distance* [12,8] and *reversal distance* [11,5] are commonly used as the evolutionary distances between genomes, and they work on the premise that the total order of the genes on each chromosome has been identified. However, except for a few model genomes, most genomes have not been completely sequenced yet. For these partially sequenced/assembled genomes, only partial gene maps are available, which might have a low resolution, missing genes/markers, or conflicting ordering information among each other. Combining these partial gene maps together might only suffice to provide a partial order of genes and markers. Hence, Zheng, Lenert and Sankoff [13,14] recently proposed a new general representation of a genome in terms of genes where each chromosome is a directed acyclic graph (DAG) rather than a permutation. Any linearization of the DAGs represents a possible total order of the genome. They generalized the sorting by reversal problem to assess the distance between two partially ordered genomes. The idea is to resolve the partial orders into two total orders (*i.e.* two linearizations of the DAGs corresponding to the two genomes) with the minimum reversal distance. In the same paper, a depth-first branch-and-bound search algorithm for computing the reversal distance is presented, which runs in exponential time in the worst case.

In this paper, we study efficient computation of the reversal distance and breakpoint distance problems between two partially ordered genomes. We show that these two problems are NP-hard. We also present an efficient heuristic algorithm to compute the breakpoint dis-

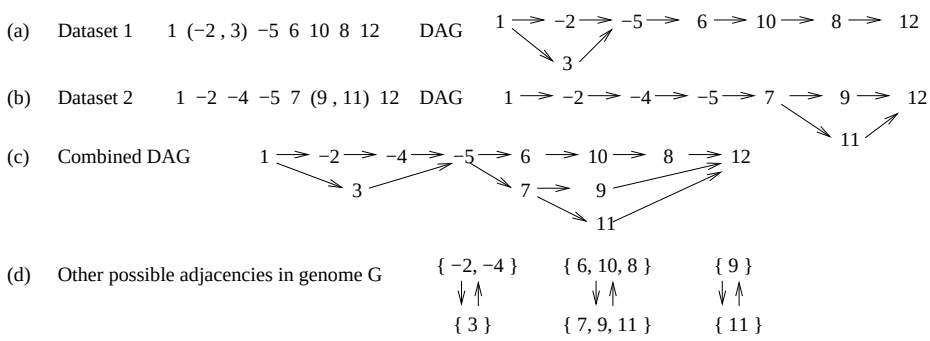

Figure 1. An example of DAG representation for a partially ordered genome.

tance between two partially ordered genomes, called *BDPOG*. The algorithm also reports a pair of total orders for the input genomes realizing the breakpoint distance. It runs in $O(n^3)$ time and uses an effi cient approximation algorithm for a natural generalization of the well-known feedback vertex set problem as a subroutine. The BDPOG algorithm has been tested on both simulated and real biological datasets. The experimental results demonstrate that it is quite effective for estimating the breakpoint distance between partially ordered genomes and inferring the total gene orders.

The rest of the paper is organized as follows. We fi rst introduce some preliminary facts and defi nitions in Section 2. Section 3 presents the NP-hardness results. Section 4 describes the algorithm BDPOG. Section 5 presents the experimental results on both simulated and real genome datasets. Finally, some concluding remarks are given in Section 6.

2. Preliminaries

Genes or markers are usually represented by signed (+ or -) symbols from a alphabet A, where the signs represent the strand of the genes. A totally ordered genome could be modeled as an ordered string of genes. However, the existing gene mapping efforts might only suffi ce to partially order the set of genes on a chromosome. If the order of some genes (*e.g.* $a_1, a_2, \cdots, a_n$) cannot be decided in a gene map, we will use $(a_1, a_2, \cdots, a_n)$ to represent the uncertainty of the ordering among them. For example, in the gene map $1\ (-2,\ 3)\ -5\ 6\ 10\ 8\ 12$, the ordering of all the genes has been decided except between genes 2 and 3.

Two or more gene maps constructed from different kinds of data or using different methodologies can be combined to form a more complicated partial order. As Zheng, Lenert, and Sankoff proposed in recent studies [13,14], directed acyclic graphs (DAGs) rather than linear permutations could be used to represent partially ordered genomes. In each DAG, all genes are represented by vertices, while the ordering relation between the genes is represented by arcs (see Figure 1).

Let Π and Γ be partially ordered genomes of size n, and the DAG representations for Π and Γ denoted as DAG(Π) and DAG(Γ). A linearization of DAG(Π) represents a possible ordering of genome Π. Let $L(\Pi)$ be the set of all possible linearizations of the DAG(Π).

Then we define the reversal distance between Π and Γ as

$$d_r(\Pi, \Gamma) = \min_{\pi \in L(\Pi), \gamma \in L(\Gamma)} d_r(\pi, \gamma) \quad (1)$$

Similarly, the breakpoint distance is

$$d_b(\Pi, \Gamma) = \min_{\pi \in L(\Pi), \gamma \in L(\Gamma)} d_b(\pi, \gamma) \quad (2)$$

We define the problem of computing $d_r(\Pi, \Gamma)$ as the *partial-order reversal distance (PRD)* problem and the problem of computing $d_b(\Pi, \Gamma)$ as the *partial-order breakpoint distance (PBD)* problem.

Clearly, all possible pairwise adjacency relationships in all possible linearizations of a DAG can be represented by the arcs of a DAG plus two arcs of opposite directions between all pairs of vertices which are not ordered by the DAG (see Figure 1d). We say that a pair of genes forms a *possible adjacency* in genome Π if they are possibly adjacent in any linearization of DAG(Π). We say that a pair of genes a and b is a *possible common adjacency* and write $a \cdot b$ if they are a possible adjacency in both genomes Π and Γ. And a is the *left end* of $a \cdot b$ and b is the *right end* of $a \cdot b$. Let $\mathcal{S}$ be the set of all possible common adjacencies. Define an order relation "$\rightsquigarrow$" between a pair of possible common adjacencies $a \cdot b$ and $c \cdot d$ in $\mathcal{S}$. We write $a \cdot b \rightsquigarrow_\Pi c \cdot d$ if one of the following four conditions is satisfied: (i) c or d is reachable from a or b in DAG(Π) (*i.e.* at least one of the genes in the second possible common adjacency is reachable from at least one of the genes in the first possible common adjacency in DAG(Π)); (ii) $a = c$ and $b \neq d$, or $b = d$ and $a \neq c$ (*i.e.* two different possible common adjacencies share the same left end or the same right end); (iii) $b = c$ (*i.e.* the right end of the first possible common adjacency is the same as the left end of the second possible common adjacency); (iv) let $a = u_{m-1}$, $b = u_m$, $c = u_1$ and $d = u_2$, and there exist possible common adjacencies $u_1 \cdot u_2,\ u_2 \cdot u_3,\ \cdots,\ u_{m-1} \cdot u_m$, $3 \leq m$ and another path in DAG(Π) from u_1 to u_m other than $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_{m-1} \rightarrow u_m$.

Based on the order relation "$\rightsquigarrow_\Pi$", we define a directed graph $\mathcal{G}_\Pi$, called the *adjacency-order graph*. The construction of $\mathcal{G}_\Pi$ is described as follows (see figure 2):

- Every possible adjacency in $\mathcal{S}$ is represented by a vertex.
- For every two possible common adjacencies, if $a \cdot b \rightsquigarrow_\Pi c \cdot d$, add an arc from the vertex $a \cdot b$ to the vertex $c \cdot d$.

A directed cycle in an adjacency-order graph usually represents a conflict among the possible common adjacencies in this cycle. And based on the construction of the adjacency-order graph, we have the following theorem.

Theorem 2.1. *All the possible common adjacencies in an acyclic adjacency-order graph $\mathcal{G}_\Pi$ could always co-exist in some linearization of DAG(Π).*

Proof. Omitted due to page limit. Please see the full version. □

The graph $\mathcal{G}_\Gamma$ can be constructed in the same way except that the arcs represent the relation $\rightsquigarrow_\Gamma$ instead of $\rightsquigarrow_\Pi$. Note that since the adjacency-order graphs $\mathcal{G}_\Pi$ and $\mathcal{G}_\Gamma$ are

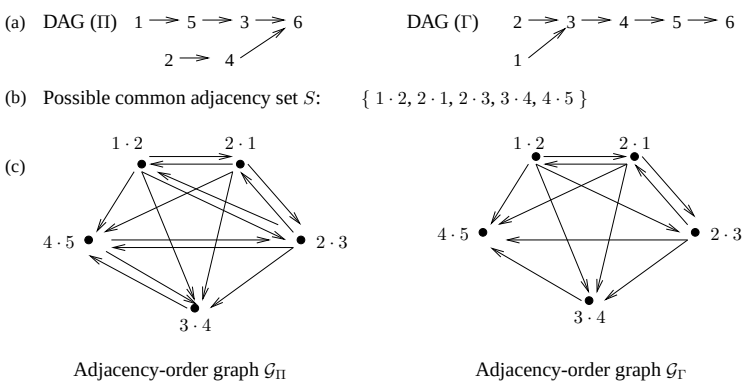

Figure 2. An example of the construction of adjacency-order graphs. In $\mathcal{G}_\Pi$, the arcs inserted by condition (i): $(1 \cdot 2, 2 \cdot 3)$, $(1 \cdot 2, 3 \cdot 4)$, $(1 \cdot 2, 4 \cdot 5)$, $(2 \cdot 1, 2 \cdot 3)$, $(2 \cdot 1, 3 \cdot 4)$, $(2 \cdot 1, 4 \cdot 5)$, $(2 \cdot 3, 3 \cdot 4)$, $(2 \cdot 3, 4 \cdot 5)$, $(4 \cdot 5, 2 \cdot 3)$, $(4 \cdot 5, 3 \cdot 4)$; the arcs inserted by condition (ii): $(2 \cdot 1, 2 \cdot 3)$, $(2 \cdot 3, 2 \cdot 1)$; the arcs inserted by condition (iii): $(1 \cdot 2, 2 \cdot 1)$, $(1 \cdot 2, 2 \cdot 3)$, $(2 \cdot 1, 1 \cdot 2)$, $(2 \cdot 3, 3 \cdot 4)$, $(3 \cdot 4, 4 \cdot 5)$; and the arc inserted by condition (iv): $(2 \cdot 3, 1 \cdot 2)$. Note that some arcs might satisfy several different conditions.

constructed by possible common adjacencies, they should share a same vertex set but may have different arc sets.

3. Computational Complexity of the PRD and PBD Problems

In this section, we show that both PRD and PBD problems are NP-hard, using different reductions.

Theorem 3.1. *The PRD problem is NP-hard.*

Proof. The proof is based on a careful analysis of the structure of the breakpoint graph for two partially ordered genomes [13,14], the Hannenhalli-Pevzner formula [5] for the reversal distance between two totally ordered genomes, and a reduction from the NP-hard problem MAX-ACD [2]. The details are omitted due to page limit. Please see the full version. □

By using a different reduction, we can prove the NP-hardness of the the breakpoint distance between two partially ordered genomes.

Theorem 3.2. *The PBD problem is NP-hard.*

Proof. We prove that the decision version of the PBD problem is NP-hard by a reduction from the decision version of *minimum feedback vertex set problem*.

Minimum Feedback Vertex Set Problem (MFVS)

INSTANCE: A directed graph $G(V, A)$ and a positive integer k.

QUESTION: Is there a subset $X \subseteq V$ with $|X| \leq k$ such that deleting all the vertices from X and their incident arcs will leave G acyclic?

Let directed graph $G(V, A)$ and positive integer k make up an arbitrary instance of the MFVS problem. The reduction to the breakpoint distance problem between partial ordered permutations (Π and Γ) works as follows: (a) For every vertex v_i in G, make two genes v_i^1

and v_i^2. (b) Add another $n+1$ genes $\{x_1, x_2 \cdots, x_{n+1}\}$, where $n = |V|$. (c) Construct a totally ordered genome $\Gamma = x_1\ v_1^1\ v_1^2\ x_2\ v_2^1\ v_2^2\ \cdots\ x_n\ v_n^1\ v_n^2\ x_{n+1}$. (d) Construct a partially ordered genome $\Pi = x_{n+1}\ (p_1, p_2, \cdots, p_m)\ x_1\ \cdots\ x_n$, where $m = |E|$ and each $p_i,\ i \in [1, m]$, represents an ordered pair of vertices. If there is an arc directed from vertex v_u to vertex v_w in G, we will have a pair $p_i = v_u^1\ v_w^2$, which means that in the genome Π gene v_u^1 is ordered before gene v_w^2. Finally, the order between p_i and $p_j,\ i \neq j$, is unknown. Figure 3 gives a simple example for this reduction.

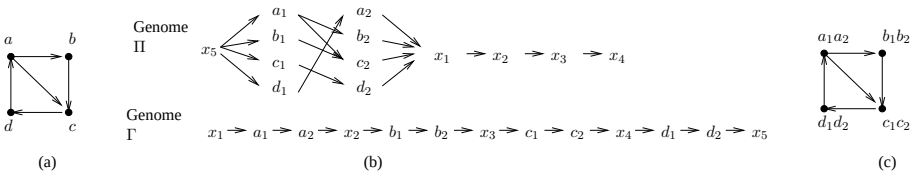

Figure 3. An example of the reduction from the minimum feedback vertex set problem to the breakpoint distance problem. (a) Directed graph $G(V, A)$. (b) Genome Π and Γ, where Γ is a totally ordered genome. (c) Adjacency-order graph of Π, $\mathcal{G}_\Pi$, which is isomorphic to $G(V, A)$.

This reduction guarantees that for Π and Γ, the set of all possible common adjacencies $\mathcal{S}= \{v_1^1 \cdot v_1^2, v_2^1 \cdot v_2^2, \cdots, v_n^1 \cdot v_n^2\}$. Adjacency-order graph $\mathcal{G}_\Pi$ is isomorphic to $G(V, A)$, while adjacency-order graph $\mathcal{G}_\Gamma$ is acyclic since Γ is totally ordered. Based on the special construction of Π and Γ, the cardinality of the minimum feedback vertex set of $\mathcal{G}_\Pi$, or graph $G(V, A)$, is exactly $d_b(\Pi, \Gamma) - 2n - 2$. Therefore, the feedback vertex set problem of $G(V, A)$ and k could be resolved by computing the $d_b(\Pi, \Gamma)$. The result of Theorem 3.2 hence follows. □

4. An Efficient Heuristic Algorithm for Computing the Breakpoint Distance

Let Π and Γ be two partially ordered genomes with possible common adjacency set $\mathcal{S}$. Computing the breakpoint distance $d_b(\Pi, \Gamma)$ is actually the problem of finding two linearizations of Π and Γ containing the maximum number of possible common adjacencies. In other words, we want to delete the smallest number of possible common adjacencies from $\mathcal{S}$ while leaving the rest of possible common adjacencies conflict free (*i.e.* they could co-exist in some linearizations). One way to delete order conflicts among possible common adjacencies is using the adjacency-order graph.

By Theorem 2.1, if the adjacency-order graph is acyclic, all the possible common adjacency vertices could be linearized by topological sort and partially ordered genomes could be totally ordered based on such a topological sort. Hence, deleting the smallest number of vertices to make both adjacency-order graphs (*i.e.* $\mathcal{G}_\Pi$ and $\mathcal{G}_\Gamma$) acyclic simultaneously could approximate the $d_b(\Pi, \Gamma)$. Formally,

Definition 4.1. *Minimum Double Feedback Vertex Set (MDFVS) problem*
Given two directed graphs with the same vertex set and different arc sets, find the minimum-cardinality subset of the vertices whose deletion leaves both graphs acyclic simultaneously.

The output vertex set is called a *minimum double feedback vertex set*.

4.1. *An Efficient Approximation Algorithm for the Minimum Double Feedback Vertex Set Problem*

Recall that the minimum feedback vertex set (MFVS) problem deals with a single graph, *i.e.*, the goal is to find the subset of vertices with the minimum cardinality whose deletion will leave the (single) input graph acyclic. We know that for the minimum feedback vertex set problem, the best-known approximation algorithm [4,10] in directed graphs achieves a performance ratio of $O(\log n \log\log n)$, where n is the number of vertices of the digraph, although the algorithm requires to the solution of a linear program. Another useful approximation algorithm [3] (denoted *APPROX-MFVS*) achieves a performance ratio bounded by the length, in terms of the number of vertices, of a longest simple cycle in the input digraph. Based on the strong relationship between the MFVS problem and the MDFVS problem, we could prove the following theorem.

Theorem 4.1. *There exists a polynomial 2λ-approximation algorithm for the* MDFVS *problem, where λ is the maximum length, in terms of the number of vertices, of a longest simple cycle in any of the two input graphs.*

Proof. In the MDFVS problem, we are given two directed graphs, say G_1 and G_2, which have the same vertex set and different arc sets. Utilizing the approximation algorithm APPROX-MFVS for the MFVS problem as a subroutine, we can easily design an approximation algorithm, denoted *APPROX-MDFVS* (see Figure 4), for the MDFVS problem as follows. Run APPROX-MFVS on G_1 and G_2 separately to get the feedback vertex sets FVS(G_1) and FVS(G_2), respectively. Denote the union of FVS(G_1) and FVS(G_2) as DFVS(G_1, G_2). DFVS(G_1, G_2) is certainly a double feedback vertex set, although not necessarily minimal. In fact, it might contain some vertices whose deletion will not affect the property of DFVS. Hence, the algorithm in its last step greedily removes vertices from DFVS(G_1, G_2) as much as possible, as long as the remaining vertices still form a DFVS. Let OPT_1 and OPT_2 be the optimal values of MFVS on G_1 and G_2 respectively. Let OPT be the optimal value of MDFVS on G_1 and G_2. It is obvious that OPT_1,$\text{OPT}_2 \leq$ OPT. Since $|\text{FVS}(G_1)| \leq \lambda_1 \text{OPT}_1$, where λ_1 is the length of a longest simple cycle in G_1, and $|\text{FVS}(G_2)| \leq \lambda_2 \text{OPT}_2$, where λ_2 is the length of a longest simple cycle in G_2, we get DFVS(G_1, G_2) $\leq 2\lambda$OPT, where $\lambda = max\{\lambda_1, \lambda_2\}$. Since the algorithm APPROX-MFVS can be implemented in $O(n^3)$ worst-case running time, the algorithm APPROX-MDFVS also runs in $O(n^3)$ time. □

4.2. *The Final Heuristic Algorithm for Breakpoint Distance*

Following the above discussion, we present an efficient heuristic algorithm, denoted as BDPOG, to calculate $d_b(\Pi, \Gamma)$ in four steps, given DAG(Π) and DAG(Γ):

(1) Add two vertices (*e.g.* v_0 and v_{n+1}) to the two input DAGs. In each DAG, add arcs from v_0 to all the vertices with in-degree 0, and add arcs from all the vertices with

Algorithm APPROX-MDFVS($G_1(V, A_1)$,$G_2(V, A_2)$)
/* G_1 and G_2 are two directed graphs with the same vertex set and different arc sets.*/
1. FVS(G_1) $\leftarrow$ APPROX-MFVS(G_1)
2. FVS(G_2) $\leftarrow$ APPROX-MFVS(G_2)
3. DFVS $\leftarrow$FVS(G_1) $\bigcup$FVS(G_2)
5. **for each** $w \in$ DFVS
6. **if** $G_1(V \setminus$ DFVS $\cup\{w\})$ and $G_2(V \setminus$ DFVS $\cup\{w\})$ are both acyclic
7. **then** DFVS $\leftarrow$ DFVS $\setminus\{w\}$
8. Output DFVS

Figure 4. The approximation algorithm for MDFVS.

out-degree 0 to v_{n+1}.

(2) Derive the possible common adjacency set $\mathcal{S}$ from the DAGs and construct the adjacency-order graphs $\mathcal{G}_\Pi$ and $\mathcal{G}_\Gamma$.
(3) Find a double feedback vertex set for $\mathcal{G}_\Pi$ and $\mathcal{G}_\Gamma$, denoted as DFVS$(\mathcal{G}_\Pi, \mathcal{G}_\Gamma)$, by applying the APPROX-MDFVS algorithm.
(4) Output $n + 1 - |\mathcal{S}| + |\text{DFVS}(\mathcal{G}_\Pi, \mathcal{G}_\Gamma)|$ as $d_b(\Pi, \Gamma)$ and the corresponding total orders of Π and Γ.

It is obvious that the performance of the BDPOG algorithm directly depends on the performance of the APPROX-MDFVS algorithm. The construction of the adjacency-order graphs in step 2 takes $O(n^3)$ time, where n is the total number of genes, since it involves a transitive closure construction. Since the APPROX-MDFVS algorithm runs in $O(n^3)$ time, the overall running time of the BDPOG algorithm is $O(n^3)$.

5. Experimental Results

In order to test the performance of the BDPOG algorithm, we have applied it to both simulated data and real biological data. We will also use an example from the Comparative Grass Genomics database (http://www.gramene.org) to illustrate the application of our method on real data.

5.1. *Simulated Data*

We use simulated data to assess the performance of our algorithm on computing the breakpoint distance between two partially ordered genomes. The simulated data is generated as follows. Start from a genome G with n distinct symbols whose signs are generated randomly. Perform r reversals on the genome G to obtain another genome H. The boundaries of these reversals are uniformly distributed within the range of the genome. The maps of these two simulated genomes are generated according to two parameters: the *group rate* p corresponding to the probability of a gene being placed at the same position as the next gene, and the *missing rate* q that determines how many genes are missing from the map. Each gene is subjected independently to these two events. Note that every gene has to exist in at least one map of each genome. Then we combine all the map datasets for each genome into a DAG. Clearly, these two DAGs represents two partially ordered genomes g

and h generated from genomes G and H. The quadruple (n, r, p, q) specifies the parameters for generating two partially ordered genomes as test data.

We run BDPOG on 20 random instances for each combination of parameters. The average breakpoint distance between partially ordered genomes g and h, computed by BDPOG, is compared with the average breakpoint distance between totally ordered genomes G and H. The results are shown in Figure 5. As we can see from the figure, our heuristic algorithm is quite reliable in computing the breakpoint distance between two partially ordered genomes. On average, the distance computed by BDPOG algorithm is very close to the real breakpoint distance between the totally ordered genomes. The difference between two breakpoint distances generally increases as two genomes become more related, or the uncertainty of gene orders increases, *e.g.*, increasing (p, q) from $(0.2, 0.1)$ to $(0.4, 0.2)$.

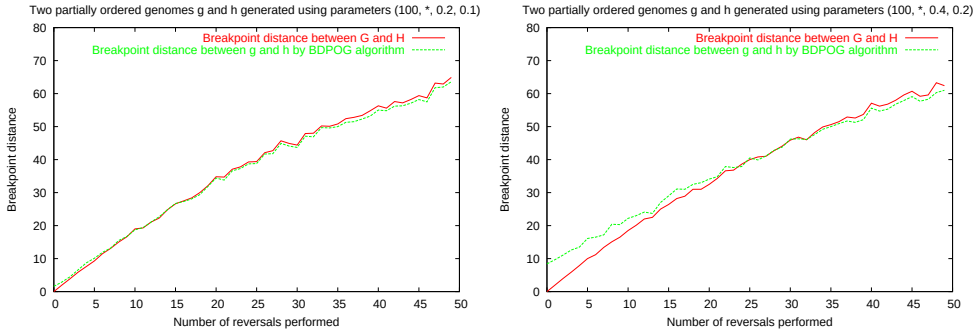

Figure 5. Performance of our heuristic algorithm BDPOG on simulated data.

5.2. *Real Data*

We use the X chromosomes of human (*Homo sapiens*, UCSC hg18, March 2006), mouse (*Mus musculus*, UCSC mm8, March 2006), and rat (*Rattus norvegicus*, UCSC rn4, November 2004) genomes in our real data test. In these three datasets downloaded from the UCSC Genome Browser [6] website (http://genome.ucsc.edu), all the genes are totally ordered. We perform the test between each pair of genomes, where we extract all the gene orthologs between the two compared genomes. Then we generate two partially ordered genomes for the compared genomes using the method described in the previous section, although we need to specify the group rate p and missing rate q here. By using our heuristic algorithm, the breakpoint distance between the simulated partially ordered genomes and the possible total order for each genome can be determined. We run our heuristic algorithm BDPOG on ten random instances, and compare the average estimated breakpoint distance and the gene orders with the real ones. The results are shown in Table 1. For example, if we generate the partially ordered chromosomes for the X chromosomes of human and mouse by using parameters $p = 0.2$ and $q = 0.1$, we get 44.15 as the average estimated breakpoint distance. In the total gene orders output by our algorithm, an average of 384.05 gene adjacencies

among 388 are kept for the human X chromosome and an average of 382.9 gene adjacencies among 388 are kept for the mouse X chromosome. Note that, the average estimated breakpoint distance 44.15 is smaller than the real breakpoint distance between human and mouse, *i.e.*, 45. A possible reason is that a small amount of uncertainties in gene order might actually decrease the number of reversals between two genomes. Overall, the results demonstrate that our algorithm performs very well on estimating breakpoint distance and recovering the gene orders for partially ordered genomes.

Table 1. Comparison of the estimated breakpoint distances and the gene orders with the real ones. $^{\zeta}$ The number of the common gene adjacencies exist in both the real genome G and the total order of the partially ordered genome g obtained by BDPOG.

G/H	#orthologs	$d_b(G, H)$	g and h ($p = 0.2, q = 0.1$)			g and h ($p = 0.4, q = 0.2$)		
			estimated $d_b(g, h)$	common adjs in g and G^{ζ}	common adjs in h and H	estimated $d_b(g, h)$	common adjs in g and G	common adjs in h and H
human/mouse	389	45	44.15	384.05	382.9	63.75	380.3	379.4
human/rat	132	22	21.3	129.9	131	27.2	128.5	126.4
mouse/rat	126	17	15.65	124.3	124.05	20	123.45	121.95

To further illustrate the application of our method on real data, we use an example from the Comparative Grass Genomics database (http://www.gramene.org). We examine two closely related genomes, maize and sorghum. We used the "IBM2 neighbors 2004" and the "IBM neighbors maps" [9] for chromosome 1 of maize, and compared it with the "Paterson 2003" [1] and the "Klein 2004" [7] maps for the chromosome labeled C and $LG-01$, respectively, of sorghum. All markers of maize indicated as having a homolog in one of the datasets of sorghum are extracted, and vice versa. We extracted 21 markers in total. The two DAGs constructed from the maize datasets and sorghum datasets and the total order of the DAGs output by our algorithm are shown in Figure 6.

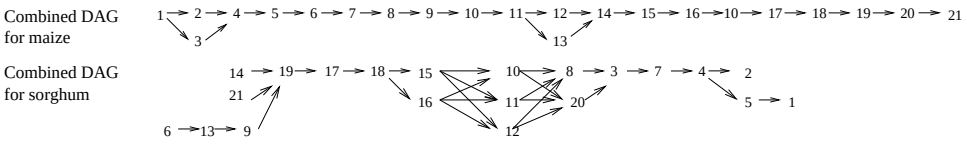

Figure 6. A comparison of maize and sorghum chromosomes using partially ordered data from the Gramene database.

6. Conclusion

In this paper, we have presented some complexity and algorithmic results for the problem of comparing two partially ordered genomes. In particular, we proposed an effi cient heuris-

tic algorithm to estimate the breakpoint distance between two partially ordered genomes and infer the corresponding linearizations achieving the distance. In our construction, we defined a useful tool, called the adjacency-order graph, and introduced a new optimization problem (MDFVS), for which we designed an efficient approximation algorithm. Our preliminary experiments on simulated and real data have demonstrated that our algorithm performs very well on estimating breakpoint distance and recovering the gene orders for partially ordered genomes. Considering the breakpoint distance is just the first step. In the future, we plan to look into other distances between partially ordered genomes, *e.g.*, the reversal distance, and try to design more efficient algorithms.

7. Acknowledgement

This project is supported in part by NSF grant CCR-0309902, National Key Project for Basic Research (973) grant 2002CB512801, NSFC grant 60528001, and a Changjiang Visiting Professorship at Tsinghua University.

References

1. J.E. Bowers *et al.*, "A high-density genetic recombination map of sequence-tagged sites for sorghum, as a framework for comparative structural and evolutionary genomics of tropical grains and grasses," *Genetics*, 165:367-386, 2003.
2. A. Caprara, "Sorting by reversal is difficult," *Proc. 1St RECOMB*, pp.75-83, 1997.
3. C. Demetrescu and I. Finocchi, "Combinatorial Algorithms for Feedback Problems," *Information Processing Letters*, 86(3):129-136, 2003.
4. G. Even *et al.*, "Approximating minimum feedback sets and multi-cuts in directed graphs," *Proc. 4th Int. Conf. on Integer Prog. and Combinatorial Optimization, Lecture Notes in Comput. Sci.*, 920, Springer-Verlag, 14-28, 1995.
5. S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip (Polynomial algorithm for sorting signed permutations by reversals)," *Proc. of 27th Annual ACM Symposium on the Theory of Computing*, pp.178-187, 1995.
6. D. Karolchik *et al.*, "The UCSC Genome Browser Database," *Nucleic Acids Res.*, 31(1): 51-54, 2003.
7. M.A. Menz *et al.*, "A high-density genetic map of Sorghum bicolor (L.) Moench based on 2926 AFLP, RFLP and SSR markers," *Plant molecular biology* 48:483-499, 2002
8. J. Nadeau and B. Taylor, "Lengths of chromosomal segments conserved since divergence of man and mouse," *Proc. Natl. Acad. Sci.*, 81:814-818, 1984.
9. M.L. Polacco and Jr Coe E., "IBM neighbors: a consensus GeneticMap," 2002
10. P.D. Seymour, "Packing directed circuits fractionally," *Combinatorica*, 15:281-288, 1995.
11. D. Sankoff, "Mechanisms of genome evolution: models and inference," *Bull. Int. Stat. Institut.*, 47:461-475, 1989.
12. G. Watterson *et al.*, "The chromosome inversion problem," *J. Theor. Biol.*, 99:1-7, 1982.
13. C. Zheng and D. Sankoff, "Genome rearrangements with partially ordered chromosomes," *COCOON*, 2005.
14. C. Zheng, A. Lenert, and D. Sankoff, "Reversal distance for partially ordered genomes," *Bioinformatics*, 21(Suppl.1):i502-i508, 2005.