

Heuristic Algorithm for Computing Reversal Distance with MultiGene Families via Binary Integer Programming

J. Suksawatchon and C. Lursinsap

Advanced Virtual and Intelligent Computing (AVIC) Center,
Department of Mathematics,
Chulalongkorn University, Thailand.
Email: jakkarin@buu.ac.th and lchidcha@chula.ac.th

M. Bodén

School of Information Technology
and Electrical Engineering,
The University of Queensland, Australia.
Email: mikael@itee.uq.edu.au

Abstract—Hannenhalli and Pevzner developed the first polynomial-time algorithm for the combinatorial problem of sorting of signed genomic data. Their algorithm solves the minimum number of reversals required for rearranging a genome to another when gene duplication is nonexisting. In this paper, we show how to extend the Hannenhalli-Pevzner approach to genomes with multigene families. We propose a new heuristic algorithm to compute the reversal distance between two genomes with multigene families via the concept of binary integer programming without removing gene duplicates. The experimental results on simulated and real biological data demonstrate that the proposed algorithm is able to find the reversal distance accurately.

I. INTRODUCTION

The appearance of gene content and gene order data has taken phylogenetic studies to a new level. The difference between the order and orientation of genes on a chromosome in two genomes can be used as a measure of evolutionary distance. The gene order and orientation change very slowly due to evolutionary events such as reversal, transposition and translocation. Hannenhalli and Pevzner [3] developed a theory for *signed permutations* (of genes) and an algorithm for computing the reversal distance between two genomes in polynomial time – indicating the extent of genome rearrangement required. Henceforth, a sign (+ or -) is associated with each gene representing its transcriptional orientation, i.e. on which of the two complementary DNA strands the gene is located. El-Mabrouk [2] extended the Hannenhalli-Pevzner theory to compute the edit distance for inversions and deletions, and for inversions and insertions not resulting from gene duplication. The best running time for *sorting by reversals* is quadratic [4], while the *reversal distance* can be computed in linear time [1]. With the more general aim of constructing a phylogenetic tree, the method of *sorting by reversals* can evaluate the distance between any two genomes (species) in the tree.

However, duplicated genes account for about 38% of the human genome, 44% in bacterial genomes, and about 40% in archaeobacterial genomes [11]. The Hannenhalli-Pevzner theory does not account for duplicated genes and the algorithm is

consequently producing misleading results for more complex genomic data. Several models have been developed to account for gene duplication for the analysis of genome rearrangement. Sankoff [6] designed a branch-and-bound algorithm to find an *exemplar* distance between genomes. The idea is to delete all but one gene in a gene family, so as to minimize the reversal distance between the related genomes. The *exemplar* problem is not only NP-hard [9] but also unlikely to have polynomial time constant ratio approximation algorithm unless NP=P [10]. Nguyen *et al.* [8] proposed a divide-and-conquer approach for calculating the *exemplar* distance by partitioning the gene families into disjoint subsets. All copies but one of each gene had to be deleted in each two genomes causing side-effects like loss of data and accuracy [12]. Recently, Chen *et al.* [7] proposed an efficient and effective heuristic algorithm (SRDD) for solving the *signed reversal distance with duplicates* problem using the techniques of *minimum common partition* and *maximum cycle decomposition*. The signed reversal distance with duplicates is NP-Hard as are both techniques [7]. Due to the size of genomes, the computational efficiency is an extremely important aspect of the analysis of genome rearrangements. SRDD is a part of a state-of-the-art system for ortholog assignment by reversals (SOAR).

In this paper we ask how the minimum reversal distance can be determined efficiently and accurately in the presence of multigene families. We propose a new heuristic algorithm to find the canonical permutation of gene duplicates that obtains the nearest *minimal signed reversal distance for multigene families* (SRDMF). We extend the concept of a breakpoint graph [3] to cope with multigene families. The key idea of our method is to generate an *incomplete breakpoint graph* which allows for exploring gene-gene relationships across the two genomes, efficiently and accurately, by maximizing the number of graph cycles. The technique of *Binary Integer Programming* (BIP) is applied to find a set of *gray edges* that minimize the reversal distance.

The SRDMF algorithm has been tested on both synthetic

and real biological datasets (from human, mouse and rat X chromosomes), and compared with the SRDD algorithm [7]. The experimental results demonstrate that our algorithm generally outperforms the SRDD algorithm in terms of accuracy of determining the minimal reversal distance (based on the known minimal reversal distance).

The rest of the paper is organized as follows. Section II introduces the preliminaries of our approach. Section III describes our heuristic algorithm (SRDMF). Section IV presents the experimental results. The conclusion and further work are discussed in section V.

II. PRELIMINARIES

Our approach is based on Hannenhalli and Pevzner's *breakpoint graph* [3] for sorting signed permutations by reversals. To understand our extension we briefly introduce the basic concepts of the Hannenhalli-Pevzner theory.

Let π and ϕ be signed permutations of size n , such that $\pi = (\pm\pi_1, \dots, \pm\pi_n)$ and $\phi = (\pm\phi_1, \dots, \pm\phi_n)$. Typically, ϕ is an identity, i.e. $\phi_i = i$. Let the unsigned permutation $\pi' = \{\pi'_0, \pi'_1, \dots, \pi'_{2n}, \pi'_{2n+1}\}$ be defined such that $\pi'_0 = 0$, $\pi'_{2n+1} = 2n + 1$ and for all i , each π_i is replaced by the $(2\pi_i - 1, 2\pi_i)$ if $\pi_i > 0$, and replaced by the $(2|\pi_i|, 2|\pi_i| - 1)$, otherwise. The unsigned permutation ϕ' is defined in the same way as π' . We will only discuss the distance from a genome π to the *identity* genome $\phi = (1, 2, \dots, n)$. Without loss of generality, we always compute the distance between any two genomes π and ϕ .

To find the *minimum number of the reversals* required to transform π to ϕ , Hannenhalli and Pevzner's algorithm depends on a bi-colored cycle graph (G), called the *breakpoint graph*, constructed from π' and ϕ' . Two genes π_i and π_j ($1 \leq j < i \leq n$) in a genome π are said to be *consecutive* if $|\pi_i - \pi_j| = 1$. There is a *breakpoint* between π_i and π_j in π if π_i and π_j are not consecutive. The *breakpoint graph* $G = (V, E)$ of π' (with respect to the unsigned identity permutation) is constructed by arranging in a sequence of $2n + 2$ vertices, corresponding to the element of π' . Each edge joins every pair of consecutive elements of π' , i.e. starting with 0, called *black edge* or *reality edge*, and every other pair of consecutive integers, i.e. starting with $(0, 1), (2, 3), \dots, (2n, 2n + 1)$ called *gray edge* or *desire edge*. A gray edge is *oriented* if it links two left vertices of two black edges, or two right vertices of two black edges, otherwise it is called *unoriented*. An example of a breakpoint graph is shown in Fig. 1. The example graph decomposed into a set of disjoint color-alternating cycles. By the size of a cycle, we mean the number of black edges it contains. The number of cycles of breakpoint graph is maximized when $\pi = \phi$. A *component* of G is a maximal subset of crossing cycles. A component is *good* (oriented) if it contains at least one oriented

gray edge, and *bad* (unoriented) otherwise¹.

A *hurdle* is an unoriented component which does not contain other unoriented component and does not separate other unoriented components. A hurdle is called a *superhurdle* if it were eliminated, a nonhurdle would emerge as a hurdle; otherwise it is a *simple hurdle*. A *fortress* exists iff there is an odd number of hurdles and all are superhurdles.

A *reversal* $\rho_k(i, j)$, for any k and $1 \leq i, j \leq n$, of $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ transforms π into $\rho_k^{-1}(i, j) = (\pi_1, \dots, -\pi_j, -\pi_{j-1}, \dots, -\pi_i, \dots, \pi_n)$. A *reversal distance* is the number of reversals $\rho_1(i_1, j_1) \rho_2(i_2, j_2), \dots, \rho_t(i_t, j_t)$ of minimal length t required to transform π to ϕ . We use $d(\pi)$ to denote the *minimal reversal distance* between π and ϕ . The minimal reversal distance necessary to transform π to ϕ of permutation length n is given by the formula [3]:

$$d(\pi) = b(\pi) - c(\pi) + h(\pi) + f(\pi) \quad (1)$$

where $b(\pi)$ is the number of black edges, $c(\pi)$ is the number of cycles in G , $h(\pi)$ its number of hurdles, and $f(\pi)$ is equal to 1 if fortresses exist in G and zero otherwise.

From Fig. 1, the permutation π has 15 black edges, 6 cycles, 2 hurdles, and nonfortress. By using the equation (1), the reversal distance of the permutation is $15 - 6 + 2 + 0 = 11$.

The probability for a given component to be good is greater than its probability to be bad, the number of hurdles is close to 0 [13]. The number of cycles is the dominant parameter in the HP formulate for $d(\pi)$, if $b(\pi)$ is considered as constant. Notice that more cycles mean less reversals.

III. COMPUTING THE NEAREST MINIMUM REVERSAL DISTANCE FOR MULTIGENE FAMILIES

In the presence of duplicated genes, let the set $A = \{g_1, g_2, \dots, g_k\}$ be the set of genes (as a uni-chromosome permutation) π of length n . Each g_i is represented by a symbol, a character or a number. For each gene g_i ($1 \leq i \leq k$), let $K(g_i)$ be the number of occurrences (+ or -) of a symbol g_i in π and ϕ such that $K(g_i) \geq 1$. A gene is called a *single* if it is the only member of a gene family in that genome, $K(g_i) = 1$. Otherwise the gene is referred to as a member of a *multigene family*. Since the breakpoint graph is unable to account for members of gene families, each member element, π_i , must be systematically renamed to achieve the minimum reversal distance.

Consider an example shown in Fig. 2. Genes π_3 and π_4 contain the same element 2 and π_8, π_9 , and π_{10} contain the same element 3. These two gene families are rearranged and relocated at new positions in ϕ . To uniquely rename each element, first, all elements in π are renamed according to their positions with respect to ϕ . Symbols x_1, x_2 are introduced

¹Hannenhalli and Pevzner proved that the good components can be transformed to a set of cycles of size one, by a sequence of *good reversals* that do not create any new bad components. Bad components require *bad reversals* to be solved.

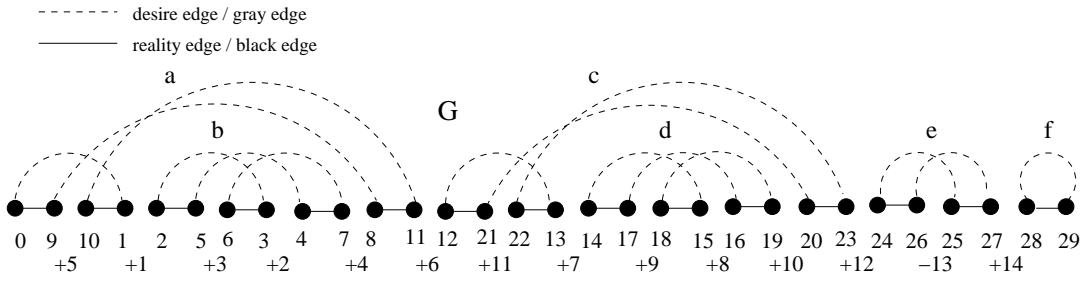


Fig. 1. Breakpoint graph G for the permutation π with respect to the identity permutation of size $n = 14$. Connected component e is oriented, and a, b, c and d are unoriented. Unoriented component b and d are hurdles and also superhurdles, but unoriented component a and c are not hurdles. The f is the cycle of size 1.

as temporary names for all duplicated elements labelled 2 and symbols y_1, y_2 and y_3 are introduced as a temporary names for all duplicated elements labelled 3. The renaming results for both π_{new} and ϕ_{new} (identity) are shown in Fig. 2.

position	1	2	3	4	5	6	7	8	9	10	11	12
π	4	1	2	2	5	6	7	3	3	3	8	9
ϕ	1	2	2	5	4	6	3	3	3	8	7	9
π_{new}	5	1	x_1	x_2	4	6	11	y_1	y_2	y_3	10	12
ϕ_{new}	1	2	3	4	5	6	7	8	9	10	11	12

Fig. 2. The example of two copies of element 2 and three copies of element 3 and the renaming results, π_{new} and ϕ_{new} by temporary name x and y .

The elements x_1 and x_2 in π can be assigned two possible labels (2 or 3) and, similarly, the elements y_1, y_2 , and y_3 in π can be assigned three possible labels (7, 8 or 9). Therefore, there are $2! \times 3!$ possible name assignments. Each assignment may result in a different reversal distance, $d(\pi)$, estimated from equation (1) or computed by other reliable computing reversal distance tools [1][5]. The minimal $d(\pi)$ can be found by generating and testing all assignments. However, time grows exponentially with the number of duplications. Therefore, we turn to a heuristic method for assigning a final name for each temporary name.

In the example π' of $\pi = (5, 1, x_1, x_2, 4, 6, 11, y_1, y_2, y_3, 10, 12)$ becomes $\pi' = (0, 9, 10, 1, 2, x_{11}, x_{12}, x_{21}, x_{22}, 7, 8, 11, 12, 21, 22, y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}, 19, 20, 23, 24, 25)$ where x_1, x_2 are replaced by $(x_{11}, x_{12}), (x_{21}, x_{22})$, and y_1, y_2 and y_3 are replaced by $(y_{11}, y_{12}), (y_{21}, y_{22})$ and (y_{31}, y_{32}) , respectively. From π' , first, its **incomplete breakpoint graph**, IG , is created (incomplete in the sense that the actual names of x_i , for $1 \leq i \leq 2$ and y_j , for $1 \leq j \leq 3$ are so far unknown). The example of incomplete breakpoint graph, IG , is shown in Fig. 3.

The names of x_1, x_2 can be either 2 or 3 and the names of y_1, y_2 and y_3 can be either 7, 8 or 9. Hence, the names of x_{i1} must be 3 or 5, for $1 \leq i \leq 2$ and the names of x_{j2} must be 4 or 6, for $1 \leq j \leq 2$. In the same way, the name of y_{m1} , for $1 \leq m \leq 3$, must be 13, 15, or 17 and the name of y_{k2} , for

$1 \leq k \leq 3$, must be 14, 16, or 18.

With all temporary names in place, all black edges and gray edges are identified. The pair (π'_i, π'_j) in IG , $1 \leq i < j \leq 2n+2$, is connected by a black edge if π'_i or π'_j is a variable, or both π'_i and π'_j are variables with different temporary names. The gray edges are determined as usual but by handling variables as follows.

Let E_p be the set of gray edge groups (p) separated by the pair of non-existing gray edges in IG . Each group is composed of all possible gray edges (e_k) with respect to its non-existing gray edge. For the example in Table I, there are seven groups (E_1, E_2, \dots, E_7) corresponding to non-existing gray edges, $(2, 3), (4, 5), \dots (18, 19)$. To form the possible gray edges for group $E_1 (2, 3)$, the element 2 exists in the IG but the element 3 does not exist for any point in IG . According to the assumption, the possible gray edges in this case are $e_1:(2, x_{11})$ and $e_2:(2, x_{21})$ because either x_{11} or x_{21} must be named 3. The other groups are handled similarly. Table I summarizes all possible edge groups for the temporary names x and y . Each e_k denotes each possible gray edge and is the decision variable for the integer programming procedure. The value is equal to 1 if the edge is selected, otherwise 0.

With the complication introduced by gene families, we classify the type of cycle into three types which are

- 1) Type A is the complete cycle (as per the breakpoint graph, size of cycle ≥ 1).
- 2) Type B is the incomplete cycle with constants.
- 3) Type C is the incomplete cycle with variables.

Note that transforming from an incomplete breakpoint graph to its complete breakpoint graph, the number of type A cycles is constant, while the number of type B cycles become 0, and the number of type C cycles is constant or is reduced. There are two cycles of type A and seven cycles of type C in Fig. 4.

While completing the breakpoint graph the set of gray edges needs to satisfy the properties: (1) every non-linking vertex is incident to exactly one gray edge; (2) only a gray edge from each edge group (E_k) is selected.

Equation (1) provides the essential clue to achieve the minimum number of reversals by minimizing $b(\pi) - c(\pi)$. This implies that the selected gray edges must create the maximum

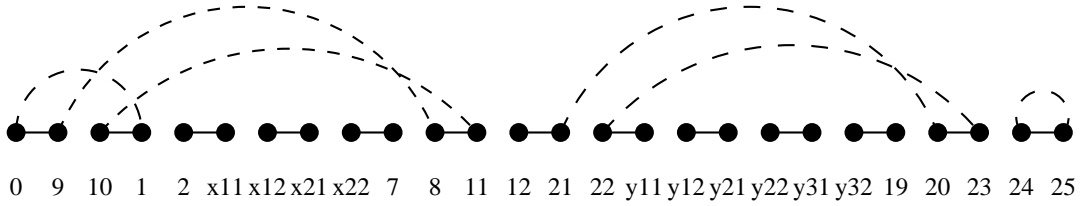


Fig. 3. The example of incomplete breakpoint graph IG for π' .

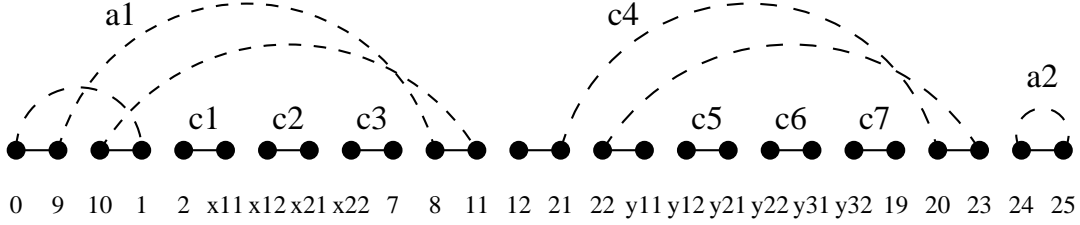


Fig. 4. All of cycles in IG , name $a1$ and $a2$ are the *type A* and $c1, c2, c3, c4, c5, c6, c7$ are the *type C*.

number of cycles. To find the maximum cycles, we set the weight w_i for each edge e_i is defined as the following cases and the examples are shown in Fig. 5.

case 1: $w_i = 1$ if e_i forms the cycle size 1.

case 2: $w_i = 2$ if e_i joins by two vertices in the same incomplete cycle.

case 3: $w_i = 3$ if e_i joins by two vertices in the different incomplete cycles.

The following notation is used for describing the application of binary integer programming.

- l_{ik} : equal to 1 if edge e_k belongs to an incomplete cycle c_i , otherwise 0.
- T_c : the number of incomplete cycles type c .
- T_{c_i} : the number of variables for a cycle c_i .
- T_e : the number of possible gray edges.
- T_g : the number of possible gray edge groups.
- T_{E_p} : the number of possible gray edges for group p .

$$\text{Objective function: } \text{Minimize } \sum_{i=1}^{T_e} w_i \cdot e_i \quad (2)$$

$$\text{Constraints: } \sum_{k=1}^{T_e} l_{ik} = T_{c_i} \quad \text{for } i = 1 \text{ to } T_c \quad (3)$$

$$\sum_{i=1}^{T_{E_p}} e_i = 1 \quad \text{for } p = 1 \text{ to } T_g \quad (4)$$

$$e_i \in \{0, 1\} \quad (5)$$

The objective function (2) minimizes the weight of candidate edges (the minimal weight forms the maximum cycles). The constraint (3) ensures that the number of selecting edges e_k does not exceed total number of variables for each cycle c_i , constraint (4) ensures that only one possible gray edge is selected for each group. The outline for the SRDMF algorithm is given in Fig. 6.

Algorithm SRDMF
Input: two genomes π and ϕ (included gene families)
Output: The selected gray edges are completed the incomplete breakpoint graph.

1. Rename each family size 1 of π according to its position in ϕ , and rename each family of size greater than 1 by temporary variables and keep the mapping position.
2. Transform signed to unsigned permutation. For each variable g_i , replace by g_{i1} and g_{i2} .
3. Construct the incomplete breakpoint graph (IG) for unsigned permutation.
4. Identify all black edges and gray edges in IG.
5. Identify possible gray edge groups (E) and gray edges (e) and weight for each gray edge.
6. Formulate the binary integer programming from equations (2) – (5).
7. Solve the binary integer programming.

Fig. 6. The outline for heuristic algorithm SRDMF.

Once this algorithm is completed, all variables are assigned actual values. Equation (1) can then be applied to compute the minimal reversal distance between the two genomes.

IV. EXPERIMENTAL RESULTS

We have implemented our algorithm (SRDMF) in MATLAB version 7.0 and tested it for correctness and performance². Test data fell into two categories: synthetic data and real biological data.

A. Synthetic data

In order to rigorously test correctness we compare the calculated reversal distance with the exact minimal reversal distance that obtained from the reliable program (GRAPPA)³. The accuracy was compared to that achieved with the SRDD

²All testing was performed on a Sony laptop with 1.6 MHz Pentium IV processor and 1 GB of RAM, and running the Linux operating system Redhat version 8.0. The BIP was solved by optimization toolbox for MATLAB version 7.0 (the module *binprog*).

³We use the GRAPPA version 2.0 [1], module of *invdist* only, to compute the reversal distance for all possible combinations to identify the one that has the minimal reversal distance.

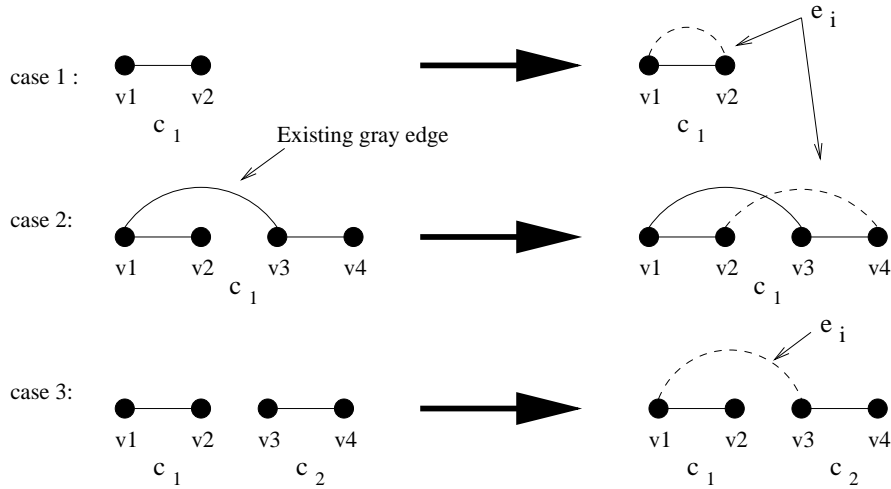


Fig. 5. The example of connecting edge e_i for all three cases.

algorithm⁴. Although the SRDD was not originally proposed to compute the reversal distance, its objective is closely related to that of the problem addressed here. The synthetic data set is generated as follows:

- 1) Start from π with n distinct symbols whose signs are also generated randomly.
- 2) After that, randomly generate f families, where each family has random size $3 \leq K(f_i) \leq 5$, i.e. recursively combining single gene until the size equals to $K(f_i)$.

To obtain the genome ϕ , we perform t random reversals uniformly over the genome π . We ran the SRDMF and SRDD on ten different instances of ϕ . Fig. 7 shows the average performance of both algorithms over the ten instances in term of reversal distance, contrasted with the exact minimal reversal distance as determined by GRAPPA (*invdist*).

On average, each run of the SRDMF algorithm takes about 10 seconds while the SRDD takes less than 5 seconds (we believe this is because SRDD uses a simple greedy algorithm to find the maximum number of cycles). Our heuristic algorithm consistently produces a closer estimate of the exact minimal reversal distance compared to SRDD, for each $t > 35$. However, both algorithms overestimate the actual reversal distance (t), for each $t > 35$. These statistics indicate that our algorithm is quite reliable in finding the nearest minimal reversal distance with multigene families.

B. Real biological data

We downloaded the X chromosome of human, (Homo Sapien, NCBI build 34, July 2003 UCSC hg 16) mouse (*Mus musculus*, NCBI build 32, October 2003; UCSC mm4) and rat (*Rattus norvegicus*, Baylor HGSC v.31, June 2003; UCSC rn3) from the SOAR web page (<http://www.cs.ucr.edu/xinchen/soar.html>). There are 922 genes

from human X chromosome, 1030 genes from mouse and 899 genes from rat, respectively. We also have used information about gene families from this site (as shown in Table II). There are 355 families of size one between human-mouse, 321 between human-rat, and 348 between mouse-rat. By using the SRDD, the breakpoint and reversal distances between human-mouse are 143 and 123, between human-rat are 135 and 117, between mouse-rat are 188 and 155, respectively. We ran SRDMF for all genome combinations. The results are shown in Table II.

The comparative results for all three pairs of genomes are summarized in Table II. The results show that SRDD and SRDMF determine similar distances for human-mouse and human-rat. We believe that the non-distinct result is due to the low estimated reversal distance between human-mouse and human-rat (34 and 35% of the number of single genes, respectively). According to the synthetic data, both SRDD and SRDMF obtain similar minimal distances when $t \leq 35$. However, the SRDMF improves slightly on SRDD for the mouse-rat calculation, both in term of breakpoint and reversal distance. Noteworthy, the estimated reversal distance between mouse and rat is 45% of the number of single genes. We expect to see a greater improvement for SRDMF for genomes that are less related.

V. CONCLUSION

We present a new heuristic algorithm to find the nearest minimal reversal distance between genomes with multigene families. The approach uses the notion of a breakpoint graph, but readily provides means for exploring possible combinations of duplicate genes across genomes. The exploration is done using binary integer programming optimization based on pre-determined penalties for properties of an incomplete version of the breakpoint graph.

⁴The executable program is provided from the authors [7].

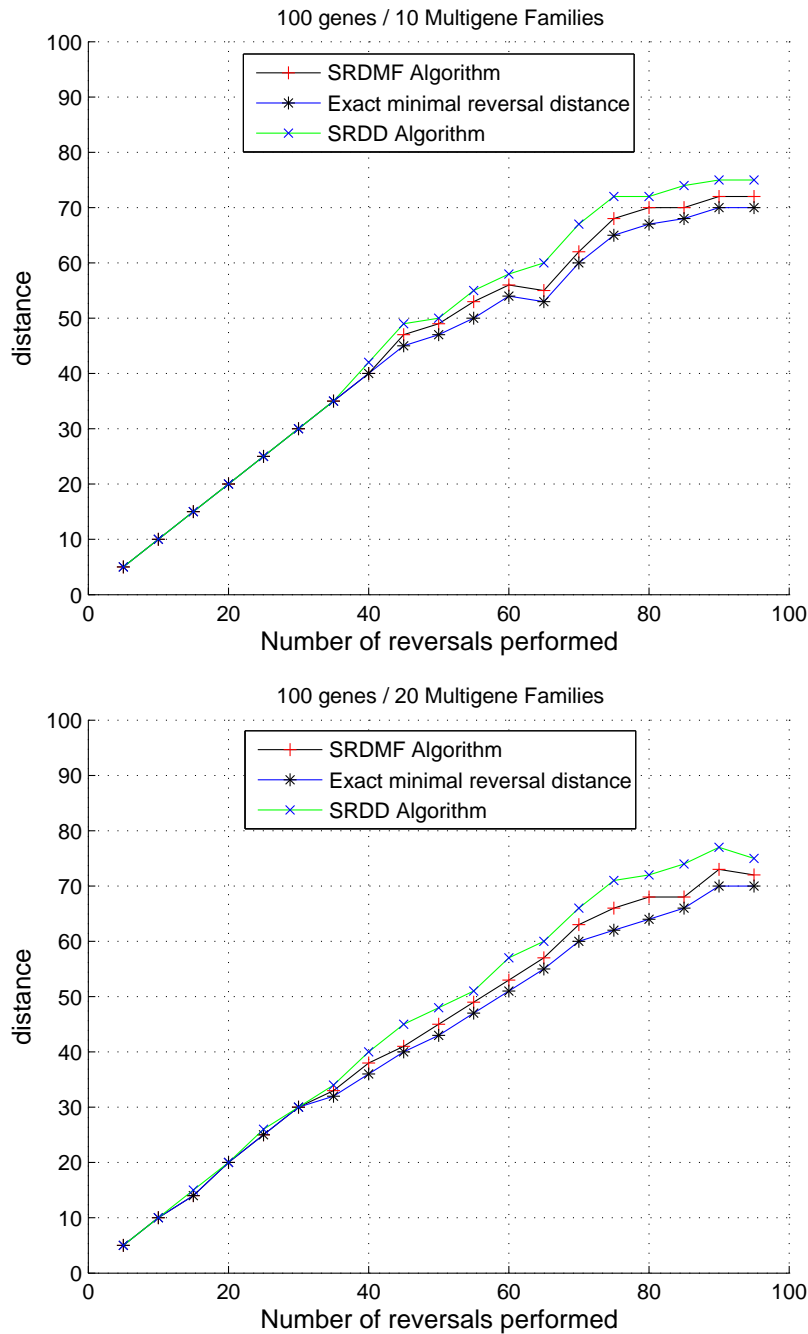


Fig. 7. The reversal distance of both algorithms comparing with the exact reversal distance.

Experimental results on synthetic and real biological data sets show that our approach exceeds the accuracy provided by the SRDD algorithm (Signed Reversal Distance with Duplicates) but with a drop in computation time.

Future work will investigate how to account for multi-chromosome and other rearrangement operations such as transposition, insertion, deletion.

REFERENCES

- [1] D.Bader, B.Moret, and M. Yan, "A linear-time algorithm for computing inversion distance between signed permutations with an experimental study," *Algorithms and Data Structures: Seventh International Workshop, WADS 2001*, pp.365-376, 2001.
- [2] N. El-Mabrouk, "Genome rearrangement by reversals and insertions/deletions of contiguous segments," *In Proc. 11 Ann. Symp. Combi-*

TABLE I

THE 7 POSSIBLE EDGE GROUPS AND THE 24 POSSIBLE GRAY EDGES OF THE INCOMPLETE BREAKPOINT GRAPH (IG).

$E_1(2, 3)$	$E_2(4, 5)$	$E_3(6, 7)$	$E_4(12, 13)$	$E_5(14, 15)$	$E_6(16, 17)$	$E_7(18, 19)$
$e_1 : 2-x_{11}$	$e_3 : x_{12}-x_{21}$	$e_5 : x_{12}-7$	$e_7 : 12-y_{11}$	$e_{10} : y_{12}-y_{21}$	$e_{16} : y_{12}-y_{21}$	$e_{22} : y_{12}-19$
$e_2 : 2-x_{21}$	$e_4 : x_{22}-x_{11}$	$e_6 : x_{22}-7$	$e_8 : 12-y_{21}$	$e_{11} : y_{12}-y_{31}$	$e_{17} : y_{12}-y_{31}$	$e_{23} : y_{22}-19$
			$e_9 : 12-y_{31}$	$e_{12} : y_{22}-y_{11}$	$e_{18} : y_{22}-y_{11}$	$e_{24} : y_{32}-19$
				$e_{13} : y_{22}-y_{31}$	$e_{19} : y_{22}-y_{31}$	
				$e_{14} : y_{32}-y_{11}$	$e_{20} : y_{32}-y_{11}$	
				$e_{15} : y_{32}-y_{21}$	$e_{21} : y_{32}-y_{21}$	

TABLE II

RESULTS OF THE CALCULATION OF REVERSAL DISTANCE AND BREAKPOINT DISTANCE BY SRDMF COMPARED WITH SRDD. THE RESULTS IN PARENTHESIS ARE OBTAINED USING THE SRDMF ALGORITHM.

X chromosome of	# of families	# number of families of size					SRDD vs (SRDMF)		Computing time SRDMF (minute)
		2-5	6-10	11-15	16-20	>20	reversal distance	breakpoint distance	
human-mouse	447	68	14	2	2	3	123(123)	143(143)	12
human-rat	408	67	14	2	2	2	117(117)	135(135)	8
rat-mouse	460	73	25	1	1	2	155(153)	188(184)	13

natorial Pattern Matching (CPM'00), Lecture Notes in Computer Science vol. 1848, pp. 222-234, 2000.

- [3] S. Hannenhalli and P.A. Pevzner, "Transforming Cabbage Into Turnip," *Proceedings of the 27th Annual ACM-SIAM Symposium on Theory of Computing*, pp. 178-189, 1995.
- [4] H.Kaplan, R.Shamir and R.Tarjan, "A faster and Simpler algorithm for sorting signed permutations by reversals," *SIAM Journal of Computing*, vol.29(3), pp.880-892, 1999.
- [5] I. Mantin and R. Shamir, "An Algorithm for Sorting Signed Permutations by reversal," <http://www.math.tau.ac.il/~rshamir/GR>, 1999.
- [6] D. Sankoff, "Genome Rearrangement with gene families," *Bioinformatics*, vol. 15, pp.909-917, 1999.
- [7] X.Chen et.al, "Computing the assignment of orthologous genes via genome rearrangement," *Proceedings of Asia Pacific Bioinformatics Conference (APBC2005)*, pp.363-378, 2005.
- [8] C.Thach Nguyen, Y.C. Tay and L. Zhang, "Divide-and-Conquer approach for the exemplar breakpoint distance," *Bioinformatics*, vol.21(10), pp.2171-2176, 2005.
- [9] D. Bryant, "The complexity of calculating exemplar distances," *Comparative Genomics*, pp.207-212, 2000.
- [10] C.T. Nguyen, "The complexity of the exemplar problem," unpublished manuscript, 2004.
- [11] J. Zhang, "Evolution by gene duplication : an update," *TRENDS in Ecology and Evolution*, vol.18(6), 2003.
- [12] Joel V., E. Young, E. Lerat, B. Moret, "Reversing Gene Erosion-Reconstructing Ancestral Bacteria Genomes from Gene Content and Order Data," *Proc. 4th Int'l Workshop on Algorithms in Bioinformatics WABI'04 in Lecture Notes in Computer Science*, vol.3240, pp.1-13, 2004.
- [13] A. Caprara, "On the practical solution of reversal median problem," *Algorithm in Bioinformatics, Proceeding of WABI 2001*, vol.2149, pp.238-251, 2001.