Pacific Symposium on Biocomputing 7:259-270 (2002)

Multiple Genome Rearrangement By Reversals

Shiquan Wu and Xun Gu
Center of Bioinformatics and Biological Statistics
Iowa State University
Ames, IA 50011, USA
{sqwu,xgu}@iastate.edu

In this paper, we discuss a multiple genome rearrangement problem: Given a collection of genomes represented by permutations, we generate the collection from some fixed genome, e.g., the identity permutation, in a minimum number of signed reversals. It is NP-hard, so efficient heuristics is important for finding its optimal solution. We at first discuss how to generate two and three genomes from a fixed genome by polynomial algorithms for some special cases. Then based on the polynomial algorithms, we obtain some approximation algorithms for generating two and three genomes in general, respectively. Finally, we apply these approximation algorithms to design a new approximation algorithm for generating more genomes. We also show by some experimental examples that the algorithms are efficient.

1 Introduction

Comparative genomics is one of the most important areas in computational biology and bioinformatics. Sorting by reversal plays a central role in Comparative genomics. The problem was originated in last decade [2,3,4]. Its theme is to determine the evolutionary distances between organisms by using genomic data. Transformations of genomes are widely studied under evolutionary events such as insertion, deletion, point mutation (substitution), reversal, etc [11,12]. Recently, optimal recombination is also discussed[19]. So far, most of the study on comparative genomics has been focused on sorting by reversal[18]. A genome is represented by a permutation and an optimal reversal process is found from any given permutation to the identity permutation.

Sorting by reversal is categorized into two classes: sorting by unsigned and signed reversals, respectively. At first, sorting by unsigned reversals is NP-hard[7,8]. Therefore, only efficient approximation algorithms can be expected to find for the solution of the problem. So far, the best approximation algorithm has been a 1.5-approximation algorithm[10]. It is proved that there exists no polynomial-time 1.0008-approximation algorithm[6].

However, sorting by signed reversals is polynomial-time solvable[13,14]. Many quadratic-time algorithms are widely used for finding the optimal solutions of the problem[5,15,16]. Recently, a linear-time algorithm is found for computing the signed reversal distance between any two signed permutations[1].

Sorting by reversal can be regarded as a problem that generates a permutation from some fixed permutation by a minimum number of reversals. Multiple genome rearrangement by reversals is a generalization of sorting by reversal. It is to generate a given collection of permutations (genomes) from a fixed permutation, e.g., the identity, in a minimum number of reversals. For the unsigned case, the problem is obviously NP-hard (since sorting by unsigned reversals is NP-hard). For the signed case, it is proved that the problem is NP-hard even if two permutations are generated from a permutation[8]. This implies that the problem is extremely hard. Therefore, it is interesting, also our purpose in this paper, to find efficient heuristics, or special cases that are polynomial-time solvable. Heuristics can be combinatorial or experimental algorithms[1,9].

A similar genome rearrangement problem was discussed and an approximation algorithm was given by a local search for the optimal solution on a grid by Sankoff *et al*[17]. In this paper, we discuss a multiple genome rearrangement problem for generating a collection of permutations from some fixed permutation in a minimum number of signed reversals.

The rest of the paper includes five parts. (1) Definitions and models, (2) Related problems, (3) Theorems and algorithms, (4) Experimental applications, (5) Discussion and future work.

2 Mathematical model of multiple genome rearrangement

First of all, we introduce our main definitions and notations. The mathematical model of multiple genome rearrangement problem is also described.

Definition 1 For a signed permutation $p = (p_1 p_2 \cdots p_{|X|})$ on an alphabet X, a signed reversal on the segment $[i, j]$ of p is defined as the following operation from p to $r(p; i, j)$:

$$\begin{aligned} p \quad &= (p_1\ p_2\ \cdots\ p_{i-1}\ \underline{p_i\ \ p_{i+1}\ \ \cdots\ \ p_j}\ p_{j+1} \cdots\ p_{|X|}) \\ r(p;i,j) \quad &= (p_1\ p_2\ \cdots\ p_{i-1} \underline{-p_j \cdots -p_{i+1} - p_i}\ p_{j+1} \cdots\ p_{|X|}) \end{aligned}$$

Definition 2 Let T_0 be a collection of permutations. Define $N(T_0) = \{p | p = r(q; i, j)$ for some $q \in T_0, 1 \leq i < j \leq n\}$, called the reversal neighborhood of T_0. Define $N_1(T_0) = N(T_0)$, $N_2(T_0) = N(N_1(T_0))$, and $N_k(T_0) = N(N_{k-1}(T_0))$, the k-neighborhood of T_0.

Definition 3 A collection T_0 of permutations is called a $k-$Bottleneck family if for any $u, v \in T_0$, the reversal distance between u and v is at most k.

Multiple Genome Rearrangement By Signed Reversal (denoted by $(m,n)-$MGRBSR) Given two collections of permutations $P = \{p_1, p_2, \cdots, p_m\}$ and $Q = \{q_1, q_2, \cdots, q_n\}$ on an alphabet X, we generate Q from P in a minimum number of signed reversals, i.e., find a collection of signed permutations $t_r (1 \leq r \leq s)$ on X such that (1) any q_j is obtained from some p_i by a series of signed reversals $t_{r_1}, t_{r_2}, \cdots, t_{r_u}$, where each $t_{r_{j+1}}$ is obtained from t_{r_j} by one signed reversal, and (2) s is minimized. Denote $d(P, Q) = s$, the (optimal) signed reversal distance for generating Q from P.

If $m = n = 1$, i.e., $P = \{p\}$ and $Q = \{q\}$, then the problem is reduced to sorting by signed reversal and $d(p, q)$ is the signed reversal distance between p and q. In any optimal reversal process, each q_j is generated from only one p_i. Therefore, it is sufficient for us to consider the case $m = 1$. For $m > 1$, the problem can be split into m $(1, n_j)-$MGRBSR problems and similarly discussed. By rearranging the elements of X, we get $p_1 = 12 \cdots |X|$. Therefore, we discuss how to generate all permutations q_j from p_1.
$(1, n)-$MGRBSR Problem Generate all given permutations $q_j (1 \le j \le n)$ from the identity $p = 12 \cdots |X|$ in a minimum number of signed reversals.

We at first consider the $(1, 2)-$, $(1, 3)-$MGRBSR problems and then split the $(1, n)-$MGRBSR problem into some $(1, 2)-$, $(1, 3)-$MGRBSR problems.

3 Related problems

Our $(1, n)-$MGRBSR problem is similar to the genome rearrangement problem discussed by Sankoff *et al*[17] and is closely connected to the following problems[11,12,20].
Multiple alignment Given some sequences, find the alignment with minimum pairwise score.

In our $(1, n)-$MGRBSR problem, we do not consider the pairwise score, but the minimum score of Steiner trees on the given permutations.
Sorting by reversal Given a permutation p, transform it into the identity permutation in a minimum number of signed (or unsigned) reversals.

It generates the identity permutation from a given permutation in a minimum number of signed (or unsigned) reversals. Our $(1, n)-$MGRBSR problem generalizes the problem to generating a collection of permutations.
Star alignment Given some sequences, find one median sequence such that the total alignment score between the median sequence and each given sequence is minimized.

Our $(1, n)-$MGRBSR problem may contain a number of median sequences.
Fixed topology alignment[20] Given some sequences and a topological structure (usually, a tree) T. Each leaf of T is labeled by one given sequence. Assign one sequence to each internal node of T such that the total alignment score for all edges of T is minimized.

Our $(1, n)-$MGRBSR problem is not restricted to a fixed topology and it is a topology-free alignment problem.

4 Theorems and algorithms

In this part, we find algorithms for $(1,n)-$MGRBSR problems. We at first consider $(1,2)-$ and $(1,3)-$MGRBSR problems. If a $(1,2)-$MGRBSR problem contains a pair of close permutations, then we get a polynomial algorithm. If a $(1,3)-$MGRBSR problem consists of two pairs of close permutations, then we also get a polynomial algorithm. Based on these polynomial algorithms, we design approximation algorithms for the general $(1,2)-$ and $(1,3)-$MGRBSR problems, respectively. Next, we discuss a $k-$Bottleneck family for a $(1,n)-$MGRBSR problem. Finally, we split a $(1,n)-$MGRBSR problem into some $(1,2)-$ and $(1,3)-$MGRBSR problems and obtain an approximation algorithm for the general $(1,n)-$MGRBSR problem.

First of all, it is shown that [1]

Theorem 1 The $(1,1)-$MGRBSR problem is solvable in a run time $O(|X|)$.

A linear-time algorithm is presented for computing the reversal distance between two signed permutations [1] (finding the optimal reversal process still costs $O(|X|^2)$). We denote it BMY algorithm and will use it in our algorithms.

We easily have the following approximation algorithm. We at first construct a weighted graph with all given permutations as its vertices. All pairs of the given permutations form its edges. The weight of an edge is defined as the reversal distance of the pair of permutations representing the edge, which is computed by the BMY algorithm. Next we find a minimum weight spanning tree of the graph. Finally, all permutations can be generated from a given permutation along the edges of the spanning tree. With the theorem and the BMY algorithm, the run time is reduced. The steps are stated in the following.

Algorithm A

Input Sequences: $p, q_1, q_2, \cdots, q_n$.

Output Reversal process.

```
Step 1   Apply the BMY algorithm to construct a graph
         $G = (V, E, W)$ with $V = \{p, q_1, q_2, \cdots, q_n\}$, $E =$
         $\{[u,v]|\ u, v \in V, u \neq v\}$, and $W([u,v]) = d(u,v)$.
Step 2   Find a minimum weight spanning tree $T$ of $G$.
Step 3   Generate all permutations from $p$ along $T$.
```

Theorem 2 Algorithm A finds an approximated solution for any $(1,n)-$MGRBSR problem in a run time $O(n^2|X| + n|X|^2)$.

Proof Step 1 has a run time $O(n^2|X|)$ since it takes $O(|X|)$ time to find each $W([u,v])$ and G has $O(n^2)$ edges. Step 2 has a run time $O(n\ log\ n)$ to find T. Step 3 has a run time $O(n|X|^2)$ since it takes a run time $O(|X|^2)$ to find the optimal reversal process for each edge $[u,v]$ and there are $n-1$ edges T.

It is obvious that the algorithm is a 2-approximation, i.e., the approximated distance is within two times of the optimal one. Furthermore, The number of reversals can be decreased by introducing some median permutations. Suppose we want to generate q_1 and q_2 from p. We at first generate a median permutation q_0 from p, then generate q_1 and q_2 from q_0, respectively. When q_0 is properly chosen, the number of reversals can be improved. The median permutation q_0 is called a Steiner permutation. If we want to generate a collection of permutations, many Steiner permutations will be applied so as to minimize the total reversal distance. These Steiner permutations are called optimal if they minimize the total reversal distance. For a $(1, n)-$MGRBSR problem, there may be $n-1$ optimal Steiner permutations.

In order to find an optimal Steiner permutation q_0 for a $(1, 2)-$MGRBSR problem, we can try each permutation on X and finally get it. However, there are $|X|!$ permutations, so the run time is at least $|X|!$. We find some special cases that are polynomial-time solvable.

Theorem 3 (1) Let $V = \{p, q_1, q_2\}$. Assume q_0 is an optimal Steiner permutation. Then $d(q_0, x) \leq d(x, y)$ for any $x, y \in V$.

(2) If $V = \{p, q_1, q_2\}$ contains a pair with a reversal distance at most k, then an optimal Steiner permutation q_0 is found in a run time $O(|X|^{2k+1})$.

(3) If $V = \{p, q_1, q_2, q_3\}$ consists of two pairs with reversal distances at most k, then two optimal Steiner permutations are found in a run time $O(|X|^{4k+1})$.

Proof (1) By contradiction. Suppose that $d(q_0, q_i > d(q_1, q_2)(i = 1, 2)$. Then $d(p, q_0) + d(q_0, q_1) + d(q_0, q_2) > d(p, q_1) + d(q_1, q_2)$, a contradiction.

(2) By (1), the optimal Steiner permutation $q_0 \in N_k(x)$ for some $x \in V$. Since $|N(x)| \leq |X|^2$ and $|N_k(x)| \leq |X|^2|N_{k-1}(x)| \leq |X|^{2k}$. We need a run time $O(|X|)$ to find $d(y, V)(y \in N_k(x))$. Therefore, the run time is $O(|X|^{2k+1})$.

(3) Suppose that p and q_1, and q_2 and q_3 have reversal distances at most k. By (2), for each pair $q_{01} \in N_k(p)$ and $q_{02} \in N_k(q_3)$, compute the total reversal distance from q_{01} and q_{02} to p,q_1,q_2 and q_3. We then get the optimal pair q_{01} and q_{02} as the Steiner permutations. By (2), the run time is $O(|X|^{4k+1})$.

Based on Theorem 3, we can find an optimal Steiner permutation in the k-neighborhood of a permutation in the closest pair for a $(1, 2)-$MGRBSR problem. For a $(1, 3)-$MGRBSR problem, we can also find two optimal Steiner permutations in the k-neighborhoods of two permutations, each of which corresponds to a closest pair. We have the following algorithms (see Figure 1).

Algorithm B1

Input Sequences: p, q_1, q_2 (with a pair of reversal distance at most k).

Output Optimal reversal process.

```
Step 1   Find the pair, say $p$ and $q_1$, with the minimum reversal
         distance (at most $k$).
```

```
Step 2   Loop over all $u \in N_k(p)$ and update the reversal distance
         $d = d(u,p) + d(u,q_1) + d(u,q_2)$ and $q_0 = u$ if a better
         one is found.
Step 3   Generate $q_0$ from $p$, $q_1$ and $q_2$ from $q_0$ by BMY algorithm.
```

Algorithm B2
Input Sequences: p, q_1, q_2, q_3 (two pairs with reversal distances $\leq k$)
Output Optimal reversal process.

```
Step 1   Find the pairs, say $p, q_1$ and $q_2, q_3$, with two minimum
         reversal distances (at most $k$).
Step 2   Loop over $u \in N_k(p), v \in N_k(q_3)$. Update the total reversal
         distance $d = d(u,v) + d(u,p) + d(u,q_1) + d(v,q_2) + d(v,q_3)$
         if a better one is found. Also update $q_{01} = u$ and $q_{02} = v$.
Step 3   Generate $q_{01}$ from $p$, $q_1$ from $q_{01}$, $q_{02}$ from $q_{01}$, $q_1$ and $q_2$
         from $q_{02}$ by the BMY algorithm.
```

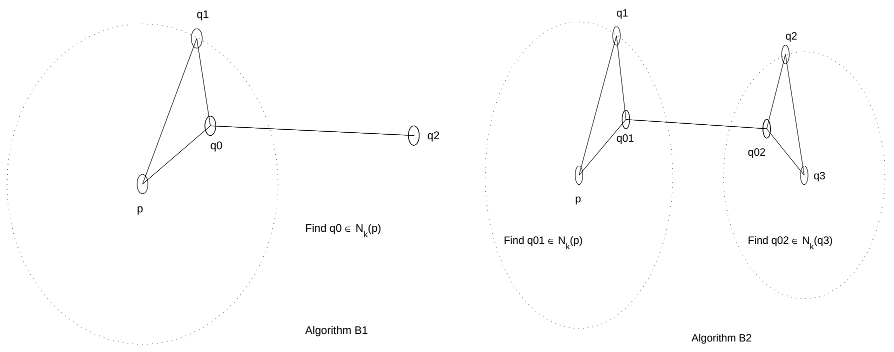

Figure 1: Algorithm B1/B2: Each optimal Steiner permutation is located in some $N_k(x)$

Figure 1 shows that Algorithm B1 (or B2) finds the optimal Steiner permutations in some $N_k(x)$ and terminates within $O(|X|^{2k+1})$ (or $O(|X|^{4k+1})$) run time. Similarly, we generate a collection of close permutations.

Theorem 4 Let $V = \{p, q_1, q_2, \cdots, q_s\}$ be a $k-$Bottleneck family. Then all Steiner permutations can be found in $O(|X|^{2k(s-2)+1})$ run time.

Based on Theorem 4, we can find the optimal reversal process for a small collection of permutations.

Algorithm C
Input Sequences: $p, q_1, q_2, \cdots, q_s$ ($k-$Bottleneck).
Output Steiner permutations and reversal process.

```
Step 1   Find $N_k(p)$.
Step 2   Loop over all $x_1, x_2 \cdots, x_{s-2} \in N_k(p)$ and update the
         total reversal distance if a better minimum spanning tree
         is found for $\{p, q_1, q_2, \cdots, q_s; x_1, x_2 \cdots, x_{s-2}\}$.
```

By Theorem 4, we find the $s-2$ optimal Steiner permutations in a run time $O(|X|^{2k(s-2)+1})$. For collections that are not $k-$Bottleneck families, we design two approximation algorithms to find their optimal Steiner permutations on the grids constructed from a series of optimal reversal paths.

Algorithm D1

Input Sequences: p, q_1, q_2.

Output Steiner permutations and reversal process.

```
Step 1   Choose a minimum reversal distance pair, say $p, q_1$.
Step 2   Find the optimal reversal path $P_1$ from $p$ to $q_1$.
Step 3   For $i \geq 2$, find $M_i \in N_k(P_{i-1})(k = 1, 2, 3)$ minimizing
         $d(M_i, p, q_1, q_2) = d(M_i, p) + d(M_i, q_1) + d(M_i, q_2)$.
         Find an optimal reversal path $P_i$ from $p$ to $M_i$, then to $q_2$.
Step 4   For each $u$ in $P_d$(the last path in Step 3), find an optimal
         reversal path $W$ from $u$ to $q_2$. We get $W_1, W_2, \cdots, W_t$.
Step 5   For each $x$ in all $W_i$, do a local optimal search to find
         $q_0 \in N_k(x)$ minimizing $d(q_0, p, q_1, q_2)$.
Step 6   Choose the best $q_0$ in Step 5 as the global optimal solution.
Step 7   Generate $q_0$ from $p$, $q_1$ and $q_2$ from $q_0$ by BMY algorithm.
```

Algorithm D2

Input Sequences: p, q_1, q_2, q_3.

Output Steiner permutations and reversal process.

```
Step 1   Choose two minimum reversal distance pairs: $(p, q_1), (q_2, q_3)$.
Step 2   Find optimal reversal paths, $P_1 : p$ to $q_1$, and $Q_1 : q_2$ to $q_3$.
Step 3   For $i \geq 2$, find $M_i \in N_k(P_{i-1})$, $N_i \in N_k(Q_{i-1})(k = 1, 2, 3)$
         minimizing $d(M_i, N_i)$. Find optimal reversal paths, $P_i : p$ to
         $M_i$, then to $q_2$, and $Q_i : q_2$ to $N_i$, then to $q_3$.
Step 4   For each $u \in P_d, v \in Q_c$ (the final paths), find an optimal
         reversal path $W$ from $u$ to $v$. We get $W_1, W_2, \cdots, W_t$.
Step 5   For each pair $x, y$ in the grid, do a local optimal search to
         find $q_{01} \in N_k(x), q_{02} \in N_k(y)$ minimizing $d(q_0, p, q_1, q_2)$.
Step 6   Choose the best $q_{01}, q_{02}$ in Step 5 as the optimal solution.
Step 7   Generate $q_{01}$ from $p$, $q_1$ from $q_{01}$, $q_{02}$ from $q_{01}$, and $q_1$ and
         $q_2$ from $q_{02}$ by the BMY algorithm.
```

In fact, in Algorithm D1, we find a series of paths from p to q_1 such that they get closer and closer to q_2. Then construct a grid by using q_2 and the closest path to q_2. Finally, do local optimal searches on the grid.

In Algorithm D2, we find two collections of paths from p to q_1, and from q_2

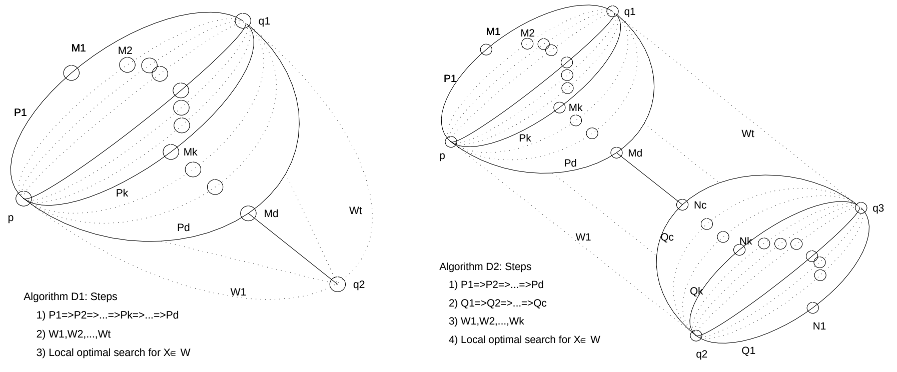

Figure 2: Algorithm D1/D2: Their steps

to q_3, respectively, such that the two collections of paths get closer and closer. Then construct a grid by using the two closest paths. Finally, do local optimal searches on the grid (see Figure 2).

Theorem 5 (1) For any p, q_1, q_2, the approximated Steiner permutation q_0 and reversal process can be found in a run time $O(|X|^{2(k+1)})$.

(2) For any p, q_1, q_2, q_3 the approximated Steiner permutations q_{01} and q_{02} and reversal process can be found in a run time $O(|X|^{2(k+1)})$.

Proof (1) Algorithm D1 at first finds an optimal reversal path from p to q_1. In the next step, it finds another optimal reversal path from p to q_1 that is closer to q_2. After some steps, it constructs a grid by using q_2 and the last optimal reversal path. The algorithm tries each possible permutation x in the grid and then finds an approximated Steiner permutation q_0 from some $N_k(x)$. Each path has length at most $|X|$ and the algorithm goes for at most $|X|$ paths. For each u in the paths, $|N_k(u)| = O(|X|^{2k})$. So the algorithm terminates in $O(|X|^{2(k+1)})$ run time.

(2) Similar to (1).

Based on Algorithm D1/D2 and Theorem 5, we designed an approximation algorithm for finding the Steiner permutations for the $(1, n)-$MGRBSR problem. The main idea is splitting the $(1, n)-$MGRBSR problem into a collection of $(1, 2)-$ and $(1, 3)-$MGRBSR problems. For any given permutations, $p, q_1, q_2, \cdots, q_n$, we at first find a minimum matching $A_i = \{x_i, y_i\}(1 \leq i \leq c)$ such that (1) $x_i, y_i \in \{p, q_1, q_2, \cdots, q_n\}$, and (2) $\sum_i d(x_i, y_i)$ is minimized. Then we find a minimum matching $w_j = \{u_j, v_j\}(j = 1, 2, \cdots, d)$ such that (1) $u_j, v_j \in \{A_i | i = 1, 2, \cdots, c\}$, (2) $\sum_j d(u_j, v_j)$ is minimized. Next, we apply Al-

gorithm D1/D2 to find two Steiner permutations q_{j1}, q_{j2} for u_j and v_j. Finally, replace all $\{u_j, v_j\}$ by all $\{q_{j1}, q_{j2}\}$ and repeat the process until it terminates.
Algorithm E (See Figure 3)
Input Sequences: $p, q_1, q_2, \cdots, q_n$.
Output Steiner permutations and reversal process.

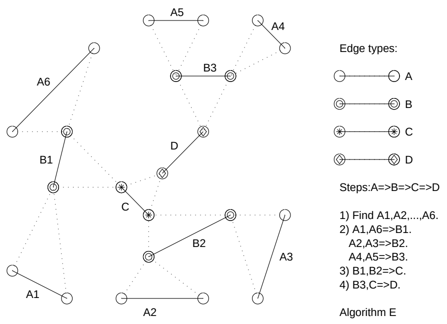

Figure 3: Algorithm E

Theorem 6 Algorithm E approximates the optimal Steiner permutations and the reversal process in a run time $O(|X|^{2(k+1)}n^2)$.

5 Experimental applications

Based on our algorithms, we design a computer program. Applying the program to some specific permutations, we obtain the optimal Steiner permutations for the permutations with different lengths. These examples show that our approximation algorithms are efficient. The three permutations, p, q_1, q_2, are chosen from the genomes of human, sea urchin, and fruit fly, respectively.

$p =$ 26 13 17 12 − 24 15 18 − 2 − 16 − 3 4 − 28 7 5 1 10 19 25 22 11 29 14 20 − 21 − 8 6 30 − 23 9 27.

$q_1 =$ 26 4 25 22 5 1 − 28 19 11 29 20 − 21 6 9 27 8 30 23 − 24 16 14 − 2 3 15 − 7 10 13 17 12 18.

$q_2 =$ −26 − 27 12 − 24 15 18 − 3 4 13 5 7 1 10 19 2 25 16 29 8 9 −20 − 11 − 22 30 23 21 6 28 17 − 14.

By our program, we obtain an optimal Steiner permutation.

$q_0 =$ 26 − 2 − 14 − 29 − 11 − 22 − 25 − 19 − 10 − 1 − 5 13 17 12 − 24 15 18 − 7 28 − 4 3 16 20 − 21 − 8 6 30 − 23 9 27.

If we choose the first k genes of p, q_1, q_2 and apply the program for $k = 5$, 10, 15, 20, 25, then we obtain the optimal Steiner permutations $q_0(k)$ from $p(k), q_1(k), q_2(k), q_3(k)$, respectively.

$p(5) =$ $-2\ -3\ 4\ 5\ 1.$
$q_1(5) =$ $4\ 5\ 1\ -2\ 3.$
$q_2(5) =$ $-3\ 4\ 5\ 1\ 2.$
$q_0(5) =$ $-2\ -1\ -5\ -4\ 3.$
$p(10) =$ $-2\ -3\ 4\ 7\ 5\ 1\ 10\ -8\ 6\ 9.$
$q_1(10) =$ $4\ 5\ 1\ 6\ 9\ 8\ -2\ 3\ -7\ 10.$
$q_2(10) =$ $-3\ 4\ 5\ 7\ 1\ 10\ 2\ 8\ 9\ 6.$
$q_0(10) =$ $-10\ -1\ -5\ -7\ -4\ 3\ 2\ -8\ 6\ 9.$
$p(15) =$ $13\ 12\ 15\ -2\ -3\ 4\ 7\ 5\ 1\ 10\ 11\ 14\ -8\ 6\ 9.$
$q_1(15) =$ $4\ 5\ 1\ 11\ 6\ 9\ 8\ 14\ -2\ 3\ 15\ -7\ 10\ 13\ 12.$
$q_2(15) =$ $12\ 15\ -3\ 4\ 13\ 5\ 7\ 1\ 10\ 2\ 8\ 9\ -11\ 6\ -14.$
$q_0(15) =$ $13\ 12\ 15 - 2 - 10 - 1 - 5 - 7 - 4\ 3\ 11 - 9 - 6\ 8 - 14.$
$p(20) =$ $13\ 17\ 12\ 15\ 18 - 2 - 16 - 3\ 4\ 7\ 5\ 1\ 10\ 19\ 11\ 14\ 20 - 8\ 6\ 9.$
$q_1(20) =$ $4\ 5\ 1\ 19\ 11\ 20\ 6\ 9\ 8\ 16\ 14 - 2\ 3\ 15 - 7\ 10\ 13\ 17\ 12\ 18.$
$q_2(20) =$ $12\ 15\ 18 - 3\ 4\ 13\ 5\ 7\ 1\ 10\ 19\ 2\ 16\ 8\ 9 - 20 - 11\ 6\ 17 - 14.$
$q_0(20) =$ $-20 - 14 - 11 - 19 - 10 - 1 - 5 - 7 - 4\ 3\ 16\ 8\ 13\ 17$
$12\ 15\ 18 - 2\ 6\ 9.$
$p(25) =$ $13\ 17\ 12\ -24\ 15\ 18\ -2\ -16\ -3\ 4\ 7\ 5\ 1\ 10\ 19\ 25\ 22\ 11$
$14\ 20\ -21\ -8\ 6\ -23\ 9.$
$q_1(25) =$ $4\ 25\ 22\ 5\ 1\ 19\ 11\ 20\ -21\ 6\ 9\ 8\ 23\ -24\ 16\ 14\ -2\ 3\ 15$
$-7\ 10\ 13\ 17\ 12\ 18.$
$q_2(25) =$ $12 - 24\ 15\ 18 - 3\ 4\ 13\ 5\ 7\ 1\ 10\ 19\ 2\ 25\ 16\ 8\ 9\ -20 - 11$
$-22\ 23\ 21\ 6\ 17\ -14.$
$q_0(25) =$ $2\ -18\ -15\ 24\ -12\ -17\ -13\ 25\ 22\ 11\ 20\ -21\ 5\ 1\ 10$
$19\ -14\ -16\ -3\ 4\ 7\ -8\ 6\ -23\ 9.$

n	$d(p,q_1)$	$d(p,q_2)$	$d(q_1,q_2)$	Optimal $d(p,\{q_1,q_2\})$
5	3	2	3	4
10	8	8	9	15
15	12	9	14	19
20	15	13	19	26
25	19	18	24	33
30	21	22	29	40

The optimal reversal distances are computed by our program. They are almost the same as the lengths of the Steiner trees in a metric space. For example, for $n = 10$, we have $(d_1, d_2, d_3) = (8, 8, 9)$. In the Euclidean metric space, we compute the optimal Steiner tree and find that its length is 14.5. The optimal $d(q_0, p) + d(q_0, q_1) + d(q_0, q_2) = 15$. Both are close each other.

Our approximation algorithms can find the optimal solutions for most collections of genomes. In many cases, they are more efficient than the one on local search for optimal solution on a grid[17].

6 Discussion and future work

In this paper, we discuss a $(1, n)-$MGRBSR problem. We design some polynomial algorithms for several special cases and some efficient approximation algorithms for the general problem. The $(1, n)-$MGRBSR problem is one of the most important problems in comparative genomics. We are interested in designing more efficient approximation algorithms for finding optimal solutions for the general $(1, n)-$MGRBSR problem. The problem is very similar to Steiner tree problems in a metric space. With the application of Steiner tree theory, the problem can be solved efficiently. On the other hand, stochastics can also be applied to the discussion of the $(1, n)-$MGRBSR problem. This will be the subject of our future work.

Acknowledgement This work is supported by the NIH grant RO1 GM62118 (to X.G.) and Wu is supported in part by NSF of China (19771025).

References

1. Bader, D.A., Moret, B.M.E. and Yan, M., *A linear-time algorithm for computing inversion distances between signed permutations with an experimental study.* Proc. 7th Workshop on Algorithms and Data Structures (WADS 01), Providence (2001), to appear in Lecture Notes in Computer Science, Springer Verlag.
2. Bafna,V. and Pevzner, P. 1994. *Genome rearrangements and sorting by reversals.* In Proc. 34th IEEE Symp. of the Foundations of Computer Science, 148-157. IEEE Computer Society Press.
3. Bafna, V. and Pevzner, P. 1995. *Sorting permutations by transpositions.* Proceedings of the 6th Annual Symposium on Discrete Algorithms, pages 614-623. ACM Press, January 1995.
4. Bafna, V. and Pevzner, P. 1996. *Genome rearrangements and sorting by reversals.* SIAM Journal on Computing, 25(2):272-289.
5. Berman, P. and Hannenhalli, S. 1996. *Fast sorting by reversal.* Proc. Combinatorial Pattern Matching (CPM), 168-175. Also Lecture Notes in Computer Science.1075.
6. Berman, P., and Karpinski, M. 1998. *On some tighter inapproximability results.* Technical Report TR98-065, ECCC.
7. Caprara, A. 1997. *Sorting by Reversals is Difficult.* Proceedings of the First Annual International Conference on Computational Molecular Bi-

ology (RECOMB'97), ACM Press.

8. Caprara, A. 1999. *Formulations and hardness of multiple sorting by yeversals.* Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB'99), ACM Press.
9. Caprara, A. and Lancia, G. 2000. *Experimental and Statistical Analysis of Sorting by Reversals.* in D. Sankoff and J.H. Nadeau (eds.) Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Kluwer Academic Publishers.
10. Christie, D. A. 1998. *A 3/2-approximation algorithm for Sorting by reversals.* Proc. 9th Ann. ACM-SIAM Symp. on Discrete Algorithms, ACM-SIAM, 244-252.
11. Durbin, R., Eddy, S. R., Krogh, A. and Mitchison, G., 1998. Biological sequence analysis: Probabilistic models of proteins and nucleic acids, Cambridge University Press.
12. Gusfield,D., Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.
13. Hannenhalli, S. and Pevzner, P. 1995. *Transforming men into mice (polynomial algorithm for genomic distance problems.* Proc. IEEE Symp. of the Foundations of Computer Science.
14. Hannenhalli, S. and Pevzner, P. 1995. *Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals).* Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 178-189.
15. Kaplan, H., Shamir, R. and Tarjan, R. E. 1997. *Faster and simpler algorithm for sorting signed permutations by reversals.* Proc. eighth annual ACM-SIAM Symp. on Discrete Algorithms (SODA 97). ACM Press.
16. Kaplan, H., Shamir, R. and Tarjan, R. E. 2000. *Faster and simpler algorithm for sorting signed permutations by reversals.* SIAM Journal on Computing, 29(3):880-892.
17. Sankoff, D., Sudaram, G. and Kececioglu, J. 1996. *Steiner points in the space of genome rearrangements.* International Journal of the Foundations of Computer Science, 7:1-9.
18. Sankoff,D. and Nadeau,J.H. . 2000. *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics.* Kluwer Academic Publishers.
19. Wu, S. and Gu. X., 2001. *A greedy algorithm for optimal recombination.* Lecture Notes on Computer Science, 2108:86-90.
20. Wang,L., B. Ma and M. Li, 2000, *Fixed topology alignment with recombination, Discrete Applied Mathematics* 104: 281-300