A New Approach for Approximating the Transposition Distance

M. E. M. T. Walter
University of Brasília
Department of Computer Science
Brasília, DF, Brazil
emilia@dcc.unb.br

Z. Dias
University of Campinas
Institute of Computing
Campinas, São Paulo, Brazil
zanoni@ic.unicamp.br

J. Meidanis
University of Campinas
Institute of Computing
Campinas, São Paulo, Brazil
meidanis@ic.unicamp.br

Abstract

One of the proposed ways to compare genomes or other large DNA molecules is by computing a rearrangement distance, defined as the minimum number of rearrangement events necessary to transform one molecule into another, taking into account only the relative order of similar genes. In this work we study the problem of computing the transposition distance between two linear gene orders, represented by permutations. To help solve it, we present a very simple structure, the breakpoint diagram, and a 2.25-approximation algorithm for the problem based on this structure. While there are better approximation algorithms, they are based on more complex data structures. Our algorithm was implemented in the C programming language and we show experimental results obtained with it on all permutations of up to 11 genes, plus selected permutations of higher size.

1. Introduction

With the advent of fast sequencing techniques, we are witnessing today a spectacular increase in the quantity of molecular data (DNA and protein sequences). The great challenge we face now is how to process this huge amount of data and extract from it relevant biological information that could help design drugs, understand life and disease, improve crops, and so on. One way to structure this information is by comparative genomics, where we analyze data coming from distinct species and learn from the similarities and differences in related genomes. Among the several proposed ways of comparing genomes, the area of **genome rearrangements** has received a lot of attention recently [1, 2, 8, 10, 9, 6, 13, 15, 4, 5, 16]. In this area, very large DNA molecules (usually entire chromosomes or large pieces of chromosomes) are investigated with respect to the relative order of genes in them. The goal is to determine a rearrangement distance, which is the minimum number of rearrangement events that could explain the differences between two such DNA molecules.

Many different events have been considered. Reversals, transpositions and translocations are the best studied ones from a theoretical point of view, altough in practice events such as duplications and deletions are at least as important. As far reversals are concerned, Hannenhalli and Pevzner presented the first polynomial time algorithm [10], subsequently improved by Kaplan, Shamir, and Tarjan [12]. Caprara showed that the reversal problem is NP-hard if we disregard the orientation of genes [3]. Hannenhalli and Pevzner also solved in polynomial time a multichromosomal problem involving translocation, fusion and fission [9]. Bafna and Pevzner [2] studied the transposition distance between two linear unsigned chromosomes, presenting several approximation algorithms, the best one having approximation factor 1.5 and running in $O(n^2)$ time, and suggesting some open problems. Guyer, Heath, and Vergara [7] implemented several algorithms for computing the transposition distance, based on subsequences and runs in a permutation. Christie [5] devised an alternative 1.5-approximation algorithm that runs in $O(n^4)$ time. The computational complexity of determining the transposition distance is still open: nobody knows whether the problem is polynomial.

In this work we present a structure named the *breakpoint diagram*, and, based on it, a 2.25-approximation algorithm for computing the transposition distance between two permutations. The algorithm runs in $O(b^2)$ time, where b is the number of breakpoints in the diagram (see Section 2). We also present experimental results. While our algorithm exhibits a larger approximation factor compared to the Bafna and Pevzner algorithm, the implementation is simpler because it relies on the breakpoint diagram, while the one by Bafna and Pevzner requires the computation of cycles and their properties, a more involved task. Christie's algorithm is also based on cycles, and has a higher running time.

In the following sections we present definitions, the approximation algorithm, experimental results, and conclusions and plans for future work.

2. Definitions

This section briefly reviews the standard definitions in the field. For more detailed definitions see, for instance, the work by Bafna and Pevzner [2].

A permutation $\pi : [1..n] \mapsto [1..n]$ denotes the sequence of genes in a chromosome. We represent permutations as lists:

$$(\pi_1 \ \ \pi_2 \ \ \ldots \ \ \pi_n).$$

We define a **transposition** as an operation that removes a contiguous block of π and places it elsewhere. For $1 \leq i < j < k \leq n+1$, we define $\varrho(i,j,k)$ as the transposition that removes the block from π_i to π_j (including π_i but excluding π_j) and places it right before π_k:

$$\varrho(i,j,k) \cdot \pi = (\pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_{k-1} \pi_i \ldots \pi_{j-1} \pi_k \ldots \pi_n)$$

We are interested in the minimum number of transpositions that transform one permutation into the identity (1 2 3 ... n). This number is the **transposition distance** of π.

Before presenting the next definitions we extend permutations with two elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$. The extended permutation will still be called π.

Given a permutation π, we write $\pi_i \prec \pi_j$ if $\pi_j - \pi_i = 1$, and $i \not\prec j$ otherwise. A **breakpoint**, is a pair of adjacent elements that are not consecutive, that is, $\pi_i \not\prec \pi_{i+1}$ with $0 \leq i \leq n$. The **number of breakpoints** of π is $b(\pi)$. When we apply a transposition, the increase in the number of breakpoints is denoted by $\Delta b(\varrho, \pi) = b(\varrho \cdot \pi) - b(\pi)$. A **strip** is a maximal series of consecutive elements without a breakpoint. We write $s_1 \prec s_2$ for two strips s_1 and s_2 when $\pi_l \prec \pi_m$ where π_l is the last element of s_1 and π_m is the first element of s_2.

The observation that we can remove at most three breakpoints per transposition leads immediately to following lemma.

Lemma 2.1 *Given the permutation π, we have*

$$\frac{b(\pi)}{3} \leq d(\pi)$$

We define now a structure named **breakpoints diagram**, denoted by $D(\pi)$, of a permutation π, in the following way:

- we define a set of nodes, $V = \{p_1, p_2, \ldots, p_{b(\pi)}\}$, one for each breakpoint of π, in the order they appear in π. Let s_i denote the i-th strip, so that $p_i = s_{i-1} \cdot s_i$.
- we define three types of edges, $\rightarrow$, $\Rightarrow$ and $--\ >$: given two nodes $p_i = s_{i-1} \cdot s_i$ and $p_j = s_{j-1} \cdot s_j$ with $i < j$, we have
  - an edge $p_i \rightarrow p_j$ when $s_{i-1} \prec s_j$ and $n+1 \notin s_j$.
  - an edge $p_i \Rightarrow p_j$ when $s_{i-1} \prec s_j$ and $0 \notin s_{i-1}$.
  - an edge $p_i --\ > p_j$ when $s_{j-1} \prec s_i$.

To clarify, let us look at an example. Given the permutation $\pi = (\ 0\ 5\ 3\ 1\ 4\ 2\ 6\ )$, we have nodes $p_1 = 0.5$, $p_2 = 5.3$, $p_3 = 3.1$, $p_4 = 1.4$, $p_5 = 4.2$ and $p_6 = 2.6$, and edges $p_1 \rightarrow p_3$, $p_1 --\ > p_5$, $p_2 \Rightarrow p_6$, $p_2 --\ > p_6$, $p_3 \rightarrow p_4$, $p_3 \Rightarrow p_4$, $p_4 \rightarrow p_5$ e $p_4 \Rightarrow p_5$ (Figure 2). Observe that the edges $\rightarrow$ and $\Rightarrow$ coincide, except when their origin is the first node or when their destination is the last node. Only $\rightarrow$ can leave from the first node, and only $\Rightarrow$ can reach the last node.

Denote by $T(i,j,k)$ the transposition that "cuts" breakpoints p_i, p_j, and p_k, that is, the transposition exchanges the block containing all strips between p_i and p_j with the block containing all strips between p_j and p_k. Notice that this is not the same as $\varrho(i,j,k)$, but there are indices i', j', k' such that $T(i,j,k) = \varrho(i',j',k')$. Observe that $T(i,j,k)$ has one of the following effects:

- it removes three breakpoints, when $p_i \rightarrow p_j$, $p_j \Rightarrow p_k$ and $p_i --\ > p_k$,
- it removes two breakpoints, when $p_i \rightarrow p_j$ and $p_j \Rightarrow p_k$ but $p_i \not--\ > p_k$, or $p_i \rightarrow p_j$ and $p_i --\ > p_k$, but $p_j \not\Rightarrow p_k$, or yet when $p_j \Rightarrow p_k$ and $p_i --\ > p_k$ but $p_i \not\rightarrow p_j$,
- it removes a single breakpoint, when there is only one of the three edges between these three nodes, and there is no one of the two other edges.
- it removes no breakpoints, when there are no edges among these three nodes.

An x-transposition, for $x \in \{-3,-2,-1,0,1,2,3\}$, is a transposition ϱ such that $\Delta b(\varrho, \pi) = x$.

Lemma 2.2 *Given a permutation π, if $V \neq \emptyset$ on $D(\pi)$, then $|V| \geq 3$.*

Lemma 2.3 *Given a permutation π, if $r = |V| \geq 4$ on $D(\pi)$, then there are at least four edges $p_1 \rightarrow p_j$, $p_i \Rightarrow p_r$, $p_1 --\ > p_l$ and $p_m --\ > p_r$, with $2 \leq i,j,l,m \leq r-1$, where i,j,l,m are not necessarily distinct.*

In terms of the breakpoints diagram, sorting by transpositions means transforming V into an empty set with the least number of operations, that is, we want to remove all nodes of V with the minimum number of steps. From Lemma 2.3 we have the following result.

Lemma 2.4 *Given a permutation π, if $|V| \geq 4$ on $D(\pi)$, then there is always a -1-transposition.*

From Lemma 2.4 we have an upper bound for $d(\pi)$.

Theorem 2.1 *Given a permutation π, we have*

$$d(\pi) \leq b(\pi)$$

3. An approximation algorithm

First, we observe that using Lemma 2.1, Lemma 2.3 and Theorem 2.1, we have immediately a 3-approximation algorithm for the problem of sorting by transpositions.

But, in this section, we will present a very simple approximation algorithm with factor 2.25, based on the breakpoint diagram.

We can verify that on a diagram with $|V| = 3$ we have necessarily a -3-transposition, while on a diagram with $|V| = 4$ we have necessarily a -1-transposition, followed by a -3-transposition.

We can verify still that when $|V| = 5$ we have only one permutation, $\pi = (4\ 3\ 2\ 1)$, generating a diagram that needs two -1-transpositions, followed by a -3-transposition. All the other permutations generate diagrams that need a -2-transposition followed by a -3-transposition.

The next two results show that, when $V \geq 6$, we can remove at least four breakpoints on three steps.

Theorem 3.1 *Given the permutation π, and $D(\pi)$, with $|V| = r \geq 6$, having no -2-transpositions nor -3-transpositions, then it is possible to remove at least four nodes on three steps.*

Proof:

On diagrams with $|V| \geq 4$ there is always a -1-transposition (Lemma 2.4).

We have two general forms of diagrams (Figure 3), a form where node $(r-1, c)$ stays on the interval between $(0, a)$ and $(b, 1)$, and another one where $(r-1, c)$ is on the interval between $(b, 1)$ and (d, r).

For the first case, we have three possibilities for the edge $(0, a) - - > (e, f)$ (Figure 4): (A) (e, f) is between $(0, a)$ and $(r-1, c)$; (B) (e, f) is between $(r-1, c)$ and $(b, 1)$; and (C) $(e, f) = (b, 1)$.

For subcase (A), we have necessarily a node $i = (g, h)$ between $(0, a)$ and (e, f). Taking $j = (r-1, c)$ and $k = (d, r)$, and applying the -1-transposition $T(i, j, k)$ we obtain a diagram with $|V| = r - 1$ (node $(r-1, r)$ is removed), on which we have a -2-transposition, $i = (0, a)$, $j = (b, 1)$ and $k = (e, f)$ (Figure 5).

For subcase (B), when (e, f) stays between $(r-1, c)$ and $(b, 1)$, we have another five cases for $(g, h) - - > (d, r)$: $(g, h) = (r-1, c)$, (g, h) between $(r-1, c)$ and (e, f), $(g, h) = (e, f)$, (g, h) between (e, f) and $(b, 1)$, and at last (g, h) between $(b, 1)$ and (d, r) (Figure 6):

- $(g, h) = (r-1, c)$: taking $i = (0, a)$, $j = (r-1, c)$ and $k = (d, r)$, the -1-transposition $T(i, j, k)$ generates a diagram with $|V| = r-1$ (node $(r-1, r)$ is removed), on which we have a -2-transposition, $i = (0, c)$, $j = (b, 1)$ and $k = (d, a)$.

- (g, h) between $(r-1, c)$ and (e, f): taking $i = (0, a)$, $j = (r-1, c)$ and $k = (d, r)$, the -1-transposition $T(i, j, k)$ generates a diagram with $|V| = r - 1$ (node $(r-1, r)$ is removed), on which we have a -2-transposition, $i = (g, h)$, $j = (e, f)$ and $k = (d, a)$.

- $(g, h) = (e, f)$: taking $i = (0, a)$, $j = (b, 1)$ and $k = (d, r)$, the -1-transposition $T(i, j, k)$ generates a diagram with $|V| = r - 1$ (node $(0, 1)$ is removed). Following, taking $i = (d, a)$, $j = (r-1, c)$ and $k = (e, f)$, and applying the -1-transposition $T(i, j, k)$ (node (e, a) is removed), we generate a diagram on which we have a -2-transposition, $i = (d, c)$, $j = (r-1, f)$ and $k = (b, r)$.

- (g, h) between (e, f) and $(b, 1)$: in this case, we have six cases for the edge $(l, m) - - > (b, 1)$: (l, m) between (g, h) and $(b, 1)$, (l, m) between (e, f) and (g, h), $(l, m) = (e, f)$, (l, m) between $(r-1, c)$ and (e, f), $(l, m) = (r-1, c)$ and finally (l, m) between $(0, a)$ and $(r-1, c)$ (Figure 7).

  1. (l, m) between (g, h) e $(b, 1)$: we have necessarily a node (s, t) between (l, m) and $(b, 1)$. Then, taking $i = (g, h)$, $j = (s, t)$ and $k = (d, r)$, the -1-transposition $T(i, j, k)$ generates a diagram with $|V| = r - 1$ (node (d, h) is removed), having a -2-transposition, $i = (0, a)$, $j = (b, 1)$ and $k = (l, m)$.

  2. (l, m) between (e, f) and (g, h): taking $i = (r-1, c)$, $j = (g, h)$ and $k = (d, r)$, the 0-transposition $T(i, j, k)$ generates a diagram with $|V| = r$, having a -2-transposition, $i = (0, a)$, $j = (b, 1)$ and $k = (l, m)$. This -2-transposition generates a diagram with $|V| = r - 2$ nodes (nodes $(0, 1)$ and (b, m) are removed), having already another -2-transposition, $i = (d, c)$, $j = (r-1, h)$ and $k = (g, r)$.

  3. $(l, m) = (e, f)$: taking $i = (r-1, c)$, $j = (g, h)$ and $k = (d, r)$, the 0-transposition $T(i, j, k)$ generates a diagram with $|V| = r$, having a -3-transposition, $i = (0, a)$, $j = (b, 1)$ and $k = (e, f)$.

  4. (l, m) between $(r-1, c)$ and (e, f): taking $i = (0, a)$, $j = (g, h)$ and $k = (d, r)$, the 0-transposition $T(i, j, k)$ generates a diagram with $|V| = r$, having a -2-transposition, $i = (0, h)$, $j = (b, 1)$ and $k = (d, a)$. This -2-transposition generates a diagram with $|V| = r - 2$ (nodes $(0, 1)$ and (d, h) are removed), having another -2-transposition, $i = (b, a)$, $j = (l, m)$ and $k = (e, f)$.

5. $(l,m) = (r-1,c)$: taking $i = (0,a)$, $j = (g,h)$ and $k = (d,r)$, the 0-transposition $T(i,j,k)$ generates a diagram with $|V| = r$, having a -2-transposition, $i = (0,h), j = (b,1)$ and $k = (d,a)$. This -2-transposition generates a diagram with $|V| = r-2$ (nodes $(0,1)$ and (d,h) are removed), having another -2-transposition, $i = (b,a), j = (r-1,c)$ and $k = (g,r)$.
6. (l,m) between $(0,a)$ and $(r-1,c)$: taking $i = (0,a)$, $j = (b,1)$ and $k = (d,r)$, the -1-transposition $T(i,j,k)$ generates a diagram with $|V| = r-1$ (node $(0,1)$ is removed), having a -2-transposition, $i = (l,m), j = (r-1,c)$ and $k = (b,r)$.

- (g,h) between $(b,1)$ and (d,r): we have necessarily a node (l,m) between (g,h) and (d,r). Then, taking $i = (0,a)$, $j = (b,1)$ and $k = (l,m)$, the -1-transposition $T(i,j,k)$ generates a diagram with $|V| = r-1$ (node $(0,1)$ is removed), on which we have a -2-transposition, $i = (g,h), j = (r-1,c)$ and $k = (d,r)$.

Finally, for the last subcase, $(e,f) = (b,1)$, we have yet other three cases for $(g,h) - - > (d,r)$: $(g,h) = (r-1,c)$, (g,h) between $(r-1,c)$ and $(b,1)$, and (g,h) between $(b,1)$ and (d,r) (Figure 8).

- $(g,h) = (r-1,c)$: taking $i = (0,a)$, $j = (b,1)$ and $k = (d,r)$, the -1-transposition $T(i,j,k)$ generates a diagram with $|V| = r-1$ (node $(0,1)$ is removed), having a -3-transposition, $i = (d,a)$, $j = (r-1,c)$ and $k = (b,r)$.

- (g,h) between $(r-1,c)$ and $(b,1)$: taking $i = (0,a)$, $j = (r-1,c)$ and $k = (b,1)$, the -1-transposition $T(i,j,k)$ generates a diagram with $|V| = r-1$ (node (b,a) is removed), having a -2-transposition, $i = (0,c)$, $j = (r-1,1)$ and $k = (d,r)$.

- (g,h) between $(b,1)$ and (d,r): there is necessarily a node $k = (l,m)$ between the nodes (g,h) and (d,r). Therefore, taking $i = (0,a)$ and $j = (b,1)$, the -1-transposition $T(i,j,k)$ generates a diagram with $|V| = r-1$ (node $(0,1)$ is eliminated), containing a -2-transposition, $i = (g,h)$, $j = (r-1,c)$ and $k = (d,r)$.

The other general case, when $(r-1,c)$ is on the interval between nodes $(b,1)$ and (d,r), can be reduced to the previous case as described now. Let us take between nodes $(r-1,c)$ and (d,r) the minimum label m on node (y,m). Then $m-1$ must appear necessarily on a node $(m-1,x)$ to the left of $(r-1,c)$, and the diagram have an edge $(m-1,x) \rightarrow (y,m)$. We note here that $(y,m) \neq (r-1,c)$ for if not there would have a -2-transposition $T(i,j,k)$, with $i = (m-1,x), j = (r-1,c)$ and $k = (d,r)$. Besides, if there would exist another node $(x-1,z)$ to the left of $(m-1,x)$, then the diagram would have the edge $(x-1,z) \rightarrow (m-1,x)$, and the the diagram would have -2-transposition $T(i,j,k)$, with $i = (x-1,z)$, $j = (m-1,x)$ and $k = (y,m)$. Then, node $(x-1,z)$ is necessarily to the right of $(m-1,x)$, in such a way that there exists the edge $(m-1,x) - - > (x-1,z)$. We can apply exactly the cases described for the previous general case. ■

Lemma 2.4 implies immediately our next result.

Lemma 3.1 *Given a permutation π, if on $D(\pi)$ there is a -3-transposition or a -2-transposition, then it is possible to remove at least four nodes on three steps.*

Based on Theorem 3.1 and on Lemma 3.1 we have a new upper bound for the sorting by transpositions problem.

Theorem 3.2 *Given a permutation π, the we have*

$$d(\pi) \leq \frac{3}{4} b(\pi)$$

Using Lemma 3.1 and Theorem 3.1, we have the Approx algorithm, shown on Figure 1. This algorithm has factor 2.25, according to Theorems 2.1 and 3.2.

4. Some experiments

To observe the approximation factor of algorithm Approx in practice we coded it and an exact algorithm in the C programming language and ran three kinds of experiments.

The first kind was to test all permutations of size n for n up to 11. Table 1 contains the results, listing the number of permutations for which our algorithms did not get the exact distance, which percent of the total this represents, and what was the larger discrepancy in these cases. In practice, the algorithm reached a 2.0 approximation factor. It is conceivable that with larger inputs the theoretical bound of 2.25 would be attained. We could not run larger experiments of this kind because for $n = 12$ we estimate that a machine with a 18GB RAM and a 40GB free disk space would need to run for a week to produce the results.

The second kind of test involved inverse permutations, that is, permutations of the form

$$r_n = (n \;\; n-1 \;\; n-2 \;\; \ldots \;\; 2 \;\; 1).$$

It is known that $d(r_n) = \lfloor \frac{n}{2} \rfloor + 1$ [13, 5]. The Approx algorithm performed as shown in Table 2.

Another family of permutations is: $\pi_2 = (5\ 4\ 3\ 2\ 1)$, $\pi_3 = (8\ 7\ 3\ 2\ 1\ 6\ 5\ 4)$, $\pi_4 = (11\ 10\ 3\ 2\ 1\ 6\ 5\ 4\ 9\ 8\ 7)$, $\pi_5 = (14\ 13\ 3\ 2\ 1\ 6\ 5\ 4\ 9\ 8\ 7\ 12\ 11\ 10)$, and so

Algorithm *Approx*
input: π
output: u, a sequence $\varrho_1, \varrho_2, \ldots, \varrho_u$
such that $\varrho_u \cdot \ldots \cdot \varrho_2 \cdot \varrho_1 \cdot \pi = (1\ 2\ 3 \ldots n)$

$u \leftarrow 0$
generate $D(\pi)$
while $|V| \neq \emptyset$ **do**
  $u \leftarrow u + 1$
  if there is a -3-transposition **then**
    $\varrho_u \leftarrow -3$-transposition
  else
    if there is a -2-transposition **then**
      $\varrho_u \leftarrow -2$-transposition
    else
      $\varrho_u \leftarrow -1$-transposition
        satisfying Theorem 3.1
  $\pi \leftarrow \varrho_u \cdot \pi$
  generate $D(\pi)$
return u, ϱ_1, ϱ_2, $\ldots$, ϱ_u

Figure 1. The approximation algorithm with factor 2.25.

n	Differences	Percent	Largest Discrepancy	Approx. Factor
6	6	0.83%	4/3	1.34
7	72	1.42%	6/4	1.50
8	1167	2.89%	7/4	1.75
9	14327	3.95%	7/4	1.75
10	213352	5.88%	7/4	1.75
11	2870035	7.19%	10/5	2.00

Table 1. Results for all permutations of a fixed size.

n	$d(r_n)$	Approx	Factor
7	4	5	1.3
8	5	6	1.3
9	5	6	1.3
10	6	6	1
11	6	6	1
12	7	9	1.3
13	7	9	1.3
14	8	10	1.3
15	8	11	1.3
16	9	12	1.4
17	9	12	1.4

Table 2. Results for inverse permutations.

on. This family is interesting because we suspect that these permutations produce distances arbitrarily larger than the lower bound. We ran both Approx and the exact algorithm with the first three permutations above as input, with both programs obtaining the same answer for π_2, π_3 and π_4. For π_5, we ran the approximation algorithm only, obtaining $u = 7$. The exact algorithm could not be run because of memory and time requirements. A lower bound for $d(\pi_5)$ is $\frac{b(\pi_5)}{3} = 4$, but it cannot be reached because if we draw the corresponding breakpoints diagram, we observe that there is no possible -3-transposition. So, $d(\pi_5)$ is at least 5, and the approximation factor is at most 1.4.

5. Conclusions

In this work, we presented a simple structure named the breakpoint diagram, and a 2.25-approximation algorithm to compute the transposition distance between a permutation and the identity, based on this structure. We ran experiments on some representative data and a comparison to other algorithms is planned for the near future.

Some questions arose from that study. First, can we lower the factor still using the breakpoint diagram? Bafna and Pevzner gave a 1.5-approximation algorithm, but using the cycle structure of the diagram, which requires more complex data structures. Second, can one decide whether $d(\pi) = \frac{b(\pi)}{3}$ in polynomial time? A similar question was given a positive answer for unsigned reversals [14, 11]. Finally, is there a permutation reaching the 2.25 factor?

Other open questions are the complexity of the problem, and the value of the diameter (the greatest distance between two chromosomes) as a function of n.

6. Acknowledgments

We thank the support of FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, and CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico.

References

[1] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals. In *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 148–157, 1993. Extended version published with same title in SIAM Journal of Computing, 25(2):272–289,1996.

[2] V. Bafna and P. Pevzner. Sorting by transpositions. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 614–623, 1995.

[3] A. Caprara. Sorting by reversals is difficult. Technical report, DEIS,University of Bologna, abril 1996.

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

[5] D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, University of Glasgow, Scotland, 1998.

[6] Q.-P. Gu, S. Peng, and H. Sudborough. Approximation algorithms for genome rearrangements. In *Proc. 7th Workshop on Genome Informatics-GIW'96*, 1996.

[7] S. A. Guyer, L. S. Heath, and J. P. C. Vergara. Subsequences and run heuristics for sorting by transpositions. In *4th Dimacs International Algorithm Implementation Challenge*, 1995.

[8] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30:299–311, 1995.

[9] S. Hannenhalli and P. Pevzner. Tranforming men into mice (polynomial algorithmic for genomic distance problem). *36th Annual IEEE Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[10] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 178–189, 1995.

[11] R. W. Irving and D. A. Christie. Sorting by reversals: a conjecture of Kececioglu and Sankoff. Working Paper, Dept. of Computer Science, University of Glasgow, 1996.

[12] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SODA'97*, 1997.

[13] J. Meidanis, M. Walter, and Z. Dias. Transposition distance of strictly decreasing sequences. In *Proc. 4th South American Workshop on String Processing*, pages 70–79, 1997.

[14] N. Tran. An easy case of sorting by reversals. *Journal of Computational Biology*, 5(4):741–746, 1998.

[15] J. P. Vergara. *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytechnic Institute and State University, 1997.

[16] M. E. Walter. *Algoritmos para Problemas em Rearranjo de Genomas*. PhD thesis, University of Campinas, 1999. In Portuguese.

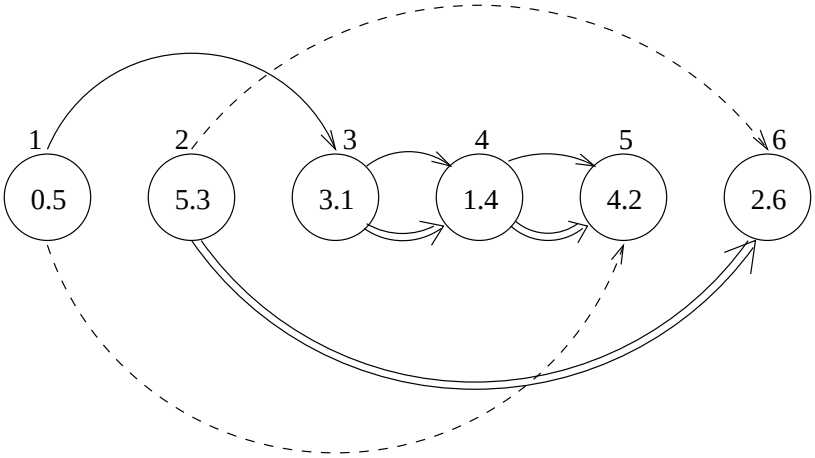

Figure 2. The breakpoints diagram $D(\pi)$, for $\pi = (\ 0\ 5\ 3\ 1\ 4\ 2\ 6\ )$.

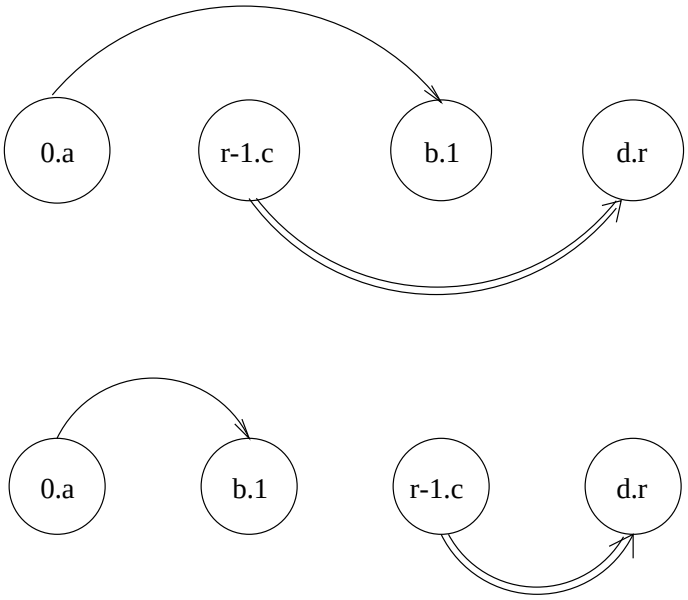

Figure 3. The unique two general forms of the breakpoint diagram when $|V| \geq 6$.

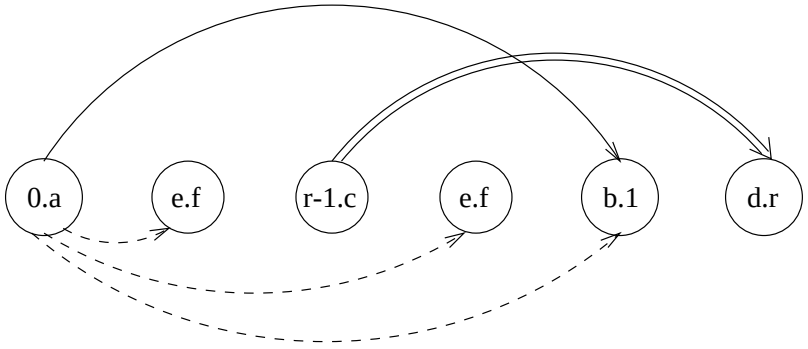

Figure 4. The general forms for the breakpoint diagrams with $|V| \geq 6$, for the three possible cases of the edge $(0, a) - - > (e, f)$, when $(r - 1, c)$ stays on the interval between $(0, a)$ and $(b, 1)$.

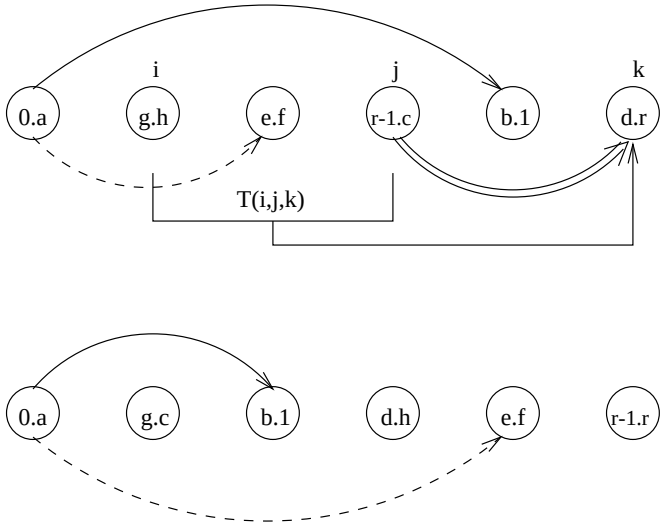

Figure 5. The -1-transposition $T(i, j, k)$ indicated in the figure generates a new diagram having a -2-transposition.

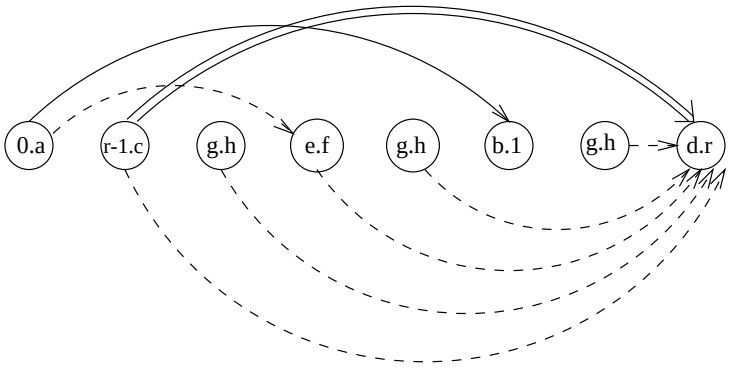

Figure 6. The general forms for the breakpoint diagrams with $|V| \geq 6$, for the five cases of the edge $(g, h) - - > (d, r)$.

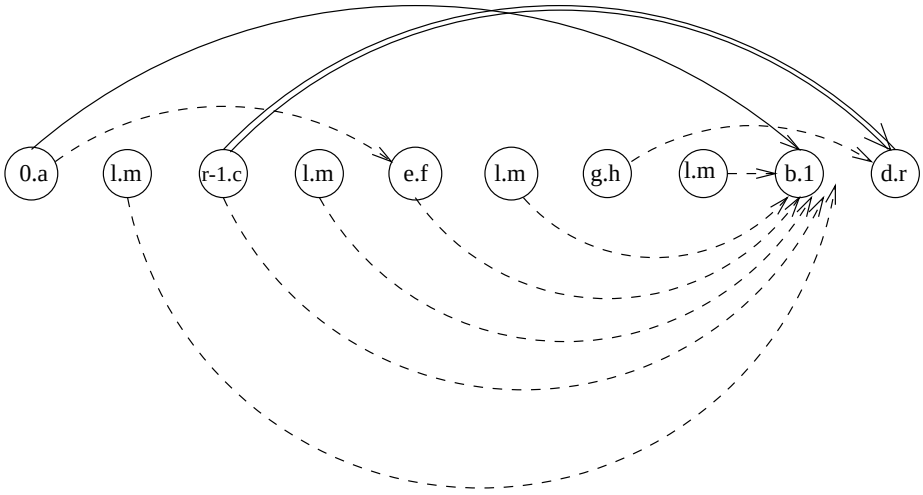

Figure 7. The general forms for the breakpoint diagrams with $|V| \geq 6$, for the six cases of the edge $(l, m) - - > (b, 1)$, when (g, h) stays on the interval (e, f) and $(b, 1)$.

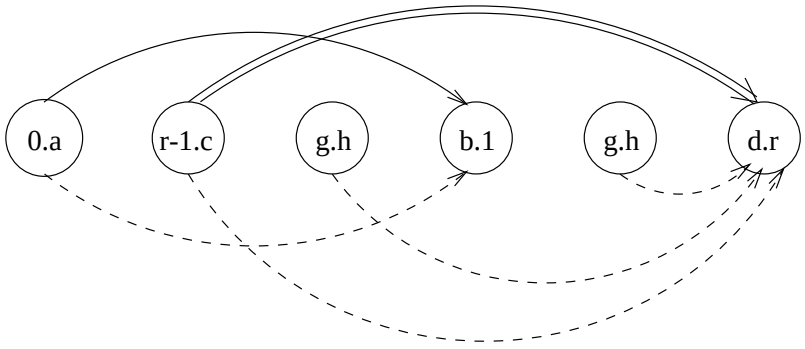

Figure 8. The general forms for the breakpoint diagrams with $|V| \geq 6$, for the three cases of the edge $(g, h) - - > (d, r)$, when $(e, f) = (b, 1)$.