Sorting by Prefix Transpositions

Zanoni Dias[1] * and João Meidanis[1] **

University of Campinas, Institute of Computing,
P.O.Box 6167, 13084-971,
Campinas, Brazil
{zanoni,meidanis}@ic.unicamp.br

Abstract. A transposition is an operation that exchanges two consecutive, adjacent blocks in a permutation. A prefix transposition is a transposition that moves the first element in the permutation. In this work we present the first results on the problem of sorting permutations with the minimum number of prefix transpositions. This problem is a variation of the transposition distance problem, related to genome rearrangements. We present approximation algorithms with performance ratios of 2 and 3. We conjecture that the maximum prefix transposition distance is $D(n) = n - \left\lfloor \frac{n}{4} \right\rfloor$ and present the results of several computational tests that support this. Finally, we propose an algorithm that decides whether a given permutation can be sorted using just the number of transpositions indicated by the breakpoint lower bound.

1 Introduction

Sequence comparison is one of the most studied problems in computer science. Usually we are interested in finding the minimum number of local operations, such as insertions, deletions, and substitutions that transform a given sequence into another given sequence. This is the edit distance problem, described in many Computational Biology textbooks [19]. Several studies, however, have shown that global operations such as reversals and transpositions (also called rearrangement events) are more appropriate when we wish to compare the genomes of two species [18].

A new research area called Genome Rearrangements appeared in the last years to deal with problems such as, for instance, to find the minimum number of rearrangement events needed to transform one genome into another. In the context of Genome Rearrangements, a genome is represented by an n-tuple of genes (or gene clusters). When there are no repeated genes, this n-tuple is a permutation. We proceed with a brief overview of the literature related to the present work.

The best studied rearrangement event is the reversal. A reversal inverts a block of any size in a genome. Caprara [5] proved that finding the minimum number of reversals needed to transform one genome into another is an NP-Hard

* Research supported by FAPESP
** Research supported in part by CNPq and FAPESP

problem. Bafna and Pevzner [3] have presented an algorithm with approximation factor 2 for this problem. Later Christie [6] presented the best known algorithm for the problem, with factor $\frac{3}{2}$.

Hannenhalli and Pevzner [10] have studied the reversal distance problem when the orientation of genes is known. In this case they proved that there is a polynomial algorithm for the problem. This algorithm has been refined successively until Kaplan, Shamir and Tarjan [14] presented a quadratic algorithm. When just the distance is needed, a faster, linear algorithm due to Bader, Moret, and Yan [2] can be used. Meidanis, Walter e Dias [17] have shown that all the reversal theory developed for linear genomes can be easily adapted to circular genomes.

Another interesting variation of this problem is the so-called prefix reversal problem or pancake problem as it was originally called [8]. In this variation only reversals involving the first consecutive elements of a genome are permitted. Heydari and Sudborough [11] have proved that this problem is NP-Hard. Gates and Papadimitriou [9] and Heydari and Sudborough [12] have studied the diameter of prefix reversals (see further details on diameter problems in Section 4).

The rearrangement event called transposition has the property of exchanging two adjacent blocks of any size in a genome. The transposition distance problem, that is, the problem of finding the minimum number of transpositions necessary to transform one genome into another, has been studied by Bafna and Pevzner [4], who presented the best approximation algorithm for the problem, with factor $\frac{3}{2}$. The transposition distance problem is still open: we do not know of any NP-Hardness proof, and there are no evidences that an exact polynomial algorithm exists. Christie [7] and Meidanis, Walter and Dias [16] have proved partial results on the transposition diameter.

In this work we present the first known results on the variation of the transposition distance problem that we call prefix transposition distance, that is, the rearrangement distance problem where only transpositions affecting two consecutive blocks of the genome, with one of these blocks formed by the first consecutive elements of the genome.

The paper is divided as follows. Initially, in Section 2, we define important concepts that will be used throughout. In Section 3 we present two approximation algorithms for the prefix transposition distance problem, with factors 3 and 2. In Section 4 we present several results on the prefix transposition diameter, leading to the conjecture that $D(n) = n - \left\lfloor \frac{n}{4} \right\rfloor$, and tests with programs that we implemented to help validate our conjectures. We show in Section 5 an algorithm that verifies whether a given genome can be sorted using the minimum number of prefix transpositions according to the breakpoint lower bound (Lemma 5). Finally, in Section 6, we exhibit our conclusions and suggestions for future work.

2 Definitions

Here we introduce a number of basic concepts used in Genome Rearrangements. Notice that some definitions, for instance that of transposition, is different from the definition used in other areas.

Definition 1. *An arbitrary genome formed by n genes will be represented as a permutation* $\pi = [\pi[1]\ \pi[2]\ \dots\ \pi[n]]$ *where each element of* π *represents a gene. The identity genome* ι_n *is defined as* $\iota_n = [1\ 2\ \dots\ n]$.

Definition 2. *A transposition* $\rho(x, y, z)$, *where* $1 \leq x < y < z \leq n+1$, *is an rearrangement event that transforms* π *into the genome* $\rho\pi = [\pi[1]\ \dots\ \pi[x-1]\ \pi[y]\ \dots\ \pi[z-1]\ \pi[x]\ \dots\ \pi[y-1]\ \pi[z]\ \dots\ \pi[n]]$.

Definition 3. *A prefix transposition* $\rho(1, x, y)$, *where* $1 < x < y \leq n+1$, *is an rearrangement event that transforms* π *into the genome* $\rho\pi = [\pi[x]\ \dots\ \pi[y-1]\ \pi[1]\ \dots\ \pi[x-1]\ \pi[y]\ \dots\ \pi[n]]$.

Definition 4. *Given two genomes* π *and* σ *we define the transposition distance* $d_\tau(\pi, \sigma)$ *between these two genomes as being the least number of transpositions needed to transform* π *into* σ, *that is, the smallest* r *such that there are transpositions* $\rho_1, \rho_2, \dots \rho_r$ *with* $\rho_r \dots \rho_2\rho_1\pi = \sigma$. *We call sorting distance by transpositions,* $d_\tau(\pi)$, *the transposition distance between the genomes* π *and* ι_n, *that is,* $d_\tau(\pi) = d_\tau(\pi, \iota_n)$.

Definition 5. *Given two genomes* π *and* σ *we define the prefix transposition distance* $d(\pi, \sigma)$ *between these two genomes as being the least number of prefix transpositions needed to transform* π *into* σ, *that is, the smallest* r *such that there are prefix transpositions* $\rho_1, \rho_2, \dots \rho_r$ *with* $\rho_r \dots \rho_2\rho_1\pi = \sigma$. *We call sorting distance by prefix transpositions,* $d(\pi)$, *the prefix transposition distance between genomes* π *and* ι_n, *that is,* $d(\pi) = d(\pi, \iota_n)$.

3 Approximation Algorithms

The first important observation is the following.

Lemma 1. *For any permutation* π, *we have* $d(\pi) \geq d_\tau(\pi)$.

Proof: This follows from the observation that every prefix transposition is a transposition. The converse is not always true. □

3.1 Approximation Algorithm with Factor 3

Lemma 2. *For every transposition* $\rho(x, y, z)$ *with* $x \neq 1$, *there are prefix transpositions* $\rho_1(1, r, s)$ *and* $\rho_2(1, t, u)$ *such that* $\rho_2\rho_1\pi = \rho\pi$.

Proof: Indeed, it suffices to take $r = y$, $s = z$, $t = z - y + 1$ and $u = z - y + x$, or, alternatively, $r = x$, $s = y$, $t = y - x + 1$ and $u = z$. Figure 1 shows how two prefix transpositions can simulate a transposition. □

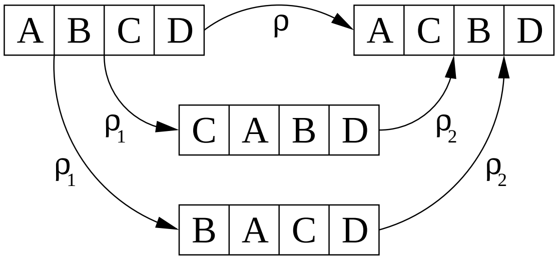

Fig. 1. Two examples of how it is possible to obtain prefix transpositions ρ_1 and ρ_2 such that $\rho\pi = \rho_2\rho_1\pi$, for a given transposition $\rho = \rho(x, y, z)$, with $x \neq 1$

Lemma 3. *Any k-approximation algorithm for the transposition distance problem can be transformed into a $2k$-approximation algorithm for the prefix transposition distance problem.*

Proof: Immediate from Lemma 2. □

Therefore it is easy to obtain an approximation algorithm with factor 3 for the prefix transposition distance problem using the approximation algorithms with factor $\frac{3}{2}$ for the transposition distance problem given by Bafna and Pevzner [4] and by Christie [7].

3.2 Approximation Algorithm with Factor 2

We need to define a few important concepts before proceeding.

Definition 6. *A breakpoint for the prefix transposition problem is a position i of a permutation π such that $\pi[i] - \pi[i-1] \neq 1$, and $2 \leq i \leq n$. By definition, position 1 (beginning of the permutation) is always considered a breakpoint. Position $n+1$ (end of the permutation) is considered a breakpoint when $\pi[n] \neq n$. We denote by $b(\pi)$ the number of breakpoints of permutation π.*

By the former definition $b(\pi) \geq 1$ for any permutation π and the only permutations with exactly one breakpoint are the identity permutations ($\pi = \iota_n$, for all n).

Definition 7. *A strip is a subsequence $\pi[i..j]$ of π ($i \leq j$) such that i and $j+1$ are breakpoints and there are no breakpoints between these positions.*

Definition 8. *Given a permutation π and a prefix transposition ρ, we define $\Delta b(\pi, \rho)$ as the variation on the number of breakpoints due to operation ρ, that is, $\Delta b(\pi, \rho) = b(\rho\pi) - b(\pi)$.*

The first important observation about breakpoints is the following.

Lemma 4. *Given a permutation π and a prefix transposition ρ, we have that $\Delta b(\pi, \rho) \in \{-2, -1, 0, 1, 2\}$.*

Lemma 5. *For every permutation π, we have that $d(\pi) \geq \left\lceil \frac{b(\pi)-1}{2} \right\rceil$.*

Proof: Immediate from Lemma 4. □

Lemma 6. *Given a permutation $\pi \neq \iota_n$, where $n = |\pi|$, it is always possible to obtain a prefix transposition ρ such that $\Delta b(\pi, \rho) \leq -1$.*

Proof: Let k be the last element of the first strip of π. If $k < n$, then there is a strip beginning with the element $k+1$, such that $\pi^{-1}[k] < \pi^{-1}[k+1]$ and $\rho = \rho(1, \pi^{-1}[k]+1, \pi^{-1}[k+1])$ suffices. If $k = n$, take $\rho = \rho(1, \pi^{-1}[k]+1, n+1)$. □

Lemma 7. *Let π be a permutation and $\rho(1, x, y)$ a prefix transposition such that $\rho\pi = \iota_n$, where $n = |\pi|$. Then $\pi[x] = 1$ and $\Delta b(\pi, \rho) = -2$.*

Lemma 8. *For every permutation π, we have $d(\pi) \leq b(\pi) - 2$.*

Proof: Immediate by Lemmas 6 and 7. □

Theorem 1. *For every permutation π, we have $\left\lceil \frac{b(\pi)-1}{2} \right\rceil \leq d(\pi) \leq b(\pi) - 2$.*

Theorem 2. *Any algorithm that produces the prefix transpositions according to Lemmas 6 and 7 is an approximation algorithm with factor 2 for the prefix transposition distance problem.*

Another important point regarding genome rearrangments is the possibility of sorting a permutation without ever increasing the number of breakpoints. Christie [7] has proved that this is true for transposition events. The following lemma establishes the analogous result for prefix transpositions. The proof is a bit lengthy and is omitted here, but appears in the full version of this paper.

Lemma 9. *Let π be an arbitrary permutation and $d(\pi) = k$ its prefix transposition distance. Then there exists an optimal sequence of prefix transpositions $\rho_1, \ldots, \rho_k$, such that $\rho_k \ldots \rho_1 \pi = \iota_n$, where $n = |\pi|$, and $\Delta b(\rho_{i-1} \ldots \rho_1 \pi, \rho_i) \leq 0$ for every $1 \leq i \leq k$.*

4 The Diameter of Prefix Transpositions

We call rearrangement diameter the largest rearrangement distance between two permutations of a certain size n. We Denote by $D(n)$ the diameter of prefix transpositions and by $D_\tau(n)$ the diameter of transpositions. Bafna and Pevzner [4] proved the following result.

Theorem 3. *The diameter of transpositions for permutations of size n is such that $\frac{n}{2} \leq D_\tau(n) \leq \frac{3n}{4}$.*

We can present a similar result for the prefix transposition distance problem.

Theorem 4. *The diameter of prefix transpositions for permutations of size n is such that $\frac{n}{2} \leq D(n) \leq n-1$.*

Proof: To begin with note that $D(n) \geq D_\tau(n)$, since $d(\pi) \geq d_\tau(\pi)$ for any permutation π (Lemma 1). We can then use the result of Aigner and West [1] that says that the diameter for the rearrangement distance problem that considers only insertion of the first element, that is, transpositions of the form $\rho(1,2,x)$, is $n-1$. □

The following result was proved independently by Christie [7] and Meidanis, Walter and Dias [16].

Theorem 5. *For $n \geq 3$, we have $d_\tau(R_n) = \lfloor \frac{n}{2} \rfloor + 1$.*

When dealing with prefix transpositions, we could state, based solely on Theorem 1, that $\lceil \frac{n}{2} \rceil \leq d(R_n) \leq n-1$. However, a stronger statement holds.

Theorem 6. *For $n \geq 4$, we have $d(R_n) \leq n - \lfloor \frac{n}{4} \rfloor$.*

Proof: The algorithm of Figure 2 sorts R_n using exactly $n - \lfloor \frac{n}{4} \rfloor$ prefix transpositions. A step-by-step execution of this algorithm on permutation R_{13} can be seen in Figure 3. □

Algorithm to Sort $R_n()$

Input: $\pi = R_n$, with $n \geq 4$
$m \leftarrow 4\lfloor \frac{n}{4} \rfloor$
{Phase 1: Shuffling}
for $i \leftarrow 1$ to $(\frac{m}{4}) - 1$
do $\pi \leftarrow \rho(1, 5, m - 2(i-1))\pi$
$\pi \leftarrow \rho(1, 3, \frac{m}{2} + 2)\pi$
{Phase 2: Greedy Phase}
$x \leftarrow \pi^{-1}[n] + 1$
$y \leftarrow m + 1$
for $i \leftarrow 1$ to $2(\frac{m}{4})$
do $z \leftarrow \pi[x]$
    $\pi \leftarrow \rho(1, x, y)\pi$
    $y \leftarrow \pi^{-1}[z-1] + 1$
    $w \leftarrow \pi[y] - 1$
    $x \leftarrow \pi^{-1}[w] + 1$
{Phase 3: Positioning the Last Elements}
for $i \leftarrow (m+1)$ to n
do $\pi \leftarrow \rho(1, i, i+1)\pi$
Output: $n - \lfloor \frac{n}{4} \rfloor$

Fig. 2. Algorithm to sort R_n

Lemma 10. *For $n \geq 1$, we have $d(R_{n+1}) \geq d(R_n)$.*

• 13 • 12 • 11 • 10 • 9 • 8 • 7 • 6 • 5 • 4 • 3 • 2 • 1 •

• 9 • 8 • 7 • 6 • 5 • 4 • 3 • 13 • 12 • 11 • 10 • 2 • 1 •

• 5 • 4 • 3 • 13 • 12 • 9 • 8 • 7 • 6 • 11 • 10 • 2 • 1 •

• 3 • 13 • 12 • 9 • 8 • 5 • 4 • 7 • 6 • 11 • 10 • 2 • 1 •

• 12 • 9 • 8 • 5 • 4 • 7 • 6 • 11 • 10 • 2 3 • 13 • 1 •

• 8 • 5 • 4 • 7 • 6 • 11 12 • 9 10 • 2 3 • 13 • 1 •

• 4 • 7 8 • 5 6 • 11 12 • 9 10 • 2 3 • 13 • 1 •

• 9 10 • 2 3 4 • 7 8 • 5 6 • 11 12 13 • 1 •

• 7 8 9 10 • 2 3 4 5 6 • 11 12 13 • 1 •

• 2 3 4 5 6 7 8 9 10 11 12 13 • 1 •

• 1 2 3 4 5 6 7 8 9 10 11 12 13

Fig. 3. Steps to sort R_{13}

Proof: It is easy to see that any series of prefix transpositions that sorts R_{n+1} will also sort R_n, provided we adapt the movements that include the element $n+1$. □

Christie [7] and Meidanis, Walter and Dias [16] have proposed the following conjecture, still open today.

Conjecture 1. The transposition diameter $D_\tau(n)$, for $n \geq 3$, is given by $D_\tau(n) = d_\tau(R_n) = \left\lfloor \frac{n}{2} \right\rfloor + 1$.

Likewise, we believe that the following statement is true.

Conjecture 2. The diameter of prefix transpositions $D(n)$, for $n \geq 4$, is given by $D(n) = d(R_n) = n - \left\lfloor \frac{n}{4} \right\rfloor$.

4.1 Tests

The tests that will be presented in this section were performed in a Digital Alpha Server GS140 computer, with 10 Alpha 21264 EV6 processors of 524MHz and 64-bit word length, with 8 GB of physical memory and running the OSF1 version 4.0 operating system. All programs were written in C++ and compiled with g++ using compilation directive "-O3". Our programs use just one processor and during the tests the machine was always executing other processes as well. The measured times are the times effectively spent by the programs (user + system time) and not the total time of execution (real time).

We implemented two "branch and bound" algorithms to compute the exact distance of prefix transpositions. The first version considers all possible prefix transpositions, while the second version considers only prefix transpositions that do not create new breakpoints, according to Lemma 9. Using these programs it was possible to obtain directly the prefix transposition distance for all reverse permutations R_n with $n \leq 15$. Table 1 and Figure 4 show result summaries.

To further support the correctness of Theorem 6, we implemented the algorithm that sorts reverse permutations R_n in polynomial time (Figure 2). We tested our implementation using all reverse permutations R_n for $n \leq 50000$. The algorithm correctly sorted all tested instances. Note that these instances are several times bigger than the biggest instances used in practice in genome rearrangement problems. Execution times for this algorithm are plotted in Figure 5.

Lastly we implemented two programs to verify the conjectures proposed in Section 4. The two programs are based in the same strategy. We built a graph as follows: we created a vertex for each of the $n!$ permutations with n elements and an edge for each pair of permutations that differ by a rearrangement event. In this graph we search for the permutations that posses the largest distance from the identity permutation. This strategy can be implemented in linear time on the graph size. With this method we could certify in slightly over 20 hours that both conjectures are true for permutations with $n \leq 11$ elements. Unfortunately 30 GB of physical memory are need to build the graph for $n = 12$, what made the test of our conjectures for $n \geq 12$ impossible.

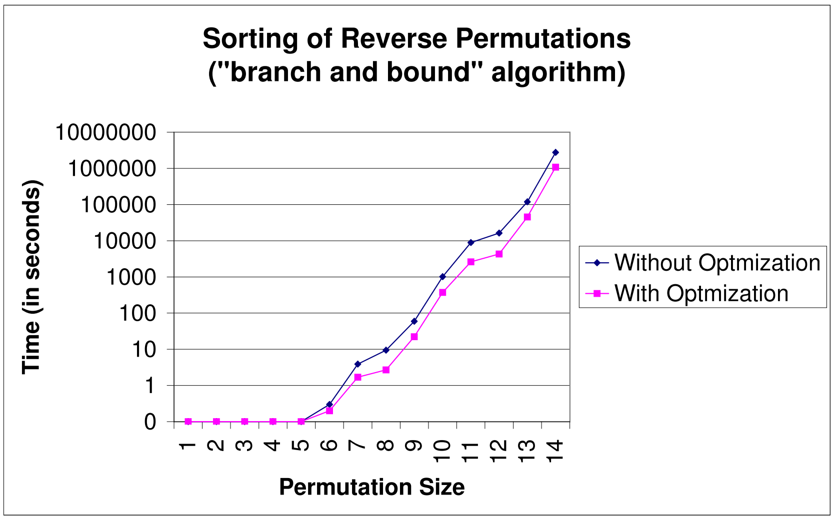

Fig. 4. Results for the "branch and bound" algorithm. Total approximate time of 47 days nonstop processing, with about 34 days for the version not optimized and 13 days for the optimized version

Table 1. distance of prefix transposition for reverse permutations with 16 or less elements. The times in column "without optimization" refer to the "branch and bound" algorithm that considers all prefix transpositions possible, while the column "with optimization" presents the results of the implementation that considers only prefix transpositions that do not create new breakpoints, according to Lemma 9. We could not compute $d(R_{16})$ directly using any of the two implementations; instead we present an estimate of the time necessary for each algorithm to compute correctly the distance. Note also that it is possible to infer the distance $d(R_{16})$ from Theorem 6 and Lemma 10

n	$d(R_n)$	Time without optimization (seconds)	Time with optimization (seconds)
02	01	0	0
03	02	0	0
04	03	0	0
05	04	0	0
06	05	0	0
07	06	0	0
08	06	4	2
09	07	9	3
10	08	59	22
11	09	1011	373
12	09	8872	2607
13	10	16294	4305
14	11	118463	45168
15	12	2771374	1081631
16	12*	750 days *	300 days *

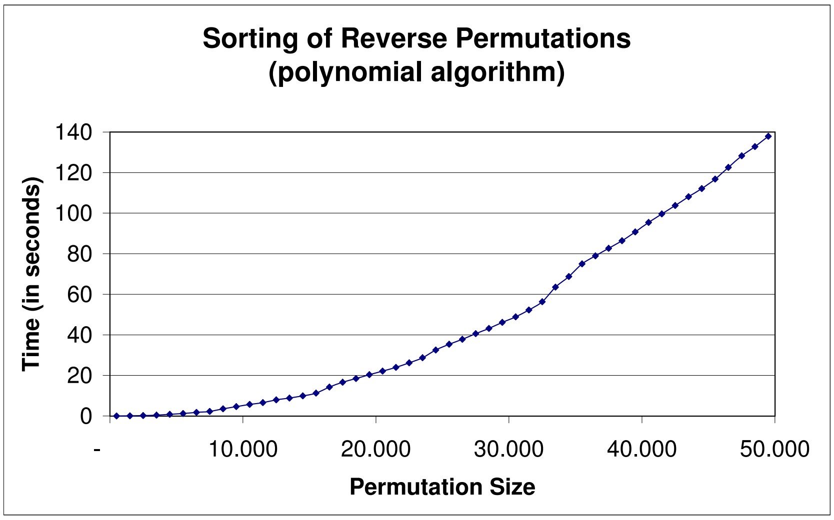

Fig. 5. Results for the polynomial algorithm. Total approximate time of 27 days non-stop processing

5 Permutations that Satisfy the Breakpoint Lower-Bound

Kececioglu and Sankoff [15] conjectured that to determine whether a permutation can be sorted using the minimum number of reversals indicated by the breakpoint lower bound for reversals was an NP-Hard problem, just like the general problem of sorting by reversals. Irving and Christie [13] and Tran [20] independently proved that this conjecture is false, exhibiting a polynomial algorithm for the problem.

In the case of prefix transpositions we know from Lemma 5 that for every permutation π we have $d(\pi) \geq \lceil (b(\pi) - 1)/2 \rceil$. However, given a permutation π, is it possible to determine whether $d(\pi) = (b(\pi) - 1)/2$? The following results prove that the answer is yes.

Lemma 11. *Let π be an arbitrary permutation. Then there exists at most one prefix transposition ρ such that $\Delta b(\pi, \rho) = -2$.*

Proof: Suppose that π and $\rho(1, x, y)$ are such that $\Delta b(\pi, \rho) = -2$. In this case we have $\pi = [\pi[1] \ldots \pi[x-1]\pi[x] \ldots \pi[y-1]\pi[y] \ldots]$ and $\rho\pi = [\pi[x] \ldots \pi[y-1]$ $\pi[1] \ldots \pi[x-1]\pi[y] \ldots]$, where $\pi[x-1] \neq \pi[x] - 1$, $\pi[y-1] \neq \pi[y] - 1$, $\pi[y-1] = \pi[1] - 1$ and $\pi[x-1] = \pi[y] - 1$. Finally, note that $\pi[1]$ determines uniquely the index y, and y determines uniquely the index x. □

Theorem 7. *Let π be an arbitrary permutation. Then it is possible to determine in polynomial time whether $d(\pi) = \frac{b(\pi)-1}{2}$.*

Proof: Immediate by the algorithm of Figure 6, that has complexity $O(n^2)$. □

Given an integer k, is it always possible to find a permutation π such that there is a series of k prefix transpositions $\rho_1, \ldots, \rho_k$ with $\Delta b(\rho_{i-1}\rho_{i-2} \ldots \rho_1\pi, \rho_i)$ $= -2$, for $1 \leq i \leq k$? Once again the answer is affirmative.

VERIFYING WHETHER π SATISFIES THE BREAKPOINTS LOWER-BOUND()

Input: π
$n \leftarrow |\pi|$
OK $\leftarrow$ TRUE
While $\pi \neq \iota_n$ and OK
do $y \leftarrow \pi^{-1}[\pi[1] - 1] + 1$
  $x \leftarrow \pi^{-1}[\pi[y] - 1] + 1$
  {Verifies whether there exists a movement that removes two breakpoints}
  if $x < y$
    then $\pi \leftarrow \rho(1, x, y)\pi$
    else OK $\leftarrow$ FALSE
Output: OK

Fig. 6. The algorithm that verifies whether π has distance $d(\pi) = \frac{b(\pi)-1}{2}$

Definition 9. *Let B_k be the family of permutations defined as follows: $B_k = [k+1\ k\ k+2\ k-1\ k+3\ k-2\ \ldots\ 2k-1\ 2\ 2k\ 1]$. Permutation B_k possesses $2k+1$ breakpoints.*

Lemma 12. *For every integer k it is possible to obtain a series of k prefix transpositions $\rho_1, \rho_2, \ldots, \rho_k$ that sort B_k such that $\Delta b(\rho_{i-1}\rho_{i-2}\ldots\rho_1\pi, \rho_i) = -2$, for $1 \leq i \leq k$.*

Proof: Immediate from the algorithm of Figure 7, that can be implemented in linear time. □

ALGORITHM TO SORT $B_k()$
Input: $\pi = B_k$, with $k \geq 1$
for $i \leftarrow 1$ to k
do $\pi \leftarrow \rho(1, 2i, 2i+1)\pi$
Output: k

Fig. 7. The algorithm that sorts B_k

6 Conclusions

We introduced in this work a new problem of Genome Rearrangement that we called distance of prefix transpositions. We showed a number of results for this problem, including two approximation algorithms (the best of them with factor 2), a proof that any permutation can be sorted without "cutting strips," a conjecture on the prefix transposition diameter stating that $D(n) = n - \lfloor \frac{n}{4} \rfloor$, and an algorithm for determining whether a permutation can be sorted using a series of prefix transpositions removing two breakpoints per step. The problem of sorting an arbitrary permutation with prefix transpositions remains open.

Acknowledments

We gratefully acknowledge the financial support of Brazilian agencies CNPq (National Council of Scientific and Technological Development) and FAPESP (The State of São Paulo Research Foundation).

References

1. M. Aigner and D. B. West. Sorting by insertion of leading element. *Journal of Combinatorial Theory*, 45:306–309, 1987.
2. D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

3. V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.
4. V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, May 1998.
5. A. Caprara. Sorting by reversals is difficult. In *Proceedings of the First International Conference on Computational Molecular Biology - (RECOMB'97)*, pages 75–83, New York, USA, January 1997. ACM Press.
6. D. A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 244–252, San Francisco, USA, January 1998.
7. D. A. Christie. *Genome Rearrangement Problems.* PhD thesis, Glasgow University, 1998.
8. H. Dweighter. *American Mathematical Monthly*, volume 82, page 1010. The Mathematical Association of America, 1975.
9. W. H. Gates and C. H. Papadimitriou. Bounds for sorting by prefix reversals. *Discrete Mathematics*, 27:47–57, 1979.
10. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, January 1999.
11. M. H. Heydari and I. H. Sudborough. Sorting by prefix reversals is np-complete. To be submitted.
12. M. H. Heydari and I. H. Sudborough. On the diameter of the pancake network. *Journal of Algorithms*, 25:67–94, 1997.
13. R. W. Irving and D. A. Christie. Sorting by reversals: on a conjecture of kececioglu and sankoff. Technical Report TR-95-12, Department of Computing Science, University of Glasgow, May 1995.
14. H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29(3):880–892, January 2000.
15. J. D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, January 1995.
16. J. Meidanis, M. E. Walter, and Z. Dias. Transposition distance between a permutation and its reverse. In R. Baeza-Yates, editor, *Proceedings of the 4th South American Workshop on String Processing (WSP'97)*, pages 70–79, Valparaiso, Chile, 1997. Carleton University Press.
17. J. Meidanis, M. E. M. T. Walter, and Z. Dias. Reversal distance of signed circular chromosomes. Technical Report IC-00-23, Institute of Computing - University of Campinas, December 2000.
18. J. D. Palmer and L. A. Herbon. Plant mitochondrial dna evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 27:87–97, 1988.
19. J. C. Setubal and J. Meidanis. *Introduction to Computional Molecular Biology.* PWS Publishing Company, 1997.
20. N. Q. Tran. An easy case of sorting by reversals. In A. Apostolico and J. Hein, editors, *Proceedings of the 8th Annual Symposium of the Combinatorial Pattern Matching (CPM'97)*, volume 1264 of *Lecture Notes in Computer Science*, pages 83–89, Aarhus, Denmark, June 1997. Springer.