Listing All Sorting Reversals in Quadratic Time

Krister M. Swenson[1,2], Ghada Badr[3], and David Sankoff[1]

[1] Department of Mathematics and Statistics, University of Ottawa, Ontario, K1N 6N5, Canada
[2] LaCIM, UQAM, Montréal Québec, H3C 3P8, Canada
[3] SITE, School of Information Technology and Engineering, University of Ottawa, Ontario, K1N 6N5, Canada

Abstract. We describe an average-case $O(n^2)$ algorithm to list all reversals on a signed permutation π that, when applied to π, produce a permutation that is closer to the identity. This algorithm is optimal in the sense that, the time it takes to write the list is $\Omega(n^2)$ in the worst case.

1 Introduction

In 1995 Hannenhalli and Pevzner [9] presented an algorithm to transform one genome into another in a minimum number of biologically plausible moves. They modeled a genome as a signed permutation and the move that they considered was the reversal: the order of a substring of the permutation is reversed, and the sign of each element in the substring is flipped. Since then many refinements and speed improvements have been developed [4,8,10,11,16,18,19].

In 2002 Siepel and Ajana et. al. [1,15] showed how to list every parsimonious scenario of reversals, each scenario being a proposed candidate for the true evolutionary history. Fundamental to their algorithms are $O(n^3)$ techniques for finding all *sorting* reversals; the reversals that at each step produce a permutation that is closer to the target permutation than the last. Ajana et. al. [1] used these results to support the replication-directed reversal hypothesis. Lefebvre et. al. [12] and Sankoff et. al. [14] used similar methodology to gain insight into the distribution of reversal lengths between genomes. Algorithms that attempt to more succinctly represent all shortest-length scenarios [3,6] have also been developed.

In this paper we show how to list all sorting reversals in $O(n^2)$ time on average. This algorithm is optimal in the sense that there are $\Omega(n^2)$ safe cycle-splitting reversals in the worst-case. This affords a significant speedup of the aforementioned methods [1,3,6,12,14,15], since listing all sorting reversals is the kernel of repeated computation in each of them, especially when applied to permutations of sizes 3×10^3 to 3×10^5 (the size of bacterial or mammalian genomes).

2 Background

Take a signed permutation $\pi = \pi_1, \ldots, \pi_n$ on the integers from 1 to n. Define a (signed) *reversal* $\rho(i, j)$ as the signed permutation

$$1, 2, \ldots, (i-1), {}^{-}j, \ldots, {}^{-}i, (j+1), \ldots, n.$$

V. Moulton and M. Singh (Eds.): WABI 2010, LNBI 6293, pp. 102–110, 2010.

That way, applying the reversal $\rho(i,j)$ to permutation π gives

$$\rho(i,j)(\pi) = \pi \circ \rho(i,j) = \pi_1, \ldots, \pi_{i-1}, {}^{-}\pi_j, \ldots, {}^{-}\pi_i, \pi_{j+1}, \ldots, \pi_n.$$

Given signed permutations π_1 and π_2, the *reversal distance* $d(\pi_1, \pi_2)$ is the smallest k such that $\pi_2 = \pi_1 \circ \rho_1 \circ \rho_2 \circ \cdots \circ \rho_k$. Without loss of generality[1] we consider $\pi_2 = I = 1, 2, \ldots, n$ to be the identity permutation. In this paper, we describe our methods using circular permutations (where the leftmost element follows the rightmost element), as any sorting reversal on a circular permutation has its counterpart on a linear version of the permutation.

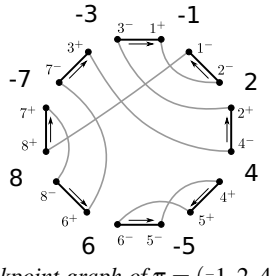
2.1 All Sorting Reversals

A reversal ρ is a *sorting* reversal on π if $d(\pi \circ \rho) = d(\pi) - 1$. Although the definition is simple, a characterization of all sorting reversals requires effort; to do so we must introduce the *breakpoint graph* [9]. Each element π_i of permutation π has two vertices associated with it denoted by π_i^- and π_i^+ ($\pi^\pm$ can denote either). Embed the graph on a circle as follows: place all $2n$ vertices on the circle so that:

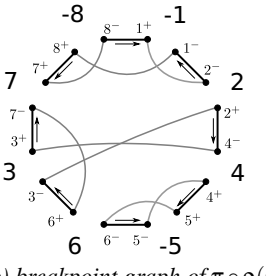
1. π_i^+ and π_i^- are adjacent on the circle,
2. π_i^- is before (in the clockwise direction) π_i^+ if and only if π_i is positive, and
3. a $\pi_i^\pm$ is adjacent to a $\pi_{i+1}^\pm$ if and only if π_i and π_{i+1} are adjacent in π.

For two vertices $v_1 = \pi_i^\pm$ and $v_2 = \pi_j^\pm$ ($i \neq j$) that are adjacent on the circle, add the edge (v_1, v_2)—a *reality* edge (also called a black edge); also add edges (π_i^+, π_{i+1}^-) for all i and (π_n^+, π_1^-)—the *desire* edges (also called gray edges). Figure 1(a) shows the breakpoint graph for $\pi = ({}^{-}1\ 2\ 4{}^{-}5\ 6\ 8{}^{-}7{}^{-}3)$. Note that every vertex has indegree 2 and outdegree 2, so the graph has a unique decomposition into cycles of even length (alternating between reality and desire edges).

A reversal $\rho(i,j)$ is said to *act on* the reality edges $(\pi_{i-1}^\pm, \pi_i^\pm)$ and $(\pi_j^\pm, \pi_{j+1}^\pm)$ because these are the only edges in the breakpoint graph of π that are not in the graph of



(a) breakpoint graph of $\pi = ({}^{-}1\ 2\ 4{}^{-}5\ 6\ 8{}^{-}7{}^{-}3)$



(b) breakpoint graph of $\pi \circ \rho(6,8)$

Fig. 1. Two breakpoints graphs. The direction that reality edges are traversed on a tour of the cycles is labeled with arrows. $\rho(6,8)$ is an unsafe reversal on π.

[1] Since $I = \pi_2^{-1} \circ \pi_1 \circ \rho_1 \circ \rho_2 \circ \cdots \circ \rho_k$.

$\pi \circ \rho(i,j)$. In Figure 1, the reversal $\rho(6,8)$ acts on reality edges $(3^-,1^+)$ and $(6^+,8^-)$. Two reality edges on the same cycle are *convergent* if a traversal of their cycle visits each edge in the same direction in the circular embedding; otherwise they are *divergent*. The following definitions classify the action of a reversal on the cycles of the breakpoint graph [9].

Definition 1 (cycle-splitting reversal). *A reversal that acts on a pair of divergent reality edges splits the cycle to which the edges belong, so are called* cycle-splitting *reversals.*

Conversely, no reversal that acts on a pair of convergent reality edges splits their common cycle. A reversal that acts upon a pair of reality edges in two different cycles merges the two cycles. The permutation of Figure 1(a) has 10 cycle-splitting inversions including $\rho(1,2)$, $\rho(4,4)$, and $\rho(6,8)$. Notice that at most one cycle can be created by a reversal, yielding the inequality

$$d(\pi) \geq n - c(\pi), \tag{1}$$

where $c(\pi)$ is the number of cycles in the breakpoint graph. Most cycle-splitting reversals are sorting reversals [17], but not all sorting reversals are cycle-splitting reversals, which indicates a gap between this lower bound and the reversal distance.

We must further explore structure in the permutation that allows us to predict the reversal distance when the lower bound is not realized.

Definition 2 (FCI [5]). *A* framed common interval *(FCI) of a permutation (made circular by considering the first and last elements as being adjacent) is a substring of the permutation,* $as_1s_2\ldots s_kb$ *or* $^-bs_1s_2\ldots s_k{}^-a$ *such that*

- *for each i,* $1 \leq i \leq k$, $|a| < |s_i| < |b|$, *and*
- *for each l,* $|a| < l < |b|$, *there exists a j with* $|s_j| = l$, *and*
- *it is not a concatenation of substrings satisfying the previous two properties.*

So the substring $s_1s_2\ldots s_k$ is a (possibly empty) signed permutation of the integers that are greater than a and less than b; a and b are the *frame* elements, while those of $s_1\ldots s_k$ are *trunk* elements if they are not trunk elements of a smaller FCI. The framed interval is said to be common, in that it also exists as an interval $\big(a(a+1)(a+2)\ldots b\big)$ in the identity permutation.

A *component* of a permutation is comprised of the trunk elements of an FCI that are not trunk elements of a shorter FCI, plus the frame elements. The permutation of Figure 1(b) has three components: one framed by elements 2 and 7, another framed by 4 and 6. The third is an interval in the circular sense, framed by elements 7 and 2 with the trunk comprised of elements 8 and 1; in the circular sense we have $7 < 8 < 1 < 2$ here.

Definition 3 (bad component[5]). *A* bad component *of a permutation is a component with at least 4 elements, where the sign of every element is the same.*

In Figure 1(b), the component $(2\ 4\ 6\ 3\ 7)$ is bad. The existence of one or more bad components in a permutation indicate exactly those situations where the lower bound

cannot be met [9]. Siepel's paper [15] describes in detail an $O(n^3)$ algorithm for finding the set of sorting reversals when bad components exist. While further exploration of Siepel's characterization of sorting reversals in the presence of bad components could eventually lead to a worst-case $O(n^2)$ algorithm, we do not address the issue here. Suffice it to say that the average-case complexity is $O(n^2)$ even when the trivial $O(n^3)$ algorithm[2] is used on permutations with bad components; the probability that a permutation chosen uniformly at random has a bad component is $O(n^{-2})$ [7,17] and we can detect the presence of bad components in linear time [2,5].

We focus on the bottleneck of sorting FCIs that do not correspond to bad components: cycle-splitting reversals that create bad components (cycle-splitting reversals that are not sorting reversals).

Definition 4 (bad reversal). *A* bad *reversal is a reversal that creates a bad component.*

Definition 5 (unsafe reversal[9]). *An* unsafe *reversal is a cycle-splitting reversal that is bad.*

In Figure 1(a), the reversal $\rho(6,8)$ is unsafe.

2.2 Outline

Known algorithms that list all sorting reversals check, one by one, if each of the potentially $\Omega(n^2)$ cycle-splitting reversals is unsafe by applying the reversal and then running a linear time check as to whether it produced a bad (unoriented) component [1,15]. Instead of listing all cycle-splitting reversals and then checking them, we do the inverse: we predict which reversals may be *unsafe* (whether cycle-splitting or otherwise) and avoid listing them. We first characterize what we call *ominous* substrings of the permutation, those substrings that could be turned into a bad component with one reversal. Our algorithm searches for ominous substrings by doing the following: for each element of the permutation we posit that it is a smallest element of a potential (after a reversal) bad component and continue by scanning the permutation to detect an ominous substring.

In Section 3 we introduce ominous substrings while Section 4 describes how to detect the set of all ominous substrings efficiently. Section 5 presents the algorithm. Finally Section 6 discusses open problems.

3 Ominous Substrings

Take any unsafe cycle-splitting reversal ρ on permutation π. Since it is unsafe, the permutation $\pi \circ \rho$ has at least one bad component created by ρ. In this section we will show that there exists in π a particular pattern — an *ominous* substring of π — indicating that ρ is unsafe. We first describe ominous substring of permutations with a single component.

[2] Each of the $O(n^2)$ reversals can in turn be applied to the permutation and the linear time algorithm for reversal distance [2] can be used to check if it was a sorting reversal.

3.1 Permutation with a Single Component

A substring of a permutation is *ominous* if and only if there exists some elements e and f such that the substring fits one of the following templates (or their reverse):

1. $(eA\underline{X^{-}f^{-}B})$: where $A, {}^{-}B$, and X are substrings of the permutation. A has only positive while ${}^{-}B$ has only negative elements.
2. $(\underline{{}^{-}A^{-}eX}Bf)$: where ${}^{-}A, B$, and X are substrings of the permutation. ${}^{-}A$ has only negative while B has only positive elements. e is negative.
3. $(eA\underline{{}^{-}BC}f)$: where $A, {}^{-}B$, and C are substrings of the permutation. A and C have only positive while ${}^{-}B$ has only negative elements.

and A and B (and C if it exists) are comprised of exactly those elements with absolute value i for $e < i < f$.

In template 3, there already exists an FCI with frame elements e and f; the reversal that acts on exactly the elements of B fixes the elements of the interval to have the same sign. In the other two templates, a new interval is created with e and f as the frame elements. For template 1, $\{f\} \cup B \cup X$ are the elements reversed while for template 2, $\{e\} \cup A \cup X$ are the elements reversed. For example, $({}^{-}7\ 1\ \underline{{}^{-}3\ {}^{-}4\ {}^{-}5\ {}^{-}2}\ 6)$ matches template 3 with the unsafe reversal acting upon the elements $\{2,3,4,5\}$; A and C are empty in this case. $({}^{-}1\ 2\ 4\ \underline{6\ {}^{-}5\ {}^{-}3})$ matches template 1 with the unsafe reversal acting upon the elements $\{3,5,6\}$; $f = {}^{-}5$, $B = \{{}^{-}3\}$, and $X = \{6\}$ in this case. $({}^{-}2\underline{{}^{-}6\ {}^{-}8\ {}^{-}4\ 1}\ 5\ 7\ 9\ {}^{-}3)$ matches template 2 with the unsafe reversal acting upon the elements $\{1,4,6,8\}$; $A = \{6,8\}$, $B = \{5,7\}$, and $X = \{1\}$ in this case.

We are ready to state the main lemma of this section.

Lemma 1. *There is a one to one correspondence between bad reversals and ominous substrings.*

Proof. By definition, there exists at least one reversal that creates a bad component from an ominous substring. On the other hand, take a permutation $\pi \circ \rho$ that has a bad component — with frame elements e and f — created by the reversal ρ. Say that the elements of the bad component are positive, then e is on the left and f is on the right. If ρ includes both e and f, this implies that the bad component already exists in π, which is a contradiction. Now let us examine the other three possibilities. If ρ does not include e and f, then the ominous substring in π corresponds to template 3. If ρ includes only f, then the ominous substring in π corresponds to template 1. If ρ includes only e, then the ominous substring in π corresponds to the template 2. If the elements of the bad component are negative then the negative analogue holds for each case. Since each ominous substring implies exactly one reversal dictated by the A, B, C, and X, we have the bijection.

3.2 Permutations with Multiple Components

We described ominous substrings on permutations with a single component. Since sorting reversals act only upon adjacencies in a single component [9], we adapt the techniques for single components to the case of multiple components in the following manner.

Consider a component of a permutation with some frame elements of a smaller FCI contained in it. We obtain the *condensed* version of the component by doing the following: for each pair of frame elements a and b (or ^-a and ^-b) of a smaller FCI contained in it, we replace the pair by a (resp. ^-a) and change the magnitude m of every element $m > b$ in the component to be $m-(b-a)$. The templates can be applied directly to the condensed component. For example, take the component $C = (2\ 4\ 6\ 3\ 7)$ in Figure 1 where the component $(4^-5\ 6)$ is contained in it. The condensed version of C is $(2\ 4\ 3\ 5)$. The condensed version of any component can be computed in linear time.

4 Detecting Ominous Substrings

We now turn to the task of detecting an ominous substring associated with a smallest element e. The following methods can be adapted to detect the negative analogue of each template, so we only describe the detection of the templates as they were presented in Section 3.1. The general outline used in each of the following algorithms is the same: we visit the permutation starting with element e, proceeding to element $e+1$, then $e+2$ and so on. At each step we maintain enough information to check whether certain conditions hold that indicate we have found an ominous substring.

Call the set of elements that we visit through the first i steps S_i (those with absolute value in the interval $[e, e+i]$). To check for each template at step i, so that f would be the element $e+i$, we maintain the following values.

- Rightmost positive index visited: $rp = \max(\{|\pi^{-1}(|j|)| \mid j \in S_i, j > 0\})$
- Leftmost positive index visited: $lp = \min(\{|\pi^{-1}(|j|)| \mid j \in S_i, j > 0\})$
- Rightmost negative index visited: $rn = \max(\{|\pi^{-1}(|j|)| \mid j \in S_i, j < 0\})$
- Leftmost negative index visited: $ln = \min(\{|\pi^{-1}(|j|)| \mid j \in S_i, j < 0\})$

Template 1 (eAX^-f^-B) exists, with unsafe reversal $\rho(rp+1, rn)$, if and only if the following conditions hold:

1. $lp = \pi^{-1}(|e|)$
   (e is the leftmost element visited)
2. $ln > rp$
   (the negative elements are to the right of the positive)
3. $rn - ln + rp - lp = i - 1$
   (the positive and negative elements are all contiguous)
4. $\pi^{-1}(|e+i|) = ln$
   (the last element visited is the leftmost negative element)
5. $i \geq 3$
   (the FCI has at least 4 elements)

Template 2 ($^-A^-eXBf$) exist, with unsafe reversal $\rho(ln, lp-1)$, if and only if the following conditions hold:

1. $rn = \pi^{-1}(|e|)$
   (e is the rightmost negative element)
2. $lp > rn$
   (the negative elements are to the left of the positive)

3. $rn - ln + rp - lp = i - 1$
   (the positive and negative elements are all contiguous)
4. $\pi^{-1}(|e+i|) = rp$
   (the last element visited is the rightmost element visited)
5. $i \geq 3$
   (the FCI has at least 4 elements)

To check for template 3 we maintain another value $neg = |\{j \mid j \in S_i, j < 0\}|$, the number of negative values visited. We know that we have found template 3 ($eA^{-}BCf$) with unsafe reversal $\rho(ln, rn)$ if and only if all of the following conditions hold:

1. $lp = \pi^{-1}(|e|)$
   (e is the leftmost element visited)
2. $ln > lp$
   (the negative elements are to the right of some positive)
3. $rp > rn$
   (the negative elements are to the left of some positive)
4. $rp - lp = i$
   (we have visited a contiguous substring)
5. $rn - ln = neg - 1$
   (the negative elements of B are contiguous)
6. $\pi^{-1}(|e+i|) = rp$
   (the last element visited is the rightmost element visited)
7. $i \geq 3$
   (the FCI has at least 4 elements)

Note that if at some iteration i during our scan conditions 1 or 2 for any of the templates are broken, we know that e can no longer match that template.

5 The Algorithm

We begin by proving the following theorem.

Theorem 1. *For a permutation without a bad component, there is an $O(n^2)$ algorithm for listing all sorting reversals.*

Proof. Use the methods of Section 4 to obtain a blacklist of all ominous substrings associated with each possible smallest frame element e. Since the list of all ominous substrings associated with a single smallest frame element is obtained by a linear scan for all possible right endpoints f, the time to build the blacklist is $O(n^2)$. Each element of the list is associated with a bad reversal, the indices of which we mark in an n by n matrix; an entry r at row i and column j indicates that the bad reversal r acts on elements from position i to position j in the permutation. Obtain the list of all cycle-splitting reversals in $O(n^2)$ time using the standard methods [9]. Finally, examine this list one reversal at a time, removing from the list any reversal that has a corresponding entry marked in the matrix.

The methods described so far are applicable to permutations with no bad components. Permutations with bad components can be easily handled by combining our algorithm with that of Siepel [15] in the following way. First make a linear scan of the permutation to detect bad components [2,5]. If there are bad components, use the $O(n^3)$ algorithm of Siepel, otherwise, use our algorithm.

Theorem 2. *Pick a signed permutation uniformly at random, the expected time the above algorithm takes to list all sorting reversals is* $O(n^2)$.

Proof. The probability of seeing a bad component in a permutation taken uniformly at random from the set of all signed permutations is $O(n^{-2})$ [17]. The bound follows since $n^3 \times n^{-2} < n^2$.

6 Conclusions

We presented the first quadratic time algorithm for listing all sorting reversals for a signed permutation. This pattern matching algorithm is simple in that it requires no special data structures. It is optimal in the sense that most permutations can have $\Omega(n^2)$ sorting reversals [20,13]. An improvement on our bound would be an algorithm that runs in $O(n+k)$ time where k is the number of sorting reversals. It is currently unclear how to modify our algorithm to obtain this bound.

References

1. Ajana, Y., Lefebvre, J.-F., Tillier, E.R.M., El-Mabrouk, N.: Exploring the set of all minimal sequences of reversals - an application to test the replication-directed reversal hypothesis. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 300–315. Springer, Heidelberg (2002)
2. Bader, D.A., Moret, B.M.E., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J. Comput. Biol. 8(5), 483–491 (2001); A preliminary version appeared in WADS 2001, pp. 365–376
3. Baudet, C., Dias, Z.: An Improved Algorithm to Enumerate All Traces that Sort a Signed Permutation by Reversals. In: SIGAPP 2010: Proceedings of the Twenty Fifth Symposium on Applied Computing (2010)
4. Bergeron, A.: A very elementary presentation of the Hannenhalli–Pevzner theory. Discrete Applied Mathematics 146(2), 134–145 (2005)
5. Bergeron, A., Heber, S., Stoye, J.: Common intervals and sorting by reversals: a marriage of necessity. In: Proc. 2nd European Conf. Comput. Biol. ECCB 2002, pp. 54–63 (2002)
6. Braga, M.D.V., Sagot, M., Scornavacca, C., Tannier, E.: The Solution Space of Sorting by Reversals. In: Măndoiu, I.I., Zelikovsky, A. (eds.) ISBRA 2007. LNCS (LNBI), vol. 4463, pp. 293–304. Springer, Heidelberg (2007)
7. Caprara, A.: On the tightness of the alternating-cycle lower bound for sorting by reversals. J. Combin. Optimization 3, 149–182 (1999)
8. Hannenhalli, S., Pevzner, P.A.: Transforming mice into men (polynomial algorithm for genomic distance problems). In: Proc. 36th Ann. IEEE Symp. Foundations of Comput. Sci. (FOCS 1995), pp. 581–592. IEEE Press, Piscataway (1995)
9. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. J. ACM 46(1), 1–27 (1999)

10. Kaplan, H., Shamir, R., Tarjan, R.E.: Faster and simpler algorithm for sorting signed permutations by reversals. SIAM J. Computing 29(3), 880–892 (1999)
11. Kaplan, H., Verbin, E.: Efficient data structures and a new randomized approach for sorting signed permutations by reversals. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 170–185. Springer, Heidelberg (2003)
12. Lefebvre, J.-F., El-Mabrouk, N., Tillier, E.R.M., Sankoff, D.: Detection and validation of single gene inversions. In: Proc. 11th Int'l. Conf. on Intelligent Systems for Mol. Biol. (ISMB 2003). Bioinformatics, vol. 19, pp. i190–i196. Oxford U. Press (2003)
13. Sankoff, D., Haque, L.: The distribution of genomic distance between random genomes. Journal of Computational Biology 13(5), 1005–1012 (2006)
14. Sankoff, D., Lefebvre, J.-F., Tillier, E.R.M., Maler, A., El-Mabrouk, N.: The distribution of inversion lengths in bacteria. In: Lagergren, J. (ed.) RECOMB-WS 2004. LNCS (LNBI), vol. 3388, pp. 97–108. Springer, Heidelberg (2005)
15. Siepel, A.C.: An algorithm to find all sorting reversals. In: Proc. 6th Ann. Int'l. Conf. Comput. Mol. Biol. (RECOMB 2002). ACM Press, New York (2002)
16. Swenson, K.M., Rajan, V., Lin, Y., Moret, B.M.E.: Sorting signed permutations by inversions in $O(n \log n)$ time. In: Batzoglou, S. (ed.) RECOMB 2009. LNCS, vol. 5541, pp. 386–399. Springer, Heidelberg (2009)
17. Swenson, K.M., Lin, Y., Rajan, V., Moret, B.M.E.: Hurdles hardly have to be heeded. In: Nelson, C.E., Vialette, S. (eds.) RECOMB-CG 2008. LNCS (LNBI), vol. 5267, pp. 239–249. Springer, Heidelberg (2008)
18. Tannier, E., Bergeron, A., Sagot, M.-F.: Advances on sorting by reversals. Disc. Appl. Math. 155(6-7), 881–888 (2007)
19. Tannier, E., Sagot, M.: Sorting by reversals in subquadratic time. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 1–13. Springer, Heidelberg (2004)
20. Yang, Y., Szkely, L.A.: On the expectation and variance of reversal distance. Acta Univ. Sapientiae, Mathematica 1(1), 5–20 (2009)