Journal of Universal Computer Science, vol. 25, no. 2 (2019), 98-121
submitted: 15/10/18, accepted: 27/2/19, appeared: 28/2/19 © J.UCS

Sorting Permutations by λ-Operations

Guilherme Henrique Santos Miranda
(Institute of Computing
University of Campinas, Campinas, Brazil
guilherme.miranda@students.ic.unicamp.br)

Carla Negri Lintzmayer
(Center for Mathematics, Computation and Cognition
Federal University of ABC, Santo André, Brazil
carla.negri@ufabc.edu.br)

Zanoni Dias
(Institute of Computing
University of Campinas, Campinas, Brazil
zanoni@ic.unicamp.br)

Abstract: For estimating the evolutionary distance between genomes of two different organisms, many sorting permutation problems have emerged. A well accepted way to do this is considering the smallest sequence of rearrangement events – mutations which affect large portions of the genomes – that transform one genome into the other. In these problems, both genomes are represented as permutations of integer numbers, but one of them can be represented as the identity permutation, so that the problem is reduced to sort a permutation. Moreover, rearrangement models define which type of operations (rearrangement events) can be applied over a permutation in order to modify it. Reversals, which are operations that revert a genome segment, and transpositions, which are operations that swap two adjacent genome segments, are two of the most studied types of rearrangements in the literature. In this paper, we consider rearrangement models that allow reversals, transpositions, and both operations together. Since there exist evidences that large mutations rarely occur, we add a restriction with biological relevance: any operation can affect at most λ elements of a permutation. For this variation of sorting permutation problem, we present approximation algorithms with approximation factors based on the size of the permutation and/or on λ, for both signed and unsigned permutations (which represent known and unknown gene orientations, respectively).

Key Words: Sorting permutations, Genome rearrangements, Reversals, Transpositions, Computational Biology.
Category: F.2.0, G.2.3.

1 Introduction

One way for estimating the evolutionary distance between two genomes is using the length of the shortest sequence of genome rearrangements that transform a genome into another, called *rearrangement distance*. This is the most likely to occur, based the Principle of Parsimony, since a genome rearrangement is an event that occurs with relative rarity, because it changes large stretches of a genome.

We can represent a genome computationally by a permutation of integers if we assume that there are no duplicate blocks (which can be a gene or a sequence of genes) and that its composition is a single linear chromosome. If the orientation of genes are known, the permutation can be *signed* to represent that. Otherwise, we use unsigned permutations. With such representation, the problem of estimating the evolutionary distance with the rearrangement distance is modeled as the problem of calculating the minimum number of rearrangements that transform a permutation into another. As it turns out, one of the permutations can be written as the identity permutation so that the previous problem can be reduced to the problem of finding the minimum number of operations that sort a permutation.

Two of the most studied genome rearrangements considered in the literature are *reversals*, which invert a determined segment of the genome/permutation, and *transpositions*, which exchange two adjacent segments in the genome/permutation. There are several results regarding sorting permutations using such rearrangements. The problem of Sorting Unsigned Permutations by Reversals is NP-Hard [Caprara, 1999] and the best-known result is a 1.375-approximation algorithm [Berman et al., 2002]. On the other hand, Sorting Signed Permutations by Reversals can be solved in polynomial time [Hannenhalli and Pevzner, 1999] and the best-known result is an algorithm with subquadratic running time [Tannier et al., 2007]. Also, a linear time algorithm was presented [Bader et al., 2001] for the case when just the rearrangement distance is desired. The problem of Sorting (Unsigned) Permutations by Transpositions is also NP-Hard [Bulteau et al., 2012] and the best-known result is also a 1.375-approximation algorithm [Elias and Hartman, 2006]. The complexity of the problems of Sorting Unsigned and Signed Permutations by Reversals and Transpositions is unknown. For them, there exist 3-approximation and 2-approximation algorithms, respectively [Walter et al., 1998]. Moreover, a 2α-approximation algorithm for unsigned permutations was presented [Rahman et al., 2008], where α is the approximation factor of a cycle decomposition algorithm for breakpoint graphs [Lin and Jiang, 2004]. Given the best-known value of α [Chen, 2013], the approximation factor of such algorithm is $2.8334 + \epsilon$, where $\epsilon > 0$.

The above mentioned results involve a traditional approach, but there are some studies considering variants that restrict the application of the rearrangements. For instance, one can limit in which parts of the genome the rearrangements will be applied [Dias and Meidanis, 2002, Lintzmayer et al., 2017] or limit the amount of elements of the permutation that can be affected by the rearrangements [Jerrum, 1985, Galvão et al., 2015, Heath and Vergara, 2003,

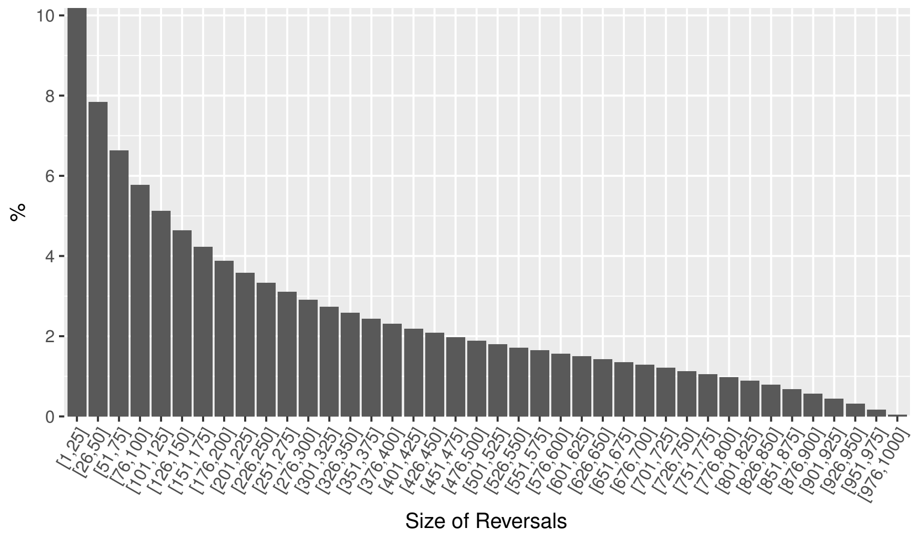


Figure 1: Frequency of size of reversals given by GRIMM for Sorting Signed Permutations by Reversals, considering 10000 arbitrary permutations of size 1000.

Jiang et al., 2014, Vergara, 1998], which is the rearrangement's size.

A motivation for considering a limit on the size of the rearrangements comes from the observation that rearrangement events are mutations that modify large stretches of the genome and, as such, they rarely occur, prevailing rearrangements that involve few blocks [Lefebvre et al., 2003]. We performed an experiment to analyze the frequency that each reversal size appears in the optimal solution considering sorting signed reversals. We used the software GRIMM (Genome Rearrangements In Man and Mouse) [Tesler, 2002] and considered 10000 arbitrary permutations of size 1000. Figure 1 shows GRIMM's default solution while Figure 2 shows the solution when we ask it to prioritize smaller reversals.

If the size limit of an operation is at most 2, Sorting Unsigned Permutations by Reversals (or by Transpositions) can be solved in polynomial time [Jerrum, 1985]. If it is at most 3, the best results in the literature are (i) a 2-approximation [Heath and Vergara, 2003] and a 5-approximation [Galvão et al., 2015] algorithm for Sorting Unsigned and Signed Permutations by Reversals, respectively, (ii) a $\frac{5}{4}$-approximation algorithm for Sorting Permutations by Transpositions [Jiang et al., 2014], and (iii) a 2-approximation [Vergara, 1998] and a 3-approximation [Galvão et al., 2015] algorithm for Sorting Unsigned and Signed Permutations by Reversals and Trans-

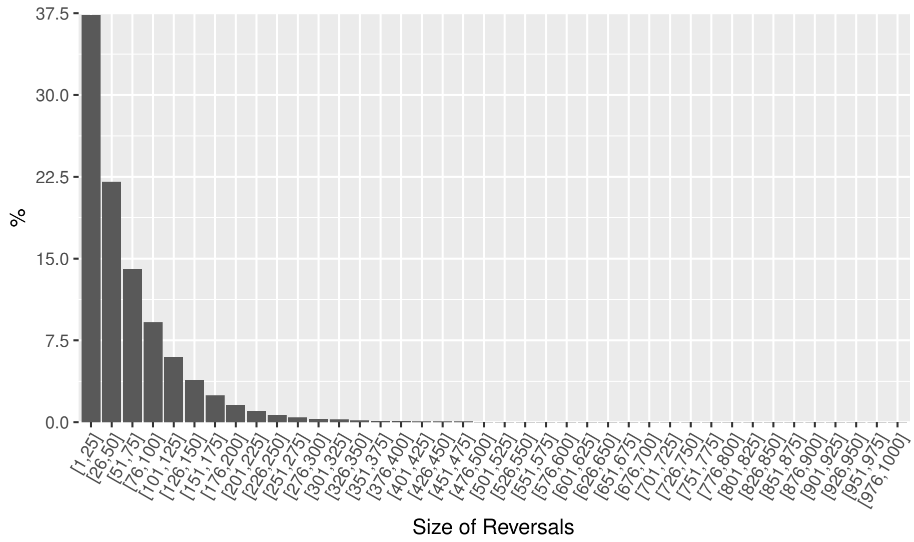


Figure 2: Frequency of size of reversals given by GRIMM for Sorting Signed Permutations by Reversal when it prioritizes smaller operations, considering 10000 arbitrary permutations of size 1000.

positions, respectively. If the size limit of any operation is at most a value λ, we call it a λ-operation. Also, a λ-permutation is a permutation such that any element is less than λ positions away from its correct position (in the identity permutation). For the problem of Sorting Unsigned λ-Permutations by λ-Operations there are approximation algorithms with factors $O(\lambda^2)$, $O(\lambda)$, and $O(1)$ [Miranda et al., 2018a].

In a previous work [Miranda et al., 2018b], we presented approximation algorithms for Sorting Unsigned Permutations by λ-Reversals, by λ-Transpositions, and by both of them. In this paper, we extend such work giving new results for such problems and we also consider Sorting Signed Permutations by λ-Operations.

Next sections are organized as follows. Section 2 presents the definitions and notations regarding problems of sorting permutations. Section 3 presents the proposed approximation algorithms. Section 4 presents some practical experiments performed over the proposed algorithms. Section 5 presents the concluding remarks.

2 Definitions

We represent a *signed permutation* of size n by $\pi = (\pi_1\ \pi_2\ \dots\ \pi_n)$, where $\pi_i \in \{-n, -(n-1), \dots, -2, -1, +1, +2, \dots, +(n-1), +n\}$ and $|\pi_i| \neq |\pi_j|$ if, and only if, $i \neq j$. An *unsigned permutation* is represented similarly, but there are no signs in the elements. The *identity permutation* of size n, which is the goal of the problems, is defined as $\iota = (1\ 2\ \dots\ n)$.

Given two permutations π and σ of size n, the composition operation "$\cdot$" generates a new permutation defined as $\alpha = \pi \cdot \sigma = (\alpha_1\ \alpha_2\ \dots\ \alpha_n)$ where $\alpha_i = -\pi_{|\sigma_i|}$ if $\sigma_i < 0$ or $\alpha_i = \pi_{\sigma_i}$ otherwise. Compositions are used to indicate the application of a rearrangement over a permutation, as we see in the following.

An *unsigned reversal* $\rho(i,j)$, with $1 \leq i < j \leq n$, is an operation that transforms an unsigned permutation π into the permutation $\pi \cdot \rho(i,j) = (\pi_1\ \pi_2 \dots \pi_{i-1}\ \underline{\pi_j\ \pi_{j-1} \dots \pi_{i+1}\ \pi_i}\ \pi_{j+1} \dots \pi_{n-1}\ \pi_n)$. For example, if $\pi = (1\,2\,3\,4\,5)$ is an unsigned permutation, then $\pi \cdot \rho(2,4) = (1\ \underline{4\ 3\ 2}\ 5)$.

A *signed reversal* $\bar{\rho}(i,j)$, with $1 \leq i \leq j \leq n$, is an operation that transforms a signed permutation π into the permutation $\pi \cdot \bar{\rho}(i,j) = (+\pi_1\ +\pi_2 \cdots + \pi_{i-1}\ \underline{-\pi_j\ -\pi_{j-1}\ \cdots -\pi_{i+1}\ -\pi_i}\ +\pi_{j+1} \cdots + \pi_{n-1}\ +\pi_n)$. For example, if $\pi = (+1 +2 -3 +4 +5)$ is a signed permutation, then $\pi \cdot \bar{\rho}(2,4) = (+1\underline{-4\ +3\ -2}+5)$.

A *transposition* $\tau(i,j,k)$, with $1 \leq i < j < k \leq n+1$, is an operation that transforms a (signed or unsigned) permutation π into the permutation $\pi \cdot \tau(i,j,k) = (\pi_1\ \pi_2\ \dots\ \pi_{i-1}\ \underline{\pi_j\ \dots\ \pi_{k-1}}\ \underline{\pi_i \dots\ \pi_{j-1}}\pi_k \dots \pi_n)$. For example, if $\pi = (1\,2\,3\,4\,5)$ is an unsigned permutation, then $\pi \cdot \tau(1,3,5) = (\underline{3\ 4}\ \underline{1\ 2}\ 5)$.

A λ-*reversal* is a reversal $\rho(i,j)$ such that $j - i + 1 \leq \lambda$, where $j - i + 1$ is the *size* of the reversal. A λ-*transposition* is a transposition $\tau(i,j,k)$ such that $k - i \leq \lambda$, where $k - i$ is the *size* of the transposition.

Every sorting problem is defined by a *rearrangement model*, denoted by β, which indicates what are the operations allowed to be applied in order to sort the permutation. Given a rearrangement model β and a permutation π, the *sorting distance*, denoted by $d_\beta(\pi)$, is the minimum amount of operations in β needed to transform π into ι. For an unsigned permutation π, if β allows only reversals, only transpositions, or both operations, we denote the sorting distance of π by $d_r(\pi), d_t(\pi)$, and $d_{rt}(\pi)$, respectively. For a signed permutation π, if β allows only reversals or both reversals and transpositions, we denote the sorting distance by $d_{\bar{r}}(\pi)$ and $d_{\bar{r}t}(\pi)$, respectively. Similarly, we denote by $d_r^\lambda(\pi)$, $d_{\bar{r}}^\lambda(\pi)$, $d_t^\lambda(\pi)$, $d_{rt}^\lambda(\pi)$, and $d_{\bar{r}t}^\lambda(\pi)$ the sorting distances for when we allow only λ-operations.

The next two definitions are used by some of the algorithms presented by us in the next sections. The *entropy of an element* π_i, denoted by $\text{ent}(\pi_i)$, is given by $||\pi_i| - i|$, that is, the distance between π_i and its position in ι. The *entropy of a permutation* π, denoted by $\text{ent}(\pi)$, is given by the sum of all values of $\text{ent}(\pi_i)$, for $1 \leq i \leq n$. For example, the entropy of the unsigned permutation

$\pi = (2\ 3\ 5\ 4\ 1)$ is $\text{ent}(\pi) = 1 + 1 + 2 + 0 + 4 = 8$. Note that, by definition, the entropy of any permutation is an even number.

For a permutation π, we say that there is an *inversion* (π_i, π_j) when we have $|\pi_i| > |\pi_j|$ and $i < j$. We denote the *number of inversions* in π by $\text{Inv}(\pi)$. For example, the inversions of the unsigned permutation $\pi = (3\,1\,4\,2)$ are the pairs (π_1, π_2), (π_1, π_4), and (π_3, π_4), so $\text{Inv}(\pi) = 3$.

3 Approximation Algorithms

We now present approximation algorithms for the five problems we are studying. Section 3.1 presents algorithms whose approximation factors are better for large values of λ, while Section 3.2 presents algorithms whose approximation factors are better for small values of λ. Regardless, all algorithms work for any $\lambda \geq 2$.

3.1 Approximation algorithms for large values of λ

The approximation algorithms shown in this section were obtained by using algorithms that already exist in the literature for the variants where the size of the rearrangements is not limited. Thus, Lemma 1 relates the distance of our problems with the distance of such variants.

Lemma 1. *For all permutations π and all $\lambda \geq 2$, we have $d_r^\lambda(\pi) \geq d_r(\pi)$, $d_{\bar{r}}^\lambda(\pi) \geq d_r(\pi)$, $d_t^\lambda(\pi) \geq d_t(\pi)$, $d_{rt}^\lambda(\pi) \geq d_{rt}(\pi)$, and $d_{\bar{r}t}^\lambda(\pi) \geq d_{\bar{r}t}(\pi)$.*

Proof. Any sorting sequence where the size of the rearrangements is limited by λ is valid for the case with no restriction. $\square$

Lemmas 2 and 3 show how to mimic any given (signed or unsigned) reversal and transposition with a sequence of (signed or unsigned) λ-reversals and λ-transpositions, respectively.

Lemma 2. *For a (signed or unsigned) permutation π and $\lambda \geq 2$, the effect of a reversal $\rho(i,j)$ of size $j-i+1 > \lambda$ can be obtained by at most $\frac{q(q+1)}{2}$ λ-reversals, where $q = \left\lceil \frac{j-i+1}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Proof. Initially, we divide the segment of π between positions i and j into subsegments of size $\lfloor \lambda/2 \rfloor$, except maybe for the one closest to j, which results in $q = \left\lceil \frac{j-i+1}{\lfloor \lambda/2 \rfloor} \right\rceil$ subsegments. Formally, for each $1 \leq \ell < q$, the ℓth subsegment contains elements of π from position $i + \lfloor \lambda/2 \rfloor(\ell - 1)$ to $i + \lfloor \lambda/2 \rfloor \ell - 1$, and the qth subsegment contains elements of π from position $i + \lfloor \lambda/2 \rfloor(q-1)$ to j. Note that the subsegments are defined by the elements contained in them. For example, in π the qth subsegment ends at position j but in $\pi \cdot \rho(i,j)$ (or in $\pi \cdot \bar{\rho}(i,j)$) it starts at position i. Even so, we can still refer to it as the qth subsegment.

The idea now is to move each subsegment until their respective positions in $\pi \cdot \rho(i,j)$ (or in $\pi \cdot \bar{\rho}(i,j)$) by exchanging a subsegment with the one to its right. Note that the reversals to do this have size at most $2\lfloor\lambda/2\rfloor$, and so they are λ-reversals.

For each value of ℓ, starting from $\ell = 1$ and going up to $\ell = q-1$, we apply a sequence of λ-reversals that first exchanges the ℓth subsegment with the $(\ell+1)$th subsegment, then it exchanges the ℓth with the $(\ell+2)$th, and so on, until it exchanges the ℓth with the qth subsegment. Note that after applying this sequence over the ℓth subsegment, it is correctly placed in its final order (relative to $\pi \cdot \rho(i,j)$ or $\pi \cdot \bar{\rho}(i,j)$) and the $(\ell+1)$th subsegment is currently starting at position i. Also, after applying the final sequence (of one λ-reversal) over the $(q-1)$th subsegment (the last one considered), subsegments $q-1$ and q are correctly placed in their final order.

Note that exactly $q-\ell$ λ-reversals are performed over the ℓth subsegment, so a total of $(q-1)+(q-2)+\cdots+1 = \frac{q(q-1)}{2}$ λ-reversals are required to correctly position all segments with this procedure. Now, if q is even, then at the end of the process we will directly have $\pi \cdot \rho(i,j)$ (or $\pi \cdot \bar{\rho}(i,j)$). Otherwise, all subsegments still have to be reversed and, therefore, another q reversals of size $\lfloor\lambda/2\rfloor$ (except, maybe, for the one over the qth subsegment) are applied, one for each subsegment. Thus, the effect of a reversal can be obtained by at most $\frac{q(q-1)}{2} + q = \frac{q(q+1)}{2}$ λ-reversals. □

As an example of the previous lemma, let $\pi = (1\,2\,3\,4\,5\,6\,7\,8)$ be an unsigned permutation and suppose we want to obtain $\pi \cdot \rho(2,6)$ with 4-reversals. Let $A = (2\ 3)$, $B = (4\ 5)$, and $C = (6)$ be the subsegments to be moved, as described in Lemma 2. Note that the total size of any two consecutive segments is less than or equal to $\lambda = 4$. The process described in the lemma first exchanges A with B, generating (1 5 4 3 2 6 7 8), and then with C, generating (1 5 4 6 2 3 7 8). Then it exchanges B with C, generating (1 6 4 5 2 3 7 8), and the process ends. Note that $q = \lceil (j-i+1)/\lfloor\lambda/2\rfloor \rceil = \lceil (6-2+1)/\lfloor 4/2\rfloor \rceil = \lceil 5/2 \rceil = 3$ is an odd number so, although the segments are in their correct positions, we still have to revert A, B, and C (actually, note that it is not necessary to revert C, since $|C| = 1$). Thus, we obtain $\pi \cdot \rho(2,6)$ with a total of $5 = 3+2 \leq q(q+1)/2 = (3\times 4)/2 = 6$ 4-reversals.

Lemma 3. *For a (signed or unsigned) permutation π and $\lambda \geq 2$, the effect of a transposition $\tau(i,j,k)$ of size $k-i > \lambda$ can be obtained by at most $\left\lceil \frac{j-i}{\lceil\lambda/2\rceil} \right\rceil \left\lceil \frac{k-j}{\lfloor\lambda/2\rfloor} \right\rceil$ λ-transpositions.*

Proof. Let F denote the first segment of the transposition, which contains elements of π comprised between positions i and $j-1$, and let S denote the second segment, which contains elements of π comprised between positions j and $k-1$.

We divide F into $f = \lceil (j-i)/\lceil \lambda/2 \rceil \rceil$ subsegments of size $\lceil \lambda/2 \rceil$, except maybe for the one that ends at $j-1$, and we divide S into $s = \lceil (k-j)/\lfloor \lambda/2 \rfloor \rceil$ subsegments of size $\lfloor \lambda/2 \rfloor$, also except maybe for the one that ends at $k-1$, in the following manner. For $1 \leq \ell < f$, the ℓth subsegment of F contains elements of π from position $i + \lceil \lambda/2 \rceil (\ell - 1)$ to $i + \lceil \lambda/2 \rceil \ell - 1$ and the fth subsegment of F contains elements of π from position $i + \lceil \lambda/2 \rceil (f-1)$ to $j-1$. For $1 \leq \ell < s$, the ℓth subsegment of S contains elements of π from position $j + \lfloor \lambda/2 \rfloor (\ell - 1)$ to $j + \lfloor \lambda/2 \rfloor \ell - 1$ and the sth subsegment of S contains elements of π from position $j + \lfloor \lambda/2 \rfloor (s-1)$ to $k-1$. Again, note that the segments F and S and their subsegments were defined by the elements they contain.

The idea now is to move each subsegment of F to their respective positions in $\pi \cdot \tau(i,j,k)$ by exchanging them with subsegments of S. The transpositions used will have size at most $\lceil \lambda/2 \rceil + \lfloor \lambda/2 \rfloor$ and so they are λ-transpositions.

For each value of ℓ, starting from $\ell = f$ and going down to $\ell = 1$, we apply a sequence of λ-transpositions that first exchanges the ℓth subsegment of F with the 1st segment of S, then exchanges the ℓth of F with the 2nd of S, and so on, until it exchanges the ℓth of F with the sth subsegment of S. Note that after applying this sequence on the ℓth subsegment of F, it ends up correctly placed in its final position (relative to $\pi \cdot \tau(i,j,k)$) and segment S is the same as in π, but beginning to the right of the $(\ell-1)$th subsegment of F. Thus, the next iteration (for $\ell - 1$) will correctly exchange the $(\ell-1)$th subsegment of F with all subsegments of S and place it in its final position.

Note that exactly s λ-transpositions are performed over the ℓth subsegment of F, so a total of $fs = \lceil (j-i)/\lceil \lambda/2 \rceil \rceil \lceil (k-j)/\lfloor \lambda/2 \rfloor \rceil$ λ-transpositions are required by this procedure to correctly position all segments. □

Theorems 4 to 12 use Lemmas 2 and 3 to obtain approximation algorithms for the five problems we are considering.

Theorem 4. *[Miranda et al., 2018b] Sorting Unsigned Permutations by λ-Reversals has an approximation algorithm of factor $0.6875p(p+1)$, where $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Corollary 5. *[Miranda et al., 2018b] Sorting Unsigned Permutations by λ-Reversals has an approximation algorithm of factor 8.25 for all $n > 3$ and $\lambda > \lceil n/2 \rceil$.*

Theorem 6. *Sorting Signed Permutations by λ-Reversals has an approximation algorithm of factor $\frac{p(p+1)}{2}$, where $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Proof. Similar to the proof of Theorem 4, but using the optimum algorithm for Sorting Signed Permutations by Reversals [Hannenhalli and Pevzner, 1999]. □

Corollary 7. *Sorting Signed Permutations by λ-Reversals has an approximation algorithm of factor 6 for all $n > 3$ and $\lambda > \lceil n/2 \rceil$.*

Theorem 8. *[Miranda et al., 2018b] Sorting Permutations by λ-Transpositions has an approximation algorithm of factor* $1.375 \left\lceil \frac{\lceil n/2 \rceil}{\lceil \lambda/2 \rceil} \right\rceil \left\lceil \frac{\lfloor n/2 \rfloor}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Corollary 9. *[Miranda et al., 2018b] Sorting Permutations by λ-Transpositions has an approximation algorithm of factor* 5.5 *for all $n > 3$ and $\lambda > \lceil n/2 \rceil$.*

Theorem 10. *[Miranda et al., 2018b] Sorting Unsigned Permutations by λ-Reversals and λ-Transpositions has an approximation algorithm of factor $\alpha\, p(p+1)$, where α is the approximation factor of the cycle decomposition algorithm for breakpoint graphs and $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Corollary 11. *[Miranda et al., 2018b] Sorting Unsigned Permutations by λ-Reversals and λ-Transpositions has an approximation algorithm of factor* 12α *for all $n > 3$ and $\lambda > \lceil n/2 \rceil$.*

Theorem 12. *Sorting Signed Permutations by λ-Reversals and λ-Transpositions has an approximation algorithm of factor $p(p+1)$, where $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Proof. Similar to the proof of Theorem 10, but using the 2-approximation algorithm for sorting signed permutations by reversals and transpositions [Walter et al., 1998]. □

Corollary 13. *Sorting Signed Permutations by λ-Reversals and λ-Transpositions has an approximation algorithm of factor* 12 *for all $n > 3$ and $\lambda > \lceil n/2 \rceil$.*

3.2 Approximation algorithms for small values of λ

In this section we present algorithms that have a better approximation when the value of λ is small. To do this, we first need to give some new definitions.

3.2.1 Entropy-based Algorithms for Unsigned and Signed Permutations

Let $\Delta_{\text{ent}}(\pi, \sigma) = \text{ent}(\pi \cdot \sigma) - \text{ent}(\pi)$ be the variation of the entropy after the application of an operation σ. Note that to calculate $\Delta_{\text{ent}}(\pi, \sigma)$, it suffices to determine the entropy of the elements affected by σ (the entropy of other elements does not change).

Observe that $\text{ent}(\iota) = 0$ and $\text{ent}(\pi) > 0$ for all unsigned permutations $\pi \neq \iota$. Lemma 14 gives an upper bound on the variation of entropy caused by any λ-reversal or λ-transposition. It implies in Corollary 15, which presents lower bounds for the distance of sorting unsigned permutation problems that we are addressing.

Lemma 14. *[Miranda et al., 2018b] Let $\Delta_{\text{ent}}^{max}(\pi, \sigma)$ be the maximum variation of entropy caused by a λ-operation σ over a (signed or unsigned) permutation π. We have $\Delta_{\text{ent}}^{max}(\pi, \sigma) = 2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor$.*

Corollary 15. *[Miranda et al., 2018b] For any unsigned permutation π, $\lambda \geq 2$, and $\beta \in \{r, t, rt\}$, we have $d_\beta^\lambda(\pi) \geq \text{ent}(\pi)/(2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor)$.*

Let $\text{neg}(\pi)$ denote the number of negative elements of a signed permutation π and observe that the only permutation with $\text{neg}(\pi) = \text{ent}(\pi) = 0$ is the identity. Let $\text{score}_{ent}(\pi, \sigma) = ((\text{ent}(\pi) - \text{ent}(\pi \cdot \sigma)) + (\text{neg}(\pi) - \text{neg}(\pi \cdot \sigma))$ be the entropy score of a λ-operation σ applied over π. Note that, if π is an unsigned permutation, then $\text{score}_{ent}(\pi, \sigma) = \text{ent}(\pi) - \text{ent}(\pi \cdot \sigma)$ is the change in the entropy after applying σ. Lemma 16 gives an upper bound for the entropy score of any λ-reversal or λ-transposition. It implies in Corollary 17, which presents lower bounds for the distance of sorting signed permutation problems that we are addressing.

Lemma 16. *Let* $\text{score}_{ent}^{max}(\pi, \sigma)$ *be the maximum possible entropy score of a λ-operation σ. We have* $\text{score}_{ent}^{max}(\pi, \sigma) = 2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda$.

Proof. By Lemma 14, we have that the maximum entropy variation of a λ-operation is $2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor$. By definition, a λ-operation σ involves at most λ elements of a permutation and so at most λ of them can become positive ones. Therefore, $\text{score}_{ent}^{max}(\pi, \sigma) = 2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda$. □

Corollary 17. *For any signed permutation π, $\lambda \geq 2$, and $\beta \in \{\bar{r}, \bar{r}t\}$, we have $d_\beta^\lambda(\pi) \geq (\text{ent}(\pi) + \text{neg}(\pi))/((2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor) + \lambda)$.*

Let π be a (signed or unsigned) permutation of size n and let i and j be two integers such that $1 \leq i < j \leq n$. Function $\phi(\pi, i, j)$ returns a permutation π' such that $\pi'_i = \pi_j$, $\pi'_j = \pi_i$, and $\pi'_k = \pi_k$ for all $k \notin \{i, j\}$. In other words, only elements π_i and π_j are exchanged and, if π is a signed permutation, the sign of all elements from π remain the same in π'. Lemmas 18, 19, and 20 show how to obtain $\phi(\pi, i, j)$ with λ-reversals and λ-transpositions, respectively. The proof of Lemma 18 is reproduced [Miranda et al., 2018b] here because it is used in Lemma 19.

Lemma 18. *[Miranda et al., 2018b] Let π be an unsigned permutation, $\lambda \geq 2$, and i and j be positions such that $1 \leq i < j \leq n$. It is possible to obtain $\phi(\pi, i, j)$ by applying at most $2x$ λ-reversals on π, where $x = \left\lceil \frac{j-i}{\lambda-1} \right\rceil$.*

Proof. We show that the result follows by considering two cases, according to the relation between i and j. If $j - i \leq \lambda - 1$, then at most two λ-reversals are necessary. First we apply the operation $\rho(i, j)$ that exchanges the position of elements π_i and π_j. It is easy to see that, if $j - i + 1 \leq 3$, we already obtained $\phi(\pi, i, j)$ with only this operation. Otherwise, observe that we will have the segment $\pi_{i+1}, \ldots, \pi_{j-1}$ in reverse order (regarding $\phi(\pi, i, j)$). Thus, we have to apply a second operation $\rho(i + 1, j - 1)$ to revert it again and, then, we obtain $\phi(\pi, i, j)$ with two λ-reversals.

Otherwise, $j - i > \lambda - 1$. In a first step, we move element π_i to position j by repeatedly increasing its position by $\lambda - 1$ (except, maybe, at the last movement) with exactly $x = \lceil (j-i)/(\lambda-1) \rceil$ λ-reversals applied successively. Formally, this is done by applying the sequence $\rho(i, i + (\lambda - 1))$, $\rho(i + (\lambda - 1), i + 2(\lambda - 1))$, $\rho(i + 2(\lambda - 1), i + 3(\lambda - 1))$, $\ldots$, $\rho(i + (x - 1)(\lambda - 1), j)$ of λ-reversals. Now element π_j is at position $i + (x-1)(\lambda-1)$ and elements π_t, for $i < t < j$, are not necessarily at position t. To correct this and, at the same time, move element π_j to position i, we have a second step that applies a sequence with the same λ-reversals that were used before (except for the last λ-reversal) in reversed order, to repeatedly decrease the position of π_j by $\lambda - 1$. Thus, a total of $x - 1$ extra operations are needed. Formally, the sequence is $\rho(i + (x - 2)(\lambda - 1), i + (x - 1)(\lambda - 1))$, $\rho(i + (x - 3)(\lambda - 1), i + (x - 2)(\lambda - 1))$, $\ldots$, $\rho(i + 2(\lambda - 1), i + 3(\lambda - 1))$, $\rho(i + (\lambda - 1), i + 2(\lambda - 1))$, $\rho(i, i + (\lambda - 1))$. At this point, if the size of the λ-reversal $\rho(i + (x - 1)(\lambda - 1), j)$ (the last λ-reversal of the first step) is less than or equal to 3, then at the end of the process we will directly have $\phi(\pi, i, j)$. Otherwise, we have to apply one more λ-reversal $\rho(i + (x - 1)(\lambda - 1) + 1, j - 1)$ to obtain $\phi(\pi, i, j)$, which totalizes $2x$ λ-reversals. □

Lemma 19. *Let π be a signed permutation and i and j be positions such that $1 \leq i < j \leq n$. It is possible to obtain $\phi(\pi, i, j)$ through π by applying at most $2x + 2$ λ-reversals on π, where $x = \left\lceil \frac{j-i}{\lambda-1} \right\rceil$.*

Proof. We show the result by considering two cases, according to the relation between i and j. If $j - i \leq \lambda - 1$, then at most four λ-reversals are necessary. First we apply the three operations $\rho(\bar{i}, j)$, $\rho(\bar{i}, i)$, and $\rho(\bar{j}, j)$ that exchange the position of elements π_i and π_j and also correct their signs (regarding $\phi(\pi, i, j)$). It is easy to see that, if $j - i + 1 \leq 2$, we already obtained $\phi(\pi, i, j)$ with three operations. Otherwise, observe that we will have the segment $\pi_{i+1}, \ldots, \pi_{j-1}$ in reverse order and with the opposite signs (regarding $\phi(\pi, i, j)$). Thus, we have to apply an extra operation $\rho(i + \bar{1}, j - 1)$ to correct it and, then, we obtain $\phi(\pi, i, j)$ with four operations.

Otherwise, $j - i > \lambda - 1$. Let $\pi' = \phi(\pi, i, j)$. Initially we apply the same (at most) $2x$ λ-reversals Lemma 18 (but now they are signed λ-reversals) to obtain a permutation π'' such that $|\pi''_t| = |\pi'_t|$ for all $i \leq t \leq j$. Note that this sequence applies exactly two λ-reversals over elements $\pi_{t'}$ where $i < t' < i+(x-1)(\lambda-1)$, so we actually have $\pi''_{t'} = \pi'_{t'}$ for all such t'. Also, it is easy to see that if x is an even number, then we will have $\pi''_i = \pi'_i$ and $\pi''_j = \pi'_j$. Otherwise, two extra unitary λ-reversal $\bar{\rho}(i, i)$ and $\bar{\rho}(j, j)$ have to be applied to fix the signs of π_i and π_j.

Now we look at elements $\pi''_{(i+(x-1)(\lambda-1))}, \ldots, \pi''_{j-1}$. Observe that, if exactly $2x$ λ-reversals were used in the previous sequence, then these elements were also affected twice each and, thus, we have $\pi'' = \pi'$ with at most $2x + 2$ operations. Otherwise, it means that $2x - 1$ λ-reversals were used to obtain π'' and the λ-reversal $\bar{\rho}(i + (x - 1)(\lambda - 1), j)$ (the last operation in the sequence of λ-reversals described at first step of the proof of Lemma 18) had size less than or equal to 3. If the size was equal to 3, that λ-reversal affected the elements π_i, π_{j-1}, and π_j, so we just have to apply one more unitary λ-reversal $\bar{\rho}(j - 1, j - 1)$ to obtain $\pi'' = \pi'$, which results in at most $2x + 2$ λ-reversals. Otherwise, the size of $\bar{\rho}(i + (x - 1)(\lambda - 1), j)$ was equal to 2, so only the elements π_i and π_j were involved by the operation. Thus, we already have $\pi'' = \pi'$ with a total of at most $2x + 1$ λ-reversals. □

Lemma 20. *Let π be a (signed or unsigned) permutation, $\lambda \geq 2$, and i and j be positions such that $1 \leq i < j \leq n$. It is possible to obtain $\phi(\pi, i, j)$ by applying $x + y$ λ-transpositions on π, where $x = \left\lceil \frac{j-i}{\lambda-1} \right\rceil$ and $y = \left\lceil \frac{j-i-1}{\lambda-1} \right\rceil$.*

Proof. We show that the result follows by considering two cases, according to the relation between i and j. If $j - i \leq \lambda - 1$, then at most two λ-transpositions are necessary. First we apply $\tau(i, i + 1, j + 1)$, that places element π_i at position j. When $j = i + 1$, it also places element π_j at position i and, since there are no elements between π_i and π_j, we obtain $\phi(\pi, i, j)$ with one operation. Otherwise, observe that there will be π_t, for $i < t \leq j$, exactly one position to the left of their original position in π. Thus, we have to apply a second operation $\tau(i, j - 1, j)$ in order to correct this. Note that, after applying this second operation, we have π_j at position i at the same time that elements $i < t < j$ were moved one position to the right and, then, we got $\phi(\pi, i, j)$ with two λ-transpositions.

Otherwise, $j - i > \lambda - 1$. Initially we move element π_i to position j by repeatedly increasing its position by $\lambda - 1$ (except, maybe, at the last movement) with exactly $x = \lceil (j-i)/(\lambda-1) \rceil$ λ-transpositions applied successively. Formally, this is done by applying the sequence of λ-transpositions $\tau(i, i + 1, i + \lambda)$, $\tau(i + (\lambda-1), i+(\lambda-1)+1, i+2(\lambda-1))$, $\tau(i+2(\lambda-1), i+2(\lambda-1)+1, i+2(\lambda-1)+\lambda)$, $\ldots$, $\tau(i + (x - 1)(\lambda - 1), i + (x - 1)(\lambda - 1) + 1, j + 1)$. After this, each element π_t, for $i < t \leq j$, is exactly one position to the left of its original position in π.

To correct this, we can apply a similar sequence of λ-transpositions, but now we will repeatedly decrease the position of π_j (which is at position $j-1$) by $\lambda-1$ (except, maybe, at the last operation) with exactly $y = \lceil (j-1-i)/(\lambda-1) \rceil$ λ-transpositions applied successively. Formally, the sequence is $\tau(j-\lambda, j-1, j)$, $\tau(j-(\lambda-1)-\lambda, j-(\lambda-1)-1, j-(\lambda-1))$, $\tau(j-2(\lambda-1)-\lambda, j-2(\lambda-1)-1, j-2(\lambda-1))$, $\ldots$, $\tau(i, j-y(\lambda-1)-1, j-y(\lambda-1))$. Note that each λ-transposition moves $\lambda-1$ (again, except maybe for the last operation) elements π_t, for $i < t < j$, one position to the right by exchanging all of them with element π_j. At the end of this process we directly have $\phi(\pi, i, j)$. □

Lemma 21 is auxiliar to Lemma 22, which shows that it is always possible to reduce the entropy of any permutation π such that $\mathrm{ent}(\pi) > 0$.

Lemma 21. *For all (signed or unsigned) permutations π with $\mathrm{ent}(\pi) > 0$, there exists a pair of elements π_i and π_j, with $1 \leq i < j \leq n$, such that $|\pi_i| \geq j$ and $|\pi_j| \leq i$.*

Proof. Let G_π be the directed graph such that $V(G_\pi) = \{1, 2, \ldots, n\}$ and $E(G_\pi) = \{(|\pi_i|, i)\colon 1 \leq i \leq n\}$. Note that each vertex has in-degree 1 and out-degree 1, and, therefore, the components of G_π are cycles. Also note that only G_ι has n unitary cycles.

Let C be any cycle of G_π with at least two vertices and let u be the smallest-value vertex of C. Let $B = (v_1, v_2, \ldots, v_\ell)$ be a maximal sequence of vertices of C such that $v_1 = u$, $v_i < v_{i+1}$ for all $1 \leq i < \ell$, and $(v_i, v_{i+1}) \in E(G_\pi)$.

Since the vertices of B are in a cycle and B is maximal, the edge incident to v_ℓ is of the form (v_ℓ, x), with $x < v_\ell$. If $v_{\ell-1} \leq x$, then take $i = x$ and $j = v_\ell$. In this case, we have $|\pi_i| = |\pi_x| = v_\ell = j$ and $|\pi_j| = |\pi_{v_\ell}| = v_{\ell-1} \leq x = i$ and the lemma follows. If $v_{\ell-1} > x$, then let k, with $1 \leq k < \ell-1$, be such that $v_k \leq x < v_{k+1}$ and take $i = x$ and $j = v_{k+1}$. In this case, we have $|\pi_i| = |\pi_x| = v_\ell > v_{k+1} = j$ and $|\pi_j| = |\pi_{v_{k+1}}| = v_k \leq x = i$ and the lemma follows. See Figure 3 for an example. □

Lemma 22. *For all (signed or unsigned) permutations π with $\mathrm{ent}(\pi) > 0$, it is possible to obtain another permutation $\phi(\pi, i, j)$ such that $\mathrm{ent}(\phi(\pi, i, j)) = \mathrm{ent}(\pi) - 2(j-i)$, for some pair $1 \leq i < j \leq n$.*

Proof. Let i and j be as in Lemma 21. By definition, $\mathrm{ent}(\pi) - \mathrm{ent}(\phi(\pi, i, j)) = ||\pi_i|-i| + ||\pi_j|-j| - ||\pi_i|-j| - ||\pi_j|-i|$. Since $|\pi_i| \geq j > i$, we have $||\pi_i|-j| = |\pi_i|-j$ and $||\pi_i|-i| = |\pi_i|-i$. Since $|\pi_j| \leq i < j$, we have $||\pi_j|-i| = i-|\pi_j|$ and $||\pi_j|-j| = j-|\pi_j|$. Therefore, $\mathrm{ent}(\pi) - \mathrm{ent}(\phi(\pi, i, j)) = |\pi_i| - i + j - |\pi_j| - |\pi_i| + j - i + |\pi_j|$ $= 2(j-i)$. □

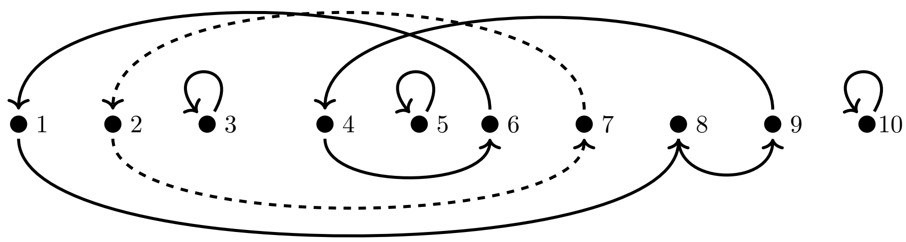


Figure 3: Graph G_π for $\pi = (6\ 7\ 3\ 9\ 5\ 4\ 2\ 1\ 8\ 10)$. Considering the notation of Lemma 21's proof, if $C = (1, 8, 9, 4, 6, 1)$, then we can take $B = (1, 8, 9)$.

A permutation $\pi \neq \iota$ is called *normal* if there exists at least one λ-operation σ such that $\mathrm{ent}(\pi) > \mathrm{ent}(\pi \cdot \sigma)$ and $\mathrm{score}_{ent}(\pi, \sigma) > 0$. Moreover, since the entropy of any permutation is an even number, we have $\mathrm{score}_{ent}(\pi, \sigma) \geq 2$ and $\mathrm{score}_{ent}(\pi, \sigma) \geq 1$ for unsigned and signed normal permutations π, respectively. We call π a *special* permutation in case it is not normal.

Consider the signed normal permutation $\pi = (-4\ +5\ -2\ -3\ +1)$, for which $\mathrm{ent}(\pi) = 12$ and $\mathrm{neg}(\pi) = 3$. Note that $\mathrm{score}_{ent}(\pi, \bar{\rho}(2, 3)) = 2$ but $\mathrm{score}_{ent}(\pi, \bar{\rho}(1, 5)) = 9$. The following algorithm is greedy in the sense of always choosing a λ-operation with highest entropy score, if possible. When the permutation is special, it applies Lemma 22. Note that, in both cases, the algorithm reduces the amount of entropy, thus it eventually reaches a permutation γ such that $\mathrm{ent}(\gamma) = 0$. For the sorting unsigned permutations problems, at this point we already have $\gamma = \iota$. For the sorting signed permutations problems, the algorithm still applies $\mathrm{neg}(\gamma)$ unitary reversals to reach ι.

The algorithm receives the rearrangement model β, since it works for any of the problems we are considering, and it runs in polynomial time, as we discuss later. We show its approximation factor in Theorem 23 and Theorem 24, for the problems of sorting unsigned and signed permutations, respectively.

Theorem 23. *[Miranda et al., 2018b] Sorting Unsigned Permutations by λ-Reversals, by λ-Transpositions, or by λ-Reversals and λ-Transpositions has an approximation algorithm of factor* $4\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor$.

Theorem 24. *Sorting Signed Permutations by λ-Reversals, or by λ-Reversals and λ-Transpositions has an approximation algorithm of factor* $8\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda$.

Proof. If π is a normal permutation, then the algorithm chooses a λ-operation σ with $\mathrm{score}_{ent}(\pi, \sigma) \geq 1$.

function GreedyAlgorithm(π, λ, β)
  while $\text{ent}(\pi) > 0$ **do**
    if π is a normal permutation **then**
      Let σ be a λ-operation such that $\text{score}_{ent}(\pi, \sigma)$ is maximum and $\text{ent}(\pi) - \text{ent}(\pi \cdot \sigma) > 0$
      $\pi \leftarrow \pi \cdot \sigma$
    else
      Let i and j be such that $|\pi_i| \geq j$, $|\pi_j| \leq i$, and $1 \leq i < j \leq n$
      $\pi \leftarrow \phi(\pi, i, j)$, according to Lemma 18, 19, or 20
  if π is a signed permutation **then**
    Apply a unitary reversal $\bar{\rho}(i, i)$ in each $\pi_i < 0$

Otherwise, let i and j be as in GreedyAlgorithm and note that $\pi' = \phi(\pi, i, j)$ is a permutation with $\text{ent}(\pi') = \text{ent}(\pi) - 2(j - i)$ (see Lemma 22). Also, we obtain π' through the operations described in Lemmas 19 and 20 and so $\text{neg}(\pi') = \text{neg}(\pi)$ in this case.

If $j - i \leq \lambda - 1$, then we obtain π' by applying at most 4 λ-operations. Therefore, the entropy score obtained per operation is at least $(2(j - i))/4 \geq 2/4 = 1/2$.

If $j - i > \lambda - 1$, then we obtain π' by applying at most $2\lceil (j-i)/(\lambda-1) \rceil + 2$ λ-reversals or $\lceil (j-i)/(\lambda-1) \rceil + \lceil (j-i-1)/(\lambda-1) \rceil$ λ-transpositions. Since $2\lceil (j-i)/(\lambda-1) \rceil + 2 \geq \lceil (j-i)/(\lambda-1) \rceil + \lceil (j-i-1)/(\lambda-1) \rceil$, the entropy score obtained per operation is at least $(2(j-i))/(2\lceil \frac{j-i}{\lambda-1} \rceil + 2) \geq (2(j-i))/(4\lceil \frac{j-i}{\lambda-1} \rceil) \geq (j-i)/(4\frac{j-i}{\lambda-1}) = \frac{\lambda-1}{4} \geq \frac{1}{4}$.

Thus, in the worst case we have 1/4 entropy score per operation for both sorting signed permutation problems. When we obtain a permutation γ with $\text{ent}(\gamma) = 0$, the algorithm applies $\text{neg}(\gamma)$ extra unitary λ-reversals in order to obtain the identity. This means that the total amount of operations used by GreedyAlgorithm is at most

$$\begin{aligned} &\frac{(\text{ent}(\pi) + \text{neg}(\pi)) - (\text{ent}(\gamma) + \text{neg}(\gamma))}{1/4} + \text{neg}(\gamma) \\ &= 4\left(\text{ent}(\pi) + \text{neg}(\pi) - \text{neg}(\gamma)\right) + \text{neg}(\gamma) \\ &= 4\left(\text{ent}(\pi) + \text{neg}(\pi)\right) - 4\,\text{neg}(\gamma) + \text{neg}(\gamma) \\ &\leq 4\left(\text{ent}(\pi) + \text{neg}(\pi)\right) \leq \left(8\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda\right) d_\beta^\lambda(\pi) \ , \end{aligned}$$

where the last inequality follows from Lemma 14 and $\beta = \{\bar{r}, \bar{r}t\}$. □

The following definition is used by Lemma 25 and by Theorem 26, which show that the algorithm guarantees constant approximation factor for all signed permutations $\pi \neq \iota$ such that $\text{ent}(\pi) = 0$. A *breakpoint* of a signed permutation π, with two extra elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$, is defined as a pair of elements (π_i, π_{i+1}), for $0 \leq i \leq n$, such that $\pi_{i+1} - \pi_i \neq 1$. The amount of breakpoints in π is denoted by $b(\pi)$. Note that the while loop of GreedyAlgorithm does not change the sign of elements of π. Also observe that, when $\pi \neq \iota$ and $\text{ent}(\pi) = 0$, we have $\text{neg}(\pi) = b(\pi) - 1$. Since any reversal or transposition can remove at most two or three breakpoints, respectively, we conclude Lemma 25.

Lemma 25. *For all signed permutations π such that $\text{ent}(\pi) = 0$ and $\lambda \geq 2$, we have $d_{\bar{r}}^{\lambda}(\pi) \geq \frac{b(\pi)}{2} = \frac{\text{neg}(\pi)+1}{2}$ and $d_{\bar{r}t}^{\lambda}(\pi) \geq \frac{b(\pi)}{3} = \frac{\text{neg}(\pi)+1}{3}$.*

Theorem 26. *Sorting Signed Permutations by λ-Reversals and by λ-Reversals and λ-Transpositions have approximation algorithms of factors 2 and 3, respectively, for all signed permutations π such that $\text{ent}(\pi) = 0$.*

Proof. When $\text{ent}(\pi) = 0$, the algorithm only applies $\text{neg}(\pi)$ unitary λ-reversals to reach ι. By the lower bound of Lemma 25, we have $\text{neg}(\pi) \leq 2d_{\bar{r}}(\pi) - 1 \leq 2d_{\bar{r}}(\pi)$ and $\text{neg}(\pi) \leq 3d_{\bar{r}t}(\pi) - 1 \leq 3d_{\bar{r}t}(\pi)$. □

Regarding the time complexity of the algorithm, first observe that the while loop at the first line runs $O(n^2)$ times because $\text{ent}(\pi) + \text{neg}(\pi)$ is $O(n^2)$ and, as seen in theorems 23 and 24, one operation in the worst case has entropy score at least 1/2 or 1/4. In the "if" command we must decide whether a permutation is normal, which can be done by finding the desired λ-operation. We can simply test all possible λ-operations, which takes time $O(n\lambda)$ for λ-reversals and $O(n\lambda^2)$ for λ-transpositions. Furthermore, it takes $O(\lambda)$ time to calculate the entropy score of a λ-operation. Therefore, testing if a permutation is normal takes time $O(n\lambda^3)$. In the "else" command, we must first obtain positions i and j as desired, which takes time $O(n)$. Now consider obtaining $\phi(\pi, i, j)$ with λ-transpositions. By Lemma 20, we can use at most $\lceil (j-i)/(\lambda-1) \rceil + \lceil (j-i-1)/(\lambda-1) \rceil$ such operations, each one of size at most λ, which means the time to perform each transposition is $O(\lambda)$ and so the total time to obtain $\phi(\pi, i, j)$ is at most $O(\lambda)\left(\frac{j-i+1}{\lambda-1} + \frac{j-i}{\lambda-1}\right) \leq O(\lambda)2\frac{n}{\lambda-1} = O(n\lambda)$, since $\lambda \geq 2$. Similarly, the time to obtain $\phi(\pi, i, j)$ with reversals is also $O(n\lambda)$. For signed permutations, at most $\text{neg}(\pi) \leq n$ extra unitary λ-reversals are performed, taking time $O(n)$. Therefore, the total time of GreedyAlgorithm is $O(n^3\lambda^3)$ for any of the problems we are considering. Note that this is polynomial because $\lambda = O(n)$.

3.2.2 Inversions-based Algorithms for Unsigned Permutations

Next lemma shows that we always have an inversion in a permutation $\pi \neq \iota$, and then Lemma 28 shows an upper bound on the number of inversions that a λ-operation can remove.

Lemma 27. *For all (signed or unsigned) permutations π such that* $\mathrm{Inv}(\pi) > 0$ *there exists an inversion* (π_i, π_{i+1}).

Proof. Let π_1, π_2, ..., π_i be a maximal subsequence such that $|\pi_1| < |\pi_2| < \cdots < |\pi_i|$. Since $\mathrm{Inv}(\pi) > 0$, we have that $i < n$ and, thus, $|\pi_i| > |\pi_{i+1}|$ and (π_i, π_{i+1}) is an inversion. □

Lemma 28. *Let π be a unsigned permutation. Let π' be the resulting permutation after that one λ-operation is applied over π. We have* $\mathrm{Inv}(\pi) - \mathrm{Inv}(\pi') \leq \lambda(\lambda - 1)/2$.

Proof. Directly from the observation that any λ-reversal or λ-transposition can remove or add at most $\lambda(\lambda - 1)/2$ inversions. □

Since the only unsigned permutation with $\mathrm{Inv}(\pi) = 0$ is the identity, we conclude lower bounds for the problems of Sorting Unsigned Permutations by λ-Operations in Corollary 29. Then, a greedy approximation algorithm is presented in Theorem 30, which is based on the idea of always reducing as many inversions as possible from a given unsigned permutation π.

Corollary 29. *For all unsigned permutations π and $\lambda \geq 2$ we have $d^{\lambda}_{\beta}(\pi) \geq 2\,\mathrm{Inv}(\pi)/(\lambda(\lambda - 1))$, for $\beta \in \{r, rt, t\}$.*

Theorem 30. *The problems of Sorting Unsigned Permutations by λ-Reversals, by λ-Transpositions and by λ-Reversals and λ-Transpositions have* $(\lambda(\lambda-1)/2)$-*approximation algorithms.*

Proof. Let $\lambda \geq 2$ be an integer and let $\pi \neq \iota$ be an unsigned permutation. Since the only permutation with $\mathrm{Inv}(\pi) = 0$ is the identity, we can design a greedy algorithm that repeatedly chooses a λ-operation which decreases $\mathrm{Inv}(\pi)$ by the largest possible amount and it will occasionally sort the permutation. By Lemma 27, we always have an inversion (π_i, π_{i+1}) in π and, so, our algorithm decreases at least one inversion at a time. Therefore, the number of operations needed by such greedy algorithm is at most $\mathrm{Inv}(\pi)$ and its approximation factor follows directly from Corollary 29. □

Note that $\mathrm{Inv}(\pi)$ is $O(n^2)$ and that $O(\lambda^2)$ λ-reversals and/or $O(\lambda^3)$ λ-transpositions have to be considered in order to find the best operation in the greedy step of the algorithms. Since we can calculate the number of inversions' variation in time $O(\lambda\sqrt{\log(\lambda)})$ [Chan and Pătraşcu, 2010], the time complexity of the algorithms for Sorting Unsigned Permutations by λ-Reversals and for the other two problems of sorting unsigned permutations that we are addressing is $O(n^2\lambda^3\sqrt{\log(\lambda)})$ and $O(n^2\lambda^4\sqrt{\log(\lambda)})$, respectively.

3.2.3 Algorithms based on Entropy and Inversions for Signed Permutations

We denote by $E_\pi^{even^-}$ (resp. $E_\pi^{odd^+}$) the set of negative (resp. positive) elements in π such that $\text{ent}(\pi_i)$ is even (resp. odd). As an example, given the signed permutation $\pi = (-5\ +2\ -1\ -3\ +4)$, we have $E_\pi^{even^-} = \{-5, -1\}$, because $\text{ent}(-5) = 4$ and $\text{ent}(-1) = 2$, and we have $E_\pi^{odd^+} = \{+4\}$, because $\text{ent}(+4) = 1$.

Lemmas 31, 32, and 33 are used to obtain lower bounds for the problems of Sorting Signed Permutations by λ-Reversals and Sorting Signed Permutations by λ-Reversals and λ-Transpositions.

Lemma 31. *[Galvão et al., 2015] Let π be a signed permutation and let $\pi' = \pi \cdot \bar{\rho}(i, i+1)$. We have $|E_\pi^{even^-}| + |E_\pi^{odd^+}| = |E_{\pi'}^{even^-}| + |E_{\pi'}^{odd^+}|$.*

Lemma 32. *Let π be a signed permutation. Let π' be the resulting permutation after one λ-operation is applied over π. We have $(|E_\pi^{even^-}| + |E_\pi^{odd^+}|) - (|E_{\pi'}^{even^-}| + |E_{\pi'}^{odd^+}|) \leq \lambda$.*

Proof. One λ-operation involves at most λ elements. □

Let $\text{score}_{inv}(\pi, \sigma) = (\text{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|) - (|\text{Inv}(\pi \cdot \sigma)| + |E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$ be the inversions score of one λ-operation σ when applied over π.

Lemma 33. $\text{score}_{inv}(\pi, \sigma) \leq \lambda(\lambda - 1)/2 + \lambda$.

Proof. Directly from lemmas 28 and 32. □

Since the only signed permutation with $\text{Inv}(\pi) = |E_\pi^{even^-}| = |E_\pi^{odd^+}| = 0$ is the identity, we conclude lower bounds for the problems of Sorting Signed Permutations by λ-Operations in Corollary 34.

Corollary 34. *For all signed permutations $\pi \neq \iota$ and all $\lambda \geq 2$, we have $d_\beta^\lambda(\pi) \geq 2(\text{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|)/(\lambda(\lambda - 1) + 2\lambda)$, for $\beta \in \{\bar{r}, \bar{r}t\}$.*

Next lemma shows that we always have a λ-operation with inversions score 1. Then, Theorem 36 presents greedy approximation algorithms based on the idea of always choosing λ-operations which have the greatest inversions score.

Lemma 35. *For any signed permutation $\pi \neq \iota$ and $\lambda \geq 2$, there always exist a λ-operation σ such that* $\text{score}_{inv}(\pi, \sigma) = 1$.

Proof. We divide the proof into two cases, according to the value of $\text{Inv}(\pi)$.

If $\text{Inv}(\pi) > 0$, then, by Lemma 27, there exists an inversion (π_i, π_{i+1}) and, by Lemma 31, we can apply a λ-reversal $\sigma = \bar{\rho}(i, i+1)$ to get $\text{Inv}(\pi \cdot \sigma) = \text{Inv}(\pi) - 1$

at the same time we hold $(|E_{\pi}^{even^-}|+|E_{\pi}^{odd^+}|) = (|E_{\pi\cdot\sigma}^{even^-}|+|E_{\pi\cdot\sigma}^{odd^+}|)$. Therefore, $\text{score}_{inv}(\pi,\sigma) = 1$.

If $\text{Inv}(\pi) = 0$, then all elements have entropy equal to zero. Thus, we also have $|E_{\pi}^{odd^+}| = 0$. With this, we can apply a unitary λ-reversal σ over each negative element in order to get $|E_{\pi\cdot\sigma}^{even^-}| = |E_{\pi}^{even^-}| - 1$ at the same time we hold $\text{Inv}(\pi\cdot\sigma) = |E_{\pi\cdot\sigma}^{odd^+}| = 0$. Therefore, $\text{score}_{inv}(\pi,\sigma) = 1$. $\square$

Theorem 36. *The problems of Sorting Signed Permutations by λ-Reversals and by λ-Reversals and λ-Transpositions have $(\frac{\lambda(\lambda-1)}{2}+\lambda)$-approximation algorithms.*

Proof. Let $\lambda \geq 2$ be an integer and let $\pi \neq \iota$ be a signed permutation. Since the only permutation with $\text{Inv}(\pi) = |E_{\pi}^{even^-}| = |E_{\pi}^{odd^+}| = 0$ is the identity, a greedy algorithm that repeatedly chooses a λ-operation σ with the greatest value of $\text{score}_{inv}(\pi,\sigma)$ will occasionally sort the permutation. By Lemma 35, we always have a λ-operation σ such that $\text{score}_{inv}(\pi,\sigma) = 1$. Therefore, the number of operations needed by our algorithm is at most $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ and its approximation factor follows directly from Corollary 34. $\square$

Since $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ is $O(n^2)$, the time complexity analysis of these greedy algorithms is analogous to the analysis done for the inversions-based algorithms presented for unsigned permutations.

4 Experimental Results

We have implemented all the approximation algorithms presented in order to analyze how they work from a practical perspective. The goal of the experiments was to compare the algorithms according to the approximation factor they get in practice. We calculated the approximation factor for each permutation used as input dividing the size of the sorting sequences generated by the algorithms by the maximum value among 4 different lower bounds. One of the lower bounds comes from the literature algorithms for the problems without limited-size restriction, which are the ones we used to create the approximation algorithms for large values of λ described in Section 3.1. Those lower bounds are valid for λ-operations, as Lemma 1 shows. In the following, we describe the literature algorithms used in the experiments and the considered lower bounds:

- Sorting Signed Permutations by Reversals: the optimal algorithm for Sorting Signed Permutations by Reversals was implemented and, for each permutation, the distance was used as a lower bound for the problem of Sorting Signed Permutations by λ-Reversals;
- Sorting Unsigned Permutations by Reversals: each unsigned permutation was transformed into a signed one using the $(1.4193+\epsilon)$-approximation

algorithm, for $\epsilon > 0$, for cycle decomposition of the breakpoint graph [Lin and Jiang, 2004] and its optimal sorting sequence was retrieved. This sequence's size divided by 1.42, assuming $\epsilon = 0.007$, was used as a lower bound for Sorting Unsigned Permutations by λ-Reversals;

- Sorting Permutations by Transpositions: a 1.5-approximation algorithm [Bafna and Pevzner, 1998] was implemented. The authors who presented this algorithm defined $c_{odd}(\pi)$ as the number of odd components in the cycle graph, a structure presented by them, and gave the lower bound $d_t(\pi) \geq (n + 1 - c_{odd}(\pi))/2$ for the problem. We are considering the same lower bound for Sorting Permutations by λ-Transpositions;

- Sorting Signed Permutations by Reversals and Transpositions: a 2-approximation algorithm [Walter et al., 1998] was implemented. The authors who presented this algorithm gave the lower bound $d_{\bar{r}t}(\pi) \geq n + 1 - c_{odd}(\pi)$. We are considering the same as a lower bound for Sorting Signed Permutations by λ-Reversals and λ-Transpositions.

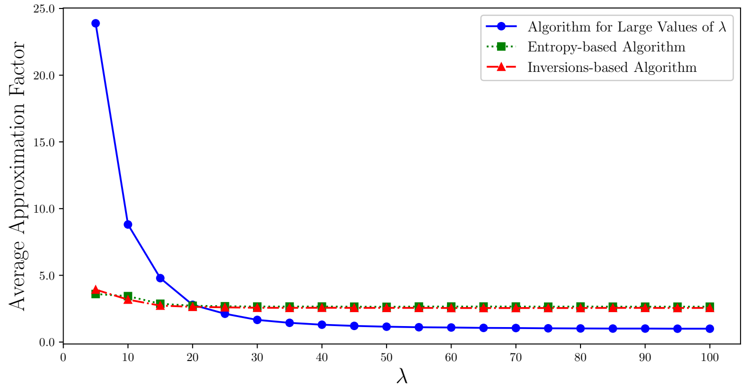
- Sorting Unsigned Permutations by Reversals and Transpositions: a 3-approximation algorithm [Walter et al., 1998] was implemented. The authors who presented this algorithm considered the lower bound $d_{rt}(\pi) \geq \frac{b(\pi)}{3}$ showed in Lemma 25. We are considering the same as a lower bound for Sorting Unsigned Permutations by λ-Reversals and λ-Transpositions.

The other three lower bounds considered in our experiments are the ones (i) based on entropy, presented in Lemma 15 and Lemma 17, (ii) based on inversions, presented in Lemma 29 and Lemma 34, and (iii) based on breakpoints, presented in Lemma 25.

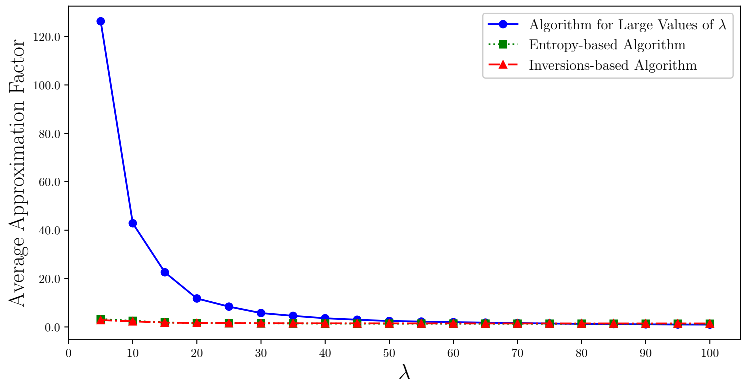
As input for our algorithms we considered a total of 1000 random permutations of size 100, and values of $\lambda = 5, 10, 15, \ldots, 100$. We show the results of our experiments in Figure 4 and Figure 5.

We point out that for the problems of Sorting (Signed or Unsigned) Permutations by λ-Reversals, the algorithm for large values of λ had the best average approximation factor for most values of λ. For the problem of Sorting Permutations by λ-Transpositions, the algorithm for large values of λ had the best average approximation factor for values of λ at least 60. Different from the expected, the algorithm for large values of λ was not the best (in terms of approximation factor) for the problems when both operations were allowed. For these problems, the same algorithm is worse than the entropy-based one and the inversions-based one for almost all values of λ considered in our experiments.

For values of λ at most 20, the algorithms for small values of λ indeed were better than the algorithms for large values of λ. In the problem of Sorting Unsigned Permutations by λ-Reversals, the entropy-based algorithm tended to have



(a) Sorting Signed Permutations by λ-Reversals.



(b) Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions.

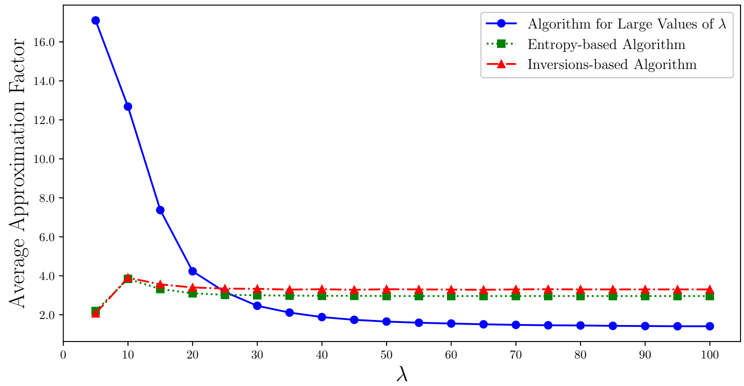
Figure 4: Average approximation factors of the algorithms for Sorting Signed Permutations by λ-Operations, with permutations of size 100.

a smaller approximation factor than the inversions-based one. Other than that, in general, the average approximation factor of the entropy-based and inversions-based algorithms behaved similarly in all problems addressed.

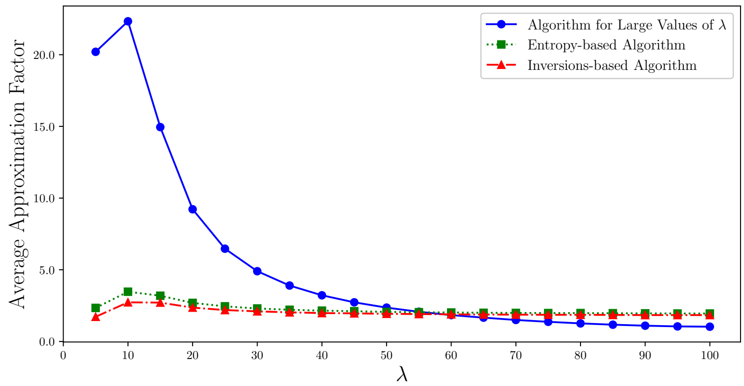
5 Conclusion

We have presented 15 approximation algorithms for the problems of sorting permutations by λ-reversals and/or λ-transpositions, being 3 algorithms for each problem we considered, where one of them works better when we have large values of λ and the other ones work better when we have small values of λ.

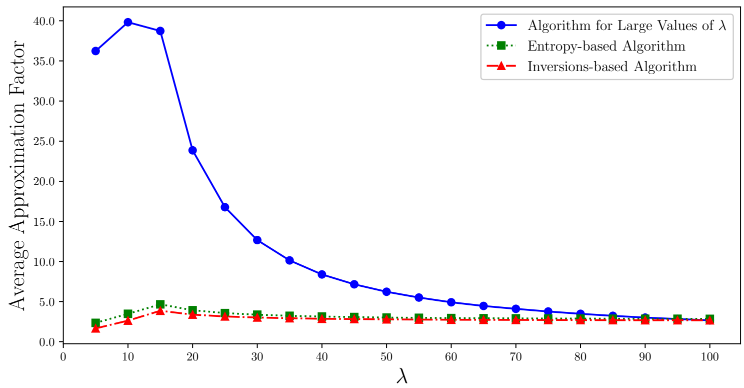
We also emphasize that the approximation factors of all algorithms we presented depend on λ and/or on the size of the permutation.



(a) Sorting Unsigned Permutations by λ-Reversals.



(b) Sorting Permutations by λ-Transpositions.



(c) Sorting Unsigned λ-Permutations by λ-Reversals and λ-Transpositions.

Figure 5: Average approximation factors of the algorithms for Sorting Unsigned Permutations by λ-Operations, with permutations of size 100.

Acknowledgments

This work was supported by the Brazilian Federal Agency for the Support and Evaluation of Graduate Education, Capes, the National Counsel of Technological and Scientific Development, CNPq (grants 400487/2016-0, 425340/2016-3, and 131182/2017-0), São Paulo Research Foundation, FAPESP (grants 2013/08293-7, 2015/11937-9, 2016/14132-4, and 2017/12646-3), and the program between the Brazilian Federal Agency for the Support and Evaluation of Graduate Education, CAPES, and the French Committee for the Evaluation of Academic and Scientific Cooperation with Brazil, COFECUB (grant 831/15).

References

[Bader et al., 2001] Bader, D. A., Moret, B. M. E., and Yan, M. (2001). A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8:483–491.

[Bafna and Pevzner, 1998] Bafna, V. and Pevzner, P. A. (1998). Sorting by Transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240.

[Berman et al., 2002] Berman, P., Hannenhalli, S., and Karpinski, M. (2002). 1.375-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany.

[Bulteau et al., 2012] Bulteau, L., Fertin, G., and Rusu, I. (2012). Sorting by Transpositions is Difficult. *SIAM Journal on Computing*, 26(3):1148–1180.

[Caprara, 1999] Caprara, A. (1999). Sorting Permutations by Reversals and Eulerian Cycle Decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110.

[Chan and Pătraşcu, 2010] Chan, T. M. and Pătraşcu, M. (2010). Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 161–173. Society for Industrial and Applied Mathematics.

[Chen, 2013] Chen, X. (2013). On Sorting Unsigned Permutations by Double-Cut-and-Joins. *Journal of Combinatorial Optimization*, 25(3):339–351.

[Dias and Meidanis, 2002] Dias, Z. and Meidanis, J. (2002). Sorting by Prefix Transpositions. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002)*, volume 2476 of *Lecture Notes in Computer Science*, pages 65–76. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany.

[Elias and Hartman, 2006] Elias, I. and Hartman, T. (2006). A 1.375-Approximation Algorithm for Sorting by Transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379.

[Galvão et al., 2015] Galvão, G. R., Lee, O., and Dias, Z. (2015). Sorting Signed Permutations by Short Operations. *Algorithms for Molecular Biology*, 10(1):1–17.

[Hannenhalli and Pevzner, 1999] Hannenhalli, S. and Pevzner, P. A. (1999). Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. *Journal of the ACM*, 46(1):1–27.

[Heath and Vergara, 2003] Heath, L. S. and Vergara, J. P. C. (2003). Sorting by Short Swaps. *Journal of Computational Biology*, 10(5):775–789.

[Jerrum, 1985] Jerrum, M. R. (1985). The Complexity of Finding Minimum-length Generator Sequences. *Theoretical Computer Science*, 36(2-3):265–289.

[Jiang et al., 2014] Jiang, H., Feng, H., and Zhu, D. (2014). An 5/4-Approximation Algorithm for Sorting Permutations by Short Block Moves. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC'2014)*, volume 8889 of *Lecture Notes in Computer Science*, pages 491–503. Springer International Publishing.

[Lefebvre et al., 2003] Lefebvre, J.-F., El-Mabrouk, N., Tillier, E. R. M., and Sankoff, D. (2003). Detection and validation of single gene inversions. *Bioinformatics*, 19(1):i190–i196.

[Lin and Jiang, 2004] Lin, G. and Jiang, T. (2004). A Further Improved Approximation Algorithm for Breakpoint Graph Decomposition. *Journal of Combinatorial Optimization*, 8(2):183–194.

[Lintzmayer et al., 2017] Lintzmayer, C. N., Fertin, G., and Dias, Z. (2017). Sorting Permutations by Prefix and Suffix Rearrangements. *Journal of Bioinformatics and Computational Biology*, 15(1):1750002.

[Miranda et al., 2018a] Miranda, G. H. S., Alexandrino, A. O., Lintzmayer, C. N., and Dias, Z. (2018a). Sorting λ-Permutations by λ-Operations. In *Proceedings of the 11th Brazilian Symposium on Bioinformatics (BSB'2018)*, pages 1–13. Springer International Publishing, Heidelberg, Germany.

[Miranda et al., 2018b] Miranda, G. H. S., Lintzmayer, C. N., and Dias, Z. (2018b). Sorting Permutations by Limited-Size Operations. In *Algorithms for Computational Biology*, volume 10849, pages 76–87. Springer International Publishing, Heidelberg, Germany.

[Rahman et al., 2008] Rahman, A., Shatabda, S., and Hasan, M. (2008). An Approximation Algorithm for Sorting by Reversals and Transpositions. *Journal of Discrete Algorithms*, 6(3):449–457.

[Tannier et al., 2007] Tannier, E., Bergeron, A., and Sagot, M.-F. (2007). Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888.

[Tesler, 2002] Tesler, G. (2002). GRIMM: Genome Rearrangements Web Server. *Bioinformatics*, 18(3):492–493.

[Vergara, 1998] Vergara, J. P. C. (1998). *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytechnic Institute and State University.

[Walter et al., 1998] Walter, M. E. M. T., Dias, Z., and Meidanis, J. (1998). Reversal and transposition distance of linear chromosomes. In *Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE'1998)*, pages 96–102, Los Alamitos, CA, USA. IEEE Computer Society.