algorithms

MDPI

Article

Approximation Algorithms for Sorting λ-Permutations by λ-Operations

Guilherme Henrique Santos Miranda [1], Alexsandro Oliveira Alexandrino [1,*], Carla Negri Lintzmayer [2] and Zanoni Dias [1]

[1] Institute of Computing, University of Campinas, Campinas SP 13083-970, Brazil; guilherme.miranda@students.ic.unicamp.br (G.H.S.M.); zanoni@ic.unicamp.br (Z.D.)
[2] Center for Mathematics, Computation and Cognition, Federal University of ABC, Santo André SP 09210-580, Brazil; carla.negri@ufabc.edu.br
* Correspondence: alexsandro@ic.unicamp.br

Abstract: Understanding how different two organisms are is one question addressed by the comparative genomics field. A well-accepted way to estimate the evolutionary distance between genomes of two organisms is finding the rearrangement distance, which is the smallest number of rearrangements needed to transform one genome into another. By representing genomes as permutations, one of them can be represented as the identity permutation, and, so, we reduce the problem of transforming one permutation into another to the problem of sorting a permutation using the minimum number of rearrangements. This work investigates the problems of sorting permutations using reversals and/or transpositions, with some additional restrictions of biological relevance. Given a value λ, the problem now is how to sort a λ-permutation, which is a permutation whose elements are less than λ positions away from their correct places (regarding the identity), by applying the minimum number of rearrangements. Each λ-rearrangement must have size, at most, λ, and, when applied to a λ-permutation, the result should also be a λ-permutation. We present algorithms with approximation factors of $O(\lambda^2)$, $O(\lambda)$, and $O(1)$ for the problems of Sorting λ-Permutations by λ-Reversals, by λ-Transpositions, and by both operations.

Keywords: genome rearrangements; approximation algorithms; sorting permutations

check for updates

Citation: Miranda, G.H.S.; Alexandrino, A.O.; Lintzmayer, C.N.; Dias, Z. Approximation Algorithms for Sorting λ-Permutations by λ-Operations. *Algorithms* **2021**, *14*, 175. https://doi.org/10.3390/a14060175

Academic Editor: Evangelos Kranakis

Received: 30 April 2021
Accepted: 31 May 2021
Published: 1 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

One challenge in biology is to understand how species evolve, considering that new organisms arise from mutations that occurred in others. Using the *principle of parsimony*, the minimum number of rearrangements that transform one genome into another, called *rearrangement distance*, is a widely adopted way to estimate the evolutionary distance between two genomes. A *genome rearrangement* is a global mutation that alters the order and the orientation of the genes in a genome.

Depending on the genomic information available and the problems considered, a genome can be modeled in different ways. Considering that a genome has no repeated genes, we can model it as a *permutation*, with each element representing a gene. Each gene has an orientation inside the genome. Gene orientation is represented by a plus or minus sign in each element of the permutation. In this case, we say that the permutation is signed. Due to mapping problems in the genome, orientation information could not be available. In this scenario, we use unsigned permutations to represent the genomes. By representing one of the genomes as the identity permutation, we reduce the problem of transforming one permutation into another to the problem of sorting a permutation with the minimum number of rearrangements, which is called sorting rearrangement distance or simply *distance*.

A *rearrangement model* $\mathcal{M}$ is the set of valid rearrangements used to calculate the distance. A *reversal* rearrangement inverts a segment of the genome, and a *transposition*

rearrangement swaps two adjacent segments of the genome. Genome rearrangements are also sometimes called operations.

The problems of Sorting Permutations by Transpositions and Sorting Unsigned Permutations by Reversals are NP-Hard [1,2]. The best known results for both problems are approximation algorithms with factor 1.375 [3,4]. Despite that, the problem of Sorting Signed Permutations by Reversals is solvable in polynomial time [5]. The variation in which the rearrangement model contains both (signed or unsigned) reversals and transpositions is NP-Hard [6]. For signed permutations, there are 2-approximation algorithms [7,8]. For unsigned permutations, the best approximation factor is $2k$ [7], where k is the approximation factor of an algorithm used for cycle decomposition of the breakpoint graph [9]. Given the best known value for k [9], this algorithm guarantees an approximation factor of $2.8334 + \epsilon$, where $\epsilon > 0$.

Extra restrictions can be added to these sorting problems, such as applying operations that only affect specific parts of a genome [10,11]. Other variants of the sorting problems emerged from the assumption that operations which affect large portions of a genome are less likely to occur [12]. Most of these variants add a constraint that limits the size of a valid operation [13–15]. The size of an operation is equal to the number of elements affected by it. Considering a size limit of 2, the problems of Sorting Unsigned Permutations by Reversals and/or Transpositions are solvable in polynomial time [13]. Considering a size limit of 3, the best known approximation factors for Sorting Unsigned Permutations by Reversals, by Transpositions, and by Reversals and Transpositions are 2 [13], 5/4 [14], and 2 [15], respectively. For Sorting Signed Permutations by Reversals and by Reversals and Transpositions, the best known approximation factors are 5 [16] and 3 [16], respectively. Recently, Zhang et al. [17] presented polynomial algorithms to decide if a permutation can be sorted using a number of short operations equal to the lower bound for the distance, which are based on the number of inversions and entropy of the permutation (these concepts are described in Sections 3 and 4).

The problem of Sorting (Unsigned or Signed) Permutations by λ-Operations is a generalization of the size limited variants in which an operation is valid if its size is less than or equal to λ. There are $O(\lambda^2)$-approximation algorithms for Sorting Unsigned Permutations by λ-Reversals, by λ-Transpositions, and by λ-Reversals and λ-Transpositions [18]. The study of λ-operations is motivated by the observation that the rearrangements occurred in some species do not act on very large segments of the genome [12,19]. Using size limited operations makes more sense when one knows that the elements are not so far away from their original positions, so we introduce the study of Sorting λ-Permutations by λ-Operations. A permutation π is a *λ-permutation* if all elements of π are at a distance less than λ from their correct positions considering the identity permutation.

We consider the problems of sorting (unsigned or signed) λ-permutations by λ-reversals, by λ-transpositions, and by λ-reversals and λ-transpositions. Some of the results for the problems of sorting unsigned λ-permutations were previously presented by us [20], but, to the best of our knowledge, there were no results for the sorting signed λ-permutations problems.

The paper is organized as follows. In Section 2, we present all concepts used in this paper and an NP-hardness proof for the sorting λ-permutations by λ-operations problems, considering four rearrangement models. In Sections 3 and 4, we present $O(\lambda^2)$-approximation algorithms for the sorting unsigned and signed λ-permutation problems, respectively. In Section 5, we present algorithms with approximation factors of $O(\lambda)$ and $O(1)$ for Sorting λ-Permutations by λ-Reversals and the other problems addressed, respectively. In Section 6, we show experimental results which compare the algorithms presented in Sections 3–5. We conclude the paper in Section 7.

2. Preliminaries and Basic Facts

We denote a signed permutation by $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, where $\pi_i \in \{-n, -(n-1), \ldots, -2, -1, +1, +2, \ldots, +(n-1), +n\}$ and $|\pi_i| \neq |\pi_j|$, for all $1 \leq i < j \leq n$. We denote an unsigned permutation similarly but neglect the signs of the elements.

We assume that there are two extra elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$ in π, but, for convenience, they are omitted from the permutation's representation. Given an integer $\lambda \geq 2$ as input, we say that π is a *λ-permutation* if we have $||\pi_i| - i| < \lambda$ for all $1 \leq i \leq n$. We define the size of a permutation as the number of elements in it, without counting π_0 and π_{n+1}.

Given two permutations π and σ, the *composition* of π and σ is equal to $\pi \cdot \sigma = (\alpha_1\ \alpha_2\ \ldots\ \alpha_n)$, such that $\alpha_i = -\pi_{|\sigma_i|}$, if $\sigma_i < 0$, and $\alpha_i = \pi_{|\sigma_i|}$, otherwise. The inverse permutation of π, denoted by π^{-1}, is the permutation such that $\pi \cdot \pi^{-1} = (1\ 2\ \ldots\ n)$. For example, given $\pi = (+2\ +4\ -5\ -1\ +3)$, we have that $\pi^{-1} = (-4\ +1\ +5\ +2\ -3)$.

An *unsigned reversal* is denoted by $\rho(i,j)$, with $1 \leq i < j \leq n$, and it transforms an unsigned permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ into $\pi \cdot \rho(i,j) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_i}\ \pi_{j+1}\ \ldots\ \pi_n)$. A *signed reversal* is denoted by $\bar{\rho}(i,j)$, with $1 \leq i \leq j \leq n$, and when applied on a signed permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, the resulting permutation is $\pi \cdot \bar{\rho}(i,j) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{-\pi_j\ \ldots\ -\pi_i}\ \pi_{j+1}\ \ldots\ \pi_{n-1}\ \pi_n)$. The size of a (signed or unsigned) reversal is given by $j-i+1$. For example, given the signed permutation $\pi = (-1\ -4\ +3\ -2\ +5)$, we have $\pi \cdot \bar{\rho}(2,4) = (-1\ +2\ -3\ +4\ +5)$, and the size of such operation is $4-2+1=3$. We say that $\rho(i,j)$ or $\rho(\bar{i},j)$ is a *λ-reversal* if $j-i+1 \leq \lambda$.

An operation of *transposition* is denoted by $\tau(i,j,k)$, with $1 \leq i < j < k \leq n+1$, and when applied on a (unsigned or signed) permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, the result is the permutation $\pi \cdot \tau(i,j,k) = (\pi_1\ \pi_2\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$. The size of a transposition is given by $k-i$. For example, given the unsigned permutation $\pi = (4\ 5\ 6\ 1\ 2\ 3)$, we have $\pi \cdot \tau(1,4,7) = (1\ 2\ 3\ 4\ 5\ 6)$ and the size of such operation is $7-1=6$. We say that $\tau(i,j,k)$ is a *λ-transposition* if $k-i \leq \lambda$.

The goal of these problems is to transform a λ-permutation π into the *identity permutation* $\iota = (1\ 2\ \ldots\ n)$ by applying the minimum number of λ-operations, which defines the *sorting distance*, such that each permutation generated during the process is also a λ-permutation.

We denote by $d_r^\lambda(\pi)$, $d_t^\lambda(\pi)$, and $d_{rt}^\lambda(\pi)$ the sorting distance for the sorting unsigned λ-permutation problems when we only have λ-reversals, λ-transpositions, and when we have both operations, respectively. Similarly, we denote by $d_{\bar{r}}^\lambda(\pi)$ and $d_{\bar{r}t}^\lambda(\pi)$ the sorting distance for the sorting signed λ-permutation problems when we only have λ-reversals and when we have both operations, respectively.

2.1. Complexity of Sorting λ-Permutations by λ-Operations

Next, we show an NP-Hardness proof for the sorting distance problem considering some rearrangement models. For this purpose, we define the following decision problems:

- Sorting λ-Permutations by λ-Operations: given a positive integer λ, a rearrangement model $\mathcal{M}$ containing only λ-operations, a λ-permutation π, and a non-negative integer k, decide if it is possible to sort π using, at most, k operations that belong to $\mathcal{M}$.
- Sorting Permutations by Rearrangements: given a rearrangement model $\mathcal{M}$, a permutation π, and a non-negative integer k, decide if it possible to sort π using, at most, k operations that belong to $\mathcal{M}$.

When $\lambda = n$, the problems of Sorting λ-Permutations by λ-Operations are NP-Hard (except for signed reversals), since they are equivalent to the sorting problems without restrictions in the size of operations. Next, we show that the problem is NP-Hard for a wide range of λ values. When the rearrangement model is clear from the context, it is omitted from the instance.

Theorem 1. *The problem of Sorting λ-Permutations by λ-Operations is NP-Hard when $\lambda = cn^d$, for positive constants c and $0 < d \leq 1$, considering unsigned reversals, transpositions, transpositions and unsigned reversals, and transpositions and signed reversals.*

Proof. Consider the model containing only unsigned reversals. The problem of Sorting Permutations by Unsigned Reversals is NP-Hard [2]. Now, we show a reduction from this problem to the problem of Sorting λ-Permutations by Unsigned λ-Reversals when $\lambda = cn^d$, for positive constants c and $0 < d \leq 1$.

Given an instance (π, k), where π has size n, for Sorting Permutations by Unsigned Reversals, we create an instance $(\lambda, \mathcal{M}, \pi', k)$, where $\lambda = n$, $\mathcal{M}$ is a model with unsigned λ-reversals, and $\pi' = (\pi_1\ \pi_2\ \ldots\ \pi_n\ (n+1)\ \ldots\ n')$, such that $\lambda = n = cn'^d$, which is a λ-permutation. We note that, since c and d are constants, this is a polynomial-time reduction.

Now, we show that the instance (π, k) is satisfied for Sorting Permutations by Unsigned Reversals iff $d_r^\lambda(\pi') \leq k$.

($\Rightarrow$) Suppose that there exists a sorting sequence S of reversals for π with k reversals or less. Note that π has size n, and all reversals in S are λ-reversals, since $\lambda = n$. The sequence S can sort the λ-permutation π'; therefore, $d_r^\lambda(\pi') \leq k$.

($\Leftarrow$) Suppose that there exists a sorting sequence S' of λ-reversals for π' with k operations or less. We now construct a sorting sequence S for π with k or less reversals. If all operations of S' affect only elements between the positions 1 and n of π', then $S = S'$ is also a sorting sequence for π. Otherwise, there exists operations in S' which affect the elements in the interval $[n+1, n']$. Let $S' = \rho'_1, \rho'_2, \ldots, \rho'_\ell$, where ℓ is the length of S'. We construct $S = \rho_1, \rho_2, \ldots, \rho_\ell$, where ρ_i starts at the first element of ρ'_i, from left to right, which is in π, and it ends at the last element of ρ'_i which is in π. Since S inverts the elements of π in the same order as they are inverted in π' by S', this sequence sorts π. Note that the length of S is less than or equal to ℓ because S' may have reversals which do not affect elements that are in π, which corresponds to an empty reversal in S that can be ignored.

The proof is analogous for the other three models. □

2.2. Inversions

An *inversion* is defined as a pair of elements (π_i, π_j) such that $i < j$ and $|\pi_i| > |\pi_j|$. The number of inversions in π is denoted by $\mathrm{Inv}(\pi)$.

For example, the permutation $\pi = (1\ 5\ 4\ 3\ 2)$ has 6 inversions. The 3-reversal $\rho(3,5)$ transforms π into $\pi' = (1\ 5\ 2\ 3\ 4)$ that has 3 inversions.

Lemma 1. *For all λ-permutations $\pi \neq \iota$ and all $\lambda \geq 2$, we have $d_r^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda(\lambda-1)/2}$, $d_t^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda^2/4}$, and $d_{rt}^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda(\lambda-1)/2}$.*

Proof. A λ-reversal of length k can only change the inversions between elements of the segment affected by it. Therefore, the maximum number of inversions that a reversal of length k can remove is equal to $\binom{k}{2}$, since this is the maximum number of inversions between the elements of a segment of length k. By definition, the length of a λ-reversal is less than or equal to λ, so a λ-reversal can remove, at most, $\binom{\lambda}{2}$ inversions. Since the identity permutation is the only one with $\mathrm{Inv}(\pi) = 0$, any sorting sequence has to remove all inversions of the permutation, which results in the lower bound $d_r^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda(\lambda-1)/2}$.

A λ-transposition $\tau(i,j,k)$ of length k' also affects only the inversions between elements affected by it. Moreover, the inversions between elements on the segment $\pi_i, \ldots, \pi_{j-1}$ are not changed. The same is valid for inversions between elements of the segment $\pi_j, \ldots, \pi_{k-1}$. So, if an inversion (π_x, π_y), with $x < y$, is removed, then π_x is in $\pi_i, \ldots, \pi_{j-1}$ and π_y is in $\pi_j, \ldots, \pi_{k-1}$. Therefore, the maximum number of inversions that a transposition can remove is equal to $(j-i)(k-j) \leq (k'/2)^2$, since $(j-i) + (k-j) = k'$. This bound is maximum when $k' = \lambda$, which results in $\lambda^2/4$. As before, this results in the lower bound $d_t^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda^2/4}$.

Using a similar argument, we have $d^{\lambda}_{rt}(\pi) \geq \frac{\text{Inv}(\pi)}{\lambda(\lambda-1)/2}$. □

2.3. Breakpoints

An *unsigned reversal breakpoint* is defined as a pair of elements (π_i, π_{i+1}) such that $|\pi_{i+1} - \pi_i| \neq 1$, for all $0 \leq i \leq n$, in the problems of Sorting Unsigned λ-Permutations by λ-Reversals and Sorting Unsigned λ-Permutations by λ-Reversals and λ-Transpositions. A *transposition breakpoint* or *signed reversal breakpoint* is defined as a pair of elements (π_i, π_{i+1}) such that $\pi_{i+1} - \pi_i \neq 1$, for all $0 \leq i \leq n$, in the problems of Sorting Unsigned λ-Permutations by Transpositions, Sorting Signed λ-Permutations by λ-Reversals, and Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions. Considering a specific type of breakpoint, the number of breakpoints in π is denoted by $b(\pi)$.

Fact 1. *[21,22] For all λ-permutations $\pi \neq \iota$ and all $\lambda \geq 2$, we have $d^{\lambda}_{r}(\pi) \geq \frac{b(\pi)}{2}$, $d^{\lambda}_{t}(\pi) \geq \frac{b(\pi)}{3}$, $d^{\lambda}_{rt}(\pi) \geq \frac{b(\pi)}{3}$, $d^{\lambda}_{\bar{r}}(\pi) \geq \frac{b(\pi)}{2}$, and $d^{\lambda}_{\bar{r}t}(\pi) \geq \frac{b(\pi)}{3}$.*

2.4. Strips

A maximal subsequence $(\pi_i\ \pi_{i+1}\ \ldots\ \pi_j)$ without any breakpoints (π_k, π_{k+1}), for all $i \leq k < j$, is called a *strip*.

For the problems of sorting unsigned λ-permutations, if the strip's elements are in ascending (resp. descending) order, then we call it an *increasing* (resp. *decreasing*) strip. Strips containing only one element are considered to be increasing ones. For example, considering sorting unsigned λ-permutations by both operations and $\pi = (6\,4\,5\,3\,2\,1)$, we have two increasing strips (6) and $(4\,5)$, and a decreasing strip $(3\,2\,1)$. Note, however, that segment $(3\,2\,1)$ is not a decreasing strip for the problem of Sorting λ-Permutations by λ-Transpositions. This is because, direct from the definition of strips and breakpoints, there are no decreasing strips when this problem is being considered.

For the problems of sorting signed λ-permutations, if the strip's elements are positive, then we call it an *increasing* strip. Otherwise, the strip is called a *decreasing* strip. Strips containing only one element are considered to be increasing ones if such an element is positive. Otherwise, they are considered to be decreasing strips. For example, considering sorting signed λ-permutations and $\pi = (+6\ -5\ -4\ +1\ +2\ -3)$, we have two increasing strips, $(+6)$ and $(+1\ +2)$, and two decreasing strips, $(-5\ -4)$ and (-3). Note that, despite the elements of strip $(-5\ -4)$ are in ascending order, it still is a decreasing strip, according to our definition.

The number of elements in a strip S of a λ-permutation π is denoted by $|S|$. We say that an element π_i is out-of-place if $|\pi_i| \neq i$.

Lemma 2. *Let π be a λ-permutation, and let $|\pi_j| = i$ be the smallest element out-of-place in π (i.e., $|\pi_j|$ is minimum and $\pi_j \neq j$). The strip S that contains π_j is such that $|S| \leq \lambda - 1$.*

Proof. First, suppose we have the increasing strip $S = (\pi_j\ \ldots\ \pi_k)$. Let $R = (\pi_i\ \ldots\ \pi_{j-1})$ be the segment of elements to the left of S. Note that the absolute value of any element in R is greater than any element in S. Therefore, $|\pi_i| \geq |\pi_k| + 1 = |\pi_j| + k - j + 1 = i + |S|$, and then $|S| \leq |\pi_i| - i < \lambda$.

When we have π_j in a decreasing strip, the proof is analogous. □

3. Inversions-Based Approximation Algorithms for Unsigned Permutations

In this section, we present approximation algorithms based on the concept of inversions for the sorting unsigned λ-permutations problems we are addressing.

The next lemma shows that it is always possible to remove at least one inversion from a λ-permutation by applying one λ-operation which results in a λ-permutation.

Lemma 3. *For any λ-permutation $\pi \neq \iota$, we can apply a 2-reversal or a 2-transposition to obtain a λ-permutation with $\mathrm{Inv}(\pi) - 1$ inversions.*

Proof. Let $\pi_j = i$ be the smallest element out-of-place in π. Initially, note we have inversion (π_{j-1}, π_j), since $\pi_{j-1} > \pi_j$ and $j - 1 < j$. Let σ be a 2-operation that swaps elements π_{j-1} and π_j, and let $\pi' = \pi \cdot \sigma$. It is easy to see that $\mathrm{Inv}(\pi') = \mathrm{Inv}(\pi) - 1$, since such inversion was removed, and note that there always exists a λ-reversal or a λ-transposition equivalent to σ, because the elements are adjacent, and, so, both only swap two elements.

Observe that, in π', element π_j is closer to its correct position, since it was moved to the left. Hence, we follow by showing that π' is a λ-permutation by considering two cases according to the values of $\pi_{j-1} = \pi'_j$.

If $\pi_{j-1} \geq j$, then π_{j-1} is also closer to its correct position, in π'. Otherwise, $\pi_{j-1} < j$. Thus, element π_{j-1} will be, in π', one position away from its correct position. Then, note that $|j - \pi_j| + 1 = (j - i) + 1 \leq \lambda$ because π is a λ-permutation. In addition, observe that we have $|j - \pi'_j| = j - \pi'_j$ because $\pi'_j = \pi_{j-1} < j$, and $j - \pi'_j < j - i$ because $\pi'_j > i$, and, so, $j - i \leq \lambda - 1$. Therefore, π' is a λ-permutation, and the result follows. □

A generic greedy approximation algorithm for the three problems we are addressing in this section is presented in the next theorem. It receives an integer $\lambda \geq 2$ and a λ-permutation $\pi \neq \iota$ as input. It is greedy because it always tries to decrease the largest amount of inversions in π. Since an unsorted permutation always contains inversions, the algorithm will finally sort π.

Theorem 2. *There exist $O(\lambda^2)$-approximation algorithms for the problems of Sorting Unsigned λ-Permutations by λ-Reversals, by λ-Transpositions, and by λ-Reversals and λ-Transpositions.*

Proof. Let $\lambda \geq 2$ be an integer, and let $\pi \neq \iota$ be a λ-permutation. Consider an algorithm which chooses a λ-operation σ, such that $\pi \cdot \sigma$ is a λ-permutation and $\mathrm{Inv}(\pi \cdot \sigma)$ is as small as possible, and then it applies such operation over π. The algorithm repeats the same process over the resulting permutation until it reaches the identity permutation.

In the worst case, we always have one λ-operation reducing the number of inversions by one unit, as shown in Lemma 3. Therefore, the number of operations of such greedy algorithm is, at most, $\mathrm{Inv}(\pi)$, and the approximation factor follows immediately from Lemma 1. □

Note that the distance is $O(n^2)$ because any unsigned permutation can be sorted with $O(n^2)$ λ-reversals or λ-transpositions. For Sorting Unsigned Permutations by λ-Reversals, at each step the algorithm considers $O(\lambda^2)$ possible reversals that can be chosen. Since the variation in the number of inversions caused by an operation can be calculated in $O(\lambda\sqrt{\log \lambda})$ time [23], the algorithm has total time complexity $O(n^2\lambda^3\sqrt{\log \lambda})$. Using the same analysis, we conclude that the algorithms involving transpositions have total time complexity $O(n^2\lambda^4\sqrt{\log \lambda})$.

4. Algorithms Based on Inversions and Entropy for Signed λ-Permutations

The *entropy* of an element π_i from a permutation π is given by $\mathrm{ent}(\pi_i) = ||\pi_i| - i|$, that is, the distance between π_i and its position in ι. We denote by $E_\pi^{even^-}$ the set of negative elements in π such that $\mathrm{ent}(\pi_i)$ is even, for all $1 \leq i \leq n$. Similarly, we denote by $E_\pi^{odd^+}$ the set of positive elements in π such that $\mathrm{ent}(\pi_i)$ is odd. For example, given $\pi = (-4\ +3\ -1\ -2\ +5)$, we have $E_\pi^{even^-} = \{-1, -2\}$ and $E_\pi^{odd^+} = \{+3\}$, since -1 and -2 are negative elements with even entropy $(\mathrm{ent}(-1) = \mathrm{ent}(-2) = 2)$, and $+3$ is a positive element with odd entropy $(\mathrm{ent}(+3) = 1)$, respectively.

Fact 2. *[16] Let π be a signed permutation, and let σ be a 2-reversal. We have $|E_\pi^{even^-}| + |E_\pi^{odd^+}| = |E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|$.*

Fact 3. *[24] Let π be a signed permutation, and let σ be a λ-operation. We have* $(|E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|) - (|E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|) \leq \lambda$.

The score of a λ-operation σ over a signed permutation π is given by $\text{score}(\pi,\sigma) = (\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|) - (|\text{Inv}(\pi\cdot\sigma)| + |E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$.

Combining Fact 3 with the fact that a λ-reversal and a λ-transposition can remove, at most, $\frac{\lambda(\lambda-1)}{2}$ inversions, we have an upper bound for the score of a permutation in Lemma 4. This upper bound implies Corollary 1.

Lemma 4. *Let π be a signed permutation. We have* $\text{score}(\pi,\sigma) \leq \lambda(\lambda-1)/2 + \lambda$.

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

We follow by showing in Lemma 5 that there always exists a λ-operation with score at least 1. Having this in mind, a generic greedy approximation algorithm for the two sorting signed permutations problems we are addressing is presented in next theorem. It receives an integer $\lambda \geq 2$ and a λ-permutation $\pi \neq \iota$ as input. It is greedy because it always chooses a λ-operation with the largest score. Since the only permutation with $\text{Inv}(\pi) = |E_{\pi}^{even^-}| = |E_{\pi}^{odd^+}| = 0$ is the identity, it will, eventually, sort π.

Lemma 5. *For any signed permutation $\pi \neq \iota$ and $\lambda \geq 2$, there always exists a λ-operation σ such that $\pi\cdot\sigma$ is a λ-permutation and* $\text{score}(\pi,\sigma) \geq 1$.

Proof. The proof is divided into two cases, according to $\text{Inv}(\pi)$.

First, consider $\text{Inv}(\pi) = 0$. Note that, in this case, we only have zero entropy elements in π, implying that $|E_{\pi}^{odd^+}| = 0$. So, by applying an unitary λ-reversal σ over a negative element, we get a λ-permutation $\pi\cdot\sigma$ such that $|E_{\pi\cdot\sigma}^{even^-}| = |E_{\pi}^{even^-}| - 1$. Since such an operation holds $\text{Inv}(\pi\cdot\sigma) = |E_{\pi\cdot\sigma}^{odd^+}| = 0$, we have $\text{score}(\pi,\sigma) = 1$.

For $\text{Inv}(\pi) > 0$, Lemma 3 shows how to decrease the amount of inversions by applying one 2-reversal σ and, since Fact 2 shows that $(|E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|) = (|E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$, we have $\text{score}(\pi,\sigma) = 1$.

Note that, in both cases, the resulting permutation is also a λ-permutation. □

Theorem 3. *There exist $(\frac{\lambda(\lambda-1)}{2} + \lambda)$-approximation algorithms for the problems of Sorting Signed λ-Permutations by λ-Reversals and Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions.*

Proof. Let $\lambda \geq 2$ be an integer, and let $\pi \neq \iota$ be a λ-permutation. Consider an algorithm which chooses the λ-operation σ such that $\pi\cdot\sigma$ is a λ-permutation and $\text{score}(\pi\cdot\sigma)$ is as great as possible and then it applies such operation over π. The algorithm repeats the same process in the resulting permutation until it reaches the identity permutation.

In the worst case, as shown in Lemma 5, we always have one λ-reversal with score 1. Therefore, the number of operations of such greedy algorithm is, at most, $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$, and the approximation factor follows immediately from Corollary 1. □

Since $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ is $O(n^2)$ and the time to calculate the variation in the number of $|E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ is $O(\lambda)$, the analysis of time complexity is analogous to the algorithms previously presented for the sorting unsigned permutation problems.

5. Breakpoints-Based Approximation Algorithms

In this section, we present approximation algorithms based on the concept of breakpoints for the five problems we are addressing.

In the next lemma, we suppose that the smallest element out-of-place is in an increasing strip of a λ-permutation $\pi \neq \iota$. We show how to reduce the number of breakpoints of π by moving this strip to its correct position using an operation that may not be a λ -operation, but assuring that the resulting permutation is also a λ-permutation.

Lemma 6. *Let π be a λ-permutation. Let $|\pi_j| = i$ be the smallest element out-of-place in π. Suppose that π_j is in an increasing strip $S = (\pi_j \ldots \pi_k)$. Then, $\pi \cdot \tau(i, j, k+1)$ is a λ-permutation, $b(\pi \cdot \tau(i, j, k+1)) \leq b(\pi) - 1$, and $(k + 1 - i) \leq 2(\lambda - 1)$.*

Proof. Let $R = (\pi_i \ldots \pi_{j-1})$ be the segment of elements in π that will be transposed with S. Observe that the absolute value of any element in R is greater than any element in S, so $\pi \cdot \tau(i, j, k+1)$ is a λ-permutation, since greater elements (in their absolute values) are moved to the right and smaller elements to the left. In addition, observe that, in π, we have the three breakpoints (π_{i-1}, π_i), (π_j, π_{j+1}), and (π_{k-1}, π_k), where the first one is because $\pi_{i-1} = \pi_j - 1 = i - 1$ and $|\pi_i| > i = \pi_j$, and the second and third ones are because the strip's start and strip's end are at positions j and k, respectively. Transposition $\tau(i, j, k+1)$ moves the elements of S to their correct positions by transposing them with elements of R, thus removing at least breakpoint (π_{i-1}, π_i). Since a transposition can add, at most, three breakpoints, but we already had all of them and we removed at least (π_{i-1}, π_i), we have $b(\pi \cdot \tau(i, j, k+1)) \leq b(\pi) - 1$.

By Lemma 2, we have $|S| \leq \lambda - 1$; thus, $k + 1 - j \leq \lambda - 1$. Since π is a λ-permutation, we have $||\pi_j| - j| \leq \lambda - 1$, and, by construction, $\pi_j = i$; thus, $|i - j| + 1 = j - i + 1 \leq \lambda - 1$. Therefore, $k + 1 - i \leq 2(\lambda - 1)$. □

For a λ-permutation π with only increasing strips, the next lemma shows that it is possible to find a sequence with, at most, 4 transpositions that decreases the number of breakpoints in π, assuring that all permutations generated in the process are λ-permutations. Using these operations, we can create an algorithm that sorts π using, at most, $4b(\pi)$ λ-transpositions.

Lemma 7. *Let π be a λ-permutation. Let $|\pi_j| = i$ be the smallest element out-of-place in π. Suppose that π_j is in an increasing strip $S = (\pi_j \ldots \pi_k)$. It is always possible to obtain a λ-permutation $\pi \cdot \tau(i, j, k+1)$ with, at most, $b(\pi) - 1$ breakpoints by applying, at most, 4 λ-transpositions such that all intermediary permutations are λ-permutations.*

Proof. Let $R = (\pi_i \ldots \pi_{j-1})$ be the segment that will be moved to the right in $\tau(i, j, k+1)$. Note that $|S| \leq \lambda - 1$, by Lemma 2, and $|R| \leq \lambda - 1$, because π is a λ-permutation.

The idea is to apply a sequence with, at most, four λ-transpositions that divide both segments $R = (\pi_i \ldots \pi_{j-1})$ and $S = (\pi_j \ldots \pi_k)$ into, at most, two parts each, where each part has, at most, $\lfloor \lambda/2 \rfloor$ elements, and then exchange each part of S, at most, twice (and at least once), with the (possible) two parts of R. If we had exactly $\lambda - 1$ elements in each of S and R, such sequence would be $\tau(i + \lfloor \lambda/2 \rfloor, j, j + \lfloor \lambda/2 \rfloor), \tau(i, i + \lfloor \lambda/2 \rfloor, j), \tau(j, j + \lfloor \lambda/2 \rfloor, k+1), \tau(i + \lfloor \lambda/2 \rfloor, j, j + \lfloor \lambda/2 \rfloor)$.

Now, we have to show that, after each of the, at most, four operations is applied, we have a λ-permutation as result.

Observe that the absolute value of any element in R is greater than any element in S. Since each λ-transposition puts elements of S closer to their correct positions by transposing them with greater elements (in their absolute values) of R, we have a λ-permutation after each λ-operation applied. After all λ-transpositions are applied, the elements of S are at positions from i to $i + k - j$, and the elements of R are at positions from $i + k - j + 1$ to k, resulting in $\pi \cdot \tau(i, j, k+1)$, which is a λ-permutation with at least $b(\pi) - 1$ breakpoints, as shown in Lemma 6. □

For a λ-permutation π with only increasing strips, the next lemma shows how to decrease the number of breakpoints in π using only λ-reversals, also assuring that all permutations generated in the process are λ-permutations.

Lemma 8. *Let π be a λ-permutation. Let $|\pi_j| = i$ be the smallest element out-of-place in π. Suppose that π only has increasing strips and that π_j is in a strip $S = (\pi_j \ \dots \ \pi_k)$. It is always possible to obtain a λ-permutation $\pi \cdot \tau(i, j, k+1)$ with, at most, $b(\pi) - 1$ breakpoints by applying, at most, $5 + \lambda - 1$ λ-reversals such that all intermediary permutations are λ-permutations.*

Proof. Let $R = (\pi_i \ \dots \ \pi_{j-1})$ be the segment that will be moved to the right in $\tau(i, j, k+1)$. Note that $|S| \leq \lambda - 1$, by Lemma 2, and $|R| \leq \lambda - 1$, because π is a λ-permutation.

The idea is to move elements from S to their correct positions by applying, at most, two sequences of pairs of λ-reversals, where each one puts, at most, $\lfloor \lambda/2 \rfloor$ elements of S in their correct positions at a time.

In the first sequence of λ-reversals, there are two possibilities. If $|S| \leq \lfloor \lambda/2 \rfloor$, then the first operation of each pair reverses $|S|$ elements contained in both S and R. If $|S| > \lfloor \lambda/2 \rfloor$, then it reverses $\lfloor \lambda/2 \rfloor$ elements contained in both S and R. In any case, the second operation of each pair reverses back the elements of R affected by the first one, in order to leave π with only increasing strips again (except for the elements of S which were affected by the first operation).

After the sequence is applied, we have, at most, $\lfloor \lambda/2 \rfloor$ elements of S from positions i to $i + \min(\lfloor \lambda/2 \rfloor, |S|)$, and, maybe, they are in a decreasing strip. If this is the case, then one more λ-reversal has to be applied to put these elements in their correct places, by reversing such decreasing strip.

The second sequence of λ-reversals puts the (at most) $\lfloor \lambda/2 \rfloor$ remaining elements of S in their correct positions, following the same idea, and, also, maybe one extra λ-reversal will be necessary after it is applied. Note that, if there are no remaining elements (in case of $|S| \leq \lfloor \lambda/2 \rfloor$), this sequence is not necessary.

The largest amount of operations needed in the process described above happens when we have $\lfloor \lambda/2 \rfloor + 1$ elements in S. Since $|S| > \lfloor \lambda/2 \rfloor$, our process starts by moving the first $\lfloor \lambda/2 \rfloor$ elements of S to their correct positions. Observe that, for each pair of reversals applied in the first sequence, one of them moves $\lfloor \lambda/2 \rfloor$ elements $\lfloor \lambda/2 \rfloor$ positions to the left (except, maybe, by the last pair), and then the other one reverses again the elements of R affected by the first reversal. Besides that, by the definition of λ-permutations, the elements of S are, at most, $\lambda - 1$ positions away from their correct positions, and, so, at most, 2 pairs of reversals are needed to put them at position i. As we have told before, maybe such elements of S ended up in reversed order and, in order to fix it, one more reversal is needed. Hence, we already have a total of, at most, 5 reversals applied.

To move the remaining element of S to its correct position, all the λ-reversals of the second sequence will have size 2 (note that, in this case, we do not need the second operation of each pair), which means such element will be moved only 1 position to the left per operation, giving an extra amount of $\lambda - 1$ λ-reversals. Therefore, the number of λ-reversals to move S to its correct position is, at most, $5 + \lambda - 1$.

Now, we have to show that, after applying each operation, we have a λ-permutation as result and, after the last operation is applied, we have $\pi \cdot \tau(i, j, k+1)$.

Observe that any element in R is greater than any element in S. Then, since the first operation of each pair moves elements of R to the right and elements of S to the left, all elements affected will be closer to their correct positions, resulting in a λ-permutation. The second operation of each pair reverses elements of R to ascending order again, so it also results in a λ-permutation. After both sequences of λ-reversals are applied, all elements of S are at positions from i to $i + k - j$ and all elements of R are at positions from $i + k - 1$ to k, resulting in $\pi \cdot \tau(i, j, k+1)$, which is a λ-permutation with at least one less breakpoint than π, as shown in Lemma 6. □

Lemma 9 shows how to decrease the number of breakpoints of a λ-permutation π, considering that the smallest element out-of-place of π is in a decreasing strip. Since Lemma 8 deals with the case where π has no decreasing strips, we can use these two lemmas together to develop an algorithm to sort a λ-permutation π using only λ-reversals.

Lemma 9. *Let $|\pi_k| = i$ be the smallest element out-of-place in a λ-permutation π. Suppose that π_k is in a decreasing strip $S = (\pi_j \dots \pi_k)$. It is always possible to obtain a λ-permutation with, at most, $b(\pi) - 1$ breakpoints by applying, at most, one λ-transposition and one λ-reversal.*

Proof. When $j = i$, one reversal $\rho(j, k)$ put elements of S in their correct positions. It is easy to see that $\pi \cdot \rho(j, k)$ is a λ-permutation and, since $|S| = k - j + 1 \leq \lambda - 1$ by Lemma 2, we have that $\rho(j, k)$ is a λ-reversal.

Now, assume $j > i$. Note that, in this case, we have the three breakpoints (π_{i-1}, π_i), (π_j, π_{j+1}), and (π_{k-1}, π_k), where the first one is because $\pi_{i-1} = |\pi_k| - 1 = i - 1$ and $|\pi_i| > i = |\pi_k|$, and the second and third ones are because the strip's start and strip's end are at positions k and j, respectively. Thus, we can apply the λ-transposition $\tau(i, j, k+1)$ followed by the λ-reversal $\rho(i, i + (k - j))$ to obtain $b(\pi \cdot \tau(i, j, k+1) \cdot \rho(i, i + (k - j))) \leq b(\pi) - 1$, since a λ-transposition can add, at most, three breakpoints but we already had (π_{i-1}, π_i), (π_{j-1}, π_j), and (π_k, π_{k+1}), and the second λ-reversal can add, at most, two breakpoints, but we already had (π_{i-1}, π_j) and (π_k, π_i) and we removed the first one, since all elements of S will be in their correct positions in $\pi \cdot \tau(i, j, k+1) \cdot \rho(i, i + (k - j))$.

Now, we have to show that, after each operation is applied, we have a λ-permutation as result.

Let $R = (\pi_i \dots \pi_{j-1})$ be the segment of elements that should be moved in order to put S in its correct position. Observe that the absolute value of any element in R is greater than any element in S. The first operation, a λ-transposition, transposes S only with greater elements (in their absolute values); thus, the result is a λ-permutation. The second operation, a λ-reversal, just reverses a decreasing strip to put the elements of S in their correct positions; thus, it also results in a λ-permutation. Hence, we have as result a λ-permutation with at least one less breakpoint. □

The next theorems describe approximation algorithms for the problems we are addressing. Lemma 10 is auxiliary to Theorem 4. The algorithms receive an integer $\lambda \geq 2$ and a λ-permutation $\pi \neq \iota$ as input. The goal is to decrease at least one unit on the number of breakpoints in π by moving elements to their correct positions (applying Lemmas 7 and 9). Since the only permutation with no breakpoints is the identity, they will, eventually, sort π.

Lemma 10. *Let π be a λ-permutation. Let $S = (j \dots i)$ be a decreasing strip in π (thus, $i < j$). Let $\pi' = \pi \cdot \rho(\pi^{-1}_{|j|}, \pi^{-1}_{|i|})$ be the resulting permutation after reversing S in π. Then, π' is a λ-permutation.*

Proof. First, note that element π_j is to the right of element π_i. We show that the lemma follows by considering four cases, according to the positions of elements i and j in relation with the elements $\pi_{|i|}$ and $\pi_{|j|}$.

Case (i), $|i| < |j| < \pi^{-1}_{|j|} < \pi^{-1}_{|i|}$: note that both $\pi_{|i|}$ and $\pi_{|j|}$ are to the left of S. Then, after reversing S, element i is closer to its correct position, while element j is moved away from its correct position. Despite this, the distance between $\pi_{|j|}$ and j in π' is smaller than the distance between $\pi_{|i|}$ and i in π, and, so, if π is a λ-permutation, π' is also a λ-permutation.

Case (ii), $|i| < \pi^{-1}_{|j|} \leq |j| < \pi^{-1}_{|i|}$: note that $\pi_{|i|}$ is to the left of S, and $\pi_{|j|}$ is in S. Then, after reversing S, the element i is closer to its correct position and the distance of j to its correct position will still be less than λ, since the size of S is, at most, λ, as Lemma 2 shows.

Case (iii), $\pi^{-1}_{|j|} \leq |i| < \pi^{-1}_{|i|} \leq |j|$: similar to (ii).

Case (iv), $\pi^{-1}_{|j|} < \pi^{-1}_{|i|} \leq |i| < |j|$: similar to (i). □

Theorem 4. *The problems of Sorting (Unsigned or Signed) λ-Permutations by λ-Reversals have $(10 + 2\lambda)$-approximation algorithms.*

Proof. Let $\lambda \geq 2$ be an integer and $\pi \neq \iota$ be a λ-permutation. Consider an algorithm which first applies one λ-reversal over each decreasing strip of π in order to get a λ-permutation with only increasing strips. By Lemma 10, we guarantee that all intermediary permutations generated by these λ-reversals are λ-permutations.

Then, the algorithm will repeatedly take the smallest element out-of-place and move the increasing strip that contains it to its correct position, obtaining a λ-permutation with at least one less breakpoint, until it reaches the identity permutation.

As shown in Lemma 8, at most $5 + \lambda - 1$ λ-reversals are needed to move each strip to its correct position. Since, maybe, one extra λ-reversal could have been applied in the beginning of the algorithm to transform such strip into an increasing one, we have that, at most, $6 + \lambda - 1$ λ-reversals can be applied to remove at least one breakpoint. Therefore, the number of operations of our algorithm is, at most,

$$(6 + \lambda - 1)b(\pi) \leq 2(6 + \lambda - 1)d_{\beta}^{\lambda}(\pi) = (10 + 2\lambda)d_{\beta}^{\lambda}(\pi) \ ,$$

for $\beta \in \{r, \bar{r}\}$, and the inequality follows from Fact 1. □

Theorem 5. *The problem of Sorting λ-Permutations by λ-Transpositions has a* 12*-approximation algorithm.*

Proof. Let $\lambda \geq 2$ be an integer, and let $\pi \neq \iota$ be a λ-permutation. The algorithm will repeatedly take the smallest element out-of-place and move the increasing strip that contains such element to its correct position, obtaining a λ-permutation with at least one less breakpoint, until it reaches the identity permutation.

As shown in Lemma 7, at most, 4 λ-transpositions are needed to move each strip to its correct position. Then, in the worst case, we remove 1 breakpoint every 4 λ-transpositions applied. With this and Fact 1, the number of operations of our algorithm is, at most, $4b(\pi) \leq 12d_{t}^{\lambda}(\pi)$. □

Theorem 6. *The problems of Sorting (Unsigned or Signed) λ-Permutations by λ-Reversals and λ-Transpositions have 12-approximation algorithms.*

Proof. Let $\lambda \geq 2$ be an integer, and let $\pi \neq \iota$ be a λ-permutation. Let $\pi_j = i$ be the smallest element out-of-place in π.

We have two cases to consider: when the strip which contains π_j is decreasing or not. In both cases, we can at least remove breakpoint (π_{i-1}, π_i) from π without adding other ones by applying, at most, 4 λ-transpositions (if the strip is increasing) or, at most, 2 λ-operations (if the strip is decreasing), as shown in Lemmas 7 and 9, respectively.

Then, considering both cases described, the algorithm will repeatedly take the smallest element out-of-place and move the strip that contains it to its correct position, decreasing at least one breakpoint at a time, until it reaches the identity permutation.

Note that, in the worst case, we remove 1 breakpoint every 4 λ-transpositions, and, so, the result is analogous to Theorem 5. □

Note that $b(\pi)$ is $O(n)$ and the time complexity to find the strip with the smallest element out-of-place in π is $O(n)$. So, we conclude that the time complexity for the $O(1)$-approximation algorithms and the $O(\lambda)$-approximation algorithm are $O(n(n + \lambda))$ and $O(n(n + \lambda^2))$, respectively.

6. Experimental Results

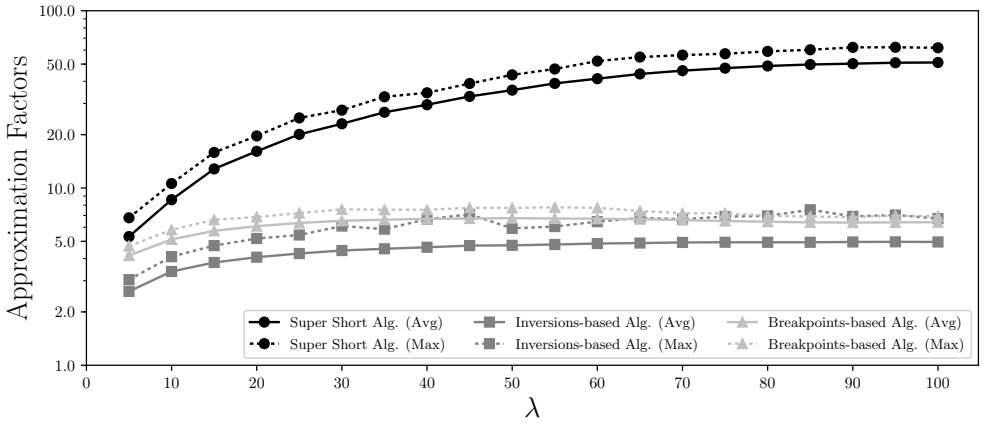
In order to analyze how the proposed algorithms work from a practical perspective, we have implemented the inversions-based (Theorems 2 and 3) and the breakpoints-based (Theorems 4–6) greedy algorithms.

We can also achieve an approximation factor of $O(\lambda^2)$ with algorithms that always apply an operation of size 2 according to Lemma 3, for unsigned permutations, or Lemma 5,

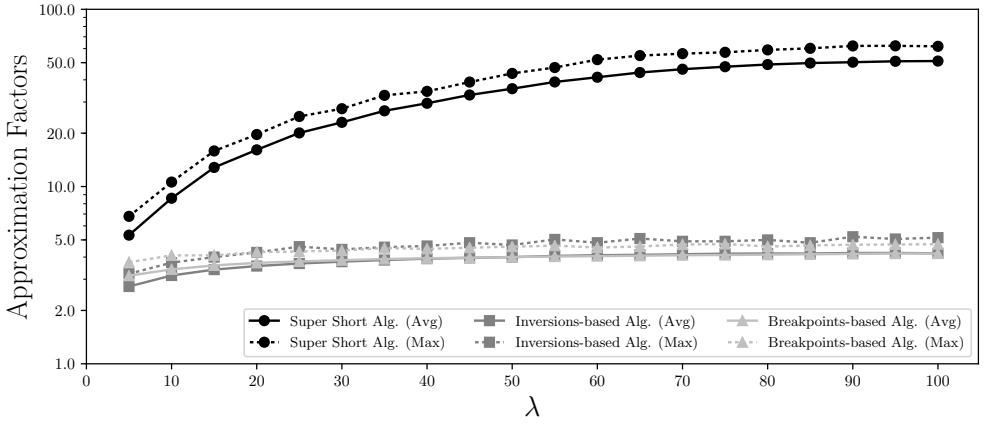
for signed permutations. There exists a way to implement such algorithms so that they have time complexity of $O(n \log n)$ [25]. We also implemented these algorithms, which we named super short algorithms, for comparison purposes. The difference between the super short algorithms and the inversions-based algorithms from Theorems 2 and 3 is that the super short algorithms only use operations of size 2, while the algorithms from Theorems 2 and 3 use, at each iteration, the operation that most decrease the number of inversions, which can have size, at most, λ.

We performed experiments considering a total of 1000 signed and unsigned λ-permutations, with size equal to 100 and values of $\lambda \in \{5, 10, 15, \ldots, 100\}$, as input for the algorithms. The considered λ-permutations were generated in two different ways: (i) totally random, and (ii) by applying 20 random λ-operations (according to the rearrangement model of each problem) over the identity. We reinforce that all generated permutations in both cases, including the intermediary ones of case (ii), are λ-permutations. For (i), the λ-permutations tend to have a high amount of breakpoints, and for (ii) they tend to have a high amount of strips and, moreover, we know the distance is, at most, 20. Then, we compared the results according to the average and maximum approximation factors obtained for all permutations. For each permutation, we calculated the approximation factor by dividing the size of the sorting sequence by the maximum value of lower bound between the ones shown in Lemma 1 and Fact 1.

We show the results for totally random λ-permutations in Figures 1 and 2, and for λ-permutations generated from the identity in Figures 3 and 4.

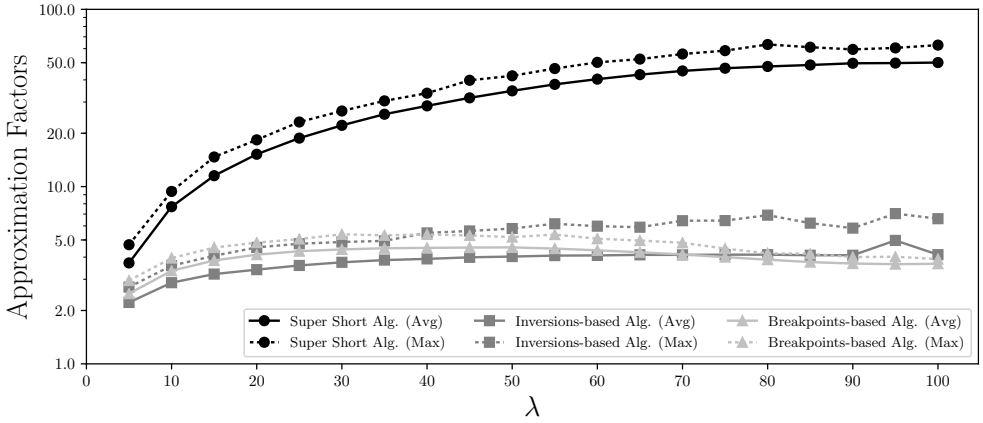


(**a**) Sorting Signed λ-Permutations by λ-Reversals.

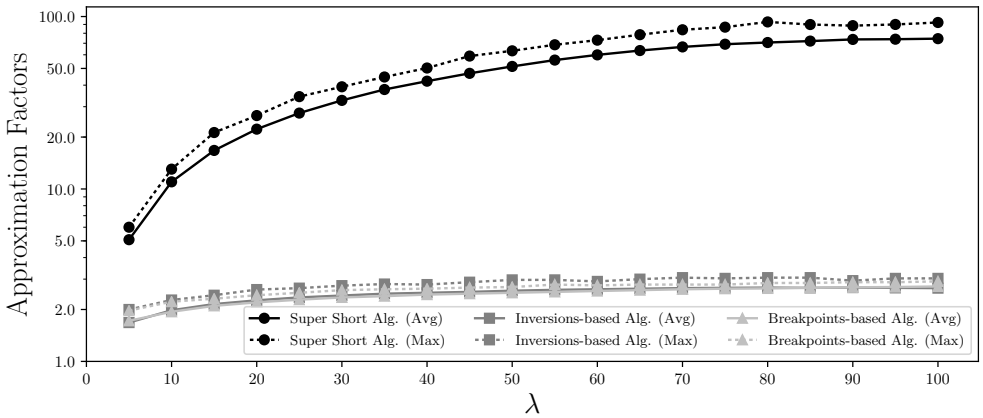


(**b**) Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions.

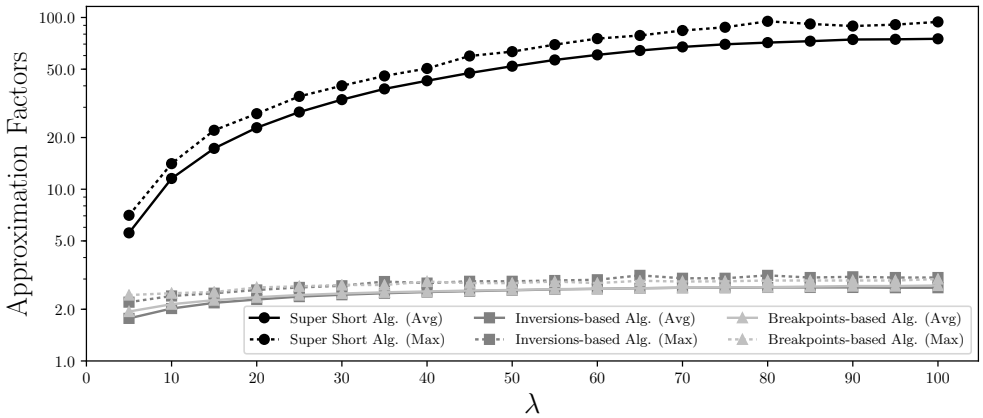
Figure 1. Average and maximum approximation factors of the algorithms for Sorting Signed λ-Permutations by λ-Operations, with totally random λ-permutations of size 100 as inputs. Note that a logarithmic scale is used in the y-axis of each graph.



(**a**) Sorting Unsigned λ-Permutations by λ-Reversals.

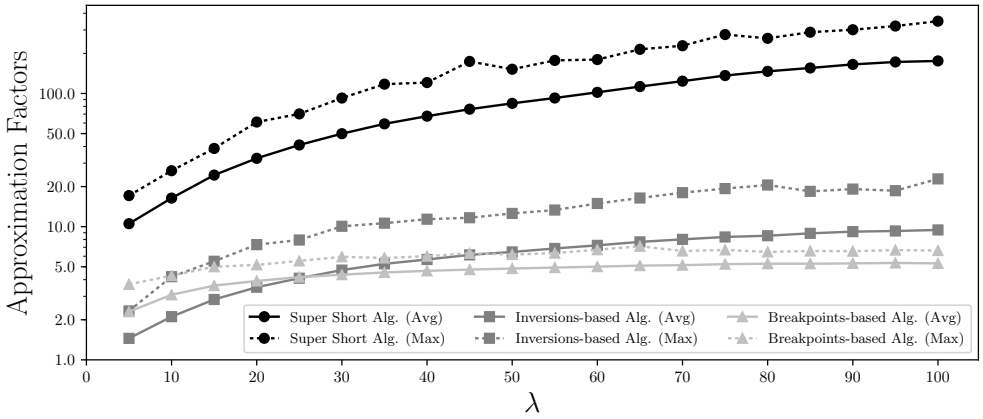


(**b**) Sorting Unsigned λ-Permutations by λ-Transpositions.

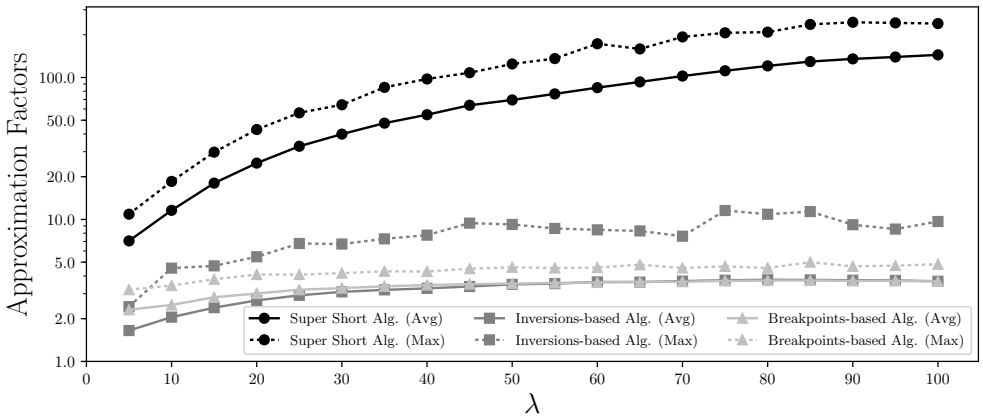


(**c**) Sorting Unsigned λ-Permutations by λ-Reversals and λ-Transpositions.

Figure 2. Average and maximum approximation factors of the algorithms for Sorting Unsigned λ-Permutations by λ-Operations, with totally random λ-permutations of size 100 as inputs. Note that a logarithmic scale is used in the y-axis of each graph.

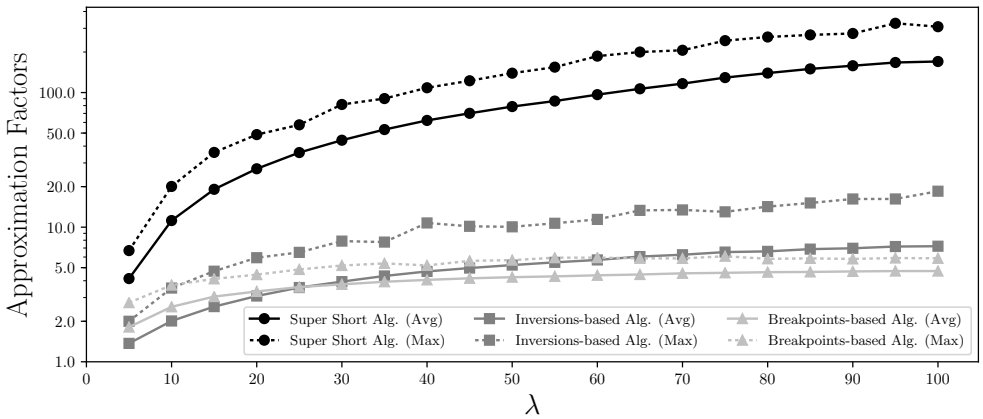


(**a**) Sorting Signed λ-Permutations by λ-Reversals.



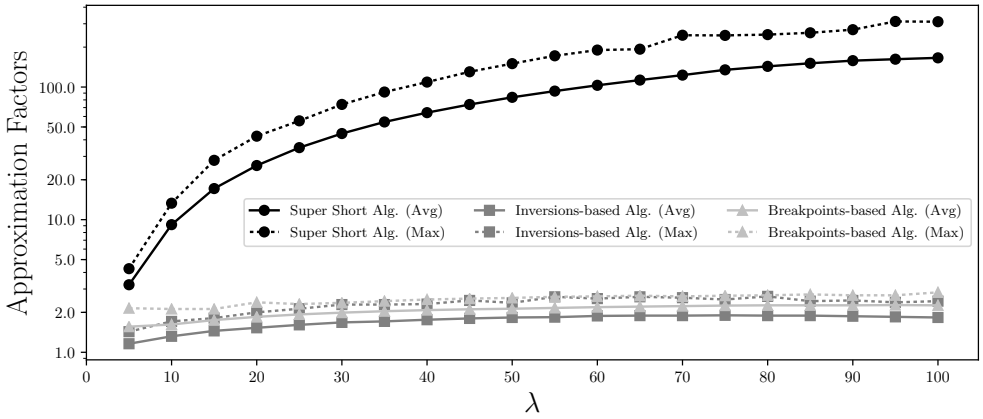
(**b**) Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions.

Figure 3. Average and maximum approximation factors of the algorithms for Sorting Signed λ-Permutations by λ-Operations, with random λ-permutations generated from ι of size 100 as inputs. Note that a logarithmic scale is used in the y-axis of each graph.

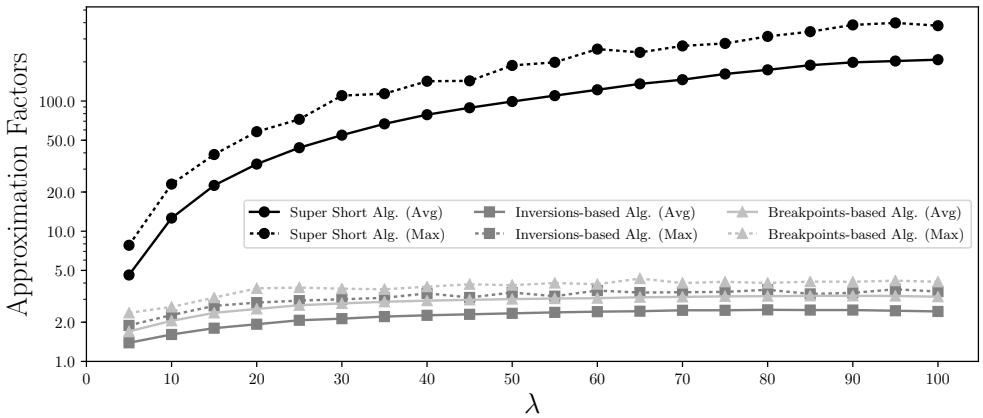


(**a**) Sorting Unsigned λ-Permutations by λ-Reversals.

Figure 4. *Cont*.



(**b**) Sorting Unsigned λ-Permutations by λ-Transpositions.



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

Figure 4. Average and maximum approximation factors of the algorithms for Sorting Unsigned λ-Permutations by λ-Operations, with random λ-permutations generated from ι of size 100 as inputs. Note that a logarithmic scale is used in the y-axis of each graph.

Even though the inversions-based and the super short algorithms have the same theoretical approximation factor, the experiments show that the inversions-based algorithms are much better in practice. Hence, for the rest of this analysis, we only compare the results of the inversions-based and breakpoints-based algorithms. Considering the results for totally random λ-permutations, we observed that the maximum approximation factors for sorting unsigned λ-permutations problems were 5.38 and 7.04 for λ-reversals, 2.91 and 3.06 for λ-transpositions, and 3.00 and 3.15 for when both λ-operations are allowed, considering the breakpoints-based and the inversions-based algorithms, respectively. For the sorting signed λ-permutations problems, the maximum approximation factors were 7.78 and 7.54 for λ-reversals and 4.74 and 5.21 for when both λ-operations are allowed, considering the breakpoints-based and the inversions-based algorithms, respectively.

Another observation is that, for totally random λ-permutations, the average approximation factor of the inversions-based algorithm and the breakpoints-based algorithm were similar (except for Sorting Signed λ-Permutations by λ-Reversals), even with the relevant difference among their theoretical approximation factors.

Regarding the results for random λ-permutations generated from the identity, the maximum approximation factors for sorting unsigned λ-permutations problems were 6.08 and 18.5 for λ-reversals, 2.83 and 2.64 for λ-transpositions, and 4.33 and 3.58 for when both λ-operations are allowed, considering the breakpoints-based and the inversions-based algorithm, respectively. For the sorting signed λ-permutations problems, the maximum approximation factors were 7.13 and 22.85 for λ-reversals and 5.00 and 11.56 for when both

λ-operations are allowed, considering the breakpoints-based and the inversions-based algorithm, respectively.

Furthermore, for this second type of λ-permutations, the maximum approximation factors of the inversions-based algorithm for the sorting signed λ-permutations problems and for the Sorting Unsigned λ-Permutations by λ-Reversals were considerably greater when compared with their average approximation factors. Despite that, the average and maximum factors given by the inversions-based algorithm were better for the two sorting unsigned λ-permutations problems which allow λ-transpositions. For both problems which allow only λ-reversals, the breakpoints-based algorithm had better average and maximum approximation factors. For Sorting Signed λ-Permutations by λ-Reversals and λ-Transpositions, both algorithms had similar average approximation factor, but the breakpoints-based algorithm had a better and more constant maximum approximation factor.

Our experiments reveal that the average approximation factor for the breakpoint-based algorithms slight increases for the interval $\lambda \in [30..100]$, while the average approximation for the inversion-based algorithms has a greater increase in the same interval. This occurred because there are more λ-reversals that decreases a breakpoint, when using bigger values of λ, since the strips tend to get smaller compared to the value of λ, which is a crucial point in Lemmas 8 and 9. A similar relation is not so common for inversions, since extending the segment to be inverted may lead to inversions being added.

7. Conclusions

In this work, we introduced the problems of Sorting λ-Permutations by λ-Operations, considering reversals and transpositions on signed and unsigned permutations.

We presented an NP-hardness proof for the models containing unsigned reversals, transpositions, and the combination of (signed or unsigned) reversals and transpositions, when $\lambda = cn^d$ for positive constants c and $0 < d \leq 1$. We developed algorithms with approximation factors of $O(\lambda^2)$, $O(\lambda)$, and $O(1)$ for all problems studied. Besides that, experiments were also performed in order to see their performance on simulated data. Tables 1 and 2 summarize these results.

For future work, we intend to develop approximation algorithms with better approximation factors for the problems of Sorting λ-Permutations by λ-Operations. Another direction of future work is to study the complexity for the model containing only signed reversals and other values of λ.

Table 1. Summary of the approximation results for the problems considering signed permutations. The columns "Approx." and "Time" refer to the theoretical approximation factor and the worst-case complexity time, respectively.

	Reversals		**Reversals and Transpositions**	
	Approx.	**Time**	**Approx.**	**Time**
Breakpoints-based	$10 + 2\lambda$	$O(n(n + \lambda^2))$	12	$O(n(n + \lambda))$
Inversions-based	$\frac{\lambda(\lambda-1)}{2} + \lambda$	$O(n^2\lambda^3\sqrt{\log \lambda})$	$\frac{\lambda(\lambda-1)}{2} + \lambda$	$O(n^2\lambda^4\sqrt{\log \lambda})$

Table 2. Summary of the approximation results for the problems considering unsigned permutations. The columns "Approx." and "Time" refer to the theoretical approximation factor and the worst-case complexity time, respectively.

	Reversals		**Transpositions**		**Reversals and Transpositions**	
	Approx.	**Time**	**Approx.**	**Time**	**Approx.**	**Time**
Breakpoints-based	$10 + 2\lambda$	$O(n(n + \lambda^2))$	12	$O(n(n + \lambda))$	12	$O(n(n + \lambda))$
Inversions-based	$O(\lambda^2)$	$O(n^2\lambda^3\sqrt{\log \lambda})$	$O(\lambda^2)$	$O(n^2\lambda^4\sqrt{\log \lambda})$	$O(\lambda^2)$	$O(n^2\lambda^4\sqrt{\log \lambda})$

Author Contributions: Conceptualization, G.H.S.M., A.O.A., C.N.L. and Z.D.; Data curation, G.H.S.M.; Formal analysis, G.H.S.M., A.O.A. and C.N.L.; Funding acquisition, Z.D.; Investigation, G.H.S.M., A.O.A., C.N.L. and Z.D.; Writing—original draft, G.H.S.M. and A.O.A.; Writing—review & editing, A.O.A., C.N.L and Z.D. All authors have read and agreed to the published version of the manuscript.

Funding: 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 304380/2018-0), and the São Paulo Research Foundation, FAPESP (grants 2013/08293-7, 2015/11937-9, 2017/12646-3, 2017/16246-0, and 2017/16871-1).

Data Availability Statement: All algorithms were implemented and they are available at https://github.com/compbiogroup/lambda-permutations, alongside the instances used in the experiments.(accessed on 29 April 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bulteau, L.; Fertin, G.; Rusu, I. Sorting by Transpositions is Difficult. *SIAM J. Comput.* **2012**, *26*, 1148–1180. [CrossRef]
2. Caprara, A. Sorting Permutations by Reversals and Eulerian Cycle Decompositions. *SIAM J. Discret. Math.* **1999**, *12*, 91–110. [CrossRef]
3. Berman, P.; Hannenhalli, S.; Karpinski, M. 1.375-Approximation Algorithm for Sorting by Reversals. In *Lecture Notes in Computer Science, Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002), Rome, Italy, 17–21 September 2002*; Möhring, R., Raman, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2461, pp. 200–210.
4. Elias, I.; Hartman, T. A 1.375-Approximation Algorithm for Sorting by Transpositions. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2006**, *3*, 369–379. [CrossRef] [PubMed]
5. Hannenhalli, S.; Pevzner, P.A. Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. *J. ACM* **1999**, *46*, 1–27. [CrossRef]
6. Oliveira, A.R.; Brito, K.L.; Dias, U.; Dias, Z. On the Complexity of Sorting by Reversals and Transpositions Problems. *J. Comput. Biol.* **2019**. [CrossRef] [PubMed]
7. Rahman, A.; Shatabda, S.; Hasan, M. An Approximation Algorithm for Sorting by Reversals and Transpositions. *J. Discret. Algorithms* **2008**, *6*, 449–457. [CrossRef]
8. Walter, M.E.M.T.; Dias, Z.; Meidanis, J. Reversal and Transposition Distance of Linear Chromosomes. In Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE'1998), Santa Cruz, Bolivia, 9–11 September 1998; IEEE Computer Society: Los Alamitos, CA, USA, 1998; pp. 96–102.
9. Chen, X. On Sorting Unsigned Permutations by Double-Cut-and-Joins. *J. Comb. Optim.* **2013**, *25*, 339–351. [CrossRef]
10. Dias, Z.; Meidanis, J. Sorting by Prefix Transpositions. In *Lecture Notes in Computer Science, Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002), Lisbon, Portugal, 11–13 September 2002*; Laender, A.H.F., Oliveira, A.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2476, pp. 65–76.
11. Lintzmayer, C.N.; Fertin, G.; Dias, Z. Sorting Permutations by Prefix and Suffix Rearrangements. *J. Bioinform. Comput. Biol.* **2017**, *15*, 1750002. [CrossRef] [PubMed]
12. Lefebvre, J.F.; El-Mabrouk, N.; Tillier, E.R.M.; Sankoff, D. Detection and validation of single gene inversions. *Bioinformatics* **2003**, *19*, i190–i196. [CrossRef] [PubMed]
13. Heath, L.S.; Vergara, J.P.C. Sorting by Short Swaps. *J. Comput. Biol.* **2003**, *10*, 775–789. [CrossRef] [PubMed]
14. Jiang, H.; Feng, H.; Zhu, D. An 5/4-Approximation Algorithm for Sorting Permutations by Short Block Moves. In *Lecture Notes in Computer Science, Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC'2014), Jeonju, Korea, 15–17 December 2014*; Ahn, H., Shin, C., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2014, Volume 8889, pp. 491–503.
15. Vergara, J.P.C. Sorting by Bounded Permutations. Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 1998.
16. Galvão, G.R.; Lee, O.; Dias, Z. Sorting Signed Permutations by Short Operations. *Algorithms Mol. Biol.* **2015**, *10*, 1–17. [CrossRef] [PubMed]
17. Zhang, S.; Zhu, D.; Jiang, H.; Guo, J.; Feng, H.; Liu, X. Sorting a Permutation by Best Short Swaps. *Algorithmica* **2021**, 1–27. [CrossRef]
18. Miranda, G.H.S.; Lintzmayer, C.N.; Dias, Z. Sorting Permutations by Limited-Size Operations. In *Algorithms for Computational Biology*; Springer International Publishing: Heidelberg, Germany, 2018; Volume 10849, pp. 76–87.
19. Blanchette, M.; Kunisawa, T.; Sankoff, D. Parametric Genome Rearrangement. *Gene* **1996**, *172*, GC11–GC17. [CrossRef]
20. Miranda, G.H.S.; Alexandrino, A.O.; Lintzmayer, C.N.; Dias, Z. Sorting λ-Permutations by λ-Operations. In *Proceedings of the 11th Brazilian Symposium on Bioinformatics (BSB'2018), Niterói, Brazil, 30 October–1 November 2018*; Springer International Publishing: Heidelberg, Germany, 2018; pp. 1–13.
21. Bafna, V.; Pevzner, P.A. Genome Rearrangements and Sorting by Reversals. *SIAM J. Comput.* **1996**, *25*, 272–289. [CrossRef]

22. Bafna, V.; Pevzner, P.A. Sorting by Transpositions. *SIAM J. Discret. Math.* **1998**, *11*, 224–240. [CrossRef]
23. Chan, T.M.; Pătraşcu, M. Counting inversions, offline orthogonal range counting, and related problems. In Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, USA, 17–19 January 2010; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2010; pp. 161–173.
24. Alexandrino, A.O.; Miranda, G.H.S.; Lintzmayer, C.N.; Dias, Z. Length-weighted λ-rearrangement distance. *J. Comb. Optim.* **2021**, *41*, 579–602. [CrossRef]
25. Swenson, K.M.; Rajan, V.; Lin, Y.; Moret, B.M.E. Sorting Signed Permutations by Inversions in $O(n \log n)$ time. *J. Comput. Biol.* **2010**, *17*, 489–501. [CrossRef] [PubMed]