Sorting Circular Permutations by Super Short Reversals

Gustavo Rodrigues Galvao, Christian Baudet, Zanoni Dias

Sorting Circular Permutations by Super Short Reversals

Gustavo Rodrigues Galvão[1], Christian Baudet[2,3,4], and Zanoni Dias[1]
[1] Institute of Computing, University of Campinas, Brazil.
E-mail: {ggalvao,zanoni}@ic.unicamp.br
[2] Erable Team, Inria Grenoble Rhône-Alpes, Montbonnot, France.
[3] Laboratoire Biométrie et Biologie Evolutive, Université de Lyon, Université Lyon 1, CNRS, Villeurbanne, UMR5558, France
[4] Cancer Research Center of Lyon, France.
E-mail: christian.baudet@lyon.unicancer.fr

Abstract

We consider the problem of sorting a circular permutation by super short reversals (*i.e.* reversals of length at most 2), a problem that finds application in comparative genomics. Polynomial-time solutions to the unsigned version of this problem are known, but the signed version remained open. In this paper, we present the first polynomial-time solution to the signed version of this problem. Moreover, we perform experiments for inferring phylogenies of two different groups of bacterial species and compare our results with the phylogenies presented in previous works. Finally, to facilitate phylogenetic studies based on the methods studied in this paper, we present a web tool for rearrangement-based phylogenetic inference using short operations, such as super short reversals.

Keywords : Genome Rearrangement, Short Reversals, Circular Permutations.

http://doi.ieeecomputersociety.org/10.1109/TCBB.2016.2515594

1 Introduction

Distance-based methods are one of the three large groups of methods to infer phylogenetic trees from sequence data [1, Chapter 5]. Such methods proceed in two steps. First, the evolutionary distance is computed for every sequence pair and stored in a matrix of pairwise distances. Then, a phylogenetic tree is constructed from this matrix using a specific algorithm, such as *Neighbor-Joining* [2]. Note that, to complete the first step, we need some method to estimate the evolutionary distance between a sequence pair. Assuming the sequence data correspond to complete genomes, we can resort to the genome rearrangement approach [3] to estimate the evolutionary distance.

In genome rearrangements, one estimates the evolutionary distance between two genomes by finding the rearrangement distance between them, which is the length of the shortest sequence of rearrangement operations that transforms one genome into the other. Assuming genomes consist of a single chromosome, share the same set of genes, and contain no duplicated genes, we can represent them as permutations of integers, where each integer corresponds to a gene. If, besides the order, the orientation of the genes is also regarded, then each integer has a sign, + or −, and the permutation is called a signed permutation. Similarly, we also refer to a permutation as an unsigned permutation when its elements do not have signs. Moreover, if the genomes are circular, then the permutations are also circular; otherwise, they are linear.

Many publications address the problem of finding the rearrangement distance between two permutations, which is equivalent to the problem of sorting a permutation into the identity permutation. For a detailed survey, the reader is referred to the book of Fertin *et al.* [3]. This problem varies according to the rearrangement operations allowed to sort a permutation. Reversals are one of the most common rearrangement operations observed in genomes. They reverse the order and orientation of a sequence of genes within a genome. Although the problem of sorting a permutation by reversals is a well-studied problem, most of the works do not take into account the length of the reversals (*i.e.*, the number of genes affected by it). As observed, short reversals are prevalent in the evolution of some species [4–7], therefore efforts have been made to address this issue algorithmically.

Jerrum [8] proved that the problem of sorting an unsigned linear permutation by reversals of length 2 is solvable in polynomial time. He has also shown a polynomial-time solution to the problem of sorting an unsigned circular permutation by reversals of length 2. However, Egri-Nagy *et al.* [9] recently presented a different solution (Jerrum's solution is based on a combinatorial approach while Egri-Nagy *et al.* solution is based on an algebraic approach). Heath and Vergara [10] considered the problem of sorting an unsigned linear permutation by reversals of length at most 3 and presented a 2-approximation algorithm. Galvão *et al.* [11] presented a polynomial-time solution to the problem of sorting a signed linear permutation by reversals of length at most 2 and a 5-approximation algorithm to the problem of sorting a signed linear permutation by reversals of length at most 3. Finally, Feng *et al.* [12] provided lower and upper bounds for the problems of sorting an unsigned circular permutation by reversals of length 2 and by reversals of length at most 3.

In this work, we add to those efforts and present a polynomial-time solution to the problem of sorting a signed circular permutation by super short reversals, that is, reversals which affect at most 2 elements (genes) of a permutation (genome). This solution closes a gap in the literature, as polynomial-time solutions to the problem of sorting by super short reversals are known for unsigned circular permutations [8, 9], unsigned linear permutations [8], and signed linear permutations [11]. In an attempt to evaluate the proposed methods, we perform two experiments for inferring distances and phylogenies of different groups of bacterial species. Firstly, we reproduce the experiment performed by Egri-Nagy *et al.* [9] to infer the phylogeny of 8 bacteria of *Yersinia* genus using super short reversal distance. However, we also consider the orientation of the genes, while the authors have ignored it to treat the permutations as unsigned. Secondly, we reproduce the experiment performed by Belda *et al.* [13] to infer the phylogeny of γ-proteobacteria using the reversal distance, but this time we use the super short reversal distance. In order to facilitate phylogenetic studies based on the proposed methods, we present a web tool for rearrangement-based phylogenetic inference using short operations, called ShortPhy.

We would like to note that this paper corresponds to the full version of a previously published conference paper [14]. The first relevant difference between both papers is that this one contains expanded explanations and expositions of the methods. In particular, this paper uses some of the formalism introduced by Meidanis *et al.* [15] and Solomon *et al.* [16] for dealing with the problem of sorting circular permutations by reversals. The second relevant difference is that this paper shows additional experimental results. Specifically, the experiment with γ-proteobacteria was not present in the conference paper. Lastly, the third relevant difference is that this paper presents the web tool for rearrangement-based phylogenetic inference.

The rest of this paper is organized as follows. Section 2 presents the solution developed by Jerrum [8] to the problem of sorting by cyclic super short reversals. Section 3 builds upon the previous section and presents the solution to the problem of sorting by signed cyclic super short reversals. Section 4 explains how we can use the solutions described in sections 2 and 3 to solve the problem of sorting a (signed) circular permutation by (signed) super short reversals. Section 5 presents the experimental results performed on real data and also describes ShortPhy. Finally, Section 6 concludes the paper.

2 Sorting by Cyclic Super Short Reversals

A *permutation* π is a bijection of $\{1, 2, \ldots, n\}$ onto itself. A classical notation used in combinatorics for denoting a permutation π is the two-row notation

$$\pi = \begin{pmatrix} 1 & 2 & \ldots & n \\ \pi_1 & \pi_2 & \ldots & \pi_n \end{pmatrix},$$

$\pi_i \in \{1, 2, \ldots, n\}$ for $1 \leq i \leq n$. This notation indicates that $\pi(1) = \pi_1$, $\pi(2) = \pi_2$, $\ldots$, $\pi(n) = \pi_n$. The notation used in genome rearrangement literature, which is the one we will adopt, is the one-row notation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$. We say that π has size n. The set of all permutations of size n is S_n. Finally, we use the notation π_i^{-1} to denote the position of element i in the permutation π, that is, $\pi_i^{-1} = k$ such that $\pi_k = i$. In other words, π^{-1} is the inverse of π.

For $i, j \in \{1, 2, \ldots, n\}$, define the interval $[i,j]$ to be the set $\{i, i \oplus 1, \ldots, j \ominus 1, j\}$, where $\oplus$ and $\ominus$ are the usual operations of subtraction and addition, respectively, except that we take the result modulo n and choose n rather than 0 as the representative of the class of multiples of n. For instance, if n is 6, then $[1, 3] = \{1, 2, 3\}$ and $[3, 1] = \{3, 4, 5, 6, 1\}$. A *cyclic reversal* $\rho(i, j)$ is an operation that transforms a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ into the permutation $\pi' = \pi \cdot \rho(i, j)$ such that

$$\pi'_x = \begin{cases} \pi_{i \oplus j \ominus x} & \text{if } x \in [i,j], \\ \pi_x & \text{otherwise.} \end{cases}$$

The cyclic reversal $\rho(i, j)$ is called a *cyclic k-reversal* if $k = j \ominus i \oplus 1$. It is called *super short* if $k = 2$. In other words, a cyclic super short reversal $\rho(i, i \oplus 1)$ is equivalent to the permutation $(1 \ldots i+1\ i \ldots n)$ if $i < n$ and to the permutation $(n\ 2 \ldots n-1\ 1)$ if $i = n$. For this reason, the action of $\rho(i, i \oplus 1)$ over a permutation π, denoted by $\pi \cdot \rho(i, i \oplus 1)$, can be seen as a composition of permutations performed from right to left.

The problem of sorting by cyclic super short reversals consists in finding the minimum number of cyclic super short reversals that transform a permutation $\pi \in S_n$ into $\iota_n = (1\ 2 \ldots n)$. This number is referred to as the *cyclic super short reversal distance* of permutation π and it is denoted by $d(\pi)$.

Let $S = \rho_1, \rho_2, \ldots, \rho_t$ be a sequence of cyclic super short reversals that sorts a permutation $\pi \in S_n$, that is,

$$\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \iota_n.$$

Then, we have that

$$\iota_n \cdot \rho_t^{-1} \cdot \rho_{t-1}^{-1} \cdots \rho_1^{-1} = \pi.$$

We say that the sequence $S^{-1} = \rho_t^{-1}, \rho_{t-1}^{-1}, \ldots, \rho_1^{-1}$ generates π and, therefore, is a generator sequence for π. In this section, we revise the solution presented by Jerrum [8] to the problem of finding a minimum-length generator sequence and explain how it relates to the solution to the problem of finding a minimum-length sorting sequence.

Before presenting Jerrum's solution [8], it is important to note that he uses a different convention for compositions: in his paper, they are performed from left to right; in ours, from right to left. Therefore, the solution to the problem of finding a minimum-length generator sequence using left-to-right convention is also a solution to the problem of finding a minimum-length sorting sequence using the right-to-left convention, as Lemma 1 shows.

Lemma 1. *A sequence $S = \rho_1, \rho_2, \ldots, \rho_t$ is a generator sequence for π when the compositions are performed from left to right if, and only if, S is a sorting sequence for π when the compositions are performed from right to left.*

Proof. Suppose that the sequence S generates π when the compositions are performed from left to right. Then, this sequence generates π^{-1} when the compositions are performed from right to left because it will give the same result as of performing the compositions from left to right in the sequence $S^{-1} = \rho_t^{-1}, \rho_{t-1}^{-1}, \ldots, \rho_1^{-1}$ (note that $\rho_i^{-1} = \rho_i$). Since $\pi \cdot \pi^{-1} = \iota_n$, it follows that S sorts π when the compositions are performed from right to left. Now, suppose that S sorts π when the compositions are performed from right to left. Then, this sequence generates π when the compositions are performed from left to right because it will give the same result as of performing the compositions from right to left in the sequence S^{-1}. Therefore the lemma follows. □

From this point, we will consider that the compositions are performed from left to right when a permutation is being generated. On the other hand, we will consider that the compositions are performed from right to left when a permutation is being sorted. For this reason, according to Lemma 1, saying that a sequence S sorts a permutation π is equivalent to saying that S generates π.

Another consequence of the convention difference is that some notions need to be translated from his convention to ours. For instance, given a sequence $S = \rho_1, \rho_2, \ldots, \rho_t$ that generates a permutation $\pi \in S_n$, Jerrum [8] defines the *generator trace* of an element $i \in [1,n]$ as the sequence $(p_0, \ldots, p_t)$ of images of i under the partial compositions of S. More specifically, if π^k is the permutation generated by the partial composition $\iota_n \cdot \rho_1 \cdots \rho_k$, with $\pi^0 = \iota_n$, then $p_k = \pi_i^k$. Now, let $\phi : \mathbb{Z} \to \{-1,0,+1\}$ be the function given by

$$\phi(i) = \begin{cases} -1 & \text{if } i \equiv -1 \pmod{n} \\ +1 & \text{if } i \equiv +1 \pmod{n} \\ 0 & \text{otherwise.} \end{cases}$$

Then, he defines the *generator displacement vector* $d^S \in \mathbb{Z}^n$ of the sequence S as the vector whose ith component is given by

$$d_i^S = \sum_{i=1}^{t} \phi(p_i - p_{i-1}),$$

where $(p_0, \ldots, p_t)$ is the generator trace of i.

Now, considering that S sorts π, we define the *sorting trace* of an element π_i as the sequence $(p_0, \ldots, p_t)$ of positions of π_i under the partial compositions of S. More specifically, if π^k is the permutation generated by the partial composition $\pi \cdot \rho_1 \cdots \rho_k$, with $\pi^0 = \pi$, then $p_k = \pi^{k^{-1}}_{\pi_i}$. Then, we define the *sorting displacement vector* $d^S(\pi) \in \mathbb{Z}^n$ of π with respect to S as the vector whose ith component is given by

$$d_i^S(\pi) = \sum_{i=1}^{t} \phi(p_i - p_{i-1}),$$

where $(p_0, \ldots, p_t)$ is the sorting trace of π_i.

Lemma 2. *Let $S = \rho_1, \rho_2, \ldots, \rho_t$ be a sequence that sorts a permutation $\pi \in S_n$. Then, $d^S = d^S(\pi)$.*

Proof. We prove this lemma by induction on the length of S. The base case $|S| = 1$ is trivial so assume $|S| > 1$. Let S' be sequence yielded by removing ρ_1 from S, that is, $S' = \rho_2, \rho_3, \ldots, \rho_t$. Then, on the one hand, S' generates the permutation $\pi' = \rho_1 \cdot \pi$. On the other hand, S' sorts permutation $\pi'' = \pi \cdot \rho_1$. We argue that $\pi' = \pi''$. To see why this is true, suppose that $\rho_1 = \rho(i, i \oplus 1)$. Then, $\pi'_k = \pi_k$ for $k \notin \{i, i \oplus 1\}$, $\pi'_i = \pi_{i\oplus 1}$, and $\pi'_{i\oplus 1} = \pi_i$. In other words, π' can be obtained from π by switching the positions of elements π_i and $\pi_{i\oplus 1}$, which is precisely the result of performing the composition $\pi \cdot \rho(i, i \oplus 1)$ from right to left. Therefore, $\pi' = \pi''$ and, by induction hypothesis, $d^{S'} = d^{S'}(\pi')$.

For analyzing the effect of adding $\rho_1 = \rho(i, i \oplus 1)$ back to S', let $k \notin \{i, i \oplus 1\}$ be an integer. If the generator trace of k with respect to S' is $(k, p_1, \ldots, p_{t-1})$, then the generator trace of k with respect to S is $(k, k, p_1, \ldots, p_{t-1})$. Similarly, the sorting trace of an element π'_k with respect to S' is $(k, p_1, \ldots, p_{t-1})$ and the sorting trace of π_k with respect to S is $(k, k, p_1, \ldots, p_{t-1})$. Therefore, $d_k^S = d_k^{S'} = d_k^{S'}(\pi') = d_k^S(\pi)$.

Now, let $(i, p_1, \ldots, p_{t-1})$ be the generator trace of i and $(i \oplus 1, p'_1, \ldots, p'_{t-1})$ be the generator trace of $i \oplus 1$ with respect to S'. Then, we have that $(i, i \oplus 1, p'_1, \ldots, p'_{t-1})$ is the generator trace of i and $(i \oplus 1, i, p_1, \ldots, p_{t-1})$ is the generator trace of $i \oplus 1$ with respect to S. Therefore, $d_i^S = d_{i\oplus 1}^{S'} + 1$ and $d_{i\oplus 1}^S = d_i^{S'} - 1$.

Similarly, let $(i, p_1, \ldots, p_{t-1})$ be the sorting trace of π'_i and $(i \oplus 1, p'_1, \ldots, p'_{t-1})$ be the sorting trace of $\pi'_{i\oplus 1}$ with respect to S'. Then, we have that $(i, i \oplus 1, p'_1, \ldots, p'_{t-1})$ is the sorting trace of π_i and $(i \oplus 1, i, p_1, \ldots, p_{t-1})$ is the sorting trace of $\pi_{i\oplus 1}$ with respect to S. Therefore, $d_i^S(\pi) = d_{i\oplus 1}^{S'}(\pi') + 1$ and $d_{i\oplus 1}^S(\pi) = d_i^{S'}(\pi') - 1$.

Since $d_i^{S'} = d_i^{S'}(\pi')$ and $d_{i\oplus 1}^{S'} = d_{i\oplus 1}^{S'}(\pi')$, it follows that $d_i^S = d_i^S(\pi)$ and $d_{i\oplus 1}^S = d_{i\oplus 1}^S(\pi)$, what completes the proof. □

Example 1. *Let $S = \rho_1, \rho_2, \rho_3$ be a sequence of cyclic super short reversals such that $\rho_1 = \rho(1, 4)$, $\rho_2 = \rho(1, 2)$, and $\rho_3 = \rho(2, 3)$. Assuming that S is a generator sequence, the partial compositions of S, with $\pi^0 = (1\ 2\ 3\ 4)$, produce the permutations $\pi^1 = (4\ 2\ 3\ 1)$, $\pi^2 = (4\ 1\ 3\ 2)$, and $\pi^3 = (4\ 1\ 2\ 3)$. Therefore, the generator traces of elements 1, 2, 3, and 4 are respectively (1, 4, 4, 4), (2, 2, 1, 1), (3, 3, 3, 2), and (4, 1, 2, 3). Moreover, the generator displacement vector of S is $d^S = [-1, -1, -1, 3]$.*

Now, considering that S sorts $\pi = (4\ 1\ 2\ 3)$, the partial compositions of S produce the permutations $\pi^1 = (3\ 1\ 2\ 4)$, $\pi^2 = (1\ 3\ 2\ 4)$, and $\pi^3 = (1\ 2\ 3\ 4)$. Therefore, the sorting traces of elements $\pi_1 = 4$, $\pi_2 = 1$, $\pi_3 = 2$, and $\pi_4 = 3$ are respectively (1, 4, 4, 4), (2, 2, 1, 1), (3, 3, 3, 2), and (4, 1, 2, 3). Moreover, the sorting displacement vector of π with respect to S is $d^S(\pi) = [-1, -1, -1, 3]$.

As Lemma 2 shows, if S is a sequence of cyclic super short reversals that sorts a permutation $\pi \in S_n$, then the generator displacement vector of S is equal to the sorting displacement vector of π with respect to S (see Example 1). Therefore, every property about the generator displacement vector proved by Jerrum [8] also applies to the sorting displacement vector. In particular, Lemma 3 states two properties of special interest.

Lemma 3. *Let $S = \rho_1, \rho_2, \ldots, \rho_t$ be a sequence of cyclic super short reversals that sorts a permutation $\pi \in S_n$. Then, we have that*

$$\sum_{i=1}^{n} d_i^S(\pi) = 0, \tag{1}$$

$$d_i^S(\pi) \equiv \pi_i - i \pmod{n}. \tag{2}$$

Note that, in one hand, we can think of a sequence of cyclic super short reversals as specifying a displacement vector. On the other hand, we can also think of a displacement vector as specifying a sequence of cyclic super short reversals. Let $x = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$ be a vector and $\pi \in S_n$ be a permutation. We say that x is a *valid vector* for π if $\sum_i x_i = 0$ and $x_i \equiv \pi_i - i \pmod{n}$. Given a vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$ and two distinct integers $i, j \in \{1, 2, \ldots, n\}$, let $r = i - j$ and $s = (i + x_i) - (j + x_j)$. The *crossing number* of i and j with respect to x is defined by

$$c_{ij}(x) = \begin{cases} |\{k \in [r, s] : k \equiv 0 \pmod{n}\}| & \text{if } r \leq s, \\ -|\{k \in [s, r] : k \equiv 0 \pmod{n}\}| & \text{if } r > s. \end{cases}$$

The crossing number of x is defined by $C(x) = \frac{1}{2}\sum_{i \neq j} |c_{ij}(x)|$.

Using the notion of crossing number, Jerrum [8, Lemma 3.6] was able to prove a fundamental lemma: if S is a minimum-length sequence that generates a permutation π, then the length of S is equal to the crossing number of the generator displacement vector of S. Lemma 4 is the restatement of that lemma considering lemmas 1 and 2.

Lemma 4. *Let S be a minimum-length sequence of cyclic super short reversals that sorts a permutation $\pi \in S_n$ and let $x \in \mathbb{Z}^n$ be a valid vector for π. If $d^S(\pi) = x$, then $d(\pi) = C(x)$.*

Lemma 4 allows the problem of sorting a permutation π by cyclic super short reversals to be recast as the optimization problem of finding a valid vector $x \in \mathbb{Z}^n$ for π with minimum crossing number. More specifically, as Jerrum [8] pointed out, this problem can be formulated as the integer program:

$$\begin{gathered} \text{Minimize } C(x) \text{ over } \mathbb{Z}^n \\ \text{subject to } \textstyle\sum_i x_i = 0,\ x_i \equiv \pi_i - i \pmod{n}. \end{gathered}$$

Although solving an integer program is NP-hard in the general case, Jerrum [8] presented a polynomial-time algorithm for solving this one.

Firstly, Jerrum [8] introduced a transformation $T_{ij} : \mathbb{Z}^n \to \mathbb{Z}^n$ defined as follows. For any vector $x \in \mathbb{Z}^n$, the result, $x' = T_{ij}(x)$, of applying T_{ij} to x is given by $x'_k = x_k$ for $k \notin \{i,j\}$, $x'_i = x_i - n$, and $x'_j = x_j + n$. Let $\max(x)$ and $\min(x)$ respectively denote the maximum and minimum component values of a vector $x \in \mathbb{Z}^n$. The transformation T_{ij} is said to *contract* x iff $x_i = \max(x)$, $x_j = \min(x)$ and $x_i - x_j \geq n$. Moreover, T_{ij} is said to *strictly contract* x iff, in addition, the final inequality is strict. Lemma 5 shows what is the effect of contracting transformations on the crossing number of a valid vector.

Lemma 5. *Let x and x' be two valid vectors over $\mathbb{Z}^n$ such that $x' = T_{ij}(x)$ and T_{ij} is a contracting transformation. Then, $C(x') - C(x) = 2(n + x_j - x_i)$.*

Proof. The proof of this lemma is given by Jerrum [8, Theorem 3.9]. We note, however, that he mistakenly wrote that $C(x') - C(x) = 4(n + x_j - x_i)$. In other words, he forgot to divide the result by 2. This division is necessary because the crossing number of a vector is the half of the sum of the crossing numbers of its indices. □

The algorithm proposed by Jerrum [8] starts with a feasible solution to the integer program and performs a sequence of strictly contracting transformations that decrease the value of the crossing number. When no further strictly contracting transformation can be performed, the solution is guaranteed to be optimal. It is explained by Jerrum [8]: any two local optimum solutions (*i.e.*, solutions that admit no strictly contracting transformation) can be brought into agreement with each other via a sequence of contracting transformations. The detailed algorithm is shown in Algorithm 1.

Algorithm 1: Algorithm for computing the cyclic super short reversal distance of a permutation π.

Data: A permutation $\pi \in S_n$.
Result: Cyclic super short reversal distance of π.

```
1  Let x be a n-dimensional vector
2  for k = 1 to n do
3  |  x_k ← π_k − k
4  end
5  while max(x) − min(x) > n do
6  |  Let i,j be two integers such that x_i = max(x) and x_j = min(x)
7  |  x_i ← x_i − n
8  |  x_j ← x_j + n
9  end
10 return C(x)
```

Example 2. *Suppose that we run Algorithm 1 on the permutation $\pi = (4\ 5\ 3\ 1\ 2)$. First, it will initialize the vector x to $[3, 3, 0, -3, -3]$ (note that $C(x) = 8$). The vector x admits strictly contracting transformations, hence the while loop will be executed. In the first iteration of the while loop, the algorithm will perform the transformation T_{14} to x, obtaining the vector $x = [-2, 3, 0, 2, -3]$ (note that $C(x) = 6$). In the second iteration of the while loop, the algorithm will perform the transformation T_{25} to x, obtaining the vector $x = [-2, -2, 0, 2, 2]$ (note that $C(x) = 4$). Since the resulting vector admits no strictly contracting transformation, the while loop will stop executing and the algorithm will return the value of $C(x)$, which will be equal to* 4.

Regarding the time complexity of Algorithm 1, we have that line 1 and the for loop of lines 2-4 take $O(n)$ time. Jerrum [8] observed that none of the variables x_i changes value more than once, therefore the while loop iterates at most $\frac{n}{2}$ times. Due this fact, we can implement the while loop in $O(n)$ time by using an auxiliary n-dimensional vector V. Each element of V is composed of two variables: *value* and *index*. First, we initialize V in such a way that $V_i.value = x_i + n$ and $V_i.index = i$. Since $|x_i| \leq$

n, we have that $0 \leq x_i + n \leq 2n$. This means that we can sort V with respect to the variable *value* in $O(n)$ time using counting sort. After sorting V, we can obtain $x_j = min(x)$ and $x_i = max(x)$ in constant time because, at iteration k, we have that $j = V_k.index$ and $i = V_{n+1-k}.index$. Finally, we can compute the value of $C(x)$ in $O(n^2)$ time, therefore the overall complexity of the algorithm is $O(n^2)$. In fact, this complexity analysis reveals that the time complexity of Algorithm 1 reduces to the time complexity of computing the crossing number.

Note that, in this section, we have focused on the problem of computing the cyclic super short reversal distance of a permutation rather than finding the minimum-length sequence of cyclic super short reversals that sorts it. As Jerrum [8] remarked, his proofs are constructive and directly imply an algorithm for finding the sequence of cyclic super short reversals.

3 Sorting by Signed Cyclic Super Short Reversals

A *signed permutation* π is a bijection of $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ onto itself that satisfies $\pi(-i) = -\pi(i)$ for all $i \in \{1, 2, \ldots, n\}$. The two-row notation for a signed permutation is

$$\pi = \begin{pmatrix} -n & \ldots & -2 & -1 & 1 & 2 & \ldots & n \\ -\pi_n & \ldots & -\pi_2 & -\pi_1 & \pi_1 & \pi_2 & \ldots & \pi_n \end{pmatrix},$$

$\pi_i \in \{1, 2, \ldots, n\}$ for $1 \leq i \leq n$. The one-row notation is $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$. Note that we drop the mapping of the negative elements since $\pi(-i) = -\pi(i)$ for all $i \in \{1, 2, \ldots, n\}$. By abuse of notation, we say that π has size n. The set of all signed permutations of size n is $S_n^{\pm}$.

A *signed cyclic reversal* $\rho^{\pm}(i, j)$ is an operation that transforms a signed permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ into the signed permutation $\pi' = \pi \cdot \rho^{\pm}(i, j)$ such that

$$\pi'_x = \begin{cases} -\pi_{i \oplus j \ominus x} & \text{if } x \in [i,j], \\ \pi_x & \text{otherwise.} \end{cases}$$

The signed cyclic reversal $\rho^{\pm}(i, j)$ is called a *signed cyclic k-reversal* if $k = j \ominus i \oplus 1$. It is called *super short* if $k = 1$ or $k = 2$.

The problem of sorting by signed cyclic super short reversals consists in finding the minimum number of signed cyclic super short reversals that transform a permutation $\pi \in S_n^{\pm}$ into ι_n. This number is referred to as the *signed cyclic super short reversal distance* of permutation π and it is denoted by $d^{\pm}(\pi)$.

Let the *absolute position* of an integer $i \in \{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ in the signed permutation $\pi \in S_n^{\pm}$ be the integer k such that $\pi_k = i$ or $\pi_k = -i$. Then, given a sequence $S = \rho_1^{\pm}, \rho_2^{\pm}, \ldots, \rho_t^{\pm}$ of signed cyclic super short reversals that sorts a signed permutation π, we define the *sorting trace* of an element π_i as the sequence $(p_0, \ldots, p_t)$ of absolute positions of π_i under the partial compositions of S. The *sorting displacement vector* $d^S(\pi) \in \mathbb{Z}^n$ of π with respect to S is defined the same way as in Section 2.

Example 3. *Let $\pi = (-3\ -1\ -2)$ be a signed permutation and let $S = \rho_1, \rho_2, \rho_3$ be a sorting sequence for π such that $\rho_1 = \rho(1, 2)$, $\rho_2 = \rho(2, 3)$, and $\rho_3 = \rho(3, 3)$. The partial compositions of S produce the permutations $\pi^1 = (1\ 3\ -2)$, $\pi^2 = (1\ 2\ -3)$, and*

$\pi^3 = (1\ 2\ 3)$. *Therefore, the sorting traces of elements* $\pi_1 = -3$, $\pi_2 = -1$, *and* $\pi_3 = -2$ *are respectively* (1, 2, 3, 3), (2, 1, 1, 1), *and* (3, 3, 2, 2). *Moreover, the sorting displacement vector of* π *with respect to* S *is* $d^S(\pi) = [2, -1, -1]$.

Now, let σ be the unsigned permutation such that $\sigma = (|\pi_1|\ |\pi_2|\ \ldots\ |\pi_n|)$ and let $S' = \rho_1, \rho_2, \ldots, \rho_m$, $m \leq t$, be the sequence of cyclic super short reversals obtained from S by removing the signed cyclic 1-reversals and transforming every signed cyclic 2-reversal $\rho_k^{\pm} = \rho^{\pm}(i, i \oplus 1)$ into the cyclic 2-reversal $\rho_k = \rho(i, i \oplus 1)$. Since signed cyclic 1-reversals do not affect the sorting trace of an element, it follows that $d^S(\pi) = d^{S'}(\sigma)$. For this reason, every property about the sorting displacement vector of unsigned permutations also applies to the sorting displacement vector of signed permutations. In particular, Lemma 6 is the signed analog of Lemma 3.

Lemma 6. *Let* $S = \rho_1^{\pm}, \rho_2^{\pm}, \ldots, \rho_t^{\pm}$ *be a sequence of signed cyclic super short reversals that sorts a signed permutation* $\pi \in S_n^{\pm}$. *Then, we have that*

$$\sum_{i=1}^{n} d_i^S(\pi) = 0, \tag{3}$$

$$d_i^S(\pi) \equiv |\pi_i| - i \pmod{n}. \tag{4}$$

Let $x \in \mathbb{Z}^n$ be a vector and $\pi \in S_n^{\pm}$ be a signed permutation. We say that x is a *valid vector* for π if $\sum_i x_i = 0$ and $x_i \equiv |\pi_i| - i \pmod{n}$. Given a valid vector x for the signed permutation π, we define the set $podd(\pi, x)$ as $podd(\pi, x) = \{i : \pi_i > 0 \text{ and } |x_i| \text{ is odd}\}$ and we define the set $neven(\pi, x)$ as $neven(\pi, x) = \{i : \pi_i < 0 \text{ and } |x_i| \text{ is even}\}$. Moreover, let $U(\pi, x)$ denote the union of these sets, that is, $U(\pi, x) = podd(\pi, x) \cup neven(\pi, x)$. The following lemma is the signed analog of Lemma 4.

Lemma 7. *Let* S *be a minimum-length sequence of signed cyclic super short reversals that sorts a signed permutation* $\pi \in S_n^{\pm}$ *and let* $x \in \mathbb{Z}^n$ *be a valid vector for* π. *If* $d^S(\pi) = x$, *then* $d^{\pm}(\pi) = C(x) + |U(\pi, x)|$.

Proof. Note that the sequence S can be decomposed into two distinct subsequences S_1 and S_2 such that S_1 is formed by the signed cyclic 1-reversals of S and S_2 is formed by the signed cyclic 2-reversals of S. We can assume, without loss of generality, that the signed cyclic reversals of the subsequence S_2 are applied first. This is because we can replace any pair of consecutive signed cyclic reversals $\rho(i, i)$, $\rho(j, j \oplus 1)$ with: $\rho(j, j \oplus 1)$, $\rho(i, i)$ if $i \notin \{j, j \oplus 1\}$; $\rho(j, j \oplus 1)$, $\rho(i \ominus 1, i \ominus 1)$ if $i = j \oplus 1$; and $\rho(j, j \oplus 1)$, $\rho(i \oplus 1, i \oplus 1)$ if $i = j$. By making successive replacements of this type, we eventually obtain a minimum-length sorting sequence S' in which all the signed cyclic 2-reversals are applied before the signed cyclic 1-reversals. Moreover, $d^S(\pi) = d^{S'}(\pi)$.

Now, we prove that $|S_1| = |U(\pi, x)|$ by induction on the size of S_2. For the base case, suppose that $|S_2| = 0$. Then, we have that $x_i = 0$ for $i \in [1,n]$, therefore $C(x) = 0$. Moreover, if π is not sorted, then it has $|S_1|$ negative elements. Since every $|x_i|$ is even, we can conclude that $|S_1| = |neven(\pi, x)| = |U(\pi, x)|$. Now, assume $|S_2| \geq 1$ and suppose that we apply a signed cyclic 2-reversal $\rho^{\pm}(i, i \oplus 1)$ of S_2 in π, obtaining a signed permutation π'. Moreover, let S' be the resulting sequence after we remove $\rho^{\pm}(i, i \oplus 1)$ from S_2. According to the proof of Lemma 2, we have that $d_k^{S'}(\pi') = d_k^S(\pi)$ for $k \notin \{i, i \oplus 1\}$, $d_i^{S'}(\pi') = d_{i \oplus 1}^S(\pi) + 1$, and $d_{i \oplus 1}^{S'}(\pi') = d_i^S(\pi) - 1$. Then, assuming the vector $x' \in \mathbb{Z}^n$ is equal to $d^{S'}(\pi')$, we show that:

i) $k \in U(\pi', x')$ iff $k \in U(\pi, x)$ for $k \notin \{i, i \oplus 1\}$;

ii) $i \in U(\pi', x')$ iff $i \oplus 1 \in U(\pi, x)$;

iii) and $i \oplus 1 \in U(\pi', x')$ iff $i \in U(\pi, x)$.

Statement (i) follows from the fact that $\pi'_k = \pi_k$ and $x'_k = x_k$ for $k \notin \{i, i \oplus 1\}$. To prove statement (ii), we have four cases to analyze:

a) $\pi_{i\oplus 1} > 0$ and $|x_{i\oplus 1}|$ is even. In this case, we have that $\pi'_i < 0$ and $|x'_i|$ is odd, therefore $i \notin U(\pi', x')$ and $i \oplus 1 \notin U(\pi, x)$;

b) $\pi_{i\oplus 1} > 0$ and $|x_{i\oplus 1}|$ is odd. In this case, we have that $\pi'_i < 0$ and $|x'_i|$ is even, therefore $i \in \mathit{neven}(\pi', x')$ and $i \oplus 1 \in \mathit{podd}(\pi, x)$;

c) $\pi_{i\oplus 1} < 0$ and $|x_{i\oplus 1}|$ is even. In this case, we have that $\pi'_i > 0$ and $|x'_i|$ is odd, therefore $i \in \mathit{podd}(\pi', x')$ and $i \oplus 1 \in \mathit{neven}(\pi, x)$;

d) $\pi_{i\oplus 1} < 0$ and $|x_{i\oplus 1}|$ is odd. In this case, we have that $\pi'_i > 0$ and $|x'_i|$ is even, therefore $i \notin U(\pi', x')$ and $i \oplus 1 \notin U(\pi, x)$.

Finally, the proof of statement (iii) is analogous to the proof of statement (ii). Taken together, these statements imply that $|U(\pi', x')| = |U(\pi, x)|$, completing the inductive step.

Given that $|S_1| = |U(\pi, x)|$ regardless the size of S_2 and that $|S_2| \geq C(x)$ (Lemma 4), we can conclude that $|S_2| = C(x)$, therefore the lemma follows. □

The Lemma 7 allows the problem of sorting a signed permutation π by signed cyclic super short reversals to be recast as the optimization problem of finding a valid vector $x \in \mathbb{Z}^n$ for π which minimizes the sum $C(x) + |U(\pi, x)|$. The next theorem shows how to solve this problem in polynomial time.

Theorem 1. *Let $\pi \in S_n^{\pm}$ be a signed permutation. Then, we can find a valid vector $x \in \mathbb{Z}^n$ which minimizes the sum $C(x) + |U(\pi, x)|$ in polynomial time.*

Proof. We divide our analysis into two cases:

i) n is even. In this case, we have that the value of $|U(\pi, x)|$ is the same for any feasible solution x. This is because, in order to be a feasible solution, a vector x has to satisfy the restriction $x_i \equiv |\pi_i| - i \pmod{n}$. This means that x_i is congruent modulo n with $a = |\pi_i| - i$ and belongs to the equivalence class $\{\ldots, a - 2n, a - n, a, a + n, a + 2n, \ldots\}$. Since n is even, the parities of the absolute values of the elements in this equivalence class are the same, therefore the value of $|U(\pi, x)|$ is the same for any feasible solution x. It follows that we can only minimize the value of $C(x)$ and this can be done by performing successive strictly contracting transformations.

ii) n is odd. In this case, it is possible to minimize the values of $|U(\pi, x)|$ and $C(x)$. Firstly, we argue that minimizing $C(x)$ leads to a feasible solution x'' such that $C(x'') + |U(\pi, x'')|$ is at least as low as $C(x') + |U(\pi, x')|$, where x' can be any feasible solution such that $C(x')$ is not minimum. To see this, let x' be a feasible solution such that $C(x')$ is not minimum. Then, we can perform a sequence of strictly contracting transformations which decrease the value of $C(x)$. When no further strictly contracting transformation can be performed, we obtain a solution x'' such that $C(x'')$ is minimum. On the one hand, we know from Lemma 5 that each strictly contracting transformation T_{ij} decreases $C(x)$ by at least 2 units. On the other hand, since n is odd, its possible that

the parities of $|x_i|$ and $|x_j|$ have been changed in such a way that the value of $|U(\pi, x)|$ increases by 2 units. Therefore, in the worst case, each strictly contracting transformation does not change the value of $C(x) + |U(\pi, x)|$, so $C(x') + |U(\pi, x')| \geq C(x'') + |U(\pi, x'')|$. Now, we argue that, if there exists more than one feasible solution x such that $C(x)$ is minimum, then it still may be possible to minimize the value of $|U(\pi, x)|$.

Jerrum [8, Theorem 3.9] proved that if there is more than one feasible solution such that $C(x)$ is minimum, then each of these solutions can be brought into agreement with each other via a sequence of contracting transformations. Note that a contracting transformation T_{ij} does not change the value of $C(x)$, but it can change the value of $|U(\pi, x)|$ because n is odd and the parities of $|x_i|$ and $|x_j|$ change when T_{ij} is performed. Therefore, among all feasible solutions such that $C(x)$ is minimum some of them have minimum $|U(\pi, x)|$ and these solutions are optimal. Hence, we can obtain an optimal solution by first obtaining a feasible solution with minimum $C(x)$ (this can be done by performing successive strictly contracting transformations). Then every possible contracting transformation T_{ij} which decreases the value of $|U(\pi, x)|$ can be applied to that feasible solution.

The previous analysis directly implies an exact algorithm for sorting by signed cyclic super short reversals. Such an algorithm is described below (Algorithm 2). Regarding its time complexity, we know from the previous section that lines 1-9 take $O(n)$ time. Note that, in the second while loop, none of the variables x_i changes value more than once, therefore the second while loop iterates at most $\frac{n}{2}$ times. As a consequence, we can implement the second while loop in $O(n)$ time by using an auxiliary n-dimensional vector V just as described at the end of Section 2. Finally, we can compute $C(x) + |U(\pi, x)|$ in $O(n^2)$, therefore the overall complexity of Algorithm 2 is $O(n^2)$. In fact, the time complexity of Algorithm 2 is limited by the time complexity of computing the crossing number. □

Example 4. *Suppose that we run Algorithm 2 on the permutation* $\pi = (+4\ -3\ -5\ -1\ +2)$. *First, it will initialize the vector* x *to* $[3,\ 1,\ 2,\ -3,\ -3]$ (*note that* $C(x) = 7$ *and* $|U(\pi, x)| = 3$). *The vector* x *admits strictly contracting transformations, hence the first while loop will be executed. In the first iteration of the first while loop, the algorithm will perform the transformation* T_{14} *to* x, *obtaining the vector* $x = [-2,\ 1,\ 2,\ 2,\ -3]$ (*note that* $C(x) = 5$ *and* $|U(\pi, x)| = 3$). *The resulting vector admits no strictly contracting transformation, but it admits a contracting transformation* T_{ij} *such that* $i, j \in U(\pi, x)$, *hence the second while loop will be executed. In the first iteration of the second while loop, the algorithm will perform the transformation* T_{35} *to* x, *obtaining the vector* $x = [-2,\ 1,\ -3,\ 2,\ 2]$ (*note that* $C(x) = 5$ *and* $|U(\pi, x)| = 1$). *Since the resulting vector admits no contracting transformation* T_{ij} *such that* $i, j \in U(\pi, x)$, *the second while loop will stop executing and the algorithm will return the value of* $C(x) + |U(\pi, x)|$, *which will be equal to* 6.

Until now, we have focused on the problem of computing the signed cyclic super short reversal distance of a signed permutation rather than finding the minimum-length sequence of signed cyclic super short reversals that sorts it. We remark that the proof of Lemma 7 directly implies an algorithm for finding the sequence of signed cyclic super short reversals. More specifically, given a signed permutation $\pi \in S_n^{\pm}$, we can use Algorithm 2 to obtain a valid vector $x \in \mathbb{Z}^n$ which minimizes the sum $C(x) + |U(\pi, x)|$. Then, we can find a sequence S of cyclic super short reversals that sorts the

Algorithm 2: Algorithm for computing the signed cyclic super short reversal distance of a signed permutation π.

Data: A permutation $\pi \in S_n^{\pm}$.
Result: Signed cyclic super short reversal distance of π.

```
Let x be a n-dimensional vector
for k = 1 to n do
    x_k ← |π_k| − k
end
while max(x) − min(x) > n do
    Let i,j be two integers such that x_i = max(x) and x_j = min(x)
    x_i ← x_i − n
    x_j ← x_j + n
end
if n is odd then
    while exists a contracting transformation T_ij such that i, j ∈ U(π, x)
    do
        x_i ← x_i − n
        x_j ← x_j + n
    end
end
return C(x) + |U(π, x)|
```

unsigned permutation $\sigma = (|\pi_1|\ |\pi_2|\ \ldots\ |\pi_n|)$ using the algorithm implied by Jerrum's proofs [8]. Finally, we can sort π by applying the cyclic super short reversals of S as if they were signed followed by signed cyclic 1-reversals on the remaining negative elements.

4 Sorting Circular Permutations

In this section, we explain how we can use the solution to the problem of sorting by (signed) cyclic super short reversals to solve the problem of sorting a (signed) circular permutation by (signed) super short reversals. This explanation is based on the works of Meidanis *et al.* [15], Solomon *et al.* [16], and Egri-Nagy *et al.* [9], where one can find more details.

A *circular permutation* is an arrangement of the elements of $\{1, 2, \ldots, n\}$ around the circle. A *super short reversal* is an operation that swaps two adjacent elements of a circular permutation. Similarly, a *signed circular permutation* is an arrangement of n elements of $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ around the circle such that $|i| \neq |j|$ for any two distinct elements i and j in the arrangement. A *signed super short reversal* is an operation that either swaps two adjacent elements and flips their signs or just flip the sign of a single element of a signed circular permutation.

Given an initial element of a (signed) circular permutation with n elements, we can represent it as a (signed) permutation – such as defined in the two previous sections – as follows. Walk around the circle in a clockwise direction, beginning at the initial

element, and write down the elements as they are visited. This process can produce n different (signed) permutations, one for each possible initial element. Moreover, since a (signed) circular permutation represents a circular chromosome, which lives in three dimensions, it can also be "turned over" before we begin the walk. By doing it, the process can also produce n different (signed) permutations. Because the n (signed) permutations produced in the first process are different from those produced in the second process, both processes can produce a total of $2n$ different (signed) permutations. All these $2n$ (signed) permutations are equivalent as they represent different viewpoints of a chromosome represented by a (signed) circular permutation.

In order to formalize the discussion of the last paragraph, we define two operations on (signed) permutations: rotations and reflections. A *rotation*, denoted by r, is an operation that moves the elements of a (signed) permutation one position to the right, that is, $\pi \cdot r = (\pi_n\ \pi_1\ \ldots,\ \pi_{n-2},\ \pi_{n-1})$. We define r^i for every $i \in \mathbb{Z}$ as follows. For $i > 0$, r^i is the composition of r i times and r^{-i} is the inverse of r^i. Moreover, r^0 is the identity. Note that $r^i = r^{\oplus i}$, $r^{-i} = r^{\ominus i}$, and $r^i \cdot r^j = r^{i+j}$ for all $i, j \in \mathbb{Z}$. A *reflection*, denoted by s, is an operation that reverses the order of the elements of a (signed) permutation. Besides, it also flips the signs of the elements if the permutation is signed. For instance, if π is an unsigned permutation, then $\pi \cdot s = (\pi_n\ \pi_{n-1}\ \ldots\ \pi_2\ \pi_1)$. On the other hand, if π is signed, then $\pi \cdot s = (-\pi_n\ -\pi_{n-1}\ \ldots\ -\pi_2\ -\pi_1)$. We define s^i for every $i \in \mathbb{Z}$ as follows. For $i > 0$, s^i is the composition of s i times and s^{-i} is the inverse of s^i. Moreover, s^0 is the identity. Note that $s^{-i} = s^i$, $s^{2i} = s^0$, and $s^{2i+1} = s^1$ for all $i \in \mathbb{Z}$.

Given two (signed) permutations π and σ, we define a binary relation $\sim$ between them as follows: $\pi \sim \sigma$ if, and only if, there are $i, j \in \mathbb{Z}$ such that $\pi = \sigma \cdot r^i \cdot s^j$. It is not hard to see that the binary relation $\sim$ is an equivalence relation (note that $r^i \cdot s^j = s^j \cdot r^{-i}$). For this reason, given a permutation π that represents a circular permutation, we can define the equivalence class of π, denoted by $[\pi]$, as $[\pi] = \{\sigma \in S_n : \pi \sim \sigma\}$. This equivalence class defines a circular permutation. Similarly, given a signed permutation π that represents a signed circular permutation, we can define the equivalence class of π as $[\pi] = \{\sigma \in S_n^{\pm} : \pi \sim \sigma\}$. This equivalence class defines a signed circular permutation.

Since a (signed) circular permutation is defined by the equivalence class of a (signed) permutation that represents it and a (signed) super short reversal corresponds to a (signed) cyclic super short reversal, the problem of sorting a (signed) circular permutation by (signed) super short reversals can be stated as follows: given a (signed) permutation π that represents a (signed) circular permutation, find the minimum number of (signed) cyclic super short reversals that transform a permutation in $[\pi]$ into a permutation in $[\iota_n]$. This number is referred to as the (*signed*) *super short reversal distance* of the (signed) circular permutation represented by π and it is denoted by $d^\circ([\pi])$.

Lemma 8. *Let π be an unsigned permutation that represents a circular permutation. Then, $d^\circ([\pi]) = \min_{\sigma \in [\pi]}\{d(\sigma)\}$.*

Proof. It is not hard to see that $d^\circ([\pi]) \leq \min_{\sigma \in [\pi]}\{d(\sigma)\}$, so let us see why $d^\circ([\pi]) \geq \min_{\sigma \in [\pi]}\{d(\sigma)\}$. Assume that $d^\circ([\pi]) = t$ and let $S = \rho_1, \ldots, \rho_t$ be a minimum-length sequence of cyclic super short reversals that transforms a permutation $\sigma \in [\pi]$ into a permutation $\gamma \in [\iota_n]$. Then, we have that

$$\sigma \cdot \rho_1 \cdots \rho_t = \gamma$$
$$\sigma \cdot \rho_1 \cdots \rho_t = \iota_n \cdot r^i \cdot s^j$$

$$\sigma \cdot \rho_1 \cdots \rho_t \cdot s^j \cdot r^{-i} = \iota_n$$
$$\sigma \cdot s^j \cdot r^{-i} \cdot \rho'_1 \cdots \rho'_t = \iota_n$$

The last equality comes from the fact that

$$\rho(i,\, i \oplus 1) \cdot r = r \cdot \rho(i \oplus 1,\, i \oplus 2),$$
$$\rho(i,\, i \oplus 1) \cdot s = s \cdot \rho(n \ominus i,\, n \ominus i \oplus 1).$$

Therefore, there exists a minimum-length sequence $S' = \rho'_1, \ldots, \rho'_t$ that sorts the permutation $\sigma \cdot s^j \cdot r^{-i} \in [\pi]$, and the lemma follows. □

Lemma 9. *Let π be a signed permutation that represents a signed circular permutation. Then, $d^\circ([\pi]) = \min_{\sigma \in [\pi]}\{d^\pm(\sigma)\}$.*

Proof. The proof is analogous to the proof of Lemma 8 by observing that

$$\rho^\pm(i,\, i) \cdot r = r \cdot \rho^\pm(i \oplus 1,\, i \oplus 1),$$
$$\rho^\pm(i,\, i) \cdot s = s \cdot \rho^\pm(1 \ominus i,\, 1 \ominus i),$$
$$\rho^\pm(i,\, i \oplus 1) \cdot r = r \cdot \rho^\pm(i \oplus 1,\, i \oplus 2),$$
$$\rho^\pm(i,\, i \oplus 1) \cdot s = s \cdot \rho^\pm(n \ominus i,\, n \ominus i \oplus 1).$$

□

According to lemmas 8 and 9, the problem of computing the (signed) super short reversal distance of a (signed) circular permutation represented by a (signed) permutation π can be reduced to the problem of finding a (signed) permutation in $[\pi]$ with minimum (signed) cyclic super short reversal distance. This a precisely what algorithms 3 and 4 do. In order to compute the (signed) cyclic super short reversal distance, these algorithms have to execute algorithms 1 and 2 $O(n)$ times. Therefore, the overall complexity of algorithms 3 and 4 is $O(n^3)$.

Algorithm 3: Algorithm for computing the super short reversal distance of a circular permutation.

Data: A permutation $\pi \in S_n$.
Result: Super short reversal distance of the circular permutation represented by π.

```
d ← ∞
for σ ∈ [π] do
    d ← min{d, d(σ)}
end
return d
```

5 Experimental Results and Discussion

We implemented the algorithms for computing the (signed) super short reversal distance of (signed) circular permutations (the implementation is available for download[1]) and performed experiments for inferring distances and phylogenies of two different groups of bacterial species. The first group is composed of 8 bacteria from the

[1] `https://github.com/chrbaudet/SuperShortReversals`

Algorithm 4: Algorithm for computing the super short reversal distance of a signed circular permutation.

Data: A permutation $\pi \in S_n^{\pm}$.
Result: Signed super short reversal distance of the signed circular permutation represented by π.

$d \leftarrow \infty$
for $\sigma \in [\pi]$ **do**
  $d \leftarrow \min\{d, d^{\pm}(\sigma)\}$
end
return d

Yersinia genus (Section 5.1) and the second group is composed of 28 γ-proteobacteria (Section 5.2). In all experiments, we used PHYLIP package [17] to build the phylogenetic trees.

In order to facilitate phylogenetic studies based on the methods studied in this paper, we have implemented a web tool. Section 5.3 describes it in more detail.

5.1 Dataset 1 : *Yersinia* genomes

Table 1 shows a list of 8 *Yersinia* bacteria which were first explored in the work of Darling *et al.* [7]. Starting from the complete genome sequence of these organisms, the authors constructed a multiple alignment using Mauve [18] software to identify conserved blocks. Based on the alignment, they performed phylogenetic studies of the *Yersinia* using Bayesian statistical methods. More recently, Egri-Nagy *et al.* [9] performed phylogenetic studies of this same group of bacteria using distance-based methods. More precisely, they computed the super short reversal distance between the permutations obtained from Darling *et al.* [7] and constructed a phylogenetic tree using Neighbor-Joining.

Table 1: List of *Yersinia* bacteria

Code	**Species**
YPK	*Yersinia pestis Kim*
YPA	*Yersinia pestis Antiqua*
YPM	*Yersinia pestis Microtus 91001*
YPC	*Yersinia pestis CO92*
YPN	*Yersinia pestis Nepal516*
YPP	*Yersinia pestis Pestoides F 15-70*
YT1	*Yersinia pseudotuberculosis IP31758*
YT2	*Yersinia pseudotuberculosis IP32953*

We reproduced the experiment performed by Egri-Nagy *et al.*, but we also considered the signed super short reversal distance. Moreover, differently from them, we considered that each permutation has 78 elements rather than 79. It is due to Darling *et al.* [7], who claimed that they could identify 78 conserved segments (or blocks)

using Mauve, but they provided permutations with elements ranging from 0 to 78. In a personal communication, Darling confirmed that there are actually 78 blocks, with 0 and 78 being part of the same block (see Supplementary Material File 1 to obtain the 8 signed permutations for this group of species). The computed distance matrices are shown in tables 2 and 3. We also constructed phylogenetic trees from each matrix using Neighbor-Joining. The resulting topologies are illustrated by figures 1 and 2. As we can see, the topology of both trees is the same.

Table 2: *Yersinia* dataset – Super short reversal distance matrix

	YPK	YPA	YPM	YPC	YPN	YPP	YT1	YT2
YPK	0000	0216	0728	0179	0319	0502	0740	0728
YPA	0216	0000	0736	0313	0255	0470	0691	0730
YPM	0728	0736	0000	0697	0729	0596	0327	0345
YPC	0179	0313	0697	0000	0358	0617	0746	0734
YPN	0319	0255	0729	0358	0000	0507	0590	0591
YPP	0502	0470	0596	0617	0507	0000	0399	0418
YT1	0740	0691	0327	0746	0590	0399	0000	0157
YT2	0728	0730	0345	0734	0591	0418	0157	0000

Table 3: *Yersinia* dataset – Signed super short reversal distance matrix

	YPK	YPA	YPM	YPC	YPN	YPP	YT1	YT2
YPK	0000	0243	0752	0205	0338	0533	0764	0760
YPA	0243	0000	0772	0352	0279	0510	0724	0773
YPM	0752	0772	0000	0728	0747	0643	0361	0385
YPC	0205	0352	0728	0000	0381	0656	0776	0760
YPN	0338	0279	0747	0381	0000	0547	0617	0624
YPP	0533	0510	0643	0656	0547	0000	0434	0457
YT1	0764	0724	0361	0776	0617	0434	0000	0189
YT2	0760	0773	0385	0760	0624	0457	0189	0000

For comparison purposes, we computed the pairwise super short reversal distance between (unsigned) permutations with 79 elements, but our results differed from the ones of Egri-Nagy *et al.* After personal communications with one of the authors, we found a small bug in the script they used to compute the distance matrix. The corrected script, which is available for download,[2] generates the same results as ours.

Despite the observed differences regarding the distance matrices, we remark that the topology of the phylogenetic tree obtained by Egri-Nagy *et al.* (Figure 3) is almost the same as of the trees we obtained. Considering the pair of *Y. pseudotuberculosis* (`YT1` and `YT2`) as outgroup, our trees show that `YPM` was the first to diverge, followed by the divergences of `YPP`, `YPN`, `YPA`, and the final divergence of `YPK` and `YPC`. In their tree, the divergence of `YPN` happened before the divergence of `YPC`, which occurred previous to the divergence of `YPK` and `YPA`. This slight difference does not invalidate Egri-Nagy

[2] `https://bitbucket.org/egri-nagy/biogap`

et al. analysis that the tree produced using super short reversal distance is consistent with the tree of Bos *et al.* [19], which was produced using sequence variation.

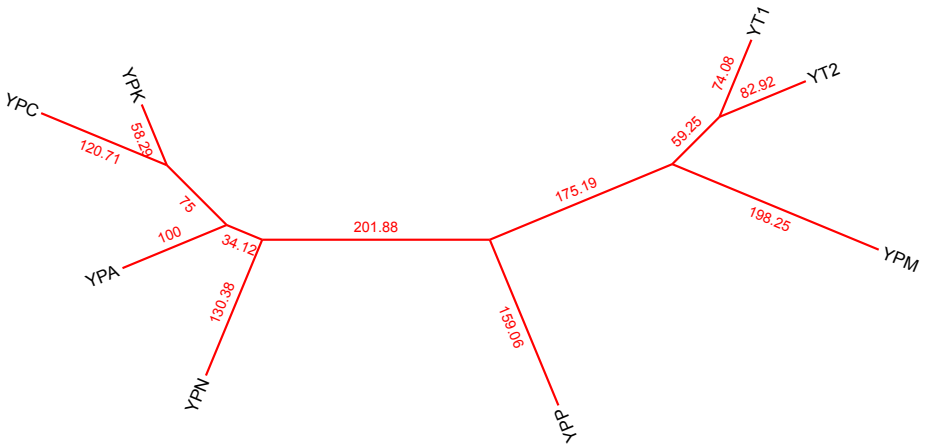

Figure 1: *Yersinia* dataset – Super short reversal distance phylogeny

We can also compare our trees with the consensus phylogenetic network presented by Darling *et al.* [7], who used reversal information without length restriction. As for the similarities, the pair of *Y. pseudotuberculosis* groups together. Moreover, considering this pair as an outgroup, `YPP` and `YPM` were the first to diverge, and the remaining bacteria group together. As for the differences, we can observe that `YPP` diverged before `YPM` in Darling *et al.* phylogeny. We can also observe that `YPN` is sibling of `YPK`, that `YPC` is sibling of `YPA`. Additionally, these four bacteria have a common ancestor that is a descendant of the least common ancestor of these four bacteria and `YPM`.

5.2 Dataset 2 : γ-proteobacteria genomes

The list of 30 γ-proteobacteria shown in Table 4 was extracted from the work of Belda *et al.* [13]. In their work, the authors provided a table containing a list of 244 orthologous genes found within these species. However, as we will further explain below, we could not reconstruct the original signed permutations.

Belda *et al.* adopted the nomenclature of the Microbial Genome Database for Comparative Analysis (MGBD) [20,21] for the gene names. For this reason, we downloaded the MBGD tables (release 2014-02) and we tried to match the names found in the orthology table produced by Belda *et al.* to obtain gene directions. During this process, we observed that the orthology information had significant changes for two species (wgl and sfl), therefore we removed them from our study.

In the orthology table provided by Belda et al., each entry is composed of a list of orthologous genes (one gene for each species). Most of the 244 entries in this table were correct and we could retrieve the missing gene directions by looking for the correspondent genes in the MBGD database. On the other hand, some of the entries had incorrect or outdated names, so we updated the ones whose correspondence could be found without ambiguity. After doing this, we discarded a total of 15 entries of the

Table 4: List of γ-proteobacteria

MBGD 3-letter code	**Species**
buc	*Buchnera aphidicola APS*
bab	*Buchnera aphidicola Bp (Baizongia pistaciae)*
bas	*Buchnera aphidicola Sg*
bfl	*Blochmannia floridanus*
ecc	*Escherichia coli CFT073*
eco	*Escherichia coli K-12 MG1655*
ecs	*Escherichia coli Sakai*
ece	*Escherichia coli EDL933*
hdu	*Haemophilus ducreyi 35000HP*
hin	*Haemophilus influenzae Rd KW20*
pae	*Pseudomonas aeruginosa PAO1*
ppu	*Pseudomonas putida KT2440*
pst	*Pseudomonas syringae DC3000*
pmu	*Pasteurella multocida PM70*
sfx	*Shigella flexneri 2457T*
son	*Shewanella oneidensis MR-1*
stm	*Salmonella typhimurium LT2 SGSC 1412; ATCC 700720*
stt	*Salmonella enterica Ty2*
sty	*Salmonella enterica CT18*
vch	*Vibrio cholerae N16961*
vpa	*Vibrio parahaemolyticus O3:K6 RIMD 2210633*
vvu	*Vibrio vulnificus CMCP6*
xac	*Xanthomonas axonopodis 306*
xcc	*Xanthomonas campestris ATCC 33913*
xfa	*Xylella fastidiosa 9a5c*
xft	*Xylella fastidiosa Temecula1*
ype	*Yersinia pestis CO92*
ypk	*Yersinia pestis KIM*
Excluded species	
wgl	*Wigglesworthia glossinidia endosymbiont of Glossina morsitan*
sfl	*Shigella flexneri 301*

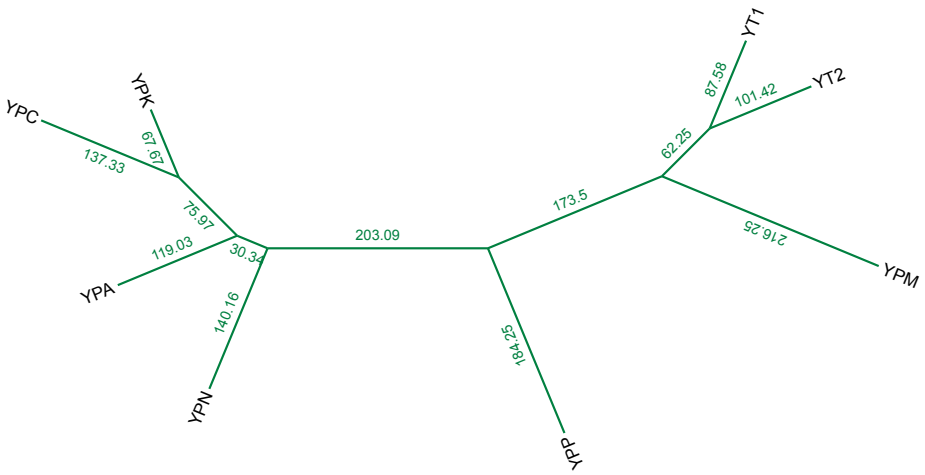

Figure 2: *Yersinia* dataset – Signed super short reversal distance phylogeny

table of orthologous genes because they had at least one gene whose name could not be updated. Furthermore, a total of 17 entries of the table were discarded because the current release of the MBGD database does not mark their listed genes as orthologous genes. Thus, at the end of this process, we obtained permutations composed of 212 genes. Supplementary Material File 2 contains the list of orthologous genes of the 28 species and Supplementary Material File 3 contains the resulting signed permutations.

Similarly to what we have done in Section 5.1, we computed the (signed) super short reversal distance between every permutation pair and used the Neighbor-Joining to produce the phylogenetic trees. The obtained distance matrices can be found in the Supplementary Material File 4. The resulting phylogenetic trees are illustrated by figures 4 and 5.

In the work of Belda *et al.*, the species `xcc` and `xac` (*Xanthomonas* genus) and `xfa` and `xfl` (*Xylella* genus) are used as outgroup. We could not define this species as an outgroup in any of the trees we obtained. For comparison purposes, we rooted our trees in the branch that leads to the least common ancestor of that four bacteria.

Differently from the *Yersinia*, the phylogenetic trees obtained for the γ-proteobacteria using super short reversal distance for unsigned and signed permutations (Figures 4 and 5) show a slight difference in the topology. In both trees, we can define three groups of species: $A = \{$`son`,`hdu`,`xfl`,`xfa`,`pmu`,`hin`$\}$, $B = \{$`bas`,`bab`,`buc`,`stt`,`sty`,`sfx`,`stm`,`ecc`,`ecs`,`eco`,`ece`$\}$, and $C = \{$`vvu`,`vpa`,`vch`,`ypk`,`ype`,`bfl`,`pae`,`xcc`,`xac`,`pst`,`ppu`$\}$. Taking this groups into account, the tree inferred with super short reversal distance shows a topology of the type $(B,(A,C))$ while the tree inferred with signed super short reversal distance shows a topology of the type $(A,(B,C))$.

Figure 6 shows the tree produced by Belda *et al.* using Maximum Likelihood [13, Figure 2] and Figure 7 shows the tree produced by Belda *et al.* using Neighbor-Joining applied to a reversal distance matrix [13, Figure 3b)]. We assigned colors to the groups of species for helping in the comparison. Although both trees have many similarities, two big differences can be observed: a) in the Maximum Likelihood tree, the first group

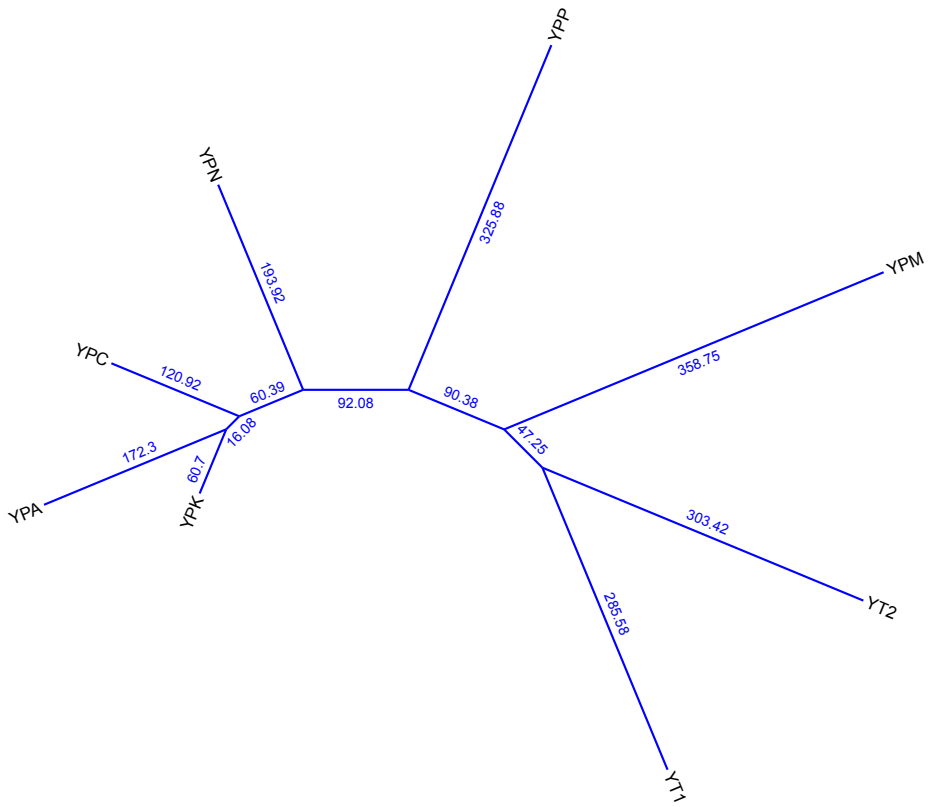

Figure 3: Phylogenetic tree obtained by Egri-Nagy *et al.* [9]

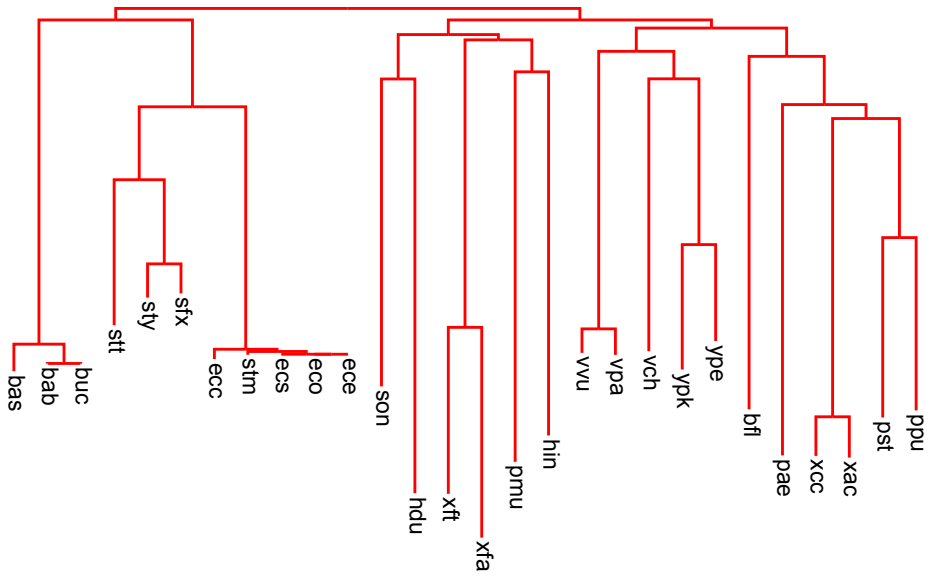

Figure 4: γ-proteobacteria dataset – Super short reversal distance phylogeny

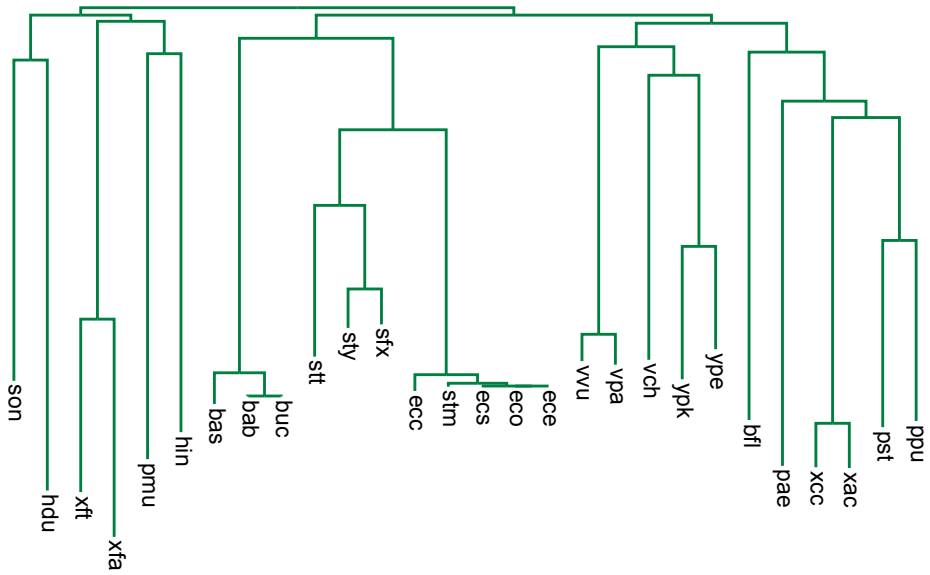

Figure 5: γ-proteobacteria dataset – Signed super short reversal distance phylogeny

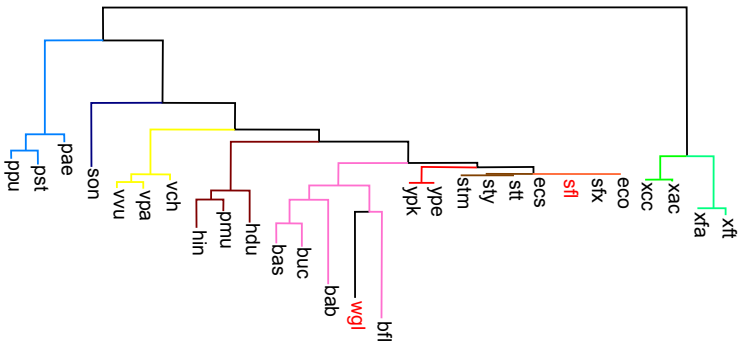

Figure 6: Phylogenetic tree obtained by Belda *et al.* [13] through Maximum Likelihood. The authors used Tree-Puzzle 5.2 and performed an alignment of concatenated amino acid sequences from the proteins encoded by *rpoC*, *rpoB*, *rho*, *rpoA*, *rpsC*, *rpsD*, *nusG*, *rpsG*, *rplK*, and *rpsK* genes. The colors identify groups of species to facilitate comparison among different phylogenetic trees. The bacteria `wgl` and `sfl` (colored in red) were excluded from our study. The bacteria `ecc` and `ece` were not considered by Belda *et al.* in the construction of this tree.

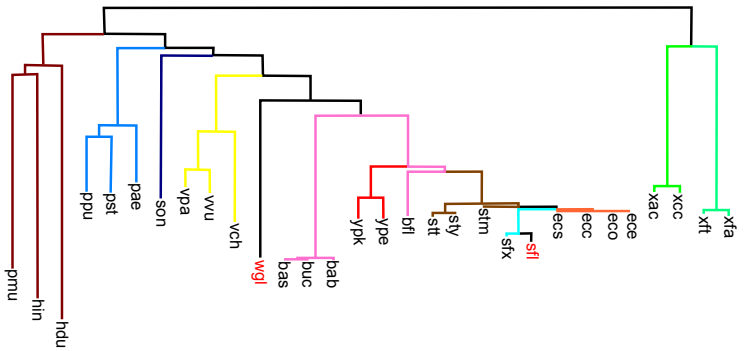

Figure 7: Phylogenetic tree obtained by Belda *et al.* [13] using Neighbor-Joining applied to a reversal distance matrix. The colors identify groups of species to facilitate comparison among different phylogenetic trees. The bacteria `wgl` and `sfl` (colored in red) were excluded from our study.

to diverge (after the root) is composed of *Pseudomonas* bacteria (`pae`, `pst`, and `ppu`) while in the reversal distance tree the first group to diverge is composed of `pmu`, `hdu`, and `hin`; and b) in the reversal distance tree, the pair of *Yersinia* bacteria appears as an inner-group among the bacteria of the genus *Buchnera* (`buc`, `bab`, and `bas`).

In order to compare our inferred phylogeny with the ones inferred by Belda *et al.*, we also colored the tree we produced using Neighbor-Joining applied to the signed super short reversal distance matrix (Figure 8). As we pointed out before, in our tree there is an absence of the outgroup composed of *Xanthomonas* and *Xylella* bacteria: between this two pairs of species, our tree shows many speciation events that give rise to other groups of species. A similar case can be observed among the bacteria of the genus *Buchnera* (`buc`, `bab`, and `bas`) and the species `bfl` (*Blochmannia floridanus*): while these species are grouped together in the Maximum Likelihood tree, they are separated by many speciation events in our tree.

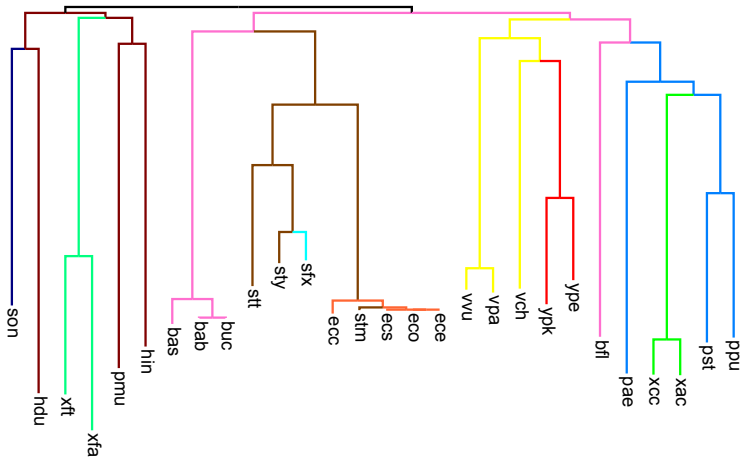

Figure 8: Phylogenetic tree obtained by us using Neighbor-Joining applied to the signed super short reversal distance matrix. The colors identify groups of species to facilitate comparison among different phylogenetic trees.

For most of the groups of species, we can observe that the internal relationships of our tree are consistent with the internal relationships of the trees produced by Belda *et al.* On the other hand, the ancestry relationships among different groups of species are distinct. We believe that these observed differences indicate that the super short reversal distance may be not enough to explain the evolutionary history of different groups of species. One could say that this was expected because considering only super short reversals is a very stringent requirement when we compare genomes of not closely related species.

5.3 ShortPhy: Rearrangement-Based Phylogeny Using Short Operations

ShortPhy[3] is a web tool for rearrangement-based phylogenetic inference using short operations. The phylogenies are inferred by ShortPhy as usual: first, a pairwise distance matrix is calculated from the input data; then, a phylogenetic tree is constructed from this matrix using Neighbor-Joining. The input data is a file containing the permutations that represent the genomes. The distances are computed by finding the rearrangement distance between every permutation pair.

The computed rearrangement distances may vary according to three parameters: (i) flag to consider, or not, the gene orientation, (ii) the type of the genomes (linear or circular), and (iii) the type of the operations. Combining these parameters, we can derive a specific variant of the problem of sorting a permutation. The variants supported by ShortPhy are:

1. Sorting Unsigned Linear Permutations by Super Short Reversals;
2. Sorting Signed Linear Permutations by Super Short Reversals;
3. Sorting Signed Linear Permutations by Super Short Reversals and Super Short Transpositions;
4. Sorting Unsigned Circular Permutations by Super Short Reversals;
5. Sorting Signed Circular Permutations by Super Short Reversals;

These variants are the only ones for which exact polynomial-time solutions are known, considering the special case of short operations. Moreover, we have used the solutions presented by Jerrum [8] for variant 1, the solutions presented by Galvão *et al.* [11] for variants 2 and 3, and the solutions presented in sections 2 and 3 for variants 4 and 5, respectively.

ShortPhy comprises its algorithms and the web server. They were implemented in Java. The web server runs on a machine featuring an Intel Core 2 Quad CPU at 3.0 GHz and 16GB of RAM running GNU/Linux 2.6.32 and Apache Tomcat 6.0.26.

In order to generate a phylogeny, the user has to fill a form with six fields:

- **Permutations file**. This file must contain permutations representing the genomes. Each line of this file must begin with a string, which uniquely identifies the genome, followed by the elements of the permutation separated by commas. Circular genomes must be represented by linear permutations, just as described in Section 4.
- **Gene orientation**. This field indicates whether the orientation of genes must be considered. If the orientation is considered, then the permutations are treated as signed. Otherwise, they are treated as unsigned.
- **Type of the genomes**. This field indicates whether the genomes are circular or linear.
- **Type of the operations**. This field indicates which types of operations must be considered. Note that the available types of operations may vary according to the other fields.
- **Tree building method**. This field indicates which Neighbor-Joining version/implementation must be used.

[3] `http://mirza.ic.unicamp.br:8080/shortphy`

- **E-mail**. This field is optional. If it is filled, an e-mail is sent to the user after the phylogeny has been generated.

In addition to the phylogeny, ShortPhy generates three output files:

- a file containing the pairwise distance matrix in CSV (Comma-Separated Values) format;
- a file containing the phylogenetic tree in NEXUS file format [22];
- and a file containing the phyloXML [23] description of the phylogenetic tree.

We recommend the users to download these files because they are temporary.

6 Conclusion

In this paper, we presented a polynomial-time solution to the problem of sorting a signed circular permutation by super short reversals. From a theoretical perspective, this solution is important because it closes a gap in the literature. From a biological perspective, it is important because signed permutations constitute a more suitable model for genomes. Moreover, we performed experiments to infer distances and phylogenies of two distinct groups of bacterial species: one group composed of 8 bacteria from the *Yersinia* genus and the other composed of 28 γ-proteobacteria. The experimental results indicate that, while producing consistent results for closely related species, the super short reversal distance may not be suitable to explain the evolutionary history of more distant species. Finally, we presented ShortPhy, the first web tool for rearrangement-based phylogenetic inference using short operations.

Even if the model of super short reversals does not reflect the evolution of a group of species, we think that the super short reversal distance can be used to measure the degree of difference between their genomes with respect to the order and the orientation of their genes. For instance, suppose that we have three bacterial species: A, B, and C. Besides, suppose that we can observe the occurrence of two super short reversals between A and B, and the occurrence of a single but very long reversal between A and C. Using the reversal distance, one would consider A closer to C than to B because a single reversal can transform A into C while two reversals are necessary to transform A into B. On the other hand, using super short reversal distance, we have the opposite scenario: one would consider A closer to B than to C. It reflects the fact that A and B have much more genes whose order and orientation are the same than A and C. In this sense, the super short reversal distance is closely related to the shuffling distance [24, 25], which is one of the first measures proposed to compare whole genomes.

We see some possible directions for future work. One is to improve the time complexity of the algorithms that compute the (signed) cyclic super short reversal distance. As we have discussed, the time complexity of these algorithms is limited by the time complexity of computing the crossing number. Therefore, all one has to do is to find a faster way to compute it. Another possibility is to consider other rearrangement operations, such as super short transpositions, for sorting a circular permutation.

Acknowledgments

GRG acknowledges the support by the FAPESP and CAPES under grant #2014/04718-6, São Paulo Research Foundation (FAPESP). CB acknowledges the support by the French Project ANR MIRI BLAN08-1335497 and by the ERC Advanced Grant SISYPHE (FP7/2007-2013)/ERC grant agreement no. [247073]10. ZD acknowledges the support by the CNPq under grants 306730/2012-0, 477692/2012-5, and 483370/2013-4. Finally, the authors thank the Center for Computational Engineering and Sciences at Unicamp for financial support through the FAPESP/CEPID Grant 2013/08293-7.

References

[1] P. Lemey, M. Salemi, and A. Vandamme, *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing.* Cambridge University Press, 2009.

[2] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406–425, 1987.

[3] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette, *Combinatorics of Genome Rearrangements.* The MIT Press, 2009.

[4] C. Seoighe, N. Federspiel, T. Jones, N. Hansen, V. Bivolarovic, R. Surzycki, R. Tamse, C. Komp, L. Huizar, R. W. Davis, S. Scherer, E. Tait, D. J. Shaw, D. Harris, L. Murphy, K. Oliver, K. Taylor, M. A. Rajandream, B. G. Barrell, and K. H. Wolfe, "Prevalence of small inversions in yeast gene order evolution," *Proceedings of the National Academy of Sciences USA*, vol. 97, no. 26, pp. 14 433–14 437, 2000.

[5] D. A. Dalevi, N. Eriksen, K. Eriksson, and S. G. E. Andersson, "Measuring genome divergence in bacteria: A case study using chlamydian data," *Journal of Molecular Evolution*, vol. 55, no. 1, pp. 24–36, 2002.

[6] J. F. Lefebvre, N. El-Mabrouk, E. Tillier, and D. Sankoff, "Detection and validation of single gene inversions," *Bioinformatics*, vol. 19, no. suppl 1, pp. i190–i196, 2003.

[7] A. E. Darling, I. Miklós, and M. A. Ragan, "Dynamics of genome rearrangement in bacterial populations," *PLoS Genetics*, vol. 4, no. 7, p. e1000128, 2008.

[8] M. R. Jerrum, "The complexity of finding minimum-length generator sequences," *Theoretical Computer Science*, vol. 36, pp. 265–289, 1985.

[9] A. Egri-Nagy, V. Gebhardt, M. M. Tanaka, and A. R. Francis, "Group-theoretic models of the inversion process in bacterial genomes," *Journal of Mathematical Biology*, vol. 69, no. 1, pp. 243–265, 2014.

[10] L. S. Heath and J. P. C. Vergara, "Sorting by short swaps," *Journal of Computational Biology*, vol. 10, no. 5, pp. 775–789, 2003.

[11] G. R. Galvão, O. Lee, and Z. Dias, "Sorting signed permutations by short operations," *Algorithms for Molecular Biology*, vol. 10, no. 12, 2015.

[12] X. Feng, B. Chitturi, and H. Sudborough, "Sorting circular permutations by bounded transpositions," in *Advances in Computational Biology*, ser. Advances in Experimental Medicine and Biology, H. R. Arabnia, Ed. Springer New York, 2010, vol. 680, pp. 725–736.

[13] E. Belda, A. Moya, and F. J. Silva, "Genome rearrangement distances and gene order phylogeny in γ-proteobacteria," *Molecular Biology and Evolution*, vol. 22, no. 6, pp. 1456–1467, 2005.

[14] G. R. Galvão, C. Baudet, and Z. Dias, "Sorting signed circular permutations by super short reversals," in *Bioinformatics Research and Applications*, ser. LNCS, R. Harrison, Y. Li, and I. Mandoiu, Eds. Springer International Publishing, 2015, vol. 9096, pp. 272–283.

[15] J. Meidanis, M. E. M. T. Walter, and Z. Dias, "Reversal distance of signed circular chromosomes," Institute of Computing, University of Campinas, Tech. Rep., 2000.

[16] A. Solomon, P. Sutcliffe, and R. Lister, "Sorting circular permutations by reversal," in *Algorithms and Data Structures*, ser. LNCS, F. Dehne, J.-R. Sack, and M. Smid, Eds. Springer Berlin Heidelberg, 2003, vol. 2748, pp. 319–328.

[17] J. Felsenstein, "PHYLIP – phylogeny inference package (version 3.2)," *Cladistics*, vol. 5, pp. 164–166, 1989.

[18] A. E. Darling, B. Mau, and N. T. Perna, "progressivemauve: Multiple genome alignment with gene gain, loss and rearrangement," *PLoS ONE*, vol. 5, no. 6, p. ee11147, 2010.

[19] K. I. Bos, V. J. Schuenemann, G. B. Golding, H. A. Burbano, N. Waglechner, B. K. Coombes, J. B. McPhee, S. N. DeWitte, M. Meyer, S. Schmedes, J. Wood, D. J. Earn, D. A. Herring, P. Bauer, H. N. Poinar, and J. Krause, "A draft genome of *Yersinia pestis* from victims of the black death," *Nature*, vol. 478, no. 7370, pp. 506–510, 2011.

[20] I. Uchiyama, "MBGD: microbial genome database for comparative analysis," *Nucleic Acids Research*, vol. 31, no. 1, pp. 58–62, 2003.

[21] I. Uchiyama, M. Mihara, H. Nishide, and H. Chiba, "MBGD update 2015: microbial genome database for flexible ortholog analysis utilizing a diverse set of genomic data," *Nucleic Acids Research*, vol. 43, no. D1, pp. D270–D276, 2015.

[22] D. R. Maddison, D. L. Swofford, and W. P. Maddison, "NEXUS: An extensible file format for systematic information," *Systematic Biology*, vol. 46, no. 4, pp. 590–621, 1997.

[23] M. V. Han and C. M. Zmasek, "phyloXML: XML for evolutionary biology and comparative genomics," *BMC Bioinformatics*, vol. 10, no. 356, 2009.

[24] D. Sankoff and M. Goldstein, "Probabilistic models of genome shuffling," *Bulletin of Mathematical Biology*, vol. 51, no. 1, pp. 117–124, 1989.

[25] V. Bafna, D. Beaver, M. Fürer, and P. A. Pevzner, "Circular permutations and genome shuffling," in *Comparative Genomics*, ser. Computational Biology, D. Sankoff and J. H. Nadeau, Eds. Springer Netherlands, 2000, vol. 1, pp. 199–206.

Gustavo Rodrigues Galvão received his B.S. and M.Sc. degrees in Computer Science from the University of Campinas, Brazil, in 2010 and 2012, respectively. He is currently a doctoral student at the same university. His main interests include computational biology and combinatorial optimization.

Christian Baudet received the B.S. degree in Computer Engineering, the M.Sc. degree in Computer Science and the Ph.D. degree in Computer Science from the University of Campinas, Brazil, in 2003, 2006 and 2010, respectively. From January 2011 to April 2015, he was a postdoctoral research at Inria Erable Team at Lyon, France. Currently, he is a member of the Bioinformatics team at the Cancer Research Center of Lyon (Centre Léon Bérard). His research interests include the study of problems related to genome rearrangement, co-phylogeny, and evolution.

Zanoni Dias received his B.S. and Ph.D degrees in Computer Science from the University of Campinas (Unicamp), Brazil, in 1997 and 2002, respectively. He is an associate professor at the Institute of Computing, Unicamp, Brazil. His main interests include computational biology, bioinformatics, and documents phylogeny.