On the Complexity of the Median and Closest Permutation Problems

Luís Felipe I. Cunha, Ignasi Sau, Uéverton dos Santos Souza

On the Complexity of the Median and Closest Permutation Problems

Luís Cunha
Instituto de Computação, Universidade Federal Fluminense, Brasil

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, France

Uéverton Souza
Instituto de Computação, Universidade Federal Fluminense, Brasil

Abstract

Genome rearrangements are events where large blocks of DNA exchange places during evolution. The analysis of these events is a promising tool for understanding evolutionary genomics, providing data for phylogenetic reconstruction based on genome rearrangement measures. Many pairwise rearrangement distances have been proposed, based on finding the minimum number of rearrangement events to transform one genome into the other, using some predefined operation. When more than two genomes are considered, we have the more challenging problem of rearrangement-based phylogeny reconstruction. Given a set of genomes and a distance notion, there are at least two natural ways to define the "target" genome. On the one hand, finding a genome that minimizes the sum of the distances from this to any other, called the *median genome*. On the other hand, finding a genome that minimizes the maximum distance to any other, called the *closest genome*. Considering genomes as permutations of distinct integers, some distance metrics have been extensively studied. We investigate the median and closest problems on permutations over the following metrics: *breakpoint* distance, *swap* distance, *block-interchange* distance, *short-block-move* distance, and *transposition* distance. In biological applications some values are usually very small, such as the solution value d or the number k of input permutations. For each of these metrics and parameters d or k, we analyze the closest and the median problems from the viewpoint of parameterized complexity. We obtain the following results: NP-hardness for finding the median/closest permutation regarding some metrics of distance, even for only $k = 3$ permutations; Polynomial kernels for the problems of finding the median permutation of all studied metrics, considering the target distance d as parameter; NP-hardness result for finding the closest permutation by short-block-moves; FPT algorithms and infeasibility of polynomial kernels for finding the closest permutation for some metrics when parameterized by the target distance d.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Permutations and combinations

Keywords and phrases Median problem, Closest problem, Genome rearrangements, Parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.WABI.2024.2

Related Version *Full Version*: http://arxiv.org/abs/2311.17224

Funding *Luís Cunha*: FAPERJ-JCNE (E-26/201.372/2022), and CNPq-Universal (406173/2021-4).
Ignasi Sau: project ELIT (ANR-20-CE48-0008-01), and CAPES/PRINT Programa Institucional de Internacionalização, edital nº 41/2017, grant 88887.717401/2022-00.
Uéverton Souza: FAPERJ-JCNE (E-26/201.344/2021), and CNPq (309832/2020-9).

24th International Workshop on Algorithms in Bioinformatics (WABI 2024).
Editors: Solon P. Pissis and Wing-Kin Sung; Article No. 2; pp. 2:1–2:23
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Ancestral reconstruction is a classic task in comparative genomics, which is based on consensus word analysis, with a vast applicability [11, 17, 29]. In this field, genome rearrangement problems study large-scale mutations on a set of DNAs in living organisms, and have been studied extensively in computational biology and computer science for decades. From a mathematical point of view, a genome is represented by a permutation (a sequence of distinct integers). Based on that, as proposed by Watterson et al. [33], a genome rearrangement problem is interpreted as transforming one permutation into another by a minimum number of operations depending on the possible allowed rearrangements, i.e., the chosen metric. In this model we consider the following assumptions: the order of genes in each genome is known; all genomes we compare share the same gene set; all genomes contain a single copy of each gene; and all genomes consist of a single chromosome. So, genomes can be modeled as permutations, once each gene is encoded as an integer.

Finding the minimum number of operations is equivalent to sorting the permutation with a given rearrangement. Many metrics received attention in recent years, and among the studied distances or sorting problems the following are the most natural ones. The *breakpoint* distance is the number of consecutive elements in one permutation that are not consecutive in another one. Note that on the breakpoint distance we do not apply any operation to transform a permutation into another one. The *reversal* operation transforms one permutation into another one by reversing the elements of a block of one permutation (a *block* is an interval of consecutive elements). The *block-interchange* operation exchanges two disjoint blocks, and generalizes the *transposition*, where the blocks are restricted to be consecutive. A *swap* is a block-interchange where each block has a unique element. A *short-block-move* is a transposition whose blocks have at most three elements. Concerning the computational complexity with respect to these metrics, the corresponding problems for breakpoint distance, sorting by block-interchanges, and sorting by swap can be solved in polynomial time [9, 17], sorting by transpositions and sorting by reversals are NP-complete [6, 7], and the complexity of sorting by short-block-moves is still unknown. Some restrictions or generalizations of the presented metrics have been considered and algorithmic aspects have been developed [27, 32].

When an input has more than two genomes, there are many approaches for finding ancestral genomes. The main application is to infer common ancestor configurations and eventually phylogenies, which are trees that show the relationships between organisms or between species [1, 8, 10, 14, 21, 30]. A relevant approach is the MEDIAN problem, where, for a fixed metric M, the goal is to find a solution genome that minimizes the sum of the distances between the solution and all the input genomes.

METRIC M MEDIAN

Instance: A set S of genomes.

Goal: A genome that minimizes the sum of the distances, according to metric M, between the solution genome and all other genomes of S.

The BREAKPOINT MEDIAN problem is NP-hard [30] for a general input. The REVERSAL MEDIAN and TRANSPOSITION MEDIAN problems are NP-hard even when the input consists of three permutations [1, 8]. Prior to this work, the complexity of BLOCK-INTERCHANGE MEDIAN was not known. The same applies to the MEDIAN problem regarding swap or short-block-move operations.

Haghighi and Sankoff [21] observed that, with respect to the breakpoint metric, a tendency for medians is to fall on or to be close to one of the input genomes, which contain no useful information for the phylogeny reconstruction. They also conjectured the same behavior concerning other metrics. Hence, an alternative approach is to consider the Closest problem for a fixed metric M, which aims to find a genome that minimizes the *maximum* distance to any genome in the input, which can be seen as finding a genome in the center of all others, i.e., a genome minimizing the radius of the ball containing all the genomes of the input set.

Metric M Closest

Instance: A set S of genomes.

Goal: A genome that minimizes the maximum distance, according to metric M, between the solution genome and any other genome in S.

Lanctot et al. [28] studied the Closest problem over strings with respect to the Hamming distance, and settled that this problem is NP-hard even for binary strings. Popov [31] studied the Closest problem over permutations with respect to the swap operation distance, and showed that it is NP-hard. Cunha et al. [10] showed that the Closest problem is NP-hard for several well-known genome rearrangement distances, such as the breakpoint and the block-interchange ones.

The parameterized complexity (see Appendix A for the basic definitions, which can also be found in [15]) of the Median and Closest problems has been studied mostly regarding strings on an alphabet Σ with respect to the Hamming distance. These problems, and some variations, have been considered with respect to parameters that are combinations of k, d, $|\Sigma|$, and n, where d is the solution value, k is the number of input permutations, and n is the length of the strings. Gramm et al. [19, 20] investigated the Closest String problem on binary strings considering some parameters, and showed how to solve it in linear time for fixed d (the exponential growth in d is bounded by $O(d^d)$), and when k is fixed and d is arbitrary. Fu et al. [18] developed a polynomial kernelization parameterized by d and k, of size $O(k^2 d \log k)$. Basavaraju et al. [3] presented a comprehensive study of Closest String and some related problems from the kernelization complexity perspective, and showed that Closest String parameterized by d and the length of the strings n does not admit a polynomial kernel under a standard complexity assumption. Furthermore, recently parameterized results regarding Median and Closest problems with respect to edit distance were developed [25].

Considering the input genomes as permutations, some few results are known, such as the fact that the Swap Closest problem is FPT when parameterized by the number of input permutations and the solution radius [31]. On the other hand, the Transposition Median problem parameterized by the number of input permutations is para-NP-hard, since Bader proved that it is NP-hard even if the input consists of three permutations [1]. To the best of our knowledge, a multivariate investigation of the parameterized complexity of computing the median/closest genome by the considered metrics on permutations has not been thoroughly studied in the literature. Therefore, our goal is to map sources of computational tractability for both consensus problems (Median and Closest) defined above, and consequently identify features that make it tractable through the lenses of metrics over permutations and the parameterized complexity.

Our contribution. In this article we obtain the following results:

- In Section 3, we develop polynomial kernels for finding median permutations considering swap, breakpoint, block-interchange, transposition, and short-block-move operations, all of them parameterized by the target distance d. This result is in sharp contrast with the fact that, as we have also managed to prove, for most of the above metrics the problem of finding the closest permutation does not admit a polynomial kernel parameterized by d.

- In Section 3, we prove the NP-hardness of Block-interchange Median, even for only $k = 3$ permutations. Based on that, in Section 4 we are able to reduce Block-interchange Median to Block-interchange Closest, as well to Transposition Closest, even for only $k = 3$ permutations.
- In Section 4, we prove the NP-hardness of Short-block-moves Closest. Since it is still an open question to decide whether the sorting problem by short-block-moves can be solved in polynomial time, it is natural to consider the "closest" version of the problem that, somehow surprisingly, had not been considered in related previous work [10].
- In Section 4, we also provide FPT algorithms for the Closest problem parameterized by the target distance d, for some of the above metrics. Our approach is inspired from FPT algorithms for Closest String (see [15]).

The above results provide an accurate picture of the parameterized complexity of the considered problems with respect to the parameters k (in particular, for $k = 3$) and d. Note that in biological applications some of these values are very small [3, 19, 25]. Table 1 summarizes our results, considering only d as the parameter, and the open questions.

Table 1 Some results obtained in this paper comparing Median (Theorem 14) and Closest (Corollary 24 and Theorem 25) problems parameterized by d. Open questions are denoted by "??".

	Swap	Block-interchange	Short-Block-Moves	Transposition	Breakpoint
Median Par. d	poly kernel	poly kernel	poly kernel	poly kernel	poly kernel
Closest Par. d	FPT no poly kernel	FPT no poly kernel	FPT no poly kernel	?? no poly kernel	?? no poly kernel

Organization. In Section 2 we provide a detailed explanation on rearrangement operations, associated graphs, and bounds on the distances we deal with. In Section 3 (resp. Section 4) we present our results for the Median (resp. Closest) problems, and detailed proofs from Section 3 can be found in Appendix B. For the sake of readability, the results whose proofs are in the appendices are marked with "($\star$)".

2 Preliminaries on genome rearrangements

Genome rearrangements are events where large blocks of DNA exchange places during evolution. For models, we may consider genomes as strings or permutations. An *alphabet* Σ is a nonempty set of letters, and a *string* over Σ is a finite sequence of letters of Σ. The *Hamming distance of two strings* s and s' of the same length, denoted by $d_H(s, s')$, is the number of mismatched positions between s and s'. The *Hamming distance of a string* s of length n, denoted by $d_H(s)$, is the Hamming distance of s and the string $0 \ldots 0$, where $0 \in \Sigma$.

A *permutation* of length n is a bijection from the set $\{1, 2, \ldots, n\}$ onto itself $\pi = [\boldsymbol{\pi(0)}\, \pi(1)\, \pi(2)\, \ldots\, \pi(n) \boldsymbol{\pi(n+1)}]$, such that $\boldsymbol{\pi(0)} = 0$ and $\boldsymbol{\pi(n+1)} = n+1$. For simplicity $\pi = [\boldsymbol{\pi(0)}\, \pi(1)\, \pi(2)\, \ldots\, \pi(n) \boldsymbol{\pi(n+1)}] = [\boldsymbol{\pi_0}\, \pi_1\, \pi_2\, \ldots\, \pi_n \boldsymbol{\pi_{n+1}}]$. The operations will never act on $\boldsymbol{\pi_0}$ or $\boldsymbol{\pi_{n+1}}$, but these are used to define graphs useful for determining bounds on some distances, as discussed later. When not needed, $\pi = [\pi_1\, \pi_2\, \ldots\, \pi_n]$. Similarly to the above, given a metric M and two permutations π and σ of the same length, we define $d_M(\pi, \sigma)$ as their distance with respect to metric M, and the *distance of a permutation* π of length n, denoted by $d_M(\pi)$, is the distance between π and the *identity permutation* $\iota = [\mathbf{0}\ 1\ 2\ \ldots\ n\ \boldsymbol{n}\ +\ \mathbf{1}]$.

2.1 Sorting by rearrangement operations

Note that the distance between two permutations can also be seen as the sorting of a permutation. Indeed, once permutations π and σ are given, one can relabel σ to be equal to $\sigma\sigma^{-1} = \iota$, and then the distance between π and σ is the same as sorting $\pi\sigma^{-1}$, i.e., the distance between $\pi\sigma^{-1}$ and ι. Hence, throughout this paper we may use interchangeably distance or sorting problem. Block-interchange generalizes a transposition and generalizes also a swap operation. Nevertheless, with respect to the distance problem, if we are dealing with an operation M that generalizes another M', if computing the distance by the metric M is in P, it does not imply that with respect to the metric M', the distance problem is also in P. For instance, concerning the block-interchange distance, it can be computed in polynomial time [9], whereas computing the transposition distance is NP-hard [6], and computing the swap distance is in P, as discussed later. On the other hand, if a distance problem is NP-hard, then the Median/Closest problems for the same operation are also NP-hard. Indeed, this follows by considering an input set of permutations consisting of two permutations π, ι such that $\pi \neq \iota$, and asking for a permutation with distance at most d from each, for a metric M. Then, we can consider the problem of computing the distance as a particular case of the Closest problem.

The breakpoint distance. An *adjacency of a permutation π with respect to permutation σ* is a pair (π_i, π_{i+1}) of consecutive elements in π such that this pair is also consecutive in σ, i.e., $\pi_i = \sigma_j$ and $\pi_{i+1} = \sigma_{j+1}$. If a pair of consecutive elements is not an adjacency, then it is called a *breakpoint*, and we denote by $d_{\mathsf{BP}}(\pi, \sigma)$ the number of breakpoints of π with respect to σ. The set $\mathsf{Adj}(\pi)$ is the set of *adjacencies of* π, given by $\mathsf{Adj}(\pi) = \{\{\pi_i, \pi_{i+1}\} \mid i = 1, \ldots, n-1\}$. Thus, in other words, the breakpoint distance between π and σ is $d_{\mathsf{BP}}(\pi, \sigma) = |\mathsf{Adj}(\pi) - \mathsf{Adj}(\sigma)|$.

The block-interchange and the transposition distances. Bafna and Pevzner [2] proposed a useful graph, called the reality and desire diagram, which allowed to obtain non-trivial bounds on the transposition distance [2], and also provided, as established by Christie [9], the exact block-interchange distance. Nevertheless, when considering the transposition distance, the reality and desire diagram is a tool to only deal with lower and upper bounds for a permutation, as discussed below.

Given a permutation π of length n, the *reality and desire diagram* $G(\pi, \iota)$ (or just $G(\pi)$ when convenient) from π to ι, is a multigraph $G(\pi) = (V, R \cup D)$, where $V = \{0, -1, +1, -2, +2, \ldots, -n, +n, -(n+1)\}$, each element of π corresponds to two vertices and we also include the vertices labeled by 0 and $-(n+1)$, and the edges are partitioned into two sets: the reality edges R and the desire edges D. The *reality edges* represent the adjacency between the elements on π, that is $R = \{(+\pi_i,\ -\pi_{i+1}) \mid i = 1, \ldots, n-1\} \cup \{(0,\ -\pi_1),\ (+\pi_n,\ -(n+1))\}$; and the *desire edges* represent the adjacency between the elements on ι, that is $D = \{(+i,\ -(i+1)) \mid i = 0, \ldots, n\}$. Figure 1 illustrates the reality and desire diagram of a permutation. A general definition considers $G(\pi, \sigma)$ where the reality (resp. desire) edges represent the adjacency between elements of π (resp. σ), and then $D = \{(+\sigma_i,\ -\sigma_{i+1}) \mid i = 0, \ldots, n\}$.

As a direct consequence of the construction of this graph, the sets of reality edges and desire edges define two perfect matchings (that is, a set of edges that contains all vertices of the graph and each of them appears exactly once), denoted by $M(\pi)$ and $M(\iota)$, respectively. Each of these perfect matchings is called a *permutation matching*.

Since every vertex in $G(\pi)$ has degree two, $G(\pi)$ can be partitioned into disjoint cycles. We say that a cycle in π has *length* k, or that it is a *k-cycle*, if it has exactly k reality edges (or, equivalently, k desire edges). Hence, the identity permutation of length n has $n+1$ cycles

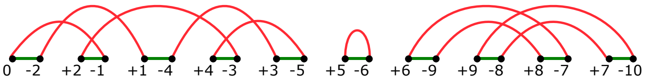


Figure 1 The reality and desire diagram between permutation [**0** 2 1 4 3 5 6 9 8 7 **10**] and ι, where green edges are the reality edges and red edges are the desire edges.

of length one. We denote by $c(G(\pi,\iota))$ (or just $c(G(\pi))$ for convenience) the number of cycles in $G(\pi)$. After applying a block-interchange $b\ell$ in a permutation π, the number of cycles $c(G(\pi))$ changes in such a way that $c(G(\pi b\ell)) = c(G(\pi)) + x$, for some $x \in \{-2, 0, 2\}$ (see [9]). The block-interchange $b\ell$ is thus classified as an *x-move* for π. Analogously, after applying a transposition t in a permutation π, the number of odd cycles, denoted by $c_{\mathsf{odd}}(G(\pi))$, changes by an *x-move*, $x \in \{-2, 0, 2\}$, for π (see [2]). Christie [9] proved, for the block-interchange operation, the existence of a 2-move for any permutation, which says that the number of cycles yields the exact block-interchange distance:

▶ **Theorem 1** (Christie [9]). *The block-interchange distance of a permutation π of length n is $d_{\mathsf{BI}}(\pi) = \frac{(n+1)-c(G(\pi))}{2}$.*

On the other hand, by allowing only the particular case of the transposition operation, it is not always possible to use a 2-move. We say that a transposition *affects* a cycle if the extremities of the two blocks of the transposition eliminate a reality edge of a cycle and create another edge. This new edge may increase, decrease, or preserve the number of cycles.

A *transposition* $t(i, j, k)$, where $1 \leq i < j < k \leq n+1$, is a permutation that exchanges the contiguous blocks $i\ i{+}1 \ldots j{-}1$ and $j\ j{+}1 \ldots k{-}1$; when composed with a permutation π, it yields the following permutation: $\pi \cdot t(i, j, k) = [\pi_1\ \pi_2\ \ldots \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \ldots \pi_{j-1}}\ \pi_k \ldots \pi_n]$. Bafna and Pevzner [2] showed conditions of a cycle for a transposition to be an x-move. If a transposition t is a -2-move, then t affects three distinct cycles. However, if a transposition t is a 0-move or a 2-move, then t affects at least two elements of the same cycle [2].

▶ **Theorem 2** (Bafna and Pevzner [2]). *The transposition distance of a permutation π of length n satisfies $d_{\mathsf{T}}(\pi) \geq \frac{(n+1)-c_{\mathsf{odd}}(G(\pi))}{2}$.*

Permutations whose transposition distances are equal to the lower bound of Theorem 2 are called *hurdle-free permutations* [1,9]. Cunha et al. [12,13] presented upper bounds on the distance of any permutation by using permutation trees data structure (cf. [13]), and based on that, an 1.375-approximation algorithm for Sorting by Transpositions was developed, improving the time complexity to $O(n \log n)$. An interesting transformation on permutations is the reduction operation, since the permutation obtained after its reduction preserves both the block-interchange and the transposition distances. The *reduced permutation* of π, denoted by $\mathsf{gl}(\pi)$ (also called as a *glued* operation), is the permutation whose reality and desire diagram $G(\mathsf{gl}(\pi))$ is equal to $G(\pi)$ without the cycles of length one (recall that the length of a cycle is its number of reality edges), and has its vertices relabeled accordingly. For instance, the reduced permutation corresponding to the permutation in Figure 1 is [**0** 2 1 4 3 5 8 7 6 **9**].

▶ **Theorem 3** (Christie [9]). *The block-interchange distances of a permutation π and its reduced permutation $\mathsf{gl}(\pi)$ satisfy $d_{\mathsf{BI}}(\pi) = d_{\mathsf{BI}}(\mathsf{gl}(\pi))$.*

Swap distance. Permutations can also be represented by each element followed by its image. For example, given a set $\{1, 2, 3\}$, the sequence (1 2 3) maps 1 into 2, 2 into 3, and 3 into 1, corresponding to the permutation [2 3 1]. This representation is not unique; for instance, (2 3 1) and (3 1 2) are equivalent. Permutations are composed of one or more algebraic cycles, where each algebraic cycle of a permutation π is formed by an element i, followed by its image $\pi(i)$, followed by the image of $\pi(i)$, i.e., $\pi(\pi(i))$, and so on. We continue this process until we reach a repeated element. This procedure uniquely defines the permutation. We denote by $c(\pi)$ the number of algebraic cycles of π. For example, given $\pi = [8\ 5\ 1\ 3\ 2\ 7\ 6\ 4] = (1\ 8\ 4\ 3)(2\ 5)(6\ 7)$, we have $c(\pi) = 3$. An exchange of elements involving elements a and b such that a and b are in the same cycle is an exchange that *breaks* the cycle into two, whereas if a and b belong to different cycles, the exchange of these elements *unites* the two cycles [17]. Thus, when considering the *swap* metric, the *swap* distance of a permutation π is determined as follows: $d_{\mathsf{swap}}(\pi) = n - c(\pi)$, where $c(\pi)$ is the number of algebraic cycles of π and n is the length of π.

Short-block-move distance. A *p-bounded block-move* is a transposition $t(i, j, k)$ such that $k - i \leq p$, and a 3-bounded block-move is called a *short-block-move*. Hence, a short-block-move is either a transposition $t(i, i+1, i+2)$, called a *skip*, a transposition $t(i, i+1, i+3)$, or a transposition $t(i, i+2, i+3)$, the two latter ones called *hops*. If the transpositions are restricted only to p-bounded block-moves, then one obtains the *p-bounded block-move distance*, denoted by $d_{\mathsf{pbbm}}(\pi)$. When $p = 3$, this defines the *short-block-move distance*, denoted by $d_{\mathsf{sbm}}(\pi)$. Previous works investigated variants of block-move distances where bounds are imposed on the lengths of at least one of the moved blocks [22,23]. The problem of sorting permutations using 2-bounded block-moves, i.e., adjacent swaps, is easily solved by the *Bubble-Sort* algorithm [26]. In general, the complexity of sorting a permutation by p-bounded block-moves is unknown for fixed $p > 2$, whereas the analogous problem of limiting $k - i \leq f(n)$, is NP-hard [22], since Sorting by Transpositions is NP-hard.

To estimate the short-block-move distance, Heath and Vergara [22,23] used the *permutation graph* $PG(\pi) = (V^p_\pi, E^p_\pi)$, where $V^p_\pi = \{1, 2, \ldots, n\}$ and $E^p_\pi = \{(i, j) \mid \pi_i > \pi_j,\ i < j\}$; each edge of PG is called an *inversion* in π. Heath and Vergara proved that on a shortest sequence of operations for π, every short-block-move decreases the number of inversions by at least one unit, and by at most two units, therefore: $\left\lceil \frac{|E^p_\pi|}{2} \right\rceil \leq d_{\mathsf{sbm}}(\pi) \leq |E^p_\pi|$. Given a permutation, our aim is to minimize the number of operations that decrease only one inversion in PG. Examples of permutations that are tight with respect to the above lower and upper bounds are [2 4 3 5 1] and [2 1 4 3 6 5], respectively.

A short-block-move is a *correcting* move if it is a skip that eliminates one inversion, or a hop that eliminates two inversions in π. Otherwise, the block-move is called *non-correcting*. Heath and Vergara [23] proved that each sorting sequence can be performed by using just correcting moves. Table 2 shows replacements from non-correcting moves to correcting moves in an optimal sorting sequence, which we will use later in Lemma 20.

Table 2 How to replace a non-correcting move β_i with a correcting move β'_i [23]; in all cases, $e < f$, and x is arbitrary. Case 1 is an exception in this discussion, since it is the case where β_i is a skip, so that it suffices to simply omit β_i instead of replacing it with some β'_i.

Case	π	$\pi' = \pi\beta_i$	$\pi'' = \pi\beta'_i$
1	$\ldots ef \ldots$	$\ldots fe \ldots$	$\ldots ef \ldots$
2	$\ldots exf \ldots$	$\ldots xfe \ldots$	$\ldots xef \ldots$
3	$\ldots exf \ldots$	$\ldots fex \ldots$	$\ldots efx \ldots$
4	$\ldots xef \ldots$	$\ldots fxe \ldots$	$\ldots exf \ldots$
5	$\ldots efx \ldots$	$\ldots fxe \ldots$	$\ldots exf \ldots$

2.2 Relationship between sorting and median/closest problems

Median problems. Caprara [8] proved that the Reversal Median problem (RM) is NP-complete by the following strategy. It begins with the Eulerian Cycle Decomposition problem (ECD), which consists in, given an Eulerian graph, find a partition of its edges into the maximum number of cycles. The ECD problem was proved to be NP-complete by Holyer [24]. First, Caprara reduced ECD to Alternating Cycle Decomposition (ACD), which is the problem of finding a maximum cycle decomposition of a reality and desire diagram (defined in Subsection 2.1). Then, he reduced ACD to Cycle Median (CM), which is the problem of finding a permutation that maximizes the sum of the number of cycles in the reality and desire diagram of a given set of three permutations. Finally, Caprara reduced CM to Reversal Median. Summarizing, he proved ECD $\leq_{\mathsf{p}}$ CM $\leq_{\mathsf{p}}$ RM.

In 2011, just before it was proved that Sorting by Transpositions is NP-hard [6], Bader [1] proved the NP-completeness of the Transposition Median problem (TM for short). It was based on the following definition: given three input permutations π^1, π^2, π^3, find a permutation σ such that $\sum_{i=1}^{3} d_{\mathsf{T}}(\sigma, \pi^i)$ is minimized, where $d_{\mathsf{T}}(\sigma, \pi^i)$ is the transposition distance between σ and π^i.

Bader [1] proved the hardness as an adaptation of Caprara's reductions considering reversals. This adaptation was done by reducing 3SAT $\leq_{\mathsf{p}}$ MDECD $\leq_{\mathsf{p}}$ OCM $\leq_{\mathsf{p}}$ TM, where MDECD is the Marked Directed Eulerian Cycle Decomposition problem, proved NP-hard by Bader [1] and defined as follows. Let k be an integer, let $G = (V, E)$ be a directed graph, and let $E_k \subseteq E$ be a subset of its edges with $|E_k| = k$. The edges in E_k are called the marked edges of G. $(G, E_k) \in$ MDECD if and only if $E(G)$ can be partitioned into edge-disjoint cycles such that each marked edge is in a different cycle. OCM denotes the Odd Cycle Median problem defined as follows. Let π^1, π^2, π^3 be permutations of $\{1, \ldots, n\}$ and let k be an integer. Then, $(\pi^1, \pi^2, \pi^3, k) \in$ OCM if and only if there is a permutation σ with $\sum_{i=1}^{3} c_{\mathsf{odd}}(G(\sigma, \pi^i)) \geq k$ (because it is known that $d_{\mathsf{T}}(\pi) \geq \frac{(n+1) - c_{\mathsf{odd}}(G(\pi))}{2}$, where $c_{\mathsf{odd}}(G(\pi))$ is the number of odd cycles in the reality and desire diagram of π, see Theorem 2). Solving an OCM instance is equivalent to finding a permutation matching $M(\sigma)$ such that $\sum_{i=1}^{3} c_{\mathsf{odd}}(G(\sigma, \pi^i))$ is maximized. This sum is also called the solution value of $M(\sigma)$. TM was proved to be NP-hard by a transformation from any instance σ' that maximizes $\sum_{i=1}^{3} c_{\mathsf{odd}}(G(\sigma', \pi^i))$ to an instance that minimizes $\sum_{i=1}^{3} d_{\mathsf{T}}(\sigma', \pi^i)$. This could be done by ensuring that the distance between σ' and each π^i achieves the lower bound of Theorem 2.

In order to examine TM, the *multiple reality and desire diagram*[1] was used in [1,8]. Given the permutations $\pi^1, \ldots, \pi^q$ each one with length n, the multiple reality and desire diagram $MG(\pi^1, \ldots, \pi^q) = (V, E)$ is a multigraph with $V = \{0, -1, +1, \ldots, -n, +n, -(n+1)\}$ and $E = M(\pi^1) \cup \ldots \cup M(\pi^q)$, i.e., the edge set is formed by the union of all permutation matchings of the permutations.

MDECD is NP-hard even when the degree of all nodes is bounded by four. Furthermore, this result still holds for graphs $G = (V, E)$ that $|V| + |E| - k$ is odd, where k is the number of marked edges. Based on that, Bader described a polynomial transformation from G being an instance of MDECD to an MG. Hence, it is necessary to guarantee conditions on graphs to be a multiple reality and desire diagram MG. To this end, we have the following properties.

[1] Also called *multiple breakpoint graph* in [1], but not called in this way here so as not to create confusion with breakpoint distances we also deal with.

▶ **Lemma 4** (Caprara [8]). *Let V^t and V^h be two disjoint node sets, and let $G' = (V^t \cup V^h, M^1 \cup \ldots \cup M^q)$ be a graph, where each M^i is a perfect matching, each edge in M^i has color i, and each edge connects a node in V^t with a node in V^h. Furthermore, let H be a perfect matching such that each edge in H connects a node in V^t with a node in V^h, and $H \cup M^i$ defines a Hamiltonian cycle of G' for $1 \leq i \leq q$. Then, there exist permutations $\pi^1, \ldots, \pi^q$ such that G' is isomorphic to $MG(\pi^1, \ldots, \pi^q)$.*

A matching H as described in Lemma 4 is called a *base matching* of the graph. An important operation over MG was introduced in [8]. Given a perfect matching M on a node set V and an edge $e = (u, v)$, the operation M/e is defined as follows. If $e \in M$, then $M/e = M \backslash \{e\}$. If $e \notin M$, and $(a, u), (b, v)$ are the two edges in M incident to u and v, then $M/e = M \backslash \{(a, u), (b, v)\} \cup \{(a, b)\}$.

▶ **Lemma 5** (Caprara [8]). *Given two perfect matchings M, L of a given graph G and an edge $e = (u, v) \in M$ with $e \notin L$, $M \cup L$ defines a Hamiltonian cycle of G if and only if $(M/e) \cup (L/e)$ defines a Hamiltonian cycle of $G - \{u, v\}$.*

Given an MG graph $G = (V, M(\pi^1) \cup \ldots \cup M(\pi^q))$, the *contraction* of an edge $e = (u, v)$ yields the graph $G/e = (V \backslash \{u, v\}, M(\pi^1)/e \cup \ldots \cup M(\pi^q)/e)$. By induction on the node size and contracting merging cycles, as a consequence of Lemma 4 we have the following result.

▶ **Lemma 6** (Bader [1]). *Let V^t and V^h be two disjoint sets, and let $G = (V^t \cup V^h, M^1 \cup M^2)$ be a graph where M^1 and M^2 are disjoint perfect matchings where each edge connects a node in V^t with a node in V^h. If $M^1 \cup M^2$ defines an even number of even cycles on V, then G has a base matching H.*

Closest problems. We start with the following result.

▶ **Theorem 7** (Basavaraju et al. [3]). *Closest String parameterized by d and n does not admit a polynomial kernel unless* NP ⊆ coNP/poly.

Considering Closest problems, Popov [31] proved the NP-completeness of the Swap Closest problem, and Cunha et al. [10] developed an NP-completeness framework of closest permutation regarding some rearrangements, such as breakpoint and block-interchange. The proposed reduction was the following: by considering any set of k strings of length n, obtain a particular set of k permutations of length $f(n)$, which is $f(n) = 4n$ for the breakpoint case, while $f(n) = 2n$ for the block-interchange case. Based on that transformation, Cunha et al. [10] showed a polynomial transformation where a solution for Closest string yields a solution for Closest permutation, as follows.

▶ **Lemma 8** (Cunha et al. [10]). *Given a set of k permutations obtained by the transformed set of binary strings, there is a breakpoint closest permutation with maximum distance equal to $2d$ if and only if there is a Hamming closest string with maximum distance equal to d.*

▶ **Lemma 9** (Cunha et al. [10]). *Given a set of k permutations obtained by the transformed set of binary strings, there is a block-interchange closest permutation with maximum distance at most d if and only if there is a Hamming closest string with maximum distance equal to d.*

Hence, the developed technique is a polynomial parameter transformation (PPT, as defined in Appendix A) from Closest String to Breakpoint Closest Permutation and to Block-interchange Closest Permutation.

3 Results for the Median problems

Given a set of k permutations, each of length n, we can store these permutations as a $k \times n$ matrix. The columns of this matrix are called the *columns* of the set of permutations, which are the elements in a same position over the k permutations. For convenience, we denote by S the input matrix and by $s \in S$ a permutation of the instance.

First, we show that Block-interchange Median (BIM for short) is NP-complete even if the input consists of only three permutations. The proof follows a similar approach considered by Caprara [8] for the reversal rearrangement.

▶ **Theorem 10** ($\star$). *The* Block-interchange Median *problem is* NP*-complete even when the input consists of three permutations.*

From Theorem 10, one can conclude that BIM is para-NP-hard when parameterized by the number of input permutations. Transposition Median and Breakpoint Median are also known to be para-NP-hard when parameterized by the number of input permutations [1, 30].

Lemma 11, Lemma 12, and Lemma 13 present useful conditions to reduce the size of the input permutations in order to obtain polynomial kernels (Theorem 14).

▶ **Lemma 11** ($\star$). *If an adjacency occurs in all of the input permutations, it occurs also in a solution of the* Block-interchange Median *and the* Transposition Median *problems.*

The argument of Lemma 11 does not hold when dealing with short-block-moves, because a simulation operation (i.e., an operation that must be applied in non-reduced permutations analogous to the reduced ones) may be affected when it exceeds the size of a block. Nevertheless, an analogous result is proved in Lemma 12. We define the d-M Median problem as the Median problem for a metric M parameterized by the sum d of the distances between the solution and all the input instances, i.e., the median solution.

▶ **Lemma 12.** *For d-*Short-Block-Move Median*, let I be an interval with $6d + 1$ consecutive columns where in each column of I all elements are equal, and let the middle column be with element c, i.e., the $(3d + 1)$th column of I has only element c. Then there is a median solution $s^\star$ that satisfies following properties:*

1. *Element c occurs in $s^\star$ in the same position as in the input permutations, i.e., c occurs in the $(3d + 1)$th column of $s^\star$, which is the same column of I.*
2. *For any element e that occurs before I in the input, e does not occur after the $(3d + 1)$th column of I in $s^\star$.*
3. *All elements of the input that occur in I and take place before (resp. after) the $(3d + 1)$th column of I also occur in $s^\star$ before (resp. after) the $(3d + 1)$th column.*

Proof. For the first statement, assume that s is a median solution in which the column ℓ containing c in the input contains a in such a position and c is in a position j. Since I contains $6d + 1$ columns, with respect to the input permutations, a must also be in I, otherwise at least $3d/2 > d$ moves would have to be applied, and so we could safely conclude that we are dealing with a no-instance. For each input permutation π^i, assume a is at a position p_i^a. Hence, the short-block-move distance between s and each π^i must use at least $|p_i^a - \ell|/2$ operations to move a to column ℓ, plus $|\ell - j|/2$ operations to move c to column j. In any case, $\ell < j < p_i^a$, $p_i^a < \ell < j$, or $j < \ell < p_i^a$. Based on that, we can transform s into $s^\star$ by keeping c in position ℓ and a in position j, applying the same number of short-block-moves as before, once it is necessary to apply at least $|j - p_i^a|/2$ operations.

For the second statement, note that if there is a permutation where the element e comes after I, then it is necessary to apply at least $(6d+1)/2 > d$ operations, which contradicts the existence of a median solution for d-Short-Block-Move Median.

Now, let us consider the third statement. Let A be the interval inside I before the column of c's, i.e., before the $(3d+1)$th column, and B the interval inside I after the column of c's. As proved in the previous statement, a median solution does not have an element e that occurs in the input before I and in the solution it is after I. Let us consider that in $s^\star$ an element of $x \in A$ occurs in B. By the pigeonhole principle, one of the elements of B must occur outside of B. Let us consider the following cases:

Case 1) An element of B occurs after B in $s^\star$. In this case, let us consider that x moves from A to B and it takes place where $y \in B$ was. Hence, y moves to another place and, also by the pigeonhole principle, some element of B takes place after B in $s^\star$. Since there are $3d$ elements in B, more than d operations were necessary in total, similar to the previous second statement, which contradicts a median solution for d-Short-Block-Move Median (see Figure 2(a));

Case 2) An element of B occurs inside A in $s^\star$. In this case, with respect to an element of A (resp. of B) of the input that occurs in B (resp. in A) in $s^\star$, by the pigeonhole principle, there must be a cycle $\mathcal{C}$ according to the moves necessary to be applied between the positions that the elements change their positions among to $s^\star$ and the input permutations. Thus, we can transform $s^\star$ into s' by keeping all elements of A (resp. of B) in A (resp. in B), since in $s^\star$ there must be moves following $\mathcal{C}$ to correct the elements according to the input columns (see Figure 2(b)). ◀

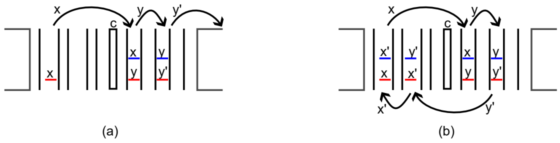


Figure 2 Cases of the third statement of Lemma 12. In red are elements before their moves to other positions and in blue elements in $s^\star$ (after the moves). (a) Case 1, where element x of A takes place in B and element y' of B takes place after B. (b) It represents whenever an element of A takes place in B. Arrows follow a cycle $\mathcal{C}$ representing new positions of the elements. Element x moves to the position where there was y, element y moves to the position where there was y', and so on.

The complexity of the Swap Median problem is still open even if the number of input strings is three. Bryant [5] proved that some variations of the Breakpoint Median problem are NP-hard having three input permutations, by dealing with the cases of linear, circular, signed, or unsigned permutations. One condition of a breakpoint median solution for given three input permutations is that if there are adjacencies common to the three input permutations, then these adjacencies can be assumed to be in a median genome [5]. This result can be directly generalized, analogous to Lemma 11, as follows.

▶ **Lemma 13** (⋆). *If an adjacency occurs in all of the input permutations, then it occurs also in a solution of the* Breakpoint Median *problem.*

Next, we consider the parameterized complexity of some median problems parameterized by the distance d. The previous lemmas allow us to develop reduction rules in order to obtain Theorem 14.

▶ **Theorem 14** (⋆)**.** *The following problems admit a polynomial kernel parameterized by the value d of the desired median solution: d-*Swap Median*, d-*Breakpoint Median*, d-*Block-interchange Median*, d-*Transposition Median*, and d-*Short-Block-Move Median*.*

To conclude this section, to give an insight on the proof of Theorem 14, we present the description of the kernel for the d-Swap Median problem, the other ones being based on slight variations of it. The kernel consists on the application of the following rules:

1. If there is a column with more than $d + 1$ elements, return `no`.
2. If there is a column with at least two elements occurring at least $d + 1$ times, return `no`.
3. If an element occurs more than d times in at least two columns, return `no`.
4. If a row has at least d copies in the matrix, either the solution is a copy of such a row, or the answer is `no`.

   We say that a column i is *heavy* for an element x if x occurs more than d times in it; otherwise, i is said *light* for x.
5. For each element x if the sum of occurrences of x in its light columns is more than $2d$ then the answer is `no`.
6. If the previous rules were not applied, remove the columns whose all elements are the same (and reduce the universe size accordingly).

After proving that Rules 1-5 are safe, to complete the proof it just remains to show (cf. Lemma 29) that if S' is a `yes`-instance of the d-Swap Median problem for which Rules 1-5 cannot be applied and that has been reduced according to Rule 6, then S' has at most $2d$ columns and $4d^2 + d$ rows. See Appendix B for all the details.

4 Results for the Closest problems

First, we present a framework transformation from the median to the closest problem. Since BIM is NP-hard even for three input permutations, we show that Block-Interchange Closest (BIC) is NP-hard even for three input permutations. This is a stronger result compared to the NP-hardness presented by Cunha et al. [10] for the case where there is an arbitrary number of input permutations.

4.1 Reducing median to closest

The polynomial reduction presented in Theorem 10 allows us to show that not only the Block-Interchange Closest problem is NP-hard for three input permutations, but also a closest problem where the corresponding median with a constant number of input permutations is NP-hard. Next we show that it is the case for the block-interchange rearrangement.

▶ **Definition 15.** *Given π^1 with p elements and π^2 with q elements, the* union *of π^1 and π^2 is a permutation $\pi^1 \uplus \pi^2$ with $p+q+1$ elements such that $\pi^1 \uplus \pi^2 = [\pi^1_1, \pi^1_2, \ldots, \pi^1_p, (p+1), (\pi^2_1 + p + 1), (\pi^2_2 + p + 1), \ldots, (\pi^2_q + p + 1)]$. For simplicity, $\pi^1 \uplus \pi^2$ is denoted by $\pi^{1,2}$. Permutations π^1 and π^2 are called* parts *of the union.*

▶ **Lemma 16.** *Given permutations π^1 and π^2, we have that $d_{\mathsf{BI}}(\pi^{1,2}) = d_{\mathsf{BI}}(\pi^1) + d_{\mathsf{BI}}(\pi^2)$.*

Proof. Assuming that π^1 has p elements and π^2 has q elements, since $p+1$ is greater than all elements of π^1 and smaller than all elements of π^2, the reality and desire diagram $G(\pi^1 \uplus \pi^2)$ is obtained by gluing $G(\pi^1)$ and $G(\pi^2)$, i.e., the reality and the desire edges do not change when the union operation is applied to permutations. As a direct consequence of Theorem 1, we have $d_{\mathsf{BI}}(\pi^{1,2}) = \frac{p+q+2-c(G(\pi^1))-c(G(\pi^2))}{2} = d_{\mathsf{BI}}(\pi^1) + d_{\mathsf{BI}}(\pi^2)$. ◀

▶ **Theorem 17.** *Given three permutations π^1, π^2 and π^3, σ is a solution of* BIM *if and only if $\biguplus_{i=1}^{6} \sigma$ is a solution of* BIC *for the permutations $\pi^{1,2,3,1,2,3}$, $\pi^{2,1,1,3,3,2}$, and $\pi^{3,3,2,2,1,1}$.*

Proof. Since permutations $\pi^{1,2,3,1,2,3}, \pi^{2,1,1,3,3,2}$, and $\pi^{3,3,2,2,1,1}$ are composed by six parts of unions and, considering BIC, each part corresponds to columns yielding π^1, π^2, and π^3. Moreover, by Lemma 16 each part can be treated separately without loss of optimality. Hence, there is a solution of BIC where all parts have the same solution δ. Therefore, there is a permutation $x \in \{\pi^{1,2,3,1,2,3}, \pi^{2,1,1,3,3,2}, \pi^{3,3,2,2,1,1}\}$ such that $d_{\mathsf{BI}}(\biguplus_{i=1}^{6} \delta, x) = 2(d_{\mathsf{BI}}(\delta, \pi^1) + d_{\mathsf{BI}}(\delta, \pi^2) + d_{\mathsf{BI}}(\delta, \pi^3))$. Since we want δ such that $d_{\mathsf{BI}}(\biguplus_{i=1}^{6} \delta, x)$ is minimized, we want δ such that $d_{\mathsf{BI}}(\delta, \pi^1) + d_{\mathsf{BI}}(\delta, \pi^2) + d_{\mathsf{BI}}(\delta, \pi^3)$ is minimized. Hence, this happens if and only if $\delta = \sigma$, where σ is solution of BIM. ◀

Since Block-Interchange Median is NP-complete (Theorem 10), as a consequence of Theorem 17, we have Corollary 18.

▶ **Corollary 18.** *The* Block-Interchange Closest *problem is* NP*-hard even when the input consists of three permutations.*

When dealing with transpositions, sorting each part of a union separately does not yield an optimum sequence in order to sort a permutation in general, as proved by Cunha et al. [11]. Hence, an analogous strategy of the one in Theorem 17 does not apply to reduce the median to the closest problems regarding transpositions rearrangement, given that Lemma 16 does not hold for sorting by transpositions. However, if each part of a union is *hurdle-free* i.e., a permutation in which the transposition distance is equal to the lower bound on the transposition distance $d_{\mathsf{T}}(\pi) \geq \frac{(n+1)-c_{\mathsf{odd}}(G(\pi))}{2}$, it follows that $d_{\mathsf{T}}(\pi^{1,2}) = d_{\mathsf{T}}(\pi^1) + d_{\mathsf{T}}(\pi^2)$ in the same matter as Theorem 17. Therefore, we have Corollary 19.

▶ **Corollary 19.** Transposition Closest *is* NP*-hard even when the input consists of three permutations which are unions of hurdle-free permutations.*

Proof. Transposition Median is NP-hard when $k = 3$ even for hurdle-free permutations [1], i.e., permutations in which the transposition distances are equal to the lower bound of Theorem 2. Since the distance of unions of hurdle-free permutations can be obtained by the sum of the distances of each part of the union, Theorem 17 holds in the same way. ◀

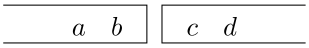
4.2 The Short-block-move Closest problem

Sufficient condition to sort by short-block-moves. We refer to block-moves that introduce elements in connected components of the permutation graph $PG(\pi)$ of π (defined in Section 2) as *merging moves*. For instance, $[2\ 3\ \underline{1\ 6}\ \underline{4}\ 5] \rightarrow [2\ 3\ 4\ 1\ 6\ 5]$ is a merging move.

▶ **Lemma 20.** *For every permutation π, sorting each connected component of $PG(\pi)$ separately is optimal.*

Proof. We allow ourselves to use merging moves, which can be replaced by correcting moves as in Table 2. The modified sequence is not longer than the original, and we observe that these new moves never merge components.

A merging move must act on contiguous components of π. Let us assume that the leftmost component the move acts on ends with elements a and b, and that the rightmost component starts with elements c and d, as represented below:



It follows that $a < c, a < d, b < c$, and $b < d$. We now replace any merging move involving those component's extremities with correcting moves. There are five cases to consider:

- $\underline{a}\ \underline{b\ c}\ d \rightarrow b\ c\ a\ d$: this move satisfies the conditions of Case 2 in Table 2, so we replace it with $\underline{a}\ \underline{b}\ c\ d \rightarrow b\ a\ c\ d$.
- $\underline{a\ b}\ \underline{c}\ d \rightarrow c\ a\ b\ d$: this move satisfies the conditions of Case 4 in Table 2, so we replace it with $\underline{a}\ \underline{b}\ c\ d \rightarrow b\ a\ c\ d$.
- $a\ \underline{b}\ \underline{c}\ d \rightarrow a\ c\ b\ d$: this move satisfies the conditions of Case 1 in Table 2, and in this case we just remove that block-move from the sorting sequence.
- $a\ \underline{b}\ \underline{c\ d} \rightarrow a\ c\ d\ b$: this move satisfies the conditions of Case 5 in Table 2, so we replace it with $a\ b\ \underline{c}\ \underline{d} \rightarrow a\ b\ d\ c$.
- $a\ \underline{b\ c}\ \underline{d} \rightarrow a\ d\ b\ c$: this move satisfies the conditions of Case 3 in Table 2, so we replace it with $a\ b\ \underline{c}\ \underline{d} \rightarrow a\ b\ d\ c$.

None of the correcting moves that we use to replace the non-correcting moves in those five cases is a merging move, and no such a replacement increases the length of our sorting sequence. Given any sorting sequence, we repeatedly apply the above transformation to the merging move with the smallest index until no such move remains; in particular, the transformation applies to optimal sequences as well, and the proof is complete. ◀

There exist cases where allowing merging moves still yields an optimal solution. This is the case for [2 1 4 3], which can be sorted optimally as follows: $[2\ \underline{1\ 4}\ \underline{3}] \rightarrow [\underline{2\ 3}\ \underline{1}\ 4] \rightarrow \iota$, where $\iota = [1\ 2\ \ldots\ n]$. It is natural to wonder whether Lemma 20 generalizes to p-bounded block-moves, for $p > 3$. However, the following counterexample shows that it is not the case, even when a block-move is bounded by four (i.e., a 4-bounded block move): sorting each component of [3 2 1 6 5 4] separately yields a sequence of length four, but one can do better by merging components as follows: $[3\ 2\ \underline{1\ 6}\ \underline{5\ 4}] \rightarrow [3\ \underline{2\ 5}\ \underline{4\ 1}\ 6] \rightarrow [\underline{3\ 4}\ \underline{1\ 2}\ 5\ 6] \rightarrow \iota$.

SHORT-BLOCK-MOVE CLOSEST problem is NP-hard. First, we apply Algorithm 1 to transform any string s of length m into a particular permutation λ_s of length $2m$.

Algorithm 1 Permut$_{\mathsf{BI}}(s)$.

```
Input  : A binary string $s$ of length $m$.
Output : A permutation $\lambda_s$.
For each occurrence of 0 in position $i$ of $s$, set the elements $2i-1$ and $2i$ in positions
  $2i-1$ and $2i$ of $\lambda_s$, respectively.
For each occurrence of 1 in position $i$ of $s$, set the elements $2i-1$ and $2i$ in positions
  $2i$ and $2i-1$ of $\lambda_s$, respectively.
```

Since from Lemma 20 each connected component can be sorted separately, and each bit set to 1 in s corresponds to an inversion in λ_s from Algorithm 1, it implies Lemma 21, which is an equality between the Hamming distance of an input string s and the short-block-move distance of its output permutation λ_s.

▶ **Lemma 21.** *Given a string of length m and a permutation λ_s of length $2m$ obtained by Algorithm 1, the short-block-move distance of λ_s is $d_{\mathsf{sbm}}(\lambda_s) = d_H(s)$.*

▶ **Lemma 22.** *Given a set of k permutations obtained by Algorithm 1, there is a short-block-move closest permutation with maximum distance at most d if and only if there is a Hamming closest string with maximum distance at most d.*

Proof. If λ' can be built by Algorithm 1 for some input string s', then, by Lemma 21, s' is a closest string. Otherwise, we search from left to right along the permutation to find the first position where the corresponding element is different from the one intended to be by the algorithm, which is a position $x \in \{2i-1, 2i\}$. In this case, all elements from position x until the position where the first element $y \in \{2i-1, 2i\}$ appear form inversions with respect to each input permutation, implying the short-block-move distance between the solution $[A\,x\,B\,y\,C]$ and any input greater than the distance between the new permutation $[A\,y\,x\,B\,C]$ and any input permutation, such that A, B, and C are blocks of elements. By repeating this process, a string agreeing with the algorithm output can be found and, by Lemma 21, a string with maximum distance at most d can be constructed. Given a solution string s, we obtain the associated permutation λ_s given by Algorithm 1. By Lemma 21 we have the solution s regarding the closest string corresponding to the permutation λ_s with the same value of maximum distance d, concluding the proof of the lemma. ◀

Since Hamming Closest String is NP-complete [28], Lemma 22 implies Theorem 23.

▶ **Theorem 23.** Short-block-move Closest Permutation *is* NP*-hard.*

Theorem 7, proved by Basavaraju et al. [3], states that Closest String does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly. Since the results presented in Lemma 8, Lemma 9, Lemma 22, Theorem 23, Corollary 19, as well as the results from Popov [31], are PPT reductions from Closest String, we have the following corollary.

▶ **Corollary 24.** Breakpoint Closest*,* Block-interchange Closest*,* Transposition Closest*,* Swap Closest *and* Short-block-move Closest *do not admit polynomial a kernel unless* NP $\subseteq$ coNP/poly*.*

4.3 FPT algorithms

Popov [31] solved the Swap Closest problem in time $\mathcal{O}(kn + g(k,d))$ parameterized by the number of permutations k (each of them of size n) and the distance d, where g is a function which depends only on k and d. Now, we propose FPT algorithms for finding closest permutations of a given set of permutations, parameterized just by the distance d (the function $\mathcal{O}^*(f(n))$ means that there exists an algorithm which runs in time $\mathcal{O}(f(n)) \cdot \mathsf{poly}(n)$, where $\mathsf{poly}(n)$ is a polynomial function in n). Our approach is inspired by the algorithm for the Closest String problem [15,19], considering the three metrics below.

▶ **Theorem 25.** *d-*Swap Closest*, d-*Short-Block-Move Closest*, and d-*Block-interchange Closest *can be solved in time $\mathcal{O}^*(d)^{\mathcal{O}(d)}$.*

Proof. First, we consider d-Swap Closest. The other problems follow in a similar way, as we discuss below. Let $\pi^1, \ldots, \pi^k$ be the input permutations. Recursively, we solve these problems using a bounded search tree technique as follows: First, set $z = \pi^1$ as a candidate permutation solution. If $d_{\mathsf{swap}}(y,z) \leq d$ for each permutation y of the input, then return `yes`. Otherwise, if $d = 0$ then return `no`. In the remaining case, $d > 0$ and there exists a permutation π^i with $d_{\mathsf{swap}}(\pi^i, z) > d$. From the triangular inequality, $d_{\mathsf{swap}}(\pi^i, z) \leq 2d$ for each input permutation z; otherwise, the answer is `no`. Since each swap operation corrects at most two positions, there are at most $4d$ positions on which π^i and z differ. Let P be a set of $4d$ positions on which π^i and z differ. Hence, we branch into $|P| = 4d$ subcases: for every $p \in P$, we define z^p to be equal to z except for the swap putting the element π^i_p in the position p of z^p, and we recursively solve the problem for the pair $(z^p, d-1)$.

We build a search tree of depth at most d, and every node has at most $4d$ children. Thus, the size of the search tree does not exceed $\mathcal{O}((4d)^d)$.

For d-Short-Block-Move Closest, it is known that each operation involves at most two edges on the associated permutation graph. Since the current solution must be at distance at most $2d$ from any input permutation, it holds that the associated permutation graph between a current solution z and any π^i has at most $4d$ edges and at most $8d$ vertices incident to some edge of the associated permutation graph; otherwise the answer is `no`. Therefore, either we are already dealing with an instance with universe of small size, or there are many isolated vertices in the associated permutation graph. By the definition of permutation graphs of strings, these isolated vertices represent positions that coincide in both permutations, and we may assume that they are not involved in any move to obtain one from the other. Since we do not need to consider moves involving these isolated vertices of the associated permutation graph, we can consider only moves involving $\mathcal{O}(d)$ many vertices. Thus, we can perform a similar bounded search tree algorithm as previously described.

For d-Block-Interchange Closest, it is known that each operation changes the number of cycles in the reality and desire diagram by -2, 0, or $+2$ (see [9]). Moreover, from Theorem 1, there exists an optimum sequence of block-interchanges that only applies 2-moves, i.e., each operation increases the number of cycles by two. This implies that there is no optimum sequence that uses -2 or 0 moves. Recall that we obtain a sorted permutation when we achieve only cycles of size one ($n + 1$ cycles in total); so, sorting is equivalent to maximizing the number of cycles in the reality and desire diagram. Thus, our focus is only analyzing possible 2-moves to approximate one permutation to another one in our bounded search tree algorithm. It is known that there is no 2-move that affects a 1-cycle (cycle of length one in the diagram), because a 2-move can only be performed into a unique cycle (cf. [9, 11]). Thus, there is no 2-move that affects an adjacency (a pair is an adjacency if and only if it yields a 1-cycle in the reality and desire diagram [2, 9]).

At this point, we have that we can safely reduce the permutation, since all optimum block-interchange sequences do not affect adjacencies (this is a stronger result than Theorem 3). Hence, as each block-interchange affects at most four breakpoints, the permutation must have at most $8d$ breakpoints (i.e., $8d+1$ elements in the reduced permutation). Therefore, we can consider only moves involving $\mathcal{O}(d)$ many breakpoints. Thus, we can perform a similar bounded search tree algorithm as previously described. ◀

Note that Breakpoint Closest does not admit a bounded search tree analogous to the ones used in Theorem 25, since this metric does not have a sequence of operations to transform a permutation into another one; so, it is unclear how to branch. Also, for the Transposition Closest problem, it is known that there may exist optimum sequences of transpositions that apply 0-moves and 2-moves, and it is an old open problem whether there are optimum sequences using -2-moves [2, 11]; so, it seems that is not safe to use the reduced permutation in that case, because there may exist an optimum sequence of transpositions that uses moves not preserved in the reduced instance, and those moves could be good for our branch step. Therefore, we leave both cases as open questions.

References

1 Martin Bader. The transposition median problem is NP-complete. *Theor. Comput. Sci.*, 412(12-14):1099–1110, 2011.

2 Vineet Bafna and Pavel A Pevzner. Sorting by transpositions. *SIAM J. Discrete Math.*, 11(2):224–240, 1998.

3 Manu Basavaraju, Fahad Panolan, Ashutosh Rai, MS Ramanujan, and Saket Saurabh. On the kernelization complexity of string problems. *Theor. Comput. Sci.*, 730:21–31, 2018.

4 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

5 David Bryant. The complexity of the breakpoint median problem. *Centre de recherches mathematiques, Technical Repert*, 1998.

6 Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by transpositions is difficult. *SIAM J. Discrete Math.*, 26(3):1148–1180, 2012.

7 Alberto Caprara. Sorting by reversals is difficult. In *Proceedings of the first annual international conference on Computational molecular biology*, pages 75–83, 1997.

8 Alberto Caprara. The reversal median problem. *INFORMS J. Comput.*, 15(1):93–113, 2003.

9 David Alan Christie. *Genome Rearrangement Problems.* University of Glasgow (United Kingdom), 1998.

10 Luís Felipe I. Cunha, Pedro Feijão, Vinícius F dos Santos, Luis Antonio B Kowada, and Celina MH de Figueiredo. On the computational complexity of closest genome problems. *Discrete Applied Mathematics*, 274:26–34, 2020.

11 Luís Felipe I. Cunha, Luis Antonio B Kowada, Rodrigo de A. Hausen, and Celina MH de Figueiredo. Advancing the transposition distance and diameter through lonely permutations. *SIAM J. Discrete Math.*, 27(4):1682–1709, 2013.

12 Luís Felipe I. Cunha, Luis Antonio B Kowada, Rodrigo de A. Hausen, and Celina MH de Figueiredo. A faster 1.375-approximation algorithm for sorting by transpositions. In *WABI 2014*, pages 26–37. Springer Berlin Heidelberg, 2014.

13 Luís Felipe I. Cunha, Luis Antonio B Kowada, Rodrigo de A Hausen, and Celina MH De Figueiredo. A faster 1.375-approximation algorithm for sorting by transpositions. *J. Comput. Biol.*, 22(11):1044–1056, 2015.

14 Luís Felipe I. Cunha and Fábio Protti. Genome rearrangements on multigenomic models: Applications of graph convexity problems. *J. Comput. Biol.*, 26(11):1214–1222, 2019.

15 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

16 Rodney G Downey and Michael Ralph Fellows. *Parameterized Complexity.* Springer Science & Business Media, 2012.

17 Guillaume Fertin, Anthony Labarre, Irena Rusu, Stéphane Vialette, and Eric Tannier. *Combinatorics of Genome Rearrangements.* MIT press, 2009.

18 Zheng Fu, Xin Chen, Vladimir Vacic, Peng Nan, Yang Zhong, and Tao Jiang. Msoar: a high-throughput ortholog assignment system based on genome rearrangement. *J. Comput. Biol.*, 14(9):1160–1175, 2007.

19 Gramm, Niedermeier, and Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37:25–42, 2003.

20 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Exact solutions for closest string and related problems. In *ISAAC 2001*, pages 441–453. Springer, 2001.

21 Maryam Haghighi and David Sankoff. Medians seek the corners, and other conjectures. In *BMC bioinformatics*, volume 13, pages 1–7. Springer, 2012.

22 Lenwood S Heath and John Paul C Vergara. Sorting by bounded block-moves. *Discrete Appl. Math.*, 88:181–206, 1998.

23 Lenwood S Heath and John Paul C Vergara. Sorting by short block-moves. *Algorithmica*, 28:323–352, 2000.

24 Ian Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

25 Gary Hoppenworth, Jason W Bentley, Daniel Gibney, and Sharma V Thankachan. The fine-grained complexity of median and center string problems under edit distance. In *28th Annual European Symposium on Algorithms, ESA 2020*, 2020.

26 D Knuth. The Art of Computer Programming: Sorting and Searching, vol 3, 1998.

27 Anthony Labarre. Sorting by Prefix Block-Interchanges. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation ISAAC 2020*, volume 181, pages 55:1–55:15, 2020.
28 J Kevin Lanctot, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003.
29 Pavel Pevzner. *Computational Molecular Biology: An Algorithmic Approach.* MIT press, 2000.
30 Itsik Pe'er and Ron Shamir. The median problems for breakpoints are NP-complete. In *Elec. Colloq. on Comput. Complexity*, 1998.
31 V Yu Popov. Multiple genome rearrangement by swaps and by element duplications. *Theor. Comput. Sci.*, 385(1-3):115–126, 2007.
32 Andrew J Radcliffe, Alex D Scott, and Elizabeth L Wilmer. Reversals and transpositions over finite alphabets. *SIAM J. Discrete Math.*, 19(1):224–244, 2005.
33 Geoffrey A Watterson, Warren J Ewens, Thomas Eric Hall, and Alexander Morgan. The chromosome inversion problem. *J. Theor. Biol.*, 99(1):1–7, 1982.

A Preliminaries on parameterized complexity

A *parameterized* problem is a decision problem whose instances are pairs $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is called the *parameter.* A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an FPT *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$.

A parameterized problem is *slice-wise polynomial* (XP) if there exists an algorithm $\mathcal{A}$ and two computable functions f, g such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an XP *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^{g(k)}$. Within parameterized problems, the class W[1] may be seen as the parameterized equivalent to the class NP of classical optimization problems. Without entering into details (see [15, 16] for the formal definitions), a parameterized problem being W[1]-*hard* can be seen as a strong evidence that this problem is not FPT. The canonical example of W[1]-hard problem is Clique parameterized by the size of the solution.

A parameterized problem is *para*-NP-*hard* if it is NP-hard for some fixed value of the parameter, such as the k-Coloring problem parameterized by the number of colors for every fixed $k \geq 3$.

▶ **Definition 26** (Bodlaender et al. [4]). *Let $P, Q \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. We say that a polynomial computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is a* polynomial parameter transformation *(PPT) from P to Q if for all $(x, k) \in \Sigma^* \times \mathbb{N}$ the following holds: $(x, k) \in P$ if and only if $(x', k') = f(x, k) \in Q$ and $k' \leq k^{O(1)}$.*

▶ **Definition 27** (Bodlaender et al. [4]). *A* kernelization algorithm*, or in short, a* kernel *for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that*

- *$(x, k) \in L \Leftrightarrow (x', k') \in L$, and*
- *$|x'|, k' \leq f(k)$,*

where f is some computable function and p is a polynomial. Any function f as above is referred to as the size *of the kernel.*

If we have a kernel for L, then for any $(x, k) \in \Sigma \times \mathbb{N}$, we can obtain in polynomial time an equivalent instance with respect to L whose size is bounded by a function of the parameter. Of particular interest are *polynomial kernels*, which are kernels whose size is bounded by a polynomial function.

▶ **Theorem 28** (Bodlaender et al. [4]). *Let P and Q be parameterized problems and P' and Q' be, respectively, the unparameterized versions of P and Q. Suppose that P' is* NP*-hard and Q' is in* NP*. Assume that there is a polynomial parameter transformation from P to Q. Then if Q admits a polynomial kernel, so does P. Equivalently, if P admits no polynomial kernel under some assumption then neither does Q.*

B Proofs deferred from Section 3

In this appendix we provide the proofs deferred from Section 3. For the sake of readability, we restate the corresponding result before proving it.

▶ **Theorem 10.** *The* Block-interchange Median *problem is* NP*-complete even when the input consists of three permutations.*

Proof. Let π^1, π^2, π^3 be permutations of $\{1, \dots, n\}$, and let k be an integer. Then, $(\pi^1, \pi^2, \pi^3, k) \in \mathsf{CM}$ if and only if there is a permutation σ satisfying $\sum_{i=1}^{3} c(G(\sigma, \pi^i)) \geq k$. Since solving an CM instance is equivalent to finding a permutation matching $M(\sigma)$ such that $\sum_{i=1}^{3} c(G(\sigma, \pi^i))$ is maximized and the block-interchange distance between any two permutations $d_{\mathsf{BI}}(\sigma, \pi^i) = \frac{n+1-c(G(\sigma,\pi^i))}{2}$, then $\mathsf{CM} \leq_{\mathsf{p}} \mathsf{BIM}$. ◀

▶ **Lemma 11.** *If an adjacency occurs in all of the input permutations, it occurs also in a solution of the* Block-interchange Median *and the* Transposition Median *problems.*

Proof. Assume $\mathtt{ab}$ is an adjacency that occurs in all k input permutations. Let σ be a solution median permutation satisfying a and b are not adjacent. Suppose, w.l.o.g. $\sigma = [\sigma_1, \dots, \sigma_{i-1}, \mathtt{a}, \sigma_{i+1}, \dots, \sigma_{j-1}, \mathtt{b}, \sigma_{j+1}, \dots, \sigma_n]$. Thus, we obtain the permutation σ' from σ by setting adjacencies $\mathtt{ab}$, $\mathtt{b}\sigma_{i+1}$ and $\sigma_{j-1}\sigma_{j+1}$, removing $\mathtt{a}\sigma_{i+1}$, $\sigma_{j-1}\mathtt{b}$ and $\mathtt{b}\sigma_{j+1}$, and keeping all other adjacencies of σ, i.e., $\sigma' = [\sigma_1, \dots, \sigma_{i-1}, \mathtt{a}, \mathtt{b}, \sigma_{i+1}, \dots, \sigma_{j-1}, \sigma_{j+1}, \dots, \sigma_n]$. Now, considering any optimum sequence of block-interchanges (or transpositions) from σ to π^i, we present a simulation sequence from σ' to π^i. Any operation applied on a sequence from σ to π^i that does not change adjacencies $\mathtt{a}\sigma_{i+1}$, $\sigma_{j-1}\mathtt{b}$ and $\mathtt{b}\sigma_{j+1}$ can be simulated properly from σ' to π^i without any loss, once any impact on the decreasing number of breakpoints is the same, and so the number of cycles on the reality and desire diagram. If an operation applied on a sequence from σ to π^i affects i) $\mathtt{a}\sigma_{i+1}$, ii) $\sigma_{j-1}\mathtt{b}$, or iii) $\mathtt{b}\sigma_{j+1}$, then we simulate it on a sequence from σ to π^i as follows: i) instead of cut a block just after $\mathtt{a}$, it is cut after the two elements $\mathtt{ab}$; ii) instead of cut a block just before $\mathtt{b}$, it is cut just after σ_{j-1}; iii) instead of cut a block just after $\mathtt{b}$ it is cut just before σ_{j+1} as well. Since all input permutations have the adjacency $\mathtt{ab}$, no extra operation must be applied from σ' to π^i. Thus, we conclude that $\sum\limits_{i=1}^{k} d_{\mathsf{BI}}(\sigma', \pi^i) \leq \sum\limits_{i=1}^{k} d_{\mathsf{BI}}(\sigma, \pi^i)$ (or $\sum\limits_{i=1}^{k} d_{\mathsf{T}}(\sigma', \pi^i) \leq \sum\limits_{i=1}^{k} d_{\mathsf{T}}(\sigma, \pi^i)$). ◀

▶ **Lemma 13.** *If an adjacency occurs in all of the input permutations, then it occurs also in a solution of the* Breakpoint Median *problem.*

Proof. Suppose that $x = [x_1, x_2, \dots, x_n]$ is a breakpoint median for the input $\pi^1, \pi^2, \dots, \pi^k$, and $\{x_i, x_j\}$ is a pair in $(\mathsf{Adj}(\pi^1) \cap \dots \cap \mathsf{Adj}(\pi^k)) \setminus \mathsf{Adj}(x)$. We obtain a set Y being $Y = \mathsf{Adj}(x) \cup \{\{x_i, x_j\}\}$. Hence, we modify Y in such a way to generate a set of adjacencies which forms a median solution. Given a pair of adjacency $\{u, v\} \in X$, let $w(u, v) = |\{X \in \{\pi^1, \dots, \pi^k\} : \{u, v\} \in \mathsf{Adj}(x)\}|$, i.e., $w(u, v)$ is the number of input permutations that have the adjacency $\{u, v\}$. If $w(x_{i-1}, x_i) \leq w(x_i, x_{i+1})$ we remove $\{x_{i-1}, x_i\}$ from Y, otherwise we remove $\{x_i, x_{i+1}\}$. In the same way, if $w(x_{j-1}, x_j) \leq w(x_j, x_{j+1})$ we remove $\{x_{j-1}, x_j\}$

from Y, otherwise we remove $\{x_j, x_{j+1}\}$. Note that in each one of the four possible cases, in the resulting set Y, $\{x_i, x_j\}$ happens exactly once, the same as the others elements. Since $w(x_i, x_j) = k$, $\sum_{i=1}^{k} d_{\mathsf{BP}}(\pi^i, Y) < \sum_{i=1}^{k} d_{\mathsf{BP}}(\pi^i, x)$, which is a contradiction. ◀

▶ **Theorem 14.** *The following problems admit a polynomial kernel parameterized by the value d of the desired median solution:*

1. *d-Swap Median.*
2. *d-Breakpoint Median.*
3. *d-Block-interchange Median.*
4. *d-Transposition Median.*
5. *d-Short-Block-Move Median.*

Proof. First, we consider a polynomial kernelization for the d-Swap Median problem based on the following reduction rules:

1. If there is a column with more than $d + 1$ elements, then return `no`.
2. If there is a column with at least two elements occurring at least $d + 1$ times, then return `no`.
3. If an element occurs more than d times in at least two columns, then return `no`.
4. If a row has at least d copies in the matrix, then either the solution is a copy of such a row, or the answer is `no`.

   We say that a column i is *heavy* for an element x if x occurs more than d times in it; otherwise, i is said *light* for x.
5. For each element x, if the sum of occurrences of x in its light columns is more than $2d$, then return `no`.
6. If the previous rules were not applied, remove the columns whose all elements are the same, and reduce the universe accordingly.

Since the goal is to determine whether there is a permutation $s^\star$ whose sum of the distances by swaps between $s^\star$ and all permutations of S is at most d, Rules 1-4 are clearly safe. Now, we discuss Rule 5. If S is a `yes`-instance then an element x having a heavy column i (by Rule 2, there is at most one heavy column) must have x in the position i of any optimum solution $s^\star$. Thus, the number of rows that contains x in positions different than i is at most d. Also, if x has no heavy column, then such a $s^\star$ contains x in some position i whose column has at most d occurrences of x, while the number of rows having x in other positions is also at most d (hence, $2d$ occurrences in total). Thus, Rule 5 is safe.

Regarding Rule 6, as each $s \in S$ is a permutation, it holds that if a column i of S contains only one element x, then all permutations of S have x in position i, implying that any optimal solution for the problem should contain x in position i. Thus, it is safe to ignore that column i and element x from the input. (Recall that $s^\star$ having x in position i implies that for any $s \in S$, there is an algebraic cycle of length one between $s^\star$ and s, which is the best possible because the swap metric can be seen as the minimum number of swaps to get only algebraic cycles of size one.)

At this point, we may suppose that Rules 1-5 were not applied and that S' is the resulting instance after the application of Rule 6. To complete the kernelization algorithm for d-Swap Median, we need the following lemma.

▶ **Lemma 29.** *If S' is a* `yes`*-instance of the d-*Swap Median *problem, then S' has at most $2d$ columns and $4d^2 + d$ rows.*

Proof. For a column i having more than one element, the distance between the optimum solution $s^\star$ and some permutation of S needs to count a swap move involving column i. In addition, each swap affects only two columns, implying that d moves can affect at most $2d$ columns. However, by Rule 6, any column of S' has more than one element. Therefore, if S' is a yes-instance of the d-Swap Median problem, it must have at most $2d$ columns.

Now, let us argue about the number of rows of S'. Recall that a row s is light if it has an element x in position i such that column i is light for x; otherwise it is heavy. By Rule 5, the number of rows that contain an element x in a light column for it is at most $2d$. Since, S' has at most $2d$ columns it also has at most $2d$ elements. Therefore, the number of light rows is at most $4d^2$. Finally, by definition, heavy rows have only elements in positions for which they are heavy. By Rule 3, each element is heavy for only one column, which implies that heavy columns are copies. By Rule 4, we conclude that we have at most d heavy rows in S'. Hence, S' has at most $4d^2 + d$ rows. ◀

Therefore, either the size of S' certifies a no-answer, or S' is returned as a kernel for the d-Swap Median problem.

Next, we discuss a kernelization algorithm for the d-Breakpoint Median problem. Recall that in the breakpoint metric, one does not care about occurrences of elements in columns but adjacencies of elements instead (regardless of their position in the rows). Thus, we should adapt the previous arguments accordingly.

A kernel for the d-Breakpoint Median problem can be found as follows:

1. If there is an element having with more than $d + 1$ distinct successor elements (distinct adjacencies) in the matrix, then return no.
2. If there is an element x with at least two elements occurring at least $d + 1$ times as successor of x in the matrix, then return no.
3. If an element occurs more than d times as successor of at least two other elements in the matrix, then return no.
4. If a row has at least d copies in the matrix, then either the solution is a copy of such a row, or the answer is no.

   We say that an element y is a *heavy successor* for an element x if xy occurs more than d times in the matrix; a successor of x that is not heavy is said to be a *light successor* for x.

5. For each element x, if the sum of occurrences of x with light successors is more than $2d$, then return no.
6. Assuming that the previous rules were not applied, if there is an adjacency between x and y (i.e., xy) occurring in all of the input permutations, then consider xy as a single element and reduce the universe accordingly. Repeat this until there is no such adjacencies.

The safety of Rules 1-4 is straightforward, for Rule 5 the argument is similar to the swap case replacing columns by successors, and for Rule 6 the safety proof follows from Lemma 13. Again, we suppose that Rules 1-5 were not applied and S' is the resulting instance after the application of Rule 6.

Similarly as above, to complete the kernelization algorithm for d-Breakpoint Median, we need the following lemma.

▶ **Lemma 30.** *If S' is a yes-instance of the d-Breakpoint Median problem, then S' has at most $2d$ columns and $4d^2 + d$ rows.*

Proof. For an element x having more than one element as successor or more than one element as predecessor in the matrix, the distance between the optimum solution $s^\star$ and some permutation of S needs to count a breakpoint involving element x. In addition, each breakpoint involves only two elements, implying that d breakpoints can involve at most $2d$ elements. However, by Rule 6, any element of S' is involved in at least one breakpoint. Therefore, if S' is a `yes`-instance of the d-Breakpoint Median problem, then, since we are dealing with permutations, it must have at most $2d$ elements and at most $2d$ columns.

Now, let us argue about the number of rows of S'. Recall that a row s is light if it has an adjacency xy such that y is a light successor for x; otherwise it is heavy. By Rule 5, for each element x, the number of rows that contain an adjacency xy where y is a light successor of x is at most $2d$. Since S' has at most $2d$ elements, the number of light rows is at most $4d^2$. Finally, by definition, heavy rows have only heavy successors. By Rule 3, each element is heavy for only one predecessor in the matrix, implying that heavy rows are copies. By Rule 4, there are at most d heavy rows in S'. Hence, S' has at most $4d^2 + d$ rows. ◀

Therefore, as above, either the size of S' certifies a `no`-answer, or S' is returned as a kernel for the d-Breakpoint Median problem.

Next, we discuss a kernelization for the d-Block-interchange Median problem and the d-Transposition Median problem. Recall that for both metrics, whenever there is a breakpoint there is a move to be "played" to obtain the identity. Thus, a large set of breakpoints being one per row is enough to certify a `no`-answer for both problems as well. As Rules 1-5 of the previous kernelization deal only with these kind of sets of breakpoints, they also hold as reduction rules for these two problems.

On the other hand, an analogous of Rule 6 may depend on the kind of move to be used. However, Lemma 11 shows that a similar reduction rule can also be applied for the d-Block-interchange Median problem and the d-Transposition Median problem. Regarding an analogous of Lemma 30, it is enough to observe that any block-interchange involves the adjacency of at most eight elements (at most four adjacencies involved), and then one can conclude that S' has at most $8d$ elements/columns and $16d^2 + d$ rows. Similarly, concerning transpositions, each move involves the adjacency of at most six elements (at most three adjacencies involved), and then one can conclude that S' has at most $6d$ elements/columns and $12d^2 + d$ rows.

Finally, we discuss a kernelization for the d-Short-Block-Move Median problem. As previously discussed, Rules 1-5 described for the breakpoint distance can be also applied to any metric where the existence of a breakpoint certifies the existence of a move to be "played" in order to obtain the identity. Thus, they work for the short-block-move distance as well. However, unlike with the d-Block-interchange Median problem and the d-Transposition Median problem, an immediate analogue of Rule 6 does not apply to the short-block-move distance, because it may be necessary to traverse some positions to get an element from one point to another, temporarily breaking some "good" adjacencies. To get around this problem, we introduce the notion of homogeneous columns.

A column of an input matrix/set S is *homogeneous* if it contains only one element, and *heterogeneous* otherwise. Note that the existence of a heterogeneous column implies the existence of a move involving such a column. Since we are looking for a permutation $s^\star$ whose sum of distances from the input permutations is at most d, it follows that S contains at most $3d$ heterogeneous columns. So, either we have already a kernel or too many homogeneous columns where many of them are not involved in moves needed for the calculation of the distance between $s^\star$ and any $s \in S$. Then, an analogous of Rule 6 for this problem must identify these homogeneous columns, remove them, and reduce the universe properly. Due to Lemma 12, we can safely apply the following reduction rule.

- If there is an interval I with $6d + 2$ consecutive homogeneous columns, then remove the middle columns of I and reduce the universe size accordingly. Repeat this until there is no such interval.

After applying the above rule, we claim that the number of columns of a `yes`-instance is at most $18d^2 + 9d + 1$, because it has at most $3d$ heterogeneous columns and a sequence of at most $6d + 1$ homogeneous columns before/after a heterogeneous one. This remark together with the reduction rules applied implies that the number of rows is at most $36d^3 + 18d^2 + 3d$. This concludes the existence of a polynomial kernel for the d-Short-Block-Move Median problem. ◀