Theoretical Computer Science 958 (2023) 113857

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Sorting by prefix block-interchanges ☆

Anthony Labarre

Laboratoire d'Informatique Gaspard Monge, Université Gustave Eiffel, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454, Marne-la-Vallée, France

ARTICLE INFO

Article history:
Received 4 October 2022
Received in revised form 10 February 2023
Accepted 28 March 2023
Available online 31 March 2023
Communicated by T. Calamoneri

Keywords:
Permutations
Genome rearrangements
Interconnection network
Sorting
Distance
Prefix block-interchange

ABSTRACT

We initiate the study of sorting permutations using *prefix block-interchanges*, which exchange any prefix of a permutation with another non-intersecting interval. The goal is to transform a given permutation into the identity permutation using as few such operations as possible. We give a 2-approximation algorithm for this problem, as well as a 4/3-approximation for *simple permutations*; we prove tight lower and upper bounds on the corresponding distance; and we bound the largest value that the distance can reach.

1. Introduction

The problem of transforming two sequences into one another using a specified set of operations has received a lot of attention in the last decades, with applications in computational biology as *(genome) rearrangement problems* [13] as well as interconnection network design [23]. In the context of permutations, it can be equivalently formulated as follows: given a permutation π of $[n] = \{1, 2, \ldots, n\}$ and a generating set S (also consisting of permutations of $[n]$), find a minimum-length sequence of elements from S that sorts π. The problem is known to be NP-hard in general [16] and W[1]-hard when parameterised by the length of a solution [6], but some families of operations that are important in applications lead to problems that can be solved in polynomial time (e.g. *exchanges* [18], *block-interchanges* [10] and *signed reversals* [15]), while other families yield hard problems that admit good approximations (e.g. 11/8 for *reversals* [3] and for *block-transpositions* [12]).

Several restrictions of these families have also been studied, one of which stands out in the field of interconnection network design: the so-called *prefix constraint*, which forces operations to act on a prefix of the permutation rather than on an arbitrary interval. Those restrictions were introduced as a way of reducing the size of the generated network while maintaining a low value for its *diameter*, thereby guaranteeing a low maximum communication delay [23]. The most famous example is perhaps the restriction of reversals (which reverse the order of elements along an interval) to *prefix reversals*, and the corresponding problem known as *pancake flipping*, introduced by Kleitman et al [17] and whose complexity was only settled thirty years later [5].

As Table 1 shows (see Fertin et al [13] for undefined terms), although sorting problems using interval transformations are now fairly well understood, progress on the corresponding prefix sorting problems has been lacking, with only three

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

E-mail address: Anthony.Labarre@univ-eiffel.fr.
URL: http://igm.univ-mlv.fr/~alabarre/.

https://doi.org/10.1016/j.tcs.2023.113857
0304-3975/

Table 1
Complexity of some sorting problems on permutations in the unrestricted setting and under the prefix constraint.

Operation	Unrestricted	Prefix-constrained
reversal	NP-hard [7]	NP-hard [5]
signed reversal	in P [15]	open
double cut-and-join	NP-hard [8]	open
signed double cut-and-join	in P [26]	in P [14]
exchange	in P [18]	in P [1]
block-transposition	NP-hard [4]	open
block-interchange	in P [10]	open

families whose status has been settled and no approximation ratio smaller than 3/2 for those problems not known to be in P. As a result, while the topology of the *Cayley graph* generated by those operations might exhibit attractive properties, efficient routing algorithms (which achieve exactly the same task as the sorting algorithms in genome rearrangements) are still needed for the network to be of practical interest.

In this work, we choose to focus on the family of block-interchanges for the following reasons:

1. along with double cut-and-joins, they constitute one of the most general kind of operations on permutations, including both exchanges and block-transpositions as special cases;
2. their behaviour in the unrestricted setting is understood well enough that we can hope for the corresponding prefix sorting problem to be in P;
3. knowledge about these operations in the prefix setting is lacking and will be needed for more general studies; for instance, rearrangement problems on strings are usually NP-hard, and efficient algorithms to solve them exactly or approximately routinely rely on techniques developed for permutations [13, part II], which currently do not exist for prefix block-interchanges.

We are only aware of two other works on prefix block-interchanges. The first one is a paper by Chou et al [9], who studied them on strings and showed that binary strings can be sorted in linear time, whereas transforming two binary strings into one another using the minimum number of prefix block-interchanges is NP-complete. The second one is a paper by Pai and Chitturi [24], published after the conference version of our work [21], which gives another 2-approximation for sorting permutations by prefix block-interchanges, which we will discuss in section 7, and a 4/3-approximation for permutations with $O(1)$ cycles. Our main contributions in this paper are:

1. tight upper (Theorem 1, Theorem 5, Lemma 12) and lower (Theorem 3, Corollary 2, Corollary 3) bounds on the so-called *prefix block-interchange distance*;
2. an approximation algorithm (Algorithm 1) which we prove to be a 2-approximation with respect to two different measures;
3. tractable instances (Lemma 9) and a 4/3-approximation for *simple permutations* (Algorithm 2);
4. an upper bound of $\lfloor 2n/3 \rfloor$ on the maximum value of the distance (Corollary 4), an important parameter in some applications [23].

A large portion of the results in this paper first appeared in a conference version [21]. The main additions and differences are:

- a simpler presentation of Lemma 2, and a slight practical improvement of Algorithm 1 (which unfortunately does not improve its theoretical worst-case performance);
- the tighter lower bound of Corollary 2;
- all results in section 5;
- the removal of the results of Theorems 21 and 27 from [21]: the statement of Theorem 21 is erroneous, and can therefore not be used to obtain the lower bound needed in Theorem 27. Therefore, we can currently only give an upper bound on the diameter instead of an exact value;
- the distribution of the prefix block-interchange distance up to $n = 12$ (Table 2);
- an experimental assessment of our algorithm, and a discussion of Pai and Chitturi's algorithm (section 7).

We implemented Algorithm 1 and Algorithm 2 in Python. The source files are freely available at http://igm.univ-mlv.fr/~alabarre/software.php.

2. Notation and definitions

A *permutation* is a bijective function of a set (usually $[n] = \{1, 2, \ldots, n\}$ in this work) onto itself. The *symmetric group* S_n is the set of all permutations of $[n]$ together with the usual function composition applied from right to left. We write permu-

tations using lower case Greek letters, viewing them as sequences $\pi = \langle \pi_1\ \pi_2\ \cdots\ \pi_n \rangle$, where $\pi_i = \pi(i)$, and occasionally rely on the two-line notation to denote them. The permutation $\iota = \langle 1\ 2\ \cdots\ n \rangle$ is the *identity permutation*.

Permutations are well known to decompose in a unique way into *disjoint cycles* (up to the ordering of cycles and of elements within each cycle), leading to another notation for π based on its *disjoint cycle decomposition*. For instance, when $\pi = \langle 7\ 1\ 4\ 5\ 3\ 2\ 6 \rangle$, the disjoint cycle notation is $\pi = (1, 7, 6, 2)(3, 4, 5)$. A permutation that contains only one cycle of length $k > 1$ is commonly referred to as a k-cycle.

Definition 1. [10] The *block-interchange* $\beta(i, j, k, \ell)$ with $1 \leq i < j \leq k < \ell \leq n+1$ is the permutation that exchanges the closed intervals determined respectively by i and $j-1$ and by k and $\ell-1$. If $i = 1$, then β is called a *prefix block-interchange*.

Block-interchanges generalise several well-studied operations: when $j = k$, the resulting operation exchanges two adjacent intervals, and is known as a *(block-)transposition* [2]; when $j = i+1$ and $\ell = k+1$, the resulting operation swaps elements in respective positions i and k, and is called an *exchange* (or *(algebraic) transposition*). Prefix block-transpositions and prefix exchanges are defined in a fashion similar to prefix block-interchanges, i.e., prefix block-transpositions are block-transpositions with $i = 1$ and prefix exchanges are exchanges with $i = 1$. We study the following problem.

SORTING BY PREFIX BLOCK-INTERCHANGES (SBPBI)
Input: a permutation π in S_n, a number $K \in \mathbb{N}$.
Question: is there a sequence of at most K prefix block-interchanges that sorts π?

The length of a shortest sorting sequence of prefix block-interchanges for a permutation π is its *(prefix block-interchange) distance*, which we denote $pbid(\pi)$. For instance, a possible sorting sequence for $\pi = \langle 1\ 4\ 3\ 2 \rangle$ is:

$$\langle \boxed{1\,4}\,3\,\boxed{2} \rangle \rightarrow \langle \boxed{2\,3}\,\boxed{1}\,4 \rangle \rightarrow \langle 1\,2\,3\,4 \rangle.$$

Proving its optimality will require a lower bound (see Theorem 3). Distances based on other operations are defined similarly; they can be computed between any pair of permutations, and since they are left-invariant, computing $pbid(\pi, \sigma)$ is equivalent to computing $pbid(\sigma^{-1}\pi, \iota)$, which allows us to restrict our attention to sorting problems and sequences.

3. A 2-approximation based on the breakpoint graph

We give in this section a 2-approximation algorithm for SBPBI based on the *breakpoint graph*. We first use this structure in subsection 3.1 to derive an upper bound on *pbid* and present our algorithm, then derive a lower bound in subsection 3.2 which allows us to prove its performance guarantee. The breakpoint graph is well known to be equivalent [19] to another structure known as the *cycle graph* [2], which allows us to use results based on either graph indifferently.

Definition 2. [15] For any π in S_n, let π' be the permutation of $\{0, 1, 2, \ldots, 2n+1\}$ defined by $\pi'_0 = 0$, $\pi'_{2n+1} = 2n+1$, and $(\pi'_{2i-1}, \pi'_{2i}) = (2\pi_i - 1, 2\pi_i)$ for $1 \leq i \leq n$. The *breakpoint graph* of π is the undirected edge-bicoloured graph $G(\pi) = (V, E_b \cup E_g)$ with ordered vertex set $(\pi'_0, \pi'_1, \ldots, \pi'_{2n+1})$ and whose edge set consists of:

- $E_b = \{\{\pi'_{2i}, \pi'_{2i+1}\} \mid 0 \leq i \leq n\}$, called the set of *black* edges;
- $E_g = \{\{2i, 2i+1\} \mid 0 \leq i \leq n\}$, called the set of *grey* edges.

Fig. 1 shows an example of a breakpoint graph. Since $G(\pi)$ is 2-regular, it decomposes in a unique way into edge-disjoint cycles which alternate black and grey edges. The *length* of a cycle in $G(\pi)$ is the number of black edges it contains, and a *k-cycle* in $G(\pi)$ is a cycle of length k. We let $c(G(\pi))$ (resp. $c_k(G(\pi))$) denote the number of cycles (resp. k-cycles) in $G(\pi)$, and refer to 1-cycles as *trivial cycles*.

A crucial insight of strategies based on the breakpoint graph is the observation that the transformations that we apply never affect grey edges, whereas they remove black edges and replace them with new black edges. This point of view conveniently allows us to define block-interchanges in terms of the black edges on which they *act*: using the notation $b_i = \{\pi'_{2i-2}, \pi'_{2i-1}\}$ for a black edge, a quadruplet (b_i, b_j, b_k, b_ℓ) of black edges with $i < j \leq k < \ell$ naturally defines the block-interchange $\beta(i, j, k, \ell)$ and conversely.

3.1. An upper bound based on the breakpoint graph

For any π in S_n, let $f(\pi) = 0$ if π *fixes* 1 (i.e. $\pi_1 = 1$) and 1 otherwise. The following quantity, defined for any π in S_n, has been shown to be a tight lower bound on the prefix block-transposition distance [20]:

$$g(\pi) = \frac{n + 1 + c(G(\pi))}{2} - c_1(G(\pi)) - f(\pi). \tag{1}$$

We prove in Theorem 1 that this quantity is also an *upper* bound on the prefix block-interchange distance. To that end, we use the following notation, based on the one introduced by Bafna and Pevzner [2]; for any two permutations π and σ, define:

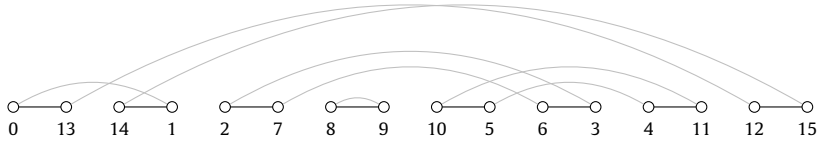


Fig. 1. The breakpoint graph of ⟨7 1 4 5 3 2 6⟩.

- $\Delta c(\pi, \sigma) = c(G(\sigma)) - c(G(\pi))$,
- $\Delta c_1(\pi, \sigma) = c_1(G(\sigma)) - c_1(G(\pi))$,
- $\Delta f(\pi, \sigma) = f(\sigma) - f(\pi)$, and
- $\Delta g(\pi, \sigma) = g(\sigma) - g(\pi)$.

These parameters allow us to obtain the following expression, which will be useful in our proofs:

$$\Delta g(\pi, \sigma) = \Delta c(\pi, \sigma)/2 - \Delta c_1(\pi, \sigma) - \Delta f(\pi, \sigma). \tag{2}$$

We start by proving in Lemma 2 the existence of a prefix block-interchange that decreases $g(\pi)$ by at least one if $\pi_1 \neq 1$. The proof uses the following structural result, where grey edges $\{\pi'_a, \pi'_b\}$ and $\{\pi'_c, \pi'_d\}$ (with $a < b$ and $c < d$) are said to *intersect* if $a < c < b < d$ or $c < a < d < b$.

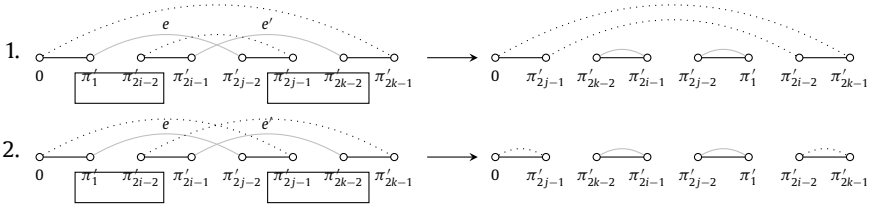
Lemma 1. *[15] For every permutation π, let e be a grey edge in a nontrivial cycle of $G(\pi)$; then there exists another grey edge e' in $G(\pi)$ that intersects e.*

We refer to the grey edge of $G(\pi)$ that contains π'_1 as the *first* grey edge, and to the cycle that contains 0 as the *leftmost cycle*. Our figures represent alternating subpaths (i.e., paths that alternate black and grey edges) as dotted edges; therefore, such a dotted edge might correspond to a unique grey edge, or to a black edge framed by two grey edges, and so on.

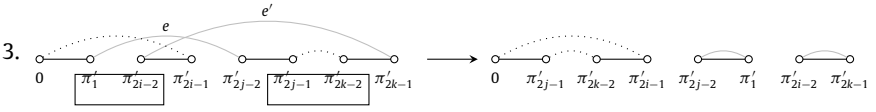
When the need arises, we will distinguish between two categories of grey edges: assuming without loss of generality that $a < b$, we call a grey edge $\{\pi'_a, \pi'_b\}$ an *inner* grey edge if a is odd, and an *outer* grey edge otherwise. For instance, grey edges $\{0, 1\}$, $\{2, 3\}$, $\{8, 9\}$, $\{10, 11\}$, $\{14, 15\}$ in Fig. 1 are outer grey edges, while grey edges $\{5, 4\}$, $\{7, 6\}$, and $\{13, 12\}$ are inner grey edges. In particular, the first grey edge is always an inner grey edge. Although Lemma 1 guarantees intersection relationships between grey edges, some grey edges might intersect only inner or only outer grey edges, and will therefore yield different options for applying prefix block-interchanges.

Lemma 2. *For any π in S_n: if $\pi_1 \neq 1$, then there exists a prefix block-interchange β such that $\Delta c(\pi, \pi\beta) = 2$, $\Delta c_1(\pi, \pi\beta) \geq 2$, and $\Delta g(\pi, \pi\beta) \leq -1$.*

Proof. We first present a case analysis explaining how to compute β in each case, then prove that β meets the stated requirements. Let e be the first grey edge. Lemma 1 guarantees the existence of a grey edge e' that intersects e; if e' is an inner grey edge, we apply the prefix block-interchange defined by the black edges connected by e and by e'. There are two cases to handle, depending on whether or not e' belongs to the leftmost cycle:



If no inner grey edge intersects e, then there must be an outer grey edge e' that intersects e. We first assume that e and e' connect four different black edges; again, we apply the prefix block-interchange defined by the black edges connected by e and by e', whether or not e and e' both belong to the leftmost cycle:



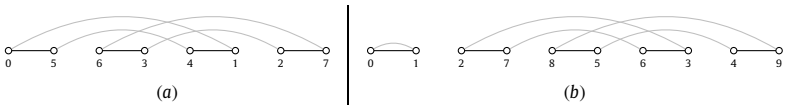
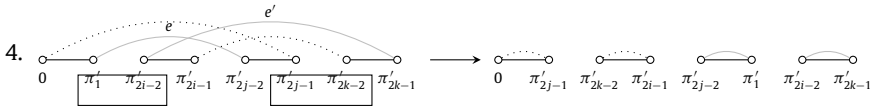
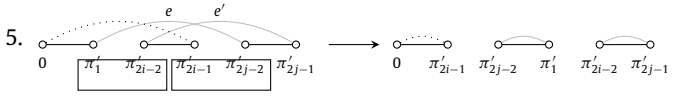


Fig. 2. The breakpoint graphs of (*a*) ⟨3 2 1⟩ and (*b*) ⟨1 4 3 2⟩.



Finally, if all outer grey edges that intersect e share a black edge with e, then we may assume that one such grey edge, which we again write e', does not contain 0. Indeed, assume $\{0, k\}$ intersects e; then by definition $k < i$, and there exists an alternating path connecting π'_{2k-1} and π'_{2i+1}. Therefore, either the grey edge incident with π'_{2i+1} connects it to a vertex located before π'_{2i+1}, in which case it is an outer edge and we have found the wanted e'; or that grey edge connects it with a vertex located after π'_{2i+1}, in which case the analysis of case 3 applies.

If there is an outer grey edge e' that shares a black edge with e, then we apply the prefix block-interchange β defined by the corresponding three distinct black edges, which is in fact a prefix block-transposition:



We now show that in all five cases, the prefix block-interchange we apply meets the requirements in our statement. It is easy to verify that all choices we describe for β satisfy $\Delta c(\pi, \pi\beta) = 2$ and $\Delta c_1(\pi, \pi\beta) \geq 2$. Since $\pi_1 \neq 1$, the leftmost cycle in $G(\pi)$ is never trivial, so $f(\pi) = -1$. The leftmost cycle in $G(\pi\beta)$ remains nontrivial in cases 1 and 3, but might be trivial in cases 2, 4 and 5, which would then yield $\Delta f(\pi) = 1$ but also $\Delta c_1(\pi, \pi\beta) \geq 3$. Equation 2 therefore allows us to conclude that in all cases, we obtain $\Delta g(\pi, \pi\beta) \leq -1$. □

We handle the case where $\pi_1 = 1$ in the proof of our upper bound below.

Theorem 1. *For any π in S_n, we have $pbid(\pi) \leq g(\pi)$.*

Proof. If $\pi_1 \neq 1$, then we apply Lemma 2 to decrease $g(\pi)$ by at least 1. Otherwise, $\{\pi'_0, \pi'_1\}$ induces a 1-cycle in $G(\pi)$ and $f(\pi) = 0$. Assume $\pi \neq \iota$ to avoid triviality; then $G(\pi)$ contains a nontrivial cycle, from which we select a grey edge $\{\pi'_{2i-2}, \pi'_{2j-1}\}$ with $j > i$. Applying the prefix block-interchange $\beta(1, i, i, j)$ then makes π_i and $\pi_i + 1$ contiguous in $\pi\beta$, and that pair corresponds to a new 1-cycle in $G(\pi\beta)$. On the other hand, β merges the 1-cycle induced by $\{\pi'_0, \pi'_1\}$ in $G(\pi)$ with the cycle that contains $\{\pi'_{2i-2}, \pi'_{2j-1}\}$, so $\Delta c(\pi, \pi\beta) = 0 = \Delta c_1(\pi, \pi\beta)$, $\Delta f(\pi, \pi\beta) = 1$ and Equation 2 yields $\Delta g(\pi, \pi\beta) = 0/2 - 0 - 1 = -1$. □

The smallest example of a permutation for which the inequality in Theorem 1 is strict is $\pi = \langle 3\ 2\ 1 \rangle$, with $pbid(\pi) = 1 < g(\pi) = 2$ (Fig. 2(*a*) shows its breakpoint graph). Algorithm 1 implements the strategy described in Theorem 1, favouring *proper* block-interchanges (i.e., $\beta(i, j, k, \ell)$ with $j < k$) over block-transpositions whenever possible. We prove in the next subsection that Algorithm 1 is a 2-approximation.

Algorithm 1 can be implemented to run in time $O(n^2)$: the main loop undergoes $O(n)$ iterations, and products of permutations can be computed in time $O(n)$. The properties that must be checked in order to decide which prefix block-interchange should be applied can be verified in time $O(n)$. We note that the *log-lists* introduced by Rusu [25] can be used to compute $\pi\sigma$ in time $O(\log n)$ rather than $O(n)$, but we still need a more efficient way of performing the more expensive checks in order to improve the complexity of Algorithm 1.

3.2. A lower bound based on the breakpoint graph

We now prove a lower bound on *pbid* that allows us to show that Algorithm 1 is a 2-approximation for SBPBI. To that end, we use a framework we introduced in a previous paper [20]. The starting point is the following mapping, in which the symmetric group on $[n+1]$ is identified with the symmetric group on $\{0\} \cup [n]$ and where A_n is the subgroup of S_n formed by the set of all permutations that are the product of an even number of exchanges:

$$\psi : S_n \to A_{n+1} : \pi \mapsto \overline{\pi} = (0, 1, 2, \ldots, n)(0, \pi_n, \pi_{n-1}, \ldots, \pi_1). \tag{3}$$

This mapping associates to every permutation π another permutation $\overline{\pi}$ whose disjoint cycles are in one-to-one correspondence with the cycles of $G(\pi)$. As a result, terminology based on the disjoint cycle decomposition of $\overline{\pi}$ or on the alternating

Algorithm 1: ApproximateSbpbi(π).

Input: A permutation π of $[n]$.
Output: A sorting sequence of prefix block-interchanges for π.

```
1  S ← empty sequence;
2  while π ≠ ι do
3      if π_1 ≠ 1 then
4          j ← the position of π_1 − 1;
5          if {1, j} intersects an inner grey edge {i, k} then
6              σ ← β(1, i, j, k);                                   // Lemma 2 cases 1 and 2
7          else
               /* condition k > j+1 below favours proper block-interchanges over block-transpositions; Lemma 2
               guarantees the existence of either option                                          */
8              {i, k} ← an outer grey edge that intersects {0, j} with i > 0 and k > j+1 if one exists; otherwise, an outer grey edge that intersects
                 {0, j} with i > 0 and k = j+1;
9              if k = j+1 then σ ← β(1, i, i, k);                   // Lemma 2 case 5
10             else σ ← β(1, i, j, k);                              // Lemma 2 cases 3 and 4
11     else                                                         // Theorem 1
12         i ← smallest index such that π_{i+1} ≠ π_i + 1;
13         j ← the position of π_i + 1;
14         σ ← β(1, i, i, j);
15     π ← πσ;
16     S.append(σ);
17 return S;
```

cycle decomposition of $G(\pi)$ can conveniently be used indifferently, including the notation introduced at the beginning of section 3 (e.g. $c(\overline{\pi}) = c(G(\pi))$, and therefore $\Delta c(\overline{\pi}, \overline{\pi\sigma}) = c(\overline{\pi\sigma}) - c(\overline{\pi}) = c(G(\pi\sigma)) - c(G(\pi)) = \Delta c(\pi, \pi\sigma)$). Recall that two permutations π and π' are *conjugate* if there exists a permutation σ such that $\pi' = \sigma\pi\sigma^{-1}$. This equivalence relation partitions S_n into *conjugacy classes*, each of which contains all permutations that have the same cycle structure. The following result will be our main tool for proving our lower bound.

Theorem 2. *[20] Let S be a subset of S_n whose elements are mapped by $\psi(\cdot)$ onto $S' = \psi(S) \subseteq A_{n+1}$. Moreover, let $\mathscr{C}$ be the union of the conjugacy classes (of S_{n+1}) that intersect with S'; then for any π in S_n, any factorisation of π into the product of t elements of S yields a factorisation of $\overline{\pi}$ into the product of t elements of $\mathscr{C}$.*

Consequently, for any $S \subseteq S_n$, if we let $d_S(\sigma)$ denote the length of a shortest sorting sequence for σ consisting solely of elements from S, then Theorem 2 implies that for any π in S_n, we have $d_S(\pi) \geq d_{S'}(\overline{\pi})$. In order to use Theorem 2, we need a translation of the effect of an operation on π in terms of a transformation on $\overline{\pi}$, as well as a precise characterisation of the image of a prefix block-interchange under the mapping ψ. Both are provided, respectively, by the following results.

Lemma 3. *[20] For all π, σ in S_n, we have $\overline{\pi\sigma} = \overline{\pi}\pi\overline{\sigma}\pi^{-1}$.*

Lemma 4. *[20] For any block-interchange $\beta(i, j, k, \ell)$ in S_n, we have $\overline{\beta(i, j, k, \ell)} = (j, \ell)(i, k)$.*

As is well-known, a 2-cycle σ containing elements from different cycles in a permutation π merges those cycles in $\pi\sigma$, while a 2-cycle in σ containing elements from the same cycle in π splits that cycle into two cycles in $\pi\sigma$. Lemma 3 and Lemma 4 therefore provide us with a very simple way of analysing the effects of a block-interchange: the effect of β on the cycles of $G(\pi)$ is the same as the effect of $\pi\overline{\beta}\pi^{-1}$ on the cycles of $\overline{\pi}$, and therefore bounds on the (prefix) block-interchange distance of π can be obtained by studying the effects of pairs of 2-cycles on $\overline{\pi}$. The following lemma will be useful in restricting the number of cases in the proof of our lower bound (Theorem 3).

Lemma 5. *For any π in S_n and any block-interchange β, we have $\Delta c(\pi, \pi\beta) \in \{-2, 0, 2\}$.*

Proof. By Lemma 4, $\overline{\beta}$ consists of two 2-cycles, each of which might split a cycle into two cycles or merge two cycles into one (Lemma 3). Combining all possible cases yields the set $\{-2, 0, 2\}$ as possible values for $\Delta c(\overline{\pi}, \overline{\pi\beta}) = \Delta c(\pi, \pi\beta)$. □

Finally, the following technical observation will be useful in ruling out impossible values for $\Delta f(\pi, \sigma)$, whose set of possible values is $\{-1, 0, 1\}$ when no restrictions apply.

Lemma 6. *For any π in S_n and every prefix block-interchange β: if $\Delta c_1(\pi, \pi\beta) \geq 2$, then $\Delta f(\pi, \pi\beta) \neq 1$.*

Proof. If $\Delta c_1(\pi, \pi\beta) \geq 2$, then the new 1-cycles are obtained in one of the following ways:

1. if at least one of them is the result of a split of the leftmost cycle of $G(\pi)$, then that cycle is nontrivial and therefore $f(\pi) = 1$, thereby forbidding the value $\Delta f(\pi, \pi\beta) = 1$;
2. otherwise, all new 1-cycles are extracted from a cycle in $G(\pi)$ other than the leftmost cycle; since that cycle can only be split into at most two new cycles (Lemma 3 and Lemma 4), we have $\Delta c_1(\pi, \pi\beta) \leq 2$ in this case. Moreover, we also have $\pi_1 = 1$, otherwise the 1-cycle containing π_1 would vanish in $G(\pi\beta)$ and contradict our assumption that $\Delta c_1(\pi, \pi\beta) \geq 2$. Therefore, the value $\Delta f(\pi, \pi\beta) = 1$ is also excluded in this case. □

We now have everything we need to prove our lower bound on *pbid*.

Theorem 3. *For any π in S_n, we have $pbid(\pi) \geq g(\pi)/2$.*

Proof. By Theorem 2 and Lemma 4, we have $pbid(\pi) \geq d(\overline{\pi})$, where $d(\overline{\pi})$ is the length of a shortest sorting sequence for $\overline{\pi}$ where the only nontrivial cycles of each transformation in the sequence are two 2-cycles, exactly one of which contains 1. As a result, any lower bound on $d(\overline{\pi})$ is a lower bound on $pbid(\pi)$, and therefore we only need to show that a transformation of the kind we have just described can decrease the value of $g(\pi)$ by at most 2.

Let $\overline{\beta} = (1, a)(b, c)$ be the image of a prefix block-interchange under the mapping $\psi(\cdot)$. By Lemma 5, we only need to distinguish between the following three cases; in each situation, we aim to minimise the value of $\Delta g(\overline{\pi}, \overline{\pi\beta})$.

1. If $\Delta c(\overline{\pi}, \overline{\pi\beta}) = -2$, then clearly $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) \leq 0$, and Equation 2 allows us to conclude that $\Delta g(\overline{\pi}, \overline{\pi\beta}) \geq -1 - 0 - 1 = -2$.
2. If $\Delta c(\overline{\pi}, \overline{\pi\beta}) = 0$, then either 2-cycle of β merges two cycles while the other splits a cycle into two. The lengths of the involved cycles in $\overline{\pi}$ and in $\overline{\pi\beta}$ may vary, but this observation is enough to deduce that $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) \leq 2$. The lowest value of $\Delta g(\overline{\pi}, \overline{\pi\beta})$ is obtained when $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) = 2$, in which case Equation 2 and Lemma 6 yield $\Delta g(\overline{\pi}, \overline{\pi\beta}) \geq 0 - 2 - 0 = -2$, or when $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) = 1$, in which case Equation 2 yields $\Delta g(\overline{\pi}, \overline{\pi\beta}) \geq 0 - 1 - 1 = -2$.
3. If $\Delta c(\overline{\pi}, \overline{\pi\beta}) = 2$, then both elements of β each split one cycle into two cycles. As in the previous case, the lengths of the involved cycles in $\overline{\pi}$ and in $\overline{\pi\beta}$ may vary, but this observation is enough to deduce that $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) \leq 4$, and as a result $\Delta f(\overline{\pi}, \overline{\pi\beta}) \in \{-1, 0\}$ (Lemma 6). The lowest value of $\Delta g(\overline{\pi}, \overline{\pi\beta})$ is obtained in two cases:
   (a) when $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) = 4$, in which case the leftmost cycle of $\overline{\pi}$ splits into two 1-cycles; therefore $\Delta f(\overline{\pi}, \overline{\pi\beta}) = -1$ and Equation 2 yields $\Delta g(\overline{\pi}, \overline{\pi\beta}) \geq 1 - 4 + 1 = -2$;
   (b) or when $\Delta c_1(\overline{\pi}, \overline{\pi\beta}) = 3$, in which case Equation 2 and Lemma 6 yield $\Delta g(\overline{\pi}, \overline{\pi\beta}) \geq 1 - 3 + 0 = -2$. □

Theorem 3 implies that Algorithm 1 is a 2-approximation for SBPBI. We note that the value of the unrestricted block-interchange distance (denoted by $bid(\pi)$) yields a trivial lower bound on *pbid* and can be computed in $O(n)$ time [10].

Theorem 4. *For any π in S_n, we have $pbid(\pi) \geq bid(\pi) = (n + 1 - c(G(\pi)))/2$.*

This lower bound often outperforms that of Theorem 3, but cases exist where the opposite holds ($\langle 1\ 4\ 3\ 2 \rangle$ is the smallest example; Fig. 2(*b*) shows its breakpoint graph). Unfortunately, a straightforward relationship linking $g(\pi)$ and $bid(\pi)$ has eluded us thus far. We will further comment on these lower bounds in section 7, where we discuss the performances of Algorithm 1 and comment on Pai and Chitturi's approximation [24], which is based on the lower bound provided by Theorem 4.

4. Tightening the bounds

Although obtaining better approximation guarantees for SBPBI seems as nontrivial as for other prefix sorting problems, the bounds obtained in the previous section can be improved. We show in this section how to tighten them.

By Theorem 3, the largest value by which the upper bound of Theorem 1 can decrease with a single prefix block-interchange is 2. In this section, we characterise all permutations that admit such a prefix block-interchange. Other nontight permutations exist (see e.g. Proposition 1), but they do not admit such an operation as the *first* step of an optimal sorting sequence. As a consequence, we obtain an improved upper bound on *pbid* in Theorem 5. The key to this improvement lies in the use of "short" cycles (i.e., cycles of length 2) in $G(\pi)$. We first show how to take advantage of 2-cycles intersecting the first grey edge.

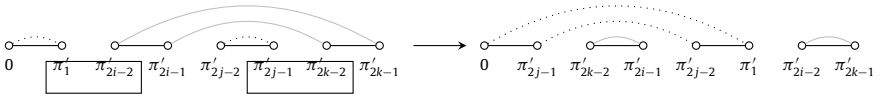
Lemma 7. *For any π in S_n: if $G(\pi)$ contains a 2-cycle that intersects the first grey edge, then there exists a prefix block-interchange β such that $\Delta g(\pi, \pi\beta) = -2$.*

Proof. Follows from cases 2 and 4 of the proof of Lemma 2, when the cycle that contains grey edge e' has length 2. □

Following Bafna and Pevzner [2], we say that a cycle C with b_i and b_k as black edges of minimum and maximum indices, respectively, *spans* a black edge b_j if $i < j < k$. We now show that some 2-cycles other than those that intersect the first grey edge are also helpful.

Lemma 8. *For any π in S_n: if $G(\pi)$ contains a 2-cycle which is not the leftmost cycle and which spans a black edge that belongs to a nontrivial cycle different from the leftmost cycle, then there exists a prefix block-interchange β such that $\Delta g(\pi, \pi\beta) = -2$.*

Proof. We apply a prefix block-interchange defined by the first black edge, both black edges of the 2-cycle, and any black edge spanned by the 2-cycle:



The number of cycles does not change, so $\Delta c(\pi, \pi\beta) = 0$. Either $\pi_1 = 1$, and then $\Delta c_1(\pi, \pi\beta) = 1$ and $\Delta f(\pi, \pi\beta) = 1$; or $\pi_1 \neq 1$, and then $\Delta c_1(\pi, \pi\beta) = 2$ and $\Delta f(\pi, \pi\beta) = 0$. In both cases, Equation 2 yields $\Delta g(\pi, \pi\beta) = -2$. □

2-cycles other than the leftmost cycle and in a different configuration from our characterisations are still helpful. We show that even though they do not allow us to apply a prefix block-interchange that decreases $g(\cdot)$ by 2 right away, they make it possible to obtain such an operation *eventually*.

Proposition 1. *For any π in S_n: if $G(\pi)$ contains a 2-cycle which is not the leftmost cycle, then π admits a sequence S of prefix-block interchanges that turns π into a permutation σ with $\Delta g(\pi, \sigma) = |S|$ and which admits a prefix block-interchange β such that $\Delta g(\sigma, \sigma\beta) = -2$.*

Proof. Let C denote the 2-cycle of interest. If C intersects the first grey edge or a cycle different from the leftmost cycle, then we are done (see respectively Lemma 7 and Lemma 8). Otherwise, C intersects another grey edge of the leftmost cycle, and Lemma 2 allows us to reduce $g(\pi)$ by one while reducing the length of the leftmost cycle without affecting C. Repeated applications of Lemma 2 eventually yield a permutation σ which satisfies one of the following conditions:

1. $\sigma_1 = 1$, in which case C necessarily spans a black edge that does not belong to the leftmost cycle and therefore we can apply Lemma 8;
2. $\sigma_1 \neq 1$ and C intersects another cycle than the leftmost cycle, in which case we can again apply Lemma 8; or
3. $\sigma_1 \neq 1$ and C intersects the first grey edge, in which case we can apply Lemma 7. □

The interactions between 2-cycles prevent us from simply reducing the upper bound of Theorem 1 by the number of 2-cycles in $G(\pi)$: indeed, the black edge spanned by the 2-cycle described in Lemma 8 may belong to a 2-cycle whose length will increase in the breakpoint graph of the resulting permutation. Therefore, we can only conclude the following.

Theorem 5. *For any π in S_n, we have $pbid(\pi) \leq g(\pi) - \lceil c_2^{\emptyset}(G(\pi))/2 \rceil$, where $c_2^{\emptyset}(G(\pi))$ denotes the number of 2-cycles in $G(\pi)$ excluding the leftmost cycle.*

Proof. We repeatedly apply Proposition 1 to take advantage of suitable 2-cycles. Each prefix block-interchange we use transforms a 2-cycle into two 1-cycles without affecting the other 2-cycles, except possibly in the case of Lemma 8 when the edge spanned by the 2-cycle we focus on belongs to another 2-cycle. In the worst case, every 2-cycle we try to split forces us to increase the length of a 2-cycle it intersects, hence the improvement of only $\lceil c_2^{\emptyset}(G(\pi))/2 \rceil$ over Theorem 1. □

Theorem 5 again yields a tight upper bound, as shown by the permutation $\langle 1\ 4\ 3\ 2 \rangle$ for which the value of the improved upper bound matches its distance (Fig. 2(*b*) shows its breakpoint graph).

Although an improvement over Theorem 1, Theorem 5 still overestimates the distance of some permutations. However, we note that the above results allow us to easily identify other nontight permutations (with respect to Theorem 1) whose breakpoint graph contains no 2-cycle. For instance, if the first grey edge intersects a 3-cycle C, then applying a prefix-block interchange selected according to Lemma 7 decreases the lengths of both the leftmost cycle and C, which becomes a 2-cycle and which therefore eventually allows for a prefix block-interchange that decreases $g(\cdot)$ by 2 according to Proposition 1.

Finally, we observe that the trivial lower bound $pbid(\pi) \geq bid(\pi)$ can be slightly improved.

Corollary 1. *[24] For any π in S_n and any prefix block-interchange β in S_n: if $\pi_1 = 1$, then $\Delta c(\pi, \pi\beta) \in \{-2, 0\}$.*

Corollary 2. *For any $\pi \neq \iota$ in S_n: if $\pi_1 = 1$, then $pbid(\pi) > bid(\pi)$.*

Proof. If $\pi_1 = 1$, then no prefix block-interchange can create cycles in $G(\pi)$ (Corollary 1), and therefore $bid(\pi\beta) \geq bid(\pi)$ for any prefix block-interchange β. The lower bound then follows from Theorem 4. □

5. Improved ratios on some classes

In this section we identify instances of SBPBI for which an approximation ratio smaller than 2 can be obtained. We begin with permutations whose breakpoint graph contains a unique nontrivial cycle, in which case we can compute the exact value of the prefix block-interchange distance.

Lemma 9. *For any π in S_n: if $c(G(\pi)) - c_1(G(\pi)) = 1$, then $pbid(\pi) = g(\pi)$.*

Proof. Theorem 4 yields the lower bound

$$bid(\pi) = \frac{n+1-c(G(\pi)}{2} = \frac{n+1-(1+c_1(G(\pi)))}{2} = \frac{n-c_1(G(\pi))}{2}.$$

If $\pi_1 \neq 1$, then Theorem 1 yields

$$\begin{aligned} pbid(\pi) \leq g(\pi) &= \frac{n+1+c(G(\pi))}{2} - c_1(G(\pi)) - 1 \\ &= \frac{n+1+c(G(\pi)) - c_1(G(\pi)) - c_1(G(\pi)) - 2}{2} = \frac{n-c_1(G(\pi))}{2} \end{aligned}$$

which matches the lower bound. If $\pi_1 = 1$, then $g(\pi) = bid(\pi) + 1$, and $pbid(\pi) > bid(\pi)$ (Corollary 2). □

Note that Algorithm 1 finds an optimal sorting sequence for the permutations described in Lemma 9: if $\pi_1 \neq 1$, then Algorithm 1 continually applies case 1 of Lemma 2 until $\pi = \iota$; and if $\pi_1 = 1$, then Algorithm 1 selects a prefix block-interchange β such that $\Delta g(\pi, \pi\beta) = -1$ and the leftmost cycle of $G(\pi\beta)$ is its only nontrivial cycle. We now turn to permutations whose breakpoint graph contains only cycles of bounded length.

Definition 3. [15] A permutation is *simple* if its breakpoint graph contains no cycle of length greater than 2.

Proposition 2. *Let π be a simple permutation. Then there is a 4/3-approximation for* SBPBI *on π.*

Proof. If $\pi_1 \neq 1$, then the leftmost cycle of $G(\pi)$ is a 2-cycle and intersects a 2-cycle (Lemma 1). Therefore, there exists a prefix block-interchange β such that $\Delta g(\pi, \pi\beta) = -2$ (Lemma 7). The resulting permutation fixes 1 and remains simple.

If $\pi_1 = 1$, then there exists a prefix block-interchange β such that $\Delta g(\pi, \pi\beta) = -2$ (Lemma 8). The leftmost cycle of $G(\pi\beta)$ has length 3, and the prefix block-transposition that acts on it decreases the value of $g(\pi\beta)$ by 1 and yields a simple permutation that fixes 1. Therefore, in the worst case, the value of $g(\pi)$ decreases by 3 using two prefix block-interchanges, which implies that every simple permutation can be sorted using $2g(\pi)/3$ operations. The approximation ratio of 4/3 then follows from Theorem 3. □

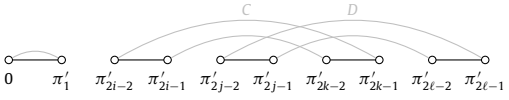
Algorithm 2 implements the approach outlined in Proposition 2. Pairs of black edges are always assumed to be sorted by increasing indices. Although the algorithm is designed for simple permutations, we allow the leftmost cycle to have length 3, since the algorithm can yield this case as an intermediate step. As in the case of Algorithm 1, Algorithm 2 can be implemented to run in time $O(n^2)$, and the same comments regarding the use of log-lists apply here.

The performances of Algorithm 2 can be improved further in the presence of *components* with more than two 2-cycles.

Definition 4. [15] Let π be a permutation. A *component* of $G(\pi)$ is a connected component of the intersection graph of the nontrivial cycles of $G(\pi)$.

Lemma 10. *Let $\pi \neq \iota$ be a simple permutation with $\pi_1 = 1$. If $G(\pi)$ contains a component $\mathscr{C}$ with $|\mathscr{C}| > 2$, then π admits a sequence S of three prefix block-interchanges such that $\Delta g(\pi, \pi S) = -6$.*

Proof. Our starting point is a pair $\{C = (c_1, c_2), D = (d_1, d_2)\}$ of intersecting 2-cycles in $\mathscr{C}$:



Algorithm 2: ApproximateSbpbiSimple(π).

Input: A permutation π of $[n]$ such that all cycles in $G(\pi)$ have length ≤ 2, with the possible exception of a leftmost cycle of length ≤ 3.
Output: A sorting sequence of prefix block-interchanges for π.

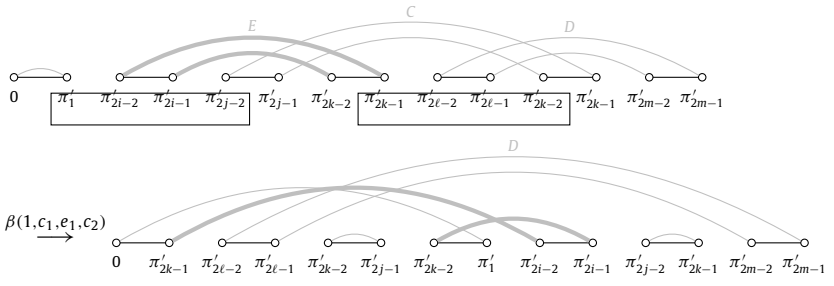
```
S ← empty sequence;
while π ≠ ι do
    if π_1 ≠ 1 then
        j ← the position of π_1 − 1;
        {1, j} intersects an inner grey edge {i, k} then
            σ ← β(1, i, j, k);
        else                                          // leftmost cycle has length 3
            b_1, b_i, b_j ← the three black edges of the leftmost cycle;
            σ ← β(1, i, i, j);
    else
        C = (b_i, b_k) ← a 2-cycle in G(π) such that i is minimal;
        D = (b_j, b_ℓ) ← any 2-cycle in G(π) with i < j < k < ℓ;
        σ ← β(1, i, j, k);
    π ← πσ;
    S.append(σ);
return S;
```

Depending on how we choose to apply Lemma 8, we can select either of the following prefix block-interchanges, both of which decrease $g(\pi)$ by 2:

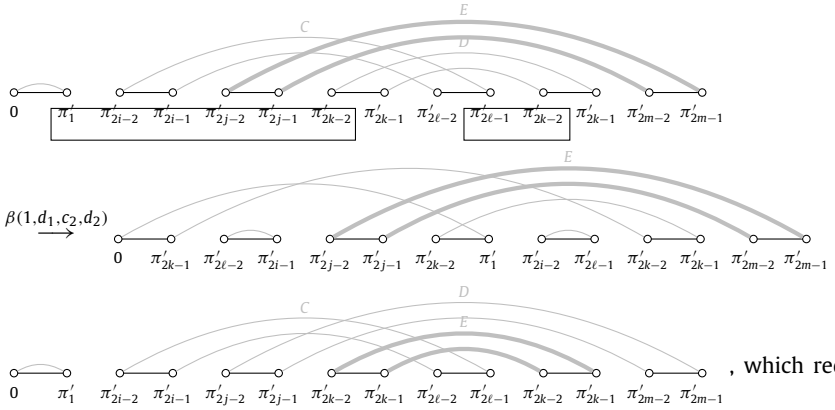
- a prefix block-interchange $\beta(1, c_1, d_1, c_2)$, which yields a leftmost 3-cycle whose first grey edge is the inner grey edge of D; or
- a prefix block-interchange $\beta(1, d_1, c_2, d_2)$, which yields a leftmost 3-cycle whose first grey edge is the outer grey edge of C.

The choice between either of the two options will be guided by our ability to obtain a 2-cycle that intersects the first grey edge of the leftmost 3-cycle in the resulting permutation. Let $E = (e_1, e_2)$ be another 2-cycle of $\mathscr{C}$; we distinguish between three families of cases depending on the interactions between C, D and E. If E intersects C but not D, then we are in one of the following three cases:

1.



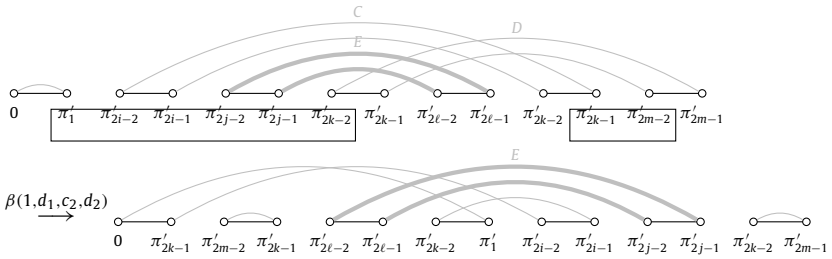
2.

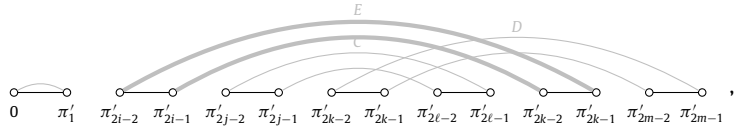


3.

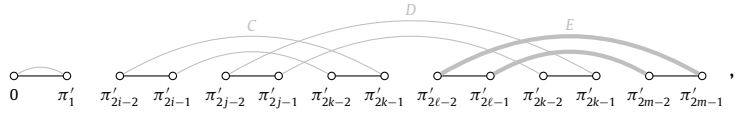
, which reduces to the previous case by swapping D and E.

If E intersects D but not C, then we are in one of the following three cases:

4. 

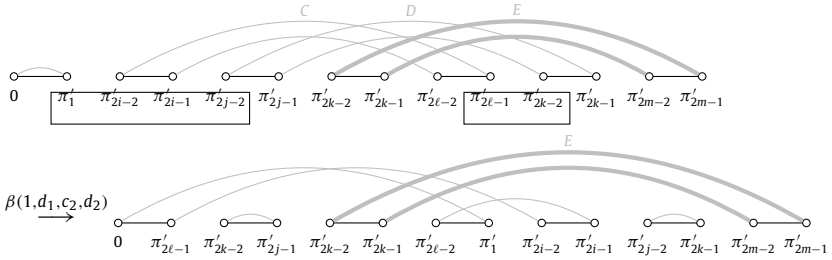
5. 

, which reduces to the previous case by swapping C and E.

6. 

, which reduces to case 1 by mapping C, D, E onto E, C, D, respectively.

Finally, if E intersects both C and D, then all remaining cases reduce to the following one:

7. 

In all cases, the leftmost cycle of the resulting breakpoint graph has length 3, and its first grey edge intersects a 2-cycle. We can therefore apply a second prefix block-interchange that decreases the value of $g(\cdot)$ by 2 (Lemma 7). Since the leftmost cycle of the resulting breakpoint graph has length 2, and since the resulting permutation is simple, a third prefix block-interchange can be applied to reduce the value of $g(\cdot)$ by 2 (Lemma 7) again, which yields the wanted optimal sequence of length 3. □

As a straightforward consequence of Lemma 10, we can sort simple permutations that fix 1 and whose breakpoint graph is a concatenation of components of size 4, possibly interspersed with trivial cycles, optimally. Unfortunately, predicting the evolution of components under block-interchanges is difficult, and it is unclear yet how this result might lead to an approximation ratio smaller than 4/3.

6. The maximum value of the prefix block-interchange distance

The *diameter* of S_n is the maximum value that a distance can reach for a particular family of operations. In this section, we obtain an upper bound on its value in the case of prefix block-interchanges, and show along the way that our 2-approximation algorithm based on the breakpoint graph is also a 2-approximation with respect to the following notion.

Definition 5. [11] Let π be a permutation of $\{0, 1, 2, \ldots, n+1\}$ with $\pi_0 = 0$ and $\pi_{n+1} = n+1$. The pair (π_i, π_{i+1}) with $0 \leq i \leq n$ is a *breakpoint* if $i = 0$ or $\pi_{i+1} - \pi_i \neq 1$, and an *adjacency* otherwise. The number of breakpoints in a permutation π is denoted by $b(\pi)$.

For readability, we slightly abuse notation by using $b(\pi)$ for π in S_n, with the understanding that it refers to $b(\langle 0\ \pi_1\ \pi_2\ \cdots\ \pi_n\ n+1\rangle)$. We let $\Delta b(\pi, \sigma) = b(\sigma) - b(\pi)$, and say that a prefix block-interchange β with $\Delta b(\pi, \pi\beta) < 0$ *removes* breakpoints, or *creates* adjacencies.

Lemma 11. *For any π in S_n and any prefix block-interchange β, we have $|\Delta b(\pi, \pi\beta)| \leq 3$.*

Table 2
The number of permutations π in S_n with $pbid(\pi) = k$.

$n \setminus k$	0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	4	1	0	0	0	0	0
4	1	10	13	0	0	0	0	0
5	1	20	78	21	0	0	0	0
6	1	35	308	375	1	0	0	0
7	1	56	938	3 264	781	0	0	0
8	1	84	2 394	18 452	19 357	32	0	0
9	1	120	5 376	78 220	231 724	47 438	1	0
10	1	165	10 956	270 082	1 766 184	1 579 231	2 181	0
11	1	220	20 691	799 810	9 870 456	24 790 471	4 435 129	22
12	1	286	36 751	2 102 188	43 989 003	243 298 111	189 361 249	214 011

Proof. A prefix block-interchange β acts on at most four pairs of adjacent elements, including the pair $(0, \pi_1)$ which is a breakpoint. Therefore, the number of breakpoints that β can remove or create lies in the set $\{0, 1, 2, 3\}$. □

Since ι is the only permutation with exactly one breakpoint, Lemma 11 immediately implies the following corollary.

Corollary 3. *For any π in S_n : $pbid(\pi) \geq \left\lceil \frac{b(\pi)-1}{3} \right\rceil$.*

Lemma 12. *For any π in S_n, we have $pbid(\pi) \leq 2 \left\lceil \frac{b(\pi)-1}{3} \right\rceil$.*

Proof. Assume $\pi \neq \iota$ to avoid triviality, and observe that adjacencies in $\langle 0\ \pi_1\ \pi_2\ \cdots\ \pi_n\ n+1 \rangle$ are in one-to-one correspondence with trivial cycles in $G(\pi)$ (except for the pair $(0, \pi_1)$ which by Definition 5 is always a breakpoint). If $\pi_1 \neq 1$, then Lemma 2 guarantees the existence of a prefix block-interchange β with $\Delta c_1(\pi, \pi\beta) \geq 2$ and in turn implies $\Delta b(\pi, \pi\beta) \geq 2$. If $\pi_1 = 1$, then we select β as in the proof of Theorem 1, which creates a new trivial cycle in $G(\pi\beta)$ that corresponds to a new adjacency in $\pi\beta$. Since $\pi\beta_1 \neq 1$, the previous case provides the next operation, and the number of breakpoints decreases by at least three using two prefix block-interchanges. □

Since $b(\pi) \leq n + 1$ for all π in S_n, we immediately obtain the following.

Corollary 4. *For any $\pi \in S_n$, we have $pbid(\pi) \leq 2n/3$.*

Table 2 shows the distribution of the prefix block-interchange distance for $n \leq 12$. The last line in particular shows that the upper bound of Corollary 4 is not tight.

7. On the performances of approximations for SBPBI

In this section, we examine the performances of our algorithm in practice using our freely available implementation, and comment on Pai and Chitturi's algorithm [24].

7.1. Practical assessment of Algorithm 1

Fig. 3 shows the behaviour of Algorithm 1 on all permutations of size at most 12. We compare the quality of our solutions to all known lower bounds, as well as to the exact distance. Each area in these plots corresponds to the percentage of permutations for which Algorithm 1 returns a solution with approximation ratio k, where the ratio is measured with respect to the lower bounds of Theorem 3, Corollary 3, Theorem 4, Corollary 2, and to the actual distance, respectively.

Fig. 3 allows us to make a few observations:

1. as expected, each lower bound provides a different lens through which the approximation guarantee can be examined. It appears that the lower bound of Theorem 3 is very pessimistic, whereas the lower bound of Theorem 4, on which Pai and Chitturi's algorithm [24] is based, yields a more satisfying outlook on the performances of Algorithm 1.
2. the performances when comparing Algorithm 1's results to the actual distance of each permutation are even more encouraging: the worst ratio in practice is actually 3/2, and Algorithm 1 finds an optimal solution for more than 80% of all instances. Performances do decrease as n increases, but the degradation is far less steep than when comparing the results against the lower bounds.
3. the comparison against the actual distance also reveals the emergence of increasingly larger areas with increasingly smaller approximation ratios; that is, as n increases, the proportion of instances we solve to optimality decreases, but

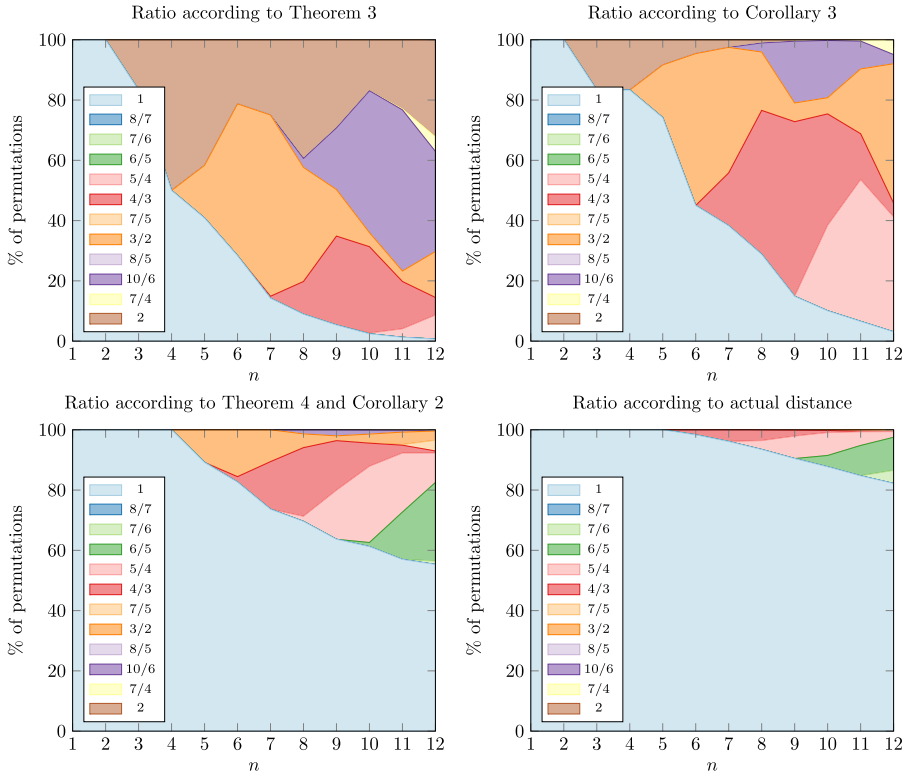


Fig. 3. The performances of Algorithm 1 on all permutations of n elements, for $1 \leq n \leq 12$. Each area depicts the percentage of permutations for which the approximation returns a solution whose quality is the specified value.

the instances on which the algorithm fails to find an optimal solution yield ratios smaller than 3/2 which keep getting closer to 1.

7.2. Comments on the approximation of Pai and Chitturi

We briefly remind the reader how the 2-approximation algorithm of Pai and Chitturi [24] works, avoiding technical definitions which we will not use. The algorithm is based on the lower bound of Theorem 4: the main idea is to try at every step to find a *2-move*, which is a prefix block-interchange that increases the number of cycles in $G(\pi)$ by 2; they show in their paper that such a 2-move always exists, provided that $\pi_1 \neq 1$. If $\pi_1 = 1$, and $\pi \neq \iota$, then no 2-move exists, but it is possible to find a prefix block-interchange that moves 1 out of the way without changing the number of cycles (i.e., that operation is a *0-move*), and as a result, produces a permutation which admits a 2-move. The 0-moves they apply also preserve the number of trivial cycles. Therefore, in the worst case, the number of cycles increases by only 2 with a sequence of two prefix block-interchanges, and the approximation ratio of 2 follows. We are not aware of an implementation of Pai and Chitturi's algorithm [24], and can therefore not provide extensive practical performance comparisons to our algorithm. We can nevertheless make a few hopefully interesting comments and suggestions for improvements.

First, Pai and Chitturi's algorithm uses the lower bound of Theorem 4, and therefore focuses solely on splitting cycles in the breakpoint graph. That strategy yielded an exact polynomial-time algorithm in the case of sorting by *unrestricted* block-interchanges; but as our bounds reveal (Theorem 3 and Theorem 1), splitting cycles is only beneficial when the newly extracted cycles are trivial, which Algorithm 1 takes into account. Note that this observation applies to other prefix sorting problems as well, since like-minded bounds have been obtained in those cases too (see e.g. SORTING BY PREFIX TRANSPOSITIONS [20] and SORTING BY PREFIX SIGNED REVERSALS [22]).

Second, care should be taken so as to avoid the problematic case $\pi_1 = 1$ as often as possible. To illustrate, let us examine the permutation $\pi = \langle \frac{n}{2}+1\ 1\ \frac{n}{2}+2\ 2\ \frac{n}{2}+3\ 3\ \cdots\ n\ \frac{n}{2} \rangle$ for any even $n \geq 4$. The breakpoint graph of this permutation contains a single, nontrivial cycle, and therefore $pbid(\pi) = n/2$ (Lemma 9). However, Pai and Chitturi's algorithm may exchange $\frac{n}{2}+1$ and 1, which is indeed a 2-move but yields a permutation that fixes 1. This mistake might be repeated and, as a result, yield a sequence whose length is asymptotically close to their announced ratio of 2. Fig. 4 compares the resulting sequence to an optimal solution in the case where π has 8 elements.

Third, our bounds also reveal that, somehow counter-intuitively, situations exist where progress can be achieved by applying -2-*moves*, i.e., operations that *decrease* the number of cycles by merging them. Such moves are never considered by Pai and Chitturi's algorithm, which will therefore miss the optimal solution.

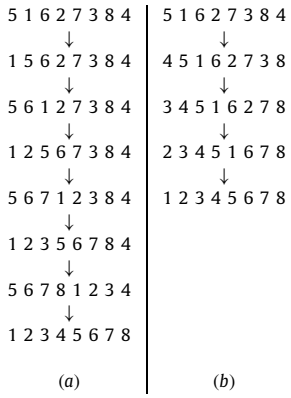


Fig. 4. An asymptotically tight example for Pai and Chitturi's algorithm with respect to the actual distance. (*a*) A sequence that their algorithm might produce; (*b*) an optimal sorting sequence.

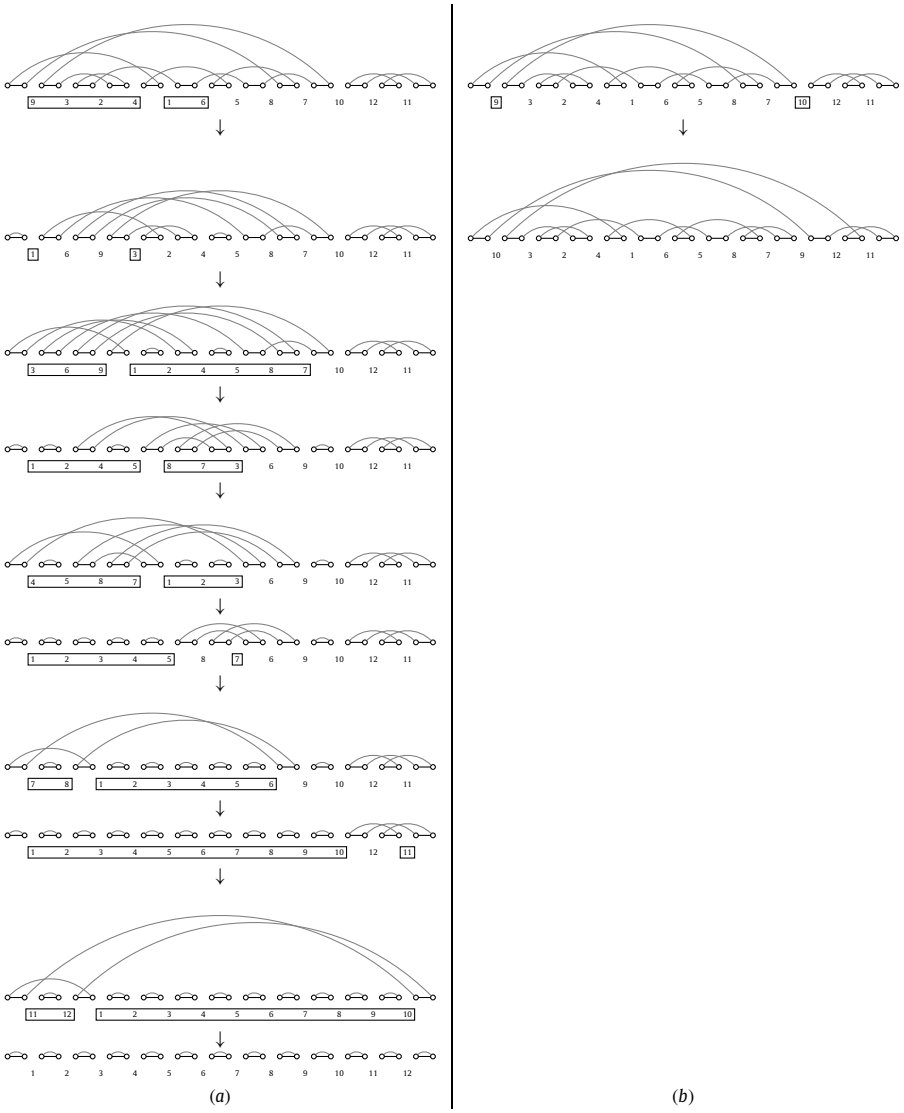


Fig. 5. An example where Pai and Chitturi's algorithm fails to find the optimal solution partly due to its disregard for operations that merge three cycles. (*a*) A sorting sequence of length 9 that their algorithm might produce; (*b*) the first move of an optimal sorting sequence, which merges the three cycles in $G(\pi)$ and produces a permutation with distance 6 (Lemma 9), thereby totalling 7 operations.

As an example, consider $\pi = \langle 9\ 3\ 2\ 4\ 1\ 6\ 5\ 8\ 7\ 10\ 12\ 11\rangle$. Fig. 5(*a*) shows a sorting sequence that Pai and Chitturi's algorithm may produce for π, along with the breakpoint graph at each step (without vertex labels for clarity). Every operation in that sequence either splits the leftmost cycle if possible, or combines the trivial leftmost cycle with a nontrivial one so as to obtain a nontrivial leftmost cycle and a trivial cycle elsewhere. By contrast, as Fig. 5(*b*) shows, applying an operation that merges the three cycles into one produces a permutation with distance 6, and this strategy turns out to be optimal. Note that π is but one of over 200 000 permutations of size 12 with distance 7, and more difficult instances are likely to arise as n further increases. Moreover, although the solution we show is optimal, we have no proof that there exists a permutation for which such merging moves are *required* (in the sense that no optimal solution can be reached without them).

8. Conclusions and future work

We initiated in this work the study of sorting permutations by prefix block-interchanges, an operation that generalises several well-studied operations in genome rearrangements and interconnection network design. We gave tight upper and lower bounds on the corresponding distance, derived a 2-approximation algorithm for the problem, as well as a 4/3-approximation for simple permutations, and an exact solution for some simple permutations as well as permutations with only one nontrivial cycle in their breakpoint graph. We also obtained an upper bound on the maximum value that the distance can reach, and finally, we examined the performances of our 2-approximation in practice and put it in perspective with that of Pai and Chitturi [24]. We outline below several leads for further investigations.

Hardness and approximability The complexity of SBPBI remains open, and existing proofs for similar problems do not seem to be adaptable. One possible lead might be to further investigate the puzzling instances discussed in subsection 7.2, and to understand when obtaining an optimal solution *requires* the use of merging operations. As far as approximability is concerned: can a ratio smaller than 2 be obtained? We note that improving this ratio will probably require improved lower bounds, since for all three upper bounds we have obtained (Theorem 1, Theorem 5 and Lemma 12) there are permutations whose actual distance matches those bounds (which does not prevent approximations from achieving better performances against the actual distance). Note that no approximation with a ratio smaller than 2 is known for any of those prefix sorting problems whose complexity remains open, except for prefix double-cut-and-joins [14]. We have seen (subsection 7.1) that in practice, at least on instances of size ≤ 12, Algorithm 1's performance ratio against the actual distance never exceeds 3/2. It may be possible to improve its performances further by combining it with Algorithm 2, although whether or not this will lead to further theoretical improvements is unclear.

Bounded cycle lengths We have shown that 2-cycles were helpful in obtaining improved upper bounds in the general case, and that the approximation ratio can be lowered from 2 to 4/3 for permutations whose breakpoint graph contains no k-cycle for $k > 2$. Can a ratio smaller than 4/3 be obtained for simple permutations, and can we obtain similar results for larger values of k? Note that simple permutations led to a polynomial-time algorithm for sorting by signed reversals [15], but we do not expect such an outcome for prefix block-interchanges since the simplification process does not preserve the prefix block-interchange distance (whereas it did preserve the signed reversal distance): the smallest counterexample is $\pi = \langle 3\ 1\ 4\ 2\rangle$, which simplifies to $\sigma = \langle 5\ 2\ 7\ 4\ 1\ 6\ 3\rangle$, and for which $pbid(\pi) = 2 \neq pbid(\sigma) = 3$.

Further tractable instances Finally, if a polynomial-time algorithm for SBPBI keeps eluding us, it would be valuable to extend the list of instances we can solve in polynomial time. For instance, Pai and Chitturi [24] show how to sort $R_n = \langle n\ n-1\ \cdots\ 2\ 1\rangle$ optimally, and we have shown that some subclasses of simple permutations, as well as permutations with $c(G(\pi)) - c_1(G(\pi)) = k$ for $k = 1$, are tractable instances (note that R_n satisfies this criterion when n is even); can positive results for larger values of k be obtained?

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A link to my source code is provided in the paper

References

[1] S.B. Akers, B. Krishnamurthy, D. Harel, The star graph: an attractive alternative to the n-cube, in: Proceedings of the Fourth International Conference on Parallel Processing, Pennsylvania State University Press, 1987, pp. 393–400, https://doi.org/10.5555/201173.201191.

[2] V. Bafna, P.A. Pevzner, Sorting by transpositions, SIAM J. Discrete Math. 11 (2) (1998) 224–240, https://doi.org/10.1137/S089548019528280X (electronic).

[3] P. Berman, S. Hannenhalli, M. Karpinski, 1.375-approximation algorithm for sorting by reversals, in: R.H. Möhring, R. Raman (Eds.), Proceedings of the 10th Annual European Symposium on Algorithms, in: Lecture Notes in Computer Science, vol. 2461, Springer, Rome, Italy, 2002, pp. 200–210, https://doi.org/10.1007/3-540-45749-6_21.

[4] L. Bulteau, G. Fertin, I. Rusu, Sorting by transpositions is difficult, SIAM J. Discrete Math. 26 (3) (2012) 1148–1180, https://doi.org/10.1137/110851390.
[5] L. Bulteau, G. Fertin, I. Rusu, Pancake flipping is hard, J. Comput. Syst. Sci. 81 (8) (2015) 1556–1574, https://doi.org/10.1016/j.jcss.2015.02.003.
[6] L. Cai, J. Chen, R.G. Downey, M.R. Fellows, On the parameterized complexity of short computation and factorization, Arch. Math. Log. 36 (4–5) (1997) 321–337, https://doi.org/10.1007/s001530050069.
[7] A. Caprara, Sorting permutations by reversals and Eulerian cycle decompositions, SIAM J. Discrete Math. 12 (1) (1999) 91–110, https://doi.org/10.1137/S089548019731994X (electronic).
[8] X. Chen, On sorting unsigned permutations by double-cut-and-joins, J. Comb. Optim. 25 (3) (2013) 339–351, https://doi.org/10.1007/s10878-010-9369-8.
[9] S. Chou, C. Yang, K. Chen, C.L. Lu, Prefix block-interchanges on binary strings, in: W.C. Chu, H. Chao, S.J. Yang (Eds.), Proceedings of the International Computer Symposium on Intelligent Systems and Applications, in: Frontiers in Artificial Intelligence and Applications, vol. 274, IOS Press, Taichung, Taiwan, 2014, pp. 1960–1969, https://doi.org/10.3233/978-1-61499-484-8-1960.
[10] D.A. Christie, Sorting permutations by block-interchanges, Inf. Process. Lett. 60 (4) (1996) 165–169, https://doi.org/10.1016/S0020-0190(96)00155-X.
[11] Z. Dias, J. Meidanis, Sorting by prefix transpositions, in: A.H.F. Laender, A.L. Oliveira (Eds.), Proceedings of the Ninth International Symposium on String Processing and Information Retrieval, in: Lecture Notes in Computer Science, vol. 2476, Springer-Verlag, Lisbon, Portugal, 2002, pp. 65–76.
[12] I. Elias, T. Hartman, A 1.375-approximation algorithm for sorting by transpositions, IEEE/ACM Trans. Comput. Biol. Bioinform. 3 (4) (2006) 369–379, https://doi.org/10.1109/TCBB.2006.44.
[13] G. Fertin, A. Labarre, I. Rusu, E. Tannier, S. Vialette, Combinatorics of Genome Rearrangements, Computational Molecular Biology, The MIT Press, 2009, https://mitpress.mit.edu/books/combinatorics-genome-rearrangements.
[14] G. Fertin, G. Jean, A. Labarre, Sorting genomes by prefix double-cut-and-joins, in: D. Arroyuelo, B. Poblete (Eds.), String Processing and Information Retrieval - 29th International Symposium, Proceedings, SPIRE 2022, Concepción, Chile, November 8-10, 2022, in: Lecture Notes in Computer Science, vol. 13617, Springer, 2022, pp. 178–190, https://doi.org/10.1007/978-3-031-20643-6_13.
[15] S. Hannenhalli, P.A. Pevzner, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, J. ACM 46 (1) (1999) 1–27, https://doi.org/10.1145/300515.300516.
[16] M.R. Jerrum, The complexity of finding minimum-length generator sequences, Theor. Comput. Sci. 36 (2–3) (1985) 265–289, https://doi.org/10.1016/0304-3975(85)90047-7.
[17] D.J. Kleitman, E. Kramer, J.H. Conway, S. Bell, H. Dweighter, Elementary problems: E2564-E2569, Am. Math. Mon. 82 (10) (1975) 1009–1010, https://doi.org/10.2307/2318260.
[18] D.E. Knuth, Sorting and Searching, The Art of Computer Programming, vol. 3, Addison-Wesley, 1995.
[19] A. Labarre, New bounds and tractable instances for the transposition distance, IEEE/ACM Trans. Comput. Biol. Bioinform. 3 (4) (2006) 380–394, https://doi.org/10.1109/TCBB.2006.56.
[20] A. Labarre, Lower bounding edit distances between permutations, SIAM J. Discrete Math. 27 (3) (2013) 1410–1428, https://doi.org/10.1137/13090897X.
[21] A. Labarre, Sorting by prefix block-interchanges, in: Y. Cao, S.W. Cheng, M. Li (Eds.), 31st International Symposium on Algorithms and Computation (ISAAC), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 181, 2020, 55, https://doi.org/10.4230/LIPIcs.ISAAC.2020.55, https://drops.dagstuhl.de/opus/volltexte/2020/13399.
[22] A. Labarre, J. Cibulka, Polynomial-time sortable stacks of burnt pancakes, Theor. Comput. Sci. 412 (8–10) (2011) 695–702, https://doi.org/10.1016/j.tcs.2010.11.004.
[23] S. Lakshmivarahan, J.S. Jwo, S.K. Dhall, Symmetry in interconnection networks based on Cayley graphs of permutation groups: a survey, Parallel Comput. 19 (4) (1993) 361–407, https://doi.org/10.1016/0167-8191(93)90054-O.
[24] J. Pai, B. Chitturi, Approximation algorithms for sorting permutations by extreme block-interchanges, Theor. Comput. Sci. 891 (2021) 105–115, https://doi.org/10.1016/j.tcs.2021.08.031, https://www.sciencedirect.com/science/article/pii/S0304397521005089.
[25] I. Rusu, Log-lists and their applications to sorting by transpositions, reversals and block-interchanges, Theor. Comput. Sci. 660 (2017) 1–15, https://doi.org/10.1016/j.tcs.2016.11.012.
[26] S. Yancopoulos, O. Attie, R. Friedberg, Efficient sorting of genomic permutations by translocation, inversion and block interchange, Bioinformatics 21 (16) (2005) 3340–3346, https://doi.org/10.1093/bioinformatics/bti535.