Nordic Journal of Computing

# Block Sorting: A Characterization and some Heuristics

Meena Mahajan
*The Institute of Mathematical Sciences,*
*Chennai 600 113, India.*
meena@imsc.res.in

Raghavan Rama
*Department of Mathematics,*
*Indian Institute of Technology, Madras,*
*Chennai 600 036, India.*
ramar@iitm.ac.in

S. Vijayakumar
*Indian Institute of Science Education and Research,*
*Pune 411 008, India.*
s.vijay@iiserpune.ac.in.

**Abstract.** Given a permutation $\pi$, the problem is to find a shortest series of block moves which, when applied in succession, sorts $\pi$. Here a block is a maximal substring of successive integers in order, and a block move is the displacement of a block to a location where it merges with another block. is an NP-hard optimization problem [3] and has a factor 2 approximation algorithm [18]. In this paper, we present a combinatorial characterization of optimal solutions of and use it to prove various computationally important properties of the problem. In particular, we identify certain block moves that are provably optimal. We also establish the equivalence of block sorting and a combinatorial puzzle.

We consider several polynomial-time heuristics for that are inspired either by the above-mentioned combinatorial characterization, or by the approach of [18] that was based on the problem, or both. Although these heuristics seem to be promising candidates for improving the approximation ratio (their approximation ratios are provably at most 2), we show that none of them leads to a better approximation ratio than 2.

**ACM CCS Categories and Subject Descriptors:** F.2.2 Analysis of Algorithms and Problem Complexity

**Key words:** Block Sorting, Approximation, Heuristics, Combinatorial Characterization

## 1. Introduction

Let $\pi$ be a permutation on $n$ elements, $\pi \in S_n$, written as a string $\pi_1\pi_2\dots\pi_n$. A *block* is a maximal substring of $\pi$ which is also a substring of the identity permutation $\mathtt{id}_n = 1\ 2\ \dots\ n$. (For example, in the permutation 6 2 3 4 1 5 on 6 elements, there are four blocks, and $\boxed{2\ 3\ 4}$ is the only block containing more than a single element.) A block move is the operation of picking a block and placing it adjacent to its predecessor or successor block so that it merges with it to form a larger block. (For instance, a block move of 5 in 6 2 3 4 1 5 will result in either 5 6 2 3 4 1 or 6 2 3 4 5 1.) The problem is to find a shortest series of block moves

Received January 28, 2008.

which, when applied in succession, sorts $\pi$. The length of this shortest series is denoted by $\mathtt{bs}(\pi)$ and is called the block sorting distance of $\pi$.

(Note: In a preliminary version [19] of this paper, as well as in [17] which is a preliminary version of [18], blocks are referred to as *strips*. We use the term blocks here to conform with [3], where NP-hardness of the problem is established.)

The problem was studied in connection with optical character recognition (OCR); see, for instance, [11, 15]. A recent paper by Bein, Larmore, Latifi, and Sudborough establishes that the decision version of this problem is NP-complete [3]. The problem also gains in importance due to the fact that it is a nontrivial variant of another well-known sorting problem, *Sorting by Transpositions*, , that arises in the study of genome rearrangements. A transposition is the operation of picking any substring of $\pi$ and placing it elsewhere in the string. The problem is the problem of finding a shortest series of transpositions that sorts a given permutation $\pi$; the length of this shortest series is denoted by $t(\pi)$. This problem has some 3/2-approximation algorithms, and most recently an 11/8-approximation has been obtained [7], but its computational complexity remains unsettled [2, 5, 9, 13, 14]. For related work, see [1, 5, 8, 10, 12, 16, 20].

The problem has a fairly straightforward 3-approximation algorithm. Let $\mathtt{\#block}(\pi)$ denote the number of blocks in $\pi$. It is easy to see that a block move, and even a transposition, can reduce the number of blocks at most by three. And a sorted permutation consists of one block. Thus reducing the number of blocks to 1 needs at least $(\mathtt{\#block}(\pi) - 1)/3$ transpositions. On the other hand, the naive algorithm of repeatedly moving the block containing 1 reduces the number of blocks by at least one in each move; hence $\mathtt{bs}(\pi)$ is no more than $\mathtt{\#block}(\pi) - 1$. Thus

$$\left\lceil \frac{\mathtt{\#block}(\pi) - 1}{3} \right\rceil \leq t(\pi) \leq \mathtt{bs}(\pi) \leq \mathtt{\#block}(\pi) - 1 \qquad (1)$$

It follows that the naive algorithm is a 3-approximation for (and for ).

The first nontrivial approximation algorithm for was obtained by Mahajan, Rama, Raman, and Vijayakumar and achieves a performance ratio of 2 [17, 18]. This is accomplished by introducing a similar problem, , that is about optimally merging a given set of increasing sequences into one increasing sequence by block moves (see Section 2 for details). A faster 2 approximation algorithm was developed later by Bein, Larmore, Morales, and Sudborough [4].

In this paper, we continue the task of exploring the combinatorial structure of optimal solutions of . We show that the problem of block sorting a given $\pi$ is equivalent to computing a largest *compatible edge set* in an associated *order graph* $G_\pi$ (Theorem 3). We note that one direction of this equivalence is established in [3] as well, but without defining compatible sets independent of block move sorting sequences as presented in this paper.

We also establish an optimum-preserving equivalence between and the following combinatorial puzzle. Let $\pi, \phi$ be a pair of permutations of the same length, viewed as strings, with $\pi \neq \phi$. Suppose we wish to quantify how different these strings are. One way to do this is the following: identify a substring common to both $\pi$ and $\phi$ and delete it. Repeat this process until the left-over strings become identical. How many such deletions are needed? We denote the minimum number of deletions needed by $\mathsf{csr}(\pi, \phi)$, and we define the *Common Substring Removals* problem, , as the problem of computing $\mathsf{csr}(\pi, \phi)$ for a given $\pi, \phi$. For example, consider the pair $\pi = 3\ 5\ 1\ 4\ 2\ 6$ and $\phi = 1\ 4\ 3\ 2\ 5\ 6$ from $S_6$. An optimal common substring removal sequence that makes this pair identical is ($\mathbf{3}$ 5 1 4 2 6, 1 4 $\mathbf{3}$ 2 5 6) $\rightarrow$ (5 $\mathbf{1\ 4\ 2}$ 6, $\mathbf{1\ 4\ 2}$ 5 6) $\rightarrow$ (5 6, 5 6). Theorem 4 establishes that $\mathsf{csr}(\pi, \phi) = \mathsf{bs}(\pi^{-1}\phi)$. An interesting consequence of this equivalence is that sorting a permutation via block moves is as hard (or as easy) as sorting its inverse (Corollary 3).

Using the characterization of block sorting via compatible sets, we establish that certain block moves are optimal: they necessarily reduce the block sorting distance. We show that a block move that reduces the number of blocks in the permutation by 2 or 3 is optimal (Theorem 5). We also show as a corollary of a more general result that if two adjacent blocks are consecutive but out of order, then a block move that exchanges them is optimal (Theorem 6). Thus, for instance, $\mathsf{bs}(6\ 4\ \boxed{7}\ 5\ 1\ 2\ 3) > \mathsf{bs}(6\ 7\ 4\ 5\ \boxed{1\ 2\ 3}) > \mathsf{bs}(6\ 7\ 1\ 2\ 3\ 4\ 5)$.

Finally, we revisit the problem and consider some heuristics for that are inspired either by block merging based approach of [18] or the combinatorial characterization obtained in this paper, or both. Initially we consider minor refinements of the algorithm of [18], and this leads to the construction of more robust tight examples for the latter. The major heuristic we discuss tries to overcome the disadvantage of the idea, used in [18], of breaking permutations into pieces (increasing runs), while at the same time using the information that may be gained from the corresponding instances. Despite its promising outlook, we show that even this heuristic is asymptotically no better than a factor 2 algorithm. The tight examples constructed to demonstrate this fact naturally suggest a generalization of the notion of noncrossing sets to strongly compatible sets. We show that even strong compatible sets cannot, in general, guarantee an approximation factor any better than 2, and so a heuristic based on these would be no better. (It is an independent issue that as yet, we do not even know how to compute largest strongly compatible sets.) But we conjecture that at least one of $\pi$ and $\pi^{-1}$ will always have a sufficiently large strong compatible set to ensure a better approximation for $\mathsf{bs}(\pi) = \mathsf{bs}(\pi^{-1})$.

This paper is organized as follows. Section 2 briefly reviews notions concerning from [17, 18], as these are central to the discussion in this paper. Section 3 establishes the combinatorial characterization of $\mathsf{bs}(\pi)$, while Section 4 shows that and are equivalent. Section 5 uses the characterization of Section 3 to show optimality of certain block moves. Finally, Section 6 examines the proposed heuristics and provides tight examples for them all.

## 2. Reviewing

Let $\mathbb{S} = \{S_1, S_2, \ldots, S_k\}$ be a set of disjoint increasing sequences whose union is $[n]$, where $[n]$ denotes the set $\{1, 2, \ldots, n\}$. (Technically, $\mathbb{S}$ is a multiset since more than one sequence could be empty.) A block in $\mathbb{S}$ is defined to be a maximal substring of the identity permutation $\mathtt{id}_n$ which is also a substring of some $S_i$. A block move on $\mathbb{S}$ consists of removing a block from some sequence $S_i$ and inserting it into some other sequence $S_j$ so that it merges with some block there to form a longer block. (Note that consequently, the new sequence $S'_j$ is still an increasing sequence.)

The problem can be stated as follows: Given a set $\mathbb{S}$ of disjoint increasing sequences whose union is $[n]$, transform $\mathbb{S}$ to the set $\mathbb{M}_n = \{\mathtt{id}_n, \epsilon, \ldots, \epsilon\}$ via the fewest possible block moves. The number of block moves in such a shortest series is denoted by $\mathsf{bm}(\mathbb{S})$.

For example, $\mathbb{S} = \{1\ 4\ 6, 2\ 3\ 8, 5\ 7\}$ is a valid instance of the problem. It has three sequences, and seven blocks. $\boxed{2\ 3}$ is the only block containing more than one element. It can be transformed using four block moves, as follows: Move block 6 to get $\{1\ 4, 2\ 3\ 8, 5\ 6\ 7\}$, then move block 2 3 to get $\{1\ 2\ 3\ 4, 8, 5\ 6\ 7\}$, then move block 5 6 7 to get $\{1\ 2\ 3\ 4\ 5\ 6\ 7, 8, \epsilon\}$, and finally move block 8 to get $\{\mathtt{id}_8, \epsilon, \epsilon\}$.

Given an instance $\mathbb{S}$ of , an associated directed graph $G$ is built as follows: $G$ has $n$ vertices numbered 1 through $n$. The multiset $\mathbb{S}$ having $k$ sequences partitions the vertices into $k$ parts, and the edges of $G$ form a total order on each part. Formally,

D 1. (D 4.1 [18]) *Let $\mathbb{S} = \{S_1, S_2, \ldots, S_k\}$. Then*

*(1) $G = (V, E)$ where $V = [n]$, and $E = \{(u, v) \mid u < v,\ \ \exists p \in [k] :\ u, v \in S_p\}$.*

*(2) Edges in $E$ of the form $(i, i+1)$ are called unit edges.*

*(3) Distinct edges $(i, j)$ and $(k, l)$ are said to* cross *if $i \leq k < j \leq l$ or $k \leq i < l \leq j$. A set $E' \subseteq E$ is said to be a non-crossing set if no two edges in $E'$ cross. (Note that $(i, j)$ and $(i, k)$ cross, but not $(i, j)$ and $(k, l)$ where $i < k < l < j$. Also, edges $(i, k)$ and $(k, l)$ do not cross.)*

*(4) By $c(\mathbb{S})$ we denote the size of the largest non-crossing set in the edge set of $G$.*

Given any non-crossing set $C$, call a block $\alpha$ free with respect to $C$ if no edge of $C$ has exactly one end-point in $\alpha$. The main idea of [18] is that as long as $\mathbb{S} \neq \mathbb{M}_n$, a free block can always be found, and moving a free block necessarily reduces $\mathsf{bm}(\mathbb{S})$. Using this, the following is established:

L 1. (L 4.4 [18]) $\mathsf{bm}(\mathbb{S}) = n - 1 - c(\mathbb{S})$.

T 1. (T 3.3 [18]) *is in P.*

Any permutation $\pi$ can be uniquely decomposed into maximal increasing substrings. Considering each of these substrings as a sequence gives an instance $\mathbb{S}_\pi$ of . The major result from [18] that also we use is that $\mathsf{bm}(\mathbb{S}_\pi)$ approximates $\mathsf{bs}(\pi)$.

T 2. (L 5.3 5.4 [18]) $\mathsf{bs}(\pi) \leq \mathsf{bm}(\mathbb{S}_\pi) \leq 2\mathsf{bs}(\pi)$.

## 3. Compatible edge sets and optimal solutions of

In this section, we define the *order graph* $G_\pi$ of a permutation $\pi \in S_n$ and show that computing optimal solutions for instances is polynomially equivalent to computing largest *compatible* subsets of edges in this graph. Compatible sets of edges is a notion that generalizes the notion of non-crossing edges from [17, 18], reviewed in the previous section. In [3], sets of "red edges" corresponding to any block sorting sequence are defined, and some properties of these edge sets are described. The red edge sets turn out to be precisely the compatible edge sets as we define below. The crucial difference in the two approaches is that red edge sets are defined based on a given sorting sequence, and can consequently be shown to possess the property of compatibility. On the other hand, compatible edge sets are substructures within the order graph and yield block sorting sequences. That is, the property of compatibility is defined independent of any block sorting sequence, and we can construct a block sorting sequence whose red set is the given compatible set. While [3] only establishes that the size of the red edge set of any block sorting sequence is at least $n - 1 - \mathsf{bs}(\pi)$, we show that in fact optimal compatible sets are *exactly* of this size. The converse direction can be obtained from the approach of [3] with considerably more effort. Instead, we present a self-contained proof of both directions, more in line with the approach of [17, 18].

D 2. *The order graph of a permutation $\pi \in S_n$ is the directed graph $G_\pi = (V_\pi, E_\pi)$ where $V_\pi = [n]$ and $E_\pi = \{(u, v) \mid (u < v) \wedge (\pi_u^{-1} < \pi_v^{-1})\} = \{(\pi_i, \pi_j) \mid (i < j) \wedge (\pi_i < \pi_j)\}$.*

D 3. *(1) Two distinct edges $(u, v)$ and $(w, x)$ of $E_\pi$ are said to* interleave by value *if $u \leq w < v \leq x$. They are said to* interleave by position *if $\pi_u^{-1} \leq \pi_w^{-1} < \pi_v^{-1} \leq \pi_x^{-1}$. They are said to interleave if they interleave either by value or by position (or by both).*

*(2) An edge $(u, v)$ is said to* contain *an edge $(w, x)$ of $E_\pi$ if $u < w < x < v$ (contain by value) or $\pi_u^{-1} < \pi_w^{-1} < \pi_x^{-1} < \pi_v^{-1}$ (contain by position).*

*(3) For any $C \subseteq E_\pi$, the inclusion graph $G(C, \pi)$ corresponding to $C$ is the directed graph $(C, A)$ where $A = \{((u, v), (w, x)) \mid (u, v)$ contains $(w, x)\}$.*

*(4) A set $C \subseteq E_\pi$ is said to be compatible if no two edges of $C$ interleave and if $G(C, \pi)$ is acyclic.*

*(5) $c_\pi$ is the size of any compatible set of edges of maximum cardinality.*

Figure 1 shows examples of various types of interleaving edge pairs. Figure 2 shows the inclusion graph of an edge set; even though no pair of edges interleave the underlying edge set is not compatible since the inclusion graph has a cycle.

We note that if $C$ is a compatible edge set, then the subgraph $([n], C)$ is a collection of vertex-disjoint directed paths. Also, if $C$ is a compatible edge set and $C' \subseteq C$, then $C'$ is also compatible.

Note that the graph of Definition 1 for $\mathbb{S}_\pi$, let us call it $H_\pi$, is a subgraph of $G_\pi$: an edge $(u, v)$ of $G_\pi$ is included in $H_\pi$ if and only if the substring from $u$ to $v$ in $\pi$ is

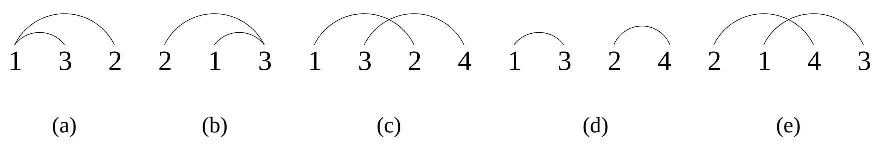

**Fig. 1**: Interleaving edge pairs

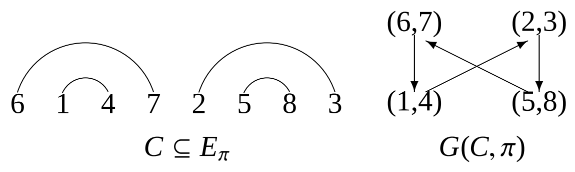

**Fig. 2**: An example inclusion graph

an increasing substring. Further, every non-crossing set in $H_\pi$ is also a compatible set in $G_\pi$, because interleaving by position in $H_\pi$ implies interleaving by value, and containment by position in $G_\pi$ implies containment by value in $H_\pi$. However, the converse is not true: a compatible set in $G_\pi$ is not necessarily a non-crossing set in $H_\pi$ for the simple reason that it may use an edge not present in $H_\pi$. Thus the notion of compatible sets properly generalizes that of non-crossing sets.

The main result of this section is an analogue of Lemma 1; we relate the size of the largest compatible set, $c_\pi$, to the block sorting distance. The result is established in Theorem 3. We need to establish several lemmas to prove this theorem.

We call $(u, v)$ a *unit edge* if $v - u = \pi_v^{-1} - \pi_u^{-1} = 1$. Thus unit edges are exactly those edges of the order graph $G_\pi$ which link the adjacent elements of a block. The following is easy to see.

L 2. *Every compatible set in $E_\pi$ of size $c_\pi$ contains all unit edges of $G_\pi$.*

D 4. *For a permutation $\pi$, let $C$ be any compatible subset of $E_\pi$. An edge $e$ is said to* touch *a block $\alpha$ if $e$ has exactly one endpoint in $\alpha$. The block $\alpha$ is said to be* free *with respect to $C$ if no edge of $C$ touches it; that is, every edge of $C$ has neither or both endpoints in $\alpha$.*

D 5. *A* (sub)block transposition *is the displacement of a (sub)block to an* arbitrary *location of the permutation. The block transposition distance* $\mathsf{s}(\pi)$ *of a permutation $\pi$ is the length of a shortest series of block transpositions that sorts it.*

Thus a block move is also a block transposition, and so $\mathsf{s}(\pi) \leq \mathsf{bs}(\pi)$.

L 3. *For a permutation $\pi$, let $C$ be a compatible subset of $E_\pi$ of size $c_\pi$, and let $\alpha$ be a block of $\pi$ that is free with respect to $C$. A transposition of $\alpha$ gives a permutation $\sigma$ with $c_\sigma \geq c_\pi$. If the transposition of $\alpha$ is indeed a block move, then $c_\sigma > c_\pi$.*

P . Since $\alpha$ is a free block, $C$ is a subset of $E_\sigma$ as well. Clearly, it has no edges interleaving by value. No edges interleave by new position either, since even in the new position no edge has just one endpoint in $\alpha$. And the graph $G(C, \sigma)$ is acyclic as well, since the edges other than those in $\alpha$ form no cycles, and the unit edges within $\alpha$ do not contain any other edge by value or by new position. So $C$ is a compatible subset of $E_\sigma$. Hence $c_\sigma \geq |C| = c_\pi$.

Further, if the transposition of $\alpha$ is a block move, then $G_\sigma$ has an extra unit edge created by this move, which is not even in $C$. So by Lemma 2, $|C| < c_\sigma$. Thus $c_\sigma > c_\pi$. □

L 4. *If $\sigma$ is obtained from $\pi$ via a block/subblock transposition, then $c_\sigma \geq c_\pi - 1$.*
*Furthermore, if $\sigma$ is obtained from $\pi$ via a block move, then $c_\sigma \geq c_\pi$.*

P . Let $C$ be a compatible set of edges in $G_\pi$ of size $c_\pi$. By Lemma 2, it contains all the unit edges of $G_\pi$. We now show that there is a compatible edge set $C'$ in $G_\sigma$ with $|C'| \geq |C| - 1$.

Let $\alpha$ be the subblock transpositioned in order to obtain $\sigma$ from $\pi$. Let $m$ be the number of edges of $C$ touching $\alpha$. Clearly, $m \in \{0, 1, 2\}$. If $m = 0$, then $\alpha$ is a free block with respect to $C$, and the result follows from Lemma 3. So now assume that $m > 0$.

**Case 1: m=1.** Let $(u, v)$ be the single edge of $C$ touching $\alpha$. (If $\alpha$ is a proper subblock, then this is in fact a unit edge.) As argued in the proof of Lemma 3, $C' = C \setminus \{(u, v)\}$ is a compatible edge set in $E_\pi$ and $E_\sigma$, showing that $c_\sigma \geq c_\pi - 1$.

Further, if the above transposition of $\alpha$ is a (sub)block move, then $G_\sigma$ has an extra unit edge created by this move, which is not in $C'$. So by Lemma 2, $|C'| < c_\sigma$ and hence $c_\sigma \geq c_\pi$.

**Case 2: m=2.** Let $\alpha$ be the (sub)block $v + 1, v + 2, \ldots, v + k$, with $\pi^{-1}_{v+j} = i + j$ for $j \in [k]$. Since $C$ contains all the unit edges of $G_\pi$, the two edges touching $\alpha$ must be of the form $(u, v + 1)$ and $(v + k, w)$. Let $C' = C \setminus \{(u, v + 1), (v + k, w)\}$, and $C'' = C' \cup \{(u, w)\}$. As argued above, $C'$ is a compatible edge set in $G_\pi$ and in $G_\sigma$. (In particular, $G(C', \sigma)$ is acyclic.) We now show that $C''$ is a compatible edge set in $G_\sigma$, proving that $c_\sigma \geq c_\pi - 1$. (In fact, $C''$ is even a compatible set in $G_\pi$ itself.) We only need to show that adding $(u, w)$ keeps this set compatible.

$C$ has a path $\rho = u, v+1, v+2, \ldots, v+k, w$ which is replaced in $C''$ by two paths: a single-edge path $\rho' = u, w$ and the path $\rho'' = v + 1, v + 2, \ldots, v + k$; all other edges of $C$ and $C''$ are the same. Since no other edge of $C$ can interleave with $\rho$ by value, it is clear that no other edge of $C''$ can interleave with paths $\rho'$ and $\rho''$ by value. Since the relative positions of all elements other than $v + 1, \ldots, v + k$ is unchanged, no edge of $C$ and $C''$ can interleave with $\rho'$ by position either. So $C''$ has no interleaving edges.

We must now show that $G(C'', \sigma)$ is acyclic. Assume it is not; then any cycle $\theta$ in it must pass through the vertex $(u, w)$, since we already know that $G(C', \sigma)$ is acyclic. Since the moved unit edges corresponding to $\alpha$ do not contain any edge by value or by position, none of the corresponding vertices can appear in $\theta$. Let $(x_1, y_1)$ be the predecessor and $(x_2, y_2)$ the successor of $(u, w)$ in $\theta$. Then $(x_1, y_1)$ must contain all edges of $\rho$ in $E_\pi$. And either $(u, v + 1)$ or $(v + k, w)$, both edges of $\rho$, must contain $(x_2, y_2)$ in $E_\pi$. The remaining edges of $\theta$ are present in $G(C, \pi)$ as well. Thus, corresponding to $\theta$, there is a cycle $\theta'$ in $G(C, \pi)$ as well, contradicting the compatibility of $C$.

Further, if the transposition of $\alpha$ is a (sub)block move, then $G_\sigma$ has an extra unit edge created by this move, which is not in $C''$. So by Lemma 2, $|C''| < c_\sigma$ and hence $c_\sigma \geq c_\pi$. □

L 5. *If $\pi$ is obtained from $\sigma$ by a block transposition, then $c_\pi \leq c_\sigma + 1$.*

P . Given $\sigma$, let $\pi$ be obtained through a transposition of some block $\alpha$ on $\sigma$. Then $\alpha$ is either a block or a subblock in $\pi$, and so, moving it back to its original place is a block or subblock transposition from $\pi$ yielding $\sigma$. Now apply Lemma 4. □

L 6. *Given $\pi \neq \mathtt{id}_n$ and a compatible set $C \subseteq E_\pi$ of size $c_\pi$, we can efficiently (in polynomial time) find a block move $\rho$ which, when applied to $\pi$, gives permutation $\sigma$ satisfying $c_\sigma = c_\pi + 1$.*

P . By Lemmas 3 and 5, it suffices to show that there exists a block $\alpha$ free with respect to $C$; finding one is easy. Consider the inclusion graph $G(C, \pi)$; it is acyclic by definition. All unit edges of $G_\pi$ have zero out-degree in $G(C, \pi)$. Let $C'$ be the set of non-unit edges of $C$; $C'$ is also a compatible set. If $C' = \emptyset$, then $C$ consists only of unit edges of $G_\pi$. Therefore, all blocks of $\pi$ are free with respect to $C$. Otherwise, consider the subgraph $G'$ of $G(C, \pi)$ induced by the vertex set $C'$. By heredity, $G'$ is also acyclic. Choose any vertex $(u, v) \in C'$ of zero out-degree in $G'$; let $i = \pi_u^{-1}$, $j = \pi_v^{-1}$. Let $A$ be the set $\{u + 1, \ldots, v - 1\} \cup \{\pi_{i+1}, \ldots, \pi_{j-1}\}$. Since $(u, v)$ is not a unit edge, $A$ is non-empty. Since $(u, v)$ is of out-degree zero in $G'$, it contains no non-unit edge of $C$. So the blocks containing any element of $A$ must all be free with respect to $C$. □

We now have all the ingredients needed to prove the main theorem of this section.

T 3. *For all $\pi \in S_n$, $n \geq 1$, $\mathsf{bs}(\pi) = n - 1 - c_\pi = \mathsf{s}(\pi)$.*

P . We will show that (i) $\mathsf{s}(\pi) \geq n - 1 - c_\pi$, and (ii) $\mathsf{bs}(\pi) \leq n - 1 - c_\pi$. Since $\mathsf{s}(\pi) \leq \mathsf{bs}(\pi)$, the result follows.

To prove (i), let $\rho_1, \rho_2, \ldots, \rho_k$ be a series of block transpositions sorting $\pi$. Let $\pi^0 = \pi$, and $\pi^i$ be the permutation obtained by applying block transposition $\rho_i$ to $\pi^{i-1}$. Let $c_i$ denote $c_{\pi^i}$. By Lemma 5, $c_i \leq c_{i-1} + 1$. Since $\pi^k = \mathtt{id}_n$, and since $c_{\mathtt{id}_n} = n - 1$, we have $n - 1 = c_k \leq c_{k-1} + 1 \leq \ldots \leq c_1 + (k - 1) \leq c_0 + k = c_\pi + k$,

hence $k \geq n - 1 - c_\pi$. This holds for any block transposition sorting sequence; in particular, it holds for an optimal sequence with $k = \mathsf{s}(\pi)$. Hence $\mathsf{s}(\pi) \geq n - 1 - c_\pi$.
To prove (ii), repeatedly apply Lemma 6 and note that $c_{\mathtt{id}_n} = n - 1$. □

For completeness, we briefly discuss how to make Theorem 3 constructive. Given a block sorting sequence of length $\mathsf{bs}(\pi)$, a compatible edge set of size $n-1-\mathsf{bs}(\pi)$ can be extracted by following the presentation of [3] (Lemma 4); the red edges corresponding to the sequence are the desired compatible set. Given a compatible edge set of size $c$, we present in Figure 3 an algorithm that sorts $\pi$ in $n - 1 - c$ steps.

```
Block-Sort($\pi, C$)
Input: A permutation $\pi$ over $n$ elements, and a compatible set $C$ in the order graph
of $\pi$.

While $\pi \neq \mathtt{id}_n$ Do
      Find a block $\alpha$ in $\pi$ free with respect to $C$.
      (The proof of Lemma 6 shows that such a block always exists.
        One can be found by direct inspection of each block with respect to $C$.)
      Let $u, v$ be the first and last elements of this block.
      If $v < n$ Then
          Move $\alpha$ to its successor, to get the permutation $\pi'$.
          If $C$ has an edge $(l, v + 1)$ for some $l$
          Then $C \longleftarrow C \cup \{(l, u)\} \setminus \{(l, v + 1)\}$
          EndIf
          $C' = C \cup \{(v, v + 1)\}$.
      Else
          Move $\alpha$ to its predecessor, to get the permutation $\pi'$.
          $C' = C \cup \{(u - 1, u)\}$.
      EndIf
      $\pi \longleftarrow \pi', C \longleftarrow C'$.
EndWhile
```

**Fig. 3**: The algorithm for

Given a permutation $\pi$, replace each block $x$ by any one representative element to obtain a sequence $S$. Now, replacing each $a \in S$ by its rank in $S$, we obtain a permutation on $k$ elements, where $k$ is the number of blocks in $\pi$. This permutation is called the kernel of $\pi$, denoted $\mathtt{ker}(\pi)$. Since blocks are indivisible in block moves, it follows that $\mathsf{bs}(\pi) = \mathsf{bs}(\mathtt{ker}(\pi))$. However, blocks are not indivisible in block transpositions, since a block may be picked up and placed elsewhere in the middle of another existing block. Intuitively, nothing would be gained by such a move; formally, this follows from Theorem 3.

C 1. $\mathsf{s}(\pi) = \mathsf{s}(\mathtt{ker}(\pi))$.

(*Note:* Corollary 1 is proved in [18] independent of $c_\pi$ using completely different techniques.)

If $\pi \in S_n$ and $\pi \neq \mathtt{rev}_n = n\ n-1 \ldots 2\ 1$, then clearly $c_\pi \geq 1$, and hence $\mathsf{bs}(\pi) \leq n-2$. For $\pi = \mathtt{rev}_n$, $c_\pi = 0$, and so $\mathsf{bs}(\mathtt{rev}_n) = n-1$. Thus we have:

C 2. $D(n) = \max\{\mathsf{bs}(\pi) | \pi \in S_n\} = n-1$.

## 4. and are equivalent

We first obtain an alternative formulation of and then show that this alternative formulation is equivalent to . It then follows that sorting a permutation via block moves is as hard (or as easy) as sorting its inverse.

We say that an increasing substring $a_i \ldots a_j$ of a string $S = a_1\, a_2 \ldots a_m$ of distinct elements from $[n]$ is *tight* if there is no $i \leq k \leq j-1$ such that $a_k < a_l < a_{k+1}$ for some $a_l$ in $S$. (In other words, if each $a_l$ in $S$ is replaced by its rank $r(a_l)$ in $S$, then the substring $r(a_i) \ldots r(a_j)$ is a subblock in the resulting string. In [4], the term *relative block* is used to denote a maximal tight increasing substring.) Let $\mathsf{isr}(S)$ denote the length of a shortest sequence of tight increasing substring removals on $S$ that leaves behind an increasing subsequence of $S$.

For example, consider 3 5 1 4 2. Neither of the increasing substrings $\langle 3, 5\rangle$ and $\langle 1, 4\rangle$ is tight, and removing any one element does not make the sequence increasing, so $\mathsf{isr}(3\ 5\ 1\ 4\ 2) > 1$. But after deleting 4, the sequence $\langle 3, 5\rangle$ is tight, and deleting it leaves behind an increasing sequence 1 2. So $\mathsf{isr}(3\ 5\ 1\ 4\ 2) = 2$.

The following is easy to see; see also [3] and Theorem 2.1 of [4].

L 7. *For any permutation* $\pi$, $\mathsf{isr}(\pi) = \mathsf{bs}(\pi)$.

Recall that $\mathsf{csr}(\pi, \phi)$ denotes the length of a shortest sequence of common substring removals that makes the pair identical. Given any $\pi, \phi \in S_n$, for any permutation $\psi \in S_n$, the pair $(\psi^{-1}\pi, \psi^{-1}\phi)$ is just a consistent relabeling for the pair $(\pi, \phi)$. Thus $\mathsf{csr}(\pi, \phi) = \mathsf{csr}(\psi^{-1}\pi, \psi^{-1}\phi)$. In particular, $\mathsf{csr}(\pi, \phi) = \mathsf{csr}(\pi^{-1}\phi, \mathtt{id}_n) = \mathsf{csr}(\phi^{-1}\pi, \mathtt{id}_n)$. But if we consider $\mathsf{csr}(\sigma, \mathtt{id}_n)$, the common substrings removed are always tight increasing substrings. Thus $\mathsf{csr}(\sigma, \mathtt{id}_n) \geq \mathsf{isr}(\sigma)$. In fact, we have equality $\mathsf{csr}(\sigma, \mathtt{id}_n) = \mathsf{isr}(\sigma)$ because the tight increasing substrings removed from $\sigma$ are necessarily substrings of $\mathtt{id}_n$. These remarks, along with Lemma 7, establish the following theorem.

T 4. *The common substring removals problem and are computationally equivalent.*
*For* $\pi, \phi \in S_n$, $\mathsf{csr}(\pi, \phi) = \mathsf{bs}(\pi^{-1}\phi) = \mathsf{bs}(\phi^{-1}\pi)$. *For* $\sigma \in S_n$, $\mathsf{bs}(\sigma) = \mathsf{csr}(\sigma, \mathtt{id}_n)$.

C 3. *For any* $\pi \in S_n$, $\mathsf{bs}(\pi) = \mathsf{bs}(\pi^{-1})$.

(*Note:* Corollary 3 is independently established in [3].)

## 5. Some optimal block moves

An important implication of Lemma 4 and Theorem 3 is that a block move never increases the block sorting distance of a permutation. However, many block moves do not reduce it either. The computational difficulty in block-sorting optimally lies precisely here: how does one identify, amongst all possible block moves, the move(s) that actually reduces the block sorting distance? In this section, we identify some block moves which provably reduce this distance. Thus any heuristic for block sorting can safely make such moves, if they are possible.

### *5.1 2-moves and 3-moves are provably optimal*

A block move reduces the number of blocks by at least 1 and at most 3. Intuitively, one wants to reduce the number of blocks as fast as possible (since $\mathtt{id}_n$ has a single block). Thus it is natural to prefer a block move which reduces the number of blocks by 3; such a move ought to be optimal. However, formally proving this is not easy. We need to argue that local "trade-offs" do not arise: if a reduction of 3 can preclude three subsequent reductions by 2, then the reduction sequence 2,2,2,1 would be better than 3,1,1,1. We show here that indeed such situations do not arise. Any block move that reduces the number of blocks by 3, or even by 2, necessarily reduces the block sorting distance.

We first consider the situation when moving a block $\alpha$ to its predecessor also joins it with its successor.

L 8. *If $\pi \in S_n$ has $\pi_i = u - 1$ and $\pi_{i+1} = v$ for some $i$ and some $u < v$, and if the elements $u, u+1, u+2, \ldots, v-1$ form a block $\alpha$ of $\pi$, then the permutation $\sigma$ obtained from $\pi$ by moving $\alpha$ to between $u$ and $v$ has $\mathsf{bs}(\sigma) < \mathsf{bs}(\pi)$.*

P . From Theorem 3 and Corollary 1, $\mathsf{bs}(\pi) = \mathsf{bs}(\mathtt{ker}(\pi))$. In $\mathtt{ker}(\pi)$, $\alpha$ is a single element. So it suffices to prove the lemma when $|\alpha| = 1$, i.e. $v = u + 1$ and $\alpha = u$.

By Lemma 3 and Theorem 3, it suffices to prove that there is some compatible edge set of size $c_\pi$ with respect to which the block containing $u$ is free.

Let $C$ be any compatible subset of $E_\pi$ of size $c_\pi$. If $u$ is free with respect to $C$ there is nothing to prove. Otherwise, without loss of generality, assume that $k = \pi_u^{-1} > i + 1$; the situation where $k < i$ is symmetric. There are three cases.
**Case 1:** There is a single edge in $C$ touching $u$, and it is of the form $(u, w)$. Then $C$ has no edge of the form $(x, u+1)$, since such an edge interleaves by value with $(u, w)$. If $C$ does not have any edge $(u-1, x)$, then let $C' = C \setminus \{(u, w)\} \cup \{(u-1, u+1)\}$. (see Figure 4(a)). Otherwise let $(u-1, x)$ be an edge in $C$ and let $y$ be the rightmost endpoint of the path in $C$ containing $u+1$. (Figure 4(b),(c)). Let $C' = C \setminus \{(u, w), (u-1, x)\} \cup \{(u-1, u+1), (y, x)\}$. It can be seen that $C'$ is a compatible subset of $E_\pi$ of size $c_\pi$. And $u$ is free with respect to $C'$.
**Case 2:** There is a single edge in $C$ touching $u$, and it is of the form $(w, u)$.

If $w = u - 1$, then $C' = C \setminus \{(u-1, u)\} \cup \{(u-1, u+1)\}$ is a compatible subset of $E_\pi$ of size $c_\pi$, in which $u$ is free.

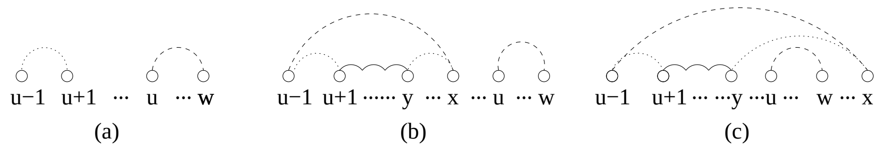
u−1 u+1 ··· u ··· w
(a)
u−1 u+1 ······ y ··· x ··· u ··· w
(b)
u−1 u+1 ··· ···y ···u ··· w ··· x
(c)

**Fig. 4**: Case 1 of Lemma 8.

If $w \neq u - 1$, then $(u - 1, y)$ cannot be in $C$ for any $y$. If $C$ has no edge with right endpoint $u+1$, then the set $C' = C \setminus \{(w, u)\} \cup \{(u-1, u+1)\}$ is a compatible subset of $E_\pi$ of size $c_\pi$, in which $u$ is free.

Otherwise, let $(x, u + 1)$ be an edge in $C$. To avoid interleaving by value with $(w, u)$, we have $x < w$. So $(x, u + 1)$ contains $(w, u)$ by value. If $\pi_w^{-1} < \pi_x^{-1}$, then $(w, u)$ contains $(x, u + 1)$ by position, creating a cycle in $G(C, \pi)$.; see Figure 5(a). So $\pi_w^{-1} > \pi_x^{-1}$. But if $\pi_w^{-1} < i$, then $(x, u+1)$ and $(w, u)$ would interleave by position. So the relative ordering of these elements must be $x, u-1, u+1, w, u$. (Figure 5(b)). Now the set $C' = C \setminus \{(w, u), (x, u + 1)\} \cup \{(x, u - 1), (u - 1, u + 1)\}$ is a compatible subset of $E_\pi$ of size $c_\pi$, and $u$ is free with respect to $C'$.

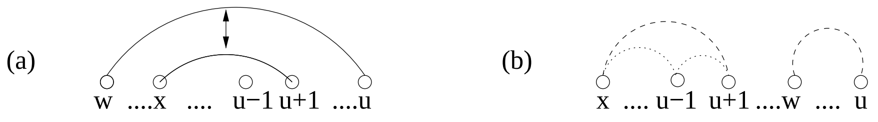

**Fig. 5**: Case 2 of Lemma 8.

**Case 3:** There are two edges touching $u$, $(w, u)$ and $(u, x)$. There can be no edge into $u+1$ in $C$ since any such edge will interleave with $(u, x)$ by value. There can be no edge out of $u - 1$ in $C$ since any such edge will interleave with $(w, u)$ by value, unless $w = u-1$. If $w = u-1$, set $C' = C \setminus \{(u-1, u), (u, x)\} \cup \{(u-1, u+1), (y, x)\}$ where $y$ is the rightmost endpoint of the path in $C$ containing $u + 1$. Otherwise set $C' = C \setminus \{(w, u), (u, x)\} \cup \{(u - 1, u + 1), (w, x)\}$. Either way, $C'$ is a compatible subset of $E_\pi$ of size $c_\pi$, and $u$ is free with respect to $C'$. □

We now consider the situation when moving a block (to either its predecessor or its successor) results in the blocks on either side of it joining up.

L 9. *If a permutation $\pi \in S_n$ has $\pi_{i-1} = u-1$ and $\pi_j = u$ for some $i, j \geq i+1, u$, and if the elements $\pi_i, \pi_{i+1}, \pi_{i+2}, \ldots, \pi_{j-1}$ form a block $\alpha$ of $\pi$, then the permutation $\sigma$ obtained from $\pi$ by a block move of $\alpha$ has* $\mathsf{bs}(\sigma) < \mathsf{bs}(\pi)$.

P . As in Lemma 8, without loss of generality, assume that $|\alpha| = 1$, and hence $j = i + 1$.

Let $\pi$ have $\pi_{i-1} = u - 1$, $\pi_{i+1} = u$, and $\pi_i = v$. By Lemma 3 and Theorem 3, it suffices to prove that there is some compatible edge set of size $c_\pi$ with respect to which the block containing $v$ is free.

Let $C$ be a compatible edge set of $G_\pi$ of size $c_\pi$. If $v$ is free with respect to $C$, there is nothing to prove.

If $C$ has a single edge $e$ touching $v$, then $(u-1, u)$ cannot be in $C$ since it interleaves with $e$ by position. If $C$ has no edge (other than possibly $e$ itself) into $u$ or out of $u-1$ (it cannot have both anyway), then let $C'$ be the set $C \setminus \{e\} \cup \{(u-1, u)\}$. Otherwise, suppose $C$ contains $(y, u)$. Then $e$ must be of the form $(w, v)$. Now set $C' = C \setminus \{(w, v), (y, u)\} \cup \{(u-1, u), (y, u-1)\}$. On the other hand, if $C$ contains $(u-1, z)$, then $e$ must be of the form $(v, x)$, and we set $C'$ to be $C \setminus \{(v, x), (u-1, z)\} \cup \{(u-1, u), (u, z)\}$. In all these cases, $C'$ is a compatible set of $G_\pi$ of size $c_\pi$, and $v$ is free with respect to it.

If $C$ has two edges $(w, v), (v, x)$ touching $v$, then we do the following. Remove $(w, v), (v, x)$ from $C$ and add $(u-1, u)$. Further, if $w = u-1$, add $(u, x)$, if $x = u$, add $(w, u-1)$, otherwise, add $(w, x)$, and let the resulting set be $C'$. Then it can be seen that $C'$ is a compatible set of $G_\pi$ of size $c_\pi$, and $v$ is free with respect to it. □

With the results we have seen, we can formalize the intuition described at the beginning of this subsection. Define an $x$-move to be a block move that results in a permutation with $x$ fewer blocks. Since a 2-move satisfies the assumption of either Lemma 8 or Lemma 9, and since a 3-move satisfies the assumptions of both, we now have the main result of this section:

T 5. *If a 2-move or a 3-move applied to $\pi$ gives $\sigma$, then* $\mathsf{bs}(\sigma) < \mathsf{bs}(\pi)$.

## *5.2 Certain 1-moves are provably optimal*

In this subsection we show that if a block is preceded by its successor block, then joining it with its successor provably reduces the block sorting distance. Note that such a move may not even be a 2-move; thus this result includes cases not covered in the previous subsection. To prove this, we establish an interesting property, which we call *additivity*, of the function $\mathsf{bs}(\pi)$, and then we use Theorem 3.

T 6. *If a permutation $\pi$ has adjacent blocks $\alpha, \beta$, with $\beta\alpha$ being a substring of* $\mathtt{id}_n$*, and if $\phi$ is the permutation obtained from $\pi$ by exchanging $\alpha$ and $\beta$ (which is a block move of $\alpha$ or $\beta$), then* $\mathsf{bs}(\phi) < \mathsf{bs}(\pi)$.

P . Let the substring $\beta\alpha$ be $u+1\ u+2\ \ldots\ u+t$ for some $u, t$, and let it occur in positions $i+1,\ i+2,\ \ldots,\ i+t$ of $\phi$ for some $i$. If $\pi_i = u$ or if $\pi_{i+t+1} = u+t+1$, then exchanging $\alpha$ and $\beta$ in $\pi$ is a 2-move or a 3-move, and hence, by Theorem 5, reduces the block move distance. So now let us assume that $\pi_i \neq u$ and $\pi_{i+t+1} \neq u+t+1$. Clearly, one way to sort $\pi$ is to exchange $\alpha$ and $\beta$ and then sort $\phi$; thus $\mathsf{bs}(\pi) \leq 1 + \mathsf{bs}(\phi)$. We show that this is in fact an equality, by defining a notion of *complete* substrings and establishing a more general result concerning them. The proof follows from the statement of Lemma 10 below and the fact that $\alpha\beta$ forms a maximal complete substring in $\pi$. □

D 6. *A substring $\pi_{i+1} \ldots \pi_{i+t}$ is* complete *if $\{\pi_{i+1}, \ldots, \pi_{i+t}\} = \{u+1, u+2, \ldots, u+t\}$ for some $u$. That is, the $t$ consecutive positions $i+1, i+2, \ldots, i+t$*

*hold t consecutive integers, though not necessarily in sorted order. If, furthermore,* $\pi_i \neq u$ *and* $\pi_{i+t+1} \neq u+t+1$*, then the substring is said to be* maximal complete.

*The block move distance for the complete substring* $\pi_{i+1} \dots \pi_{i+t}$ *is defined to be the minimum number of block moves needed to sort it i.e. to transform it into* $u + 1 \dots u + t$*.*

L 10. *If* $\pi$ *contains a maximal complete substring* $\alpha$ *with* $\mathsf{bs}(()\alpha) = q$*, then* $\mathsf{bs}(\pi) = q + \mathsf{bs}(\sigma)$*, where* $\sigma$ *is the permutation obtained from* $\pi$ *by replacing* $\alpha$*, in place, with its sorted version.*

P . Since $\alpha$ is maximal complete, it is clear that $\mathsf{bs}(\pi) \leq q + \mathsf{bs}(\sigma)$. So we only need to show that $\mathsf{bs}(\pi) \geq q + \mathsf{bs}(\sigma)$. By Theorem 3, it suffices to show that $c_\sigma \geq q + c_\pi$.

Let $\alpha = \pi_{i+1} \dots \pi_{i+t}$ where $\{\pi_{i+1}, \dots, \pi_{i+t}\} = [u+1,\ u+t]$ for some $u$. Let $C$ be any compatible edge set of $G_\pi$ of size $c_\pi$. Partition $C$ into four subsets as follows:

| | | | |
|---|---|---|---|
| $C_1$ | $=$ | $\{(v,w) \mid v,w \in [u+1,\ u+t]\}$ | both endpoints in $[u+1,\ u+t]$ |
| $C_2$ | $=$ | $\{(v,w) \mid \pi_v^{-1} \leq i, w \in [u+1,\ u+t]\}$ | right endpoint in $[u+1,\ u+t]$ |
| $C_3$ | $=$ | $\{(v,w) \mid v \in [u+1,\ u+t], \pi_w^{-1} > i+t\}$ | left endpoint in $[u+1,\ u+t]$ |
| $C_4$ | $=$ | $C \setminus (C_1 \cup C_2 \cup C_3)$ | no endpoint in $[u+1,\ u+t]$ |

It follows from Theorem 3 that $|C_1| \leq c_\alpha = t - 1 - q$; i.e. $t - 1 - |C_1| \geq q$. If $|C_2| \leq 1$ and $|C_3| \leq 1$, we construct a compatible set in $E_\sigma$ of the required size as follows: Let $C_5$ be the set of $t-1$ unit edges obtained after sorting $\alpha$; $C_5 = \{(u+i, u+i+1) \mid i \in [t-1]\}$. Define $C' = C_4 \cup C_5$. Then $C'$ is a subset of $E_\sigma$, and it is easy to see that $C'$ is compatible. If $C_2 \cup C_3 = \emptyset$, then $|C_1| + |C_4| = c_\pi$, so $C'$ is the desired set. Otherwise, $C'' = C' \cup \{(v, u+1) \mid (v, u+i) \in C_2$ for some $i \in [t]\} \cup \{(u+t, w) \mid (u+j, w) \in C_3$ for some $j \in [t]\}$ is the desired set; compatibility and size bounds are easily seen to hold.

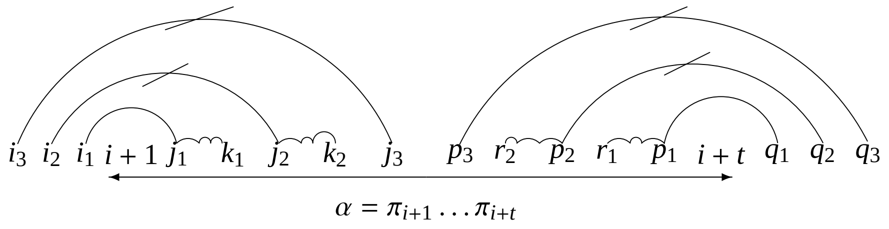

**Fig. 6**: A compatible set of $G_\pi$ with $\alpha$ complete: a=3; b=3.

If either $|C_2|$ or $|C_3|$ exceeds 1, we show how to construct another compatible subset of $E_\pi$ of size $c_\pi$, in which the corresponding subsets $|C_2|$ and $|C_3|$ are both of size at most 1. Then we can use the above argument.

Assume that $|C_2| = a \geq 2$. Let the edges of $C_2$ be $(\pi_{i_1}, \pi_{j_1}), (\pi_{i_2}, \pi_{j_2}), \dots, (\pi_{i_a}, \pi_{j_a})$, where $j_1 < j_2 < \dots < j_a$. By compatibility considerations, we have $i_a < \dots < i_2 < i_1 < j_1 < j_2 < \dots < j_a$. Let the unique path in $C$ containing $\pi_{j_h}$ end at the right at $\pi_{k_h}$. From compatibility considerations and the completeness of $\alpha$, it can be argued that $(\pi_{k_h}, \pi_{j_{h+1}}) \in E_\pi$. We define $C'_2 = \{(\pi_{i_1}, \pi_{j_1})\} \cup \{(\pi_{k_h}, \pi_{j_{h+1}}) \mid h \in [a-1]\}$.

If $|C_3| = b \geq 2$, then a set $C'_3$ of the same size is constructed in a symmetric way. Let the edges of $C_3$ be $(\pi_{p_1}, \pi_{q_1}), (\pi_{p_2}, \pi_{q_2)}), \ldots, (\pi_{p_r}, \pi_{p_r})$, where $p_b < \ldots < p_2 < p_1$. By compatibility considerations, we have $p_b < \ldots < p_2 < p_1 < q_1 < q_2 < \ldots < q_b$. Let the unique path in $C$ containing $\pi_{p_h}$ end at the left at $\pi_{r_h}$. Again, we can argue that the edges $(\pi_{p_{h+1}}, \pi_{r_h})$ are indeed present in $E_\pi$, and we define $C'_3 = \{(\pi_{p_1}, \pi_{q_1})\} \cup \{(\pi_{p_{h+1}}, \pi_{r_h}) \mid h \in [b-1]\}$.

If both $a, b \geq 2$, then it must be that $j_a < p_b$, since otherwise there will be edges in $C$ interleaving by position.

Clearly, $C'_2 \cup C'_3$ is of the same size as $C_2 \cup C_3$, and is disjoint from $C_1 \cup C_4$. We now show that $C' = C_1 \cup C'_2 \cup C'_3 \cup C_4$ is also a compatible subset of $E_\pi$.

It is easy to see that none of the edges of $C'_2 \cup C'_3$ interleave by position. Neither do edges of $C_1 \cup C_4$. And by our choice of $C'$, an edge of $C'_2 \cup C'_3$ will not interleave by position with an edge of $C_1 \cup C_4$.

If edges of $C'$ interleave by value, then the corresponding edges of $C$ can similarly be shown to interleave by value, since $\alpha$ is a complete substring.

It can be argued that if the inclusion graph $G(C', \pi)$ has a cycle, then so does $G(C, \pi)$, contradicting compatibility of $C$. $\square$

**Note:** The above result can be shown to hold even if $\alpha$ is complete but not maximal; however, the proved form is enough for our purpose.

## 6. Heuristics for the problem

We revisit the 2-approximation algorithm of [18] and consider some heuristics inspired by it, especially in conjunction with the characterization of optimal solutions of in terms of compatible edge sets.

### *6.1 Minor refinements of the heuristic*

We begin by looking at the examples presented for establishing the tightness of the performance ratio in [18].

E 1. (a) For the permutation $\pi$ on $2n$ elements where $\pi_{2k-1} = k$ and $\pi_{2k} = n + k$, we have $\mathsf{bs}(\pi) = n - 1$ and $\mathsf{bm}(\mathbb{S}_\pi) = 2n - 2$.
(e.g. 1 5 2 6 3 7 4 8.)

(b) For $\phi$ on $[2n]$ such that $\phi_i = 2i - 1$ for $i \leq n$ and $\phi_i = 2i - n$ for $i > n$, $\mathsf{bs}(\phi) = n - 1$ while $\mathsf{bm}(\mathbb{S}_\phi) = n$.
(e.g. 1 3 5 7 2 4 6 8.)

(c) For $\texttt{rev}_n$, $\mathsf{bs}(\texttt{rev}_n) = n - 1 = \mathsf{bm}(\mathbb{S}_{\texttt{rev}_n})$.

The permutation $\pi$ in Example 1(a) shows why the approximation is bad; the approximate solution is twice as long as the optimal. Thus if we stop at reporting $n - 1 - \mathsf{bm}(\mathbb{S}_\pi)$, we cannot do better than a factor of 2. However, we could perhaps do better by actually constructing the solution of , mimicking it on $\pi$ to get a block sorting sequence, and reporting the length of this sequence. This length could be shorter than $\mathsf{bm}(\mathbb{S}_\pi)$ because in the mimicking, some merge moves become redundant. (For instance, consider the optimal solution to $\mathbb{S}_\pi$ where we first move $n+1, n+2, \ldots, 2n-1$ to successors sequentially, then we move $1, 2, \ldots, n-1$ to successors sequentially. Mimicking this with block moves on $\pi$, we find that the last $n-1$ moves are redundant; though the sequences are not merged, the permutation is sorted after the first $n-1$ moves itself.) This suggests the first heuristic.

Heuristic-1: Compute an optimal solution to $\mathbb{S}_\pi$. Convert this to a block sorting sequence on $\pi$ and remove redundant moves. Report this sequence.

Another noteworthy point about $\pi$ is that it can be sorted using the provably-optimal moves of Section 5, which are easy to detect. So using the solution without even trying to detect provably optimal moves is a mistake. This suggests the next heuristic.

Heuristic-2: While there are provably-optimal moves (from Theorems 5, 6), perform these moves. Then apply Heuristic-1.

Further, note that the permutation $\phi$ in Example 1(b) is exactly $\pi^{-1}$. And for $\phi$, the approximation via itself works very well. Since $\mathsf{bs}(\pi) = \mathsf{bs}(\pi^{-1})$ (Corollary 3), this suggests the third heuristic.

Heuristic-3: Apply Heuristic-2 on $\pi$ and $\pi^{-1}$. Report the minimum.

The question is, do any of these heuristics have a performance ratio better than 2? Unfortunately, the answer is No. We find an example that nullifies all of the above expectations.

E 2. Consider the sequence of the first $2n+1$ odd numbers $1, 3, \ldots, 4n+1$. Insert the even numbers $2n+2, 2n+4, \ldots, 4n, 2, 4, \ldots 2n$, in this order between 1 and 3, between 3 and 5, and so on, and between $4n-1$ and $4n+1$. Let the resulting permutation be denoted $\sigma$.
(e.g. for $n = 2$, we get 1 6 3 8 5 2 7 4 9. For $n = 4$, see Figure 7.)
Then $\sigma^{-1} = \sigma, \mathsf{bs}(\sigma) = 2n$, and $\mathsf{bm}(\mathbb{S}_\sigma) = 4n-1$.

An instance of the type of Example 2, with a largest compatible set, is depicted in Figure 7.

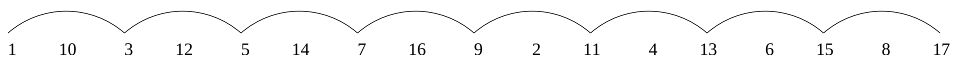

**Fig. 7**: An alternative tight example with a largest compatible set

In the permutations of Example 2, going over to the inverse permutation does not help. There are no provably optimal moves to begin with. And there are optimal merging sequences on $\mathbb{S}_\sigma$ which yield no redundant moves. (This is not to say that no optimal merging sequence has redundant moves. But we only know how to come up with some merging solution. We do not know how to explicitly search for one with many redundant moves.) Thus, Heuristic-3 on this example has an approximation factor of 2.

Another, slightly weaker, scenario worth considering is whether largest noncrossing sets $C$ and $C'$ computed for the input permutation and its inverse together contain a large enough compatible set to ensure a better than factor 2 approximation ratio for . Searching for such a compatible set in $C \cup C'$ could be easier, since there are fewer edges to begin with. Of course, Example 2 kills this hope as well, since $\sigma = \sigma^{-1}$.

## *6.2 A New Heuristic based on*

The last heuristic considered in the previous subsection uses twice, possibly after some preprocessing to remove provably optimal moves. However, once a solution is found, it is used essentially as is, with some minor preprocessing to remove redundant moves. In this section, we introduce a heuristic that uses repeatedly in a non-trivial way.

The exorbitant cost for block merging compared to block sorting in Example 2 is due to the fact that the permutation is broken up into many substrings in an inconvenient way. It appears advisable that we keep a given permutation $\pi$ as such and somehow detect a preferred block move. This gives rise to the following heuristic, defined formally in Figure 8.

Heuristic-4: Let the application of a block move $\rho$ to $\pi$ result in a permutation denoted $\pi.\rho$. Repeatedly choose the block move $\rho$ that maximizes the difference $\mathsf{bm}(\mathbb{S}_{\pi}) - \mathsf{bm}(\mathbb{S}_{\pi.\rho})$.

```
(1) Set $\pi^0 = \pi$.
(2) For $j = 1$ to $\texttt{\#block}(\pi^i)$, move the $j$th block of $\pi^i$ to its predecessor to obtain
    $\pi^{i,j}$ and compute $\mathsf{bm}(\mathbb{S}_{\pi^{i,j}})$.
(3) Set $\pi^{i+1} = \pi^{i,j}$ if $\mathsf{bm}(\pi^{i,j}) = \min_j\{\mathsf{bm}(\mathbb{S}_{\pi^{i,j}})\}$. (Break ties by choosing the
    largest $j$.)
(4) Repeat steps 2 and 3 until $\pi^{i+1} = \texttt{id}_n$.
```

**Fig. 8**: A heuristic for based on

(Note that we don't gain anything by letting the heuristic check moves to successor as well, because the two permutations resulting from the move of any block to its predecessor and to its successor will both have the same kernel.)

It is clear that we can always find a block move $\rho$ on $\pi$ for which $\mathsf{bm}(\mathbb{S}_{\pi}) - \mathsf{bm}(\mathbb{S}_{\pi.\rho}) \geq 1$. Thus the above heuristic takes no more than $\mathsf{bm}(\mathbb{S}_{\pi})$ steps to sort, and so it follows from [18] that the heuristic is obviously no worse than a factor 2 algorithm for . The question is, is its approximation factor better than 2?

The heuristic behaves well on many test cases. In particular, it sorts all examples presented so far in this paper optimally. In fact, finding permutations for which the heuristic needs more than $\mathsf{bs}(\pi)$ moves is itself non-trivial. Despite such favorable evidence, we show that this heuristic is no better than a 2-approximation algorithm. But the permutations we construct to establish this fact differ from the tight examples presented so far in that the worst case is attained only for asymptotically large permutations. Since the examples we construct appear to be typical of adversaries, it may be hoped that the heuristic will be of some practical value nonetheless.

## *6.3 Tight Examples for the Heuristic*

We show that the worst-case performance ratio of Heuristic-4 (the heuristic in Figure 8) is 2. The tight examples we provide for this purpose have long sequences in the associated merging instances, in contrast to the tight examples we have encountered so far. As mentioned above, the only way to get an approximation factor better than 2 is to often find block moves where $\mathsf{bm}(\mathbb{S}_\pi) - \mathsf{bm}(\mathbb{S}_{\pi.\rho})$ strictly exceeds 1. But note that this difference can be greater than 1 only if (and even then not always) the block moved by $\rho$ is at the extreme end of some maximal increasing substring of $\pi$. We use this intuition in constructing our examples.

We begin with a description of the permutations we are going to construct. The lengths of the permutations will be $n = (2p + 2)q = 2pq + 2q$, $p, q \geq 2$. (The associated instance will consist of $q$ increasing sequences of length $p$ and $q$ increasing sequences of length $p+2$.) The construction is formally defined below. However, the reader may find the equivalent description of Figure 9 more insightful, as it highlights how pairs of elements are placed so as to make block merging difficult. Figures 10 and 11 illustrate the construction for the case of $p = q = 3$.

D 7. *For any $p, q \geq 2$, let $n = (2p + 2)q = 2pq + 2q$. Define the following sequences for $1 \leq i, j \leq q$:*

*(1)* $w_i = i, q + i, 2q + i, \ldots, (p - 1)q + i.$

*(2)* $x_j = pq + j, pq + q + j, pq + 2q + j, \ldots, pq + (p - 1)q + j.$

*(3)* $z_i = 2pq + i, 2pq + q + i.$

*Then the permutation $\pi(p, q)$ is defined by the sequence*
$w_1\ z_1\ x_q\ \ w_2\ z_2\ x_{q-1}\ \ldots\ w_q\ z_q\ x_1.$

Consider $a(\pi(p, q))$, the number of block moves needed by Heuristic-4 on $\pi(p, q)$. In the next three lemmas, we show that $\mathsf{bm}(\mathbb{S}_{\pi(p,q)})$ is roughly $2pq$, that $a(\pi(p, q)) = \mathsf{bm}(\mathbb{S}_{\pi(p,q)})$, and that $\mathsf{bs}(\pi(p, q))$ is roughly $pq$. Together, these results give Theorem 7, namely, that the approximation factor of Heuristic-4 is no better than 2.

L 11. $\mathsf{bm}(\mathbb{S}_{\pi(p,q)}) = 2(pq - p + q) - 1.$

P . Let $\pi = \pi(p, q)$, with the corresponding instance $\mathbb{S}_\pi$. From Lemma 1, it suffices to show that $c(\mathbb{S}) = 2p$. We decompose $H_\pi$ into two graphs: $A$ is the induced subgraph on nodes labeled from the set $\{1, 2, \ldots, pq\} \cup \{2pq + 1, \ldots, 2pq + 2q\}$, and $B$ is the induced subgraph on the rest. ($A$ is the restriction

(1) Use the elements $1, 2, 3, \ldots, 2pq - 1, 2pq$ to form $pq$ nested intervals $(1, 2pq), (2, 2pq - 1), (3, 2pq - 2), \ldots, (pq - 1, pq + 2), (pq, pq + 1)$.

(2) Partition these intervals into $q$ groups of $p$ intervals each so that the $i$th group consists of every $j$th interval, $j \pmod{q} = i$.

(3) Nest the intervals in each of the $q$ groups in the natural way to form $q$ disjoint subsequences of $1, 2, 3, \ldots, 2pq - 1, 2pq$ of length $2p$ each.

(4) Insert the pair of elements $2pq + i, 2pq + q + i$, $1 \le i \le q$, after $p$ elements in the sequence corresponding to the $i$th group of intervals.

(5) Concatenate the above sequences derived from the first, second, $\ldots, q$th group of intervals in order to obtain a permutation of length $2pq + 2q$.

**Fig. 9**: Construction of tight examples $\pi(p, q)$ from Definition 7 for Heuristic-4 (Figure 8)

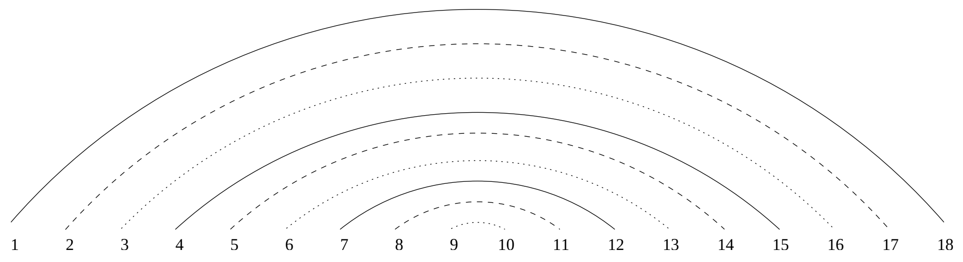

**Fig. 10**: The construction for $p = 3, q = 3$

of $H_\pi$ to the substrings $w_i z_i$, $B$ is the restriction to substrings $x_j$.) Since $H_\pi$ has no edges with one endpoint in $A$ and one in $B$, these induced subgraphs completely cover $H_\pi$.

Our first observation is that no edge from $A$ crosses any edge from $B$. So we can independently find largest non-crossing sets in these two graphs and put them together. We will show that largest noncrossing sets in $A$ and $B$ have sizes exactly $p + 1$ and $p - 1$ respectively, proving the lemma.

We first show the lower bound: we exhibit noncrossing sets of size $p + 1$ and $p - 1$ in $A$ and $B$ respectively. But this is easy; all edges linking adjacent elements

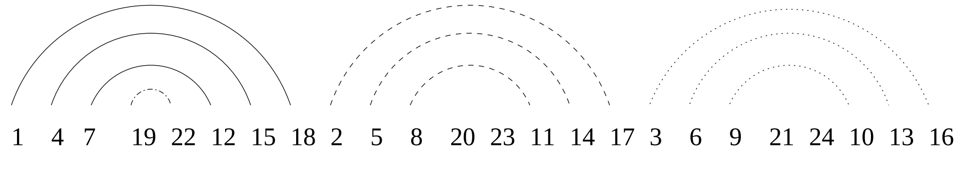

**Fig. 11**: The structure of $\pi(3, 3)$ and its largest compatible set

in any $w_i z_i$ or any $x_j$ gives the required set.

Now we come to the upper bound, which is highly non-trivial. We consider $A$ first. If we arrange the nodes of $A$ in increasing order, we can group them into $p+2$ groups each containing $q$ elements. The $l$th group $g_l$ has elements $ql+1, ql+2, \ldots, ql+q$ for $0 \leq l \leq p-1$, while the groups $g_p$ and $g_{p+1}$ have elements $2pq+1, \ldots, 2pq+q$ and $2pq+q+1, \ldots, 2pq+2q$, respectively. The edges of $A$ connect elements across groups if their position within the group is the same. Let us denote by $B_l$ the boundary between groups $g_{l-1}$ and $g_l$, for $1 \leq l \leq p+1$. An edge originating in group $g_l$ and ending in group $g_k$ spans boundaries $B_{l+1}, \ldots, B_k$. We now analyze how edges in a noncrossing set can cross boundaries.

Given any noncrossing set $C$ in $A$, consider the directed graph $G_C = (C, F)$, where $(e, f) \in F$ if $e$ contains $f$ by value and there is no $f' \in C$ distinct from $e, f$ such that $e$ contains $f'$ and $f'$ contains $f$. Now, our argument is as follows:

(1) $G_C$ is acyclic, with each component being a rooted tree.
(This is obvious; we cannot have directed cycles based on containment by value.)

(2) The sets of boundaries spanned by roots of distinct trees are disjoint. Also, the sets of boundaries spanned by siblings are disjoint.
(This follows from the fact that if the set of boundaries spanned by $e$ intersects the set of boundaries spanned by $f$, then either $e$ must contain $f$ or $f$ must contain $e$ or $e$ and $f$ must cross.)

(3) At each node $e$ of $G_C$, the corresponding edge spans at least one boundary not spanned by the edges of any of its children.
(Suppose not. Let $e$ originate in group $g_l$ and end in group $g_{l+k}$, so it spans boundaries $B_{l+1}, \ldots, B_{l+k}$. Say $e$ is $(ql+i, q(l+k)+i)$ for some $i$. Since $B_{l+1}$ is spanned by a child, the first child must originate in $g_l$; since it does not cross $e$, it must originate "to the right" of $e$, at $ql+j$ for some $j > i$. The next child must originate in the same group that the first child ends, otherwise a boundary will be left unspanned. So it must also originate to the right of $e$. Continuing this way, the edge corresponding to every child of $e$ originates to the right of $e$. But this means that the last child cannot span $B_{l+k}$; to do so, it would have to reach a node in group $g_{l+k}$ to the right of $q(l+k)+i$, thus crossing $e$.)

(4) A leaf spans at least one boundary.
(This is because there are no edges of $A$ within a group; all edges span at least one boundary.)

(5) The number of nodes in the subtree rooted at $e$ is at most the number of boundaries spanned by $e$.
(This follows from straightforward induction on the height of $e$ in $G_C$. (4) above is the base case, and (3) carries the induction through.)

It now follows from (2) and (5) above that the number of vertices in the entire forest is at most the number of boundaries, which is $p+1$. But the number of vertices in the forest is $|C|$, the size of the non-crossing set. Hence $|C| \leq p+1$.

Now consider the induced subgraph $B$. Its structure is isomorphic to that of $A$, the only difference being that it has $p$ groups, not $p + 2$, and so $p - 1$ boundaries. It follows that a largest noncrossing set in $B$ has size at most $p - 1$. □

L 12. $a(\pi(p, q)) = \mathsf{bm}(\mathbb{S}_{\pi(p,q)})$.

P . Let $\pi$ denote $\pi(p, q)$. We know that a block move $\rho$ satisfies $\mathsf{bm}(\mathbb{S}_\pi) - \mathsf{bm}(\mathbb{S}_{\pi.\rho}) > 1$ only if it leads to a permutation with a larger increasing substring. A straightforward inspection shows that there are no such block moves possible in $\pi$. Therefore the heuristic moves the last block of $\pi$. The resultant permutation, say $\pi'$, also has $\mathsf{bm}(\mathbb{S}_{\pi'}) - \mathsf{bm}(\mathbb{S}_{\pi'.\rho}) = 1$ for every block move $\rho$. So the heuristic again moves the last block of $\pi'$. In fact, the heuristic moves the last block in each of the subsequent steps, all the way until the permutation is sorted. At no point along the way does it reach a permutation $\phi$ on which there is a block move $\rho$ with $\mathsf{bm}(\mathbb{S}_\phi) - \mathsf{bm}(\mathbb{S}_{\phi.\rho}) > 1$.

Since at no step does $\mathsf{bm}(.)$ drop by more than one, and since $\mathsf{bm}(.)$ drops by one at each stage, it follows that this sequence is of length $\mathsf{bm}(\mathbb{S}_\pi)$. (In fact, one can also directly see that this sorting sequence is an optimal merging sequence for $\mathbb{S}_\pi$.) □

L 13. $\mathsf{bs}(\pi) \leq pq + 2q - 2$.

P . Let $\pi = \pi(p, q)$, $p$, and $q$ be as given. In the construction of $\pi$, the $pq$ intervals in $A = \{(1, 2pq), (2, 2pq-1), (3, 2pq-2), \ldots, (pq-1, pq+2), (pq, pq+1)\}$ are partitioned into $q$ groups each of size $p$, and the intervals in each group are so nested and placed in $\pi$ so that they constitute a compatible set of edges in the associated graph $G[\pi]$.

The remaining elements $2pq + 1, 2pq + 2, \ldots, 2pq + 2q$ occur as adjacent pairs $2pq + i$, $2pq + q + i$, $1 \leq i \leq q$, in the permutation. The corresponding edges $(2pq + i, 2pq + q + i)$, $1 \leq i \leq q$, of $G[\pi]$ interleave mutually. But none of these $q$ edges interleaves with any of the edges in $A$. Thus exactly one of these edges can be added to $A$ to get a compatible set of size $|A| + 1 = pq + 1$.

Hence $c_\pi \geq pq + 1$, and so from Theorem 3, it follows that $\mathsf{bs}(\pi) \leq pq + 2q - 2$.

A more direct way to see this is to exhibit a sorting sequence of this length. Here is one such sequence, corresponding to the compatible set $A \cup \{(2pq + 1, 2pq + 2)\}$ described above. First, collect the elements $2pq + q + 1, \ldots, 2pq + 2q$ together at the location of $2pq + q + 1$; this needs $q - 1$ moves. Second, collect the elements $2pq + 1, \ldots, 2pq + q$ together at the location of $2pq + 1$; this needs $q - 1$ moves. Now we have the elements $2pq + 1, \ldots, 2pq + 2q$ in a single block, say $z$, and we move it to the end. So we have the sequence $w_1 x_q w_2 x_{q-1} \ldots w_q x_1 z$. The $w_q x_1$ part shows that $pq, pq + 1$ are in a single block. Repeatedly move this block to its predecessor. It will also join up with its successor. Thus in $pq - 1$ such moves, the whole sequence is sorted. Overall, we used $(q-1)+(q-1)+1+(pq-1) = pq+2q-2$ moves. □

T 7. *Heuristic-4 (given in Figure 8) is a factor 2 approximation algorithm for . For any* $0 < \epsilon \leq 1$, *the approximation factor achieved by Heuristic-4 exceeds* $2 - \epsilon$.

P . That the approximation factor for the heuristic is no worse than 2 is clear. To see that it is no better, let us consider the ratio $\frac{a(\pi)}{\mathsf{bs}(\pi)}$, where $\pi = \pi(p, q)$ is as defined in Definition 7 and Figure 8.

$$\frac{a(\pi)}{\mathsf{bs}(\pi)} = \frac{\mathsf{bm}(\mathbb{S}_\pi)}{\mathsf{bs}(\pi)} \geq \frac{2(pq - p + q) - 1}{pq + 2q - 2} = 2\frac{1 - \frac{1}{q} + \frac{1}{p} - \frac{1}{2pq}}{1 + \frac{2}{p} - \frac{2}{pq}} = F(p, q)$$

Clearly, for any $\epsilon$, we can choose $p, q$ large enough so that $F(p, q) > 2 - \epsilon$. In fact, for any $\epsilon$, and for any $q > 2/\epsilon$, we can choose $p$ large enough so that $F(p, q) > 2 - \epsilon$. □

### 6.4 *An Alternative Method*

The examples we constructed seem to imply that the heuristic fails to perform well simply because of the way it breaks ties in step 3 of Figure 8. It fails to detect series of 2 or more *bad* blocks moving all of which effects a huge reduction in the block merging distance. (In the above example, moving out the $z_i$ blocks first brings down the merging distance; for $\pi' = w_1 x_q \dots w_q x_1$, it is easy to see that $\mathsf{bm}(\mathbb{S}_{\pi'}) = \mathsf{bs}(\pi') = pq - 1$.) Since "huge reductions" translates to huge increases in the number of "noncrossing edges" in the corresponding merging instance, we propose the strategy of removing all bad blocks at once. This is equivalent to finding a subsequence $\pi'$ of $\pi$ which maximizes, over the choice of all subsequences, the size of the largest noncrossing set in the corresponding merging instance. We note that this method computes the optimal solutions themselves for at least the counterexamples constructed in Subsection 6.3. On the other hand, this method is not tractable, since the number of subsequences over which we wish to maximize a choice is exponentially large.

We observe that by suitably generalizing to permutations the notion of noncrossing sets of edges defined for instances of , we can obtain a larger compatible set than the one obtainable from the method described above. One characterizing property of noncrossing sets viewed as compatible edge sets is that every inclusion by position implies inclusion by value as well. Let us define a *strongly compatible set* to be compatible set which satisfies this property: i.e., if an edge includes another edge by position, then it includes it by value as well. This property ensures the acyclicity of the inclusion graph. See for instance Figure 12. Thus, every noncrossing set in $H_\pi$ is a strongly compatible set in $G_\pi$, and every strongly compatible set in $G_\pi$ is also compatible. This raises two questions: (1) Does the length of a sorting sequence based on a largest strongly compatible set have a better approximation to $\mathsf{bs}(\pi)$ than 2? and (2) Can largest strongly compatible sets be computed efficiently?

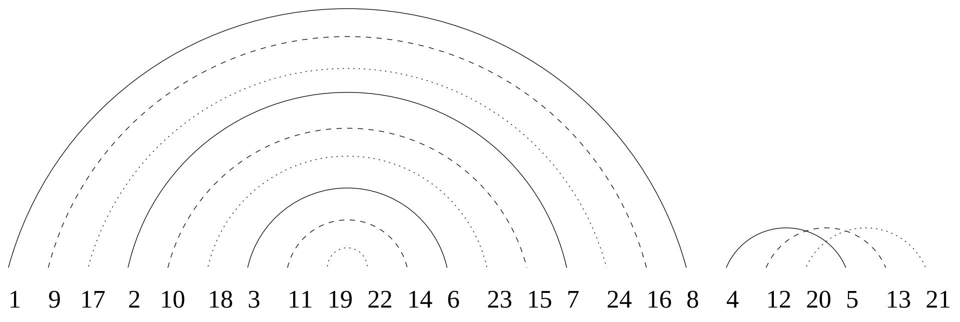

**Fig. 12**: Different strong compatible sets of largest cardinality

We do not know the answers to these questions. It appears that the computation of largest strong compatible sets may be difficult: it lacks optimal substructure property and so straightforward application of dynamic programming does not work. Also, it appears that for the inverses of the permutations $\pi(p, q)$, a largest strong compatible set is quite small (we are unable to construct one larger than $p+1$), whereas there are compatible sets of size $pq+1$ (as follows from Lemma 13). So the ratio of the approximate solution value to the optimal value again approaches 2 as $p, q$ become large.

It is interesting to note that a largest strong compatible set seems to be large for at least one of $\pi$ and $\pi^{-1}$. Thus it becomes worthwhile to analyze the ratio of the corresponding approximate solution values to $\mathsf{bs}(\pi) = \mathsf{bs}(\pi^{-1})$.

## *6.5 Other Heuristics*

For completeness, we briefly discuss other approaches, not necessarily based on . Some fast heuristics for block sorting are discussed in [11, 15]. However, these heuristics do not even have an upper bound of 2 on their approximation ratio, so we do not consider them here. (Our focus here is on explaining why attempts to beat the ratio 2 have not succeeded.)

A 2-factor approximation not based on is proposed in [4]. This algorithm is based on absolute block deletion: given a sequence $\pi$ of distinct integers, repeatedly delete a block until a monotone sequence is left behind. Let the length of the shortest such sequence be denoted $\mathsf{abd}(\pi)$ (or $\mathsf{tabd}(\pi)$ if the empty sequence is to be left behind). It is shown in [4] that $\mathsf{bs}(\pi) \leq \mathsf{abd}(\pi) \leq 2\mathsf{bs}(\pi)$ and that $\mathsf{abd}(\pi)$ is efficiently computable, giving the desired approximation.

There are instances on which absolute-block-deletion outperforms block merging; in the concluding section of [4], an example 3 7 9 4 8 1 5 2 6 is presented where the former leads to a sequence of 4 moves (delete 2, 8, 1, 4 5 6) while block merging needs 6 moves (move in that order 2, 3, 4, 6, 8, then 7 8 9). Also, for the permutations $\pi$ of Example 1, $\mathsf{abd}(\pi) = \mathsf{bs}(\pi) = n - 1$, while $\mathsf{bm}(\pi) = 2n - 2$. On the other hand, Example 3 shows that block merging can outperform absolute-block-deletion by a multiplicative factor tending to 2.

E 3. Consider the sequence $\sigma(0, n) = 1\ 3\ 5\ \ldots\ (2n-1)\ 2\ 4\ 6\ \ldots\ (2n-2)$ on $2n-1$ elements, and the sequences $\sigma(p, n)$ obtained by translating each element of $\sigma(0, n)$ by $p(2n-1)$. For each pair of positive integers $m, n$, construct the permutation $\tau(m, n)$ on $m(2n-1)$ elements by stringing together $\sigma(m-1, n), \sigma(m-2, n), \ldots, \sigma(1, n), \sigma(0, n)$. For instance, $\sigma(0, 3) = 1\ 3\ 5\ 2\ 4$, and $\tau(2, 3) = \sigma(1, 3)\sigma(0, 3) = 6\ 8\ 10\ 7\ 9\ 1\ 3\ 5\ 2\ 4$.

It is straightforward to see that $\mathsf{bs}(\sigma(0, n)) = \mathsf{bm}(\sigma(0, n)) = \mathsf{abd}(\sigma(0, n)) = n-1$.

To perform block merging on $\tau(m, n)$, we can merge each $\sigma(i, n)$, $i = 0, 1, \ldots, k-1$, with $n-1$ steps. Then we have $m$ non-empty lists of blocks, which can be trivially merged in $m-1$ more moves. Thus $\mathsf{bm}(\tau(m, n)) = (n-1)m + m - 1 = mn - 1$. Also, the first $(n-1)m$ block merge moves are optimal block moves by Theorem 5, and they leave behind a permutation with kernel $\mathtt{rev}_m$, needing $m-1$ block moves to sort. Hence $\mathsf{bs}(\tau(m, n)) = \mathsf{bm}(\tau(m, n)) = mn - 1$.

Via absolute block deletion, it is easy to see that the final monotone sequence left behind can have elements from at most one $\sigma(i, n)$. All other $\sigma(j, n)$ must be totally deleted. For each $j$, we have $\mathsf{tabd}(\sigma(j, n)) = 2n-2$ and $\mathsf{abd}(\sigma(j, n)) = n-1$. Thus we get $\mathsf{abd}(\tau(m, n)) = (2n-2)(m-1) + n - 1 = 2mn - 2m - n + 1$.

Thus, as $m, n$ tend to infinity, $\mathsf{abd}(\tau(m, n))$ tends to $2\mathsf{bm}(\tau(m, n)) = 2\mathsf{bs}(\tau(m, n))$.

We note that for the inverses of the permutations presented in Example 3, $\mathsf{abd}(\pi)$ in fact equals $\mathsf{bs}(\pi)$. But this seemingly dual nature of absolute block deletion and block merging does not go too far. For the permutations $\pi(p, q)$ of Section 6.3 as well as their inverses, it can be verified that both $\mathsf{abd}(\pi(p, q))$ and $\mathsf{bm}(\pi(p, q))$ get arbitrarily close to $2\mathsf{bs}(\pi(p, q))$ with increasing values of the parameters $p$ and $q$.

## 7. Conclusion

The combinatorial characterization of optimal solutions of has naturally triggered many computationally relevant explorations. The theorems about good moves prompted the search for nontrivial tight examples for the approximation algorithm of [18]. The heuristic inspired by the problem and its tight examples ultimately leads to the identification of a natural intermediary between noncrossing sets and compatible sets called strong compatible sets. This little understood concept opens a definite new window for further research on the problem.

## Acknowledgments

The results of Sections 3, 4 and 5 appear in a preliminary version titled “Towards constructing optimal block move sequences”[19] in the Proceedings of the 10th International Computing and Combinatorics Conference, COCOON 2004.

This work was done when the third author was with the Department of Mathematics, IIT Madras, India, and then with the Institute of Mathematical Sciences, Chennai, India.

## References

[1] B , V. P , P.. 1996. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing 25*, 272–289.

[2] B , V. P , P.. 1998. Sorting by transpositions. *SIAM Journal on Discrete Mathematics 11*, 2 (may), 224–240.

[3] B , W. W., L , L. L., L , S., S , I. H. 2003. Block Sorting is Hard. *International Journal of Foundations of Computer Science 14*, 3 (june), 425–437.

[4] B , W.W., L , L.L., M , L., S , I.H. 2005. A Faster and Simpler 2-Approximation Algorithm for Block Sorting. In *Proc. of 15th International Symposium on Fundamentals of Computation Theory, LNCS 3623*. Springer, 115–124.

[5] C , D. A. 1999. *Genome Rearrangement Problems*. PhD thesis, Univ. of Glasgow.

[6] E , I. H , T. 2005. A 1.375 approximation algorithm for sorting by transpositions. In *Proc. 5th International Workshop on Algorithms in Bioinformatics WABI, LNCS 3692*. Springer.

[7] E , I. H , T. 2006. A 1.375 approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 3*, 4, 369–379.

[8] E , N.. 2002. $1 + \epsilon$ approximation for sorting by reversals and transpositions. *Theoretical Computer Science 289*, 517–529.

[9] E , H., E , K., K , J., S , L., W¨ , J.. 2001. Sorting a bridge hand. *Discrete Mathematics 241*, 289–300.

[10] G , W. H. P , C. H. 1979. Bounds for sorting by prefix reversals. *Discrete Mathematics 27*, 47–57.

[11] G , R., L , S., B ., W.W. 2000. Adaptive sorting algorithms for evaluation of automatic zoning employed in OCR devices. In *Proceedings of the 2000 International Conference on Imaging Science, Systems, and Technology*. CSREA Press, 253–259.

[12] G , Q. P., P , S., S , H. 1999. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theoretical Computer Science 210*, 2, 327–339.

[13] H , T.. 2003. A simpler 1.5 approximation algorithm for sorting by transpositions. In *Proceedings of 14th Annual Symposium on Combinatorial Pattern Matching, LNCS 2676*. Springer-Verlag, 156–169.

[14] H , T. S , R.. 2006. A simpler and faster 1.5-approximation algorithm for sorting by transpositions. *Information and Computation 204*, 2, 275–290.

[15] K , J., R , S.V., N ., T.A. 1995. Automatic evaluation of OCR zoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 86–90.

[16] K , J. S , D. 1995. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica 13*, 180–210.

[17] M , M., R , R., R , V., V , S. 2003. Merging and sorting by strip moves. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LNCS 2914*. Springer-Verlag, 314–325.

[18] M , M., R , R., R , V., V ., S. 2006. Approximate block sorting. *International Journal of Foundations of Computer Science 17*, 2, 337–355.

[19] M , M., R , R., V , S. 2004. Towards constructing optimal block move sequences. In *Proceedings of the 10th International Computing and Combinatorics Conference, (COCOON), LNCS 3106*. Springer-Verlag, 33–42.

[20] V , J. P. C. 1997. *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytechnique Institute and State University.