On Sorting by 3-Bounded Transpositions

Meena Mahajan *

The Institute of Mathematical Sciences, Chennai 600 113, INDIA.

Raghavan Rama

Department of Mathematics, Indian Institute of Technology Madras, Chennai 600 036, INDIA

S. Vijayakumar [1]

Department of Mathematics, Anna University, Chennai 600 025, India.

Abstract

Heath and Vergara (1) proved the equivalence between sorting by 3-bounded transpositions and sorting by correcting skips and correcting hops. This paper explores various algorithmic as well as combinatorial aspects of correcting skips/hops, with the aim of understanding 3-bounded transpositions better.

We show that to sort any permutation via correcting hops and skips, $\lfloor n/2 \rfloor$ correcting skips suffice. We also present a tighter analysis of the 4/3 approximation algorithm of Heath and Vergara, and a possible simplification. Along the way, we study the class H_n of those permutations of S_n which can be sorted using *correcting hops alone*, and characterize large subsets of this class. We obtain a combinatorial characterization of the set $G_n \subseteq S_n$ of all *correcting-hop-free* permutations, and describe a linear-time algorithm to optimally sort such permutations. We also show how to efficiently sort a permutation with a minimum number of correcting moves.

* Corresponding Author.
Email addresses: `meena@imsc.res.in` (Meena Mahajan), `ramar@iitm.ac.in` (Raghavan Rama), `vjyvjy@yahoo.com` (S. Vijayakumar).
[1] This work was done when this author was with the Department of Mathematics, IIT Madras, India.

1 Introduction

A particular type of mutational event occurring in the genomic sequences of certain species is the removal of a long substring from some location and its subsequent re-insertion in some other location (2). This event may also be seen as an exchange of two adjacent non-overlapping substrings. Such an event is called a transposition. It is of interest to biologists to know how the genomic sequences of two species are related, when one is hypothesized to have evolved from the other due to a sequence of transpositions in their genomic sequences. In particular, the number of transpositions that are necessary if it is assumed that Nature operates in an optimal way is of some importance.

Since the substrings involved in a transposition are generally long, the effect of the transposition is felt accordingly in the gene order in the chromosome. As a first abstraction, different species with the same set of genes, say $1, 2, \ldots, n-1, n$, may be labelled by the order of these genes in their respective genomes, and transpositions over the genome sequences may be thought of as transpositions over the corresponding labelling permutations. This motivates the problem of finding the *transposition distance* between permutations, namely the shortest number of transpositions that will transform one permutation into another. Computing the transposition distance is easily seen to be equivalent to the problem of transforming a permutation into the identity permutation, i.e. *sorting* permutations with transpositions.

The problem of sorting by transpositions, as well as its bounded variation, first appeared in a list of genome rearrangement problems presented by Kececioglu and Sankoff (3). Bafna and Pevzner (4) first gave a 3/2-approximation algorithm for sorting permutations with a minimum number of transpositions. This was followed by simpler 3/2-approximation algorithms by Christie and Hartman (5; 6). Heath and Vergara studied bounded variations of this problem(7; 1). For sorting by 3-bounded transpositions, they gave a 4/3-approximation general algorithm, and polynomial-time algorithms for special classes of permutations. The computational complexity of both these problems remains open. Related work appears in (8; 9; 10; 11; 12); the reader is referred to (13) for an overview.

In (7), Heath and Vergara showed that while sorting by 3-transpositions, correcting moves – moves which do not introduce new inversions – suffice. A correcting 3-transposition is either a correcting hop (an element moves to a position two places away) or a correcting skip (an element moves to a position one place away), and as shown in (7), it suffices to minimize the number of correcting skips used.

In this paper we consider only correcting moves. Our study comprises of roughly four parts, organized as follows.

In Section 3, we study the class H_n, consisting of permutations that can be sorted with correcting hops alone. We find efficient recognizers for L_n and R_n, the subsets of H_n which can be sorted by correcting left hops alone and correcting right hops alone, respectively (Theorem 3.6). We also describe how to optimally sort such permutations, and we determine the cardinalities of L_n and R_n. (It is easy to see that H_n is larger than $L_n \cup R_n$. In Section 6, we extend some of the proof ideas to identify larger subsets of H_n.)

In Section 4, we focus mainly on the class G_n, consisting of correcting-hop-free permutations, and its subset CG_n consisting of connected permutations. We obtain combinatorial characterizations of G_n and CG_n (Theorems 4.3 and 4.7), and determine the cardinality of G_n (Theorem 4.4). Using the combinatorial characterizations, we describe linear time algorithms to compute the length of an optimal 3-transposition sorting sequence when $\pi \in G_n$ (Theorem 4.14), and to describe an optimal sequence of correcting skips/hops (Theorem 4.16). (The optimal sequence length need not be $O(n)$; however, it has a succinct $O(n)$ sized description.) The optimal sequences we construct have an interesting property: they consist of the minimum number of correcting skips all of which are applied initially in parallel to obtain a permutation in $L_n \cup R_n$.

In Section 5, we establish an upper bound: we show (Theorem 5.2) that $\lfloor \frac{n}{2} \rfloor$ correcting skips suffice to sort any permutation via 3-transpositions, and hence the optimal 3-transposition sorting sequence has at most $(\#\texttt{inv}(\pi) + \lfloor \frac{n}{2} \rfloor)/2$ moves, where $\#\texttt{inv}(\pi)$ is the number of inversions in π. This bound, although incomparable with the $\lceil \binom{n}{2}/2 \rceil$ bound of (7), is better for a very large class of permutations. Our bound, along with the characterization of CG_n, provides a tighter analysis (Theorem 5.6) of the 4/3 approximation algorithm of (1) and a simplification (Algorithm GREEDYHOPS-3).

We consider correcting moves in Section 7. We observe that minimizing the number of correcting skips/hops is equivalent to minimizing the number of *odd* correcting moves *over* sequences of correcting moves. We show that the minimum number of correcting moves required to sort a permutation is efficiently computable; in fact, an optimal sequence of such moves can also be efficiently computed.

2 Preliminaries and Notation

Here we formally define the basic terms/notions/notation used in this paper. We denote by $[i, j]$ the set $\{i, i + 1, \ldots, j\}$ and by $[n]$ the set $[1, n] = \{1, 2, \ldots, n\}$. We denote by $|s|_1$ the number of 1s in a string $s \in \{0, 1\}^*$. The *symmetric group* over n elements is denoted S_n, and a *permutation* $\pi \in S_n$ is a linear order of the elements of $[n]$. An inversion in π is a pair of elements $\{\pi_i, \pi_j\}$ that are not in correct relative order ($i < j$ and $\pi_i > \pi_j$). By $\texttt{Inv}(\pi)$ we denote the set of inversions in π, and the size of this set is denoted $\texttt{\#inv}(\pi)$. We denote by $\texttt{id}_n$ the identity permutation on n elements.

A component of a permutation π is a minimal substring $\pi_i ... \pi_j, i \leq j$ such that (1) $\pi_l < i$ for $l < i$, (2) $i \leq \pi_l \leq j$ for $i \leq l \leq j$, and (3) $\pi_l > j$ for $l > j$. A permutation is said to be *connected* if it has exactly one component.

A *transposition* $\tau_{(i,j,k)} : S_n \to S_n,\ 1 \leq i \leq j < k \leq n$, is defined in the following way: $\pi.\tau_{(i,j,k)} = \pi_1 \dots \pi_{i-1}\pi_{j+1} \dots \pi_k \pi_i \dots \pi_j \pi_{k+1} \dots \pi_n$; i.e., $\tau_{(i,j,k)}$ when applied on π exchanges the adjacent substrings $\pi_i \dots \pi_j$ and $\pi_{j+1} \dots \pi_k$. We say that a transposition $\tau_{(i,j,k)}$ is *b-bounded* for a given constant b if $k - i + 1 \leq b$; i.e., the total length of the two substrings involved in the transposition is bounded by b. Let $t(\pi)$ denote the length of any shortest sequence of transpositions sorting π and $t_b(\pi)$ denote the length of any shortest sequence of b-bounded transpositions sorting π. In the literature, these terms are also referred to block moves (transpositions), short block moves (b-bounded transpositions), $Min_{Bk}(\pi)(= t(\pi))$ and $Min_{Bk^b}(\pi)(= t_b(\pi))$.

A *move* of π_i in π is a transposition where one of the involved substrings is a single element, i.e., it is a transposition between π_i and an adjacent substring lying to its left or right. A *correcting move* is a move which does not introduce any new inversions. In particular, a correcting move of π_i over a substring α implies that all elements of α are less (greater) than π_i if α lies to the right (left, respectively) of π_i.

A 3-bounded transposition in π, by definition, is either an exchange of two adjacent elements or an exchange of an element and an adjacent substring of length two. Hence these transpositions may equivalently be seen as a move of some element π_i in π past one element *(a skip)* or past two elements *(a hop)*, to the left or right. *Correcting skips/hops* are similarly defined.

Let $m(\pi)$ denote the length of a shortest sequence of moves sorting π, and let $csh(\pi)$ denote the length of any shortest sequence of correcting skips and correcting hops sorting π. The following result was proved by Heath and Vergara in (7):

Theorem 2.1 (Theorem 5 (7).) *For all $\pi \in S_n$, $t_3(\pi) = csh(\pi)$. That is, to optimally sort using 3-transpositions, correcting skips/hops suffice.*

This result indicates that a complete understanding of correcting skips/hops will suffice to resolve questions about $t_3(.)$. Clearly, $csh(\pi) \leq \texttt{\#inv}(\pi)$, since starting with $i = 1$ and going up to $i = n$, we can repeatedly move i left via correcting skips alone to its correct place. On the other hand, a correcting hop (skip) reduces the number of inversions by exactly two (one, respectively), and $\texttt{\#inv}(\texttt{id}_n) = 0$. Thus it follows that

$$\left\lceil \frac{\texttt{\#inv}(\pi)}{2} \right\rceil \leq csh(\pi) \leq \texttt{\#inv}(\pi) \tag{1}$$

Any sequence of s correcting skips and h correcting hops sorting π satisfies $s+h \geq csh(\pi)$ and $s+2h = \texttt{\#inv}(\pi)$. If the sequence is optimal, then $s+h = csh(\pi)$. Thus one can see that every optimal sequence has the same number of correcting skips, say $cs(\pi)$, and that no sorting sequence (optimal or otherwise) has fewer correcting skips. Hence computing $csh(\pi)$ is equivalent to computing $cs(\pi)$; in fact,

$$csh(\pi) = \frac{\texttt{\#inv}(\pi) + cs(\pi)}{2} \tag{2}$$

We denote by H_n the set of those permutations $\pi \in S_n$ for which $cs(\pi) = 0$, or equivalently, the set of permutations that can be sorted using correcting hops alone. We denote by G_n the set of all correcting-hop-free permutations in S_n, and by CG_n its subclass of connected correcting-hop-free permutations.

3 Permutations Sortable by Correcting Hops

One implication of Theorem 2.1 and Equation 2 is that computing $t_3(\pi)$ is equivalent to computing $cs(\pi)$. This then naturally suggests a partition of S_n into classes based on $cs(.)$. Here we concentrate on the class H_n of those permutations for which $cs(\pi) = 0$ (i.e. permutations that can be sorted using correcting hops alone). Note that for $\pi \in H_n$, $t_3(\pi) = csh(\pi) = \frac{\texttt{\#inv}(\pi)}{2}$.

We define below certain structures associated with each permutation. These structures will allow us to identify large subclasses of H_n.

Definition 3.1 *Let π be any permutation in S_n. For $u \in [n]$,*

(1) $\texttt{left}(u, \pi)$ *is the set of elements v greater than u appearing to the left of u in π.* $\texttt{left}(u, \pi) = \{v > u \mid (u, v) \in \texttt{Inv}(\pi)\}$.
$p(u, \pi) = |\texttt{left}(u, \pi)| \bmod 2$.
(2) $\texttt{right}(u, \pi)$ *is the set of elements w less than u appearing to the right of u in π.* $\texttt{right}(u, \pi) = \{w < u \mid (u, w) \in \texttt{Inv}(\pi)\}$.
$q(u, \pi) = |\texttt{right}(u, \pi)| \bmod 2$.
(3) $p(\pi) = p(\pi_1, \pi)p(\pi_2, \pi) \ldots p(\pi_n, \pi)$.
$q(\pi) = q(\pi_1, \pi)q(\pi_2, \pi) \ldots q(\pi_n, \pi)$.

Note that $\texttt{Inv}(\pi) = \cup_u \texttt{left}(u, \pi) = \cup_v \texttt{right}(v, \pi)$

Proposition 3.2 *For $\pi \in S_n$ and $i \in [n]$, $\pi_i + |\texttt{left}(\pi_i, \pi)| = i + |\texttt{right}(\pi_i, \pi)|$.*

Proof: Let $\pi_i = a$, $|\texttt{left}(a, \pi)| = k$, and $|\texttt{right}(a, \pi)| = l$. Of the $i-1$ elements to the left of a, k are larger than a, so the remaining $i-1-k$ are smaller than a. Thus, of the $a-1$ elements less than a, $i-1-k$ are already to the left of a. The remaining $(a-1)-(i-1-k)$ form the set $\texttt{right}(a, \pi)$, giving the claimed equality. $\square$

Hops can be either to the left or to the right, and left hops and right hops affect the underlying permutation differently. Let σ be the permutation resulting from the application of a correcting hop or skip to π. For elements u not involved in the move, $p(u, \sigma) = p(u, \pi)$ and $q(u, \sigma) = q(u, \pi)$. For elements involved in the move, the table below summarizes the effect of the move on $p()$ and $q()$. The entries are self-explanatory.

	Initial position			Correcting left hop ($w < u$; $w < v$)			Correcting right hop ($u > v$; $u > w$)			Correcting skip ($u > v$)	
element	u	v	w	w	u	v	v	w	u	v	u
$p(x, \pi)$	p_1	p_2	p_3	p_3	p_1	p_2	$\overline{p_2}$	$\overline{p_3}$	p_1	$\overline{p_2}$	p_1
$q(x, \pi)$	q_1	q_2	q_3	q_3	$\overline{q_1}$	$\overline{q_2}$	q_2	q_3	q_1	q_2	$\overline{q_1}$

Table 1
How correcting hops/skips affect $p()$ and $q()$.

Lemma 3.3 *For all $\pi \in H_n$, the quantities $\#\texttt{inv}(\pi)$, $|p(\pi)|_1$ and $|q(\pi)|_1$ are even.*

Proof: A correcting hop decreases the number of inversions exactly by two. And $\texttt{id}_n$ has no inversions. So if π can be transformed to $\texttt{id}_n$ via correcting hops alone, then $\#\texttt{inv}(\pi)$ must be even.

As can be seen from Table 1, if a correcting hop is applied to π yielding σ, then $|p(\sigma)|_1 = |p(\pi)|_1 + x$ and $|q(\sigma)|_1 = |q(\pi)|_1 + y$ where $x, y \in \{-2, 0, 2\}$. (For instance, in $p(\pi)$ and $p(\sigma)$, a correcting left hop leaves the two underlying bits unaltered whereas a correcting right hop flips the two underlying bits, and, in both the cases, the bit that hops remains unaffected.) But $|p(\texttt{id}_n)|_1 = |q(\texttt{id}_n)|_1 = 0$. The result follows. $\square$

This result is not a characterization: for the permutation $\pi = 4\ 1\ 5\ 2\ 6\ 3$, all of $\#\texttt{inv}(\pi)$, $|p(\pi)|_1$ and $|q(\pi)|_1$ are even, but $\pi \notin H_n$.

We consider the following questions: "which permutations can be sorted using correcting left hops alone?" and "which can be sorted with correcting right hops alone?" Let us denote by L_n the set of all permutations of length n which can be sorted using correcting left hops alone and by R_n the set of all permutations which can be sorted with correcting right hops alone. That $L_n \neq R_n$ is obvious. In fact, 2 3 1 4...$n \in L_n - R_n$, 3 1 2 4 ...$n \in R_n - L_n$, and 3 4 1 2 5...$n \in L_n \cap R_n$. It follows from the definitions that $L_n \cup R_n \subseteq H_n$; this inclusion is proper, for 4 1 2 5 3 6...$n \in H_n - (L_n \cup R_n)$. Even though the classes L_n and R_n are distinct, nevertheless they are symmetric in their properties. In the rest of this section, we explicitly prove results for either L_n or R_n but not for both.

Lemma 3.4 *$p^{-1}(0^n) \subseteq L_n$, and $q^{-1}(0^n) \subseteq R_n$.*

Proof: let $\pi \in p^{-1}(0^n)$. Then every π_i has an even number of larger elements to its left in π. This is in particular true for 1. Hence 1 can be moved to the extreme left through a sequence of correcting left hops. From Table 1, the $p()$ string for the resulting permutation is still 0^n. So 2 can be moved to the immediate right of 1 via correcting left hops. Repeatedly, after having positioned elements $1, 2, \ldots i$ correctly via correcting left hops, there are an even number of larger elements to the left of $i+1$ (and to the right of i). So, $i+1$ can be taken to the immediate right of i via a sequence of correcting left hops. In this manner the permutation can be transformed to $\mathtt{id}_n$. $\square$

Lemma 3.5 *For every π in L_n, $p(\pi) = 0^n$. For every π in R_n, $q(\pi) = 0^n$.*

Proof: As we can see from Table 1, if a correcting left hop is applied to π yielding σ, then $|p(\pi)|_1 = |p(\sigma)|_1$. Using this, we see that if π is in L_n, then $|p(\pi)|_1 = |p(\mathtt{id}_n)|_1$. But $|p(\mathtt{id}_n)|_1 = 0$, hence $p(\pi) = 0^n$. $\square$

These lemmas give us the following characterization of the sets L_n and R_n.

Theorem 3.6 *$L_n = p^{-1}(0^n)$ and $R_n = q^{-1}(0^n)$*

Though the classes L_n and R_n appear rather restrictive, the following theorem shows that each of them is actually fairly large. Hence the class H_n, which contains $L_n \cup R_n$, is also fairly large.

Theorem 3.7 $|p^{-1}(0^n)| = |q^{-1}(0^n)| = \left(\left\lceil \frac{n-1}{2} \right\rceil\right)! \left(\left\lfloor \frac{n+1}{2} \right\rfloor\right)!$.

Proof: A permutation π which is constructed from the empty string ϵ by inserting successive smaller elements, starting downward from n, in such a way that they have an even number of larger elements to their left, is obviously in $p^{-1}(00...00)$. It is also obvious that every $\pi \in p^{-1}(00...00)$ has such a (unique) construction. To count $|p^{-1}(0^n)|$, we just need to count the number of ways in which such a construction can be carried out.

Consider such an incremental construction of a permutation π in $p^{-1}(0^n)$. We start with s_0 being the empty string ϵ. After i stages, we have a string s_i which has the elements $n-i+1, n-i+2, \ldots, n$. At the $(i+1)$th stage, for $i \in [n]$, we insert the element $n-i$ into the string s_i to obtain the string s_{i+1}. If at this stage we ensure that $n-i$ has an even number of larger elements to its left in s_{i+1}, then $n-i$ will have an even number of larger elements to its left in s_n as well, since all subsequent insertions are of elements smaller than $n-i$ and do not affect this property.

At the $(i+1)$th stage, since $|s_i| = i$, there are $i+1$ slots in which a new element may be inserted. Exactly $\lceil (i+1)/2 \rceil$ of these will allow $n-i$ to have an even number of larger elements to its left. Thus the construction can be carried out in $\prod_{i=0}^{n-1} \lceil (i+1)/2 \rceil$ ways, giving the theorem. $\square$

4 Correcting-Hop-Free Permutations

We know permutations which don't have any correcting hops at all: 2 1 and 3 1 4 2 are such. Here we analyze the class $G_n \subseteq S_n$ of all correcting-hop-free permutations, and its subset CG_n of connected correcting-hop-free permutations. As stated below, this class is essentially disjoint from the class H_n considered in the previous section.

Proposition 4.1 $G_n \cap H_n = \{\mathtt{id}_n\}$.

Proof: Permutations in $G_n - \mathtt{id}_n$ have no correcting hops at all, to start with. So, it is necessary that we use a positive number of correcting skips to sort them. $\square$

4.1 Characterizing G_n and CG_n

We consider the classes $G_n = \{\pi \in S_n \mid \pi \text{ is correcting-hop-free}\}$ and $CG_n = \{\pi \in G_n \mid \pi \text{ is connected}\}$. Recognizing G_n itself is trivial: there are $2n-4$ possible hops in a permutation in S_n (π_1 and π_2 can only hop right, π_{n-1} and π_n can only hop left, all other elements can hop both left and right), and in $O(n)$ time we can check whether any of them is a correcting hop. Recognizing CG_n is equally straightforward; check if π is in G_n and check if it is connected.

However, these recognition procedures do not help us to optimally sort such permutations using correcting hops/skips. In this subsection, we analyze the kinds of permutations that are in G_n and in CG_n and obtain a combinatorial characterization. This characterization will be used in the next subsection to efficiently sort these permutations using correcting hops/skips.

Definition 4.2 *Let $\pi \in S_n$.*

(1) π has the Increasing-Even-Sequence property, IES, if $\pi_2 < \pi_4 < \ldots$; that is, the even-positioned elements form an increasing sequence.
(2) π has the Increasing-Odd-Sequence property, IOS, if $\pi_1 < \pi_3 < \ldots$; that is, the odd-positioned elements form an increasing sequence.
(3) π has the Odd-Dominates-Next-2-Even property, ODN2E, if $\pi_{2k-1} > \pi_{2k}$ and $\pi_{2k-1} > \pi_{2k+2}$ for all appropriate k; that is, each odd-positioned element is larger than the succeeding two even-positioned elements.

Theorem 4.3 *A permutation is correcting-hop-free if and only if it has the IES and the IOS properties.*

Proof: First, assume that $\pi \in S_n$ has the IES and IOS properties. Then it is clear that there is no correcting hop, because any hop moves some element π_i over two of its neighbours, and by the IES and IOS properties, a move of π_i over π_{i-2} or over π_{i+2} introduces an inversion.

Now, suppose π fails either the IES or the IOS property. Then there is an index i such that $u = \pi_i > \pi_{i+2} = v$. Let $w = \pi_{i+1}$, so that uwv is a substring of π. If $w > u$, then $w > v$ and so moving v left is a correcting hop. If $w < u$, then moving u right is a correcting hop. Either way, π has a correcting hop and so is not in G_n. $\square$

Theorem 4.4 *The number of correcting-hop-free permutations on n elements is exactly $\binom{n}{\lfloor n/2 \rfloor}$.*

Proof: Theorem 4.3 gives a simple procedure to generate all permutations in G_n: choose any subset $S \subseteq [n]$ of size $\lceil n/2 \rceil$, arrange the elements of S in increasing order, arrange the elements of $[n] \setminus S$ in increasing order, and then take the perfect shuffle of the two sequences so obtained, starting with the smallest element of S. The number of ways in which this can be done is precisely $\binom{n}{\lfloor n/2 \rfloor}$. $\square$

Lemma 4.5 *For a correcting-hop-free permutation $\pi \in G_n$, for each $i \in [n]$,*

	$\pi_i \leq i$	$\pi_i \geq i$
$\lvert\texttt{left}(\pi_i, \pi)\rvert$	$i - \pi_i$	0
$\lvert\texttt{right}(\pi_i, \pi)\rvert$	0	$\pi_i - i$

Proof: Since $\pi \in G_n$, by Theorem 4.3 it has the IES and IOS properties. So $\texttt{left}(\pi_i, \pi) \subseteq \{\ldots, \pi_{i-3}, \pi_{i-1}\}$ and $\texttt{right}(\pi_i, \pi) \subseteq \{\pi_{i+1}, \pi_{i+3}, \ldots\}$. Furthermore, if $\texttt{left}(\pi_i, \pi) \neq \emptyset$, then $\pi_{i-1} > \pi_i$ and due to IES and IOS,

$\mathtt{right}(\pi_i, \pi) = \emptyset$. Similarly, if $\mathtt{right}(\pi_i, \pi) \neq \emptyset$, then $\mathtt{left}(\pi_i, \pi) = \emptyset$.

Now we use Proposition 3.2. If $\pi_i = i$, then $|\mathtt{left}(\pi_i, \pi)| = |\mathtt{right}(\pi_i, \pi)|$. But $\mathtt{left}(\pi_i, \pi)$ and $\mathtt{right}(\pi_i, \pi)$ cannot both be non-empty; so they must both be empty. If $\pi_i < i$, then $|\mathtt{left}(\pi_i, \pi)| > |\mathtt{right}(\pi_i, \pi)|$. Since at least one of these two sets is empty, it must hold that $|\mathtt{right}(\pi_i, \pi)| = 0$, and then $|\mathtt{left}(\pi_i, \pi)| = i - \pi_i$. Similarly, If $\pi_i > i$, then $|\mathtt{left}(\pi_i, \pi)| < |\mathtt{right}(\pi_i, \pi)|$, and so it must hold that $|\mathtt{left}(\pi_i, \pi)| = 0$ and $|\mathtt{right}(\pi_i, \pi)| = \pi_i - i$. $\square$

Lemma 4.6 *If a permutation $\pi \in S_n,\ n > 1$, is connected and correcting-hop-free (that is, $\pi \in CG_n$), then n is even.*

Proof: For permutations of S_2 the lemma is vacuously true. The only connected permutations of S_3 are 3 2 1, 3 1 2, and 2 3 1, and each of these has a correcting hop. Let $\pi \in S_n,\ n \geq 4$, be connected and correcting-hop-free. We establish that n is necessarily even. We first prove a claim about connected correcting-hop-free permutations.

Claim 1 *If π in CG_n, then it does not have a monotone substring (increasing or decreasing) of length 3.*

Proof-of-Claim: We will show that every internal element $\pi_i, 1 < i < n$, in π is flanked either by two smaller elements or by two larger elements.

Suppose one of π_{i-1} and π_{i+1} is smaller and the other is larger than π_i for some $1 < i < n$. From Theorem 4.3, it cannot be true that $\pi_{i-1} > \pi_{i+1}$. So assume that $\pi_{i-1} < \pi_i < \pi_{i+1}$. From connectedness, either there is a $k < i$ such that $\pi_k > \pi_i$ or there is $k > i$ such that $\pi_k < \pi_i$ (or both). Let us assume that the former is true (the other case is symmetric). Then the rightmost such π_k has two smaller elements to its immediate right and so π has a correcting right hop, which is a contradiction. Hence the claim. $\square$

Since π is connected and correcting-hop-free, it is necessary that $\pi_2 = 1$ and $\pi_{n-1} = n$. Therefore, from the above claim, it follows that every π_{2i} is flanked by two larger elements. But $\pi_{n-1} = n$ is necessarily flanked by two smaller elements; so $n - 1$ cannot be even or n cannot be odd. $\square$

Theorem 4.7 *A permutation $\pi \in S_n$, with $n > 1$, is connected and correcting-hop-free if and only if it satisfies the following properties: (1) π has the IES and IOS properties, (2) n is even, and (3) π has the ODN2E property.*

Proof: First, let us see why these conditions suffice. Let π satisfy all three conditions. Condition (1) implies, by Theorem 4.3, that π is correcting-hop-free. By condition (3), every even i satisfies $\pi_i < \pi_{i-1}$, so a component cannot begin at such an i. For every odd $i > 1$, by condition (2), there is a succeeding element π_{i+1}, and by condition (3), $\pi_{i+1} < \pi_{i-2}$. So a component cannot begin at such an i. Thus π has only one component beginning at position 1; i.e. it is connected.

Now we show that these conditions are necessary. Let π be a connected correcting-hop-free π. By Theorem 4.3, π satisfies the first condition. By IES and IOS, $n \in \{\pi_{n-1}, \pi_n\}$. Also, by Lemma 4.6, π satisfies the second condition, so n is even.

Suppose π does not satisfy the ODN2E property. Then there is an index $2j-1$ where this property is violated. Consider the largest such index $2i-1$. If $2i-1 = n-1$, then $\pi_{n-1} < \pi_n$. Since $n \in \{\pi_{n-1}, \pi_n\}$, it must be that $n = \pi_n$, making π_n a component of π. This contradicts the fact that π is connected. So $2i-1 < n-1$. Since n is even while $2i-1$ is odd, we have in fact $2i-1 < n-2$; that is, $n \geq 2i+2$. Let $\pi_{2i-1} = a$, $\pi_{2i} = b$, $\pi_{2i+1} = c$, and $\pi_{2i+2} = d$. By IES and IOS, $a < c$ and $b < d$. By choice of i, $a < d < c$. So the correct ordering of these 4 elements is either $a < b < d < c$ or $b < a < d < c$. By IES and IOS, it follows that a component of π begins at position $2i+1$, contradicting the fact that π is connected. So no such index can exist, and π must satisfy ODN2E. $\square$

Corollary 4.8 *A permutation π is connected and correcting-hop-free if and only if it can be constructed in the following way: (1) Choose an even number n, (2) Set $\pi_2 = 1$, (3) For $2 \leq i \leq n/2$ choose a value for π_{2i} from the set $\{\pi_{2i-2}+1, \ldots, 2i-2\}$, and (4) Fill up the odd positions with the unused elements of $[n]$ in increasing order.*

Proof: If a permutation π is constructed in this way, then it obviously satisfies the first two conditions of the theorem above. To see why it satisfies ODN2E, note that for each $i > 1$, of the elements $[1, 2i-2]$, exactly i elements are placed in the even positions $2, 4, \ldots, 2i$. So at most $i-2$ elements from this set are available to fill up the odd positions. But there are i odd positions to the left of π_{2i}. So the last two of these, $2i-3$ and $2i-1$, must use an element outside this set, i.e. it must hold that $\pi_{2i-1} > \pi_{2i-3} > 2i-2 \geq \pi_{2i} > \pi_{2i-2}$. Thus ODN2E holds. Hence by the above theorem, $\pi \in CG_n$.

Conversely, if π is in CG_n, then by Theorem 4.7, n is even. Since π has IES and IOS and ODN2E properties, $1 \in \{\pi_1, \pi_2\}$ and $\pi_1 > \pi_2$, so $\pi_2 = 1$. Furthermore, for each index $2i > 2$, the elements π_{2i-3} and π_{2i-1} are larger than π_{2i} by ODN2E, and then by IES and IOS, all elements to the right of π_{2i} are also larger than π_{2i}. Thus there are at most $2i-3$ elements smaller than π_{2i}, so $\pi_{2i-2} < \pi_{2i} \leq 2i-2$. $\square$

For permutations in G_n, Theorem 19 of (1) implies that optimal sorting using correcting skips/hops can be done efficiently. A permutation $\pi \in S_n$ is said to be a woven double-strip permutation if there is a set $S \subseteq [n]$ such that (1) the elements $\{\pi_i \mid i \in S\}$ appear in correct relative order in π, and (2) the elements $\{\pi_i \mid i \notin S\}$ appear in correct relative order in π. Corollaries 24 and 25 of (1) establish that for a woven double-strip permutation π, $csh(\pi)$ (and hence $cs(\pi)$) can be computed in $O(n^2)$ time and an optimal sorting sequence can be found in $O(n^3 \log n)$ time. Note that by Theorem 4.3, every permutation in G_n is a woven double-strip permutation and so these bounds apply. In this section, we improve these bounds for permutations in G_n by establishing a combinatorial result about $cs(\pi)$ when $\pi \in G_n$. We show that for $\pi \in G_n$, an optimal sorting sequence can be found in $O(n)$ time.

Proposition 4.9 *For any π in G_n, and for any $i \in [n]$,*
$\mathtt{left}(\pi_i, \pi) \neq \emptyset \Longrightarrow [\pi_{i-1} \in \mathtt{left}(\pi_i, \pi) \ \wedge \ \mathtt{left}(\pi_{i-1}, \pi) = \emptyset]$.
$\mathtt{right}(\pi_i, \pi) \neq \emptyset \Longrightarrow [\pi_{i+1} \in \mathtt{right}(\pi_i, \pi) \ \wedge \ \mathtt{right}(\pi_{i+1}, \pi) = \emptyset]$.

Proof: Let $\mathtt{left}(\pi_i, \pi) \neq \emptyset$. By Theorem 4.3, π has the IES and IOS properties, so none of the elements $\pi_{i-2}, \pi_{i-4}, \ldots$ belong to $\mathtt{left}(\pi_i, \pi)$. Of the remaining elements to the left of π_i, π_{i-1} is the largest, so if $\mathtt{left}(\pi_i, \pi)$ is non-empty, it must contain π_{i-1}. So $\pi_i < \pi_{i-1}$. But this, along with the IES and IOS properties, implies that $\mathtt{left}(\pi_{i-1}, \pi)$ is empty. The proof for $\mathtt{right}()$ is symmetric. $\square$

Lemma 4.10 $\forall \pi \in G_n, \ \ cs(\pi) \leq \min\{|p(\pi)|_1, \ |q(\pi)|_1\}$. *Furthermore, there is a sorting sequence where min$\{|p(\pi)|_1, \ |q(\pi)|_1\}$ correcting skips are applied initially, in parallel, to obtain a permutation in $L_n \cup R_n$.*

Proof: For any $i \in [n]$, let $p(\pi_i, \pi) = 1$. Clearly, $\mathtt{left}(\pi_i, \pi) \neq \emptyset$. By Proposition 4.9, $\mathtt{left}(\pi_{i-1}, \pi) = \emptyset$ and so $p(\pi_{i-1}, \pi) = 0$. Thus every 1 in $p(\pi)$ is preceded by a 0.

We now describe a sorting sequence, where $|p(\pi)|_1$ correcting skips are applied initially, in parallel, to obtain a permutation in L_n. For any $i \in [n]$, let $p(\pi_i, \pi) = 1$. From Proposition 4.9, $\pi_{i-1} > \pi_i$, so moving π_i one position left is a correcting skip. From Table 1, such a correcting (left) skip will convert the 01 at positions $i-1, i$ to a 00. Thus, after applying correcting (left) skips to each π_i satisfying $p(\pi_i, \pi) = 1$, we get a permutation in $p^{-1}(0^n) = L_n$.

The proof for $|q(\pi)|_1$ correcting skips is symmetric. $\square$

Corollary 4.11 *For $\pi \in G_n$,* $\max\{|p(\pi)|_1, \ |q(\pi)|_1\} \leq \lfloor \frac{n}{2} \rfloor$.

Proof: It follows from definition that $p(\pi_1, \pi) = 0$ (and $q(\pi_n, \pi) = 0$). And the proof of Lemma 4.10 establishes that every 1 in $p(\pi)$ is preceded by a 0 (and every 1 in $q(\pi)$ is followed by a 0). $\square$

To obtain a corresponding lower bound, we use the results about G_n and CG_n from the previous subsection, and a result from (1). First, we present some notation, also from (1). For any $\pi \in S_n$, the permutation graph G_π is the directed graph $G_\pi = (V_\pi, A_\pi)$ where $V_\pi = [n]$ and $A_\pi = \{(\pi_i, \pi_j) \mid i < j \wedge \pi_i > \pi_j\}$. We say that arcs (a, b) and (c, d) are right-compatible (left-compatible) if $b = d$ ($a = c$ respectively). A pair of arcs $(a, b), (c, d)$ is said to be compatible if it is either right-compatible or left-compatible. An element x is an obstacle for a pair of compatible arcs $(a, b), (c, d)$ if (1) $a = c$, $b < x < d$, and in π, these elements occur in relative order a, b, x, d, or (2) $b = d$, $a < x < c$, and in π, these elements occur in relative order a, x, c, b. A pair of arcs is feasible if it is a compatible pair with no obstacle. The arc graph G^a_π is an undirected graph with vertex set $V^a_\pi = A_\pi$, and $((a, b), (c, d)) \in E^a_\pi$ exactly when $(a, b), (c, d)$ is a pair of feasible arcs.

The following result appears in (1); for completeness, we sketch the proof here.

Lemma 4.12 (Lemma 14, Heath and Vergara (1)) *For every $\pi \in S_n$, $csh(\pi) \geq |M| + |U|$ where M is a maximum matching in G^a_π and U is the vertices of G^a_π left unmatched by M.*

Proof: Note that correcting moves do not introduce arcs in G_π. Every correcting skip erases exactly one arc from G_π, every correcting hop erases exactly two compatible arcs. If two compatible arcs are not feasible, then they cannot be erased by a single correcting hop, since the obstacle in between will never go away. (Correcting moves never introduce inversions.) So every correcting hop erases exactly two feasible arcs.

Consider an optimal sequence ρ of h correcting hops and s correcting skips sorting π. This sequence erases all the arcs from G_π, so $2h + s = |A_\pi| = \texttt{\#inv}(\pi)$. Each skip erases a single arc, each hop erases a pair of feasible arcs. Define $M' \subseteq E^a_\pi$ as follows:

$$M' = \{((a, b), (c, d)) \mid \text{ some correcting hop in } \rho \text{ erases both } (a, b) \text{ and } (c, d)\}$$

Clearly, M' is a matching of size h. The arcs left unmatched by M' are precisely the arcs erased by correcting skips, so $|U'| = s$. Thus the length of the sequence, $s + h$, is precisely $|M'| + |U'|$.

Let M be a maximum matching in G^a_π, and let U be the vertices unsaturated by it. Then $2|M| + |U| = |A_\pi| = 2|M'| + |U'|$. Since $|M| \geq |M'|$, it follows that $csh(\pi) = s + h = |M'| + |U'| = 2|M'| + |U'| - |M'| = 2|M| + |U| - |M'| = (|M| + |U|) + (|M| - |M'|) \geq |M| + |U|$. $\square$

We can now establish a lower bound on $cs(\pi)$ when $\pi \in G_n$.

Lemma 4.13 *$\forall \pi \in G_n$, $cs(\pi) \geq \max\{|p(\pi)|_1,\ |q(\pi)|_1\}$.*

Proof: Since G^a_π has exactly $\texttt{\#inv}(\pi)$ vertices, it follows from Lemma 4.12 and Equation 2 that for every $\pi \in S_n$, $cs(\pi) \geq |U|$ where U is the vertices left unmatched by by a maximum matching M in G^a_π. It thus suffices to prove that when π is in G_n, for any matching M in G^a_π, $|U| \geq |p(\pi)|_1$ and $|U| \geq |q(\pi)|_1$.We present the argument for $|U| \geq |p(\pi)|_1$, that for $|q(\pi)|_1$ is symmetric.

No correcting hop/skip involves elements from more than one connected component of the underlying permutation. We say that a sequence $\langle a_1, \ldots, a_n \rangle$ is isomorphic to a permutation $\pi = \pi_1 \ldots \pi_n$ if there is a constant k such that for every $1 \leq i \leq n$, $\pi_i = a_i - k$. Then a permutation π is in G_n if and only if each of its connected components is isomorphic to a permutation in CG_m, for some $m \leq n$. If a component of π is isomorphic to some permutation σ, then $p(\sigma)$ is exactly the corresponding substring of $p(\pi)$. Note that connected components of π correspond to connected components of the permutation graph G_π. Hence it suffices to prove that for each $\pi \in CG_n$, $|U| \geq |p(\pi)|_1$.

If π is in CG_n, then by Theorem 4.7, it has the IES, IOS and ODN2E properties. Let (π_i, π_j) be an arc in G_π. By the above properties, i must be odd and j must be even. If (π_i, π_k) and (π_j, π_k) are arcs in G_π, then every (π_l, π_k) where l is an odd number between i and j must also be in G_π. Thus all the incoming arcs into an even position are from consecutive odd positions. Note that $\texttt{left}(\pi_i, \pi) = \{\pi_j \mid (\pi_j, \pi_i) \text{ is an arc in } G_\pi\}$. Hence $p(\pi_i, \pi) = 0$ if i is odd, and $p(\pi_i, \pi) = \text{indegree}(\pi_i) \bmod 2$ if i is even. Also, all compatible pairs are of the form $((\pi_i, \pi_j), (\pi_{i+2}, \pi_j))$ or $((\pi_j, \pi_i), (\pi_j, \pi_{i+2}))$.

We show that there is a maximum matching in G^a_π in which every matched pair of arcs is right-compatible. Let M be any matching in G^a_π. If no arcs matched by M are left-compatible, we are done. Otherwise, choose the largest j such that M contains a left-compatible pair with left-endpoint π_j. Now choose the smallest $k > j$ such that the pair $(\pi_j, \pi_k), (\pi_j, \pi_{k+2})$ is in M. Note that j must be odd and k must be even. Let $B = \{(\pi_l, \pi_k) \mid l \text{ odd}, j < l < k\}$ and let $C = \{(\pi_l, \pi_k) \mid l \text{ odd}, j < l < k+2\}$. As argued above, $B \cup C \subseteq E_\pi$. Exactly one of B and C is odd. Assume that B is odd, the argument when C is odd is similar. Each arc in B is a vertex in G^a_π. If it is matched by M, then it must be matched with a right-compatible arc, which is also in B, since (by our choice of j) no arcs in M are left-compatible beyond j. So we can now restructure M as follows: remove from M the left-compatible pair $(\pi_j, \pi_k), (\pi_j, \pi_{k+2})$ and all arc pairs within B (at most $(|B| - 1)/2$; add to M the following $(|B| + 1)/2$ feasible arc pairs $((\pi_{j+4t}, \pi_k), (\pi_{j+4t+2}, \pi_k))$ for $0 \leq t \leq (k - j - 3)/4$. (By ODN2E, all these pairs are indeed feasible.) This gives a new matching of the same size as M and with one less left-compatible pair. Repeat this process until there are no left-compatible pairs left.

Now let M be a maximum matching in G^a_π where no matched arcs are left-compatible. For each even position i, M can pick at most $\lfloor \text{indegree}(\pi_i)/2 \rfloor$ feasible pairs of arcs right-compatible at π_{2i}. Thus, if indegree(π_i) is odd (i.e. $p(\pi_i, \pi) = 1$), then at least one vertex of G^a_π of the form (π_k, π_i) is left unsaturated by M. Hence $|U| \geq |p(\pi)|_1$, proving the theorem. $\square$

From Lemmas 4.10 and 4.13, we have the following characterization.

Theorem 4.14 *$\forall \pi \in G_n$, $cs(\pi) = |p(\pi)|_1 = |q(\pi)|_1$.*

From Lemma 4.10 and Theorem 4.14, we have the following.

Corollary 4.15 *Each permutation $\pi \in G_n$ has $cs(\pi)$ correcting skips applying which in parallel results in a permutation in H_n.*

From Theorem 4.3, we see that testing if π is in G_n can be done in $O(n)$ time. If $\pi \in G_n$, then by Lemma 4.5, computing $|p(\pi)|_1$ can also be done in $O(n)$ time. With this information, a sorting sequence of length $|p(\pi)|_1$ can be described in additional $O(n)$ time as follows: First apply $|p(\pi)|_1$ correcting skips at appropriate positions, as described in the proof of Lemma 4.10, to obtain a permutation in L_n. Then, for a going from 1 to $n-1$, apply $\lfloor |\texttt{left}(a, \pi)|/2 \rfloor$ left hops to a; the proof of Lemma 3.4 shows that all these hops are correcting. From Theorem 4.14, this sequence is optimal. We thus have the following result. (Note that $csh(\pi)$ itself need not be $O(n)$; but the $csh(\pi)$ correcting hops/skips have an $O(n)$ sized description.)

Theorem 4.16 *Given a permutation $\pi \in G_n$, we can find an optimal sorting sequence of correcting hops / skips in $O(n)$ time.*

5 Improved Upper Bounds for General Permutations

In this section, we extend the techniques developed to obtain some results for general permutations. We first obtain an upper bound for $cs(\pi)$ based on the strings $p(\pi)$ and $q(\pi)$. We then use this bound, along with the combinatorial characterization of CG_n, to present a tighter analysis of the 4/3-approximation algorithm for $csh(\pi)$ (Heath and Vergara, (1)). Our results, in particular, simplify the above mentioned algorithm.

5.1 An Upper Bound for $cs(\pi)$

Using the techniques developed in Section 3, we obtain an upper bound for $cs(\pi)$ based on the strings $p(\pi)$ and $q(\pi)$. Let s be a bit string $s \in \{0,1\}^n$. A 1-block of s is a maximal subtring of 1s. An odd (even) 1-block is a 1-block of odd (even, respectively) length. For instance, if $s = 001101110100011$, then s has four 1-blocks, of which the first and last are even and the other two are odd.

Lemma 5.1 *For $\pi \in S_n$, let m_1 denote the number of odd 1-blocks in $p(\pi)$, and let m_2 denote the number of odd 1-blocks in $q(\pi)$. Then $cs(\pi) \leq \min(m_1, m_2)$.*

Proof: We prove that $cs(\pi) \leq m_1$; the upper bound of m_2 is proved in a symmetric way.

Note that $p(\pi_1)$ is always 0. Let $j \geq 1$ be the smallest index such that $p(\pi_{j+1}) = 1$, and let $k \geq j$ denote the smallest index such that $p(\pi_{k+1}) = 0$. (That is, the first 1-block in $p(\pi)$ occurs in positions $j+1, \ldots, k$.) As argued in the proof of Lemma 3.4, the portion $\pi_1 ... \pi_j$ can be sorted with correcting left hops alone. Let us denote the permutation that results from sorting this prefix by σ. Then $p(\sigma) = p(\pi)$. Since $p(\sigma_{j+1}, \sigma) = 1$, it holds that $\texttt{left}(\sigma_{j+1}, \sigma) \neq \emptyset$. But σ_j is the largest element to the left of σ_{j+1}, so $\sigma_{j+1} < \sigma_j$.

We argue inductively that this holds for every element in this 1-block; i.e. $\sigma_l < \sigma_j$ for each $j+1 \leq l \leq k$. For $l = j+1$ we have shown this above. Now assume that this is true for $j+1 \leq i < l$, and now consider l. Since $p(\sigma_l, \sigma) = 1$, $\texttt{left}(\sigma_l, \sigma) \neq \emptyset$. If for some $t \in [j+1, l-1]$, $\sigma_t > \sigma_l$, then $\sigma_l < \sigma_t < \sigma_j$ by induction. Otherwise $\texttt{left}(\sigma_l, \sigma)$ contains no elements in the positions $j+1, \ldots, l-1$. Since it is non-empty, it must contain an element in position $1, \ldots, j$. But σ_j is the largest such element, so $\sigma_j > \sigma_l$.

Thus σ_j is larger than all the elements in the 1-block beginning at position $j+1$. We move σ_j to the right of this block using only correcting hops if the 1-block is even, and using correcting hops and one correcting skip if the 1-block is odd. Let ρ be the resulting permutation. From Table 1, we can see that $p(\rho)$ differs from $p(\sigma)$ in that the first 1-block is replaced by 0s. (A correcting right move flips the underlying bits.)

The argument can be repeated to sequentially remove each of the remaining 1-blocks. One correcting skip is required for every odd 1-block encountered. So all 1-blocks are removed using exactly m_1 correcting skips. This gives a permutation in $p^{-1}(0^n)$, which, by Lemma 3.4, is in L_n. $\square$

Since the maximum number of odd 1-blocks that can be packed into an n-bit string where the first (or last) bit is 0 is $\lfloor n/2 \rfloor$, we have the following upper bound.

Theorem 5.2 *For any $\pi \in S_n$, $cs(\pi) \leq \left\lfloor \frac{n}{2} \right\rfloor$ and $csh(\pi) \leq \frac{\texttt{\#inv}(\pi) + \lfloor \frac{n}{2} \rfloor}{2}$.*

Note that Theorem 18 of (7) establishes another upper bound; it shows that for all $\pi \in S_n$, $csh(\pi) \leq \lceil \binom{n}{2}/2 \rceil$. Our bound is incomparable with this; for some permutations our bound is tighter whereas for some it is weaker. For instance, for the reversal permutation $\texttt{rev}_n = n\ n-1 ... 2\ 1$, $p(\texttt{rev}_n)$ has $\lfloor \frac{n}{2} \rfloor$ odd blocks of 1s, so our bound shows that $\lfloor n/2 \rfloor$ correcting skips suffice and $csh(\texttt{rev}_n) \leq (\binom{n}{2} + \lfloor \frac{n}{2} \rfloor)/2$. But the bound of Heath and Vergara gives $csh(\texttt{rev}_n) \leq \lceil \binom{n}{2}/2 \rceil$ (in fact, Heath and Vergara establish that $csh(\texttt{rev}_n) = \lceil \binom{n}{2}/2 \rceil$), and since $\texttt{\#inv}(\texttt{rev}_n) = n(n-1)/2$, at most one correcting skip is required. On the other hand, for permutations $\pi \in G_n$, due to the IES and IOS properties, we have $\texttt{\#inv}(\pi) \leq \binom{n}{2} - 2\binom{n/2}{2}$, so our bound is significantly better. Essentially, our bound is better whenever $\binom{n}{2} - \texttt{\#inv}(\pi)$ exceeds $n/2$.

Corollary 5.3 *For any $\pi \in S_n$, if either $p(\pi)$ or $q(\pi)$ has only even-length blocks of 1s, then π is in H_n.*

5.2 A Tighter Analysis of the 4/3 Approximation Algorithm of (1)

In (1), Heath and Vergara present a greedy algorithm for sorting by correcting hops / skips, and show that it achieves an approximation factor of 4/3. To describe the algorithm, we first present some notation, following (1).

Let π_i, π_{i+1} form a component of π. Then it must hold that $\pi_i = i + 1$ and $\pi_{i+1} = i$. To sort π, a correcting skip exchanging π_i and π_{i+1} is essential. Such a skip, that sorts a component of size 2, is called a lone skip. For every permutation π, $cs(\pi)$ is at least as large as the number of lone skips already present in π. However, an algorithm to sort π may also introduce lone skips; since the goal is to minimize skips, a sub-goal is to minimize such introductions.

A mushroom in π is a substring of contiguous elements $a\ b\ c\ d$ such that $b < d < a < c$. A correcting skip exchanging a and b, followed by a left correcting hop of d, sorts the mushroom.

The following points are proved in (1): (1) If a permutation has correcting hops, then there are correcting hops introducing zero or one new lone skip, (2) If an unsorted permutation has no lone skip and no correcting hop, then it has a mushroom, and (3) Performing a lone skip, or sorting a mushroom with one skip and one hop as described above, does not introduce lone skips. Using these, the greedy algorithm GREEDYHOPS is formulated as follows.

1. While there are lone skips, perform lone skips.
2. While there are correcting hops, perform a correcting hop that introduces the fewest number of lone skips and go to step 1.
3. If unsorted, locate a mushroom, sort it with one skip and one hop, and go to step 1.

Algorithm GREEDYHOPS-1

Each skip used in this algorithm is either essential (a pre-existing lone skip), or introduced by a correcting hop at step 2, or followed by a correcting hop at step 3. So the non-essential skips can be matched with correcting hops, giving the 4/3 factor.

It is easy to see that performing the lone skips can be postponed to the end of the algorithm, since each lone skip acts on a component of size 2 and does not affect the rest of the permutation. Thus, the algorithm can be restated as:

1′. While there are correcting hops, perform a correcting hop that introduces the fewest number of lone skips.

2′. If there is a mushroom, pick any one mushroom, sort it with one skip and one hop, and go to step 1′.

3′. While there are lone skips, perform lone skips.

Algorithm GREEDYHOPS-2

(Algorithms GREEDYHOPS-1 and GREEDYHOPS-2 require the same number of skips and hops; only the order in which lone skips are performed is changed.)

We use our understanding of G_n and CG_n to tighten the analysis of this algorithm. Notice that step $2'$ is executed only on G_n, correcting-hop-free permutations. Since a mushroom is necessarily within a single component, step $2'$ acts on a connected correcting-hop-free permutation, in CG_k for some k. If step $2'$ is executed again on this part of the permutation, then just before this execution, this portion of the permutation is in G_k. And in the intervening sequence, exactly one skip and some hops have been performed on this part.

We show below that if $\pi \in CG_n$, then any correcting skip applied to π, followed by any sequence of correcting hops leading to a permutation in G_n, will lead to the same permutation, say σ, in G_n. Recall that from Theorem 4.7, n must be even, say $2m$, and π has the IES, IOS and ODN2E properties.

Lemma 5.4 *For $m > 1$, let $\pi \in CG_{2m}$. Denote by o_i and e_i the ith odd and even position elements respectively; then $\pi = o_1e_1o_2e_2 \dots o_me_m$. If ϕ is obtained from π through a sequence of one correcting skip followed by zero or more correcting hops, and if $\phi \in G_{2m}$, then $\phi = e_1e_2o_1e_3o_2e_4 \dots o_{m-2}e_mo_{m-1}o_m$; therefore $|p(\phi)|_1 = |p(\pi)|_1 - 1$. Furthermore, ϕ is reached from π by one correcting skip and exactly $m-1$ correcting hops.*

Before proving this lemma, we show how it helps us to analyse GREEDYHOPS-2.

Lemma 5.5 *Algorithms GREEDYHOPS -1 and GREEDYHOPS -2 perform at most $\lfloor n/2 \rfloor$ correcting skips.*

Proof: Consider a sorting sequence constructed by GREEDYHOPS-2. Let it be of length $acsh(\pi)$ (approximate csh), with acs correcting skips and ach correcting hops. Since correcting skips are performed only when the permutation is in G_n, and the skip affects only the component involved, Lemma 5.4 tells us that from one skip to the next, the number of 1s in $p(.)$ decreases exactly by 1. Thus, if σ is the permutation just before the application of the first correcting skip, and if $|p(\sigma)|_1 = k$, then $acs = k$. But $\sigma \in G_n$, so $|p(\sigma)|_1 \leq \lfloor n/2 \rfloor$. Hence $acs \leq \lfloor n/2 \rfloor$. □

Theorem 5.6 *On any permutation $\pi \in S_n$, algorithm GREEDYHOPS of (1) produces a sorting sequence of length at most*

$$\min\left\{4/3, \left(1 + \frac{n}{2 \cdot \texttt{\#inv}(\pi)}\right)\right\} \cdot csh(\pi)$$

Proof: The 4/3 bound is established in (1). To see the other bound, note that

$$\begin{aligned} acsh(\pi) &= \frac{\texttt{\#inv}(\pi) + acs}{2} \\ &\leq \frac{\texttt{\#inv}(\pi) + \lfloor \frac{n}{2} \rfloor}{2} \\ &\leq \left(\frac{\texttt{\#inv}(\pi) + \frac{n}{2}}{\texttt{\#inv}(\pi)}\right) \cdot \left(\frac{\texttt{\#inv}(\pi)}{2}\right) \\ &\leq \left(1 + \frac{n}{2 \cdot \texttt{\#inv}(\pi)}\right) \cdot csh(\pi) \end{aligned}$$

□

Note that whenever $\texttt{\#inv}(\pi) > 3n/2$, we have $4/3 > 1 + \frac{n}{2 \cdot \texttt{\#inv}(\pi)}$ and so we improve the approximation factor. In fact, when $\texttt{\#inv}(\pi) \in \theta(n^2)$, the approximation factor achieved is $O(1 + 1/n)$, very close to 1. When $\texttt{\#inv}(\pi) \leq 3n/2$, the permutation is, in some sense, already close to being sorted. An algorithm which works well for such permutations can be coupled with algorithm GREEDYHOPS for the remaining permutations to obtain a better approximation factor. Thus, to improve the approximation factor from 4/3 to 5/4, say, it suffices to devise an algorithm that achieves this factor when $\texttt{\#inv}(\pi) \leq 2n$.

Another noteworthy point is that when GREEDYHOPS-2 first uses a correcting skip, the permutation in hand is in G_n. But we know how to sort such permutations optimally. So an alternative greedy algorithm, with approximation factor at least as good as GREEDYHOPS-1 or GREEDYHOPS-2, is as follows:

---

(1) While there are correcting hops, perform a correcting hop that introduces the fewest number of lone skips.
(2) The permutation is now in G_n. Sort it optimally according to Theorem 4.16.

Algorithm GREEDYHOPS-3

We now prove Lemma 5.4.

Proof: (Of Lemma 5.4)

Let $\pi = o_1e_1o_2e_2 \dots o_me_m \in CG_{2m}$. By Theorem 4.7, we have $e_{i-1} < e_i < o_{i-1} < o_i$ for every $1 < i \leq m$. We want to show that after any one correcting skip, followed by enough correcting hops to re-enter G_n, the resulting permutation is sorted if $m = 2$, and if $m > 2$, then the smallest two elements (which are from the even subsequence) move to the extreme left, the two largest elements (which are from the odd subsequence) move to the extreme right, and the remaining elements of the odd and even subsequences appear in a perfect shuffle in the middle.

We consider three cases depending on where the first correcting skip is applied.

Case 1: *The correcting skip applied on π exchanges o_1 and e_1.* This gives the permutation $e_1o_1o_2e_2 \dots o_me_m$ which has exactly one correcting hop: move e_2 left. By a simple induction, we can show that after the first skip and i correcting hops, the only correcting hop available is: move e_{i+3} left. Thus to reach G_n again, we must perform $m-1$ hops, resulting in the permutation $e_1e_2o_1e_3o_2e_5 \dots o_{m-2}e_mo_{m-1}e_{m-1}$.

Case 2: *The correcting skip applied on π exchanges (o_m, e_m).* This case is similar to Case 1.

Case 3: *The correcting skip exchanges o_i, e_i for some $1 < i < m$.* This gives rise to a permutation $\sigma = o_1e_1 \ \dots \ o_{i-1}e_{i-1} \ e_io_i \ o_{i+1}e_{i+1} \ \dots \ o_me_m$.

We now establish by induction that after performing k correcting hops on σ, the permutation is of the form $(oe)^*\ ee\ (oe)^{k-1}\ oo\ (oe)^*$, with the elements from the original even (odd) subsequence in correct relative order.

To see the basis, when $k = 1$, note that σ has two correcting hops: move o_{i-1} right, and move e_{i+1} left. Applying the right hop of o_{i-1} on σ gives permutation $\sigma^1 = o_1e_1\ \ldots\ o_{i-2}e_{i-2}\ e_{i-1}e_i\ o_{i-1}o_i\ o_{i+1}e_{i+1}\ \ldots\ o_me_m$, and now applying the left hop of e_{i+1} on σ gives permutation $\sigma^2 = o_1e_1\ \ldots\ o_{i-1}e_{i-1}\ e_ie_{i+1}\ o_io_{i+1}\ o_{i+2}e_{i+2}\ \ldots\ o_me_m$. Thus both σ^1 and σ^2 are of the form $(oe)^*\ ee\ oo\ (oe)^*$, with the elements from the original even (odd) subsequence in correct relative order.

Assume that the claim is true after $k-1$ correcting hops, so we have a permutation ρ of the form $(oe)^*\ ee\ (oe)^{k-2}\ oo\ (oe)^*$. The elements marked o are in correct relative order, so are the elements marked e, and $o_{i-1} > e_i$. If $k = m$, then $\rho = ee\ (oe)^{k-2}\ oo$, and from Theorem 4.3 we can see that there are no more correcting hops available. If $k < m$, then let $\rho = (oe)^p\ ee\ (oe)^{k-2}\ oo\ (oe)^q$, where $p + k + q = m$. If $p = 0$, there is only one correcting hop, transforming the suffix $oo\ (oe)^q$ to $oe\ oo\ (oe)^{q-1}$. The resulting permutation is of the form $(oe)^p\ ee\ (oe)^{k-2}\ oe\ oo\ (oe)^{q-1}$, satisfying the claim. If $q = 0$, the argument is symmetric. If neither is zero, the kth hop could either transform the suffix $oo\ (oe)^q$ to $oe\ oo\ (oe)^{q-1}$ or transform the prefix $(oe)^p\ ee$ to $(oe)^{p-1}\ ee\ oe$. In either case, the resulting permutation has the form $\rho = (oe)^{p'}\ ee\ (oe)^{k-1}\ oo\ (oe)^{q'}$, satisfying the claim.

Thus to reach G_n, we must perform $m-1$ hops, after which the resulting permutation is as claimed.

□

6 The Class Two_n

From Lemma 3.3 we know that for permutations in H_n, $|p(\pi)|_1$ and $|q(\pi)|_1$ are both even. The subclass of H_n where either of these numbers is 0 is well-understood (the classes L_n and R_n). Thus, a first non-trivial extension of $L_n \cup R_n$ within H_n may be considered to be permutations with two 1s in $p(\pi)$ or in $q(\pi)$. In this section, we explore such permutations. Let $LTwo_n = \{\pi \in S_n \mid |p(\pi)|_1 = 2\}$, $RTwo_n = \{\pi \in S_n \mid |q(\pi)|_1 = 2\}$, and $Two_n = LTwo_n \cup RTwo_n$. We attempt to understand $Two_n \cap H_n$. While we cannot yet characterize this intersection, we identify fairly large subsets of Two_n which lie in H_n. Finally, we sketch a procedure to extend these ideas to permutations in S_n.

We will prove results for $LTwo_n \cap H_n$; analogous results hold for $RTwo_n \cap H_n$. Let π be a permutation in $LTwo_n$. We identify sufficient properties for π to be in H_n. The first sufficient property follows from Corollary 5.3.

Lemma 6.1 *For $\pi \in LTwo_n$, if the two 1s in $p(\pi)$ are adjacent, then $\pi \in H_n$.*

Lemma 6.2 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$. If $\pi_i > \pi_j$, then $\pi \in H_n$.*

Proof: If $j = i + 1$, then the result follows from Lemma 6.1. Otherwise, let $\pi_i = a$ and $\pi_i = b$. As in the proof of Lemma 3.4, the elements of π to the left of a can be sorted via correcting left hops. This brings $c = max\{\pi_l \mid l < i\}$ to the immediate left of a, with $c > a$, and the $p(.)$ string is unaltered. Now, scan the string between a and b from left to right. Whenever an element $d < a$ is encountered, move it left via correcting hops as long as possible. Since all elements between a and d are greater than a (by the time we reach d, we have already moved out smaller elements), and since $d < a < c$, these hops will take d to the left of a (and possibly to the left of c as well). When we reach b, which by assumption is less than a, we move it left via correcting hops until it is adjacent to a (it may be to the left or right of a). By Lemma 6.1, this permutation is in H_n, and to reach this permutation from π we used only correcting hops. Hence $\pi \in H_n$. □

Lemma 6.3 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$. If $i < j-1$ and $\pi_{j+1} < \pi_i < \pi_j < \pi_{j-1}$, or if $i < j - 2$ and $\pi_{j-1} < \pi_i < \pi_j < \pi_{j-2}$, then $\pi \in H_n$.*

Proof: If $\pi_i > \pi_j$, then Lemma 6.2 gives the result. Otherwise, in the first case, perform a right hop of π_{j-1} to obtain permutation ρ. The bits of $p(\pi)$ at positions $j-1, j, j+1$ remain as 010, with the 1 at $\rho_j = \pi_{j+1} < \pi_i = \rho_i$. In the second case, perform a right hop of π_{j-2} to obtain permutation σ. The bits of $p(\pi)$ at positions $j-2, j-1, j$ change from 001 to 100 in $p(\sigma)$, with the 1 at $\sigma_{j-2} = \pi_{j-1} < \pi_i = \sigma_i$.

In either case, after this one right hop, Lemma 6.2 is applicable. □

Lemma 6.4 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$. If $i < j-3$ and there exists a $i+1 < k < j-1$ such that $\pi_k < \pi_i < \pi_j < \pi_{k+1} < \pi_{k-1}$, then $\pi \in H_n$.*

Proof: The substring $\pi_{k-1}, \pi_k, \pi_{k+1}$ lies between π_i and π_j. Perform a right hop of π_{k-1} to get permutation σ. This creates two new 1s in $p(\sigma)$ (which were 0s in $p(\pi)$); these 1s are at $\sigma_{k-1} = \pi_k$ and $\sigma_k = \pi_{k+1}$.

Now the prefix of σ up to position $k-1$ can be sorted using only hops, according to the strategy of Lemma 6.2, since there are two 1s at i and $k-1$ and $\sigma_i > \sigma_{k-1}$. The resulting permutation ρ again satisfies the premise of Lemma 6.2 and hence is in H_n. □

Lemma 6.5 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$, and let*

$c = max\{\pi_l \mid l < i\}$. If $\pi_l < c$ for $i \leq l \leq j$, and if any of the following conditions holds, then $\pi \in H_n$.

(i) $j - i$ is odd.
(ii) $\pi_l < \pi_i$ for some $l > i$.
(iii) Some element smaller than π_j occurs to the right of π_j; the first such element occurs at position $l > j$ where $l - j - 1$ is odd.

Proof: If $j = i + 1$ or if $\pi_j < \pi_i$, then the result follows from Lemma 6.1 or Lemma 6.2.

Otherwise, as in the previous proof, let $\pi_i = a$ and $\pi_j = b$. Sort the prefix of π to the left of a using correcting left hops. This brings c to the immediate left of a.

First assume that condition (i) holds. Since $j - i$ is odd, the substring $\pi_i \ldots \pi_j$ is of even length. Since furthermore, all elements between a and b are smaller than c, c can be moved to the immediate right of b using correcting right hops. From Table 1, we see that this changes the $p(.)$ sequence only in the positions $i-1, i, \ldots, j$, from $010^{j-i-1}1$ to $01^{j-i-1}00$. Since $j-i-1$ is even, by Corollary 5.3 the resulting permutation is in H_n. Since no skips were needed to get to H_n, it follows that $\pi \in H_n$.

If condition (i) does not hold but condition (ii) holds, then $j - i$ is even and for some $l > i$, $\pi_l < a$.

Case 1: Such an index appears between i and j. Let $d = \min\{\pi_l < a \mid i < l < j\}$. Then d can be moved left via correcting hops until it is to the left of a (and possibly to the left of c as well). This gives a permutation in $LTwo_n$ satisfying condition (i) above, and so it is in H_n.

Case 2: All elements between a and b are greater than a (but less than c). For some $l > j$, $\pi_l < a$. Let $d = \min\{\pi_l < a \mid j < l\}$. Then d can be moved left via correcting hops until it is between a and b. This too gives a permutation in $LTwo_n$ satisfying condition (i) above, and so it is in H_n.

If neither condition (i) nor condition (ii) holds but condition (iii) holds, then let $t = \pi_l$. By performing correcting left hops, we can bring t to the left of b but to the right of a. This gives a permutation in $LTwo_n$ satisfying condition (i) above, and so it is in H_n. □

Lemma 6.6 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$. If there exist indices $k < l < i$ such that $\pi_k > \pi_l > \pi_j$ and $\pi_s < \pi_j$ for all $s \in [k+1, i-1] \setminus \{l\}$, then $\pi \in H_n$.*

Proof: If $\pi_i > \pi_j$, then by Lemma 6.2 $\pi \in H_n$. Otherwise, let $a = \pi_i$, $b = \pi_j$, $c = \pi_l$ and $d = \pi_k$. The permutation π is of the form $\alpha\ d\ \beta\ c\ \gamma\ a\ \delta\ b\ \kappa$ where

$a < b < c < d$, and all elements of β and γ are less than b.

First, as above, sort α using correcting left hops.

Now scan the elements of β in increasing order and move them left via correcting left hops as far as possible. When an element r is scanned, all elements between d and r are greater than r (by the time we reach r, we have already moved out smaller elements). If r goes to the left of d via correcting hops, we now re-sort the prefix to the left of d. If r stops to the immediate right of d, then it means that r exceeds the largest element before d (because we maintain the property that the prefix to the left of d is sorted). But this means that $\texttt{left}(r, \sigma) = \{d\}$ and $p(r, \sigma) = 1$ where σ is the current permutation. But in obtaining σ from π we have used only correcting left hops, so $p(x, \sigma) = p(x, \pi)$ for all $x \in [n]$. And $p(r, \pi) = 0$. So r will go to the left of d.

When we finish scanning β, we have the permutation $\beta'\ d\ c\ \gamma\ a\ \delta\ b\ \kappa$ where β' is the merged sorted sequence of elements from α and β.

In a similar procedure, we can move γ to the left of d to get $\gamma'\ d\ c\ a\ \delta\ b\ \kappa$ where γ' is the merged sorted sequence of elements from β' and γ. At this stage, the $p(.)$ sequence is still $p(\pi)$, since only correcting left hops have been used.

Now we perform a correcting right hop of d. This gives the permutation $\rho = \gamma'\ c\ a\ d\ \delta\ b\ \kappa$. Since d hops over c and a, by Table 1 we see that $p(c, \rho)$ and $p(a, \rho)$ have flipped (compared to their values in π). So in $p(\rho)$, there will be exactly two 1s, at c and b, with $c > b$ and c to the left of b. By Lemma 6.2, $\rho \in H_n$. Since ρ was obtained from π via correcting hops alone, $\pi \in H_n$. $\square$

Lemma 6.7 *For $\pi \in LTwo_n$, let $p(\pi_i, \pi) = p(\pi_j, \pi) = 1$ for $i < j$. If there exist indices $k < i$ and $i < l < j$ such that $\pi_k > \pi_l > \pi_j$, $\pi_s < \pi_j$ for all $s \in [k+1, l-1] \setminus \{i\}$, and $l - 1 - i$ is even, then $\pi \in H_n$.*

Proof: Let $\pi_i = a$, $\pi_j = b$, $\pi_l = c$ and $\pi_k = d$. If $a > b$, then by Lemma 6.2 $\pi \in H_n$. Otherwise, we have $a < b < c < d$ and their relative order in π is d, a, c, b. As in the preceding proof, all elements between d and a can be moved left of d, via left correcting hops. Let this permutation have the form $\alpha\ d\ a\ \beta\ c\ \gamma\ b\ \delta$, where $|\beta| = l - i - 1$ is even and all elements of β are less than b. Now we perform right hops of d until it goes to the right of c, giving $\alpha\ a\ \beta\ c\ d\ \gamma\ b\ \delta$. The string $p()$ flips from $010^{|\beta|}0$ in the $d\ a\ \beta\ c$ section to $01^{|\beta|}10$ in the $a\ \beta\ c\ d$ section. The prefix $\alpha\ a\ \beta$ has bit string $p()$ of the form $0^*1^{|\beta|}$, and since $|\beta|$ is even, it can be sorted using correcting hops by Corollary 5.3. This gives another permutation in $LTWO_n$, with the two 1s at c and b. Now Lemma 6.2 is applicable to this permutation to sort it using correcting hops alone. $\square$

The results of this section identify some subsets of $LTwo_n$ which are also in H_n. Unfortunately, they do not constitute a recognizer for $LTwo_n \cap H_n$. However, they are still of some use: for many permutations of S_n, we can show membership in H_n by carefully pairing the 1s in their p-strings in a balanced parenthesis structure, and by repeatedly locally applying one of the above results to an innermost pair of 1s (as in Lemma 6.4). Thus these results may be useful in designing a heuristic for approximate optimal sorting.

7 Sorting with Correcting Moves

In this section, we show how the problem of sorting by skips and hops can be rephrased as the problem of sorting by special types of correcting moves. We also give a simple procedure for optimally sorting permutations by correcting moves.

Recall that a move picks up a single element and places it elsewhere in the string. The length of a move may be defined as the number of elements past which the moving element goes. Based on its length, a move can be labelled either odd or even. Let $m(\pi)$ and $cm(\pi)$ denote the minimum number of moves and the minimum number of correcting moves, respectively, required to sort $\pi \in S_n$. Let $ocm(\pi)$ denote the minimum number of odd correcting moves, over all sequences of correcting moves sorting π. The following result is established in (7):

Theorem 7.1 (Theorem 1 (7).) *For all $\pi \in S_n$, $m(\pi) = n - |LIS(\pi)|$, where $|LIS(\pi)|$ is the length of a longest increasing subsequence of π.*

Thus $m(\pi)$ can be computed efficiently (standard algorithmic techniques using dynamic programming can be used to compute $|LIS(\pi)|$). We now consider computing $cm(\pi)$ and $ocm(\pi)$. We show that $cm(\pi)$ can be computed exactly, while computing $ocm(\pi)$ is equivalent to computing $cs(\pi)$.

Theorem 7.2 *For all $\pi \in S_n$, $cm(\pi) = m(\pi)$.*

Proof: By definition, $m(\pi) \leq cm(\pi)$. It thus suffices to show, from Theorem 7.1, that $cm(\pi) \leq n - |LIS(\pi)|$, where $|LIS(\pi)|$ is the length of any longest increasing subsequence of π. We describe a sequence of $n - |LIS(\pi)|$ correcting moves sorting π.

Let $l = |LIS(\pi)|$, and let $\pi_{i_1} ... \pi_{i_l}$ be a longest increasing subsequence LIS of π. Assume for notational convenience that $\pi_0 = \pi_{i_0} = 0$ and $\pi_{n+1} = \pi_{i_{l+1}} = n+1$. For each $j = 0, 1, \ldots, l$, and for $i_j < k < i_{j+1}$, we have $\pi_k < \pi_{i_j}$ or $\pi_k > \pi_{i_{j+1}}$, since otherwise including π_k gives a longer increasing subsequence. Thus the

LIS partitions S into $l+1$ intervals, each of which is stuffed with the wrong members. Ignore the presence of these wrong entries and concentrate instead on the entries which are missing in each of these intervals. Do a forward pass over these $l+1$ intervals, starting from the leftmost, bringing and placing in order all the missing elements which are to the right of the interval. Next do a backward pass, starting from the rightmost interval, bringing and placing in order all the missing elements in order, in order also with the elements already placed inside the interval in the forward pass.

Formally, the sorting procedure may be described as follows:

```
For j = 0 to l                                    /* forward pass*/
    For k = π_{i_j} + 1 to π_{i_{j+1}} − 1
          If k occurs to the right of π_{i_{j+1}},
          then move it as far left as a correcting move will allow.
For j = l down to 1                               /* backward pass*/
    For k = π_{i_{j+1}} − 1 down to π_{i_j} + 1
          If k occurs to the left of π_{i_j},
          then move it as far right as a correcting move will allow.
```

Given the order in which elements are processed, this procedure will end in $\mathtt{id}_n$. Since each element not in the LIS is moved exactly once (in the forward pass if it is to the right of its target interval, in the backward pass otherwise), the number of correcting moves is exactly $n-l$. $\square$

Theorem 7.3 *$ocm(\pi) = cs(\pi)$.*

Proof: Every sequence of s correcting skips and h correcting hops is a sequence of correcting moves as well, in which s moves are odd. Hence $ocm(\pi) \le s$. Since there is a sorting sequence with $cs(\pi)$ correcting skips, we have $ocm(\pi) \le cs(\pi)$.

Each correcting move can be broken up into a sequence of correcting hops, followed by one last correcting skip if the move is of odd length. Thus from a correcting move sequence with $ocm(\pi)$ odd moves, we can extract a correcting hop/skip sequence with exactly $ocm(\pi)$ skips. Hence $cs(\pi) \le ocm(\pi)$. $\square$

Theorems 7.2 and 7.3 suggest two natural variants of computing $ocm(\pi)$: (1.) Minimize the number of odd correcting moves over shortest sequences of correcting moves; (2.) Minimize the number of odd correcting moves over those sequences of correcting moves which leave the elements of some longest increasing subsequence untouched. While these problems may be of some combinatorial interest, they are not algorithmically useful, since these restricted minima do not even provide a a provably good approximation for $ocm(\pi)$. (For permutations like $\pi = 4\ 5\ 1\ 2\ 3\ 9\ 10\ 6\ 7\ 8 \ldots n-1\ n\ n-4\ n-3\ n-2$, $ocm(\pi) = 0$ but the number of odd correcting moves under either of the above two restrictions grows linearly.)

8 Conclusion

The complexity of optimal sorting using 3-transpositions has been an open problem for a few years. This paper presents an upper bound which is incomparable with previously known bounds. It also presents a tighter analysis of the best-known approximation algorithm for this problem, due to Heath and Vergara. To obtain these results, a good combinatorial understanding is developed of how optimal sorting is achieved for several classes of permutations (L_n, R_n, G_n), and some sufficient conditions for optimal sorting in larger subclasses are presented.

Acknowledgment

The authors are indebted to anonymous referees, whose comments helped immensely in improving the presentation of this paper.

References

[1] L. S. Heath, J. P. C. Vergara, Sorting by short block moves, Algorithmica 28 (2000) 323–352.

[2] P. D. Palmer, L. A. Herbon, Plant mitochondrial dna evolves rapidly in structures, Journal of Molecular Evolution 28 (1988) 87–97.

[3] J. Kececioglu, D. Sankoff, Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement, Algorithmica 13 (1995) 180–210.

[4] V. Bafna, P. Pevzner, Sorting by transpositions, SIAM Journal on Discrete Mathematics 11 (1998) 224–240.

[5] D. A. Christie, A 3/2-approximation algorithm for sorting by reversals, in: Proceedings of Ninth ACM-SIAM Symposium on Discrete Algorithms, ACM, 1998, pp. 244–252.

[6] T. Hartman, A simpler 1.5 approximation algorithm for sorting by transpositions, in: Proceedings of 14th Annual Symposium on Combinatorial Pattern Matching, LNCS 2676, Springer-Verlag, 2003, pp. 156–169.

[7] L. S. Heath, J. P. C. Vergara, Sorting by bounded block moves, Discrete Applied Mathematics 88 (1998) 181–206.

[8] V. Bafna, P. Pevzner, Genome rearrangements and sorting by reversals., SIAM Journal on Computing 25 (1996) 272–289.

[9] J. P. C. Vergara, Sorting by bounded permutations, Ph.D. thesis, Virginia Polytechnic Institute and State University (1997).

[10] D. Christie, Genome rearrangement problems, Ph.D. thesis, University of Glasgow (1998).

[11] N. Eriksen, $(1+\epsilon)$-approximation of sorting by reversals and transpositions, Theoretical Computer Science 289 (2002) 517–529.

[12] H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, J. Wästlund, Sorting a bridge hand., Discrete Mathematics 241 (2001) 289–300.

[13] P. Pevzner, Computational Molecular Biology: An Algorithmic Approach, MIT Press, Cambridge, MA, USA, 2000.