New Algorithms and Bounds for Halving Pseudolines

Sergey Bereg(✉) and Mohammadreza Haghpanah

University of Texas at Dallas, Richardson, TX 75080, USA
{besp,Mohammadreza.Haghpanah}@utdallas.edu

Abstract. Let P be a set of points in general position in the plane. A *halving line* of P is a line passing through two points of P and cutting the remaining $n-2$ points in a half (almost half if n is odd). Generalized configurations of points and their representations using allowable sequences are useful for bounding the number of halving lines.

We study a problem of finding generalized configurations of points maximizing the number of halving pseudolines. We develop algorithms for optimizing generalized configurations of points using the new notion of *partial allowable sequence* and the problem of computing a partial allowable sequence maximizing the number of k-transpositions. It can be viewed as a sorting problem using transpositions of adjacent elements and maximizing the number of transpositions at position k.

We show that this problem can be solved in $O(nk^n)$ time for any $k > 2$, and in $O(n^k)$ time for $k = 1, 2$. We develop an approach for optimizing allowable sequences. Using this approach, we find new bounds for halving pseudolines for even n, $n \leq 100$.

1 Introduction

Let S be a set of n points in the plane in general position. A *halving line* of S is a line passing through two points of S and

(i) cutting the remaining points in a half, if n is even, or
(ii) having $(n-1)/2$ and $(n-3)/2$ points of S on each side, if n is odd.

The problem of finding $h(n)$, the maximum number of halving lines for a set of n points, is one of the important open problems in the field of discrete geometry. Erdős, Lovász, Simmons Straus [13,19] raised this problem for first time.

This problem is extended from the real plane $\mathbb{R}^2$ to the real projective plane $\mathbb{P}^2$. A *generalized configuration of points* consists of n distinct points in the projective plane and an arrangement of $\binom{n}{2}$ pseudolines crossing from each pair of points and intersect each other exactly once. Halving lines in $\mathbb{R}^2$ can be similarly extended to *halving pseudolines* [15] for a generalized configuration of points in $\mathbb{P}^2$ and define $\widetilde{h}(n)$ is the maximum number of halving pseudolines.

The research is supported in part by NSF award CCF-1718994.

M. Changat and S. Das (Eds.): CALDAM 2020, LNCS 12016, pp. 463–475, 2020.
https://doi.org/10.1007/978-3-030-39219-2_37

A part of extensive research on finding bounds on $h(n)$ and $\widetilde{h}(n)$ can be found in [1,7,12,24].

Goodman and Pollack [17] introduced *allowable sequences of permutations* (allowable sequence for short) which are useful for encoding configurations of points in $\mathbb{P}^2$. Allowable sequence is a doubly infinite sequence and half-period of it can be represented by a sequence $\Pi = (\pi_0, \pi_1, \ldots, \pi_{\binom{n}{2}})$ of permutations on n elements such that:

(1) Any permutation $\pi_i, i \geq 1$ can be obtained from the previous permutation π_{i-1} by a transposition of two adjacent elements.
(2) Every two elements are transposed exactly one time.

A transposition between elements at positions k and $k+1$ is called a *k-transposition*. We denote by $\tau(k, \Pi)$ the number of k-transpositions in Π.

Allowable sequences are one of most the important tools in proving bounds for many problems of discrete geometry including bounds on the number of k-sets, halving lines, halving pseudolines, also problems of finding rectilinear crossing number of graph K_n and pseudolinear crossing number of K_n, see [1–3,6,9]. For example, the number of $(\leq k)$-sets of a set of n points in the plane in general position were studied in [11,20] using allowable sequences.

Most known upper bounds for $h(n)$ use the upper bound

$$h(n) \leq \widetilde{h}(n), \tag{1}$$

where $\widetilde{h}(n)$ is the maximum number of halving pseudolines. The definition of $\widetilde{h}(n)$ is based on generalized configurations of points in $\mathbb{P}^2$. For the sake of simplicity, we define it using allowable sequences as follows. Let Π be an allowable sequence of permutations on $[n]$. In this paper, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. First, we define $\widetilde{h}(\Pi)$ using two cases. If n is even then $\widetilde{h}(\Pi) = \tau(n/2, \Pi)$. If $n \geq 3$ is odd then $\widetilde{h}(\Pi) = \tau(\frac{n-1}{2}, \Pi) + \tau(\frac{n+1}{2}, \Pi)$. Then $\widetilde{h}(n)$ is the maximum value of $\widetilde{h}(\Pi)$ over all allowable sequences of permutations on $[n]$.

The bound (1) is used to show the tight bounds for the halving numbers by proving upper bounds for $\widetilde{h}(n)$ matching the lower bounds for $h(n)$. The tight bounds $h(n) = \widetilde{h}(n)$ are known for all $n \leq 27$ [4]. Inequality (1) can be viewed as the lower bound for $\widetilde{h}(n)$. Current lower bounds of $\widetilde{h}(n)$ and $h(n)$, for small n, are mostly can be attained by point configurations of Aichholzer's construction [5]. Can the bound (1) be improved?

In this paper, we propose to study lower bounds for $\widetilde{h}(n)$ using allowable sequences. This can be viewed as the problem of finding an allowable sequence Π maximizing $\widetilde{h}(\Pi)$ for a given n. The problem is known to be difficult for large n. Checking all possible configurations is computationally expensive as the number of simple arrangements of n pseudolines B_n, grows exponentially in n. Best upper bound for B_n is found by Felsner and Valtr [16] and best lower bound of B_n is provided by Dumitrescu and Mandal [10]. (see also [14,18,21])

$$\Omega(2^{0.2053n^2}) = B_n = O(2^{0.6571n^2}).$$

We propose an approach using *partial allowable sequences* described in Sect. 2. An interesting problem in our approach is the following sorting problem.

MAX-k **Sorting Problem**. Given a permutation π on $[n]$ and an integer $1 \leq k < n$, sort π using transpositions to $(n, n-1, n-2, \ldots, 1)$ such that

(1) The number of k-transpositions is maximized, and
(2) Every pair (i, j) can transposed at most one time.

Our Results. In this paper, we show that MAX-k sorting problem can be solved in $O(nk^n)$ time for any k and in $O(n^k)$ time for $k = 1, 2$. We develop an approach for optimizing allowable sequences and use it to find new bounds for halving pseudolines for even n, $n \leq 100$.

2 Transforming Allowable Sequences

Allowable sequences are very flexible and can be modified to increase $\widetilde{h}(\Pi)$. For example, if the transpositions in permutations π_i and π_{i+1} are non-overlapping, say transpositions at positions $j, j+1$ in π_i and positions $j', j'+1$ in π_{i+1} with $|j - j'| \geq 2$, then the transpositions in π_i and in π_{i+1} can be exchanged.

We define a *push operation* as follows. Consider a permutation π_j of an allowable sequence $\Pi = (\pi_0, \pi_1, \ldots, \pi_{\binom{n}{2}})$. Consider two elements $\pi_j(i) = a$ and $\pi_j(i+1) = b$. If $a > b$ then the transposition of a and b is at some permutation $\pi_{j'}$ before π_j, i.e. $j' < j$. We can push the transposition of a and b down from the j'th permutation to the jth permutation. Thus, a and b should be exchanged in all permutations between these two permutations, see an example at Fig. 1. We call this operation *push-down*.

	1	2	3	4		1	2	3	4
j'	1	3	2	4		2	1	3	4
	3	1	2	4		2	1	4	3
	3	1	4	2		2	4	1	3
	3	4	1	2		4	2	1	3
	4	3	1	2		4	2	3	1
j	4	3	2	1		4	3	2	1

Fig. 1. Pushing down the transposition of 2 and 3 from permutation $\pi_{j'}$ to π_j.

Now suppose that $a < b$ and the transposition of a and b is at some permutation $\pi_{j'}$ after π_j, i.e. $j' > j$. We can push the transposition of a and b up from the j'th permutation to the jth permutation. Thus, a and b should be exchanged in all permutations between these two permutations. We called the operation *push-up*.

Proposition 1. *For any two allowable sequences Π, Π' of permutations on n elements, there exists a sequence of push operations transforming Π into Π'. It holds even if the operations are restricted to push-down (or push-up) operations.*

We describe another tool for transforming allowable sequences. Let $\Pi = (\pi_0, \pi_1, \ldots, \pi_{\binom{n}{2}})$ be an allowable sequence. A *partial allowable sequence* $\Pi_{i,j}$ is a subsequence of consecutive permutations of Π, i.e. $\Pi_{i,j} = (\pi_i, \pi_{i+1}, \ldots, \pi_j)$. One way to optimize $\widetilde{h}(\Pi)$ is to choose a partial allowable sequence $\Pi_{i,j}$ and find another partial allowable sequence $\Pi'_{i,j} = (\pi'_i, \pi'_{i+1}, \ldots, \pi'_j)$ such that

(1) $\pi'_i = \pi_i, \pi'_j = \pi_j$, and
(2) the partial allowable sequence $\Pi_{i,j}$ can be transformed to $\Pi'_{i,j}$ using push operations within $\Pi_{i,j}$.

In many cases, the permutations of $\Pi_{i,j}$ have a common prefix and a common suffix. By removing them and renumbering the elements of the permutations, we reduce the size of the problem. Suppose that n is even. Then the halving transpositions correspond to k-transpositions for some value of k in the reduced problem. The elements of the reduced partial allowable sequence can be renumbered such that the last permutation is $(m, m-1, \ldots, 2, 1)$ for some $m \leq n$. Then the problem of optimizing the halving transpositions can be viewed as MAX-k sorting problem. In this paper, we mostly focus on this problem.

For odd n, the problem is to maximize the sum $\tau(k, \Pi) + \tau(k+1, \Pi)$. This problem will be discussed in Sect. 7.

The output of MAX-k sorting problem is a partial allowable sequence $\Pi_{i,j} = (\pi_i, \pi_{i+1}, \ldots, \pi_j)$. It can be represented by a *transcript* which is a sequence of integers $k_1, k_2, \ldots, k_{j-i}$ such that $\pi_s, (1 \leq s \leq j-i)$ is obtained from π_{s-1} by a k_s-transposition.

3 General MAX-k Sorting Problem

Let π a permutation on $[n]$ and k be an integer with $1 \leq k < n$. We associate a vector $v(\pi) = (v_1, v_2, \ldots, v_{n-1})$ with π where v_i is the number of elements $\pi(1), \pi(2), \ldots, \pi(i-1)$ larger than $\pi(i)$. Now, let i be the largest integer such that $v_i \geq k$. Let $m = \pi(i)$. Consider any sorting of π to the reverse of identity using adjacent transpositions. Every transposition of m in the sorting will be at positions $j, j+1$ where $j \geq k$. Thus, m will never be used in a k-transposition; therefore, we can remove it from π. By renumbering the permutation elements, we reduce π to a permutation on $[n-1]$. If $v_i < k$ for all i, we call the permutation π *k-bounded*. By repeating the above process, a permutation π can be reduced to a unique k-bounded permutation, say σ. We called $v(\sigma)$ a *k-vector* of π.

Let $\lambda_k(\pi)$ be the maximum number of k-transpositions in a sorting of permutation π. The use of vectors of k-bounded permutations is motivated by the following proposition.

Proposition 2. *Let π_1, and π_2 be two permutation on $[n]$. If the k-vectors of π_1 and π_2 are equal then $\lambda_k(\pi_1) = \lambda_k(\pi_2)$.*

Lemma 1. *The number of k-bounded permutations on $[n]$ is $(k-1)!k^{n-k+1}$.*

Theorem 3. MAX-k *sorting problem can be solved in $O(nk!k^{n-k})$ time.*

Proof. First, we will explain the main idea of the algorithm. Then, we show the improvement of the algorithm efficiency by changing the indexing process of storing array. We employ dynamic programming and compute arrays A_m for $m = 1, 2, \ldots, n$. An array A_m stores the maximum number of k-transpositions for all k-bounded permutations of size m. We use k^m entries in A_m since the number of k-bounded permutations of size m is $(k-1)!k^{m-k+1} \leq k^m$. A k-bounded permutation π of size m corresponds to $A_m[j]$ where $j = \sum_{i=1}^{m} v_i k^{i-1}$ and $v = v(\pi)$.

For a given vector v_π, the corresponding k-bounded permutation π can be computed in linear time following the proof of Proposition 1. Then the maximum number of k-transpositions in a sorting of π can be computed as follows. Apply one transposition and find the corresponding k-bounded permutation. Its length is either m or $m-1$. So, it is stored in A_m or A_{m-1}. There are at most $m-1$ possible transposition for π. We find the maximum value for them in $O(m)$ time. The running time for computing each A_m is $O(m^2k^m)$. Then the total running time is $\sum_{m \leq n} cm^2k^m = O(n^2k^n)$.

We modify the previous approach to improve both the running time and the space. The size of each array A_m can be reduced using Lemma 1. We change indexing for k-bounded permutations and introduce a one-to-one map T from set of k-bounded permutations of size m to $\{0, 1, \ldots, (k-1)!k^{m-k+1} - 1\}$ [1]. Let π be a k-bounded permutations of size m, and $v_\pi = (v_1, v_2, \ldots, v_m)$ be the associated vector with π. We define $T_a^b(\pi) = \sum_{i=a}^{b} \delta_k(i)v_i$ where $1 \leq a \leq b \leq m$, and $T(\pi) = T_1^m(\pi)$ where

$$\delta_k(i) = \begin{cases} 1 & i = 1 \\ \delta_k(i-1) \cdot i & 1 < i < k \\ \delta_k(i-1) \cdot k & i \geq k. \end{cases}$$

The vector $\delta_k = (\delta_k(1), \ldots, \delta_k(m)$ can be computed once at the beginning. Given an index $t = T(\pi)$ of a permutation π, π can be computed in $O(m)$ time. For a permutation π of size m, the values of $T_t^n(\pi)$, and $T_1^t(\pi)$ for $t \in [m]$ can be computed in $O(m)$ time. Let π' be the permutation obtained from π by applying one transposition, say p-transposition. The permutations π and π' are different only at positions p and $p+1$, the vector $v_{\pi'}$ may be different from v_π only at positions p and $p+1$. Specifically, if $v = v(\pi)$ and $u = v(\pi')$, then $u_p = v_{p+1}, u_{p+1} = v_p + 1$ and $u_i = v_i, i = 1, \ldots, p-1, p+2, \ldots, m$.

[1] This improves the space by a factor of $k^{k-1}/(k-1)!$. For example, if $k = 5$ this a factor of 26.041.

If $u_{p+1} < k$ then permutation π' is k-bounded and

$$T(\pi') = T(\pi) + \delta_k(p)v_{p+1} + \delta_k(p+1)(v_p+1) - \delta_k(p)v_p - \delta_k(p+1)v_{p+1}.$$

Suppose that $u_{p+1} \geq k$. Then $p \geq k$ and $\pi'(p+1)$ will be deleted and the elements of π' will be renumbered. Then π' corresponds to an entry of array A_{m-1} and

$$T(\pi') = T_1^p(\pi) + T_{p+2}^m(\pi)/k.$$

Notice that the computation of the final π' takes $O(m)$ time whereas $T(\pi')$ can be computed in $O(1)$ time. We avoid the computation of π' in this case and use $A_{m-1}[T(\pi')]$ directly.

The runnig time for computing each A_m is $O(m(k-1)!k^{m-k+1})$. Then the total running time is $O(nk!k^{n-k})$. □

4 MAX-1 Sorting Problem

We show that MAX-1 sorting problem can be solved by a greedy algorithm. Let π be a permutation on $[n]$. The following algorithm has two steps. The first step will maximize the number of 1-transposition and step 2 will complete the sorting of the permutation.

Step 1. While $\pi(1) \neq n$, pick the smallest i such that $\pi(i) > \pi(1)$ and move it to the first position, i.e. by swapping $\pi(i)$ with $\pi(i-1), \pi(i-2), \ldots, \pi(1)$.

Step 2. While $\pi(i) < \pi(i+1)$ for some i, swap $\pi(i)$ and $\pi(i+1)$.

To compute the maximum number of 1-transpositions for π, one can apply Step 1 without doing swaps. This can be done by computing the longest increasing sequence in π where start at first position.

Proposition 4. *MAX-1 sorting problem can be solved in linear time, i.e. the maximum number of 1-transpositions to sort π can be found in $O(n)$ time and the corresponding transcript can be found in time $O(n+I)$ where I is the size of the transcript.*

5 MAX-2 Sorting Problem

First, we solve MAX-2 sorting problem in a special case where the input permutation is $1, 2, \ldots, n$. Define a function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$

$$f(n) = \begin{cases} 3n/2 - 3, & \text{if } n \text{ is even,} \\ 3(n-1)/2 - 1, & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 5. *For any $n \geq 3$, the maximum number of 2-transpositions in a sorting of $1, 2, \ldots, n$ is $f(n)$.*

Proof. We omit the base case $M(3) \leq 2$ and $M(4) \leq 3$ due to the lack of space.

Inductive step. First, we show that, for any $n \geq 3$, $M(n) \leq f(n)$ implies $M(n+2) \leq f(n+2)$.

Let Π be an allowable sequence of size $n+2$. We transform it by pushing operators as follows. Consider elements $n, n+1$, and $n+2$. There are three transpositions in Π between these elements, and at most two of them are 2-transpositions. We push them down in the following order: transposition of n and $n+1$, transposition of n and $n+2$, and transposition of $n+1$ and $n+2$. The number of 2-transpositions in Π will not decrease. This is shown in Fig. 2 (the last four lines).

Now, $n+2$ is in third position, and it was never swapped to this position before. Therefore we can push-down the transpositions of $n+2$ with $n-1, n-2, \ldots, 1$, see Fig. 2. Similarly, we can push-down the transpositions of $n+1$ with $n-1, n-2, \ldots, 1$ as shown in Fig. 2. Let Π' be the allowable sequence of the remaining permutations after removing $n+1$ and $n+2$ from them. Let t and t' be the number of 2-transpositions in Π and Π', respectively. Then $t \leq t' + 3$. By induction hypothesis, $t' \leq f(n)$. Then $t \leq f(n) + 3 = f(n+2)$.

1	2	3	4	$\cdots$	n	$n+1$	$n+2$
$\cdots$				$\cdots$			
n	$n-1$	$n-2$	$n-3$	$\cdots$	1	$n+1$	$n+2$
n	$n-1$	$n-2$	$n-3$	$\cdots$	$n+1$	1	$n+2$
n	$n-1$	$n-2$	$n+1$	$\cdots$	2	1	$n+2$
n	$n-1$	$n+1$	$n-2$	$\cdots$	2	1	$n+2$
n	$n+1$	$n-1$	$n-2$	$\cdots$	2	1	$n+2$
n	$n+1$	$n-1$	$n-2$	$\cdots$	2	$n+2$	1
n	$n+1$	$n-1$	$n-2$	$\cdots$	$n+2$	2	1
n	$n+1$	$n-1$	$n+2$	$\cdots$	3	2	1
n	$n+1$	$n+2$	$n-1$	$\cdots$	3	2	1
n	$n+2$	$n+1$	$n-1$	$\cdots$	3	2	1
$n+2$	n	$n+1$	$n-1$	$\cdots$	3	2	1
$n+2$	$n+1$	n	$n-1$	$\cdots$	3	2	1

Fig. 2. An allowable sequence of size $n+2$.

The above argument can be used to construct an allowable sequence with $f(n)$ 2-transpositions for any $n \geq 5$, see Fig. 3 for an example. □

We show how to solve Max-2 sorting problem for a given permutation π on n elements. As in Sect. 3, consider the vector $v_\pi = (v_1, v_2, \ldots, v_{n-1})$ where v_i is the number of elements $\pi(1), \pi(2), \ldots, \pi(i-1)$ larger than $\pi(i)$. First, we observe that an element $\pi(i)$ can be removed from π if $v_i \geq 2$. We remove all elements $\pi(i)$ from π with π if $v_i \geq 2$. Without loss of generality, we assume that

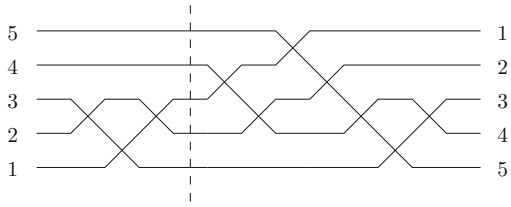

Fig. 3. The inductive step. The wiring diagrams for $n = 5$ constructed from the wiring diagrams for $n = 3$. The corresponding transcript of it is (2, 1, 2, 3, 2, 4, 3, 2, 1, 2).

all $v_i < 2$ in the vector v for π. Then vector v is simply a binary sequence. We define an *i-block*, $i = 0, 1$, as a maximal subsequence of v of consecutive i. Then vector v is a sequence of alternating blocks

$$v = B_1^0 B_1^1 B_2^0 B_2^1 \dots B_k^0 B_k^1, \tag{2}$$

where B_j^i is an i-block and block B_k^1 may be not present. Note that the first block of vector v must be a 0-block since $v_1 = 0$. Thus, the first block in (2) is B_1^0. We employ dynamic programming and compute $m_j^i, i = 0, 1, j = 1, 2, \dots, k$, the maximum number of 2-transpositions in a sorting of $\pi(1), \pi(2), \dots, \pi(l)$ where $l = |B_1^0| + |B_1^1| + |B_2^0| + |B_2^1| + \dots + |B_j^i|$. Initially $m_1^0 = f(|B_1^0|)$ if $|B_1^0| \geq 3$; otherwise $m_1^0 = 0$. If $|B_1^0| = 1$ then $m_1^1 = |B_1^1| - 1$; otherwise $m_1^1 = m_j^0 + |B_1^1|$. To make a recursive formula we consider a pair (j', j) such that $1 \leq j' < j \leq k$. Let α be the permutation obtained from sequence $\pi(1), \pi(2), \dots, \pi(l)$ by

(i) sorting first $|B_1^0| + |B_1^1| + \dots + |B_{j'}^1|$ elements, and then
(ii) deleting elements of π corresponding to blocks $B_{j'+1}^1, B_{j'+2}^1, \dots, B_{j-1}^1$.

Let $g(j', j)$ be the maximum number of 2-transpositions in a sorting of permutation α. Then,

$$m_j^0 = \max(f(l_j), \max_{1 \leq j' < j}(m_{j'}^1 + g(j', j))), \tag{3}$$

$$m_j^1 = \begin{cases} m_j^0 + |B_j^1|, & \text{if } j > 1 \text{ or } (j = 1 \text{ and } |B_1^0| > 1) \\ |B_j^1| - 1, & \text{otherwise.} \end{cases} \tag{4}$$

where $l_j = |B_1^0| + |B_2^0| + \dots + |B_j^0|$.

Computing $g(j', j)$. Let a be the total length of blocks $B_1^0, B_1^1, \dots, B_{j'}^1$ and let b be the total length of blocks $B_{j'+1}^0, B_{j'+2}^0, \dots, B_j^0$. Then vector α (after relabeling) is

$$\alpha = a, a-1, \dots, 1, a+1, a+2, \dots, a+b$$

and $g(j', j)$ can be computed as

$$g(j', j) = \begin{cases} 3b/2, & \text{if } b \text{ is even,} \\ 3(b-1)/2 + 1, & \text{if } b \text{ is odd.} \end{cases} \tag{5}$$

This can be shown similar to the proof of Theorem 5. In the base case, $b = 1, 2$ and α can be sorted using one and three 2-transpositions, respectively. The inductive case is similar to Fig. 3 and three 2-transpositions can be added.

The maximum number of 2-transpositions in a sorting of π is m_k^0 or m_k^1 if block B_k^1 exists. The corresponding transcript can be computed as follows. First, we modify the dynamic program and, for each $j = 1, 2, \ldots, k$, we store the value of j' that is used in computing m_j^0. If m_j^0 is computed as $f(l_j)$ in Eq. (3) then we store $j' = -1$. We denote by t_j^i the transcript corresponding to m_j^i, i.e. the transcript for sorting $\pi(1), \pi(2), \ldots, \pi(l)$ where $l = |B_1^0| + |B_1^1| + |B_2^0| + |B_2^1| + \cdots + |B_j^i|$. Then transcript t_j^1 is the transcript t_j^0 followed by inserting the elements of π corresponding to B_j^1 as in the insertion sort.

The transcript t_j^0 can be computed as follows. If $m_j^0 = f(l_j)$ then the transcript t_j^0 is obtained by using Theorem 5, see Fig. 3 for an example. Suppose that $m_j^0 = m_{j'}^1 + g(j', j)$ for some $1 \leq j' < j$. Then the transcript t_j^0 is the transcript $t_{j'}^1$ followed by the transcript obtained using the proof of Eq. 5.

Theorem 6. *For any permutation π on $[n], n \geq 3$, a transcript maximizing* Max-*2 sorting problem can be solved in $O(n^2)$ time.*

6 Improving Lower Bounds for $\widetilde{h}(n)$

In this section we use algorithms developed in previous sections to improve lower bounds for $\widetilde{h}(n)$ for even n. Since n is even, we use allowable sequences maximizing the number of $\frac{n}{2}$-transpositions. We can directly apply the Max-k sorting algorithm from Theorem 3 for $k = \frac{n}{2}$ but this is infeasible for large n. Instead, we devise a heuristic approach to the problem using a local optimization by

(i) creating a block and an allowable sequence and
(ii) solving the corresponding Max-k sorting problem for small k.

Let $\Pi = (\pi_0, \pi_1, \ldots, \pi_{\binom{n}{2}})$ be an allowable sequence. We define a (l, r)-*window* or simply a *window* for a permutation π_i of Π as the sequence $\pi_i(l), \pi_i(l+1), \ldots, \pi_i(r)$, (see Fig. 4(a)). In general, we define a *block*[2] of Π as a sequence of (l, r)-windows in consecutive permutations $\pi_i, \pi_{i+1}, \ldots, \pi_j$ of Π such that each window of the block has the same set of elements and every two consecutive windows are different, see Fig. 4 for examples. For a permutation in an allowable sequence, we call the $\frac{n}{2}$-th position of the permutation, the *halving position*.

Let B be a block in Π which consists of (l, r)-windows of length $r-l+1 = t$ on permutations $\pi_i, \pi_{i+1}, \ldots, \pi_j$. Suppose that there is at least one l-transposition and at least one $(r-1)$-transposition in the block. We also assume that block B overlaps with the halving position in Π. Consider the partial allowable sequence $\Pi_{i,j}$. Note that the optimization of this partial allowable sequence (as described in Sect. 2) corresponds to Max-k sorting problem derived from the block B as

[2] The blocks in this section are different from alternating blocks used in the proof of Theorem 6.

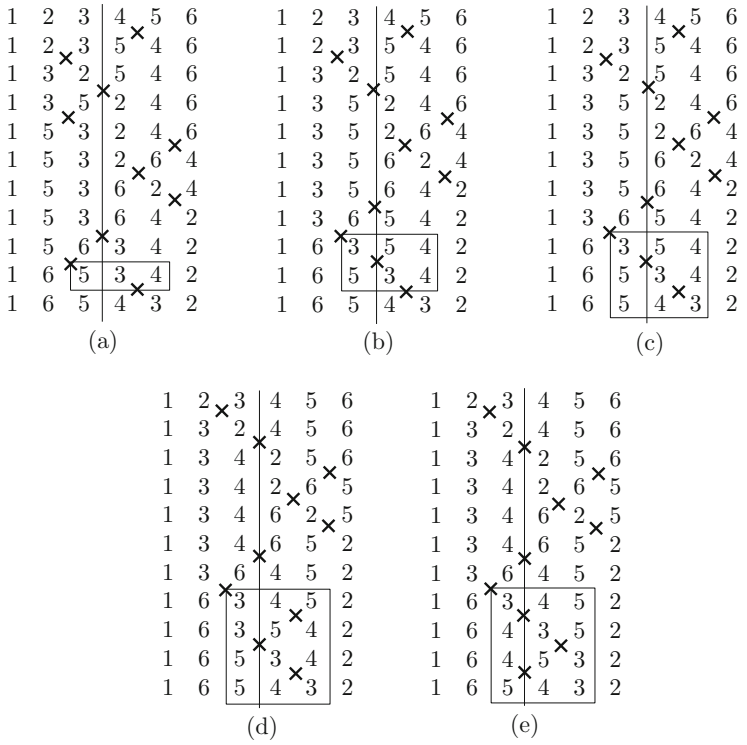

Fig. 4. (a) Initial window. (b–d) Extension of the window using push-up/push-down. (e) Applying the MAX-1 sorting algorithm.

follows. There is a bijection $\alpha : \{\pi_j(l), \pi_j(l+1), \ldots, \pi_j(r)\} \to [t]$ such that $\alpha(\pi_j(x)) = t - (x - l)$, $l \le x \le r$. This map transforms the last window of B into sequence $(t, t-1, t-2, \ldots, 1)$. Then the permutation for MAX-k sorting problem is $\alpha(\pi_i(l)), \alpha(\pi_i(l+1)), \ldots, \alpha(\pi_i(r))$ and the value of k corresponds to the halving position in the block. The solution of an algorithm for MAX-k sorting problem can be used to increase $\tau(\frac{n}{2}, \Pi)$ if the block in Π is replaced by $\alpha^{-1}(\Pi')$ where Π' is the output of the algorithm (a sequence of permutations).

A block B that allows to increase $\tau(\frac{n}{2}, \Pi)$ may not exist in the allowable sequence Π. We create new blocks using push operations as follows. First, we choose a (l, r)-window in a permutation of Π such that it includes the halving position We consider it as the initial block B_0. To construct a new block $B_i, i > 0$, we consider the permutation π_j containing the lower window w_j of B_{i-1}. Take two adjacent elements $\pi_j(t)$ and $\pi_j(t+1)$ in w_j such that $\pi_j(t) < \pi_j(t+1)$ and the corresponding transposition of $\pi_j(t)$ and $\pi_j(t+1)$ in Π is not a $\frac{n}{2}$-transposition. Apply the push operation (push-up) to this transposition such that it is a transposition between π_j and π_{j+1}. Then the block B_i is the extension of B_{i-1} using the (l, r)-window in permutation π_{j+1}. Note that this extension preserves the block property that every two consecutive windows are different.

The same process can be done to extend the block upward by pushing down transpositions (see Fig. 4(b)).

Let B be the block constructed by the above procedure. The block contains the halving position, say at its k-th position. By applying a MAX-k sorting algorithm for block B we may increase $\tau(k, B)$. This will increase $\tau(\frac{n}{2}, \Pi)$, see Fig. 4(e)) for an example. We run this algorithm all even n up to 100 for $k = 1$. The running time for each n is between two hours and two days. The results are shown in Table 1. Many of them improve known lower bounds for $\widetilde{h}$. The transcripts of these allowable sequences are available at http://www.utdallas.edu/~besp/soft/pseudo/halving/even.zip.

7 Future Work

In this paper, we used the algorithm discussed in Sect. 6 combined with the MAX-1 sorting algorithm from Sect. 4 on allowable sequences of even length to achieve the results shown in Table 1. For odd n, the problem of maximizing $\widetilde{h}(\Pi)$ for an

Table 1. Bounds for the $\widetilde{h}(n)$. **LB/UB** denotes the current lower bound/upper bound for $\widetilde{h}(n)$. The bounds in bold are obtained in this paper. **Prev** denotes the previous lower bound for $h(n)$ if they are improved in this paper. The numbers in this column are from [5,24].

n	LB	UB	Prev	n	LB	UB	Prev
28	63	64	-	66	**202**	236	197
30	69	72	-	68	**207**	246	203
32	74	79	-	70	**226**	257	211
34	**81**	86	79	72	**229**	268	211
36	**88**	94	84	74	**237**	279	228
38	**97**	102	94	76	**253**	291	237
40	**104**	110	103	78	**262**	303	242
42	111	119	-	80	**265**	315	250
44	**117**	127	112	82	**268**	327	261
46	**126**	136	122	84	**282**	339	264
48	**133**	146	129	86	**292**	351	276
50	**141**	155	139	88	**297**	363	282
52	**146**	164	143	90	**312**	376	290
54	**153**	174	152	92	**317**	388	300
56	**163**	183	158	94	**326**	401	309
58	**169**	193	165	96	**345**	414	308
60	**177**	204	172	98	**338**	427	320
62	**187**	214	180	100	**366**	440	328
64	**195**	225	187				

allowable sequence Π uses two halving positions. A heuristic solution is to use a MAX-k sorting algorithm on a block for each of halving positions separately. Another approach is to provide an algorithm for the problem of maximizing total number of k-transpositions and $(k+1)$-transpositions in sorting of a permutation. We will explore this approach for odd values of n in the future.

Every allowable sequence from Table 1 corresponds to an arrangement of pseudolines. If an arrangement of n pseudolines is stretchable, i.e., is isomorphic to an arrangement of n straight lines, then one can find a set of n points in the plane providing a new bound for $h(n)$. The problem of determining whether a pseudoline arrangement is stretchable is NP-hard, see [22,23,25]. It would be interesting to explore the stretchability of the pseudoline arrangements from Table 1, perhaps using the heuristic method by Bokowski [8].

References

1. Ábrego, B.M., Balogh, J., Fernández-Merchant, S., Leaños, J., Salazar, G.: An extended lower bound on the number of ($\leq k$)-edges to generalized configurations of points and the pseudolinear crossing number of K_n. J. Comb. Theory Ser. A **115**(7), 1257–1264 (2008)
2. Ábrego, B.M., Cetina, M., Fernández-Merchant, S., Leaños, J., Salazar, G.: On $\leq k$-edges, crossings, and halving lines of geometric drawings of k_n. Discrete Comput. Geom. **48**(1), 192–215 (2012)
3. Ábrego, B.M., Fernández-Merchant, S.: A lower bound for the rectilinear crossing number. Graphs Combin. **21**(3), 293–300 (2005)
4. Ábrego, B.M., Fernández-Merchant, S., Leaños, J., Salazar, G.: The maximum number of halving lines and the rectilinear crossing number of kn for n $\leq$ 27. Electron. Notes Discrete Math. **30**, 261–266 (2008)
5. Aichholzer, O.: On the rectilinear crossing number. http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing/
6. Alon, N., Györi, E.: The number of small semispaces of a finite set of points in the plane. J. Comb. Theory Ser. A **41**(1), 154–157 (1986)
7. Beygelzimer, A., Radziszowski, S.: On halving line arrangements. Discrete Math. **257**(2–3), 267–283 (2002)
8. Bokowski, J.: On heuristic methods for finding realizations of surfaces. In: Bobenko, A.I., Sullivan, J.M., Schröder, P., Ziegler, G.M. (eds.) Discrete Differential Geometry. Oberwolfach Seminars, vol. 38, pp. 255–260. Springer, Basel (2008). https://doi.org/10.1007/978-3-7643-8621-4_13
9. Cetina, M., Hernández-Vélez, C., Leaños, J., Villalobos, C.: Point sets that minimize ($\leq k$)-edges, 3-decomposable drawings, and the rectilinear crossing number of K_{30}. Discrete Math. **311**(16), 1646–1657 (2011)
10. Dumitrescu, A., Mandal, R.: New lower bounds for the number of pseudoline arrangements. In: Proceedings of 13th Symposium on Discrete Algorithms, pp. 410–425 (2019)
11. Edelsbrunner, H., Hasan, N., Seidel, R., Shen, X.J.: Circles through two points that always enclose many points. Geometriae Dedicata **32**, 1–12 (1989)
12. Eppstein, D.: Sets of points with many halving lines. Technical Report ICS-TR92-86, University of California, Irvine, Department of Information and Computer Science, August 1992

13. Erdős, P., Lovász, L., Simmons, A., Straus, E.: Dissection graphs of planar point sets. In: Srivastava, J.N. (ed.) A Survey of Combinatorial Theory, pp. 139–154. North-Holland, Amsterdam (1973)
14. Felsner, S.: On the number of arrangements of pseudolines. Discrete Comput. Geom. **18**, 257–267 (1997)
15. Felsner, S., Goodman, J.E.: Pseudoline arrangements. In: Handbook of Discrete and Computational Geometry, pp. 125–157. Chapman and Hall/CRC (2017)
16. Felsner, S., Valtr, P.: Coding and counting arrangements of pseudolines. Discrete Comput. Geom. **46**(3), 405–416 (2011)
17. Goodman, J.E., Pollack, R.: Semispaces of configurations, cell complexes of arrangements. J. Comb. Theory Ser. A **37**(3), 257–293 (1984)
18. Knuth, D.E. (ed.): Axioms and Hulls. LNCS, vol. 606. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55611-7
19. Lovász, L.: On the number of halving lines. Annal. Univ. Scie. Budapest. de Rolando Eötvös Nominatae, Sectio Math. **14**, 107–108 (1971)
20. Lovász, L., Vesztergombi, K., Wagner, U., Welzl, E.: Convex quadrilaterals and k-sets. In: Pach, J. (ed.) Towards a Theory of Geometric Graphs, pp. 139–148. Contemporary Mathematics, American Mathematical Society (2004)
21. Matoušek, J.: Lectures on Discrete Geometry, vol. 212. Springer-Verlag, New York (2002). https://doi.org/10.1007/978-1-4613-0039-7
22. Mnëv, N.: On manifolds of combinatorial types of projective configurations and convex polyhedra. Soviet Math. Doklady **32**, 335–337 (1985)
23. Mnev, N.E.: The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In: Viro, O.Y., Vershik, A.M. (eds.) Topology and Geometry — Rohlin Seminar. LNM, vol. 1346, pp. 527–543. Springer, Heidelberg (1988). https://doi.org/10.1007/BFb0082792
24. Rodrigo, J., López, M.D.: An improvement of the lower bound on the maximum number of halving lines in planar sets with 32 points. Electr. Notes in Discr. Math. **68**, 305–310 (2018)
25. Shor, P.: Stretchability of pseudolines is NP-hard. Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift (1991)