J Comb Optim (2016) 32:1133–1164
DOI 10.1007/s10878-015-9930-6

Approximate sorting of data streams with limited storage

Farzad Farnoud[1] · Eitan Yaakobi[1] ·
Jehoshua Bruck[1]

Published online: 28 August 2015

Abstract We consider the problem of approximate sorting of a data stream (in one pass) with limited internal storage where the goal is not to rearrange data but to output a permutation that reflects the ordering of the elements of the data stream as closely as possible. Our main objective is to study the relationship between the quality of the sorting and the amount of available storage. To measure quality, we use permutation distortion metrics, namely the Kendall tau, Chebyshev, and weighted Kendall metrics, as well as mutual information, between the output permutation and the true ordering of data elements. We provide bounds on the performance of algorithms with limited storage and present a simple algorithm that asymptotically requires a constant factor as much storage as an optimal algorithm in terms of mutual information and average Kendall tau distortion. We also study the case in which only information about the most recent elements of the stream is available. This setting has applications to learning user preference rankings in services such as Netflix, where items are presented to the user one at a time.

Keywords Approximate sorting · Data stream · Limited storage · Permutation distortion metrics · Weighted Kendall distortion · User preference ranking

✉ Farzad Farnoud
farnoud@caltech.edu

Eitan Yaakobi
yaakobi@caltech.edu

Jehoshua Bruck
bruck@caltech.edu

[1] California Institute of Technology, Pasadena, CA, USA

1 Introduction

In many applications, such as sensor networks, finance, and web applications, data may be available as a transient stream that is not permanently accessible (Babcock et al. 2002). Often, in these applications, the large volume of data or time constraints prevent storage of the whole stream before processing. Even if data is locally stored, certain storage media only allow sequential access in a time-efficient manner.

In this paper, we study the fundamental problem of sorting a data stream when internal storage is limited. As the nature of the problem makes rearranging the data into a sorted stream impossible, by sorting we mean *determining the ordering* of the elements of the stream. In our model, the amount of available internal storage limits the number of elements of the data stream that can be stored internally. Furthermore, only elements in internal storage can be compared with each other. Lack of storage capable of holding the whole data stream implies that sorting must be approximate; the goal is to produce a permutation that represents the ordering of the elements of the data stream as faithfully as possible. As in Babcock et al. (2002), we consider algorithms that make only one pass over the data stream.

To evaluate performance, we measure the distortion between the output permutation and the permutation representing the true ordering of the data. There are many possible distortion measures on permutations (Diaconis 1988), among which we consider the Kendall tau metric and its weighted version, as well as the Chebyshev metric. The Kendall tau metric can be viewed as the number of mistakes made by the algorithm, while the Chebyshev metric represents the maximum error in the rank of any element. Another quality measure considered in the paper is the mutual information between the true permutation and the output permutation, which reflects the amount of relevant information present in the output.

We first provide universal bounds on the performance of algorithms with limited storage, namely an upper bound on mutual information, and lower bounds on storage to obtain a given distortion between the true permutation and the output. Further, we present a simple algorithm that is asymptotically optimal in terms of mutual information and asymptotically requires a constant factor as much storage as any algorithm with the same average Kendall tau distortion. For the Chebyshev distortion, the algorithm is also asymptotically constant-factor-optimal, provided that normalized distortion, to be defined later, is bounded away from 0. We also consider the case in which algorithms temporarily have access to more storage in an initial phase. Additionally, we present an improvement of the aforementioned proposed algorithm for a special case of storage size. We also consider distortion in the weighted Kendall metric (Farnoud and Milenkovic 2013), which is capable of penalizing mistakes in certain positions of permutations more heavily based on the importance of those positions. Finally, we study a more restricting storage model where only the most recent elements of the stream are available, in order to model human memory.

The problem of sorting a data stream with limited storage goes back to the work of Munro and Paterson (1980), where they considered sorting of, and selecting from, data stored on a read-only tape and showed that for exact sorting of a stream of length n in p passes, one requires storage of size $\Theta(n/p)$. While they allowed making multiple passes over the data and considered only exact sorting, in this work we study the quality of *approximate* sorting that can be obtained in *one* pass. Since the work of Munro and Paterson, many papers have studied problems related to *selection* in data streams, such as finding the kth highest value or quantiles, in one or many passes, e.g., Greenwald and Khanna (2001), Manku et al. (2008), Chakrabarti et al. (2008), and McGregor and Valiant (2012). The problem of approximate sorting in one pass, however, to the best of our knowledge, has not been studied.

The rest of this paper is organized as follows. In Sect. 2, we present the formal problem statement and preliminaries. Section 3 includes universal bounds on the performance of algorithms with limited storage. In Sect. 4, an algorithm for sorting with limited storage is given and its performance is analyzed. Section 5 discusses the weighted Kendall distance as a distortion measure. Finally, in Sect. 6, we study the case where only information about the most recent elements of the stream is available, which is applicable to human memory.

2 Problem statement and preliminaries

For a positive integer n, we let $[n] = \{1, \ldots, n\}$. The set of all permutations of $[n]$ is denoted by $\mathbb{S}_n$. For a permutation $\pi \in \mathbb{S}_n$ and distinct $i, j \in [n]$, we use $i \prec_\pi j$ (resp. $i \succ_\pi j$) to denote that i appears before j (resp. after j) in π. For example, if $\pi = (2, 3, 1)$, we have $2 \prec_\pi 3$ and $1 \succ_\pi 2$. The inverse of π is denoted by π^{-1}. The *rank* of element i in π is its position in π, that is, $\pi^{-1}(i)$.

The data stream is denoted by the sequence $s = s_1, s_2, \ldots, s_n$, of length n, which may be a stream of real numbers, text files, etc. We assume that a total order $<$ exists on the elements of s. This total order is represented by a permutation $X \in \mathbb{S}_n$ as follows: For distinct $i, j \in [n]$, if $i \prec_X j$, i.e., if i appears before j in X, then $s_i < s_j$. The goal is to approximate X as closely as possible. While X is not directly accessible in our setting, the relationship between every two elements s_i and s_j of s can be queried (or computed) if they are both present in internal storage, and the result of the query is either $s_i < s_j$ or $s_j < s_i$ (equivalently, $i \prec_X j$ or $j \prec_X i$).

Example 1 Suppose $s = (98, 15, 23, 13, 2, 89, 60, 118, 104)$. Then, X is given as $X = (5, 4, 2, 3, 7, 6, 1, 9, 8)$. For instance, note that $s_2 < s_1$ implies that $2 \prec_X 1$ and vice versa.

Throughout the paper, our assumption is that X is chosen uniformly and at random among the permutations of $\mathbb{S}_n$ but we only consider deterministic algorithms.

The elements of s are revealed in a streaming fashion, i.e., one by one. If an element of the stream is not stored internally when revealed, it will not be possible to access it in the future. The storage limitation is that there are m cells each of which can store one element of s and thus any algorithm can only access m elements of the sequence s at any one time. The set of these m cells is termed *stream memory.* When a new element s_i of the stream s arrives, it can only be stored in the stream memory if there is an empty cell or if the contents of a cell is discarded; otherwise, s_i is ignored. To make a query regarding the relative order of s_i and s_j with respect to X, both s_i and s_j should be stored in the stream memory. We do not impose any other type of storage limitation. For example, there is no restriction on the number of integer values that an algorithm can store and access. In particular, one can store n integers but not the n elements of the data stream. This assumption is for simplifying the analysis and is also valid when each element of s is much larger than other types of data that an algorithm may require. To avoid trivial cases, we assume $n, m \geq 2$.

The output of the algorithms considered here is a permutation, denoted Y. To measure performance, we evaluate how "close" Y is to X. Closeness between two permutations can be quantified in a variety of ways. We use the Kendall tau and Chebyshev metrics, defined below, as well as the mutual information between X and Y.

The *Kendall tau distance* between two permutations $\pi, \sigma \in \mathbb{S}_n$ is the number of pairs of distinct elements i and j such that $i \prec_\pi j$ and $j \prec_\sigma i$, or equivalently, the number of adjacent transpositions needed to take π to σ. This distance is denoted as $d_K(\pi, \sigma)$. The Chebyshev distance between π and σ, denoted $d_C(\pi, \sigma)$, is defined as

$$\max_{i \in [n]} \left| \pi^{-1}(i) - \sigma^{-1}(i) \right| .$$

In other words, the Chebyshev distance is the maximum difference in the rank of any element in the two permutations.

Example 2 Consider s and X given in Example 1 and suppose that the output of a certain algorithm is $Y = (5, 2, 4, 3, 7, 9, 1, 6, 8)$. This output corresponds to the approximate sorting s' of s,

$$s' = (2, 15, 13, 23, 60, 104, 98, 89, 118) .$$

Note that s' is given for demonstration only; the output of the algorithm is Y and s' is not available. For the Kendall tau and Chebyshev distortions, we have $d_K(X, Y) = 4$ and $d_C(X, Y) = 2$.

We also discuss the weighted Kendall distance, defined in Sect. 5, which is a distance measure that assigns different weights to different positions in permutations. This distance is for example useful when one wants to penalize errors at the top positions more heavily than those at the bottom.

For two functions f_n and g_n of n, the notation $f_n \sim g_n$ is used to denote $\lim_{n\to\infty} f_n/g_n = 1$. Furthermore, we use lg and ln as shorthands for $\log_2$ and $\log_e$, respectively.

3 Universal bounds

In this section, we present bounds on the performance of any algorithm that can only store m elements of the sequence s. To derive these bounds, we use the fact that to make a query for comparing two elements s_i and s_j, both need to be present in the stream memory and so the amount of information that can be obtained via queries is limited because of the limitation on storage. As mentioned earlier, X is a random element of $\mathbb{S}_n$ but only deterministic algorithms are considered. We first present bounds on the mutual information between X and the output permutation Y and then consider distortion under the Kendall tau and Chebyshev metrics.

We use $H(X)$ and $I(X;Y)$ to refer to the entropy of X and the mutual information between X and Y, respectively. For these functions, logarithms are base 2. Note that as X is a random element of $\mathbb{S}_n$, we have $H(X) = \lg n!$.

Theorem 1 *For any algorithm with stream memory of size m, we have*

$$I(X;Y) \le n \lg m - m \lg e + O(\lg m).$$

Furthermore, $I(X;Y^*)/H(X) \sim \lg m / \lg n$ for $m, n \to \infty$, where Y^* is the output of an algorithm that maximizes the mutual information between X and Y.

Proof Let Z be the set of responses provided to the comparison queries made by the algorithm. Since the algorithm can only have access to X through Z, we have $X \to Z \to Y$, i.e., the random variables X, Z, Y form a Markov chain in that order (Cover and Thomas 2006), and equivalently $Y \to Z \to X$. By the data processing inequality (Cover and Thomas 2006, Theorem 2.8.1), we find $I(Y;X) \le I(Y;Z)$. Furthermore, $I(Y;Z) \le H(Z)$.

We now show that $H(Z) \le \lg m! + (n-m)\lg m$. The first m elements of s can be fully compared and so $m!$ cases arise from their ordering. Let Z_0' be an integer in $[m!]$ identifying the permutation representing the ordering of the first m elements. Each of the next $n-m$ elements can at most be compared with $m-1$ elements already present in the stream memory. These $m-1$ elements define m intervals, into one of which the new element falls. For $i \in \{m+1, \dots, n\}$, let Z_i' be an integer in $[m]$ identifying the interval in which the ith element of the stream falls. Given the algorithm, Z is a deterministic function of $\left(Z_0', Z_{m+1}', Z_{m+2}', \dots, Z_n'\right)$ and thus

$$\begin{aligned} H(Z) &\le H\left(Z_0', Z_{m+1}', Z_{m+2}', \dots, Z_n'\right) \\ &\le H\left(Z_0'\right) + \sum_{i=m+1}^{n} H\left(Z_i'\right) \\ &\le \lg m! + (n-m)\lg m. \end{aligned}$$

It follows that $I(X;Y) \le H(Z) \le \lg m! + (n-m)\lg m$. The first theorem statement then follows from the Stirling approximation: For a positive integer k, we have $\lg k! = k\lg k - k\lg e + O(\lg k)$.

Since $I(X;Y) \le \lg m! + (n-m)\lg m$ holds for $Y = Y^*$, we have

$$\frac{I(X;Y^*)}{H(X)} \le \frac{n\lg m + O(m)}{n\lg n + O(n)} = \frac{\lg m}{\lg n}(1 + o(1)), \tag{1}$$

where we have used the fact that $\frac{m}{n\lg m} = O(1/\lg n) = o(1)$. In Sect. 4, we present an algorithm that produces an output Y_1 such that

$$\frac{I(X;Y_1)}{H(X)} \ge \frac{\lg m}{\lg n}(1 + o(1)), \quad m, n \to \infty.$$

Since $I(X;Y^*) \ge I(X;Y_1)$, we have

$$\frac{I(X;Y^*)}{H(X)} \ge \frac{\lg m}{\lg n}(1 + o(1)), \quad m, n \to \infty. \tag{2}$$

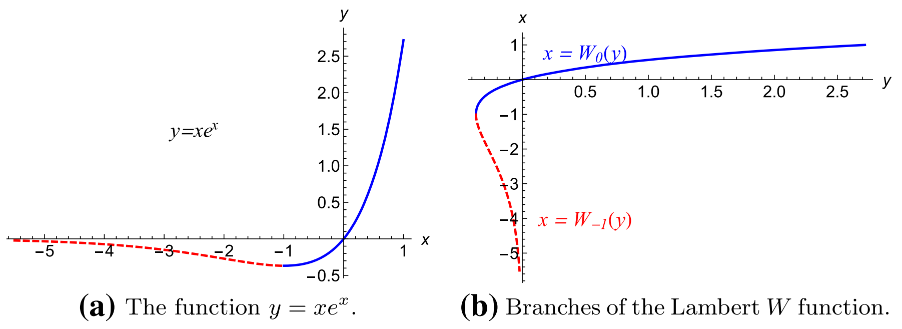

Fig. 1 The inverse of the function $y = xe^x$ (**a**) is called the Lambert W function (**b**). The function $y = xe^x$ is decreasing in $(-\infty, -1)$ and increasing in $(-1, \infty)$; these regions are indicated with *different line styles*. Each of these regions gives rise to one branch of the inverse function in (**b**), namely W_{-1} and W_0

The second statement of the theorem follows from (1) and (2). □

In particular, if $m = n^\beta + O(1)$ for a constant β, then a β fraction of the information of X can be recovered by an algorithm with stream memory m.

Next, we use the rate-distortion theory to find lower bounds on storage for a specific value of the average Kendall tau distortion between X and Y, defined as $E[d_K(X, Y)]$. We use δ to denote the normalized version of this distortion, that is, $\delta = E[d_K(X, Y)]/n$. This choice leads to simpler expressions. Note that since d_K can be of the order of n^2, δ can take on values in the range $[0, \infty)$.

The following theorem applies to any algorithm with stream memory m. We use W_0 and W_{-1} to respectively denote the principal and the lower branches of the Lambert W function. The Lambert W function (Corless et al. 1996) is defined as the inverse of $y = xe^x$. See Fig. 1 for plots of these functions and a brief note on the branches of W.

Theorem 2 *Let $\mu = \frac{m}{n}$ and $\delta = \frac{E[d_K(X,Y)]}{n}$. Suppose ϵ is a positive constant. For any algorithm with stream memory m and $\delta > \epsilon$, we have*

$$\mu \geq -W_0\left(-\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\right)\left(1 + O\left(\frac{\lg n}{n}\right)\right), \tag{3}$$

and

$$\mu \geq \frac{1}{e^2\delta}\left(1 + O\left(\frac{\lg n}{n}\right) + O\left(\frac{1}{\delta}\right)\right). \tag{4}$$

Proof Since we only consider deterministic algorithms, the number M of outputs of a given algorithm is bounded from above by $m!m^{n-m}$. This statement can be proven in a similar manner to the upper bound on $H(Z)$ in Theorem 1.

Let $A = \frac{1}{n} \lg \frac{M}{n!}$. We have $\lg M \leq n \lg m - m \lg e + O(\lg m)$ and so

$$\begin{aligned} A &\leq \tfrac{1}{n}(n \lg m - m \lg e - n \lg n + n \lg e + O(\lg n)) \\ &= \lg\left(\mu e^{1-\mu}\right) + O\left(n^{-1} \lg n\right). \end{aligned}$$

The parameter M can be viewed as the size of a rate-distortion code. Hence, from Farnoud et al. (2014a, Theorem 5), and Farnoud et al. (2014b), we have the following relationship between the average distortion $E\left[d_K(X,Y)\right]$ and M, expressed in terms of δ and A,

$$A \geq \lg \frac{\delta^\delta}{(1+\delta)^{1+\delta}} + O\left(\frac{\lg n}{n}\right).$$

From this and the fact that $A \leq \lg\left(\mu e^{1-\mu}\right) + O\left(n^{-1}\lg n\right)$, we obtain

$$\mu e^{1-\mu} \geq \frac{\delta^\delta}{(1+\delta)^{1+\delta}} e^{O\left(n^{-1}\lg n\right)}.$$

Since for all real x, we have $e^x \geq 1+x$, we find

$$\mu e^{1-\mu} \geq \frac{\delta^\delta}{(1+\delta)^{1+\delta}} \left(1 + O\left(\frac{\lg n}{n}\right)\right),$$

or equivalently,

$$-\mu e^{-\mu} \leq \frac{-\delta^\delta}{e(1+\delta)^{1+\delta}} \left(1 + O\left(\frac{\lg n}{n}\right)\right). \tag{5}$$

Hence

$$\mu \geq -W_0\left(\frac{-\delta^\delta}{e(1+\delta)^{1+\delta}} \left(1 + O\left(\frac{\lg n}{n}\right)\right)\right). \tag{6}$$

For convenience, let $g\left(\delta\right) = \frac{\delta^\delta}{e(1+\delta)^{1+\delta}}$. By taking derivatives, one can show that the function W_0 is concave. Hence,

$$\begin{aligned} W_0\left(-g(\delta) + g(\delta)O\left(n^{-1}\lg n\right)\right) &\leq W_0(-g(\delta)) + W_0'(-g(\delta))g(\delta)O\left(n^{-1}\lg n\right) \\ &= W_0(-g(\delta)) + \frac{W_0(-g(\delta))g(\delta)O\left(n^{-1}\lg n\right)}{g(\delta)\left(1 + W_0(-g(\delta))\right)} \\ &= W_0(-g(\delta))\left(1 + \frac{O\left(n^{-1}\lg n\right)}{1 + W_0(-g(\delta))}\right) \\ &= W_0(-g(\delta))\left(1 + O\left(n^{-1}\lg n\right)\right). \end{aligned}$$

Note that for $\delta \geq 0$, the expression $-g(\delta)$ is strictly increasing and $-g(\delta) \in [-1/e, 0)$. Since $\delta > \epsilon > 0$, we have $-g\left(\delta\right) > -g\left(\epsilon\right) > -1/e$. Furthermore, $W_0\left(x\right)$ is strictly increasing for $x \geq -1/e$ and so $W_0\left(-g\left(\delta\right)\right) > W_0\left(-g\left(\epsilon\right)\right) > -1$. Hence for some positive constant ϵ', we have $W_0\left(-g\left(\delta\right)\right) \geq -1 + \epsilon'$, from which the last step of the above derivation follows. We finally have,

$$\mu \geq -W_0\left(-\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\right)\left(1 + O\left(\frac{\lg n}{n}\right)\right). \tag{7}$$

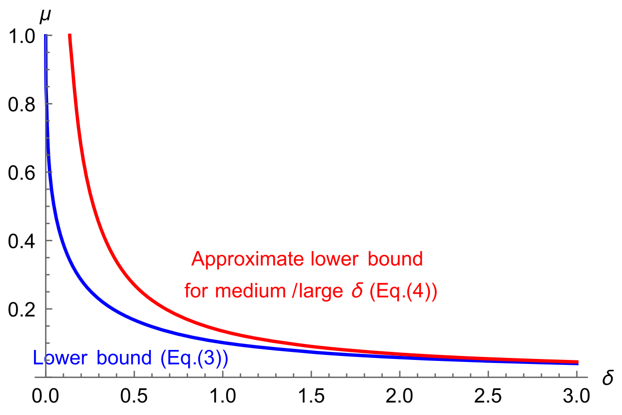

Fig. 2 The lower bound (3) on the memory requirement of any algorithm with average Kendall distortion δ and the lower bound (4), which becomes more accurate as δ becomes larger. Both bounds are plotted for $n \to \infty$, so we have ignored the term $O\left(\frac{\lg n}{n}\right)$. For (4), we have also ignored the term $O\left(\frac{1}{\delta}\right)$

To prove the second statement, note that by concavity of W_0 and the facts that $W_0(0) = 0$ and $W_0'(0) = 1$, we have

$$\begin{aligned} W_0\left(-\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\right) &\le -\frac{\delta^\delta}{e(1+\delta)^{1+\delta}} \\ &\le -\frac{1}{e^2(1+\delta)} \\ &= -\frac{1+O(1/\delta)}{e^2\delta}. \end{aligned} \tag{8}$$

The second statement of the theorem then follows from (7) and (8). □

The inequalities (3) and (4) of Theorem 2 are illustrated in Fig. 2. In this figure, as well as all following figures in the paper, we ignored the asymptotically negligible terms. It is worth noting that the approximation is close to the lower bound even for moderate values of δ.

Finally, we consider the Chebyshev distortion between X and Y. The normalized Chebyshev distortion is $\chi = E\left[d_C(X,Y)\right]/n$. We only consider the case of $\chi \le 1/2$ which is more important as it represents small distortions.

Theorem 3 *Let* $\mu = \frac{m}{n}$ *and* $\chi = \frac{E[d_C(X,Y)]}{n}$. *Suppose* $2/n \le \chi \le 1/2$. *For any algorithm with stream memory m, we have*

$$\mu \ge -W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\left(1+O\left(n^{-1}\lg n\right)\right).$$

Proof Let M be defined as in the proof of Theorem 2 and let $R = \frac{1}{n}\lg M$. Since $\lg M \le n\lg m - m\lg e + O(\lg m)$, we have $R \le \lg m - \frac{m}{n}\lg e + O(n^{-1}\lg m)$.

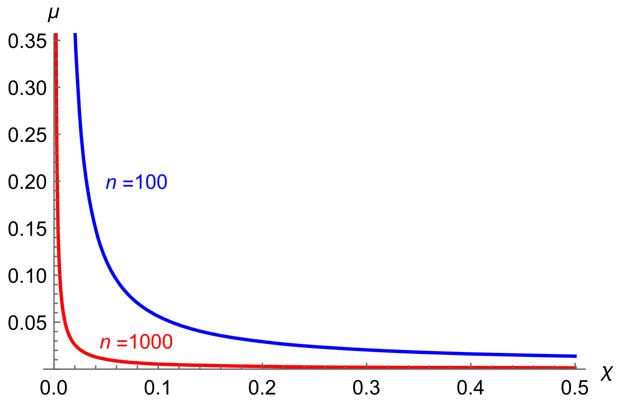

Fig. 3 The lower bound on the memory requirement of any algorithm with average Chebyshev distortion χ, for $n = 100$ and $n = 1000$. See Theorem 3

From Farnoud et al. (2014a, Theorem16), and Farnoud et al. (2014b), we find $R \geq \lg \frac{1}{2\chi} + 2\chi \lg \frac{e}{2} + O(n^{-1} \lg n)$ for $\chi \leq 1/2$. Hence,

$$\lg \frac{1}{2\chi} + 2\chi \lg \frac{e}{2} \leq \lg m - \frac{m}{n} \lg e + O(n^{-1} \lg n)$$

implying that $\mu e^{-\mu} \geq \frac{(e/2)^{2\chi}}{2\chi n} \left(1 + O\left(n^{-1} \lg n\right)\right)$, or equivalently,

$$\mu \geq -W_0 \left(-\frac{(e/2)^{2\chi}}{2\chi n} \left(1 + O\left(n^{-1} \lg n\right)\right)\right).$$

Since $\chi \leq 1/2$, we have $(e/2)^{2\chi} \leq e/2$ and since $\chi \geq 2/n$, we have $2\chi n \geq 4$. So $-\frac{(e/2)^{2\chi}}{2\chi n} \geq -\frac{e}{8} > -\frac{1}{e}$. Hence, $W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)$ is bounded away from -1. We have

$$\begin{aligned}
&W_0 \left(-\frac{(e/2)^{2\chi}}{2\chi n} \left(1 + O\left(n^{-1} \lg n\right)\right)\right) \\
&\overset{\text{(a)}}{\leq} W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) + W_0'\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) \frac{(e/2)^{2\chi} O\left(n^{-1} \lg n\right)}{2\chi n} \\
&= W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) \frac{O\left(n^{-1} \lg n\right)}{1 + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)} \\
&\overset{\text{(b)}}{=} W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\left(1 + O\left(n^{-1} \lg n\right)\right),
\end{aligned}$$

where (a) and (b) follow from the concavity of W_0 and the fact that $1 + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)$ is bounded away from 0, respectively. □

In Fig. 3, we plot the lower bound given in Theorem 3, ignoring the term $\left(1 + O\left(n^{-1} \lg n\right)\right)$. Note the dependance of the bound on the value of n.

4 Algorithm for limited-storage approximate sorting

We present the following simple algorithm for approximately sorting a stream using storage of size m and then present results regarding its performance. Let $c_1, \ldots, c_m$ denote the m memory cells capable of storing elements of the stream. Recall that $s_i < s_j$ if i appears before j in X, i.e., $i \prec_X j$.

Algorithm 1

```
1. Store the first $m-1$ elements of $s$ in memory cells $c_1, \ldots, c_{m-1}$.
2. Find permutation $y$ of $\{1, \ldots, m-1\}$ such that $s_{y_1} < s_{y_2} < \cdots < s_{y_{m-1}}$.
3. Let $Y_1 \leftarrow y$.
4. For each new element $s_i$, $i = m, m+1, \ldots, n$, of the stream:
   (a) Store $s_i$ in $c_m$.
   (b) If there exists $j$ such that $s_{y_{j-1}} < s_i < s_{y_j}$, insert $i$ immediately before $y_j$ in $Y_1$.
   (c) If $s_i < s_j$ for all $j \in [m-1]$, insert $i$ immediately before $y_1$ in $Y_1$.
   (d) If $s_i > s_j$ for all $j \in [m-1]$, append $i$ to the end of $Y_1$.
```

In this algorithm, the first $m-1$ elements, namely, $s_1, \ldots, s_{m-1}$, are stored in the memory for the duration of the algorithm and every new element is compared with these. An element that is stored in memory, for the purpose that new elements can be compared with it, is called a *pivot*.

Example 3 Same as Example 1, suppose

$$s = (98, 15, 23, 13, 2, 89, 60, 118, 104),$$
$$X = (5, 4, 2, 3, 7, 6, 1, 9, 8).$$

Furthermore, suppose $m = 3$. After step 3 of Algorithm 1, we have $y = Y_1 = (2, 1)$. For $i = 3$, Y_1 is updated to $(\mathbf{2}, 3, \mathbf{1})$, where the indices of the pivots are shown in bold. For $i = 4$ and $i = 5$, Y_1 is respectively updated to $(4, \mathbf{2}, 3, \mathbf{1})$ and $(4, 5, \mathbf{2}, 3, \mathbf{1})$. The final output is $Y_1 = (4, 5, \mathbf{2}, 3, 6, 7, \mathbf{1}, 8, 9)$, which corresponds to the approximate sorting s' of s,

$$s' = (13, 2, \mathbf{15}, 23, 89, 60, \mathbf{98}, 118, 104).$$

For the Kendall tau and Chebyshev distortions, we have $d_K(X, Y_1) = 3$ and $d_C(X, Y_1) = 1$.

In Algorithm 1, the index set of pivots is $\{1, 2, \ldots, m-1\}$ and they are in correct order in Y_1. However, indices of elements between the pivots, and between the pivots and the boundaries, are sorted in the natural increasing order which may differ from their order in X, e.g., the subsequence 3, 6, 7 of Y_1 in the preceding example. Let $r_1, \ldots, r_{m-1}$ be an increasing sequence that denotes the positions of the indices of the pivots in X (or equivalently in Y_1). Furthermore, let $r_0 = 0$ and $r_m = n + 1$. In Example 3, we have $r_0 = 0$, $r_1 = 3$, $r_2 = 7$, and $r_3 = 10$. For $j \in [m]$, the elements of Y_1 between positions r_{j-1} and r_j can have any order in X. Additionally, all possibilities are equally probable. Therefore, we have the following lemma.

Lemma 1 *Given Y_1, the number of possible cases for X is given by*

$$\prod_{j=1}^{m} \left(r_j - r_{j-1} - 1\right)!\,,$$

where all cases are equally likely.

In Theorem 4 we show that Algorithm 1 is asymptotically optimal with respect to mutual information. First however, we explain why this result is intuitively expected. The random choice of the $m-1$ pivots in Algorithm 1 divides the sorted version of the sequence s into m *segments* with lengths $r_j - r_{j-1} - 1$, $j \in [m]$. The expected length of each of these segments is approximately n/m. If the actual lengths of the segments are close to this expected value, from the preceding lemma, the number of possible cases for X given Y_1 is estimated as $\left(\frac{n}{m}!\right)^m \simeq \left(\frac{n}{m}\right)^n$ and so $I(X;Y_1) = H(X) - H(X|Y_1) \simeq n \lg m$. We thus may expect that $\frac{I(X;Y_1)}{H(X)} \simeq \frac{\lg m}{\lg n}$, which is optimal.

Theorem 4 *Algorithm 1 is asymptotically optimal for $m, n \to \infty$, with respect to mutual information.*

Proof Since $I(X;Y_1) = H(X) - H(X|Y_1)$ and $H(X) = \lg n!$, to find $I(X;Y_1)$, it suffices to find $H(X|Y_1)$. From Lemma 1, we have

$$\begin{aligned} H(X|Y_1) &= \sum_{z \in \mathbb{S}_n} P(Y_1 = z) H(X|Y_1 = z) \\ &= \binom{n}{m-1}^{-1} \sum_{r_0 < \cdots < r_m} \sum_{j=1}^{m} \lg \left(r_j - r_{j-1} - 1\right). \end{aligned}$$

For given values of $r_0, \ldots, r_m$, the set $[n]$ is divided into m blocks with lengths $r_j - r_{j-1} - 1$. To compute the above sum, we count how many times a block of size k occurs for all possible values of $r_0, \ldots, r_m$. The number of times a block of length k appears starting at position 1 equals $\binom{n-k-1}{m-2}$ since we have $r_1 = k+1$ but must choose the values of $r_2, \ldots, r_{m-1}$ among the $n-(k+1)$ possibilities. The number of times a block of size k ends at position n is the same. A similar argument shows that the number of times a block of length k starts at position i and ends at position $i+k-1$, for each $i \in \{2, \ldots, n-k\}$, is $\binom{n-k-2}{m-3}$. Thus, the total number of blocks of size k is $2\binom{n-k-1}{m-2} + (n-k-1)\binom{n-k-2}{m-3} = m\binom{n-k-1}{m-2}$. Hence,

$$H(X|Y_1) = \binom{n}{m-1}^{-1} m \sum_{k=2}^{n-m+1} \binom{n-k-1}{m-2} \lg k!. \tag{9}$$

From Lemma 6 of the appendix, we have

$$\sum_{k=1}^{n-m+1} \binom{n-k-1}{m-2} k \lg k \le \binom{n}{m} \left(\lg \frac{n}{m} + O(1)\right). \tag{10}$$

From (9), (10), and the fact that $\lg k! < k \lg k$, we obtain

$$H(X|Y_1) \leq (n-m+1)\left(\lg \frac{n}{m} + O(1)\right) = n \lg \frac{n}{m} + O(n)$$

and so $I(X; Y_1) \geq \lg n! - n \lg \frac{n}{m} + O(n) = n \lg m + O(n)$. Thus $\frac{I(X;Y_1)}{H(X)} \geq \frac{\lg m}{\lg n}(1 + o(1))$ for $m, n \to \infty$. Recall from the proof of Theorem 1 that $\frac{I(X;Y)}{H(X)} \leq \frac{\lg m}{\lg n}(1 + o(1))$ for the output Y of any algorithm. Therefore,

$$\frac{I(X; Y_1)}{H(X)} = \frac{\lg m}{\lg n}(1 + o(1)),$$

which is optimal. □

The next theorem and its corollary prove the optimality of Algorithm 1 with respect to the Kendall tau distortion, up to a constant factor. Before moving to the formal proof, we state an approximate analysis of the Kendall tau distortion of the algorithm. As mentioned earlier, the random choice of pivots in the algorithm creates segments with expected length approximately equal to n/m. The average Kendall tau distortion induced by each segment is then about $\frac{1}{2}\binom{n/m}{2} \simeq (n/m)^2/4$. Since there are m segments, we may expect $\delta \simeq \frac{1}{n} \cdot m \cdot (n/m)^2/4 = 1/(4\mu)$ or equivalently $\mu \simeq 1/(4\delta)$. Recall from Theorem 2 that if δ is bounded away from zero, then for any algorithm, $\mu = \Omega(1/\delta)$. Hence, this discussion points to the optimality of Algorithm 1. As we see in the following theorem however, this analysis is somewhat optimistic as it assumes the best case, namely, the case in which all segments have equal lengths. Nevertheless, this rough approximation correctly predicts the optimality of Algorithm 1 up to a constant factor.

Theorem 5 *Suppose Algorithm 1 has stream memory m and produces an output with average Kendall tau distortion $d_K(X, Y_1)$. We have*

$$E[d_K(X, Y_1)] = \frac{1}{m+1}\binom{n-m+1}{2}.$$

Denoting $m = \mu_1 n$ and $\delta n = E[d_K(X, Y_1)]$, it follows that

$$\mu_1 \leq \left(1 + \delta - \sqrt{\delta(\delta+2)}\right)(1 + O(1/n)),$$

and, if μ_1 is bounded away from 1, that

$$\delta \leq \frac{(1-\mu_1)^2}{2\mu_1}(1 + O(1/n)). \tag{11}$$

Proof Without loss of generality, assume $1 \prec_X \cdots \prec_X m-1$. Consider distinct $i, j \in \{m, \ldots, n\}$, with $i < j$. The pair i, j will have incorrect order in Y_1, if and only if $j \prec_X i$ and there is no $p \in \{1, \ldots, m-1\}$ such that $j \prec_X p \prec_X i$ (in

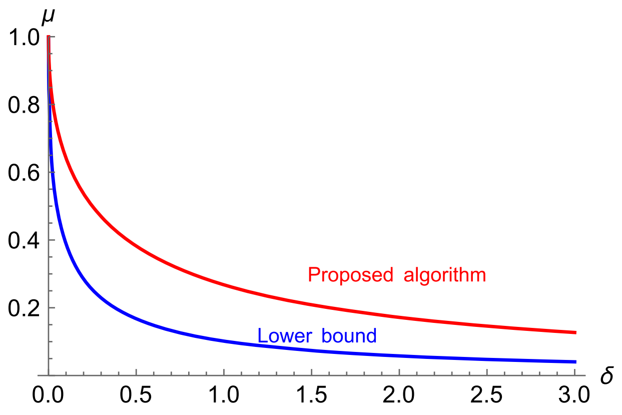

Fig. 4 Comparison of the lower bound on storage requirement of any algorithm and the storage required by Algorithm 1 for a given average Kendall tau distortion

other words, there is no pivot s_p such that $s_j < s_p < s_i$). Since X is random, it is straightforward to see that the probability of this event is $1/(m+1)$. There are $\binom{n-m+1}{2}$ possible choices for the pair i, j. By the linearity of expectation, we find $E\left[d_K(X, Y_1)\right] = \frac{1}{m+1}\binom{n-m+1}{2}$.

We have $\delta n = E\left[d_K(X, Y_1)\right] = \frac{1}{\mu_1 n+1}\binom{n-\mu_1 n+1}{2}$ and thus $\delta n \le \frac{(n-\mu_1 n+1)^2}{2\mu_1 n}$. From this, (11) follows and also

$$\begin{aligned}\mu_1 &\le 1+\delta+\frac{1}{n}-\sqrt{\delta(\delta+2+2/n)}\\ &= \left(1+\delta-\sqrt{\delta(\delta+2)}\right)(1+O(1/n)).\end{aligned}$$

□

Corollary 1 *Algorithm 1 asymptotically requires at most a constant factor as much storage as an optimal algorithm with the same average Kendall tau distortion.*

Proof From the upper bound on μ_1 given in the preceding theorem, we have $\mu_1 \le 1/(2\delta)\,(1+O(1/\delta))\,(1+O(1/n))$.

Let $\mu^* n$ be the smallest amount of stream memory of any algorithm with average Kendall tau distortion δn. From (4), we have

$$\frac{\mu_1}{\mu^*} \le \frac{1/(2\delta)}{1/\left(e^2\delta\right)}\,(1+O(1/\delta))\,(1+O(\lg n/n))\,.$$

Thus there is a constant c such that for $\delta, n \ge c$, μ_1/μ^* is bounded.

On the other hand, if $\delta < c$, from (3) and using the fact that μ^* is a decreasing function of δ, we have $\mu^* \ge -W_0\left(-c^c e^{-1}(1+c)^{-1-c}\right)(1+O(\lg n/n))$. Furthermore $\mu_1 \le 1$. Hence, if $\delta < c$, again μ_1/μ^* is bounded. □

In Fig. 4, we compare the performance of Algorithm 1 with the lower bound on storage for any algorithm, given in Theorem 2.

The next theorem and corollary are concerned with the average Chebyshev distortion of Algorithm 1.

Theorem 6 *Suppose Algorithm 1 has memory m and produces an output with average Chebyshev distortion χn. Furthermore, suppose that $\chi \leq 1/2$ and $m \geq 2$. We have*

$$\chi \leq \frac{1+\ln m}{m},$$
$$m \leq -\frac{1}{\chi} W_{-1}\left(-\frac{\chi}{e}\right).$$

Proof Consider an element i in Y_1 that is between positions r_{j-1} and r_j. We know that the position of this element in X is also between r_{j-1} and r_j. Thus, $\left|X^{-1}(i) - Y_1^{-1}(i)\right| \leq r_j - r_{j-1} - 1$ and so

$$d_C(X, Y_1) \leq \max_j \left(r_j - r_{j-1} - 1\right).$$

Suppose a stick of length n is randomly broken at $m-1$ points. Let the length of the longest piece among the m pieces be denoted by S. From Holst (1980), we have $E[S] = nE[S']/m$, where S' is the largest random variable among m iid exponential random variables with mean 1. We have $E[S'] = \sum_{i=1}^{m} 1/i \leq \ln m + \gamma_e + \frac{1}{2m}$, where the inequality follows from Chen and Qi (2008) and γ_e is Euler's constant. Since the positions of the pivots in Algorithm 1 are random, with a coupling argument one can show that the expected length of the longest segment is not more than $E[S]$. That is, $E\left[\max_j \left(r_j - r_{j-1} - 1\right)\right] \leq E\left[S\right]$. Hence,

$$E[d_C(X, Y_1)] = \chi n \leq \frac{n}{m}\left(\ln m + \gamma_e + \frac{1}{2m}\right).$$

Since $m \geq 2$ we have $\gamma_e + \frac{1}{2m} \leq 1$, and thus $\chi \leq \frac{\ln(me)}{m}$. This in turn implies that $-m\chi e^{-m\chi} \leq -\frac{\chi}{e}$, from which it follows that $-(1/\chi)W_0(-\chi/e) \leq m \leq -(1/\chi)W_{-1}(-\chi/e)$, which completes the proof. Note that for $\chi \leq 1$, we have $-(1/\chi)W_0(-\chi/e) \leq 1$ and hence the inequality $-(1/\chi)W_0(-\chi/e) \leq m$ does not give us any useful information. □

Corollary 2 *If χ is bounded away from zero, Algorithm 1 asymptotically requires at most a constant factor as much memory as an optimal algorithm with the same average Chebyshev distortion.*

Proof Let μ_1 denote m/n for Algorithm 1 and μ^* denote the smallest amount of storage of any algorithm with Chebyshev distortion χn. From the preceding theorem and Theorem 3,

$$\frac{\mu_1}{\mu^*} \leq \frac{-\frac{1}{\chi n} W_{-1}\left(-\frac{\chi}{e}\right)}{-W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)}\left(1 + O\left(\frac{\lg n}{n}\right)\right)$$
$$\sim \frac{-\frac{1}{\chi n} W_{-1}\left(-\frac{\chi}{e}\right)}{\frac{(e/2)^{2\chi}}{2\chi n}}$$

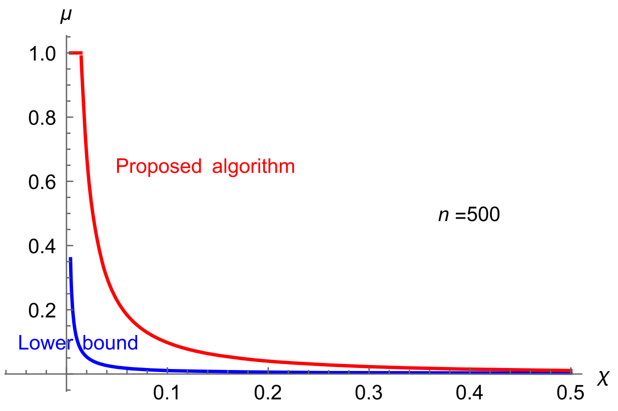

Fig. 5 The lower bound on the memory requirement of any algorithm with average Chebyshev distortion χ compared with the storage that Algorithm 1 requires for $n = 500$

$$\sim \frac{-2W_{-1}\left(-\frac{\chi}{e}\right)}{(e/2)^{2\chi}}. \tag{12}$$

Suppose χ is bounded away from 0. It follows that $-\chi/e$ is also bounded away from 0. This in turn implies that $-W_{-1}(-\chi/e)$ is bounded and so is the right side of (12). □

Figure 5 presents an illustration of the storage requirement of Algorithm 1 and the lower bound on storage of any algorithm (given in Theorem 3) versus average Chebyshev distortion.

In what follows we provide two variants of Algorithm 1. The first deals with a relaxation of the storage limitation and the second provides improved performance for $m = 2$. The main modification in both variants is that the set of pivots is no longer the same set of elements throughout the algorithm, but it changes during its execution.

4.1 Using more than *m* storage units in initial phase

In this subsection, we consider a variant of Algorithm 1, where the storage restriction is relaxed to allow larger storage in the initial phase of the algorithm. The idea is that increasing storage temporarily in the beginning may enable one to choose better pivots from a set that has more than m elements. This approach bears a resemblance to the median-of-three method for choosing pivots in Quicksort (Sedgewick and Wayne 2011), where instead of randomly choosing the pivot element, one chooses the median of three randomly chosen elements, resulting in better partitioning of the elements. Here, we study whether a similar performance improvement is observed with respect to the average Kendall tau distortion, when choose m pivots from a larger set of elements.

Since applications are rarely executed in isolation on devoted hardware, it is beneficial to consider cases in which the storage requirement can vary during different stages of execution. For example, as is the case in our setting, an application may utilize more storage in an initial phase and then a part of this space is freed up so that it can be used by other applications running on the system. In general it is useful to identify which stages of an algorithm may benefit from extra resources and how these resources, if available, can improve the performance. In Algorithm 1, since pivots play a crucial role, the initial stage of the algorithm can be modified to benefit from more storage. After this stage of the algorithm ends, this extra storage is released to the system to be used by other applications.

Suppose that the algorithm can store ℓ elements, where $\ell > m$ for the duration of receiving the first ℓ elements. After receiving the first ℓ elements, the algorithm chooses $m-1$ pivots among the ℓ possible choices, keeps them in memory, and releases the extra $\ell - m$ storage units. From this point, the algorithm proceeds similarly to Algorithm 1 and uses only stream memory of size m.

Ideally we want the pivots to be distributed as uniformly as possible in the sorted version of s. If this is the case, we will have $E\left[d_K(X,Y)\right] \simeq \frac{m}{2}\binom{n/m}{2} \simeq \frac{n^2}{4m}$, assuming m is small with respect to n. This is an improvement of a factor of 2 over the distortion given by Theorem 5, $E\left[d_K(X,Y_1)\right] = \frac{1}{m+1}\binom{n-m+1}{2} \simeq \frac{n^2}{2m}$. To get pivots that are uniformly distributed, it appears that one should choose pivots that are uniformly distributed among the initial ℓ elements. In what follows, we show that this intuition is correct and that the reduction by a factor of 2 of the average distortion can be obtained.

Let $A_\ell\left(v_0,\ldots,v_m\right)$ be the probability that a certain pair of elements among $s_{\ell+1},\ldots,s_n$ are placed in the wrong order in the final output given that *among the first* ℓ *elements*, the pivots are in ranks $(v_i)_{i=0}^m$, where we let $v_0 = 0$ and $v_m = \ell + 1$ for convenience. Using an argument similar to that of the proof of $E\left[d_K(X,Y_1)\right] = \frac{1}{m+1}\binom{n-m+1}{2}$ of Theorem 5, we obtain

$$A_\ell\left(v_0,\ldots,v_m\right) = \frac{\sum_{i=1}^m \binom{v_i - v_{i-1}+1}{2}}{(\ell+1)(\ell+2)}.$$

We now find the set of values of $(v_i)_{i=0}^m$ that minimize $A_\ell\left(v_0,\ldots,v_m\right)$. For $\alpha, \beta \in \mathbb{R}$, note that

$$\binom{\alpha}{2} + \binom{\beta}{2} > \binom{\alpha+1}{2} + \binom{\beta-1}{2}$$

if and only if $\beta > \alpha + 1$. Furthermore, note that $\sum_{i=1}^m \left(v_i - v_{i-1} + 1\right) = \ell + m + 1$. So to minimize $A_\ell\left(v_0,\ldots,v_m\right)$ we find the values of v_i such that

$$\left|\left(v_i - v_{i-1} + 1\right) - \left(v_j - v_{j-1} + 1\right)\right| \le 1.$$

Hence, the pivots should be chosen among the first ℓ elements as uniformly as possible. To do so, put $\rho = \ell + m + 1 \bmod m$ and let

$$v_i - v_{i-1} + 1 = \left\lfloor \frac{\ell+m+1}{m} \right\rfloor + 1, \quad \text{for } 1 \le i \le \rho$$

$$v_i - v_{i-1} + 1 = \left\lfloor \frac{\ell+m+1}{m} \right\rfloor, \quad \text{for } \rho + 1 \le i \le m.$$

Furthermore, let $A_\ell^*(m)$ denote the the value of A_ℓ with the aforementioned values for v_i. Then,

$$\begin{aligned} A_\ell^*(m) &= \frac{\rho\binom{\left\lfloor\frac{\ell+m+1}{m}\right\rfloor+1}{2} + (m-\rho)\binom{\left\lfloor\frac{\ell+m+1}{m}\right\rfloor}{2}}{(\ell+1)(\ell+2)} \\ &= \frac{(\ell+1+\rho)(\ell+1+m-\rho)}{2m(\ell+1)(\ell+2)} = \frac{1}{2m}\left(1+O\left(\frac{m}{\ell}\right)\right) \end{aligned}$$

and so the average Kendall tau distortion is

$$\frac{1}{2m}\binom{n-\ell}{2}\left(1+O\left(\frac{m}{\ell}\right)\right).$$

In particular, suppose $m, \ell, n \to \infty$ but ℓ is much larger than m and much smaller than n, i.e. $m = o(\ell)$ and $\ell = o(n)$. Then the average Kendall tau distortion is $\frac{n^2}{4m}(1+o(1))$. As expected, this is a reduction by a factor of 2 compared to the distortion of Algorithm 1, $\frac{n^2}{2m}(1+o(1))$, given by Theorem 5.

4.2 Updating the pivot for $m = 2$

In this subsection, we study the possibility of improving the performance of Algorithm 1 by modifying the pivots after they are chosen, based on their rank among a certain number of received elements. For simplicity, we limit ourselves to the special case of $m = 2$, that is, there is a single pivot, and we focus solely on the Kendall tau distortion. This relatively simple case illustrates the possibility of improving the performance by updating the pivots, as we show below. We expect the same updating strategy to be also useful for $m > 2$. However, a rigorous study of these cases is considerably more complex and postponed to future work.

First, we note that for $m = 2$ according from (5), we have

$$\frac{2}{n}e^{-2/n} \geq \frac{1}{e(1+1/\delta)^\delta(1+\delta)}(1+o(1)).$$

Since $e^{-2/n} \leq 1$ and $(1+1/\delta)^\delta \leq \left(e^{1/\delta}\right)^\delta = e$, we find

$$\frac{2}{n} \geq \frac{1}{e(1+\delta)}(1+o(1)),$$

implying the lower bound

$$\delta \geq \frac{n}{2e^2}(1+o(1)) \approx 0.135\frac{n}{2}(1+o(1))$$

on δ. On the other hand, according to Theorem 5, the normalized distortion guaranteed by Algorithm 1 is

$$\frac{E\left[d_K\left(X, Y_1\right)\right]}{n} = \frac{1}{3n}\binom{n-1}{2} = \frac{1}{3}\frac{n}{2} + O(1).$$

We now consider the scenario in which we change the pivot only once and the main concern here is how to decide whether the pivot should be changed. We start by executing Algorithm 1 so the pivot is s_1. Suppose that t items are received and a of them are smaller than the pivot s_1 while b of them are larger than s_1 ($t = a + b + 1$). Without loss of generality, assume that $a > b$. Intuitively, if a is much larger than b, it may be better to change the pivot to an element smaller than s_1. In what follows, we investigate circumstances under which it is indeed helpful to change the pivot. We do this by comparing between keeping the same pivot and changing it to the last received element that is smaller than s_1. This choice of the alternative pivot, simplifies our derivation. We state and prove our result to this problem in the following lemma.

Lemma 2 *Assume that after receiving $t = a + b + 1$ elements, the pivot s_1 is in rank $a + 1$ where $a > b$. If $t = o(n)$, then replacing s_1 with the last received element smaller than s_1 improves the expected distortion if and only if $a > 3b + 1$.*

Proof After receiving $a + b + 1$ elements, let A, B, and C be the following sets:

$$A = \left\{s_j : j \leq a + b + 1\right\},$$
$$C = \left\{s_j : a + b + 2 \leq j \leq n\right\}.$$

First, suppose that we keep the current pivot s_1, so the output is the permutation output Y_1 of Algorithm 1. We calculate the probability that two elements of the stream are not in the correct order in Y_1. For two elements u and v of the stream and two permutations π and σ, let $D_{u,v}(\pi, \sigma)$ be the event that π and σ disagree on the relative ranks of u and v, i.e., the event that ($u \prec_\sigma v$ & $v \prec_\pi u$) or ($u \prec_\pi v$ & $v \prec_\sigma u$). For $u, v \in C,\ u \neq v$, assuming v arrives after u, we have

$$P\left(D_{u,v}(X, Y_1)\right) = P(v < u < s_1) + P(s_1 < v < u) = \frac{\binom{a+2}{2} + \binom{b+2}{2}}{2\binom{a+b+3}{2}}.$$

Hence, the expected distortion is

$$E[d_K(X, Y_1)] = \frac{n^2}{2} \cdot \frac{\binom{a+2}{2} + \binom{b+2}{2}}{2\binom{a+b+3}{2}} + o\left(n^2\right),$$

as $|A|^2$ and $|A|\,|C|$ are $o(n^2)$.

Now, let s_i be the last received element that is smaller than s_1. Note that among the first $a + b + 1$ elements of s, s_i is ranked immediately before s_1. Now, suppose that we change the pivot to s_i after receiving the first $a + b + 1$ elements. Let the final result of the algorithm be denoted by Y_1' in this case. For $u, v \in C,\ u \neq v$, assuming v arrives after u, we have

$$\begin{aligned} P\left(D_{u,v}\left(X, Y_1'\right)\right) &= P(v < u < s_i) + P(s_i < v < u) \\ &= \tfrac{1}{a}\textstyle\sum_{j=1}^{a} \frac{\binom{j+1}{2}+\binom{a+b+3-j}{2}}{2\binom{a+b+3}{2}} \\ &= \frac{\binom{a+2}{3}+\binom{a+b+3}{3}-\binom{b+3}{3}}{2a\binom{a+b+3}{2}}, \end{aligned}$$

where the second line is the result of conditioning on the rank of s_i among the a elements that are smaller than s_1. Thus the expected distortion is

$$E[d_K(X, Y_1')] = \frac{n^2}{2} \cdot \frac{\binom{a+2}{3} + \binom{a+b+3}{3} - \binom{b+3}{3}}{2a\binom{a+b+3}{2}} + o\left(n^2\right).$$

Hence, it is better to change the pivot iff

$$\binom{a+2}{3} + \binom{a+b+3}{3} - \binom{b+3}{3} > a\binom{a+2}{2} + a\binom{b+2}{2} \iff a > 3b+1.$$

□

Suppose that after receiving the $t = a + b + 1$st element, we update the pivot to s_i, defined above, if $a > 3b + 1$. Let the output of the algorithm in this case be denoted by Y_1''. The distortion of this modified algorithm is calculated in the next theorem.

Theorem 7 *If $t = o(n)$ and is unbounded then the expected distortion δ' of Y_1'' becomes*

$$\frac{181}{576}\frac{n}{2} + o(n).$$

Proof Since $t = o(n)$, the dominant term in the expected distortion is produced by $u, v \in C$, $u \neq v$, for which from the proof of the previous lemma we have

$$\begin{aligned} P\left(D_{u,v}\left(X, Y_1''\right)\right) = \frac{1}{t} \sum_{a=0}^{\lfloor (3t-2)/4 \rfloor} \frac{\binom{a+2}{2} + \binom{t-a+1}{2}}{2\binom{t+2}{2}} \\ + \frac{1}{t} \sum_{a=\lfloor (3t-2)/4 \rfloor+1}^{t-1} \frac{\binom{a+2}{3} + \binom{t+2}{3} - \binom{t-a+2}{3}}{2a\binom{t+2}{2}}. \end{aligned} \tag{13}$$

Using this equality, in the appendix, we show that

$$P\left(D_{u,v}\left(X, Y_1''\right)\right) = \frac{181}{576} + o(1). \tag{14}$$

The expected distortion δ' then becomes

$$\frac{1}{n}\left(\frac{181}{576}\frac{n^2}{2} + o\left(n^2\right)\right).$$

Note that the coefficient $\frac{181}{576} \approx 0.3142$ should be compared with $\frac{1}{3} \approx 0.3333$, obtained from Theorem 5. □

5 Weighted distance

In many applications, such as evaluating search engine performance and preference aggregations, different positions in rankings are of different significance. In particular, often errors in the top positions in rankings should be penalized more heavily than those in the bottom positions. The metrics considered in the previous sections, as well as all conventional distance (distortion) metrics on permutations, disregard the importance of positions in which errors occur. In this section, we use a weighted distance as a measure of performance that takes into account the positions of errors.

To address the aforementioned shortcoming of conventional distances, several alternative distance measures have been recently proposed in the literature including those in Shieh (1998), Yilmaz et al. (2008), Carterette (2009), Kumar and Vassilvitskii (2010), and Farnoud and Milenkovic (2013). Here we use the method proposed by Farnoud and Milenkovic (2013) that introduces the weighted Kendall distance with an axiomatic approach based on the original set of distance axioms put forth by Kemeny (1959).

A transposition $(i\ j)$ is a permutation that is obtained from the identity by swapping the positions of i and j. For $i \in [n-1]$, $(i\ i+1)$ is called an adjacent transposition. For a permutation π, $\pi\ (i\ j)$ is the permutation obtained from π by swapping the elements of π that are in *positions* i and j. To define the weighted Kendall distance, we need a nonnegative function w that assigns weight w_i to each adjacent transposition $(i\ i+1)$. The weight of a sequence $(i_1\ i_1+1)\,,(i_2\ i_2+1)\,,\ldots,(i_k\ i_k+1)$ with respect to w is $\sum_{j=1}^{k} w_{i_j}$. A sequence $(i_1\ i_1+1)\,,(i_2\ i_2+1)\,,\ldots,(i_k\ i_k+1)$ of adjacent transpositions is said to take a permutation π to σ if $\pi\ (i_1\ i_1+1)\ (i_2\ i_2+1)\cdots(i_k\ i_k+1)=\sigma$. The *weighted Kendall distance* (Farnoud and Milenkovic 2013) between π and σ with respect to w is denoted $d_w(\pi,\sigma)$ and defined as the minimum weight of a sequence of adjacent transpositions that takes π to σ. For example, consider the weight function

$$w_i=\begin{cases}1, & 1\le i\le 2,\\ 0, & \text{else},\end{cases}$$

and let $\pi=(4,1,2,3)$ and $\sigma=(1,2,3,4)$. It can be shown that a minimum weight sequence that takes π to σ is $(1\ 2)\,,(2\ 3)\,,(3\ 4)$ and that $d_w(\pi,\sigma)=w_1+w_2+w_3=2$. Note that transpositions $(1\ 2)$ and $(2\ 3)$ contribute to the distance but $(3\ 4)$ does not.

In this section, we discuss distortion with respect to the weighted Kendall distance. We consider decreasing weight functions as they penalize errors in higher positions of rankings more severely. We assume here that after the pivots are assigned, they do not change for the duration of the data stream. In the following subsections, we find the optimum ranks of the pivots for two special cases of w. However, note that in Algorithm 1, one cannot choose the pivots. The information about the optimum positions for the pivots is thus useful in settings where one can find percentile values of the data stream from some side information or estimate them in a manner similar to that of Sect. 4.1 with a relaxed initial storage restriction. In the latter case, instead of choosing the m pivots uniformly among the ℓ elements, we choose them so that their rank is as close as possible to the optimum ranks obtained in the following subsections. We start by presenting a lemma from Hassanzadeh (2013) that will be useful in our analysis.

Lemma 3 *Consider a non-increasing weight function w and suppose that π is a randomly chosen permutation of length k. We have*

$$E\left[d_w\left(\pi, \mathsf{id}\right)\right] = \sum_{h=1}^{k-1} w_h\left(k-h\right)\left(H_k - H_{k-h}\right)$$

where id *stands for the identity permutation and* $H_i = \sum_{j=1}^{i} \frac{1}{j}$.

Suppose that the $m-1$ pivots have ranks $r_1, \ldots, r_{m-1}$ among the n elements and let $r_0 = 0$ and $r_m = n+1$. Furthermore, suppose that a non-increasing weight function $(w_i)_{i=1}^{n-1}$ is given. The portion of the ranking between each two pivots is a random permutation of the corresponding elements. By Lemma 3, the average distance contributed by the elements between ranks r_{i-1} and r_i equals

$$\sum_{h=r_{i-1}+1}^{r_i-2} w_h\left(r_i - 1 - h\right)\left(H_{r_i - r_{i-1}-1} - H_{r_i-h-1}\right).$$

Hence, the average weighted Kendall distance between the true permutation and the output permutation given that the ranks of the pivots are given by a vector r is

$$\begin{aligned} E\left[d_w(X,Y)|r\right] &= \sum_{i=1}^{m} \sum_{h=r_{i-1}+1}^{r_i-2} w_h\left(r_i-1-h\right)\left(H_{r_i-r_{i-1}-1} - H_{r_i-h-1}\right) \\ &= \sum_{i=1}^{m} \sum_{h=1}^{r_i - r_{i-1}-2} h w_{r_i-1-h}\left(H_{r_i-r_{i-1}-1} - H_h\right). \end{aligned} \tag{15}$$

In the next two subsections, we focus on two special cases of the weight function w.

5.1 Top *k*

As an example, let $w_h = 1$ for $h \in [k]$ and $w_h = 0$ otherwise. This weight function only penalizes adjacent transpositions in the top k positions. As a result, it is useful when only the correctness of the ordering of the top k elements matter to us. To illustrate the effect of this weight function, we consider the case of $m = 2$, and find the optimum value for r_1. Intuition may suggest that the pivot should be at position k to enable us to separate the top k elements from the rest. However, as shown formally below, the pivot should have rank less than k. This can be explained by observing that choosing r_1 to be less than k leads to a more accurate ordering of the top k elements, while setting $r_1 = k$ would only identify these elements but assign a random order to them.

It can be shown that we must have $r_1 \le k+1$. For $r_1 = k+1$, from (15),

$$\begin{aligned} E\left[d_w(X,Y)|r_1 = k+1\right] &= \sum_{h=1}^{r_1-2} h(H_{r_1-1} - H_h) \\ &= \sum_{h=1}^{r_1-2} \sum_{j=h+1}^{r_1-1} \frac{h}{j} = \frac{(r_1-1)(r_1-2)}{4} = \frac{k\left(k-1\right)}{4}. \end{aligned} \tag{16}$$

For $r_1 \leq k$, we have

$$E\left[d_w(X,Y)|r_1\right] = \sum_{h=1}^{r_1-2} h(H_{r_1-1} - H_h) + \sum_{h=n-k}^{n-1-r_1} h(H_{n-r_1} - H_h)$$
$$= \frac{1}{4}(r_1-1)(r_1-2) - 2(n-k)(n-k-1)\left(H_{n-r_1} - H_{n-k}\right)$$
$$+ \frac{1}{4}(k-r_1)(2n-k-r_1-1). \tag{17}$$

From (16) and (17), it can be seen that $r_1 = k$ leads to a smaller distortion than $r_1 = k+1$. Hence, it suffices to only consider the case of $r_1 \leq k$.

From (17), one can show that for $x \leq k$,

$$E\left[d_w(X,Y)|r_1 = x\right] - E\left[d_w(X,Y)|r_1 = x-1\right] \leq 0$$

if and only if $n(3r - 2k - 4) + k(k+1) - 2(r-1)^2 \leq 0$. Hence, for a constant k and sufficiently large n, the best value for r_1, given that it is at most k, is the largest integer smaller than $\frac{2k+4}{3}$, i.e, $\left\lceil \frac{2k+1}{3} \right\rceil$.

Using the method of Sect. 4.1 with ℓ units of storage at the initial phase, to pick a pivot with rank close to $\left\lceil \frac{2k+1}{3} \right\rceil$, one can pick the element among the ℓ received elements whose expected rank in s is closest to $\left\lceil \frac{2k+1}{3} \right\rceil$. It can be shown in a straightforward manner that the expected rank in s of an element that has rank i among the initial ℓ elements is $i\frac{n+1}{\ell+1}$ (see the computation of $E[U_i]$ in the proof of Lemma 5 for a similar derivation). So among the ℓ elements, we pick the element with rank $i = \left\lceil \frac{2k+1}{3} \right\rceil \frac{\ell+1}{n+1}$ as the pivot.

5.2 Linear weights

Let us now consider the linear weight function $w_h = c(n-1-h)+1$, where c is a positive constant. This weight function corresponds to the case where the importance of correctness of the ordering gradually decreases, so errors towards the bottom of the list are penalized less than those at the top. Using (15) and some manipulation, it can be shown that the average distance given r equals

$$E\left[d_w(X,Y)|r\right] = \sum_{i=1}^{m} \sum_{h=1}^{r_i - r_{i-1} - 2} h\left(c(n - r_i + h) + 1\right)\left(H_{r_i - r_{i-1} - 1} - H_h\right)$$
$$= \sum_{i=1}^{m} \binom{r_i - r_{i-1} - 1}{2} \frac{9 + c(9n - 5r_i - 4r_{i-1} - 3)}{18}.$$

To find the optimum values of r_i, we let $r_i = a_i n$ for $0 \leq a_i \leq 1$ and $\delta_i = (a_i - a_{i-1})$. We assume a_i and m are constants and (a_i) is a strictly increasing sequence. The average distortion becomes

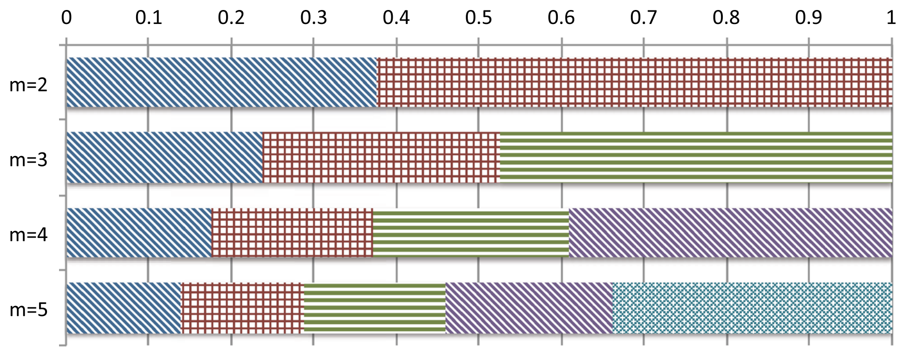

Fig. 6 Optimum positions of pivots derived from solving (18). Each *bar* corresponds to the given value for m. The positions in which a *bar* is divided represent the values of a_i corresponding to positions of the ith pivot

$$E\left[d_w(X,Y)|r\right] = \sum_{i=1}^{m} (a_i - a_{i-1})^2 \left(1 - \frac{5}{9}a_i - \frac{4}{9}a_{i-1}\right)\frac{cn^3}{4} + O\left(n^2\right)$$
$$= \left(\frac{cn^3}{4}\right)\left(1 + O\left(n^{-1}\right)\right)\sum_{i=1}^{m}\delta_i^2\left(1 - \sum_{j=1}^{i}\delta_j + \frac{4}{9}\delta_i\right).$$

Asymptotically, to optimize with respect to r_i, we can find values for $\delta_i, i \in [m]$, that minimize

$$\sum_{i=1}^{m}\delta_i^2\left(1 - \sum_{j=1}^{i}\delta_j + \frac{4}{9}\delta_i\right) \tag{18}$$

such that $\sum_j \delta_j = 1$ and $\delta_j \geq 0$.

Figure 6 illustrates the solutions to this optimization problem obtained numerically for $m = 2, 3, 4, 5$. It is evident in Figure 6 that in higher positions, pivots should be chosen closer to each other to diminish the probability of making mistakes in these positions. As a result, if storage restricted can be relaxed as in Sect. 4.1, then the pivots should no longer be chosen among the first ℓ elements to produce uniform spaces between them, but rather such that they are closer to each other at the top and farther at the bottom.

6 Storing the last *m* elements

In this section, we consider the scenario in which only the last m elements can be stored. This case is especially useful when one needs to learn user preference rankings of a set of objects, e.g., movies, that are presented to the user one by one, and after each item the user is asked to compare it with previous items. Because the number of presented objects may be large, it cannot be expected that the user remembers sufficient information about every object presented in the past. To obtain more reliable information, we may only ask the user to compare the current item with the last $m - 1$ items. Furthermore, this method decreases the number of questions asked from the user. One such application is learning user rankings of movies in streaming services such as Netflix.

For simplicity assume m divides n. We present and analyze the following simple algorithm. The presented algorithm does not use all the information that can be provided by comparison with the last $m-1$ items. However, its simplicity enables analytical analysis of its performance.

Algorithm 2

First sort each group of m consecutive elements in the stream into a block of size m and then produce the output premutation, denoted by Y_2, by interleaving the sorted blocks, where interleaving is defined as follows. For t sequences $a_1, \dots, a_t$, the interleaved sequence $a_1 \star a_2 \star \cdots \star a_t$ is defined as the sequence $a_{11}, \dots, a_{t1}, a_{12}, \dots, a_{t2}, \dots$, where a_{ij} is the jth symbol of a_i. The definition is valid for finite and infinite sequences. Intuition for using interleaving is given after the example below.

Example 4 Same as in 1, suppose $s = (98, 15, 23, 13, 2, 89, 60, 118, 104)$ and, therefore, $X = (5, 4, 2, 3, 7, 6, 1, 9, 8)$. Furthermore, suppose $m = 3$. Sorting the first 3 items gives $s_2 < s_3 < s_1$, corresponding to the sequence 2, 3, 1. The next two blocks of length 3 give $s_5 < s_4 < s_6$ and $s_7 < s_9 < s_8$ corresponding to the sequences $5, 4, 6$ and $7, 9, 8$, respectively. We have $Y_2 = (2, 3, 1) \star (5, 4, 6) \star (7, 9, 8) = (2, 5, 7, 3, 4, 9, 1, 6, 8)$, corresponding to the sorting

$$s' = (15, 2, 60, 23, 13, 104, 98, 89, 118).$$

As a result $d_K(X, Y_2) = 8$ and $d_C(X, Y_2) = 3$.

Interleaving the sorted blocks of length m is motivated by the maximum likelihood estimation of the ranks of the elements of each block among the n elements of s. To see this, consider an element α with rank i in a block of length m. What is the most likely value for the rank of α among the n elements of s? To answer this question, let f_j denote the probability of the event that the rank of α in s equals j. Note that

$$f_j = \binom{n}{m}^{-1} \binom{j-1}{i-1} \binom{n-j}{m-i} \text{ for } i \leq j \leq n-m+i.$$

By investigating the inequality $f_j \geq f_{j-1}$, it can be shown that the maximum value of f_j is attained at $j = \left\lceil n\frac{i-1}{m-1} \right\rceil$. The interleaving method described in Algorithm 2 places α between positions $n\frac{i-1}{m}$ and $n\frac{i}{m}$ and thus this method is compatible with the maximum likelihood estimate of the position of α in s.

Obviously, the results of Sect. 3, namely Theorems 1, 2, and 3, which provide an upper bound on mutual information and lower bounds on storage for a given distortion, are also valid for Algorithm 2. We now study Algorithm 2 regarding its mutual information and average Kendall tau distortion.

The following theorem compares the mutual information $I(X; Y_2)$ between the true permutation and the output produced by the algorithm of this section with the best possible value of mutual information.

Theorem 8 *Assume m divides n and $n, m \to \infty$. For Algorithm 2 with stream memory m, we have*

$$I\left(X; Y_2\right) \sim I\left(X; Y^*\right),$$

where Y^* is the result of an algorithm that maximizes the mutual information between X and Y with stream memory m.

Proof Given Y_2, the number of possible choices for X is

$$\binom{n}{m, \ldots, m} = \frac{n!}{(m!)^{n/m}},$$

where each m elements correspond to one block. Note that the order of the elements of each block in X and Y_2 is the same and so, one only needs to determine the positions of elements of each block. Hence $H\left(X|Y_2\right) = \lg \frac{n!}{(m!)^{n/m}}$ and

$$I\left(X; Y_2\right) = H(X) - H\left(X|Y_2\right) = \frac{n}{m} \lg m! \geq \frac{n}{m} \lg \left(\frac{m}{e}\right)^m = n \lg m - n \lg e$$

which, by Theorem 1, implies that

$$I\left(X; Y^*\right) - I\left(X; Y_2\right) \leq (n - m) \lg e.$$

Hence,

$$\frac{I\left(X; Y_2\right)}{I\left(X; Y^*\right)} \geq 1 + \frac{n \lg e - m \lg e}{n \lg m - m \lg e + O\left(\lg m\right)},$$

where we have again used Theorem 1. If $m, n \to \infty$, we can further simplify this expression as

$$\frac{I\left(X; Y_2\right)}{I\left(X; Y^*\right)} \geq 1 + \frac{O(n)}{n \lg m + O\left(n\right)} = 1 + o\left(1\right).$$

The fact that $\frac{I(X;Y_2)}{I(X;Y^*)} \leq 1$ completes the proof. □

We now turn to the Kendall tau distortion of the algorithm. First, we present two lemmas that will be useful in our analysis.

Lemma 4 *Let T be a discrete random variable with distribution $p_i = P\left[T = i\right]$ and t be a constant. It holds that*

$$E\left[|T - t|\right] \leq \sqrt{E[(T - t)^2]}.$$

Proof We have

$$\begin{aligned}(E\left[|T - t|\right])^2 &= \Big(\sum_i p_i \left|i - t\right|\Big)^2 = \Big(\sum_i \sqrt{p_i}\sqrt{p_i (i - t)^2}\Big)^2 \\ &\leq \sum_i p_i \sum_j p_j \left(j - t\right)^2 = E[(T - t)^2],\end{aligned}$$

where the inequality follows from the Cauchy–Schwarz inequality. □

Lemma 5 *Let P_1 and P_2 be two disjoint sets of size m; let the sequences $a = a_1, \ldots, a_m$, $b = b_1, \ldots, b_m$ be permutations of P_1, and P_2, respectively; and let $c = c_1, \ldots, c_{2m}$ be a randomly chosen permutation of $P_1 \cup P_2$ that satisfies $a_i \prec_c a_j$ and $b_i \prec_c b_j$ for all $i, j \in [m],\ i < j$. For $m \geq 3$,*

$$E[d_K(a \star b, c)] \leq \frac{m^{3/2}}{\sqrt{2}}.$$

Proof Let U_i be a random variable representing the position of b_i in c. From this definition, for the Kendall tau distance between $a \star b$ and c, we have

$$d_K(a \star b, c) = \sum_{i=1}^{m} |U_i - 2i|,$$

and so, by Lemma 4,

$$E[d_K(a \star b, c)] = \sum_{i=1}^{m} E\left[|U_i - 2i|\right] \leq \sum_{i=1}^{m} \sqrt{E[(U_i - 2i)^2]}.$$

We now show that $\sum_{i=1}^{m} \sqrt{E[(U_i - 2i)^2]} \leq \frac{m^{3/2}}{\sqrt{2}}$.

The distribution of U_i is given by

$$P\left(U_i = j\right) = \frac{\binom{j-1}{i-1}\binom{2m-j}{m-i}}{\binom{2m}{m}},$$

and thus

$$\begin{aligned} E\left[U_i\right] &= \binom{2m}{m}^{-1} \sum_j j \binom{j-1}{i-1}\binom{2m-j}{m-i} = i \binom{2m}{m}^{-1} \sum_j \binom{j}{i}\binom{2m-j}{m-i} \\ &= i \binom{2m}{m}^{-1} \binom{2m+1}{m+1} = i \frac{2m+1}{m+1} \end{aligned}$$

and

$$\begin{aligned} E\left[U_i\left(U_i+1\right)\right] &= \binom{2m}{k}^{-1} \sum_j j\left(j+1\right) \binom{j-1}{i-1}\binom{n-j}{k-i} \\ &= i \binom{2m}{m}^{-1} \sum_j \left(j+1\right) \binom{j}{i}\binom{n-j}{m-i} \\ &= i\left(i+1\right) \binom{2m}{m}^{-1} \sum_j \binom{j+1}{i+1}\binom{n-j}{k-i} \end{aligned}$$

$$\begin{aligned}
&= i\,(i+1)\binom{2m}{m}^{-1}\binom{2m+2}{m+2}\\
&= i\,(i+1)\,\frac{(2m+1)\,(2m+2)}{(m+1)\,(m+2)}\\
&= \frac{2i\,(i+1)\,(2m+1)}{(m+2)}.
\end{aligned}$$

Suppose $m \geq 3$. We observe

$$\begin{aligned}
E[(U_i-2i)^2] &= E\,[U_i\,(U_i+1)] - (4i+1)\,EU_i + 4i^2\\
&= \frac{2im(m-i)+2i^2+im}{(m+1)(m+2)}\\
&\leq \frac{m^2(2m+1)^2}{8(m-1)(m+1)(m+2)}\\
&\leq \frac{m}{2},
\end{aligned}$$

where the inequalities follow in a straightforward manner from the facts that $1 \leq i \leq m$ and $m \geq 3$, respectively. Hence,

$$E[d_K(a \star b, c)] = \sum_{i=1}^{m} E\,[|U_i - 2i|] \leq \sum_{i=1}^{m} \sqrt{E[(U_i-2i)^2]} \leq \frac{m^{3/2}}{\sqrt{2}}.$$

□

The next theorem presents an upper bound on the average Kendall tau distortion $E[d_K(X, Y_2)]$ of the algorithm.

Theorem 9 *Assume m divides n. For Algorithm 2 with stream memory m, we have*

$$E[d_K(X, Y_2)] \leq \frac{n\,(n-m)}{2\sqrt{2m}}.$$

Proof Since there are $\binom{n/m}{2}$ pairs of blocks of length m, by Lemma 5, the expected total distortion is bounded as

$$E\,[d_K\,(X, Y_2)] \leq \binom{n/m}{2}\frac{m^{3/2}}{\sqrt{2}} = \frac{n\,(n-m)}{2\sqrt{2m}}.$$

For Algorithm 2, let $\mu = \frac{m}{n}$ and $\delta = E\,[d_K\,(X, Y_2)]\,/n$. Suppose $\delta = \sqrt{n} + O(1)$. From the above theorem,

$$\mu \leq \frac{4d^2 + n - 4d^2\sqrt{1 + n/(2d^2)}}{n} = \frac{5n - 4\sqrt{\frac{3}{2}}n + O(1)}{n} \simeq 0.1 + o(1).$$

On the other hand, the lower bound on μ, from Theorem 2, is

$$\mu \geq \frac{1}{e^2\sqrt{n}}(1+o(1)).$$

Hence, in contrast with Algorithm 1, there is a large gap between the lower and upper bounds on μ for Algorithm 2. Closing this gap requires a better algorithm or improved bounds. □

7 Conclusion

In this paper, we studied the fundamental trade-off between the amount of available storage and the quality of a sorting of a data stream, where the quality of a sorting is measured via permutation distortion metrics. Our results provide lower-bounds on storage for any algorithm with given distortion in the Kendall tau and Chebyshev metrics. Furthermore, we provided an algorithm that asymptotically requires a constant factor as much storage as an optimal algorithm with the same Kendall tau distortion. By considering the special case of $m = 2$, we showed that it is possible to improve the performance of this algorithm by updating its pivots. However, analytical study of the gain in performance for $m > 2$ is still an open problem.

Since applications are usually not run in isolation, it is useful to study performance with resource requirements that may vary through the execution. For this reason, we studied the extension of the algorithm to the case where the storage requirement can be relaxed for parts of the algorithm's execution. Furthermore, we studied the case where the distortion metric is the weighted Kendall tau measure, which can reflect the fact that top of a ranking is typically more important.

Finally, we considered the case in which only information about the last m observed elements is available, to model the problem of learning user preference ranking where the user only remembers recently observed elements. While in this case, we present an algorithm that is optimal with regards to mutual information, further research is required to design algorithms that are optimal with regards to appropriate distortion measures.

Acknowledgments The authors would like to thank Ryan Gabrys and Yue Li for useful discussions and comments. Furthermore, the authors thank anonymous reviewers whose comments greatly improved this paper.

Appendix

Proof of (10)

Lemma 6 *We have*

$$\sum_{k=1}^{n-m+1}\binom{n-k-1}{m-2}k\ln k \leq \binom{n}{m}\left(H_n - H_m + 1 - \frac{m}{n}\right).$$

Proof To prove the upper bound on $\sum_{k=2}^{n-m+1}\binom{n-k-1}{m-2}k\lg k$, we use Abel's identity Apostol (1976, Theorem 4.2), which states that for an arithmetic function a, a real number x, and a function f with a continuous derivative on $[1, x]$, we have

$$\sum_{1\le k\le x} a(k)\, f(k) = A(x)\, f(x) - \int_1^x A(y)\, f'(y)\, dy,$$

where $A(y) = \sum_{1\le k\le y} a(k)$ for $y \in \mathbb{R}$. To use Abel's identity, we let $a(k) = k\binom{n-k-1}{m-2}$ and $f(k) = \ln k$. Hence,

$$\begin{aligned} A(y) &= \sum_{1\le k\le y} k\binom{n-k-1}{m-2} \\ &= \binom{n}{m} - \frac{(\lfloor y\rfloor(m-1)+n)}{m}\binom{n-\lfloor y\rfloor-1}{m-1}, \end{aligned}$$

for $y \ge 1$ and $A(y) = 0$ for $y < 1$. By Abel's identity, we have

$$\sum_{k=1}^{n-m+1}\binom{n-k-1}{m-2}k\ln k = A(n-m+1)f(n-m+1) - \int_{y=1}^{n-m+1} A(y)f'(y)dy.$$

The first term on the right side equals $\binom{n}{m}\ln(n-m+1)$ and for the second term, we find

$$\begin{aligned} &\int_{y=1}^{n-m+1} A(y)f'(y)dy \\ &\quad= \int_{y=1}^{n-m+1}\left(\binom{n}{m} - \frac{(\lfloor y\rfloor(m-1)+n)}{m}\binom{n-\lfloor y\rfloor-1}{m-1}\right)\frac{1}{y}dy \\ &\quad= \binom{n}{m}\ln(n-m+1) - \int_{y=1}^{n-m+1}\frac{(\lfloor y\rfloor(m-1)+n)}{m}\binom{n-\lfloor y\rfloor-1}{m-1}\frac{1}{y}dy. \end{aligned}$$

Hence,

$$\sum_{k=1}^{n-m+1}\binom{n-k-1}{m-2}k\ln k = \int_{y=1}^{n-m+1}\frac{(\lfloor y\rfloor(m-1)+n)}{m}\binom{n-\lfloor y\rfloor-1}{m-1}\frac{1}{y}dy.$$

We proceed as follows:

$$\begin{aligned} &\sum_{k=1}^{n-m+1}\binom{n-k-1}{m-2}k\ln k \\ &\quad\le \int_{y=1}^{n-m+1}\frac{(\lfloor y\rfloor(m-1)+n)}{m}\binom{n-\lfloor y\rfloor-1}{m-1}\frac{1}{\lfloor y\rfloor}dy \end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=1}^{n-m} \frac{k(m-1)+n}{m} \binom{n-k-1}{m-1} \frac{1}{k} \\
&= \frac{m-1}{m} \sum_{k=1}^{n-m} \binom{n-k-1}{m-1} + \frac{n}{m} \sum_{k=1}^{n-m} \binom{n-k-1}{m-1} \frac{1}{k} \\
&= \frac{m-1}{m} \binom{n-1}{m} + \frac{n}{m} \binom{n-1}{m-1} (H_{n-1} - H_{m-1}) \\
&= \frac{m-1}{m} \binom{n-1}{m} + \binom{n}{m} (H_n - H_m) + \frac{1}{m} \binom{n-1}{m-1} \left(\frac{n-m}{m}\right) \\
&= \binom{n-1}{m} + \binom{n}{m} (H_n - H_m) = \binom{n}{m} \left(H_n - H_m + 1 - \frac{m}{n}\right),
\end{aligned}
$$

where we have used the fact that for nonnegative integers ℓ, j, we have

$$
\sum_{i=1}^{\ell-j} \binom{\ell-i}{j} \frac{1}{i} = \binom{\ell}{j} \left(H_\ell - H_j\right),
$$

proved below, to obtain the third equality.

To prove

$$
\sum_{i=1}^{\ell-j} \binom{\ell-i}{j} \frac{1}{i} = \binom{\ell}{j} \left(H_\ell - H_j\right)
$$

let us write it as

$$
\sum_{i=j+1}^{\ell} \frac{(\ell-i+j)!}{(\ell-i)!} \frac{1}{i-j} = \frac{\ell!}{(\ell-j)!} \sum_{i=j+1}^{\ell} \frac{1}{i}. \tag{19}
$$

The proof is by induction. The equality (19) holds for $j = 0$ as both sides reduce to $\sum_{i=1}^{\ell} \frac{1}{i}$. As induction hypothesis, suppose (19) holds for a certain value of j. We show that it also holds for $j+1$. We have

$$
\begin{aligned}
&\sum_{i=j+2}^{\ell} \frac{(\ell-i+j+1)!}{(\ell-i)!} \frac{1}{i-j-1} \\
&= \sum_{i=j+1}^{\ell-1} \frac{(\ell-i+j)!}{(\ell-i-1)!} \frac{1}{i-j} \\
&= \sum_{i=j+1}^{\ell} \frac{(\ell-i)\,(\ell-i+j)!}{(\ell-i)!} \frac{1}{i-j} \\
&= \sum_{i=j+1}^{\ell} \frac{(\ell-j)\,(\ell-i+j)!}{(\ell-i)!} \frac{1}{i-j} - \sum_{i=j+1}^{\ell} \frac{(i-j)\,(\ell-i+j)!}{(\ell-i)!} \frac{1}{i-j}
\end{aligned}
$$

$$
\begin{aligned}
&= (\ell - j) \sum_{i=j+1}^{\ell} \frac{(\ell - i + j)!}{(\ell - i)!} \frac{1}{i - j} - \sum_{i=j+1}^{\ell} \frac{(\ell - i + j)!}{(\ell - i)!} \\
&\overset{\text{(a)}}{=} (\ell - j) \frac{\ell!}{(\ell - j)!} \sum_{i=j+1}^{\ell} \frac{1}{i} - j! \sum_{i=j+1}^{\ell} \binom{\ell - i + j}{j} \\
&= \frac{\ell!}{(\ell - j - 1)!} \sum_{i=j+1}^{\ell} \frac{1}{i} - j! \sum_{i=j}^{\ell-1} \binom{i}{j} \\
&= \frac{\ell!}{(\ell - j - 1)!} \sum_{i=j+1}^{\ell} \frac{1}{i} - \frac{\ell!}{(j+1)(\ell - j - 1)!} \\
&= \frac{\ell!}{(\ell - j - 1)!} \sum_{i=j+2}^{\ell} \frac{1}{i},
\end{aligned}
$$

where for (a) we have used the induction hypothesis. □

Proof of (14)

In this subsection, we prove (14) as follows

$$
\begin{aligned}
&P\left(D_{u,v}\left(X, Y_1''\right)\right) \\
&\overset{\text{(a)}}{=} \frac{1}{t} \sum_{a=0}^{\lfloor (3t-2)/4 \rfloor} \frac{\binom{a+2}{2} + \binom{t-a+1}{2}}{2\binom{t+2}{2}} + \frac{1}{t} \sum_{a=\lfloor (3t-2)/4 \rfloor + 1}^{t-1} \frac{\binom{a+2}{3} + \binom{t+2}{3} - \binom{t-a+2}{3}}{2a\binom{t+2}{2}} \\
&= \frac{1}{t^3 + O\left(t^2\right)} \Bigg[\sum_{a=0}^{\lfloor (3t-2)/4 \rfloor} \left(\binom{a+2}{2} + \binom{t-a+1}{2} \right) \\
&\quad + \sum_{a=\lfloor (3t-2)/4 \rfloor + 1}^{t-1} \frac{1}{a} \left(\binom{a+2}{3} + \binom{t+2}{3} - \binom{t-a+2}{3} \right) \Bigg] \\
&= \frac{1}{t^3 + O\left(t^2\right)} \Bigg[\frac{(3t/4)^3}{6} + \frac{t^3 - (t/4)^3}{6} + \frac{t^3 - (3t/4)^3}{18} + O\left(t^2\right) \\
&\quad + \sum_{a=\lfloor (3t-2)/4 \rfloor + 1}^{t-1} \frac{t^3 - (t-a)^3 + O\left(t^2\right)}{6a} \Bigg] \\
&= \frac{1}{1 + O\left(t^{-1}\right)} \left(\frac{307}{1152} + \frac{1}{6} \sum_{\lfloor (3t-2)/4 \rfloor + 1}^{t-1} \left(\frac{3}{t} - \frac{3a}{t^2} + \frac{a^2}{t^3} + O\left(\frac{1}{at}\right) \right) \right) \\
&= \frac{1}{1 + O\left(t^{-1}\right)} \left(\frac{307}{1152} + \frac{1}{6} \left(\frac{3}{4} - \frac{21}{32} + \frac{37}{192} + O\left(\frac{1}{t}\right) \right) \right) \\
&= \frac{181}{576} + o(1),
\end{aligned}
$$

where (a) follows from (13) and where we have used $\sum_{a=0}^{k+O(1)} \binom{a+O(1)}{2} = \frac{k^3}{6} + O\left(k^2\right)$ and $\frac{1}{a}\binom{a+2}{3} = \frac{1}{3}\binom{a+2}{2}$.

References

Apostol TM (1976) Introduction to analytic number theory. Springer, New York

Babcock B, Babu S, Datar M, Motwani R, Widom J (2002) Models and issues in data stream systems. In: Proceedings of 21st ACM symposium on principles of database systems (PODS), New York

Carterette B (2009) On rank correlation and the distance between rankings. In: Proceedings of 32nd international SIGIR conference on research and development in information retrieval, ACM Press, New York, pp 436–443

Chakrabarti A, Jayram TS, Pătraşcu M (2008) Tight lower bounds for selection in randomly ordered streams. In: ACM-SIAM symposium on discrete algorithms (SODA), Society for Industrial and Applied Mathematics, Philadelphia, pp 720–729

Chen CP, Qi F (2008) The best lower and upper bounds of harmonic sequence. Glob J Appl Math Math Sci 1(1):41–49

Corless RM, Gonnet GH, Hare DEG, Jeffrey DJ, Knuth DE (1996) On the Lambert W function. Adv Comput Math 5(1):329–359. doi:10.1007/BF02124750

Cover TM, Thomas JA (2006) Elements of information theory. Wiley, New York

Diaconis P (1988) Group representations in probability and statistics, vol 11. Institute of Mathematical Statistics, Hayward

Farnoud F, Milenkovic O (2013) Aggregating rankings with positional constraints. In: Proceedings of IEEE information theory workshop (ITW), Seville

Farnoud F, Schwartz M, Bruck J (2014a) Rate-distortion for ranking with incomplete information. arXiv preprint: http://arxiv.org/abs/1401.3093

Farnoud F, Schwartz M, Bruck J (2014b) Bounds for permutation rate-distortion. In: Proceedings of IEEE international symposium on information theory (ISIT), Honolulu

Greenwald M, Khanna S (2001) Space-efficient online computation of quantile summaries. In: Proceedings of ACM SIGMOD international conference on management of data, ACM, New York, pp 58–66. doi:10.1145/375663.375670

Hassanzadeh F (2013) Distances on rankings: from social choice to flash memories. Ph.D. thesis, University of Illinois at Urbana–Champaign. http://hdl.handle.net/2142/44268

Holst L (1980) On the lengths of the pieces of a stick broken at random. J Appl Probab 17(3):623–634

Kemeny JG (1959) Mathematics without numbers. Daedalus 88(4):577–591

Kumar R, Vassilvitskii S (2010) Generalized distances between rankings. In: Proceedings of 19th international world wide web conference, Raleigh, pp. 571–580

Manku GS, Rajagopalan S, Lindsay BG (1998) Approximate medians and other quantiles in one pass and with limited memory. In: Proceedings of ACM SIGMOD international conference on management of data, ACM, New York, pp 426–435. doi:10.1145/276304.276342

McGregor A, Valiant P (2012) The shifting sands algorithm. In: ACM-SIAM symposium on discrete algorithms (SODA), SIAM, pp 453–458. http://www.dl.acm.org/citation.cfm?id=2095116.2095155

Munro J, Paterson M (1980) Selection and sorting with limited storage. Theor Comput Sci 12(3):315–323. http://www.sciencedirect.com/science/article/pii/0304397580900614

Sedgewick R, Wayne K (2011) Algorithms, 4th edn. Addison-Wesley Professional, Reading

Shieh GS (1998) A weighted Kendall's tau statistic. Stat Probab Lett 39(1):17–24

Yilmaz E, Aslam JA, Robertson S (2008) A new rank correlation coefficient for information retrieval. In: Proceedings of 31st annual international SIGIR conference research and development in information retrieval, ACM, New York, pp 587–594