Algorithmica (2017) 79:960–983
DOI 10.1007/s00453-017-0295-3

Partial Sorting Problem on Evolving Data

Qin Huang[1,5] · **Xingwu Liu**[2,3,4] · **Xiaoming Sun**[4,5] · **Jialin Zhang**[4,5]

Received: 17 February 2016 / Accepted: 20 February 2017 / Published online: 10 March 2017

Abstract In this paper we investigate the top-k-selection problem, i.e. to determine and sort the top k elements, in the dynamic data model. Here *dynamic* means that the underlying total order evolves over time, and that the order can only be probed by pairwise comparisons. It is assumed that at each time step, only one pair of elements can be compared. This assumption of restricted access is reasonable in the dynamic model, especially for massive data sets where it is impossible to access all the data before the next change occurs. Previously only two special cases were studied (Anagnostopoulos et al. in 36th international colloquium on automata, languages and programming (ICALP). LNCS, vol 5566, pp 339–350, 2009) in this model: selecting the element of a given rank, and sorting all elements. This paper systematically deals with $1 \leq k \leq n$. Specifically, we identify the critical point k^* such that the top-k-selection problem can be solved error-free with probability $1 - o(1)$ if and only if $k = o(k^*)$. A lower bound

✉ Xingwu Liu
liuxingwu@ict.ac.cn

Qin Huang
huangqinscu@gmail.com

Xiaoming Sun
sunxiaoming@ict.ac.cn

Jialin Zhang
zhangjialin@ict.ac.cn

1 State Key Laboratory of Software Development Environment, Beihang University, Beijing, China

2 Research Institute of Beihang University in Shenzhen, Shenzhen, China

3 Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

4 University of Chinese Academy of Sciences, Beijing, China

5 CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

of the error when $k = \Omega(k^*)$ is also determined, which actually is tight under some conditions. In contrast, we show that the top-k-set problem, which means finding the top k elements without sorting them, can be solved error-free with probability $1-o(1)$ for all $1 \leq k \leq n$. Additionally, we consider some extensions of the dynamic data model and show that most of these results still hold.

1 Introduction

Sorting, a fundamental primitive in algorithms, has been an active research topic in computer science for decades. In the era of big data, it is the cornerstone of numerous vital applications—Web search, online ads, and recommendation systems to name but a few. While sorting has been extensively studied, little is known when the data is dynamic. Actually, dynamic data is common in practical applications: the linking topology of Web pages, the friendship network of Facebook, the daily sales of Amazon, and so on, all keep changing. The basic challenge in dealing with dynamic, massive data is that the access to the data is too restricted to catch the changes. For example, it is impossible to get an exact snapshot of Web, and a third-party vendor can query the Facebook network only via a rate-limited API. As a result, this paper is devoted to studying the sorting problem on dynamic, access-restricted data.

In the seminal paper [1], Anagnostopoulos et al. formulated a model for dynamic data as follows. Given a set U of n elements, at every discrete time t, there is an underlying total order π^t on U. For every $t \geq 1$, π^t is obtained from π^{t-1} by sequentially swapping α random pairs of consecutive elements, where α is a constant number. The only way to probe π^t is querying the relative rank of ONE pair of elements in U at every time step. The goal is to learn about the true order π^t. Obviously, it is impossible to always exactly find out the orders, so our objective is that at any time t, the algorithm estimates the correct answer (or an approximate answer) with high probability. In this paper, "with high probability" and "with probability $1-o(1)$" are used interchangeably.

Anagnostopoulos et al. [1] proved that the Kendall tau distance between π^t and $\tilde{\pi}^t$, defined in Sect. 2 and denoted by $\mathrm{KT}(\pi^t, \tilde{\pi}^t)$, is lower-bounded by $\Omega(n)$ with high probability at every t, where $\tilde{\pi}^t$ is the order estimated by any algorithm at time t. This lower bound is nearly tight, since they proposed an algorithm with $\mathrm{KT}(\pi^t, \tilde{\pi}^t) = O(n \ln \ln n)$. Furthermore, they designed an algorithm that with high probability, exactly identifies the element of a given rank.

Though elegant, this model is too restricted: the evolution is extremely slow since α is constant, and is extremely local since only consecutive elements are swapped. Hence, it is extended in this paper by allowing α to be a function of n, and is called the consecutive-swapping model. We further generalize it to the Gaussian-swapping model by relaxing the locality condition.

Inspired by [1], we study the general top-k-selection problem: at every time t, figure out the top k elements and sort them, where $k \in [n] \triangleq \{1, 2, \ldots n\}$. Its two extreme cases, namely $k = n$ and $k = 1$, correspond to the sorting problem and the selection problem in [1], respectively. The error-free solvability of the selection problem suggests that the error in solving the top-k-selection problem may vanish as k

Table 1 Results in the consecutive-swapping model

k	$X \triangleq \mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t)$
$o\left(\sqrt{\frac{n}{\alpha}}\right)$	$\Pr(X=0) = 1 - o(1)$
$\Theta\left(\sqrt{\frac{n}{\alpha}}\right)$	$\Pr(X=0) = \Theta(1), \Pr(X>0) = \Theta(1)$
$\omega\left(\sqrt{\frac{n}{\alpha}}\right)$	$\Pr\left(X = O\left(\frac{k^2\alpha}{n}\right)\right) = 1 - o(1)^{\dagger}$

In † case, this upper bound of X is tight for constant α. See Sect. 3

decreases, so it is natural to investigate the critical point where the error vanishes and to find the optimal solution beyond the critical point. Another motivation lies in the wide application of top-k-selection, also known as partial sorting. It has been used in a variety of areas such as Web and multimedia search systems and distributed systems, where massive data has to be dealt with efficiently [2,3].

Additionally, we consider a closely related top-k-set problem: at every time t, identify the set of the top k elements. The top-k-set problem is weaker in that it does not require to sort the elements. In the static data setting, when a selection algorithm identifies the kth element, it automatically determines the set of the top k elements (see for example Knuth's book [4]). However, this is not apparent in the dynamic data model.

1.1 Our Contributions

The main results of this paper lie in two aspects in the consecutive-swapping model. First, it is shown that the top-k-set problem can be solved error-free with high probability for any $k \in [n]$. Second and more important, $k^* = \Theta(\sqrt{\frac{n}{\alpha}})$ is proven to be the critical point of k for the top-k-selection problem, which means that this problem can be solved error-free with high probability if and only if $k = o(k^*)$.

In addition, for k beyond k^*, we obtain tight lower bounds of $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t)$, where π_k^t is the true order over the top k elements at time t and $\tilde{\pi}_k^t$ is the algorithmically estimated order over the estimated top k elements at time t. Specifically, if $k = \Omega(\sqrt{\frac{n}{\alpha}})$, then for any algorithm and $t > k$, $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) \neq 0$ with constant probability. When $k = \omega(\sqrt{n})$ and $\alpha = O(1)$, for any algorithm and $t > k/8$, $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = \Omega(\frac{k^2}{n})$ with high probability. These lower bounds can be reached by Algorithm 2 with parameter k, hence being tight.

The results of the top-k-selection problem in the consecutive-swapping model are summarized in Table 1. Most of the results are also generalized to the Gaussian-swapping model with constant α, as summarized in Table 2.

1.2 Related Work

The sorting/selection problem has been actively investigated for decades [2,5–7], but the study of this problem in dynamic data setting was initiated very recently [1]. In [1], Anagnostopoulos et al. considered two special cases of the top-k-selection problem,

Table 2 Results in the Gaussian-swapping model

k	$X \triangleq \mathrm{KT}\left(\tilde{\pi}_k^t, \pi_k^t\right)$
$o\left(\frac{\sqrt{n}}{\ln^{0.25} n}\right)$	$\Pr(X = 0) = 1 - o(1)$
$\Theta\left(\frac{\sqrt{n}}{\ln^{0.25} n}\right)$	$\Pr(X = 0) = \Theta(1)$
$\omega\left(\frac{\sqrt{n}}{\ln^{0.25} n}\right)$	$\Pr\left(X = O\left(\frac{k^2 \ln n}{n}\right)\right) = 1 - o(1)$

namely $k = n$ and $k = 1$, in the consecutive-swapping model with constant α. Their work has inspired the problem and the data model in this paper. The theoretical results in [1] were experimentally verified by Moreland [8] in 2014.

Dynamic data is also studied in the graph setting. [9] considered two classical graph connectivity problems (path connectivity and minimum spanning trees) where the graph keeps changing over time and the algorithm, unaware of the changes, probes the graph to maintain a path or spanning tree. Bahmani et al. [10] designed an algorithm to approximately compute the PageRank of evolving graphs. Zhuang et al. [11] considered the influence maximization problem in dynamic social networks. Kanade et al. [12] investigated the stable matching problem in the context that the preferences are evolving. On the other hand, Labouseur et al. [13] and Ren [14] dealt with the data structure and management issues, respectively, enabling efficient query processing for dynamic graphs.

It is worth noting that our dynamic data model is essentially different from noisy information or fault tolerance models [15–18]. In those models, the main difficulty is brought about by misleading or faulty information. On the contrary, in our model, the query results are correct, while the difficulty comes from the restricted access to the dynamic data. The ground truth can be probed only by local observation, so it is impossible to capture all changes in the data. The key issue is to choose query strategies in order to approximate the real data with high probability.

In the algorithm community, there are many other models dealing with dynamic and uncertain data, from various points of view. However, none of them captures the two crucial aspects of our dynamic data model: the underlying data keeps changing, and the data exposes limited information to the algorithm by probing. For example, temporal graphs [19] model graphs that change over time, but all the data is fully known; data stream algorithms [20] deal with a stream of data, typically with limited space, but the algorithms can observe the entire data that has arrived; local algorithms on graphs [21,22] probe the underlying graphs by a limited number of query, but typically the graphs are static; in online algorithms [23], though the data comes over time and is processed without knowledge of the future data, the algorithms know all the data up to now; the multi-armed-bandit model [24] tends to optimize the total gain in a finite exploration–exploitation process, while our framework concerns the performance of the algorithm at every time step in an infinite process.

The rest of the paper is organized as follows. In Sect. 2, we provide the formal definition of the models and formulate the problems. Section 3 is devoted to solving the top-k-set problem and the top-k-selection problem in the consecutive-swapping model. In Sect. 4, the problems are studied in the Gaussian-swapping model. Section 5 concludes the paper.

2 Preliminaries

We now formalize our dynamic data model.

Let U be a set of n elements, and $\mathcal{U}$ be the set of all total orders over U, that is, $\mathcal{U} = \{\pi : U \to [n] \mid \forall u \neq v \in U, \pi(u) \neq \pi(v)\}$, where $[n] \triangleq \{1, 2, \ldots n\}$. For any $\pi \in \mathcal{U}$, $k \in [n]$, and $u \in U$, we call $\pi^{-1}(k)$ the *k-th element* relative to π and $\pi(u)$ the *rank* of u relative to π. If $\pi(u) < \pi(v)$, we say $u >_\pi v$ or simply $u > v$ when π can be inferred from context.

In this paper, we consider the process where the order on U gradually changes over time. Time is discretized into steps sequentially numbered by nonnegative integers. At every time step t, there is an underlying total order π^t on U. For every $t \geq 1$, π^t is obtained from π^{t-1} by sequentially swapping α random pairs of consecutive elements, where α is an integer function of n. This is our consecutive-swapping model.

Now we introduce the Gaussian-swapping model whose defining feature is that non-consecutive pairs can be swapped in the evolution. Specifically, for every $t \geq 1$, π^t is still obtained from π^{t-1} by sequentially swapping α pairs of elements. However, each pair (not necessarily consecutive) is selected as follows, rather than uniformly randomly. First, d is sampled from a truncated Gaussian distribution $\Pr(d) = \beta e^{\frac{-d^2}{2}}$, $d \in [n-1]$, where β is the normalizing factor. Then, a pair of elements whose ranks differ by d is chosen uniformly randomly from all such pairs. Thus, the probability that a pair (u, v) gets swapped is $\frac{\beta e^{\frac{-d^2}{2}}}{n-d}$, where $d = |\pi^{t-1}(u) - \pi^{t-1}(v)|$.

In either model, at any time step t, the changes of π^t are unknown by the algorithm running on the data. The only way to probe the underlying order is by comparative queries. At any time t, the algorithm can chose an arbitrary pair of elements $u, v \in U$ and query whether $\pi^t(u) > \pi^t(v)$ or not. At most *one* pair of elements can be queried at each time step.

Now we define $\mathcal{I}$-sorting problem for any index set $\mathcal{I} \subseteq [n]$: at each time step t, find out all the elements whose ranks belong to $\mathcal{I}$, and sort them according to π^t. The concept of $\mathcal{I}$-sorting problem unifies both the sorting problem ($|\mathcal{I}| = n$) and the selection problem ($|\mathcal{I}| = 1$). This paper mainly studies the top-k-selection problem, also known as the partial sorting problem, which is the special case of the $\mathcal{I}$-sorting problem with $\mathcal{I} = [k]$ for $k \in [n]$. A closely-related problem, called the top-k-set problem, is also studied. It requires to find out $(\pi^t)^{-1}([k])$ at each time t, without sorting them.

We then define the performance metrics of the algorithms. In the top-k-set problem, we want to maximize the probability that the output set is exactly the same as the true set for sufficiently large t. In the top-k-selection problem, we try to minimize the Kendall tau distance between the estimated order and the true order on the top k elements, for sufficiently large t. Since an algorithm solving the top-k-selection problem may output an order on a wrong set, we extend the definition of Kendall tau distance to orders on equal-size different sets. Specifically, given total orders σ on set V and δ on set W with $|V| = |W|$, their Kendall tau distance is defined to be $\mathrm{KT}(\sigma, \delta) = |\{(x, y) \in V^2 : \sigma(x) < \sigma(y) \text{ and } (x \notin W \text{ or } y \notin W \text{ or } \delta(x) > \delta(y))\}|$.

Intuitively, it is the number of pairs that either are not shared by V and W or are ordered inconsistently by the two total orders.

Throughout this paper, one building block of the algorithms is the randomized quick-sort algorithm. We describe it briefly. Given an array, it works as follows: (1) Uniformly randomly pick an element, called a pivot, from the array. (2) Compare all elements with the pivot, resulting in two sub-arrays: one consisting of all the elements smaller than the pivot, and the other consisting of the other elements except the pivot. (3) Recursively apply steps 1 and 2 to the two sub-arrays until all the sub-arrays are singletons.

3 Consecutive-Swapping Model

In this section, we consider the top-k-set problem and the top-k-selection problem in the consecutive-swapping model. For the top-k-set problem, Sect. 3.1 shows an algorithm which is error-free with probability $1 - o(1)$ for arbitrary k. Section 3.2 is devoted to the top-k-selection problem. It presents an algorithm that is optimal when α is constant or k is small.

3.1 An Algorithm for the Top-k-Set Problem

The basic idea is to repeatedly run quick-sort over the data U, extract the set of the top k elements from the resulting order, and output this set during the next run. But an issue should be addressed: since the running time of quick-sort is $\Omega(n \ln n)$ with high probability, the set of the top k elements will change with high probability during the next run, leading to out-of-date outputs. Because the rank of any element does not change too much during the next run of quick-sort, a solution is to sort in parallel a small subset of U that contains the top k elements with high probability.

Algorithm 1 Top-k-set

Input: A set U of n elements
Output: $\widetilde{T}$
1: Initialize $\tilde{\pi}$, L, C, $\tilde{\pi}_C$, and $\widetilde{T}$ arbitrarily
2: **while** (true) **do**
3: **Execute in odd steps:** /*QS_1*/
4: $\tilde{\pi} \leftarrow$ quick_sort(U)
5: $L \leftarrow \tilde{\pi}^{-1}([k - c\alpha \ln n])$ and $C \leftarrow \tilde{\pi}^{-1}([k + c\alpha \ln n]) \setminus L$ /*The constant c will be determined in the proof of Theorem 1*/
6: **Execute in even steps:** /*QS_2*/
7: $\tilde{\pi}_C \leftarrow$ quick_sort(C)
8: $\widetilde{T} \leftarrow L \bigcup \tilde{\pi}_C^{-1}([c\alpha \ln n])$
9: **end while**

Specifically, the algorithm Top-k-set consists of two interleaving procedures (denoted by QS_1 and QS_2, respectively), each of which restarts once it terminates. In the odd steps, QS_1 calls quick-sort to sort U, preparing two sets L and C. The set L

consists of the elements that will remain among top k during the next run of QS_1 with high probability, while C contains the uncertain elements that might be among top k in this period. Then, QS_2 will sort the set C computed by the last run of QS_1 to produce the estimated set of top k elements. At any time t, the output $\widetilde{T}_t$ of the algorithm is the set $\widetilde{T}$ computed by the previous run of QS_2.

Theorem 1 shows that Algorithm 1 is error-free with high probability.

Theorem 1 *Assume that* $\alpha = o(\frac{\sqrt{n}}{\ln n})$. *For any* $k \in [n]$, $\Pr(\widetilde{T}_t = (\pi^t)^{-1}([k])) = 1 - o(1)$, *where* $\widetilde{T}_t$ *is the output of Algorithm 1 at time* t, π^t *is the true order on* U *at time* t, *and* t *is sufficiently large.*

The basic idea of the proof lies in two aspects. First, with high probability, the estimated rank of every element with respect to $\tilde{\pi}$ in Line 4 is at most $O(\alpha \ln n)$ away from the true rank, implying that all the elements in L are among top k and all top k elements are in $L \bigcup C$. Second, with high probability, the kth element of U does not swap throughout sorting C, so the set of top k elements remains unchanged and is exactly contained in $\widetilde{T}$.

Two lemmas are needed in proving Theorem 1.

Lemma 1 *For any* $\alpha = o(n)$, *the running time of the randomized quick-sort algorithm in the consecutive-swapping model is* $O(n \ln n)$ *in expectation and with probability* $1 - O(n^{-3})$.

Proof This proof is inspired by the proof of [1, Proposition 3] and the proof of the time complexity of the randomized quick-sort algorithm [25].

We first recall the main steps in proving that the time complexity of the randomized quick-sort on static data is $O(n \ln n)$ in expectation and with probability $1 - O(n^{-3})$.

1. A pivot is said to be good if it divides the array of size s into two sub-arrays each having at least γs elements, where $0 < \gamma < 0.5$ is a constant. Thus the number of good pivots along a given path from the root to a leaf in the quick-sort execution tree is $O(\ln n)$.
2. In a given path, a pivot is good with constant probability and bad (i.e., not good) also with constant probability. By using Chernoff bound, the length of a given path is $O(\ln n)$ with probability at least $1 - O(n^{-4})$.
3. By union bound, the lengths of all the paths are $O(\ln n)$ with probability $1 - O(n^{-3})$. Therefore, the running time is $O(n \ln n)$ with probability $1 - O(n^{-3})$ and in expectation.

In the consecutive-swapping model, since the true order keeps evolving, a pivot u that is good at the time it is chosen as pivot might divide the array of size s into two parts one of which has fewer than γs elements. Let X be the number of swaps involving u while it is playing the role of a pivot. We have $\mathbb{E}[X] = O(\frac{\alpha s}{n})$. By Markov's inequality, $\Pr(X \geq \frac{\gamma s}{2}) \leq O(\frac{\alpha}{n})$. Hence, with probability $1 - O(\frac{\alpha}{n})$, each part will contain at least $\frac{\gamma s}{2}$ elements.

As a result, in the consecutive-swapping model, we redefine the term *good pivot* to be a pivot which divides the corresponding array of size s into two sub-arrays both having at least $\frac{\gamma s}{2}$ elements. If $\alpha = o(n)$, a pivot is good with at least a constant

probability β. In any given path of length L, the number of good pivots is $O(\ln n)$, and the expected number of bad pivots is at most $(1-\beta)L$, which is at most $\frac{1-\beta}{\beta}$ times the expected number of good pivots and is $O(\ln n)$. Thus by Chernoff bound, the number of bad pivots is also $O(\ln n)$ with probability $1 - O(n^{-3})$. Therefore, steps 2 and 3 hold. □

Lemma 2 *In the consecutive-swapping model with $\alpha = o(n)$, consider a run of the randomized quick-sort algorithm from time t_0 to t_1. For any $u \in U$, the number of elements that are incorrectly ordered with respect to u is $O(\alpha \ln n)$ with probability $1 - O(\frac{1}{n^3})$. Additionally, with probability $1 - O(\frac{1}{n^2})$, $|\pi^{t_1}(u) - \tilde{\pi}^{t_1}(u)| = O(\alpha \ln n)$ for all $u \in U$.*

Proof This proof is similar to that of [1, Lemma 6].

Arbitrarily fix a $u \in U$. We first consider the set $S = \{v \in U | u <_{\pi^{t_1}} v, v <_{\tilde{\pi}^{t_1}} u\}$. Of course, $S = S_1 \bigcup S_2$, where $S_1 = \{v \in S | \exists t \in [t_0, t_1), v <_{\pi^t} u\}$ and $S_2 = \{v \in S | \forall t \in [t_0, t_1], u <_{\pi^t} v\}$.

$|S_1|$ is bounded by Z_1, the number of swaps involving u. Since at each time step, u was chosen to swap with probability at most $\frac{2\alpha}{n}$ and the running time is bounded by $O(n \ln n)$ in expectation and with probability $1 - o(1)$, $\mathbb{E}[Z_1] = O(\alpha \ln n)$. By Chernoff bound, $\Pr(Z_1 \le c_1 \alpha \ln n) = 1 - O(\frac{1}{n^3})$ for some constant c_1. Thus, $\Pr(|S_1| \le c_1 \alpha \ln n) = 1 - O(\frac{1}{n^3})$.

For any $v \in S_2$, there exists a pivot w which incorrectly places u and v. Specifically, when w is compared with u, $w < u$ (by the true order at that time), but when it is compared with v, $v < w$. We can see that w must swap with v while it is the pivot.

$|S_2|$ is thus bounded by Z_2, the number of swaps that involve any pivot which is compared with u. Let X_w be the number of steps that an element w is a pivot and Y_w be the number of swaps that involve w while it is a pivot. Let Q_u be the set of all the pivots which are compared with u. Then $Z_2 = \sum_{w \in Q_u} Y_w$. By Lemma 1, we have

$$\mathbb{E}\left[\sum_{w \in Q_u} X_w\right] = O(n \ln n).$$

Since Y_w follows the binomial distribution $B(X_w, 2\alpha/n)$, we have

$$\mathbb{E}[Z_2] = \mathbb{E}\left[\mathbb{E}\left[\sum_{w \in Q_u} Y_w | X_w\right]\right] = O(\alpha \ln n).$$

By Chernoff bound, $\Pr(Z_2 \le c_2 \alpha \ln n) = 1 - O(\frac{1}{n^3})$ for some constant c_2. Hence, $\Pr(|S_2| \le c_2 \alpha \ln n) = 1 - O(\frac{1}{n^3})$.

Altogether, $\Pr(|S| \le c_3 \alpha \ln n) = 1 - O(\frac{1}{n^3})$, where $c_3 = c_1 + c_2$. Likewise, we also have $\Pr(|S'| \le c_4 \alpha \ln n) = 1 - O(\frac{1}{n^3})$, where $S' = \{v \in U | v <_{\pi^{t_1}} u, u <_{\tilde{\pi}^{t_1}} v\}$ and c_4 is a constant. The first part of the lemma thus holds.

Now prove the second part of the lemma. By union bound, $|\pi^{t_1}(u) - \tilde{\pi}^{t_1}(u)| \leq c\alpha \ln n$ holds for every $u \in U$ with probability at least $1 - O(\frac{1}{n^2})$, where $c = c_3 + c_4$. □

Proof (of Theorem 1) Let $\mathcal{R}$ stand for the run of QS_1 which starts at t_0 and ends at t_1, and $\mathcal{R}'$ for the run of QS_1 exactly preceding $\mathcal{R}$. Applying Lemma 2 to $\mathcal{R}'$, we have that with probability $1 - O(\frac{1}{n^2})$, $|\pi^{t_0}(u) - \tilde{\pi}^{t_0}(u)| \leq c_1\alpha \ln n$ for every $u \in U$ and some constant c_1. For the run $\mathcal{R}$, by Lemma 1, we have $t_1 - t_0 = O(n \ln n)$ in expectation and with probability $1 - o(1)$. During $[t_0, t_1]$, the rank of any element u changes less than $c_2\alpha \ln n$ with probability $1 - O(\frac{1}{n^2})$ for some constant c_2. Hence, with probability $1-o(1)$, we have $|\pi^t(u)-\tilde{\pi}^{t_0}(u)| \leq (c_1+c_2)\alpha \ln n$ for all $t \in [t_0, t_1]$ and all $u \in U$. Let $c = c_1 + c_2$.

Note that L contains all the elements u such that $\tilde{\pi}^{t_0}(u) \leq k - c\alpha \ln n$. Then with probability $1 - o(1)$, we have $\pi^t(u) \leq k$ for all $t \in [t_0, t_1]$ and all $u \in L$. Consider the set $R = U\backslash(L \bigcup C) = \{v : \tilde{\pi}^{t_0}(v) \geq k + c\alpha \ln n + 1\}$. Then with probability $1 - o(1)$, we have $\pi^t(v) > k$ for all $t \in [t_0, t_1]$ and all $v \in R$. Therefore, with probability $1 - o(1)$, $L \bigcup C$ contains every element which can be among top k at some $t \in [t_0, t_1]$.

In line 7, QS_2 requires time $O(\alpha \ln n(\ln \alpha + \ln \ln n))$ to sort C, with probability $1-o(1)$ (by Lemma 1). If the element of true rank k (say $\tilde{u}$) does not swap throughout the run of QS_2, the algorithm can return the correct set of the top k elements at the end of the run of QS_2. This is because in the consecutive-swapping model, if $\tilde{u}$ does not swap, then for any $u \in U$, the sign of $\pi^t(\tilde{u}) - \pi^t(u)$ does not change. The probability that $\tilde{u}$ swaps during the run of QS_2 is bounded by $O(\frac{\alpha^2 \ln n(\ln \alpha + \ln \ln n)}{n})$. Hence, if $\alpha = o(\frac{\sqrt{n}}{\ln n})$, the algorithm can return the correct set of the top k elements with probability $1 - o(1)$. During the next run of QS_2, the set of the top k elements does not change with probability $1 - O(\frac{\alpha^2 \ln n(\ln \alpha + \ln \ln n)}{n}) = 1 - o(1)$, implying the result. □

3.2 An Algorithm for the Top-k-Selection Problem

Now we present an algorithm to solve the top-k-selection problem. The basic idea is to repeatedly run quick-sort over the data U, extracting a small subset that includes all the elements that can be among top k during the next run. To exactly identify the top k elements in order, the small set is sorted and the order of the top k elements is produced accordingly. Like in designing the top-k-set algorithm, there is also an issue to address: since sorting the small set takes time $\Omega(k \ln k)$, the order of the top k elements will soon become out of date. Again note that with high probability the rank of each element does not change too much during sorting the small set, so the order of the top k elements can be regulated locally and keeps updated.

Specifically, Algorithm 2 consists of four interleaving procedures (QS_1, QS_2, QS_3, and Local-sort), each of which restarts once it terminates. At the $(4t + 1)$-st time steps, QS_1 invokes a quick-sort on U, preparing a set C of size $k + O(\alpha \ln n)$ which with high probability, contains all the elements among top k during the next run of QS_1. At the $(4t + 2)$-nd time steps, QS_2 calls another quick-sort on the latest C computed

by QS_1, producing a set P of size k. With high probability, the set P exactly consists of the top k elements of U during the next run of QS_2. At the $(4t+3)$-rd time steps, the other quick-sort is invoked by QS_3 on the latest P computed by QS_2, periodically updating the estimated order over P. The resulting order is actually close to the true order over P during the next run of QS_3. Finally, at the $(4t)$-th time steps, an algorithm Local-sort is executed on the total order over P that is produced by the last run of QS_3, so as to locally regulate the order. At any time t, the output $\tilde{\pi}_k^t$ of Algorithm 2 is the last $\tilde{\pi}_k$ computed by Local-sort.

Algorithm 2 Top-k-selection

Input: A set U of n elements
Output: $\tilde{\pi}_k$

Let t be the time
Initialize $\tilde{\pi}, C, \tilde{\pi}_C, P, \tilde{\pi}_P,$ and $\tilde{\pi}_k$ arbitrarily
while (true) **do**
  Execute in $t \equiv 1$(**mod** 4) **steps** /*QS_1*/
  $\tilde{\pi} \leftarrow$ quick_sort(U)
  $C \leftarrow \tilde{\pi}^{-1}([k + c'\alpha \ln n])$ /*The constant c' will be determined in the proof of Theorem 2*/
  Execute in $t \equiv 2$(**mod** 4) **steps** /*QS_2*/
  $\tilde{\pi}_C \leftarrow$ quick_sort(C)
  $P \leftarrow \tilde{\pi}_C^{-1}([k])$
  Execute in $t \equiv 3$(**mod** 4) **steps** /*QS_3*/
  $\tilde{\pi}_P \leftarrow$ quick_sort(P)
  Execute in $t \equiv 0$(**mod** 4) **steps** /*Local-sort*/
  $\tilde{\pi}_k \leftarrow$ Local-sort($P, \tilde{\pi}_P, 4c+1$) /*The constant c will be determined in the proof of Theorem 2*/
end while

Algorithm 3 Local-sort

Input: A set P; an order π over P; an integer c
Output: $\tilde{\pi}$

$m \leftarrow |P|$
$B_1 \leftarrow \pi^{-1}([c])$ /* Define the first block */
$\tilde{\pi}^{-1}(1) \leftarrow$ Maximum-Find(B_1)
$j = 2$
while $(c + j - 1 \le m)$ **do**
  $B_j \leftarrow (B_{j-1} \backslash \tilde{\pi}^{-1}(j-1)) \bigcup \pi^{-1}(c+j-1)$ /* Define the j-th block */
  $\tilde{\pi}^{-1}(j) \leftarrow$ Maximum-Find(B_j)
  $j = j + 1$
end while
$B_e \leftarrow B_{j-1}$ /*Deal with the final block*/
while $|B_e| \ge 1$ **do**
  $\tilde{\pi}^{-1}(j) \leftarrow$ Maximum-Find(B_e)
  $B_e \leftarrow B_e \backslash \tilde{\pi}^{-1}(j)$
  $j = j + 1$
end while

The main idea of Algorithm 3 (Local-sort) is to regulate the order over P block by block. Since block-by-block processing takes linear time, the errors can be corrected in time and few new errors will emerge during one run of Algorithm 3. Considering

that the elements may move across blocks, it is necessary to make the blocks overlap. Actually, for each j, the element of the lowest rank in the j-th block is found, regarded as the j-th element of the final order, and removed from the block. The rest elements of the j-th block, together with the lowest-ranked element in P (according to the latest order produced by QS_3) that has not yet been processed, forms the $(j+1)$-st block. The element of the lowest rank in each block is found by calling Algorithm 4, which repeatedly runs sequential comparison. Both Algorithms 3 and 4 are self-explained, so detailed explanation is omitted.

Algorithm 4 Maximum-Find

Input: B
Output: u_{max}
1: $u_{max} \leftarrow B(1)$
2: $j = 2$
3: **while** $(j \leq |B|)$ **do**
4: **if** $u_{max} < B(j)$ **then**
5: $u_{max} \leftarrow B(j)$
6: **end if**
7: $j = j + 1$
8: **end while**

Theorem 2 *Assume $\alpha = o(\frac{\sqrt{n}}{\ln n})$ and $k = O((\frac{n}{\alpha \ln n})^{1-\epsilon})$, where $\epsilon > 0$. Let $\tilde{\pi}_k^t$ be the output of Algorithm 2 and π_k^t be the true order over the top k elements at time t. For sufficiently large t, we have that:*

1. *If $k^2\alpha = o(n)$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = 0) = 1 - o(1)$,*
2. *If $k^2\alpha = \Theta(n)$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = 0) = \Theta(1)$, and*
3. *If $k^2\alpha = \omega(n)$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = O(\frac{k^2\alpha}{n})) = 1 - o(1)$.*

The following lemma will be used in proving Theorem 2.

Lemma 3 *In the consecutive-swapping model with $\alpha = o(n)$, consider a run of the randomized quick-sort algorithm. If no swaps involve any element that is playing the role of a pivot, then for any $u \in U$, the number of elements that are incorrectly ordered with respect to u is bounded by the number of swaps that involve u.*

Proof This is in fact a by-product of the proof of Lemma 2.

Assume that no swaps involve any element that is playing the role of a pivot. In the proof of Lemma 2, we must have $S_2 = \emptyset$. The lemma immediately follows. □

The proof of Theorem 2 consists of five steps. First, with high probability, the set C produced by QS_1 includes all the top k elements during the next run of QS_1. Second, with high probability, the set P produced by QS_2 exactly consists of the top k elements during the next run of QS_2. Third, with high probability, the true rank of any element remains close to that estimated by QS_3, during the next run of QS_3. Fourth, with high probability, the number of incorrectly ordered pairs by Local-sort is upper-bounded by the number of swaps during the run of Local-sort. And fifth, a proper upper bound of

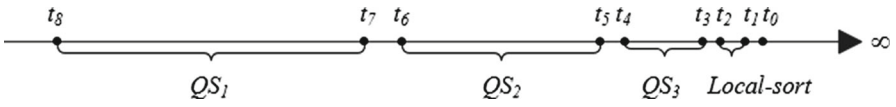

Fig. 1 The time steps $t_8, t_7, \ldots t_0$

the number of swaps during a run of Local-sort is presented. These steps immediately lead to the theorem.

Proof (of Theorem 2) We will show that the theorem holds at t_0, where t_0 is an arbitrary time step after QS_1 in Algorithm 2 has run twice.

Consider the last completed run of Local-sort before t_0, which starts at t_2 and terminates at t_1. It is easy to see that $t_0 - t_1 = O(k)$ and $t_1 - t_2 = O(k)$. Also note that the input of *Local-sort* at t_2 comes from the last completed run of QS_3 before t_2, which starts at t_4 and finishes at t_3. By Lemma 1, with probability $1-o(1)$, $t_0 - t_3 = O(k \ln k)$ and $t_3 - t_4 = O(k \ln k)$. Likewise, the input of QS_3 at t_4 comes from the last completed run of QS_2 before t_4, which starts at t_6 and finishes at t_5, and with probability $1-o(1)$, $t_0 - t_5 = O((k+\alpha \ln n)\ln(k+\alpha \ln n))$ and $t_5 - t_6 = O((k+\alpha \ln n)\ln(k+\alpha \ln n))$. Also, the input of QS_2 at t_6 comes from the last completed run of QS_1, which starts at t_8 and terminates at t_7, and with probability $1 - o(1)$, $t_0 - t_7 = O(n \ln n)$ and $t_7 - t_8 = O(n \ln n)$. The time steps t_0 through to t_8 are illustrated in Fig. 1.

Step 1 We show that with probability $1 - o(1)$, for all $u \in U$ and all $t \in [t_7, t_0]$, $|\pi^t(u) - \tilde{\pi}^{t_7}(u)| \leq c'\alpha \ln n$, where c' is a constant. Due to the union bound, it is sufficient to show that for each $u \in U$, with probability $1 - o(\frac{1}{n})$, $|\pi^t(u) - \tilde{\pi}^{t_7}(u)| \leq c'\alpha \ln n$ for all $t \in [t_7, t_0]$. Note that $|\pi^t(u) - \tilde{\pi}^{t_7}(u)| \leq |\pi^t(u) - \pi^{t_7}(u)| + |\pi^{t_7}(u) - \tilde{\pi}^{t_7}(u)|$, and $\Pr(|\pi^{t_7}(u) - \tilde{\pi}^{t_7}(u)| \leq c_1 \ln n) = 1 - O(\frac{1}{n^2})$ for some constant c_1 by Lemma 2. Hence, we just need to show that $\Pr(\forall t \in [t_7, t_0], |\pi^t(u) - \pi^{t_7}(u)| \leq c_2 \ln n) = 1 - O(\frac{1}{n^2})$ where c_2 is a constant, and let $c' = c_1 + c_2$. Actually this follows from Chernoff bound and the fact that during $[t_7, t_0]$, the rank of u changes $O(\ln n)$ in expectation and with high probability.

As a result, event $\mathcal{E}_1$ happens with probability $1 - o(1)$, where $\mathcal{E}_1$ means that the set C produced at t_7 by QS_1, denoted as C^{t_7}, contains all the top k elements during $[t_7, t_0]$.

Step 2 We show that event $\mathcal{E}_2$ happens with probability $1 - o(1)$, where $\mathcal{E}_2$ means that the set P produced by QS_2 at t_5, denoted as P^{t_5}, exactly consists of the top k elements of U during $[t_5, t_0]$. It follows from two facts. On the one hand, $\mathcal{E}_1$ happens with probability $1 - o(1)$. On the other hand, event $\mathcal{E}_3$ happens with probability $1 - O(\frac{(k+\alpha \ln n)\alpha \ln(k+\alpha \ln n)}{n}) = 1 - o(1)$, where $\mathcal{E}_3$ means that the k-th element at time t_6 doesn't swap during $[t_6, t_0]$.

Hence, in the rest of the proof we assume that $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ all happen.

Step 3 Let $\tilde{\pi}_P^{t_3}$ be the order on P^{t_5} produced by QS_3 at time t_3. We now show that with probability $1 - o(1)$, for all $u \in P^{t_5}$ and all $t \in [t_3, t_0]$, $|\pi^t(u) - \tilde{\pi}_P^{t_3}(u)| \leq c$ for some constant c.

To see why, consider Y_i, the number of swaps during $[t_4, t_0]$ that involve $(\pi^{t_4})^{-1}(i)$. Since $t_0 - t_4 = O(k \ln k)$, $\mathbb{E}[Y_i] = O(\frac{\alpha k \ln k}{n})$. Arbitrarily choose a constant $c_3 \geq e$. By Chernoff bound, $\Pr(Y_i \geq c_3) \leq (\mathbb{E}[Y_i])^{c_3}$. Because $k = O((\frac{n}{\alpha \ln n})^{1-\epsilon})$, there is

a constant $0 < \beta < 1$ such that $\frac{\alpha k \ln k}{n} = O(k^{\beta-1})$. Arbitrarily choose a constant $c_4 > \max\{(1-\beta)^{-1}, c_3\}$, and we have $\Pr(\exists i \in [k], Y_i \geq c_4) \leq k(O(\frac{k\alpha \ln k}{n}))^{c_4} = O(k^{1+(\beta-1)c_4}) = o(1)$. Consequently, $\Pr(\forall i \in [k], Y_i < c_4) = 1 - o(1)$.

On the other hand, since $t_3 - t_4 = O(k \ln k)$, with probability $1 - O(\frac{\alpha k \ln k}{n}) = 1 - o(1)$, no swaps during $[t_4, t_3]$ involve any element while it is a pivot in QS_3. By Lemma 3, with probability $1-o(1)$, $|\pi^{t_3}(u) - \tilde{\pi}_P^{t_3}(u)| \leq c_4$ for all $u \in P^{t_5}$. As a result, with probability $1-o(1)$, $|\pi^t(u) - \tilde{\pi}_P^{t_3}(u)| \leq |\pi^t(u) - \pi^{t_3}(u)| + |\pi^{t_3}(u) - \tilde{\pi}_P^{t_3}(u)| \leq c$ for all $u \in P^{t_5}$ and all $t \in [t_3, t_0]$, where $c = 2c_4$.

Step 4 Focus on the behavior of *Local-sort* during $[t_2, t_1]$. Assume that during $[t_2, t_1]$, no swaps involve any u_{max}. We claim that for any $u, v \in P^{t_5}$, if $\pi^t(u) < \pi^t(v)$ for all $t \in [t_2, t_1]$, then $\tilde{\pi}_k^{t_1}(u) < \tilde{\pi}_k^{t_1}(v)$.

The claim can be proved in three cases. Suppose that $\pi^t(u) < \pi^t(v)$ for all $t \in [t_2, t_1]$. Define $r_u \triangleq \tilde{\pi}_P^{t_3}(u)$ and $r_v \triangleq \tilde{\pi}_P^{t_3}(v)$.

- **Case 1** $u, v \in B_l$ for some l. Let l_0 be the largest such l. This implies that u_{max} of B_{l_0} is either u or v. Since no swaps involve any u_{max}, the u_{max} computed by Local-sort cannot be v. This means that u_{max} of B_{l_0} is u and $\tilde{\pi}_k^{t_1}(u) = l_0 < \tilde{\pi}_k^{t_1}(v)$.
- **Case 2** $r_u < r_v$ and there is no l such that $u, v \in B_l$. There must be some $r_u - 4c \leq l < r_v - 4c$ such that u_{max} of B_l is u. Hence, $\tilde{\pi}_k^{t_1}(u) = l < r_v - 4c \leq \tilde{\pi}_k^{t_1}(v)$.
- **Case 3** $r_u > r_v$ and there is no l such that $u, v \in B_l$. We have $r_u - r_v < r_u - \pi^t(u) + \pi^t(v) - r_v$ for any $t \in [t_2, t_1]$, so $r_u - r_v < 2c$. For any integer m, let $V_m = (\tilde{\pi}_P^{t_3})^{-1}(\{1, 2, \ldots, m+4c\})$. For all $r_v - 4c \leq l < r_u - 4c$, since $B_l \subseteq V_l$ and $|V_l \setminus V_{r_v-6c-1}| \leq 4c$, one gets $|B_l \bigcap V_{r_v-6c-1}| \geq 1$. For any $w \in B_l \bigcap V_{r_v-6c-1}$ and any $t \in [t_2, t_1]$, $\pi^t(w) \leq \tilde{\pi}_P^{t_3}(w) + c \leq r_v - c - 1 < \pi^t(v)$. By the proof of case 1 (with u replaced by w), u_{max} of B_l is not v for any $r_v - 4c \leq l < r_u - 4c$. This means that $v, u \in B_{r_u-4c}$, which is a contradiction.

Since the probability that no swaps during $[t_2, t_1]$ involve any u_{max} is $1 - O(\frac{k\alpha}{n})$, the above claim means that $\Pr(\mathrm{KT}(\tilde{\pi}_k^{t_1}, \pi_k^{t_1}) \leq Y) = 1 - o(1)$, where Y is the number of swaps occurred among P^{t_5} during $[t_2, t_1]$.

Step 5 Since $\Pr(Y = 0) = (1 - \frac{k}{n})^{O(k)\alpha}$, $\Pr(Y = 0) = 1 - o(1)$ if $k^2\alpha = o(n)$, and $\Pr(Y = 0) = \Theta(1)$ if $k^2\alpha = \Theta(n)$. When $k^2\alpha = \omega(n)$, note that $\mathbb{E}[Y] = O(\frac{k^2\alpha}{n})$, so $Pr(Y = O(\frac{k^2\alpha}{n})) = 1 - o(1)$ by Chernoff bound. Actually, we still have these results if $[t_1, t_0]$ is considered.

Altogether, the theorem holds. □

3.3 Lower Bounds for the Top-k-Selection Problem

Now we analyze the lower bounds of the performance of any top-k-selection algorithm. The lower bounds hold for both randomized and deterministic algorithms.

Let A be an arbitrary algorithm which takes our dynamic data as input and outputs a total order $\tilde{\pi}_k^t$ on a subset of size k at every time step t. Let π_k^t be the true order on the top k elements. The following theorems characterize the difference between $\tilde{\pi}_k^t$ and π_k^t when k is large.

Theorem 3 *Given* $k = \Omega(\sqrt{\frac{n}{\alpha}})$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) > 0) = \Theta(1)$ *for every* $t > k$.

Proof Arbitrarily fix $t > k$.

If $\alpha = \omega(n)$, $k = \Omega(\sqrt{\frac{n}{\alpha}})$ is equivalent to $k \in [n]$ since $k < 1$ does not make sense for the partial sorting problem. On the other hand, $k = \Omega(\sqrt{\frac{n}{\alpha}})$ exactly means $k \in [n]$ when $\alpha = \Theta(n)$. In addition, for any fixed k, if $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) > 0) = \Theta(1)$ for some α, it obviously holds for bigger α which means more frequent changes. As a result, without loss of generality, we assume $\alpha = O(n)$.

Let us focus on the special case where $k = \Theta(\sqrt{\frac{n}{\alpha}})$ because it trivially implies the general case where $k = \Omega(\sqrt{\frac{n}{\alpha}})$. Now we proceed case by case.

Case 1 $\alpha = \Theta(n)$. Assume that asymptotically, $\alpha = cn$ for some constant c. Let $u_i \in U$ be such that $\pi^{t-1}(u_i) = i$, $\mathcal{E}_i$ be the event that $\pi^t(u_i) = 1$, and $p_i = \Pr(\mathcal{E}_i)$, $i \in [n]$. We show that $\min\{p_1, p_2, p_3\} = \Theta(1)$. Actually, if u_1 is not involved in the swaps at time t, then $\pi^t(u_1) = 1$, so $\Pr(\pi^t(u_1) = 1) \geq \left(1 - \frac{1}{n-1}\right)^{\alpha}$ which approaches e^{-c}. Moreover, at time t, if exactly one swap occurs between u_1 and u_2 and the other swaps are among $u_3, \ldots u_n$, then $\pi^t(u_2) = 1$, meaning that $\Pr(\pi^t(u_2) = 1) \geq \binom{\alpha}{1} \frac{1}{n-1} \left(1 - \frac{2}{n-1}\right)^{\alpha-1}$ which approaches ce^{-2c}. Likewise, $\Pr(\pi^t(u_3) = 1) \geq c^2 e^{-3c}$ asymptotically. Therefore, $p = \min\{p_1, p_2, p_3\} = \Theta(1)$.

By the assumption of the consecutive-swapping model, at time t, only one pair of elements can be compared. No matter which pair is chosen to compare, there are just two possible results. Hence, at least one of the events $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ can't be identified, so $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) > 0) \geq p = \Theta(1)$.

Case 2 $\alpha = o(n)$. Basically, we will show that with a constant probability, in any period of $\Theta(\sqrt{\frac{n}{\alpha}})$, exactly one swap occurs among the top k elements and the swap is not observed.

Specifically, $k = \Theta(\sqrt{\frac{n}{\alpha}})$ means that asymptotically, $k = c_1\sqrt{\frac{n}{\alpha}}$ for some constant c_1. Consider the time interval $I = [t - c_2\sqrt{\frac{n}{\alpha}}, t]$, where $c_2 < c_1$ is a constant. Let $\mathcal{E}$ be the event that only one pair swaps among the top k elements during the interval I, and $q_1 = \Pr(\mathcal{E})$. Then $q_1 = \frac{(k-1)c_2\sqrt{n\alpha}}{n-1}(1 - \frac{k}{n-1})^{c_2\sqrt{n\alpha}-1}$. When n approaches infinity, $q_1 = c_1c_2e^{-c_1c_2}$. During the interval I, since at each time step only one pair is observed by the algorithm, altogether at most $2c_2\sqrt{\frac{n}{\alpha}}$ elements are checked. Now assume that $\mathcal{E}$ occurs, and let $\mathcal{E}'$ be the event that the swapped pair of elements are not observed during the interval I. Then $q_2 \triangleq \Pr(\mathcal{E}') \geq \frac{c_1\sqrt{n/\alpha}-1-4c_2\sqrt{n/\alpha}}{c_1\sqrt{n/\alpha}-1}$ which equals $\frac{c_1-4c_2}{c_1}$ asymptotically. Thus with probability at least $q_1q_2 \geq c_2^2e^{-5c_2^2}$, the algorithm cannot tell the correct order of the top k elements at time t_2. The theorem holds. □

Theorem 4 *Given $k = \omega(\sqrt{n})$ and $\alpha = O(1)$, $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = \Omega(\frac{k^2}{n})$ in expectation and with probability $1 - o(1)$ for every $t > k/8$.*

The basic idea of the proof is that with high probability, in any period of length $\Theta(k)$, $\Omega(\frac{k^2}{n})$ swaps occur among the top k elements and a majority of the swaps cannot be observed.

Proof By our definition of Kendall tau distance, if $\tilde{\pi}_k^t$ is not on the true set of top k elements, then $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) \geq k - 1 = \Omega(\frac{k^2}{n})$ and the theorem trivially holds. Hence, we now assume that $\tilde{\pi}_k^t$ is on the true set of top k elements.

When $k = \Theta(n)$, this theorem immediately follows from [1, Theorem 1], which claims the $\Omega(n)$ lower bound of error in solving the sorting problem. As a result, we assume $k = o(n)$ in the rest of the proof.

The proof is inspired by and is similar to that of [1, Theorem 1]. It is presented here not only to make this paper self-contained, but also due to subtle differences. For convenience we assume that $\alpha = 1$. The proof can be modified slightly to prove the case where $\alpha > 1$.

Consider the time interval $I = [t - \frac{k}{100}, t]$. Let X be the number of swaps that occur among the top k elements. Then $\mathbb{E}[X] = \Theta(\frac{k^2}{n})$. By Chernoff bound, we have that $X = \Theta(\frac{k^2}{n})$ with probability $1 - o(1)$.

We then use the idea of deferred decisions as in [1], but focus on the decisions involving only the top k elements. At every time step, one random pair of consecutive elements is swapped. The process is equivalent to the following process (called the nature's decision): two disjoint pairs of consecutive elements are picked uniformly randomly and then one of these two pairs is randomly selected to swap. Using the principle of deferred decisions, we fix all the nature's decisions that include at least one of the elements observed by the algorithm and defer the rest. Since a swap will influence the elements in future swaps, we also need to fix both swaps that overlap. Therefore, the deferred decisions are a random set of disjoint pairs from the elements that are not involved in the algorithm's comparisons or the nature's fixed decisions so far. An element is said to be *touched* if it is observed by the algorithm or is involved in any fixed decisions.

Initially, both the set of touched elements and the set of deferred decisions are empty. At every time step, a pair of elements is selected by the algorithm to compare. We mark each of these elements touched if it is previously untouched. For each deferred decision, we flip a coin with a suitable probability to determine if the decision involves an element newly marked as touched. If so, the decision is fixed and one pair in the decision is randomly picked to swap. The decision may involve other previously untouched elements and we mark all those elements touched. And again, we continue to determine if any of the deferred decisions involves any of those elements. The process continues until all the deferred decisions are checked against all the newly-marked touched elements. Then the comparison of the pair picked by the algorithm is answered. Next, the nature makes a new decision. We flip a coin to determine whether the new decision involves any touched elements. If it does, we fix the decision, update the set of touched elements, and iterate as before. Also, a coin is flipped to determine whether the new decision overlaps with any of the deferred decisions. If it does, we fix both of the decisions.

At the end of the interval I, the set of all the touched elements is of size at most $\frac{6k}{100}$, because at each step, the nature's decision and the algorithm's query involve at most six distinct elements. Thus at least $(1-\frac{12}{100})k$ consecutive pairs don't include any touched elements. Therefore, during the interval I, the probability that a decision is deferred is at least $(1-\frac{12}{100})^2 > \frac{1}{2}$. Recall that $\mathbb{E}[X] = \Theta(\frac{k^2}{n})$. By Chernoff bound, we have $\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = \Omega(\frac{k^2}{n})$ in expectation and with probability $1-o(1)$. □

From Theorems 2 and 3, we know that $\Theta(\sqrt{n/\alpha})$ is the critical point of k, and by Theorem 4, it is impossible to generally improve Algorithm 2 even if $k = \omega(\sqrt{n/\alpha})$. The term *critical point* means the least upper bound of k such that top-k-selection problem can be solved error-free with probability $1-o(1)$.

4 Gaussian-Swapping Model

This section is devoted to extending the algorithms for the consecutive-swapping model to the Gaussian-swapping model. We focus on the special case where α is a constant, and still assume that at each time step only one pair of elements can be compared.

Algorithms 1 and 2 can be slightly adapted to solve the top-k-set problem and the top-k-selection problem in this model, respectively. Specifically, replacing α in lines 5 and 8 of Algorithm 1 with $\ln^{0.5} n$, one gets Algorithm 5; likewise, in Algorithm 2, replacing α in line 6 with $\ln^{0.5} n$ and $4c+1$ in lines 13 with $4c\ln^{0.5} n + 1$, and removing QS_2, we get Algorithm 6. The following theorems state the performance of these algorithms.

Algorithm 5 Gaussian-top-k-set

Input: A set U of n elements
Output: $\widetilde{T}$

1: Initialize $\tilde{\pi}, L, C, \tilde{\pi}_C,$ and $\widetilde{T}$ arbitrarily
2: **while** (true) **do**
3: **Execute in odd steps:** /*QS_1*/
4: $\tilde{\pi} \leftarrow$ quick_sort(U)
5: $L \leftarrow \tilde{\pi}^{-1}([k - c\ln^{1.5} n])$ and $C \leftarrow \tilde{\pi}^{-1}([k + c\ln^{1.5} n]) \setminus L$ /*The constant c will be determined in the proof of Theorem 5*/
6: **Execute in even steps:** /*QS_2 */
7: $\tilde{\pi}_C \leftarrow$ quick_sort(C)
8: $\widetilde{T} \leftarrow L \bigcup \tilde{\pi}_C^{-1}([c\ln^{1.5} n])$
9: **end while**

Theorem 5 *For any $k \in [n]$, we have $\Pr(\widetilde{T}_t = (\pi^t)^{-1}([k])) = 1-o(1)$, where $\widetilde{T}_t$ is the output of Algorithm 5 at time t, π^t is the true order at time t, and t is sufficiently large.*

The following definition and lemmas will be needed in the proof.

Algorithm 6 Gaussian-top-k-selection

Input: A set U of n elements
Output: $\tilde{\pi}_k$

1: Let t be the time
2: Initialize $\tilde{\pi}$, C, $\tilde{\pi}_C$, P, $\tilde{\pi}_P$, and $\tilde{\pi}_k$ arbitrarily
3: **while** (true) **do**
4: **Execute in** $t \equiv 1 (\mathbf{mod}\ 3)$ **steps** /*QS_1*/
5: $\tilde{\pi} \leftarrow$ quick_sort(U)
6: $C \leftarrow \tilde{\pi}^{-1}([k + c' \ln^{1.5} n])$ /*The constant c' will be determined in the proof of Theorem 6*/
7: **Execute in** $t \equiv 2 (\mathbf{mod}\ 3)$ **steps** /*QS_2*/
8: $\tilde{\pi}_C \leftarrow$ quick_sort(C)
9: $P \leftarrow \tilde{\pi}_C^{-1}([k])$ and $\tilde{\pi}_P \leftarrow \tilde{\pi}_C|P$ /*$\tilde{\pi}_C|P$ is the restriction of $\tilde{\pi}_C$ to P */
10: **Execute in** $t \equiv 0 (\mathbf{mod}\ 3)$ **steps** /*Local-sort*/
11: $\tilde{\pi}_k \leftarrow$ *Local-sort*$(P, \tilde{\pi}_P, 4c\sqrt{\ln n} + 1)$ /*The constant c will be determined in the proof of Theorem 6*/
12: **end while**

Definition 1 In the Gaussian-swapping Model, given $u \in U$, at any time t, define u-neighbourhood at t to be the set $\{v \in U : |\pi^t(v) - \pi^t(u)| \leq 4\sqrt{\ln n}\}$.

Lemma 4 *In the Gaussian-swapping Model, the randomized quick-sort algorithm can terminate in $O(n \ln n)$ time in expectation and with probability $1 - o(n^{-3})$.*

Proof This proof is similar to that of Lemma 1. Lemma 1 relies on the key fact that with high probability, a pivot that is good at the time it is chosen will actually divide the array in a pretty balanced way. Though this fact is an easy observation in the consecutive-swapping model, it is quite hard in the Gaussian-swapping model.

In the following, we only prove this fact, since the other part of the proof of Lemma 1 can be directly used. Namely, assume that u is chosen as the pivot of an array of size s and that according to the true order when u is chosen, u divides the array into two sub-arrays A and B each having at least γs elements, where $\gamma \in (0, 0.5]$ is a constant. Let A' and B' be the two sub-arrays that are actually obtained. We want to show that $\Pr(|A'| \geq \frac{\gamma s}{2}, |B'| \geq \frac{\gamma s}{2}) = 1 - o(1)$.

Recall the Gaussian-swapping model where $\Pr(d) = \beta e^{-d^2/2}$. Given elements v and w whose ranks differ by j, the probability that v and w swap is $\frac{\beta e^{-j^2/2}}{n-j}$, denoted by p_j. Define q_i to be the probability that the element of rank i is chosen to swap. We have

$$q_i = \sum_{j=1}^{i-1} p_j + \sum_{j=1}^{n-i} p_j.$$

Since $p_j = \frac{\beta e^{-j^2/2}}{n-j}$, it's easy to see that p_j decreases as j increases. Hence, q_1 is the minimum and $q_{\lfloor \frac{n+1}{2} \rfloor}$ is the maximum and $2q_1 > q_{\lfloor \frac{n+1}{2} \rfloor}$. Thus, $q_i = \Theta(\frac{1}{n})$.

Since

$$\sum_{d \geq 4\sqrt{\ln n}} Pr(d) < \frac{1}{n^8} \times n = \frac{1}{n^7},$$

the event $\mathcal{E}$ happens with probability $1-o(\frac{1}{n^5})$, where $\mathcal{E}$ means that no pair of elements whose ranks differ by more than $4\sqrt{\ln n}$ is swapped in a run of quick-sort. In the rest of the proof, we assume that $\mathcal{E}$ does happen.

Let $C = B' \setminus B$. The elements of C must come from the following two types of events which occur while u is the pivot:

1. The pivot u swaps with another element. Each such event contributes at most $4\sqrt{\ln n}$ elements to C.
2. A pair within u-neighbourhood swaps. Each such event contributes at most 1 element to C.

Let X be the number of the elements in C caused by the first type of events. Then $\mathbb{E}[X] = O(\frac{s\sqrt{\ln n}}{n})$. By Markov Inequality, $\Pr(X \geq \frac{\gamma s}{4}) = o(1)$.

Let Y be the number of the elements in C caused by the second type of events. Since $\mathcal{E}$ happens by assumption, $\mathbb{E}[Y] = O(\frac{s\sqrt{\ln n}}{n})$. By Markov Inequality, $\Pr(Y \geq \frac{\gamma s}{4}) = o(1)$.

Altogether, $\Pr(|C| \geq \frac{\gamma s}{2}) = o(1)$. Likewise, $\Pr(|A' \setminus A| \geq \frac{\gamma s}{2}) = o(1)$. Consequently, $\Pr(|A'| \geq \frac{\gamma s}{2}, |B'| \geq \frac{\gamma s}{2}) = 1 - o(1)$. □

Lemma 5 *In the Gaussian-swapping model, consider a run of the randomized quick-sort algorithm from time t_0 to t_1. For any $u \in U$, the number of elements that are incorrectly ordered with respect to u is $O(\ln^{1.5} n)$ with probability $1 - O(\frac{1}{n^3})$. In addition, with probability $1 - O(\frac{1}{n^2})$, $|\pi^{t_1}(u) - \tilde{\pi}^{t_1}(u)| = O(\ln^{1.5} n)$ for all $u \in U$.*

Proof By Lemma 4, without loss of generality, we assume that $t_1 - t_0 = O(n \ln n)$ and no pair of elements whose ranks differ by more than $4\sqrt{\ln n}$ is swapped during $[t_0, t_1]$. We will also use the following fact in the proof of Lemma 4: at any time step, the probability that a certain element is chosen to swap is $\Theta(\frac{1}{n})$.

Arbitrarily fix $u \in U$. Similar to the proof of [1, Lemma 6], the set of elements which are incorrectly ordered with respect to u can be partitioned into two sets, A and B:

$$\begin{aligned} A = &\{v | u <_{\tilde{\pi}^{t_1}} v, u >_{\pi^{t_1}} v, \exists t \in [t_0, t_1) : u <_{\pi^t} v\} \\ &\bigcup \{v | u >_{\tilde{\pi}^{t_1}} v, u <_{\pi^{t_1}} v, \exists t \in [t_0, t_1) : u >_{\pi^t} v\} \\ B = &\{v | u <_{\tilde{\pi}^{t_1}} v, \forall t \in [t_0, t_1] : u >_{\pi^t} v\} \\ &\bigcup \{v | u >_{\tilde{\pi}^{t_1}} v, \forall t \in [t_0, t_1] : u <_{\pi^t} v\} \end{aligned}$$

Only two types of swaps can cause A:

Type 1.1: swaps involving u,

Type 1.2: swaps within u-neighbourhood but not involving u.

At each time step, a swap of type 1.1 can contribute at most $4\sqrt{\ln n}$ elements to set A. Recall that at each time step, the probability that u is chosen to swap is $\Theta(\frac{1}{n})$. By Chernoff bound, in expectation and with probability $1 - O(\frac{1}{n^3})$, the number of type 1.1 swaps is $O(\ln n)$, meaning that the elements in A caused by type 1.1 swaps is upper-bounded by $c_1 \ln^{1.5} n$ for some constant c_1.

On the other hand, at each time step, a swap of type 1.2 causes at most 2 elements in set A. Recall that at every time step, the probability that a swap occurs within u-neighborhood is $O(\frac{\sqrt{\ln n}}{n})$. Using Chernoff bound, in expectation and with probability $1 - O(\frac{1}{n^3})$, the number of type 1.2 swaps is $O(\ln^{1.5} n)$, implying that the elements in A caused by type 1.2 swaps is upper-bounded by $c_2 \ln^{1.5} n$ for some constant c_2.

Now we analyze the set B. Let's focus on the part $B_1 = \{v | u >_{\tilde{\pi}^{t_1}} v, \forall t \in [t_0, t_1] : u <_{\pi^t} v\}$ (the other part can be handled in a similar way).

According to operational semantics of the randomized quick-sort algorithm, given $v \in B_1$, there is a unique pivot, denoted by p_v, which is compared with both u and v and satisfies that $v < p_v$ and $p_v < u$ by the true orders when the comparisons occur. It is easy to see that the relative order between p_v and v must change while p_v is playing the role of a pivot. Such changes of the relative order can be caused either by swaps involving p_v or by swaps among p_v-neighbourhood. Let $P_u \subseteq U$ be the set of pivots along the path to u in the quick-sort tree. Then, B_1 can be caused only by two types of swaps:

Type 2.1: swaps involving some $v \in P_u$,

Type 2.2: swaps within v-neighbourhood but not involving v, where $v \in P_u$.

Note that in both cases, a swap will be included only if it occurs while the corresponding v is playing the role of a pivot.

It is easy to see that each swap of type 2.1 contributes at most $4\sqrt{\ln n}$ elements to set B_1. Let X_v be the number of steps at which v is a pivot and Y_v be the number of steps at which v is a pivot and is involved in a swap. Then by Lemma 4, we have

$$\mathbb{E}\left[\sum_{v \in P_u} X_v\right] = O(n \ln n).$$

We define $Z = \sum_{v \in P_u} Y_v$. Since each Y_v follows the binomial distribution $B(X_v, \Theta(\frac{1}{n}))$, we have

$$\mathbb{E}[Z] = \mathbb{E}\left[\mathbb{E}\left[\sum_{v \in P_u} Y_v | X_v\right]\right] = \mathbb{E}\left[\sum_{v \in P_u} X_v\right] \Theta\left(\frac{1}{n}\right) = O(\ln n).$$

By Chernoff bound, $Z = O(\ln n)$ with probability $1 - O(\frac{1}{n^3})$. Therefore, with probability $1 - O(\frac{1}{n^3})$, swaps of type 2.1 contribute at most $c_3 \ln^{1.5} n$ elements to B_1, for some constant c_3.

On the other hand, at every time step, a swap of type 2.2 occurs with probability $O(\frac{\sqrt{\ln n}}{n})$. Each type 2.2 swap causes at most two elements to change their relative order with respect to the pivot, hence leading to at most two elements in B_1. Let W_v be the number of type 2.2 swaps within v-neighbourhood when v is a pivot. We have that $\mathbb{E}[\sum_{v \in P_u} W_v] = O(\frac{\sqrt{\ln n}}{n}) \mathbb{E}[\sum_{v \in P_u} X_v] = O(\ln^{1.5} n)$. By Chernoff bound, with probability $1 - O(\frac{1}{n^3})$, swaps of type 2.2 lead to at most $c_4 \ln^{1.5} n$ elements in B_1, for some constant c_4.

Let $c = c_1 + c_2 + c_3 + c_4$. Then we get the first part of the lemma.

By union bound, with probability at least $1 - O(\frac{1}{n^2})$, for every $u \in U$, there are at most $O(\ln^{1.5} n)$ elements which are incorrectly ordered with respect to u, implying that $|\pi^{t_1}(u) - \tilde{\pi}^{t_1}(u)| \leq c \ln^{1.5} n$. The second part of the lemma is proven. □

Proof (of Theorem 5) Consider a run of QS_1, which starts at t_0 and ends at t_1. By Lemma 4, we have $t_1 - t_0 = O(n \ln n)$ in expectation and with probability $1 - o(1)$. By Lemma 5, we have that with probability $1 - O(\frac{1}{n^2})$, $|\pi^{t_0}(u) - \tilde{\pi}^{t_0}(u)| \leq c_1 \ln^{1.5} n$ for every $u \in U$ and some constant c_1. During $[t_0, t_1]$, the rank of an element u changes less than $c_2 \ln^{1.5} n$ with probability $1 - o(\frac{1}{n^2})$ for some constant c_2. Hence with probability $1-o(1)$, we have $|\pi^t(u)-\tilde{\pi}^{t_0}(u)| \leq (c_1+c_2) \ln^{1.5} n$ for all $t \in [t_0, t_1]$ and all $u \in U$. Let $c = c_1 + c_2$.

Note that L contains all the elements u such that $\tilde{\pi}^{t_0}(u) \leq k - c \ln^{1.5} n$. Then with probability $1-o(1)$, we have $\pi^t(u) \leq k$ for all $t \in [t_0, t_1]$ and all $u \in L$. Consider the set $R = U \backslash (L \bigcup C) = \{v : \tilde{\pi}^{t_0}(v) \geq k + c\alpha \ln n + 1\}$. With probability $1 - o(1)$, we have $\tilde{\pi}^t(v) > k$ for all $t \in [t_0, t_1]$ and all $v \in R$. Therefore, with probability $1 - o(1)$, $L \bigcup C$ contains all the elements which may be among top k at some $t \in [t_0, t_1]$.

In line 6, QS_2 takes time $O(\ln^{1.5} n \ln \ln n)$ to sort C, with probability $1 - o(1)$. Let $\tilde{u}$ stand for the element of true rank k when QS_2 begins to sort C. If no element in $\tilde{u}$-neighborhood is swapped throughout the run of QS_2, the algorithm returns the correct set of the top k elements at the end of the run of QS_2. This reason lies in two aspect. On the one hand, with probability $1 - o(1)$, each pair of elements whose ranks differ by more than $4\sqrt{\ln n}$ is not swapped during the run of QS_2. On the other hand, the rank of an element $u \in C$ remains smaller than $k - 4\sqrt{\ln n}$ if it is so at the beginning of the run of QS_2 and no element in $\tilde{u}$-neighbourhood is swap during the run of QS_2. The probability that $\tilde{u}$-neighbourhood is not involved in any swap in this period is $1 - O(\frac{\ln^3 n \ln \ln n}{n}) = 1 - o(1)$. Furthermore, with probability $1 - O(\frac{\ln^3 n \ln \ln n}{n}) = 1 - o(1)$, the set of the top k elements does not change during the next run of QS_2, implying the result. □

Theorem 6 *Assume that $k = O((\frac{n}{\ln n})^{1-\epsilon})$, where $\epsilon > 0$. Let $\tilde{\pi}_k^t$ be the output of Algorithm 6 and π_k^t be the true order over the top k elements at time t. For sufficiently large t, we have:*

1. *If $k = o(\frac{\sqrt{n}}{\ln^{0.25} n})$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = 0) = 1 - o(1)$,*
2. *If $k = \Theta(\frac{\sqrt{n}}{\ln^{0.25} n})$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = 0) = \Theta(1)$, and*
3. *If $k = \omega(\frac{\sqrt{n}}{\ln^{0.25} n})$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^t, \pi_k^t) = O(\frac{k^2 \ln n}{n})) = 1 - o(1)$.*

The proof is similar to that of Theorem 2 and also consists of five steps.

Proof We will show that the theorem holds at t_0, where t_0 is an arbitrary time step after QS_1 in Algorithm 6 has run twice.

Consider the last completed run of *Local-sort* before t_0, which starts at t_2 and terminates at t_1. It is easy to see that $t_0 - t_1 = O(k\sqrt{\ln n})$ and $t_1 - t_2 = O(k\sqrt{\ln n})$. Also note that the input of *Local-sort* at t_2 comes from the last completed run of QS_2 before t_2, which starts at t_4 and terminates at t_3. By Lemma 4, with probability $1-o(1)$,

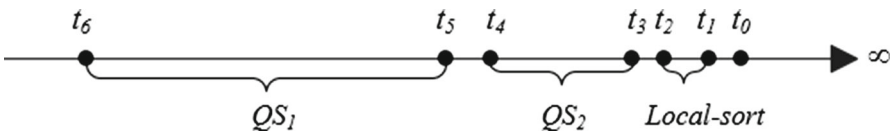

Fig. 2 The time steps $t_6, t_5, \ldots t_0$

$t_3 - t_4 = O(T)$ and $t_0 - t_3 = O(T)$, where $T = O(\ln^{1.5} n \ln\ln n)$ if $k = o(\ln^{1.5} n)$ and $T = O(k \ln k)$ otherwise. Likewise, the input of QS_2 at t_4 comes from the last completed run of QS_1 before t_4, which starts at t_6 and finishes at t_5. With probability $1 - o(1)$, $t_0 - t_5 = O(n \ln n)$ and $t_5 - t_6 = O(n \ln n)$. The time steps t_0 through to t_6 are illustrated in Fig. 2.

The following two results in the proof of Lemma 4 will be used. First, at any time t, the probability that a specified element is involved in some swap is $\Theta(\frac{1}{n})$. Second, with probability $1 - o(1)$, throughout $[t_6, t_0]$, no pair of elements whose ranks differ by more than $4\sqrt{\ln n}$ is swapped.

Step 1 We show that with probability $1 - o(1)$, $|\pi^t(u) - \tilde{\pi}^{t_5}(u)| \leq c' \ln^{1.5} n$ for all $u \in U$ and all $t \in [t_5, t_0]$, where c' is a constant.

Note that $|\pi^t(u) - \tilde{\pi}^{t_5}(u)| \leq |\pi^t(u) - \pi^{t_5}(u)| + |\pi^{t_5}(u) - \tilde{\pi}^{t_5}(u)|$, and $\Pr(|\pi^{t_5}(u) - \tilde{\pi}^{t_5}(u)| \leq c_1 \ln^{1.5} n) = 1 - o(\frac{1}{n^2})$ for some constant c_1 by Lemma 5. From Chernoff bound and the fact that the rank of u changes $O(\ln^{1.5} n)$ during $[t_5, t_0]$ in expectation, $\Pr(|\pi^t(u) - \pi^{t_5}(u)| \leq c_2 \ln^{1.5} n) = 1 - o(\frac{1}{n^2})$ for some constant c_2. Hence, due to union bound, $\Pr(\forall t \in [t_5, t_0], \forall u \in U, |\pi^t(u) - \tilde{\pi}^{t_5}(u)| \leq c' \ln^{1.5} n) = 1 - o(\frac{1}{n})$, where $c' = c_1 + c_2$.

As a result, event $\mathcal{E}_1$ happens with probability $1 - o(1)$, where $\mathcal{E}_1$ means that the set C produced at t_5 by QS_1, denoted as C^{t_5}, contains all the top k elements during $[t_5, t_0]$.

Step 2 We show that event $\mathcal{E}_2$ happens with probability $1 - o(1)$, where $\mathcal{E}_2$ stands for the event that the set P produced by QS_2 at t_3, denoted as P^{t_3}, exactly consists of the top k elements of U during $[t_3, t_0]$.

It follows from two facts. On the one hand, $\mathcal{E}_1$ happens with probability $1 - o(1)$. One the other hand, event $\mathcal{E}_3$ happens with probability $1 - O(\frac{T\sqrt{\ln n}}{n}) = 1 - o(1)$, where $\mathcal{E}_3$ means that no element in the neighborhood of the k-th element (by the true order π^{t_4}) is involved in any swap during $[t_4, t_0]$.

Therefore, in the following we'll assume that $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ all happen.

Now we prove that the theorem holds if $k = o(\ln^{1.5} n)$. When $k = o(\ln^{1.5} n)$, $|C| = O(\ln^{1.5} n)$. Therefore, with probability $1 - o(1)$, $t_0 - t_4 = O(\ln^{1.5} n \ln\ln n)$. When $t_0 - t_4 = O(\ln^{1.5} n \ln\ln n)$, with probability $1 - O(\frac{\ln^3 n \ln\ln n}{n}) = 1 - o(1)$, no element of C is involved in any swap throughout $[t_4, t_0]$. If the order of C does not change throughout $[t_4, t_0]$, the order on P^{t_3} estimated by QS_2 at t_3 is exactly $\pi^t|_{P^{t_3}}$ for all $t \in [t_3, t_0]$. Then, the order on P^{t_3} estimated by *Local-sort* at t_1 is $\pi^t|_{P^{t_3}}$ for $t \in [t_1, t_0]$, immediately leading to the theorem.

As a result, in the rest of the proof, we assume that $k = \Omega(\ln^{1.5} n)$.

Let $\tilde{\pi}_C^{t_3}$ be the order on C estimated by QS_2 at time t_3, and $\tilde{\pi}_P^{t_3} \triangleq \tilde{\pi}_C^{t_3}|_{P^{t_3}}$ be the induced order on P^{t_3}.

Step 3 We show that with probability $1 - o(1)$, $|\pi^t(u) - \tilde{\pi}_P^{t_3}(u)| \leq c\sqrt{\ln n}$ for all $u \in P^{t_3}$ and all $t \in [t_3, t_0]$, for a constant c.

To see why, first consider Y_i, the number of swaps that involve the element $(\pi^{t_4})^{-1}(i)$ during $[t_4, t_0]$. Since $t_0 - t_4 = O(k \ln k)$ in expectation, $\mathbb{E}[Y_i] = O(\frac{k \ln k}{n})$. Arbitrarily fix an constant $c_3 \geq e$. By Chernoff bound, $\Pr(Y_i \geq c_3) \leq (\mathbb{E}[Y_i])^{c_3}$. Because $k = O((\frac{n}{\ln n})^{1-\epsilon})$, there is a constant $0 < \beta < 1$ such that $\frac{k \ln k}{n} = O(k^{\beta-1})$. Arbitrarily choose a constant $c_4 > \max\{(1-\beta)^{-1}, c_3\}$, and we have $\Pr(\exists i \in [k], Y_i \geq c_4) \leq k(O(\frac{k \ln k}{n})^{c_4}) = O(k^{1+(\beta-1)c_4}) = o(1)$. Consequently, $\Pr(\forall i \in [k], Y_i < c_4) = 1 - o(1)$.

On the other hand, assume that the event $\mathcal{E}_4$ happens, which means that no swap during $[t_4, t_3]$ involves any element in v-neighborhood when v is a pivot selected by QS_2. The assumption does not lose generality, since $\Pr(\mathcal{E}_4) = 1 - O(\frac{k \ln k \sqrt{\ln n}}{n}) = 1 - o(1)$. Under this assumption, one can observe that for any pair (u, v), if $u < v$ always holds during $[t_4, t_3]$, then $u <_{\tilde{\pi}_C^{t_3}} v$ also holds. Hence, for any $u \in C$, the elements in C that are incorrectly ordered with respect to u constitute the set A as follows

$$A = \{v \in C : u <_{\tilde{\pi}_C^{t_3}} v, u >_{\pi^{t_3}} v, u <_{\pi^t} v \text{ for some } t \in [t_4, t_3)\}$$
$$\bigcup \{v \in C : u >_{\tilde{\pi}_C^{t_3}} v, u <_{\pi^{t_3}} v, u >_{\pi^t} v \text{ for some } t \in [t_4, t_3)\}$$

$|A|$ is upper-bounded by the number of elements that ever appear in the u-neighbourhood at some time step $t \in [t_4, t_0]$. Since each element is swapped at most c_4 times during $[t_4, t_0]$, there are at most $2c_4^2 \times 4\sqrt{\ln n} = 8c_4^2\sqrt{\ln n}$ elements that may be in the u-neighbourhood during $[t_4, t_0]$. Therefore, we have that $|\pi^t(u) - \tilde{\pi}_P^{t_3}(u)| \leq 8c_4^2\sqrt{\ln n}$ for any $t \in [t_4, t_0]$. In Algorithm 6, let $c = 8c_4^2$.

Step 4 We claim that with probability $1 - o(1)$, for any $u, v \in P^{t_3}$, $\tilde{\pi}_k^{t_1}(u) < \tilde{\pi}_k^{t_1}(v)$ if $\pi^t(u) < \pi^t(v)$ throughout $[t_2, t_1]$.

Since the probability that no swaps occur within u_{max}-neighbourhood during $[t_2, t_1]$ is $1 - O(\frac{k\sqrt{\ln n}}{n}) = 1 - o(1)$, we assume hereunder that no swaps occur within u_{max}-neighbourhood during $[t_2, t_1]$.

The claim can be proved in three cases. Assume that $\pi^t(u) < \pi^t(v)$ for all $t \in [t_2, t_1]$). Define $r_u \triangleq \tilde{\pi}_P^{t_3}(u)$ and $r_v \triangleq \tilde{\pi}_P^{t_3}(v)$.

- **Case 1** $u, v \in B_l$ for some l. Let l_0 be the largest such l. This implies that u_{max} of B_{l_0} is either u or v. Since no swaps occur within u_{max}-neighbourhood, the u_{max} computed by *Local-sort* cannot be v. This means that u_{max} of B_{l_0} is u and $\tilde{\pi}_k^{t_1}(u) = l_0 < \tilde{\pi}_k^{t_1}(v)$.
- **Case 2** $r_u < r_v$ and there is no l such that $u, v \in B_l$. There must be some $r_u - 4c\sqrt{\ln n} \leq l < r_v - 4c\sqrt{\ln n}$ such that u_{max} of B_l is u. Hence, $\tilde{\pi}_k^{t_1} = l < r_v - 4c\sqrt{\ln n} \leq \tilde{\pi}_k^{t_1}(v)$.
- **Case 3** $r_u > r_v$ and there is no l such that $u, v \in B_l$. We have $r_u - r_v < r_u - \tilde{\pi}^t(u) - r_v + \tilde{\pi}^t(v)$ for any $t \in [t_2, t_1]$, so $r_u - r_v < 2c\sqrt{\ln n}$. For any integer m, let $V_m = (\tilde{\pi}_P^{t_3})^{-1}(\{1, 2, \ldots, m + 4c\sqrt{\ln n}\})$. For all $r_v - 4c\sqrt{\ln n} \leq l < r_u - 4c\sqrt{\ln n}$, since $B_l \subseteq V_l$ and $|V_l \backslash V_{r_v - 6c\sqrt{\ln n} - 1}| \leq 4c\sqrt{\ln n}$, $|B_l \bigcap V_{r_v - 6c\sqrt{\ln n} - 1}| \geq 1$ holds.

For any $w \in B_l \bigcap V_{r_v - 6c\sqrt{\ln n}-1}$ and any $t \in [t_2, t_1]$, $\pi^t(w) \le \tilde{\pi}_P^{t_3}(w) + c\sqrt{\ln n} \le r_v - c\sqrt{\ln n} - 1 < \pi^t(v)$. By the proof of case 1 (with u replaced by w), u_{max} of B_l is not v for any $r_v - 4c\sqrt{\ln n} \le l < r_u - 4c\sqrt{\ln n}$. This means that there exists l such that $u, v \in B_l$, which is a contradiction.

Step 5 The claim in Step 4 means that $\Pr(\mathrm{KT}(\tilde{\pi}_k^{t_1}, \pi_k^{t_1}) \le 4Y\sqrt{\ln n}) = 1 - o(1)$, where Y is the number of swaps occurring in P during $[t_2, t_1]$. Since $\Pr(Y = 0) = (1-\frac{k}{n})^{O(k\sqrt{\ln n})}$, $\Pr(Y = 0) = 1 - o(1)$ if $k = o(\frac{\sqrt{n}}{\ln^{0.25} n})$, and $\Pr(Y = 0) = \Theta(1)$ if $k = \Theta(\frac{\sqrt{n}}{\ln^{0.25} n})$. When $k = \omega(\frac{\sqrt{n}}{\ln^{0.25} n})$, because $\mathbb{E}[Y] = O(\frac{k^2\sqrt{\ln n}}{n})$, $\Pr(\mathrm{KT}(\tilde{\pi}_k^{t_1}, \pi_k^{t_1}) = O(\frac{k^2 \ln n}{n})) = 1 - o(1)$ by Chernoff bound. Actually, we still have these results if $[t_2, t_0]$ is considered.

Altogether, the theorem holds. □

Except for the Gaussian distribution, d can also be determined by other discrete distributions, for example, $p(d) = \frac{\beta}{d^\gamma}$, where γ is a constant and β is a normalizing factor. When γ is large enough (say, $\gamma > 10$), the results similar to those in the Gaussian-swapping model can be obtained.

5 Conclusions

In this paper we identify the critical point k^* such that the top-k-selection problem can be solved error-free with high probability if and only if $k = o(k^*)$. A lower bound of the error when $k = \Omega(k^*)$ is also determined, which actually is tight under some condition. On the contrary, it is shown that the top-k-set problem can be solved error-free with probability $1 - o(1)$, for all $k \in [n]$. These results hold in the consecutive-swapping model and most of them can be extended to the Gaussian-swapping model.

A number of problems remain open for the top-k-selection problem in the consecutive-swapping model. For $\alpha = \omega(1)$, we have not shown whether the upper bound $O(\frac{k^2\alpha}{n})$ of error is tight when $k = \omega(\sqrt{\frac{n}{\alpha}})$. For $\alpha = O(1)$, there exists a gap between $k = n$ and $k = O((\frac{n}{\ln n})^{1-\epsilon})$, where the lower bound $\Omega(\frac{k^2}{n})$ of error has not yet shown to be tight. We conjecture that these bounds are tight.

Acknowledgements The work is partially supported by the National Key Research and Development Program of China (2016YFB1000201, 2016YFB1000604), State Key Laboratory of Software Development Environment Open Fund (SKLSDE-2016KF-01), Science Foundation of Shenzhen City in China (JCYJ20160419152942010), National Natural Science Foundation of China (61222202, 61433014, 61502449, 61602440), and the China National Program for support of Top-notch Young Professionals.

References

1. Anagnostopoulos, A., Kumar, R., Mahdian, M., Upfal, E.: Sort me if you can: how to sort dynamic data. In: 36th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 5566, pp. 339–350 (2009)
2. Ilyas, I., Beskales, G., Soliman, M.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. **40**(4), 11 (2008)
3. Whang, K., Kim, M., Lee, J.: Linear-time top-k sort method. US Patent 8,296,306 B1 (2012)

4. Knuth, D.E.: The Art of Computer Programming, vol. 3. Addison-Wesley, Reading (1973)
5. Kislitsyn, S.S.: On the selection of the k th element of an ordered set by pairwise comparison. Sibirskii Mat. Zhurnal **5**, 557–564 (1964)
6. Blum, M., Floyd, R., Pratt, V., Rivest, R., Tarjan, R.: Time bounds for selection. J. Comput. Syst. Sci. **7**(4), 448–461 (1973)
7. Dor, D., Zwick, U.: Selecting the median. In: SODA 1995, pp. 28–37 (1995)
8. Moreland, A.: Dynamic Data: Model, Sorting, Selection. Technical report (2014)
9. Anagnostopoulos, A., Kumar, R., Mahdian, M., Upfal, E., Vandin, F.: Algorithms on evolving graphs. In: 3rd Innovations in Theoretical Computer Science Conference (ITCS), pp. 149–160. ACM, New York (2012)
10. Bahmani, B., Kumar, R., Mahdian, M., Upfal, E.: Pagerank on an evolving graph. In: 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 24–32. ACM (2012)
11. Zhuang, H., Sun, Y., Tang, J., Zhang J., Sun, X.: Influence maximization in dynamic social networks. In: 13th IEEE International Conference on Data Mining (ICDM), pp. 1313–1318. IEEE (2013)
12. Kanade, V., Leonardos, N., Magniez, F.: Stable matching with evolving preferences (2015). arXiv:1509.01988
13. Labouseur, A.G., Olsen, P.W., Hwang, J.H.: Scalable and robust management of dynamic graph data. In: 1st International Workshop on Big Dynamic Distributed Data (BD3@VLDB), pp. 43–48 (2013)
14. Ren, C.: Algorithms for evolving graph analysis. Doctoral dissertation, The University of Hong Kong (2014)
15. Ajtai, M., Feldman, V., Hassidim, A., Nelson, J.: Sorting and selection with imprecise comparisons. In: 36th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 5566, pp. 37–48 (2009)
16. Feige, U., Raghavan, P., Peleg, D., Upfal, E.: Computing with noisy information. SIAM J. Comput. **23**(5), 1001–1018 (1994)
17. Finocchi, I., Grandoni, F., Italiano, G.: Optimal resilient sorting and searching in the presence of memory faults. In: 33th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 4051, pp. 286–298 (2006)
18. Finocchi, I., Italiano, G.: Sorting and searching in the presence of memory faults (without redundancy). In: 36th Annual ACM Symposium on Theory of Computing (STOC), pp. 101–110 (2004)
19. Erlebach, T., Hoffmann, M., Kammer, F.: On temporal graph exploration. In: 42th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 9134, pp. 444–455 (2015)
20. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS), pp. 1–16. ACM (2002)
21. Bressan, M., Peserico, E., Pretto, L.: Approximating PageRank locally with sublinear query complexity (2014). arXiv:1404.1864
22. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Mishima, T., Onizuka, M.: Fast and exact top-k algorithm for PageRank. In: 27th AAAI Conference on Artificial Intelligence, pp. 1106–1112 (2013)
23. Albers, S.: Online algorithms: a survey. Math. Programm. **97**(1–2), 3–26 (2003)
24. Kuleshov, V., Precup, D.: Algorithms for multi-armed bandit problems (2014). arXiv:1402.6028
25. Dubhashi, D.P., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, Cambridge (2009)