Noname manuscript No.
(will be inserted by the editor)

Derandomization for sliding window algorithms with strict correctness*

Moses Ganardi · Danny Hucke · Markus Lohrey

Received: date / Accepted: date

Abstract In the sliding window streaming model the goal is to compute an output value that only depends on the last n symbols from the data stream. Thereby, only space sublinear in the window size n should be used. Quite often randomization is used in order to achieve this goal. In the literature, one finds two different correctness criteria for randomized sliding window algorithms: (i) one can require that for every data stream and every time instant t, the algorithm computes a correct output value with high probability, or (ii) one can require that for every data stream the probability that the algorithm computes at every time instant a correct output value is high. Condition (ii) is stronger than (i) and is called "strict correctness" in this paper. The main result of this paper states that every strictly correct randomized sliding window algorithm can be derandomized without increasing the worst-case space consumption.

1 Introduction

Sliding window streaming algorithms process an input sequence $a_1a_2\cdots a_m$ from left to right and receive at time t the symbol a_t as input. Such algorithms are required to compute at each time instant t a value $f(a_{t-n+1}\cdots a_t)$ that depends on the n last symbols (we should assume $t \geq n$ here). The value n is called the *window size* and the sequence $a_{t-n+1}\cdots a_t$ is called the *window content* at time t. In many applications, data items in a stream are outdated after a certain time, and the sliding window model is a simple way to model this. A typical application is the analysis of a time series as it may arise in

* A conference version of this paper appeared in [17].

The first author has been supported by the DFG research project LO 748/13-1.

Universität Siegen
Hölderlinstrasse 3
57076 Siegen, Germany
E-mail: {ganardi,hucke,lohrey}@eti.uni-siegen.de

network monitoring, healthcare and patient monitoring, and transportation grid monitoring [3].

A general goal in the area of sliding window algorithms is to avoid the explicit storage of the window content, and, instead, to work in considerably smaller space, e.g. space polylogarithmic in the window size. In the seminal paper of Datar, Gionis, Indyk and Motwani [12], where the sliding window model was introduced, the authors prove that the number of 1's in a 0/1-sliding window of size n can be maintained in space $O(\frac{1}{\epsilon} \cdot \log^2 n)$ if one allows a multiplicative error of $1 \pm \epsilon$. Also a matching lower bound is shown. Other algorithmic problems that were addressed in the extensive literature on sliding window streams include the computation of statistical data (e.g. computation of the variance and k-median [5], and quantiles [4]), optimal sampling from sliding windows [9], membership problems for formal languages [13–16], computation of edit distances [10], database querying (e.g. processing of join queries over sliding windows [18]) and graph problems (e.g. checking for connectivity and computation of matchings, spanners, and minimum spanning trees [11]). The reader can find further references in [1, Chapter 8] and [8].

Many of the above mentioned papers deal with sliding window algorithms that only compute a good enough approximation of the exact value of interest. In fact, even for very simple sliding window problems it is unavoidable to store the whole window content. Examples are the exact computation of the number of 1's [12] or the computation of the first symbol of the sliding window for a 0/1-data stream [14]. In this paper, we consider a general model for sliding window *approximation problems*, where a (possibly infinite set) of *admissible output values* is fixed for each word. To be more accurate, a specific approximation problem is described by a relation $\Phi \subseteq \Sigma^* \times Y$ which associates to words over a finite alphabet Σ (the set of data values in the stream) admissible output values from a possibly infinite set Y. A sliding window algorithm for such a problem is then required to compute at each time instant an admissible output value for the current window content. This model covers exact algorithms (where Φ is a function $\Phi \colon \Sigma^* \to Y$) as well as a wide range of approximation algorithms. For example the computation of the number of 1's in a 0/1-sliding window with an allowed multiplicative error of $1 \pm \epsilon$ is covered by our model, since for a word with k occurrences of 1, the admissible output values are the integers between $(1 - \epsilon)k$ and $(1 + \epsilon)k$.

A second ingredient of many sliding window algorithms is randomization. Following our recent work [13–15] we model a randomized streaming algorithm for a given approximation problem as a *probabilistic automaton* over a finite alphabet. Probabilistic automata were introduced by Rabin [23] and can be seen as a common generalization of deterministic finite automata and Markov chains. The basic idea is that for every state q and every input symbol a, the next state is chosen according to some probability distribution. As an extension to the classical model of Rabin, states in a probabilistic automaton are not accepting or rejecting but are associated with output values from the set Y. This allows to associate with every input word $w \in \Sigma^*$ and every output value $y \in Y$ the probability that the automaton outputs y on input w.

In order to solve a specific approximation problem $\Phi \subseteq \Sigma^* \times Y$ in the sliding window model one should require that for a given window size n, a probabilistic automaton $\mathcal{P}_n$ should have a small error probability λ (say $\lambda = 1/3$) on every input stream. But what does the latter exactly mean? Two different definitions can be found in the literature:

– For every input stream $w = a_1 \cdots a_m$, the probability that $\mathcal{P}_n$ outputs on input w a value $y \in Y$ with $(a_{m-n+1} \cdots a_m, y) \notin \Phi$ is at most λ. Clearly an equivalent formulation is that, for all input streams $w = a_1 \cdots a_m$ and all $0 \le t \le m$ the probability that $\mathcal{P}_n$ outputs on input $a_1 \ldots a_t$ a value $y \in Y$ with $(a_{t-n+1} \cdots a_t, y) \notin \Phi$ is at most λ. In this case, we say that $\mathcal{P}_n$ is λ-*correct* for Φ and window size n.
– For every input stream $w = a_1 \cdots a_m$, the probability that $\mathcal{P}_n$ outputs at some time instant t $(n \le t \le m)$ a value $y \in Y$ with $(a_{t-n+1} \cdots a_t, y) \notin \Phi$ is at most λ. In this case, we say that $\mathcal{P}_n$ is *strictly* λ-*correct* for Φ and window size n.

One can rephrase the difference between strict λ-correctness and λ-correctness as follows: λ-correctness means that while the randomized sliding window algorithm runs on an input stream it returns at each time instant an admissible output value with probability at least $1 - \lambda$. In contrast, strict λ-correctness means that while the randomized sliding window algorithm reads an input stream, the probability that the algorithm returns an admissible output value at every time instant is at least $1 - \lambda$. Obviously this makes a difference: imagine that $\Omega = \{1, 2, 3, 4, 5, 6\}$ and that for every input word $w \in \Sigma^*$ the admissible output values are $2, 3, 4, 5, 6$, then the algorithm that returns at every time instant the output of a fair dice throw is $1/6$-correct. But the probability that this algorithm returns an admissible output value at every time instant is only $(5/6)^m$ for an input stream of length m and hence converges to 0 for $m \to \infty$. Of course, in general, the situation is more complex since successive output values of a randomized sliding window algorithm are not independent.

In the following discussion, let us fix the error probability $\lambda = 1/3$ (using probability amplification, one can reduce λ to any constant > 0). In our recent paper [15] we studied the space complexity of the membership problem for regular languages with respect to λ-correct randomized sliding window algorithms. It turned out that in this setting, one can gain from randomization. Consider for instance the regular language ab^* over the alphabet $\{a, b\}$. Thus, the sliding window algorithm for window size n should output “yes”, if the current window content is ab^{n-1} and “no” otherwise. From our results in [13, 14], it follows that the optimal space complexity of a *deterministic* sliding window algorithm for the membership problem for ab^* is $\Theta(\log n)$. On the other hand, it is shown in [15] that there is an λ-correct randomized sliding window algorithm for ab^* with (worst-case) space complexity $O(\log \log n)$ (this is also optimal). In fact, we proved in [15] that for every regular language L, the space optimal λ-correct randomized sliding window algorithm for L has either constant, doubly logarithmic, logarithmic, or linear space complexity, and the corresponding four space classes can be characterized in terms of simple syntactic properties.

Strict λ-correctness is used (without explicit mentioning) for instance in [7, 12].[1] In these papers, the lower bounds shown for deterministic sliding-window algorithms are extended with the help of Yao's minimax principle [24] to strictly λ-correct randomized sliding-window algorithms. The main result from the first part of the paper states that this is a general phenomenon: we show that every strictly λ-correct sliding window algorithm for an approximation problem Φ can be derandomized without increasing the worst-case space complexity (Theorem 1). To the best of our knowledge, this is the first investigation on the general power of randomization on the space consumption of sliding window algorithms. We emphasize that our proof does not utilize Yao's minimax principle, which would require the choice of a "hard" distribution of input streams specific to the problem. It remains open, whether such a hard distribution exists for every approximation problem.

We remark that the proof of Theorem 1 needs the fact that the sliding window algorithm is strictly correct on doubly exponentially long streams with high probability in order to derandomize it. In fact, we show that for a certain problem a restriction to polynomially long input streams yields an advantage of strictly correct randomized algorithms over deterministic ones, see Propositions 1 and 2. Whether such an advantage can be also obtained for input streams of length singly exponential in the window size remains open.

In the second part of the paper we come back to the problem of counting the number of 1's in a sliding window [7,12]. Datar et al. [12] proved a space lower bound of $\Omega(\frac{1}{\epsilon} \cdot \log^2 n)$ for approximating the number of 1's in a sliding window of size n with a multiplicative error of $1 \pm \epsilon$. This lower bound is first shown for deterministic algorithms and then, using Yao's minimax principle [24], extended to strictly λ-correct randomized sliding-window algorithms. We show that the same lower bound also holds for the wider class of λ-correct randomized sliding-window algorithms (Theorem 2). For the proof of this result we first show a lower bound for the one-way randomized communication complexity of the following problem: Alice holds m many ℓ bit numbers $a_1, \ldots, a_m$, and Bob holds an index $1 \leq i \leq m$ and an ℓ-bit number b. The goal of Bob is to find out whether $a_i > b$ holds. We show that Alice has to transfer at least $m\ell/3$ bits to Bob if the protocol has an error probability of at most $1/200$ (Theorem 4). From this result we can derive Theorem 2 using ideas from the lower bound proof in [12].

Let us add further remarks on our sliding window model. First of all, it is crucial for our proofs that the input alphabet (i.e., the set of data values in the input stream) is finite. This is for instance the case when counting the number of 1's in a 0/1-sliding window. On the other hand, the problem of computing the sum of all data values in a sliding window of arbitrary numbers (a problem

[1] For instance, Ben-Basat et al. write "We say that algorithm A is λ-correct on a input instance S if it is able to approximate the number of 1's in the last W bits, at every time instant while reading S, to within an additive error of $W\epsilon$".

that is considered in [12] as well) is not covered by our setting, unless one puts a bound on the size of the numbers in the input stream.

As a second remark, note that our sliding window model is non-uniform in the sense that for every window size we may have a different streaming algorithm. In other words: it is not required that there exists a single streaming algorithm that gets the window size as a parameter. Clearly, lower bounds get stronger when shown for the non-uniform model. Moreover, all proofs of lower bounds in the sliding window setting, we are aware of, hold for the non-uniform model.

2 Preliminaries

With $[0,1]$ we denote the real interval $\{p \in \mathbb{R} : 0 \leq p \leq 1\}$ of all probabilities. With log we always mean the logarithm to the base two.

The set of all words over a finite alphabet Σ is denoted by Σ^*. The empty word is denoted by ε. The length of a word $w \in \Sigma^*$ is denoted with $|w|$. The sets of words over Σ of length exactly, at most and at least n are denoted by Σ^n, $\Sigma^{\leq n}$ and $\Sigma^{\geq n}$, respectively. In the context of streaming algorithms, we also use the term “stream” for words.

2.1 Approximation problems

An *approximation problem* is a relation $\Phi \subseteq \Sigma^* \times Y$ where Σ is a finite alphabet and Y is a (possibly infinite) set of output values. The relation Φ associates with each word $w \in \Sigma^*$ a set of *admissible* or *correct* output values in Y. Typical examples include:

– exact computation problems $\varphi\colon \Sigma^* \to Y$ where we identify φ with its graph $\Phi = \{(w, \varphi(w)) : w \in \Sigma^*\}$. A typical example is the mapping $c_1\colon \{0,1\}^* \to \mathbb{N}$ where $c_1(w)$ is the number of 1's in w. Another exact problem is given by the characteristic function $\chi_L\colon \Sigma^* \to \{0,1\}$ of a language $L \subseteq \Sigma^*$ ($\chi_L(w) = 1$ if and only if $w \in L$).
– approximation of some numerical value for the data stream, which can be modeled by a relation $\Phi \subseteq \Sigma^* \times \mathbb{N}$. A typical example would be the problem $\{(w,k) : (1-\epsilon) \cdot c_1(w) \leq k \leq (1+\epsilon) \cdot c_1(w)\}$ for some $0 < \epsilon < 1$.

For a window length $n \geq 0$ and a stream $w \in \Sigma^*$ we define $\mathrm{last}_n(w)$ to be the suffix of $\Box^n w$ of length n where $\Box \in \Sigma$ is a fixed alphabet symbol. The word $\mathrm{last}_n(\varepsilon) = \Box^n$ is also called the *initial window*. To every approximation problem $\Phi \subseteq \Sigma^* \times Y$ we associate the sliding window problem

$$\mathrm{SW}_n(\Phi) = \{(x,y) \in \Sigma^* \times Y : (\mathrm{last}_n(x), y) \in \Phi\}$$

for window length n.

2.2 Probabilistic automata with output

In the following we will introduce probabilistic automata [22,23] as a model of randomized streaming algorithms which produce an output after each input symbol. A *randomized streaming algorithm* or a *probabilistic automaton* $\mathcal{P} = (Q, \Sigma, \iota, \rho, o)$ consists of a (possibly infinite) set of states Q, a finite alphabet Σ, an initial state distribution $\iota \colon Q \to [0,1]$, a transition probability function $\rho \colon Q \times \Sigma \times Q \to [0,1]$ and an output function $o \colon Q \to Y$ such that

- $\sum_{q \in Q} \iota(q) = 1$,
- $\sum_{q \in Q} \rho(p,a,q) = 1$ for all $p \in Q$, $a \in \Sigma$.

The *space* of the randomized streaming algorithm $\mathcal{P}$ (or the *number of bits used by* $\mathcal{P}$) is given by $s(\mathcal{P}) = \log |Q| \in \mathbb{R}_{\geq 0} \cup \{\infty\}$.

If ι and ρ map into $\{0,1\}$, then $\mathcal{P}$ is a *deterministic automaton*; in this case we write $\mathcal{P}$ as $\mathcal{P} = (Q, \Sigma, q_0, \delta, o)$, where $q_0 \in Q$ is the initial state and $\delta \colon Q \times \Sigma \to Q$ is the transition function. A *run* on a word $a_1 \cdots a_m \in \Sigma^*$ in $\mathcal{P}$ is a sequence $\pi = (q_0, a_1, q_1, a_2, \ldots, a_m, q_m)$ where $q_0, \ldots, q_m \in Q$ and $\rho(q_{i-1}, a_i, q_i) > 0$ for all $1 \leq i \leq m$. If $m = 0$ we obtain the empty run (q_0) starting and ending in q_0. We write runs in the usual way

$$\pi \colon q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} q_m$$

or also omit the intermediate states: $\pi \colon q_0 \xrightarrow{a_1 \cdots a_m} q_m$. We extend ρ to runs in the natural way: if $\pi \colon q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} q_m$ is a run in $\mathcal{P}$ then $\rho(\pi) = \prod_{i=1}^{m} \rho(q_{i-1}, a_i, q_i)$. Furthermore we define $\rho_\iota(\pi) = \iota(q_0) \cdot \rho(\pi)$. We denote by $\mathrm{Runs}(\mathcal{P}, w)$ the set of all runs on w in $\mathcal{P}$ and denote by $\mathrm{Runs}(\mathcal{P}, q, w)$ those runs on w that start in $q \in Q$. If $\mathcal{P}$ is clear from the context, we simply write $\mathrm{Runs}(w)$ and $\mathrm{Runs}(q, w)$. Notice that for each $w \in \Sigma^*$ the function ρ_ι is a probability distribution on $\mathrm{Runs}(\mathcal{P}, w)$ and for each $q \in Q$ the restriction of ρ to $\mathrm{Runs}(\mathcal{P}, q, w)$ is a probability distribution on $\mathrm{Runs}(\mathcal{P}, q, w)$. If Π is a set of runs (which will often be defined by a certain property of runs), then $\Pr_{\pi \in \mathrm{Runs}(w)}[\pi \in \Pi]$ denotes the probability $\sum_{\pi \in \mathrm{Runs}(w) \cap \Pi} \rho_\iota(\pi)$ and $\Pr_{\pi \in \mathrm{Runs}(q,w)}[\pi \in \Pi]$ denotes $\sum_{\pi \in \mathrm{Runs}(q,w) \cap \Pi} \rho(\pi)$.

2.3 Correctness definitions

Let $\mathcal{P} = (Q, \Sigma, \iota, \rho, o)$ be a randomized streaming algorithm with output function $o \colon Q \to Y$, let $\Phi \subseteq \Sigma^* \times Y$ be an approximation problem and let $w = a_1 a_2 \cdots a_m \in \Sigma^*$ be an input stream. Furthermore let $0 \leq \lambda \leq 1$ be an error probability.

- A run $\pi \colon q_0 \xrightarrow{w} q_m$ in $\mathcal{P}$ is *correct for* Φ if $(w, o(q_m)) \in \Phi$. We say that $\mathcal{P}$ is λ-*correct for* Φ if for all $w \in \Sigma^*$ we have

$$\Pr_{\pi \in \mathrm{Runs}(w)}[\pi \text{ is correct for } \Phi] \geq 1 - \lambda.$$

– A run $\pi \colon q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots q_{m-1} \xrightarrow{a_m} q_m$ in $\mathcal{P}$ on w is *strictly correct for Φ* if $(a_1 \cdots a_t, o(q_t)) \in \Phi$ for all $0 \le t \le m$. We say that $\mathcal{P}$ is *strictly λ-correct for Φ* if for all $w \in \Sigma^*$ we have

$$\Pr_{\pi \in \mathrm{Runs}(w)}[\pi \text{ is strictly correct for } \Phi] \ge 1 - \lambda.$$

A (strictly) λ-correct randomized streaming algorithm $\mathcal{P}_n$ for $\mathrm{SW}_n(\Phi)$ is also called a *(strictly) λ-correct randomized sliding window algorithm* for Φ and window size n. If $\mathcal{P}_n$ is deterministic and (strictly) 0-correct, we speak of a *deterministic* sliding window algorithm for Φ and window size n. The reader might think of having for every window size n a sliding window algorithm $\mathcal{P}_n$. We do not assume any uniformity here in the sense that the sliding window algorithms for different window sizes do not have to follow a common pattern. This is the same situation as in non-uniform circuit complexity, where one has for every input length n a circuit C_n and it is not required that the circuit C_n can be computed from n.

Remark 1 The trivial sliding window algorithm stores for window size n the window content with $\lceil \log |\Sigma| \rceil \cdot n$ bits. Hence every approximation problem has a deterministic sliding window algorithm $\mathcal{D}_n$ with $s(\mathcal{D}_n) \le \lceil \log |\Sigma| \rceil \cdot n$. In particular, for every (strictly) λ-correct randomized sliding window algorithm $\mathcal{P}_n$ for Φ and window size n, there exists a (strictly) λ-correct randomized sliding window algorithm $\mathcal{P}'_n$ for Φ and window size n such that

$$s(\mathcal{P}'_n) \le \min\{s(\mathcal{P}_n), \lceil \log |\Sigma| \rceil \cdot n\}.$$

3 Derandomization of strictly correct algorithms

In this section we prove the main result of this paper, which states that strictly correct randomized sliding window algorithms can be completely derandomized:

Theorem 1 *Let $\Phi \subseteq \Sigma^* \times Y$ be an approximation problem, $n \in \mathbb{N}$ be a window size and $0 \le \lambda < 1$ be an error probability. For every randomized sliding window algorithm $\mathcal{P}_n$ which is strictly λ-correct for Φ and window size n there exists a deterministic sliding window algorithm $\mathcal{D}_n$ for Φ and window size n such that $s(\mathcal{D}_n) \le s(\mathcal{P}_n)$.*

The proof idea is to successively construct a (doubly exponentially long) strictly correct run which reads all possible windows of length n from a certain subset of memory states. This strictly correct run then defines a deterministic algorithm which is always correct.

Let $\Phi \subseteq \Sigma^* \times Y$, $n \in \mathbb{N}$ be a window size and $0 \le \lambda < 1$ as in Theorem 1. Let $\mathcal{P}_n$ be a strictly λ-correct sliding window algorithm for Φ and window size n. By Remark 1, we can assume that $\mathcal{P}_n$ has a finite state set. Consider a run

$$\pi \colon q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} q_m$$

in $\mathcal{P}_n$. The run π is *simple* if $q_i \neq q_j$ for $0 \le i < j \le m$. A *subrun* of π is a run

$$q_i \xrightarrow{a_{i+1}} q_{i+1} \xrightarrow{a_{i+2}} \cdots q_{j-1} \xrightarrow{a_j} q_j$$

for some $0 \le i \le j \le m$. Consider a nonempty subset $S \subseteq Q$ and a function $\delta \colon Q \times \Sigma \to Q$ such that S is closed under δ, i.e., $\delta(S \times \Sigma) \subseteq S$. We say that the run π is *δ-conform* if $\delta(q_{i-1}, a_i) = q_i$ for all $1 \le i \le m$. We say that π is *(S,δ)-universal* if for all $q \in S$ and $x \in \Sigma^n$ there exists a δ-conform subrun $\pi' \colon q \xrightarrow{x} q'$ of π. Finally, π is *δ-universal* if it is (S,δ)-universal for some nonempty subset $S \subseteq Q$ which is closed under δ.

Lemma 1 *Let π be a strictly correct run in $\mathcal{P}_n$ for Φ, let $S \subseteq Q$ be a nonempty subset and let $\delta \colon Q \times \Sigma \to Q$ be a function such that S is closed under δ. If π is (S,δ)-universal, then there exists $q_0 \in S$ such that $\mathcal{D}_n = (Q, \Sigma, q_0, \delta, o)$ is a deterministic sliding window algorithm for Φ and window size n.*

Proof Let $q_0 = \delta(p, \Box^n) \in S$ for some arbitrary state $p \in S$ and define $\mathcal{D}_n = (Q, \Sigma, q_0, \delta, o)$. Let $w \in \Sigma^*$ and consider the run $\sigma \colon p \xrightarrow{\Box^n} q_0 \xrightarrow{w} q$ in $\mathcal{D}_n$ of length $\ge n$. We have to show that $(\mathrm{last}_n(w), o(q)) \in \Phi$. We can write $\Box^n w = x\, \mathrm{last}_n(w)$ for some $x \in \Sigma^*$. Hence, we can rewrite the run σ as $\sigma \colon p \xrightarrow{x} q' \xrightarrow{\mathrm{last}_n(w)} q$. We know that $q' \in S$ because S is closed under δ. Since π is (S,δ)-universal, it contains a subrun $q' \xrightarrow{\mathrm{last}_n(w)} q$. Strict correctness of π implies $(\mathrm{last}_n(w), o(q)) \in \Phi$. $\Box$

For the rest of this section we fix an arbitrary function $\delta \colon Q \times \Sigma \to Q$ such that for all $q \in Q$, $a \in \Sigma$,

$$\rho(q, a, \delta(q,a)) = \max\{\rho(q,a,p) \colon p \in Q\}.$$

Thus, we choose $\delta(q,a)$ as a most likely a-successor of q. Note that

$$\rho(q, a, \delta(q,a)) \ge \frac{1}{|Q|} \tag{1}$$

for all $q \in Q$, $a \in \Sigma$. Furthermore, let $\mathcal{D}_n = (Q, \Sigma, q_0, \delta, o)$ where the initial state q_0 will be defined later. We inductively define for each $i \ge 1$ a state p_i, a run π_i^* in $\mathcal{D}_n$ on some word $w_i \in \Sigma^*$, and a non-empty set $S_i \subseteq Q$, which is closed under δ. For $m \ge 0$, we abbreviate $\mathrm{Runs}(\mathcal{P}_n, w_1 \cdots w_m)$ by R_m. Note that $R_0 = \mathrm{Runs}(\mathcal{P}_n, \varepsilon)$. For $1 \le i \le m$ let H_i denote the event that for a random run $\pi = \pi_1 \cdots \pi_m \in R_m$, where each π_j is a run on w_j, the subrun π_i is (S_i, δ)-universal. Notice that H_i is independent of $m \ge i$.

First, we choose for p_i $(i \ge 1)$ a state that maximizes the probability

$$\Pr_{\pi \in R_{i-1}}[\pi \text{ ends in } p_i \mid \forall 1 \le j \le i-1 : \overline{H_j}],$$

which is at least $1/|Q|$. Note that p_1 is a state such that $\iota(p_1)$ is maximal, since R_0 only consists of empty runs (q). For S_i we take any maximal strongly connected component of $\mathcal{D}_n$ (viewed as a directed graph) which is reachable from p_i. As usual, strongly connected component means that for all $p, q \in S_i$ the state p is reachable from q and vice versa. Maximality means that for every $q \in S_i$ and every $a \in \Sigma$, also $\delta(q,a)$ belongs to S_i, i.e., S_i is closed under δ. Note that such a δ-closed strongly connected component must exist since Q is finite. Finally, we define the run π_i^* and the word w_i. The run π_i^* starts in p_i. Then, for each pair $(q,x) \in S_i \times \Sigma^n$ the run π_i^* leads from the current state to state q via a simple run and then reads the word x from q. The order in which we go over all pairs $(q,x) \in S_i \times \Sigma^n$ is not important. Since S_i is a maximal strongly connected component of $\mathcal{D}_n$ such a run π_i^* exists. Hence, π_i^* is a run on a word

$$w_i = \prod_{q \in S_i} \prod_{x \in \Sigma^n} y_{q,x}\, x,$$

where $y_{q,x}$ is the word that leads from the current state via a simple run to state q. Since we choose the runs on the words $y_{q,x}$ to be simple, we have $|y_{q,x}| \le |Q|$ and thus $|w_i| \le |Q| \cdot |\Sigma|^n \cdot (|Q| + n)$. Let us define

$$\mu = \frac{1}{|Q|^{|Q| \cdot |\Sigma|^n \cdot (|Q|+n)+1}}. \tag{2}$$

Note that by construction, the run π_i^* is (S_i, δ)-universal. Inequality (1) yields

$$\Pr_{\pi \in \mathrm{Runs}(p_i, w_i)}[\pi = \pi_i^*] \ge \frac{1}{|Q|^{|w_i|}} \ge \mu \cdot |Q|. \tag{3}$$

Lemma 2 *For all $m \ge 0$ we have $\Pr_{\pi \in R_m}[H_m \mid \forall i \le m-1 : \overline{H_i}] \ge \mu$.*

Proof In the following, let π be a random run from R_m and let π_i be the subrun on w_i. Under the assumption that the event $[\pi_{m-1}$ ends in $p_m]$ holds, the events $[\pi_m = \pi_m^*]$ and $[\forall i \le m-1 : \overline{H_i}]$ are conditionally independent.[2] Thus, we have

$$\begin{aligned} &\Pr_{\pi \in R_m}[\pi_m = \pi_m^* \mid \pi_{m-1} \text{ ends in } p_m \wedge \forall i \le m-1 : \overline{H_i}] \\ = &\Pr_{\pi \in R_m}[\pi_m = \pi_m^* \mid \pi_{m-1} \text{ ends in } p_m]. \end{aligned}$$

[2] Two events A and B are conditionally independent assuming event C if $\Pr[A \wedge B \mid C] = \Pr[A \mid C] \cdot \Pr[B \mid C]$, which is equivalent to $\Pr[A \mid B \wedge C] = \Pr[A \mid C]$.

Since the event $[\pi_m = \pi_m^*]$ implies the event $[\pi_{m-1}$ ends in $p_m]$ (recall that π_m^* starts in p_m) and π_m^* is (S_m, δ)-universal, we obtain:

$$\begin{aligned}
&\Pr_{\pi\in R_m}[H_m \mid \forall i \le m-1 : \overline{H_i}] \\
\ge\ &\Pr_{\pi\in R_m}[\pi_m = \pi_m^* \mid \forall i \le m-1 : \overline{H_i}] \\
=\ &\Pr_{\pi\in R_m}[\pi_m = \pi_m^* \wedge \pi_{m-1} \text{ ends in } p_m \mid \forall i \le m-1 : \overline{H_i}] \\
=\ &\Pr_{\pi\in R_m}[\pi_m = \pi_m^* \mid \pi_{m-1} \text{ ends in } p_m \wedge \forall i \le m-1 : \overline{H_i}] \cdot \\
&\Pr_{\pi\in R_m}[\pi_{m-1} \text{ ends in } p_m \mid \forall i \le m-1 : \overline{H_i}] \\
=\ &\Pr_{\pi\in R_m}[\pi_m = \pi_m^* \mid \pi_{m-1} \text{ ends in } p_m] \cdot \\
&\Pr_{\pi\in R_m}[\pi_{m-1} \text{ ends in } p_m \mid \forall i \le m-1 : \overline{H_i}] \\
\ge\ &\Pr_{\pi_m\in \mathrm{Runs}(p_m,w_m)}[\pi_m = \pi_m^*] \cdot \frac{1}{|Q|} \\
\ge\ &\mu,
\end{aligned}$$

where the last inequality follows from (3). This proves the lemma. □

Lemma 3 $\Pr_{\pi\in R_m}[\pi$ *is* δ*-universal*$] \ge \Pr_{\pi\in R_m}[\exists i \le m : H_i] \ge 1-(1-\mu)^m$.

Proof The first inequality follows from the definition of the event H_i. Moreover, with Lemma 2 we get

$$\begin{aligned}
\Pr_{\pi\in R_m}[\exists i \le m : H_i] &= \Pr_{\pi\in R_m}[\exists i \le m-1 : H_i] + \\
&\quad \Pr_{\pi\in R_m}[H_m \mid \forall i \le m-1 : \overline{H_i}] \cdot \Pr_{\pi\in R_m}[\forall i \le m-1 : \overline{H_i}] \\
&= \Pr_{\pi\in R_{m-1}}[\exists i \le m-1 : H_i] + \\
&\quad \Pr_{\pi\in R_m}[H_m \mid \forall i \le m-1 : \overline{H_i}] \cdot \Pr_{\pi\in R_{m-1}}[\forall i \le m-1 : \overline{H_i}] \\
&\ge \Pr_{\pi\in R_{m-1}}[\exists i \le m-1 : H_i] + \mu \cdot \Pr_{\pi\in R_{m-1}}[\forall i \le m-1 : \overline{H_i}].
\end{aligned}$$

Thus, $r_m := \Pr_{\pi\in R_m}[\exists i \le m : H_i]$ satisfies $r_m \ge r_{m-1} + \mu\cdot(1-r_{m-1}) = (1-\mu)\cdot r_{m-1} + \mu$. Since $r_0 = 0$, we get $r_m \ge 1-(1-\mu)^m$ by induction. □

We can now show our main theorem:

Proof (Theorem 1.) We use the probabilistic method in order to show that there exists $q_0 \in Q$ such that $\mathcal{D}_n = (Q, \Sigma, q_0, \delta, o)$ is a deterministic sliding window algorithm for Φ. With Lemma 3 we get

$$\begin{aligned}
&\Pr_{\pi \in R_m}[\pi \text{ is strictly correct for } \mathrm{SW}_n(\Phi) \text{ and } \delta\text{-universal}] \\
&= 1 - \Pr_{\pi \in R_m}[\pi \text{ is not strictly correct for } \mathrm{SW}_n(\Phi) \text{ or is not } \delta\text{-universal}] \\
&\geq 1 - \Pr_{\pi \in R_m}[\pi \text{ is not strictly correct for } \mathrm{SW}_n(\Phi)] - \Pr_{\pi \in R_m}[\pi \text{ is not } \delta\text{-universal}] \\
&\geq \Pr_{\pi \in R_m}[\pi \text{ is } \delta\text{-universal}] - \lambda \\
&\geq 1 - (1-\mu)^m - \lambda.
\end{aligned}$$

We have $1 - (1-\mu)^m - \lambda > 0$ for $m > \log(1-\lambda)/\log(1-\mu)$ (note that $\lambda < 1$ and $0 < \mu < 1$ since we can assume that $|Q| \geq 2$). Hence there are $m \geq 0$ and a strictly correct δ-universal run $\pi \in R_m$. We can conclude with Lemma 1. □

4 Polynomially long streams

The word $w_1 w_2 \cdots w_m$ (with $m > \log(1-\lambda)/\log(1-\mu)$) from the previous section, for which there exists a strictly correct and δ-universal run has a length that is doubly exponential in the window size n. To see this note that $0 > \ln(1-x) \geq x/(x-1)$ for $0 \leq x < 1$, which implies

$$m > \frac{\log(1-\lambda)}{\log(1-\mu)} = \frac{\ln(1-\lambda)}{\ln(1-\mu)} \geq \ln(1-\lambda)(\mu-1) \cdot \frac{1}{\mu}.$$

Here, $\ln(1-\lambda)$ is a negative constant and $\mu - 1$ is very close to -1. Moreover, $1/\mu$ grows doubly exponential in n by (2).

In other words: We need the fact that the sliding window algorithm is strictly correct on doubly exponentially long streams with high probability in order to derandomize the algorithm. In this section we show that at least we cannot reduce the length to poly(n): if we restrict to inputs of length poly(n) then strictly λ-correct sliding window algorithms can yield a proper space improvement over deterministic sliding window algorithms.

For a word $w = a_1 \cdots a_n$ let $w^{\mathsf{R}} = a_n \cdots a_1$ denote the reversed word. Take the language $K_{\text{pal}} = \{ww^{\mathsf{R}} : w \in \{a,b\}^*\}$ of all palindromes of even length, which belongs to the class **DLIN** of deterministic linear context-free languages [6], and let $L = \$K_{\text{pal}}$. As explained in Section 2.1 we identify L with the (exact) approximation problem $\chi_L \colon \{a,b,\$\}^* \to \{0,1\}$ where $\chi_L(w) = 1$ if and only if $w \in L$. We write L_n for $\mathrm{SW}_n(\chi_L)$. Note that the following proposition holds for arbitrarily long input streams.

Proposition 1 *Any deterministic sliding window algorithm for L and window size $2n+1$ uses $\Omega(n)$ space.*

Proof Let $\mathcal{D}_{2n+1}$ be a deterministic sliding window algorithm for L and window size $2n+1$, and take two distinct words $\$x$ and $\$y$ where $x, y \in \{a,b\}^n$. Since $\mathcal{D}_{2n+1}$ accepts $\$xx^{\mathsf{R}}$ and rejects $\$yx^{\mathsf{R}}$, the algorithm $\mathcal{D}_{2n+1}$ reaches two different states on the inputs $\$x$ and $\$y$. Therefore, D_{2n+1} must have at least $|\{a,b\}^n| = 2^n$ states and hence $\Omega(n)$ space. □

Proposition 2 *Fix a polynomial $p(n)$ and let $n \in \mathbb{N}$ be a window size. If n is large enough, there is a randomized streaming algorithm $\mathcal{P}_n$ with $s(\mathcal{P}_n) \le \mathcal{O}(\log n)$ such that*

$$\Pr_{\pi \in \mathrm{Runs}(\mathcal{P}_n, w)} [\pi \textit{ is strictly correct for } L_n] \ge 1 - 1/n$$

for all input words $w \in \Sigma^*$ with $|w| \le p(n)$.

Proof Babu et al. [6] have shown that for every language $K \in \mathbf{DLIN}$ there exists a randomized streaming algorithm using space $\mathcal{O}(\log n)$ which, given an input v of length n,

- accepts with probability 1 if $v \in K$,
- and rejects with probability at least $1 - 1/n$ if $v \notin K$.

We use this statement for the language $K_{\mathrm{pal}} \in \mathbf{DLIN}$. We remark that the algorithm needs to know the length of v in advance. To stay consistent with our definition, we view the above algorithm as a family $(\mathcal{S}_n)_{n \ge 0}$ of randomized streaming algorithms $\mathcal{S}_n$. Furthermore, the error probability $1/n$ can be further reduced to $1/(n+1)^d$ where d is chosen such that $p(n) \le n^d$ for sufficiently large n (by picking random primes of size $\Theta(n^{d+1})$ in the proof from [6]).

Now we prove our claim for $L = \$K_{\mathrm{pal}}$. The streaming algorithm $\mathcal{P}_n$ for window size n works as follows: After reading a \$-symbol, the algorithm $\mathcal{S}_{n-1}$ from above is simulated on the longest factor from $\{a, b\}^*$ that follows (i.e. $\mathcal{S}_{n-1}$ is simulated until the next \$ arrives). Simultaneously we maintain the length ℓ of the maximal suffix over $\{a, b\}$, up to n, using $\mathcal{O}(\log n)$ bits. If ℓ reaches $n-1$, then $\mathcal{P}_n$ accepts if and only if $\mathcal{S}_{n-1}$ accepts. Notice that $\mathcal{P}_n$ only errs if the stored length is $n-1$, which happens at most once in every n steps. Therefore the number of time instants where $\mathcal{P}_n$ errs on an input stream w of length $|w| \le p(n) \le n^d$ is at most $|w|/n \le n^d/n = n^{d-1}$ (if n is large enough). Moreover, at each of these time instants the error probability is at most $1/n^d$. By the union bound we have for every stream $w \in \{\$, a, b\}^{\le p(n)}$:

$$\Pr_{\pi \in \mathrm{Runs}(\mathcal{P}_n, w)} [\pi \text{ is not strictly correct for } L_n] \le n^{d-1} \cdot \frac{1}{n^d} = 1/n.$$

This concludes the proof. □

5 Lower bound for basic counting

For an approximation error $\epsilon > 0$ let us define the *basic counting problem*

$$C_{1,\epsilon} = \{(w, m) \in \{0,1\}^* \times \mathbb{N} : (1-\epsilon) \cdot c_1(w) \le m \le (1+\epsilon) \cdot c_1(w)\}$$

where $c_1(w)$ denotes the number of 1's in w. In [12] Datar, Gionis, Indyk and Motwani prove that any strictly λ-correct randomized sliding window algorithm for $C_{1,\epsilon}$ and window size n must use $\frac{k}{64} \log^2 \frac{n}{k} - \log(1-\lambda)$ bits where $k = \lfloor 1/\epsilon \rfloor$. We adapt their proof to show that the lower bound also holds for the weaker notion of λ-correct randomized sliding window algorithms.

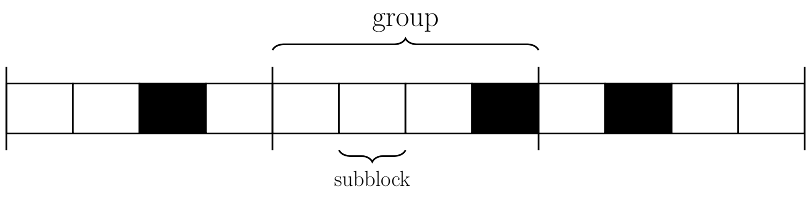

Fig. 1 A single block consisting of $B = 12$ subblocks divided into $k/4 = 3$ groups. The groups encode the numbers 2, 1, and 3 (from left to right).

Theorem 2 *Let $\epsilon > 0$ and $k = \lfloor 1/\epsilon \rfloor$. Every $1/200$-correct randomized sliding window algorithm for $C_{1,\epsilon}$ and window size $n \geq 4k$ must use $\frac{k}{48} \log^2(\frac{n}{k})$ many bits.*

In the statement above we can assume any algorithm with error probability $\lambda < 1/2$ using the *median trick*, see e.g. [2]: We run m copies of the algorithm in parallel and output the median of their outputs. Using the Chernoff bound we can choose m such that the median is a correct ϵ-approximation with error probability $1/200$. This reduces the space lower bound only by a constant.

For the rest of the section let us fix $n \in \mathbb{N}$ and $0 < \epsilon < 1$. Furthermore set $k = \lfloor 1/\epsilon \rfloor$. For the proof we use a reduction from a suitable communication problem. Let $f \colon A \times B \to \{0,1\}$ be a function. A *(one-round communication public-coin) protocol* $P = (m_A, m_B)$ with *cost* c consists of two functions $m_A \colon A \times R \to \{0,1\}^c$ and $m_B \colon \{0,1\}^c \times B \times R \to \{0,1\}$. Here R is a finite set of random choices equipped with a probability distribution. Given inputs $a \in A$ and $b \in B$ the protocol computes the random output $P(a,b) = m_B(m_A(a,r), b, r)$ where $r \in R$ at random. It computes f with error probability $\lambda < 1/2$ if

$$\Pr_{r \in R}[P(a,b) \neq f(a,b)] \leq \lambda$$

for all $a \in A, b \in B$. If $|R| = 1$ then P is *deterministic*.

We define the communication problem $\mathrm{GT}_{\ell,m}$ where Alice is given m many ℓ-bit numbers $a^{(1)}, \dots, a^{(m)}$, Bob is given a single ℓ-bit number b and an index $1 \leq p \leq m$, and the goal is to decide whether $a^{(p)} > b$. Formally, we view $\mathrm{GT}_{\ell,m}$ as a function

$$\mathrm{GT}_{\ell,m} \colon (\{0,1\}^\ell)^m \times (\{0,1\}^\ell \times \{1, \dots, m\}) \to \{0,1\}.$$

If $m = 1$ we write $\mathrm{GT}_\ell = \mathrm{GT}_{\ell,1}$.

Proposition 3 *Let $B = \sqrt{nk}$ such that $n \geq 4k$, and $j = \lfloor \log \frac{n}{B} \rfloor$. If $\mathcal{P}_n$ is a λ-correct sliding window algorithm for $C_{1,\epsilon}$ and window size n then there exists a one-round protocol for $\mathrm{GT}_{\log \frac{4B}{k}, \frac{jk}{4}}$ with cost $s(\mathcal{P}_n)$ and error probability λ.*

Proof In the following we ignore rounding issues. The idea is that Alice encodes her $jk/4$ many numbers by a bit stream consisting of $jk/4$ groups and feeding it into the sliding window algorithm. Then Bob can compare his number b with any of Alice's numbers $a^{(i)}$ with high probability, by sliding the window to the appropriate position.

As in [12] we partition the window of length n into j blocks of size

$$B, 2B, 4B, \dots, 2^{j-1}B$$

from right to left where $j = \lfloor \log \frac{n}{B} \rfloor$. Notice that $j \geq 1$ by our assumption that $n \geq 4k$. The blocks are numbered 0 to $j-1$ from right to left. The i-th block of length $2^i B$ is divided into B many subblocks of length 2^i. Each block is divided into $k/4$ groups consisting of $4B/k$ contiguous subblocks. In the following we choose from every group exactly one subblock which is filled with 1's; the remaining subblocks in the group are filled with 0's. An example is shown in Figure 1.

Let $M = \{1, \dots, 4B/k\}$. We will encode a tuple $\mathbf{a} = (a^{(1)}, \dots, a^{(jk/4)}) \in M^{jk/4}$ as a bit string of length n in unary encoding fashion as follows: For $a \in M$ and $0 \leq i \leq j-1$ define the bit string

$$u_i(a) = (0^{2^i})^{4B/k-a} 1^{2^i} (0^{2^i})^{a-1}$$

of length $2^i \cdot 4B/k$. For a tuple $\mathbf{a} = (a^{(1)}, \dots, a^{(jk/4)})$ over M of length $jk/4$ we define the *arrangement* $w(\mathbf{a}) \in \{0,1\}^n$ by

$$w(\mathbf{a}) = \prod_{i=0}^{j-1} \prod_{r=1}^{k/4} u_i(a^{(ik/4+r)})$$

where both concatenations are interpreted from right to left.

Datar et al. [12] argue that for any two distinct tuples $\mathbf{a}$ and $\mathbf{b}$, the arrangements $w(\mathbf{a})$ and $w(\mathbf{b})$ must be distinguished by a deterministic sliding window algorithm for $C_{1,\epsilon}$ and window length n.

We will present a communication protocol for $\mathrm{GT}_{\log \frac{4B}{k}, \frac{jk}{4}}$ based on a λ-correct sliding window algorithm $\mathcal{P}_n$ for $C_{1,\epsilon}$. Notice that $\log \frac{4B}{k} \geq 1$ by the assumption that $n \geq 4k$. Suppose that Alice holds the tuple $\mathbf{a} = (a^{(1)}, \dots, a^{(jk/4)})$ of numbers from M, Bob holds $b \in M$ and an index $1 \leq p \leq jk/4$. Their goal is to determine whether $a_p > b$. The protocol is defined as follows: Alice simulates $\mathcal{P}_n$ on $w(\mathbf{a})$ and sends the reached state to Bob, using $s(\mathcal{P}_n)$ bits. Suppose that $p = ik/4 + r$ for some $0 \leq i \leq j-1$ and $1 \leq r \leq k/4$. Bob then insert a suitable number of 0's in the stream such that the length-n window starts with the b-th subblock from the r-th group in the i-th block of $w(\mathbf{a})$. Notice that this is possible without knowing the tuple $\mathbf{a}$ because of the regular structure of arrangements which is known to Bob. The number of 1-bits in the obtained window is precisely

- $r \cdot 2^i + \frac{k}{4}(2^{i-1} + \cdots + 2^1 + 2^0)$ if $a_p \leq b$, and
- $(r-1) \cdot 2^i + \frac{k}{4}(2^{i-1} + \cdots + 2^1 + 2^0)$ if $a_p > b$.

Since the absolute approximation error is bounded by

$$\epsilon \frac{k}{4}(2^i + \cdots + 2^1 + 2^0) < \frac{2^{i+1}}{4} = 2^{i-1},$$

the two cases above can be distinguished by $\mathcal{P}_n$ with probability $1 - \lambda$. □

It remains to prove a lower bound for the one-round communication complexity of $\mathrm{GT}_{\ell,m}$. We start by showing that the one-round communication complexity of GT_n is $\Omega(n)$. This was already proven by Yao [25, Theorem 5]. More generally, Miltersen et al. showed that any r-round protocol for GT_n requires $\Omega(n^{1/r})$ bits using the *round elimination technique* [21]. We will first reprove the $\Omega(n)$ lower bound for GT_n, by directly plugging in $r = 1$ and GT_n into the proof of [21, Lemma 11]. Afterwards we adapt the proof to show the $\Omega(\ell m)$ lower bound for $\mathrm{GT}_{\ell,m}$.

Theorem 3 *Every one-round randomized protocol for* GT_n *with error probability* $1/200$ *has cost at least* $n/3$ *bits.*

Proof We follow the proof of [21, Lemma 11]. Consider a randomized one-round protocol for GT_n with error probability 1/200. The goal is to prove that the protocol must use $n/3$ bits.

By Yao's minimax principle [24] it suffices to exhibit a “hard” input distribution D on the set of inputs $\{0,1\}^n \times \{0,1\}^n$ and to prove that every deterministic protocol P with $\Pr_D[P(x,y) \neq \mathrm{GT}_n(x,y)] \leq 1/200$ must have cost $\Omega(n)$. See [20] for similar applications of Yao's minimax principle in the area of communication complexity.

For a bit string $x = x_1 \cdots x_n$ and an index $1 \leq i \leq n$ we define the bit string $\tau_i(x) = x_1 \cdots x_{i-1} 0 1^{n-i}$ of length n. Interpreted as binary numbers, we have the property

$$x > \tau_i(x) \iff x_i = 1. \tag{4}$$

The “hard” input distribution D is the uniform distribution on

$$\{(x, \tau_i(x)) \mid x \in \{0,1\}^n,\ 1 \leq i \leq n\}.$$

In other words, Alice holds a uniformly random string $x \in \{0,1\}^n$ and Bob holds $\tau_i(x)$ where the index $1 \leq i \leq n$ is also chosen uniformly at random and independently from x. By property (4) Bob needs to determine the value of x_i. Intuitively, the prefix $x_1 \cdots x_{i-1}$ of $\tau_i(x)$ does not help Bob, so this is basically the “index”-function, for which every one-round randomized protocol with error probability 1/3 has cost $\Omega(n)$ [19, Theorem 3.7].

Consider any deterministic protocol P with communication cost c such that

$$\Pr_D[P \text{ errs on } (x, \tau_i(x))] \leq 0.005.$$

Call an index i in x *bad* if P errs on $(x, \tau_i(x))$, and otherwise *good.* A uniformly random string $x \in \{0,1\}^n$ has at most $0.005n$ bad indices in expectation. By the Markov inequality we know

$$\Pr_{x \in \{0,1\}^n}[\text{number of bad indices of } x \geq 0.01n] \leq \frac{0.005n}{0.01n} = \frac{1}{2}.$$

Hence the set

$$R = \{x \in \{0,1\}^n \mid x \text{ has at most } 0.01n \text{ bad indices}\}$$

must contain at least 2^{n-1} bit strings. Since Alice sends at most c bits, she partitions R into at most 2^c subsets according to the bit string send to Bob. Let T be one of these subsets that has maximum cardinality. We have $|T| \geq |R|/2^c = 2^{n-1-c}$, i.e.,

$$c \geq n - 1 - \log |T|. \tag{5}$$

To prove $c \geq n/3$ we derive an upper bound on $|T|$.

In the following we successively construct a sequence of bits $a_1, \ldots, a_n$ and a sequence of nonempty sets $T = T_1 \supseteq T_2 \supseteq \cdots \supseteq T_{n+1}$ of n-bit strings such that all strings in T_i have the prefix $a_1 \cdots a_{i-1}$ (in particular, $|T_{n+1}| = 1$).

1. Set $T_1 := T$ and repeat the following for $i = 1, \ldots, n$:
2. Set $T_i^- := \{x \in T_i \mid i \text{ is bad in } x\}$ and $T_i^+ := \{x \in T_i \mid i \text{ is good in } x\}$.
3. If $|T_i^-| \geq 0.05 \cdot |T_i|$ choose $a_i \in \{0,1\}$ such that $|\{x \in T_i^- \mid x_i = a_i\}|$ is maximal. Then set $T_{i+1} := \{x \in T_i^- \mid x_i = a_i\}$.
4. Otherwise we have $|T_i^+| \geq 0.95 \cdot |T_i|$. All strings $x \in T_i^+$ must have the same i-th bit, say $x_i = a_i$, since for a string $x \in T_i^+$ Bob outputs correctly the bit x_i on input $(x, \tau_i(x))$. But since all strings in T_i^+ have the same prefix of length $i-1$ (and hence yield the same value under τ_i) and Alice communicates by definition of T the same message to Bob, Bob outputs the same bit for all $x \in T_i^+$. Set $T_{i+1} := T_i^+$.

Observe that all subsets $T_1, \ldots, T_{n+1}$ are nonempty. If point 3 is satisfied then $|T_{i+1}| \geq 0.5 \cdot 0.05|T_i| = 0.025|T_i|$, and the index i is bad in all strings of T_{i+1}. Hence, point 3 can only be satisfied at most $0.01n$ times by definition of R. If point 4 is satisfied then $|T_{i+1}| \geq 0.95|T_i|$. We can therefore bound

$$1 = |T_{n+1}| \geq 0.025^{0.01n} \cdot 0.95^{0.99n} \cdot |T| > 0.916^n \cdot |T|,$$

and thus $|T| \leq 1.0917^n$. By (5) we have established

$$c \geq (1 - \log 1.0917)n - 1 \geq 0.8735n - 1 \geq n/3.$$

for all $n \geq 2$. Clearly in the case $n = 1$ there is no zero-message randomized protocol for GT_1. □

Theorem 4 *Every one-round protocol for* $\mathrm{GT}_{\ell,m}$ *with error probability* $1/200$ *has cost at least* $\ell m/3$ *bits.*

Proof We adapt the proof above to $\mathrm{GT}_{\ell,m}$. To keep the notation consistent we view Alice's input as a single bit string $x = x^{(1)} \cdots x^{(m)} \in \{0,1\}^{\ell m}$ where $x^{(1)}, \ldots, x^{(m)} \in \{0,1\}^\ell$. We can write an index $1 \leq i \leq \ell m$ uniquely as $i = (p(i) - 1)\ell + r(i)$ where $1 \leq p(i) \leq m$ and $1 \leq r(i) \leq \ell$. For $1 \leq i \leq \ell m$ we define the bit string

$$\tau_i(x) = x_1^{(p(i))} \ldots x_{r(i)-1}^{(p(i))} 0 1^{\ell - r(i)}$$

of length ℓ. We have the property that

$$\mathrm{GT}_{\ell,m}(x, \tau_i(x), p(i)) = 1 \iff x_{r(i)}^{(p(i))} = 1.$$

The hard input distribution is the uniform distribution D over

$$\{(x, \tau_i(x), p(i)) : x \in \{0,1\}^{\ell m},\ 1 \le i \le \ell m\}.$$

Let P be a deterministic protocol with communication cost c such that

$$\Pr_D[P \text{ errs on } (x, \tau_i(x), p(i))] \le 0.005.$$

We call an index $1 \le i \le \ell m$ bad if P errs on $(x, \tau_i(x), p(i))$. Again we can find a set $T \subseteq \{0,1\}^{\ell m}$ such that

- $|T| \ge 2^{\ell m - 1 - c}$,
- all $x \in T$ have at most $0.01\ell m$ bad indices,
- and Alice sends the same message on all $x \in T$.

Using precisely the same arguments as in the previous proof we obtain $|T| \le 1.0917^{\ell m}$ and thus $c \ge \ell m/3$ whenever $\ell m \ge 2$. If $\ell m = 1$ then Alice must also send at least one bit in any communication protocol for $\mathrm{GT}_{\ell,m}$ with error probability 1/200. □

Theorem 2 now follows from Proposition 3 and Theorem 4: Let $B = \sqrt{nk}$ and $j = \lfloor \log \frac{n}{B} \rfloor \ge 1$. Then, any randomized 1/200-correct sliding window algorithm for $C_{1,\epsilon}$ and window size n must use at least

$$\frac{1}{3} \log\left(\frac{4B}{k}\right) \cdot \frac{jk}{4} = \frac{1}{3} \log\left(4\sqrt{\frac{n}{k}}\right) \cdot \left\lfloor \log\left(\sqrt{\frac{n}{k}}\right) \right\rfloor \cdot \frac{k}{4} \ge \frac{k}{48} \cdot \log^2\left(\frac{n}{k}\right)$$

many bits.

6 Open problems

In the proof of Theorem 1 we need the fact that the sliding window algorithm is strictly correct on doubly exponentially long streams with high probability. We pose the question whether this can be reduced to exponentially long streams.

Another open problem is whether one can extend Theorem 4 to arbitrary communication problems. For any function $f\colon A \times B \to \{0,1\}$ one can define an "indexed" version $f^{(m)}\colon A^m \times B \times \{1, \dots, m\} \to \{0,1\}$ where Alice holds a tuple $(a_1, \dots, a_m) \in A^m$, Bob holds $b \in B$ and $1 \le i \le m$ and their goal is to compute $f(a_i, b)$. The question is whether the one-round communication complexity of $f^{(m)}$ must be m times as large as the complexity of f, as it is the case for GT_n.

References

1. C. C. Aggarwal. *Data Streams - Models and Algorithms.* Springer, 2007.
2. N. Alon, Y. Matias, and M. Szegedy *The Space Complexity of Approximating the Frequency Moments. Journal of Computer and System Sciences*, 58(1):137–147, 1999.
3. H. C. M. Andrade, B. Gedik, and D. S. Turaga *Fundamentals of Stream Processing: Application Design, Systems, and Analytics.* Cambridge University Press, 2014.
4. A. Arasu and G. S. Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of PODS 2004*, pages 286–296. ACM, 2004.
5. B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of PODS 2003*, pages 234–243. ACM, 2003.
6. A. Babu, N. Limaye, J. Radhakrishnan, and G. Varma. Streaming algorithms for language recognition problems. *Theoretical Computer Science*, 494:13–23, 2013.
7. R. Ben-Basat, G. Einziger, R. Friedman, and Y. Kassner. Succinct Summing over Sliding Windows. *Algorithmica* 81(5): 2072–2091, 2019.
8. V. Braverman. Sliding window algorithms. In *Encyclopedia of Algorithms*, pages 2006–2011. Springer, 2016.
9. V. Braverman, R. Ostrovsky, and C. Zaniolo. Optimal sampling from sliding windows. *Journal of Computer and System Sciences*, 78(1):260–272, 2012.
10. H. Chan, T. W. Lam, L. Lee, J. Pan, H. Ting, and Q. Zhang. Edit distance to monotonicity in sliding windows. In *Proceedings of ISAAC 2011*, volume 7074 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2011.
11. M. S. Crouch, A. McGregor, and D. Stubbs. Dynamic graphs in the sliding-window model. In *Proceedings of ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013.
12. M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.
13. M. Ganardi, D. Hucke, D. König, M. Lohrey, and K. Mamouras. Automata theory on sliding windows. In *Proceedings of STACS 2018*, volume 96 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. to appear.
14. M. Ganardi, D. Hucke, and M. Lohrey. Querying regular languages over sliding windows. In *Proceedings of FSTTCS 2016*, volume 65 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
15. M. Ganardi, D. Hucke, and M. Lohrey. Randomized sliding window algorithms for regular languages. In *Proceedings of ICALP 2018*, volume 107 of *LIPIcs*, pages 127:1–127:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
16. M. Ganardi, A. Jez, and M. Lohrey. Sliding windows over context-free languages. In *Proceedings of MFCS 2018*, volume 117 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
17. M. Ganardi, D. Hucke, and M. Lohrey. Derandomization for Sliding Window Algorithms with Strict Correctness. In *Proceedings of CSR 2019*, volume 11532 of *Lecture Notes in Computer Science*, pages 237–249. Springer International Publishing, 2019.
18. L. Golab and M. T. Özsu. Processing sliding window multi-joins in continuous queries over data streams. In *Proceedings of VLDB 2003*, pages 500–511. Morgan Kaufmann, 2003.
19. Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.
20. Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.
21. P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson On data structures and asymmetric communication complexity. In *Proceedings of STOC 1995*, pages 103–111. ACM, 1995.
22. A. Paz. *Introduction to Probabilistic Automata.* Academic Press, 1971.
23. M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.
24. A. C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of FOCS 1977*, pages 222–227. IEEE Computer Society, 1977.
25. A. C. Yao. Lower Bounds by Probabilistic Arguments (Extended Abstract). In *Proceedings of FOCS 1983*, pages 420–428. IEEE Computer Society, 1983.