Derandomization for sliding window algorithms with strict correctness

Moses Ganardi[1], Danny Hucke[1], and Markus Lohrey[1]

Universität Siegen, Germany

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$a_{t-n+1}\cdots a_t$ is called the *window content* at time t. In many applications, data items in a stream are outdated after a certain time, and the sliding window model is a simple way to model this.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 medical monitoring, web tracking, or financial monitoring.

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 [10], 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 $O(\frac{1}{\varepsilon}\cdot\log^2 n)$ if one allows a multiplicative error of $1\pm\varepsilon$. 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 [3], and quantiles [2]), optimal sampling from sliding windows [7], membership problems for formal languages [11–14], computation of edit distances [8], database querying (e.g. processing of join queries over sliding windows [15]) and graph problems (e.g. checking for

connectivity and computation of matchings, spanners, and minimum spanning trees [9]). The reader can find further references in [1, Chapter 8] and [6].

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 [10] or the computation of the first symbol of the sliding window for a 0/1-data stream [12]. 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 function $\mathcal{A}\colon \Sigma^* \to 2^\Omega$ 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 Ω. 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 $\mathcal{A}$ is a function $\mathcal{A}\colon \Sigma^* \to \Omega$) 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 $1 \pm \varepsilon$ is covered by our model, since for a word with k occurrences of 1, the admissible output values are the integers from $\lfloor (1-\varepsilon)k \rfloor$ to $\lceil (1+\varepsilon)k \rceil$.

A second ingredient of many sliding window algorithms is randomization. Following our recent work [11–13] we model a randomized sliding window algorithm as a family $\mathcal{R} = (R_n)_{n \geq 0}$ of *probabilistic automata* R_n over a finite alphabet Σ, where R_n is the algorithm for window size n. Probabilistic automata were introduced by Rabin [17] and can 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. In addition to the classical model of Rabin, we require that for every probabilistic automaton R_n, (i) states are encoded by bit strings (the memory contents of the algorithm – this allows to define the space consumption of R_n on a certain input) and (ii) every state is associated with an output value from the set Ω. The second point allows to associate with every input word $w \in \Sigma^*$ and every output value $\omega \in \Omega$ the probability that the automaton outputs ω on input w. In order to solve a specific approximation problem $\mathcal{A}\colon \Sigma^* \to 2^\Omega$ one should require that for every window size n, the probabilistic automaton R_n should have a small error probability ϵ (say $\epsilon = 1/3$) on every input stream. But what does the latter exactly mean? Two different definitions can be found in the literature:

- For every input stream $w = a_1 \cdots a_m$ and every window size n ($n \leq m$), the probability that R_n outputs on input w a value $\omega \notin \mathcal{A}(a_{m-n+1} \cdots a_m)$ is at most ϵ. In this case, we say that $\mathcal{R}$ is *ϵ-correct* for $\mathcal{A}$.
- For every input stream $w = a_1 \cdots a_m$ and every window size n, the probability that R_n outputs at some time instant t ($n \leq t \leq m$) a value $\omega \notin \mathcal{A}(a_{t-n+1} \cdots a_t)$ is at most ϵ. In this case, we say that $\mathcal{R}$ is *strictly ϵ-correct* for $\mathcal{A}$.

One can rephrase the difference between strict ϵ-correctness and ϵ-correctness as follows: ϵ-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 $1 - \epsilon$. In contrast, strict ϵ-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 $1 - \epsilon$. 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 $\epsilon = 1/3$ (using probability amplification, one can reduce ϵ to any constant > 0). In our recent paper [13] we studied the space complexity of the membership problem for regular languages with respect to ϵ-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 [11, 12], 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 [13] that there is an ϵ-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 [13] that for every regular language L, the space optimal ϵ-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 ϵ-correctness is used (without explicit mentioning) for instance in [5, 10].[1] In these papers, the lower bounds shown for deterministic sliding-window algorithms are extended with the help of Yao's minimax principle [18] to strictly ϵ-correct randomized sliding-window algorithms. The main result of this paper states that this is a general phenomenon: we show that every strictly ϵ-correct sliding window algorithm for an approximation problem $\mathcal{A}$ can be derandomized without increasing the worst-case space complexity (Theorem 4). 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 4 uses exponentially long input streams in the size of the sliding window. In fact, we show that for a certain problem

[1] For instance, Ben-Basat et al. write "We say that algorithm A is ϵ-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 $W\epsilon$".

a restriction to polynomially long input streams yields an advantage of strictly correct randomized algorithms over deterministic ones, see Propositions 8 and 9.

It is possible to extend Theorem 4 to average space complexity with the cost of an additional constant factor. More precisely, for every randomized strictly ϵ-correct sliding window algorithm $\mathcal{R}$ for $\mathcal{A}$ there exists a deterministic sliding window algorithm for $\mathcal{A}$ whose space complexity is only a constant factor larger than the average space complexity of $\mathcal{R}$. This is a direct corollary of Theorem 4 and Lemma 3, which allows to go from average space to worst-case space with the cost of a constant blow-up.

Let us add further remarks on our 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 that is considered in [10] 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 $\{p \in \mathbb{R} \colon 0 \leq p \leq 1\}$. The set of all words over a finite alphabet Σ is denoted by Σ^*. The empty word is denoted by λ. 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.

2.1 Approximation problems

An *approximation problem* is a mapping $\mathcal{A} \colon \Sigma^* \to 2^\Omega$ where Σ is a finite alphabet and Ω is a (possibly infinite) set of output values. For a given input word $w \in \Sigma^*$ the set $\mathcal{A}(w)$ is the set of *admissible outputs* for w. Typical examples include:

- exact computation problems $\mathcal{A} \colon \Sigma^* \to \Omega$ (here we identify an element $\omega \in \Omega$ with the singleton subset $\{\omega\}$). 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 $L \subseteq \Sigma^*$.
- approximation of some numerical value for the data stream, which can be modeled by a function $\mathcal{A} : \Sigma^* \to 2^{\mathbb{N}}$. A typical example would be the mapping $w \mapsto \{k \in \mathbb{N} \colon (1-\epsilon) \cdot c_1(w) \leq k \leq (1+\epsilon) \cdot c_1(w)\}$ for some $0 < \epsilon < 1$.

2.2 Probabilistic automata with output

In the following we will introduce probabilistic automata [16, 17] as a model of randomized streaming algorithms which produce an output after each input symbol. A *probabilistic automaton* $R = (Q, \Sigma, \iota, \rho, \omega)$ 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 $\omega\colon Q \to \Omega$ such that

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

If ι and ρ map into $\{0,1\}$, then R is a *deterministic automaton*; in this case we write R as $R = (Q, \Sigma, q_0, \delta, \omega)$, where $q_0 \in Q$ is the initial state and $\delta : Q \times \Sigma \to Q$ is the transition function. A *run* on a word $a_1 \cdots a_m \in \Sigma^*$ in R 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

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

or also omit the intermediate states: $\pi : q_0 \xrightarrow{a_1 \cdots a_m} q_m$. We extend ρ to runs in the natural way: if $\pi : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} q_m$ is a run in R 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 $\mathrm{Runs}(R, w)$ the set of all runs on w in R and denote by $\mathrm{Runs}(R, q, w)$ those runs on w that start in $q \in Q$. If R 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 $\mathrm{Runs}(R, w)$ and for each $q \in Q$ the restriction of ρ to $\mathrm{Runs}(R, q, w)$ is a probability distribution on $\mathrm{Runs}(R, q, w)$. 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 $\sum_{\pi \in \mathrm{Runs}(q,w) \cap \Pi} \rho(\pi)$.

3 Randomized streaming and sliding window algorithms

We define a *randomized streaming algorithm* as a pair (R, enc) consisting of a probabilistic automaton $R = (Q, \Sigma, \iota, \rho, \omega)$ as above and an injective function $\mathrm{enc}\colon Q \to \{0,1\}^*$. Usually, we will only refer to the underlying automaton R. If R is deterministic, we speak of a *deterministic streaming algorithm*. The maximum number of bits stored in a run $\pi : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} q_m$ is denoted by $\mathrm{space}(R, \pi)$, i.e.,

$$\mathrm{space}(R, \pi) = \max\{|\mathrm{enc}(q_i)| : 0 \leq i \leq m\}.$$

We are interested in two space measures for an input stream w:

- worst case space:

$$\mathrm{space}(R, w) = \max\{\mathrm{space}(R, \pi) : \pi \in \mathrm{Runs}(R, w), \rho_\iota(\pi) > 0\}.$$

– expected space:

$$\text{space}_{\varnothing}(R, w) = \sum_{\pi \in \text{Runs}(R,w)} \rho_\iota(\pi) \cdot \text{space}(R, \pi)$$

Let $R = (Q, \Sigma, \iota, \rho, \omega)$ be a randomized streaming algorithm, let $\mathcal{A} : \Sigma^* \to 2^\Omega$ be an approximation problem and let $w = a_1 a_2 \cdots a_m \in \Sigma^*$ be an input stream.

– A run $\pi : q_0 \xrightarrow{w} q_m$ is *correct for* $\mathcal{A}$ if $\omega(q_m) \in \mathcal{A}(w)$. The *error probability* of R on w for $\mathcal{A}$ is

$$\epsilon(R, w, \mathcal{A}) = \sum_{\pi \in N} \rho_\iota(\pi),$$

where $N = \{\pi \in \text{Runs}(R, w) \colon \pi \text{ is not correct for } \mathcal{A}\}$.

– A run $\pi : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots q_{m-1} \xrightarrow{a_m} q_m$ is *strictly correct for* $\mathcal{A}$ if $\omega(q_t) \in \mathcal{A}(a_1 \cdots a_t)$ for all $0 \le t \le m$. The *strict error probability* of R on w for $\mathcal{A}$ is

$$\epsilon_*(R, w, \mathcal{A}) = \sum_{\pi \in N_*} \rho_\iota(\pi),$$

where $N_* = \{\pi \in \text{Runs}(R, w) \colon \pi \text{ is not strictly correct for } \mathcal{A}\}$.

3.1 Sliding window algorithms

For a window length $n \ge 0$ and a stream $w \in \Sigma^*$ we define $\text{last}_n(w)$ to be the suffix of $\square^n w$ of length n where $\square \in \Sigma$ is a fixed alphabet symbol. The word $\text{last}_n(\lambda) = \square^n$ is also called the *initial window*. Given an approximation problem $\mathcal{A} : \Sigma^* \to 2^\Omega$ and a window length $n \ge 0$ we define the sliding window problem $\mathcal{A}_n : \Sigma^* \to 2^\Omega$ as

$$\mathcal{A}_n(w) = \mathcal{A}(\text{last}_n(w))$$

for $w \in \Sigma^*$. Since we can identify a language $L \subseteq \Sigma^*$ with its characteristic function $\chi_L : \Sigma^* \to \{0, 1\}$, the definition of $\mathcal{A}_n$ specializes to

$$L_n = \{w \in \Sigma^* \colon \text{last}_n(w) \in L\}.$$

A *randomized sliding window algorithm* (*randomized SWA* for short) is a sequence $\mathcal{R} = (R_n)_{n \ge 0}$ of randomized streaming algorithms R_n over the same alphabet Σ and over the same set of output values Ω. If every R_n is deterministic, we speak of a *deterministic SWA*. Note that this is a non-uniform model in the sense that for every window length n we have a separate algorithm R_n. The *space complexity* of the randomized SWA $\mathcal{R} = (R_n)_{n \ge 0}$ is the function

$$f(\mathcal{R}, n) = \sup\{\text{space}(R_n, w) : w \in \Sigma^*\}$$

and its *expected space complexity* is the function

$$f_{\varnothing}(\mathcal{R}, n) = \sup\{\text{space}_{\varnothing}(R_n, w) : w \in \Sigma^*\}.$$

Clearly, if R_n is finite, then one can always find a state encoding such that $f(\mathcal{R}, n) = \lfloor \log_2 |R_n| \rfloor$ ($|R_n|$ denotes the number of states of R_n).

Definition 1. *Let $0 \leq \epsilon \leq 1$ be an error bound, let $\mathcal{R} = (R_n)_{n\geq 0}$ be a randomized SWA, and let $\mathcal{A}$ be an approximation problem.*

- *$\mathcal{R}$ is* ϵ-correct *for $\mathcal{A}$ if $\epsilon(R_n, w, \mathcal{A}_n) \leq \epsilon$ for all $n \geq 0$ and $w \in \Sigma^*$.*
- *$\mathcal{R}$ is* strictly ϵ-correct *for $\mathcal{A}$ if $\epsilon_*(R_n, w, \mathcal{A}_n) \leq \epsilon$ for all $n \geq 0$ and $w \in \Sigma^*$.*

A deterministic SWA for $\mathcal{A}$ is a deterministic SWA which is 0-correct (and hence strictly 0-correct) for $\mathcal{A}$.

Remark 2. Since one can store for window size n the window content with $\lceil \log_2 |\Sigma| \rceil \cdot n$ bits, every approximation problem has a deterministic SWA $\mathcal{D} = (D_n)_{n\geq 0}$ such that $f(\mathcal{D}, n) \leq \lceil \log_2 |\Sigma| \rceil \cdot n$. In particular, for every (strictly) ϵ-correct randomized SWA $\mathcal{R}$ for $\mathcal{A}$, there exists a (strictly) ϵ-correct randomized SWA $\mathcal{R}'$ for $\mathcal{A}$ such that $f(\mathcal{R}', n) \leq \min\{f(\mathcal{R}, n), \lceil \log_2 |\Sigma| \rceil \cdot n\}$.

It is not clear, whether the statement in Remark 2 also holds for average space complexity. On the other hand, the following statement holds:

Lemma 3. *Let $\mathcal{R}$ be a randomized SWA which is (strictly) ϵ-correct for $\mathcal{A}$ and let $\mu \geq 1$. Then there exists a randomized SWA $\mathcal{R}'$ which is (strictly) $(\epsilon + \frac{1}{\mu})$-correct for $\mathcal{A}$ such that $f(\mathcal{R}', n) \leq \mu \cdot f_\varnothing(\mathcal{R}, n)$.*

Proof. Fix $n \geq 0$ and let $s = f_\varnothing(\mathcal{R}, n)$ be the expected space complexity on window length n. If $s = \infty$ then the statement is trivial (take $R'_n = R_n$). So, let us assume that s is finite. Let $Q_{\geq}$ be the set of states in R_n with encoding length $\geq \mu \cdot s$. If $Q_{\geq} = \emptyset$, then $f(\mathcal{R}, n) \leq \mu \cdot f_\varnothing(\mathcal{R}, n)$ already holds. If $Q_{\geq}$ is nonempty, let R'_n be the algorithm obtained from R_n by merging all states $q \in Q_{\geq}$ into a single state $q_\perp$ encoded using an unused bit string of minimal length, which is at most $\mu \cdot s$. On an input stream $w \in \Sigma^*$, the probability that $q_\perp$ is reached is

$$\Pr_{\pi \in \mathrm{Runs}(R'_n, w)}[\pi \text{ contains } q_\perp] = \Pr_{\pi \in \mathrm{Runs}(R_n, w)}[\mathrm{space}(R_n, \pi) \geq \mu \cdot s] \leq \frac{s}{\mu \cdot s} = \frac{1}{\mu}$$

by Markov's inequality. Note that if an R'_n-run on w is not (strictly) correct, then (i) it must contain $q_\perp$ or (ii) it must be an R_n-run on w which is not (strictly) correct. Hence, an union bound yields

$$\begin{aligned} \epsilon(R'_n, w, \mathcal{A}) &\leq \frac{1}{\mu} + \epsilon(R_n, w, \mathcal{A}) \leq \frac{1}{\mu} + \epsilon, \text{ respectively} \\ \epsilon_*(R'_n, w, \mathcal{A}) &\leq \frac{1}{\mu} + \epsilon_*(R_n, w, \mathcal{A}) \leq \frac{1}{\mu} + \epsilon, \end{aligned}$$

which shows the lemma. □

4 Derandomization of strictly correct algorithms

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

Theorem 4. *Let $\mathcal{A}\colon \Sigma^* \to 2^\Omega$ be an approximation problem and let $\mathcal{R}$ be a randomized SWA which is strictly ϵ-correct for $\mathcal{A}$, where $0 \leq \epsilon < 1$. There exists a deterministic SWA $\mathcal{D}$ for $\mathcal{A}$ such that $f(\mathcal{D}, n) \leq f(\mathcal{R}, n)$ for all $n \geq 0$.*

Theorem 4 talks about the worst-case space complexity $f(\mathcal{R}, n)$. On the other hand, if we choose a constant $\mu \geq 1$ with $\epsilon + 1/\mu < 1$ then with Lemma 3 we obtain from a strictly ϵ-correct randomized SWA $\mathcal{R}$ for $\mathcal{A}$ a deterministic SWA $\mathcal{D}$ for $\mathcal{A}$ with $f(\mathcal{D}, n) \leq \mu \cdot f_\varnothing(\mathcal{R}, n)$.

Let $\mathcal{A}\colon \Sigma^* \to 2^\Omega$, $\mathcal{R} = (R_n)_{n\geq 0}$, and $0 \leq \epsilon < 1$ as in Theorem 4. By Remark 2, we can assume that every R_n has a finite state set. Fix a window size $n \geq 0$ and let $R_n = (Q, \Sigma, \iota, \rho, \omega)$. Consider a run

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

in R_n. The run π is *simple* if $q_i \neq q_j$ for $0 \leq i < j \leq m$. 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 $0 \leq i \leq j \leq m$. 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 $1 \leq i \leq m$. 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 $\pi' : q \xrightarrow{x} q'$ 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 5. *Let π be a strictly correct run in R_n for $\mathcal{A}$, 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 $D_n = (Q, \Sigma, q_0, \delta, \omega)$ is a deterministic streaming algorithm for $\mathcal{A}_n$.*

Proof. Let $q_0 = \delta(p, \square^n) \in S$ for some arbitrary state $p \in S$ and define $D_n = (Q, \Sigma, q_0, \delta, \omega)$. Let $w \in \Sigma^*$ and consider the run $\sigma : p \xrightarrow{\square^n} q_0 \xrightarrow{w} q$ in D_n of length $\geq n$. We have to show that $(\mathrm{last}_n(w), \omega(q)) \in \mathcal{A}$. We can write $\square^n w = x\,\mathrm{last}_n(w)$ for some $x \in \Sigma^*$. Thus, we can rewrite the run σ as $\sigma : p \xrightarrow{x} q' \xrightarrow{\mathrm{last}_n(w)} q$. 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 $(\mathrm{last}_n(w), \omega(q)) \in \mathcal{A}$. □

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

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

Note that

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

for all $q \in Q$, $a \in \Sigma$. Furthermore, let $D_n = (Q, \Sigma, q_0, \delta, \omega)$ where the initial state q_0 will be defined later. We define for each $i \geq 1$ a state p_i, a word $w_i \in \Sigma^*$, a

corresponding run $\pi_i^* \in \mathrm{Runs}(D_n, p_i, w_i)$ in D_n and a set $S_i \subseteq Q$. For $m \geq 0$, we use the abbreviation

$$\Pi_m = \mathrm{Runs}(R_n, w_1 \cdots w_m).$$

Note that $\Pi_0 = \mathrm{Runs}(R_n, \lambda)$. For $1 \leq i \leq m$ let H_i denote the event that for a random run $\pi = \pi_1 \cdots \pi_m \in \Pi_m$, 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 $m \geq i$.

First, we choose for p_i a state that maximizes the probability

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

which is at least $1/|Q|$. Note that p_1 is a state such that $\iota(p_1)$ is maximal, since Π_0 only consists of empty runs (q). For S_i we take any maximal strongly connected component of D_n (viewed as a directed graph) which is reachable from p_i. Here, maximality means that for every $q \in S_i$ and every $a \in \Sigma$, also $\delta(q, a)$ belongs to S_i. 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 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 $|y_{q,x}| \leq |Q|$ and thus $|w_i| \leq |Q| \cdot |\Sigma|^n \cdot (|Q| + n)$. Let us define

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

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

$$\Pr_{\pi \in \mathrm{Runs}(p_i, w_i)}[\pi = \pi_i^*] \geq \frac{1}{|Q|^{|w_i|}} \geq \mu \cdot |Q|. \tag{2}$$

Lemma 6. *For all $m \geq 0$ we have $\Pr_{\pi \in \Pi_m}[H_m \mid \forall i \leq m-1 : \overline{H_i}] \geq \mu$.*

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

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

[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 $p_m]$, we obtain:

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

where the last inequality follows from (2). This proves the lemma. □

Lemma 7. $\Pr_{\pi\in\Pi_m}[\pi$ *is* δ*-universal*$] \ge \Pr_{\pi\in\Pi_m}[\exists i \le m : H_i] \ge 1-(1-\mu)^m$.

Proof. The first inequality follows from the definition of the event H_i. Moreover, with Lemma 6 we get

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

Thus, $r_m := \Pr_{\pi\in\Pi_m}[\exists i \le m : H_i]$ 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. □

Proof of Theorem 4. We use the probabilistic method in order to show that there exists $q_0 \in Q$ such that $D_n = (Q, \Sigma, q_0, \delta, \omega)$ is a deterministic streaming algorithm for $\mathcal{A}_n$. With Lemma 7 we get

$$\begin{aligned}
&\Pr_{\pi\in\Pi_m}[\pi \text{ is strictly correct for } \mathcal{A} \text{ and } \delta\text{-universal}] \\
=\ &1 - \Pr_{\pi\in\Pi_m}[\pi \text{ is not strictly correct for } \mathcal{A} \text{ or is not } \delta\text{-universal}] \\
\ge\ &1 - \Pr_{\pi\in\Pi_m}[\pi \text{ is not strictly correct for } \mathcal{A}] - \Pr_{\pi\in\Pi_m}[\pi \text{ is not } \delta\text{-universal}] \\
\ge\ &\Pr_{\pi\in\Pi_m}[\pi \text{ is } \delta\text{-universal}] - \epsilon
\end{aligned}$$

$\geq 1 - (1-\mu)^m - \epsilon$.

We have $1 - (1-\mu)^m - \epsilon > 0$ for $m > \log(1-\epsilon)/\log(1-\mu)$ (note that $\epsilon < 1$ 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 $\pi \in \Pi_m$. We can conclude with Lemma 5. □

The word $w_1 w_2 \cdots w_m$ (with $m > \log(1-\epsilon)/\log(1-\mu)$), for which there exists a strictly correct and δ-universal run has a length that is exponential in the window size n. In other words: We need words of length exponential in n in order to transform a strictly ϵ-correct randomized SWA into an equivalent deterministic SWA. We now show that this is unavoidable: if we restrict to inputs of length $\mathrm{poly}(n)$ then strictly ϵ-correct SWAs can yield a proper space improvement over deterministic SWAs.

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 $K_{\mathrm{pal}} = \{ww^{\mathsf{R}} : w \in \{a,b\}^*\}$ of all palindromes of even length, which belongs to the class **DLIN** of deterministic linear context-free languages [4], and let $L = \$K_{\mathrm{pal}}$.

Proposition 8. *If $\mathcal{D}$ is a deterministic SWA for L, then $f(\mathcal{D}, 2n+1) = \Omega(n)$.*

Proof. Let $\mathcal{D} = (D_n)_{n \geq 0}$. Take two distinct words $\$x$ and $\$y$ where $x, y \in \{a,b\}^n$. Since D_{2n+1} accepts $\$xx^{\mathsf{R}}$ and rejects $\$yx^{\mathsf{R}}$, the automaton 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 9. *Fix a polynomial $p(n)$. There is a randomized SWA $\mathcal{R} = (R_n)_{n \geq 0}$ such that (i) $f(\mathcal{R}, n) \in \mathcal{O}(\log n)$ and (ii) $\epsilon_*(R_n, w, L_n) \leq 1/e$ (e denotes Euler's number) for all input words $w \in \Sigma^*$ with $|w| \leq p(n)$.*

Proof. Babu et al. [4] have shown that for every language $K \in$ **DLIN** there exists a randomized streaming algorithm using space $\mathcal{O}(\log n)$ which, given an input w of length n,

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

We remark that the algorithm needs to know the length of w in advance. To stay consistent with our definition, we view the algorithm above as a family $(S_n)_{n \geq 0}$ of randomized streaming algorithms S_n. Furthermore, the error probability $1/n$ can be further reduced to $1/n^d$ where $p(n) \leq n^d$ for sufficiently large n (by picking random primes of size $\Theta(n^{d+1})$ in the proof from [4]).

Now we prove our claim for $L = \$K_{\mathrm{pal}}$. The streaming algorithm R_n for window size n works as follows: After reading a \$-symbol, the algorithm S_{n-1} from above is simulated on the longest factor from $\{a,b\}^*$ that follows (i.e. 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 R_n accepts if and only if S_{n-1} accepts. Notice that R_n only errs if the stored length is $n-1$ (with probability $1/n^d$), which happens at most once in every n steps. Therefore the number of time instants where R_n errs on w is at most $|w|/n \leq n^d/n = n^{d-1}$. The union bound yields $\epsilon_*(R_n, w, L_n) \leq n^{d-1}/n^d = \frac{1}{n}$ for every stream $w \in \{\$, a, b\}^{\leq p(n)}$. This concludes the proof. □

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

References

1. C. C. Aggarwal. *Data Streams - Models and Algorithms*. Springer, 2007.
2. A. Arasu and G. S. Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of PODS 2004*, pages 286–296. ACM, 2004.
3. 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.
4. A. Babu, N. Limaye, J. Radhakrishnan, and G. Varma. Streaming algorithms for language recognition problems. *Theoretical Computer Science*, 494:13–23, 2013.
5. R. Ben-Basat, G. Einziger, R. Friedman, and Y. Kassner. Efficient summing over sliding windows. In *Proceedings of SWAT 2016*, volume 53 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
6. V. Braverman. Sliding window algorithms. In *Encyclopedia of Algorithms*, pages 2006–2011. Springer, 2016.
7. V. Braverman, R. Ostrovsky, and C. Zaniolo. Optimal sampling from sliding windows. *Journal of Computer and System Sciences*, 78(1):260–272, 2012.
8. 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.
9. 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.
10. M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. A. Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.
17. M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.
18. A. C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of FOCS 1977*, pages 222–227. IEEE Computer Society, 1977.