Classical and Quantum Computations with Restricted Memory

Farid Ablayev[1(✉)], Marat Ablayev[1], Kamil Khadiev[1,2], and Alexander Vasiliev[1]

[1] Kazan Federal University, Kazan, Russia
fablayev@gmail.com, mablayev@gmail.com, kamilhadi@gmail.com, alexander.ksu@gmail.com
[2] University of Latvia, Riga, Latvia

Abstract. Automata and branching programs are known models of computation with restricted memory. These models of computation were in focus of a large number of researchers during the last decades. Streaming algorithms are a modern model of computation with restricted memory. In this paper, we present recent results on the comparative computational power of quantum and classical models of branching programs and streaming algorithms.

In addition to comparative complexity results, we present a quantum branching program for computing a practically important quantum function (quantum hash function) and prove optimality of this algorithm.

Keywords: Complexity · Quantum computing
Branching programs · OBDDs · Hierarchy · Hashing

1 Introduction

The class of problems solvable with restricted memory and efficient algorithms (classical and quantum) for such problems are of great interest nowadays. There are many different models that process data streams and use few memory resources: automata, branching programs (BPs), or streaming algorithms. Streaming algorithms are a new model in computer science, while branching programs and automata were in the focus of a great number of computer science researchers during the last decades.

Our research group is focused on the study of automata, branching programs with the OBDD type restrictions, and models of streaming algorithms. These research interests largely coincide with the research interests of Juraj Hromkovič and his research group.

In this paper, we present an overview of our results on classical and quantum restricted computational models, especially on the comparison of the computational power of classical and quantum models.

Branching programs are a well-known model of computation that has proven useful in a variety of domains such as hardware verification, model checking, and

H.-J. Böckenhauer et al. (Eds.): Hromkovič Festschrift, LNCS 11011, pp. 129–155, 2018.
https://doi.org/10.1007/978-3-319-98355-4_9

other applications [64]. It is known that the class of Boolean functions computed by polynomial-sized branching programs coincides with the class of functions computed by non-uniform log-space Turing machines. For theoretical computer science the model of branching programs is interesting, in particular, because of the possibility to define different natural kinds of restrictions on branching programs. It is interesting that the computational model, which is known in theory as branching program was independently defined and investigated in applied computer science as well.

Streaming algorithms are a new model of computation that came to play as a tool for investigations of big data processing. This direction or research uses in part methods developed for the model of branching programs. We show that such connections (between the streaming algorithm model and the branching program model) are very close.

The paper is organized as follows. We start with the definition of various models of branching programs and hierarchy results for these models. Note that hierarchy results need both lower bound and upper bound techniques. To prove some hierarchy one needs appropriate examples of functions that separate different classes. We present a bunch of such functions and use them for hierarchy proofs. To show the practical importance of the model of branching programs one needs examples of practically important functions that can be efficiently computed by BPs. We present a quantum hash function and show a corresponding quantum branching program that is optimal for such a function.

2 Branching Programs

One of the important restricted variants of branching programs are oblivious read-once branching programs or *ordered binary decision diagrams* (OBDD) [64]. The OBDD model can be considered as a nonuniform automaton (see, for example, [10]) and is a good model for data streaming algorithms.

2.1 Definitions

A branching program over the set X of n Boolean variables is a directed acyclic graph with two distinguished nodes s (a source node) and t (a sink node). We denote such a program by $P_{s,t}$ or simply P. Each inner node v of P is associated with a variable $x \in X$. A *deterministic* P has exactly two outgoing edges labeled $x = 0$ and $x = 1$ respectively for such a node v. The program P computes the Boolean function $f(X)$ ($f\colon \{0,1\}^n \to \{0,1\}$) as follows: for each $\sigma \in \{0,1\}^n$ we let $f(\sigma) = 1$ if and only if there exists at least one $s - t$ path (called *accepting path* for σ) such that all edges along this path are consistent with σ.

A branching program is *leveled* if the nodes can be partitioned into levels $V_1, \ldots, V_\ell$ and a level $V_{\ell+1}$ such that the nodes in $V_{\ell+1}$ are the sink nodes, and the nodes in each level V_j with $j \leq \ell$ have outgoing edges only to nodes in the next level V_{j+1}. For a leveled $P_{s,t}$ the source node s is a node from the first level V_1 of nodes and the sink node t is a node from the last level $V_{\ell+1}$.

The *width* $width(P)$ of a leveled branching program P is the maximum number of nodes over all levels of P, $width(P) = \max_{1 \le j \le \ell} |V_j|$. The *size* of a branching program P is the number of nodes in P.

A leveled branching program is called *oblivious* if all inner nodes of each level are labeled by the same variable. A branching program is called *read once* if each variable is tested on each path only once. An oblivious leveled read once branching program is also called *ordered binary decision diagram* (OBDD). OBDD P reads variables in its individual order $\pi = (j_1, \ldots, j_n)$, $\pi(i) = j_i$; $\pi^{-1}(j)$ is the position of j in permutation π. We call $\pi(P)$ the order of P. Let us denote the natural order by $id = (1, \ldots, n)$. Sometimes we will use the notation id-OBDD P, which means that $\pi(P) = id$. Let $width(f) = \min_P width(P)$ for an OBDD P which computes f, and $id\text{-}width(f)$ is the same but for id-OBDD.

A branching program P is called a k-OBDD if it consists of k layers, where the i-th $(1 \le i \le k)$ layer P^i of P is an OBDD. Let π_i be an order of P^i, where $1 \le i \le k$ and $\pi_1 = \cdots = \pi_k = \pi$. We call $\pi(P) = \pi$ the order of P.

A *nondeterministic* OBDD (NOBDD) is the nondeterministic counterpart of OBDD. A *probabilistic* OBDD (POBDD) can have more than two outgoing edges for a node, and the choice between them is done probabilistically during computation. A POBDD P computes a Boolean function f with bounded error $0.5 - \varepsilon$ if the probability of a correct answer is at least $0.5 + \varepsilon$.

Next, we recall the definition of a quantum version of the branching program model [4,5].

Definition 1. *A* quantum branching program Q *over the Hilbert space* $\mathcal{H}^d$ *is defined as*

$$Q = \langle \mathbb{T}, |\psi_0\rangle, \text{Accept} \rangle \ , \tag{1}$$

where $\mathbb{T}$ *is a sequence of* l *instructions:* $\mathbb{T}_j = \big(x_{i_j}, U_j(0), U_j(1)\big)$ *is determined by the variable* x_{i_j} *tested on the step* j*, and* $U_j(0)$, $U_j(1)$ *are unitary transformations in* $\mathcal{H}^d$*.*

Vectors $|\psi\rangle \in \mathcal{H}^d$ *are called states (state vectors) of* Q*,* $|\psi_0\rangle \in \mathcal{H}^d$ *is the initial state of* Q*, and* Accept $\subseteq \{1, 2, \ldots, d\}$ *is the set of indices of accepting basis states.*

We define a computation of Q *on an input* $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^n$ *as follows:*

1. *A computation of* Q *starts from the initial state* $|\psi_0\rangle$*;*
2. *The* j*-th instruction of* Q *reads the input symbol* σ_{i_j} *(the value of* x_{i_j}*) and applies the transition matrix* $U_j = U_j(\sigma_{i_j})$ *to the current state* $|\psi\rangle$ *to obtain the state* $|\psi'\rangle = U_j(\sigma_{i_j})|\psi\rangle$*;*
3. *The final state is*

$$|\psi_\sigma\rangle = \left(\prod_{j=l}^{1} U_j(\sigma_{i_j}) \right) |\psi_0\rangle \ . \tag{2}$$

4. *After the* l*-th (last) step of quantum transformation the configuration* $|\psi_\sigma\rangle = (\alpha_1, \ldots, \alpha_d)^T$ *of* Q *is measured, and the input* σ *is accepted if and only if the result is in* Accept*, which happens with probability*

$$\Pr_{\text{accept}}(\sigma) = \sum_{i \in \text{Accept}} |\alpha_i|^2 \ . \tag{3}$$

We say that a Boolean function f is computed by Q with *bounded error* if there exists an $\varepsilon \in (0, \frac{1}{2}]$ such that Q accepts all inputs from $f^{-1}(1)$ with probability at least $\frac{1}{2} + \varepsilon$ and Q accepts all inputs from $f^{-1}(0)$ with probability at most $\frac{1}{2} - \varepsilon$. We can say that the probability of error is bounded by $\frac{1}{2} - \varepsilon$.

Two main complexity measures of a quantum branching program are its *width* (the dimension of the state space d) and its *length* (the number of instructions l). We can also introduce the complexity measures of the function f being computed by a quantum branching program (QBP):

$$\mathit{Width}(f) = \min \mathit{Width}(Q), \qquad \mathit{Length}(f) = \min \mathit{Length}(Q),$$

where the minimum is taken over all QBPs that compute f.

We can naturally define restricted variants of the QBP model. For instance, we call a quantum branching program a *quantum OBDD* (QOBDD) if each input variable appears in the sequence of instructions at most once. In this case we can define π to be a permutation of $\{1, \ldots, n\}$ that defines the order of testing input variables.

We can also define a quantum k-OBDD (k-QOBDD) to be a quantum branching program with k layers, where each layer is a QOBDD, and each layer has the same order π. We allow measurements for k-QOBDDs during the computation, but after that it should stop and accept an input or continue the computation. A k-*id*-QOBDD is a k-QOBDD with the natural order $id = (1, \ldots, n)$ of input bits.

Let k-$\mathit{QOBDD}_{\mathcal{W}}$ be a set of Boolean functions that can be computed by the bounded-error k-QOBDDs of width w, for $w \in \mathcal{W}$. k-*id*-$\mathit{QOBDD}_{\mathcal{W}}$ is defined in a similar way for bounded-error k-*id*-QOBDDs.

Let BQP_{ε}-kQOBDD be a set of Boolean functions that can be computed by polynomial-sized k-QOBDDs with the probability of error bounded by ε. We can consider similar classes for the deterministic model (P-kOBDD) and bounded-error probabilistic model (BP_{ε}-kOBDD):

- P-kOBDD and BP_{δ}-kOBDD are defined for polynomial-sized k-OBDDs, the first one is for the deterministic case and the second one is for the bounded-error probabilistic k-OBDD with error probability at most δ;
- SUPERPOLY-kOBDD and $\text{BSUPERPOLY}_{\delta}$-$k$OBDD are similar classes for superpolynomial-sized models;
- SUBEXP_{α}-kOBDD and $\text{BSUBEXP}_{\alpha,\delta}$-$k$OBDD are similar classes for a size of at most $2^{O(n^{\alpha})}$, for $0 < \alpha < 1$.

2.2 Hierarchies for Classical Read k-Times Branching Programs (k-OBDDs)

One of the first explicit hard functions for the *syntactic* BPs was introduced by Borodin, Razborov, and Smolensky [20]. For each $k \leq c \log n$ they presented

an explicit function, which needs superpolynomial size syntactic k-BPs for some appropriate constant $c > 0$.

Let P-kBP be the set of Boolean functions that can be computed by syntactic k-BP of polynomial size. The classes NP-kBP and BPP-kBP are nondeterministic and bounded-error probabilistic counterparts of the P-kBP class.

Thathachar [61] presented a family of explicit Boolean functions depending on integer parameter k, which cannot be represented by a kn-length polynomial-sized *syntactic* nondeterministic k-BP. In addition, the technique from [61] allows to prove the following proper inclusion: NP-$(k-1)$BP $\subsetneq$ NP-kBP, for $k = o(\log \log n)$.

Probabilistic k-BPs were investigated by Hromkovič and Sauerhoff in 2003 [35]. They proved a lower bound for the explicit Boolean function *m-Masked-*$PJ_{k,n}$ and showed that bounded-error probabilistic k-BPs should have a size of at least $2^{\Omega(N^\alpha/k^3)}$, for $\alpha = 1/(1+2\log 3)$. Using these results Hromkovič and Sauerhoff obtained a hierarchy for polynomial-sized bounded-error probabilistic k-BPs: BPP-$(k-1)$BP $\subsetneq$ BPP-kBP, for $k \leq \log n/3$.

For the case of k-OBDD models, Bolling, Sauerhoff, Sieling, and Wegener suggested an explicit Boolean function which cannot be represented by non-linear-length $o(n^{3/2}/\log^{3/2} n)$-polynomial-sized k-OBDDs. In addition their technique allows to prove the following proper inclusions: P-$(k-1)$OBDD $\subsetneq$ P-kOBDD, for $k = o(n^{1/2}/\log^{3/2} n)$.

Ablayev and Karpinski [8,12] introduced an explicit Boolean function which is hard for polynomial-sized nondeterministic k-OBDDs, for $k = o(n/\log n)$, but can be computed by bounded-error probabilistic k-OBDDs.

Brosenne, Homeister, and Waack [25] showed that for any constant k it holds that NP-OBDD = NP-kOBDD.

The k-OBDD model of small width is also interesting, because, for example, the class of functions computed by constant-width $poly(n)$-OBDD equals the well-known complexity class NC^1 for logarithmic-depth circuits [17,62].

Khadiev [42] considered polynomial-width deterministic and nondeterministic k-OBDDs, and sublinear-width deterministic, nondeterministic and probabilistic k-OBDDs. For polynomial, superpolynomial and subexponential width (size) he proved a lower bound for an explicit function and obtained the following hierarchies.

Theorem 1 (Khadiev [42]).

– *For* $k = o(n/\log n)$ *and* $k > \log^2 n$*, the following statements are true:*

$$P\text{-}\left(\frac{k}{\log^2 n}\right) OBDD \subsetneq P\text{-}kOBDD,\ \ NP\text{-}\left(\frac{k}{\log^2 n}\right) OBDD \subsetneq NP\text{-}kOBDD.$$

– *For* $k = o(n/\log^\alpha n)$ *and* $\log^{\alpha+1} n = o(k)$*, the following statements are true:*

$$SUPERPOLY\text{-}\left\lfloor k/\log^{\alpha+1} n\right\rfloor NOBDD \subsetneq\ SUPERPOLY\text{-}kNOBDD,$$
$$SUPERPOLY\text{-}\left\lfloor k/(\log^{\alpha+1} n)\right\rfloor OBDD \subsetneq SUPERPOLY\text{-}kOBDD.$$

– *For $k = o(n^{1-\alpha})$ and $n^\alpha \log n = o(k)$, the following statements are true:*

$$SUBEXP_\alpha\text{-}\lfloor k/(n^\alpha \log n)\rfloor\ NOBDD \subsetneq SUBEXP_\alpha\text{-}kNOBDD,$$
$$SUBEXP_\alpha\text{-}\lfloor k/(n^\alpha \log n)\rfloor\ OBDD \subsetneq SUBEXP_\alpha\text{-}kOBDD.$$

Proof (Sketch). The *shuffled equality function* was used, which was defined in [6,7]. The lower and upper bounds for separation of the classes are proven via this function.

The idea of the lower bound is presenting k-NOBDDs as 1-NOBDDs of bigger width using a product technique. Then we can use the NOBDD complexity of the shuffled equality function from [6,7].

Let us define the shuffled equality function. Let d be a multiple of 4 such that $4 \le d \le n/4$. The Boolean function EQS_d depends only on the first d bits. For any given input $\nu \in \{0,1\}^n$, we can define two binary strings $\alpha(\nu)$ and $\beta(\nu)$ in the following way. We call odd bits of the input *marker bits* and even bits *value bits*. For any i satisfying $1 \le i \le d/2$, the value bit ν_{2i} belongs to $\alpha(\nu)$ if the corresponding marker bit $\nu_{2i-1} = 0$ and ν_{2i} belongs to $\beta(\nu)$ otherwise. $EQS_d(\nu) = 1$ if $\alpha(\nu) = \beta(\nu)$, and $EQS_d(\nu) = 0$ otherwise. □

Later the results for polynomial width (size) deterministic and probabilistic k-OBDDs were improved by Khadiev and Khadieva [44]. They developed a reordering method for constructing hard functions that allow to show the following hierarchies.

Theorem 2 (Khadiev and Khadieva [44]).

– *For polynomial size (width) we have:*

$$P\text{-}kOBDD \subsetneq P\text{-}2kOBDD, \text{ for } k = o(n/\log^3 n).$$
$$BP_{1/3}\text{-}kOBDD \subsetneq BP_{1/3}\text{-}2kOBDD, \text{ for } k = o(n^{1/3}/\log n).$$

– *For superpolynomial size (width) we have:*

$$SUPERPOLY\text{-}kOBDD \subsetneq SUPERPOLY\text{-}2kOBDD,$$
$$\text{for } k = o(n^{1-\delta}), \delta > 0.$$
$$BSUPERPOLY_{1/3}\text{-}kOBDD \subsetneq BSUPERPOLY_{1/3}\text{-}2kOBDD,$$
$$\text{for } k = o(n^{1/3-\delta}), \delta > 0.$$

– *For subexponential size (width) we have:*

$$SUBEXP_\alpha - kOBDD \subsetneq SUBEXP_\alpha\text{-}2kOBDD,$$
$$\text{for } k = o(n^{1-\delta}), 1 > \delta > \alpha + \varepsilon, \varepsilon > 0.$$
$$BSUBEXP_{\alpha,1/3} - kOBDD \subsetneq BSUBEXP_{\alpha,1/3} - 2kOBDD,$$
$$\text{for } k = o(n^{1/3-\delta/3}), 1/3 > \delta > \alpha + \varepsilon, \varepsilon > 0.$$

Proof (Sketch). We can use the reordered version of the *pointer jumping function* from [19,57]. We use the lower bound for the pointer jumping function that is based on communication complexity results from [57], and modify it for the reordered version of the function. Additionally, we show the upper bound for the function.

Let us define the function. Let V_A, V_B be two disjoint sets (of vertices) with $|V_A| = |V_B| = m$ and $V = V_A \cup V_B$. Let $F_A = \{f_A : V_A \to V_B\}$, $F_B = \{f_B : V_B \to V_A\}$, and $f = (f_A, f_B) : V \to V$ is defined by $f(v) = f_A(v)$ if $v \in V_A$, and $f = f_B(v)$ if $v \in V_B$. For each $k \geq 0$ we define $f^{(k)}(v)$ as $f^{(0)}(v) = v$, $f^{(k+1)}(v) = f(f^{(k)}(v))$. Let $v_0 \in V_A$. We will be interested in computing $g_{k,m} : F_A \times F_B \to V$ defined by $g_{k,m}(f_A, f_B) = f^{(k)}(v_0)$. The function $PJ_{t,n} : \{0,1\}^n \to \{0,1\}$ is the Boolean version of $g_{k,m}$, where we encode f_A by a binary string using $m \log m$ bits, and do this for f_B as well. The result of the function is the parity of the binary representation of the resulting vertex.

Let us apply the reordering method from [44] to the $PJ_{k,n}$ function. $RPJ_{k,n}$ is the total version of the reordered $PJ_{k,n}$. Formally, the Boolean function $RPJ_{k,n} : \{0,1\}^n \to \{0,1\}$ is defined as follows. Let us separate the whole input $X = (x_1, \ldots, x_n)$ into b blocks, such that $b\lceil \log_2 b + 1 \rceil = n$; therefore $b = O(n/\log n)$. Let $Adr(X,i)$ be the address whose binary representation is given by the first $\lceil \log_2 b \rceil$ bits of the i-th block and let $Val(X,i)$ be the value of the bit at position $\lceil \log_2 b + 1 \rceil$ of block i, for $i \in \{0, \ldots, b-1\}$. Let a be a number such that $b = 2a\lceil \log_2 a \rceil$ and $V_A = \{0, \ldots, a-1\}$, $V_B = \{a, \ldots, 2a-1\}$.

Let the function $BV : \{0,1\}^n \times \{0, \ldots, 2a-1\} \to \{0, \ldots, a-1\}$ be defined as follows:

$$BV(X,v) = \sum_{i:(v-1)\lceil \log_2 b \rceil < Adr(X,i) \leq v\lceil \log_2 b \rceil} 2^{Adr(X,i)-(v-1)\lceil \log_2 b \rceil} \cdot Val(X,i) \pmod a.$$

Then $f_A(v) = BV(X,v) + a$, $f_B(v) = BV(X,v)$.
Let $r = g_{k,a}(f_A, f_B)$; then

$$RPJ_{k,n}(X) = \bigoplus_{i:(r-1)\lceil \log_2 b \rceil < Adr(X,i) \leq r\lceil \log_2 b \rceil} Val(X,i).$$

□

Khadiev [42] also investigated the sublinear-width case. He showed the following results for deterministic, nondeterministic and probabilistic models of constant, polylogarithmic and sublinear width.

Theorem 3 (Khadiev [42]).

– *For $k = o(n/\log n)$, the following statements are true:*

$$\lfloor k/(\log_2 \log_2 n) \rfloor\text{-}OBDD_{\texttt{const}} \subsetneq k\text{-}OBDD_{\texttt{const}}, \text{for } \log n = o(k);$$
$$\lfloor k/(\log_2 \log_2 n) \rfloor\text{-}NOBDD_{\texttt{const}} \subsetneq k\text{-}NOBDD_{\texttt{const}}, \text{for } \log n = o(k);$$
$$\lfloor k/(\log_2 n \cdot \log_2 \log_2 n) \rfloor\text{-}POBDD_{\texttt{const}} \subsetneq k\text{-}POBDD_{\texttt{const}},$$
$$\text{for } \log_2 n \cdot \log_2 \log_2 n = o(k).$$

– *For* $\varepsilon, \varepsilon_1 > 0, k = o(n^{1-\varepsilon}), n^{\varepsilon_1} < k$, *the following statements are true:*

$$\lfloor k/n^{\varepsilon_1} \rfloor\text{-}OBDD_{\mathtt{polylog}} \subsetneq k\text{-}OBDD_{\mathtt{polylog}};$$
$$\lfloor k/n^{\varepsilon_1} \rfloor\text{-}NOBDD_{\mathtt{polylog}} \subsetneq k\text{-}NOBDD_{\mathtt{polylog}};$$
$$\lfloor k/n^{\varepsilon_1} \rfloor\text{-}POBDD_{\mathtt{polylog}} \subsetneq k\text{-}POBDD_{\mathtt{polylog}}.$$

– *the following statements are true:*

$$\lfloor k/(n^{\alpha}(\log_2 n)^2) \rfloor\text{-}OBDD_{\mathtt{sublinear}_\alpha} \subsetneq k\text{-}OBDD_{\mathtt{sublinear}_\alpha},$$
$$\text{for } 0 < \alpha < 0.5 - \varepsilon, \varepsilon > 0, k > n^{\alpha}(\log_2 n)^2, k = o(n^{1-\alpha}/\log_2 n);$$
$$\lfloor k/(n^{\alpha}(\log_2 n)^2) \rfloor\text{-}OBDD_{\mathtt{sublinear}_\alpha} \subsetneq k\text{-}OBDD_{\mathtt{sublinear}_\alpha},$$
$$\text{for } 0 < \alpha < \frac{1}{3} - \varepsilon, \varepsilon > 0, k > n^{2\alpha}(\log_2 n)^2, k = o(n^{1-\alpha}/\log_2 n);$$
$$\lfloor k/(n^{2\alpha}(\log_2 n)^2) \rfloor\text{-}POBDD_{\mathtt{sublinear}_\alpha} \subsetneq k\text{-}POBDD_{\mathtt{sublinear}_\alpha},$$
$$\text{for } 0 < \alpha < 0.25 - \varepsilon, \varepsilon > 0, k > n^{2\alpha}(\log_2 n)^2, k = o(n^{1-2\alpha}/(\log_2 n)^2).$$

Proof (Sketch). The *shuffled address function* from [41] is used. The results also can be shown using the modification of the pointer jumping function from [19,57] that is called *matrix XOR pointer jumping function* [3]. We use lower bounds for memoryless communication protocols from [11,42] and prove upper bound for the shuffled address function.

Let us define the matrix XOR pointer jumping that is easier and also useful for the proof. The matrix XOR pointer jumping function ($MXPJ_{2k,d}$) is a modification of PJ. First, we give the definition of the $MatrixPJ_{2k,d}$ function. Let us consider functions $f_{A,1}, \ldots, f_{A,k} \in F_A$ and $f_{B,1}, \ldots, f_{B,k} \in F_B$. On iteration $j+1$ function $f^{(j+1)}(v) = f_{j+1}(f^{(j)}(v))$, where $f_i(v) = f_{A,\lceil \frac{i}{2} \rceil}(v)$ if i is odd, and $f_i(v) = f_{B,\lceil \frac{i}{2} \rceil}(v)$ otherwise. The value of $MatrixPJ_{2k,d}(f_{A,1}, \ldots, f_{A,k}, f_{B,1}, \ldots, f_{B,k})$ is given by $f^{(k)}(v_0)$. $MXPJ_{2k,d}$ is a modification of $MatrixPJ_{2k,d}$. Here, we take

$$f^{(j+1)}(v) = f_{j+1}(f^{(j)}(v)) \oplus f^{(j-1)}(v), \text{ for } j \geq 0.$$

Finally, we consider a Boolean version of these functions. The Boolean function $PJ_{t,n} \colon \{0,1\}^n \to \{0,1\}$ is $g_{k,d}$, where we encode f_A in a binary string using $d \log d$ bits, and do this for f_B as well. The result of the function is the parity of the bits from the binary representation of the resulting vertex's number. For encoding input functions of $MXPJ_{2k,d}$ we use the following order: $f_{A,1}, \ldots, f_{A,k}, f_{B,1}, \ldots, f_{B,k}$. The function $f_{A,i}$ is encoded by $a_{i,1}, \ldots, a_{i,d}$, and $f_{B,i}$ is encoded by $b_{i,1}, \ldots, b_{i,d}$, for $i \in \{1, \ldots, k\}$. We assume that $v_0 = 0$. □

The results for deterministic k-OBDDs extend the Bolling-Sauerhoff-Sieling-Wegener hierarchy for larger k, but they are not tight. If we compare the results of [44] and [42], then we can see that the latter results are true for larger k. At the same time, the hierarchy of [44] has a smaller "jump" between the classes. Additionally, these two papers show hierarchies for sublinear, superpolynomial and subexponential width. These cases were not considered before.

In the nondeterministic case, the best possible hierarchy for polynomial width is shown in [42]. A smaller jump is not possible, because increasing k by a constant factor does not give more power for the model [25]. So Khadiev's result shows a hierarchy which extends Thathachar's and Okolnishnikova's hierarchies but for the model with a more regular structure (k-NOBDDs). As for the deterministic case, the improvement was shown for sublinear, superpolynomial and subexponential width.

For the probabilistic case, the authors of [44] prove a hierarchy which extends the Hromkovič-Sauerhoff hierarchy but for the model with a more regular structure (probabilistic k-OBDDs). The papers [42,44] show an extension of the Hromkovič-Sauerhoff hierarchy in case of sublinear, superpolynomial and subexponential width.

2.3 Hierarchies for Quantum Read k-Times Branching Programs (k-OBDDs)

In the last decades, a quantum model of OBDDs was considered [4,54,58,59]. Researchers are also interested in the read-k-times quantum model of OBDDs (k-QOBDDs, see, for example, [34]). k-QOBDDs can be explored from an automata point of view. In that situation, we can find good algorithms for two-way quantum classical automata and related models [15,65]. It is known that if we consider unbounded-error k-OBDDs, then we don't have known hierarchies of complexity classes. Let us consider two classes of Boolean functions: functions computed by polynomial-sized unbounded-error k-QOBDDs and 1-QOBDDs. Homeister and Waack [34] have shown equality of these two classes.

Ablayev, Ambainis, Khadiev and Khadieva [3] modified the technique from [44] and proved hierarchies for quantum k-OBDDs with bounded error.

Theorem 4 (Ablayev et al. [3]). *$BQP_{1/8}$-$kOBDD \subsetneq BQP_{1/8}$-$(2k)QOBDD$, for $k > 0, k = const$.*

Proof (Sketch). We use the XOR-reordered version of the pointer jumping function from Theorem 2 and the modified version of the lower bound for the pointer jumping function based on communication complexity results from [39,48]. Additionally, we show an upper bound for the function. □

In [3] we also showed hierarchies for sublinear width and the natural order of the input variables:

Theorem 5 (Ablayev et al. [3]).

- *For $k = o(\sqrt{n})$, $\sqrt{k} > r$, $1 = o(r)$, we have*

$$\lfloor \sqrt{k}/r \rfloor\text{-}id\text{-QOBDD}_{\texttt{const}} \subsetneq k\text{-}id\text{-QOBDD}_{\texttt{const}}.$$

- *For $k = o(n^{0.5-\delta})$, $\sqrt{k} > n^r$, $r > 0, \delta > 0$, we have*

$$\lfloor \sqrt{k}/n^r \rfloor\text{-}id\text{-QOBDD}_{\texttt{polylog}} \subsetneq k\text{-}id\text{-QOBDD}_{\texttt{polylog}}.$$

– *For* $k = o(n^{0.5-\delta})$, $\sqrt{k} > n^r$, $r > 0, \delta > 0$, *we have*

$$\lfloor \sqrt{k}/n^{\alpha+r} \rfloor\text{-}id\text{-QOBDD}_{\mathtt{sublinear}_\alpha} \subsetneq k\text{-}id\text{-QOBDD}_{\mathtt{sublinear}_\alpha}.$$

Proof (Sketch). We use the matrix XOR pointer jumping function and lower bounds for memoryless communication protocols that are modifications of results from [11,42]. Also we present an upper bound for the matrix XOR pointer jumping function. □

The hierarchies in [3] are the first such hierarchies for bounded-error quantum k-OBDDs, and these results show that unbounded-error and bounded-error models have different properties in this point of view.

3 Las-Vegas Models

The Las-Vegas model is an alternative probabilistic computational model. In this case, an algorithm gives the right answer 0 or 1 with probability $1 - \varepsilon$ and with probability ε it fails. The relations between Las-Vegas and deterministic versions of automata and OBDDs were explored by different groups and using different techniques. The first group were Ďuriš, Hromkovič, Rolim, and Schnitger [29,37]; another result was produced by Klauck with the focus on quantum models [49]; and the third research group were Hirvensalo and Seibert [32].

The relations between Las-Vegas and deterministic automata complexities are the following.

Fact 1 (Ďuriš et al. [29,37], Klauck [49], Hirvensalo and Seibert [32]). *For any regular language* L *and error bound* $\varepsilon < 1$, *we have the following lower bounds for probabilistic finite automata and quantum finite automata:*

$$(\mathsf{DFA}(\mathsf{L}))^{1-\varepsilon} \leq \mathsf{LV}_\varepsilon(\mathsf{L}) \text{ and } (\mathsf{DFA}(\mathsf{L}))^{1-\varepsilon} \leq \mathsf{ULV}_\varepsilon(\mathsf{L}).$$

Here $\mathsf{DFA}(\mathsf{L})$ is the size of the best deterministic finite automaton for the language L, $\mathsf{LV}_\varepsilon(\mathsf{L})$ is the size of the best Las-Vegas automaton and $\mathsf{ULV}_\varepsilon(\mathsf{L})$ is the size of the best measure-once quantum automaton. So, if $\varepsilon \leq 1/2$, then the best gap can be quadratic.

Up to a constant, this quadratic gap is achieved [29,37] by using the language $\mathtt{END}_\mathtt{k} = \{u1v \mid u, v \in \{0,1\}^* \text{ and } |v| = k-1\}$:

$$\mathsf{DFA}(\mathtt{END}_\mathtt{k}) = 2^k \text{ and } \mathsf{LV}_{0.5}(\mathtt{END}_\mathtt{k}) \leq 4 \cdot 2^{k/2}.$$

Later Ibrahimov, Khadiev, Prūsis and Yakaryılmaz [38] suggested a modification $\mathtt{MODXOR}_\mathtt{k}$ of the language. The language $\mathtt{MODXOR}_\mathtt{k}$ for $k > 0$ is formed by the strings

$$\{0,1\}^{<2k} x_1 \{0,1\}^{2k-1} x_2 \{0,1\}^{2k-1} \cdots x_m \{0,1\}^{2k-1},$$

where $m > 0$, each $x_i \in \{0,1\}$ for $1 \leq i \leq m$, and $\bigoplus_{i=0}^{m} x_i = 1$, taking $x_0 = 0$.

Using this language, better gaps for the probabilistic and quantum cases were shown.

Theorem 6 (Ibrahimov et al. [38]). *For each $k > 0$, we have:*

$$\mathsf{DFA}(\mathtt{MODXOR_k}) \geq 2^{2k}, \quad \mathsf{LV}_{0.5}(\mathtt{MODXOR_k}) \leq 2 \cdot 2^k, \quad \mathsf{ULV}_{0.5}(\mathtt{MODXOR_k}) \leq 2 \cdot 2^k.$$

The same relation exists for the OBDD model. The relation between the quantum Las-Vegas model and the deterministic one was shown by Sauerhoff and Sieling [60]. For the probabilistic model, the relation was investigated by Ibrahimov, Khadiev, Prūsis and Yakaryılmaz [38].

Theorem 7 (Sauerhoff and Sieling [60], Ibrahimov et al. [38]). *For any Boolean function f over $X = (X_1, \ldots, X_n)$ and error bound $\varepsilon < 1$:*

$$(\mathsf{OBDD}(\mathsf{f}))^{1-\varepsilon} \leq \mathsf{ULV}-\mathsf{OBDD}_\varepsilon(\mathsf{f}) \text{ and } (\mathsf{OBDD}(\mathsf{f}))^{1-\varepsilon} \leq \mathsf{LV}-\mathsf{OBDD}_\varepsilon(\mathsf{f}).$$

For OBDDs with $\varepsilon = \frac{1}{2}$, the lower bound can be at most quadratic. Up to a logarithmic factor, this quadratic gap was achieved by using the $\mathtt{SA_d}$ function in [60] for id-OBDDs. The authors of [38] presented result for OBDDs (for any order) using the $\mathtt{SSA_n}$ function.

Let us define this function. We start with the well-known *storage access* Boolean function $\mathtt{SA_d}(x, y) = x_y$, where the input is split into the *storage* $x = (x_1, \ldots, x_{2^d})$ and the *address* $y = (y_1, \ldots, y_d)$. By using the idea of "shuffling" from [1,6,7,12] we define the *shuffled storage access* Boolean function $\mathtt{SSA_n}\colon \{0,1\}^n \to \{0,1\}$, for even n. Let $x \in \{0,1\}^n$ be an input. We form two disjoint sorted lists of even indexes of bits $I_0 = (2i_1, \ldots, 2i_m)$ and $I_1 = (2j_1, \ldots, 2j_k)$ with the following properties: (i) if $x_{2i-1} = 0$ then $2i \in I_0(x)$, (ii) if $x_{2i-1} = 1$ then $2i \in I_1(x)$, (iii) $i_r < i_{r+1}$, for $r \in \{1, \ldots, m-1\}$ and $j_r < j_{r+1}$, for $r \in \{1, \ldots, k-1\}$.

Let d be a number such that $2^d + d = n/2$. We can construct two binary strings by the following procedure: Initially we set $\alpha(x) := 0^{2^d}$ (the all-zero sequence of length 2^d); then for r from 1 to m we do $\alpha(x) := \mathtt{ShiftX}(\alpha(x), x_{2i_r})$, where $\mathtt{ShiftX}((a_1, \ldots, a_m), b) = (a_2, \ldots, a_m, a_1 \oplus b)$. We initially set $\beta(x) := 0^d$ (the all-zero sequence of length d); then for r from 1 to k: $\beta(x) := \mathtt{ShiftX}(\beta(x), x_{2j_r})$. Then, $\mathtt{SSA_n}(x) = \mathtt{SA_d}(\alpha(x), \beta(x))$.

We can provide a lower bound for OBDDs and upper bounds for Las-Vegas-OBDDs (LV-OBDDs), computing the $\mathtt{SSA_n}$ function.

Theorem 8 (Ibrahimov et al. [38]). *For $2^d + d = n/2$, we have:*

$$\mathsf{OBDD}(\mathtt{SSA_n}) \geq 2^{2^d}, \ \mathsf{LV}-\mathsf{OBDD}_{0.5}(\mathtt{SSA_n}) \leq 2^{2^d/2+d+3},$$
$$\mathsf{ULV}-\mathsf{OBDD}_{0.5}(\mathtt{SSA_n}) \leq 2^{2^d/2+d+3}.$$

4 Quantum Hashing

Hashing has a lot of fruitful applications in cryptography. Note that in cryptography functions satisfying (i) the *one-way property* and (ii) the *collision resistance property* (in different specific meanings) are called hash functions, and we use

this notion when we are considering cryptographic aspects of quantum functions with the above properties. So we suggest to call a quantum function (which maps classical words into quantum states) that satisfies properties (i) and (ii) (in the quantum setting) a *cryptographic quantum hash function* or just quantum hash function. One of the first considerations of a quantum function as a cryptographic primitive, having the one-way property and the collision resistance property, is due to [31], where the quantum fingerprinting function from [26] was used. Another approach towards constructing quantum hash functions from quantum walks was considered in [51,52,66], and it resulted in privacy amplification in quantum key distribution and other useful applications.

In this section, we recall the notion of a quantum (δ, ε)-resistant hash function based on [9].

We let $\mathbb{Z}_q$ be the finite additive group of $\mathbb{Z}/q\mathbb{Z}$, i.e., the integers modulo q. Let Σ^k be the set of words of length k over a finite alphabet Σ. Let $\mathbb{X}$ be a finite set (in [63] we have considered $\mathbb{X}$ as an arbitrary finite Abelian group, and in Subsect. 4.3, we will restrict ourselves to $\mathbb{X} = \mathbb{Z}_q$). For $K = |\mathbb{X}|$ and integer an $s \geq 1$ we define a $(K; s)$ classical-quantum function (or just quantum function) to be the following mapping:

$$\psi \colon \mathbb{X} \to (\mathcal{H}^2)^{\otimes s} \quad \text{or} \quad \psi \colon w \mapsto |\psi(w)\rangle.$$

In order to outline a computational aspect and present a procedure for the quantum function ψ, we define a unitary transformation (determined by an element $w \in \mathbb{X}$) of the initial state $|\psi_0\rangle \in (\mathcal{H}^2)^{\otimes s}$ to a quantum state $|\psi(w)\rangle \in (\mathcal{H}^2)^{\otimes s}$:

$$|\psi(w)\rangle = U(w)|\psi_0\rangle,$$

where $U(w)$ is a unitary matrix.

Extracting information about w from $|\psi(w)\rangle$ is a result of the measurement of the quantum state $|\psi(w)\rangle$. In this paper, we consider quantum transformations and measurements of quantum states with respect to computational basis.

4.1 One-Way δ-Resistance

We present the following definition of a quantum δ-resistant one-way function. Let M be a function M: $(\mathcal{H}^2)^{\otimes s} \to \mathbb{X}$. Informally speaking, an "information extracting" mechanism M makes some measurement of the state $|\psi\rangle \in (\mathcal{H}^2)^{\otimes s}$ and decodes the result to $\mathbb{X}$.

Definition 2 (Ablayev and Ablayev [9]). *Let X be a random variable with $\Pr[X = w]$ for all $w \in \mathbb{X}$. Let $\psi \colon \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a quantum function. Let Y be a random variable over $\mathbb{X}$ obtained by some M making a measurement to the encoding ψ of X and decoding the result to $\mathbb{X}$. Let $\delta > 0$. We call a quantum function ψ a one-way δ-resistant function if*

1. *it is easy to compute, i.e., a quantum state $|\psi(w)\rangle$ for a particular $w \in \mathbb{X}$ can be computed using a polynomial-time algorithm;*

2. *for any mechanism M, the probability* $\Pr[Y = X]$ *that M successfully decodes Y is bounded by δ, i.e.,*

$$\Pr[Y = X] \leq \delta.$$

For cryptographic purposes it is natural to expect (and we do this in the rest of the paper) that the random variable X is uniformly distributed.

A quantum state of $s \geq 1$ qubits can theoretically contain an infinite amount of information. On the other hand, Holevo's Theorem [33] states that $O(s)$ bits of information can be extracted from this state. Here we use the result of [56] motivated by Holevo's Theorem.

Property 1 (Nayak [56]). Let X be a random variable uniformly distributed over $\{0,1\}^k$. Let $\psi\colon \{0,1\}^k \to (\mathcal{H}^2)^{\otimes s}$ be a quantum function. Let Y be a random variable over $\{0,1\}^k$ obtained by some M making a measurement of the encoding ψ of X and decoding the result to $\{0,1\}^k$. Then the probability of a correct decoding is given by

$$\Pr[Y = X] \leq \frac{2^s}{2^k}.$$

So, extracting information about the input σ from the state $|\psi(\sigma)\rangle$ is rather limited. The efficiency of computing $|\psi(\sigma)\rangle$ depends on the construction of the quantum hash function ψ. In Subsect. 4.3, we consider the quantum hash function that is based on small biased sets and prove the efficiency of this construction.

4.2 Collision ε-Resistance

The following definition was presented in [2].

Definition 3. *Let $\varepsilon > 0$. We call a quantum function $\psi\colon \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ a collision ε-resistant function if for any pair w, w' of different inputs,*

$$|\langle\psi(w)|\psi(w')\rangle| \leq \varepsilon.$$

Informally speaking, we need two states $|\psi(w)\rangle$ and $|\psi(w')\rangle$ to be almost orthogonal in order to get a small probability of collision, i.e., if one is testing different states $|\psi(w)\rangle$ and $|\psi(w')\rangle$ for equality, then a testing procedure should give a positive result with a small probability .

The following result [2] proves that a quantum collision ε-resistant function needs at least $\log\log K - c(\varepsilon)$ qubits.

Property 2 (Ablayev and Ablayev [2]). Let $s \geq 1$ and $K = |\mathbb{X}| \geq 4$. Let $\psi\colon \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a collision ε-resistant quantum hash function. Then

$$s \geq \log\log K - \log\log\left(1 + \sqrt{2/(1-\varepsilon)}\right) - 1.$$

Properties 1 and 2 provide a basis for building a "balanced" one-way δ-resistant and collision ε-resistant quantum hash function. Roughly speaking, if we need to hash elements w from the domain $\mathbb{X}$ with $|\mathbb{X}| = K$ and if we can build for an $\varepsilon > 0$ a collision ε-resistant $(K; s)$ hash function ψ with $s \approx \log\log K - c(\varepsilon)$ qubits, then the function f is one-way δ-resistant with $\delta \approx (\log K/K)$. Such a function is balanced with respect to Property 2.

To summarize the above considerations we can state the following. A quantum (δ, ε)-hash function satisfies all of the properties that a "classical" hash function should satisfy. Pre-image resistance follows from Property 1. Second pre-image resistance and collision resistance follow, because all inputs are mapped to states that are nearly orthogonal. Therefore, we see that quantum hash functions can satisfy the properties of classical cryptographic hash functions.

4.3 Constructing Quantum Hash Functions via Small-Biased Sets

This section is based on the paper [63]. We first present a brief background on ε-biased sets. For more information see [27]. Note that ε-biased sets are generally defined for arbitrary finite groups, but here we restrict ourselves to $\mathbb{Z}_q$.

For an $a \in \mathbb{Z}_q$ a character χ_a of $\mathbb{Z}_q$ is a homomorphism $\chi_a \colon \mathbb{Z}_q \to \mu_q$, where μ_q is the (multiplicative) group of complex q-th roots of unity. That is, $\chi_a(x) = \omega^{ax}$, where $\omega = e^{\frac{2\pi i}{q}}$, is a primitive q-th root of unity. The character $\chi_0 \equiv 1$ is called a trivial character.

Definition 4. *A set $S \subseteq \mathbb{Z}_q$ is called ε-biased if for any nontrivial character $\chi \in \{\chi_a \mid a \in \mathbb{Z}_q\}$*

$$\frac{1}{|S|}\left|\sum_{x \in S} \chi(x)\right| \leq \varepsilon.$$

These sets are interesting when $|S| \ll |\mathbb{Z}_q|$ (as $S = \mathbb{Z}_q$ is 0-biased). In their seminal paper [55] Naor and Naor defined these small-biased sets, gave the first explicit constructions of such sets, and demonstrated the power of small-biased sets for several applications.

Remark 1. Note that a set S of $O(\log q/\varepsilon^2)$ elements selected uniformly at random from $\mathbb{Z}_q$ is ε-biased with positive probability [14].

Many other constructions of small-biased sets followed during the last decades.

Vasiliev [63] showed that ε-biased sets generate (δ,ε)-resistant hash functions. We present the result of [63] in the following form.

Theorem 9. *Let $S \subseteq \mathbb{Z}_q$ be an ε-biased set. Let*

$$H_S = \{h_a(x) = ax \pmod q, \quad a \in S, h_a \colon \mathbb{Z}_q \to \mathbb{Z}_q\}$$

be a set of functions determined by S. Then the quantum function $\psi_{H_S}\colon \mathbb{Z}_q \to (\mathcal{H}^2)^{\otimes \log |S|}$

$$|\psi_{H_S}(x)\rangle = \frac{1}{\sqrt{|S|}} \sum_{a \in S} \omega^{h_a(x)} |a\rangle$$

is a (δ, ε)-resistant quantum hash function, where $\delta \leq |S|/q$.

As a corollary from Theorem 9 and the above considerations we can state the following.

Property 3. For a small-size ε-biased set $S = \{a_1, \ldots, a_T\} \subset \mathbb{F}_q$ with $T = O(\log q/\varepsilon^2)$, for $s = \log T$, for $\delta = O(1/(q\varepsilon^2))$, H_S generates the quantum (δ, ε)-hash function

$$\psi_{H_S}\colon \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s} \tag{4}$$

$$|\psi_{H_S}(x)\rangle = \frac{1}{\sqrt{T}} \sum_{j=0}^{T-1} \omega^{a_j x} |j\rangle. \tag{5}$$

5 Computing Quantum Hash Function by QOBDD

The complexity of computing the quantum hash function ψ_{H_S} in the QOBDD model is given by the following theorem.

Theorem 10. *The quantum (δ, ε)-hash function (4)*

$$\psi_{H_S}\colon \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s}$$

can be computed by a quantum OBDD Q composed of $s = O(\log \log q)$ qubits.

Proof. The quantum function ψ_{H_S} (4) maps an input $x \in \mathbb{F}_q$ to a quantum state (5), i.e.,

$$|\psi_{H_S}(x)\rangle = \frac{1}{\sqrt{T}} \sum_{j=0}^{T-1} \omega^{a_j x} |j\rangle,$$

which is the result of a transformation (similar to the well-known *quantum Fourier transformation*, QFT) of the initial state

$$|\psi_0\rangle = \frac{1}{\sqrt{T}} \sum_{j=0}^{T-1} |j\rangle.$$

Such a QFT-like operator is controlled by the input x. We represent an integer $x \in \{0, \ldots, q-1\}$ as the bit string $x = x_0 \ldots x_{\log q - 1}$, i.e., $x = x_0 + 2^1 x_1 + \cdots + 2^{\log q - 1} x_{\log q - 1}$.

For a binary string $x = x_0 \dots x_{\log q-1}$ a quantum OBDD Q over the space $(\mathcal{H}^2)^{\otimes s}$ for computing $\psi_{H_S}(x)$ (composed of $s = \log T$ qubits) is defined as

$$Q = \langle \mathbb{T}, |\psi_0\rangle \rangle,$$

where $|\psi_0\rangle$ is the initial state and $\mathbb{T}$ is a sequence of $\log q$ instructions:

$$\mathbb{T}_j = (x_j, U_j(0), U_j(1))$$

is determined by the variable x_j tested on step j, and $U_j(0)$, $U_j(1)$ are unitary transformations in $(\mathcal{H}^2)^{\otimes s}$. More precisely, $U_j(0)$ is the $T \times T$ identity matrix. $U_j(1)$ is the $T \times T$ diagonal matrix whose diagonal entries are $\omega^{a_0 2^j}, \omega^{a_1 2^j}, \dots, \omega^{a_{T-1} 2^j}$ and the off-diagonal elements are all zero, i.e.,

$$U_j(1) = \begin{bmatrix} \omega^{a_0 2^j} & & & \\ & \omega^{a_1 2^j} & & \\ & & \ddots & \\ & & & \omega^{a_{T-1} 2^j} \end{bmatrix}.$$

Note that here we compute a quantum function instead of a Boolean function, and thus we abandoned the measurement phase of the computation and the corresponding accepting set in the construction of a quantum branching program.

We define a computation of Q on an input $x = x_0 \dots x_{\log q-1} \in \{0,1\}^{\log q}$ as follows:

1. A computation of Q starts from the initial state $|\psi_0\rangle$;
2. The j-th instruction of Q reads the input symbol x_j (the value of x) and applies the transition matrix $U_j(x_j)$ to the current state $|\psi\rangle$ to obtain the state $|\psi'\rangle = U_j(x_j)|\psi\rangle$;
3. The final state is

$$|\psi_{H_S}(x)\rangle = \left(\prod_{j=0}^{\log q-1} U_j(x_j) \right) |\psi_0\rangle.$$

□

Upper bounds. From Theorem 10 we have the following corollary.

Corollary 1. *It holds that*

$$\mathit{Width}(\psi_{H_S}) = O(\log q),$$
$$\mathit{Length}(\psi_{H_S}) = O(\log q).$$

Proof. $O(\log q)$ width corresponds to $O(\log \log q)$ qubits used by the QOBDD, and $O(\log q)$ length is due to the binary encoding of the input $x \in \mathbb{F}_q$.

Lower bounds. Here we show that the quantum OBDD from Theorem 10 is optimal for the function ψ_{H_S}.

Theorem 11. *It holds that*

$$Width(\psi_{H_S}) = \Omega(\log q), \tag{6}$$

$$Length(\psi_{H_S}) = \Omega(\log q). \tag{7}$$

Proof. Let Q be a QOBDD computing the function ψ_{H_S}. Since QOBDDs are defined for binary inputs, we use the following encoding for ψ_{H_S}:

$$\psi_{H_S} \colon \{0,1\}^{\log q} \to (\mathcal{H}^2)^{\otimes s}.$$

The lower bound (6) for $Width(\psi_{H_S})$ follows immediately from the minimal number of qubits required by Property 2.

$$s \geq \log\log q - \log\log(1 + \sqrt{2/(1-\varepsilon)}).$$

The lower bound (7) for $Length(\psi_{H_S})$ follows from the fact that ψ_{H_S} is a collision ε-resistant function. Indeed, the assumption that the QOBDD Q for ψ_{H_S} can test less than $\log q$ (that is, not all $\log q$) variables encoding the input $x \in \mathbb{F}_q$ means the existence of (at least) two different inputs $w, w' \in \mathbb{F}_q$, such that Q produces the same quantum hashes $|\psi(w)\rangle$ and $|\psi(w')\rangle$ for w and w', that is, $|\psi(w)\rangle = |\psi(w')\rangle = |\psi\rangle$. The latter contradicts the fact that the states $|\psi(w)\rangle$ and $|\psi(w')\rangle$ are ε-orthogonal:

$$|\langle\psi(w)|\psi(w')\rangle| \leq \varepsilon.$$

□

6 Online Streaming Algorithms

Online algorithms are well-known as a computational model for solving optimization problems. The defining property of this model is that the algorithm reads an input piece by piece and should return output variables after some of the input variables immediately, even if the answer depends on whole input. An online algorithm should return an output for minimizing an objective function. There are different methods to define the efficiency of online algorithms [22,23,28], but the most standard is the competitive ratio [40].

Typically, online algorithms have unlimited computational power and the main restriction is the lack of knowledge on future input variables. There are many problems that can be formulated in these terms. At the same time it is quite interesting to solve online minimization problems in the case of large input streams. We consider a large stream such that it cannot be stored in memory. In this situation, we can discuss online algorithms with restricted memory. In this paper, we consider streaming algorithms as online algorithms. This classical model was considered in [18,24,30,47]. Automata for online minimization problems were considered in [43]. We are interested in the investigation of a new model of online algorithms, the *quantum online algorithms*, that use the power of quantum computing for solving online minimization problems. This model was introduced in [47]. Here, we focus on quantum online streaming algorithms.

6.1 Definitions

Let us define online optimization problems. We give all following definitions with respect to [42,45,47,67]. An *online minimization problem* consists of a set $\mathcal{I}$ of inputs and a cost function. Every input $I \in \mathcal{I}$ is a sequence of requests $I = (x_1, \ldots, x_n)$. Furthermore, a set of feasible outputs (or solutions) is associated with every I; every output is a sequence of answers $O = (y_1, \ldots, y_n)$. The cost function assigns a positive real value $cost(I, O)$ to every input I and any feasible output O. For every input I, we call any feasible output O for I that has the smallest possible cost (i.e., that minimizes the cost function) an optimal solution for I.

Let us define an online algorithm for a given online problem as an algorithm which gets requests x_i from $I = (x_1, \ldots, x_n)$ one by one and should return answers y_i from $O = (y_1, \ldots, y_n)$ immediately, even if an optimal solution can depend on future requests. A *deterministic online algorithm* A computes the output sequence $A(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $x_1, \ldots, x_i$. This setting can also be regarded as a request-answer game: an adversary generates requests, and an online algorithm has to serve them one at a time [13].

We use the competitive ratio as the main measure of quality for online algorithms. It is the ratio of the cost of the algorithm's solution and the cost of a solution of an optimal offline algorithm in the worst case. We say that some deterministic online algorithm A is *c-competitive* if there exists a non-negative constant α such that, for every n and for any input I, we have: $cost(A(I)) \leq c \cdot cost(Opt(I)) + \alpha$, where Opt is an optimal offline algorithm for the problem, $|I| \leq n$ and $|I|$ is length of I. We also call c the *competitive ratio* of A. If $\alpha = 0$, then A is called strictly c-competitive; A is optimal if it is strictly 1-competitive.

Let us define an online algorithm with advice. We can say that advice is some information about the future input. An *online algorithm A with advice* computes the output sequence $A^{\phi}(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $\phi, x_1, \ldots, x_i$, where ϕ is the message from an *adviser*, who knows the whole input. A is c-competitive with advice complexity $b = b(n)$ if there exists a non-negative constant α such that, for every n and for any input I, there exists some ϕ such that $cost(A^{\phi}(I)) \leq c \cdot cost(Opt(I)) + \alpha$ and $|\phi| \leq b$, $|I| \leq n$.

Next, let us define a randomized online algorithm. A *randomized online algorithm* R computes the output sequence $R^{\psi} := R^{\psi}(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $\psi, x_1, \ldots, x_i$, where ψ is the content of a random tape, i.e., an infinite binary sequence, where every bit is chosen uniformly at random and independently of all the others. By $cost(R^{\psi}(I))$ we denote the random variable expressing the cost of the solution computed by R on I. R is c-competitive in expectation if there exists a non-negative constant α such that, for every I, $\mathbb{E}[cost(R^{\psi}(I))] \leq c \cdot cost(Opt(I)) + \alpha$.

We use streaming algorithms for online minimization problems as online algorithms with restricted memory. You can read more about streaming algorithms in [53]. Shortly, these are algorithms that use a small amount of memory and read input variables one by one. Suppose A is a *deterministic online streaming*

algorithm with $s = s(n)$ bits of memory that processes the input $I = (x_1, \ldots, x_n)$. Then we can describe a state of the memory of A before reading input variable x_{i+1} by a vector $d^i = (d^i_1, \ldots, d^i_s) \in \{0,1\}^s$. The algorithm computes the output $A(I) = (y_1, \ldots, y_n)$ such that y_i depends on d^{i-1} and x_i; d^i depends on d^{i-1} and x_i. *Randomized online streaming algorithms* and *deterministic online streaming algorithms with advice* have similar definitions, but with respect to the definitions of the corresponding models of online algorithms.

Now we are ready to define a *quantum online algorithm*. A *quantum online algorithm* Q computes the output sequence $Q(I) = (y_1, \ldots, y_n)$ such that y_i depends on $x_1, \ldots, x_i$. The algorithm can measure qubits several times during a computation. Note that quantum computation is a probabilistic process. Q is c-competitive in expectation if there exists a non-negative constant α such that, for every I, $\mathbb{E}[cost(Q(I))] \leq c \cdot cost(Opt(I)) + \alpha$.

Let us consider a *quantum online streaming algorithm*. For a given $n > 0$, a quantum online algorithm Q with q qubits is defined on the input $I = (x_1, \ldots, x_n) \in \{0, \ldots, \alpha - 1\}^n$ and outputs $(y_1, \ldots, y_m) \in \{0, \ldots, \beta - 1\}^m$. The algorithm is a triple $Q = (\mathbb{T}, |\psi_0\rangle, Result)$, where $\mathbb{T} = \{\mathbb{T}_j \mid 1 \leq j \leq n \text{ and } \mathbb{T}_j = (U^0_j, \ldots, U^{\alpha-1}_j)\}$ are (left) unitary matrices representing the transitions. Here, $U^{x_j}_j$ is applied on the j-th step. $|\psi_0\rangle$ is an initial vector from the 2^q-dimensional Hilbert space. $Result = \{Result_1, \ldots, Result_n\}$, where $Result_i \colon \{0, \ldots, 2^q - 1\} \to \{0, \ldots, \beta - 1\}$ is a function that converts the result of the measurement to an output variable. For any given input I, the computation of A on I can be traced by a 2^q-dimensional vector from the Hilbert space. The initial one is $|\psi_0\rangle$. In each step $j \in \{1, \ldots, n\}$ the input variable x_j is tested and then the $U^{x_j}_j$ unitary operator is applied: $|\psi_j\rangle = U^{x_j}_j(|\psi_{j-1}\rangle)$, where $|\psi_j\rangle$ represents the state of the system after the j-th step. The algorithm can measure one of the qubits or more on any step after a unitary transformation; or it can skip the measurement. Suppose that Q is in the state $|\psi\rangle = (v_1, \ldots, v_{2^q})^T$ before the measurement, and the i-th qubit is measured. Let the states with numbers $a^0_1, \ldots, a^0_{2^{q-1}}$ correspond to the 0 value of the i-th qubit, and the states with numbers $a^1_1, \ldots, a^1_{2^{q-1}}$ correspond to the 1 value of the qubit. The result of the measurement of the qubit is 1 with probability $pr_1 = \sum_{j=1}^{2^{q-1}} |v_{a^1_j}|^2$ and 0 with probability $pr_0 = 1 - pr_1$. If r qubits were measured on the j-th step, then we get the number $\gamma \in \{0, \ldots, 2^r - 1\}$ as a result and A returns $Result_j(\gamma)$.

The following lemma describes relations between automata, *id*-OBDDs and streaming algorithms that are folklore.

Lemma 1. *If a quantum (probabilistic) id-OBDD P of width 2^w computes a Boolean function f, then there is a quantum (randomized) streaming algorithm computing f that uses $w + \lceil \log_2 n \rceil$ qubits (bits). If a quantum (probabilistic) automaton A of size 2^w recognizes a language L, then there is a quantum (randomized) streaming algorithm recognizing L that uses w qubits (bits). If any deterministic (probabilistic) id-OBDD P computing a Boolean function f has width at least 2^w, then any deterministic (randomized) streaming algorithm computing f uses at least w bits. If any deterministic (probabilistic) automaton A*

recognizing a language L has size at least 2^w, then any deterministic (randomized) streaming algorithm recognizing L uses at least w bits.

Let us describe the "black hats method" from [45] that allows to construct hard online minimization problems. In this paper, we discuss a Boolean function f, but in fact we consider a family of Boolean functions $f = \{f_1, f_2, \dots\}$, where $f_m \colon \{0,1\}^m \to \{0,1\}$. We use the notation $f(X)$ for $f_m(X)$ if the length of X is m and it is clear from context.

Definition 5 (Black Hats Method). *Let f be a Boolean function. Then $BH^t_{k,r,w}(f)$, for positive integers k, r, w, t, where $k \bmod t = 0$, is the following online minimization problem: Suppose we have an input $I = (x_1, \dots, x_n)$ and k positive integers $m_1, \dots, m_k$, where $n = \sum_{i=1}^{k}(m_i + 1)$. Lets assume that $I = 2, X_1, 2, X_2, 2, X_3, 2, \dots, 2, X_k$, where $X_i \in \{0,1\}^{m_i}$, for $i \in \{1, \dots, k\}$. Let O be an output and $O' = (y_1, \dots, y_k)$ be the output bits corresponding to input variables with value 2 (in other words, output variables for guardians). Output y_j corresponds to an input variable x_{i_j}, where $i_j = j + \sum_{r=1}^{j-1} m_r$. Let $g_j(I) = \bigoplus_{i=j}^{k} f_{m_i}(X_i)$. We split all output variables y_i into t blocks each of length $u = k/t$. The cost of the i-th block is c_i, where $c_i = r$ if $y_j = g_j(I)$ for $j \in \{(i-1)u + 1, \dots, i \cdot u\}$, and $c_i = w$ otherwise. The cost of the whole output is $cost^t(I, O) = c_1 + \dots + c_t$.*

6.2 Quantum vs. Classical Online Streaming Algorithms

Khadiev, Ziatdinov, Mannapov, Khadieva and Yamilov [46,47] showed that quantum online streaming algorithms can be better than classical ones in the case of sublogarithmic memory (see the following theorem).

Theorem 12 (Khadiev et al. [46], Khadiev et al. [47]). *There is an online minimization problem BHP with the following properties:*

- *There is a quantum online streaming algorithm Q for BHP such that it uses 1 qubit of memory and has expected competitive ratio c.*
- *Any deterministic online algorithm D with unlimited computational power solving BHP is c'-competitive, for $c' > c$.*
- *Any randomized online streaming algorithm R using $o(\log n)$ bits and solving BHP is c''-competitive in expectation, for $c'' > c$.*

Proof (Sketch). To define the *BHP* problem, we use the Boolean function $PartialMOD^s_n$ from [6,7,16] and the black hats method for constructing hard minimization problems from [45]. We propose a single qubit quantum online streaming algorithm for the problem and show lower bounds for classical models.

Let us present the definition of the Boolean function $PartialMOD^s_n$. The feasible inputs for the problem are $X = (x_1, \dots, x_n) \in \{0,1\}^n$ such that $\#_1(X) = v \cdot 2^s$, where $\#_1(X)$ is the number of 1s in X and $v \geq 2$. $PartialMOD^s_n(X) = v \bmod 2$. □

The case of polylogarithmic memory was considered by Khadiev, Khadieva, Kravchenko, Rivosh, Yamilov and Mannapov [45]. Similar supremacy is shown in the following result.

Theorem 13 (Khadiev et al. [46]). *There is an online minimization problem BHR with the following properties:*

- *There is a quantum online streaming algorithm Q for BHR such that it uses $\log^{O(1)} n$ qubits of memory and has expected competitive ratio c.*
- *Any deterministic online algorithm D with unlimited computational power solving BHR is c'-competitive, for $c' > c$.*
- *Any randomized online streaming algorithm R using $n^{o(1)}$ bits and solving BHR is c''-competitive in expectation, for $c'' > c$.*

Proof (Sketch). For the definition of the *BHR* problem, we use $R_{\nu,l,m,n}$ from [60] and the black hats method. We propose a single qubit quantum online streaming algorithm for the problem and show lower bounds for classical models.

Let us define $R_{\nu,l,m,n}$. Let $|1\rangle, \ldots, |n\rangle$ be the standard computational basis of the n-dimensional Hilbert space. Let V_0 and V_1 denote the subspaces spanned by the first and last $n/2$ of these basis vectors. Let $0 < \nu < 1/\sqrt{2}$. The input for the function $R_{\nu,l,m,n}$ consists of $3l(m+1)$ Boolean variables $a_{i,j}, b_{i,j}, c_{i,j}, 1 \leq i \leq l, 1 \leq j \leq m+1$, which are interpreted as universal (ε, l, m)-codes for three unitary $n \times n$ matrices A, B, C, where $\varepsilon = 1/(3n)$. The function takes the value $z \in \{0,1\}$ if the Euclidean distance between $CBA|1\rangle$ and V_z is at most ν; otherwise the function is undefined. □

6.3 Advice Complexity of Quantum and Classical Online Streaming Algorithms

We are also interested in the *advice complexity* of online algorithms [21,50]. In this model an online algorithm gets some bits of advice about the input. The trusted *adviser* sending these bits knows the whole input and has unlimited computational power. The question is "How many advice bits are sufficient to reduce the competitive ratio or to make the online algorithm as efficient as the offline algorithm?" This question has different interpretations. One of them is "How much information should an algorithm have about the future for solving a problem efficiently?" Another one is "If we have an expensive channel which can be used for pre-processed information about the future, then how many bits should we send via this channel to solve a problem efficiently?" Deterministic and probabilistic or randomized online algorithms with advice were investigated in [36,50]. It is interesting to compare the power of quantum online streaming algorithms and classical ones.

Let us consider the case of sublogarithmic memory. Khadiev, Ziatdinov, Mannapov, Khadieva and Yamilov [46,47] showed that quantum online streaming algorithms can be better than classical ones even if the classical algorithms get advice bits. Moreover, if a quantum online streaming algorithm gets a single advice bit, then it becomes optimal.

Theorem 14 (Khadiev et al. [46], Khadiev et al. [47]). *There is an online minimization problem BHP with the following properties:*

- *There is a quantum online streaming algorithm Q for BHP such that it uses a single qubit of memory and has expected competitive ratio c, for $c \geq 1$.*
- *There is an* optimal *quantum online streaming algorithm Q' for BHP such that it uses a single qubit of memory and a single advice bit.*
- *Any deterministic online streaming algorithm D using $o(\log n)$ bits of memory, $o(\log n)$ advice bits and solving BHP is c'-competitive, for $c' > c$.*
- *Any randomized online streaming algorithm R using $o(\log n)$ bits of memory, $o(\log n)$ advice bits and solving BHP is c''-competitive in expectation, for $c'' > c$.*

Proof (Sketch). We use the same problem *BHP* as in Theorem 12 that is based on the Boolean function $PartialMOD_n^s$ from [6,7,16]. We can construct an optimal quantum algorithm, because we have an exact quantum streaming algorithm for $PartialMOD_n^s$. □

Quantum online streaming algorithms can also be better than deterministic algorithms with unlimited computational power and a constant number of advice bits.

Theorem 15 (Khadiev et al. [46]). *There is an online minimization problem $IBHP_\lambda$ with the following properties:*

- *There is a quantum online streaming algorithm Q for $IBHP_\lambda$ such that it uses constant memory, less than λ advice bits and has expected competitive ratio c, for $c \geq 1$.*
- *Any deterministic online algorithm D with unlimited computational power using less than λ advice bits and solving $IBHP_\lambda$ is c'-competitive, for $c' > c$.*

Proof (Sketch). We use a modification of the *BHP* problem from Theorems 12 and 14. This modification uses λ copies of the *BHP* problem. □

At the same time, randomized online streaming algorithms cannot achieve a similar supremacy for the *BHP* problem as quantum online streaming algorithms. The reason is that there is no randomized streaming algorithm that can compute the Boolean function $PartialMOD_n^s$ using constant memory.

Let us again consider the case of polylogarithmic memory. Khadiev, Ziatdinov, Mannapov, Khadieva and Yamilov [46] and same authors with Kravchenko and Rivosh [45] considered this case. They showed that we have a situation similar to the sublogarithmic case. A quantum online streaming algorithm can be better than classical ones even if classical algorithms get advice.

Theorem 16 (Khadiev et al. [46], Khadiev et al. [45]). *There is an online minimization problem BHR with the following properties:*

- *There is a quantum online streaming algorithm Q for BHR such that it uses $\log^{O(1)} n$ qubits of memory and has expected competitive ratio c, for $c \geq 1$.*

- *Any deterministic online streaming algorithm D using $n^{o(1)}$ bits of memory, $o(\log n)$ advice bits and solving BHR is c'-competitive, for $c' > c$.*
- *Any randomized online streaming algorithm R using $n^{o(1)}$ bits of memory, $o(\log n)$ advice bits and solving BHR is c''-competitive in expectation, for $c'' > c$.*

Proof (Sketch). We use the same problem *BHP* as in Theorem 13 that is based on the Boolean function $R_{\nu,l,m,n}$ from [60]. □

Acknowledgements. Partially supported by ERC Advanced Grant MQC. The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University. Work was in part supported by the Russian Foundation for Basic Research (under the grant 17-07-01606).

References

1. Ablayev, F.: Randomization and nondeterminism are incomparable for ordered read-once branching programs. In: Electronic Colloquium on Computational Complexity (ECCC) (021) (1997)
2. Ablayev, F., Ablayev, M.: Quantum hashing via ε-universal hashing constructions and classical fingerprinting. Lobachevskii J. Math. **36**(2), 89–96 (2015). https://doi.org/10.1134/S199508021502002X
3. Ablayev, F., Ambainis, A., Khadiev, K., Khadieva, A.: Lower bounds and hierarchies for quantum memoryless communication protocols and quantum ordered binary decision diagrams with repeated test. In: Tjoa, A.M., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 197–211. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_14
4. Ablayev, F., Gainutdinova, A., Karpinski, M.: On computational power of quantum branching programs. In: Freivalds, R. (ed.) FCT 2001. LNCS, vol. 2138, pp. 59–70. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44669-9_8
5. Ablayev, F., Gainutdinova, A., Karpinski, M., Moore, C., Pollett, C.: On the computational power of probabilistic and quantum branching program. Inf. Comput. **203**(2), 145–162 (2005)
6. Ablayev, F., Gainutdinova, A., Khadiev, K., Yakaryılmaz, A.: Very narrow quantum OBDDs and width hierarchies for classical obdds. Lobachevskii J. Math. **37**(6), 670–682 (2016). https://doi.org/10.1134/S199508021606007X
7. Ablayev, F., Gainutdinova, A., Khadiev, K., Yakaryılmaz, A.: Very narrow quantum OBDDs and width hierarchies for classical OBDDs. In: Jürgensen, H., Karhumäki, J., Okhotin, A. (eds.) DCFS 2014. LNCS, vol. 8614, pp. 53–64. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09704-6_6
8. Ablayev, F., Karpinski, M.: On the power of randomized ordered branching programs. In: Electronic Colloquium on Computational Complexity (ECCC) (004) (1998)
9. Ablayev, F., Ablayev, M.: On the concept of cryptographic quantum hashing. Laser Phys. Lett. **12**(12), 125204 (2015). http://stacks.iop.org/1612-202X/12/i=12/a=125204
10. Ablayev, F., Gainutdinova, A.: Complexity of quantum uniform and nonuniform automata. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 78–87. Springer, Heidelberg (2005). https://doi.org/10.1007/11505877_7

11. Ablayev, F., Khadiev, K.: Extension of the hierarchy for k-OBDDs of small width. Russ. Math. **53**(3), 46–50 (2013)
12. Ablayev, F., Karpinski, M.: On the power of randomized branching programs. In: Meyer, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 348–356. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61440-0_141
13. Albers, S.: BRICS, Mini-Course on Competitive Online Algorithms. Aarhus University (1996)
14. Alon, N., Roichman, Y.: Random Cayley graphs and expanders. Random Struct. Algorithms **5**(2), 271–284 (1994). https://doi.org/10.1002/rsa.3240050203
15. Ambainis, A., Watrous, J.: Two-way finite automata with quantum and classical states. Theor. Comput. Sci. **287**(1), 299–311 (2002)
16. Ambainis, A., Yakaryılmaz, A.: Superiority of exact quantum automata for promise problems. Inf. Process. Lett. **112**(7), 289–291 (2012)
17. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. J. Comput. Syst. Sci. **38**(1), 150–164 (1989)
18. Becchetti, L., Koutsoupias, E.: Competitive analysis of aggregate max in windowed streaming. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 156–170. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_15
19. Bollig, B., Sauerhoff, M., Sieling, D., Wegener, I.: Hierarchy theorems for kOBDDs and kIBDDs. Theor. Comput. Sci. **205**(1), 45–60 (1998)
20. Borodin, A., Razborov, A., Smolensky, R.: On lower bounds for read-k-times branching programs. Comput. Complex. **3**(1), 1–18 (1993)
21. Boyar, J., Favrholdt, L., Kudahl, C., Larsen, K., Mikkelsen, J.: Online algorithms with advice: a survey. ACM Comput. Surv. (CSUR) **50**(2), 19 (2017)
22. Boyar, J., Irani, S., Larsen, K.S.: A comparison of performance measures for online algorithms. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 119–130. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03367-4_11
23. Boyar, J., Irani, S., Larsen, K.S.: A comparison of performance measures for online algorithms. Algorithmica **72**(4), 969–994 (2015)
24. Boyar, J., Larsen, K.S., Maiti, A.: The frequent items problem in online streaming under various performance measures. Int. J. Found. Comput. Sci. **26**(4), 413–439 (2015)
25. Brosenne, H., Homeister, M., Waack, S.: Nondeterministic ordered binary decision diagrams with repeated tests and various modes of acceptance. Inf. Process. Lett. **98**(1), 6–10 (2006)
26. Buhrman, H., Cleve, R., Watrous, J., de Wolf, R.: Quantum fingerprinting. Phys. Rev. Lett. **87**(16), 167902 (2001). https://doi.org/10.1103/PhysRevLett.87.167902. http://www.arXiv.org/quant-ph/0102001v1
27. Chen, S., Moore, C., Russell, A.: Small-bias sets for nonabelian groups. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM -2013. LNCS, vol. 8096, pp. 436–451. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40328-6_31
28. Dorrigiv, R., López-Ortiz, A.: A survey of performance measures for on-line algorithms. SIGACT News **36**(3), 67–81 (2005)
29. Duriš, P., Hromkovič, J., Rolim, J.D.P., Schnitger, G.: Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 117–128. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0023453

30. Giannakopoulos, Y., Koutsoupias, E.: Competitive analysis of maintaining frequent items of a stream. Theor. Comput. Sci. **562**, 23–32 (2015)
31. Gottesman, D., Chuang, I.: Quantum digital signatures. Technical report, Cornell University Library, November 2001. http://arxiv.org/abs/quant-ph/0105032
32. Hirvensalo, M., Seibert, S.: Lower bounds for Las Vegas automata by information theory. RAIRO-Theor. Inform. Appl. **37**(1), 39–49 (2003)
33. Holevo, A.S.: Some estimates of the information transmitted by quantum communication channel (russian). Probl. Pered. Inform. [Probl. Inf. Transm.] 9(3), 3–11 (1973)
34. Homeister, M., Waack, S.: Quantum ordered binary decision diagrams with repeated tests. arXiv preprint quant-ph/0507258 (2005)
35. Hromkovič, J., Sauerhoff, M.: The power of nondeterminism and randomness for oblivious branching programs. Theory Comput. Syst. **36**(2), 159–182 (2003)
36. Hromkovic, J.: Zámecniková. design and analysis of randomized algorithms: introduction to design paradigms (2005)
37. Hromkovič, J., Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. Inf. Comput. **169**(2), 284–296 (2001)
38. Ibrahimov, R., Khadiev, K., Prusis, K., Yakaryılmaz, A.: Zero-error affine, unitary, and probabilistic obdds. Technical report. arXiv 1703.07184 (2017)
39. Klauck, H., Nayak, A., Ta-Shma, A., Zuckerman, D.: Interaction in quantum communication and the complexity of set disjointness. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 124–133. ACM (2001)
40. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. In: 1986 27th Annual Symposium on Foundations of Computer Science, pp. 244–254. IEEE (1986)
41. Khadiev, K.: Width hierarchy for k-obdd of small width. Lobachevskii J. Math. **36**(2), 178–183 (2015)
42. Khadiev, K.: On the hierarchies for deterministic, nondeterministic and probabilistic ordered read-k-times branching programs. Lobachevskii J. Math. **37**(6), 682–703 (2016)
43. Khadiev, K., Khadieva, A.: Quantum automata for online minimization problems. In: Ninth Workshop on NCMA 2017 Short Papers, pp. 25–33. Institute fur Computersprachen TU Wien (2017)
44. Khadiev, K., Khadieva, A.: Reordering method and hierarchies for quantum and classical ordered binary decision diagrams. In: Weil, P. (ed.) CSR 2017. LNCS, vol. 10304, pp. 162–175. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58747-9_16
45. Khadiev, K., Khadieva, A., Kravchenko, D., Rivosh, A., Yamilov, R., Mannapov, I.: Quantum versus classical online algorithms with advice and logarithmic space. arXiv:1710.09595 (2017)
46. Khadiev, K., Ziatdinov, M., Mannapov, I., Khadieva, A., Yamilov, R.: Quantum online streaming algorithms with constant number of advice bits. arXiv:1802.05134 (2018)
47. Khadiev, K., Khadieva, A., Mannapov, I.: Quantum online algorithms with respect to space complexity. arXiv:1709.08409 (2017)
48. Klauck, H., Nayak, A., Ta-Shma, A., Zuckerman, D.: Interaction in quantum communication. IEEE Trans. Inf. Theory **53**(6), 1970–1982 (2007)

49. Klauck, H.: On quantum and probabilistic communication: Las vegas and one-way protocols. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, STOC 2000, pp. 644–651 (2000)
50. Komm, D.: An Introduction to Online Computation: Determinism, Randomization, Advice. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-42749-2
51. Li, D., Zhang, J., Guo, F.Z., Huang, W., Wen, Q.Y., Chen, H.: Discrete-time interacting quantum walks and quantum hash schemes. Quantum Inf. Process. **12**(3), 1501–1513 (2013). https://doi.org/10.1007/s11128-012-0421-8
52. Li, D., Zhang, J., Ma, X.W., Zhang, W.W., Wen, Q.Y.: Analysis of the two-particle controlled interacting quantum walks. Quantum Inf. Process. **12**(6), 2167–2176 (2013). https://doi.org/10.1007/s11128-012-0516-2
53. Muthukrishnan, S., et al.: Data streams: algorithms and applications. Found. Trends® Theor. Comput. Sci. **1**(2), 117–236 (2005)
54. Nakanishi, M., Hamaguchi, K., Kashiwabara, T.: Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. In: Du, D.-Z.-Z., Eades, P., Estivill-Castro, V., Lin, X., Sharma, A. (eds.) COCOON 2000. LNCS, vol. 1858, pp. 467–476. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44968-X_46
55. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. In: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing, STOC 1990, pp. 213–223. ACM, New York (1990). https://doi.org/10.1145/100216.100244
56. Nayak, A.: Optimal lower bounds for quantum automata and random access codes. In: 1999 40th Annual Symposium on Foundations of Computer Science, pp. 369–376 (1999). https://doi.org/10.1109/SFFCS.1999.814608
57. Nisan, N., Widgerson, A.: Rounds in communication complexity revisited. In: Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, pp. 419–429. ACM (1991)
58. Sauerhoff, M.: Quantum vs. classical read-once branching programs. In: Krause, M., Pudlák, P., Reischuk, R., van Melkebeek, D. (eds.) Complexity of Boolean Functions. No. 06111 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2006). http://drops.dagstuhl.de/opus/volltexte/2006/616
59. Sauerhoff, M., Sieling, D.: Quantum branching programs and space-bounded nonuniform quantum complexity. Theor. Comput. Sci. **334**(1–3), 177–225 (2005)
60. Sauerhoff, M., Sieling, D.: Quantum branching programs and space-bounded nonuniform quantum complexity. Theor. Comput. Sci. **334**(1), 177–225 (2005)
61. Thathachar, J.S.: On separating the read-k-times branching program hierarchy. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 653–662. ACM (1998)
62. Vasiliev, A.V.: Functions computable by Boolean circuits of logarithmic depth and branching programs of a special type. J. Appl. Ind. Math. **2**(4), 585–590 (2008). https://doi.org/10.1134/S1990478908040145
63. Vasiliev, A.: Quantum hashing for finite abelian groups. Lobachevskii J. Math. **37**(6), 751–754 (2016). http://arxiv.org/abs/1603.02209
64. Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. SIAM, Philadelphia (2000)
65. Yakaryılmaz, A., Say, A.C.C.: Succinctness of two-way probabilistic and quantum finite automata. Discrete Math. Theor. Comput. Sci. **12**(2), 19–40 (2010)

66. Yang, Y.G., Xu, P., Yang, R., Zhou, Y.H., Shi, W.M.: Quantum hash function and its application to privacy amplification in quantum key distribution, pseudorandom number generation and image encryption. Sci. Rep. **6**, 19788 (2016). https://doi.org/10.1038/srep19788
67. Yuan, Q.: Quantum Online Algorithms. University of California, Santa Barbara (2009)