Quantum Online Algorithms with Respect to Space Complexity

Kamil Khadiev\textsuperscript{1,2}, Aliya Khadieva\textsuperscript{2}, and Ilnaz Mannapov\textsuperscript{2}

\textsuperscript{1} University of Latvia, Riga, Latvia
\textsuperscript{2} Kazan Federal University, Kazan, Russia
kamilhadi@gmail.com, aliya.khadi@gmail.com, ilnaztatar5@gmail.com

Abstract. Online algorithm is a well-known computational model. We introduce quantum online algorithms and investigate them with respect to a competitive ratio in two points of view: space complexity and advice complexity. We start with exploring a model with restricted memory and show that quantum online algorithms can be better than classical ones (deterministic or randomized) for sublogarithmic space (memory), and they can be better than deterministic online algorithms without restriction for memory. Additionally, we consider polylogarithmic space case and show that in this case, quantum online algorithms can be better than deterministic ones as well.

Another point of view to the online algorithms model is advice complexity. So, we introduce quantum online algorithms with a quantum channel with an adviser. Firstly, we show that quantum algorithms have at least the same computational power as classical ones have. And we give some examples of quantum online algorithms with advice. Secondly, we show that if we allow to use shared entangled qubits (EPR-pairs), then quantum online algorithm can use two times less advise qubits comparing to a classical one. We apply this approach to the well-known Paging Problem.

Keywords: quantum computing, online algorithms, advice complexity, quantum vs classical, quantum models, computational complexity

1 Introduction

Online algorithms are a well-known computational model for solving optimization problems. The peculiarity is that algorithm reads an input piece by piece and should return an answer piece by piece immediately, even if an answer can depend on feature pieces of the input. An online algorithm should return an output for minimizing an objective function. There are different methods to define the effectiveness of the algorithms [BIL09, BIL15, DLO05]. But a most standard is the competitive ratio [KMRS86, ST85]. It is a ratio between output’s price for an online algorithm and optimal offline algorithms.

We suggest a new model for online algorithms, quantum online algorithms that use a power of quantum computing for solving online minimization problem. In this model, an algorithm can have quantum and classical part. Additionally, we consider restricted version of the algorithms, pure quantum online algorithms.
These algorithms have only quantum part. We focus on two measures of complexity: space complexity and advice complexity.

When we consider space, we discuss the online algorithms with restricted memory. For this kind of algorithms, we allow to use only \( s \) bits of memory, for given integer \( s \). Another point of view to the same model is streaming algorithms for an online minimization problem. And such classical models were considered in [BK09], [GK15], [BLM15]. In this case, we show that quantum online algorithms with single qubit can be better than any classical (deterministic or randomized) online algorithms with sublogarithmic memory. And this quantum algorithm still can be better than any deterministic online algorithms without a restriction for memory. It is also interesting to investigate the model with poly-logarithm memory or logarithmic memory (LOGSPACE). Here the algorithm can use \( (\log n)^O(1) \) bits of memory. We show that for \((n, k, r, w)\)-Parity for Number of Equality Hats problem the quantum algorithm can be better than any deterministic one. Additionally, we show that quantum online algorithm can simulate a classical (deterministic or randomized) one with almost same memory. It means that quantum model has at least the same computational power.

Note that for other models, quantum and classical cases can be incomparable, for example, communication complexity model [Kla00] or Ordered Binary Decision Diagrams [SS05]. Online algorithms with restricted memory are similar to streaming algorithms [LG06], [GKK+07]. Branching programs [Weg00], Automata [AY12], [AY15]. Researchers also compare classical and quantum cases for these models [AGK+05], [AGKY14], [AGKY16], [Gai15], [SS05], [KK17], [AY12], [AY15], [LG06], [AAKK17], [IKP+17], [GKK+07].

Another interesting complexity measure for the model with respect to the competitive ratio is advice complexity [Kom16], [BFK+16], [BFK+17]. In this model online algorithm gets some bits of advice about an input. Trusted Adviser sending these bits knows the whole input and has an unlimited computational power. The question is “how many advice bits are enough to reduce competitive ratio or to make the online algorithm as the same effective as the offline algorithm in the worst case?” This question has different interpretations. One of them is “how many information an algorithm should know about a future for solving a problem effectively?”. Another one is “If we have an expensive channel which can be used for pre-processed information about the future, then how many bits we should send by this channel to solve a problem effectively?”. Researchers pay attention to deterministic and probabilistic or randomized online algorithms with advice [Hro05], [KT06], [MR10], [Kom16].

We suggest quantum online algorithms with advice. In this model, an algorithm has a quantum channel with Adviser. We consider two kinds of the model. The first one is an algorithm with private qubits. In this case, an algorithm and an Adviser have not shared qubits. Advice bits are only communication between these two “players”. For this model, we show that we also can simulate a classical (deterministic or randomized) online algorithm with the same number of advice bits. And we demonstrate examples of quantum online algorithms with advice for special \((n, k, r, w)\)-Parity for Number of Hats problem. The second kind of
algorithms is a model with shared entangled qubits or EPR-pairs. Before the computational process, we prepare pairs of entangled qubits (EPR-pairs), and for each pair, we give one qubit to the algorithm and another to the Adviser. Then the Adviser sends twice less advice qubits using a trick from [BW92]. We apply this idea to the well-known Paging problem [Hro05,BEY05].

The paper is organized in the following way. We present definitions in Section 2. In Section 3 we explore online algorithms with restricted memory. The quantum online algorithms with advice are explored in Section 4. We focus on a model with private quantum bits in Section 4.1 and with shared entangled qubits in Section 4.2.

2 Preliminaries

Firstly, let us define an online optimization problem. All following definitions we give with respect to [Kom16].

**Definition 1 (Online Minimization Problem).** An online minimization problem consists of a set \( I \) of inputs and a cost function. Every input \( I \in I \) 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 \( \text{cost}(I, O) \) 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 this problem as an algorithm which gets requests \( I \) one by one and should return answers \( O \) immediately, even if an optimal solution can depend on future requests.

**Definition 2 (Deterministic online algorithm).** Consider an input \( I \) of an online minimization problem. An online algorithm \( A \) computes the output sequence \( A(I) = (y_1, \ldots, y_n) \) such that \( y_i \) is computed from \( x_1, \ldots, x_i, y_1, \ldots, y_{i-1} \). We denote the cost of the computed output by \( \text{cost}(A(I)) = \text{cost}(I, A(I)) \).

This setting can also be regarded as a request-answer game: an adversary generates requests, and an algorithm has to serve them one at a time [Alb96].

As the main measure of quality of an online algorithm, we use a competitive ratio. It is the ratio of two costs: cost for an online algorithm’s solution; and cost for an optimal offline algorithm solution. We consider the worst case.

**Definition 3 (Competitive Ratio).** An online algorithm \( A \) is \( c \)-competitive if there exists a non-negative constant \( \alpha \) such that, for every input \( I \), we have: \( \text{cost}(A(I)) \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha \), where Opt is an optimal offline algorithm for the problem. 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.
Definition 4 (Online Algorithm with Advice). Consider an input $I$ of an online minimization problem. 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 $\phi$ is the message from Adviser, who knows the whole input. $A$ is $c$-competitive with advice complexity $b = b(n)$ if there exists a non-negative constant $\alpha$ such that, for every $n$ and for any input $I$ of length at most $n$, there exists some $\phi$ such that $\text{cost}(A^\phi(I)) \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha$ and length of $\phi$ is at most $b$ bits.

Next, let us define a randomized online algorithm.

Definition 5 (Randomized Online Algorithm). Consider an input $I$ of an online minimization problem. 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 $\psi$ 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 $\text{cost}(R^\psi(I))$ 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 $\alpha$ such that, for every $I$, $\mathbb{E}[\text{cost}(R^\psi(I))] \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha$, where $\text{Opt}$ is an optimal offline algorithm for the problem.

We consider online algorithms that base their computation on both advice bits and randomness.

Definition 6 (Randomized Online Algorithm with Advice). Consider an input $I$ of an online minimization problem. A randomized online algorithm $R$ computes the output sequence $R^{\psi, \phi} := R^{\psi, \phi}(I) = (y_1, \ldots, y_n)$ such that $y_i$ is computed from $\psi, \phi, x_1, \ldots, x_i$, where $\psi$ 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, and $\phi$ is the message from Adviser, who knows whole input. By $\text{cost}(R^{\psi, \phi}(I))$ 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 $\alpha$ such that, for every $I$ of length at most $n$, there is $\phi$ such that, $\mathbb{E}[\text{cost}(R^{\psi, \phi}(I))] \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha$, where $\text{Opt}$ is an optimal offline algorithm for the problem and length of $\phi$ is at most $b$ qubits.

A randomized online algorithm with advice is allowed to make random choices (i.e., “toss coins”) to determine its actions and the advice scheme. Formally, then, a randomized online algorithm with advice is a probability distribution over deterministic online algorithms with advice.

Now we are ready to define a quantum online algorithm. You can read more about quantum computation in [AY15].

Definition 7 (Quantum Online Algorithm). Consider an input $I$ of an online minimization problem. A quantum online algorithm $Q$ computes the output sequence $Q(I) = (y_1, \ldots, y_n)$ such that $y_i$ is computed from $x_1, \ldots, x_i$. $Q$ can have classical and quantum part. The algorithm can measure qubits several times during computation. If the algorithm has not classical part, then we call it “pure quantum online algorithm”. By $\text{cost}(Q(I))$ we denote the cost of the solution
computed by $Q$ on $I$. Note that quantum computation is probabilistic process. $Q$ is $c$-competitive in expectation if there exists a non-negative constant $\alpha$ such that, for every $I$, $E[\text{cost}(Q(I))] \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha$, where $\text{Opt}$ is an optimal offline algorithm for the problem.

Let us define a quantum online algorithm with advice. We define two kinds of the model: a model with shared and a model with private qubits. (We allow or do not allow to share qubits between Adviser and Algorithm before the start of computation.)

Definition 8 (Quantum Online Algorithm with Advice and Private qubits). Consider an input $I$ of an online minimization problem. A quantum online algorithm $A$ with advice and private qubits (or quantum online algorithm with advice) computes the output sequence $A^{|\phi\rangle}(I) = (y_1, \ldots, y_n)$ such that $y_i$ is computed from $|\phi\rangle, x_1, \ldots, x_i$, where $|\phi\rangle$ is the quantum message from Adviser, who knows the whole input. $A$ can have classical and quantum part and do not have any restrictions on measurement. $A$ is $c$-competitive with advice complexity $b = b(n)$ if there exists a non-negative constant $\alpha$ such that, for every $n$ and for any input $I$ of length at most $n$, there exists some $|\phi\rangle$ such that $E[\text{cost}(A^{|\phi\rangle}(I))] \leq c \cdot \text{cost}(\text{Opt}(I)) + \alpha$ where, as above, $\text{Opt}$ is an optimal offline algorithm for the problem and length of $|\phi\rangle$ is at most $b$ qubits.

Definition 9 (Quantum Online Algorithm with Advice and Shared qubits). Consider an input $I$ of an online minimization problem. A quantum online algorithm $A$ with advice, and shared qubits is a quantum online algorithm with advice, but before the process of computation, the Algorithm prepares shared qubits (may be entangled) and sends part of them to the Adviser.

Let us define online algorithms with restricted memory. Let deterministic online algorithm $A_s$ be an algorithm which uses at most $s$ bits of memory on processing any input $I$. We can define similar restrictions for randomized algorithms and algorithms with advice. Let quantum online algorithm $Q_{s,t}$ be an algorithm which uses at most $s$ classical bits of memory and $t$ quantum bits of memory on processing any input $I$. Let pure quantum online algorithm $Q_t$ be an algorithm using at most $t$ quantum bits of memory on processing any input $I$.

3 Space Complexity for Quantum Online Algorithms

Let us focus on space complexity of online algorithms. It is interesting to analyze the size of memory that is required by the algorithm. In a case of the restricted memory, a quantum algorithm can be better than classical ones (deterministic or probabilistic). We present this result in Theorems 12 and 3. And the quantum algorithm can be still better than any deterministic online algorithm without a restriction to a size of memory (Theorem 4).

Let us consider the special problem which allows us to show the separation: $(n, k, r, w)$-Parity for Number of Hats ($(n, k, r, w)$-PNH).
Definition of \((n, k, r, w)\)-PNH problem is based on definition of \(\text{Partial}\text{MOD}_n^k\) function from \cite{AY12,AGKY14,AGKY16}. Feasible inputs for the problem are \(X = (x_1, \ldots, x_n)\), for \(x_1, \ldots, x_n \in \{0, 1\}\) such that \(#_1(X) = v \cdot 2^k\), where \(#_1(X)\) is the number of 1s and \(v \geq 2\) is a positive integer. \(\text{Partial}\text{MOD}_n^k(X) = v \mod 2\).

Firstly, let us describe \((n, k, r, w)\)-PNH problem informally. There are 3 guardians and 3 prisoners. They stay one by one in a line \(\ldots P_1P_2G_1P_3G_2P_4G_3\ldots\), where \(P_i\) is guardian and \(P_i\) is prisoner. Prisoner \(P_i\) has an input \(X_i\) of length \(m_i\) and computes function \(\text{Partial}\text{MOD}_n^k(X_i)\) for \(i \in \{1, 2, 3\}\). If the result is 1 then he paints his hat in black. Otherwise, he paints it in white. Each guardian wants to know if the number of following black hats is odd or even. The cost of right guardian’s answer is \(r\), and the cost of the wrong answer is \(w\). We want to minimize the cost of output, and assume that \(r < w\).

Formal definition of \((n, k, r, w)\)-PNH is following: Feasible inputs for the problem are \(I = (x_1, \ldots, x_n)\) of length \(n\) such that \(n = m_1 + m_2 + m_3 + 3\), for some integer \(m_1, m_2, m_3 \geq 2^{k+1}\). It is guaranteed that \(I\) is always such that \(I = 2, X_1, 2, X_2, 2, X_3\), where \(X_i \in \{0, 1\}^{m_i}\), for \(i \in \{1, 2, 3\}\). Additionally, \(#_1(X_i) = v_i \cdot 2^k\), where \(v_i\) is some integer, \(i \in \{1, 2, 3\}\). Let \(O\) be output of \((n, k, r, w)\)-PNH and \(O' = y_1, y_3, y_3\) be output bits corresponding to input variables with value 2 (in other words, variables of guardians). Output \(y_1\) corresponds to \(x_1\), \(y_2\) corresponds to \(x_2+m_1\), and \(y_3\) corresponds to \(x_3+m_1+m_2\). Let \(z_j(I) = \bigoplus_{i=1}^{3} \text{Partial}\text{MOD}_{m_i}^k(X_i)\). The cost \(\text{cost}(I, O) = r\), if \(y_j = z_j\) for all \(j \in \{1, 2, 3\}\), and \(\text{cost}(I, O) = w\) otherwise. We consider numbers \(r\) and \(w\) such that \(r < w\).

Let us present a pure quantum online algorithm which uses single qubit of memory for this problem. It uses ideas from quantum automata \cite{AY12} and branching programs theories \cite{AGKY14,AGKY16}.

**Algorithm 1. (Quantum Online Algorithm for \((n, k, r, w)\)-PNH)** The pure quantum algorithm \(Q_1\) uses single qubit.

- **Step 1.** The algorithm emulates guessing for \(z_1(I)\). \(Q_1\) starts on a state \(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\). And it measures the qubit before reading any input variables. It gets \(|0\rangle\) or \(|1\rangle\) with equal probability. The result of measurement is \(y_1\).

- **Step 2.** The algorithm reads \(X_1\). Let angle \(\alpha = \pi/2^{k+1}\). Then \(Q_1\) rotates the qubit by an angle \(\alpha\), if the algorithm meets 1. And it does not do anything otherwise.

- **Step 3.** If \(Q_1\) meets 2 then it measures the qubit \(|\psi\rangle = a|0\rangle + b|1\rangle\). If \(\text{Partial}\text{MOD}_{m_1}^k(X_1) = 1\) then the qubit is rotated by an angle \(\pi/2 + v \cdot \pi\), for some integer \(v\), else the qubit is rotated by an angle \(w \cdot \pi\), for some integer \(w\). If \(y_1 = 1\), then \(a \in \{1, -1\}\) and \(b = 0\). And if \(y_1 = 0\), then \(a = 0\) and \(b \in \{1, -1\}\). The result of measurement is \(y_2\).

- **Step 4.** The step is similar to Step 2, but algorithm reads \(X_2\).

- **Step 5.** The step is similar to Step 3, but algorithm outputs \(y_3\).

- **Step 6.** The algorithm reads and skips the last part of the input. \(Q_1\) does not need these variables, because it guesses \(y_1\) and using this value we already can obtain \(y_2\) and \(y_3\) without \(X_3\).
Assume, that algorithm did right guess that \( z_1 = y_1 \). So, if the parity of the passing part is the same as the parity of the future part of the input, then the algorithm returns the right answer with probability 1. And if the guess is not correct and \( z_1 \neq y_1 \), then the algorithm returns a wrong answer with probability 1.

With equal probabilities 0.5 we have \( z_1 = y_1 \) or \( z_1 \neq y_1 \). Thus competitive ratio is \((0.5 \cdot r + 0.5 \cdot w)/r = (r + w)/(2r)\).

As a result we have the following theorem:

**Theorem 1.** There is \((r + w)/(2r)\)-competitive in expectation pure quantum algorithms for \((n, k, r, w)\)-PNH Problem \(Q_1\) with a single qubit of memory.

At the same time, if a deterministic online algorithm for \((n, k, r, w)\)-PNH Problem uses less than \( k \) bits; then it is \((w/r)\)-competitive. To show this claim, let us discuss properties of \(\text{PartialMOD}_k^n\) function.

**Lemma 1.** Let integer \( s, k \) be such that \( s < k = o(\log(n)) \), where \( n \) is the length of input. Then there is no deterministic algorithm that reads an input variable by variable, uses \( s \) bits of memory and computes \(\text{PartialMOD}_k^n(X)\).

**Proof.** Let us assume that we can construct such a deterministic online algorithm \(A_s\). Therefore we can construct an automaton \(T\), with \(2^s\) states that emulates the same algorithm and computes \(\text{PartialMOD}_k^n(X)\). But it is impossible, due to [AY12,AGKY14,AGKY16]. \(\square\)

**Lemma 2.** Let integer \( s, k \) be such that \( s < k = o(\log(n)) \), where \( n \) is the length of input. Then there is no randomized algorithm that reads an input variable by variable, uses \( s \) bits of memory and computes \(\text{PartialMOD}_k^n(X)\) with probability of error less than 0.5.

**Proof.** We can prove the claim using the same technique as in Lemma 1 because of the same lower bound for probabilistic automata [AY12,AGKY14,AGKY16]. \(\square\)

Now we can discuss deterministic online algorithms for \((n, k, r, w)\)-PNH Problem.

**Theorem 2.** Let integer \( s, k \) be such that \( s < k = o(\log(n)) \), where \( n \) is the length of input. Any deterministic online algorithm \(A_s\) computing \((n, k, w, r)\)-PNH Problem is \((w/r)\)-competitive.

**Proof.** Let us assume that we have such an algorithm \(A_s\). Then we suggest the input \( I = (x_1, \ldots, x_n) \) such that \(A_s\) returns the wrong answer on all requests of guardians. Let \(m_1 = m_2 = m_3 = m = 3 \cdot 2^k\).

The first guardian answers \( y_i \). Due to Lemma 1 we can choose input \(X_1 \in \{0,1\}^m\) such that \(A_s\) cannot compute \(\text{PartialMOD}_k^n(X_1)\). It means that we can choose \(X_1\) such that \(y_2 = v_1 \oplus y_1\), for \(v_1 = (\#_1(X_1)/2^k) \mod 2\). By the
same reason we can pick $X^2$ such that $y_3 = v_2 \oplus y_2$, for $v_2 = (\#_1(X_2)/2^k) \mod 2$. Let us choose the input $X^3$ such that $v_3 \oplus y_3 = 1$, for $v_3 = (\#_1(X_3)/2^k) \mod 2$. Note that we guarantee that $\#_1(X_i)/2^k$ is an integer, for $i \in \{1, 2, 3\}$.

Therefore, we have: $z_3 = (\#_1(X^3)/2^k) \mod 2 \neq y_3$, $z_2 = (\#_1(X_2, X_3)/2^k) \mod 2 = v_2 \oplus v_3 \neq y_2$, $z_1 = (\#_1(X_1, X_2, X_3)/2^k) \mod 2 = v_1 \oplus v_2 \oplus v_3 \neq y_1$. We got a contradiction for input $(2, X^1, 2, X^2, 2, X^3)$. Hence the cost of output is $w$ and the competitive ration is $w/r$. □

**Theorem 3.** Let integer $s, k$ be such that $s < k = o(\log(n))$, where $n$ is the length of input. Any randomize online algorithm $R_s$ computing $(n, k, w, r)$-PNH Problem is $(r + 7w)/(8r)$-competitive.

**Proof.** By the same way as in the previous Theorem we can show that for any algorithm $R_s$ we can suggest the input such that it cannot say anything better than just guessing answers with probabilities 0.5. Therefore expected cost is $(r + 7w)/8$ and expected competitive ratio is $(r + 7w)/(8r)$. □

Note that for any deterministic online algorithm without memory restriction, we can construct input such that at least two of three guardians return wrong answers.

**Theorem 4.** Suppose a deterministic online algorithm $A$ computes $(n, k, w, r)$-PNH; then $A$ is $(w/r)$-competitive.

**Proof.** Let the algorithm $A$ receives the input $I = (x_1, \ldots, x_n) = (2, X_1, 2, X_2, 2, X_3)$, such that $X_1, X_2, X_3 \in \{0, 1\}^m$, for $m = 3 \cdot 2^k$. Let $X_1, X_2$ be such that $\text{PartialMOD}_{m}^{k}(X_1) = \text{PartialMOD}_{m}^{k}(X_2) = 0$.

Then $A$ receives part $(2, X_1, 2, X_2, 2)$ of the input and returns $y_1, y_2, y_3$. Let $b = 1$, if $y_1 + y_2 + y_3 \geq 2$; and $b = 0$, otherwise. Then we choose $X_3$ such that $\text{PartialMOD}_{m}^{k}(X_3) \neq b$. In that case $z_1 = z_2 = z_3 = \text{PartialMOD}_{m}^{k}(X_3) \neq b$. Therefore at least two of three guardians return wrong answers. Therefore cost$(I, A(I)) = w$ and $A$ is $(w/r)$-competitive. 3 □

It is easy to see that $(r + 7w)/(8r) > (w + r)/(2r)$ and $w/r > (w + r)/(2r)$ due to $r < w$. Therefore, a pure quantum algorithm is better than any deterministic or randomize online algorithm that uses less than $k$ bits of memory. And the same pure quantum online algorithm is better than any deterministic online algorithm without memory restriction.

### 3.1 Polylogarithmic Space Complexity Separation between Quantum and Deterministic Online Algorithms with Polylogarithmic Memory

Note, that above results show separation for sublogarithmic memory. For polylogarithmic memory case, we present separation between quantum and deterministic models in Theorems 5 and 6. Also, same results are right in logarithmic memory case.
Let us consider modification of \((n,k,r,w)\)-PNH problem called \((n,r,w)\)-Parity Number of Equality Hats or \((n,r,w)\)-PNEH. It is the same problem, but we use \(EQ_m(X)\) function instead of \(PartialMOD^m_n\). Boolean function \(EQ_m : \{0,1\}^r \rightarrow \{0,1\}\) is such that \(EQ(x_1,\ldots,x_{m/2},x_{m/2}+1,\ldots,x_m) = 1\), if \((x_1,\ldots,x_{m/2})) = (x_{m/2}+1,\ldots,x_m)\), and 0 otherwise. So \(z_j(I) = \bigoplus_{i=j} EQ_m(X_i)\). We suppose that \(m > 1, r < w\).

Let us construct a pure quantum online algorithm that uses \(O(\log n)\) and solves \((n,r,w)\)-PNEH.

**Algorithm 2. (Quantum Algorithm for \((n,r,w)\)-PNEH)** The pure quantum algorithm \(Q = Q_{O(\log n)}\) uses \(O(\log n)\) qubits.

**Step 1.** The algorithm emulates guessing for \(z_1(I)\). \(Q\) initializes the qubit \(|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\). And it measures the qubit before reading any input variables. It gets \(|0\rangle\) or \(|1\rangle\) with equal probability. The result of measurement is \(y_1\).

**Step 2.** The algorithm takes \(X_1\) and obtains a value of the function \(v_1 = EQ(X_1)\) using quantum fingerprinting method presented in [AV09,AV08,AKV10,AN08,AN09,AF98]. This method allows to compute the function \(EQ_m\), with probability of error by any fixed constant \(\varepsilon > 0\). The method uses \(O(\log m)\) qubits. For 0-instances, probability of error is \(\varepsilon\). And for 1-instances probability of error is 0.

**Step 3.** If \(Q\) meets 2 then it takes a value \(v_1\). Let the value be in qubit \(|\phi\rangle\). Then the algorithm applies \(CNOT\) gate for \(|\phi\rangle|\psi\rangle\). After that \(|\psi\rangle = |y_1 + v_1\rangle\). Then \(Q\) measure \(|\psi\rangle\) and returns \(y_2\).

**Step 4.** The step is similar to Step 2, but algorithm reads \(X_2\).

**Step 5.** The step is similar to Step 3, but algorithm outputs \(y_3\).

**Step 6.** The algorithm reads and skips the last part of the input. \(Q\) does not need these variable, because it guesses \(y_1\) and using this value we already can obtain \(y_2\) and \(y_3\) without \(X_3\).

Let us compute an expected cost of pairs \((I,Q(I))\). For this we construct a Table \(\|\) of probabilities and costs for all possible values \((z_1, z_2, z_3)\) in columns and all possible answers \((y_1, y_2, y_3)\) in rows.

Let us compute expected cost for each kind of inputs \(I_{z_1 z_2 z_3}:\)

\[
\begin{align*}
\text{cost}(Q(I_{100})) &= \text{cost}(Q(I_{011})) = r(1 - \varepsilon)^2/2 + w((1 - \varepsilon^2)/2 + \varepsilon), \\
\text{cost}(Q(I_{101})) &= \text{cost}(Q(I_{101})) = \text{cost}(Q(I_{100})) = \text{cost}(Q(I_{101})) = r(1 - \varepsilon)/2 + w(1 + \varepsilon)/2, \\
\text{cost}(Q(I_{110})) &= \text{cost}(Q(I_{111})) = r/2 + w/2, \\
\end{align*}
\]

So, expected ratio is \((r(1 - \varepsilon)^2/2 + w(1 - \varepsilon^2)/2 + \varepsilon))/r\). As a result we have the following theorem:

**Theorem 5.** There is \((r(1 - \varepsilon)^2/2 + w(1 - \varepsilon^2)/2 + \varepsilon))/r\)-competitive in expectation pure quantum algorithms for \((n,r,w)\)-PNEH Problem \(Q_{O(\log n)}\) with \(O(\log n)\) qubits of memory.

At the same time, if a deterministic online algorithm for \((n,r,w)\)-PNEH Problem uses a polylogarithmic number of bits; then it is \((w/r)\)-competitive. To show this claim, let us discuss a required property of \(EQ_m\) function.
Theorem 6. Let integer $s$ be such that $s = o(n)$, where $n$ is the length of input. Any deterministic online algorithm $A_s$ computing $(n, w, r)$-PNEH Problem is $(w/r)$-competitive.

Proof. Using Lemma 3 we can prove the theorem by the same way as in proof of Theorem 2.

Note that for any deterministic online algorithm without memory restriction, we can construct input such that at least two of three guardians return wrong answers.

Theorem 7. Suppose a deterministic online algorithm $A$ computes $(n, k, w, r)$-PNEH; then $A$ is $(w/r)$-competitive.
Proof. We can prove the theorem by the same way as in proof of Theorem 4.

It is easy to see that \((r(1 - \varepsilon)^2/2 + w(\frac{1+\varepsilon^2}{r^2} + \varepsilon))/r < W/r\) due to \(r < w\). Therefore, the pure quantum algorithm is better than any deterministic online algorithm without memory restriction.

3.2 On an Emulation of a Probabilistic Online Algorithm by a Quantum One

Let us show that a quantum model has at least the same power as a classical model. We use a simple technique, but it is important to show this result, because as we discussed in the introduction, for some models, quantum and classical cases are incomparable.

**Theorem 8.** Let an online optimization problem \(S\) is solved by \(c\)-competitive randomized online algorithm \(A\) using \(s\) bits of memory, then there is a \(c\)-competitive quantum online algorithm \(Q\) that computes this problem using at most \(s\) bits of classical memory and one qubit. (See Appendix A)

Pure quantum algorithms also can emulate random algorithms.

**Theorem 9.** Let an online optimization problem \(S\) is solved by \(c\)-competitive randomized online algorithm \(A\) using \(s\) bits of memory, then there is a \(c\)-competitive pure quantum online algorithm \(Q\) that computes \(S\) using at most \(s + 1\) qubits.

**Proof.** We can use the same technique as in the previous theorem, but we will store a state of classical memory in the quantum bits.

4 Advice complexity of Quantum online algorithms

4.1 Results on Model with Private Qubits

Firstly, let us show that situation with advice complexity is similar to model with restricted memory, and quantum model with advice also has at least the same power as a classical one. All proofs of theorems from this section are presented in Appendix B.

**Theorem 10.** Let an online optimization problem \(S\) is solved by \(c\)-competitive randomized online algorithm \(A\) with \(b\) advice bits, then there is a \(c\)-competitive quantum online algorithm \(Q\) that computes \(S\) using at most \(b\) advice qubits.

Let us apply the above theorem to the \((n,k,r,w)\)-PNH problem. We have shown in Theorem 4 which for any deterministic online algorithm, we can construct an input such that at least two of three guardians return wrong results. At the same time, one advice bit is enough to construct optimal an online solution.
Theorem 11. There is optimal deterministic online algorithm $B$ with 1 advice bit for $(n, k, r, w)$-PNH problem.

We will formulate two quantum online algorithms for the problem. The first one emulates a random algorithm, the second one is an optimal pure quantum online algorithm with advice, and it is based on ideas from Theorem 1.

Theorem 12. There are $(w + r)/(2r)$-competitive randomized online algorithm $R$ and $(w + r)/(2r)$-competitive quantum online algorithm $Q$ for $(n, k, r, w)$-PNH Problem.

Theorem 13. There is an optimal pure quantum algorithm $Q$ with single advice qubit that uses a single qubit of memory for solving $(n, k, r, w)$-PNH Problem.

4.2 Results on Model with Shared Entangled Qubits

For any classical online algorithm with advice, we can construct a quantum algorithm with shared qubits and two times less advice qubits. The idea is based on paper [BW92], where authors use EPR-pairs. It allows to send 2 bits of classical information using one qubit.

Theorem 14. Let $P$ be an online minimization problem, and $A$ be $c_A$-competitive deterministic online algorithm with $b_A$ advice bits, $R$ be $c_B$-competitive randomized online algorithm with $b_B$ advice bits. Then, there are $c_A$-competitive quantum online algorithm with shared qubits and $\lceil b_A/2 \rceil$ advice qubits, and $c_B$-competitive quantum online algorithm with shared qubits and $\lceil b_B/2 \rceil$ advice qubits for the same problem. (See Appendix C)

The Paging problem. Let us apply this result to the well-known Paging problem [DKP09]. We describe it in a very simplified way; a practical view of the problem can be found in the standard literature, for example, in [Kom16].

Consider a two-level memory system that consists of a small fast memory and a large slow memory. Here, each request specifies a page in the memory system. A request is served if the corresponding page is in the fast memory. If a requested page is not in the fast memory, a page fault occurs. Then some page must be moved from the fast memory to the slow memory so that the requested page can be loaded into the vacated location. A paging algorithms specifies which page to evict on a fault. The cost is the total number of evicted pages. The large slow memory has $N$ pages. The fast memory (cache) has $k < N$ pages.

Theorem 15 [DKP09]. There is an optimal deterministic online algorithm $A$ for Paging Problem, which uses $n$ bits of advice.

Hence, for the quantum case, we get the following result due to Theorem 14.

Theorem 16. There is an optimal quantum online algorithm $A$ with shared qubits for Paging Problem, which uses $n/2$ qubits of advice.

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
References

AAKK17. Farid Ablayev, Andris Ambainis, Kamil Khadiev, and Aliya Khadieva. Lower bounds and hierarchies for quantum automata communication protocols and quantum ordered binary decision diagrams with repeated test. 2017.

AF98. Andris Ambainis and Rusins Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In FOCS’98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, pages 332–341, 1998. (http://arxiv.org/abs/quant-ph/9802062).

AGK+05. Farid Ablayev, Aida Gainutdinova, Marek Karpinski, Cristopher Moore, and Christopher Pollett. On the computational power of probabilistic and quantum branching program. Information and Computation, 203(2):145–162, 2005.

AGKY14. Farid Ablayev, Aida Gainutdinova, Kamil Khadiev, and Abuzer Yakaryilmaz. Very narrow quantum obdds and width hierarchies for classical obdds. In Descriptional Complexity of Formal Systems, volume 8614 of Lecture Notes in Computer Science, pages 53–64. Springer, 2014.

AGKY16. F. Ablayev, A. Gainutdinova, K. Khadiev, and A. Yakaryilmaz. Very narrow quantum obdds and width hierarchies for classical obdds. Lobachevskii Journal of Mathematics, 37(6):670–682, 2016.

AKV10. Farid Ablayev, Airat Khasianov, and Alexander Vasiliev. On complexity of quantum branching programs computing equality-like boolean functions. ECCC, 2010.

Alb96. Susanne Albers. BRICS, Mini-Course on Competitive Online Algorithms. Aarhus University, 1996.

AN08. Andris Ambainis and Nikolajs Nahimovs. Improved constructions of quantum automata. In TQC, pages 47–56. Springer, 2008.

AN09. Andris Ambainis and Nikolajs Nahimovs. Improved constructions of quantum automata. Theoretical Computer Science, 410(20):1916–1922, 2009.

AV08. Farid Ablayev and Alexander Vasiliev. On the computation of boolean functions by quantum branching programs via fingerprinting. In Electronic Colloquium on Computational Complexity (ECCC), volume 15, 2008.

AV09. Farid Mansurovich Ablayev and AV Vasilyev. On quantum realisation of boolean functions by the fingerprinting technique. Discrete Mathematics and Applications, 19(6):555–572, 2009.

AY12. Andris Ambainis and Abuzer Yakaryilmaz. Superiority of exact quantum automata for promise problems. Information Processing Letters, 112(7):289–291, 2012.

AY15. Andris Ambainis and Abuzer Yakaryilmaz. Automata and quantum computing. Technical Report 1507.01988, arXiv, 2015.

BEY05. Allan Borodin and Ran El-Yaniv. Online computation and competitive analysis. Cambridge University Press, 2005.

BFK+16. Joan Boyar, Lene M Favrholdt, Christian Kudahl, Kim S Larsen, and Jesper W Mikkelsen. Online algorithms with advice: A survey. ACM Sigact News, 47(3):93–129, 2016.

BFK+17. Joan Boyar, Lene M Favrholdt, Christian Kudahl, Kim S Larsen, and Jesper W Mikkelsen. Online algorithms with advice: A survey. ACM Computing Surveys (CSUR), 50(2):19, 2017.
BIL09. Joan Boyar, Sandy Irani, and Kim S Larsen. A comparison of performance measures for online algorithms. In Workshop on Algorithms and Data Structures, pages 119–130. Springer, 2009.

BIL15. Joan Boyar, Sandy Irani, and Kim S Larsen. A comparison of performance measures for online algorithms. Algorithmica, 72(4):969–994, 2015.

BK09. Luca Becchetti and Elias Koutsoupias. Competitive analysis of aggregate max in windowed streaming. In Automata, Languages and Programming: 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I, pages 156–170, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

BLM15. Joan Boyar, Kim S Larsen, and Abhayandana Maiti. The frequent items problem in online streaming under various performance measures. International Journal of Foundations of Computer Science, 26(4):413–439, 2015.

BW92. Charles H Bennett and Stephen J Wiesner. Communication via one-and two-particle operators on einstein-podolsky-rosen states. Physical review letters, 69(20):2881, 1992.

DKP09. Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. RAIRO-Theoretical Informatics and Applications, 43(3):585–613, 2009.

DLO05. Reza Dorrigiv and Alejandro López-Ortiz. A survey of performance measures for on-line algorithms. SIGACT News, 36(3):67–81, 2005.

Gai15. A. F. Gainutdinova. Comparative complexity of quantum and classical obdds for total and partial functions. Russian Mathematics, 59(11):26–35, 2015.

GK15. Yiannis Giannakopoulos and Elias Koutsoupias. Competitive analysis of maintaining frequent items of a stream. Theoretical Computer Science, 562:23–32, 2015.

GKK+07. Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald De Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. In Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, pages 516–525. ACM, 2007.

Hro05. Ji Hromkovic. Záměnecniková. design and analysis of randomized algorithms: Introduction to design paradigms, 2005.

IKP+17. Rishat Ibrahimov, Kamil Khadiev, Krisjānis Prūsis, Jevgenijs Vihrovs, and Abuzer Yakaryılmaz. Zero-error affine, unitary, and probabilistic obdds. arXiv preprint arXiv:1703.07184, 2017.

KK17. Kamil Khadiev and Aliya Khadieva. Reordering Method and Hierarchies for Quantum and Classical Ordered Binary Decision Diagrams, pages 162–175. Springer International Publishing, Cham, 2017.

Kla00. Hartmut Klauck. Quantum communication complexity. In In Proc. Intl. Colloquium on Automata, Languages, and Programming (ICALP). Citeseer, 2000. arXiv preprint quant-ph/0005032.

KMRS86. Anna R Karlin, Mark S Manasse, Larry Rudolph, and Daniel D Sleator. Competitive snoopy caching. In Foundations of Computer Science, 1986., 27th Annual Symposium on, pages 244–254. IEEE, 1986.

Kom16. Dennis Komm. An Introduction to Online Computation: Determinism, Randomization, Advice. Springer, 2016.

KT06. Jon Kleinberg and Eva Tardos. Algorithm design. Pearson Education India, 2006.
LG06. François Le Gall. Exponential separation of quantum and classical online space complexity. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 67–73. ACM, 2006.

MR10. Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

SS05. Martin Sauerhoff and Detlef Sieling. Quantum branching programs and space-bounded nonuniform quantum complexity. *Theoretical Computer Science*, 334(1):177–225, 2005.

ST85. Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

Weg00. Ingo Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. SIAM, 2000.
A  The proof of Theorem 8

Let us construct the quantum algorithm $Q$ which emulates the algorithm $A$. Let $A$ be a randomized online algorithm which uses $d$ random bits from random tape. Note that a deterministic online algorithm is the partial case of a randomized algorithm with $d = 0$. Before reading the input, $Q$ initializes a qubit $|\psi\rangle = |0\rangle$. In each case, when the algorithm $Q$ should emulate reading of a random bit from random tape, it applies to $|\psi\rangle$ the Hadamard transformation $H$. Here $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. After that $Q$ measures the qubit and gets 1 or 0 with probability 0.5, this action emulates the uniform distributed random bit. By construction, $Q$ returns the same results with the same probability as $A$.

B  Proofs of Theorems 10, 11, 12, 13

**Theorem 10.** Let an online optimization problem $S$ is solved by $c$-competitive randomized online algorithm $A$ with $b$ advice bits, then there is a $c$-competitive quantum online algorithm $Q$ that computes $S$ using at most $b$ advice qubits.

**Proof.** Let us construct the quantum algorithm $Q$ which emulates the algorithm $A$. Let $A$ be a randomized online algorithm which uses $d$ random bits from random tape. Note that deterministic online algorithm is the partial case of a randomized algorithm with $d = 0$. The Adviser sends $b$ qubits with the same information as for $A$, using pure states of qubits. When $Q$ gets advice bits, then it measures them immediately and gets classical advice bits. After that $Q$ does the same actions as $A$. Let us describe the emulation of the random process. Before reading the input, $Q$ initializes qubit $|\psi\rangle$ in $|0\rangle$ state. In each case, when Algorithm $Q$ is required to emulate reading of a random bit from random tape, it applies the Hadamard transformation $H$ to $|\psi\rangle$. After that $Q$ measures the qubit and gets 1 or 0 with probability 0.5, this action emulates the uniform distributed random bit. By construction, $Q$ returns result with the same probability as $A$. □

**Theorem 11.** There is an optimal deterministic online algorithm $B$ with 1 advice bit for $(n, k, r, w)$-PNH problem.

**Proof.** Adviser sends parity of numbers of 1s over $2^k$. Let the value of this bit be $p$. Algorithm $B$ computes the number of 1s, when it gets full $2^k$ ones, then it inverts $p$. The algorithm returns $p$ on each request. It is easy to see that $B$ always returns right answers. Thus it is optimal. □

**Theorem 12.** There are $(w + r)/(2r)$-competitive randomized online algorithm $R$ and $(w+r)/(2r)$-competitive quantum online algorithm $Q$ for $(n, k, r, w)$-PNH Problem.

**Proof.** Firstly, algorithm $R$ guesses $z_1$ with equal probability. Let value of guess be $p \in \{0, 1\}$. Algorithm $R$ computes the number of 1s, when it gets full $2^k$ ones, then it inverts $p$. The algorithm returns $p$ for each request. It is easy to see that if the guess is right, then $R$ always returns the right answers, and
$R$ always returns the wrong answer otherwise. Therefore, competitive ratio is $(0.5 \cdot r + 0.5 \cdot w)/r = (r+w)/(2r)$. For the quantum case, we apply Theorem 8 and get the claim of the theorem.

**Theorem 13** There is an optimal pure quantum algorithm $Q$ with single advice qubit that uses single qubit of memory for solving $(n,k,r,w)$-PNH Problem.

*Proof.* We can construct an algorithm like in the proof of Theorem 1. The difference is that the Adviser sends one qubit $|\psi\rangle$ for $z_1$. □

## C The Proof of Theorem 14

Let us consider a deterministic online algorithm. The proof for the randomized case is the same. We interpreted the sending of advice bits as a communication game between the algorithm and the Adviser. Let the algorithm wants to receive some information from the Adviser. To perform this feat, the algorithm prepares an EPR-pair (two qubits in a state $\sqrt{1/2}|01\rangle - \sqrt{1/2}|10\rangle$) and sends one qubit of the pair to the Adviser. After that, the Adviser applies some unitary operation described in [BW92], to the qubit to encode information with one of the states. And Adviser sends the qubit to the algorithm. Then the pair of qubits can be in one of these states (basis):

$$\sqrt{1/2}(|00\rangle + |11\rangle), \sqrt{1/2}(|01\rangle + |10\rangle), \sqrt{1/2}(|01\rangle - |10\rangle), \sqrt{1/2}(|00\rangle - |11\rangle) \quad (1)$$

Each state encodes two bits of information. The algorithm can measure qubits jointly in the orthonormal basis (1), and so reliably learn which operator the Adviser applied. For $b_A$ bits the algorithm prepares $\lceil b_A/2 \rceil$ pairs and sends one qubit for each pair. Then the Adviser applies the described transformation and sends $\lceil b_A/2 \rceil$ qubits back. After that, the algorithm can get $b_A$ classical bits from received qubits.