Testing Group Commutativity

Laura Mancinska
(ID 20286922)

April 14, 2008

Contents

1 Introduction

Often we are interested in finding problems that can be solved faster if we employ the phenomena of quantum world. In this essay we will see how quantum strategies can help us in testing group commutativity. For group commutativity problem the quantum speedup will not be so impressing as we have seen, for example, in special cases of hidden subgroup problem.

Let us now remind what a *commutative* or *abelian* group is.

Definition. Group G is said to be *commutative* or *abelian* if $xy = yx$, for all $x, y \in G$.

In many cases we cast our problem in a black box model, where the input to the problem is provided by a black-box. It turns out that it is often easier to prove lower bounds in a black-box model. This model leads us to notion of query complexity. Query complexity of a problem is the number of queries we have to make in order to solve the problem. In this essay we will put group G to be tested for commutativity in a black-box.
Black-box group model

- Elements of group are encoded as words over a finite alphabet (in this essay we assume that the encoding of group elements is unique)
- Group operation is performed by a black box containing oracles O_G and O_G^{-1}

$$O_G |g, h\rangle = |g, gh\rangle$$
$$O_G^{-1} |g, h\rangle = \left|g, g^{-1}h\right\rangle$$

Note that using oracle O_G^{-1} we can get identity of the group with just one query:

$$O_G^{-1} |g, g\rangle = \left|g, g^{-1}g\right\rangle = |g, id\rangle$$

Once we have identity element we can invert any group element g using one query:

$$O_G^{-1} |g, id\rangle = \left|g, g^{-1}id\right\rangle = \left|g, g^{-1}\right\rangle$$

Thus we see, that although oracle O_G^{-1} might seem strange at first, it can serve us in finding inverses of group elements. Black-box groups were first discussed by Babai and Szemerédi in 1984. Mosca and Watrous have studied black-box groups and their properties in quantum context.

Now we are ready to formally state the problem, we want to solve.

Group commutativity problem.

Input: Generators $g_1, \ldots, g_k$ of G (specified as $n-$bit strings and n is given)
Black box: Oracles O_G and O_G^{-1}
Task: Determine whether G is abelian

2 Algorithms for group commutativity

2.1 Classical algorithms

The straightforward classical algorithm for testing group commutativity is to check each pair of group generators $g_1, \ldots, g_k$ to see if they commute. This algorithm requires $2 \cdot \binom{k}{2} \in \Theta(k^2)$ queries. Note that we are using deterministic strategy and we always get the correct answer. Pak [2] has shown that this naive algorithm has optimal query complexity up to a constant factor if only deterministic strategies are allowed.

If we allow randomized strategies, it turns out that we can do better. Pak [2] has come up with a randomized algorithm that determines whether group G is abelian in time linear in k, the number of generators of G. The algorithm itself is very simple. A bit harder it is to understand why it really works (see lemma 1). Yet first, for the purpose of explaining Pak's randomized algorithm, we need to introduce notion of *random subproduct*.

Definition. Define *random subproduct* as

$$h = g_1^{a_1} \ldots g_k^{a_k},$$

where $g_1, \ldots, g_k$ are generators of group G and $a_i \in \{0, 1\}$ takes each of the values with equal probability.

Since Pak's algorithm is not exact, there is some probability that it will give incorrect answer. In fact, the error is one-sided. If algorithm decides that group is not abelian, we can be completely sure, that it is indeed the case. This is because in such a case algorithm has found two group elements that do not commute. However, algorithm could be wrong, if it decides that group is not abelian. If we want error probability to be lower than p, we have to pick $c > \log p / \log \frac{3}{4}$, where c is the number of iterations (see algorithm below). Note that the relationship between error probability and the number of iterations is good, since exponential rate of decrease in error probability corresponds to only linear increase in the number of iterations.

Pak's randomized algorithm:

1. Pick two random subproducts h_1, h_2
2. Test whether $h_1 h_2 = h_2 h_1$
3. Repeat steps 1 and 2 for c times or until non-commuting random subproducts are found
4. Answer that G is abelian if all of the tested subproducts commuted, otherwise say that G is not abelian

Note that we need to make queries only for the execution of steps 1 and 2. In order to pick each of the two random subproducts in step 1, we need to do no more than k group operations. In order to test whether h_1 and h_2 commute, we need to do 2 group operations. Therefore, Pak's algorithm requires $2c(k+1)$ queries to ensure that the error probability is less than $1 - (3/4)^c$.

Now we turn to the task of showing that Pak's algorithm really works. This will be done with the help of the following Lemma 1 and Theorem 1.

Lemma 1 (Pak)**.** Let $g_1, \ldots, g_k$ be generators of G, H be a proper subgroup of G and let h be a random subproduct. Then

$$\Pr[h \notin H] \geq \frac{1}{2}$$

Proof. Let i be the smallest index for which $g_i \notin H$. Now we can write our random subproduct as

$$h = \left(g_1^{a_1} \ldots g_{i-1}^{a_{i-1}}\right) g_i^{a_i} \left(g_{i+1}^{a_{i+1}} \ldots g_k^{a_k}\right) = u g_i^{a_i} v,$$

where $u \in H$. We consider two cases.

1. Assume $v \in H$. Then with probability $1/2$ we have $a_i = 1$ and therefore $h = u g_i v \notin H$.
2. Assume $v \notin H$. Then with probability $1/2$ we have $a_i = 0$ and therefore $h = uv \notin H$.

Since in both of the possible cases with probability at least one half, we have $h \notin H$, we have proved the desired result. □

Theorem 1 (Pak)**.** Let $g_1, \ldots, g_k$ be generators of G and let h_1, h_2 be random subproducts. If group G is not abelian, then

$$\Pr[h_1 h_2 = h_2 h_1] < \frac{3}{4}$$

Proof. If G is not abelian then its largest abelian subgroup, the centralizer of G, $C(G)$ is a proper subroup of G. According to Lemma 1 we have

$$\Pr[h_1 \notin C(G)] \geq 1/2$$

Assume $h_1 \notin C(G)$. Then elements of G commuting with h_1, the centralizer of h_1, $C(h_1)$ form a proper subgroup of G. Therefore, again according to Lemma 1 we have

$$\Pr[h_2 \notin C(h_1)] \geq 1/2$$

If we sum up all the above, we get the desired result:

$$\Pr[h_1 h_2 = h_2 h_1] = 1 - \Pr[h_1 h_2 \neq h_2 h_1] = 1 - \Pr[h_1 \notin C(G)] \cdot \Pr[h_2 \notin C(h_1)] < \\ < 1 - \frac{1}{4} = \frac{3}{4}$$

□

It is easy to see that the above theorem implies the expressions relating the number of iterations and error probability given in the description of the Pak's randomized algorithm.

2.2 Quantum algorithm

In this section we will present a quantum algorithm for testing group commutativity with query complexity $O(k^{2/3} \log k)$. Thus, we see that with quantum strategies we can solve group commutativity problem with less queries than by using only classical strategies. The quantum algorithm discussed in this section will be based on quantum walk. We will make use of Szegedy's construction that allows us to quantize classical walk on a graph and obtain a quantum algorithm. More precisely, we will use the algorithm whose existence is asserted in theorem given by Szegedy in [3] that will follow shortly.

Now let us introduce some notation that we will need in the statement of Szegedy's theorem. Let P be a symmetric Markov chain on graph $G = (V, E)$, where δ is the eigenvalue gap of P. Let $M \subset V$ be the set of marked vertices. At each vertex we store some auxiliary data. We have complete knowledge about Markov chain P. Yet in order to find out which vertices are marked and determine the data to be stored at each vertex, we have to query a black box. Consider the following query costs:

- *Setup cost, S* - the cost associated with the preparation of the initial state which is uniform superposition over vertices V. Note that this cost in fact arises from determining the actual data to be stored at a vertex.
- *Setup cost, U* - the cost associated with updating the data stored as we move from some vertex to an adjacent one.
- *Setup cost, C* - the cost associated with checking, whether a particular vertex is marked.

Theorem 2 (Szegedy)**.** Assume that we are promised that either the set of marked vertices M is empty or $|M| \geq \varepsilon |V|$. Then there is a quantum algorithm for determining which of these two cases holds using

$$S + O\left(\frac{1}{\sqrt{\delta\varepsilon}}\right)(U + C) \tag{1}$$

queries to the black box.

In the next two subsections we will present a graph on which the walk will take place and estimate the parameters in equation (1).

2.2.1 Constructing random walk

Let S_l be the set of all l-tuples of *distinct* elements from $\{1, \ldots, k\}$. We define the group element corresponding to a l-tuple $u = (u_1, \ldots, u_l) \in S_l$ to be

$$g_u = g_{u_1} \cdots g_{u_l}$$

Finally, let t_u be the balanced binary tree with generators $g_{u_1}, \ldots, g_{u_l}$ as leaves. We put the generators in the same order as their indices where in l-tuple u, i.e., g_{u_1} is the leftmost leave and g_{u_l} is the rightmost leave. If l is not a power of two, we place the deepest leaves to the left. The value of each inner node of t_u is obtained by composing its children group elements with group operation. Note that the root of t_u will be g_u.

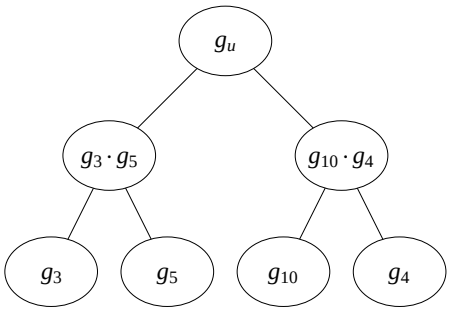

Figure 1: Binary tree t_u corresponding to l-tuple $u = (u_1, u_2, u_3, u_4) = (3, 5, 10, 4) \in S_4$

Example 1. Assume $l = 4, k = 20, u = (u_1, u_2, u_3, u_4) = (3, 5, 10, 4) \in S_4$. Then $g_u = g_3 \cdot g_5 \cdot g_{10} \cdot g_4$ and the corresponding binary t_u can be seen in Fig. 1.

Random walk on S_l

- Vertices are l-tuples from S_l. At each vertex $u = (u_1, \ldots, u_l) \in S_l$ we store the corresponding binary tree t_u
- Transitions from each vertex $u = (u_1, \ldots, u_l) \in S_l$ (denote the transition matrix with P) :
  - With probability 1/2 stay at $u = (u_1, \ldots, u_l)$
  - With probability 1/2 do:
    1. Pick a random position (leave) $i \in \{1, \ldots, l\}$ and a random generator index $j \in \{1, \ldots, k\}$
    2. If $j = u_m$ for some m, exchange u_i and u_m, else set $u_i = j$
    3. Update tree t_u

The random walk (Markov chain) on $S_l \times S_l$ that will form a basis for quantum algorithm consists of two independent random walks on S_l:

- Vertices are pairs of l-tuples $(u, v) \in S_l \times S_l$. At each vertex we store the corresponding binary trees (t_u, t_v).
- If the transition matrix for the walk on S_l is P, then the transition matrix for the walk on $S_l \times S_l$ is $P \otimes P$.

We mark those vertices $(u, v) \in S_l \times S_l$ which provide us with a certificate that group is not abelian, i.e.

$$M = \{(u, v) \in S_l \times S_l : g_u g_v \neq g_v g_u\}$$

2.2.2 Evaluating parameters

Evaluating fraction of marked vertices (ε)

Recall from Theorem 2 that $\varepsilon \leq |M| / |V|$. By taking the largest possible ε we get the best estimate for the query complexity (see equation (1)). Therefore, we choose ε to be the fraction of marked vertices, i.e., $\varepsilon = |M| / |V|$. We proceed with a lemma that will leed us to a lower bound of ε.

Lemma 2 (Magniez, Nayak)**.** If G is not abelian and $l = o(k)$, then

$$\Pr_{u,v \in S_l}[g_u g_v \neq g_v g_u] \geq const \cdot \left(\frac{l}{k}\right)^2$$

We omit the proof of this lemma (proof can be found in [1]). In order to prove the above lemma Magniez and Nayak use similar ideas as we have seen in Pak's proof of Lemma 1 and Theorem 1. However, they do not use random subproducts, since group elements g_u, where $u \in S_l$, are obtained by taking product of exactly l generators.

Theorem 3 (Magniez, Nayak)**.** If G is not abelian and $l = o(k)$, then $\varepsilon = \Omega\left((l/k)^2\right)$.

Proof. With the help of Lemma 2, we can easily lower bound the fraction of marked vertices:

$$\varepsilon = \frac{|M|}{|V|} = \frac{|\{(u,v) \in S_l \times S_l : g_u g_v \neq g_v g_u\}|}{|S_l \times S_l|} = \Pr_{u,v \in S_l}[g_u g_v \neq g_v g_u] = \Omega\left(\left(\frac{l}{k}\right)^2\right)$$

□

Evaluating spectral gap δ

Definition. Let P be a Markov chain on state space Ω. Then a *coupling* of Markov chain P is a stochastic process Q on $\Omega \times \Omega$ such that

$$\sum_{t \in \Omega} Q[(u,v),(s,t)] = P[u,s] \quad \text{for all } u,v,s \in \Omega$$
$$\sum_{s \in \Omega} Q[(u,v),(s,t)] = P[v,t] \quad \text{for all } u,v,t \in \Omega$$

where $P[u,v]$ is the probability of transition from state u to v.

These conditions say that if we view each of the components of a pair $(u,v) \in \Omega \times \Omega$ marginally, then it evolves according to the original Markov chain P.

Definition. The *coupling time* T of a coupling Q is the maximum expected time (over all initial pairs $(u,v) \in \Omega \times \Omega$) it takes for the both states in the pair to coincide:

$$T = \max_{(u,v)} \mathrm{E}\left[\min\left\{t : u(t) = v(t), u(0) = u, v(0) = v\right\}\right],$$

where $\big(u(t), v(t)\big) \in \Omega \times \Omega$ is *a* state after t steps of evolution under Q from initial state $(u,v) = \big(u(0), v(0)\big)$.

Definition. Markov chain P is called *ergodic* if there exists t such that $P^t[u,v] > 0$, for all u, v.

Now we are ready to state a lemma that can easily be deduced from theorems given in textbooks on Markov processes (see [1] for references to textbooks). This lemma will turn out useful in proving Lemma 4 that will directly lead us to a lower bound for spectral gap δ of a random walk on S_l.

Lemma 3. Let P be an ergodic Markov chain with non-negative eigenvalues and let δ be the spectral gap of P. Let T be the coupling time for any valid coupling defined on $\Omega \times \Omega$, where Ω is the state space of Markov chain P. Then

$$\delta \leq \frac{1}{4e\ T}$$

Lemma 4 (Magniez, Nayak)**.** If $l \leq \frac{k}{2}$, then the spectral gap for the walk on S_l is at least $\frac{1}{8e\ l \log l}$.

Proof. Recall that P is the transition matrix for the walk on S_l (see Section 2.2.1). Note that, because of self-loops, P can be expressed as $P = 1/2(I + P')$, where P' is a stochastic matrix. Moreover, P' is symmetric, as transition from $u \in S_l$ to $v \in S_l$ occurs with the same probability as transition from v to u (see section 2.2.1). Since eigenvalues of a symmetric stochastic matrix lie in the interval $[-1, 1]$, the eigenvalues of P lie within $1/2([1,1] + [-1,1]) = [0,1]$. Thus, eigenvalues of P are non-negative.

Note that in the random walk on S_l every state $u \in S_l$ is reachable from any state $v \in S_l$. Also, once we reach some state, we will stay there with non-zero probability due to the self-loops. Therefore, we conclude that P is ergodic.

Now we see that the walk on S_l meets the requirements in Lemma 3. Thus, we proceed with a construction of a valid coupling of transition matrix (Markov chain) P. Consider stochastic process Q on $S_l \times S_l$, where each of the states in pair $(u, v) \in S_l \times S_l$ evolve according to Markov chain P and where all of the random choices made are the same for both states in the pair. This means, if we start from (u, v) and we stay at let's say u, then we stay at v, too. On the other hand, if we move away, then we pick the same position i and index j (see Section 2.2.1) for both of the states in the pair (u, v). Let $\mathrm{Dist}(u, v)$ stand for the number of positions in which l-tuple u differs from v. Suppose that we start at $(u, v) \in S_l \times S_l$ and after one step of process Q we go to $(s, t) \in S_l \times S_l$. Then $\mathrm{Dist}(u, v) \geq \mathrm{Dist}(s, t)$. Also if $d := \mathrm{Dist}(u, v)$, then we have

$$\Pr[\mathrm{Dist}(u,v) > \mathrm{Dist}(s,t)] \geq \Pr[u_i \neq v_i] \cdot \Pr[\neg\exists t : u_t = v_t = j] =$$
$$= \frac{d}{l} \cdot \frac{k - (l-d)}{k} \geq \frac{d}{l} \cdot \frac{k-l}{k} \geq \frac{d}{l} \cdot \frac{k - \frac{k}{2}}{k} = \frac{d}{2l},$$

where i is the randomly chosen position and j is the randomly chosen generator index. This shows that distance decreases after one step of process Q with probability at least $d/2l$.

The next step is to evaluate coupling time T for coupling Q. It can be shown that if we start from state (u, v) and evolve according to Q, then both states in a pair are expected to become the same after no more than $2l \log l$ steps (see [1]). If we combine this estimate of T with Lemma 3 we obtain the desired result. □

Theorem 4 (Magniez, Nayak)**.** If $l \leq \frac{k}{2}$, then the spectral gap for the walk on $S_l \times S_l$ is at least $\frac{1}{8e\ l \log l}$.

Proof. Recall that if the transition matrix for the walk on S_l is P, then the transition matrix for the walk on $S_l \times S_l$ is $P \otimes P$. The eigenvalues of $P \otimes P$ can be obtained by taking products of two eigenvalues of P. Since P is an ergodic, stochastic matrix one of its eigenvalues is 1 and the absolute value for all the other eigenvalues is less than 1. Therefore, if λ is an eigenvalue of P that lies closest to 1, then λ plays the same role for matrix $P \otimes P$. Thus, the spectral gap for the walk on $S_l \times S_l$ is the same as for the walk on S_l, which according to Lemma 4 is at least $\frac{1}{8e\ l \log l}$. □

Evaluating setup, update and checking costs (S, U, C)

- *Setup cost*, $S = \Theta(l)$. The number of leaves and inner nodes in a binary tree is roughly the same. Since we need one group operation to determine inner node of a binary tree t_u ($u \in S_l$), it takes roughly l group operations to construct tree t_u.
- *Update cost*, $U = \Theta(\log(l))$. Note that per one step of the walk on S_l we replace no more than 2 leaves in the tree. Binary trees t_u stored at vertices have l leaves, thus their depth is $\log l$. Therefore, we can update tree as we ascend from the replaced leaves to root using $2 \log l$ operations. In fact our walk is composed of two independent walks on S_l and also we need to do uncomputation. Therefore, we need around $8 \log l$ group operations.
- *Checking cost*, $C = O(1)$. Recall that vertex $(u, v) \in S_l \times S_l$ is marked if $g_u g_v \neq g_v g_u$. Moreover, at vertex (u, v) we store trees t_u and t_v that have roots g_u and g_v respectively. Thus, the only thing we need to do is to compute $g_u g_v$ and $g_v g_u$ and compare them.

Now we have evaluated all the parameters in equation (1). So, we are ready for the theorem that gives us an upper bound of the quantum query complexity for the group commutativity problem.

Theorem 5. There is a quantum algorithm that solves group commutativity problem using $O(k^{2/3} \log k)$ queries.

Proof. We apply Theorem 2 to the random walk on $S_l \times S_l$ constructed in Section 2.2.1. Note that determining whether the set of marked vertices M is empty is equivalent to determining whether group G is abelian. Therefore, there is a quantum algorithm solving group commutativity with $S + O\left(1/\sqrt{\delta\epsilon}\right)(U + C)$ queries. If we plug in this formula the expressions obtained by evaluating parameters $\delta, \epsilon, S, U, C$, we get

$$S + \frac{1}{\sqrt{\delta\varepsilon}}(U + C) = O\left(l + \frac{k \log^{3/2} l}{\sqrt{l}}\right) \tag{2}$$

Suppose we take $l = k^{2/3}$. Note that in this way we are satisfying requirements made both in Theorem 4 and Theorem 3. Now equation (2) turns into

$$O\left(k^{2/3} \log k\right).$$

□

3 Lower bounds

Let us now state problems which will turn out useful in proving lower bounds for query complexity of group commutativity problem.

Unique collision problem.

Black-box: Function $F : \{1, \ldots, k\} \to \{1, \ldots, k\}$
Input: Value of k
Task: Output YES if there exists a *unique pair* $x \neq y \in \{1, \ldots, k\}$ such that $F(x) = F(y)$. Output NO if F is a *permutation*

This is a promise problem, since we can give whatever answer, for example, in case when there are two collisions. A variation of this problem is *unique split collision problem.* In unique split collision we output 'YES' if one of the elements from the colliding pair is in $\{1, \ldots, k/2\}$ while other is in $\{k/2 + 1, \ldots, k\}$ and k has to be even.

Lemma 5 (Szegedy, Nayak)**.** Randomized and quantum query complexity for unique split collision problem is $\Omega(k)$ and $\Omega(k^{2/3})$ respectively.

The above lemma can be proved by reducing unique split collision problem to unique collision problem (see [1] for details of the proof).

Theorem 6 (Szegedy, Nayak)**.** Randomized and quantum query complexity for group commutativity problem is $\Omega(k)$ and $\Omega(k^{2/3})$ respectively.

Proof. We prove the theorem by reducing unique split collision to group commutativity. We will construct group G that will be non-abelian if and only if oracle function $F : \{1, \ldots, k\} \to \{1, \ldots, k\}$ has a split collision.

Let I, X, Y, Z be Pauli matrices, where Y is defined unconventionally as $Y = XZ$. However, set of matrices $\{\pm I, \pm X, \pm Y, \pm Z\}$ still form a group under multiplication and no two of X, Y, Z commute. Let us define block diagonal $4k \times 4k$ unitary matrices g_i as follows:

$$g_i := \begin{cases} \bigoplus_{t=1}^{i-1} I \oplus Y \oplus \bigoplus_{t=i+1}^{k+F(i)-1} I \oplus X \oplus \bigoplus_{t=k+F(i)+1}^{2k} I & \text{if } i \leq \frac{k}{2} \\ \bigoplus_{t=1}^{i-1} I \oplus Y \oplus \bigoplus_{t=i+1}^{k+F(i)-1} I \oplus Z \oplus \bigoplus_{t=k+F(i)+1}^{2k} I & \text{if } i > \frac{k}{2} \end{cases}$$

For matrix g_i only i-th and $k + F(i)$-th block differ from identity. Consider group $G = \langle g_1, \ldots, g_k \rangle$. Note that G is not abelian if and only if there are two generators g_i and g_j such that $i \leq k/2 < j$ and $F(i) = F(j)$. Thus G is not abelian if and only if function F has a split collision. The input for group commutativity is set $\{1, \ldots, k\}$, where element i encodes generator g_i.

Now we show how to simulate oracles O_G and O_G^{-1} for the matrix group G if we are given black-box function F for the unique split collision problem. Note that we need to query F only in cases when a previously unused generator g_i is involved, since in all the other cases we already know the $4k \times 4k$ matrix explicitly. In the case when a previously unused generator g_i is involved we need

to query the value of $F(i)$ in order to find the number of the block which has to be X or Z (X if $i \leq k/2$ and Z if $i > k/2$). In each query at most 2 previously unused generators are involved, thus we need to make no more than 2 queries to F per each query to O_G or O_G^{-1}. In quantum case due to uncompution we need twice as much, i.e., no more than 4 queries. In any case the number of queries needed to simulate oracles O_G and O_G^{-1} is bounded by a constant.

We have reduced unique split collision problem to group commutativity problem of a matrix group G. We have also seen that the number of queries needed to simulate oracles O_G and O_G^{-1} is bounded by a constant. Therefore, we can claim that query complexity of group commutativity problem is at least of the same order as unique split collision problem. In other words, the same lower bounds apply for query complexities of group commutativity problem as for unique split collision problem. By applying Lemma 5 we get our theorem. □

4 Conclusion

We have seen both randomized and quantum algorithms for testing group commutativity. Both of the algorithms were based on a random walk. However, random walk that formed a basis for the quantum algorithm was not the same as the one used in Pak's randomized algorithm.

Obtained bounds for the query complexity of group commutativity problem are summarized in the the table below.

	Lower bound	Upper bound (query complexity of the algorithm presented)
Randomized	$\Omega(k)$	$O(k)$
Quantum	$\Omega\left(k^{2/3}\right)$	$O\left(k^{2/3}\log k\right)$

From the above table we see that the query complexity for Pak's randomized algorithm was optimal up to a constant. Whereas, the query complexity for the quantum algorithm presented in Section 2.2 differs from optimal by no more than a logarithmic factor.

References

[1] Frédéric Magniez, Ashwin Nayak: "Quantum Complexity of Testing Group Commutativity", Proceedings of 32nd International Colloquium on Automata, Languages and Programming, Springer-Verlag, 2005

[2] Igor Pak: "Testing commutativity of a group and the power of randomization", Electronic version at `http://www.math.mit.edu/~pak/research.html`, 2000

[3] Mario Szegedy: "Spectra of Quantized Walks and a $\sqrt{\delta\varepsilon}$-Rule", arXiv.org report quant-ph/0401053, 2004