

On the Complexity of Searching Maximum of a Function on a Quantum Computer

Maciej Goćwin¹

Abstract

We deal with a problem of finding maximum of a function from the Hölder class on a quantum computer. We show matching lower and upper bounds on the complexity of this problem. We prove upper bounds by constructing an algorithm that uses the algorithm for finding maximum of a discrete sequence. To prove lower bounds we use results for finding logical OR of sequence of bits. We show that quantum computation yields a quadratic speed-up over deterministic and randomized algorithms.

1 Introduction

Quantum algorithms yield a speed-up over deterministic and Monte Carlo algorithms for many problems. Many papers deal with quantum solution of discrete problems, starting from the work of Shor [15], followed by database search algorithm of Grover [6]. Other discrete problems were also studied, such as discrete summation, computation of the mean, median and k th-smallest element [3],[4],[5],[7],[12].

There is also a progress in studying the quantum complexity of numerical problems. The first paper dealing with a continuous problem was that of Novak [14], who considered integration of a function from the Hölder class. The problem of function approximation on quantum computer was studied by Heinrich [8],[9]. Also path integration [16] and differential equations [10],[11] on a quantum computer were investigated.

In this paper, we deal with a problem of finding maximum of a function from the Hölder class on a quantum computer. The complexity of this problem in deterministic and randomized settings on a classical computer is well known [13].

We present matching upper and lower complexity bounds in the quantum setting. We

¹*Department of Applied Mathematics, AGH University of Science and Technology,
Al. Mickiewicza 30, paw. B7, II p., pok. 24,
30-059 Cracow, Poland
gocwin@uci.agh.edu.pl, tel. +48(12)617 4405*

show that quantum computations yield a quadratic speed-up compared to deterministic and randomized algorithms over entire range of class parameters. Upper bounds are shown by constructing a suitable algorithm, which uses the optimal algorithm for finding maximum of a discrete sequence. To prove the lower bound we use the result of Nayak and Wu [12] for finding logical OR of sequence of bits.

In the next section necessary definitions are presented. Existing results for searching maximum of a sequence are shown in Section 3. The main result of this paper is contained in Theorem 1 in Section 4.

2 Quantum setting

In this section we briefly describe the model of computation. More details about quantum computing can be found in [14]. Suppose that we have a numerical problem given by a solution operator $S : F \rightarrow G$, where F is a subset of a linear function space and G is a normed space. We are interested in computing an approximation of $S(f)$ for $f \in F$ on a quantum computer. This is done by an algorithm A . The algorithm can access the input element f only by a quantum query. An output of the algorithm A for a given f is a random variable $A(f, \omega)$. For a detailed discussion of quantum algorithms and the quantum query operator, the reader is referred to [14].

We now recall what is meant by the error of an algorithm. Let $0 < \theta < 1$. The local error of algorithm A on input element f is defined by

$$e^{\text{quant}}(S, A, f, \theta) = \inf\{\varepsilon : P\{\|S(f) - A(f, \omega)\| > \varepsilon\} \leq \theta\}.$$

The number $1 - \theta$ is thus the lower bound on success probability. For $\varepsilon > 0$ the bound $e^{\text{quant}}(S, A, f, \theta) \leq \varepsilon$ holds iff algorithm A computes $S(f)$ with error at most ε and probability at least $1 - \theta$. The global error in the class F is defined as

$$e^{\text{quant}}(S, A, F, \theta) = \sup_{f \in F} e^{\text{quant}}(S, A, f, \theta).$$

For $\theta = 1/4$, we denote the error for f by $e^{\text{quant}}(S, A, f)$ and the error in the class F by $e^{\text{quant}}(S, A, F)$.

The cost of the algorithm on the input element f , $\text{cost}(A, f)$ is defined as a number of accesses to an oracle. In classical settings it is a number of function values or derivative values that is used to compute maximum of the function. In the quantum setting, by

an oracle we mean the quantum query operator. The global cost of algorithm is defined as

$$\text{cost}(A, F) = \sup_{f \in F} \text{cost}(A, f).$$

For $\varepsilon > 0$, the ε -complexity is defined as the minimal cost of an algorithm that produces an ε -approximation:

$$\text{comp}_\varepsilon^{\text{quant}}(S, F) = \min_A \{ \text{cost}(A, F) \mid e^{\text{quant}}(S, A, F) \leq \varepsilon \}.$$

In the next section some known results about complexity of finding maximum of a sequence of numbers in the quantum and classical settings are presented.

3 Searching maximum of a sequence

We recall results on a discrete maximization (minimization) problem. Consider the following problem: given a sequence $X = (x_0, x_1, \dots, x_{n-1})$ of real numbers in $[0, 1]$, find the number $x_i = \max(\min)\{x_j : j = 0, 1, \dots, n-1\}$. The cost of an algorithm is defined as a number of accesses to the oracle, which returns the input number x_i , $i = 0, 1, \dots, n-1$. Another possibility is to count a number of comparisons. In this model the oracle returns the logical value of a comparison $x_i < x_j$, where $i, j \in \{0, 1, \dots, n-1\}$. Clearly, the complexity of this problem on a classical computer is

$$\text{comp}^{\text{worst}}(n) = \Theta(n) \quad , \quad \text{comp}^{\text{rand}}(n) = \Theta(n), \quad (1)$$

where by $\text{comp}(n)$ we mean the minimal cost of an algorithm computing the maximum (minimum) value from the sequence of n numbers in suitable setting. Better results can be obtained on a quantum computer. In 1996 C. Dürr and P. Høyer in [5] presented comparison quantum algorithm for finding the minimum. This algorithm finds minimum value from the list of n items with probability greater than $1/2$ and its running time is $O(\sqrt{n})$. They based their algorithm on quantum exponential searching algorithm [2], which is a generalization of Grover's search algorithm introduced in [6]. This result establishes the upper bound on the complexity of the problem of finding maximum of a discrete sequence on a quantum computer.

Lower bounds on this problem were established by A. Nayak and F. Wu in [12]. They examine the more general problem: for $X = (x_0, x_1, \dots, x_{n-1}) \in [0, 1]^n$ and $\Delta > 0$

compute Δ -approximate k th-smallest element, i.e., a number x_i that is a j th-smallest element of X for some integer $j \in (k - \Delta, k + \Delta)$.

If $\Delta = 1$ (or less) this problem reduces to the problem of finding k th-smallest element exactly. For $k = n - 1$ k th-smallest element is the maximum value from the sequence. In [12], a quantum algorithm has been presented with the cost $O(N \log(N) \log \log(N))$, where $N = \sqrt{n/\Delta} + \sqrt{k(n-k)}/\Delta$. The algorithm is inspired by the minimum finding algorithm of Dürr and Høyer [5], and uses exponential search algorithm of Boyer *et al.* [2]. It finds Δ -approximation of k th-smallest element for any $k \in \{0, \dots, n - 1\}$ and $\Delta \geq 1$ with probability at least $2/3$.

Nayak and Wu in [12] established essentially matching lower bounds for this problem. To derive the bounds they used polynomial method introduced by R. Beals *et al.* in [1].

These results show that the complexity of searching maximum of n elements on a quantum computer is of order

$$\text{comp}^{\text{quant}}(n) = \Theta(\sqrt{n}). \quad (2)$$

The comparison of this result with (1) shows that quantum computers make a quadratic speed-up over classical computers for this problem.

4 Searching maximum of a function

We consider the problem of finding the maximum of a function from the Hölder class

$$F_d^{r,\rho} = \left\{ f : [0, 1]^d \rightarrow \mathbf{R} \mid f \in C^r, \|f\| \leq 1, \right. \\ \left. \left| D^{(r)} f(x) - D^{(r)} f(y) \right| \leq \|x - y\|^\rho \quad \forall x, y \in [0, 1]^d, \quad \forall D^{(r)} \right\},$$

where $D^{(r)}$ run through the set of all partial derivatives of order r , $r \in \mathbf{N}_0$, $0 < \rho \leq 1$ and $\|\cdot\| = \|\cdot\|_\infty$. We want to find a number

$$M(f) = \max_{t \in [0, 1]^d} f(t)$$

up to some given precision $\varepsilon > 0$, for any function f from class $F_d^{r,\rho}$ with probability not less than $3/4$.

The complexity of this problem on a classical computer in the deterministic and randomized settings is presented in [13]. We shall recall these results for a further comparison.

In the deterministic worst-case setting the local error of algorithm A on input function $f \in F_d^{r,\rho}$ is defined by

$$e^{\text{worst}}(M, A, f) = |M(f) - A(f)|$$

and the global error by

$$e^{\text{worst}}(M, A, F_d^{r,\rho}) = \sup_{f \in F_d^{r,\rho}} e^{\text{worst}}(M, A, f).$$

In the randomized setting, algorithm $A = (A(\omega))_{\omega \in \Omega}$ is a random variable on some probabilistic space (Ω, B, m) . The local error of this algorithm is defined by

$$e^{\text{rand}}(M, A, f) = \int_{\Omega} |M(f) - A(\omega)(f)| dm(\omega),$$

and the global error by

$$e^{\text{rand}}(M, A, F_d^{r,\rho}) = \sup_{f \in F_d^{r,\rho}} e^{\text{rand}}(M, A, f).$$

The cost of an algorithm in the deterministic and randomized settings is meant as a number of function values accessed by an algorithm. In the randomized setting, points where f is evaluated can be chosen randomly.

It is shown in [13] (pp. 34 and 59) that the complexity of function maximization in the Hölder class in both worst-case and randomized settings is given by

$$\text{comp}_{\varepsilon}^{\text{worst}}(M, F_d^{r,\rho}) = \Theta \left(\left(\frac{1}{\varepsilon} \right)^{\frac{d}{r+\rho}} \right) \quad \text{and} \quad \text{comp}_{\varepsilon}^{\text{rand}}(M, F_d^{r,\rho}) = \Theta \left(\left(\frac{1}{\varepsilon} \right)^{\frac{d}{r+\rho}} \right).$$

We now pass to the quantum setting. In the following result we prove that a significant improvement is achieved on a quantum computer.

Theorem 1 *Let $\varepsilon > 0$. The quantum ε -complexity of computing maximum of a function from Hölder class $F_d^{r,\rho}$ is*

$$\text{comp}_{\varepsilon}^{\text{quant}}(M, F_d^{r,\rho}) = \Theta \left(\varepsilon^{-\frac{d}{2(r+\rho)}} \right).$$

Proof:

First we prove the upper bound. We divide each edge of the cube $[0, 1]^d$ into n subintervals of equal length. We get $N = n^d$ cubes K^i , $i = 1, \dots, N$. Let t^i denote the center of cube K^i . On every cube K^i we use Taylor's expansion of f . For $t \in K^i$ we have

$$f(t) = w^i(t) + R_r(t, t^i),$$

where

$$w^i(t) = \sum_{k=0}^r \frac{1}{k!} f^{(k)}(t^i) (t - t^i)^k,$$

and

$$R_r(t, t^i) = \int_0^1 \left(f^{(r)}(\theta t + (1 - \theta)t^i) - f^{(r)}(t^i) \right) (t - t^i)^r \frac{(1 - \theta)^{r-1}}{(r - 1)!} d\theta.$$

Let

$$m_i(f) = \max_{t \in K^i} w^i(t).$$

We consider the algorithm A^* of the form

$$A^*(f) = \max_{i=1, \dots, N} \tilde{m}_i(f),$$

where $\tilde{m}_i(f)$ is an approximation of $m_i(f)$ computed by some classical algorithm. We assume that

$$|\tilde{m}_i(f) - m_i(f)| \leq \varepsilon_1 \quad \forall i = 1, \dots, N,$$

for some $\varepsilon_1 > 0$ independent of i and f . To compute $\tilde{m}_i(f)$ on a classical computer we do not need any new evaluations of f or its partial derivatives, so that information cost does not increase. (Of course, it is still not an easy task to compute $\tilde{m}_i(f)$ and it increases combinatory cost of the algorithm.) The maximum of the discrete set of numbers $\tilde{m}_1(f), \dots, \tilde{m}_N(f)$ we compute on a quantum computer, with probability not less than $\frac{3}{4}$. This is done by the optimal algorithm described in Section 3.

We now estimate the error of the algorithm defined above

$$\begin{aligned} e^{\text{quant}}(M, A^*, f) &= |M(f) - A^*(f)| = \left| \max_{t \in [0, 1]^d} f(t) - \max_{i=1, \dots, N} \tilde{m}_i(f) \right| \\ &= \left| \max_{i=1, \dots, N} \max_{t \in K^i} f(t) - \max_{i=1, \dots, N} \tilde{m}_i(f) \right| \\ &\leq \max_{i=1, \dots, N} \left| \max_{t \in K^i} f(t) - \tilde{m}_i(f) \right| \\ &\leq \max_{i=1, \dots, N} \left(\left| \max_{t \in K^i} f(t) - m_i(f) \right| + |m_i(f) - \tilde{m}_i(f)| \right) \end{aligned}$$

$$\begin{aligned}
&= \max_{i=1,\dots,N} \left| \max_{t \in K^i} f(t) - \max_{t \in K^i} w^i(t) \right| + \varepsilon_1 \\
&\leq \max_{i=1,\dots,N} \max_{t \in K^i} |f(t) - w^i(t)| + \varepsilon_1.
\end{aligned} \tag{3}$$

For $t \in K^i$ we have

$$\begin{aligned}
|f(t) - w^i(t)| &= |R_r(t, t^i)| \\
&= \left| \int_0^1 \left(f^{(r)}(\theta t + (1-\theta)t^i) - f^{(r)}(t^i) \right) (t - t^i)^r \frac{(1-\theta)^{r-1}}{(r-1)!} d\theta \right| \\
&\leq \sup_{\theta \in [0,1]} |f^{(r)}(\theta t + (1-\theta)t^i) - f^{(r)}(t^i)| \|t - t^i\|^r \int_0^1 \frac{(1-\theta)^{r-1}}{(r-1)!} d\theta \\
&\leq H \sup_{\theta \in [0,1]} \|\theta t + (1-\theta)t^i - t^i\|^\rho \|t - t^i\|^r \\
&= H \|t - t^i\|^{r+\rho} \leq H \left(\frac{1}{n} \right)^{r+\rho}.
\end{aligned}$$

The constant H depends on d and r but not on n . From this, and inequality (3) we get the error bound

$$e^{\text{quant}}(M, A^*, f) \leq H \left(\frac{1}{n} \right)^{r+\rho} + \varepsilon_1.$$

We now choose $\varepsilon_1 = (1/n)^{r+\rho}$. Then for some constant G independent of n we have

$$e^{\text{quant}}(M, A^*, f) \leq G \left(\frac{1}{n} \right)^{r+\rho}. \tag{4}$$

We examine the cost of algorithm A^* .

To compute $\tilde{m}_i(f)$ we need to know the value of f and the values of all its partial derivatives of order up to r at point t^i . So the cost of computing $\tilde{m}_i(f)$ is

$$\text{cost}(\tilde{m}_i) = \sum_{k=0}^r \binom{d+k-1}{k} = \frac{(d+r)!}{d! r!},$$

which is independent on n .

Due to (2) the cost of computing the maximum of numbers $\tilde{m}_1(f), \dots, \tilde{m}_1(f)$ on quantum computer is $O(\sqrt{N}) = O(\sqrt{n^d})$ accesses to the numbers $\tilde{m}_i(f)$. Thus, the total cost is

$$\text{cost}(A^*, f) \leq C n^{d/2} \tag{5}$$

for some constant C . Due to (4), to obtain $e^{\text{quant}}(M, A^*, f) \leq \varepsilon$ it suffices to take $K \varepsilon^{-\frac{d}{2(r+\rho)}}$ function and derivative values, where K is a constant. This completes the proof of the upper bound.

We now prove the lower bound. Assume that A is any algorithm that computes $M(f) = \max_{t \in [0,1]^d} f(t)$ for any $f \in F_d^{r,\rho}$ up to the error ε with probability not less than $\frac{3}{4}$. We denote the cost of this algorithm by $c(\varepsilon)$.

For $\varepsilon_1 > 0$, the class $F_d^{r,\rho}$ contains $n = \Theta\left(\varepsilon_1^{-\frac{d}{r+\rho}}\right)$ functions f_1, \dots, f_n with disjoint supports such that $\max_{t \in [0,1]^d} f_i(t) = \varepsilon_1$ (see [13], p. 35).

Let $\varepsilon_1 = 4\varepsilon$. Let $X = (x_1, \dots, x_n)$ be any sequence such that $x_i \in \{0, 1\} \quad \forall i = 1, \dots, n$.

Then the function

$$f_{\varepsilon_1} := \sum_{i=1}^n x_i f_i$$

belongs to the class $F_d^{r,\rho}$. Thus, algorithm A applied to f_{ε_1} computes $\max_{t \in [0,1]^d} f_{\varepsilon_1}(t)$ up to the error $\varepsilon = \frac{\varepsilon_1}{4}$, with the cost $c(\varepsilon) = c(\frac{\varepsilon_1}{4})$. That is

$$\left| \max_{t \in [0,1]^d} f_{\varepsilon_1}(t) - A(f_{\varepsilon_1}) \right| \leq \frac{\varepsilon_1}{4} \quad (6)$$

with probability not less than $\frac{3}{4}$, and cost $c(\varepsilon) = c(\frac{\varepsilon_1}{4})$.

From the definition of f_{ε_1} we see that

$$\max_{t \in [0,1]^d} f_{\varepsilon_1}(t) = \begin{cases} \varepsilon_1 & \text{if } \max_{i=1, \dots, n} x_i = 1 \\ 0 & \text{if } \max_{i=1, \dots, n} x_i = 0 \end{cases}. \quad (7)$$

If $A(f_{\varepsilon_1}) \geq \frac{3}{4}\varepsilon_1$, then due to (6)

$$\max_{t \in [0,1]^d} f_{\varepsilon_1}(t) \geq A(f_{\varepsilon_1}) - \frac{1}{4}\varepsilon_1 \geq \frac{1}{2}\varepsilon_1.$$

So, in this case, due to (7), $\max_{t \in [0,1]^d} f_{\varepsilon_1}(t) = \varepsilon_1$ and $\max_{i=1, \dots, n} x_i = 1$ with probability at least $\frac{3}{4}$. Similarly, with probability at least $\frac{3}{4}$, if $A(f_{\varepsilon_1}) \leq \frac{1}{4}\varepsilon_1$, then $\max_{t \in [0,1]^d} f_{\varepsilon_1}(t) = 0$ and $\max_{i=1, \dots, n} x_i = 0$.

Based on algorithm A , we now define an algorithm \tilde{A} , which finds the maximum of a sequence $X = (x_1, \dots, x_n)$. This algorithm is constructed as follows:

$$\begin{aligned} &\text{if } \frac{3}{4}\varepsilon_1 \leq A(f_{\varepsilon_1}) \leq \frac{5}{4}\varepsilon_1, \text{ then we put } \tilde{A}(X) = 1, \text{ and} \\ &\text{if } -\frac{1}{4}\varepsilon_1 \leq A(f_{\varepsilon_1}) \leq \frac{1}{4}\varepsilon_1, \text{ then we put } \tilde{A}(X) = 0. \end{aligned}$$

In the other cases we put $\tilde{A}(X) = 0$. With probability at least $\frac{3}{4}$, we have that $\max_{i=1, \dots, n} x_i = 0$ and $\tilde{A}(X) = 0$, or $\max_{i=1, \dots, n} x_i = 1$ and $\tilde{A}(X) = 1$. Hence, the algorithm \tilde{A} computes the maximum of n numbers x_1, \dots, x_n , such that $x_i \in \{0, 1\}$

(logical OR of the input bits), with probability not less than $\frac{3}{4}$. From [12] (see proof of Theorem 1.5 in the case of $x_i \in \{0, 1\}$) we know that the cost of such an algorithm is $\Omega(\sqrt{n})$. The cost of the algorithm A is not less than the cost of the algorithm \tilde{A} . Since

$$n = \Theta\left(\left(\frac{1}{\varepsilon_1}\right)^{\frac{d}{r+\rho}}\right) = \Theta\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{r+\rho}}\right),$$

we have that

$$c(\varepsilon) = \Omega(\sqrt{n}) = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{2(r+\rho)}}\right).$$

This completes the proof of the theorem. ■

Comparing this to the classical deterministic or random complexity of this problem, which is $\Theta\left(\varepsilon^{-\frac{d}{r+\rho}}\right)$, we see that a quantum computer makes a quadratic speed-up over classical settings. This is achieved over the entire range of r , ρ and d . For the integration problem, a quadratic speed-up over the randomized setting holds only for $(r + \rho)/d$ small.

References

- [1] R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. Quantum lower bounds by polynomials. *Proceeding of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998, see also <http://arXiv.org/abs/quant-ph/9802049>.
- [2] M. Boyer, G. Brassard, P. Høyer and A. Tapp. Tight bounds on quantum searching. *Fortschritte Der Physik* **46**, 1998, pp. 493-505, see also <http://arXiv.org/abs/quant-ph/9605034>.
- [3] G. Brassard, P. Høyer, M. Mosca, A. Tapp. Quantum amplitude amplification and estimation, 2000, <http://arXiv.org/abs/quant-ph/0005055>.
- [4] G. Brassard, P. Høyer, A. Tapp. Quantum Counting, Lect. Notes on Comp. Science **1443**, 1998, pp. 820-831, see also <http://arXiv.org/abs/quant-ph/9805082>.
- [5] C. Dürr, P. Høyer. A quantum algorithm for finding the minimum. *Proceeding of the 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 1516-1524, see also <http://arXiv.org/abs/quant-ph/9607014>.

- [6] L.K. Grover. A fast quantum mechanical algorithm for database search, *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996, pp. 212-219.
- [7] L.K. Grover. A framework for fast quantum mechanical algorithms, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 212-219, see also <http://arXiv.org/abs/quant-ph/9711043>.
- [8] S. Heinrich. Quantum approximation I. Embeddings of finite dimensional L_p spaces, *Journal of Complexity* 20 (2004), pp. 2-26, see also <http://arXiv.org/abs/quant-ph/0305030>.
- [9] S. Heinrich. Quantum approximation II. Sobolev embeddings, *Journal of Complexity* 20 (2004), pp. 27-45, see also <http://arXiv.org/abs/quant-ph/0305031>.
- [10] B. Kacewicz. Randomized and quantum algorithms yield a speed-up for initial-value problems, *Journal of Complexity* 20 (2004), pp. 821-834, see also <http://arXiv.org/abs/quant-ph/0311148>.
- [11] B. Kacewicz. Improved bounds on the randomized and quantum complexity of initial-value problems, <http://arXiv.org/abs/quant-ph/0405018>.
- [12] A. Nayak, F. Wu. The quantum query complexity of approximating the median and related statistics. *Proceedings of 31th STOC*, 1999, pp. 384-393, see also <http://arXiv.org/abs/quant-ph/9804066>.
- [13] E. Novak. *Deterministic and Stochastic Error Bounds in Numerical Analysis*. Lecture Notes in Mathematics **1349**, Springer, 1988.
- [14] E. Novak. Quantum complexity of integration, *Journal of Complexity* 17 (2001), pp. 2-16, see also <http://arXiv.org/abs/quant-ph/0008124>.
- [15] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Computing* 26 (1997), pp. 1484-1509, see also <http://arXiv.org/abs/quant-ph/9508027>.
- [16] J. F. Traub, H. Woźniakowski. Path integration on quantum computer, *Quantum Information Processing* 1 (2002), pp. 365-388, see also <http://arXiv.org/abs/quant-ph/0109113>.