Quantum Information and Computation, Vol. 0, No. 0 (2003) 000–000

Optimizing the Number of Gates in Quantum Search

Srinivasan Arunachalam[a]
QuSoft, CWI, Amsterdam, the Netherlands
srinivasan1390@gmail.com

Ronald de Wolf [b]
QuSoft, CWI and University of Amsterdam, the Netherlands
rdewolf@cwi.nl

Received (October 18, 2016)
Revised (February 1, 2017)

In its usual form, Grover's quantum search algorithm uses $O(\sqrt{N})$ queries and $O(\sqrt{N}\log N)$ other elementary gates to find a solution in an N-bit database. Grover in 2002 showed how to reduce the number of other gates to $O(\sqrt{N}\log\log N)$ for the special case where the database has a unique solution, without significantly increasing the number of queries. We show how to reduce this further to $O(\sqrt{N}\log^{(r)} N)$ gates for every constant r, and sufficiently large N. This means that, on average, the circuits between two queries barely touch more than a constant number of the $\log N$ qubits on which the algorithm acts. For a very large N that is a power of 2, we can choose r such that the algorithm uses essentially the minimal number $\frac{\pi}{4}\sqrt{N}$ of queries, and only $O(\sqrt{N}\log(\log^\star N))$ other gates.

Keywords: Quantum computing, Quantum search, Gate complexity.

Communicated by: to be filled by the Editorial

1 Introduction

One of the main successes of quantum algorithms so far is Grover's *database search* algorithm [1, 2]. Here a database of size N is modeled as a binary string $x \in \{0,1\}^N$, whose bits are indexed by $i \in \{0,\ldots,N-1\}$. A *solution* is an index i such that $x_i = 1$. The goal of the search problem is to find such a solution given access to the bits of x. If our database has Hamming weight $|x| = 1$, we say it has a *unique* solution.

The standard version of Grover's algorithm finds a solution with high probability using $O(\sqrt{N})$ database queries and $O(\sqrt{N}\log N)$ other elementary gates. It starts from a uniform superposition over all database-indices i, and then applies $O(\sqrt{N})$ identical "iterations," each of which uses one query and $O(\log N)$ other elementary gates. Together these iterations concentrate most of the amplitude on the solution(s). A measurement of the final state then yields a solution with high probability. For the special case of a database with a unique solution its number of iterations ($=$ number of queries) is essentially $\frac{\pi}{4}\sqrt{N}$, and Zalka [3] showed that this number of queries is optimal. Grover's algorithm, in

[a]Supported by ERC Consolidator Grant QPROGRESS.
[b]Partially supported by ERC Consolidator Grant QPROGRESS and by the European Commission FET-Proactive project Quantum Algorithms (QALGO) 600700.

various forms and generalizations, has been applied as a subroutine in many other quantum algorithms, and is often the main source of speed-up for those. See for example [4, 5, 6, 7, 8, 9].

In [10], Grover gave an alternative algorithm to find a unique solution using slightly more (but still $(\frac{\pi}{4}+o(1))\sqrt{N})$ queries, and only $O(\sqrt{N}\log\log N)$ other elementary gates. The algorithm is more complicated than the standard Grover algorithm, and no longer consists of $O(\sqrt{N})$ identical iterations. Still, it acts on $O(\log N)$ qubits, so on average a unitary sitting between two queries acts on only a tiny $O(\log\log N/\log N)$-fraction of the qubits. It is quite surprising that such mostly-very-sparse unitaries suffice for quantum search.

In this paper we show how Grover's reduction in the number of gates can be improved further: for every fixed r, and sufficiently large N, we give a quantum algorithm that finds a unique solution in a database of size N using $O(\sqrt{N})$ queries and $O(\sqrt{N}\log^{(r)}N)$ other elementary gates.[c] To be concrete about the latter, we assume that the set of elementary gates at our disposal is the Toffoli (controlled-controlled-NOT) gate, and all one-qubit unitary gates.

Our approach is recursive: we build a quantum search algorithm for a larger database using amplitude amplification on a search algorithm for a smaller database.[d] Let us sketch this in a bit more detail. Suppose we have an increasing sequence of database-sizes $N_1,\ldots,N_r=N$, where $N_{i+1}\approx 2^{\sqrt{N_i}}$ (of course, N needs to be sufficiently large for such a sequence to exist). The basic Grover algorithm can search a database of size N_1 using

$$Q_1=O(\sqrt{N_1}),\quad E_1=O(\sqrt{N_1}\log N_1)$$

queries and gates, respectively. We can build a search algorithm for database-size N_2 as follows. Think of the N_2-sized database as consisting of N_2/N_1 N_1-sized databases; we can just pick one such N_1-sized database at random, use the smaller algorithm to search for a solution in that database, and then use $O(\sqrt{N_2/N_1})$ rounds of amplitude amplification to boost (to 1) the N_1/N_2 probability that our randomly chosen N_1-sized database happened to contain the unique solution. Each round of amplitude amplification involves one application of the smaller algorithm, one application of its inverse, a reflection through the $\log N_2$-qubit all-0 state, and one more query. This gives a search algorithm for an N_2-sized database that uses

$$Q_2=O\left(\sqrt{\frac{N_2}{N_1}}Q_1\right)=O(\sqrt{N_2}),\quad E_2=O\left(\sqrt{\frac{N_2}{N_1}}(E_1+\log N_2)\right)$$

queries and gates respectively. Note that by our choice of $N_2\approx 2^{\sqrt{N_1}}$, we have $E_1\geq\log N_2$, so $E_2=O(\sqrt{N_2/N_1}E_1)$. Repeating this construction gives a recursion

$$Q_{i+1}=O\left(\sqrt{\frac{N_{i+1}}{N_i}}Q_i\right),\quad E_{i+1}=O\left(\sqrt{\frac{N_{i+1}}{N_i}}E_i\right).$$

[c] The constant in the $O(\cdot)$ for the number of gates depends on r. The iterated binary logarithm is defined as $\log^{(s+1)}=\log\circ\log^{(s)}$, where $\log^{(0)}$ is the identity function. The function $\log^\star N$ is the number of times the binary logarithm must be iteratively applied to N to obtain a number that is at most 1: $\log^\star N=\min\{r\geq 0:\log^{(r)}N\leq 1\}$.

[d] The idea of doing recursive applications of amplitude amplification to search increasingly larger database-sizes is reminiscent of the algorithm of Aaronson and Ambainis [11] for searching an N-element database that is arranged in a d-dimensional grid. However, their goal was to design a search algorithm for the grid with nearest-neighbor gates and with optimal number of *queries* (they succeeded for $d>2$). It was not to optimize the number of *gates*. If one writes out their algorithm as a quantum circuit, it still has roughly $\sqrt{N}\log N$ gates.

The constant factor in the $O(\cdot)$ blows up by a constant factor in each recursion, so after r steps this unfolds to

$$Q_r = O(\exp(r)\sqrt{N}), \quad E_r = O(\exp(r)\sqrt{N}\log N_1).$$

Since $N_1, \ldots, N_r = N$ is (essentially) an exponentially increasing sequence, we have $\log N_1 = O(\log^{(r)} N)$.

The result we prove in this paper is stronger: it does not have the $\exp(r)$ factor. Tweaking the above idea to avoid this factor is somewhat delicate, and will take up the remainder of this paper. In particular, in order to get close to the optimal query complexity $\frac{\pi}{4}\sqrt{N}$, it is important (and different from Grover's approach) that the intermediate amplitude amplification steps do *not* boost the success probability all the way to 1. The reason is that amplitude amplification is less efficient when boosting large success probabilities to 1 than when boosting small success probabilities to somewhat larger success probabilities. Our final algorithm will boost the success probability to 1 only at the very end, after all r recursion steps have been done. Because the calculations involved are quite fragile, and tripped us up multiple times, the proofs in the body of the paper are given in much detail.

If N is a power of 2, then choosing $r = \log^\star N$ in our result and being careful about the constants, we get an exact quantum algorithm for finding a unique solution using essentially the optimal $\frac{\pi}{4}\sqrt{N}$ queries, and $O(\sqrt{N}\log(\log^\star N))$ elementary gates. Note that our algorithm on average uses only $O(\log(\log^\star N))$ elementary gates in between two queries, which is barely more than constant. Once in a while a unitary acts on many more qubits, but the average is only $O(\log(\log^\star N))$.

Possible objections. To pre-empt the critical reader, let us mention two objections one may raise against the fine-grained optimization of the number of elementary gates that we do here. First, one query acts on $\log N$ qubits, and when itself implemented using elementary gates, any oracle that's worth its salt would require $\Omega(\log N)$ gates. Since $\Omega(\sqrt{N})$ queries are necessary, a fair way of counting would say that just the queries themselves already have "cost" $\Omega(\sqrt{N}\log N)$, rendering our (and Grover's [10]) gate-optimizations moot. Second, to do exact amplitude amplification in our recursion steps, we allow infinite-precision single-qubit phase gates. This is not realistic, as in practice such gates would have to be approximated by more basic gates. Our reply to both would be: fair enough, but we still find it quite surprising that query-efficient search algorithms only need to act on a near-constant number of qubits in between the queries on average. It is interesting that after nearly two decades of research on quantum search, the basic search algorithm can still be improved in some ways. It may even be possible to optimize our results further to use $O(\sqrt{N})$ elementary gates, which would be even more surprising.

2 Preliminaries

Let $[n] = \{1, \ldots, n\}$. We use the binary logarithm throughout this paper. We will typically assume for simplicity that the database-size N is a power of 2, $N = 2^n$, so we can identify indices i with their binary representation $i_1 \ldots i_n \in \{0,1\}^n$. We can access the database by means of *queries*. A query corresponds to the following unitary map on $n+1$ qubits:

$$O_x : |i, b\rangle \mapsto |i, b \oplus x_i\rangle,$$

where $i \in \{0, \ldots, N-1\}$ and $b \in \{0,1\}$. Given access to an oracle of the above type, we can make a phase query $O_{x,\pm} : |i\rangle \to (-1)^{x_i}|i\rangle$ as follows: start with $|i,1\rangle$ and apply the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ to the last qubit to obtain $|i\rangle|-\rangle$, where $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Apply O_x

to $|i\rangle|-\rangle$ to obtain $(-1)^{x_i}|i\rangle|-\rangle$. Finally, apply the Hadamard gate to the last qubit, sending the state to $(-1)^{x_i}|i,1\rangle$.

Let $D_n = 2|0^n\rangle\langle 0^n| - \mathrm{Id}$ be the n-qubit unitary that reflects through $|0^n\rangle$. It is not hard to see that this can be implemented using $O(n)$ elementary gates and $n-1$ ancilla qubits that all start and end in $|0\rangle$ (and that we often will not even write explicitly). Specifically, one can apply $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ gates to each of the n qubits, then use $n-1$ Toffoli gates into $n-1$ ancilla qubits to compute the logical AND of the first n qubits, then apply $-Z$ to the last qubit (which negates the basis states where this AND is 0), and reverse the Toffolis and Xs.

Amplitude amplification is a technique that can be used to efficiently boost quantum search algorithms with a known success probability a to higher success probability. We will invoke the following theorem from [2] in the proof of Theorem 2 later. For the sake of completeness we include a proof in the appendix.

Theorem 1 *Let $N = 2^n$. Suppose there exists a unitary quantum algorithm $\mathcal{A}$ that finds a solution in database $x \in \{0,1\}^N$ with known probability a, in the sense that measuring $\mathcal{A}|0^n\rangle$ yields a solution with probability exactly a. Let $a' \in [a,1]$ and $w = \lceil \frac{\arcsin(\sqrt{a'})}{2\arcsin(\sqrt{a})} - \frac{1}{2} \rceil$. Then there exists a quantum algorithm $\mathcal{B}$ that finds a solution with probability exactly a' using $w+1$ applications of algorithm $\mathcal{A}$, w applications of $\mathcal{A}^{-1}$, w additional queries, and $4w(n+2)$ additional elementary gates. In total, $\mathcal{B}$ uses $(2w+1)Q + w$ queries and $w(4n+2E+8)+E$ elementary gates.*

A very simply algorithm to which we can apply this theorem is $\mathcal{A} = H^{\otimes n}$. If our N-bit database has a unique solution, then the success probability is $a = 1/N$. Let $a' = 1/k$ for some integer $k \geq 2$. Then, Theorem 1 implies an algorithm $\mathcal{C}^{(1)}$ that finds a solution with probability exactly $1/k$ using w queries and at most $O(w \log N)$ other elementary gates, where $w \leq \lceil \frac{\sqrt{N}(1+1/k)}{2\sqrt{k}} - \frac{1}{2} \rceil$ (this upper bound on w follows because $\arcsin(z) \geq z$, and $\sin(\frac{1+1/k}{\sqrt{k}}) \geq \frac{1}{\sqrt{k}}$ since $\sin(z) \geq z - z^3/6$ for $z \geq 0$).

In order to amplify the probability of an algorithm from $1/k$ to 1 we use the following corollary.

Corollary 1 *Let $k \geq 2$, n be integers, $N = 2^n$. Suppose there exists a quantum algorithm $\mathcal{D}$ that finds a unique solution in an N-bit database with probability exactly $1/k$ using $Q \geq \sqrt{k}$ queries and E elementary gates. Then there exists a quantum algorithm that finds the unique solution with probability 1 using at most $\frac{\pi}{2}Q\sqrt{k}(1+\frac{2}{\sqrt{k}})^2$ queries and $O(\sqrt{k}(n+E))$ elementary gates.*

Proof: Applying Theorem 1 to algorithm $\mathcal{D}$ with $a = 1/k, a' = 1$, we obtain an algorithm that succeeds with probability 1 using at most $w'(2Q+1)+Q$ queries and $O(w'(n+E))$ gates, where

$$w' = \left\lceil \frac{\arcsin(1)}{2\arcsin(1/\sqrt{k})} - \frac{1}{2} \right\rceil \leq \frac{\pi}{4}(\sqrt{k}+1),$$

using $\arcsin(x) \geq x$ and $\lceil \frac{\pi}{4}\sqrt{k} - \frac{1}{2} \rceil \leq \frac{\pi}{4}(\sqrt{k}+1)$. Hence, the total number of queries in this new algorithm is at most

$$\begin{aligned} \frac{\pi}{4}(\sqrt{k}+1)(2Q+1)+Q &= \frac{\pi}{2}Q(\sqrt{k}+1)\Big(1+\frac{1}{2Q}+\frac{2}{\pi(\sqrt{k}+1)}\Big) \\ &\leq \frac{\pi}{2}Q(\sqrt{k}+1)(1+\frac{2}{\sqrt{k}}) \\ &\leq \frac{\pi}{2}Q\sqrt{k}(1+\frac{2}{\sqrt{k}})^2, \end{aligned}$$

where we used $Q \geq \sqrt{k}$ and $\pi(\sqrt{k}+1) \geq 2\sqrt{k}$ in the first inequality. The total number of gates is $O(\sqrt{k}(n+E))$. $\square$

The following easy fact will be helpful to get rid of some of the ceilings that come from Theorem 1.

Fact 1 *If $k \geq 2$ and $\alpha \geq k$, then* $\lceil \frac{\alpha}{2}(1+\frac{1}{k}) - \frac{1}{2} \rceil \leq \frac{\alpha}{2}(1+\frac{2}{k})$.

Fact 2 *If $k \geq 3$ and $i \geq 2$, then* $(2i+8)\log k < k^{i+1}$.

Proof: Fixing $i = 2$, it is easy to see that $12 \log k < k^3$ for $k \geq 3$. Similarly, fix $k = 3$ and observe that $(2i+8)\log 3 < 3^{i+1}$ for all $i \geq 2$. This implies the result for all $k \geq 3$ and $i \geq 2$, because the right-hand side grows faster than the left-hand side in both i and k. $\square$

3 Improving the gate complexity for quantum search

In this section we give our main result, which will be proved by recursively applying the following theorem.

Theorem 2 *Let $k \geq 4$, $n \geq m + 2\log k$ be integers, $M = 2^m$ and $N = 2^n$. Suppose there exists a quantum algorithm $\mathcal{G}$ that finds a unique solution in an M-bit database with a known success probability that is at least $1/k$, using $Q \geq k+2$ queries and E other elementary gates. Then there exists a quantum algorithm that finds a unique solution in an N-bit database with probability exactly $1/k$, using Q' queries and E' other elementary gates where,*

$$Q' \leq Q\sqrt{N/M}(1+4/k), \qquad E\sqrt{N/M} \leq E' \leq (3n+E)\sqrt{N/M}(1+3/k).$$

Proof: Consider the following algorithm $\mathcal{A}$:

1. Start with $|0^n\rangle$.
2. Apply the Hadamard gate to the first $n-m$ qubits, leaving the last m qubits as $|0^m\rangle$. The resulting state is a uniform superposition over the first $n-m$ qubits $\frac{1}{\sqrt{N/M}} \sum_{y \in \{0,1\}^{n-m}} |y\rangle|0^m\rangle$.
3. Apply the unitary $\mathcal{G}$ to the last m qubits (using queries to x, with the first $n-m$ address bits fixed).

The final state of algorithm $\mathcal{A}$ is

$$(H^{\otimes(n-m)} \otimes \mathcal{G})|0^n\rangle = \frac{1}{\sqrt{N/M}} \sum_{y \in \{0,1\}^{n-m}} |y\rangle\mathcal{G}|0^m\rangle.$$

The state $|y\rangle\mathcal{G}|0^m\rangle$ depends on y, because here $\mathcal{G}$ restricts to the M-bit database that corresponds to the bits in x whose n-bit address starts with y. Let t be the n-bit address corresponding to the unique solution in the database $x \in \{0,1\}^N$. Then the probability of observing $|t_1 \dots t_n\rangle$ in the state $|t_1 \dots t_{n-m}\rangle\mathcal{G}|0^m\rangle$ is at least $1/k$. Suppose $\sqrt{a}$ is the amplitude of t in the final state after $\mathcal{A}$, then we have that $a \geq \frac{M}{kN}$. The total number of queries of algorithm $\mathcal{A}$ is Q (from Step 3) and the total number of elementary gates is $n-m+E$ (from Steps 2 and 3).

Applying Theorem 1 to algorithm $\mathcal{A}$ by choosing $a' = 1/k$, we obtain an algorithm $\mathcal{B}$ using at most $w(2Q+1)+Q$ queries and $w(4n+2E+8)+E$ gates (from Theorem 1), where

$$w = \left\lceil \frac{\arcsin(\sqrt{a'})}{2\arcsin(\sqrt{a})} - \frac{1}{2} \right\rceil \leq \left\lceil \frac{\sqrt{1/k}(1+1/k)}{2\sqrt{a}} - \frac{1}{2} \right\rceil \leq \left\lceil \frac{\sqrt{N}(1+1/k)}{2\sqrt{M}} - \frac{1}{2} \right\rceil \leq \frac{\sqrt{N}(1+2/k)}{2\sqrt{M}},$$

where the first inequality uses $\sin(\frac{1+1/k}{\sqrt{k}}) \geq \frac{1}{\sqrt{k}}$ (since $\sin(z) \geq z - z^3/6$ for $z \geq 0$), the second inequality follows from $\arcsin(z) \geq z$ and the third inequality uses Fact 1 ($\sqrt{N/M} \geq k$ because $n \geq m + 2\log k$).

The total number of queries in algorithm $\mathcal{B}$ is at most

$$\begin{aligned}
w(2Q+1)+Q &\leq Q\sqrt{N/M}(1+2/k) + \frac{1}{2}\sqrt{N/M}(1+2/k) + Q \\
&\leq Q\sqrt{N/M}(1+2/k) + \frac{Q}{2k}\sqrt{N/M} + \frac{Q}{k}\sqrt{N/M} \\
&\leq Q\sqrt{N/M}(1+4/k)
\end{aligned}$$

where we used $Q \geq k+2$ and $\sqrt{N/M} \geq k \geq 4$ in the second inequality. The number of gates in $\mathcal{B}$ is

$$w(4n+2E+8)+E \leq \sqrt{N/M}(1+2/k)(2n+E+4)+E \leq (3n+E)\sqrt{N/M}(1+3/k),$$

where we used $\sqrt{N/M} \geq 4$ in the second inequality.

It is not hard to see that the number of gates in $\mathcal{B}$ is at least $E\sqrt{N/M}$. □

Applying Theorem 1 once to an algorithm that finds the unique solution in an M-bit database with probability $1/\log\log N$, we get the following corollary, which was essentially the main result of Grover [10].

Corollary 2 *Let $n \geq 25$ and $N = 2^n$. There exists a quantum algorithm that finds a unique solution in a database of size N with probability 1, using at most $(\frac{\pi}{4}+o(1))\sqrt{N}$ queries and $O(\sqrt{N}\log\log N)$ other elementary gates.*

Proof: Let $m = \lceil \log(n^2k^3) \rceil$ and $k = \log\log N$. Let $\mathcal{C}^{(1)}$ be the algorithm (described after Theorem 1) on an M-bit database with $M = 2^m$ that finds the solution with probability $1/k$. Observe that $k \geq 4$ and $m + 2\log k \leq \log(2n^2k^5) \leq n$ (where the last inequality is true for $n \geq 25$), hence we can apply Theorem 2 using $\mathcal{C}^{(1)}$ as our base algorithm. This gives an algorithm $\mathcal{C}^{(2)}$ that finds the solution with probability exactly $1/k$. The total number of queries in algorithm $\mathcal{C}^{(2)}$ is at most

$$\begin{aligned}
\left\lceil \frac{\sqrt{M}(1+1/k)}{2\sqrt{k}} - \frac{1}{2} \right\rceil \cdot \left(\sqrt{N/M}(1+4/k)\right) &\leq \frac{\sqrt{M}(1+2/k)}{2\sqrt{k}}\sqrt{N/M}(1+4/k) \\
&\leq \sqrt{\frac{N}{4k}}(1+4/k)^2,
\end{aligned}$$

where the expression on the left is the contribution from Theorem 2. The first inequality above follows from Fact 1 (since $m \geq 4\log k$). The total number of gates in $\mathcal{C}^{(2)}$ is

$$\begin{aligned}
&O\Big(\Big(3n + \Big\lceil \frac{\sqrt{M}(1+\frac{1}{k})}{2\sqrt{k}} - \frac{1}{2} \Big\rceil \log M\Big)\sqrt{\frac{N}{M}}(1+\frac{3}{k})\Big) \\
&\leq O\Big(\sqrt{\frac{N}{k}}\Big(\frac{3n\sqrt{k}(1+3/k)}{\sqrt{M}} + (1+3/k)^2\log M\Big)\Big) \\
&\leq O\Big(\sqrt{\frac{N}{k}}\Big(\frac{3}{k} + (1+3/k)^2\log M\Big)\Big) \\
&\leq O\Big(\sqrt{\frac{N}{k}}\Big(1+\frac{3}{k}\Big)^3\log\log N\Big),
\end{aligned}$$

where we used Fact 1 in the first inequality, $n\sqrt{k}(1+3/k) \leq \sqrt{M}/k$ (since $m \geq \log(n^2k^3)$) in the second inequality and $\log M = O(\log\log N)$ in the last inequality. Applying Corollary 1 to algorithm $\mathcal{C}^{(2)}$, we obtain an algorithm that succeeds with probability 1 using at most

$$\frac{\pi}{2}\Big(\sqrt{\frac{N}{4k}}(1+\frac{4}{k})^2\Big)\cdot\Big(\sqrt{k}(1+\frac{2}{\sqrt{k}})^2\Big) \leq \frac{\pi}{4}\sqrt{N}\Big(1+\frac{4}{\sqrt{k}}\Big)^4$$

queries and

$$O\Big(n\sqrt{k}+\sqrt{N}\Big(1+\frac{3}{k}\Big)^3\log\log N\Big) \leq O\Big(\sqrt{N}\Big(1+\frac{3}{k}\Big)^3\log\log N\Big)$$

gates, since $n\sqrt{k} \leq \sqrt{N}\log\log N$ (which is true for $n \geq 25$). Since $k = \log\log N$, it follows that the query complexity is at most $(\frac{\pi}{4}+o(1))\sqrt{N}$ and the gate complexity is $O(\sqrt{N}\log\log N)$. □

We can now use Theorem 2 recursively by starting from the improved algorithm from Corollary 2. This gives query complexity $O(\sqrt{N})$ and gate complexity $O(\sqrt{N}\log\log\log N)$. Doing this multiple times and being careful about the constant (which grows in each step of the recursion), we obtain the following result:

Theorem 3 *Let k be a power of 2 and N a sufficiently large power of 2. For every $r \in [\log^\star N]$, $k \in \{4,\ldots,\log\log N\}$, there exists a quantum algorithm that finds a unique solution in a database of size N with probability exactly $1/k$, using at most*

$$\sqrt{\frac{N}{4k}}(1+4/k)^r \text{ queries and } O\left(\sqrt{\frac{N}{k}}(1+6/k)^{2r-1}\max\{\log k, \log^{(r)} N\}\right) \text{ other elementary gates.}$$

Proof: We begin by defining a sequence of integers $n_1,\ldots,n_r$ such that $n_r = \log N$ and $n_{i-1} = \max\{(2i+6)\log k, \lceil\log(n_i^2k^3)\rceil\}$ for $i \in \{2,\ldots,r\}$. Note that $n_1 \geq 10\log k \geq 20$ (since $k \geq 4$). We first prove the following claim about this sequence.

Claim 1 *If $i \in \{2,\ldots,r\}$, then $n_{i-1}+2\log k \leq n_i$.*

Proof: We use downward induction on i. For the base case $i = r$, note that $n_r = \log N$. Firstly, note that $(2r+6)\log k \leq \lceil\log(n_r^2k^3)\rceil$ for sufficiently large N and $k \leq \log\log N$, hence $n_{r-1} = \max\{(2r+6)\log k, \lceil\log(n_r^2k^3)\rceil\} = \lceil\log(n_r^2k^3)\rceil$. It follows that

$$n_{r-1}+2\log k = \lceil\log(n_r^2k^3)\rceil + 2\log k \leq \log(2n_r^2k^5) \leq \log N = n_r,$$

where the last inequality assumed N is sufficiently large and used $k \leq \log\log N$.

For the inductive step, assume we have $n_j + 2\log k \leq n_{j+1}$. We now prove $n_{j-1}+2\log k \leq n_j$ by considering the two possible values for n_{j-1}.

Case 1. $n_{j-1} = (2j+6)\log k$. Then we have

$$n_{j-1}+2\log k = (2j+8)\log k \leq \max\{(2j+8)\log k, \lceil\log(n_{j+1}^2k^3)\rceil\} = n_j.$$

Case 2. $n_{j-1} = \lceil\log(n_j^2k^3)\rceil$. We first show $n_{j-1} \leq n_j$:

$$n_{j-1} \leq \lceil\log(n_{j+1}^2k^3)\rceil \leq \max\{(2j+8)\log k, \lceil\log(n_{j+1}^2k^3)\rceil\} = n_j,$$

where the first inequality uses the induction hypothesis. We can now conclude the inductive step:

$$n_{j-1}+2\log k \leq \log(2n_j^2k^5) = 1+2\log n_j + 5\log k \leq n_j/2 + 5\log k \leq n_j/2 + n_j/2 = n_j.$$

In the first inequality above we use $n_{j-1} \le \log(2n_j^2 k^3)$. In the second inequality we use $n_j \ge n_1 \ge 10\log k \ge 20$ (since $n_{j-1} \le n_j$ for $j \in \{2,\ldots,r\}$ and $k \ge 4$) to conclude $1 + 2\log n_j \le n_j/2$ (which is true for $n_j \ge 20$) and in the last inequality we use $5\log k \le n_j/2$. $\square$

Using the sequence $n_1,\ldots,n_r$, we consider r database-sizes $2^{n_1} = N_1 \le 2^{n_2} = N_2 \le \cdots \le 2^{n_r} = N_r = N$. For each $i \in [r]$, we will construct a quantum algorithm $\mathcal{C}^{(i)}$ on a database of size N_i that finds a unique solution with probability exactly $1/k$. Let Q_i and E_i be the query complexity and gate complexity, respectively, of algorithm $\mathcal{C}^{(i)}$. We have already constructed the required algorithm $\mathcal{C}^{(1)}$ (described after Theorem 1) on an N_1-bit database using

$$Q_1 = \left\lceil \frac{\sqrt{N_1}(1+1/k)}{2\sqrt{k}} - \frac{1}{2} \right\rceil \le \frac{\sqrt{N_1}(1+2/k)}{2\sqrt{k}}$$

queries, where the inequality follows from Fact 1 (since $N_1 \ge k^{10}$). Also, note that

$$Q_1 \ge \frac{\sqrt{N_1}(1+1/k)}{2\sqrt{k}} - 1 \ge k+2,$$

where the first inequality uses $N_1 \ge k^{10}$, and the second inequality uses $k \ge 4$. From Theorem 1, the number of gates E_1 used by $\mathcal{C}^{(1)}$ is

$$\begin{aligned}
\left\lceil \frac{\sqrt{N_1}(1+1/k)}{2\sqrt{k}} - \frac{1}{2} \right\rceil (6\log N_1 + 8) + \log N_1 &\le \frac{\sqrt{N_1}(1+2/k)}{\sqrt{k}}(3\log N_1 + 4) + \log N_1 \\
&\le \frac{4\sqrt{N_1}(1+2/k)}{\sqrt{k}} \log N_1 + \log N_1 \\
&\le \frac{4\sqrt{N_1}(1+3/k)}{\sqrt{k}} \log N_1,
\end{aligned}$$

where we use Fact 1 (since $N_1 \ge k^{10}$) in the first inequality and $N_1 \ge k^{10}$ in the second and third inequality. It is not hard to see that $E_1 \ge \sqrt{N_1/(4k)}$.

For $i \in \{2,\ldots,r\}$, we apply Theorem 2 using $\mathcal{C}^{(i-1)}$ as the base algorithm and we obtain an algorithm $\mathcal{C}^{(i)}$ that succeeds with probability exactly $1/k$. We showed earlier in Claim 1 that $n_{i-1} + 2\log k \le n_i$ and it also follows that $k+2 \le Q_1 \le \cdots \le Q_r$ (since the database-sizes $N_1,\ldots,N_r$ are non-decreasing). Hence both assumptions of Theorem 2 are satisfied. The total number of queries used by $\mathcal{C}^{(i)}$ is

$$Q_i \le \sqrt{\frac{N_i}{N_{i-1}}} Q_{i-1}\Big(1 + \frac{4}{k}\Big). \tag{1}$$

In order to analyze the number of gates used by $\mathcal{C}^{(i)}$ we need the following two claims.

Claim 2 $E_i \ge \sqrt{N_i/(4k)}$ *for all* $i \in [r]$.

Proof: The proof is by induction on i. For the base case, we observed in Theorem 2 that $E_1 \ge \sqrt{N_1/(4k)}$. For the induction step assume $E_{i-1} \ge \sqrt{N_{i-1}/(4k)}$. The claim follows immediately from the lower bound on E' in Theorem 2 since $E_i \ge E_{i-1}\sqrt{N_i/N_{i-1}} \ge \sqrt{N_i/(4k)}$. $\square$

Claim 3 *Suppose* $n_1 = \lceil \log(n_2^2 k^3) \rceil$. *Then* $n_{i-1} = \lceil \log(n_i^2 k^3) \rceil$ *for all* $i \in \{2,\ldots,r\}$.

Proof: We prove the claim by induction on i. The base case $i = 2$ is the assumption of the claim. For the inductive step, assume $n_{i-1} = \lceil \log(n_i^2 k^3) \rceil$ for some $i \ge 2$. We have

$$\log(n_i^2 k^4) \ge \lceil \log(n_i^2 k^3) \rceil \ge (2i+6)\log k = \log(k^{2i+6})$$

where the second inequality is because of the definition of n_{i-1}. Hence

$$n_i \geq k^{i+1} > (2i+8)\log k$$

using Fact 2 ($k \geq 3$ and $i \geq 2$ hold by the assumption of the theorem and claim respectively). Hence $n_i = \max\{(2i+8)\log k, \lceil \log(n_{i+1}^2 k^3)\rceil\}$ must be equal to the second term in the max. This concludes the proof of the inductive step and hence of the claim. □

Recursively it follows that the number of gates E_i used by $\mathcal{C}^{(i)}$ is at most

$$\begin{aligned}\sqrt{\frac{N_i}{N_{i-1}}}(E_{i-1}+3n_i)(1+3/k) &\leq \sqrt{\frac{N_i}{N_{i-1}}}E_{i-1}\Big(1+3n_i\sqrt{\frac{4k}{N_{i-1}}}\Big)(1+3/k) \\ &\leq \sqrt{\frac{N_i}{N_{i-1}}}E_{i-1}(1+6/k)^2,\end{aligned} \tag{2}$$

where we used Claim 2 in the first inequality and $n_i \leq \sqrt{\frac{N_{i-1}}{k^3}}$ in the last inequality (which clearly holds if $n_{i-1} = (2i+6)\log k \geq \lceil \log(n_i^2 k^3)\rceil$). Unfolding the recursion in Equations (1) and (2), we obtain

$$Q_r \leq \sqrt{\frac{N_r}{4k}}\Big(1+\frac{4}{k}\Big)^r, \qquad E_r \leq 4\sqrt{\frac{N_r}{k}}\Big(1+\frac{6}{k}\Big)^{2r-1}\log N_1.$$

It remains to show that n_1, defined as $\max\{10\log k, \lceil \log(n_2^2 k^3)\rceil\}$, is $O(\max\{\log k, \log^{(r)} N\})$. If $n_1 = 10\log k$, then we are done. If $n_1 = \lceil \log(n_2^2 k^3)\rceil$, we can use Claim 3 to write

$$n_{i-1} = \lceil 2\log n_i + 3\log k\rceil \leq 4\log n_i, \quad \text{for } i \in \{2,\ldots,r\},$$

where the last inequality follows from $k \leq n_2^{1/3} \leq n_i^{1/3}$ (using $\lceil \log(n_2^2 k^3)\rceil \geq 10\log k$ and Claim 1 for the first and second inequality respectively). Since $n_r = \log N$, it follows that $n_1 = O(\log^{(r)} N)$. We conclude $n_1 = O(\max\{\log k, \log^{(r)} N\})$. □

The following is our main result:

Corollary 3

- *For every constant integer $r > 0$ and sufficiently large $N = 2^n$, there exist a quantum algorithm that finds a unique solution in a database of size N with probability 1, using $(\frac{\pi}{4}+o(1))\sqrt{N}$ queries and $O(\sqrt{N}\log^{(r)} N)$ gates,*
- *For every $\varepsilon > 0$ and sufficiently large $N = 2^n$, there exist a quantum algorithm that finds a unique solution in a database of size N with probability 1, using $(\frac{\pi}{4}+\varepsilon)\sqrt{N}$ queries and $O(\sqrt{N}\log(\log^\star N))$ gates.*

Proof: Applying Corollary 1 to algorithm $\mathcal{C}^{(r)}$ (as described in Theorem 3), with some $k \leq \log\log N$ to be specified later, we obtain an algorithm that succeeds with probability 1 using at most

$$\frac{\pi}{2}\Big(\sqrt{\frac{N}{4k}}\Big(1+\frac{4}{k}\Big)^r\Big)\cdot\Big(\sqrt{k}\Big(1+\frac{2}{\sqrt{k}}\Big)^2\Big) \leq \frac{\pi}{4}\sqrt{N}\Big(1+\frac{4}{\sqrt{k}}\Big)^{r+2}$$

queries and

$$O\Big(\sqrt{k}n+\sqrt{N}\Big(1+\frac{6}{k}\Big)^{2r-1}\max\{\log k, \log^{(r)} N\}\Big) \leq O\Big(\sqrt{N}\Big(1+\frac{6}{k}\Big)^{2r}\max\{\log k, \log^{(r)} N\}\Big)$$

gates. To obtain the two claims of the corollary we can now either pick:

- $k = (c_1 \log^\star N)^2$, where $c_1 \in [1,2]$ ensures k is a power of 2. Observe that $(1+\frac{4}{c_1 \log^\star N})^{r+2} = 1 + o(1)$ for constant r. Since $\log^\star N \in o(\log^{(r)} N)$ for every constant r, it follows that $\max\{\log k, \log^{(r)} N\} = \log^{(r)} N$. Hence the query and gate complexities are $(\frac{\pi}{4} + o(1))\sqrt{N}$ and $O(\sqrt{N} \log^{(r)} N)$, respectively.

- $r = \log^\star N$ and $k = (c_2(\log^\star N + 2))^2$, where we choose c_2 as the smallest number that is at least $4/\ln(1+\varepsilon)$ and that makes k a power of 2. We have $(1 + \frac{4}{\sqrt{k}})^{r+2} \leq (1 + \frac{4}{c_2(\log^\star N+2)})^{\log^\star N+2} \leq 1 + \varepsilon$. Hence the query and gate complexities are $\frac{\pi}{4}\sqrt{N}(1+\varepsilon)$ and $O(\sqrt{N} \log(\log^\star N))$, respectively. $\square$

4 Future work

Our work could be improved further in a number of directions:

- Can we reduce the $\log(\log^\star N)$ factor in the gate complexity to the optimal $O(\sqrt{N})$? This may well be possible, but requires a different idea than our roughly $\log^\star$ recursion steps, which will inevitably end up with $\omega(\sqrt{N})$ gates.

- Our construction only works for specific values of N. Can we generalize it to work for all sufficiently large N, even those that are not powers of 2, while still using close to the optimal $\frac{\pi}{4}\sqrt{N}$ queries?

- Can we obtain a similar gate-optimized construction when the database has *multiple* solutions instead of one unique one? Say when the exact number of solutions is known in advance?

- Most applications of Grover's algorithm deal with databases with an unknown number of solutions, and focus only on the number of queries. Are there application where our reduction in the number of elementary gates for search with one unique solution is both applicable and significant?

Acknowledgements

We thank Peter Høyer and Andris Ambainis for helpful comments related to [11], and an anonymous referee for helpful comments on the presentation.

1. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.
2. G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. 2002. quant-ph/0005055.
3. Ch. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60:2746–2751, 1999. quant-ph/9711070.
4. G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology Column)*, 28:14–19, 1997. quant-ph/9705002.
5. H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998. quant-ph/9802040.
6. H. Buhrman, Ch. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *SIAM Journal on Computing*, 34(6):1324–1330, 2005. quant-ph/0007016.
7. C. Dürr and P. Høyer. A quantum algorithm for finding the minimum. quant-ph/9607014, 18 Jul 1996.

8. C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. In *Proceedings of 31st ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 481–493, 2004.
9. S. Dörn. *Quantum Complexity of Graph and Algebraic Problems*. PhD thesis, Institut für Theoretische Informatik, Universität Ulm, 2007.
10. L. K. Grover. Trade-offs in the quantum search algorithm. *Physical Review A*, 66(052314), 2002. quant-ph/0201152.
11. S. Aaronson and A. Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005. Earlier version in FOCS'03. quant-ph/0303041.

Appendix A

For the sake of completeness we present the construction of quantum algorithm $\mathcal{B}$ from Theorem 1. The idea is to lower the success probability from a in such a way that an integer number of rounds of amplitude amplification suffice to produce a solution with probability exactly a'.

Define $\theta = \frac{\arcsin(\sqrt{a'})}{2w+1}$ and $\tilde{a} = \sin^2(\theta)$, where w is defined in Theorem 1. Let $R_{\tilde{a}/a}$ be the one-qubit rotation that maps $|0\rangle \mapsto \sqrt{\tilde{a}/a}|0\rangle + \sqrt{1-\tilde{a}/a}|1\rangle$. Call an $(n+1)$-bit string (i,b) a "solution" if $x_i = 1$ *and* $b = 0$. Define the $(n+1)$-qubit unitary $O'_x = (I \otimes XH)O_x(I \otimes HX)$. It is easy to verify that O'_x puts a $-$ in front of the solutions (in the new sense of the word), and a $+$ in front of the non-solutions.

Let $\mathcal{A}' = \mathcal{A} \otimes R_{\tilde{a}/a}$, and define $|U\rangle = \mathcal{A}'|0^{n+1}\rangle$ to be the final state of this new algorithm. Let $|G\rangle$ be the normalized projection of $|U\rangle$ on the (new) solutions and $|B\rangle$ be the normalized projection of $|U\rangle$ on the (new) non-solutions. Measuring $|U\rangle$ results in a (new) solution with probability exactly $\sin^2(\theta)$, hence we can write

$$|U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

Define $\mathcal{Q} = \mathcal{A}'D_{n+1}(\mathcal{A}')^{-1}O'_x$. This is a product of two reflections in the plane spanned by $|G\rangle$ and $|B\rangle$: O'_x is a reflection through $|G\rangle$, and $\mathcal{A}'D_{n+1}(\mathcal{A}')^{-1} = 2|U\rangle\langle U| - I$ is a reflection through $|U\rangle$. As is well known in the analysis of Grover's algorithm and amplitude amplification, the product of these two reflections rotates the state through an angle 2θ. Hence after applying $\mathcal{Q}$ w times to $|U\rangle$ we have the state

$$\mathcal{Q}^w|U\rangle = \sin((2w+1)\theta)|G\rangle + \cos((2w+1)\theta)|B\rangle = \sqrt{a'}|G\rangle + \sqrt{1-a'}|B\rangle,$$

since $(2w+1)\theta = \arcsin(\sqrt{a'})$. Thus the algorithm $\mathcal{A}'$ can be boosted to success probability a' using an integer number of applications of $\mathcal{Q}$.

Our new algorithm $\mathcal{B}$ is now defined as $\mathcal{Q}^w\mathcal{A}'$. It acts on $n+1$ qubits (all initially 0) and maps

$$|0^{n+1}\rangle \mapsto \sqrt{a'}|G\rangle + \sqrt{1-a'}|B\rangle,$$

so it finds a solution with probability exactly a'. $\mathcal{B}$ uses $w+1$ applications of algorithm $\mathcal{A}$ together with elementary gate $R_{\tilde{a}/a}$; w applications of $\mathcal{A}^{-1}$ together with $R^{-1}_{\tilde{a}/a}$; w applications of O'_x (each of which involves one query to x and two other elementary gates, counting XH as one gate); and w applications of D_{n+1} (each of which takes $4n+3$ elementary gates). Hence the total number of queries that $\mathcal{B}$ makes is at most $(2w+1)Q + w$ and the number of gates used by $\mathcal{B}$ is at most $(2w+1)E + 4w(n+2)$.