COCOON'01 Submission

title: Deterministic application of Grover's quantum search algorithm

author: Kyoichi Okamoto and Osamu Watanabe

affiliation:

Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152-8552

email: watanabe@is.titech.ac.jp

Abstract:

Grover's quantum search algorithm finds one of t solutions in N candidates by using $(\pi/4)\sqrt{N/t}$ basic steps. It is, however, necessary to know the number t of solutions in advance for using the Grover's algorithm directly. On the other hand, Boyer etal proposed a randomized application of Grover's algorithm, which runs, on average, in $O(\sqrt{N/t})$ basic steps (more precisely, $(9/4)\sqrt{N/t}$ steps) without knowing t in advance. Here we show a simple (almost trivial) deterministic application of Grover's algorithm also works and finds a solution in $O(\sqrt{N/t})$ basic steps (more precisely, $(8\pi/3)\sqrt{N/t}$ steps) on average.

1. Introduction

Grover [Gro96, Gro97] proposed a quantum algorithm — Grover's search algorithm — that solves the following general search problem much faster than any randomized/deterministic algorithm designed on classical computers.

Search Problem

Given: For any $n > 0$, an oracle Boolean function f on the set $\{0,1\}^n$ of binary sequences of length n.

Question: Find some sequence $x \in \{0,1\}^n$ such that $f(x) = 1$.

(Remark. In general, a solution, i.e., a sequence $x \in \{0,1\}^n$ satisfying $f(x) = 1$, is not unique, in which case it is sufficient to output *any one* of such solutions.)

For any given n, since there are $N = 2^n$ binary strings in $\{0,1\}^n$, it is (almost obvious) that any algorithm solving the above problem needs N steps to find the desired sequence. Surprisingly, though, Grover's search algorithm finds the desired sequence in $O(\sqrt{N})$ quantum computation steps, where each quantum step (which we refer *G-steps*) can be implemented by some poly(n) number of basic quantum gates. More precisely, when there are t solutions, i.e., t binary sequences satisfying f, Grover's algorithm finds some of them in $O(\sqrt{N/t})$ G-steps. Note, however, that one needs to know the number t in advance in order to achieve this better bound; but t, the number of solutions, is usually unknown in advance. For solving this problem, Boyer etal [BBHT96] proposed an algorithm that applies Grover's algorithm with randomly chosen parameters, that is, a randomized application of Grover's algorithm, which runs, on average, in $O(\sqrt{N/t})$ G-steps without knowing t in advance. Here we show that a simple (in fact, almost trivial) deterministic application of Grover's algorithm also works and finds a solution in $O(\sqrt{N/t})$ G-steps on average without knowing t in advance.

Grover's Algorithm and Its Randomized Application

Let us discuss our problem specifically. We start with recalling Grover's algorithm and some basic facts about the algorithm. Notions and notations for quantum computation we use here are standard; see, e.g., [BV97, Gru99, Hos99, For00].

Consider any $n > 0$ and any oracle function f on $\{0,1\}^n$ for specifying the above general search problem, and let us fix them in the following discussion. That is, our task is to find a sequence $x \in \{0,1\}^n$ that satisfies $f(x) = 1$. A sequence $x \in \{0,1\}^n$ satisfying $f(x) = 1$ is simply called a *solution*. Let $N = 2^n$, i.e., the total number of sequences in $\{0,1\}^n$, and let t denote the number of all solutions among N candidates.

In Grover's algorithm, each sequence $x \in \{0,1\}^n$ corresponds to a quantum base state $|x\rangle$ consisting of n qubits. (In the following, we identify an n qubit base state with the corresponding n bit binary sequence.) The main ingredients of Grover's algorithm are the following three unitary transformations on n qubit states.

Walsh-Hadmard transformation:

$$W: \ |x\rangle \ \mapsto \ \frac{1}{\sqrt{N}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$$

(Here $x \cdot y$ denotes the bit-wise inner product.)

Sign flipping on f:

$$S_f: \ |x\rangle \ \mapsto \ (-1)^{f(x)}|x\rangle$$

Sign flipping on 0:

$$S_0: \ |0\rangle \ \mapsto \ -|0\rangle \ \text{ and } \ |x\rangle \ \mapsto \ |x\rangle \ (\text{if } x \neq 0^n)$$

By using these transformations, one G-step is defined as the following unitary transformation U.

$$U \ = \ -WS_0WS_f$$

That is, *one G-step* is to apply this U to a current state. Grover's algorithm is to apply U for some appropriate number of times to the following initial state ϕ_0.

$$|\psi_0\rangle \ = \ \sum_{a\in\{0,1\}^n} \frac{1}{\sqrt{N}}|a\rangle.$$

Formally, by *the j G-step execution of Grover's algorithm* (or more simply $G(j)$) we mean to apply $\underbrace{UUU\cdots U}_{j}$ to ϕ_0 and then observe the obtained state. (We assume that the observation is made so that some n qubit base state (i.e., n bit sequence) is observed with the probability that is the square of its amplitude.)

For justifying this procedure, the following property of U plays a key role [Gro96, Gro97].

Lemma 1. Consider the quantum state obtained by applying U to ϕ_0 for j times. Then each solution state, i.e., a base state corresponding to a solution, has the same amplitude (which we denote a_j) while the other base state also has the same amplitude (which we denote b_j). Furthermore, by using θ that satisfies $\sin\theta = \sqrt{t/N}$, these amplitudes are stated as follows.

$$\begin{aligned} a_j &= \frac{1}{\sqrt{t}}\sin((2j+1)\theta), \\ b_j &= \frac{1}{\sqrt{N-t}}\cos((2j+1)\theta). \end{aligned}$$

The angle $(2j+1)\theta$ determines the amplitude a_j. In the following we call this angle *the angle* (*of a solution*) *after applying U for j times* (*to the initial state*) (or more simply, *the angle after executing $G(j)$*). Note that after applying U for $\lfloor(\pi/4)\theta\rfloor$ times, the angle gets close to $\pi/2$; hence, the amplitude of each solution is close to $1/\sqrt{t}$, which means that the total probability that solution states are observed is close to 1. Note also that $(\pi/4)\theta$ is approximately $(\pi/4)\sqrt{N/t}$ by using the approximation $\theta \approx \sin\theta$ $(= \sqrt{t/N})$. This argument leads us to the following theorem of Grover [Gro96, Gro97].

Theorem 2. Define m_0 by

$$m_0 \ = \ \left\lfloor \frac{\pi}{4\theta} \right\rfloor \ \left(\approx \ \frac{\pi}{4}\sqrt{\frac{N}{t}} \right).$$

Then a state observed by the execution of $G(m_0)$ is one of the solutions with probability approximately $1 - 1/N$ (≈ 1).

Therefore, if we can compute m_0, we would simply execute $G(m_0)$ to get some solution. It is, however, not so easy to compute it because t is usually unknown in advance. Note that we cannot simply execute $G(m)$ with some $m > m_0$; in this case, the probability that some solution is observed could become much smaller. In order to solve this problem, Boyer etal [BBHT96] proposed the following randomized algorithm.

Algorithm Randomized-Grover
$\lambda \leftarrow 6/5$; $i \leftarrow 0$;
for $i \leftarrow 1$ **to** ∞ **do**
$m \leftarrow \lambda^i$;
select j, $0 \leq j \leq m-1$, uniformly at random;
execute $G(j)$ and let x be the observed state;
if $f(x) = 1$ **then** output x and halt;
end-for;

Figure 1: A randomized application of Grover's algorithm

It is clear that this algorithm finds a solution. On the other hand, we can show the following time bound.

Theorem 3. The average number of G-steps executed in the above algorithm is at most $(9/4)\sqrt{N/t}$.

Remark. In the proof of this theorem and in the following discussion in general, we often assume that t/N is small enough, say, $t/N < 1/100$ and use some approximations like $\sqrt{t/N} \approx \sin\sqrt{t/N}$ under this assumption. Thus, our bounds may not be precise; but even so, it is easy to see that the bounds shown here certainly hold by changing constant factors. We leave the tedious precise calculation to the interest reader. Also though our arguments need an assumption such as $t < N/100$, this assumption is not essential for designing a fast search algorithm. In fact, if $t \geq N/100$, then we can search for a solution by simply picking any n bit sequence randomly, say, 1,000 times.

2. Deterministic Application of Grover's Algorithm

Here we show that a simple and *deterministic* execution of Grover's algorithm still yields a similar result. More specifically, we consider the algorithm given in Figure 2.

Again for this algorithm, we can show that its average running time is $O(\sqrt{N/t})$. In the following we will prove this fact after preparing some tools.

First let us see that at some point of the for-iteration of the algorithm, the angle after executing $G(m)$ in this iteration gets reasonably close to $\pi/2$.

Claim 1. There exists some integer $i_0 > 0$ such that

$$\frac{\pi}{3} \;\leq\; (2 \cdot 2^{i_0} + 1)\theta \;\leq\; \frac{2\pi}{3}.$$

Furthermore, since $\theta \approx \sqrt{t/N}$, we have $2^{i_0} < (\pi/3)\sqrt{N/t}$.

Algorithm Deterministic-Grover

% The parameter k is fixed to 2.
for $i \leftarrow 1$ **to** ∞ **do**
  $m \leftarrow 2^i$;
  execute $G(m)$ for k times and let x_1, ..., x_k be the observed states;
  if $f(x_u) = 1$ for some u, $1 \leq u \leq k$ **then** output x_u and halt;
end-for;

Figure 2: A determinisitic application of Grover's algorithm

Proof. Clearly, $(2 \cdot 2^i + 1)\theta$ $(\approx (2 \cdot 2^i + 1)\sqrt{t/N}) > \pi/3$ for some $i > 0$. Let i be the smallest integer such that $(2 \cdot 2^i + 1)\theta < \pi/3$ holds. Then we have

$$(2 \cdot 2^{i+1} + 1)\theta \;<\; 2(2 \cdot 2^i + 1)\theta \;<\; \frac{2\pi}{3}.$$

Therefore, the claim holds with $i_0 = i + 1$. □

Below we will keep using i_0 to denote the one satisfying this claim. Also let W denote the above range of angles; that is, $W = [\pi/3, 2\pi/3]$. As stated in the above claim, the angle at the i_0th for-iteration gets into W; then the probability that a solution is found at this for-iteration is, as we will see below, more than $1 - (1/4)^k$. In general, for any i, let us simply call the angle $(2 \cdot 2^i + 1)\theta$ *the angle at the ith for-iteration.* Recall that the success probability is defined by $\sin^2$ of angles; hence, we had better discuss angles with their residues under modulo π. Thus, in the following, by "an angle" we mean the residue of a real angle under modulo π. For any for-iteration, we have the following relation between the angle and the failure probability at this for-iteration.

Claim 2. For any $i > 0$, let ω be the angle at the ith for-iteration. Let $\alpha = |\pi/2 - \omega|$ and $\delta = \sin^2 \alpha$. Then the probability that no solution is found at this for-iteration is δ^k. (Note that if $\omega \in W$, then we have $\delta \leq 1/4$.)

Proof. Since the angle at the ith for-iteration is ω, the probability that some solution is found by one execution of $G(m)$ (where $m = 2^i$) is

$$t \cdot \left(\frac{1}{\sqrt{t}} \sin \omega \right)^2 \;=\; \sin^2 \omega.$$

Hence, the probability that no solution is found at this iteration is

$$(1 - \sin^2 \omega)^k \;=\; (\cos^2 \omega)^k \;=\; \left(\sin^2 |\pi/2 - \omega| \right)^k \;=\; (\sin^2 \alpha)^k \;=\; \delta^k. \qquad \square$$

It would be nice if we could argue that the angle belongs to W reasonably often. Unfortunately, however, if the angle (at some for-iteration) gets very close to $\pi/2$, then it takes rather long time to have an angle in W again. Note, on the other hand, that if the angle is close to $\pi/2$, then the probability of finding a solution at this iteration is close to 1. The crucial point of our analysis is to estimate this trade-off.

For simplifying our discussion, instead of an angle ω, we will argue by using $\alpha = |\pi/2 - \omega|$, which we call a *co-angle*. For example, the above claim shows that if the co-angle (at some for-iteration) gets less than $\pi/6$, then the error probability at this iteration becomes less than $(1/4)^k$. Let A denote this good co-angle range; that is, $A = \{\alpha : 0 \le \alpha \le \pi/6\}$.

For our analysis, we estimate the number of for-iterations until having a co-angle in A again.

Claim 3. Consider any ith for-iteration with the co-angle $\alpha \in A$. (We assume that $\alpha > 0$.) Define $h = \lceil \pi/(3\alpha) \rceil$. Then there exists some $\overline{h}$, $1 \le \overline{h} \le h$, such that the co-angle at the $(i+\overline{h})$th for-iteration gets into A again.

Remark. For simplifying our discussion. the argument here is a bit unprecise. More specifically, it may be the case that the co-angle at the $(i+\overline{h})$th for-iteration is slightly larger than $\pi/3$; but the difference is at most $+3\theta$, which should be small enough under our assumption on t/N.

Proof. Here we show that the angle at the $(i+\overline{h})$th for-iteration belongs to the range $W = [\pi/3, 2\pi/3]$.

Since the claim clearly holds if $\alpha = \pi/6$, we may assume that $\alpha < \pi/6$, which implies that $h \ge 2$. Recall that the definitions of angle and co-angle; that is, we have $(\omega =)\ \pi/2 - \alpha = (2 \cdot 2^i + 1)\theta$, where ω is the angle at the ith for-iteration. (Here we consider only the case that $\omega < \pi/2$. The case that $\omega > \pi/2$ can be treated in a similar way.) Then for any h', $2 \le h' \le h$, we can estimate the angle ω' at the $(i+h')$th for-iteration as follows.

$$
\begin{aligned}
\omega' &= (2 \cdot 2^{i+h'} + 1)\theta \pmod{\pi} \\
&= 2^{h'} \cdot ((2 \cdot 2^i + 1)\theta) - (2^{h'} - 1)\theta \pmod{\pi} \\
&= 2^{h'} \cdot \left(\frac{\pi}{2} - \alpha\right) - (2^{h'} - 1)\theta \pmod{\pi} \\
&= -(2^{h'}\alpha + (2^{h'} - 1)\theta - m\pi) \pmod{\pi}.
\end{aligned}
$$

Here m is some nonnegative integer such that $0 \le 2^{h'}\alpha + (2^{h'} - 1)\theta - m\pi < \pi$. Since $\sin\omega' = \sin(\pi - \omega')$, we can discuss by using $\overline{\omega}' = 2^{h'}\alpha + (2^{h'} - 1)\theta - m\pi$ instead of the above ω'.

Now we consider the first $h' \le h$ such that $2^{h'}\alpha + (2^{h'} - 1)\theta \ge \pi/3$. Such h' exists because we have $2^h\alpha \ge \pi/3$ from our choice of h. Furthermore, we may assume that $h' \ge 2$, because in the special case that $h' = 1$, we can easily show that $\overline{h} = 2$ satisfies the claim. Also notice that $\overline{\omega}' = 2^{h'}\alpha + (2^{h'} - 1)\theta$; that is, $m = 0$ for this h'.

Clearly, we are done if $\overline{\omega}' \le 2\pi/3$. But if otherwise, i.e., if $2^{h'}\alpha + (2^{h'} - 1)\theta > 2\pi/3$, then the angle $\overline{\omega}'' = 2^{h'-1}\alpha + (2^{h'-1} - 1)\theta$ ($\approx (2^{h'}\alpha + (2^{h'} - 1)\theta)/2$) should be $\ge \pi/3$. On the other hand, since $2^{h'}\alpha + (2^{h'} - 1)\theta < \pi$, we have $\overline{\omega}'' = 2^{h'-1}\alpha + (2^{h'-1} - 1)\theta < \pi/2$ ($< 2\pi/3$). Therefore, the claim is satisfied with either $\overline{h} = h'$ or $\overline{h} = h' - 1$. $\square$

Next we divide the co-angle range A. Below let j be any nonnegative integer. Let $\delta_j = 2^{-(j+2)}$, and let $\widehat{\alpha}_j$ be the co-angle $< \pi/2$ such that $\delta_j = \sin^2 \widehat{\alpha}_j$. (For example, we have $\delta_0 = 1/4$ and $\widehat{\alpha}_0 = \pi/6$.) Then define $A_j = \{\alpha : \widehat{\alpha}_{j+1} < \alpha \le \widehat{\alpha}_j\}$. Also we define h_j as follows.

$$
h_j = \left\lceil \log \frac{\pi}{3\widehat{\alpha}_{j+1}} \right\rceil.
$$

Since $\widehat{\alpha}_j$ gets very small, we may assume that $\sin\widehat{\alpha}_j \approx \widehat{\alpha}_j$. In particular, we may assume that $\widehat{\alpha}_{j+1} \geq \widehat{\alpha}_j/2$. (In fact, we would have $\widehat{\alpha}_{j+1} \geq \widehat{\alpha}_j/\sqrt{2}$ for large j.) Hence, we have $h_{j+1} \leq h_j + 1$.

Here let us summarize our discussion. Consider any ith for-iteration, and suppose that the co-angle at this iteration belongs to A_j. Then it follows from the above claims that (i) the probability that the algorithm does not halt at this for-iteration is $\leq \delta_j^k$ $(= 2^{-(j+2)k})$, and (ii) the co-angle gets into A again before the $(i + h_j)$th for-iteration.

Now we are ready to prove our theorem.

Theorem 4. The average number of G-steps executed in Deterministic Grover (the algorithm of Figure 2) is $(8\pi/3)\sqrt{N/t}$.

Proof. Recall that the co-angle belongs to A at the i_0th for-iteration. We may assume that it is indeed the first for-iteration having a co-angle in A. In general, we use i_1, i_2, ..., to denote the first, the second, ... for-iteration (after the i_0th for-iteration) where the co-angle belongs to A. Also let α_0, α_1, ... denote the co-angles at the i_0th, i_1th, ... iterations.

To bound the expected number of executed G-steps, we may assume that the execution of the algorithm reaches to the i_0th for-iteration. Thus, we first estimate the number of G-steps executed by the end of the i_0th for-iteration, which is bounded as follows.

$$k \cdot (1 + 2 + 2^2 + \cdots + 2^{i_0}) \;=\; k \cdot (2^{i_0+1} - 1) \;<\; \frac{2k\pi}{3}\sqrt{\frac{N}{t}}. \tag{1}$$

Next we analyze the expected number E_0 of G-steps executed after the i_0th for-iteration. (Precisely speaking, E_0 that we will analyze below is the expected number E_0 of G-steps executed after the i_0th for-iteration under the condition that the i_0th for-iteration is executed.) Although we know that the co-angle α_0 at the i_0th for-iteration is in A, we do not know which A_j it belongs to. But for each $j \geq 0$, by assuming that $\alpha \in A_j$, we can estimate the failure probability and the length to the i_1th for-iteration. Thus, we can bound E_0 in the following way. (Here we use E_1 to denote the amount corresponding to E_0; that is, E_1 is the expected number of G-steps executed after the i_1th for-iteration under the condition that the i_1th for-iteration is executed.)

$$\begin{aligned}
E_0 \;&\leq\; \sum_{j\geq 0} \Pr\{\,\text{no solution is found at the } i_0\text{th for-iteration} \mid \alpha_0 \in A_j\,\} \\
&\qquad\quad \times (\#\text{ of G-steps executed between the } (i_0+1)\text{th and } i_1\text{th for-iteration} + E_1) \\
&\leq\; \sum_{j\geq 0} \delta_j^k \cdot \Big(k \cdot (2^{i_0+1} + 2^{i_0+2} + \cdots + 2^{i_0+h_j}) + E_1\Big) \\
&=\; \sum_{j\geq 0} \delta_j^k \cdot k \cdot \Big(2^{i_0+h_j+1} - 1\Big) + \sum_{j\geq 0} \delta_j^k \cdot E_1 \;\leq\; k2^{i_0} \cdot \left(\sum_{j\geq 0} \delta_j^k \cdot 2^{h_j+1}\right) + \sum_{j\geq 0} \delta_j^k \cdot E_1.
\end{aligned}$$

Here by using the fact that $h_j \leq h_0 + j$ and $h_0 \leq 2$, and our choice of δ_j and k (i.e., $\delta_j = 2^{-j+2}$ and $k = 2$), we can bound the last expression as follows.

$$E_0 \;\leq\; k2^{i_0} \cdot \left(\sum_{j\geq 0} 2^{-(j+2)k} \cdot 2^{h_0+j+1}\right) + \sum_{j\geq 0} \delta_j^k \cdot E_1$$

$$= \quad k2^{i_0}\sum_{j\geq 0}2^{-j-1} + \sum_{j\geq 0}\delta_j^k \cdot E_1 \ = \ k2^{i_0} + \sum_{j\geq 0}\delta_j^k\cdot E_1. \tag{2}$$

At this point, let us see how E_1 is estimated. Notice that E_1 depends on the value of i_1. Otherwise, the calculation is quite similar to the one for E_0. (Here we use E_2 denoting the amount corresponding to E_1 for i_2.)

$$\begin{aligned}
E_1 &\leq \sum_{j\geq 0}\Pr\{\text{ no solution is found at the } i_1\text{th for-iteration} \mid \alpha_1 \in A_j\ \} \\
&\qquad \times (\#\text{ of G-steps executed between the } (i_1+1)\text{th and } i_2\text{th for-iteration} + E_2) \\
&\leq \sum_{j\geq 0}\delta_j^k\cdot\left(k\cdot(2^{i_1+1}+2^{i_1+2}+\cdots+2^{i_1+h_j})+E_2\right) \\
&\leq k2^{i_1}\cdot\left(\sum_{j\geq 0}\delta_j^k\cdot 2^{h_j+1}\right)+\sum_{j\geq 0}\delta_j^k\cdot E_2 \ \leq \ k2^{i_1}+\sum_{j\geq 0}\delta_j^k\cdot E_2.
\end{aligned} \tag{3}$$

We substitute E_1 of (2) by (3). Here notice that i_1 depends on the choice of α_0; in fact, if $\alpha_0 \in A_j$, then it is bounded by $i_0 + h_j$ ($\leq i_0 + h_0 + j \leq i_0 + j + 2$). Thus, we have the following bound.

$$\begin{aligned}
E_0 &\leq k2^{i_0}+\sum_{j_1\geq 0}\delta_{j_1}^k\cdot\left(k2^{i_1}+\sum_{j_2\geq 0}\delta_{j_2}^k\cdot E_2\right) \\
&\leq k2^{i_0}+\sum_{j_1\geq 0}2^{-2(j_1+2)}\cdot k2^{i_0+j_1+2}+\sum_{j_1\geq 0}\delta_{j_1}^k\cdot\left(\sum_{j_2\geq 0}\delta_{j_2}^k\cdot E_2\right) \\
&= k2^{i_0}+k2^{i_0}\sum_{j_1\geq 0}2^{-(j_1+2)}+\sum_{j_1\geq 0}\delta_{j_1}^k\cdot\left(\sum_{j_2\geq 0}\delta_{j_2}^k\cdot E_2\right) \\
&= k2^{i_0}+\frac{1}{2}\cdot k2^{i_0}+\sum_{j_1\geq 0}\delta_{j_1}^k\cdot\left(\sum_{j_2\geq 0}\delta_{j_2}^k\cdot E_2\right).
\end{aligned}$$

We can similarly expand E_2, E_3, ..., thereby deriving the following bound.

$$E_0 \ \leq \ k2^{i_0}+2^{-1}\cdot k2^{i_0}+2^{-2}\cdot k2^{i_0}+\cdots \ = \ 2k2^{i_0} \ \leq \ \frac{2k\pi}{3}\sqrt{\frac{N}{t}}. \tag{4}$$

Now the bound of the theorem is immediate from (1) and (4), and our choice of k. □

References

[BV97] E. Bernstein and U. Vazirani, Quantum complexity theory, *SIAM Journal on Computing* 26, 1411–1473 (1997).

[BBHT96] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp, Tight bounds on quantum searching, quant-ph/9605034.

[For00] L. Fortnow, One complexity theorist's view of quantum computing, quant-ph/0003035.

[Gro96] L.K. Grover, A fastquantum mechanical algorithm for database search, in *Proc. 28th Annual ACM Symposium on Theory of Computing*, ACM, 212–219 (1996).

[Gro97] L.K. Grover, Quantum mechanics helps in searching for a needle in a haystack, *Phys.Rev.Lett* 79, 325–328 (1997).

[Gru99] J. Gruska, *Quantum Computing*, McGrow-Hill, 1999.

[Hos99] A. Hosoya, *Lectures on Quantum Computation* (in Japanese), Science Pub. Co., 1999.