アルゴリズム 73－8
（2000．5．19）

Shor のアルゴリズム専用シミュレーションシステム

山崎智弘, 今井浩

東京大学大学院理学系研究科情報科学専攻
113-0033 東京都文京区本郷 7-3-1
{yamasaki, imai}@is.s.u-tokyo.ac.jp

要旨 多数の量子ビットを持つ量子コンピュータが存在しない現状では、量子計算を模倣するシミュレータをつくって実際にアルゴリズムの振る舞いを検証することの意義は大きい。しかし従来のシミュレータは汎用的に作られているため、計算時間だけでなく必要なメモリ量も指数関数的に増加してしまうという欠点があった。そこで本論文では Shor のアルゴリズム専用のシミュレータを構築した。その結果計算の途中での確率振幅を保持しておく必要がなくなったため、必要なメモリ量を少なく抑えることに成功した。

Shor's algorithm simulation system

Tomohiro Yamasaki, and Hiroshi Imai

Department of Information Science, Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113–0033, Japan
{yamasaki, imai}@is.s.u-tokyo.ac.jp

Abstract No quantum computers with a lot of quantum bits have not appeared yet. Therefore, it is significant to implement a system which simulates quantum computation and verifies the behavior of algorithms in practice. Traditional simulators, however, have a weak point such that not only computation time but also necessary memory size increases exponentially, since they are made for general purpose. In this paper, we implement a simulator specialized for Shor's algorithm. As a result, small memory size are needed because this simulator does not need to hold all amplitudes on computation in progress.

1 はじめに

現在量子コンピュータの実現の研究はさかんに行われているが、それはいくつかの有用性が理論的に示されたからである。1994 年、P.Shor は量子計算を用いれば自然数の素因数分解を多項式時間で行えることを示し [1]、注目を集めた。しかし多数の量子ビットを持つ量子コンピュータが存在しない現状では、量子計算を模倣するシミュレータをつくって実際にアルゴリズムの振る舞いや計算量を検証することの意義は大きい。しかし従来のシミュレータは汎用的に作られているため、量子計算のモデルである量子回路のすべての基底状態に対する確率振幅を保存する必要があった。そのために計算時間だけでなく必要なメモリ量も指数関数的に増加してしまうという欠点があり、大規模な実験を行うことができない。そこで本論文では、Shor のアルゴリズム専用のシミュレータを構築した。その結果各ラベルに対する確率振幅を逐次的に計算できるようになったため計算の途中で保持しておく必要がなくなったので、必要なメモリ量を少なく抑えることに成功した。さらには実験から得た結果に基づいて Shor のアルゴリズムの振る舞いや性質について考察を加える。

2 準備

2.1 Walsh–Hadamard (W–H) 変換

量子計算の利点の一つは、重ねあわせの原理を用いて指数個の計算を並列的に計算できることである。ところが重ねあわせを効率よく作る方法が存在しないとすると、この利点を生かすことができない。しかしながら、Walsh–Hadamard 変換という変換によって効率よく重ね合わせ状態をつくることができることが知られている。

定義 1 Walsh–Hadamard (W–H) 変換とは、2 状態系のビットに対して行う以下のような変換である。

$$W : |x\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{y=0,1} (-1)^{xy} |y\rangle, \quad x \in \{0,1\}$$

これは以下のような 2×2 ユニタリ行列

$$W = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

で表される変換を量子ビットに対して行うことにほかならない。したがって一般の n ビットに対する W–H 変換は、$\boldsymbol{x} = x_0 x_1 \cdots x_{n-1}$、$\boldsymbol{y} = y_0 y_1 \cdots y_{n-1}$、$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=0}^{n-1} x_i y_i$ であるとき以下のように表せる。

$$\begin{aligned} W : |\boldsymbol{x}\rangle &\mapsto \bigotimes_{i=0}^{n-1} \frac{1}{\sqrt{2}} \sum_{y_i} (-1)^{x_i y_i} |y_i\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{y}=0}^{2^n-1} (-1)^{\boldsymbol{x} \cdot \boldsymbol{y}} |\boldsymbol{y}\rangle \end{aligned}$$

2.2 離散フーリエ変換 (DFT)

離散フーリエ変換は、量子計算のもう一つの特徴である干渉効果を効率よく検出できるように状態を変化させるための前処理である。なんらかの周期を発見するような計算においては、この変換によって欲しい解を観測する確率が高くなるようにすることができる。

定義 2 離散フーリエ変換とは n ビットの 2 状態系に作用する以下のような変換である。ただし $N = 2^n$ である。

$$DFT : |\boldsymbol{x}\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\boldsymbol{y}=0}^{N-1} e^{2\pi i \boldsymbol{x}\boldsymbol{y}/N} |\boldsymbol{y}\rangle$$

2.3 整数論

2.3.1 連分数近似

任意の正の実数 x は正整数の数列 $\{a_n\}$ を用いて以下のように表すことができる。

$$x = a_0 + \frac{1}{a_1 + \dfrac{1}{a_2 + \cdots + \dfrac{1}{a_n \cdots}}}$$

これを x の連分数表現という。また途中の有限項までで打ち切ったものを x の連分数近似という。連分数に関しては以下のような定理が知られている。

定理 1 $|x - p/q| < 1/2q^2$ ならば p/q は x の連分数近似である。

2.3.2 Euler 関数

定義 3 n 以下で n と互いに素な数の個数を Euler 関数といい、$\phi(n)$ で表す。

Euler 関数の値に関して、以下のような定理が知られている。

定理 2

$$\inf_{n\to\infty} \frac{\phi(n)}{n/\log\log n} = e^{\gamma}$$

3 Shor のアルゴリズム

P.Shor [1] による量子計算を用いた素因数分解のアルゴリズムは、以下の通りである。このアルゴリズムは誤った解を返すことがないので、量子計算によるラスベガスアルゴリズムである。

3.1 素因数分解のアルゴリズム

1. n が素数かどうかを適当なアルゴリズムを用いて判定する。素数と判定されたら、n を出力して終了。
2. n が素数の自然数乗かどうかを適当なアルゴリズムを用いて判定する。素数の自然数乗と判定されたら、n とベキを出力して終了。
3. $n^2 \le q < 2n^2$ を満たす 2 のべき乗 q をとる。
4. ランダムに自然数 x $(1 < x < n)$ を選び、Euclid の互除法によって $g = \gcd(n, x)$ を計算する。$1 < g < n$ ならば g を出力し、n を n/g に置き換えて 1 に戻る。
5. $\log q$ ビットで状態数 q の量子コンピュータを生成し、初期状態を $|0\rangle$ とする。次に Walsh–Adamard 変換を行って全ての状態が等しい重みで重ね合わさった状態にする。

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle \tag{1}$$

6. $x^a \pmod n$ を計算して第 2 レジスタに蓄える。

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |x^a \pmod n\rangle \tag{2}$$

7. 第 2 レジスタを観測する。観測により $x^{a_0} = k$ $(0 \le a_0 < r)$ が得られたとすると、状態の収縮により第 1 レジスタが $x^a \equiv k \pmod n$ となる a 以外の振幅はすべて 0 になる。$A = \lceil (q - a_0)/r \rceil$ とするとそれらの振幅は $1/\sqrt{A}$ となり、位数 r を周期として等間隔に並ぶ。

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |a_0 + jr\rangle |k\rangle \tag{3}$$

8. 第 1 レジスタに対して離散フーリエ変換をかける。

$$\frac{1}{\sqrt{qA}} \sum_{c=0}^{q-1} \sum_{j=0}^{A-1} e^{2\pi i (a_0 + jr)c/q} |c\rangle |k\rangle \tag{4}$$

9. 第1レジスタを観測する。観測値が c であったとき、c/q を連分数展開により分母が n を越えないあいだ展開し、得られた最大の分母を x の位数 r の候補とする。

10. r が偶数でなければ4に戻る。

11. Euclid の互除法により $g = \gcd(x^{r/2} - 1, n)$ を計算する。$1 < g < n$ ならば g を出力し、n を n/g に置き換えて1に戻る。そうでなければそのまま4に戻る。

3.2 シミュレータの作成

命題 1 Shor のアルゴリズムの5–7段階目においてどのような k が得られたとしても、r が同じであれば8段階目の離散フーリエ変換を施したあとの確率分布は a_0 や k をあらわには含まない。また、ラベル c を得る確率は、A と r が与えられていればほかのラベル c' を得る確率とは独立に計算することができる。

証明: 8段階目の離散フーリエ変換を施したあとの系の状態は

$$
\begin{aligned}
&= \frac{1}{\sqrt{qA}} \sum_{c=0}^{q-1} \sum_{j=0}^{A-1} e^{2\pi i(a_0 + jr)c/q} |c\rangle |k\rangle \\
&= \frac{1}{\sqrt{qA}} \sum_{c=0}^{q-1} e^{2\pi i a_0 c/q} \sum_{j=0}^{A-1} e^{2\pi i jrc/q} |c\rangle |k\rangle
\end{aligned}
$$

ここで観測を行ってラベル c を得る確率は

$$
\begin{aligned}
\mathrm{Prob}(c) &= \frac{1}{qA} \left| e^{2\pi i a_0 c/q} \sum_{j=0}^{A-1} e^{2\pi i jrc/q} \right|^2 \\
&= \frac{1}{qA} \left| \sum_{j=0}^{A-1} e^{2\pi i jrc/q} \right|^2 \\
&= \frac{1}{qA} \left| \frac{e^{2\pi i Arc/q} - 1}{e^{2\pi i rc/q} - 1} \right|^2 \\
&= \frac{1}{qA} \frac{\sin^2(\pi Arc/q)}{\sin^2(\pi rc/q)}
\end{aligned}
$$

すなわち a_0 や k をあらわには含んでおらず、また c だけの閉じた式で表されているためほかのラベル c' を得る確率とは独立に計算できることが証明された。 □

Shor のアルゴリズムにおいて量子計算を必要とするのは5–9段階目である。量子コンピュータの計算過程とは、$n$qubit あるとき 2^n 個の基底状態の振幅が移り変わっていく様子であるので、本来ならばここの計算において 2^n 次元の配列ベクトルを用意してそれぞれの振幅を保存していかなければならない。実際、徳永ほかによる量子計算機シミュレーションシステム [2] においてはそのようなシミュレーションを行っている。しかし上の命題で見たように Shor のアルゴリズムに限定した場合は、A と r が与えられれば最終的にそれぞれのラベルを観測する確率がいくらかになるかは直接求めることができる。そのため今回は r は通常の計算で求め、Shor のアルゴリズムを用いるとどの程度の確率で素因数分解が成功するかを見るために8–9のシミュレートを行うことにした。

各確率振幅が逐次的に求められるため配列に保存しないで済むので、時間計算量としては指数時間かかるものの空間計算量は多項式サイズでシミュレーションを行うことができる。

4 シミュレーション

4.1 理論的評価

1. 式4において観測を行うと干渉効果により $4/\pi^2$ 以上の確率で

$$-\frac{r}{2} \leq rc \pmod{n} \leq \frac{r}{2} \tag{5}$$

となる c を観測する。

2. 式5を同値変形すると

$$\exists d\ (0 \leq d \leq r-1)\ s.t.\ \left|\frac{c}{q} - \frac{d}{r}\right| \leq \frac{1}{2q} \tag{6}$$

という式が得られる。$r^2 < n^2 \leq q$ という条件は c/q を d/r で近似して良い条件となっているため、$\gcd(d,r) = 1$ ならば正しい r が得られることがわかる。ここで d と r が互いに素になる確率は

$$\inf_{d\to\infty} \frac{\phi(d)}{d} = \frac{e^{-\gamma}}{\log\log d} > \frac{e^{-\gamma}}{\log\log n} \tag{7}$$

なので、$e^{-\gamma}/\log\log n$ 以上の確率で $\gcd(d,r) = 1$ となることが保証される。

3. r が偶数のとき $\gcd(x^{r/2}-1, n)$ が $1, n$ 以外の素因数を与える確率は $1/2$ 以上である。

したがって、素因数分解の成功確率の理論的な保証値は以下のようになる。

$$\frac{4}{\pi^2} \times \frac{e^{-\gamma}}{\log\log n} \times \frac{1}{2} = O\left(\frac{1}{\log\log n}\right)$$

4.2 理論的評価の改良

1. 式5を満たす c は1つしか存在しないが、シミュレータによる実験の結果 $c \pm 1, c \pm 2$ でも r が得られることがあることがわかった。これは式6の $1/2q$ という評価がきつすぎるためであると思われる。そのため $\frac{4}{\pi^2}$ は0.9くらいに改良できそうである。

2. 正しくない位数が得られた場合でも、それを記録しておいて次に得られた位数との最小公倍数をとることによって正しい位数が得られる可能性がある。なぜなら、正しくない位数というのは正しい位数の約数であることが大半であるからである。このような改良により繰り返し回数を $O(\log\log n)$ よりも下げられる可能性がある。

4.3 実際的評価

4.3.1 試行回数

100程度までの2つの素数をかけあわせた数それぞれに対してShorのアルゴリズムを1000回ずつ適用し、素因数が得られるまでの試行回数の平均値を求めた。OSはLinux、メモリは64MB、CPU周波数は233MHzのマシン上でのシミュレートである。

表における"theory"、"average0"、"average1"はそれぞれ、「一般的な場合の理論値」、「Shorのアルゴリズムをそのまま実装したものによる平均値」、「正しくない位数が得られた場合に次に得

た位数との最小公倍数をとるようにShor のアルゴリズムを改良して実装したものによる平均値」を表している。

図を見ると理論値はかなり緩いことがわかった。アルゴリズムを改良した結果、反復回数が減少する効果があったこともわかった。しかし今回は2素数の積のみをシミュレートしており $O(\log \log n)$ よりも低いオーダでおさえられるかどうかまではわからなかった。

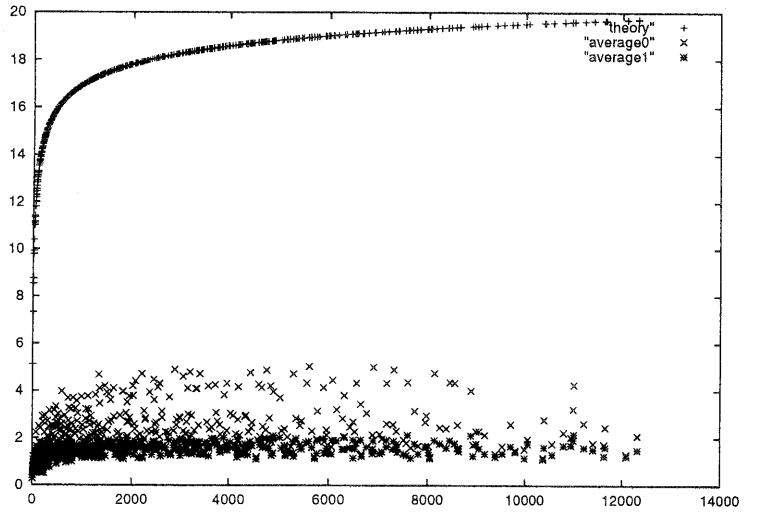

図 1: 理論値と実測による平均値 (横軸が素因数分解する数、縦軸が素因数分解が成功するまでの繰り返し回数)

5 まとめ

Shor のアルゴリズム専用のシミュレータを実装した結果、以前の汎用的なシミュレータでは実行できなかったような大規模な実験を行えるようになった。アルゴリズムの実際の振る舞いを確かめられるようになった意義は大きい。実験結果により、もし量子コンピュータが実現できた場合は理論よりも効率よく素因数分解が行えることがわかる。しかしながら必要なメモリ量は少なくてすむようになったものの、いまだに計算時間は指数関数的に増加してしまうという欠点がある。そのためいかに計算時間を抑えるかが今後の課題である。

参考文献

[1] P. Shor. Algorithms for quantum computation: Discrete log and factoring. In *Proceedings of the 35th Annual Symposium on Foundation of Computer Science*, pp. 56–65, 1994.

[2] Yuuki Tokunaga, Ayumu Nagai and Hiroshi Imai. Quantum computer simulation system. 京都大学数理解析研究所講究録, 1999 Oct 27–29.