Der Shor Algorithmus

Faktorisierungsverfahren in polynomialer Laufzeit mithilfe eines Quantencomputers

Lukas Lentner

24. November 2009

Inhaltsverzeichnis

1 Einführung

Das Faktorisieren von ganze Zahlen ist eine Aufgabenstellung aus der Zahlentheorie und besitzt eine langen Geschichte.

Neben Euklid haben sich schon viele berühmte Mathematiker wie Fermat mit der Suche nach einem effizienteren Verfahren als der Probedivision beschäftigt. Seit den 80er Jahren ist das Quadratische Sieb von Carl Pomerance (1981) einer der besten Algorithmen (exponentielle Laufzeit).

Da die Primfaktorzerlegung bei großen Zahlen immer noch sehr langsam ist, basieren viele Systeme auf diesem Problem.

Beispiel: Das Public-Key Kryptosystem RSA, welches zur Internetverschlüsselung eingesetzt wird, vertraut auf die Schwierigkeit ein Produkt zweier großen Primzahlen zu faktorisieren.

Der Shor Algorithmus stellt ein neues, schnelleres Verfahren da, welches auf der Quanteninformationsverarbeitung aufbaut.

Er wurde 1994 von Peter Shor in seiner Arbeit »Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer« veröffentlicht und ist bislang das einzige Faktorisierungsverfahren mit einer polynominalen Laufzeit. Allerdings ist die Nutzung noch stark von der verfügbaren Hardware abhängig (2001 faktorisierte eine Forschungsgruppe von IBM die Zahl 15. Hierfür benötigt man einen Quantencomputer mit 7 Qubits).

2 Vorangehende Erläuterungen

2.1 Diskrete Quanten-Fouriertransformation

Allgemein ordnet die diskrete Fouriertransformation einem Vektor x_j mit $0 \leq j \leq N-1$ einen Vektor y_k so zu, dass

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N} \ .$$

Die Quanten-Fouriertransformation auf einer orthonormale Basis $|0\rangle, \ldots, |N-1\rangle$ wird also folgendermaßen als linearer Operator definiert:

$$|j\rangle \longmapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle \ .$$

Verwendet man die reinen Zustände eines n-QBit-Systems als Basis, so ist $N = 2^n$.

Weiterhin macht es Sinn Ganzzahlen und Brüche binär darzustellen: $j = j_1 2^{n-1} + \ldots + j_n 2^0 = j_1 \ldots j_n$ und $0.j_1 \ldots j_n = j_1 2^{-1} + \ldots + j_n 2^{-n}$

2.1.1 Produktdarstellung

Wir formen um:

$$
\begin{aligned}
|j\rangle &\longmapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \sum_{l=1}^{n} k_l 2^{-l}} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} (|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle) \\
&= \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle)}{\sqrt{2^n}}
\end{aligned}
$$

Dieser Zustand lässt sich - wie folgend gezeigt - relativ leicht präparieren.

2.1.2 Algorithmus

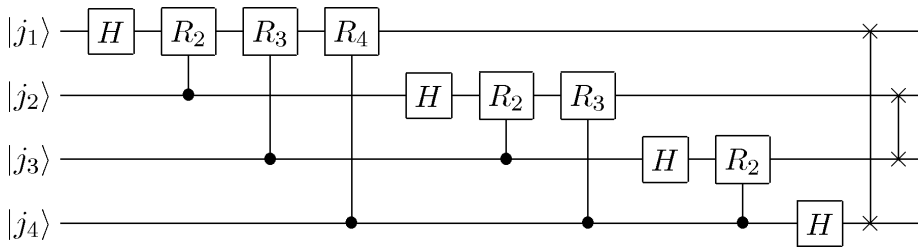

Abbildung 1: 4-QBit Quanten-Fouriertransformation

Wir beginnen mit dem höchsten QBit $|j_1\rangle$ und wenden darauf ein Hadamard-Gatter an. Darauf folgt eine Serie von kontrollierten Phasen- bzw. R-Gattern, wobei

$$
R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.
$$

Die Serie beginnt mit $k = 2$ und wird sukzessive von dem nächst tieferen QBit kontrolliert. Dieses Verfahren wird jeweils auf das nächst-niedriger QBit angewandt (Bsp. 2.-höchstes QBit: $|\widetilde{j_2}\rangle = R_{n-1} \dots R_2 H |j_2\rangle$, wobei R_2 von $|j_3\rangle$ und R_{n-1} von $|j_n\rangle$ kontrolliert wird).

Zum Schluss werden alle QBits achsensymetrisch vertauscht (SWAP). Die gewünschte Produktdarstellung ist erreicht.

2.1.3 Beispiel $|j_2\rangle$

Wir wollen den Algorithmus für das 2.-höchste QBit ausführen, um zu zeigen, dass wir auf die gewünschte Produktdarstellung kommen.

- Das Hadamard-Gatter:

$$\Rightarrow \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2}|1\rangle)$$

- Das R_2-Gatter:

$$\Rightarrow \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_2+j_3/2^2)}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)$$

- Die Anwendung der weiteren R-Gatter:

$$\Rightarrow \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle)$$

2.1.4 Laufzeit und Resourcen

Da auf jedes QBit $|j_k\rangle$ $n-k+1$ Gatter angewandt werden und die Vertauschung gegen Ende (SWAP) aus maximal $n/2$ Elementarvertauschungen (3 kontrollierte NOT-Gatter) besteht, benötigt man insgesamt $\frac{n(n+1)+3n}{2}$ Gatter. Also ist die Ordnung des Algorithmus $\Theta(n^2)$ und somit exponentiell schneller als die besten klassischen Algorithmen (FFT - Fast Fourier Transformation: $\Theta(n2^n)$).

2.2 Quanten-Phasen-Approximation

Häufig ist es nötig die Eigenwerte von unitären Operatoren zu bestimmen, welche sich als $U|u\rangle = e^{2\pi i\varphi}|u\rangle$ schreiben lassen; sie sind also durch Ihre Phase bereits vollständig identifiziert. Das Nähern dieser Phase soll hier weiterhin beschrieben werden, wobei es zu beachten gilt, dass es sich hierbei mehr um ein Konzept handelt, als um einen Algorithmus, da die Machbarkeit und Laufzeit stark von U abhängt.

2.2.1 Verfahren

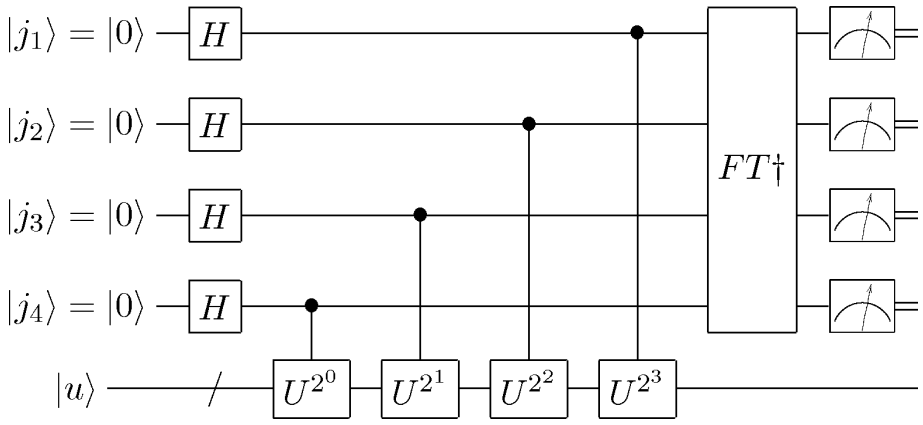

Abbildung 2: 4-QBit Phasen-Approximation

Das Verfahren verwendet 2 Quantenregister. Das erste wird mit $|0\rangle^{\otimes t}$ initialisiert, wobei t sowohl von der Genauigkeit der Näherung, als auch von gewünschten Erfolgswahrscheinlichkeit abhängt. Das zweite wird mit $|u\rangle$ initialisiert und behält diesen Wert. Nun werden folgende Schritte ausgeführt:

- Auf das erste Quantenregister wird ein Hadamard-Gatter angewandt:

  $$\Rightarrow \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t}|u\rangle$$

- Auf das zweite Quantenregister werden sukzessive U^{2^k}-Gatter angewandt, welche vom k-ten QBit des ersten Quantenregisters kontrolliert werden (beginnend mit dem niedrigsten QBit). Die Änderung der relativen Phase kann man im Tensorprodukt auch dem 1. Quantenregister zuordnen:

  $$\Rightarrow \frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle)\dots(|0\rangle + e^{2\pi i 2^0\varphi}|1\rangle)|u\rangle = \frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle|u\rangle$$

- Auf das erste Quantenregister wird die inverse Fouriertransformation angewandt:

  $$\Rightarrow |\widetilde{\varphi}\rangle|u\rangle$$

- Das erste Quantenregister wird gemessen:

  $$\Rightarrow \widetilde{\varphi}$$

2.2.2 Genauigkeit bzw. Erfolg

Falls $\varphi = \frac{b}{2^t}$ messen wir φ genau ($\varphi = \widetilde{\varphi}$). Ansonsten liefert $\widetilde{\varphi}$ eine gute Näherung von φ. Soll unser Messergebnis eine n-bit Näherung der Phase sein, so wissen wir aus [2, S. 224]

$$\epsilon \leq \frac{1}{2(2^{t-n}-2)}$$

und folgern

$$t = n + \left\lceil \mathrm{lb}(\frac{1}{2\epsilon} + 2) \right\rceil .$$

2.2.3 Laufzeit und Resourcen

Grundsätzlich werden für die Approximation der Phase t Hadamard-Gatter, die U^{2^k}-Gatter und ein Schaltkreis der inversen Fouriertransformation ($\Theta(t^2)$) benötigt. Da der Operator nicht näher spezifiziert wird, sagen wir die Laufzeit beträgt $\Theta(t^2)$ ausschließlich der Anwendung der U^{2^k}-Gatter. Außerdem sollte der Aufwand einen Eigenzustand $|u\rangle$ herzustellen beachtet werden!

2.3 Quantenmechanische Berechung der Ordnung

Für die Ganzzahlen x und N mit $0 < x, N$ und $x \perp N$ kann man die kleinste, positive Ganzzahl r berechnen, sodass

$$x^r \equiv 1 \,(\mathrm{mod}\, N) ,$$

wobei »mod N« angibt, dass wir uns in einer Restklassengruppe ($\mathbb{Z}_N^\star$) befinden. D. h. $1 \equiv N + 1 \,(\mathrm{mod}\, N)$. r nennen wir Ordnung.

2.3.1 Algorithmus

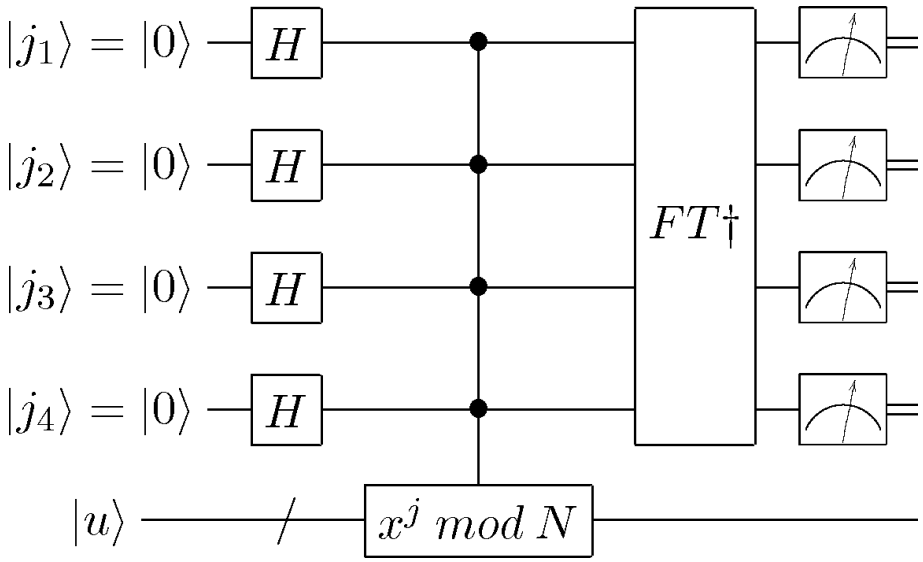

Abbildung 3: 4-QBit Berechnung der Ordnung

Um die Ordnung von x und N zu berechnen, führt man eine Phasen-Approximation des unitären Operators $U|y\rangle = |xy \,(\mathrm{mod}\, \mathrm{N})\rangle$ aus, wobei $y \in \{0,1\}^L$ mit $L = \lceil \mathrm{lb(N)} \rceil$. Man sieht leicht, dass

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^k \,(\mathrm{mod}\, N\rangle$$

mit $0 \leqq s \leqq r-1$ die Eigenzustände von U mit den dazugehörigen Eigenwerten $\lambda_s = e^{2\pi i s/r}$ sind.

Um die Phase nähern zu können, werden die Operatoren U^{2^m} in einer schnellen Implementierung benötigt. Hierfür verwenden wir die Tatsache, dass die Anwendung der U^{2^m}-Operatoren wie folgt umgeformt werden kann $|j\rangle U^{j_t 2^{t-1}} \ldots U^{j_1 2^0} |y\rangle = |j\rangle |x^{j_t 2^{t-1}} \ldots x^{j_1 2^0} y \,(\mathrm{mod}\, N)\rangle = |j\rangle |x^j y \,(\mathrm{mod}\, N)\rangle$ und benutzen den schnellen, klassischen Algorithmus, um Zahlen module zu exponentieren. Weiterhin benötigen wir einen Eigenzustand $|u_s\rangle$, was allerdings die Kenntnis von r erfordert und deshalb nicht möglich ist. Folgendes hier nicht bewiesene Theorem aus [2, S. 227] erlaubt es uns trotzdem die Phasen-Approximation auszuführen:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

Benutzen wir $|1\rangle$ als Eingabe für das 2. Quantenregister, werden wir später $\varphi \approx s/r$ für einen Eigenzustand s erhalten. Obwohl wir s nicht kennen, können wir mit der Kettenbruchentwicklung schnell ein Paar s' und r' erhalten, welches φ gut nähert (hierfür muss das n der Phasen-Approximation $2L+1$ Qbits betragen [2, S. 229]).

Sodann testen wir r', ob wirklich $x^{r'} \equiv 1 \,(\mathrm{mod}\, N)$.

2.3.2 Genauigkeit bzw. Erfolg

Natürlich hängt die Genauigkeit bzw. der Erfolg der Ordnungssuche von der Genauigkeit der Phasen-Approximation ab. Wir haben aber gesehen, dass diese durch hinzufügen von QBits effizient erhöht werden kann.

Das zweite - weitaus schwerwiegendere - Problem tritt auf, falls r und s gemeinsame Faktoren haben. Dann ist r' nämlich nur ein Faktor von r. Mehrere Abschätzungen z.B. für die Verteilung von Primzahlen für s geben hier an, wie wahrscheinlich es ist, dass die Zahl r' die Ordnung ist. Eine gute Abschätzung benötigt lediglich eine konstante Erhöhung der Ausführungsanzahl [2, S. 231].

2.3.3 Laufzeit und Resourcen

Es werden $\Theta(L)$ Hadamard-Gatter und eine inverse Fouriertransformation mit $\Theta(L^2)$ benötigt. Außerdem schlägt die modulare Exponentierung und die Kettenbruchentwicklung mit je $\Theta(L^3)$ zu Buche. Die Ausführungshäufigkeit muss konstant erhöht werden, um die Genauigkeit zu steigern. So benötigt der Algorithmus insgesamt $\Theta(L^3)$ Gatter.

3 Der Shor-Algorithmus

Nun haben wir das quantenmechanische Rüstzeug, um den auf Zahlentheorie basierenden Faktorisierungsalgorithmus zu verstehen.

3.1 Algorithmus

Die folgenden Anweisungen werden solange wiederholt, bis ein Teiler gefunden wurde. Die Wahrscheinlichkeit hierfür steigt exponentiell mit der Anzahl der Wiederholungen.

Man möchte einen nicht-trivialen Teiler von $N \in \mathbb{N}$ finden.

1. Wähle zufällig $x \in \mathbb{N}$ mit $1 < x < N$.
2. Berechne $\gcd(x, N)$. Falls dies $\neq 1$, gib den Teiler als Lösung zurück und beende den Algorithmus.
3. Bestimme die Ordnung r von x in der primen Restklassengruppe $\mathbb{Z}_n^\star$, also die kleinste Lösung von $x^r \equiv 1 \,(\mathrm{mod}\, N)$.
4. Sollte r ungerade oder $x^{r/2} \equiv -1 \,(\mathrm{mod}\, N)$, so ist x ungeeignet. Wiederhole den Algorithmus.
5. Die Lösung ergibt sich nun als $\gcd(x^{r/2} \pm 1, N)$.

3.2 Beweis

Da eine Ordnung nur existiert, falls x und n teilerfremd ist, wird dies zu Anfangs überprüft.

Ansonsten betrachten wir $\underbrace{(x^{r/2}+1)}_{a}\underbrace{(x^{r/2}-1)}_{b} = x^r - 1$ und stellen fest:

- $x^r - 1 \equiv 0 \,(\mathrm{mod}\, N)$, wegen der Definition von r. Deshalb können wir dafür auch $N\mathbb{Z}$ schreiben
- $a \not\equiv 0 \,(\mathrm{mod}\, N)$, was aus Schritt 4 folgt.
- $b \not\equiv 0 \,(\mathrm{mod}\, N)$, da r in Schritt 3 die kleinste Lösung war und natürlich $r/2 < r$ gilt.

Mit $ab = N\mathbb{Z} = (\prod_{i\in\mathbb{N}} N_i)\mathbb{Z}$ sehen wir, dass die Primfaktoren N_i von N in a und b enthalten sein müssen (die Aufteilung ist unbekannt). Allerdings können nicht alle Primfaktoren in a oder alle in b stecken, da sonst die obigen Äquivalenzausdrücke erfüllt wären. So enthalten sowohl a als auch b Teiler von N, welche wir mit gcd() schnell finden können.

3.3 Laufzeit und Resourcen

Neben dem Finden der Ordnung wird nur noch gcd() verwendet. Hierfür existieren aber klassische, schnelle Algorithmen wie z.B. der Euklidische Algorithmus.

Der Faktorisierungsalgorithmus benötigt also $\Theta(L^3)$ Gatter, wobei $L = \lceil \mathrm{lb(N)} \rceil$.

4 Abschließendes Beispiel

Zum Abschluss möchten wir den Algorithmus noch kurz am Beispiel der Zahl $N = 15$ nachvollziehen:

Als erstes wählen wir eine Zahl $x = 7$, deren Ordnung in Bezug auf N wir nun berechnen wollen. Hierfür initialisieren wir unsere Quantenregister mit $|0\rangle|1\rangle$ und wenden auf das erste (Länge $t = 11$) jeweils das Hadamard-Gatter an:

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle|1\rangle$$

Weiterhin wenden wir die U^{2^k}-Gatter auf das zweite Quantenregister an

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle|x^k \bmod N\rangle = \frac{1}{\sqrt{2^t}} \left[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \ldots\right],$$

führen die inverse Fouriertransformation auf das erste Quantenregister aus und messen es. Nachdem z. B. $1536/2048 = 3/4$ gemessen wurde, vermuten wir $r = 4$. Die Ordnung ist sowohl gerade, als auch $x^{r/2} = 49 \not\equiv -1 \pmod{15}$; d. h. $\gcd(7^2 \pm 1,\ 15) = 3,\ 4$ sind Teiler von 15!

Literatur

[1] Artur Ekert and Richard Jozsa. Quantum algorithms: Entanglement enhanced information processing, 1998.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 1 edition, 2000.

[3] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.SCI.STATIST.COMPUT.*, 26:1484, 1997.