Quantenalgorithmen

Robert Feldmann

Institut für Informatik
Universität Leipzig
Augustusplatz 10/11, 04109 Leipzig
RobertFeldmann@gmx.de
Januar 2002

Vorwort

Die vorliegende Arbeit geht auf einen Vortrag zurück, den ich im Dezember 2001 im Rahmen des Proseminar „Quantencomputer“ hielt. Quantenalgorithmen bilden das quantenmechanische Analogon zum klassischen Algorithmusbegriff. Sie können z.B. als Sequenz von Operatoren im Hilbertraum oder als Quantenschaltkreise repräsentiert werden und überführen einen gegebenen Startzustand in einen gewünschten Endzustand, an dem schließlich durch eine Messung das Ergebnis ausgelesen wird. Es zeigt sich nun, dass für verschiedene Klassen von Problemen der Einsatz von Quantenalgorithmen zu einem erheblichen Geschwindigkeitszuwachs bei der Lösungsberechnung führt.
Wir werden in dieser Arbeit einen Blick auf verschiedene Standardalgorithmen werfen und versuchen, die Grundprinzipien ihrer Wirkungsweise zu verstehen. Allerdings werden wir uns nicht mit der experimentellen Umsetzung beschäftigen, die auch nur für elementare Systeme bislang gelungen ist, sondern auf einem eher abstrakten Niveau verbleiben.

Inhaltsverzeichnis

Einführung

Ein Quantenalgorithmus ist intuitiv ein Algorithmus, der als elementare Operationen Transformationen von Zuständen aus einem Hilbertraum besitzt. Dabei wird ein Ausgangszustand in einen Endzustand überführt, d.h. völlig analog zum klassischen Fall. Der entscheidende Unterschied besteht darin, dass für die Zustände ein Superpositionsprinzip gilt und die Transformationen linear sind. Weiterhin sind fast alle physikalisch bedeutsamen Transformationen unitär , da sie die Übergangswahrscheinlichkeit erhalten. Schauen wir uns dazu folgendes Paradebeispiel an. Sei ein Zustand

$\|\Psi\rangle = \sum_{i=0}^{k} c_i \|i,0\rangle$	1

Gegeben. Nun wenden wir einen linearen, unitären Operator U_f darauf an, der die Funktion f realisiert. Dieser Operator überführt einen Zustand $|x, y\rangle$ wie folgt:

$U_f \|x, y\rangle = \|x, y + f(x) \pmod 2\rangle$, und deshalb $U_f \|\Psi\rangle = \sum_{i=0}^{k} c_i \|i, f(i)\rangle$	2

D.h. eine einmalige Anwendung von U_f liefert uns alle Werte der Funktion *f*, klassisch wären dazu k Aufrufe der Funktion *f* nötig. Dies verstehen wir unter „Quantenparallelismus“, eine in der Tat beeindruckende Leistung. Allerdings hat dieser Ansatz einen entscheidenden Nachteil. Die Regeln der Quantenmechanik erlauben nämlich keine Bestimmung des Zustandes $U_f|\Psi\rangle$, bei einer Messung wird stattdessen eine Komponente $|j, f(j)\rangle$ herausprojeziert.

Die Leistung eines Quantencomputers hängt entscheidend davon ab, ob und inwieweit es unter Nutzung von verschiedenen Techniken gelingt, selektiv gewünschte Informationen aus Quantenzuständen zu erhalten.

Wie schon erwähnt, lassen sich Zustände als Vektoren in einem n-dimensionalen Hilbertraum deuten[1]. Ein wichtiger Spezialfall ist hierbei der 2-dimensionale Hilbertraum $\mathcal{H}$. Die Zustände aus $\mathcal{H}$ werden auch als quantum-bits, kurz *qubits* bezeichnet, und lassen sich physikalisch leicht realisieren. Die Standardbasis in $\mathcal{H}$ ist gegeben durch die (normierten) Vektoren $|0\rangle$ und $|1\rangle$. Eine weitere wichtige Basis, die uns im folgenden immer wieder begegnen wird, ist die sogenannte duale Basis von $\mathcal{H}$

$\|0'\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle), \|1'\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	3

[1] wir beschränken uns im folgenden auf den endlichdimensionalen Fall

Die *Hadamard-Transformation H* überführt die Standardbasis in die duale Basis und wieder zurück, d.h.

$H\|i\rangle = \|i'\rangle, i \in \{0,1\} \quad H^2 = \mathrm{I}$	4

Dem Übergang von qubits zu Multi-qubits entspricht der Übergang vom 2-dimensionalen Hilbertraum $\mathcal{H}$ zum Tensorprodukt-Hilbertraum $\mathcal{H}^{\mathsf{n}}$. Die Hadamardtransformation H lässt sich für diesen Fall kanonisch erweitern über das Tensorprodukt, also:

$\mathcal{H}^{\mathsf{n}} = \bigotimes_{i=1}^{n} \mathcal{H} \quad \text{und} \quad H_n = \bigotimes_{i=1}^{n} H$	5

Aus 3,4,5 ergibt sich für die Hadamardtransformation H_n sofort

$H_n\|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\circ y}\|y\rangle = \|x'\rangle \quad , \quad \text{mit } x \circ y = \sum_{l=1}^{n} x_l y_l$	6

Eine weitere wichtige Transformation ist die diskrete Quantenfouriertransformation Q_{q}
Die Wirkung auf die Basis ist wie folgt definiert:

$Q_q\|a\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{i2\pi\frac{a}{q}c}\|c\rangle$	7

Mit $Q_q\|f\rangle = \|\hat{f}\rangle$ gilt für die Koeffizienten $f(a) = \langle a\|f\rangle$, $\hat{f}(c) = \langle c\|\hat{f}\rangle$ folgendes Transformationsverhalten

$\hat{f}(c) = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} e^{i2\pi\frac{c}{q}a} f(a)$	8

Dies entspricht der bekannten diskreten Fouriertransformation.

Offensichtlich ist Q_q eine Verallgemeinerung von H und es gilt: $H_n = \bigotimes_{i=1}^{n} Q_2$

Das Deutsch-Problem

Eines der ersten Probleme, für die ein Geschwindigkeitsgewinn durch einen Quantenalgorithmus gefunden wurde, ist das Problem von Deutsch[2]: Gegeben sei eine Funktion $f:\{0,1\}^n \to \{0,1\}$, die entweder konstant oder ausgeglichen ist, d.h. für alle Argumente den selben Funktionswert liefert bzw. Nullen und Einsen in gleicher Anzahl. Die Aufgabe ist es nun herauszufinden, ob der erste oder der zweite Fall vorliegt. Ein klassischer Ansatz benötigt, falls z.B. der erste Fall vorliegt, $2^{n-1}+1$ Auswertungen der Funktion f um dies sicher zu entscheiden.

Nun ist jedoch zu klären, was genau im *quantenphysikalischen Sinne* unter einem Aufruf einer Funktion zu verstehen ist. Eine Funktion, manifestiert als Quantenschaltkreis bzw. als Operator über dem Hilbertraum, besitzt als fundamentale Eigenschaft die Möglichkeit, Superpositionen von Zuständen gleichzeitig zu bearbeiten. Einem Aufruf einer Funktion entspricht dabei die einmalige Anwendung des entsprechenden Operators. Unter einem Quantenalgorithmus stellen wir uns im weiteren die Sequenz von Operatoren vor, die einen Ausgangszustand in einen Endzustand überführen. Unter der Anzahl der Funktionsaufrufe in einem Quantenalgorithmus wollen wir dabei die Anzahl der die Funktion f quantenphysikalisch repräsentierenden Operatoren verstehen, die in dieser Sequenz vorkommen.

Bei der Betrachtung des vorliegenden Problemes zeigt sich, dass *ein* Aufruf der „Quanten“funktion f bereits ausreichend ist. Es ist damit für diese Problemklasse eine exponentielle Beschleunigung erreichbar. Widmen wir uns nun dem Algorithmus, der dieses Problem löst.

Wir beginnen mit einem System im Zustand $|0^n\rangle \otimes |1\rangle$ und wenden darauf die Transformation $H_{n+1}^{-1} U_f H_{n+1}$ an. Schließlich messen wir die ersten n-qubits und werten das Ergebnis aus.

$$|0^n\rangle \otimes |1\rangle = \underbrace{|0\ldots0\rangle}_{n} \otimes |1\rangle \xrightarrow{H_{n+1}} \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} |i\rangle \otimes \left(|0\rangle - |1\rangle\right) \xrightarrow{U_f} \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} \left(|i\rangle \otimes |\, f(i)\rangle - |i\rangle \otimes |\, f(i)+1\rangle\right)$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} |i\rangle \otimes \left(|0\rangle - |1\rangle\right) \xrightarrow{H_{n+1}^{-1}=H_{n+1}} \left(\frac{1}{\sqrt{2^n}}\right)^2 \sum_{i=0}^{2^n-1} (-1)^{f(i)} \sum_{j=0}^{2^n-1} (-1)^{i\circ j} |\, j\rangle \otimes |1\rangle$$

$$= \frac{1}{2^n} \sum_{j=0}^{2^n-1} A_f(j) |\, j\rangle \otimes |1\rangle \quad , \text{ mit } A_f(j) = \sum_{i=0}^{2^n-1} (-1)^{f(i)} (-1)^{i\circ j}$$

Untersuchen wir den Koeffizienten $A_f(j)$ für die entsprechenden Fälle:

a) f ist konstant: $A_f(j) = \sum_{i=0}^{2^n-1} (-1)^{i\circ j} = 2^n \delta_0^j$

[2] Hier in dieser etwas verallgemeinerten Form auch als Deutsch-Jozsa Problem bekannt

Der Beweis des zweiten Gleichheitszeichens ist schnell erbracht.

Es gibt 3 Fälle

1) $k=hw(j)=0$ d.h. $j=0$, dann gilt sofort $A_f(j)=2^n$

2) $2 \nmid k$, dann folgt sofort $A_f(j)=0$, da für jedes k die Anzahl der Zahlen i (dargestellt als Bitfolge), die eine –1 bzw. 1 erzeugen, gleich ist

3) $2 \mid k \;\&\; k \neq 0$, die Anzahl der Zahlen, die eine –1 beisteuern beträgt: $\binom{k}{1}+\binom{k}{3}+\ldots+\binom{k}{k-1}$, während $\binom{k}{0}+\binom{k}{2}+\cdots+\binom{k}{k}$ Zustände eine 1 addieren. Nun gilt jedoch $(1-1)^n = 0 = \binom{k}{0}-\binom{k}{1}+\binom{k}{2}-\binom{k}{3}\pm\ldots+\binom{k}{k}$ und somit $A_f(j)=0$.

Also $A_f(j)=\sum_{i=0}^{2^n-1}(-1)^{i\circ j}=2^n\delta_0^j$

Es ergibt sich somit $H_{n+1}^{-1}U_f H_{n+1}\left|0^n\right\rangle\otimes\left|1\right\rangle=\left|0^n\right\rangle\otimes\left|1\right\rangle$.

Eine Messung der ersten n-qubits liefert somit mit Sicherheit den Wert 0.

b) f ist ausgeglichen: Wie groß ist jetzt die Wahrscheinlichkeit , dass die ersten n-qubits 0 sind? Nun für $j=0$ ergibt sich: $A_f(0)=\sum_{i=0}^{2^n-1}(-1)^{f(i)}(-1)^{i\circ 0}=\sum_{i=0}^{2^n-1}(-1)^{f(i)}=0$, d.h. die Wahrscheinlichkeit ist null, dass eine Messung der ersten n-qubits den Wert 0 ergibt.

Somit liefert eine Messung der ersten n-qubits (dazu benötigen wir eine Observable mit der Standardbasis als nichtentartete Eigenzustände), d.h. eine Projektion auf die Standardbasis im $\mathcal{H}^n$, das jeweilige Ergebnis. Damit ist das Problem zumindest prinzipiell gelöst.

Das Problem von van Dam

Das Problem ist hier, eine gegebene Funktion $f:\{1,\ldots,n\}\to\{0,1\}$ zu bestimmen. Klassisch sind dazu sicher n-Aufrufe der Funktion nötig. Der folgende Algorithmus schafft dies mit einer Wahrscheinlichkeit von 95% bereits mit gut der Hälfte der Aufrufe, im Sinne der obigen Definition.

Der Ansatz des folgenden Algorithmus besteht darin, die Identität $H_n^2 = I$ (folgt sofort aus 4 und 5) für eine approximative Lösung auszunutzen. Mit w_f bezeichnen wir im folgenden den String *(f(1),...,f(n))*, der im $\mathcal{H}^n$ auch durch einen Basisvektor $|w_f\rangle$ repräsentiert wird. Es gilt demnach:

$\|w_f\rangle = H_n^2\|w_f\rangle = \frac{1}{\sqrt{2^n}} H_n \sum_{x\in\{0,1\}^n} (-1)^{x\circ w_f}\|x\rangle$	8

Der Trick besteht nun darin, die Summe in 8 nicht vollständig auszurechnen, sondern lediglich für solche *x* mit $hw(x)\le k$, für ein geeignet gewähltes *k*. Der so erhaltene Zustand klappt dann bei Auslesung aller qubits mit großer Wahrscheinlichkeit auf den Lösungsvektor $|w_f\rangle$.

Für diesen Quantenalgorithmus ist der folgende lineare, unitäre Operator V_{F_K} von Bedeutung.

$V_{F_K}\|x\rangle = (-1)^{F_k(x)}\|x\rangle$, wobei $F_k(x) = \begin{cases} x\circ w_f = \sum_{i=1}^{n} x_i f(i), & \text{für } hw(x)\le k \\ 0 \quad \text{sonst} \end{cases}$	9

Zur Analyse des Operators V_{F_K} definieren wir eine neue Funktion G_m mit

$G_m(x) = \begin{cases} f(i_m), \text{mit } i_m \text{ die } m-\text{kleinste Zahl, so daß } x_{i_m} = 1 \\ 0 \quad \text{sonst} \end{cases}$	10

Der daraus erhaltene Operator V_{G_m}, für den analog $V_{G_m}|x\rangle = (-1)^{G_m(x)}|x\rangle$ gilt, entspricht demnach *einer* Auswertung der Funktion *f*.

Die Anwendung von $V_{F_K} = V_{G_k} V_{G_{k-1}} \ldots V_{G_1}$ entspricht somit k Funktionsaufrufen.

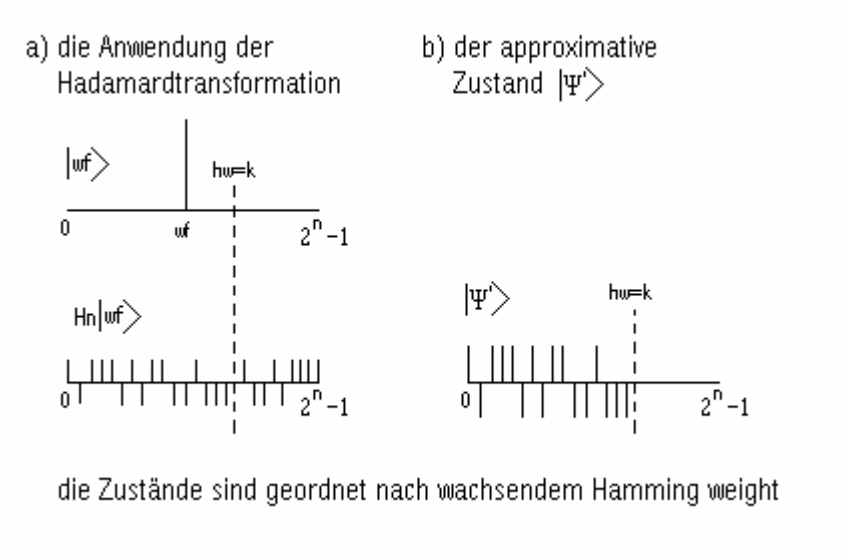

Abbildung 1

I) Wir beginnen mit dem Startzustand $|\Psi\rangle$, der wie gegeben präpariert sei:

$$\|\Psi\rangle = \frac{1}{\sqrt{M_k}} \sum_{x\in\{0,1\}^n}^{hw(x)\le k} \|x\rangle$$	11

dabei ist *hw(x)* das Hamming weight von x, d.h. die Anzahl der Einsen in der Zeichenkette $(x_1,...,x_n)$, und M_k ist eine Normierungskonstante, welche sich sofort zu $M_k = \sum_{l=0}^{k} \binom{n}{l}$ ergibt.

II) Wir wenden den Operator $H_n V_{F_K}$ auf $|\Psi\rangle$ an und erhalten

$$\|\Psi''\rangle = H_n V_{F_k} \|\Psi\rangle = H_n \|\Psi'\rangle = \frac{1}{\sqrt{M_k}} H_n \sum_{x\in\{0,1\}^n}^{hw(x)\le k} (-1)^{x\circ w_f} \|x\rangle$$	12

III) Schließlich messen wir alle n qubits von $|\Psi''\rangle$ aus und projizieren damit auf einen Basiszustand. Berechnen wir nun die Wahrscheinlichkeit, dass wir dabei $|w_f\rangle$ erhalten, wobei wir die Hermitezität von H_n ausnutzen (Abb.1):

$$P(|\Psi''\rangle \text{ ergibt } |w_f\rangle) = |\langle w_f | \Psi''\rangle|^2 = |\langle H_n w_f | \Psi'\rangle|^2$$

$$= \left| \frac{1}{\sqrt{2^n M_k}} \sum_{y\in\{0,1\}^n} (-1)^{y\circ w_f} \sum_{x\in\{0,1\}^n}^{hw(x)\le k} (-1)^{x\circ w_f} \langle y|x\rangle \right|^2 = \frac{1}{2^n M_k} \left| \sum_{y\in\{0,1\}^n}^{hw(y)\le k} (-1)^{2(y\circ w_f)} \right|^2$$

$$= \frac{1}{2^n M_k} \left(\sum_{y\in\{0,1\}^n}^{hw(y)\le k} 1 \right)^2 = \frac{M_k}{2^n}$$, es lässt sich nun zeigen, dass dies größer 0.95 ist, falls $k \ge \frac{n}{2} + \sqrt{n}$.

Wir erhalten demnach den richtigen String *(f(1),...,f(n))* mit einer Wahrscheinlichkeit, die sich als das Verhältnis der Anzahl M_k der Zustände mit $hw(x) \le k$ zur Gesamtanzahl 2^n aller Zustände ergibt (Abb.2) und zwar zu 95%, falls $k \ge \frac{n}{2} + \sqrt{n}$.

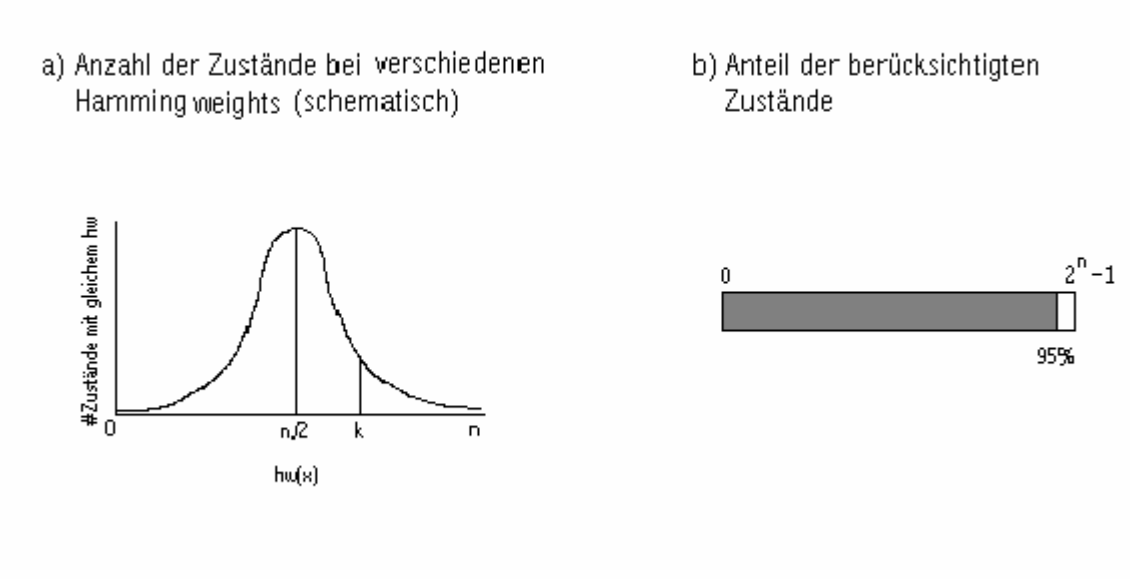

Abbildung 2

Sofern wir als Startzustand

$\left\|\Psi\right\rangle = \frac{1}{\sqrt{\tilde{M}_k}} \sum_{x\in\{0,1\}^n}^{hw(x)\le k \wedge hw(x)\ge n-k} \left\|x\right\rangle$	13

wählen, d.h. nur Zustände nahe des Maximums berücksichtigen, gewinnen wir nichts. Aufgrund $\tilde{M}_k \le M_k$ ist die Wahrscheinlichkeit[3] auf $\left|w_f\right\rangle$ zu projizieren etwas geringer. Entscheidend ist jedoch, dass die Anzahl der Funktionsaufrufe durch das Hamming weight der Zustände bestimmt wird. Unverändert kommen aber Zustände vor, welche das maximale Hamming weight k besitzen, woraus k Funktionsaufrufe resultieren.

[3] gegeben durch $\frac{\tilde{M}_k}{2^n}$ nach analoger Rechnung

Shor's Algorithmus und die Primfaktorisierung

Der Algrorithmus von Shor ist wohl einer von wenigen Quantenalgorithmen, der uns auf so beeindruckende Art und Weise die Vorzüge eines Quantencomputers vor Augen hält. Im Gegensatz zu obigen Problemen ist die Lösung von Shor zur effizienten Primfaktorzerlegung sehr praxisrelevant.

RSA

Ein Großteil heutiger Kryptosysteme beruht auf Einwegfunktionen. Unter anderem auch die RSA-Verschlüsselung, welches die praktische Unmöglichkeit des Zerlegens des Produktes zweier Primzahlen in seine Faktoren ausnutzt. Das Grundprinzip von RSA ist dabei wie folgt:

- 2 Primzahlen (512-1024bit) p, q werden miteinander multipliziert : $n=pq$
- eine Zahl d, die zu $\Phi(n)=(p-1)(q-1)$ teilerfremd ist, wird gewählt
- eine Zahl e mit $ed \equiv 1 \bmod \Phi(n)$ wird gewählt

Nun bilden n und e den öffentlichen Schlüssel, während der private Schlüssel aus p, q, d besteht. Im weiteren bezeichnen wir den Klartext mit m und das Chiffre mit c. Die Verschlüsselung bzw. Entschlüsselung geschieht nach folgenden Gleichungen:

$c = m^e \bmod n \quad , \quad m = c^d \bmod n$	14

Man kann nun zeigen, dass RSA unsicher ist, sobald p, q aus n berechnet werden können[4].

Primfaktorisierung

Der beste bekannte klassische Algorithmus zur Primfaktorisierung hat eine exponentielle Komplexität in *log(n)* , d.h. ist exponentiell in der Zahl der bits. Shor's Algorithmus besitzt hingegen asymptotisch eine quadratische Komplexität (/1/).
Der gesamte Algorithmus besteht aus mehreren Teilen, wovon jedoch nur eine Unterroutine inhärent ein Quantenalgorithmus ist. Werfen wir zunächst einen Blick auf das generelle Konzept der Faktorisierung.
Gegeben sei eine Funktion $f_{n,k}(k) = x^k \bmod n$. Unter der Periode von $f_{n,k}$ verstehen wir die kleinste von null verschiedene Zahl r für die $x^r \equiv 1$ gilt[5]. Da r auch als die (Element-)Ordnung

[4] Allerdings ist dies dies nicht einzige Möglichkeit der Attacke auf RSA. So ist z.B. im Moment nicht bekannt, ob die Berechnung von $\Phi(n)$ für gegebenes n nicht doch in polynomialer Zeit geleistet werden kann.

von x bezeichnet wird, ist das Problem, die Periode von $f_{n,k}$ zu bestimmen, auch als „order problem“ bekannt. Falls x Element aus $Z_n^* = \{i \mid i \le n,\ \text{ggT}(i,n) = 1\}$ und zu n teilerfremd ist, so ist die Existenz einer Periode gesichert, da stets $x^{\Phi(n)} \equiv 1$ (Satz von Euler-Fermat) mit $\Phi(n) = |Z_n^*|$.

Sei r bekannt und gerade, dann gilt:

$$x^r \equiv 1 \rightarrow \left(x^{r/2}+1\right)\left(x^{r/2}-1\right) \equiv 0$$

Klar ist, dass $\left(x^{r/2}-1\right) \not\equiv 0$, da ja r die kleinste Zahl ist, die dies erfüllt.

Wir fordern jetzt $\left(x^{r/2}+1\right) \not\equiv 0$. In diesem Fall, d.h. wenn r gerade und $x^{r/2} \not\equiv \pm 1$ ist,

ergeben die Berechnungen von

$\text{ggT}\left(x^{r/2}+1, n\right), \text{ggT}\left(x^{r/2}-1, n\right)$	15

nichttriviale Faktoren von *n*. Der Algorithmus besteht nun darin, ein *x* zu wählen und es zu verwerfen, falls *r* ungerade oder $x^{r/2} \equiv \pm 1$ ist, im anderen Falle jedoch die nichttrivialen Teiler nach 15 zu bestimmen. Dies ist mit dem Euklidischen Algorithmus in polynomialer Zeit möglich. Die Wahrscheinlichkeit, ein zufällig gewähltes x aufgrund obiger Bedingungen verwerfen zu müssen, liegt bei weniger als 50% (/1/). Somit erhält man i.a. bereits nach wenigen Versuchen eine Faktorisierung von *n*. Derjenige Teil, der im gänzlich klassischen Algorithmus für die exponentielle Komplexität sorgt, ist die Bestimmung von *r* für gegebenes *x*. Der quantenphysikalische Teil des Shor-Algorithmus setzt an dieser Stelle an und liefert eine Berechnung der Ordnung von x in polynomialer Zeit.

Folgendes Schema gibt eine Übersicht über den Algorithmus von Shor

A Wähle ein *x* zufällig aus $\{1,\ldots,n\}$

B Berechne y = ggT(*x*,*n*) , falls dies nicht 1 ist, so springe nach F

C Quantenroutine: Berechne die Periode r der Funktion $f_{n,k}(k) = x^k \bmod n$

D Falls r ungerade ist, so beginne bei A

E Berechne $y = \text{ggT}\left(x^{r/2}+1, n\right), \text{ggT}\left(x^{r/2}-1, n\right)$, falls nur triviale Teiler, so beginne bei A

F gib *y* aus

[5] Im weiteren immer mod *n*

Die Quantenroutine

Wir wollen eine *m*-bit Zahl *n* faktorisieren. Gegeben ist ein zufällig gewähltes *x*. Wir wählen eine Zahl *q* mit $n^2 \le q < 2n^2$. Diese Wahl der Zahl *q* hat technische Gründe, soll uns aber im weiteren nicht zu sehr interessieren (siehe /2/ für ausführlichere Informationen). Die Quanteroutine besteht aus 3 Schritten. Im ersten wird ein Superpositionszustand hergestellt und durch Messung ein Unterraum herausprojiziert, im zweiten Schritt wird durch Anwendung der Quantenfouriertransformation der Effekt eines vorhandenen Offsets aufgehoben und schließlich wird im dritten Schritt die Periode ausgelesen.

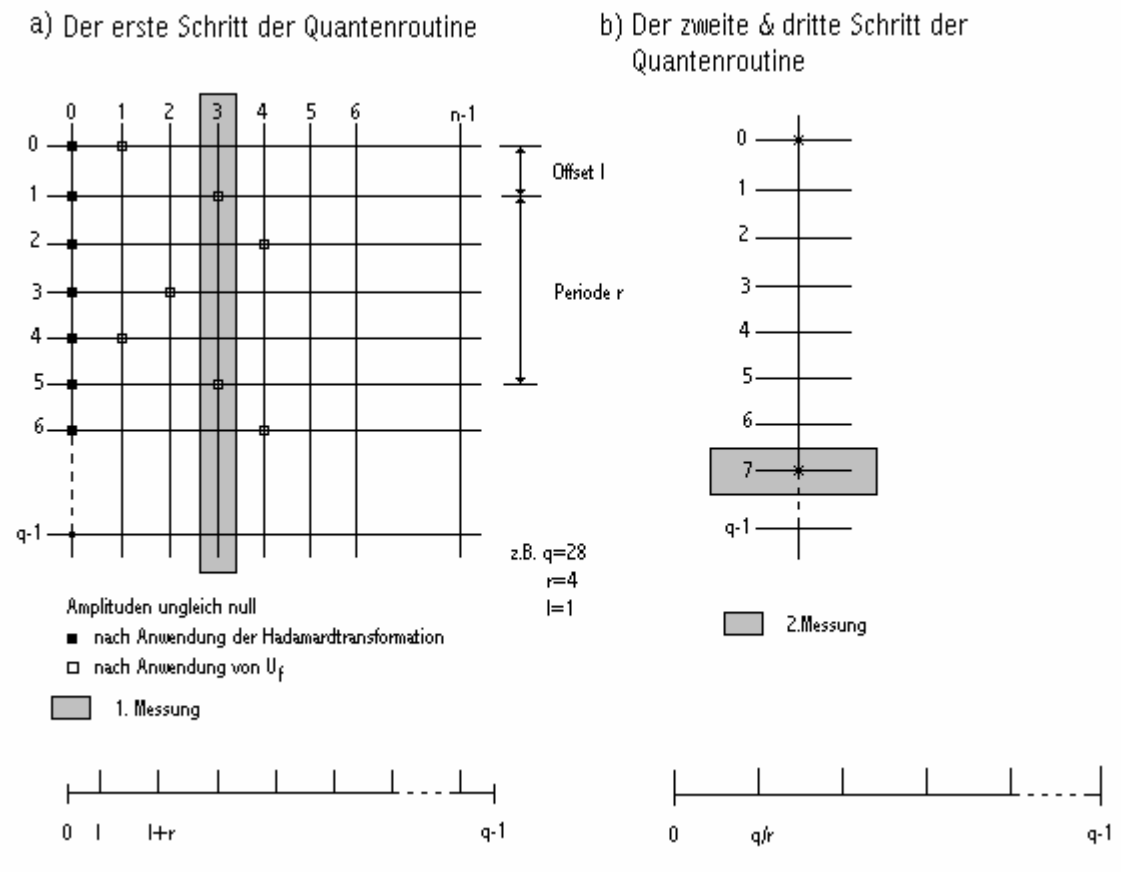

Abbildung 3

I) Unser Startvektor sei $\left|0^q\right\rangle \otimes \left|0^n\right\rangle$. Wir wenden nun die Hadamardtransformation $H_q \otimes \mathbf{1}^n$ und dann $U_{f_{n,x}}$ an. Damit erhalten wir:

$$\left|0^q\right\rangle \otimes \left|0^n\right\rangle \xrightarrow{H_n \otimes 1^n} \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} \left|a\right\rangle \otimes \left|0^n\right\rangle \xrightarrow{U_{fn,x}} \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} \left|a\right\rangle \otimes \left|x^a \bmod n\right\rangle \qquad (16)$$

Nach einer Messung an den letzten n-qubits (der Messwert $y=x^l$ ist unerheblich) erhalten wir (Abb.3)

$\left\|\phi_l\right\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} \left\|l + jr\right\rangle \otimes \left\|x^l\right\rangle$	17

II) Als hinderlich zur Bestimmung der Periode r erweist sich nun der Offset l. Inbesondere lässt sich aus einer Messung an den ersten q-qubits nichts über r erfahren, da l gleichverteilt jeden Wert zwischen 0 und r-1 annimmt. Deshalb wenden wir nun die Quantenfouriertransformation auf $\left\|\phi_l\right\rangle$ an. Zur Vereinfachung gehen wir ab jetzt von $r \mid q$ aus, und damit von $A = \frac{q}{r}$. Für den Fall $r \nmid q$ bleibt das Prinzip aber das gleiche, in diesem Kontext wird auch die Wahl von $n^2 \le q < 2n^2$ erst verständlich (siehe /2/). Um die Notation zu vereinfachen, führen wir die letzten n Tensorkomponenten nicht mehr explizit auf (sie bleiben ohnehin unverändert). Also:

$$\left\|\phi_l\right\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} \left\|l + jr\right\rangle$$

Anwendung des Operators Q_q liefert $\left\|\phi_{out}\right\rangle = Q_q \left\|\phi_l\right\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{i2\pi c(l+jr)/q} \left\|c\right\rangle$

$$= \sum_{c=0}^{q-1} \alpha_c \left\|c\right\rangle \text{ mit } \alpha_c = \frac{\sqrt{r}}{q} \sum_{j=0}^{\frac{q}{r}-1} e^{i2\pi c(l+jr)/q}$$

α_c ergibt sich zu $\alpha_c = \begin{cases} \frac{1}{\sqrt{r}} e^{i2\pi cl/q} & \text{falls } q \mid cr \\ 0 & \textit{sonst} \end{cases}$

Der Beweis ist recht elementar, der erste Fall ist trivial, im zweiten Fall ergibt sich α_c als Partialsumme einer geometrischen Reihe oder einfacher als die Summe von $\frac{q}{r}$ gleichverteilten komplexen Zahlen auf dem Einheitskreis und damit zu null.
Somit berechnen wir für

$\left\|\phi_{out}\right\rangle = Q_q \left\|\phi_l\right\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{i2\pi jl/r} \left\| j\frac{q}{r} \right\rangle$	18

Da jeglicher Offset verschwunden ist, korrelieren nun die verschiedene Messungen miteinander und wir sind in der Lage die Periode r auszulesen. Allerdings hat der vorliegende Zustand die Periode $\frac{q}{r}$, weshalb wir noch nicht fertig sind (Abb 3).

III) Wir messen nun die ersten q qubits und erhalten ein $c = \lambda \frac{q}{r}$ mit $\lambda \in \{0,\ldots,r-1\}$, wobei wir jedes λ mit der Wahrscheinlichkeit 1/r erhalten. Bekannt ist also c, q und es gilt $\frac{c}{q} = \frac{\lambda}{r}$ und somit erhalten wir, falls $\mathrm{ggT}(r,\lambda) = 1$ gilt:

$r = \frac{q}{\mathrm{ggT}(c,q)}$	19

Die Wahrscheinlichkeit, dass $\mathrm{ggT}(r,\lambda) = 1$ lässt sich leicht errechnen als $\frac{|Z_r^*|}{r} = \frac{\Phi(r)}{r}$.

Benutzen wir die sehr grobe Abschätzung $\Phi(r) \geq \frac{r}{\ln(r)}$, so sehen wir, dass wir mit ca. $\ln r \leq \ln n$ Versuchen auskommen sollten. Die Quanten Routine lässt sich somit effektiv implementieren und daher der gesamte Algorithmus.

Die folgenden Tabellen belegen das enorme Potential, dass in diesem Quantenalgorithmus steckt. Einmal wird die Zeit, welche die leistungsfähigsten klassischen Computer zur Faktorisierung benötigen, für die nächsten Jahre geschätzt (unter Berücksichtigung des stetigen Geschwindigkeitszuwachses), zum anderen die Zeit, die ein Quantencomputer aufwenden würde (aus /1/).

Klassischer Computer

Bitanzahl	1024	2048	4096
2006	10^5 a	$5*10^{15}$ a	$3*10^{29}$ a
2024	38 a	10^{12} a	$7*10^{25}$ a
2042	3 Tage	$3*10^8$ a	$2*10^{22}$ a

Quantencomputer

Bitanzahl	1024	2048	4096
Faktorisierungszeit	4.5min	36min	4.8h

Quellen und weiterführende Literatur

/1/ J. Gruska, *Quantum computing*, McGraw-Hill, University Press, Cambridge, 1999

/2/ B. Crell & A. Uhlmann, *Einführung in die Grundlagen und Protokolle der Quanteninformatik*, Institut für Theoretische Physik, Universität Leipzig

/3/ D. Deutsch et al.: *Quantum Computation*, Physics World, März 1998, S41-45

/4/ C.H. Bennet: *Quantum Information and Computation*, Physics, Today, Oktober 1995, S24-30

/5/ J.I. Cirac et al.: *Quantum Computations with Cold Trapped Ions*, Phys. Rev. Lett. 74 (1995) , 4091

/6/ S. Lloyd: *A Potentially Realizable Quantum Computer*, Science 261 (1993) , 1569

/7/ R. Der, *Vorlesungsskript Quantencomputer*, Institut für Informatik, Universität Leipzig