

Quantenberechnungen - Einführung

[Seminar Computing Beyond Turing]

Gunnar Bergmann
Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck, Deutschland
gunnar.bergmann@student.uni-luebeck.de

ABSTRACT

In dieser Ausarbeitung werden die Grundlagen über Quantencomputer vorgestellt. Es werden grundlegende Begriffe und mathematische Formulierungen zur Beschreibung von Quantenrechner vorgestellt. Es wird eine Beurteilung der Berechenbarkeit und Komplexität gegeben und am Beispiel von Deutschs Algorithmus erläutert, wie man die Eigenschaften von Quantenzuständen ausnutzen kann, um Berechnungen zu parallelisieren. Desweiteren werden einige Probleme erklärt, die die Konstruktion von Quantenalgorithmen erschweren.

1. EINFÜHRUNG

In der Quantenmechanik sind Quantensysteme nicht auf klassische Zustände beschränkt, sondern können auch Superpositionen einnehmen. Das bedeutet, sie sind in mehreren Zuständen gleichzeitig. Quanten in Superposition können immer noch mit anderen Quantensystemen interagieren, aber es ist nicht möglich, den Zustand direkt zu messen. Stattdessen wechselt das Quantensystem beim Messen in einen Basiszustand.

Quantencomputer benutzen die Eigenschaften der Quantenmechanik, um über Quantenparallelismus bei bestimmten Problemen schneller rechnen zu können, als klassische Computer.

Wir werden sehen, dass Quantenrechner jeden anderen Rechner simulieren können, aber bestimmte Algorithmen Quantenparallelismus ausnutzen können, um exponentiell viele gleichzeitige Berechnungen durchzuführen.

Quantenmechanik hat oft unintuitive Effekte, aber mathematische Modelle erlauben eine akkurate Beschreibung und Vorhersage von experimentell beobachtbarem Verhalten. Diese unintuitive Natur von Quanten erschwert die Konstruktion von Quantenalgorithmen.

Ein wichtiges Anwendungsgebiet für Quantenrechner ist die

Simulation von Quantensystemen zu Forschungszwecken. Außerhalb dessen sind nur wenige Algorithmen bekannt, welche die Vorteile von Quantencomputern ausnutzen können. Am bekanntesten sind Shors Algorithmus zur Faktorisierung in Polynomialzeit und Grovers Algorithmus, der einen Suchraum der Größe N in Laufzeit $\Theta(\sqrt{N})$ durchsucht [2].

Alle diese Algorithmen gehen über diese Arbeit hinaus. Stattdessen wird hier ein künstliches Problem betrachtet, das keine realen Anwendungsgebiete hat, aber in Linearzeit auf einem Quantenrechner gelöst werden kann, während ein klassischer Rechner Exponentialzeit benötigt.

2. QUANTENBITS

Quantenbits (gewöhnlich Qubits genannt) generalisieren klassische Bits. Es gibt zwei Basiszustände, $|0\rangle$ und $|1\rangle$, die man wie klassische Bits interpretieren und verwenden kann.

Anders als gewöhnliche Bits erlauben Qubits Superpositionen, die Anteile von beiden Zuständen gleichzeitig enthalten.

Ein mathematisches Modell für ein Qubit ist ein Vektorraum

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

mit $\alpha, \beta \in \mathbb{C}$ und $|\alpha|^2 + |\beta|^2 = 1$, wobei $|0\rangle$ und $|1\rangle$ eine Orthonormalbasis bilden. Für viele Probleme, insbesondere alle in dieser Ausarbeitung thematisierten, reichen $\alpha, \beta \in \mathbb{R}$ aus.

Durch geeignete Wahl von α und β können dann die Superpositionen modelliert werden. Häufig tritt zum Beispiel der Zustand $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ auf.

Es ist nicht möglich, den exakten Zustand eines Qubits zu bestimmen. Stattdessen ergibt eine Messung $|0\rangle$ mit Wahrscheinlichkeit $|\alpha|^2$ und $|1\rangle$ mit Wahrscheinlichkeit $|\beta|^2$. Darüber hinaus verändert die Messung den Zustand des Qubits selber und das Qubit nimmt dann den Zustand $|0\rangle$ bzw. $|1\rangle$ an.

Durch Messung kann man also nicht den genauen Zustand bestimmen und erhält anschließend nur noch die Basiszustände, die keine weiteren Informationen als klassische Bits enthalten. Wie in Abschnitt 6 noch an einem Beispiel gezeigt wird, kann man aber mit Qubits in Superpositionen

weiterrechnen und so Quantenparallelismus erhalten.

Theoretisch kann man Qubits auch in andere Basen als $|0\rangle$ und $|1\rangle$ angeben und messen. Sämtliche Eigenschaften bleiben erhalten, aber der Einfachheit halber beschränkt man sich üblicherweise auf $|0\rangle$ und $|1\rangle$.

Betrachtet man ein System aus mehreren Qubits, bezeichnet man dieses als Quantenregister und stellt es gut lesbar in einer Formel dar.

Zum Beispiel besteht das Quantenregister

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$$

mit

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$$

aus zwei Qubits.

Verallgemeinert ist also das Quantenregister aus n Qubits durch

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} c_x |x\rangle, \quad \sum_{x \in \{0,1\}^n} |c_x|^2 = 1$$

definiert und hat eine Gesamtanzahl von 2^n möglichen Zuständen.

3. QUANTENSCHALTKREISE

Quantengatter sind analog zu klassischen Logikgattern die kleinsten Bausteine für Schaltungen.

Einige Quantengatter können über das Generalisieren von klassischen Gattern erzeugt werden.

Eines der einfachsten Schaltkreise ist das NOT-Gatter, das als Qubit folgendermaßen aussieht:

$$NOT(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle$$

Es ist eine Verallgemeinerung des klassischen NOT-Gatters, denn offensichtlich gelten

$$\begin{aligned} NOT(|0\rangle) &= |1\rangle \\ NOT(|1\rangle) &= |0\rangle \end{aligned}$$

und der Tausch der Komponenten verändert die Länge des Vektors nicht, sodass auch die Gesamtwahrscheinlichkeit von $1 = |\alpha|^2 + |\beta|^2$ erhalten bleibt.

Im Falle von klassischen Gattern reicht eine Wahrheitstabelle, aber für Quantenrechner ist dieses aufgrund der überabzählbaren Zustandskombinationen nicht mehr möglich. Stattdessen werden Gatter über eine Matrix definiert, sodass die Anwendung als Multiplikation mit dem Vektor des Quantenregisters beschrieben wird.

Aufgrund von Eigenschaften, deren Umfang über diese Ausarbeitung hinaus gehen, interagieren Quantensysteme immer in linearer Weise. Deswegen ist die Darstellung als Matrix ausreichend [1].

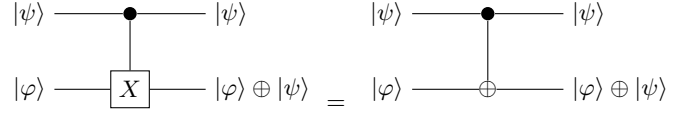


Figure 1: Verschiedene Darstellung des CNOT-Gatters als Schaltplan. CNOT wird als NOT-Gatter gezeichnet, auf das noch ein weiteres Kontroll-Bit einwirkt. Da CNOT ein sehr häufiges Gatter ist, gibt es ein spezielles Symbol, das rechts dargestellt wird.

Diese Matrizen müssen die Länge des Vektors erhalten, damit sich die Wahrscheinlichkeiten auch nach der Anwendung noch auf 1 aufsummieren.

Dafür müssen die Matrizen unitär sein. Eine Matrix U ist unitär, wenn für die konjugiert transponierte Matrix U^\dagger das Inverse ist, also $U^\dagger \cdot U = U \cdot U^\dagger = I$, wobei I die Einheitsmatrix entsprechender Größe bezeichnet. Dies bedeutet auch, dass es zu jedem Quantengatter ein Inverses gibt und somit jede Berechnung umkehrbar ist.

Wenn man die Anwendung des NOT-Gatter als Matrix-Vektor-Multiplikation schreibt, erhält man:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

Offensichtlich ist $NOT^\dagger = NOT$ das Inverse und somit ist die Matrix unitär.

Da jedes Gatter invertiert werden kann, müssen Eingabe- und Ausgabegröße gleich sein. Aufgrund dieser Einschränkung ist es nicht möglich, Bits zu kopieren, wie es im Fall klassischer Bits durch Verbinden mehrerer Kabel sehr einfach möglich ist. Diese Operation ist auf Quantenrechnern im Allgemeinen nicht mehr möglich und wird in Abschnitt 4 noch genauer untersucht.

Ebenso sind auch bestimmte grundlegende Operationen wie AND, OR und XOR nicht ohne weiteres möglich. Dies lässt sich aber über Einfügen weiterer Eingaben, sogenannter Kontrollbits, umgehen.

Ein Beispiel ist das CNOT. Es ist eine Verallgemeinerung des XOR-Gatters, stellt die Funktion

$$|\psi, \varphi\rangle \rightarrow |\psi, \varphi \oplus \psi\rangle$$

dar. In Abbildung 1 wird das CNOT-Gatter als Schaltplan dargestellt. Als Matrix dargestellt ist CNOT

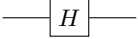
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

CNOT ist unitär und es gilt $CNOT^\dagger = CNOT$.

Ähnliche Generalisierungen sind auch für andere Gatter möglich. In Abschnitt 5 wird gezeigt, wie jedes klassische Gatter über eine bestimmte Schaltung simuliert werden kann.

Für gewöhnlich werden bei Quantenalgorithmen nur azyklische Gatter betrachtet.

Ein weiteres, wichtiges Gatter für viele Algorithmen ist das Hadamard-Gatter

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$


Auf den Eingaben $|0\rangle$ und $|1\rangle$ mischt das Hadamard-Gatter die beiden Zustände zu gleichen Anteilen.

$$|+\rangle = H \cdot |0\rangle = H \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$|-\rangle = H \cdot |1\rangle = H \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Beim Messen von $|+\rangle$ und $|-\rangle$ erhält man 0 und 1 mit jeweils gleichen Wahrscheinlichkeiten, aber man kann $|+\rangle$ und $|-\rangle$ gebrauchen, um mit beiden Zuständen gleichzeitig zu rechnen und so Quantenparallelismus ausnutzen zu können. In Abschnitt 6 wird dies auch an einem Beispiyalgorithmus erklärt.

4. NO CLONING THEOREM

In klassischen Schaltkreisen kann ein Bit einfach kopiert werden indem man mehrere Kabel an einer Ausgabe befestigt. Dies ist bei Quantenbits nicht mehr möglich. Es gibt keinen Weg, um im Allgemeinen ein Bit zu klonen.

Jedes Quantengatter ist reversibel, da die zugehörige Matrix unitär sein muss. Eine Klon-Funktion wie $|\psi, \varphi\rangle \rightarrow |\psi, \psi\rangle$ ist nicht reversibel und damit ist Klonen von Quanten über Quantengatter nicht möglich. Dies ist auch über andere Effekte nicht möglich, denn es ist fundamentale Eigenschaft von Quanten, dass sie nicht exakt kopiert werden können, die über diese Ausarbeitung hinaus geht [1].

Diese Aussage kann wie ein Widerspruch mit dem vorher definierten CNOT aussehen, denn für $\varphi = |0\rangle$ erhält man genau $|\psi, 0\rangle \rightarrow |\psi, \psi\rangle$.

Es ist allerdings doch kein Widerspruch, wie recht einfach gezeigt werden kann: Für $\psi = \alpha|0\rangle + \beta|1\rangle$ ergibt die Anwendung des vorherigen CNOT-Gatters genau die Ausgabe

$$\alpha|00\rangle + \beta|11\rangle$$

aber für zwei komplett unabhängige Quanten mit Wert ψ erhält man

$$|\psi\psi\rangle = \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle$$

Dieses ist nur ein Klon, wenn $a = 0$ oder $b = 0$ gilt, also die Qubits nur in den Zuständen $\pm|0\rangle$ oder $\pm|1\rangle$ sind. Spezifische Bits können also sehr wohl geklont werden, allerdings übersteigt der dabei kopierte Informationsgehalt nicht denen klassischer Bits.

5. SIMULATION KLASSISCHER COMPUTER

Das Toffoli-Gatter spielt eine zentrale Rolle in der Simulation klassischer Computer. Es berechnet die Funktion.

$$|a, b, c\rangle \rightarrow |a, b, c \oplus (a \wedge b)\rangle$$

Die entsprechende Matrix ist sehr groß und nur wenig instruktiv. Stattdessen ist die Funktionsweise an der Zeichnung 2 verdeutlicht.

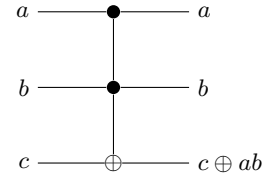


Figure 2: Toffoli-Gatter

Das Toffoli-Gatter ist unitär. Es ist sein eigenes Inverses, denn zweimal Anwenden ergibt

$$|a, b, c \oplus (a \wedge b) \oplus (a \wedge b)\rangle = |a, b, c\rangle$$

Eine Menge von Toffoli-Gattern kann jeden klassischen Schaltkreis simulieren. Für diese werden nur die Qubits $|0\rangle$ und $|1\rangle$ verwendet, aber keine weiteren Quantenzustände.

Als erstes zeigen wir, dass das Toffoli-Gatter zum Kopieren dieser Bits eingesetzt werden kann. In Abschnitt 4 haben wir gezeigt, dass das CNOT-Gatter trotz des *no-cloning-theorems* klassische Bits klonen kann und dieses ist durch das Toffoli-Gatter ebenso möglich, indem man die Eingangsbits entsprechend setzt. Durch die Wahl $a = |1\rangle$, $c = |0\rangle$, erhält man die Funktion $|1, b, 0\rangle \rightarrow |1, b, b\rangle$,

Wenn man stattdessen $c = |1\rangle$ setzt, erhält man $|a, b, 1\rangle \rightarrow |a, b, \neg(a \wedge b)\rangle$ und kann über dieses das NAND-Gatter simulieren. NAND ist in der klassischen Logik ein universelles Gatter, d.h. jede andere logische Funktion kann über Folgen von NANDs realisiert werden.

Mit dem NAND-Gatter und der Fähigkeit zu kopieren kann dann jeder klassischer Schaltkreis und damit auch klassischer Computer nur mit Toffoli-Gatter simuliert werden. Dafür werden nur konstant viele Quantengatter benötigt, wodurch sich die Komplexität der Berechnungen nicht ändert.

6. DEUTSCHS ALGORITHMUS

Deutschs Algorithmus ist ein einfacher Algorithmus, der Quantenparallelismus zeigen kann. Der von Deutsch vorgestellte Algorithmus ist randomisiert, aber hier wird eine vereinfachte Variante gezeigt, die das gleiche Problem deterministisch lösen kann.

Sei dazu $f(x) : \{0, 1\} \rightarrow \{0, 1\}$ eine beliebige Funktion. f ist unbekannt, es kann aber ein Quantenschaltkreis dazu konstruiert werden.

Deutschs Algorithmus testet nun, ob $f(0) \neq f(1)$, bzw. berechnet $f(1) \oplus f(0)$. Dieses Problem könnte ein klassischer Rechner durch Auswertung der beiden Funktionsaufrufe sehr einfach entscheiden, aber hier werden Eigenschaften der Quantenmechanik benutzt, um f nur einmal auszuwerten.

Dazu wird als erstes ein Quantenschaltkreis U_f auf Zweibit-registern konstruiert, der die Funktion $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$ umsetzt.

Durch $y = |0\rangle$ kann offensichtlich $f(x)$ berechnet werden. Mit Hilfe eines Hadamard-Gatters kann man nun $x = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ setzen und wendet anschließend U_f an. Das Ergebnis ist

$$|x, f(x)\rangle = \frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$$

Hier kann man bereits sehen, dass ein einzelner Schaltkreis ausreicht, um über Quantenparallelismus sowohl $f(0)$, als auch $f(1)$ zu berechnen, allerdings kann man dieses Ergebnis noch nicht nutzen, da man beim Messen der Ausgabe jeweils nur $|0, f(0)\rangle$ oder $|1, f(1)\rangle$ erhält.

In Deutschs Algorithmus wird nun eine weitere Eigenschaft der Quantenmechanik, die sogenannte Interferenz genutzt. Dafür wird über ein weiteres Hadamard-Gatter y auf $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$ gesetzt. Anwenden von U_f ergibt

$$U_f \cdot \left(|x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \right) = (-1)^{f(x)} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \quad (1)$$

Nach diesen Vorüberlegungen kann der eigentliche Algorithmus beginnen. Er ist im Schaltplan in Abbildung 3 dargestellt.

Als erstes wird ein Quantenregister $|01\rangle$ über zwei Hadamard-Gatter zu

$$|\psi_1\rangle = \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \quad (2)$$

transformiert.

Anwendung von U_f ergibt

$$|\psi_2\rangle = \begin{cases} \pm \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{für } f(0) = f(1) \\ \pm \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{für } f(0) \neq f(1) \end{cases} \quad (3)$$

Anwendung eines weiteren Hadamard-Gatters ergibt

$$|\psi_3\rangle = \begin{cases} \pm |0\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{für } f(0) = f(1) \\ \pm |1\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{für } f(0) \neq f(1) \end{cases} \quad (4)$$

$$= \pm |f(0) \oplus f(1)\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

Durch Messung des ersten Qubits erhält man nun

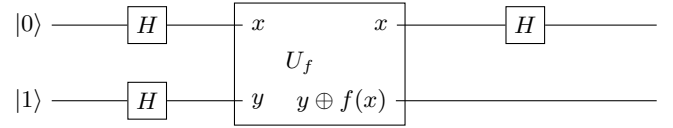


Figure 3: Deutschs Algorithmus als Schaltplan. Die Eingabewerte werden über Hadamard-Gatter transformiert, bevor die Funktion f ausgerechnet wird.

$f(0) \oplus f(1)$, wobei f während des ganzen Durchganges nur einmal ausgewertet wurde.

7. DEUTSCH-JOZSA ALGORITHMUS

Der Deutsch-Jozsa Algorithmus ist eine Verallgemeinerung von Deutschs Algorithmus. Gegeben eine n -bit Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$, entscheidet der Algorithmus, ob f eine konstante Funktion ist (immer den gleichen Wert zurückgibt) oder balanciert (in der Hälfte der Fälle 0, in der anderen 1). Andere Fälle treten nicht auf.

Die Lösung des Problems ist analog zu Deutschs Algorithmus, aber statt der Eingabe x werden n Qubits verwendet, die alle über Hadamard-Gatter zuerst in Superposition gebracht werden.

Obwohl das Problem an sich noch keinen realen Nutzen hat, existiert ein Quantenalgorithmus, der die Antwort in einer einfachen Auswertung von f unter Verwendung von $\Theta(n)$ vielen Qubits bestimmt, während eine klassische Maschine 2^n Auswertungen braucht.

8. KOMPLEXITÄT

Wie in Abschnitt 3 gezeigt wurde, kann jeder klassische Computer effizient simuliert werden.

Umgedreht kann auch ein Quantenalgorithmus auf einem klassischen Computer simuliert werden, indem die Komponenten der Quantenregister einzeln berechnet werden.

Bei der Berechenbarkeit gibt es also keinen Unterschied, aber wenn man die effizient berechenbaren Probleme betrachtet, ist der Quantenrechner über Quantenparallelismus möglicherweise effizienter.

Die Klasse BQP (bounded error quantum polynomial time) bezeichnet die Klasse der Probleme, die ein Quantenrechner in Polynomialzeit berechnen kann.

Wie in Abschnitt 5 gezeigt wurde, gilt $P \subseteq BQP$.

Quantenrechner verfügen über Zufallsquellen. Misst man ein Qubit im Zustand $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$, erhält man die Bits $|0\rangle$ und $|1\rangle$ mit einer Wahrscheinlichkeit von jeweils 0.5. Also können Quantencomputer auch randomisierte Algorithmen ohne Zeitverlust umsetzen und zusammen mit der effizienten Simulation von klassischen Computern folgt $BPP \subseteq BQP$.

Simons Problem stellt eine Fragestellung dar, die analog zu Deutschs Algorithmus zeigt, dass Quantenrechner auch bei randomisierten Algorithmen mächtiger sind als klassische Computer.

Zusätzlich kann kein Problem, das nicht in PSPACE liegt effizient in Polynomialzeit gelöst werden. Damit gilt also $BQP \subseteq PSPACE$ [1].

Es gilt $L \subsetneq PSPACE$, aber es ist nicht bekannt, welche der Inklusionen bei $L \subseteq P \subseteq BQP \subseteq PSPACE$ echte Teilmengen oder Gleichheiten sind.

Eine genauere Untersuchung des Verhältnisses zwischen BQP und P bzw. PSPACE kann auf die offenen Fragen noch weitere Antworten liefern, denn sollte $P \neq BQP$ gezeigt werden, dann folgt auch $P \neq PSPACE$ [1].

Andererseits ist die Frage $P = PSPACE$ aber auch oft untersucht worden und da bisher keine Antwort gefunden wurde, ist der entsprechende Beweis vermutlich sehr schwer.

Deswegen ist anzunehmen, dass die genaue Beziehung zwischen BQP und PSPACE auch noch länger offen bleibt. Ähnliches gilt für die Untergrenze, denn sollte $BQP = P$ gelten, dann gäbe es mit Shors Algorithmus auch einen Polynomialzeitalgorithmus für die Primfaktorzerlegung.

Die Beziehung zwischen NP und BQP ist nicht bekannt. Einige Probleme wie Primfaktorzerlegung können zwar in Polynomialzeit von einem Quantenrechner gelöst werden, aber bisher wurde noch für kein NP-hartes Problem ein effizienter Algorithmus angegeben [1].

9. ZUSAMMENFASSUNG UND AUSBLICK

Wir haben gesehen, dass es möglich ist, bestimmte Probleme über Quantenalgorithmen effizienter zu lösen.

Es sind bisher nur wenige Quantenalgorithmen verfügbar und die unintuitive Natur der Quantenmechanik erschwert es weitere zu finden.

Die genaue Beziehung zwischen BQP und P bzw. NP ist noch unklar. Insbesondere letzteres ist interessant, denn sollte $NP \subseteq BQP$ gelten, dann könnte man für viele relevante Probleme effiziente Lösungen entwickeln, unter der Annahme, dass irgendwann nutzbare Quantencomputer existieren werden.

Bisher wurden keine Quantencomputer in nutzbarer Größe gebaut. Allerdings wurden bereits Quantenalgorithmen erfolgreich an kleinen, experimentellen Systemen mit wenigen Qubits getestet.

Sollten allerdings funktionsfähige Systeme entwickelt werden, dann können durch steigendes Interesse und die Verfügbarkeit von Testsystemen möglicherweise noch viele neue Algorithmen entwickelt werden.

10. REFERENCES

- [1] A. Barenco. Quantum computation: an introduction. In H.-K. Lo, S. Popescu, and T. Spiller, editors, *Introduction to Quantum Computation Information*, chapter 10, pages 143–183. World Scientific Publishing Co., Inc., 1998.
- [2] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*, chapter 1. Cambridge University Press, 2000.