

Quanten Computing verstehen: Suchen und Grovers Algorithmus

Sabrina Cielas,¹ Till Pilarczyk²

Abstract: Suchprobleme können von klassischen Computern nicht effizient gelöst werden. Bei einer Suche mit Quantencomputer und dem Grover Algorithmus wird eine quadratische Beschleunigung erreicht. Der Grover Algorithmus setzt diese Suche in unstrukturierten Daten perfekt um. Dabei wird in Schritten die Amplitude des gesuchten Elementes gesteigert und alle anderen Amplituden gesenkt. Der Algorithmus hat verschiedene Varianten, so kann nach mehreren oder unbekannten vielen Lösungen und dem Minimum gesucht werden. Die Laufzeit wird dabei nicht schlechter. Mithilfe des Grover Algorithmus lassen sich jedoch nach aktuellem Stand keine NP Probleme effizient lösen. Daher wird der Quantencomputer mithilfe des Algorithmus nicht zu einem Allheilmittel.

Keywords: Quantencomputer; Quantum Computing; Grovers-Algorithmen

1 Einleitung

Viele Berechnungsprobleme sind in ihrer Essenz Suchprobleme: Optimierungsprobleme sind die Suche nach der optimalen Lösung, der Versuch eine kryptographische Verschlüsselung zu brechen ist die Suche nach dem korrekten Schlüssel. Um diese Probleme effizient lösen zu können, ergibt sich die große Relevanz der Suchalgorithmen.

Diese Arbeit ist im Rahmen des Moduls “Quantencomputer verstehen - Grundlagen und Anwendungen” an der Hochschule Düsseldorf bei Prof. Dr. Holger Schmidt entstanden. In dem Modul wurden zunächst gemeinsam die Grundlagen des Themengebiets Quantencomputer erarbeitet. Darauf aufbauend wurden durch die Studierenden Seminarvorträge vorbereitet, welche verschiedene Themengebiete tiefergehend beleuchten und im Kurs vorgestellt wurden. Dies ist die schriftliche Ausarbeitung zu dem Seminarvortrag “Suchen & Grovers Algorithmus”. Es werden die im Modul erarbeiteten Grundlagen als bekannt vorausgesetzt.

Diese Seminararbeit fokussiert sich auf den von Dr. Lov Grover entwickelten Quantensuchalgorithmus. Außerdem werden dessen verschiedene Varianten und die möglichen Anwendungsgebiete vorgestellt. Von zentralem Interesse ist dabei, ob Quantencomputer mit dem Grover-Suchalgorithmus zukünftig jene Probleme effizient lösen können, für die es mit klassischen Computern bisher keine geeigneten Lösungen gibt. Aus diesem Grund

¹ Hochschule Düsseldorf, Gebäude 4, Münsterstraße 156, 40476 Düsseldorf, Deutschland sabrina.cielas@study.hs-duesseldorf.de, Matrikelnummer: 771074

² Hochschule Düsseldorf, Gebäude 4, Münsterstraße 156, 40476 Düsseldorf, Deutschland till.pilarczyk@study.hs-duesseldorf.de, Matrikelnummer: 765335

findet die Laufzeit des Suchalgorithmus besondere Betrachtung, um im Fazit diese Frage angemessen beantworten zu können. Als Hauptquelle diene Kapitel 6 “Suchen” aus dem Buch “Quantum Computing verstehen: Grundlagen - Anwendungen - Perspektiven” von Matthias Homeister.

TODO: Aufbau der Arbeit

2 Aufgabenstellung

Das grundlegende Problem, welches im Folgenden betrachtet wird, ist die unstrukturierte Suche. Es soll ein gegebenes Datum aus einer Menge von Datensätzen gefunden werden. Die Datensätze sind dabei entweder gänzlich unsortiert oder nach einem Kriterium, welches für die Suche keine Relevanz besitzt. Die Suche kann mit einer Funktion f_x abgebildet werden, welche die Datensätze x als Eingabe akzeptiert. Der gesuchte Datensatz wird mit x_{DACH} bezeichnet. Ist der Datensatz x nicht der gesuchte, so gilt $f_x=0$, andernfalls $f_x DACH=1$. Die Datensätze sind dabei Teil der Menge N mit $N \gg 1$. Die Suche ist dann erfolgreich, wenn ein Element x_{DACH} gefunden wurde, für das $f_x DACH=1$ gilt.

Wird die Suche auf einem klassischen Rechner ausgeführt, so müsste dieser im schlechtesten Fall $f(x)$ N -mal auswerten. In diesem Fall wäre das gesuchte Element genau das letzte, welches aus der Menge N betrachtet wird. Dieser Fall ist jedoch sehr unwahrscheinlich und ein klassischer Rechner wertet $f(x)$ im Durchschnitt $(N+1)/2$ mal aus, bis er das gesuchte Element findet.

Diese Laufzeit lässt sich unter der Verwendung von einem Quantencomputer mit dem Grover-Algorithmus deutlich verbessern. Der Grovers-Algorithmus erreicht eine quadratische Beschleunigung der Suche, indem $f(x)$ nur noch \sqrt{N} -mal ausgewertet werden muss. Abbildung X zeigt anschaulich den Vergleich der verschiedenen Laufzeiten. Wie genau der Grovers-Algorithmus diese Beschleunigung erreicht wird in den folgenden Abschnitten erläutert.

3 Grovers-Algorithmus

Um den Algorithmus besser verstehen zu können werden die Rahmenbedingungen wie folgt formalisiert. Die Datenbank, besitzt N Elemente, wobei $N = 2^n$ entspricht. Den Datensätzen werden die Elementen $\{0, 1\}^n$ zugeordnet. Wenn $n = 2$ entspräche, besäße die Datenbank folgende Elemente: **00, 01, 10, 11**. Das gesuchte Element wird als \hat{x} bezeichnet. Die Datenbank wird als eine Funktion $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit folgender Eigenschaft umgesetzt:

$$f(x) = \begin{cases} 1 & \text{für } x = \hat{x} \\ 0 & \text{sonst} \end{cases}$$

Ist die Eingabe in die Funktion das gesuchte Element, so gibt diese **1** zurück, in allen anderen Fällen **0**. Mithilfe dieser Funktion wird das Quantenorakel. $U_f: |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$ erstellt. Das Quantenorakel, negiert das Vorzeichen des gesuchten Elements. Diese Funktionsweise ist aus dem Deutsch-Jozsa Algorithmus bekannt

3.1 Prinzip

Der Grover Algorithmus lässt sich in drei Schritte aufteilen.

1. **Superpositionen aufbauen**

Im ersten Schritt werden alle Quantenbits in die Superposition gebracht.

2. **Amplitudenveränderung durchführen (Grover Iteration G)**

Der zweite Schritt verändert die Amplituden der Elemente. Dabei wird die Amplitude des gesuchten Elementes erhöht und alle anderen verringert. Dieser Schritt wird auch *Grover Iteration (G)* genannt und abhängig von der Anzahl der Elementen öfters wiederholt. Wie oft G ausgeführt werden muss wird im Abschnitt 4.1.2 erläutert.

3. **Messen**

Im letzten Schritt werden die Quantenbits gemessen. Mit einer hohen Wahrscheinlichkeit kommt das gesuchte Element \hat{x} heraus.

3.1.1 Amplitudenveränderung

Die Amplitudenveränderung besteht aus zwei Schritten. Beim ersten Schritt wird das Vorzeichen der Amplitude von \hat{x} negiert. Im zweiten Schritt wird die negative Amplitude ausgenutzt um diese zu verstärken. Dies passiert in dem alle Amplituden am Mittelwert aller Amplituden gespiegelt werden.

Negieren der Amplitude

Um die Amplitude von \hat{x} zu negieren, wird ein Hilfsbit und das Quantenorakel benötigt. Das Hilfsbit wird in den Zustand $\mathbf{H}|1\rangle$ mithilfe eines Hadamar Gatters gebracht. Dadurch erhalten wird $|\mathbf{x}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Anschließend wird das Quantenorakel angewendet und damit das Vorzeichen der Amplitude von \hat{x} negiert. Das Hilfsbit wird nun nicht mehr benötigt und kann in folgenden Berechnungen weggelassen werden. Dies lässt sich auch an der Abbildung 1 erkennen. Dort ist Anwendung des Quantenorakel und graphisch die Amplituden aller Elemente der Datenbank mit $N = 4$ vor und nach der Anwendung des Quantenorakels zu sehen. Die negative Amplitude von \hat{x} hat keinen Einfluss auf das Messen. Schritt 2 wird benötigt, um mit einer erhöhten Wahrscheinlichkeit das Element \hat{x} nach dem Messen zu erhalten.

Spiegelung am Mittelwert

Um zu zeigen, dass die Spiegelung der Amplituden am Mittelwert den gewünschten Effekt hat, folgen nun ein paar Beispielrechnungen. Eine Spiegelung an einem Wert \mathbf{m} entspricht der Abbildung: $\alpha \rightarrow 2 \times \mathbf{m} - \alpha$.

Bei folgender Rechnung wird angenommen, dass die Datenbank vier Elemente enthält ($N = 4$). Der Mittelwert - nach der Negation von \hat{x} - hat den Wert:

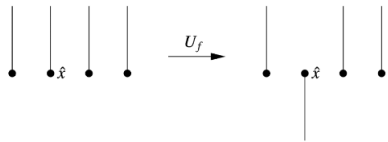
$$\begin{aligned}
|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) &\xrightarrow{U_f} |x\rangle \frac{1}{\sqrt{2}}(|f(x)\rangle - |1 \oplus f(x)\rangle) \\
&= |x\rangle (-1)^{f(x)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)
\end{aligned}$$


Abb. 1: Amplitudenveränderung

Quelle: [Ho17, S. 141]

$\mathbf{m} = \frac{1}{4} \times (\frac{1}{2} - \frac{1}{2} + \frac{1}{2} + \frac{1}{2}) = \frac{1}{4}$. Die Spiegelung von \hat{x} entspricht $-\frac{1}{2} \times \frac{1}{4} - (-\frac{1}{2}) = 1$. Damit ist die Amplitude des gesuchten Elements gleich eins. Das Messen würde mit einer Wahrscheinlichkeit von 100% das gesuchte Element \hat{x} zurückgeben. Die Amplituden aller anderen Elemente entwickeln sich wie folgt: $\frac{1}{2} \times \frac{1}{4} - \frac{1}{2} = 0$

Sei $N = 8$, so würde die Amplitude von \hat{x} nach der ersten Grover Iteration $\frac{5}{5\sqrt{8}}$ und alle anderen Amplituden $\frac{1}{2\sqrt{8}}$ betragen. Nach einer Spiegelung am Mittelwert ist die Amplituden von \hat{x} wieder positiv, bevor ein erneutes Spiegeln möglich ist, um die Amplituden weiter zu verstärken/verringern, muss erneut das Quantenorakel angewandt werden. Nach einer erneuten Negation und Spiegelung, hätte das gesuchte Element \hat{x} eine Amplitude von **0,973**, alle anderen Elemente eine von **-0,088**. Würde nun gemessen werden erhielte man mit einer Wahrscheinlichkeit von 93 % das gesuchte Element. Ein erneutes Spiegeln würde die Amplituden von \hat{x} im Gegensatz zu Erwartung wieder verringern und alle anderen erhöhen. Es ist daher besonders wichtig, dass nicht zu viele Grover Iterationen ausgeführt werden. Wie die Genaue Anzahl an Iterationen berechnet werden kann folgt im Abschnitt 4.1.2

3.1.2 Graphische Darstellung des Grover Algorithmus

In Abbildung 2 ist der Quantenschaltkreis des Grover Algorithmus nach bisherigen Erklärungsstand zu sehen. Alle QBits werden mithilfe der Hadamar Gatter in Superpositionen gebracht. Alles danach bis zum Messen am Ende der Abbildung ist die Grover Iteration. Der Äußere Kasten steht in der Abbildung für das mehrfache Wiederholen der Grover-Iteration. V_f steht für das Quantenorakel, jedoch wird hier das Hilfsbit nicht mit eingezeichnet. Anschließend folgt die Spiegelung am Mittelwert. Wie diese genau mithilfe von Gattern umgesetzt wird, folgt im nächsten Abschnitt (3.2). Nach dem ausführen der Grover Iterationen werden die QBits gemessen.



Abb. 2: Graphische Darstellung des Grover Algorithmus
Quelle: Anlehnung an [Ho17, S. 146]

3.2 Realisierung der Spiegelung am Mittelwert

Die Abbildung $\alpha \rightarrow 2 \times \mathbf{m} - \alpha$ lässt sich mithilfe folgender Matrixberechnung umsetzen.

$$\mathbf{D}_N \times \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{pmatrix}, \text{ mit } \mathbf{D}_N = \begin{pmatrix} -1 + \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & -1 + \frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & -1 + \frac{2}{N} \end{pmatrix}$$

Dies wird in Abschnitt 3.2.1 anhand einer Beispielrechnung verdeutlicht.

3.2.1 Beispielrechnung: Realisierung der Spiegelung am Mittelwert

Sei $N = 4$ so ergibt sich folgende Rechnung:

$$\mathbf{D}_4 \times \begin{pmatrix} 0,5 & -0,5 & 0,5 & 0,5 \end{pmatrix}^T.$$

$$\begin{pmatrix} -0,5 & 0,5 & 0,5 & 0,5 \\ 0,5 & -0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & -0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 & -0,5 \end{pmatrix} \times \begin{pmatrix} 0,5 \\ -0,5 \\ 0,5 \\ 0,5 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Dieses Ergebnis gleicht sich mit dem Ergebnis aus Abschnitt 3.1.1. In diesem wurden ebenfalls alle Elemente einer Datenbank mit $N = 4$ an dem Mittelwert der Amplituden gespiegelt. Folgende Abbildung 3 zeigt graphisch, wie sich die Amplituden verändert haben.

Wenn eine $N \times N$ Matrix verwendet wird, dann verstößt dies gegen das Lokalitätsprinzip.

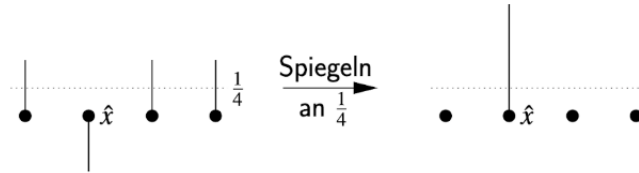


Abb. 3: Spiegelung am Mittelwert
Quelle: [Ho17, S. 142]

Daher muss die \mathbf{D}_n Matrix in verschiedene unitäre Matrizen zerlegt werden. \mathbf{D}_n kann in ein Produkt aus drei unitäre Matrizen zerlegt werden.

$$\mathbf{D}_n = -\mathbf{H}_n \times \mathbf{R}_N \times \mathbf{H}_n, \text{ mit } \mathbf{R} = \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

3.2.2 Beispielrechnung: Realisierung der Spiegelung am Mittelwert mithilfe von unitären Matrizen

Um zu zeigen dass die Matrix \mathbf{D}_N wie oben abgebildet als Produkt von drei Matrizen zerlegt werden kann, folgt ein Beispiel mit $N = 4$. Das Ergebnis der Berechnung ist, wie in der

$$\begin{aligned} \frac{1}{2} \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} &\times \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \\ &\quad \quad \quad -\mathbf{H}_2 \quad \quad \quad \mathbf{R}_4 \quad \quad \quad \text{Zwischenergebnis} \\ \\ \frac{1}{2} \begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} &\times \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} -2 & 2 & 2 & 2 \\ 2 & -2 & 2 & 2 \\ 2 & 2 & -2 & 2 \\ 2 & 2 & 2 & -2 \end{pmatrix} \\ &\quad \quad \quad \text{Zwischenergebnis} \quad \quad \quad \mathbf{H}_2 \quad \quad \quad \mathbf{D}_4 \end{aligned}$$

Abb. 4: Beispielzerlegung von \mathbf{D}_4
Quelle: Eigene Darstellung

beschriftetet Rechnung 4 zu sehen, gleich mit der zu erwarteten Matrix \mathbf{D}_4 . Der Beweis, das dies auch für beliebige N zutrifft, befindet sich in dem Buch vom M. Hoemeister [Ho17, S. 309].

3.2.3 Matrix \mathbf{R} als lokale Transformation

Es wurde gezeigt das die Matrix \mathbf{D}_N in ein Produkt aus Matrizen zerlegt werden kann. Das die Hadamar Matrix mit Hilfe eines Gatters als lokale Transformation umgesetzt werden kann, ist bekannt. Dies muss jedoch auch noch für \mathbf{R}_N gezeigt werden.

Um ein Gatter zu entwickeln zu können, welches die Transformation \mathbf{R}_N umsetzt, muss verstanden werden, wie sich \mathbf{R}_N bei einer Multiplikation von Matrizen auswirkt. Alle Werte einer Matrix die mit \mathbf{R}_N multipliziert wird bleiben gleich. Lediglich die erste Zeile oder Spalte wird negiert. Dies ist davon abhängig, ob die Matrix \mathbf{R}_N auf der Linken oder Rechten Seite der Multiplikation steht. In ersten Zeile der Abbildung 4 ist die Negation der ersten Spalte zu sehen.

Multipliziert man \mathbf{R}_N mit Amplituden, bedeutet dies, dass lediglich die Amplitude des ersten Elementes ($|0\dots 0\rangle$) negiert wird. Die Abbildung 5 zeigt den Quantenschaltkreis, wenn $N = 4$ gilt.

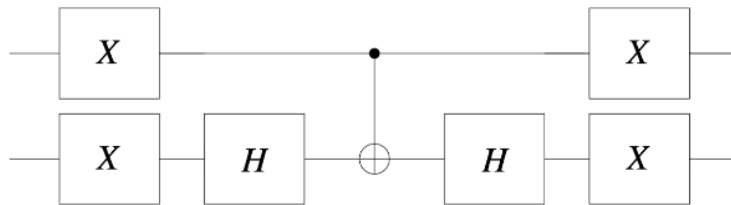


Abb. 5: \mathbf{R}_4 Realisierung
Quelle: [Ho17, S. 145]

Es folgen weitere Beispielrechnungen, um zu verdeutlichen, dass der Quantenschaltkreis die Transformation \mathbf{R}_4 ausführt.

3.2.4 Beispielrechnung: Matrix \mathbf{R} als lokale Transformation

In den Abbildungen 6 werden die Werte der QBits $|00\rangle$ und wie diese sich durch die einzelnen Gatter verändern dargestellt. Die Pauli X-Gatter invertieren den Wert eines QBits. Aus $|0\rangle$ wird $|1\rangle$ und andersherum. Das erste Bit wird bis zum letzten Pauli-X Gatter nicht mehr verändert. Es wird ausschließlich genutzt um zu schauen, ob das CNOT aktiviert wird. Das zweite QBit wird durch das erste Hadamar Gatter, in folgende Superposition $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ gebracht. Da das erste QBit den Wert $|1\rangle$ hat wird das zweite QBit durch das

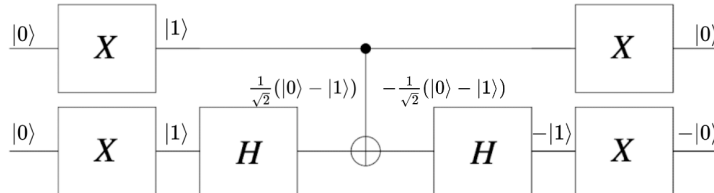


Abb. 6: $|00\rangle$ Werteveränderungen durch das R_4 -Gatter
Quelle: Anlehnung an [Ho17, S. 145]

CNOT negiert, und hat den Wert: $-\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Anschließend durch läuft das zweite Bit wieder ein Hadamar Gatter. Das QBit hat anschließend folgenden Wert: $-|1\rangle$. Zum Schluss werden beide Bits ($|1\rangle$ und $-|1\rangle$) durch die Pauli-X Gatter invertiert und wir erhalten das gewünschte und erwartete Ergebnis $-|00\rangle$.

Die Abbildung 7 zeigt die QBits $|01\rangle$ und wie diese von den Gattern verändert

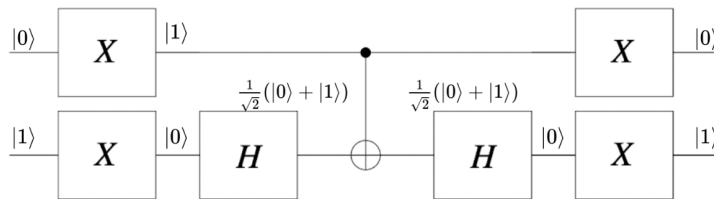


Abb. 7: $|00\rangle$ Werte Veränderungen durch das R_4 -Gatter
Quelle: Anlehnung an [Ho17, S. 145]

werden. Die ersten Pauli-X Gatter invertieren abermals die QBits, diese haben nun den Wert $|10\rangle$. Das erste QBit wird bis auf von dem letzten Pauli-X Gatter nicht verändert und nur zur Aktivierung der Operation CNOT zuhelfe genommen. Das zweite QBit wird durch das Hadamar Gatter in die Superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ gebracht. Diese verändert sich nicht durch die CNOT Operation. Nach der CNOT Operation wird das QBit durch das zweite Hadamar Gatter wieder in den Basiszustand $|0\rangle$ gebracht. Zuletzt durchlaufen die beiden QBits ein Pauli-X Gatter welches die QBits von den Basiszustände $|10\rangle$ in den erwarteten Zustand $|01\rangle$ überführt. Die QBits haben wie erwartet, nach dem durchlaufen des Quantenschaltkreises die selben Zustände wie vorher.

Bei den QBits $|10\rangle$ und $|11\rangle$, wird die CNOT Operation nicht ausgeführt, da das erste QBit in den Basiszustand $|0\rangle$ überführt werden. Die QBits werden durch die doppelte Ausführung der Gatter ebenfalls nicht verändert. Aus diesen Gründen ist für diese beiden

Beispiele keine Abbildung vorhanden.

An den Beispielen kann gesehen werden, dass der Quantenschaltkreis die gewünschte Operation R_4 ausführen.

3.3 Graphische Darstellung des Grover Algorithmus

In der Abbildung 8 ist wie in der Abbildung 2 der Quantenschaltkreis des Grover Algorithmus dargestellt. Die Abbildung enthält alle Gatter die für den Grover Algorithmus benötigt werden. Hilfsbits sind in der Abbildung nicht eingezeichnet.

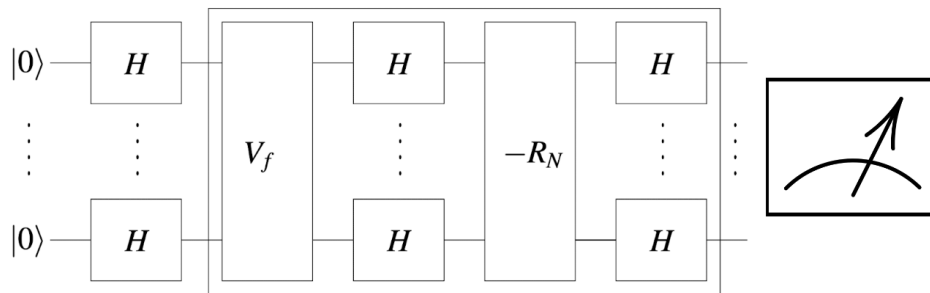


Abb. 8: Graphische Darstellung des Grover Algorithmus
Quelle: Anlehnung an [Ho17, S. 146]

4 Bestimmung der Anzahl an Grover-Iterationen

Nachdem der genaue Ablauf des Algorithmus erklärt wurde, bleibt die Frage: Wie oft muss die Grover Iteration durchgeführt werden, um mit einer hohen Wahrscheinlichkeit das gesuchte Element \hat{x} zu messen? noch offen.

4.1 Geometrische Veranschaulichung

Mithilfe von Regeln aus der Geometrie kann die Anzahl der Iterationen bestimmt werden. Das Schrittweise Erhöhen der Amplitude des gesuchten Elements kann auf der Bloch Kugel als Rotation aufgefasst werden.

In der Geometrie entspricht die Spiegelung eines Punktes an zwei Ebenen eine Drehung um den Winkel $2 \times \beta$, wobei β der Winkel zwischen den Ebenen ist. Dies ist in der Abbildung 9 verdeutlicht. Diese Eigenschaft kann sich zunutze gemacht werden. Ist der Winkel bekannt, kann ausgerechnet werden, um wie viel der Grad gedreht werden muss, um das gesuchte

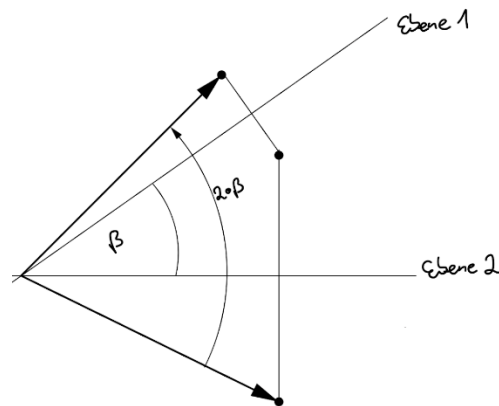


Abb. 9: R_4 Spiegelung an zwei Ebenen
 Quelle: Anlehnung an [Ho17, S. 149]

Element zu erreichen und damit wie oft die Grover-Iteration durchgeführt werden muss. Wie der Name schon sagt, handelt sich bei der Spiegelung am Mittelwert um eine Spiegelung. Die zweite Spiegelung ist das negieren von dem gesuchten Element \hat{x} . Die Formel zur Spiegelung an einem Wert \mathbf{m} lautet: $\alpha \rightarrow 2 \times \mathbf{m} - \alpha$. Ist $\mathbf{m} = \mathbf{0}$ erhalten wir $\alpha \rightarrow -\alpha$. Dies zeigt, dass die Negation eines Elementes, eine Spiegelung an dem Wert $\mathbf{0}$ ist.

Der Winkel lautet $\sin(\beta) = \langle \mathbf{s} | \hat{\mathbf{x}} \rangle$, mit $\mathbf{s} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |\mathbf{x}\rangle$, welches die Allgemeine Superposition ist. Ausrechnet ergibt dies $\sin(\beta) = \frac{1}{\sqrt{N}}$. Daran lässt sich erkennen, dass die Anzahl der Iterationen abhängig von der Anzahl der Datenbankelemente ist und nicht von $\hat{\mathbf{x}}$ ist. Wäre dies nicht so, hätte das zur Folge, dass für jedes gesuchte Element die Anzahl der Grover Iterationen angepasst werden werden müsste.

4.1.1 Beispielrechnung: Geometrische Veranschaulichung

Sei $N = 4$, dann folgt daraus, dass $\sin(\beta) = \frac{1}{\sqrt{4}}$ ist. Nach Beta aufgelöst ergibt sich $\beta = \frac{\pi}{6}$. Wird nun eine Grover Iteration durchgeführt, ändert sich der Winkel wie folgt: $\beta = \frac{\pi}{6} + 2 \times \frac{\pi}{6} = \frac{\pi}{2}$. Wird nun $\sin(\frac{\pi}{2})$ ausgerechnet, erhalten wir 1 . Dies bedeutet, dass wir mit einer Drehung das gesuchte Element erreicht haben. Wird nun gemessen erhalten wir mit einer Wahrscheinlichkeit von 100 % das gesuchte Element. Dies wird auch wie im Abschnitt 3.1.1 beschrieben erwartet.

4.1.2 Anzahl an Grover Iterationen

Durch die Rechnung konnte gezeigt werden, dass der Startwinkel $\frac{1}{\sqrt{N}}$ beträgt und der Winkel nach T Grover Iterationen den Wert $(2 \times T + 1) \times \frac{1}{\sqrt{N}}$ hat. Falls für $T = \frac{\pi}{4} \times \sqrt{N}$ gewählt wird, wird immer sehr nah an das gesuchten Element rotiert. In der Praktischen Umsetzung, muss T immer abgerundet werden, da nur eine gerade Zahl an Iterationen durchgeführt werden kann. Wird T aufgerundet, werden zu viele Grover-Iterationen durchgeführt und es wird sich vom gesuchten Element wieder entfernt. QUELLE DAS STEHT NICHT IN HOEHMEISTER

Dadurch dass die Anzahl der Grover-Iterationen $\frac{\pi}{4} \times \sqrt{N}$ beträgt und in jeder Iteration einmal das Quantenorakel aufgerufen wird, beträgt die Laufzeit des Grover Algorithmus $O(\sqrt{N})$

5 Varianten der Quantensuche

Grovers Suchalgorithmus lässt sich so anpassen, dass er auch auf ähnliche verwandte Probleme angewandt werden kann. Bisher wurde für den Algorithmus vorausgesetzt, dass f genau ein Element x_{DACH} auf 1 abbildet und die anderen Elemente auf 0. Diese Voraussetzung ist aber nicht bei jeder Suche gegeben. In den folgenden Abschnitten werden Suchalgorithmen betrachtet, bei denen es mehr als eine korrekte Lösung und unbekannt viele Lösungen gibt. Abschließend wird in Abschnitt x die Suche nach dem Minimum betrachtet.

5.1 Suche nach einer von mehreren Lösungen

Angenommen in der Menge der möglichen Lösungen N befindet sich nicht nur ein Element x_{DACH} , welches f auf 1 abbildet, sondern mehrere. Bei der Suche mit einem klassischen Computer verringert sich die Anzahl der Auswertungen von $f(x)$ entsprechend der Anzahl der vorhandenen Lösungen. So würde bei vier korrekten Lösungen in N nur noch ein Viertel der ursprünglichen Auswertungen zu erwarten sein. Auch bei der Quantensuche verringert sich der Aufwand entsprechend, ohne eine große Modifizierung am Algorithmus vorzunehmen. Für die Suche nach einer von mehreren Lösungen ist wieder ein Orakel $UVON_f$ gegeben, wobei die Funktion f genau k Elemente $x_{DACH1}, \dots, x_{DACHk}$ auf 1 abbildet. k ist somit die bekannte Anzahl der Lösungen in N , mit $N = 2^n$. Der Algorithmus besteht wieder aus den folgenden Schritten:

1. Superpositionen aufbauen

Es wird die gleichverteilte Superposition aufgebaut, indem alle Qubits in die Superposition gebracht werden: $R \leftarrow H_n 0 \dots 0$

2. Superpositionen aufbauen

Grover-Iteration auf R anwenden und so die Amplitudenveränderungen durchführen.
 $R \leftarrow -H_n E N H_n V_f x$ Die Grover-Iteration wird dabei $G(N,k)$ -mal ausgeführt

3. Superpositionen aufbauen

Das Quantenregister R wird gemessen und die Ergebnisse ausgegeben.

Der einzige Punkt, in dem sich dieser Algorithmus von dem ursprünglichen Grover-Algorithmus unterscheidet, ist im zweiten Schritt die Anzahl der Grover-Iteration. Im Regelfall wird die Grover-Iteration $T = \pi/4 \times \sqrt{N}$ durchgeführt. $G(N,k)$ ist jedoch definiert mit:

Damit reduziert die Anzahl der vorhandenen Lösungen in N die Anzahl der benötigten Grover-Iterationen. Für den Fall, dass $k \geq 3/4 N$ kann ein zufälliges Element x aus N gewählt und $f(x)$ ausgewertet werden. Mit einer 75 prozentigen Wahrscheinlichkeit wird $f(x)$ zu 1 ausgewertet. Laufzeit

5.2 Suche nach unbekannt vielen Lösungen

Eine größere Herausforderung stellt die Suche dar, wenn im Vorhinein nicht bekannt ist, wie viele Lösungen in der durchsuchten Menge N vorhanden sind. Bisher wurde die Anzahl der Lösungen benötigt, um die Anzahl der Grover-Iterationen zu bestimmen, damit durch die Amplitudenveränderung mit großer Wahrscheinlichkeit das gesuchte Element gefunden wird. Ist die Anzahl der richtigen Elemente vorher nicht bekannt, so muss die Anzahl der Iterationen auf eine andere Weise bestimmt werden. Dazu wurde der Grover-Algorithmus von Boyer, Brassard, Høyer und Tapp weiterentwickelt und wird dementsprechend als G-BBHT-Suche bezeichnet. Wie bei der Suche nach einer von mehreren Lösungen ist ein Orakel U_f gegeben, mit der Funktion f , die genau k Elemente $x_{DACH1}, \dots, x_{DACHk}$ auf 1 abbildet und die restlichen Elemente auf 0. N ist wieder gegeben durch $N \approx 2^n$, allerdings ist dieses mal nicht bekannt wie groß k ist. Der angepasste Algorithmus läuft folgendermaßen ab

1. Superpositionen aufbauen

Ein zufälliges Element x wird aus N ausgewählt. Wenn $f(x)$ zu 1 ausgewertet wurde erfolgreich eins der gesuchten Elemente gefunden und es kann an dieser Stelle gestoppt werden. Hintergrund ist, dass $k \geq 3/4 N$ sein könnte und in diesem Fall mit großer Wahrscheinlichkeit auch ohne Iterationen ein korrektes Ergebnis erreicht werden kann. Wertet $f(x)$ zu 0 aus, dann wird im 2. Schritt fortgefahren.

2. Superpositionen aufbauen

Es wird ein zufälliger Wert r ausgewählt mit r Element aus $1, \dots, \sqrt{N}$. Dieses r bestimmt die Anzahl der Iterationen in Schritt 4.

3. Superpositionen aufbauen

Es wird die gleichverteilte Superposition aufgebaut, indem alle Qubits in die Superposition gebracht werden: $R \leftarrow H_n 0 \dots 0$

4. Superpositionen aufbauen

Die Grover-Iteration wird r -mal auf R angewandt und so die Amplitudenveränderungen durchgeführt: $R \leftarrow -H_n R H_n V_f R$

5. Superpositionen aufbauen

Das Quantenregister R wird gemessen und $f(x)$ für das ausgegebene Ergebnis x ausgewertet. Wertet $f(x)$ zu 1 aus, so wurde eins der gesuchten Elemente gefunden und es kann an dieser Stelle gestoppt werden. Wertet $f(x)$ zu 0 aus, so wird wieder beim 1. Schritt begonnen.

Aus der Veröffentlichung von Boyer, Brassard, Høyer und Tapp geht hervor, dass bei Anwendung dieses Algorithmus nach einem Durchlauf eine Lösung mit einer Wahrscheinlichkeit von 25% gefunden wurde. Trotz der zufällig gewählten Anzahl von Iterationen beträgt die Laufzeit für diesen Algorithmus $O(\sqrt{N})$. Boyer, Brassard, Brassard und Tapp beschreiben eine weitere Modifizierung des Algorithmus, die es erlaubt die Laufzeit auf $O(\sqrt{N})$ zu senken. Dazu wird vor Beginn des Algorithmus ein $SIGMA$ element aus $\{1, \dots, 4/3\}$ gewählt und $m=1$ initialisiert. Die Anzahl der Iterationen r wird dann im 2. Schritt nicht mehr aus $\{1, \dots, \sqrt{N}\}$ gewählt, sondern aus $0, \dots, m$. Wertet $f(x)$ im 5. Schritt dann zu 0 aus, wird m mit dem Faktor $SIGMA$ multipliziert, sodass bei dem nächsten Durchlauf des Algorithmus im 2. Schritt r aus $\{0, \dots, SIGMA \cdot m\}$ gewählt wird. Im schlechtesten Fall wird so lange keine Lösung gefunden, dass $SIGMA \cdot m > \sqrt{N}$ gilt. Dann wird im 2. Schritt r wieder zufällig aus $\{0, \dots, \sqrt{N}\}$ gewählt, bis der Algorithmus terminiert. [QUELLE]

5.3 Suche nach dem Minimum

Bei den bisherigen Suchen wurde nur eine lokale Eigenschaft der Elemente berücksichtigt: handelt es sich bei dem aktuellen Element um jenes, welches ich suche? Vollkommen unabhängig von den anderen Elementen, die sich in N befinden. Bei der Suche nach dem Minimum ist jedoch eine globale Eigenschaft ausschlaggebend: Ist dieses Element kleiner, als alle anderen Elemente aus dem Feld? Zur Lösung dieses Problems haben 1996 Dürr und Høyer basierend auf der G-BBHT-Suche weiter gearbeitet und den folgenden Algorithmus entwickelt:

Gegeben ist ein Orakel U , welches auf ein Feld $T[0], \dots, T[N-1]$ zugreift. Dieses dient als Eingabe des Algorithmus, während das kleinste Element aus diesem Feld die Ausgabe des Algorithmus darstellt. N ist wieder gegeben durch 2^n . Für das Orakel werden zwei Quantenregister $x>$ und $y>$ der Länge n , sowie ein Hilfsbit $h>$ verwendet:

U formel und f formel

1. **Superpositionen aufbauen**
 Schrankenindex wählen Zu Beginn wird ein Anfangswert j für den Schrankenindex zufällig aus $0 \leq j \leq N-1$ ausgewählt. Dieser wird genutzt, um das Register y zu initialisieren: $y \leftarrow j$
2. **Superpositionen aufbauen**
 Nach Dürr und HoMITSTRICHyer werden die folgenden Schritte wiederholt, bis die Gesamtlaufzeit $22.5 \sqrt{N} + 1.4 \log[2, N]^2$ überschreitet [QUELLE]. Anschließend wird mit Schritt 2.3. fortgefahren:
 - a) **Superpositionen aufbauen**
 Es wird die gleichverteilte Superposition aufgebaut, indem alle Qubits in die Superposition gebracht werden: $x \leftarrow H_n 0 \dots 0$
 - b) **Superpositionen aufbauen**
 G-BBHT-Suche Auf dem Register x wird mit dem Quantenorakel U die G-BBHT-Suche ausgeführt. So wird aus allen Elementen des Feldes eines zufällig ausgewählt, welches kleiner als der Schrankenindex $T[y]$ ist.
 - c) **Superpositionen aufbauen**
 Das Quantenregister x wird gemessen. Gilt für das daraus resultierende Ergebnis i $T[i] < T[y]$, dann wird $y \leftarrow i$ gesetzt.
3. **Superpositionen aufbauen**
 Abschließend wird das Register y gemessen und das ausgegebene Ergebnis ist das kleinste Element des Feldes.

Der Algorithmus ist auf den ersten Blick, insbesondere durch die Verwendung der G-BBHT-Suche, recht abstrakt. Abbildung x soll das Verständnis durch eine vereinfachte, grafische Darstellung des Vorgehens vereinfachen. Mit einer Laufzeit von O findet dieser Algorithmus mit über 50 % das minimale Element des Feldes.

6 Anwendungen des Grovers Algorithmus

Wofür kann man den Algorithmus alles verwenden ?

7 Folgen für die Fähigkeit von Quantencomputern

Hier kommen die Folgen für die Fähigkeit von Quantencomputern hin

Literatur

[Ho17] Homeister, M.: Quantum Computing verstehen. Springer Vieweg Wiesbaden, 2017.