

# Video-Hausarbeit

## Einführung in Quantencomputing

**Adrian Uffmann<sup>1</sup>**

adrian.uffmann@campus.lmu.de

LMU München

Tuesday 4<sup>th</sup> August, 2020



Beispiel für solch einen Zustand in der Basis  $\{|+\rangle, |-\rangle\}$  (Hadamard Basis).

$$|\Psi\rangle = \hat{\alpha} \cdot |+\rangle + \hat{\beta} \cdot |-\rangle$$

$$\hat{\alpha}^2 = 20\% = 0,2 \Rightarrow \hat{\alpha} = \sqrt{0,2} = \frac{1}{\sqrt{5}}$$

$$|\hat{\beta}|^2 = 1 - |\hat{\alpha}|^2 \Rightarrow |\hat{\beta}| = \pm \sqrt{1 - |\hat{\alpha}|^2} = \pm \sqrt{1 - \left|\frac{1}{\sqrt{5}}\right|^2} = \pm \sqrt{1 - \frac{1}{5}} = \pm \frac{2}{\sqrt{5}}$$

$\Rightarrow$  Ein Beispiel für  $\hat{\beta}$  ist  $\frac{2}{\sqrt{5}}$

$\Rightarrow$  Ein Beispiel für  $\Psi$  ist  $\frac{1}{\sqrt{5}} |+\rangle + \frac{2}{\sqrt{5}} |-\rangle$

Für  $|\Phi\rangle = \frac{1}{\sqrt{5}}|+\rangle + \frac{2}{\sqrt{5}}|-\rangle$ :

$$\begin{aligned} H|\Phi\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{5}} + \frac{1}{\sqrt{2}} \cdot \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{5}} - \frac{1}{\sqrt{2}} \cdot \frac{2}{\sqrt{5}} \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{10}} \\ -\frac{1}{\sqrt{10}} \end{pmatrix} \\ &= \frac{3}{\sqrt{10}}|+\rangle - \frac{1}{\sqrt{10}}|-\rangle \\ \Rightarrow |\Phi\rangle &= \frac{3}{\sqrt{10}}|0\rangle - \frac{1}{\sqrt{10}}|1\rangle \end{aligned}$$

## Komplexität

*"Quantencomputer sind einfach schnellere Rechner, die Probleme beliebiger Komplexität im Handumdrehen lösen."*

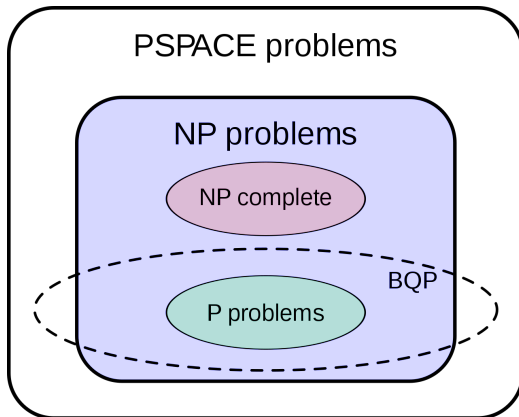
Das stimmt nicht.

Quantencomputer können alles berechnen, was normale Computer auch berechnen können, **aber** sie sind nicht bei jedem Problem schneller.

Für Quantenalgorithmen gibt es eine eigene Komplexitätsklasse.



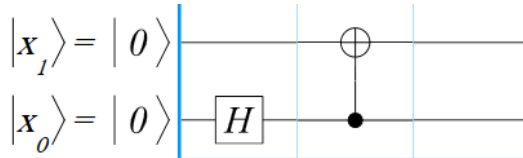
## BQP (*bounded error quantum polynomial time*)





## Bell Zustände

- Die Bell-Zustände sind verschränkte Zustände.
- lassen sich mit einem sehr einfachen Schaltkreis erzeugen.

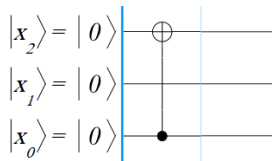
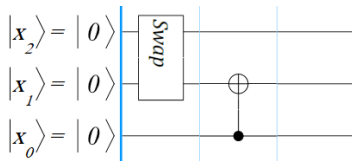


## Die vier Bell Zustände

- $|\beta_{00}\rangle = |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\beta_{10}\rangle = |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\beta_{01}\rangle = |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\beta_{11}\rangle = |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

## SWAP-Gatter logisch

- SWAP-Gatter tauschen die Zustände zweier QuBits.
- ⇒ SWAP-Gatter sind logisch gesehen nie nötig, denn man könnte auch mit nicht-ver tauschten QUBits weiter rechnen.



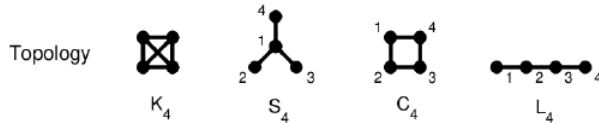




## SWAP-Gatter physikalisch

Echte Quantencomputer ...

- ... können kontrollierte Operationen nicht zwischen beliebigen Qubits machen.
- ... haben eine Netzwerktopologie.



Beispiel für Netzwerktopologien



## Grover's Algorithmus

- untersucht Funktionen der Form  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ .
- Ziel ist es, eine Eingabe zu finden, bei der die Funktion eine Ausgabe von 1 liefert.
- Funktioniert besser, wenn es weniger mögliche Ergebnisse gibt.
- Eignet sich für unser Problem, wenn wir eine Funktion finden, die nur für den kürzesten Pfad 1 liefert.

- Kann effizient Zahlen faktorisieren.
- Eignet sich um RSA zu brechen.
- Eignet sich nicht für unser Problem.

## Quanten Fourier Transformation

- die Quanten-Variante der DFT (*diskrete Fourier-Transformation*).
- Eignet sich um die Periode einer Funktion zu finden.
- Eignet sich nicht für unser Problem.



## Simon's Algorithmus

- untersucht Funktionen der Form  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ .
- Ziel ist es herauszufinden, ob die Funktion bijektiv ist, oder
- ob es einen Vektor  $s$  gibt, sodass  $f(x) = f(\hat{x}) \Leftrightarrow x \oplus s = \hat{x}$ .  
( $\oplus$  steht hier für element-weises XOR)
- Beispiel: Wenn  $f(001_2) = f(111_2)$  ist der gesuchte Vektor

$$s = 110_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \text{ denn } 001_2 \oplus 110_2 = \begin{pmatrix} 1 \oplus 0 \\ 0 \oplus 1 \\ 0 \oplus 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 111_2$$

- Eignet sich nicht für unser Problem.

## Überlegung: Optimale Funktion

- Mit Grover's Algorithmus erhalten wir für eine Funktion der Form  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  eine Eingabe, die eine Ausgabe von 1 erzeugt.
- ⇒ Optimal wäre eine Funktion, der man den Pfad als Eingabe gibt und die 1 liefert, genau dann, wenn es der kürzeste Pfad von  $S$  zu  $Z$  ist.
- nur leicht zu implementieren, wenn kürzester Pfad bereits bekannt.
  - Dazu mehr in Aufgabe 2f.

## Modellierung der Eingabe

Die Eingabe muss einen Pfad repräsentieren. Hier gibt es verschiedene Varianten:

1. Jedes Bit steht für einen Knoten (gesetzt bedeutet Teil des Pfades).  
⇒ Pfad nicht eindeutig, Knoten-Reihenfolge nicht klar.
2. Jedes Bit steht für eine Kante (gesetzt bedeutet Teil des Pfades).  
⇒ Pfad schwer zu validieren, da Kanten-Reihenfolge implizit.
3. Mehrere Bits kodieren einen Knoten, mehrere kodierte Knoten ergeben den Pfad.  
⇒ Leicht zu validieren, da Knote-Reihenfolge Teil der Eingabe.
4. Mehrere Bits kodieren eine Kante, mehrere kodierte Kanten ergeben den Pfad.  
⇒ Wie 3., aber es gibt mehr Kanten als Knoten, daher ineffizienter.

## Modellierung der Eingabe

Wir verwenden Variante 3.

Bei vier Knoten  $\{a, b, c, d\}$  reichen 2 Bit um einen Knoten zu kodieren:

$a = |00\rangle$ ,  $b = |01\rangle$ ,  $c = |10\rangle$  und  $d = |11\rangle$ .

Wir schreiben Knoten, die weiter vorne im Pfad sind auf die QBits mit niedrigerem Index (also weiter rechts).

Z. B. wird der Pfad  $b \rightarrow c \rightarrow a$  als  $|001001\rangle$  dargestellt.



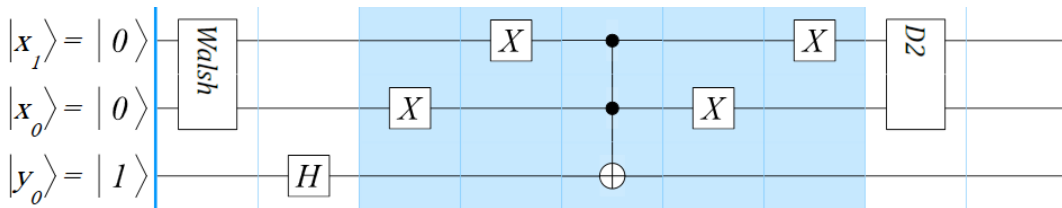
## Modellierung von Startknoten und Zielknoten

- Ein valider Pfad muss immer bei  $S$  beginnen und bei  $Z$  enden.
- ⇒ Wir müssen nur den Teilpfad zwischen  $S$  und  $Z$  betrachten.
- Es macht keinen Sinn, wenn  $S$  oder  $Z$  an anderer Stelle im Pfad auftauchen.
- ⇒ Wir brauchen keine Zahlen um  $S$  und  $Z$  kodieren zu können.

## Funktionsdefinition

- Der Graph hat nur einen kürzesten Pfad:  $S \rightarrow a \rightarrow Z$ .
- Dieser wird als  $|00\rangle$  dargestellt.

$$\Rightarrow f(x) = \begin{cases} 1 & x = |00\rangle \\ 0 & \text{otherwise.} \end{cases}$$



$U_f$  ist blau hinterlegt

Als Erinnerung  $f(x) = \begin{cases} 1 & x = |00\rangle \\ 0 & \text{otherwise.} \end{cases}$

Zustand vor  $U_f$ :

Value	Qubits	Probability	Amplitude
0>	000>	P = 0.125	0.35 + 0.00i
1>	001>	P = 0.125	-0.35 + 0.00i
2>	010>	P = 0.125	0.35 + 0.00i
3>	011>	P = 0.125	-0.35 + 0.00i
4>	100>	P = 0.125	0.35 + 0.00i
5>	101>	P = 0.125	-0.35 + 0.00i
6>	110>	P = 0.125	0.35 + 0.00i
7>	111>	P = 0.125	-0.35 + 0.00i

Zustand nach  $U_f$ :

Value	Qubits	Probability	Amplitude
0>	000>	P = 0.125	-0.35 + 0.00i
1>	001>	P = 0.125	0.35 + 0.00i
2>	010>	P = 0.125	0.35 + 0.00i
3>	011>	P = 0.125	-0.35 + 0.00i
4>	100>	P = 0.125	0.35 + 0.00i
5>	101>	P = 0.125	-0.35 + 0.00i
6>	110>	P = 0.125	0.35 + 0.00i
7>	111>	P = 0.125	-0.35 + 0.00i

## Ergebniszustand:

Value	Qubits	Probability	Amplitude
0>	000>	P = 0.5	-0.71 + 0.00i
1>	001>	P = 0.5	0.71 + 0.00i



## Kürzeste Wege

Es gibt 7 kürzeste Wege (alphabetisch sortiert):

1.  $S \rightarrow i \rightarrow a \rightarrow d \rightarrow Z$

2.  $S \rightarrow i \rightarrow a \rightarrow e \rightarrow Z$

3.  $S \rightarrow i \rightarrow a \rightarrow g \rightarrow Z$

4.  $S \rightarrow i \rightarrow h \rightarrow d \rightarrow Z$

5.  $S \rightarrow l \rightarrow c \rightarrow d \rightarrow Z$

6.  $S \rightarrow l \rightarrow c \rightarrow j \rightarrow Z$

7.  $S \rightarrow l \rightarrow h \rightarrow d \rightarrow Z$

## Abändern der Funktion

Dieser Graph hat 2 Unterschiede:

1. Es gibt mehr Knoten 12 und  $S$  und  $Z$ .

⇒ Es werden nun  $\lceil \log_2(12) \rceil = 4$ , statt 2 Bits pro Knoten benötigt.

2. Es gibt mehr kürzeste Pfade.

⇒ Bei allen muss die Funktion 1 liefern.

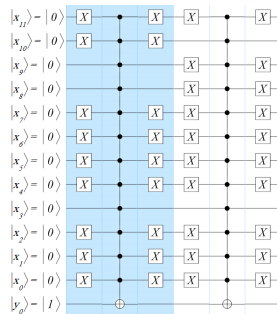
## Abändern des Schaltkreises

Eingabedarstellung der zwei ersten kürzesten Pfade:

1.  $S \rightarrow i \rightarrow a \rightarrow d \rightarrow Z \equiv |001100001000\rangle$
2.  $S \rightarrow i \rightarrow a \rightarrow e \rightarrow Z \equiv |010000001000\rangle$

Die Schaltkreise für die kürzesten Pfade können einfach hintereinander ausgeführt werden.

Aufeinander folgende X-Gatter können weggelassen werden.



Teile des Orakels: blau für Pfad 1,  
weiß für Pfad 2.

## Zustand nach $U_f$

- Ähnlich wie in 2d.
  - Das angepasste Orakel wird bei 7 statt nur einem Pfad aktiv.
- ⇒ Es sollten bei 14 statt nur bei 2 Zuständen der Superposition die Vorzeichen negiert worden sein.

Value	Qubits	Probability	Amplitude
0>	000>	P = 0.125	-0.35 + 0.00i
1>	001>	P = 0.125	0.35 + 0.00i
2>	010>	P = 0.125	0.35 + 0.00i
3>	011>	P = 0.125	-0.35 + 0.00i
4>	100>	P = 0.125	0.35 + 0.00i
5>	101>	P = 0.125	-0.35 + 0.00i
6>	110>	P = 0.125	0.35 + 0.00i
7>	111>	P = 0.125	-0.35 + 0.00i

Zustand nach  $U_f$  aus Aufgabe 2d



## Verallgemeinerung des Schaltkreises

- Bisherige Annahme: Funktion liefert 1, genau dann, wenn Eingabe der kürzeste Pfad von  $S$  zu  $Z$ .
  - **Aber:** Im Allgemeinen zu schwer, man kann nicht leicht prüfen, ob kürzerer Pfad existiert.
  - Wie beim Max-Cut-Problem reicht es, wenn die Funktion nur für valide Pfade mit bestimmter Länge eine 1 liefert.
- ⇒ Dann kann mit binärer Suche die kürzeste Länge bestimmt werden.

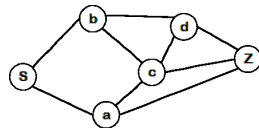
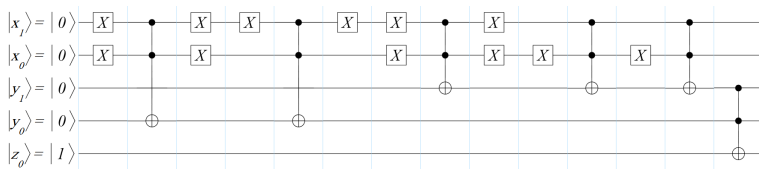
## Abbilden des Problems in eine analysierbare Funktion

- Unsere Funktion soll nur für valide Pfade mit bestimmter Länge eine 1 liefern.
  - Die Pfadlänge ist durch die Länge der Eingabe bereits festgelegt.
- ⇒ Die Funktion muss daher nur noch den Pfad validieren.

## Welche Eigenschaften hat ein valider Pfad?

1. Der erste Pfad-Knoten muss durch eine Kante mit  $S$  verbunden sein.
  2. Für aufeinander folgende Knoten muss der Graph eine Kante enthalten.
  3. Der letzte Pfad-Knoten muss durch eine Kante mit  $Z$  verbunden sein.
- Die Eigenschaften lassen sich leicht prüfen, indem für jedes aufeinander folgende Knotenpaar geprüft wird, ob es so eine Kante gibt.
  - Wenn ja wird der Wert in einem Ancilla QuBit gespeichert.
  - Die Ancilla QBits können dann mit einem mehrfachen Toffoli verundet werden.
- ⇒ Es werden so viele Ancilla QuBits benötigt, wie der Pfad Kanten hat.

## Allgemeines Orakel für Problem aus 2b



- $y_0$ : gibt es eine Kante zwischen  $S$  und dem ersten (einzigen) Knoten des Pfades.
- $y_1$ : gibt es eine Kante zwischen dem letzten (einzigen) Knoten des Pfades und  $Z$ .
- Anschließend werden die Ancilla Qubits wieder entschränkt, indem die Operationen (außer das letzte Toffoli) noch einmal rückwärts ausgeführt werden.

## Laufzeit

Die Laufzeit hängt von der Anzahl der Knoten  $V$ , der Anzahl der Kanten  $E$  und der Länge des zu untersuchenden Pfades  $|P|$  ab.

Das Orakel aus 2f benötigt für jede Kante des Pfades eine zu der Anzahl der Knoten logarithmische Anzahl an Operationen pro Kante des Graphes. Das Orakel hat also eine Komplexität von  $O(|P| \cdot |E| \cdot \log(|V|))$ .

Dies muss man noch multiplizieren mit der Anzahl der Aufrufe des Orakels.

Diese hängt logarithmisch von der Größe des Suchraumes der binären Suche und von der optimalen Anzahl von Grover-Iterationen ab.

## Laufzeit

Die binäre Suche liegt in  $O(\log(|V|))$ , da der kürzeste Pfad höchstens alle Knoten besucht.

Das Register  $x$ , auf dem Grover's Algorithmus agiert hat eine Größe  $n$  in  $O(|P| \cdot \log(|V|))$ , da  $O(|P|)$  Knoten in der Eingabe sind und jeder  $O(\log(|V|))$  QuBits für die Kodierung braucht.

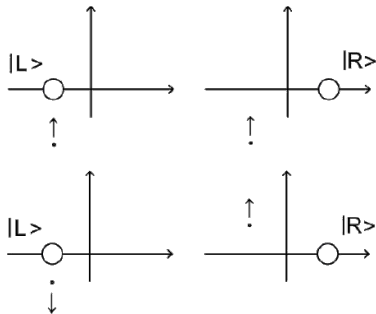
Die optimale Anzahl an Grover-Iterationen bei  $M$  möglichen Messergebnissen liegt in  $O(\sqrt{\frac{2^n}{M}})$ .

Es ist also nicht offensichtlich, ob der Algorithmus den klassischen Algorithmus von Dijkstra mit  $O(|E| + |V|\log(|V|))$  schlagen kann.

## Dekohärenz

- ist das Phänomen, dass QuBits mit der Umgebung interagieren können.

Beispiel aus der Vorlesung:



$$\frac{1}{\sqrt{2}}(|L\rangle + |R\rangle) |\uparrow\rangle = \frac{1}{\sqrt{2}}(|L\rangle |\uparrow\rangle + |R\rangle |\uparrow\rangle)$$

Interaktion  
→

$$\frac{1}{\sqrt{2}}(|L\rangle |\downarrow\rangle + |R\rangle |\uparrow\rangle)$$

Um Dekohärenz zu vermeiden werden Quantencomputer stark gekühlt und von äußeren Einflüssen isoliert.





## Bitflip vs. Phaseflip

- *Bitflip*
  - kennt man auch aus der klassischen Informatik.
  - an einer Stelle, an der eine 0 sein sollte, ist eine 1 (oder anders herum).
  - entspricht der Anwendung eines X-Gatters.
- *Phaseflip*
  - Die Amplitude von  $|1\rangle$  hat das falsche Vorzeichen (Minus statt Plus, oder anders herum).
  - entspricht der Anwendung eines Z-Gatters.

Da sowohl das X-Gatter, als auch das Z-Gatter ihre eigenen Inversen sind, können diese Gatter die Fehler auch wieder beheben.

## Dichte Kodierung