LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

Anwendung von Graphpartitionierungsalgorithmen und Genetischen Algorithmen zur Optimierung der Teleportationskosten in verteilten Quantenschaltkreisen

Teodor Slaveykov

Entwurf vom 9. Juni 2023

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

Anwendung von Graphpartitionierungsalgorithmen und Genetischen Algorithmen zur Optimierung der Teleportationskosten in verteilten Quantenschaltkreisen

Teodor Slaveykov

Aufgabensteller:	Prof. Dr. Claudia Linnhoff-Popien
Betreuer:	Leo Sünkel Thomas Gabor
Abgabetermin:	9. Juni 2023

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 9. Juni 2023

. . . . . .

(Unterschrift des Kandidaten)

Abstract

Derzeit befinden wir uns in der Noisy Intermediate Scale Quantum (NISQ) - Ära, in der die Anzahl der Qubits, die in einem einzelnen Quantencomputer verwendet werden können, zunimmt. Mit dieser Entwicklung entstehen jedoch Herausforderungen bei der Handhabung großer Quantensysteme. Die verteilte Quantenberechnung gewinnt daher an Bedeutung, um diese Herausforderungen zu bewältigen. Dabei werden mehrere Quantencomputer oder Quantenverarbeitungseinheiten miteinander verbunden, um gemeinsam an einem Problem zu arbeiten. Dies ermöglicht die Nutzung größerer Rechenkapazitäten und effizientere Lösungen komplexer Aufgaben. In der verteilten Quantenberechnung kommunizieren verschiedene Einheiten oder Teilsysteme miteinander, um Quanteninformation auszutauschen. Dabei spielt das grundlegende Teleportationsprotokoll eine wichtige Rolle. Es ermöglicht die Übertragung von Quanteninformationen zwischen den Teilsystemen. Ein wichtiger Aspekt besteht darin, die Anzahl der Teleportationen zu minimieren. Somit wird angestrebt, die Genauigkeit der Quantenberechnungen zu steigern, die Fehleranfälligkeit der Qubits zu reduzieren und gleichzeitig den Ressourcenverbrauch effizienter zu gestalten. In dieser Arbeit werden verschiedene Graphpartitionierungsalgorithmen, wie der Kernighan-Lin-Algorithmus und die Spektrale Partitionierung, ein Genetischer Algorithmus (GA) sowie zwei hybride Genetische Algorithmen (HGA), die eine Kombination aus den Graphpartitionierungsalgorithmen und einem GA sind, angewendet und untersucht, um die Anzahl globaler Quantengatter und die damit verbundenen Teleportationskosten zu minimieren. Zunächst werden die Graphpartitionierungsalgorithmen verwendet, um die Knoten möglichst gleichmäßig zu partitionieren. Zusätzlich wird ein GA implementiert, der sich um die Aufteilung der Qubits mittels zufälliger Partitionen kümmert. Die beiden HGA führen zu einer nahezu optimalen Anordnung der globalen Quantengatter, nachdem die Qubits mithilfe der Graphpartitionierungsalgorithmen partitioniert sind. Schließlich werden die vorgeschlagenen Ansätze anhand von neun Benchmark-Schaltkreisen untersucht und hinsichtlich der Anzahl globaler Quantengatter und Teleportationskosten verglichen. Außerdem werden zufällige Suchläufe für den GA und der beiden HGA durchgeführt, um deren Leistungsfähigkeit in Bezug auf das Optimierungsziel zu überprüfen. Die Ergebnisse deuten auf eine signifikante Verbesserung der Teleportationskosten hin.

Schlüsselwörter: verteilte Quantenberechnung, Quantenschaltkreis, Teleportationskosten, Graphpartitionierungsalgorithmen, Genetische Algorithmen

Inhaltsverzeichnis

1 Einleitung **1**

2 Theoretische Grundlagen **3**
2.1 Quantenberechnung . . . 3
2.1.1 Qubit . . . 3
2.1.2 Quantengatter . . . 3
2.1.3 Quantenschaltkreise . . . 5
2.1.4 Quantenverschränkung . . . 6
2.1.5 Quantenteleportation . . . 7
2.2 Verteilte Quantenberechnung . . . 9
2.2.1 Verteilte Quantenarchitekturen . . . 9
2.2.2 Verteilte Quantenschaltkreise . . . 9
2.3 Genetische Algorithmen . . . 10
2.3.1 Kodierung der Chromosomen . . . 10
2.3.2 Fitnessfunktion . . . 11
2.3.3 Selektion . . . 11
2.3.4 Kreuzung . . . 12
2.3.5 Mutation . . . 13
2.3.6 Ersetzungsschema . . . 13
2.3.7 Abbruchskriterien . . . 13
2.3.8 Exploitation gegen Exploration . . . 14
2.4 Graphpartitionierungsalgorithmen . . . 15
2.4.1 Kernighan-Lin-Algorithmus . . . 15
2.4.2 Spektrale Partitionierung . . . 16
2.4.3 Graphische Darstellung eines Quantenschaltkreises . . . 18

3 Verwandte Arbeiten **21**

4 Methodik **23**
4.1 Strategie zur Bestimmung der Teleportationskosten . . . 23
4.2 Graphpartitionierungsalgorithmen . . . 26
4.2.1 Kernighan-Lin-Algorithmus . . . 26
4.2.2 Spektrale Partitionierung . . . 29
4.3 Genetischer Algorithmus . . . 30
4.3.1 Kodierung der Chromosomen und Anfangspopulation . . . 30
4.3.2 Fitness Funktion und Selektionsstrategie . . . 30
4.3.3 Kreuzung und Mutation . . . 31
4.4 Hybridgenetische Algorithmen . . . 32

5 Ergebnisse und Diskussion **35**

6 Fazit **43**

Abbildungsverzeichnis **45**

Tabellenverzeichnis **47**

Literaturverzeichnis **49**

1 Einleitung

Quantenberechnung hat die Informatik revolutioniert, indem sie gezeigt hat, dass die Verarbeitung von Quantenzuständen im Vergleich zu herkömmlichen Algorithmen, die mit klassischen Bits arbeiten, zu einer signifikanten Beschleunigung bei der Lösung bestimmter Problemtypen führen kann [Sho99] [Gro96] [Gro97]. Daher hat die Quantenberechnung das Potenzial, den gesamten aktuellen Markt und die Industrie zu transformieren.

Im Jahr 2017 startete die Europäische Kommission ein Flagship-Programm mit einem Budget von einer Milliarde Euro, um die Quantenforschung für einen Zeitraum von zehn Jahren ab 2018 zu unterstützen. In den folgenden drei Jahren wurde bereits eine erste Tranche von 132 Millionen Euro zur Verfügung gestellt [RBTC17]. Im Jahr 2018 hingegen starteten die Vereinigten Staaten von Amerika die nationale Quanteninitiative mit einer Finanzierung von 1,2 Milliarden US-Dollar über einen Zeitraum von zehn Jahren. Auch China investiert Milliardenbeträge, um quantentechnologische Entwicklungen zu kommerzialisieren [Gib19].

Ende 2019 erzeugte Google einen Quantencomputer mit einem 54-Qubit-Quantenprozessor namens Sycamore [PGN+19]. Das Unternehmen schätzt, dass ein Supercomputer 10.000 Jahre brauchen würde, um die gleiche Arbeit zu berechnen, die Sycamore in wenigen Sekunden erledigt. Darüber hinaus arbeiten auch andere große Unternehmen wie Intel, IBM, Microsoft usw. aktiv an die Entwicklung von Quantencomputern.

Die Quantenberechnung bietet viele Vorteile, aber die Implementierung von großen Quantensystemen mit vielen Qubits ist mit Herausforderungen verbunden. Die Wechselwirkungen der Qubits mit der Umgebung führen zu Rauschen oder Dekohärenz [CCVMH20] und die Quanteninformation wird mit zunehmender Anzahl von Qubits empfindlicher und fehleranfälliger [CCT+19]. Tatsächlich ist Dekohärenz nicht die einzige Fehlerquelle in der Quantenberechnung. Fehler treten praktisch bei jeder Operation auf einem Quantenzustand auf. Jedoch ist die Isolierung der Qubits von der Umgebung allein nicht ausreichend, da die Qubits manipuliert werden müssen, um die Anforderungen an Kommunikation und Berechnung zu erfüllen, wie zum Beispiel Lese-/Schreiboperationen. Daher ist die Fehlerkorrektur in Quantensystemen wichtig, um die Robustheit und Zuverlässigkeit der Quanteninformation zu gewährleisten. Die Fehlerkorrektur in der Quantenberechnung zielt darauf ab, Fehler zu erkennen und zu korrigieren, die während der Durchführung von Operationen auf Qubits auftreten.

Darüber hinaus müssen zur Herstellung von Quantensystemen viele Qubits involviert sein, um ein Qubit zu korrigieren. Es ist daher möglich, dass alle Qubits nicht auf einem einzigen Chip passen [VMLFY10]. Eine vernünftige Lösung, um diese Probleme zu überwinden, besteht in der Anwendung von verteilten Quantencomputern. Ein verteilter Quantencomputer ist eine Sammlung von Quantencomputern bzw. Knoten, wobei jeder Quantencomputer in der Sammlung eine bestimmte Anzahl von Qubits hat, auf denen er arbeiten kann. Die Quantencomputer sind über ein Netzwerk miteinander verbunden, das ihnen ermöglicht, sowohl klassische als auch Quanteninformation zwischen sich auszutauschen. Die Idee hinter einem verteilten Quantencomputer besteht darin, die

Leistungsfähigkeit und Skalierbarkeit eines Quantencomputers zu erhöhen, indem mehrere Quantenverarbeitungseinheiten parallel zusammenarbeiten. Durch die Verteilung der Aufgaben auf mehrere Quantensysteme kann ein verteilter Quantencomputer größere Probleme lösen, als es ein einzelner Quantencomputer könnte.

Um ein verteiltes Quantensystem umzusetzen, benötigt man eine zuverlässige Kommunikationsverbindung zwischen den Teilsystemen. Die langstreckige Quantenkommunikation stellt eine technologische Herausforderung für die physische Umsetzung der Quantenkommunikation dar [BLMS00]. In diesem Zusammenhang ist die Teleportation [BBC$^+$93] ein grundlegendes Protokoll zur Verteilung der verschränkten Qubits und mit Hilfe dieses Protokolls kann Quanteninformation durch Quantenverbindungen und -systeme verteilt werden.

Die Hauptidee der Teleportation besteht darin, die Zustände der Qubits von einem Ort zum anderen zu transformieren, ohne sie physisch zu bewegen [MMNI08]. Demzufolge werden nach der Übertragung der Qubitinformation alle Berechnungen lokal am Zielort durchgeführt.

In der Quantenberechnung existiert ein Qubit nach der Teleportation nicht mehr im ursprünglichen Subsystem und kann dort nicht mehr verwendet werden. Um dieses Qubit erneut im ursprünglichen System zu verwenden, muss es über einen Kommunikationskanal zurückgeführt werden. Die Teleportation ist folglich eine kostspielige Operation. Dementsprechend gewinnen Lösungen zur Minimierung der Anzahl von Teleportationen zunehmend an Bedeutung.

In dieser Arbeit wird die Optimierung der Teleportationskosten zwischen Subsystemen als Maßstab dafür betrachtet. Die vorliegende Arbeit baut auf den Ergebnissen von [HMZMH20] auf. Das Ziel der Arbeit ist die Minimierung der Anzahl globaler Quantengatter und der damit verbundenen Teleportationskosten. Zur Erreichung dieses Ziels werden Graphpartitionierungsalgorithmen wie der Kernighan-Lin-Algorithmus und die Spektrale Partitionierung verwendet, um die Qubits möglichst gleichmäßig in Partitionen aufzuteilen. Zusätzlich wird ein Genetischer Algorithmus (GA) als Teil dieser Arbeit implementiert, um die Aufteilung der Qubits mittels zufälliger Partitionen zu optimieren, sodass die Kommunikationskosten minimiert werden. Schließlich werden zwei hybride Genetische Algorithmen (HGA) vorgestellt, bei denen die Graphpartitionierungsalgorithmen zur Aufteilung der Qubits in Partitionen genutzt werden und daraufhin jeweils ein GA für eine optimale Anordnung der globalen Quantengatter sorgt. Die verschiedenen Ansätze werden anhand der Anzahl globaler Quantengatter und der Teleportationskosten verglichen und präsentiert.

Der Rest dieser Arbeit ist wie folgt strukturiert: In Kapitel 2 werden wichtige Grundlagen erläutert, die zur besseren Verständnis der Forschungsfrage beitragen. Kapitel 3 stellt verwandte Arbeiten vor. In Kapitel 4 werden die verschiedenen Ansätze zur Erreichung des Ziels beschrieben. Kapitel 5 enthält die Ergebnisse und Diskussion, in der die Ergebnisse interpretiert und bewertet werden. Im abschließenden Kapitel 6 werden alle vorangegangenen Kapitel zusammengefasst und es wird ein Schlusswort gegeben.

2 Theoretische Grundlagen

Im folgenden Kapitel werden die für den weiteren Verlauf der Arbeit benötigten Grundlagen gelegt. Insbesondere werden die wichtigsten Begriffe aus dem Bereich der Quantenberechnung und der verteilten Quantenberechnung erläutert. Des Weiteren wird näher auf die Bedeutung der verschiedenen Ansätze eingegangen, die im Laufe dieser Arbeit präsentiert werden.

2.1 Quantenberechnung

2.1.1 Qubit

Als **Qubit** bezeichnen wir, analog zu den Bits eines klassischen Computers,die kleinstmögliche Berechnungseinheit eines Quantencomputers. Ein klassisches Bit befindet sich stets entweder im Zustand 0 oder 1. Auch für ein Qubit gibt es nur diese beiden sogenannten Basiszustände, allerdings kann es sich in einer beliebigen Überlagerung beider Zustände befinden, man spricht von einer **Superposition** . Mögliche Zustände schreiben wir zukünftig in der **Dirac-Notation** als $|\psi\rangle$, die beiden Zustände eines klassischen Bits schreiben wir also als $|0\rangle$ und $|1\rangle$. Damit lässt sich jeder Zustand $|\psi\rangle$ eines Qubits notieren als $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ mit $\alpha, \beta \in \mathbb{C}$, wobei wir fordern, dass stets $|\alpha|^2 + |\beta|^2 = 1$. Die Zahlen, noch als **Amplituden** genannt, α und β beschreiben jeweils den Anteil der beiden Zustände $|0\rangle$ und $|1\rangle$ am Zustand des Qubits [Hom08].

Um den Wert eines Qubits zu erfahren, müssen wir es messen. Dadurch wird die Superposition zerstört, wodurch das Qubit in einen **nicht-überlagerten** Zustand übergeht, nämlich genau einen der beiden Basiszustände $|0\rangle$ und $|1\rangle$. Die Wahrscheinlichkeit für eine Messung von $|0\rangle$ ist dabei genau $|\alpha|^2$, für eine Messung von $|1\rangle$ entsprechend $|\beta|^2$. Den Zustand eines Quantenbits kann man auch als Vektor in einem zweidimensionalen Vektorraum über den komplexen Zahlen betrachten. Die Standardbasis dieses Vektorraums sieht so aus: $|0\rangle := \begin{pmatrix}1\\0\end{pmatrix}$, $|1\rangle := \begin{pmatrix}0\\1\end{pmatrix}$. Jeder Zustandsvektor kann als Linearkombination der Vektoren dieser Standardbasis geschrieben werden:

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

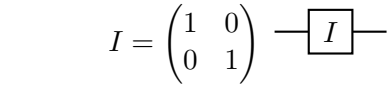
2.1.2 Quantengatter

Ein **Quantengatter** funktioniert wie ein klassisches Gatter und ändert den Zustand von einem oder mehreren Qubits bei Anwendung. Aber Quantenlogikgatter sind reversibel. Das bedeutet, dass die Anzahl der Qubits in der Eingabe gleich der in der Ausgabe ist. Ein n-Qubit-Quantengatter U wird durch eine $2^n \times 2^n$-Matrix definiert. Wenn U auf einen Quantenzustand $|\psi\rangle$ angewendet wird, ändert sich das Ergebnis zu einem anderen Zustand, der durch den Vektor $U|\psi\rangle$ dargestellt wird.

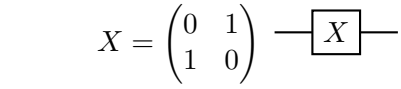
Das Schaltkreismodell der Quantenberechnung basiert auf der unitären Evolution von Qubits durch Netzwerke von Gattern. Jedes n-Qubit-Quantengatter repräsentiert eine lineare Transformation, die durch eine unitäre Matrix definiert wird, die auf einem n-Qubit-Hilbertraum definiert ist. Die Matrix U ist unitär, wenn $UU^\dagger = 1$, wobei $U^\dagger$ die konjugiert transponierte Matrix von U darstellt.

Es gibt eine Vielzahl von Quantengattern, die auf unterschiedliche Anzahlen von Qubits angewendet werden können. Einzelne Qubits können beispielsweise mit Pauli- und Rotationsgattern manipuliert werden, die häufig verwendet werden, um spezifische Zustände zu erzeugen oder Operationen durchzuführen. Pauli-Gatter wie das $X-$, $Y-$ und Z-Gatter ermöglichen es, einen Qubit-Zustand entlang der entsprechenden Achsen des Bloch-Sphärenmodells zu drehen. Rotationsgatter wie das R_x, R_y und R_z-Gatter ermöglichen es, den Zustand eines Qubits um einen bestimmten Winkel um die entsprechende Achse zu drehen. [BBC+95]:

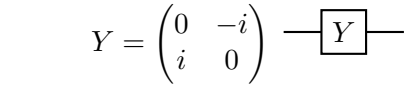
$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



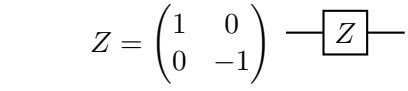
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



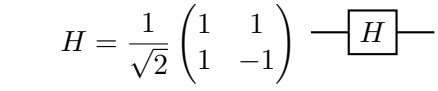
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$



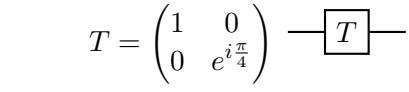
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



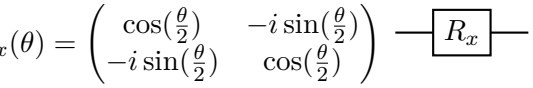
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



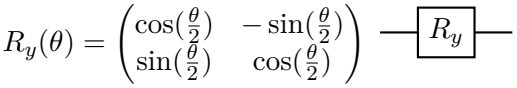
$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$



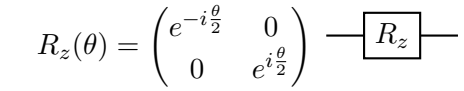
$$R_x(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$$



$$R_y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$$



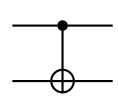
$$R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$



Wenn X ein Gatter ist, das auf einem einzelnen Qubit arbeitet, ist **Controlled-**X oder **Controlled-**NOT ein Gatter, das auf zwei Qubits arbeitet. Wenn das Steuer-Qubit $|1\rangle$ ist, wird X auf das Ziel-Qubit angewendet. Andernfalls bleibt das Ziel-Qubit unverändert

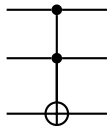
[NC01].

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



Das Toffoli-Gatter [Hom08] ist ein weiteres Beispiel für ein gesteuertes Quantengatter und wird auch als **Controlled-Controlled-NOT (CCNOT)** bezeichnet. Es arbeitet auf drei Qubits, wobei das erste und das zweite Qubit als Steuer-Qubits fungieren und das dritte Qubit als Ziel-Qubit. Wenn die beiden Steuer-Qubits in dem Zustand $|1\rangle$ sind, wird das Toffoli-Gatter auf das Ziel-Qubit angewendet. Andernfalls bleibt das Ziel-Qubit unverändert. Das Toffoli-Gatter ist universell, was bedeutet, dass es als Basisgatter für alle anderen Quantengatter verwendet werden kann.

$$\text{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



2.1.3 Quantenschaltkreise

Ein **Quantenschaltkreis** besteht aus Quantengattern, die über Quantendrähte miteinander verbunden sind und Qubits transportieren [NC01]. Die unitäre Matrix des Quantenschaltkreises wird entweder durch das Skalarprodukt oder das Tensorprodukt der unitären Matrizen der Quantengatter ermittelt [ZMHH18]. Der Gesamteffekt der Gatter, die nacheinander auf eine gleiche Teilmenge von Qubits angewendet werden, wird durch das Skalarprodukt berechnet, das der bekannten Matrixmultiplikation entspricht. Die benachbarten Gatter, die auf unabhängigen Teilmengen von Qubits wirken, können parallel angewendet werden und ihr Gesamteffekt wird durch ihr Tensorprodukt ermittelt und wie folgt definiert.

Sei A eine $m \times n$ Matrix und B eine $p \times q$ Matrix. Dann ist $A \otimes B$ eine $(mp) \times (nq)$ Matrix, die als Tensorprodukt (**Kronecker-Produkt**) von A und B bezeichnet wird und wie folgt definiert ist:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

wobei a_{ij} das Element in der i-ten Spalte und j-ten Zeile der Matrix A darstellt.

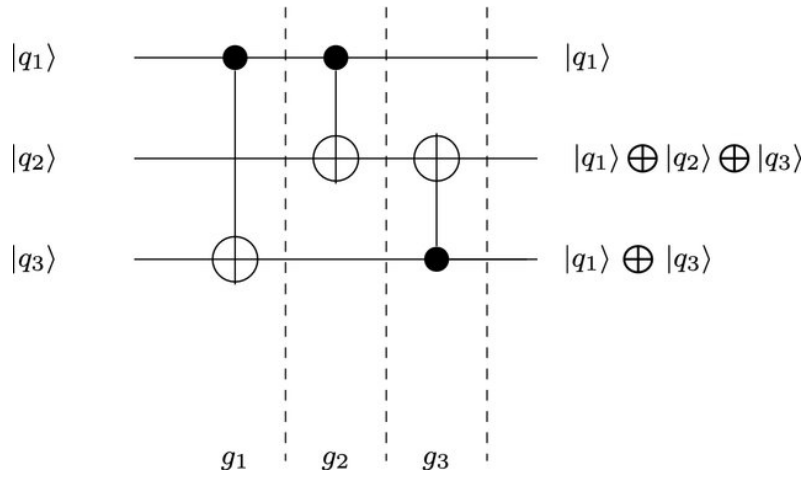


Abbildung 2.1: Quantenschaltkreis bestehend nur aus CNOT-Gattern [DZMHNb20]

2.1.4 Quantenverschränkung

Verschränkung ist ein quantenmechanisches Phänomen, das eine Schlüsselrolle bei vielen der interessantesten Anwendungen der Quantenberechnung und der Quanteninformation spielt. In einem verschränkten Zustand beeinflusst die Manipulation eines Bits das Ergebnis der Messung eines anderen. Ein Multi-Qubit-Quantenzustand $|v\rangle$ gilt als verschränkt, wenn er nicht als das Tensorprodukt $|v\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle$ von zwei reinen Zuständen geschrieben werden kann. Um dieses Konzept zu veranschaulichen, können wir uns folgenden Quantenschaltkreis mit zwei Qubits und zwei Quantengattern vorstellen:

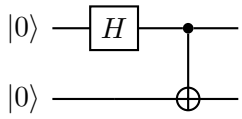


Abbildung 2.2: Verschränkung zweier Qubits

Auf das Quantenregister bestehend aus den beiden Qubits, die sich beide jeweils im Anfangszustand $|0\rangle$ befinden, werden Quantengatter angewendet. Das erste Qubit wird durch ein Hadamard-Gatter transformiert und im Anschluss folgt eine Transformation beider Qubits durch das CNOT-Gatter, wie in Abbildung 2.2 dargestellt. Das Hadamard Gatter überführt das Qubit A in eine Superposition, das heißt bei Messung ist das Ergebnis $|0\rangle$ oder $|1\rangle$, mit jeweils genau 50 Prozent Wahrscheinlichkeit. Messen wir für Qubit A den Zustand $|0\rangle$, so ist der Folgezustand und somit der Zustand des Quantenregisters $|00\rangle$. Messen wir für Qubit A den Zustand $|1\rangle$, so ist der Folgezustand des Quantenregisters $|11\rangle$. Somit ist der Zustand von Qubit B abhängig vom Zustand von Qubit A.

Im Folgenden werden die **Bell-Zustände**, auch EPR-Paare (Einstein-Podolsky-Rosen-Paar) [NC01] genannt, präsentiert:

$$\left|\Phi^+\right\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$\left|\Phi^{-}\right\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$\left|\Psi^{+}\right\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

$$\left|\Psi^{-}\right\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

Tatsächlich wird gemäß der Quantenmechanik, sobald eines der beiden Qubits gemessen wird, der Zustand des anderen Qubits ebenfalls augenblicklich bestimmt. Dieses quantenmechanische Verschränkungsverhalten führte Einstein und seine Kollegen zum sogenannten EPR-Paradoxon. Es besagt, dass die Messung eines Qubits den Zustand des anderen Qubits augenblicklich verändert, unabhängig von der Entfernung zwischen den beiden Qubits. Dieses Phänomen spielt eine wichtige Rolle in der Quantenteleportation, die im nächsten Unterkapitel erläutert wird.

2.1.5 Quantenteleportation

Die grundlegende Idee bei der **Quantenteleportation** [Hom08] besteht darin, einen unbekannten Quantenzustand eines Qubits mithilfe von zwei klassischen Bits auf eine Weise zu übertragen, dass der Empfänger genau denselben Zustand reproduziert wie der des ursprünglichen Qubits. Der ursprüngliche Zustand wird dabei zerstört, so dass die Quantenteleportation nicht im Widerspruch zum **No-Cloning-Theorem** steht. Das No-Cloning-Theorem [Hom08] ist eines der frühesten Ergebnisse der Quantenberechnung und Quanteninformation und besagt, dass ein unbekanntes Quantensystem nicht durch unitäre Transformationen kopiert werden kann.

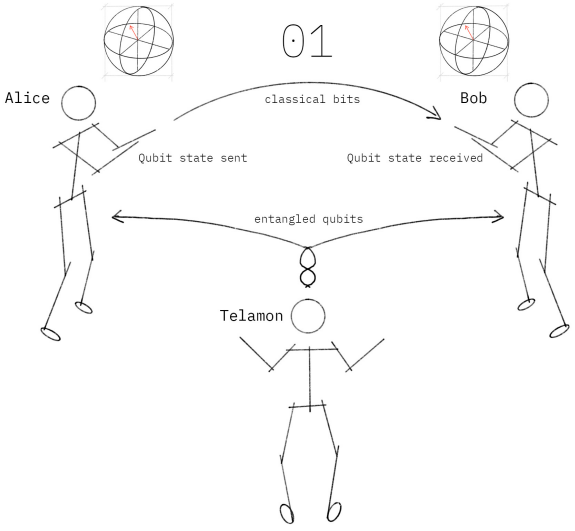


Abbildung 2.3: Quantenteleportation mit EPR-Quelle Telamon [Qis23]

Grundsätzlich teilen sich in der Quantenteleportation zwei Parteien, Alice und Bob, ein verschränktes Paar von Qubits, z. B. $|\beta_{00}\rangle$. Zum Beispiel ist das EPR-Paar, das unten gezeigt wird, ein verschränkter Quantenzustand:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

In Abbildung 2.3 versucht Alice, ein unbekanntes Qubit $|\psi\rangle$ an Bob zu senden, hat aber dazu nur einen klassischen Kanal mit zwei klassischen Bits zur Verfügung, aber einen gemeinsamen verschränkten Zustand mit Bob über eine EPR-Quelle Telamon. Der Gesamtzustand des Systems $|\phi\rangle$ lautet wie folgt:

$$|\phi\rangle = |\psi\rangle \otimes |\beta_{00}\rangle \ = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{|00\rangle + |11\rangle}{\sqrt{2}} \ = \frac{1}{\sqrt{2}}(\alpha(|000\rangle + |011\rangle) + \beta(|100\rangle + |111\rangle))$$

Die ersten beiden Qubits gehören zu Alice und das dritte Qubit gehört zu Bob. Alice wendet ein CNOT-Gatter auf den ersten beiden Qubits an und dann wendet sie ein Hadamard-Gatter auf den ersten Qubit an, was zu folgendem Ergebnis führt:

$$\begin{aligned} &\frac{1}{2}[a(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + b(|010\rangle + |001\rangle - |110\rangle - |101\rangle)] \\ =&\frac{1}{2}[|00\rangle (a\,|0\rangle + b\,|1\rangle) + |01\rangle (a\,|1\rangle + b\,|0\rangle) \ + |10\rangle (a\,|0\rangle - b\,|1\rangle) + |11\rangle (a\,|1\rangle - b\,|0\rangle)] \end{aligned}$$

Schließlich misst Alice beide Qubits, die zu ihr gehören. Sie wird einen der vier Ergebniszustände $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$ mit gleicher Wahrscheinlichkeit von $\frac{1}{4}$ erhalten. Wegen des verschränkten Zustands schaltet das Bit von Bob augenblicklich in einen mit dem Resultat von Alice korrelierten Zustand. Abhängig vom Ergebnis von Alice Messung kollabiert Bob Qubit zu $|0\rangle + b\,|1\rangle$, $|1\rangle + b\,|0\rangle$, $|0\rangle - b\,|1\rangle$ oder $|1\rangle - b\,|0\rangle$, jeweils. Alice sendet dann die Ergebnisse ihrer Messung an Bob unter Verwendung von zwei klassischen Bits. Alice ursprüngliches Qubit $|\psi\rangle$ wird durch ihre Messung vollständig zerstört, was die Quantenteleportation mit dem No-Cloning-Theorem in Einklang bringt. Schließlich kann Bob nach Erhalt der beiden klassischen Bits den Zustand des Qubits in seiner Hand kennen, indem er $I-$, $X-$, $Z-$ Gatter anwendet, wenn die klassischen Bits $|00\rangle$, $|01\rangle$, $|10\rangle$ und $|11\rangle$ entsprechen, um den ursprünglichen Zustand von Alice wiederherzustellen, siehe Tabelle 2.1. Im Quantenschaltkreis von Abbildung 2.4 ist die Übertragung der zwei klassischen Bits durch eine Doppellinie dargestellt.

Damit ist klar, dass bei einer Quantenteleportation auf die augenblicklich (über die

Tabelle 2.1: Transformationen von Bob

Zustand von Bob	**Erhaltene Bits**	**Angewendete Gatter**
$\alpha\|0\rangle + \beta\|1\rangle$	00	I
$\alpha\|1\rangle + \beta\|0\rangle$	01	X
$\alpha\|0\rangle - \beta\|1\rangle$	10	Z
$\alpha\|1\rangle - \beta\|0\rangle$	11	XZ

EPR-Verschränkung) beim Empfänger vorliegende Information nur zugegriffen werden kann, wenn das Messergebnis vorher über den klassischen Kanal mitgeteilt wurde.

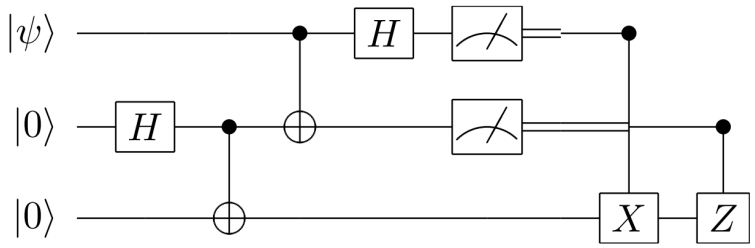


Abbildung 2.4: Der Schaltkreis für die Quantenteleportation [NC01]

2.2 Verteilte Quantenberechnung

2.2.1 Verteilte Quantenarchitekturen

Eine verteilte Quantenarchitektur besteht aus [AMH19]:

- Mehreren Quantenverarbeitungseinheiten (QVE), wobei jede Einheit eine bestimmte Anzahl von Qubits enthält und in der Lage ist, darauf einige universelle Quantengatter auszuführen.

- Einem klassischen Kommunikationsnetzwerk, über das die QVE Nachrichten senden oder empfangen können, insbesondere während sie ihre Qubits messen.

- Hardware zur Erzeugung von Ebits. Ein Ebit ist ein maximal verschränkter bipartiter Quantenzustand, der zwischen zwei QVE geteilt wird und aus zwei Qubits besteht. Jedes Qubit befindet sich in einer separaten QVE. Ein Ebit enthält auch die erforderlichen Informationen zum Senden eines einzelnen Qubits von einer QVE zur anderen. Die Hardware zur Erzeugung und gemeinsamen Nutzung von Ebits kann entweder in jeder QVE vorhanden sein oder von einem zentralen Gerät bereitgestellt werden.

2.2.2 Verteilte Quantenschaltkreise

Ein **verteilter Quantenschaltkreis** [SMZ21] ist eine Erweiterung des Quantenschaltkreismodells. Dieser besteht aus n-begrenzt leistungsfähigen Quantenschaltkreisen, noch bekannt als Partitionen oder Knoten. Diese Knoten sind an einem bestimmten Ort voneinander entfernt und insgesamt implementieren die Funktionalität eines Quantensystems. Die verschiedenen Teile eines verteilten Quantenschaltkreises sollten miteinander kommunizieren, indem sie ihre Qubits über einen Quantenkanal via Teleportation austauschen.

In einem verteilten Quantenschaltkreis sind zwei Arten von Quantengattern [HMZMH20] zu betrachten: Ein **lokales Gatter** hat ihre Steuer- und Ziel-Qubits im selben Subsystem, d.h. alle Qubits des Quantengatters befinden sich in derselben Partition und können lokal ohne Teleportation ausgeführt werden. In einem **globalen Gatter** hingegen werden die Qubits in verschiedenen Partitionen positioniert. Um ein globales Gatter in einem verteilten Quantenschaltkreis auszuführen, muss ein Qubit in ein anderes Subsystem teleportiert und dann zum Heim-Subsystem zurückgebracht werden.

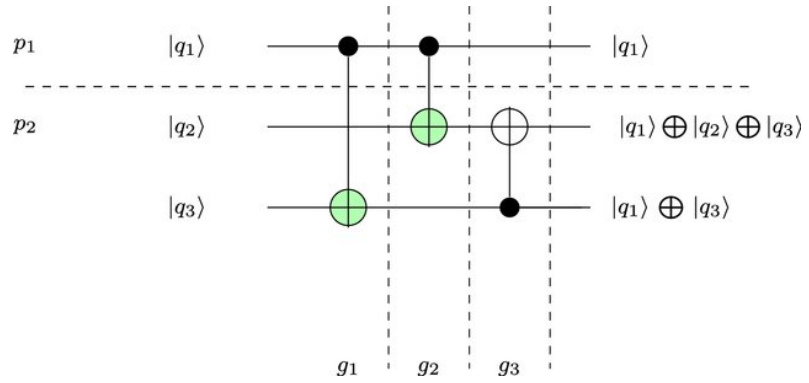


Abbildung 2.5: Verteilter Quantenschaltkreis in 2 Partitionen [DZMHNb20]

Aus Abbildung 2.5 erkennt man, dass der verteilte Quantenschaltkreis aus drei Qubits besteht, die in zwei Partitionen aufgeteilt sind. Darüber hinaus sieht man, dass dieser aus 3 CNOT-Gatter besteht. Die ersten zwei Gatter (g_1 und g_2) sind globale Gatter, da sich ihre Qubits in verschiedenen Partitionen (p_1 und p_2) befinden. Das letzte Gatter ist nämlich ein lokales Gatter, da sich ihre Qubits in derselben Partition befinden (p_2).

Obwohl verteilte Quantenschaltkreise auf den ersten Blick ähnlich wie Quantenschaltkreise sind, ist das Problem bei einem verteilten Quantenschaltkreis grundsätzlich anders als bei einem Quantenschaltkreis. In einem verteilten Quantenschaltkreis liegt der Optimierungsprozess darin, die Anzahl der globalen Gatter und damit die Teleportationskosten zu minimieren. Um dieses Ziel zu erreichen, werden der Kernighan-Lin-Algorithmus, die Spektrale Partitionierung und die genetischen Algorithmen angewendet und als Nächstes verdeutlicht.

2.3 Genetische Algorithmen

2.3.1 Kodierung der Chromosomen

Die **genetischen Algorithmen** (GA) [Wei15] [GKK13] basieren auf der Darwinschen Theorie der Evolution (**Survival of the Fittest**), bei der das Überleben fitter Lebewesen im Mittelpunkt steht. GA ist ein populationsbasierter Algorithmus, bei dem jede Lösung einem **Chromosom** entspricht und jeder Parameter ein **Gen** repräsentiert. Um die Entwicklung einer Population von Lösungen eines Ausgangsproblems überhaupt auf dem Computer simulieren zu können, müssen die Lösungen zunächst in geeigneter Form dargestellt werden. Man spricht dabei von der Form der Kodierung, die zu wählen ist. Letztlich muss es möglich sein, mit der gewählten Repräsentation jede denkbare Lösung des Problems darstellen zu können. Ursprünglich wurde bei GA ausschließlich die binäre Kodierung verwendet. Gruppen von Bits repräsentieren dabei jeweils ein bestimmtes Merkmal der Lösung.

0	**1**	**1**	**0**	**0**	**1**	**0**

Abbildung 2.6: Beispiel für die binäre Kodierung

Heutzutage werden jedoch zunehmend auch andere Kodierungsformen wie reelle Zah-

len oder Permutationsfolgen verwendet. Diese sind oftmals besser auf das Problem zugeschnitten und leichter verständlich.

2.3.2 Fitnessfunktion

Hat man eine geeignete Kodierung gewählt, muss man noch definieren, was als **Fitness** der Individuen betrachtet wird. Üblicherweise ist dies eine Funktion, die von allen kodierten Parametern abhängig ist. Das Ziel des GA ist es nun, für diese Funktion ein möglichst globales Maximum zu bestimmen und somit das Individuum zu finden, das die beste Lösung für das Ausgangsproblem darstellt. Es kann aber auch sein, dass nicht das Maximum, sondern das Minimum der Funktion gesucht wird, dass also das Individuum mit minimaler Fitness als beste Lösung betrachtet wird.

Nachdem die Kodierung der Individuen und die Form der Bestimmung der Fitnesswerte festgelegt wurden, werden bei Start des Algorithmus eine bestimmte Anzahl Individuen aus dem Suchraum zufällig oder in Kombination mit verschiedenen Kriterien bestimmt. Diese Individuen bilden die sogenannte Startpopulation, deren Größe im Vorhinein festgelegt wird. Anschließend wird von allen Individuen innerhalb der Startpopulation der Fitnesswert berechnet. Im Folgenden werden die genetischen Operatoren in Betracht gezogen.

2.3.3 Selektion

Die natürliche **Selektion** operiert auf den Individuen einer Population und ist von entscheidender Bedeutung zum Fortschreiten der simulierten Evolution. Ihre Aufgaben innerhalb eines GA sind vielfältig, z.B. Auswahl der an einer Rekombination beteiligten Individuen; Auswahl eines Individuums, das mutiert werden soll; Auswahl eines Individuums, das in die neue Population übernommen werden soll (Reproduktion); Auswahl einer bestimmten Anzahl von Individuen, die eine neue Population bilden sollen, usw.

Bei einem der am häufigsten verwendeten Schemata, die Roulette-Wheel-Selektion, wird erst die Gesamtfitness berechnet und danach fur jedes Individuum das Verhältnis zwischen seiner Fitness und der Gesamtfitness ermittelt. Sein Fitnessverhältnis ist (als Prozentwert gesehen) die Wahrscheinlichkeit, dass ein Individuum zur Kreuzung ausgewählt wird. Auf dieselbe Art und Weise wird auch der Partner für das so ausgesuchte Elternteil ermittelt.

Aus der Abbildung 2.7 sieht man, dass Chromosom 3 und Chromosom 1 für Reproduktion ausgewählt werden, da sie die höchsten Wahrscheinlichkeiten aufweisen. Chromosom 2 hingegen besitzt die niedrigste Wahrscheinlichkeit und wird somit nicht mehr in Betracht gezogen.

Bei der Turnier-Selektion hingegen werden k Chromosomen zufällig ausgewählt. Der Fitteste wird als Turniergewinner und dementsprend als Elternteil ausgewählt. Bei einer Population der Größe N kann der Wert von k von 2 bis N variieren. Niedrigere Werte von k bieten einen geringeren Selektionsdruck, während höhere Werte einen höheren Selektionsdruck. Wenn beispielsweise k = N gilt, ist der fitteste Chromosom in der aktuellen Population immer der Turniergewinner [BOM15].

Abbildung 2.8 zeigt so einen Ablauf dieser Selektion. Zuerst werden 3 aus insgesamt 7 Chromosomen zufällig ausgewählt. Der Chromosom mit dem höchsten Fitnesswert (in die-

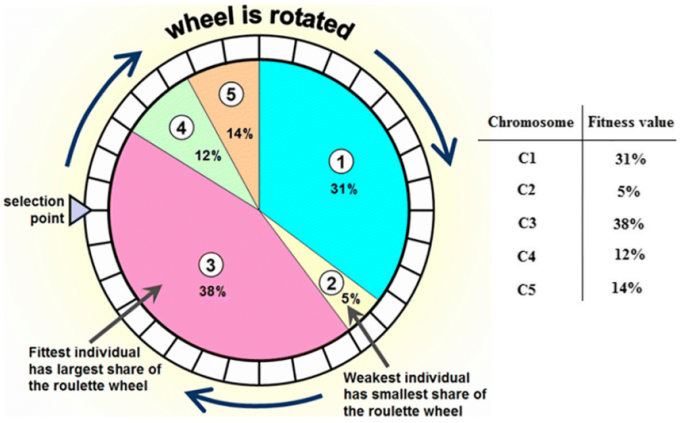


Abbildung 2.7: Roulette-Wheel-Selektion [SFHBA+18]

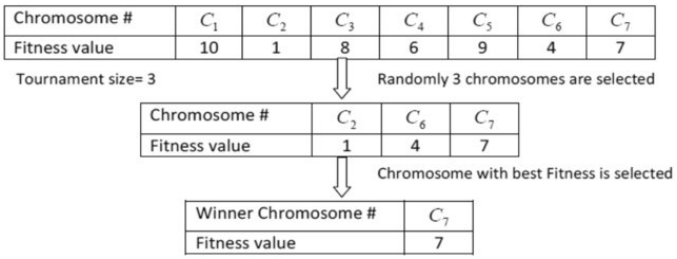


Abbildung 2.8: Turnier-Selektion [BA21]

sem Fall Chromosom 7 mit Fitnesswert 7) wird für Reproduktion ausgewählt. Der interessierte Leser findet eine Beschreibung dieser und anderer Selektionsformen in [GKK13].

2.3.4 Kreuzung

Die Kreuzung ist der wichtigste genetische Operator beim GA, bei dem Gene der Eltern kopiert und an die Nachkommen weitergegeben werden. Es gibt verschiedene Arten von Kreuzungsoperatoren. Aus Abbildung 2.9 erkennt man, dass beim One-Point Crossover eine Rekombination an einem zufällig ausgewählten Punkt zwischen den einzelnen Bits erfolgt. Die Wahl des Schnittpunktes ist zufallsbasiert und wird auf die beiden zufällig gewählten Chromosomen aus der Population angewendet. Ab diesem Punkt erfolgt ein Austausch der Gensequenzen.

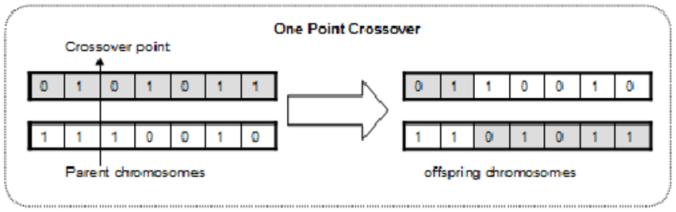


Abbildung 2.9: One-Point-Crossover [KUT11]

Der Abbildung 2.10 lässt sich entnehmen, dass beim Two-Point Crossover zwei Kreuzungspunkte für die Reproduktion gewählt werden. Der Austausch der Gensequenzen

erfolgt zwischen den beiden Kreuzungsstellen.

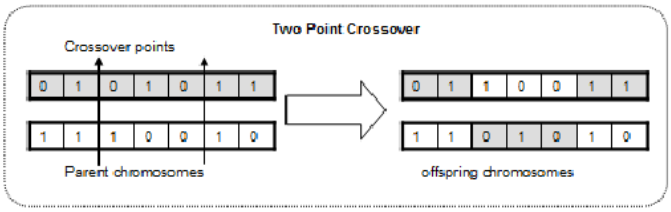


Abbildung 2.10: Two-Point-Crossover [KUT11]

Eine Beschreibung dieser und anderer Kreuzungsformen findet man in [GKK13].

2.3.5 Mutation

Die Mutation erhöht die Vielfalt der Individuen in der Population und das explorative Verhalten vom GA, indem sie zufällig die Gene in den Chromosomen ändert. Im Vergleich zur Kreuzung sollte die Mutation jedoch nur sehr selten angewandt werden – man spricht von einer kleinen Mutationsrate, da sonst Allelfolgen, die eine gute Fitness bewirken, zu leicht zerstört werden könnten. Es gibt verschiedene Varianten der Mutation, wie z.B. Bit-Flip-Mutation (Hier wird an einer zufällig bestimmten Stelle des binären genetischen Codes eines Individuums der binäre Wert invertiert,d.h. eine 1 wird zu einer 0, beziehungsweise eine 0 zu einer 1, wie in Abbildung 2.11 dargestellt), Swap-Mutation (Dieser Operator funktioniert durch zufällige Auswahl von zwei Elementen der Permutationsfolge und Austausch ihrer Werte.), usw.

Abbildung 2.11: Bit-Flip-Mutation [Rif16]

2.3.6 Ersetzungsschema

Das **Ersetzungsschema** in einem GA ist das Prinzip, nach dem die alten Individuen durch neue Individuen in der Population ersetzt werden. Das Ziel des Ersetzungsschemas ist, sicherzustellen, dass die Population eine ausreichende Vielfalt aufweist, um lokale Minima zu vermeiden, und dass die besten Individuen beibehalten werden, um das optimale Ergebnis zu erreichen (**Elitismus**).

2.3.7 Abbruchskriterien

Nun liegt eine vollständige neue Population vor. Die alte Population wird durch die neue ersetzt und es wird getestet, ob das Abbruchkriterium erfüllt ist. Mögliche **Abbruchskriterien** sind z.B. keine Verbesserung zur vorherigen Generation, das Erreichen einer gewissen Anzahl von Iterationen, die Verifikation einer gefundenen Optimallösung oder einfach das Erreichen einer vorgegebenen Anzahl von Generationen. Ist das Abbruchkriterium nicht erfüllt, arbeitet der Algorithmus mit der neuen Population weiter. Ist

es jedoch erfüllt, wird in der neuen Population das Individuum mit dem höchsten Fitnesswert gesucht und als beste vom Algorithmus ermittelte Lösung zurückgegeben. Der Algorithmus 1 zeigt den allgemeinen Ablauf eines GA.

Algorithm 1: Allgemeiner Ablauf eines GA

```
1  Initialisiere eine zufällige Population von Chromosomen
2  while Abbruchbedingung nicht erfüllt do
3  |  Bewerte die Fitness jedes Chromosoms in der Population
4  |  Selektiere Chromosomen für die Fortpflanzung basierend auf ihrer Fitness
5  |  Erzeuge Nachkommen durch Anwendung der genetischen Operatoren
6  |  Kreuzung und Mutation auf die ausgewählten Chromosomen
7  |  Evaluiere die Fitness der Nachkommen
8  |  Wähle Chromosomen für die nächste Generation aus der Elternpopulation und
   |   den Nachkommen aus
9  end
10 return Beste Lösung als Chromosom
```

2.3.8 Exploitation gegen Exploration

Die Größe der Startpopulation stellt einen wichtigen Parameter im GA dar. Es wurden zahlreiche Studien[Ala92], [RG02] durchgeführt, um den Einfluss unterschiedlicher Populationsgrößen auf die Leistungsfähigkeit vom GA zu analysieren. Die grundlegende Idee ist, dass eine größere Populationsgröße die Diversität der Population erhöhen kann und somit die Leistungsfähigkeit vom GA verbessern kann. Es wurde jedoch gezeigt [CTCY12], dass es Bedingungen gibt, unter denen größere Populationen schädlich sein können. Im Allgemeinen hängt die Populationsgröße von der Komplexität des Problems ab.

Es ist wichtig zu betonen, dass GA auf zwei grundlegenden Konzepten basieren: **Exploitation** - die Nutzung von bereits gefundenem vielversprechendem Material, und **Exploration** - die Untersuchung weiterer vielversprechender Bereiche im Suchraum [ČLM13]. Eine größere Populationsgröße, höhere Mutationsraten oder Kreuzungsraten können die Exploration fördern, indem sie mehr Vielfalt in der Population erzeugen und es dem Algorithmus ermöglichen, potenziell bessere Lösungen in unbekannten Bereichen des Suchraums zu finden. Eine kleinere Populationsgröße, niedrigere Mutationsraten oder Kreuzungsraten hingegen können die Exploitation fördern, indem sie dem Algorithmus erlauben, bereits gute Lösungen zu behalten und diese weiter zu verfeinern. Es ist entscheidend, ein ausgewogenes Verhältnis zwischen Exploration und Exploitation zu finden, um optimale Ergebnisse mit einem GA zu erzielen. Eine zu starke Betonung der Exploration kann dazu führen, dass der Algorithmus in weniger vielversprechenden Bereichen des Suchraums verweilt und keine konvergenten Lösungen findet. Auf der anderen Seite kann eine zu starke Betonung der Exploitation dazu führen, dass der Algorithmus in einem lokalen Optimum stecken bleibt. Die optimale Balance hängt von der spezifischen Problemstellung und den gewünschten Zielen ab und ist oft Gegenstand von Experimenten und Optimierung bei der Anwendung von GA. Im Folgenden werden wir uns mit Graphpartitionierungsalgorithmen befassen.

2.4 Graphpartitionierungsalgorithmen

Einer der Vorteile der Verwendung von **Graphentheorie** besteht darin, dass sie für viele Probleme dieselbe Form bereitstellt. Formal besteht ein Graph aus einem Paar von Mengen (V, E), wobei V die Menge der Knoten und E die Menge der Kanten ist, die aus Paaren von Knoten gebildet werden. In der Mathematik umfasst eine Graphpartitionierung die Aufteilung seiner Knoten in sich gegenseitig ausschließende Mengen. Die Kanten, die die Mengen verbinden, stellen die Verbindung zwischen diesen Mengen dar. Üblicherweise werden Graphpartitionierungsprobleme als **NP-schwer** [ABKL07] eingestuft. Daher verwenden Lösungen für solche Probleme in der Regel Heuristiken und Annäherungsalgorithmen[SBH+16].

Angenommen, ein Graph $G = (V, E)$, wobei V und E eine Menge von n Knoten bzw. eine Menge von Kanten repräsentieren. Ein balanciertes Partitionierungsproblem teilt G in K Teile ähnlicher Größe auf, so dass das Gewicht der Kanten zwischen getrennten Komponenten minimal ist [AR04]. Lokale und globale Algorithmen sind zwei Methoden für diesen Zweck.

Der **Kernighan-Lin-Algorithmus** [KL70] verwendet eine lokale Suche für zwei gleichmäßige Partitionen. Ein effektives Problem dieses Algorithmus, das die Qualität des Endergebnisses beeinträchtigen kann, ist die zufällige Initialisierung der Knotenmenge. Globale Ansätze erfordern keine zufällige Anfangspartition und hängen von den Eigenschaften des gesamten Graphen ab. Die **spektrale Partitionierung** [McS01] ist die am weitesten verbreitete Methode.

Für die beiden Partitionierungsalgorithmen verwenden wir einen ungerichteten Graphen, um die Struktur des jeweiligen Quantenschaltkreises darzustellen. Ein **ungerichteter** Graph $G = (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge E in $V \times V$, wobei jeder Kante $e \in E$ von G zwei (nicht notwendig verschiedene) Knoten aus V zugeordnet sind.

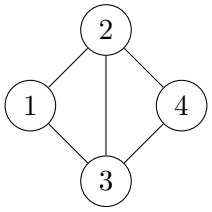


Abbildung 2.12: Ein ungerichter Graph G

2.4.1 Kernighan-Lin-Algorithmus

Der Kernighan-Lin (K-L) Algorithmus ist ein heuristischer Algorithmus zur Graphpartitionierung. Die Eingabe des Algorithmus ist ein ungerichteter gewichteter Graph $G = (V, E, W)$, wobei V die Menge der Knoten, E die Menge der Kanten und W die Menge der den Kantengewichte ist. Der K-L Algorithmus partitioniert V in zwei Teilmengen A und B. Die Partitionierung erfolgt so, dass die Summe der Gewichte der Kanten, die von A nach B führen, minimiert wird. Wenn die Kanten keine Gewichte haben, nehmen wir an, dass die Gewichte der Kanten gleich eins sind.

Zunächst werden die Knoten zufällig in Partitionen aufgeteilt. Dann müssen zwei Parameter berechnet werden: Erstens, die Kommunikationskosten (Gewichte der Kanten) jedes Knotens mit den anderen Knoten in seiner Partition, bekannt als **interne Kosten**. Zweitens, die Kommunikationskosten jedes Knotens mit anderen Partitionen, bekannt als **externe Kosten**. Diese beiden Parameter werden durch I und E repräsentiert. Angenommen, der Unterschied zwischen diesen beiden Parametern wird mit D ($D = E - I$) angezeigt. Die Werte von D für alle Knoten im Schaltkreis werden berechnet und der Gewinn zwischen den beiden Knoten x und y, die in zwei Abschnitte A und B unterteilt sind, wird als $g_{xy} = D_x + D_y - 2C_{xy}$ berechnet. Der Gewinnwert wird für jeden der beiden Knoten berechnet und die größte Zahl, die als g_1 bezeichnet wird und die Beziehung zwischen den beiden Knoten angibt, wird ausgewählt. Als nächstes werden diese beiden Knoten gesperrt und aus den Mengen A und B entfernt. Jetzt werden die Werte von D aktualisiert und die Berechnungen werden für die verbleibenden Knoten durchgeführt. Dieser Prozess wird fortgesetzt, bis es keine Knoten mehr in den beiden Mengen gibt. Bei jedem Schritt werden g_2, g_3, usw. identifiziert.

Der Parameter k repräsentiert die Anzahl jeder Ausführungsstufe und $G_k = \max \sum_{i=1}^{k} g_i$ wird in jeder Stufe berechnet. Der größte Wert gibt an, welche zwei Knoten ausgetauscht werden müssen. Im ersten Durchlauf des Algorithmus werden diese Knoten ausgetauscht und alle gesperrten Knoten werden für die nächste Runde freigeschaltet. Dieser Algorithmus wird fortgesetzt, solange $G_k > 0$.

Algorithm 2: Kernighan-Lin-Algorithmus [KL70]

```
1  Function Kernighan-Lin(G(V,E)):
2      Bestimme eine gleichmäßige Anfangspartitionierung der Knoten in die Mengen
         A und B
3      repeat
4          Berechne die D-Werte für alle a ∈ A und b ∈ B
5          for i = 1 to |V|/2 do
6              Wähle a_i ∈ A und b_i ∈ B, sodass g_i = D_a + D_b − 2C(a, b) maximal ist
7              Entferne a_i und b_i für weitere Betrachtung in diesem Durchlauf
8              Aktualisiere D-Werte für die Elemente von A = A/a_i und B = B/b_i
9          end
10         Bestimme k, das G_k maximiert, die Summe von g_1 bis g_k if G_k > 0 then
11             Tausche a_1, a_2, ..., a_k mit b_1, b_2, ..., b_k
12         end
13     until G_k ≤ 0
14     return G(V,E)
15 end
```

Der K-L-Algorithmus kann erweitert werden, um jede Partition in kleinere Partitionen aufzuteilen, so dass die Kommunikation zwischen diesen Partitionen optimiert wird.

2.4.2 Spektrale Partitionierung

Um den Ablauf von der **spekralen Partitionierung** besser zu verstehen, werden im Folgenden einige Definitionen aus der linearen Algebra vorgestellt.

Definition 1 (Adjazenzmatrix) [BR02]

Die Darstellung eines Graphen G folgt meist in Form seiner **Adjazenzmatrix** $A(G)$.

Sei G ein Graph mit n Knoten, dann ist $A(G)$ eine $n \times n$ Matrix mit den Einträgen

$$a_{i,j} = \begin{cases} 1, & \text{i und j sind adjazent,} \\ 0, & \text{sonst.} \end{cases}$$

Die Adjazenzmatrix des Graphen G aus der Abbildung 2.12 lautet:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Bemerkung:
Die Adjazenzmatrix eines ungerichteten Graphen ist stets symmetrisch, d.h. $a_{i,j} = a_{j,i}$. Die Summe der Elemente einer Spalte oder Zeile ist gleich dem Knotengrad:

$$\sum_{i=1}^{n} a_{i,k} = \sum_{j=1}^{n} a_{k,j} = \deg(k)$$

Definition 2 (Gradmatrix) [BR02]
Eine weitere Graphmatrix ist die **Gradmatrix** des Graphen G. Die Gradmatrix ist eine $n \times n$-Diagonalmatrix, deren Diagonalelemente die Knotengrade von G sind, d.h. $d_i = \sum_{j=1}^{n} a_{i,j}$. Für den G ergibt sich die Gradmatrix

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Definition 3 (Laplacematrix) [BR02]
Damit lässt sich die **Laplacematrix** eines Graphen als Differenz aus der Gradmatrix und der Adjazenzmatrix definieren:

$$\mathcal{L} = D - A = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

Dabei ist D die Diagonalmatrix, welche die Gradzahlen der Knoten enthält, und A ist die Adjazenzmatrix, welche angibt, welche Knoten direkt miteinander verbunden sind.

Definition 4 (Normierte Laplacematrix) [But16]
Die **normierte Laplacematrix** $\mathcal{L}_{\text{norm}}$ eines Graphen G mit n Knoten wird durch die folgende Formel definiert:

$$\mathcal{L}_{norm} = D^{-1/2} \mathcal{L} D^{-1/2}$$

wobei $\mathcal{L}$ die Laplacematrix von G ist, A die Adjazenzmatrix von G ist, D die Diagonalmatrix der Knotengrade ist und $D^{-1/2}$ die Diagonalmatrix der inversen Wurzeln der Knotengrade.

$$\mathcal{L}_{norm} = \begin{pmatrix} 1 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \end{pmatrix}$$

Definition 4 (Eigenvektor und Eigenwert) [BR02]

Ein **Eigenvektor** einer quadratischen Matrix $A \in \mathbb{R}^{n \times n}$ ist ein nichtnull-Vektor $v \in \mathbb{R}^n$, so dass, wenn A mit v multipliziert wird, das konstante Vielfache von v ergibt. Das heißt

$$Av = \lambda v$$

Die Zahl $\lambda \in \mathbb{R}$ wird **Eigenwert** von A genannt, der zum Eigenvektor v gehört. Algorithmus 3 beschäftigt sich mit dem Pseudocode der spektralen Partitionierung.

Algorithm 3: Spektrale Partitionierung [VL07]

Data: Adjazenzmatrix $A \in \mathbb{R}^{n \times n}$, Anzahl k der zu konstruierenden Cluster
Result: Cluster $A_1, \ldots, A_k$ mit $A_i = \{j \mid y_j \in C_i\}$

```
Berechnung der Laplace-Matrix L oder normalisierte Laplace-Matrix L_norm
Bestimme die ersten k Eigenvektoren u_1, ..., u_k von der Laplacematrix
Erzeuge die Matrix U ∈ R^{n×k}, welche die Vektoren u_1, ..., u_k als Zeilen enthält
for i = 1, ..., n do
    Sei y_i ∈ R^k ∀i = 1, ..., n der zur i-ten Spalte von U zugehörige Vektor
end
Führe den k-Means-Algorithmus auf den Punkten (y_i)_{i=1,...,n} durch, um die
  Cluster C_1, ..., C_k zu erhalten
```

2.4.3 Graphische Darstellung eines Quantenschaltkreises

Ein gewichteter Graph wird normalerweise definiert als $G = (V, E, W)$, wobei V die Menge der Knoten $V = v_1, v_2, ..., v_n$ ist, E die Menge der Kanten $e_1, e_2, ..., e_k$ und W die Menge der Kantengewichte, die durch $w(e_1), w(e_2), ..., w(e_k)$ repräsentiert werden. Um einen Quantenschaltkreis in ein Graphmodell zu transformieren, müssen zunächst die Qubits und die Verbindungen zwischen ihnen in Bezug auf Knoten und Kanten in einem Graphmodell dargestellt werden. Angenommen, $q_1, q_2, ..., q_n$ sind Qubits eines Quantenschaltkreises, wobei n die Anzahl der Qubits in der Darstellung des Schaltkreises ist. Qubits werden in einem Graphen als Knoten interpretiert, sodass der zugehörige Graph n Knoten hat. Die Knotenmenge V des Graphen ist $V = q_1, q_2, ..., q_n$. Jedes Quantengatter besteht aus einem oder mehreren Qubits und Qubits eines Quantenschaltkreises sind über Quantengatter miteinander verbunden. Ein-Qubit-Quantengatter benötigen nur ein Qubit zur Ausführung und fügen somit keine Kanten im Graphenmodell hinzu. Wenn zwei Qubits durch ein Gatter miteinander verbunden sind, gibt es eine Kante zwischen ihren entsprechenden Knoten im Graphen, andernfalls sind die Knoten disjunkt. Diese Verbindungen werden durch Kanten mit einem Anfangsgewicht von eins dargestellt. Wenn mehrere Gatter die gleichen Qubits benötigen, erhöht sich das Gewicht um eine Einheit entsprechend. Abbildung 2.13 zeigt ein Beispiel für einen zufälligen Quantenschaltkreis bestehend aus vier Qubits und Abbildung 2.14 den entsprechenden ungerichteten gewichteten Graphen dazu.

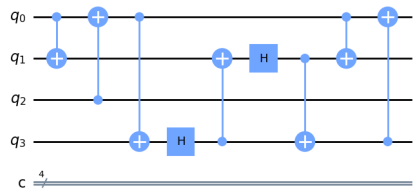


Abbildung 2.13: Zufälliger Quantenschaltkreis [ZMHH18]

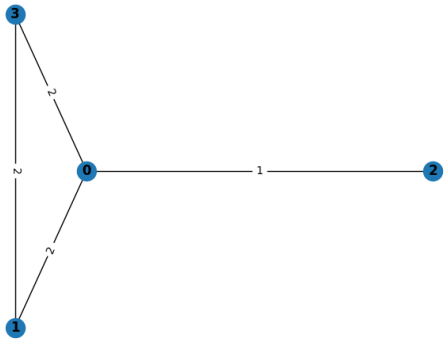


Abbildung 2.14: Der gewichtete Graph dazu

3 Verwandte Arbeiten

Verteilte Quantenberechnung wird seit mehr als zwanzig Jahren untersucht und die ersten Vorschläge in dieser Richtung stammen von Grover [Gro97], Cleve und Buhrmann [CB97] und später von Cirac et al. [CEHM99]. Grover präsentiert in seiner Arbeit ein verteiltes Quantensystem [Gro97], bei dem sich einige Teilchen an entfernten Orten befinden und jeder seine Berechnungen durchführt und bei Bedarf die erforderliche Information an eine Basisstation sendet. Grover zeigt, dass die Berechnungszeit proportional zur Anzahl der Teilsysteme des verteilten Systems ist.

In [CB97] untersuchen Cleve und Buhrman die Quantenkommunikation. Sie präsentieren, dass die Quantenverschränkung als Ersatz für die Kommunikation eingesetzt werden kann, wenn wir eine Funktion berechnen möchten, deren Eingabedaten auf verschiedene entfernte Orte verteilt sind. Auch Cirac et al. [CEHM99] zeigen, dass die Verwendung von maximal verschränkten Zuständen für ideale Quantenkanäle und eine ausreichend große Anzahl von Knoten gegenüber unkorrelierten Zuständen vorteilhaft ist.

Zuerst zeigen Yimsiriwattana et al. in [YLJ04b], dass für beliebige zusammenhängende globale CNOT Gatter, bei denen das Steuer-Qubit gemeinsam ist, die Steuerlinie nur einmal verteilt werden muss, da sie wiederverwendet werden kann. Diese Idee ermöglicht eine Minimierung der Teleportationskosten.

In [YLJ04a] wird ein verteiltes Modell des **Shor-Algorithmus** vorgestellt. Hierfür verwendet man Teleportation als Kommunikationsprotokoll. Globale Gatter werden eingesetzt, um den verteilten Quantenschaltkreis für den Shor-Algorithmus zu implementieren. Allerdings bemühen sich die Autoren nicht, die beste Positionierung der Qubits zwischen den verteilten Knoten zu bestimmen. Daher ist ihre Methode nicht systematisch.

Im Jahr 2012 schlagen STRELTSOV et al. [SKB12] eine Form von verteilter Verschränkung vor und liefern die minimalen Quantenkosten, um einen verschränkten Zustand über eine lange Distanz zu senden. Es ist die erste Arbeit, die die Kommunikationskosten und den Einfluss verteilter Kommunikation auf Quantenverschränkung diskutiert.

Die Autoren von [AMH19] stellen eine automatisierte Methode vor, um einen Quantenschaltkreis in mehrere Partitionen aufzuteilen, wobei die Teleportationskosten minimiert werden. Sie wandeln die Eingangsschaltung in eine äquivalente Schaltung um, die nur aus Clifford+T-Gattern besteht, und tauschen die Reihenfolge zwischen CNOT-Gatter und den 1-Qubit-Gattern von Clifford+T aus, indem sie alle CNOT-Gatter so früh wie möglich in der Schaltung platzieren. Dadurch rücken die CNOT-Gatter näher zusammen. Darüber hinaus diskutieren die Autoren nicht die allgemeine Anordnung der globalen Gatter in verschiedenen Partitionen. Sie diskutieren nur globale Gatter unter besonderen Bedingungen.

Daei et al. präsentieren in [DNZM20] ein Graphmodell, um verteilte Quantenschaltkreise aus monolithischen Quantenschaltkreisen zu generieren, sodass die Kommunikation zwischen den Partitionen eines verteilten Quantenschaltkreises minimiert wird. Hier betrachten die Autoren vier Partitionen.

In [DZMHNb20] stellt man einen **dynamischen Programmierungsansatz** vor, um

beliebige Quantenschaltkreise in K Partitionen aufzuteilen. Zunächst wandelt man den Quantenschaltkreis in einen bipartiten Graphen um und dann platziert man die Gatter und Qubits in jedem Teil des Graphen. Damit bezweckt man, die Anzahl der globalen Gatter zu minimieren.

Die Autoren in [DNZ21] haben auch das Problem der Minimierung der Kommunikationskosten in einem verteilten Quantenschaltkreis, der aus bis zu dreiquantigen Gatter besteht, diskutiert und eine neue heuristische Methode, die auf der Kommutierung von Quantengatter basiert, vorgestellt, um es zu lösen. Außerdem ziehen sie auch die Ausführungszeit der Benchmark-Schaltkreise in Betracht.

In einer anderen Arbeit [NMSZ21] verwenden die Autoren einen effizienten Ansatz zur Partitionierung, um die Kommunikationskosten zu minimieren. Sie kombinieren die Konzepte der Gatter- und Qubit-Teleportation, um die Anzahl der Teleportationen zu minimieren. Sie schlagen einen hybriden Partitionierungsansatz namens **WQCP** (Window-Based Partitioning of Quantum Circuits) vor, der sowohl die Ideen von **Teledata** als auch von **Telegate** kombiniert.

In [ZMHH18] ermittelt man basierend auf dem Kernighan-Lin-Algorithmus die Positionierung der globalen Gatter in einem beliebigen verteilen Quantenschaltkreis. Die Autoren erreichen die beste Reihenfolge der Ausführung von globalen CNOT-Gatter. In dieser Methode untersucht man alle möglichen Situationen für die Teleportation und Ausführung von globalen Gatter. Da hier alle möglichen Szenarien untersucht werden, steigt die Anzahl der zu untersuchenden Fälle **exponentiell** mit der Anzahl der globalen Gatter. Mit zunehmender Anzahl von Partitionen wird die Ausführungszeit des Algorithmus daher exponentiell länger, was in dem Artikel nicht erwähnt wird. Später verbessern sie die Ausführungszeit in [HMZMH20] mit Hilfe der GA, die ein Teil von den **evolutionären Algorithmen** (EA)[GKK13] sind. In diesem Artikel betrachtet man die Teleportationskosten als Fitnessfunktion des GA. Dieser versucht, die genaueste mögliche Konfiguration für die Ausführung von globalen Gatter zu finden.

4 Methodik

In diesem Kapitel werden Ansätze zur Minimierung der Anzahl von globalen Gattern und somit der Teleportationskosten in einem verteilten Quantenschaltkreis vorgestellt. Zur Erreichung dieses Ziels werden Graphpartitionierungsalgorithmen wie der Kernighan-Lin-Algorithmus und die Spektrale Partitionierung, genetische Algorithmen (GA) und hybridgenetische Algorithmen (HGA) untersucht, die eine Kombination aus Graphpartitionierungsalgorithmen und genetischen Algorithmen darstellen.

Die verschiedenen Ansätze wurden in Python implementiert, um die richtige Konfiguration zu finden, die die Teleportationskosten in verteilten Quantenschaltkreisen minimiert. In dieser Arbeit betrachten wir die Teleportationskosten (TK) als die Anzahl der Teleportationen, die zur Ausführung des Schaltkreises benötigt werden. Viele verschiedene Quantenschaltkreise wurden verwendet, um die Leistung der vorgeschlagenen Algorithmen zu testen. Eine Reihe von Tests waren die Quanten-Fourier-Transformations-(QFT(n))-Schaltkreise für $n \in \{4, 8, 16\}$, wobei n die Anzahl der Qubits im Schaltkreis ist. Die QFT-Schaltkreise [FH04] wurden in einigen Quantenalgorithmen, wie dem Shor-Algorithmus[NC01], verwendet. Eine weitere Gruppe von Schaltkreisen wurde aus der Revlib-Webseite [WGT$^+$08] als Benchmark-Schaltkreise entnommen. Quantenschaltkreise wie Parity_247, 4gt5-76, Sym9_147, 4mod7, Rd73_140 werden in dieser Arbeit verwendet. Der Schaltkreis "Figure4" stammt aus dem folgenden Artikel [ZMHH18]. Um die Effizienz der vorgeschlagenen Methoden zu implementieren und zu überprüfen, müssen die Benchmark-Schaltkreise zunächst in ihre grundlegenden Gatter zerlegt werden. Dies wurde mit Hilfe von Qiskit[Qis23] erreicht. Die Schaltkreise bestehen aus Ein-Qubit-Gattern wie $\{H, X, R_x, R_z, R_y\}$ und CNOT-Gattern, welche dann in das OpenQASM 2.0[CBSG17]-Format transformiert wurden. Um die Ergebnisse mit anderen Ansätzen vergleichen zu können, wurden die Schaltkreise aus den Veröffentlichungen[ZMHH18] [HMZMH20], außer Rd73_140, übernommen. Die Methoden wurden auf neun Benchmark-Schaltkreise angewendet, die in K Partitionen unterteilt sind, wobei K die Anzahl der Partitionen darstellt und hier K = 2, 3 und 4 betrachtet wurde.

4.1 Strategie zur Bestimmung der Teleportationskosten

In diesem Abschnitt wird die Strategie zur Bestimmung der Teleportationskosten in einem verteilten Quantenschaltkreis mithilfe eines Pseudocodes und eines anschließenden Beispiels erläutert.

In einem verteilten Quantenschaltkreis enthält jede Partition eine bestimmte Anzahl von Qubits. Wie in Kapitel 2 erwähnt, existiert ein teleportiertes Qubit, das als **migriertes Qubit** [ZMHH18] bezeichnet wird, nicht mehr am Ursprungsort und muss zurückteleportiert werden, um am Ursprungsort wiederverwendet zu werden. Jedes globale Gatter erfordert daher mindestens zwei Teleportationen: Zuerst migriert ein Qubit zum Ziel und anschließend wird das migrierte Qubit nach Ausführung des globalen Gatters zurück in das Heim-Subsystem teleportiert. Wenn jedoch ein Qubit nach dem Teleportieren zum

Ziel nicht mehr am Ursprungsort benötigt wird, kann seine Rückkehr verzögert werden. Dadurch können mehrere globale Gatter am Ziel mit derselben vorherigen Teleportation ausgeführt werden, ohne die Qubit-Information erneut zu teleportieren, da sie bereits dort vorhanden sind. Auf diese Weise können mehrere globale Gatter hintereinander mit nur einer Teleportation ausgeführt werden.

Algorithm 4: Hauptalgorithmus zur Bestimmung der Teleportationskosten

```
1  Function: min_telep(gate_list, part_qbits):
2      global_gates ← find_global_gates(gate_list, part_qbits)
3      local_gates ← find_local_gates(gate_list, part_qbits)
4      executed_gates ← []
5      num_teleportations ← 0
6      foreach gate ∈ gate_list do
7          if gate ∈ local_gates then
8              executed_gates.append(gate)
9          end
10     end
11     i ← 0
12     while i < len(global_gates) do
13         j ← i + 1 while j < len(global_gates) do
14             if check_partition_crossing(global_gates[i], global_gates[j], part_qbits)
                 and set(global_gates[i]).intersection(set(global_gates[j])) then
15                 j ← j + 1
16             end
17             else
18                 break
19             end
20         end
21         if j > i + 1 then
22             executed_gates += global_gates[i:j]
23             num_teleportations += 2
24             i ← j
25         end
26         else
27             executed_gates.append(global_gates[i])
28             num_teleportations += 2
29             i ← i + 1
30         end
31     end
32     return num_teleportations
33 end
```

Algorithmus 4 beschreibt die Strategie zur Bestimmung der Teleportationskosten in einem verteilten Quantenschaltkreis. Dabei werden eine Liste von Quantengattern und eine Liste von partitionierten Qubits als Eingabe verwendet und die Anzahl der benötigten Teleportationen als Ausgabe zurückgegeben. Zunächst werden die globalen Gatter und die lokalen Gatter identifiziert. Für jedes Quantengatter wird geprüft, ob es aus einem

Qubit oder aus zwei lokalen Qubits besteht. Falls dies der Fall ist, wird das Gatter als bereits ausgeführt markiert. Anschließend durchläuft der Algorithmus dann die Liste der globalen Gatter und überprüft, ob diese die gleichen Partitionen kreuzen und gemeinsame Qubits aufweisen. Wenn ja, werden sie zusammen ausgeführt und es werden insgesamt zwei Teleportationen benötigt. Andernfalls wird das Gatter einzeln ausgeführt und es werden ebenfalls zwei Teleportationen benötigt. Abschließend gibt der Algorithmus die Gesamtzahl der Teleportationen aus.

Im Folgenden wird die Strategie an einem Beispiel erläutert: Der verteilte Quantenschaltkreis in Abbildung 4.1 ist in drei Partitionen aufgeteilt und besteht aus vier Qubits und neun Quantengattern. Der erste Gatter G1 ist ein lokales Gatter und muss nicht durch Teleportation ausgeführt werden, da er sich nur in Partition 1 befindet. G2 hingegen ist ein globales Gatter, da es von Partition 2 startet und in Partition 1 endet. Das nächste Gatter, G3, ist ebenfalls ein globales Gatter. Hier wird geprüft, ob es gemeinsame Qubits gibt, was in diesem Fall nicht der Fall ist, da G2 die Partitionen 1 und 2 kreuzt. G3 hingegen kreuzt die Partitionen 1, 2 und 3. Daher muss G2 allein durch zwei Teleportationen ausgeführt werden und G3 ist noch nicht ausgeführt. Dann werden die nächsten Gatter auf gemeinsame Qubits überprüft. G4 ist ein lokales Gatter und wird daher lokal in seiner Partition ausgeführt. G5 ist ein globales Gatter und teilt ein gemeinsames Qubit mit G3, nämlich q3, da beide Gatter die Partitionen 1, 2 und 3 durchlaufen. Die Ausführung dieser beiden Gatter ist noch nicht abgeschlossen. G6 ist ein lokales Gatter und wird ebenfalls lokal ausgeführt. G7 ist ein globales Gatter, das mindestens ein gemeinsames Qubit q3 mit G3 und G5 aufgrund der Kreuzung derselben Partitionen aufweist. G8 ist ein lokales Gatter und wird lokal ausgeführt. Das letzte Gatter G9 ist global, kreuzt dieselben Partitionen wie G3, G5 und G7 und hat ein gemeinsames Qubit q3 mit ihnen. Daher können G3, G5, G7 und G9 zusammen durch zwei Teleportationen ausgeführt werden. Insgesamt verfügt dieser verteilte Quantenschaltkreis über vier lokale Gatter und fünf globale Gatter und erfordert insgesamt vier Teleportationen, um ausgeführt zu werden.

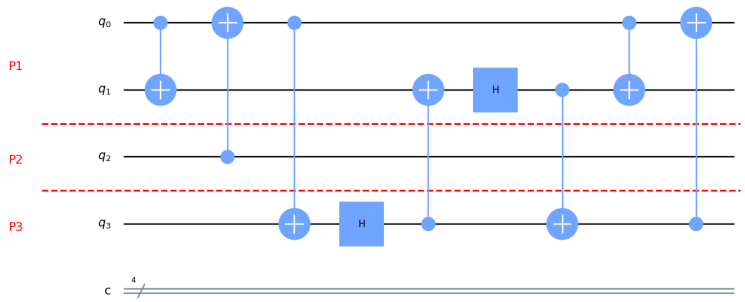


Abbildung 4.1: Figure4 [ZMHH18] in 3 Partitionen aufgeteilt

4.2 Graphpartitionierungsalgorithmen

In diesem Abschnitt wird zunächst die Modellierung von Quantenschaltkreisen erläutert, um verteilte Quantenschaltkreise zu erhalten. Am Ende werden die Methoden durch die Verwendung von Pseudocode beschrieben.

Zunächst wird ein beliebiger Quantenschaltkreis durch einen ungerichteten, gewichteten Graphen dargestellt, wie in Abbildung 2.14 gezeigt. Jeder Quantenschaltkreis besteht aus n Qubits $\{q_1, q_2, \ldots, q_n\}$ und m Quantengatter. Die Anzahl der Qubits bestimmt die Anzahl der Knoten im Graphenmodell. Daher ist die Menge der Knoten für n Qubits in dem Graphenmodell $V = \{q_1, q_2, \ldots, q_n\}$. Um ein ungerichtetes gewichtetes Graphenmodell zu erstellen, werden die Verbindungen zwischen den Qubits in den Gattern als Kantenmenge E betrachtet. Für Ein-Qubit-Gatter wird keine Kante definiert. CNOT-Gatter werden repräsentiert wie folgt:

$$E_i = \{(q_j, q_k) \,|g_i(j,k) \in Schaltkreis; j,k : 1, \ldots, n\}$$

Als Beispiel: Wenn ein Zwei-Qubit-Gatter $g_i(1,2)$ zur Ausführung zwei Qubits q_1 und q_2 benötigt, wird die Kante zwischen q_1 und q_2 als $e_i = (q_1, q_2)$ dargestellt.

Die Kanten werden zunächst mit einem Gewicht von eins erstellt. Wenn mehrere Gatter zwei Qubits verbinden, wird das Gewicht der entsprechenden Kante inkrementiert: $w(e_i) = w(e_i) + 1$. Dadurch entsteht ein ungerichteter, gewichteter Graph, bei dem das Gewicht jeder Kante die Anzahl der Gatter darstellt, die diese beiden Qubits zur Ausführung benötigen.

Mit dem Graphmodell werden die Knoten in Partitionen aufgeteilt. Durch Anwendung eines geeigneten Partitionierungsalgorithmus auf das Graphmodell erhält man K Partitionen $\{P_1, P_2, ..., P_K\}$, wobei jede Partition eine bestimmte Anzahl von Knoten und Kanten enthält und Verbindungen zu anderen Partitionen haben kann.

Jede Partition P_i des Graphen repräsentiert einen Teil des verteilten Quantenschaltkreises und die Knoten in jeder Partition repräsentieren Qubits in dieser Partition. Die Kanten zwischen zwei Knoten innerhalb einer Partition deuten auf ein lokales Gatter hin. Die Kanten hingegen, die die Partitionen verbinden, repräsentieren globale Gatter, deren Qubits sich in mehr als einer Partition befinden. Das bedeutet, dass $w(e_{k1})$ globale Gatter Qubits q_i und q_j verwendet, wenn es eine Kante e_{k1} von q_i der Partition P_x zu q_j der Partition P_y gibt und das Gewicht dieser Kante durch $w(e_{k1})$ angegeben wird. Die Summe der Gewichte dieser Kanten über alle Partitionen $W(E_K) = w(e_{k1}) + w(e_{k2}) + \ldots$ bestimmt letztendlich die Anzahl der globalen Gatter.

Wenn die Knoten logisch in den Partitionen verteilt sind, wird die Anzahl der globalen Quantengatter minimiert, was letztendlich weniger Teleportationen erfordert, um den Quantenschaltkreis auszuführen. Nachdem wir die Graphpartitionierungsalgorithmen implementiert haben, werden die Teleportationskosten anhand Algorithmus 4 bestimmt.

4.2.1 Kernighan-Lin-Algorithmus

Algorithmus 5 stellt die erste Methode zur Graphpartitionierung von Quantengattern dar. Zunächst wird ein Graphmodell erstellt, indem jeder aufeinanderfolgende Knoten in jedem Gatter betrachtet wird und eine Kante zwischen diesen Knoten dem Graphen hinzugefügt wird. Dabei wird gezählt, wie oft diese Kante vorkommt.

Die Funktion *multi_kerlin* erwartet als Eingabe eine Liste von Quantengattern und eine Anzahl von Partitionen. Sie gibt eine Partitionierung der Knoten des Graphen in k Teile zurück. Hierbei wird die Python-Bibliothek **NetworkX** verwendet, die Graphen und Netze erstellen, manipulieren und analysieren kann [HSSC08]. Wenn die Anzahl der Partitionen gleich 2 ist, wird eine Kernighan-Lin-Bisektion auf den Graphen angewendet. Andernfalls wird der Graph in Teilgraphen unterteilt, bis die Anzahl der Teilgraphen der gewünschten Anzahl von Partitionen entspricht. Jeder Teilgraph wird mit Hilfe der Kernighan-Lin-Bisektion in zwei Teile unterteilt und die entstehenden Teilgraphen werden in einer Liste *subgraphs* gespeichert. Falls es mehr als k Teilgraphen gibt, werden die beiden kleinsten Teilgraphen ausgewählt und zu einem einzigen Teilgraphen verschmolzen. Dieser neue Teilgraph wird der Liste *subgraphs* hinzugefügt, und die beiden ursprünglichen Teilgraphen werden daraus entfernt. Schließlich gibt der Algorithmus die resultierende Partitionierung zurück, wobei jede Partition eine Gruppe von Qubits enthält.

Algorithm 5: Erweiterter Kernighan-Lin-Algorithmus

```
Function: multi_kerlin(gate_list, k):
    G ← nx.Graph()
    edges ← defaultdict(int)
    for gate in gate_list do
        for i in range(1, len(gate)) do
            edge ← tuple(sorted([gate[i − 1], gate[i]]));
            if edge in edges then
                edges[edge] ← edges[edge] + 1
            end
            else
                edges[edge] ← 1
            end
        end
    end
    for edge, count in edges.items() do
        G.add_edge(edge[0], edge[1], weight=count)
    end
    if k = 2 then
        partition ← nx.community.kernighan_lin_bisection(G, seed=0)
    end
    else
        subgraphs ← [G]
        while len(subgraphs) < k do
            new_subgraphs ← []
            for subgraph in subgraphs do
                for part in nx.community.kernighan_lin_bisection(subgraph,
                  seed=0) do
                    new_subgraphs.append(subgraph.subgraph(part))
                end
            end
            subgraphs ← new_subgraphs
            if len(subgraphs) > k then
                smallest_subgraphs ← heapq.nsmallest(2, subgraphs, key=lambda
                  sg: sg.number_of_nodes())
                merged_subgraph ← nx.compose_all(smallest_subgraphs)
                  subgraphs.remove(smallest_subgraphs[0])
                  subgraphs.remove(smallest_subgraphs[1])
                  subgraphs.append(merged_subgraph)
            end
        end
        partition ← [set(subgraph.nodes) for subgraph in subgraphs]
    end
    return partition
end
```

4.2.2 Spektrale Partitionierung

Algorithm 6: Spektrale Partitionierung

```
1  Function: spectral_partitioning(gate_list, k):
2      G ← nx.Graph()
3      edges ← defaultdict(int)
4      for gate in gate_list do
5          for i in range(1, len(gate)) do
6              edge ← tuple(sorted([gate[i − 1], gate[i]]));
7              if edge in edges then
8                  edges[edge] ← edges[edge] + 1
9              end
10             else
11                 edges[edge] ← 1
12             end
13         end
14     end
15     for edge, count in edges.items() do
16         G.add_edge(edge[0], edge[1], weight=count)
17     end
18     l_norm ← nx.normalized_laplacian_matrix(G).todense()
19     eigenvalues, eigenvectors ← np.linalg.eig(l_norm)
20     idx ← np.argsort(eigenvalues)[1:k]
21     eigenvectors ← eigenvectors[:, idx]
22     kmeans ← KMeans(n_clusters=k, random_state=0,init='k-means++')
23     clusters ← kmeans.fit_predict(eigenvectors)
24     partition ← [[] for _ in range(k)]
25     for i, qubit in enumerate(G.nodes()) do
26         partition[clusters[i]].append(qubit)
27     end
28     max_size ← max(len(p) for p in partition)
29     min_size ← min(len(p) for p in partition)
30     while max_size - min_size > 1 do
31         max_idx ← np.argmax([len(p) for p in partition])
32         min_idx ← np.argmin([len(p) for p in partition])
33         qubit ← partition[max_idx].pop()
34         partition[min_idx].append(qubit)
35         max_size ← max(len(p) for p in partition)
36         min_size ← min(len(p) for p in partition)
37     end
38     return partition
39 end
```

Dieser Pseudocode 6 implementiert den Algorithmus der spektralen Partitionierung für Quantenschaltkreise und erzeugt eine Liste von Partitionen, die jeweils eine Untergruppe von Qubits enthalten.

Der Algorithmus benötigt als Eingabe eine Liste von Gattern sowie eine Anzahl von

Partitionen, die erstellt werden sollen. Zu Beginn des Algorithmus wird der Quantenschaltkreis in ein Graphmodell umgewandelt. Anschließend wird die normierte Laplace-Matrix des Graphen G berechnet und die zugehörigen Eigenwerte und -vektoren bestimmt. Die Eigenvektoren werden dann auf die ersten k sortierten Eigenwerte minimiert, um k Cluster von Qubits zu bilden, die später verwendet werden, um die Partitionen zu erstellen.

Der Algorithmus weist jedem Qubit im Graphen G ein Cluster zu, indem er jedes Qubit dem Cluster zuweist, dessen zugehöriger Eigenvektor im K-Means-Clustering den kleinsten Abstand zum Qubit aufweist. Die Implementierung des K-Means-Algorithmus wird dabei mithilfe von Scikit-learn [PVG+11] durchgeführt. Scikit-learn ist eine umfangreiche Python-Bibliothek für maschinelles Lernen, die viele Algorithmen für Klassifizierung, Regression, Clustering, Dimensionsreduktion und Modellauswahl sowie Funktionen für Datenvorbereitung, -manipulation und -visualisierung bietet. Am Ende gibt der Algorithmus die resultierende Partitionierung zurück, wobei jede Partition eine Gruppe von Qubits enthält.

4.3 Genetischer Algorithmus

In diesem Abschnitt wird ein heuristischer Ansatz auf Basis von einem GA vorgeschlagen und untersucht, um eine optimierte Aufteilung der Qubits in K Partitionen mit minimierten Teleportationskosten zu finden. Jedes Chromosom, das die Individuen in der Startpopulation repräsentiert, enthält eine Konfiguration, deren Fitness durch den Algorithmus 4 berechnet wird und dann durch die genetischen Operatoren optimiert wird. Abbildung 4.2 zeigt den Ablauf des gesamten Prozesses und die Einzelheiten des vorgeschlagenen Ansatzes werden in den folgenden Unterabschnitten erläutert.

4.3.1 Kodierung der Chromosomen und Anfangspopulation

Eine der wichtigsten Komponenten bei genetischen Algorithmen ist die Kodierung der Chromosomen. In diesem Algorithmus wurde eine Chromosomenstruktur verwendet, deren Elemente die Partitionierung der Qubits darstellen. Jedes Gen wird je nach Zugehörigkeit zu einer bestimmten Partition von Qubits mit einer Nummer von 1 bis k belegt. Im folgenden Chromosom 4.3 ist beispielsweise Qubit 1 in Partition 1, Qubit 2 in Partition 2, Qubit 3 in Partition 1, Qubit 4 in Partition 3 und so weiter. Die Länge des Chromosoms ergibt sich somit aus der Anzahl der Qubits, die ein beliebiger Quantenschaltkreis hat. Jede Konfiguration kann eine potenzielle Lösung für das Problem sein. Der Algorithmus beginnt mit einer zufälligen Anfangspopulation der Chromosomen.

4.3.2 Fitness Funktion und Selektionsstrategie

Die Fitnessfunktion zur Bestimmung der Teleportationskosten wurde im Algorithmus 4 definiert. Zur Selektion wird eine Kombination aus der **Turnier-Selektion** und dem **Elitismus** angewendet. Elitismus [Ree10] bedeutet, dass die besten Chromosomen aus der vorherigen Generation direkt in die nächste Generation übernommen werden. Die Kombination aus Elitismus und Turnier-Selektion ermöglicht es, gute Lösungen aus vorherigen Generationen zu behalten und gleichzeitig neue Lösungen durch das Turnierauswahlverfahren zu finden.

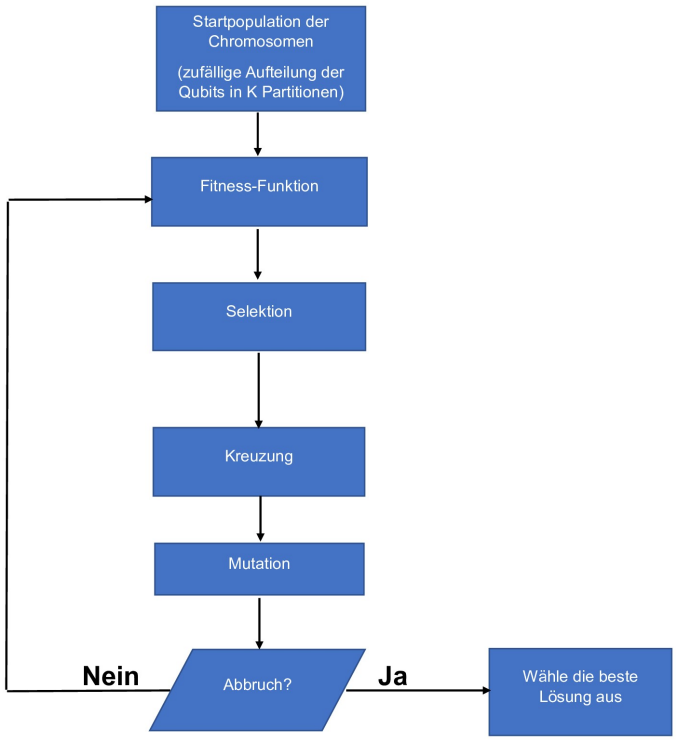


Abbildung 4.2: Das Flussdiagramm der vorgeschlagenen Methode

1	2	1	3	1	3	2

Abbildung 4.3: Beispielhafte Darstellung der Chromosomen

4.3.3 Kreuzung und Mutation

Zur Kreuzung wird die PMX-Methode (Partially-Mapped-Crossover) [DFAP17] angewendet. Diese Methode eignet sich besonders gut für Permutationsfolgen, da sie die Entstehung von Duplikaten verhindert. Zunächst werden für die zwei Kinder Kopien der Eltern erstellt und mittels Zufallszahlen zwei Trennstellen bestimmt. Anschließend werden die Positionen zwischen den Trennstellen nacheinander bearbeitet, wobei ein Wert, der von einem Kind auf das andere übernommen werden soll, auf dem übernehmenden Kind gesucht wird und die Werte auf beiden Positionen des übernehmenden Kindes getauscht werden. In dieser Weise wird mit allen Werten zwischen den Trennstellen auf beiden Kindern verfahren, wie in Abbildung 4.4 dargestellt.

Jeder Nachkomme hat die Chance, einer Mutation zu unterliegen. Der **Swap-Operator** funktioniert durch zufällige Auswahl von zwei Elementen der Permutationsfolge und Austausch ihrer Werte, wie in Abbildung 4.5 dargestellt. Mutation und Kreuzung erfolgen jeweils mit ihren entsprechenden Wahrscheinlichkeiten, siehe Tabelle 5.2.

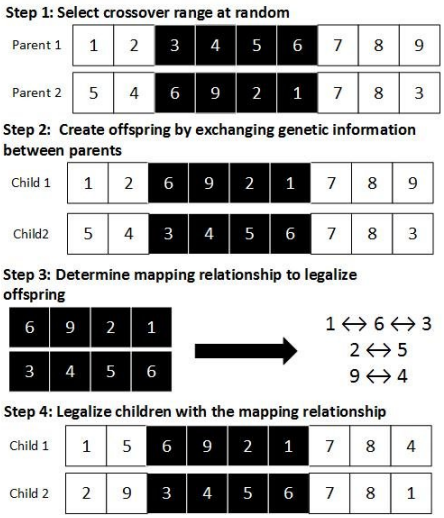


Abbildung 4.4: Beispiel für die PMX-Kreuzungsmethode [DFAP17]

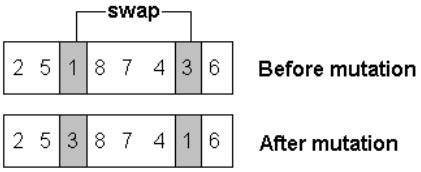


Abbildung 4.5: Beispiel für die Swap-Mutation [EM20]

4.4 Hybridgenetische Algorithmen

Im Folgenden werden zwei HGA vorgestellt, die denselben Aufbau haben, aber unterschiedliche Graphpartitionierungsalgorithmen verwenden. Der erste GA wird in Verbindung mit dem Kernighan-Lin-Algorithmus eingesetzt und der zweite mit der spektralen Partitionierung.

Zuerst wird der Quantenschaltkreis in ein Graphmodell umgewandelt und anschließend werden die Graphpartitionierungsalgorithmen 5, 6 verwendet, um die Qubits in K Partitionen aufzuteilen. Zum Schluss werden die GA eingesetzt, um die Teleportationskosten durch die optimale Anordnung der globalen Quantengatter zu minimieren. Abbildung 4.7 zeigt den Ablauf des gesamten Prozesses.

In diesen beiden HGA wurde eine Chromosomenstruktur verwendet, bei der die Elemente die Anordnung der globalen Quantengatter darstellen. Jedes Gen wird mit einer Nummer von 1 bis n belegt, je nach Zugehörigkeit, wie in Abbildung 4.6 dargestellt. Die Länge des Chromosoms ergibt sich aus der Anzahl der globalen Quantengatter, für die die Graphpartitionierungsalgorithmen verantwortlich sind. Jede Konfiguration kann eine potenzielle Lösung für das Problem sein. Der Algorithmus beginnt mit einer zufälligen Anfangspopulation der Chromosomen.

2	**4**	**1**	**3**	**5**	**7**	**6**

Abbildung 4.6: Beispielhafte Darstellung der Chromosomen

Die Fitnessfunktion zur Bestimmung der Teleportationskosten wurde im Algorithmus 4 definiert. Für die Selektion wird eine Kombination aus der Turnier-Selektion und dem Elitismus verwendet. Durch den Elitismus können gute Lösungen aus vorherigen Generationen beibehalten werden, während die Turnier-Selektion neue Lösungen finden kann.

Für die Kreuzung der Chromosomen wird in beiden Algorithmen die PMX (Partially-Mapped-Crossover) Methode [DFAP17] verwendet, da sie besonders effektiv bei Permutationsfolgen ist und die Entstehung von Duplikaten verhindert. Als Mutationsvariante nutzen beide Algorithmen den Swap-Operator.

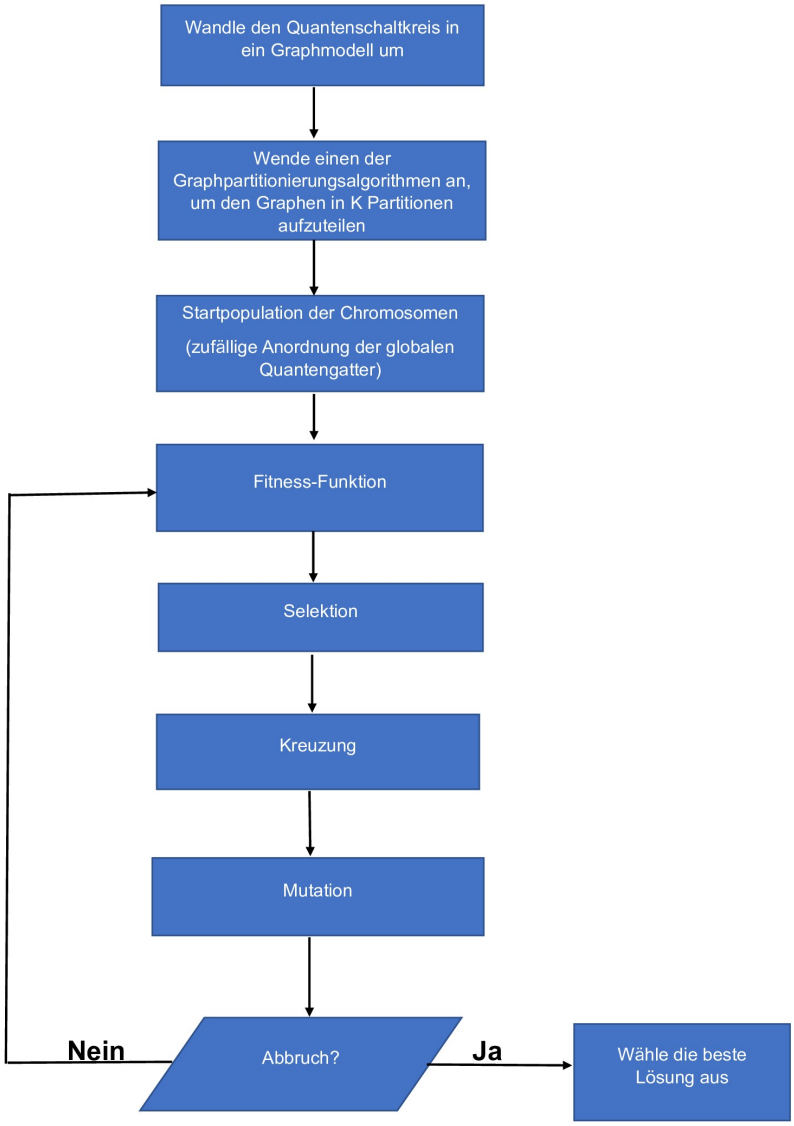


Abbildung 4.7: Das Flussdiagramm der zwei HGA

5 Ergebnisse und Diskussion

Im Folgenden werden die Ergebnisse der bereits vorgestellten Ansätze zur Minimierung globaler Quantengatter und damit verbundener Teleportationskosten mithilfe von verschiedenen Tabellen und Abbildungen präsentiert und im Anschluss interpretiert. Die Analyse dieser Ergebnisse liefert wertvolle Einblicke in die Wirksamkeit und Effizienz der angewandten Methoden sowie deren Auswirkungen auf die Minimierung der Teleportationskosten.

Die GA-Parameter, die in Tabelle 5.1 und 5.2 präsentiert sind, wurden durch verschiedene Simulationen angepasst und die Ergebnisse analysiert, um diese Parameter unter optimalen Bedingungen einzustellen und bessere Lösungen zu erzeugen. Hierbei beschreibt Pc die Kreuzungsrate und Pm die Mutationsrate. Zusätzlich wurde die Anzahl der Generationen als Abbruchbedingung in den vorgeschlagenen GA berücksichtigt.

Tabelle 5.1: Parameter der zwei hybriden genetischen Algorithmen (GA+KL und GA+SP)

Populations-größe	**Selektions-strategie**	**Eliten-individuen**	**Kreuzung**	**Mutation**	**Pc**	**Pm**	**Generationen**
400	Turnierselektion+ Elitismus	4	PMX	Swap	0.9	0.1	200

Tabelle 5.2: Parameter des GA

Populations-größe	**Selektions-strategie**	**Eliten-individuen**	**Kreuzung**	**Mutation**	**Pc**	**Pm**	**Generationen**
100	Turnierselektion+ Elitismus	2	PMX	Swap	0.9	0.1	100

Tabelle 5.3 stellt den Vergleich der Ergebnisse dieser Arbeit mit dem bereits erwähnten Ansatz[HMZMH20] in Bezug auf die Anzahl globaler Gatter dar. In den ersten drei Spalten werden die Benchmark-Schaltkreise, die Anzahl von Qubits und die Anzahl von Partitionen angegeben. Die verbleibenden vier Spalten zeigen die Anzahl globaler Gatter zwischen den verschiedenen Methoden. Bei sorgfältiger Analyse der Tabelle 5.3 wird ersichtlich, dass die vorgeschlagenen Ansätze im Gegensatz zu den Ergebnissen im wissenschaftlichen Artikel[HMZMH20] für die QFT-Schaltkreise die gleiche Anzahl globaler Gatter aufweisen. Es konnte jedoch beobachtet werden, dass bei den Schaltkreisen Parity_247, Figure4 und Sym9_147 eine Verbesserung hinsichtlich der Minimierung von globalen Gattern durch die vorgeschlagenen Methoden vorliegt, mit Ausnahme von 4gt5_76 und 4mod7.

Es zeigt sich noch, dass der Kernighan-Lin-Algorithmus (KL) und der GA ähnliche Ergebnisse hinsichtlich der Minimierung von globalen Gattern bei K = 2 und K = 4 liefern, was auf eine vergleichbare Effektivität beider Algorithmen für diese spezifischen Werte

von K hinweist. Im Gegensatz dazu liefert die Spektrale Partitionierung (SP) entweder schlechtere oder gleich gute Ergebnisse im Kontrast zu den anderen Methoden für diese Werte von K. Daraus kann geschlossen werden, dass SP an Effektivität verlieren kann, wenn es mit komplexeren Schaltkreisen und unregelmäßigen Graphen konfrontiert wird. Weiterhin zeigt sich bei der Betrachtung der untersuchten Ansätze für K = 3 eine Variation in der Anzahl der globalen Gatter. Die Auswertung der vorliegenden Tabelle legt nahe, dass der GA im Vergleich zur SP bessere Ergebnisse in Bezug auf die Minimierung der globalen Gatter erzeugt. Der GA ist ein suchbasierter Ansatz, der globale Optimierungsfähigkeiten besitzt und in der Lage ist, verschiedene Lösungsräume zu erkunden. Es kann gute Ergebnisse bei komplexen Schaltkreisen und vielfältigen Partitionierungsaufgaben liefern, ist skaliert und anpassungsfähig an verschiedene Problemstellungen. Es kann jedoch in der Ausführung zeitaufwändig sein und erfordert oft Parameterabstimmung und Anpassung für optimale Leistung. KL ist ein effizienter Algorithmus zur Partitionierung von Schaltkreisen und einfach zu implementieren. Es kann gute Ergebnisse bei bestimmten Schaltkreisen und Partitionierungsaufgaben liefern. Allerdings handelt es sich um einen heuristischen Ansatz, der nicht immer das globale Optimum findet und anfällig für lokale Minima sein kann.

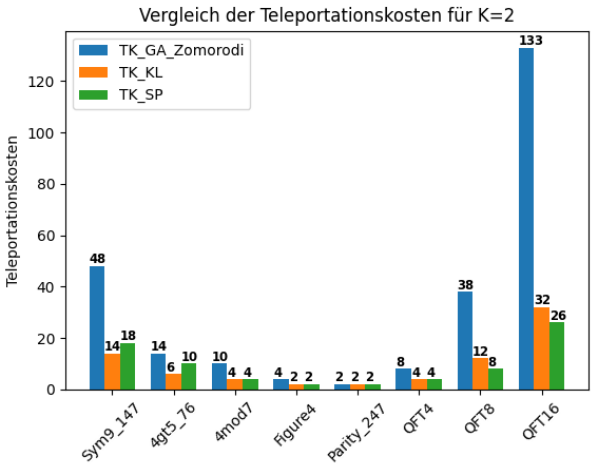


Abbildung 5.1: Vergleich der Teleportationskosten für K=2 zwischen GA[HMZMH20], KL, SP

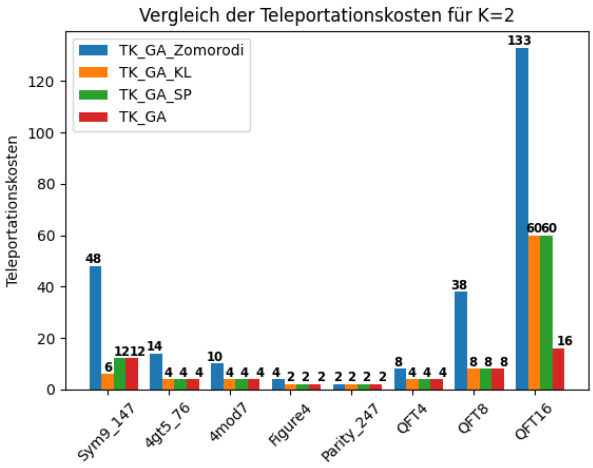


Abbildung 5.2: Vergleich der Teleportationskoste für K=2 zwischen GA[HMZMH20] GA+KL, GA+SP, GA

Die Balkendiagramme in den Abbildungen 5.1 und 5.2 zeigen einen Vergleich der Teleportationskosten für K = 2 Partitionen zwischen den vorgeschlagenen Ansätzen und dem GA [HMZMH20]. Die horizontalen Achsen repräsentieren die Benchmark-Schaltkreise für K = 2, während die vertikalen Achsen die Anzahl der Teleportationen darstellen. Basierend auf den Ergebnissen lässt sich feststellen, dass die vorgestellten Ansätze zur Minimierung der Teleportationskosten in verteilten Quantenschaltkreisen für den Fall K = 2 im Vergleich zum GA [HMZMH20] signifikant verbesserte Ergebnisse liefern. Es ist jedoch zu beachten, dass im Fall des Schaltkreises Parity_247 keine Verbesserung erkennbar ist. Besonders signifikant sind diese Verbesserungen bei QFT8, QFT16 und Sym9_147, bei denen die Anzahl der Qubits zunimmt. Mit steigender Anzahl von

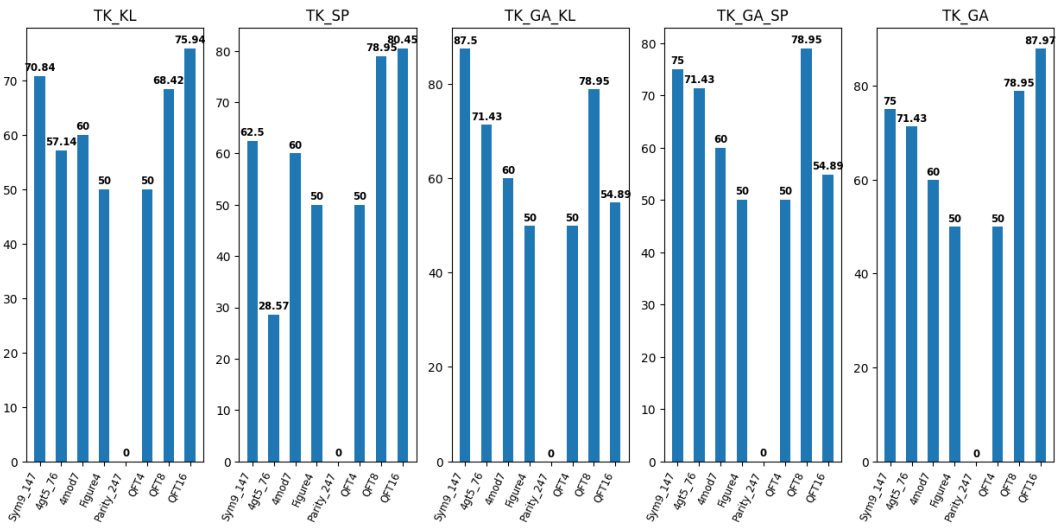


Abbildung 5.3: Prozentuale Verbesserung der Teleportationskosten in den vorgeschlagenen Methoden im Vergleich zu GA[HMZMH20]

Qubits in den Quantenschaltkreisen erhöhen sich die Teleportationskosten in den untersuchten Methoden linear, während sie im GA [HMZMH20] exponentiell ansteigen und stark zunehmen. Abbildung 5.3 besteht aus fünf Balkendiagrammen, die die prozentuale Verbesserung der Teleportationskosten für jeden einzelnen Quantenschaltkreis in den vorgeschlagenen Ansätzen im Kontrast zum GA [HMZMH20] darstellen. Auf der horizontalen Achse sind die Benchmark-Schaltkreise aufgeführt, während die vertikale Achse die prozentuale Verbesserung der Teleportationskosten angibt. Außerdem fasst Tabelle 5.5 die durchschnittliche prozentuale Verbesserung jeder einzelnen Methode im Vergleich zu der Studie [HMZMH20] zusammen. Es wurde festgestellt, dass alle Methoden eine signifikante Verbesserung der Teleportationskosten von über 50% aufweisen.

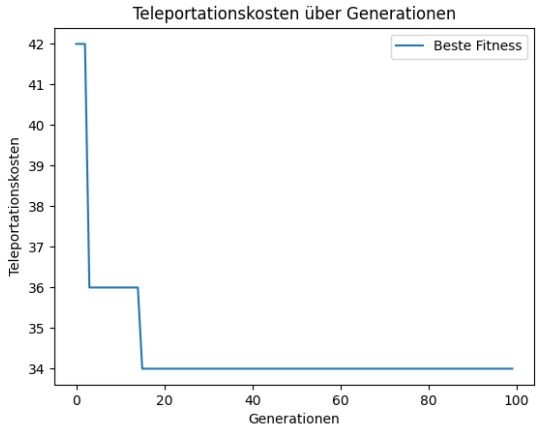


Abbildung 5.4: Fitnessverlauf des GA für Sym9_147 und K = 3

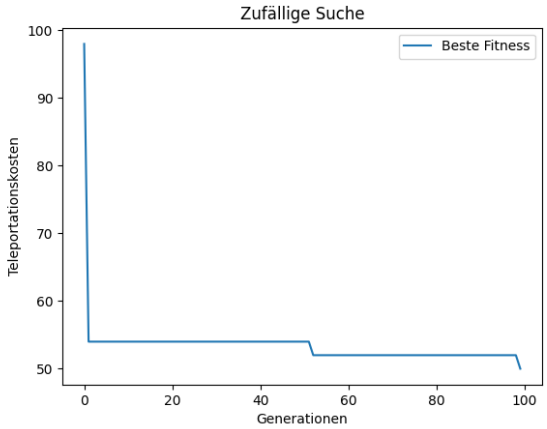


Abbildung 5.5: Zufällige Suche des GA für Sym9_147 und K = 3

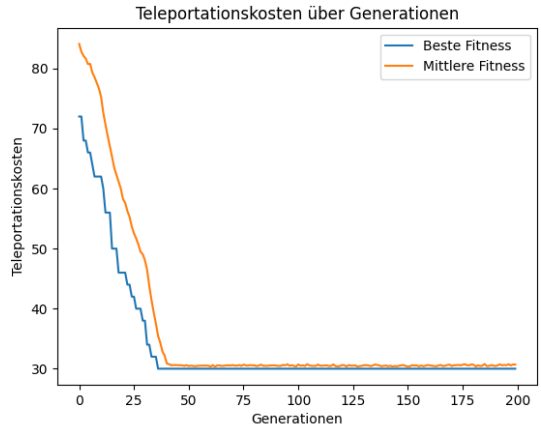


Abbildung 5.6: Fitnessverlauf vom GA+KL für QFT8 und K = 4

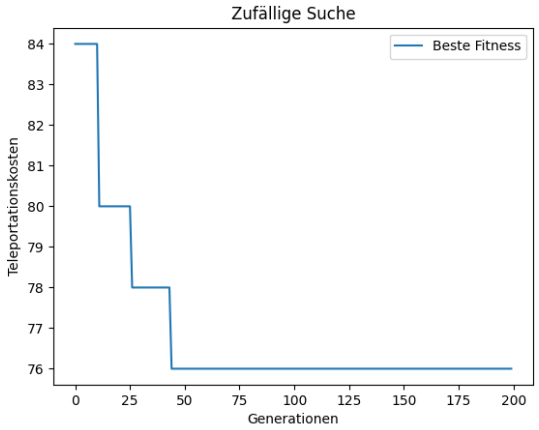


Abbildung 5.7: Zufällige Suche vom GA+KL für QFT8 und K = 4

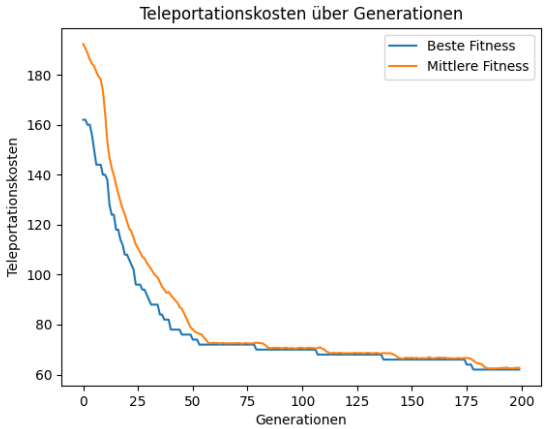


Abbildung 5.8: Fitnessverlauf vom GA+SP für QFT16 und K = 2

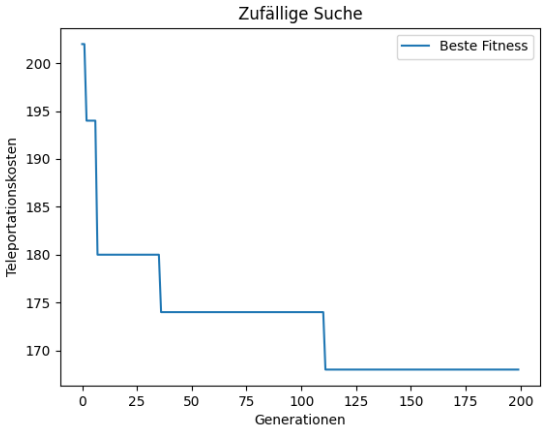


Abbildung 5.9: Zufällige Suche vom GA+SP für QFT16 und K = 2

Um die Leistungsfähigkeit des GA und der beiden HGA-Varianten zu überprüfen, wurden zufällige Suchläufe durchgeführt. Dabei wurden für jeden Lauf die gleiche Populationsgröße und Anzahl der Generationen verwendet, wie in den Tabellen 5.2 und 5.1 angegeben. Zusätzlich nutzte die zufällige Suche die gleiche Chromosomenkodierung wie der entsprechende GA/HGA und die gleiche Fitnessfunktion (siehe Algorithmus 4) zur Bestimmung der Teleportationskosten. Der Hauptunterschied zwischen der zufälligen Suche und dem GA/HGA besteht darin, dass bei der zufälligen Suche keine genetischen Operatoren (Selektion, Rekombination und Mutation) angewendet wurden.

Diese Untersuchung wurde für mehrere Quantenschaltkreise durchgeführt. Zur Veranschaulichung des Verlaufs des Fitnesswerts über mehrere Generationen basierend auf der zufälligen Suche des GA für denselben Quantenschaltkreis und dieselbe Anzahl von Partitionen dient Abbildung 5.5. Auf der horizontalen Achse sind die Anzahl der Generationen aufgetragen, während die vertikale Achse die Teleportationskosten darstellt.

Die gleiche Vorgehensweise wurde für QFT8 und QFT16 angewendet, wobei die beiden HGA-Varianten analysiert wurden. Die Abbildungen 5.6 und 5.8 zeigen die besten und durchschnittlichen Fitnesswerte des GA, während die Abbildungen 5.7 und 5.9 den Verlauf des Fitnesswerts aufgrund der zufälligen Suche darstellen.

Die Ergebnisse zeigen eine eindeutige Tendenz: Die zufällige Suche führt zu deutlich schlechteren Ergebnissen als die Verwendung der GA und der beiden HGA. Dies ist darauf zurückzuführen, dass der GA und die HGA eine Kombination aus Exploration (Erkunden des Suchraums) und Exploitation (Ausnutzen guter Lösungen) verwenden, um nach optimalen Lösungen zu suchen. Durch die Verwendung von Selektion, Rekombination und Mutation kann der GA das Suchraumpotenzial besser ausschöpfen. Im Gegensatz dazu beruht die zufällige Suche ausschließlich auf zufällig erzeugten Lösungen und hat keine Mechanismen zur Anpassung und Verbesserung der Lösungen. Insgesamt belegen diese Ergebnisse die hohe Effektivität des GA und der beiden HGA-Varianten bei der Minimierung der Teleportationskosten in verteilten Quantenschaltkreisen im Gegensatz zur zufälligen Suche.

Darüber hinaus zeigt Tabelle 5.4 die Ergebnisse der verschiedenen Ansätze bezüglich der Teleportationskosten für neun Quantenschaltkreise. Da die Ergebnisse für mehr als zwei Partitionen (dargestellt als N.A.) nicht verfügbar waren, wurden die Ergebnisse für $K = 3$ und $K = 4$ in Tabelle 5.4 gemessen und präsentiert. Zusätzlich wurden sie in den Abbildungen 5.10 und 5.11 graphisch veranschaulicht. Es ist zu beobachten, dass mit steigender Anzahl globaler Gatter auch die Anzahl der Teleportationen zunimmt, was auf die steigende Anzahl der Partitionen zurückzuführen ist.

Interessanterweise zeigen die untersuchten Ansätze für QFT4 ($K = 2$ und $K = 4$), Figure4 ($K = 2$), Parity_247 ($K = 2$) und 4mod7 ($K = 2$) gleiche Teleportationskosten. Die Ergebnisse verdeutlichen, dass im Durchschnitt die beiden HGA die besten Teleportationskosten erzielen, mit Ausnahme von QFT8 und QFT16. Ein möglicher Grund dafür sein kann, dass die Chromosomenkodierung auf der Anzahl der globalen Gatter basiert und die HGA keine nahezu optimale Anordnung der globalen Gatter findet. Im Gegensatz dazu erzielt der GA die besten Ergebnisse für die QFT-Schaltkreise und übertrifft dabei den KL-Algorithmus und die SP. Zudem wird ersichtlich, dass die Graphpartitionierungsalgorithmen (KL und SP) allein im Durchschnitt die geringste Leistung bei der Minimierung der Teleportationskosten erbringen. Durch die Kombination mit dem GA können jedoch aufgrund der genetischen Operatoren, die auf den Prinzipien der Evolution basieren, signifikant bessere Ergebnisse erzielt werden.

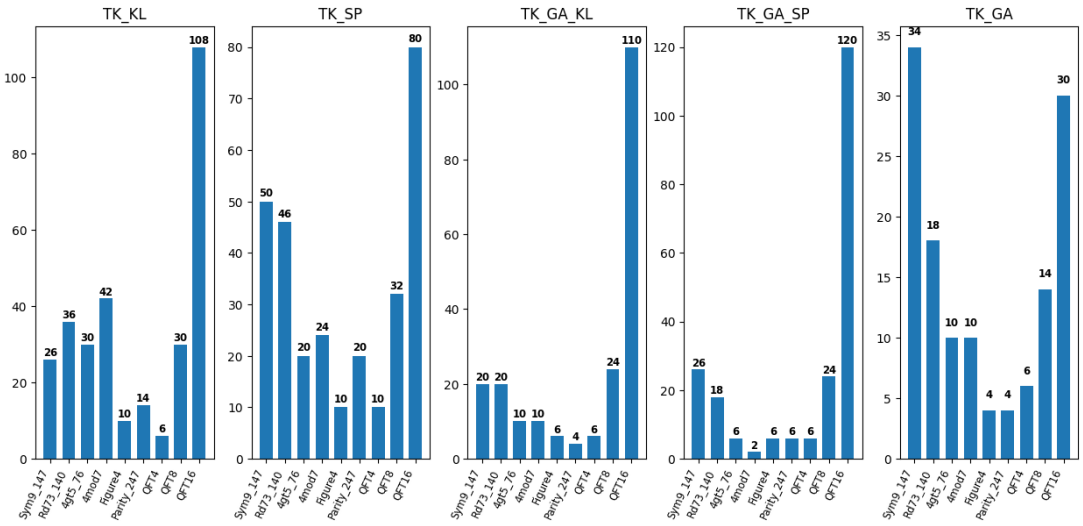


Abbildung 5.10: Vergleich der Teleportationskosten für K = 3

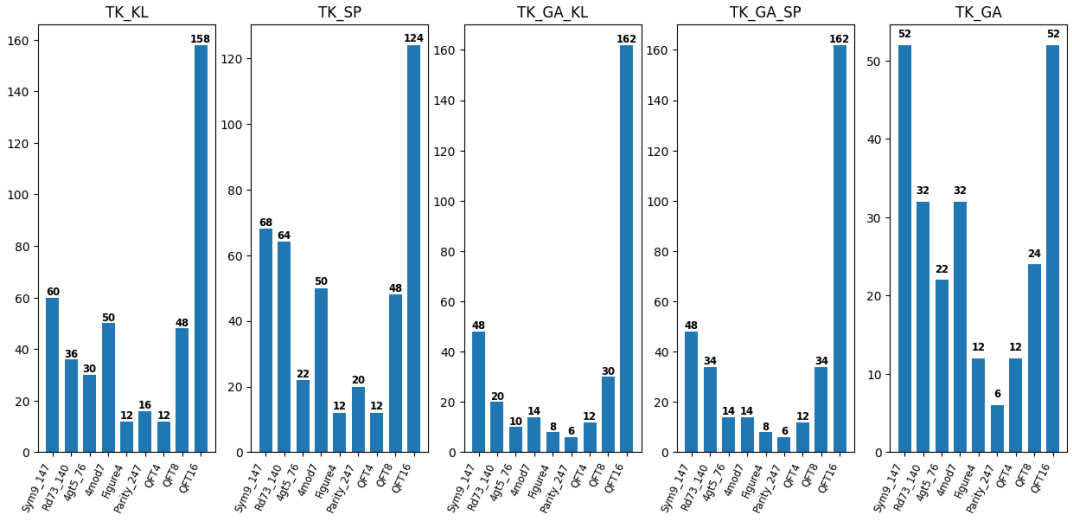


Abbildung 5.11: Vergleich der Teleportationskosten für K = 4

Tabelle 5.3: Vergleich der Anzahl der globalen Gatter(GG) zwischen den folgenden Ansätzen für die ausgewählten Schaltkreise für K Anzahl an Partitionen: GA[HMZMH20], Kernighan-Lin-Algorithmus(KL), Spektrale Partitionierung(SP), Genetischer Algorithmus(GA)

Schaltkreis	**#Qubits**	**#K**	**GG [HMZMH20]**	**#GG (KL)**	**#GG (SP)**	**#GG (GA)**
QFT4	4	2	8	8	8	8
		3	N.A.	10	10	10
		4	N.A.	12	12	12
QFT8	8	2	32	32	32	32
		3	N.A.	40	42	42
		4	N.A.	48	48	48
QFT16	16	2	128	128	128	128
		3	N.A.	160	170	170
		4	N.A.	192	192	192
4gt5_76	5	2	11	13	16	13
		3	N.A.	24	22	21
		4	N.A.	24	25	24
Figure4	4	2	5	4	4	4
		3	N.A.	6	6	5
		4	N.A.	7	7	7
Parity_247	17	2	9	8	8	8
		3	N.A.	12	11	11
		4	N.A.	12	12	12
Sym9_147	12	2	54	45	52	45
		3	N.A.	58	67	65
		4	N.A.	76	78	76
4mod7	5	2	11	18	21	18
		3	N.A.	29	24	24
		4	N.A.	30	30	30
Rd73_140	10	2	N.A.	29	31	29
		3	N.A.	47	46	40
		4	N.A.	47	52	47

Tabelle 5.4: Vergleich der Teleportationskosten(TK) zwischen den folgenden Ansätzen für die ausgewählten Schaltkreise für K Anzahl an Partitionen: GA[HMZMH20], Kernighan-Lin-Algorithmus(KL), Spektrale Partitionierung(SP), Genetischer Algorithmus(GA), Hybribgenetischer Algorithmus 1(GA+KL), Hybridgenetischer Algorithmus 2(GA+SP)

Schaltkreis	**#Qubits**	**#K**	**TK [HMZMH20]**	**TK (KL)**	**TK (SP)**	**TK (GA)**	**TK (GA+KL)**	**TK (GA+SP)**
QFT4	4	2	8	4	4	4	4	4
		3	N.A.	6	10	6	6	6
		4	N.A.	12	12	12	12	12
QFT8	8	2	38	12	8	8	8	8
		3	N.A.	30	32	14	24	24
		4	N.A.	48	48	24	30	34
QFT16	16	2	133	32	26	16	60	60
		3	N.A.	108	80	30	110	120
		4	N.A.	158	124	52	162	162
4gt5_76	5	2	14	6	10	4	4	4
		3	N.A.	30	20	10	10	6
		4	N.A.	30	22	22	10	14
Figure4	4	2	4	2	2	2	2	2
		3	N.A.	10	10	4	6	6
		4	N.A.	12	12	12	8	8
Parity_247	17	2	2	2	2	2	2	2
		3	N.A.	14	20	4	4	4
		4	N.A.	16	20	6	6	6
Sym9_147	12	2	48	14	18	12	6	12
		3	N.A.	26	50	34	20	26
		4	N.A.	60	68	52	48	48
4mod7	5	2	10	4	4	4	4	4
		3	N.A.	42	24	10	10	6
		4	N.A.	50	50	32	14	14
Rd73_140	10	2	N.A.	10	10	10	4	6
		3	N.A.	36	46	18	18	18
		4	N.A.	36	64	32	20	32

Tabelle 5.5: Prozentuale Verbesserung der Teleportationskosten im Vergleich zu [HMZMH20]

Ansatz	**Prozentuale Verbesserung**
TK_SP	51,30%
TK_KL	54,04%
TK_GA+SP	55,03%
TK_GA+KL	56,60%
TK_GA	59,16%

6 Fazit

In Anbetracht der steigenden Anzahl von Qubits in modernen Quantencomputern wird die Handhabung von Quanteninformation zunehmend anspruchsvoller und fehleranfälliger. Daher gewinnt die Betrachtung verteilter Quantencomputer an Bedeutung. In der verteilten Quantenberechnung interagieren verschiedene Teilsysteme miteinander mithilfe des Teleportationsprotokolls, das für die Übertragung der Quanteninformation verantwortlich ist. Ein wesentlicher Aspekt besteht darin, die Anzahl der Teleportationen zu minimieren, um die Genauigkeit der Quantenberechnungen zu steigern, die Fehleranfälligkeit der Qubits zu reduzieren und gleichzeitig den Ressourcenverbrauch effizienter zu gestalten. Dieses Streben ist von grundlegender Bedeutung, um die praktische Anwendbarkeit von Quantencomputern in verschiedenen Anwendungsbereichen zu maximieren.

In dieser Arbeit wurden verschiedene Graphpartitionierungsalgorithmen, darunter der Kernighan-Lin-Algorithmus (KL) und die Spektrale Partitionierung (SP), ein Genetischer Algorithmus (GA) sowie am Ende zwei hybride Genetische Algorithmen (HGA), die aus einer Kombination von Graphpartitionierungsalgorithmen und einem Genetischen Algorithmus bestehen, angewendet, um die Minimierung der Anzahl globaler Quantengatter und der damit verbundenen Teleportationskosten zu erreichen. Die Benchmark-Schaltkreise wurden zuerst in ein Graphmodell umgewandelt und die Knoten mittels der Graphpartitionierungsalgorithmen möglichst gleichmäßig in K Partitionen aufgeteilt. Der GA war für die zufällige Aufteilung der Qubits in Partitionen zuständig, um die Teleportationskosten zu minimieren. Die beiden HGA führten zu einer nahezu optimalen Anordnung der globalen Quantengatter, nachdem die Qubits mit Hilfe der Graphpartitionierungsalgorithmen partitioniert wurden.

Es wurde beobachtet, dass jede der untersuchten Methoden im Vergleich zum Ansatz [HMZMH20] eine deutliche Verbesserung der Teleportationskosten, insbesondere für die QFT-Schaltkreise, um mehr als 50% zeigte. Es wurde auch festgestellt, dass die Verbesserung besonders signifikant war, wenn die Anzahl der globalen Gatter mit gemeinsamen Qubits in einem Schaltkreis höher war und sie logisch in Partitionen entsprechend ihrer Anordnung verteilt waren. Interessanterweise wiesen die beiden HGA die besten Teleportationskosten auf, mit Ausnahme von QFT8 und QFT16. Zusätzlich wurde eine zufällige Suche für mehrere Quantenschaltkreise durchgeführt, um die Effektivität des GA und der beiden HGA zu vergleichen. Die Ergebnisse dieser Suche deuten darauf hin, dass die zufällige Suche nicht in der Lage war, bessere Ergebnisse als die vorgestellten Methoden zu liefern. Darüber hinaus zeigte sich, dass die Graphpartitionierungsalgorithmen im Allgemeinen schlechtere Leistungen im Vergleich zu den anderen angewendeten Methoden erbrachten.

Als zukünftige Arbeit bieten sich die Untersuchung von Matrixmodellen für Quantenschaltkreise, indem Quantengatter in Matrixform präsentiert werden, sowie die Anwendung von Optimierungsalgorithmen wie Simulated Annealing oder Partikelschwarmoptimierung zur weiteren Minimierung der Teleportationskosten an. Darüber hinaus könnte die Anwendbarkeit der untersuchten Methoden auf größere Quantenschaltkreise

mit mehr als vier Partitionen erweitert werden, um deren Effektivität zu überprüfen. Ein weiterer wichtiger Aspekt neben der Minimierung der Teleportationskosten ist die Ausführungszeit von Quantenschaltkreisen. Die großen Interaktionsdistanzen zwischen den Qubits innerhalb eines Quantenschaltkreises können die Generierungszeit des verteilten Quantenschaltkreises erhöhen. Daher ist es wichtig, die Qubits vor der Partitionierung neu anzuordnen und dann einen beliebigen Optimierungsalgorithmus anzuwenden, um die Ausführungszeit zu minimieren.

Abbildungsverzeichnis

2.1 Quantenschaltkreis bestehend nur aus CNOT-Gattern [DZMHNb20] . . . 6
2.2 Verschränkung zweier Qubits . . . 6
2.3 Quantenteleportation mit EPR-Quelle Telamon [Qis23] . . . 7
2.4 Der Schaltkreis für die Quantenteleportation [NC01] . . . 9
2.5 Verteilter Quantenschaltkreis in 2 Partitionen [DZMHNb20] . . . 10
2.6 Beispiel für die binäre Kodierung . . . 10
2.7 Roulette-Wheel-Selektion [SFHBA+18] . . . 12
2.8 Turnier-Selektion [BA21] . . . 12
2.9 One-Point-Crossover [KUT11] . . . 12
2.10 Two-Point-Crossover [KUT11] . . . 13
2.11 Bit-Flip-Mutation [Rif16] . . . 13
2.12 Ein ungerichter Graph G . . . 15
2.13 Zufälliger Quantenschaltkreis [ZMHH18] . . . 19
2.14 Der gewichtete Graph dazu . . . 19

4.1 Figure4 [ZMHH18] in 3 Partitionen aufgeteilt . . . 25
4.2 Das Flussdiagramm der vorgeschlagenen Methode . . . 31
4.3 Beispielhafte Darstellung der Chromosomen . . . 31
4.4 Beispiel für die PMX-Kreuzungsmethode [DFAP17] . . . 32
4.5 Beispiel für die Swap-Mutation [EM20] . . . 32
4.6 Beispielhafte Darstellung der Chromosomen . . . 32
4.7 Das Flussdiagramm der zwei HGA . . . 33

5.1 Vergleich der Teleportationskosten für K=2 zwischen GA[HMZMH20], KL, SP . . . 36
5.2 Vergleich der Teleportationskosten für K=2 zwischen GA[HMZMH20], GA+KL, GA+SP, GA . . . 36
5.3 Prozentuale Verbesserung der Teleportationskosten in den vorgeschlagenen Methoden im Vergleich zu GA[HMZMH20] . . . 37
5.4 Fitnessverlauf des GA für Sym9_147 und K = 3 . . . 37
5.5 Zufällige Suche des GA für Sym9_147 und K = 3 . . . 37
5.6 Fitnessverlauf vom GA+KL für QFT8 und K = 4 . . . 38
5.7 Zufällige Suche vom GA+KL für QFT8 und K = 4 . . . 38
5.8 Fitnessverlauf vom GA+SP für QFT16 und K = 2 . . . 38
5.9 Zufällige Suche vom GA+SP für QFT16 und K = 2 . . . 38
5.10 Vergleich der Teleportationskosten für K = 3 . . . 40
5.11 Vergleich der Teleportationskosten für K = 4 . . . 40

Tabellenverzeichnis

2.1 Transformationen von Bob 8

5.1 Parameter der zwei hybriden genetischen Algorithmen (GA+KL und GA+SP) 35
5.2 Parameter des GA 35
5.3 Vergleich der Anzahl der globalen Gatter(GG) zwischen den folgenden Ansätzen für die ausgewählten Schaltkreise für K Anzahl an Partitionen: GA[HMZMH20], Kernighan-Lin-Algorithmus(KL), Spektrale Partitionierung(SP), Genetischer Algorithmus(GA) 41
5.4 Vergleich der Teleportationskosten(TK) zwischen den folgenden Ansätzen für die ausgewählten Schaltkreise für K Anzahl an Partitionen: GA[HMZMH20], Kernighan-Lin-Algorithmus(KL), Spektrale Partitionierung(SP), Genetischer Algorithmus(GA), Hybribgenetischer Algorithmus 1(GA+KL), Hybridgenetischer Algorithmus 2(GA+SP) 42
5.5 Prozentuale Verbesserung der Teleportationskosten im Vergleich zu [HMZMH20] 42

Literaturverzeichnis

[ABKL07] Alekseev, Vladimir E, Rodica Boliac, Dmitry V Korobitsyn und Vadim V Lozin: *NP-hard graph problems and boundary classes of graphs.* Theoretical Computer Science, 389(1-2):219–236, 2007.

[Ala92] Alander, Jarmo T: *On optimal population size of genetic algorithms.* In: *CompEuro 1992 Proceedings computer systems and software engineering*, Seiten 65–70. IEEE, 1992.

[AMH19] Andres-Martinez, Pablo und Chris Heunen: *Automated distribution of quantum circuits via hypergraph partitioning.* Physical Review A, 100(3):032308, 2019.

[AR04] Andreev, Konstantin und Harald Räcke: *Balanced graph partitioning.* In: *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, Seiten 120–124, 2004.

[BA21] Banihashemian, Seyed Saber und Fazlollah Adibnia: *A novel robust soft-computed range-free localization algorithm against malicious anchor nodes.* Cognitive Computation, 13(4):992–1007, 2021.

[BBC+93] Bennett, Charles H, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres und William K Wootters: *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels.* Physical review letters, 70(13):1895, 1993.

[BBC+95] Barenco, Adriano, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin und Harald Weinfurter: *Elementary gates for quantum computation.* Physical review A, 52(5):3457, 1995.

[BLMS00] Brassard, Gilles, Norbert Lütkenhaus, Tal Mor und Barry C Sanders: *Limitations on practical quantum cryptography.* Physical review letters, 85(6):1330, 2000.

[BOM15] Brabazon, Anthony, Michael O'Neill und Seán McGarraghy: *Natural computing algorithms*, Band 554. Springer, 2015.

[BR02] Blyth, Thomas Scott und Edmund F Robertson: *Basic linear algebra.* Springer Science & Business Media, 2002.

[But16] Butler, Steve: *Algebraic aspects of the normalized Laplacian.* Recent Trends in Combinatorics, Seiten 295–315, 2016.

[CB97] Cleve, Richard und Harry Buhrman: *Substituting quantum entanglement for communication.* Physical Review A, 56(2):1201, 1997.

[CBSG17] Cross, Andrew W, Lev S Bishop, John A Smolin und Jay M Gambetta: *Open quantum assembly language.* arXiv preprint arXiv:1707.03429, 2017.

[CCT+19] Cacciapuoti, Angela Sara, Marcello Caleffi, Francesco Tafuri, Francesco Saverio Cataliotti, Stefano Gherardini und Giuseppe Bianchi: *Quantum internet: networking challenges in distributed quantum computing.* IEEE Network, 34(1):137–143, 2019.

[CCVMH20] Cacciapuoti, Angela Sara, Marcello Caleffi, Rodney Van Meter und Lajos Hanzo: *When entanglement meets classical communications: Quantum teleportation for the quantum internet.* IEEE Transactions on Communications, 68(6):3808–3833, 2020.

[CEHM99] Cirac, J Ignacio, AK Ekert, Susana F Huelga und Chiara Macchiavello: *Distributed quantum computation over noisy channels.* Physical Review A, 59(6):4249, 1999.

[ČLM13] Črepinšek, Matej, Shih-Hsi Liu und Marjan Mernik: *Exploration and exploitation in evolutionary algorithms: A survey.* ACM computing surveys (CSUR), 45(3):1–33, 2013.

[CTCY12] Chen, Tianshi, Ke Tang, Guoliang Chen und Xin Yao: *A large population size can be unhelpful in evolutionary algorithms.* Theoretical Computer Science, 436:54–70, 2012.

[DFAP17] Desjardins, Benjamin, Rafael Falcon, Rami Abielmona und Emil Petriu: *Planning robust sensor relocation trajectories for a mobile robot with evolutionary multi-objective optimization.* Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges, Seiten 179–210, 2017.

[DNZ21] Daei, Omid, Keivan Navi und Mariam Zomorodi: *Improving the teleportation cost in distributed quantum circuits based on commuting of gates.* International Journal of Theoretical Physics, 60(9):3494–3513, 2021.

[DNZM20] Daei, Omid, Keivan Navi und Mariam Zomorodi-Moghadam: *Optimized quantum circuit partitioning.* International Journal of Theoretical Physics, 59(12):3804–3820, 2020.

[DZMHNb20] Davarzani, Zohreh, Mariam Zomorodi-Moghadam, Mahboobeh Houshmand und Mostafa Nouri-baygi: *A dynamic programming approach for distributing quantum circuits by bipartite graphs.* Quantum Information Processing, 19:1–18, 2020.

[EM20] El Majdoubi, Omayma: *Artificial Intelligence Approach for Multi-Objective Design Optimization of Composite Structures: Parallel Genetic Immigration.* International Journal of Advanced Trends in Computer Science and Engineering, 9:2508–2516, 06 2020.

[FH04] Fowler, Austin G und Lloyd CL Hollenberg: *Scalability of Shor's algorithm with a limited set of rotation gates.* Physical Review A, 70(3):032329, 2004.

[Gib19] Gibney, Elizabeth: *The quantum gold rush.* Nature, 574(7776):22–24, 2019.

[GKK13] Gerdes, Ingrid, Frank Klawonn und Rudolf Kruse: *Evolutionäre Algorithmen: Genetische Algorithmen—Strategien und Optimierungsverfahren—Beispielanwendungen.* Springer-Verlag, 2013.

[Gro96] Grover, Lov K: *A fast quantum mechanical algorithm for database search.* In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, Seiten 212–219, 1996.

[Gro97] Grover, Lov K: *Quantum telecomputation.* arXiv preprint quant-ph/9704012, 1997.

[HMZMH20] Houshmand, Mahboobeh, Zahra Mohammadi, Mariam Zomorodi-Moghadam und Monireh Houshmand: *An evolutionary approach to optimizing teleportation cost in distributed quantum computation.* International Journal of Theoretical Physics, 59:1315–1329, 2020.

[Hom08] Homeister, Matthias: *Quantum Computing verstehen.* Springer, 2008.

[HSSC08] Hagberg, Aric, Pieter Swart und Daniel S Chult: *Exploring network structure, dynamics, and function using NetworkX.* Technischer Bericht, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[KL70] Kernighan, Brian W und Shen Lin: *An efficient heuristic procedure for partitioning graphs.* The Bell system technical journal, 49(2):291–307, 1970.

[KUT11] Kaya, Yilmaz, Murat Uyar und Ramazan Tekin: *A Novel Crossover Operator for Genetic Algorithms: Ring Crossover.* CoRR, abs/1105.0355, 01 2011.

[McS01] McSherry, Frank: *Spectral partitioning of random graphs.* In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, Seiten 529–537. IEEE, 2001.

[MMNI08] Meter, Rodney Van, WJ Munro, Kae Nemoto und Kohei M Itoh: *Arithmetic on a distributed-memory quantum multicomputer.* ACM Journal on Emerging Technologies in Computing Systems (JETC), 3(4):1–23, 2008.

[NC01] Nielsen, Michael A und Isaac L Chuang: *Quantum computation and quantum information.* Phys. Today, 54(2):60, 2001.

[NMSZ21] Nikahd, Eesa, Naser Mohammadzadeh, Mehdi Sedighi und Morteza Saheb Zamani: *Automated window-based partitioning of quantum circuits.* Physica Scripta, 96(3):035102, 2021.

[PGN+19] Pednault, Edwin, John A Gunnels, Giacomo Nannicini, Lior Horesh und Robert Wisnieff: *Leveraging secondary storage to simulate deep 54-qubit sycamore circuits.* arXiv preprint arXiv:1910.09534, 2019.

[PVG+11] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay: *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011.

[Qis23] Qiskit contributors: *Qiskit: An Open-source Framework for Quantum Computing*, 2023.

[RBTC17] Riedel, Max F, Daniele Binosi, Rob Thew und Tommaso Calarco: *The European quantum technologies flagship programme.* Quantum Science and Technology, 2(3):030501, 2017.

[Ree10] Reeves, Colin R: *Genetic algorithms.* Handbook of metaheuristics, Seiten 109–139, 2010.

[RG02] Rylander, STANLEY GOTSHALL BART und Bart Gotshall: *Optimal population size and the genetic algorithm.* Population, 100(400):900, 2002.

[Rif16] Rifki, Omar: *A Study on Robust Structures Discovery and its Economic Applications : Combination of Algorithm and Simulation.* Doktorarbeit, 03 2016.

[SBH+16] Schulz, Christian, SK Bayer, C Hess, C Steiger, M Teichmann, J Jacob, F Bernardes-lima, R Hangu und S Hayrapetyan: *Course notes: Graph partitioning and graph clustering in theory and practice.* Karlsruhe Institute of Technology, 2016.

[SFHBA+18] Shirani Faradonbeh, Roohollah, Mahdi Hasanipanah, Hassan Bakhshandeh Amnieh, Danial Jahed Armaghani und Masoud Monjezi: *Development of GP and GEP models to estimate an environmental issue induced by blasting operation.* Environmental Monitoring and Assessment, 190, 05 2018.

[Sho99] Shor, Peter W: *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM review, 41(2):303–332, 1999.

[SKB12] Streltsov, Alexander, Hermann Kampermann und Dagmar Bruss: *Quantum cost for sending entanglement.* Physical review letters, 108(25):250501, 2012.

[SMZ21] Sarvaghad-Moghaddam, Moein und Mariam Zomorodi: *A general protocol for distributed quantum gates.* Quantum Information Processing, 20(8):265, 2021.

[VL07] Von Luxburg, Ulrike: *A tutorial on spectral clustering.* Statistics and computing, 17:395–416, 2007.

[VMLFY10] Van Meter, Rodney, Thaddeus D Ladd, Austin G Fowler und Yoshihisa Yamamoto: *Distributed quantum computation architecture using semiconductor nanophotonics.* International Journal of Quantum Information, 8(01n02):295–323, 2010.

[Wei15] Weicker, Karsten: *Evolutionäre algorithmen.* Springer-Verlag, 2015.

[WGT+08] Wille, Robert, Daniel Grosse, Lisa Teuber, Gerhard W Dueck und Rolf Drechsler: *RevLib: An online resource for reversible functions and reversible circuits.* In: *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, Seiten 220–225. IEEE, 2008.

[YLJ04a] Yimsiriwattana, Anocha und Samuel J Lomonaco Jr: *Distributed quantum computing: A distributed Shor algorithm.* In: *Quantum Information and Computation II*, Band 5436, Seiten 360–372. SPIE, 2004.

[YLJ04b] Yimsiriwattana, Anocha und Samuel J Lomonaco Jr: *Generalized GHZ states and distributed quantum computing.* arXiv preprint quant-ph/0402148, 2004.

[ZMHH18] Zomorodi-Moghadam, Mariam, Mahboobeh Houshmand und Monireh Houshmand: *Optimizing teleportation cost in distributed quantum circuits.* International Journal of Theoretical Physics, 57:848–861, 2018.