

QUANTUM COMPUTING PROGRAMMIERUNG

# **Projektarbeit am Lehrstuhl für mobile und verteilte Systeme**

**BEARBEITER:** Ege Çimşir, Jonas Gottal, Sebastian Silva,  
Tobias Rohe, Viktoria Patapovich

**BETREUER:** Jonas Stein, Sebastian Zielinski

**AUFGABENSTELLER:** Prof. Dr. Claudia Linnhoff-Popien

Dokument erstellt  
11. Juli 2021





# Projektarbeit am Lehrstuhl für mobile und verteilte Systeme

## Zusammenfassung

Das Job-Shop-Scheduling Problem ist NP-hart und beschäftigt gleichermaßen Wissenschaft und Industrie. Im Rahmen der QC-Optimization-Challenge hat die Firma TRUMPF eine Erweiterung dieser Problemstellung geboten: Hierbei geht es sowohl um die zeitliche Planung als auch Zuweisung verschiedener Bauteile auf eine Reihe von Maschinen. Zur Lösung dieses Problems stehen mehrere Quantencomputer – darunter zwei Quantum Annealer und zwei Quantum Gate Model-Computer – zur Verfügung. Die folgende Ausarbeitung zeigt, dass bisher lediglich hybride Annealing-Ansätze solche Problemstellungen bewältigen können.

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
<b>2 Grundlagen</b>	<b>3</b>
2.1 Job Shop Scheduling Probleme . . . . .	3
2.2 Quantenmechanische Postulate . . . . .	4
2.3 Quantum Annealing . . . . .	5
2.4 Quantum Gate Model . . . . .	7
<b>3 Verwandte Arbeiten</b>	<b>7</b>
<b>4 Konzept</b>	<b>8</b>
4.1 QUBO-Formulierung . . . . .	8
4.2 Pruning zur Komplexitätsreduktion . . . . .	10
4.3 Allgemeines Vorgehen . . . . .	11
<b>5 Evaluation</b>	<b>13</b>
5.1 Quantum Annealing . . . . .	13
5.1.1 Ergebnisse . . . . .	13
5.1.2 Direkter Vergleich der QPUs . . . . .	14
5.1.3 Weitere Parameter des D-Wave Advantage . . . . .	15
5.1.4 Hybrider Ansatz mit D-Wave Leap und Qbsolv . . . . .	16
5.2 Quantum Gate Model . . . . .	17
5.2.1 Qulacs-Simulator . . . . .	18
5.2.2 IBM Quantum System One . . . . .	18
5.2.3 Rigetti Aspen-9 . . . . .	19
<b>6 Fazit</b>	<b>20</b>
<b>Literatur</b>	<b>27</b>



# 1 Einleitung

In den letzten Jahrzehnten folgte der Fortschritt der Rechenleistung dem Moore-schen Gesetz. Basierend auf der Miniaturisierung von Transistoren verdoppelte sich die Hardwareleistung für von-Neumann-Architekturen alle 18 Monate. So wurde die Lösung von schwierigen Optimierungsproblemen vorangetrieben, aber physikalische Gesetze begrenzen diese Entwicklung, wenn die Chip-Größe bei atomarer Skala ankommt. Dies erfordert einen neuen Ansatz: Quantum Computing [5].

Unsere Aufgabe für die *QC-Optimization-Challenge* des *QAR-Lab* wurde von dem Unternehmen TRUMPF bereitgestellt: Es handelt sich um ein flexibles Job-Shop-Scheduling-Problem (JSSP) aus der Blechfertigung. Für aus Rohblech-chen geschnittene Kleinteile muss zur Weiterverarbeitung eine richtige und ef-fiziente Maschinenbelegung gefunden werden. Dieses Kombinationsproblem ist NP-hart<sup>1</sup> weshalb es nicht in polynomieller Zeit, deterministisch gelöst werden kann. Das bedeutet, dass solch eine Problemstellung auch mit modernen Com-putern lediglich approximativ lösbar ist.

Daher stellt sich die Frage, wie man diese Problemstellung in der Zukunft ef-fizient lösen kann. Hierzu untersuchen wir im Nachfolgenden den Einsatz von Quantencomputern.

# 2 Grundlagen

Als flexibles JSSP gehen wir im Nachfolgenden näher auf die Komplexität und Struktur des Problems ein. Im Anschluss erklären wir die quantenmechanischen Grundlagen für das Quantencomputing und gehen dann auf die einzelnen Teil-bereiche – Annealing und Gate Model – ein.

## 2.1 Job Shop Scheduling Probleme

Der Begriff Job-Shop-Scheduling-Problem beschreibt eine Problemklasse in der ein Set von Jobs einem Set von Maschinen zugeordnet werden muss und dabei die sogenannte *makespan* – die Zeitspanne in der alle Jobs abgearbeitet werden – so kurz wie möglich gehalten werden soll. Ein Job besteht dabei aus ein oder mehreren Operationen eines oder mehrerer Operationstypen, wobei diese un-terschiedlich lange Durchführungzeiten haben und einer Operationsreihenfolge unterliegen. Schließlich hat jeder Job noch eine individuelle Deadline zu der alle seine Bauteile abgearbeitet sein sollen. In unserem Fall hat jede Maschine einen Operationstyp und kann zu jedem Zeitpunkt maximal eine Operation dieses Operationstyps ausführen [24].

---

<sup>1</sup>Da das bekannteste NP-harte Problem (Travelling Salesman Problem) eine simple Instanz eines JSSPs ist, folgt, dass unsere Problemstellung ebenfalls NP-hart ist.

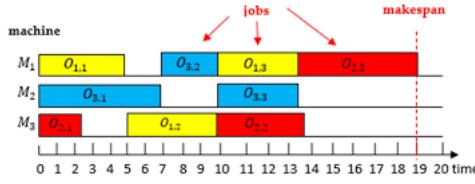


Abbildung 1: JSS-Problembispiel mit drei Jobs und drei Maschinen

Bei unserer Problemstellung handelt es sich um den Spezialfall eines *flexiblen* JSSPs, welches sich dadurch kennzeichnet, dass mindestens einem Operations- typ mehr als eine Maschine zur Verfügung steht auf der er bearbeitet werden kann. Es gibt also mindestens einen Maschinentyp mehrmals. [1]

## 2.2 Quantenmechanische Postulate

Am Ende des 19. Jahrhundert ist die klassische Mechanik bei Beschreibung mancher Phänomene an ihre Grenzen gestoßen, weshalb sich eine neue Disziplin der theoretischen Physik entwickelt hat: die Quantenmechanik. Um den Zustand eines quantenmechanischen Objekts zu beschreiben, benötigt man die aus der Schrödinger Gleichung abgeleitete Wellenfunktion [14]:

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x, t) \right] \Psi(x, t) \quad (1)$$

Diese gibt uns die Wahrscheinlichkeit, welche Zustände ein Objekt annehmen darf und zeigt uns weshalb die Quantenmechanik nicht-deterministisch ist.

Mit diesem Wissen kann das Pendant zu analogen Bits konstruiert werden: Quantenbits (nachfolgend *Qubits*) werden durch zweidimensionale Zustandsvektoren realisiert mit der Orthonormalbasis  $|0\rangle$  und  $|1\rangle$ . Ein Qubit nimmt also Zustände [15] der folgenden Form an:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad |\alpha|^2 + |\beta|^2 = 1 \quad (2)$$

Welcher Zustand mit welcher Wahrscheinlichkeit auftritt, ist von den Amplituden  $\alpha$  und  $\beta$  abhängig. Das Qubit kann sich also auch in beiden klassischen Zuständen gleichzeitig befinden, dies nennt man *Superposition*. Messen wir ein Quantenbit in diesem Zustand, wird die Superposition zerstört. Ein weiteres wichtiges quantenmechanisches Postulat ist die *Verschränkung*: Ein Phänomen von mehreren Quantenobjekten, bei dem der Zustand eines Teilchens nicht unabhängig vom Zustand der Anderen beschrieben werden kann – auch wenn diese durch eine große Entfernung getrennt sind. Albert Einstein bezeichnete dies als *spukhafte Fernwirkung*.

Quantenparallelismus – der Grund für die höhere Rechenleistung eines Quantencomputers gegenüber einem klassischen Computer – begründet sich in diesen beiden Phänomenen.

## 2.3 Quantum Annealing

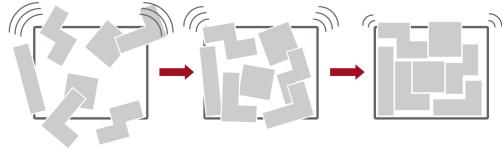


Abbildung 2: Fujitsu’s Bildhafte Erklärung

Quantum Annealing ist eine Metaheuristik zum Lösen komplexer Optimierungsprobleme [12]. Fujitsu – der Hersteller des Digitalen Annealers, den wir untersuchen – veranschaulicht Annealing im Allgemeinen anhand eines Puzzles wie in Abbildung 2 zu

sehen. Um alle Teile schnell einzupassen, ’schüttelt’ man zuerst das ganze System und reduziert dann das Schütteln kontinuierlich (daher der Name Annealing – ’Ausglühen’). Es ist also ein Heuristisches Näherungsverfahren bei dem alle Lösungen, einschließlich derjenigen, die weit vom Optimum entfernt sind, erkundet werden um sich sukzessive einer optimalen Lösung anzunähern.

Nach Lucas [19] lassen sich viele NP-harte Probleme auf eine Energieminimierungsaufgabe abbilden, die als Ising-Modell formuliert sind, und (Quantum) Annealing ist in der Lage, solche Ising-Modell-Probleme in polynomieller Zeit zu lösen [16].

Der Digitale Annealer, nachführend auch Fujitsu DAU genannt, bewältigt exakt diese Probleme [23]. In der von uns getesteten Version 2 kann er theoretisch vollständig verbundene Ising-Modelle mit 8.192 Bits lösen.

Quantum Annealing gehört zum Adiabatischen Quantencomputing, da es auf dem Adiabatischen Theorem der Quantenmechanik basiert: Ein System verbleibt in seinem Eigenzustand, wenn der zeitabhängige Hamiltonoperator sich nur sehr langsam ändert [14]. Der D-Wave Quantencomputer implementiert das Adiabatische Quantencomputing in seiner Hardware und manipuliert die verfügbaren Quantenbit-Zustände (Qubit) gemäß des zeitabhängigen Hamiltonian, nachfolgend als  $H(t)$  bezeichnet [22].

$$H(t) = s(t)H_A + (1 - s(t))H_P \quad (3)$$

Der Grundgedanke ist die Interpolation zwischen einem Anfangs-Hamiltonian  $H_A$ , dessen Grundzustand einfach zu konstruieren ist, und einem Problem-Hamiltonian  $H_P$ , dessen minimale Energiekonfiguration dem globalen Optimum des definierten Problems entspricht [11]. Unter Einhaltung des Adiabatischen Theorems fällt  $s(t)$  von 1 auf 0 [22] und liefert mit hoher Wahrscheinlichkeit den Grundzustand von  $H_P$  [2]. Dieses Konzept des adiabatischen Quantencomputers, wie ihn D-Wave herstellt, liegt nahe, dass je langsamer eine Annäherung vonstattengeht, desto wahrscheinlicher ist das Resultat ein Optimum. Jedoch sind die Qubits aufgrund der Dekohärenz [15] stark mit der Umgebung gekoppelt und werden kontinuierlich beeinflusst. Daher darf der Annealing Prozess nicht zu lange dauern [20]. Die benötigte Zeit hängt von der minimalen Energiedifferenz zwischen den beiden niedrigsten Zuständen des interpolierenden Hamiltonians ab [11]. Um Quantum Annealing auf der Hardware von D-Wave zu verwenden, muss das JSSP jedoch zunächst als ein *Quadratic Unconstrained Binary Opti-*

*mization* Problem formuliert werden. Im D-Wave Advantage sind die Qubits in einem sogenannten Pegasusgraphen (Abbildung 3) angeordnet – mathematisch ein ungerichteter Graph mit Qubits als Knoten und Kopplern als Kanten. In Bezug auf das QUBO-Problem repräsentiert jedes Qubit eine Variable und Koppler zwischen Qubits repräsentieren die Kosten, die mit Qubit-Paaren verbunden sind. Wenn die Problemstruktur nicht direkt in den Pegasusgraphen eingebettet werden kann, können Hilfs-Qubits zu einer sogenannten *Chain* kombiniert werden, um die verfügbaren Kopplungen zu erweitern.

Der D-Wave Advantage enthält mindestens 5.000 Qubits mit einer Kopplerdichte von 15 pro Qubit [21]. Im Detail besteht die Pegasus-Einheitszelle aus 48 Hälften von Qubits, die zwischen benachbarten Einheitszellen aufgeteilt sind, wie in Abbildung 3: Dabei werden Qubits als abgeschnittene Schleifen (Doppellinien), interne Koppler als Punkte und externe und ungerade Koppler als kurze Linien dargestellt [4]. Das aktuelle Hauptproblem besteht darin, dass Quantenhardware hinsichtlich der Anzahl der Qubits und ihrer Konnektivität auf dem Chip begrenzt sind. Da der D-Wave Solver keine All-to-All-Konnektivität implementiert, erschwert es für große QUBO-Probleme ein *minor-embedding* zu finden. Wenn – wie bei unserer Problemstellung von TRUMPF – große Datensätze verwendet werden, kann das resultierende QUBO-Problem die Anzahl der verfügbaren Qubits überschreiten und nicht mehr vollständig auf ein Embedding gebracht werden. Für diesen Fall haben wir noch zwei hybride Lösungsansätze verfolgt, die wir später genauer vorstellen.

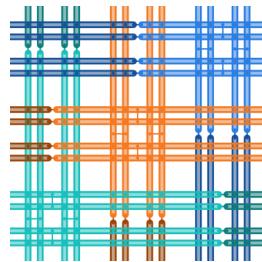


Abbildung 3: Pegasus Topologie mit internen, externen und ungeraden Kopplern

## 2.4 Quantum Gate Model

Quantum Gate Modeling (QGM) unterscheidet sich in zweierlei Hinsicht zu klassischen Computern.

Zum einen nutzt es das sogenannte „Superpositionsprinzip“, welches einen Qubit-zustand als eine Linearkombination aus zwei klassischen Zuständen darstellt. Zum anderen nutzt ein QC das quantenphysikalische Phänomen der „Quantenverschränkung“. Sind zwei quantenphysische Teilchen miteinander verschränkt, so können sie nicht unabhängig voneinander betrachtet werden, sondern werden durch einen gemeinsamen Zustand beschrieben. Diese beiden Konzepte werden in einem sogenannten „Qubit“ vereint.

Ähnlich wie bei klassischen Bits werden Qubits zu Registern, sog. Quantenregistern zusammengefasst. Ein Quantenregister mit  $n$  Qubits repräsentiert dabei die Superposition von  $2^n$  Zuständen. Genau hier kommt nun das Phänomen der Quantenverschränkung zum Tragen. Einzelne Qubits können miteinander verschränkt werden und liefern damit bei einer Messung des Zustandes kausal verbundene Ergebnisse.

Um nun mit den Qubits zu rechnen, gibt es die sogenannten Quantengatter. Diese Gatter ändern mit Hilfe einer physikalischen Aktion, bspw. eines Laserstrahls, die Superposition des Qubits, in dem sie den Spin des Qubits verändern. Daher ist die Dauer der Anwendung von großer Bedeutung. Mathematisch entspricht ein Quantengatter einer unitären Transformation auf ein oder mehrere Qubits, welche als Vektoren dargestellt werden. Damit ist ein Quantengatter eine bijektive unitäre Abbildung und kann mit Hilfe seines adjungierten Operators rückgängig gemacht werden. Dies ist eine Eigenschaft, die klassische Computer nicht erfüllen.

Durch diesen Aufbau und diese Funktionsweise kann das Ergebnis eines komplexen Problems in einem Schritt zurückgegeben werden. Ein klassischer Computer bräuchte für ein entsprechendes Ergebnis mehrere Teilschritte.

Das Ergebnis dieser Berechnung wird durch die Superpositionen der einzelnen Qubits erfasst. Leider können diese Superpositionen nicht direkt physikalisch gemessen werden, denn wird ein Qubit gemessen, so nimmt es einen einzigen Zustand an (entweder  $|0\rangle$  oder  $|1\rangle$ ). Durch Wiederholungen dieser Berechnungen und erneutes Messen, kann allerdings die zugrundeliegende Superposition beliebig genau approximiert werden. Auch wenn viele Wiederholungen dabei nötig sind, hat der QC einen erheblichen Geschwindigkeitsvorteil gegenüber klassischen Computern.

## 3 Verwandte Arbeiten

Auch andere Forschungsarbeiten haben verschiedene Typen von Quantenprozessoren miteinander verglichen. Die Hauptmethode dieser Arbeiten besteht darin, denselben Algorithmus auf verschiedenen Prozessoren laufen zu lassen und die Ergebnisse miteinander zu vergleichen.

2017 veröffentlichten Forscher an der University of Maryland den Artikel: „Experimental comparison of two quantum computing architectures“ in dem sie zwei Gattermodell-Architekturen, supraleitende Qubits und Trapped Ion Qubits, vergleichen. Obwohl die beiden Systeme unterschiedliche native Quanteninteraktionen haben, können sie auf eine Weise programmiert werden, die unabhängig für die zugrundeliegende Hardware ist, was einen Vergleich von identischen Quantenalgorithmen auf verschiedenen physikalischen Systemen ermöglicht. Die Forscher haben gezeigt, dass Quantenalgorithmen und -schaltungen, welche eine höhere Konnektivität nutzen, eindeutig von einem besser verbundenen System von Qubits profitieren. Dieses Experiment zeigt kritische Faktoren der Skalierung von Quantencomputern, wie z.B. die Konnektivität der Qubits und die Expressivität der Gatter. Darüber hinaus deuten die Ergebnisse darauf hin, dass die Abstimmung bestimmter Quantenanwendungen mit der zugrunde liegenden Hardware von entscheidender Bedeutung für den erfolgreichen Einsatz ist [18].

Des Weiteren haben Forscher des Pharmaunternehmens GlaxoSmithKline (GSK) mit Hilfe von Quantencomputern Experimente durchgeführt und die unterschiedlichen Herangehensweisen von IBM und D-Wave verglichen. Der Quantum Annealer von D-Wave, kann dabei bereits mit klassischen Computern konkurrieren und einige Probleme lösen. Auf der anderen Seite haben Gate-basierte Quantencomputer, wie der von IBM, noch nicht genügend Qubits, um Probleme von relevanter Komplexität zu lösen. Die GSK-Forscher gehen davon aus, dass die zu erwartende Erhöhung der Qubit-Anzahl in solchen Computern dazu führen wird, dass Quantengeräte einen signifikanten Leistungsvorteil gegenüber klassischer Hardware aufweisen werden [13].

## 4 Konzept

Im folgenden Kapitel werden wir unser grundlegendes Konzept und Vorgehen aufzeigen. Zunächst werden wir auf unsere Constraints und die Kostenfunktion für die QUBO eingehen und die Komplexität des JSSP diskutieren. Anschließend beschreiben wir die angewandten Pruning Methoden, das allgemeine Vorgehen und die eingesetzten Trainingsmethoden.

### 4.1 QUBO-Formulierung

*Quadratic Unconstrained Binary Optimization*, kurz QUBO, ist eine Form zur Beschreibung kombinatorischer Optimisierungsprobleme.

Sie besteht meist aus einer zu minimierenden Kostenfunktion  $K$  sowie Straftermen  $H_0$  bis  $H_n$ , welche dazu dienen, bestimmte Tupelbelegungen zu verbieten. Da ein Quantencomputer nach der Lösung mit dem *niedrigsten* energetischen Level sucht, füllen Strafterme die Matrixfelder, welche diese Belegungen beschreiben, mit *positiven* Energiewerten.

Wir definieren die QUBO-Variablen binär:

$$x_{j,p,o,m,t} = \begin{cases} 1 & : \text{Operation } o \text{ aus der Order } (j,p) \text{ wird auf} \\ & \text{der Maschine } m \text{ zum Zeitpunkt } t \text{ gestartet} \\ 0 & : \text{sonst} \end{cases} \quad (4)$$

Im folgenden sind  $O$  – die Menge aller Operationen,  $T$  – die Gesamtzeit, die nötig ist, um alle Operationen *nacheinander* auszuführen,  $M$  - die Menge aller verfügbaren Maschinen.

Im Folgenden eine Übersicht über unsere Strafterme und Kostenfunktion.

$$H_0 = \sum_{o \in O} \left( 1 - \sum_{m \in M} \sum_{t=0}^{T-1} x_{j,p,o,m,t} \right)^2 \quad (5)$$

Dieser Strafterm stellt sicher, dass für jeden Job jede Operation *genau* einmal gestartet wird. Er bestraft also mehrmalige und fehlende Belegung einer Operation [10].

$$H_1 = \sum_{j \in J} \left( \sum_{k, k' \in V_j} x_k x_{k'} \right) \quad (6)$$

$$V_J = \left\{ \begin{array}{l} ((j, p, o, m, t), (j', p', o', m', t')) \in (J \times P \times O_j \times M \times T)^2 \mid \\ j = j' \wedge p = p' \wedge (c(o) < c(o')) \wedge (t' < t + d_{o,m}) \end{array} \right\} \quad (7)$$

$$\text{Und } c|_{O_j} : O_j \rightarrow \{1, \dots, n_j\} = \text{Reihenfolge der Operationen innerhalb eines Jobs} \quad (8)$$

Der zweite Strafterm stellt für alle Bauteile aller Jobs sicher, dass die Reihenfolge der Operationstypen untereinander eingehalten wird [10].

$$H_2 = \sum_{m \in M} \left( \sum_{k, k' \in W_m} x_k x_{k'} \right) \quad (9)$$

$$W_M = \left\{ \begin{array}{l} ((j, p, o, m, t), (j', p', o', m', t')) \in (J \times P \times O \times M \times T)^2 \mid \\ m = m' \wedge (t \leq t' < t + d_{o,m}) \end{array} \right\} \quad (10)$$

Dieser letzte Strafterm bewirkt, dass zu jedem Zeitpunkt auf einer Maschine jeweils maximal ein Bauteil bearbeitet wird [10].

$$K = \sum_{j \in J} (1 + sgn(t_j^* - t_j, \text{deadline})) (t_j^* - t_j, \text{deadline}) \quad (11)$$

$$\text{mit } t_j^* = \sum_{t=0}^T (x_{o^*, m, t} * d_{o^*, m}) \quad (12)$$

Zuletzt folgt hier noch die Kostenfunktion, welche die Verspätung der Bauteile des Jobsets in Sekunden zählt und diese dabei für jede weitere Zeiteinheit (30s)

die ein spezifisches Bauteil zu spät ist, härter bestraft [17]. Hier repräsentiert  $O^*$  die letzte Operation  $o$  eines Jobs  $j$

Diese vier Teilgleichungen können nun in der Form:

$$H = \delta K + \alpha H_0 + \beta H_1 + \gamma H_2 \quad (13)$$

zusammengefasst werden und bilden die zu optimierende Funktion  $H$  der QUBO. Die Konstanten  $\alpha$  bis  $\gamma$  sind die *Gewichtungen*, welche bestimmen wie sehr ein jeweiliger Strafterm die spätere Wahl der Variablen beeinflussen soll.

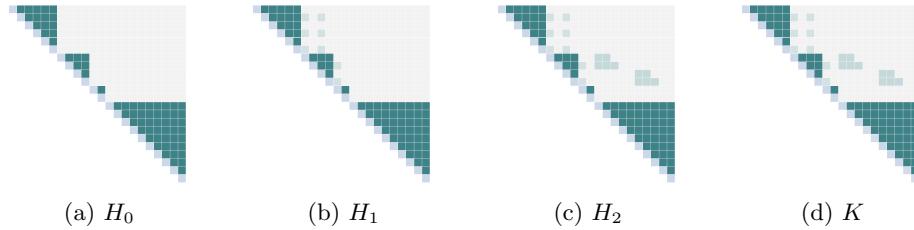


Abbildung 4: Visualisierung des Progressiven Aufbaus der Constraints und Kostenfunktion – Letztere hat einen merkbar geringeren Effekt vergleichbar zu den Constraints

## 4.2 Pruning zur Komplexitätsreduktion

Eine der wichtigsten Aufgaben während der QUBO-Formulierung war die Eliminierung von möglichst vielen Variablen während des Pruning-Prozesses. Um die Gesamtzahl der Variablen zu reduzieren, haben wir als erstes alle *unmöglichen* Variablen entfernt, deren Index gar nicht existieren kann [10]. Zum Beispiel kann eine Biegeoperation nicht auf einer Schweißanlage durchgeführt werden.

Zusätzlich haben wir das sogenannte *Heads and Tails* [24] Pruning durchgeführt: für jede Operation muss Zeit für Vorgänger- und Nachfolgeroperationen vorhanden sein. In diesem Schritt haben wir alle Operationen mit unmöglichen Startzeiten entfernt, z. B. Lackieren bei  $t = 0$ , wenn wir zuvor noch biegen und schweißen müssen. So haben wir alle Variablen  $x_{j,p,o,m,t}$  mit  $0 \leq t < S$ , entfernt, wobei  $S$  die Summe der Ausführungszeiten aller Operationen vor der betrachteten ist. Dann haben wir auch alle Variablen  $x_{j,p,o,m,t}$  mit  $T - S \leq t < T$  entfernt, wobei  $S$  die Summe der Ausführungszeiten aller Operationen nach der betrachteten ist [17].

Den signifikantesten Unterschied machte das *Time-Modulo* Pruning und hat die Gesamtzahl der Variablen deutlich reduziert. Der Grundgedanke ist es die Gesamtzeiten für jede Operation auf den größten gemeinsamen Teiler zu bringen ohne die Lösungsqualität zu verändern. Um die Größe unserer QUBO weiter zu reduzieren, haben wir jedoch Zeitstritte von  $t_{step} = 30$  Sekunden angenommen. Das ändert nichts für Schweiß- und Lackier-Operationen ( $\equiv \mod 30\text{sec}$ ), aber addiert in manchen Fällen einen vernachlässigbar kleinen zusätzlichen Zeitpuffer  $\Delta t = 10\text{sec}$  am Ende der Biegeoperationen. Dies reduziert die Lösungsqualität geringfügig.

### 4.3 Allgemeines Vorgehen

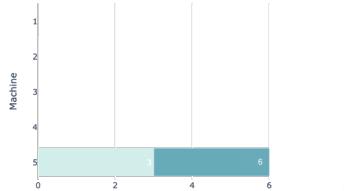
Um eine Vergleichbarkeit unserer Ergebnisse – die auf verschiedenen Ansätzen und Hardware basieren – zu erreichen, haben wir einheitliche Problemsets definiert. Diese Problemsets bestehen aus zufällig generierten Job Sets, welche eine gestaffelte Komplexität vorweisen. In Abbildung 5 sind die Gantt-Diagramme mit einer der möglichen optimalen Lösungen (im Nachfolgenden *Best Known Solution – BKS*) der jeweiligen JSSPs grafisch dargestellt. Die Komplexität dieser Problemsets leitet sich dabei aus der gerundeten Anzahl der Kombinationsmöglichkeiten der verschiedenen Jobs ab. Die unterschiedlichen Balkenfarben repräsentieren verschiedene Jobs, wobei die Deadline dieser Jobs als kleine Zahl am Balkenende angegeben wurde. (Maschinen, welche nicht genutzt wurden, sind teilweise aus Gründen der Übersichtlichkeit ausgeblendet worden.) Während das Gate Modeling aufgrund seiner geringen Anzahl an zur Verfügung stehenden Qubits nur Problemsets der Klasse 10 löst, werden beim Annealing die Job Sets 10 bis 500 direkt mit einer QPU und 500 bis 5000 mit Hybriden Ansätzen gelöst. Die QUBO Parameter  $\alpha$ ,  $\beta$ ,  $\gamma$  und  $\delta$  wurden wie folgt gewählt:

Tabelle 1: Parameter der QUBO

$\alpha$	$\beta$	$\gamma$	$\delta$
2.0	1.5	2.0	0.035

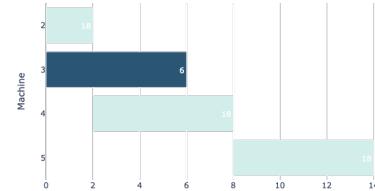
Die Constraints sind im Allgemeinen wichtiger als die Kostenfunktion, da diese nicht gebrochen werden dürfen. Sollten sie dennoch gebrochen werden, so wird eine unmögliche Lösung berechnet. Aber auch innerhalb der Constraints gibt es Unterschiede: So führt ein Bruch von  $H_0$  ( $\alpha$ ) und  $H_2$  ( $\gamma$ ) zu weiteren Seiteneffekten, aber ein Bruch von  $H_1$  ( $\beta$ ) lediglich zum Verlust eines einzelnen Bauteils. Daher sind  $H_0$  und  $H_2$  als *harte* Constraints zu deklarieren, während  $H_1$  als ein *weiches* Constraint gilt. Die Kostenfunktion muss anschließend mit Parameter  $\delta$  so gewählt werden, dass sie die zuvor definierten Constraints nicht überlagert, jedoch groß genug ist, die optimale Lösung zu finden.

Job Shop Schedule [Timeline in 10sec Steps]



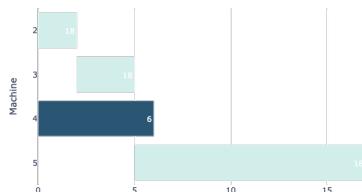
(a) Job Set 5

Job Shop Schedule [Timeline in 10sec Steps]



(b) Job Set 10

Job Shop Schedule [Timeline in 10sec Steps]



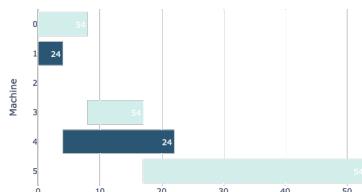
(c) Job Set 20

Job Shop Schedule [Timeline in 10sec Steps]



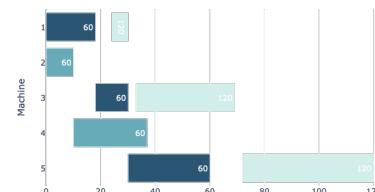
(d) Job Set 50

Job Shop Schedule [Timeline in 10sec Steps]



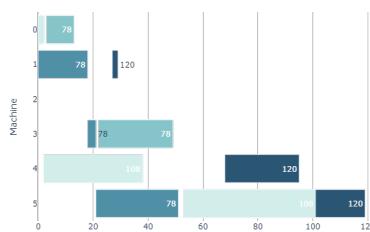
(e) Job Set 100

Job Shop Schedule [Timeline in 10sec Steps]



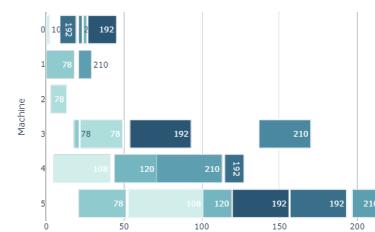
(f) Job Set 500

Job Shop Schedule [Timeline in 10sec Steps]



(g) Job Set 1000

Job Shop Schedule [Timeline in 10sec Steps]



(h) Job Set 5000

Abbildung 5: Alle Job Sets gestaffelt nach Komplexität

## 5 Evaluation

In diesem Kapitel werden die Ergebnisse der Ansätze des Quantum Annealings und des Quantum Gate Models vorgestellt. Für das Quantum Annealing analysieren wir zunächst die beiden QPUs, deren Parameter und die hybriden Ansätze um größere Probleme zu lösen. Im Anschluss werden im Gate Model die Ergebnisse von Quantenschaltkreisen vorgestellt.

In beiden Teilen werden *Pyschedule* als generischer Solver und *Qbsolv* als klassischer QUBO Solver für Benchmarks genutzt. Für das Annealing werden die beiden Annealer von D-Wave und Fujitsu, der Simulator *Neal* und die hybriden Divide-and-Conquer Ansätze *Leap* und *Qbsolv* verglichen. Für das Gate Model dienen *Qulacs* und *Qiskit* als Simulatoren und die beiden Quantencomputer von IBM und Rigetti liefern die realen Ergebnisse.

### 5.1 Quantum Annealing

Im Nachfolgenden werden folgende Solver für die Job-Sets 10 bis 5.000 aus Abbildung 5 miteinander verglichen: *Pyschedule*, *Qbsolv*, *D-Wave Neal*, *Fujitsu DAU*, *D-Wave Advantage*, *Qbsolv + D-Wave*, *D-Wave Leap*. Als Vergleichswert dient das globale Energieminimum einer *Best Known Solution (BKS)*, also einem Ergebnis, bei dem weder Constraints noch Kostenfunktion verletzt wurden. Des Weiteren definieren wir eine *Good Known Solution (GKS)* als ein Ergebnis bei dem maximal ein Zeitschritt über die Deadline verstrichen ist. Da dies für Alltagssituationen der Lohnfertiger von TRUMPF noch konservativ geschätzt ist, haben wir diesen Maßstab hinzugenommen. Ein Zeitschritt Verspätung entspricht einer Abweichung von 0.5 Energieeinheiten vom Optimum – also bei D-Wave:  $-8.0 \hat{=} \text{BKS}$  und  $< -7.5 \hat{=} \text{GKS}$ .

#### 5.1.1 Ergebnisse

Unten stehend sind die finalen Ergebnisse der einzelnen Solver nach *BKS* aufgeschlüsselt. Für *Pyschedule* und D-Wave Leap sind keine granularen *BKS*-Werte möglich – lediglich eine Verifikation.

Tabelle 2: Ergebnisse der unterschiedlichen Solver nach Komplexitätsklassen

Type	Solver	Komplexitätsklassen der Job-Sets							
		Runs	10	20	50	100	500	1.000	5.000
CPU	Pyschedule	10	✓	✓	✓	✓	✓	✓	✗
CPU	Qbsolv	10	100 %	100 %	100 %	62.5 % <sup>2</sup>	100 %	8 %	5 %
CPU	D-Wave Neal	10	84.2 %	79.2 %	56.2 %	1.6 %	0.6 %	0.2 %	✗
QPU	D-Wave Advantage	20	26.6 %	1.8 %	0.3 %	0.2 %	✗	✗	✗
QPU	Fujitsu DAU	15	99.5 %	31.1 %	4.4 %	1.3 %	0.9 %	✗	✗
Hybrid	Qbsolv + D-Wave	1	100 %	100 %	100 %	73.2 %	52 %	45 %	100 % <sup>3</sup>
Hybrid	D-Wave Leap	1	✓	✓	✓	✓	✓	✓	✓

<sup>1</sup>In Kapitel 5.1.4 werden wir auf diese Anomalie näher eingehen.

<sup>3</sup>Aufgrund von mangelnder Rechenzeit nur mit *num\_reads = 1* berechnet.

Von besonderem Interesse ist der Vergleich der beiden QPUs in Bezug auf Performance und Fehleranfälligkeit.

### 5.1.2 Direkter Vergleich der QPUs

Im Nachfolgenden haben wir alle QPU Runs nach Komplexität aufgeschlüsselt, um zwei Eigenschaften zu verdeutlichen: Die Solver sind nur für kleine Probleme zuverlässig und die Quantum-Inspired Technologie ist im Streuverhalten dem tatsächlichen Adiabatischen Quantencomputer überlegen. Dies lässt sich auf Quanteneffekte zurückführen, welche jetzt noch zu starken Schwankungen bei den Ergebnissen führen. Außerdem konnte lediglich bis einschließlich Job100 ein Embedding gefunden werden.

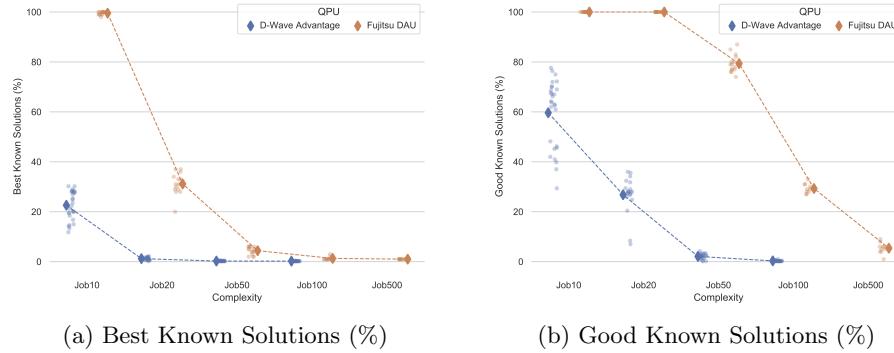


Abbildung 6: Alle Ergebnisse nach Komplexität

Um dieses Streuverhalten noch genauer zu verdeutlichen, beschränken wir uns nicht nur auf *BKS* und *GKS*, sondern visualisieren das Gesamtergebnis der Energieminimierung im Ising-Model. Hier erkennt man noch deutlicher, dass bei größeren Komplexitätsklassen der D-Wave Advantage scheinbar zufällige Ergebnisse liefert, während der Fujitsu DAU noch *gute* Lösungen findet.

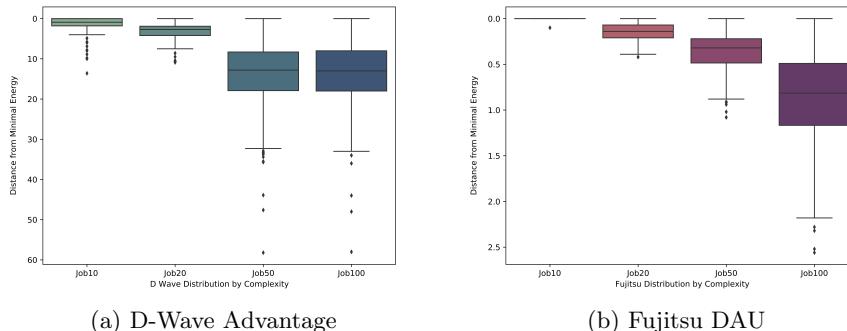


Abbildung 7: Alle Abweichungen vom Energieminimum ( $\equiv 0$ ) nach QPU Solver

In Abbildung 7 erkennt man die starke Streuung genauer. Die Distanz zum Energieminimum (dem globalen Optimum) ist bei allen Ergebnissen des Fujitsu DAU deutlich geringer. Hier bedeutet 0 das globale Optimum (*BKS*) und 0.5 entspricht der *GKS*. Die Ergebnisse des D-Wave streuen hingegen bis zu über 50 Schritte weiter, als die des DAU.

### 5.1.3 Weitere Parameter des D-Wave Advantage

Für die D-Wave Advantage QPU haben wir unseren Lösungsansatz weiter präzisiert um das Maximum an *BKS* zu ermöglichen. D-Wave QPUs können Ising-Modelle lösen, aber die erforderliche Konnektivität und Präzision in realen Problemen kann nur erreicht werden, indem die Bits des logischen Modells auf physikalische Qubits abgebildet werden. [9] Dieser sogenannte Embedding-Prozess reduziert die Anzahl der nutzbaren Modellbits um den Faktor 30-50 im Vergleich zu den verfügbaren Qubits. [7]

Embeddings lassen sich lokal mithilfe von *minorminer* [8] oder remote von D-Wave berechnen und speichern. Dazu haben wir diese verglichen und bewertet nach Gesamt-Performance, Platzierung auf der Pegasus Topologie, Chain-Length, sowie Chain-Breaks. Chains entstehen bei der physischen Implementierung eines logischen Qubits mit mehr als einem physischen Qubit. Da diese Konstellation mit zunehmender Distanz (Chain-Length) instabil wird und zu Chain-Breaks führen kann, ist die Chain-Strength als Parameter ausschlaggebend. Als einer der wichtigsten Annealing-Parameter, stellt sie sicher, dass die physischen Qubits auch den selben Wert abbilden, den die logischen Qubits vorgeben. Und um diesen Parameter zu bestimmen, haben wir drei Ansätze zur Berechnung verglichen: Den Maximalwert der QUBO (Empfehlung als Ausgangspunkt von D-Wave), *scale* – die Skalierung zum größten Problem-Bias und *torque-compensation* – eine Kompensation für Drehmomente um Chain-Breaks entgegenzuwirken. In Abbildung 8 kann man anhand der Ergebnisse sehen, dass letzteres am ehesten stetige *BKS* liefert.

Da sich Quantum Annealing wie in Abschnitt 2.3 vorgestellt, als zeitliche Balance zwischen Adiabatischem Theorem und der Dekoheranz interpretieren lässt, stellt dies eine weitere Stellschraube in unserer Untersuchung dar. Die Annealingzeiten haben wir als Fenster um die erwartete optimale Zeit mit wachsender Problemgröße verschoben und analysiert: Die Default Zeit von  $20\mu\text{s}$  war meist unter den besten Ergebnissen und andere Einstellungen lieferten kaum Mehrerfolg. Dass die Annealingzeit mit der Problemgröße steigen muss, hatte nämlich kaum Einfluss: Mit wachsender Komplexität wurden die Ergebnisse insgesamt zunehmend zufällig und damit nicht mehr objektiv vergleichbar unter dem Kriterium Annealingzeit. Auch die diversen Anneal Schedules (Quench and Pause oder Reverse

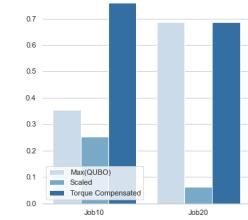
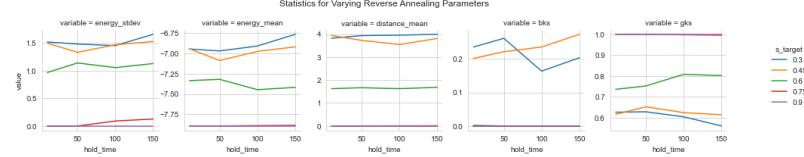
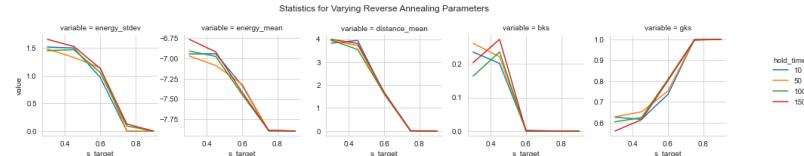


Abbildung 8: Die Chain Settings für normalisierte BKS von aussagekräftigen Job-Sets



(a) Reverse Annealing in Abhängigkeit der Slopes.



(b) Reverse Annealing in Abhängigkeit der Hold Time.

Abbildung 9: Reverse Annealing Parameter mit Random Walk

Annealing), um die Energiewellenform zu ändern, konnten die Lösungsqualität nicht positiv beeinflussen. Nur für die am besten gewählten Parameter (Abbildung 9) konnten vergleichbare Ergebnisse von Forward Annealing mit Default Annealing Time erreicht werden. Zusammenfassend lässt sich sagen, dass sich mit zum Problem passenden Chain Strengths und guten Embeddings die besten Resultate erzielen lassen. Lokal mit *minorminer* oder remote über D-Wave konnten wir jedoch ab Job500 keine Embeddings mehr berechnen, was für uns die Grenze des D-Wave Advantage als alleinstehenden Solver markiert.

#### 5.1.4 Hybrider Ansatz mit D-Wave Leap und Qbsolv

Da wie erwähnt die QPU Solver der aktuellen Generation noch nicht leistungsstark genug sind, um unsere Problemstellungen anzugehen, haben wir uns zusätzlich mit den beiden hybriden Ansätzen auseinandergesetzt: Die Komplettlösung D-Wave Leap [6] – eine Kombination aus Decomposer und QPU als Solver – die aber nur eine finale Lösung zurück gibt und wenig Einblick in die Funktionsweise bietet. Sowie die Kombination aus Qbsolv [3] als Decomposer, mit dem D-Wave Advantage als QPU. Dieser Ansatz lässt nach jeder Iteration eine Auswertung der Ergebnisse zu. Qbsolv zerlegt dabei die QUBO in kleinere Komponenten (Sub-QUBOs) einer definierten Größe, die dann unabhängig voneinander gelöst werden. Dieser Prozess wird, solange es eine Verbesserung gibt, iterativ ausgeführt. Dabei wird die QUBO-Matrix in jeder Iteration in verschiedene Komponenten aufgeteilt und von der D-Wave QPU gelöst. Neben dem Embedding und Aufspalten der QUBOs in Sub-QUBOs kümmert sich Qbsolv auch um das Zusammenführen der Lösungen der Teilprobleme. Wir verwenden die Standardkonfiguration von D-Wave's Qbsolv mit  $20\mu\text{s}$ .

Da Qbsolv für Job100 merkbar schlechtere Ergebnisse liefert hat im Vergleich zu Job50 und Job500 und wir jetzt die Funktionsweise von Qbsolv als Decom-

poser kennen, kommen wir zu folgender Betrachtung. Da die Sub-QUBO Größe manuell zu bestimmen ist, ist nachfolgender Vergleich interessant: Für zwei Job-Sets – die sehr ähnlich sind (beide Komplexität 100) und bei direktem QPU Zugriff sehr ähnliche Resultate liefern – können die Ergebnisse drastisch variieren. Job100 A ist unser zufällig generiertes Problem und Job100 B ein leicht vereinfachtes und manuell konstruiertes Problem.

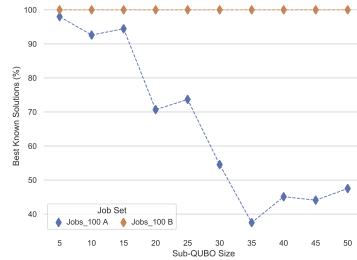


Abbildung 10: BKS (%) in Abhängigkeit der Sub-QUBO Größe.

Während Job100 A mit zunehmender Sub-QUBO Größe deutlich an BKS verliert, kann man bei Job100 B keine Verluste erkennen. Qbsolv als Decomposer schafft es die kleineren Teilprobleme in Job100 B wesentlich effizienter zu strukturieren. Das liegt an einer sehr langen Lackier-Operation in Job100 A, die es erschwert ein Sub-Problem ohne Seiteneffekte zu erstellen.

Da jedoch alle Teilprobleme einzeln der Reihe nach remote über D-Wave berechnet werden, erhöht sich die Gesamtzeit signifikant: Während man  $24.7\text{sec}$  reine Rechenzeit für die  $5000 \times 5000$  Matrix auf der QPU benötigt, dauerte der hybride Run insgesamt  $35\text{min}$ . Ein lokaler Vergleich zum Cloud-Service wäre hier interessant. Abschließend zeigt sich aber, dass schon komplexe und größere Probleme mit einem Hybriden Ansatz zu lösen sind. Der Mehrwert der QPU in dieser Konstellation steht jedoch noch aus.

## 5.2 Quantum Gate Model

Im folgenden Teil besprechen und vergleichen wir die Ergebnisse des Quantum Gate Models. Zuerst betrachten wir die Resultate des Qulacs-Simulators, anschließend die Ergebnisse der Quantencomputer *IBM Quantum System One in Ehninghen* und des *Rigetti Aspen-9 in Kalifornien*.

Das von uns modellierte Jobset benötigt lediglich vier Qubits. Obwohl uns der IBM Quantum System One 27 Qubits und der Rigetti Aspen-9 32 Qubits zur Verfügung stellt, stammt die gewählte Komplexität unseres Jobsets aus der Erfahrung, dass mit zunehmender Anzahl der genutzten Qubits die Wahrscheinlichkeit für falsche Ergebnisse durch Quantenrauschen und Dekohärenz ebenfalls enorm wächst.

Im folgenden werden die Ergebnisse von Rigetti und IBM durch jeweils zwei Histogramme repräsentiert. Dies liegt an den unterschiedlichen verwendeten Optimierungsansätzen. Auf der rechten Abbildung wurde der Schaltkreis mit Parametern betrieben, welche zuerst auf einem klassischen Computer voroptimiert und dann zusätzlich auf dem Quantum Computer während der Laufzeit nachoptimiert wurden. Das Diagramm auf der linken Seite hingegen gibt die Ergebnisse einer Einspeisung des Schaltkreises mit lediglich klassisch voroptimierten Parametern wieder.

### 5.2.1 Qulacs-Simulator

Aufgrund der geringen Komplexität der von uns genutzten Jobssets, erlauben es uns klassische Computer noch das Verhalten von Quantencomputern zu simulieren und dabei zuverlässige Ergebnisse zu produzieren. Wir stützen uns daher bei den folgenden Vergleichen auf die Ergebnisse des Qulacs-Simulators und benutzen diese als Benchmark. Wie in Abbildung 11 zu sehen ist, stehen zwei Belegungen deutlich hervor. Die größere von beiden, Belegung Nr.9 (Bitstring 1001), repräsentiert hierbei das richtige Ergebnis. Der zweite Ausschlag hingegen gehört zu einer unzulässigen Belegung und ist auf die probabilistische Natur des QCPs zurückzuführen.

### 5.2.2 IBM Quantum System One

Die folgenden Ergebnisse stammen von dem neuen Quantencomputer IBM Quantum System One in Ehningen auf den wir im Rahmen dieses Seminars bereits zugreifen konnten.

Aus diesem Ergebnis geht hervor, dass der IBMQ ein falsches Ergebnis, Belegung Nr.6 (Bitstring 0110), gewählt hat. Anzumerken ist jedoch, dass das Ergebnis bei dem die Parameter lediglich einfach optimiert wurden (Abbildung 12a), wesentlich näher an der korrekten Lösung lagen. Bei einer doppelten Optimierung der Parameter, schneidet die korrekte Belegung deutlich schlechter ab (Abbildung 12b). Dies könnte darauf hindeuten, dass eine doppelte Optimierung, zumindest auf dem IBM Quantum System One und unter Benutzung des Adam-Optimizers, negative Auswirkungen haben kann.

Weiterhin lässt sich beobachten, dass die Varianz der Belegungen bei der einfachen Optimierung (Abbildung 12a) deutlich höher ausfällt als bei der doppelten Optimierung (Abbildung 12b). Im Allgemeinen kann gesagt werden, dass eine höhere Varianz bei richtigem Ergebnis erstrebenswert ist. Ein einziger großer Ausschlag erhöht hierbei das Vertrauen in die Richtigkeit der berechneten Belegung.

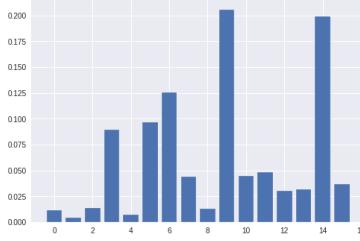
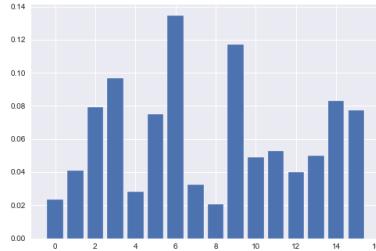
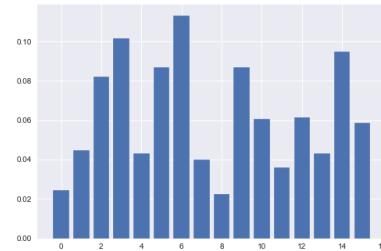


Abbildung 11: Qulacs



(a) Einfach optimiert



(b) Zweifach optimiert

Abbildung 12: Ergebnisse des IBM Ehningen für ein vier Qubit Jobset

### 5.2.3 Rigetti Aspen-9

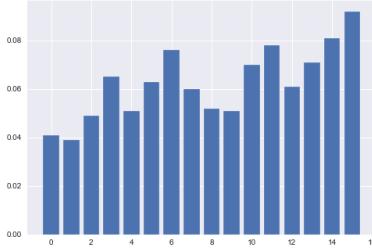
Die Ergebnisse des Rigetti Aspen-9 in Kalifornien, angesteuert über AWS Bracket, vervollständigen unseren Vergleich von Quanten-Hardware.

Auch hier hat die richtige Belegung nicht den stärksten Ausschlag. Sowohl bei dem einfach optimierten, als auch dem zweifach optimierten Versuch ist die häufigste Belegung die 15 (Bitstring 1111). Diese Belegung spiegelt die Wahl aller Tupel wieder, was unzulässig ist.

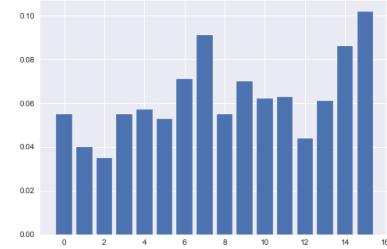
Im Gegensatz zu den Ergebnissen des IBM Quantum System One, führt hier die doppelte Parameteroptimierung zu einem besseren Ergebnis. Der Ausschlag der richtigen Belegung ist stärker und die Varianz der Häufigkeiten größer.

Das bessere Abschneiden der doppelten Optimierung entspricht viel eher dem erwarteten Unterschied zwischen den beiden Optimierungsansätzen.

Dies hebt die Unterschiede der beiden Quantencomputer nochmals hervor und lässt vermuten, dass die Auffälligkeiten bei unseren IBM Ergebnissen auf die unterschiedlichen Hardwarearchitekturen der beiden Rechner zurückzuführen sind. Zu guter Letzt stellen wir fest, dass der IBM Quantum System One im Allgemeinen besser auf unseren Schaltkreis anspricht, was die Tatsache, dass er nach der doppelten Optimierung *schlechtere* Ergebnisse zurückgibt, noch bemerkenswerter macht.



(a) Einfach optimiert



(b) Zweifach optimiert

Abbildung 13: Ergebnisse des Rigetti Aspen-9 für ein vier Qubit Jobset

## 6 Fazit

In unserem Projekt haben wir die Leistung von mehreren Quantenrechnern anhand eines flexiblen JSSPs für verschiedene Komplexitätssklassen verglichen.

Unsere Experimente zeigen, dass Quantencomputer in der heutigen NISQ-Ära noch immer keine adäquate Antwort auf das Quantenrauschen haben. Beide Quantum Gate Computer – IBM Quantum System One und Rigetti Aspen-9 – konnten keine zufriedenstellenden Lösungen für die gegebenen Jobsets finden. Für kleine Probleminstanzen konnten die Quantum Annealer hingegen schnell qualitativ hochwertige Lösungen finden: beide getesteten QPU Solver waren in der Lage ein JSSP mit bis zu 100 Variablen zu lösen. Dabei war die Quantum-Inspired Technologie von Fujitsu in der Performance und im Streuverhalten dem tatsächlichen Adiabatischen Quantencomputer überlegen. Die besten Ergebnisse lieferten aber hybride Ansätze, mit denen man auch schon jetzt industriennahe Probleme lösen kann.

Die QPUs entwickeln sich rasant, so soll die nächste Generation der Digitalen Annealer bis zu einer Millionen binärer Variablen bieten. Bei D-Wave verdoppeln sich alle drei Jahre die Anzahl der verfügbaren Qubits und bei IBM werden in den nächsten Jahren voraussichtlich mehr als ein Tausend Qubits zur Verfügung stehen.

All das deutet auf eine vielversprechende Technologie im Kampf gegen exponentiell wachsende Optimierungsprobleme hin.

## Appendix

### Job Sets nach Komplexität

Tabelle 3: Job5 mit 4 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	0	0.5	0.5
1	0	2	0	0.5	1

Tabelle 4: Job10 mit 12 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	2	1	3
1	0	2	0	0	1

Tabelle 5: Job20 mit 22 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	1	2	3
1	0	0	2	0	1

Tabelle 6: Job50 mit 49 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	1	2	3
1	0	2	3	0	3

Tabelle 7: Job100 mit 105 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	4	3	6	9
1	0	2	6	0	4

Tabelle 8: Job500 mit 507 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	3	12	8	20
1	0	5	9	0	10
1	1	9	4	5	10

Tabelle 9: Job1000 mit 1006 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	12	8	18
1	0	5	9	0	13
1	1	9	1	5	13
2	0	1	9	3	20

Tabelle 10: Job5000 mit 5004 Variablen

OrderNo	PartNo	BendingLines	Weldingpoints	PaintTime	DueDate
0	0	1	12	8	18
1	0	5	9	0	13
1	1	9	1	5	13
2	0	1	9	3	20
3	0	4	14	3	35
3	1	1	11	0	35
4	0	5	4	6	32
4	1	9	13	6	32

## Verwendete Einstellungen der Solver

Tabelle 11: Unsere Default Annealing Einstellung bei Fujitsu

Parameter	Wert	Bedeutung
solver	DAU	Digital Annealer Unit
solution mode	COMPLETE	Gibt von jedem Durchlauf die beste Lösung an
number runs	100	Gesamtanzahl der Durchläufe
number iterations	500	Gesamtanzahl der Iterationen pro Durchlauf
temperature start	1000.0	Starttemperatur
temperature end	1.0	Endtemperatur
temperature interval	100	Anzahl Iterationen (Temperatur konst.)
temperature mode	0	Temperatur wird pro temperaturenterval um den Faktor $1-temperature-decay$ verringert
offset increase rate	0.0	Dynamische Energiefunktion
optimization method	annealing	Unterstützte Optimierungsmethode
annealer version	2	Digital Annealer Version
auto tuning	0	Automatisches Tuning des QUBOs
bit precision	16	Bit-Präzision (In DAU Version 2)

Tabelle 12: Unsere Default Annealing Einstellung bei D-Wave Advantage

Parameter	Wert	Bedeutung
solver	DWave Advantage	Solver mit 5000 Qubits
sampler	Embedding Composite	D-Waves Sampler für automatisches Minor-embedding
sampler	FixedEmbedding Composite	D-Waves Sampler für fixes Minor-embedding
num reads	500	Gesamtanzahl der Durchläufe
annealing time	$20\mu s$	Zeit in Mikrosekunden für ein Annealing (Durchlauf)

Tabelle 13: Unsere Default Einstellung bei Hybrid: Qbsolv + D-Wave

Parameter	Wert	Bedeutung
solver	D-Wave Advantage	Solver mit 5000 Qubits
solver	D-Wave Neal	Quantum Annealing Simulator
num reads	20	Gesamtanzahl der Durchläufe
solver limit	20	Maximale Größe der subQUBO Matrix

## Berechnungsgrundlagen

Formeln zur Berechnungsgrundlage für die Chain Strength mit:

- QUBO  $Q = (c_{ij})$ ,
- $n$ : Anzahl von  $c_{ij} \neq 0$  in  $Q$ ,
- mit  $i > j$ ,
- und  $\langle d \rangle$ : Durchschnitt der Verbindungen von Variablen in der QUBO

### MaxQUBO:

$$\max(\text{abs}(c_{ij})) + 1 \quad (14)$$

### Scaled<sup>4</sup>:

$$\max \left( \max \left( \max_{i>i} c_{ij}, -\min_{i>i} c_{ij} \right), \max \left( \max_{i=i} c_{ij}, -\min_{i=j} c_{ij} \right) \right) \quad (15)$$

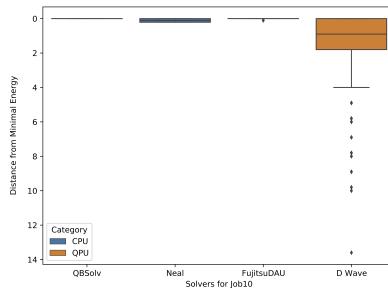
### Uniform Torque Compensation:

$$\sqrt{\langle d \rangle * \frac{2 \sum_{i>j} c_{ij}^2}{n}} \quad (16)$$

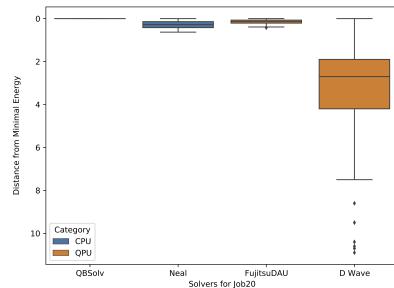
---

<sup>4</sup>Scaled gibt 1 zurück, falls kein *Binary Quadratic Model* erkannt wird.

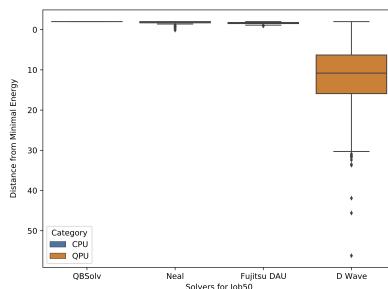
## Weitere Auswertungen



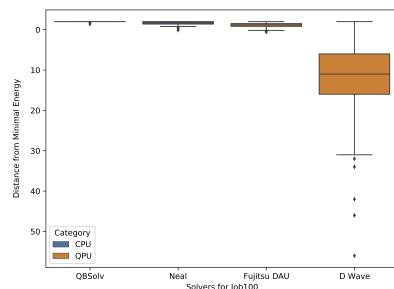
(a) Solver Abweichungen für Job10



(b) Solver Abweichungen für Job20



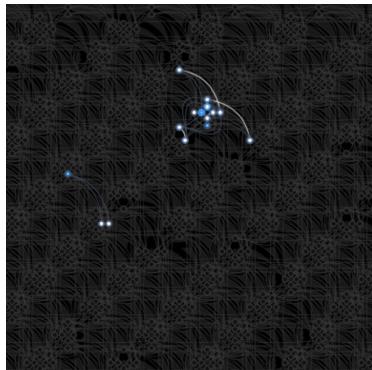
(c) Solver Abweichungen für Job50



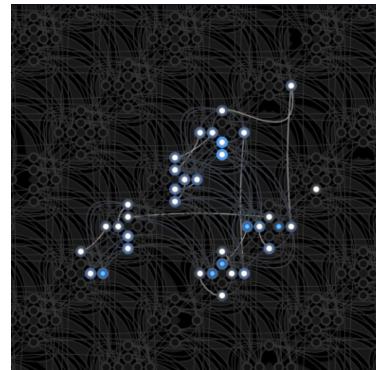
(d) Solver Abweichungen für Job100

Abbildung 14: Alle Abweichungen vom Energie-Minimum ( $\hat{=} 0$ ) nach Solver-Art für Job10, Job20, Job50 und Job100

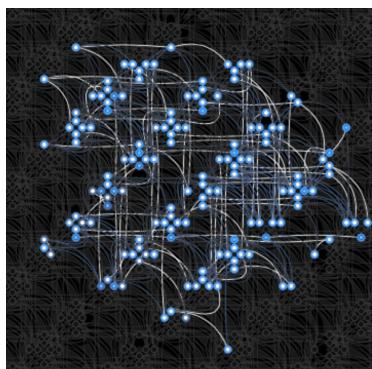
## Weitere Grafiken



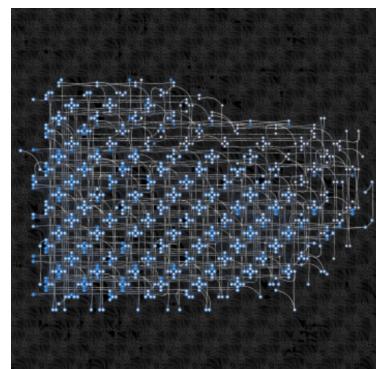
(a) Embedding für Job10



(b) Embedding für Job20



(c) Embedding für Job50



(d) Embedding für Job100

Abbildung 15: Embeddings mit der besten Performance für Job10, Job20, Job50 und Job100

## Autorenschaft

Ege Çimşir hat den Abschnitt 2.2 verfasst. Den Abschnitt 3 haben Tobias Rohe und Ege Çimşir gemeinsam verfasst. Den Abschnitt 2.4 hat Tobias Rohe verfasst. Jonas Gottal und Viktoria Patapovich haben die Abschnitte 2.3, 4.2, und 5.1 verfasst. Jonas Gottal und Tobias Rohe haben den Abschnitt 4.3 gemeinsam verfasst. Sebastian Silva hat die Abschnitte 2.1 und 4.1 verfasst. Den Abschnitt 5.2 haben Ege Çimşir, Sebastian Silva und Tobias Rohe gemeinsam verfasst. Die Abschnitte 1 und 6 wurden von allen Autoren gemeinsam verfasst. Wir bedanken uns bei den Betreuern Jonas Stein, Sebastian Zielinski und Leo Sünkel sowie bei Frau Prof. Dr. Claudia Linnhoff-Popien für ein spannen-

des Semester und die Chance am *QAR-Lab* mit Quantencomputern arbeiten zu dürfen.

## Literatur

- [1] Majid Abdolrazzaghi-Nezhad and Salwani Abdullah. Job shop scheduling: Classification, constraints and objective functions. September 2017.
- [2] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), January 2018.
- [3] Michael Booth, Steven P. Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical/quantum execution. Technical report, D-Wave, October 2017.
- [4] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors. Technical report, D-Wave, February 2019.
- [5] R. Colwell. The chip design game at the end of moore's law. *2013 IEEE Hot Chips 25 Symposium (HCS)*, pages 1–16, 2013.
- [6] D-Wave. *D-Wave Hybrid Solver Service: An Overview*, February 2020.
- [7] D-Wave. *D-Wave Problem-Solving Handbook*, June 2021.
- [8] D-Wave. *D-Wave Solver Properties and Parameters Reference*, June 2021.
- [9] D-Wave. *Technical Description of the D-Wave Quantum Processing Unit*, May 2021.
- [10] Berend Denkena, Fritz Schinkel, Jonathan Pirnay, and Sören Wilmsmeier. Quantum algorithms for process parallel flexible job shop scheduling. *CIRP Journal of Manufacturing Science and Technology*, 33:100–114, May 2021.
- [11] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. February 2000.
- [12] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Frontiers in ICT*, 6, June 2019.
- [13] Dillion M. Fox, Kim M. Branson, and Ross C. Walker. mrna codon optimization on quantum computers. *bioRxiv*, 2021.
- [14] David J. Griffiths. *Introduction of Quantum Mechanics*. Cambridge University Press, second edition, 2017.

- [15] Matthias Homeister. *Quantum Computing verstehen - Grundlagen – Anwendungen – Perspektiven*. Springer-Verlag, Wiesbaden, 2018.
- [16] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58:5355–5363, November 1998.
- [17] Krzysztof Kurowski, Jan Węglarz, Marek Subocz, Rafał Różycski, and Grzegorz Waligóra. Hybrid quantum annealing heuristic method for solving job shop scheduling problem. In Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira, editors, *Computational Science – ICCS 2020*, pages 502–515, Cham, 2020. Springer International Publishing.
- [18] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305 – 3310, 2017.
- [19] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014.
- [20] Jeffrey Marshall, Eleanor G. Rieffel, and Itay Hen. Thermalization, freeze-out, and noise: Deciphering experimental quantum annealers. *Physical Review Applied*, 8(6), December 2017.
- [21] Catherine McGeoch and Pau Faré. The d-wave advantage system: An overview. Technical report, D-Wave, September 2020.
- [22] Christoph Roch, Thomy Phan, Sebastian Feld, Robert Müller, Thomas Gabor, Carsten Hahn, and Claudia Linnhoff-Popien. A quantum annealing algorithm for finding pure nash equilibria in graphical games. In Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira, editors, *Computational Science – ICCS 2020*, pages 488–501, Cham, 2020. Springer International Publishing.
- [23] Sanroku Tsukamoto, M. Takatsu, S. Matsubara, and Hirotaka Tamura. An accelerator architecture for combinatorial optimization problems. *Fujitsu Scientific and Technical Journal*, 53:8–13, September 2017.
- [24] Davide Venturelli, Dominic J. J. Marchand, and Galo Rojo. Quantum annealing implementation of job-shop scheduling. 2016.