AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Solving Optimization problems using Qiskit Aqua

Rozwiązywanie wybranych problemów optymalizacyjnych za pomocą Qiskit Aqua

Autor:	*Małgorzata Agnieszka Stachoń*
Kierunek studiów:	*Informatyka*
Typ studiów:	*Stacjonarne*
Opiekun pracy:	*dr inż. Katarzyna Rycerz*

Kraków, 2020

Oświadczenie studenta

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

……………………………………………
(czytelny podpis studenta)

Abstract

Nowadays, quantum computers are attracting more and more attention to pave the way for new opportunities in computer science. One of the main areas that can benefit from the 'quantum revolution' are optimization problems. This work focuses on exploring the possibility of solving a popular optimization problem, workflow scheduling (assigning tasks in a cloud environment) using a quantum computer provided by IBM with the use of Qiskit quantum framework. In this thesis, the selected optimization problem is solved with the use of a hybrid VQE (Variational Quantum Eigensolver) algorithm. The solution was tested on a simulator from the Qiskit framework and on a real quantum device with 15 qubits, provided by IBM Q. The results presented in this thesis, obtained on a simulator, prove that it is possible to use a quantum solution to solve the workflow scheduling problem and it yields very good results. Unfortunately, due to the imperfections of current generation quantum computers, the results obtained from a real quantum computer are not ideal, imperfections are caused by the phenomena of decoherence and noise. Good results from simulators give good prospects for the future in terms of the possibility of applying quantum solutions in practice.

Streszczenie

W dzisiejszych czasach komputery kwantowe przyciągają coraz to większą uwagę, gdyż otwierają drogę na nowe możliwości w Informatyce. Jednym z obszarów mogących szczególnie zyskać na “kwantowej rewolucji” są problemy optymalizacyjne. Praca ta, skupia się na sprawdzeniu możliwość rozwiązania popularnego problemu optymalizacyjnego "workflow scheduling"(przydział zadań w środowisku chmurowym) przy użyciu kwantowego komputera dostarczonego przez firmę IBM z wykorzystaniem kwantowego frameworka Qiskit. W pracy przedstawione jest jak rozwiązać wybrany problem optymalizacyjny, przy użyciu hybrydowego algorytmu VQE (Variational Quantum Eigensolver). Rozwiązanie zostało przetestowane na symulatorze udostępnionym przez framework Qiskit oraz prawdziwym kwantowym komputerze posiadającym 15 qubitów, udostępnionym przez IBM Q. Przedstawione w pracy wyniki otrzymane na symulatorze, pokazują, że możliwe jest wykorzystanie kwantowego podejścia do rozwiązania problemu workflow scheduling i daje ono bardzo dobre rezultaty. Niestety ze względu na niedoskonałości obecnej generacji komputera kwantowego wyniki uzyskane na prawdziwym kwantowym komputerze odbiegają od ideału, związane jest to ze zjawiskiem dekoherencji oraz szumów. Dobre wyniki z symulatorów dają dobre rokowania na przyszłość pod względem możliwości wykorzystania rozwiązania kwantowego w praktyce.

Acknowledgments

I would like to express my genuine gratitude to my supervisor, dr inż. Katarzyna Rycerz for her patience, guidance, and immense knowledge. Her valuable substantive remarks and comprehensive supervision helped me a lot with the research.

Contents

Chapter 1

Introduction

1.1 Preface

Nowadays quantum computing is becoming more and more popular. Big companies invest in the development of quantum machines as they see big opportunities in using them. Quantum computing, in theory, enables solving some difficult problems in a faster and more efficient way [1][42]. Sometimes it is even not possible to solve a problem without quantum mechanisms.

A universal quantum computer that can simulate any quantum machine or simulator with arbitrary precision may have many usages like: cracking RSA [9], quantum cryptography [10] or exploring multiple possibilities of execution in a single QPU clock. One of the most promising and useful ways of exploiting quantum computers is solving optimization problems. This area of research could open the door of solving problems currently unsolvable. The goal of this thesis is to choose one of the optimization problems and try to solve it with a quantum computer.

Cloud computing [8] is currently one of the major topics in software development. Cloud computing is based on offloading workload to remote data centers, typically by the internet. The major cloud computing providers like Amazon, Google, or Microsoft gain popularity and are widely and willingly used by millions of users as well as other companies. Cloud computing gained a lot of popularity thanks to its ease of use, fault tolerance, flexibility, and cost-effectiveness. It utilizes the pay-as-you-go model, therefore effective scheduling and planning can produce large cost reductions. This is the reason why the workflow scheduling is a very relevant problem. Efficient planning may have a big impact on system behavior. Therefore in this work, the chosen problem for optimization is workflow scheduling.

The general purpose of workflow scheduling is to find the most optimal way of resource allocation. The workflow scheduling problem is an NP-hard problem, therefore it is not possible to test all cases and choose the best one in a reasonable time. There are many different techniques of how to approach this problem, that focus on different objectives. Some of them are presented in [37].

1.2 Motivation

Since quantum computers can give new opportunities, this work will explore the possibility of solving an optimization problem with the use of a quantum computer delivered by IBM. IBM invests in quantum computing and created a framework called Qiskit [25] for developing quantum algorithms and a web tool – IBM Quantum Experience (IBM Q) [15] that enables cloud access to real quantum devices. Qiskit Aqua [26] is a part of Qiskit and provides high-level structures for creating quantum algorithms. As workflow scheduling is an NP-hard problem using quantum algorithms may in the future result in producing better result than traditional computation, therefore this work will examine the possibility of solving this problem with the use of quantum mechanics and its usage for quantum computation provided by IBM Q.

1.3 Goal of the thesis

The following issues will be addressed in this thesis.

- Explore and evaluate the possibilities of Qiskit Aqua framework, in terms of solving optimization problems.
- Find a solution to the workflow scheduling problem with the use of a quantum computer provided by IBM through the IBM Quantum Experience interface.

- Comparison of the results received from using a classical computer and quantum computer.
- Comparison and evaluation of the results received from quantum computers and simulators.
- Check the usability of the DOcplex module in terms of solving optimization problems.

1.4 Content description

The chapter 2 presents a technical background and a short introduction to quantum computing. In the second part, the IBM Quantum and Qiskit framework are introduced. The chapter 3 depicts a workflow scheduling problem as well as the workflow scheduling objectives and solutions. Chapter 4 describes the necessary methodology and steps to solve the problem with the IBM quantum computer. In the chapter 5, steps for mapping the problem to an executable form in a quantum computer and the solution to the workflow problem with the Qiskit tool are depicted. In chapter 6 the exact testing problem and results received from experiments are presented. The conclusion and future extensions are described in chapter 7.

Chapter 2

IBM Support for quantum computing

This chapter presents some basic knowledge about quantum computing and the current state of quantum computers, focusing on computers delivered by IBM. In the second part, there is information about the Qiskit framework and what parts it consists of.

2.1 Quantum computing

Using quantum computers to solve problems may greatly reduce the execution time and resources usage in comparison with traditional computers thanks to the usage of quantum mechanics phenomena [1]. Two most important quantum phenomena are called superposition [24] and quantum entanglement [14]. The first one gives the ability for the system to exists in several states at the same time. The second, creates a bond between particles, even if they are separated, which results in an effect that the change of the state of one particle affects the other.

Quantum computers instead of operating on classical two state bits operate on quantum bits called qubits. Qubit is the basic unit of quantum information and conventionally is represented by $|0\rangle$ and $|1\rangle$. Using Dirac notation qubit can also be written as in (2.1).

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

The representation of a single qubit is a linear combination of two states: $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix},\ \ \alpha,\beta \in C,\ \ |\alpha|^2 + |\beta|^2 = 1, \tag{2.2}$$

where α and β represents probability amplitudes and $|\alpha|^2 + |\beta|^2 = 1$. It is not possible to measure quantum state without changing it. After measurement value 0 is received with probability $|\alpha|^2$ and value 1 with $|\beta|^2$, where α and β are complex numbers.

The quantum system may consist of many qubits. The state of the whole system is a tensor product of all individual qubits. For example the state that is composed by two quits $|\psi\rangle$ and $|\psi'\rangle$ is presented in equation (2.3). The $\otimes$ is a tensor product but it can be omitted and the simplified notation is $|\psi\psi'\rangle$.

$$|\psi\rangle \otimes |\psi'\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \otimes \begin{bmatrix}\alpha'\\ \beta'\end{bmatrix} = \begin{bmatrix}\alpha\alpha'\\ \alpha\beta'\\ \beta\alpha'\\ \beta\beta'\end{bmatrix} \tag{2.3}$$

The current state of the system is a superposition of all possible states of individual qubits, therefore the problem space is much wider. For the quantum state that consists of n qubits the problem space is 2^n.

To operate on qubits, quantum gates are used, which are unitary matrices. Basic quantum gates that operate on a single qubit are presented in (2.4). The I represents a unit matrix. This gate does not change the qubit state. The σ_x gate operates on a single qubit and acts as NOT gate. When the qubit has value $|0\rangle$ this gate sets it to $|1\rangle$ and when the qubit has value $|1\rangle$ this gate set value to $|0\rangle$. The σ_y gate maps qubit $|0\rangle$ to $i|1\rangle$ and qubit $|1\rangle$ to $-i|0\rangle$. The σ_z gate does not change the $|0\rangle$ and maps $|1\rangle$ to $-|1\rangle$. The gates: σ_x, σ_y and σ_z are called Pauli gates.

$$I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \quad \sigma_x = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \quad \sigma_y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} \quad \sigma_z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \tag{2.4}$$

The (2.5) is called controlled not gate and acts on two qubits. This gate operates on the second qubit only when the first qubit is set to $|1\rangle$, otherwise, the qubits remain unchanged.

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

Current quantum computers do not work ideally. There are several challenges, that engineers are faced with. The biggest limitation imposed on quantum computers is quantum decoherence [38]. It is extremely hard to isolate qubits from the environment and the external world. Right now few real quantum computers exist and even fewer are available publicly. Those computers have restricted capabilities and do not work ideally, but are capable enough to test and experiment with some ideas.

Publicly available quantum computers

Below, publicly available quantum computers are shortly presented:

- **D-Wave 2000Q** [3] - people can use this computer and run experiments through a web interface called Leap [18]. This computer implements a special-purpose quantum annealing processor, which is not a general-purpose quantum processor. However, people can successfully run experiments on this machine. It theoretically provides 2048 qubits.

- **IBM Q** enables access to several types of quantum computers through the online platform called IBM Quantum Experience [15]. As opposed to D-Wave, IBM Q provides real general-purpose quantum computers. There are a couple of different types of quantum computers with 1, 5 and 15 physical qubits.

- **Rigetti** [36] provides cloud platform – Forest. This platform allows user to run their algorithms on a simulated quantum chip with 36 qubits. This platform provides a custom API called Quil for python.

These computers have different properties and a different number of qubits but they should not be directly compared to each other since they have different architectures and purposes.

Problems solvable by quantum computers

There were several theoretical quantum algorithms proposed in the 1990s. Factorization of large numbers with the use of quantum superposition and quantum Fourier transform is feasible in polynomial time. This algorithm was proposed by Peter Shor in 1994 and is commonly known as Shor's algorithm [40]. Grover algorithm from 1996 [13] enables searching unordered data in time $O(\sqrt{N})$, what is much faster than using classical algorithms. Those algorithms are just theoretical examples, however, quantum computing may benefit many real-life fields of studies. The known usages are for example traffic optimization, weather forecasting, or fertilization optimization. There are already companies [11] that use quantum computers for different reasons and purposes like DAIMLER AG for development of car batteries, or PROTEINQURE for molecule simulation and drug discovery.

2.2 IBM Quantum

IBM in recent years put a lot of focus on quantum computing. It developed tools for simulating and running quantum experiments. IBM Q System One released in January 2019 made a breakthrough. This quantum computer with 20 qubits was a first commercial quantum machine, designed in a way that it does not have to be kept in a special sterile environment since it is kept inside a cube with a side less than 3 meters. It is important, as previous quantum computers easily could take a whole room and were not movable. This situation resembles early development in traditional computers as the first systems were enormous and latter became more and more compact. The next big leap forward was releasing quantum computer with 53 qubits called Rochester. It was created in the same year and since October 2019 is available to researchers [34]. All quantum computers are available through platforms:

- IBM Q Experience - this web platform is available to everyone and enables access to 1, 5 and 15 qubit quantum computers.
- IBM Q Network - this platform is for researchers and business and makes access also to 20 and 53 qubits processors.

This work will focus on IBM Quantum Experience since it is open and available to everyone. IBM Quantum Experience is an online platform that enables access to IBMs quantum computers. It also delivers materials for educational purposes and a forum for discussing IBM Q related issues. The IBM Q delivers a web tool that enables creating experiments. It contains a graphical circuit composer and provides qasm language for running experiments.

2.2.1 Available Quantum computers at IBM Quantum Experience

IBM Quantum Experience delivers simulators as well as real quantum hardware for performing quantum computing. For now there are 9 real quantum computers presented in table 2.1. IBM also provides ibmq_qasm_simuator with 32 qubits.

Table 2.1: Table presents available quantum computers through IBM Q interface.

name of quantum computer	number of qubits
ibmq_armonk	1
ibmqx2	5
ibmq_vigo	5
ibmq_ourense	5
ibmq_london	5
ibmq_burlington	5
ibmq_essex	5
ibmq_rome	5
ibmq_16_melbourne	15

2.3 Qiskit

Qiskit [25] is a novel framework for quantum computing. The roots of Qiskit development date back to the beginning of 2017, but since the end of 2018, it began to be quickly developed. Qiskit is an open-source framework for performing quantum computations. It provides different simulators, noise models, and algorithms. Qiskit consists of 4 main components: Qiskit Terra, Qiskit Aer, Qiskit Ignis and Qiskit Aqua. Figure 2.1 presents Qiskit components and how they interact with each other. Aer element is the lowest in the hierarchy. It provides simulators and emulators. The next component is Terra and it contains API for creating quantum circuits. Aqua and Ignis are additional libraries build on top of Terra. Aqua component is a high-level element that provides algorithms and high-level structures. Ignis is an element for noise imitation.

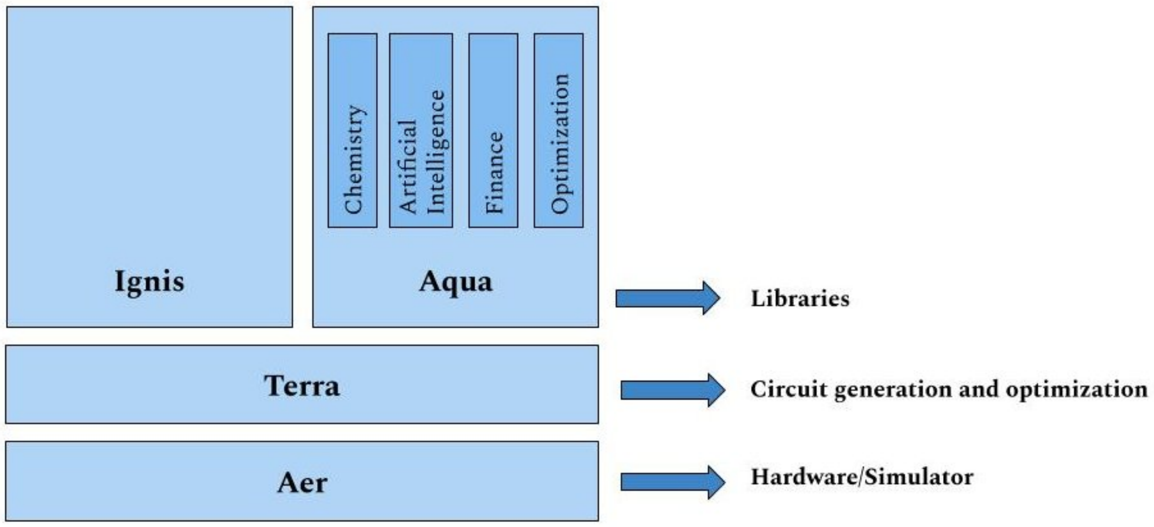

Figure 2.1: Qiskit software stack

Qiskit provides API in Python programming language for easy and simple experimenting and learning. It enables its users to create quantum circuits and visualize them as well as use more complex features. Qiskit also comes up with a great amount of ready to use algorithms and functions. The biggest advantage is that Qiskit is free and open to everyone. It is integrated well with IBM Quantum Experience and users can run their programs on real quantum computers in a very easy and robust way.

2.3.1 Qiskit Terra

This is a main module of Qiskit, it contains basic elements that enable quantum circuit creation. It provides an interface between high-level end-user API and handling of backend communication and pulse scheduling. Qiskit Terra contains elements that transpile experiments so that they can be performed on quantum hardware. It also delivers components for the optimization of all layers. It also implements many visualization functions for displaying quantum circuits as well as receiving results from quantum experiments.

2.3.2 Qiskit Aer

Qiskit Aer provides simulators, emulators, and debuggers. Current quantum computers have many limitations and do not perform ideally, therefore it is crucial to test experiments on simulators. Qiskit Aer provides high-performance simulator backends for running experiments compiled with the use of the Terra module. It also contains tools for constructing noise models so that it is possible to perform high-quality noise simulation. The Aer module delivers several backends for simulation:

- **statevector_simulator** - this simulator executes a quantum circuit exactly once and returns the exact result of the quantum state, the received results are state vector amplitudes.
- **qasm_simulator** - this simulator is designed to imitate the actual quantum device, it yields results in a histogram form.
- **unitary_simulator** - it creates a unitary matrix, that represents a quantum circuit.

2.3.3 Qiskit Ignis

Qiskit Ignis is used to present and mitigate noise and errors that appear in quantum circuits. This module focuses on gates improvement, error characterization improvement, and computation improvement with the presence of noise. It enables three different types of experiments that can be performed:

- characterization - measurements of system parameters,
- verification - verification of gates and circuit performance,
- mitigation - running calibration circuit that helps with creation of error mitigation methods.

2.3.4 Qiskit Aqua

Qiskit Aqua [26] is the library written in Python language that enables building algorithms. It is built on top of the Terra module and delivers high-level mechanisms to build and develop quantum algorithms, that may be used to find solutions to real-life problems [19]. Qiskit Aqua put focus on finding solutions to real-life problems with quantum computers and so far it focuses on four areas:

- Optimization
- Chemistry
- Artificial Intelligence
- Finance

Qiskit Aqua delivers several ready to use algorithms [27] such as VQE [33], QAOA [30], Quantum Fourier Transform[31] and Grover's [28] that are domain-independent, but also several domain-specific knowledge algorithms including Quantum Kernel algorithm and Support Vector Machine (SVM). The Aqua interface is easy to understand and use even for non-technical people and does not require to learn new programming language.

There are already some optimization problems that are solved with Qiskit Aqua like Traveling Salesmen Problem or Max-Cut. They are solved with Variational Quantum Eigensolver (VQE) [22] algorithm.

2.4 Summary

Although available quantum computers have limited resources it is possible to test and experiment with some optimization problems. Thanks to IBM Quantum Experience and Qiskit Aqua framework quantum optimization is available to everyone.

Chapter 3

Workflow scheduling

This chapter describes the optimization problem chosen for this work – workflow scheduling. It also contains a short overview of the workflow scheduling objectives and currently available solutions. In the second part, there is a mathematical definition of workflow scheduling problem solved and an example problem with a solution.

3.1 Scientific workflow overview

Scientific workflow enables an easier understanding of complex processes that consists of many tasks. It presents dependencies between tasks. Scientific workflows are popular in different fields of studies such as astronomy, biology, or chemistry. Below are some examples:

- CyberShake [12] - CybyberShake is a workflow-based solution for probabilistic seismic hazard analysis. It uses a physics-based 3D wave propagation simulation for forecasting ground motions. This approach is expected to produce more accurate results that an empirical-based solution.
- Montage [2] - montage uses a scientific workflow to provide the tools needed for building image mosaics in Flexible Image Transport System (FITS) format. It supports all common astronomical coordinate systems, all World Coordinate System (WCS) map projections, arbitrary image sizes (including full-sky images) and rotations, and user-specified spatial sampling. Mosaic is suitable for large scale processing of the sky.

3.2 Objectives of workflow scheduling problem

The main goal of the workflow scheduling problem is to assign some resources with the minimalization of some objective. It is usually represented as a DAG (Directed Acyclic Graph), therefore the tasks must be performed in a specific order and only some parts can be parallelized.

Since the workflow scheduling problem nowadays is very relevant, there are many propositions and solutions to it. In [37], there is a short retrospection of available solutions. Workflow scheduling algorithms differ by their goals which are for instance: cost, time limit, load balancing, Quality of Service, energy consumption, security awareness, and with service level agreement. Different workflow scheduling problems focus attention on different measures however the most popular are cost and time minimization.

3.3 Current solutions to workflow scheduling problems

There are many solutions to workflow scheduling problems that are focused on different optimization criteria [37]. Generally, those algorithms can be divided into two categories: dynamic scheduling algorithms [4] and static scheduling algorithms [17]. The main difference between those two categories is the time when scheduling happens, static algorithms create a schedule before any computation is started, whereas dynamic ones perform scheduling in real-time. All existing algorithms deliver approximate or near-optimal solutions as workflow scheduling is an NP-hard problem it is impossible to generate an optimal solution that works in polynomial time. As an example of a workflow scheduling algorithm, one could provide HEFT (Heterogeneous Earliest Finish Time) [7], which is a greedy heuristic algorithm. Other popular are genetic and PSO (Particle Swarm Optimization) [16].

3.4 Workflow scheduling problem formal definition

We have a given set of machines, each of them has a specific operating cost and efficiency. Tasks are represented by a DAG and must be performed in a specific order. There is also a time limit set in advance. The goal is to assign for each task a type of machine that the overall cost will be the smallest and all tasks will finish within a specific time limit. The child task can be started only when all parent tasks are finished.

The formal, mathematical definition of the workflow scheduling problem [43] consists of the following assumptions:

1. There is given a time matrix $T = [t_{i,j}]_{N\times M}$ where N is the number of tasks, M is the number of machines and $t_{i,j}$ represents the execution time of task with number i on the machine with number j.
2. There is also a given machine usage cost per time unit $K = [k_j]_M$, which is measured in a currency per time unit. This data is used to calculate the cost of running a task on a specific machine $C = [c_{i,j}]_{N\times M}$, where $c_{i,j}$ represents the cost of performing the task with the number i on the machine with number j.
3. There is also a provided deadline d for the workflow to finish, which is represented as a number.
4. There is also a list of paths R from the first vertex to the final in a DAG, that represents the problem.

Based on these assumptions it is possible to create a formal definition of the problem. The goal is to choose $x_{i,j} = \{0,1\}$, that represents performing a task i on machine j (when $x_{i,j}$ is set to 1 this means that the task i will be performed on machine j, 0 otherwise), so that all constraints ((3.2), (3.3)) will be satisfied with the minimization of the cost function (3.1):

$$f(x) = \sum_{i}^{N}\sum_{j}^{M} c_{i,j}x_{i,j}, \tag{3.1}$$

where $c_{i,j}$ represents cost of performing a task i on machine j.

Constraints:

1. Only one machine can be assigned to every task:

$$\sum_{j}^{M} x_{i,j} = 1,\ \forall i \in \{1,..,N\} \tag{3.2}$$

2. All tasks should finish before time limit:

$$\forall r \in R \sum_{i}^{r} \sum_{j}^{M} t_{i,j}\, x_{i,j} \le d, \tag{3.3}$$

   for every path execution time should be less or equal than the deadline, where R consists of list of tasks in path and $t_{i,j}$ represents time that is required to perform a task i on machine j.

The correct solution for workflow scheduling problem is when all constraints are satisfied. The optimum (most efficient result) is when the result is correct with the lowest possible cost. The wrong result is when the solution does not meet at least one constraint.

Diagram 3.1 presents a graph as an example workflow problem with four tasks. The nodes represent tasks and edges paths and dependencies between tasks. In the example below there are 2 machines with different speeds and cost per time unit. Machine 0 with 1 cost unit per time unit and machine 1 with 4 cost units per time unit. The table 3.1 presents the time that is required to finish a task on a specific machine. The table 3.2 presents the cost of performing a task on a specific machine. The time limit is set to 26.

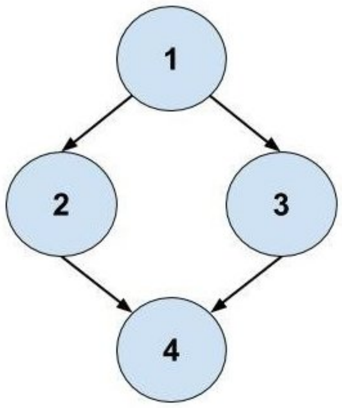

Figure 3.1: Example workflow diagram

Table 3.1: The table presents the time that is required to finish a task on a specific machine. The $t_{i,j}$ represents a time that is required to perform task i on the machine with number j for an example workflow scheduling problem.

	task 1	task 2	task 3	task 4
machine 0	6	3	12	9
machine 1	2	1	4	3

Table 3.2: The table presents the cost of performing a task on a specific machine. The $c_{i,j}$ represents a cost that is required to perform task i on the machine with number j for the example workflow scheduling problem.

	task 1	task 2	task 3	task 4
machine 0	6	3	12	9
machine 1	8	4	16	12

Table 3.3 presents all configurations, in which every task is executed exactly once. The $t_{i,j}$ represents the time that is required to perform task i on the machine with number j. The first path consists of tasks with numbers 1, 2, and 4. The second path consists of tasks with numbers 1, 3, and 4. Configurations that fit in time limit are marked with gray. Among the configurations that fit in the time limit, the one with the lowest cost is chosen and marked with blue color. For this problem, the optimal configuration is when the first task is performed on a machine with number 1 and all the rest tasks are executed on the machine with number 0. This configuration in total takes 23-time units and 32 cost units.

Table 3.3: Table presents all configurations for the example problem, where every task is assigned to exactly one machine. All correct solutions are marked with gray. The most efficient solution is marked with blue.

Configuration	total time for path 1	total time for path 2	total cost
$t_{1,0}+t_{2,0}+t_{4,0}$, $t_{1,0}+t_{3,0}+t_{4,0}$	18.0	27.0	30.0
$t_{1,0}+t_{2,0}+t_{4,1}, t_{1,0}+t_{3,0}+t_{4,1}$	12.0	21.0	33.0
$t_{1,0}+t_{2,0}+t_{4,0}, t_{1,0}+t_{3,1}+t_{4,0}$	18.0	19.0	34.0
$t_{1,0}+t_{2,0}+t_{4,1}, t_{1,0}+t_{3,1}+t_{4,1}$	12.0	13.0	37.0
$t_{1,0}+t_{2,1}+t_{4,0}, t_{1,0}+t_{3,0}+t_{4,0}$	16.0	27.0	31.0
$t_{1,0}+t_{2,1}+t_{4,1}, t_{1,0}+t_{3,0}+t_{4,1}$	10.0	21.0	34.0
$t_{1,0}+t_{2,1}+t_{4,0}, t_{1,0}+t_{3,1}+t_{4,0}$	16.0	19.0	35.0
$t_{1,0}+t_{2,1}+t_{4,1}, t_{1,0}+t_{3,1}+t_{4,1}$	10.0	13.0	38.0
$t_{1,1}+t_{2,0}+t_{4,0}, t_{1,1}+t_{3,0}+t_{4,0}$	14.0	23.0	32.0
$t_{1,1}+t_{2,0}+t_{4,1}, t_{1,1}+t_{3,0}+t_{4,1}$	8.0	17.0	35.0
$t_{1,1}+t_{2,0}+t_{4,0}, t_{1,1}+t_{3,1}+t_{4,0}$	14.0	15.0	36.0
$t_{1,1}+t_{2,0}+t_{4,1}, t_{1,1}+t_{3,1}+t_{4,1}$	8.0	9.0	39.0
$t_{1,1}+t_{2,1}+t_{4,0}, t_{1,1}+t_{3,0}+t_{4,0}$	12.0	23.0	33.0
$t_{1,1}+t_{2,1}+t_{4,1}, t_{1,1}+t_{3,0}+t_{4,1}$	6.0	17.0	36.0
$t_{1,1}+t_{2,1}+t_{4,0}, t_{1,1}+t_{3,1}+t_{4,0}$	12.0	15.0	37.0
$t_{1,1}+t_{2,1}+t_{4,1}, t_{1,1}+t_{3,1}+t_{4,1}$	6.0	9.0	40.0

3.5 Summary

Although there are many objectives in the workflow scheduling problem the most common are cost and time, these objectives are also the ones that this work focuses on. In this work, the workflow must finish within a specified time limit and its cost should be minimal.

Chapter 4

Solving optimization problems on gate-based quantum computers

This chapter describes all the necessary structures that have to be used and steps that have to be taken to solve an optimization problem on an IBM quantum computer using the VQE optimization algorithm.

4.1 Solving optimization problem

Quantum computers that are available today are called near-term quantum machines since these machines do not provide error-free execution with the use of millions of qubits. They have limited resources and because of that very popular are optimization algorithms that are hybrid and consist of using both classical and quantum computing [39].

4.1.1 Hamiltonian

Ising is a mathematical model that consists of particles and every particle can be in state $\{-1,1\}$. Hamiltonian is an energy function that describes the energy of the system. In the classical Ising model with N spins that has values $s_i = \{-1,1\}$, the Hamiltonian have the following form:

$$H(s_1, ..., s_N) = -\sum_{i<j} J_{i,j} s_i s_j - \sum_{i=1}^{N} h_i s_i, \tag{4.1}$$

where $J_{i,j}$ represents the interaction force between spins s_i and s_j whereas h_i the external force that interacts on the spin s_i. The quantum version of classical Hamiltonian (also called Hamiltonian Operator) represents energy of the quantum system. It is the sum of all kinetic and potential energies in the system. Its eigenvalues represent the energy values that this system may have. Ising Hamiltonian is constructed as a sum of Pauli Operators σ_x, σ_y, σ_z. The quantum version of Ising Hamiltonian has the following form:

$$H(\sigma_z^1, ..., \sigma_z^n) = -\sum_{i<j} J_{i,j} \sigma_z^i \sigma_z^j - \sum_{i=1}^{N} h_i \sigma_x^i, \tag{4.2}$$

where $J_{i,j}$ represents the interaction force between spins σ^i and σ^j whereas h_i the external force that interacts on the spin σ^i.

In this work in the following chapters, occurrences of Ising Hamiltonian mean its quantum version.

4.1.2 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) [22] is a hybrid algorithm, that approximates eigenvalues of a matrix H, that is in most cases the system's Hamiltonian. The basic concept is that finding the minimal eigenvalue of the matrix is used to approximate energy of the system ground state. The ground state of the system is the state when the system has the lowest energy. This algorithm consists of classical and quantum part:

- **classical part** is run on classical CPU,
- **quantum part**, that is run on quantum hardware.

Variational Quantum Eigensolver Algorithm shown on figure 4.1 consists of several steps:

1. Preparation of quantum state, that will be processed on quantum modules.

2. Computation of the expectation values for the prepared state. Quantum state is passed to quantum modules, that compute H_i - part of the sum of the Hamiltonian.

3. The results from quantum modules are passed to the CPU, where the sum of H_i – H is computed and then according to the variational principle the eigenvalues are computed. The variational principle is based on the assumption that the smallest eigenvalue of H is always smaller than the expectation value [44].

4. The classical optimization algorithm is run on the CPU and creates a new parameter set.

5. Result from classical optimizer is passed to the Quantum processing unit. Steps 1-4 are repeated until the minimum is not reached.

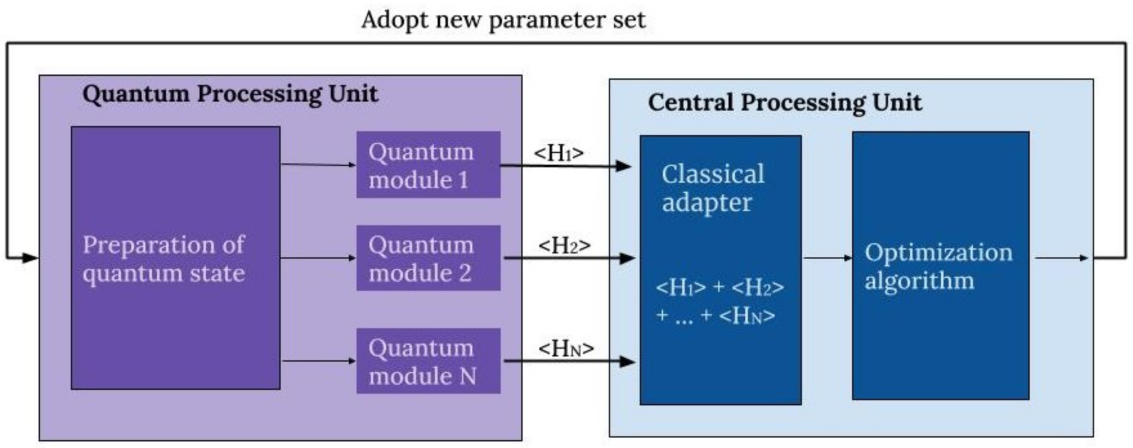

Figure 4.1: Schema of VQE Algorithm. The figure is modeled on [22].

4.1.3 Solving optimization problem on quantum computer

Steps that have to be taken to solve the workflow problem with hybrid VQE algorithm with Qiskit Aqua framework [23]:

1. Represent the problem as quantum Ising Hamiltonian, so that ground state of H is the solution to the optimisation question in a problem.

2. Choose the sub algorithm and parameters of the VQE algorithm.

    - Choose the classical optimization algorithm. All available algorithms are presented in [29], but it is also possible to add a new one. The optimizer must be chosen carefully and may depend on the specific problem. According to [32] for the computations with the presence of noise the Simultaneous Perturbation Stochastic Approximation (SPSA) [41] optimizer is recommended. The advantage of this algorithm is that it makes only two measurements, regardless of the dimension of the problem.
    - Choose the variational form, for the quantum part of the computation which is a parametrized circuit with a fixed form. The variational form presented in Qiskit is RealAmplitudes [35], that is a circuit that consists of σ_y pauli gates and CX entanglement gates. Two most important RealAmpludes parameters are entanglement and reps. Entanglement specifies which qubits are entagled and the reps parameter defines how often the sets of gates are repeated.

3. Choose the backend, on which the algorithm will be run. This can be a simulator or real quantum computer.

4. Choose the shots number - this number represents how many repetitions of the circuit are executed. This parameter is used for sampling.

5. Run the problem on the previously chosen backend. In order to get the same results every time it is necessary to set the seed parameter.

4.2 Summary

Ising Hamiltonian represents the energy of the quantum system. Creating this form is necessary to solve an optimization problem in a quantum computer. VQE the hybrid – classical-quantum algorithm is used to optimize the workflow scheduling problem. The next chapter provides a description of how to map the workflow scheduling problem to a form executable on a quantum computer.

Chapter 5

Creating Hamiltonian for workflow scheduling problem

This chapter describes how to map the workflow scheduling optimization problem to the form executable on a quantum computer and how to use the DOcplex module to simplify the process.

In order to solve an optimization problem with the VQE algorithm on the IBM quantum computer, it is necessary to present the problem in Ising Hamiltonian form. This process consists of several steps. The first step is to map the problem to BILP (Binary Integer Linear Programming) form. In this form, only binary variables are allowed and the objective function and all constraints have to be linear. The BILP formulation enables mapping into the QUBO (Quadratic unconstrained binary optimization) form, from which the problem is translated to Ising Hamiltonian. To simplify the Ising Hamiltonian creation process it is possible to present the problem in BILP form and then use DOcplex [5] module to translate it.

5.1 BILP formulation

The BILP (Binary Integer Linear Programming) is a model where only binary variables are allowed. The goal of solving BILP problem is to find the vector $X = [x_1, ..., x_n]$, for which the function (5.1) reaches minimal value.

$$f(X) = \sum_{i=1}^{N} c_i x_i \tag{5.1}$$

The BILP consists of objective function and constraints that must be linear. Below is presented the mapping of a workflow scheduling problem to BILP, which is based on [43]. In the section 5.1.1 it is presented how to map problem to 0-1 values, in section 5.1.2 is presented how to map the problem's constraints and in 5.1.3 the objective function for minimization is presented.

5.1.1 Binary variables

The problem must consist of variables that take only 0 or 1 values, therefore the vector that represents the problem takes the following form: $X = [x_0, .., x_{N\times M-1}]$, where N is a number of tasks and M is number of machines. The variable x_i is set to 1 if the task with number $i \bmod N+1$ is run on machine with number $i//N$, where // is total division.

Table 5.1 represents an example mapping. There are two machines $M = 2$ and four tasks $N = 4$. The first n variables represent tasks performing on the first machine. The next n variables represent performing tasks on the second machine. In this example, if the x_0 is set to 1 this means that the first task will be performed on a machine with number 0. The vector with values: $X = [1,1,1,1,0,0,0,0]$ means that all tasks will be run on the machine with number 0.

Table 5.1: Table represents an example mapping to BILP problem.

	task 1	task 2	task 3	task 4
machine 0	x_0	x_1	x_2	x_3
machine 1	x_4	x_5	x_6	x_7

5.1.2 Equality constraints

Constraint: machine usage constraint

Exactly one machine must be chosen for every task. A task should not be performed on more that one machine, therefore, it is crucial to have a constraint that exactly one machine is chosen. This is guaranteed by the constraint presented in (5.2).

$$\sum_{j=i,j=j+N}^{M\times N-1} x_j = 1,\ \ \forall i \in \{0,..,N-1\} \tag{5.2}$$

Constraint: time limit constraint

The second constraint is that all tasks should finish within a specific time. In order to satisfy this constraint, every path should finish within the deadline – d. R represents a list of paths. Every path r_γ consists of a vector with $N \times M$ variables that represent the existence of a task in the specific path: $r_\gamma = [r_{i,\gamma}]_{N\times M}$. The (5.3) presents how the vector r_γ is constructed.

$$r_{i,\gamma} = \begin{cases} 1 & \text{if path } \gamma \text{ contains a task number } i\ mod\ N+1, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

The constraint can be written as follows:

$$\forall r_\gamma \in R \quad \sum_{i=0}^{N\times M-1} r_{i,\gamma}\, t_i\, x_i \leq d, \tag{5.4}$$

where t_i represents the time that is required to perform a task $i\ mod\ N+1$ on the machine $i//N$. Workflow execution time will equal to the maximal execution time of paths that the graph consists of. The execution time can be described by the g(x) function (5.5).

$$g(x) = \max_{r_\gamma\,\in\,R}\ (\sum_{i}^{N\times M} r_{i,\gamma}\, t_i\, x_i) \tag{5.5}$$

It is only possible to represent equality constraint, therefore in order to satisfy (5.4) some additional variables are introduced. They are called slack variables and the idea is represented in (5.6).

$$a \in A,\ b \in B,\ a \leq b \leftrightarrow\ \exists\, c \in C,\ a+c = b \tag{5.6}$$

The number of slack variables – S presented on equation (5.7), depends on the difference between the deadline and the fastest completion of the workflow of the problem.

$$S = \lceil \log_2 (d - min(g(x))) \rceil \tag{5.7}$$

The final form of the constraint is presented in (5.8).

$$h(x) = \max_{r_\gamma \in R} \left(\sum_{i=0}^{N\times M-1} r_{i,\gamma}\, t_i\, x_i \right) + \sum_{s=0}^{S} 2^s\, x_s = d \tag{5.8}$$

5.1.3 Function for optimization

In the workflow scheduling problem, the overall cost of running all tasks is optimized, therefore the goal is to find the minimal cost, which is the sum of costs of performing single tasks. This objective function has the following formula:

$$f(x) = \sum_{i=0}^{N\times M-1} c_i x_i, \tag{5.9}$$

where $c_i > 0$ and c_i represents cost of performing a task $i \; mod \; N+1$ on the machine $i//N$.

5.1.4 Summary

In order to map the workflow scheduling problem, it is necessary to have available a specific number of qubits. To map the workflow scheduling problem to the form executable on a quantum computer it is necessary to use $N \times M + S \times R$ number of qubits, where N represents the number of tasks, M the number of machines, R the number of paths and S the number of slack variables that can be computed with the use of equation (5.7).

5.2 Strength of objective function and constraints

In the formulas defining the workflow scheduling problem, later in this work (equations (5.10) and (5.11), figure 5.2), there are three (A, B and C) important variables responsible for setting strength for necessary parts of the optimization problem. In particular the variable A indicates the strength of the objective function (5.9), the variable B the strength of the machine usage constraint (5.2) and variable C the time limit constraint (5.8). All these variables indicate how important the according part is. These variables have to be set for a specific problem. This is a hard and time-consuming part since the values cannot be too big and too small so that all parts have similar importance and strength. This process has to be done manually which takes a lot of time and is not trivial.

5.3 QUBO form and Ising Hamiltonian form

All the above statements enable defining the QUBO form for workflow scheduling problem. The goal is to minimize a quadratic polynomial, with binary variables. The created formula is presented in (5.10). The first part resembles optimization function (5.9), the second part – machine usage constraint (5.2) and the third – time limit constraint (5.8).

$$A(\sum_{i}^{N\times M} c_i\, x_i) + B(\sum_{i=0}^{N}(1 - \sum_{j=i,j=j+N}^{N\times M} x_j)^2) + C(\sum_{r_\gamma}^{R}(d - \sum_{i}^{N\times M} r_{i,\gamma}\, t_i\, x_i + \sum_{s=0}^{S} 2^s\, x_s)^2), \tag{5.10}$$

In order to translate the problem from QUBO form to Ising Hamiltonian it is necessary to map the domain from $x_i = \{0,1\}$ to $x_i = \{-1,1\}$. The Ising Hamiltonian form given by the equation (5.11) is constructed by replacing x_i in (5.10) with $\frac{I-\sigma_z^i}{2}$, so its ground state is equal to the solution to (5.10) and its corresponding minimal eigenvalue is the same as minimum of function (5.10) for $x_i = \{0,1\}$.

$$A(\sum_{i}^{N\times M} c_i \frac{I-\sigma_z^i}{2}) + B(\sum_{i=0}^{N}(1 - \sum_{j=i,j=j+N}^{N\times M} \frac{I-\sigma_z^j}{2})^2) + C(\sum_{r_\gamma}^{R}(d - \sum_{i}^{N\times M} r_{i,\gamma}\, t_i \frac{I-\sigma_z^i}{2} + \sum_{s=0}^{S} 2^s \frac{I-\sigma_z^i}{2})^2), \tag{5.11}$$

5.4 Method 1: DOcplex generation of Hamiltonian

Qiskit Aqua provides DOcplex [5] module that generates Ising Hamiltonian for the specific optimization problem. Creating this operator by ourselves, without any libraries is a very complex process, therefore the possibility of generating it automatically is very useful. In order to call the translator and generate Ising Hamiltonian, the classical optimization model must be in a specified form. Some restrictions must be fulfilled to run the DOcplex translator. Firstly, only binary variables can be used to represent the problem, secondly set constraints can only be equality constrains, thirdly function that will be optimized (minimalized or maximalized) must be linear or quadratic. All these requirements are fulfilled by BILP formulation and are presented in section 5.1.

First, it is necessary to prepare the values presented in figure 5.1, which depends on the specific problem and are in detail described in section 5.1.

Figure 5.1: Variables that have to be set to run the code presented in figure 5.2, that generates Ising Hamiltonian.

```
d # time limit
time # time array
cost # cost array
S # number of slack variables
N # number of tasks
M # number of machines
R # list of paths
```

The code for generating Ising Hamiltonian with the DOcplex module for the workflow scheduling problem is presented in figure 5.2. In order to do that, firstly the model [6] is created. Model creation consists of specifing the number of variables that will be mapped to qubits, specifying the objective function and adding constraints, that are in BILP form. Initiated model can be passed to DOcplex which translates it to the Hamiltonian form. The DOcplex automatically sets the values of variables A, B, and C (see section 5.2), based on the objective function values.

Figure 5.2: Code for generating Ising Hamiltonian with DOcplex module. The variables presented in figure 5.1 must be set before.

```
mdl = Model(name='workflow')
x = {i : mdl.binary_var(name='x_{0}'.format(i)) for i in range(len(time))}
x.update({i : mdl.binary_var(name='x_{0}'.format(i))
             for i in range(len(time), len(time) + S)})

# function for minimalization
tsp_func = mdl.sum(A * cost[i] * x[i] for i in range(len(time)))

# machine usage constraint
for j in range(N):
    mdl.add_constraint(B * mdl.sum(x[i] for i in range(j, M*N, N)) == B)

# time limit constraint for every path
for r in range(0, len(R)):
    mdl.add_constraint(mdl.sum(C * time[i] * x[i] * R[r][i]
            for i in range(len(time)))+ mdl.sum(2 ** (S - j) * x[len(time) + j + S * r]
            for j in range(0, S)) == C * d)

mdl.minimize(tsp_func)
hamiltonian, offset = docplex.get_operator(mdl)
```

5.5 Method 2: manual construction of Hamiltonian

To manually construct the Ising Hamiltonian (5.11), the QUBO formula (5.10) is used. Each part of the QUBO form (5.10) is described as DOcplex Model [6] and then translated to Ising Hamiltonian by using Qiskit functions supporting quadratic programming. The final Hamiltonian is the sum of the created parts. In figure 5.3 this steps are presented as pseudo code. This method requires manual weights (A, B, C) adjustment (see section 5.2), which results in greater control of what is happening.

Figure 5.3: The pseudo code for translating the problem from QUBO formula to Ising Hamiltonian formula.

```
# weights for objective function and constraints
A = ...
B = ...
C = ...

# models creation
objective_function_model = get_model_for_objective_function(weight=A)
machine_usage_constraint_model= get_model_for_machine_usage_constraint(weight=B)
time_limit_constraint_model = get_model_for_time_limit_constraint(weight=C)

objectives = [
    objective_function_model,
    machine_usage_constraint_model,
    time_limit_constraint_model
]

# creating subparts of result Hamiltonian
subparts = []
for objective in objectives:
    quadratic_program = QuadraticProgram()
    quadratic_program.from_docplex(model)
    quadratic_program_to_ising = QuadraticProgramToIsing()
    H, offset = quadratic_program_to_ising.encode(quadratic_program)
    subparts.append(H)

# creating final Hamiltonian
H = subparts[0] + subparts[1]  + subparts[2]
```

5.6 Summary

Hamiltonian represents the energy of a quantum system and in order to run the optimization problem on a quantum machine, this model must be created. This chapter presented two ways of creating an Ising Hamiltonian, one with use of DOcplex module and one with manual weights adjustment. The next chapter presents experiments based on these two creation methods.

Chapter 6

Evaluation of the results

This chapter presents the problem used for testing and achieved results from running three different sets of experiments for different Ising Hamiltonians. Experiments are run on a simulator and a real quantum computer provided by IBM Q.

6.1 Workflow scheduling problem used for testing

The largest quantum computer provided by IBM Quantum Experience has 15 qubits and the chosen problem must fit in that limit, therefore the selected problem consists of 3 tasks and 2 machines and according to mapping in section 5.1.4 could be computed using 10 qubits.

The figure 6.1 represents the workflow scheduling problem used for testing. The problem consists of three tasks $Z = [12, 6, 24]$ and one path $R = [r_1],\ r_1 = [1, 1, 1, 1, 1, 1]$. There are two available machines with different speed $M = [2, 6]$ and cost $K = [1, 4]$ per time unit. The deadline is set to $d = 19$. Table 6.1 presents time $[t_i]$ that is required for specific task to be run on a specific machine. Table 6.2 present the cost $[c_i]$ of performing a specific task on a specific machine.

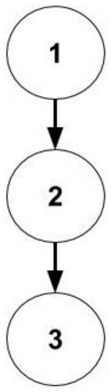

Figure 6.1: Workflow diagram with 3 tasks used for testing.

Table 6.1: The table presents the time that is required to finish a task on a specific machine for the testing problem.

	task 1	task 2	task 3
machine 0	6	3	12
machine 1	2	1	4

Table 6.2: The table presents the cost of performing a task on a specific machine for the testing problem.

	task 1	task 2	task 3
machine 0	6	3	12
machine 1	8	4	16

In the above problem the number of slack variables is $S = 4$ (see equation (5.7)). The vector will have ten values, that will be mapped to ten qubits. The meaning of the individual result values in the result vector is presented in table 6.3.

Table 6.3: The meaning of individual result values in result vector for the testing problem.

position in the result vector	meaning
0	performing task 1 on machine 0
1	performing task 2 on machine 0
2	performing task 3 on machine 0
3	performing task 1 on machine 1
4	performing task 2 on machine 1
5	performing task 3 on machine 1
6	slack variable, represents 8 time units
7	slack variable, represents 4 time units
8	slack variable, represents 2 time units
9	slack variable, represents 1 time unit

The optimization function based on (5.9) is shown in (6.1). The machine usage constraint, based on formula (5.2) is presented in (6.2) and the deadline constraint based on formula (5.8) for this problem have the form presented in (6.3). The *correct solution* is when all constraints ((6.3) and (6.2)) are satisfied. The *optimal solution* is when the solution is correct and the function (6.1) have minimal value. All other results are *incorrect*. The QUBO form for the testing problem based on (5.10) is presented in (6.4).

$$f(x) = \sum_{i=0}^{5} c_i\, x_i, \tag{6.1}$$

$$\sum_{i\in\{0,3\}} x_i = 1 \ \wedge \sum_{i\in\{1,4\}} x_i = 1 \ \wedge \sum_{i\in\{2,5\}} x_i = 1 \tag{6.2}$$

$$\max_{r_\gamma \,\in\, [r_1]} \left(\sum_{i=0}^{5} r_{i,\gamma}\, t_i\, x_i\right) + \sum_{s=0}^{3} 2^s\, x_s = 19 \tag{6.3}$$

$$A\left(\sum_{i=0}^{5} c_i\, x_i\right) + B\left(\sum_{i=0}^{2}\left(1 - \sum_{j=i,j=j+3}^{5} x_j\right)^2\right) + C\left(d - \sum_{i} t_i\, x_i + \sum_{s=0}^{3} 2^s\, x_s\right)^2\Big), \tag{6.4}$$

The example result is [1 1 0 0 0 1 0 1 1 0]. It should be interpreted in the following way:

the tasks 1 and 2 will be performed on a machine with number 0 and task 3 will be performed on a machine with number 1. The execution time of all tasks is 13-time units and cost 25 cost units. When counting slack variables the total time is 19.

The optimal solution for this problem is [1 0 1 0 1 0 0 0 0 0], which means that the task 1 and 3 will be performed on a machine with number 0 and task 2 will be performed on a machine with number 1. The total performing cost will be 22 cost units and time 19-time units.

6.2 Creating Ising Hamiltonian

There were three types of conducted experiments. Two of them used DOcplex Module to translate the problem to Ising Hamiltonian form and the method is presented in section 5.4. The third one was constructed without this module, with the method described in section 5.5. These experiments are presented and in detail described below. Every Ising Hamiltonian constructed below is correct, since the optimal solutions have the lowest energy values.

6.2.1 Ising Hamiltonian with DOcplex model without modifications

In the first experiment set, the model was constructed without setting variables A, B, C (which means that these values were set to 1) (see section 5.2) and then passed to DOcplex to generate Hamiltonian (see section 5.4). All eigenvalues, received with classical solver [21] for this model are presented in figure 6.2.

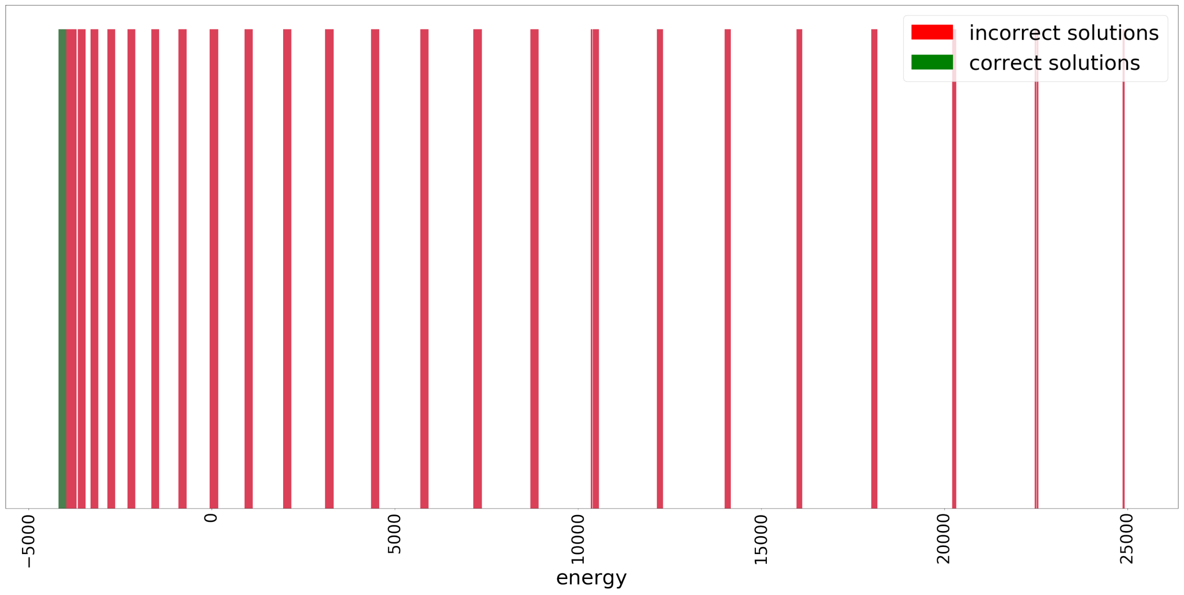

Figure 6.2: Energy for all eigenvalues, for the Ising Hamiltonian generated with DOcplex model with A, B, C were equal and set to 1. The correct values are marked with green.

After running the VQE algorithm on qasm_simulator with this Ising Hamiltonian the machine usage constraint in most cases is violated, therefore the model in the next set of experiments is slightly changed.

6.2.2 Ising Hamiltonian with DOcplex model with modifications

In the second set of experiments, the previous model was slightly modified and the value of B, corresponding to the machine usage constraint was experimentally increased to 15, the values of A and C remained unchanged (set to 1) (see section 5.2) and then passed to DOcplex to generate Ising Hamiltonian (see section 5.4). The increase of B value was obtained by simple grid search between values 1 and quintuple of maximum from maximal task execution time and maximal task execution cost. This Ising Hamiltonian has a larger range of energy values, although the values are less grouped, in comparison to the previous form. All eigenvalues obtained for this form of Ising Hamiltonian, were computed with classical solver [21] and are presented in figure 6.3.

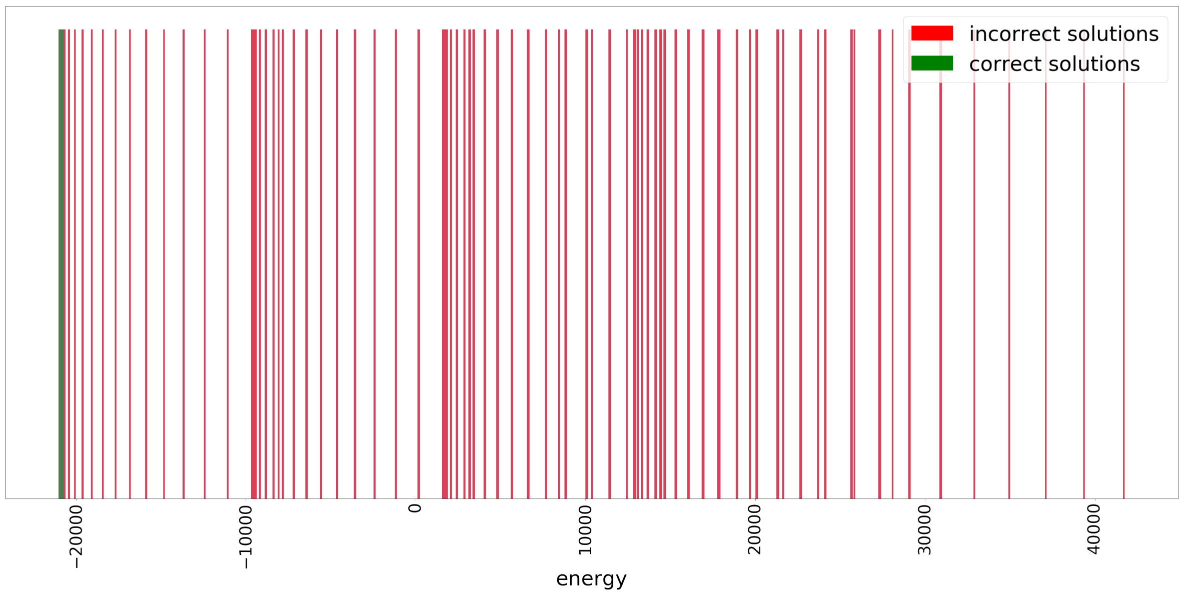

Figure 6.3: Energy for all eigenvalues for the Ising Hamiltonian created with DOcplex model with A, B, C variables set to 1, 15, 1. The correct configurations are marked with green.

6.2.3 Setting weights manually

In the third set of experiments, the Ising Hamiltonian was constructed with the method described in 5.5. In order to find the best A, B, C parameters the grid search method was used. The lower bound was set to 1 and the upper bound was set to quintuple of maximum from maximal task execution time and maximal task execution cost. In order to find the best A, B, C values, first, the energies for all eigenvalues were computed with the use of method NumPyEigensolver [21], and then the configurations were tested in qasm_simulator from Aer provider. The best results were achieved when setting parameters A, B, C (see section 5.2) to 1, 40, 1. All eigenvalues for constructed Ising Hamiltonian, computed with classical solver [21] are presented in 6.4.

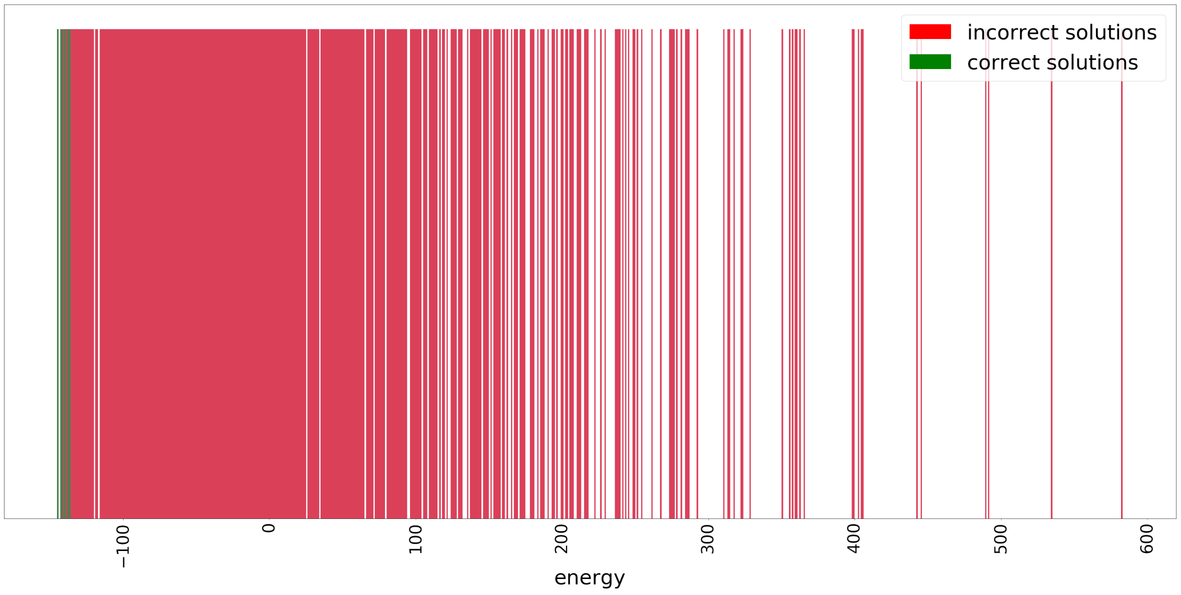

Figure 6.4: Energy for all eigenvalues, for the Ising Hamiltonian with manually setting variables A, B, C to 1, 40, 1. The correct configurations are marked with green.

6.3 Referencing methods

6.3.1 The NumPyEigensolver method

The first used reference method is the **NumPyEigensolver** [21] – classical method to find the solutions. For every previously prepared Ising Hamiltonian, this method is used to find eigenvalues of the matrix. This method uses NumPy [20] library to find all eigenvalues. Every Ising Hamiltonian created for the testing problem was run with this deterministic method and the optimal result was returned for every form.

6.3.2 Results from state vector simulation

For the second reference method, the statevector_simulator backend from Aer provider was used. This simulator imitates the ideal circuit simulation. The state vector simulator returns the possibilities of occurrences for all the possible states. The most frequently returned state for every Ising Hamiltonian is the optimal one, with the accuracy of numerical errors equal to 1.

6.3.3 Bruteforce method

The third reference method involves checking all possible configurations. Since the problem is not big it is possible to test and check the time and cost for every configuration. There are $2^{10} = 1024$ possible configurations, 7 of them are correct, one solution is optimal, the rest are incorrect. All correct configurations are presented in table 6.4 and marked with gray the optimal solution is marked with blue. The term *correct configs* represents the percentage of unique correct configurations that occurred in the experiment result.

Table 6.4: Table presents all configurations that meets machine usage constraint. All correct solutions for the testing problem are marked with gray. The most efficient – optimal solution is marked with blue. The $t_{i,j}$ represents execution time of performing task i on machine j.

Configuration	Result vector	Total time	Total cost
$t_{1,1}+t_{2,1}+t_{3,1}$	0001111100	7	28
$t_{1,1}+t_{2,1}+t_{3,0}$	0011100100	15	24
$t_{1,1}+t_{2,0}+t_{3,1}$	0101011010	9	27
$t_{1,1}+t_{2,0}+t_{3,0}$	0111000010	17	23
$t_{1,0}+t_{2,1}+t_{3,1}$	1000111000	11	26
$t_{1,0}+t_{2,1}+t_{3,0}$	1010100000	19	22
$t_{1,0}+t_{2,0}+t_{3,1}$	1100010110	13	25
$t_{1,0}+t_{2,0}+t_{3,0}$	111000xxxxx	21	21

6.4 Results

For each of the three Ising Hamiltonian creation methods in detail described in sections 6.2.1, 6.2.2, 6.2.3, one experiment set has been run. Every Ising Hamiltonian used to solve the testing problem was used for experiments run on simulator – qasm_simulator backend from Aer provider and on the real quantum computer – ibmq_16_melbourne backend from IBM Q provider. The simulator represents a quantum device without noise, therefore in order to find the best parameters set, first the experiments were run on the simulator and after finding the best execution parameters the experiments were run on the quantum computer. All experiments were run with SPSA optimizer and RealAmplitudes variational form with reps parameter set to 2 (described in section 4.1.3).

Simulator based experiments and all classical parts of real experiments were run on a server with 2GB RAM, 2 AMD EPYC CPU cores clocked at 2.5GHZ. The Qiskit version used for all configurations was 0.19.2.

In order to get better results, it is possible to increase the optimizer iterations, although this change increases the execution time. Therefore it is necessary to find the optimal number of iterations that yields satisfactory results in a reasonable time. It is important not to assign too many iterations, as after some point the results are getting slightly better, but the execution time increases linearly. For instance, the execution times on a simulator for the tested problem and settings presented above and maximal iterations number set to 1000 were in the order of 60 seconds, however the execution time differs for different configurations and executed problem.

It is not possible to measure execution time on a quantum computer since access to this hardware is through the IBM Q interface. The VQE algorithm is a hybrid algorithm and every quantum part run remotely is followed by the classical part run locally. Every quantum iteration requires waiting for access to a quantum computer. This time varies and is dependent on the number of jobs submitted on this specific quantum hardware.

6.4.1 Experiment set no. 1

The first set of experiments was conducted with the Ising Hamiltonian, which was created with the method described in section 6.2.1. Table 6.5 presents a summary of conducted experiments with execution parameters and received results for this Ising Hamiltonian. The histogram diagrams with received results are presented in figure 6.5.

Table 6.5: The table presents execution parameters and results for the testing problem with the use of Ising Hamiltonian generated with method described in section 6.2.1. The optimal, correct and incorrect solutions and correct configs column labels are described in section 6.3.3.

experiment number	backend	optimizer iterations	optimal solutions	correct solutions	incorrect solutions	correct configs
1.1	qasm_simulator	1000	0%	22,7%	72,3%	14,29%
1.2	qasm_simulator	2000	0%	37,3%	62,7%	14,29%
1.3	ibmq_16_melbourne	1000	0%	1,1%	98,9%	71,43%
1.4	ibmq_16_melbourne	2000	0,15%	0,83%	99,17%	85,71%

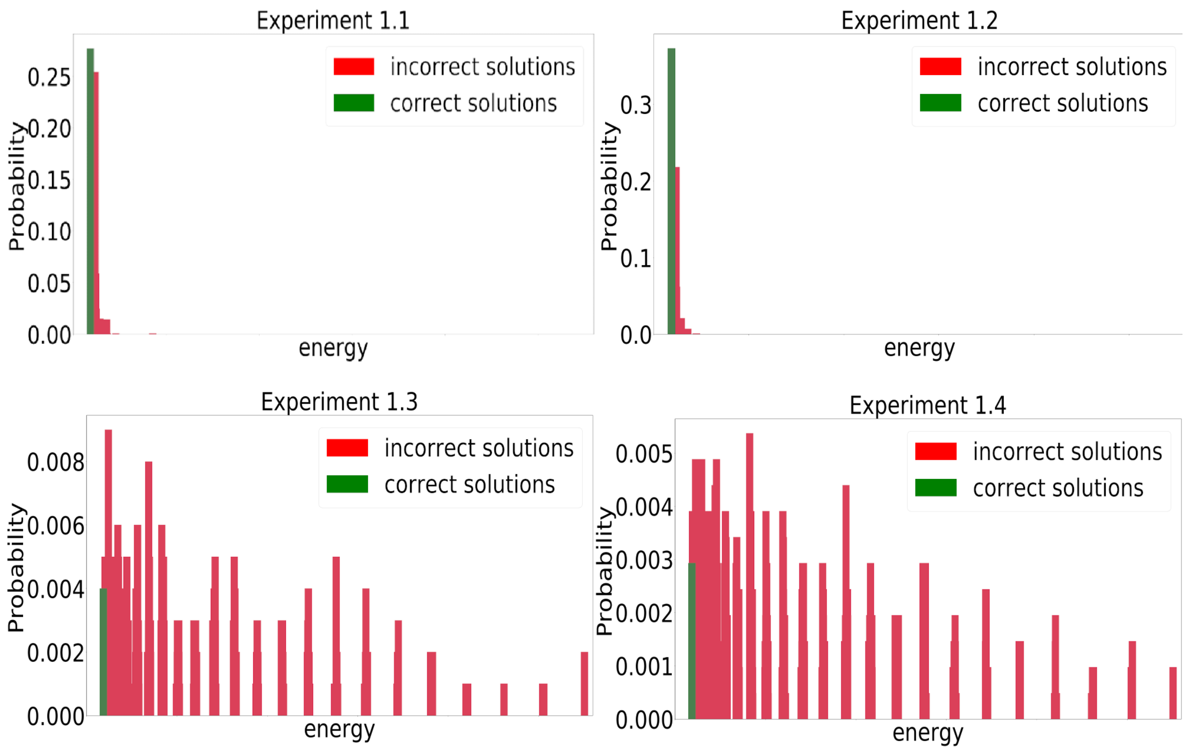

Figure 6.5: Probability density from energy diagrams for experiments described in table 6.5.

Results from simulator

Figures 6.6 and 6.7 presents exact results from the simulations that were performed on qasm_simulator. The y-axis represents the number of returned results for the specific configuration, where the total number of shots in both experiments was set to 1000. In simulation 1.1, the optimal result was not reached. In total the experiment returned 42 different configurations. 1 configuration was correct. In simulation 1.2 the optimal solution also was not reached, and the total number of different returned configurations was 28, where only one of them was correct. Table 6.6 presents three highest reached results received from experiment 1.1 and 1.2. It can be observed that these configurations are very similar, and with the increasing number of optimizer iterations, the percentage of the correct solution increases almost by 10%.

Table 6.6: Three most frequently returned configurations for testing problem from experiments 1.1 and 1.2.

result	experiment 1.1	experiment 1.2	energy	correct result
0101011010	27,7%	37,3%	-4072.5	yes
0100011010	25,4%	21,8%	-3830.5	no
0100011011	4,9%	6,3%	-3980.5	no

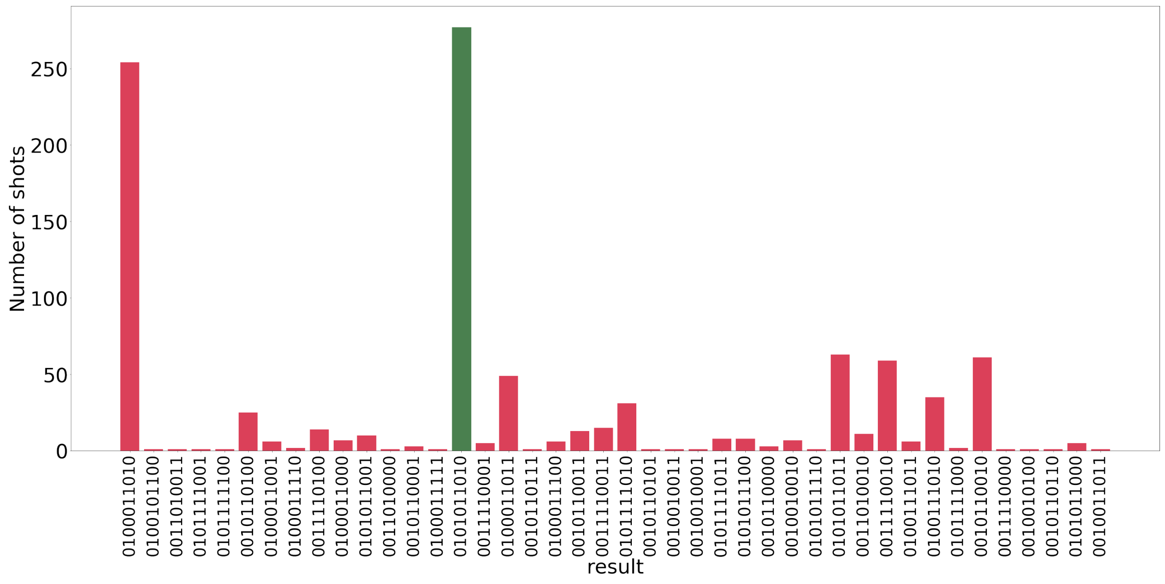

Figure 6.6: Experiment 1.1: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. The correct configurations are marked with green.

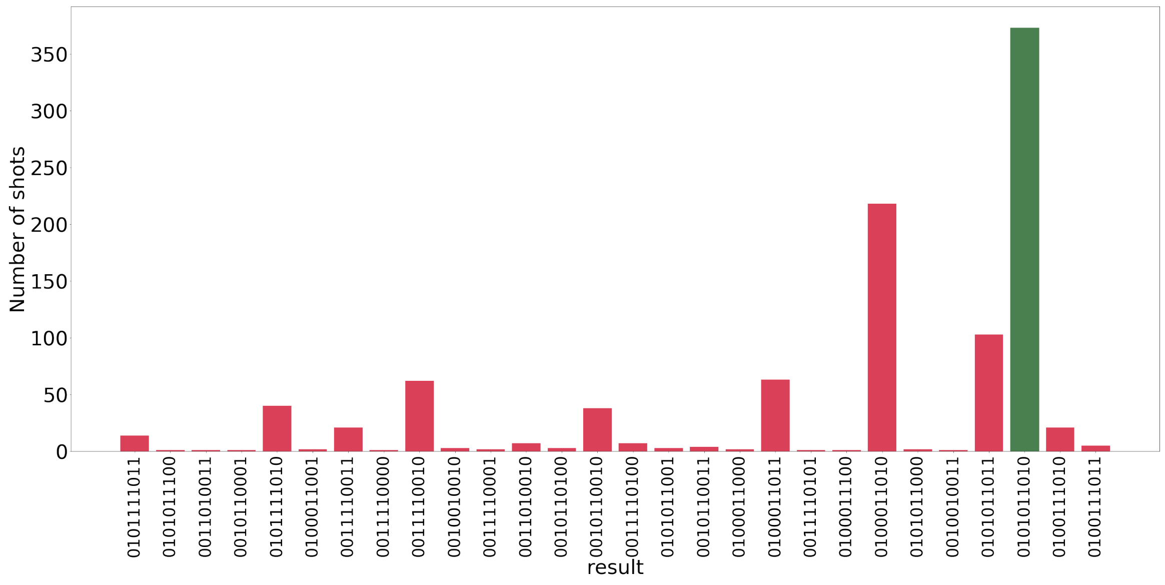

Figure 6.7: Experiment 1.2: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 2000 with shots number set to 1000. The correct configurations are marked with green.

Results from quantum computer

Figures 6.8 and 6.9 presents exact results from the experiments that were performed on ibmq_16_melbourne. The y-axis represents the number of returned results for the specific configuration, where the total number of shots in experiment 1.3 was set to 1000 and in experiment 1.4 to 2048. In experiment 1.3, the optimal result was not reached. In total, the 1.3 experiment returned 579 different configurations, 5 of them were correct. In experiment 1.4 the optimal solution was reached by 0,15% results. It returned 788 different configurations, 5 of them were correct.

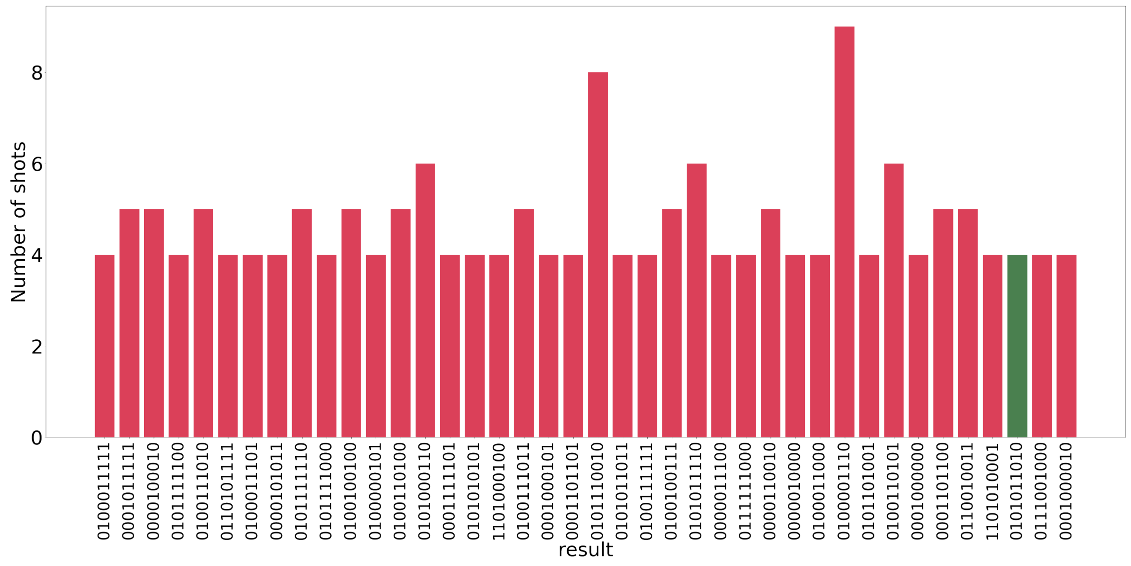

Figure 6.8: Experiment 1.3: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. Only results with shots above 3 are presented. The correct configuration is marked with green.

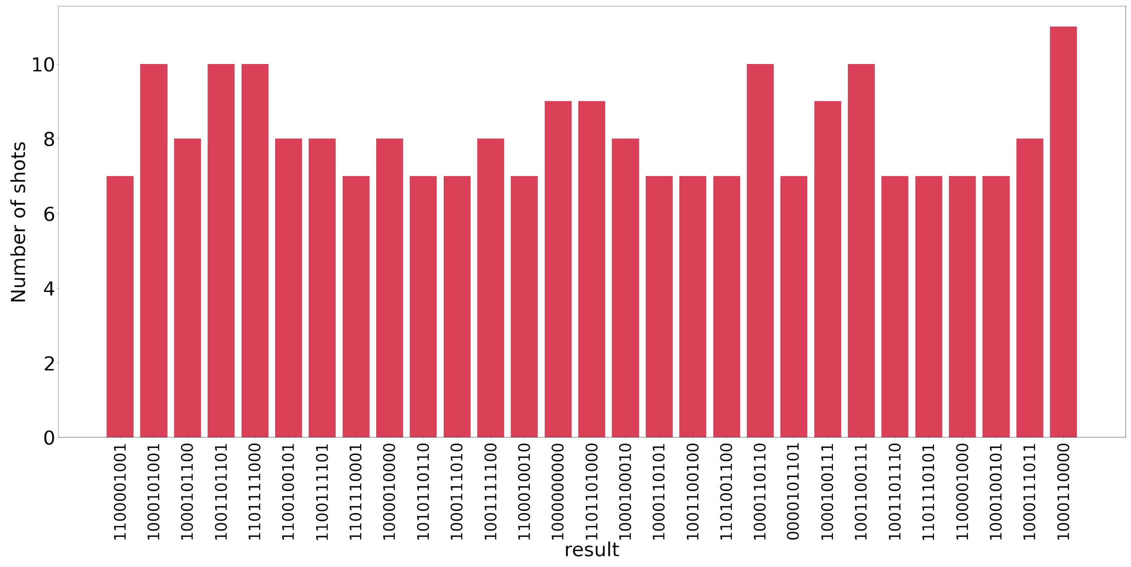

Figure 6.9: Experiment 1.4: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 2000 with shots number set to 2048. Only results with shots above 6 are presented.

Summary

Despite the increasing number of optimizer iterations on the simulator, the optimal solution was not found. The simulation returned only one correct result. After doubling the optimizer iterations on the simulator the number of correct results increased by almost 10%. The optimal solution was found on a quantum computer, but only on 0,15% of results with maximal optimizer iterations set to 2000. The percentage of received correct configurations from the quantum computer is very small – about 1%. These results are not satisfactory since the correct results in a simulator without any noise get less than 40%. Probably this happens because the strength for constraints is chosen only once, by DOcplex, and is based only on objective function values without looking at constraints values. Especially the machine usage constraint is violated, therefore in the next section, this constraint is slightly modified.

6.4.2 Experiment set no. 2

This section presents achieved results from experiments that use the improved version of Hamiltonian, which is in detail described in section 6.2.2. The improved Hamiltonian is also generated by DOcplex module, but the passed form has different constraint values. Diagrams 6.10 presents probability values from energy values for experiments described in table 6.7.

Table 6.7: The table presents execution parameters and results for the testing problem with the use of Ising Hamiltonian generated with the method described in section 6.2.2. The optimal, correct and incorrect solutions and correct configs column labels are described in section 6.3.3.

experiment number	backend	optimizer iterations	optimal solutions	correct solutions	incorrect solutions	correct configs
2.1	qasm_simulator	1000	79,9%	79,9%	20,1%	14,29%
2.2	qasm_simulator	2000	88,1%	88,1%	11,9%	14,29%
2.3	ibmq_16_melbourne	1000	0,1%	0,6%	99,4%	57,14%
2.4	ibmq_16_melbourne	2000	0%	0,7%	99,3%	57,14%

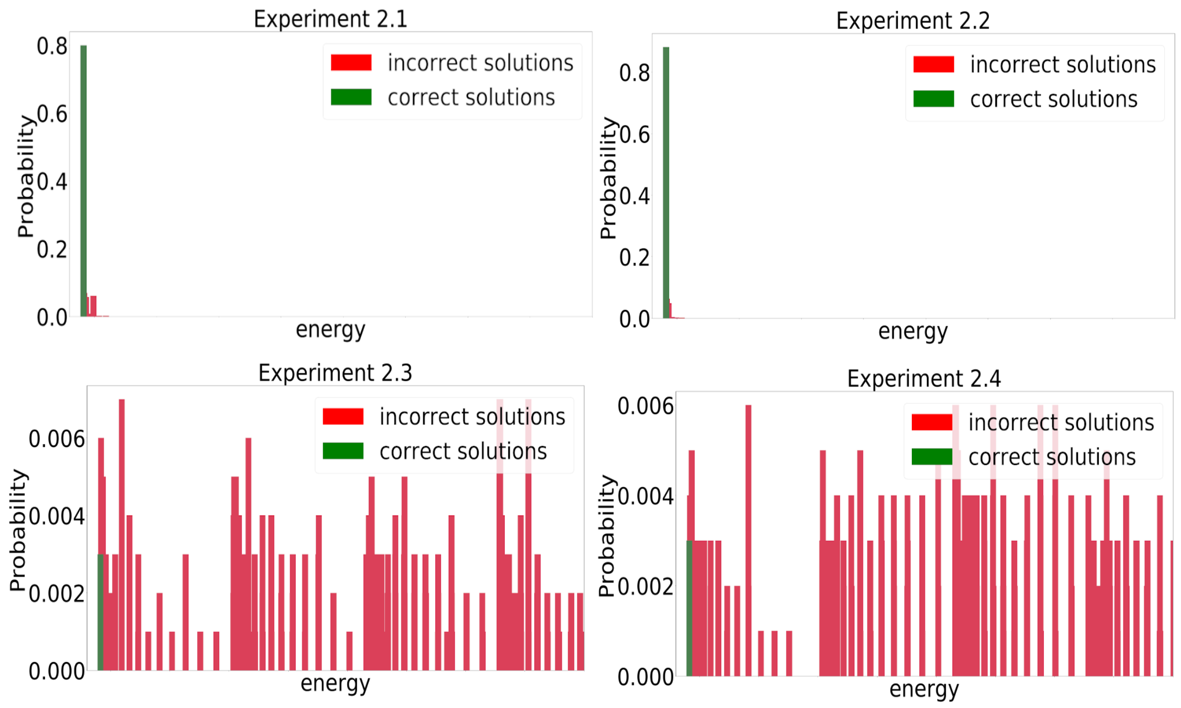

Figure 6.10: Probability density from energy diagrams for experiments described in table 6.7

Results from simulator

Figures 6.11 and 6.12 present exact results from the simulations that were performed on qasm_simulator. The y-axis represents the number of returned results for the specific configuration, where the total number of shots was set to 1000. In experiment 2.1, the correct and at the same time optimal result was returned in 79,9% of tests. In total the experiments returned 8 different configurations, only one was correct. In simulation 2.2 the correct and optimal solution got 88,1%. In total the simulation returned 6 different configurations, only one of them was correct. Table 6.6 presents 3 highest reached results received from experiment 2.1 and experiment 2.2.

Table 6.8: Three most frequently returned configurations for testing problem from experiment 2.1 and 2.2.

result	experiment 2.1	experiment 2.2	energy	correct result
1010100000	79,9%	88,1%	-20877.5	yes
1010100001	7,0%	6,3%	-20827.5	no
1010100100	6,1%	0,2%	-20077.5	no

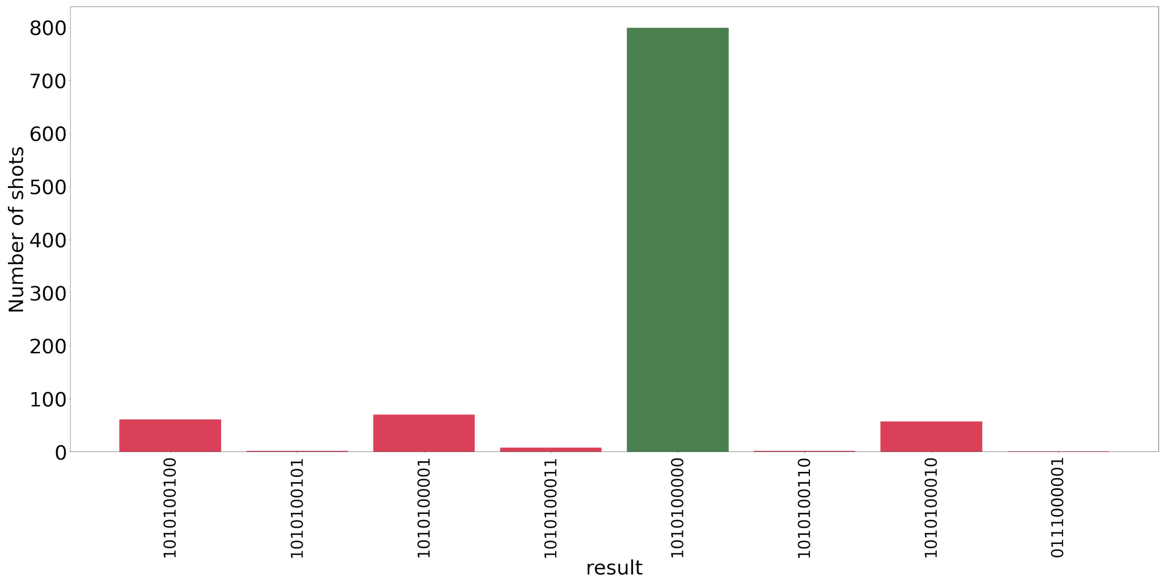

Figure 6.11: Experiment 2.1: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. The correct configurations are marked with green.

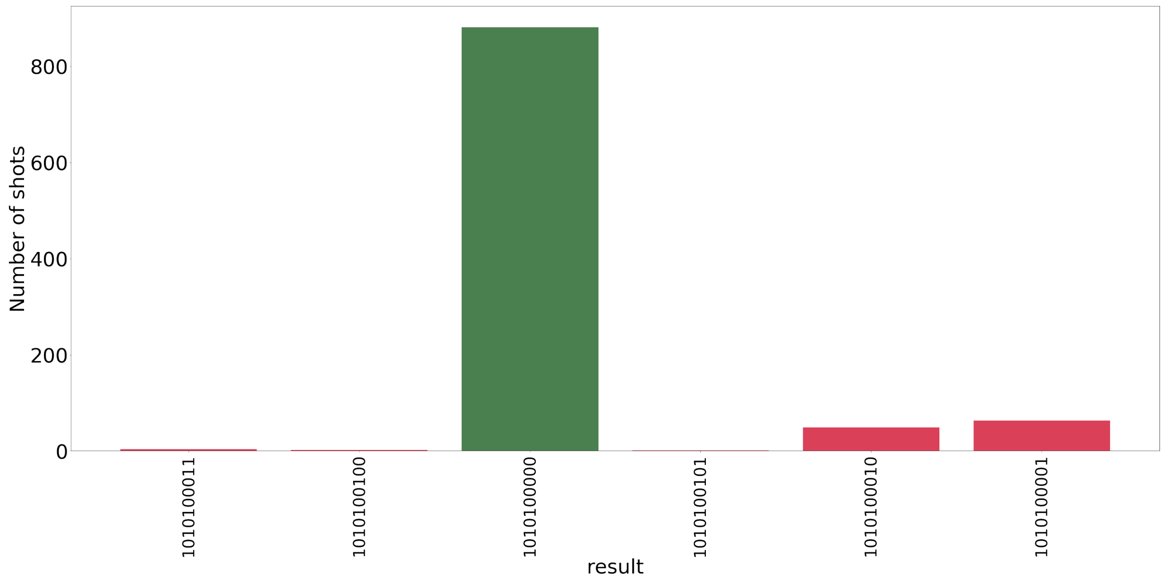

Figure 6.12: Experiment 2.2: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 2000 with shots number set to 1000. The correct configurations are marked with green.

Results from quantum computer

Figures 6.13 and 6.14 presents exact results from the experiments that were performed on ibmq_16_melbourne. The y-axis represents the number of returned results for the specific configuration, where the total number of shots in both experiments was set to 1000. Experiment 2.3 only once returned optimal result, and in total returned 583 different configurations. 4 configurations were correct. The experiment 2.4 did not return optimal configuration, in total 542 different configurations were returned, 4 of them were correct.

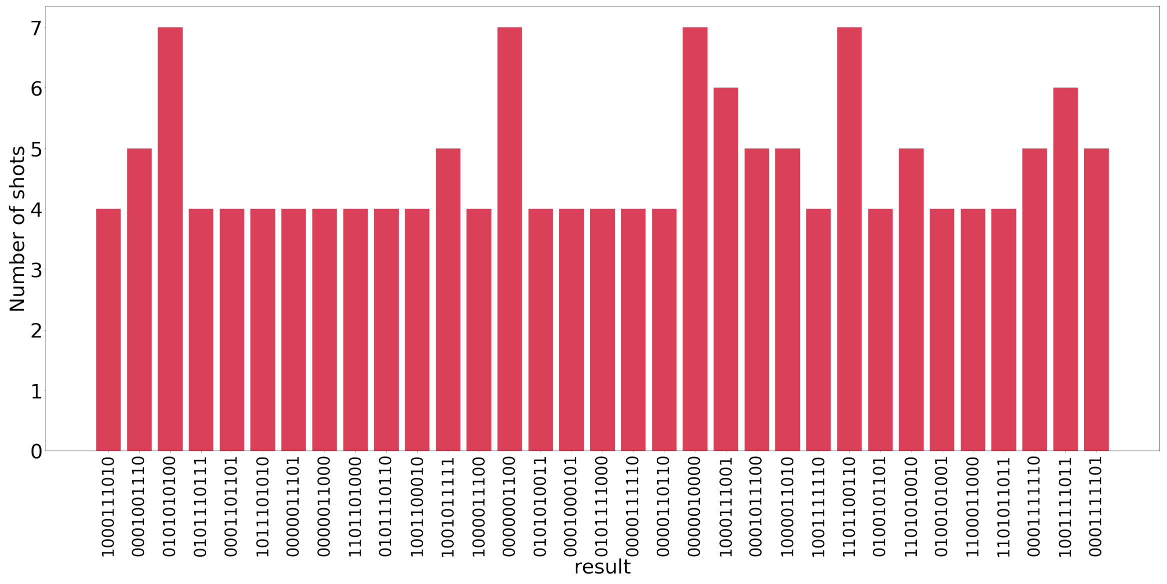

Figure 6.13: Experiment 2.3: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. Olny results with shots number above 3 are presented.

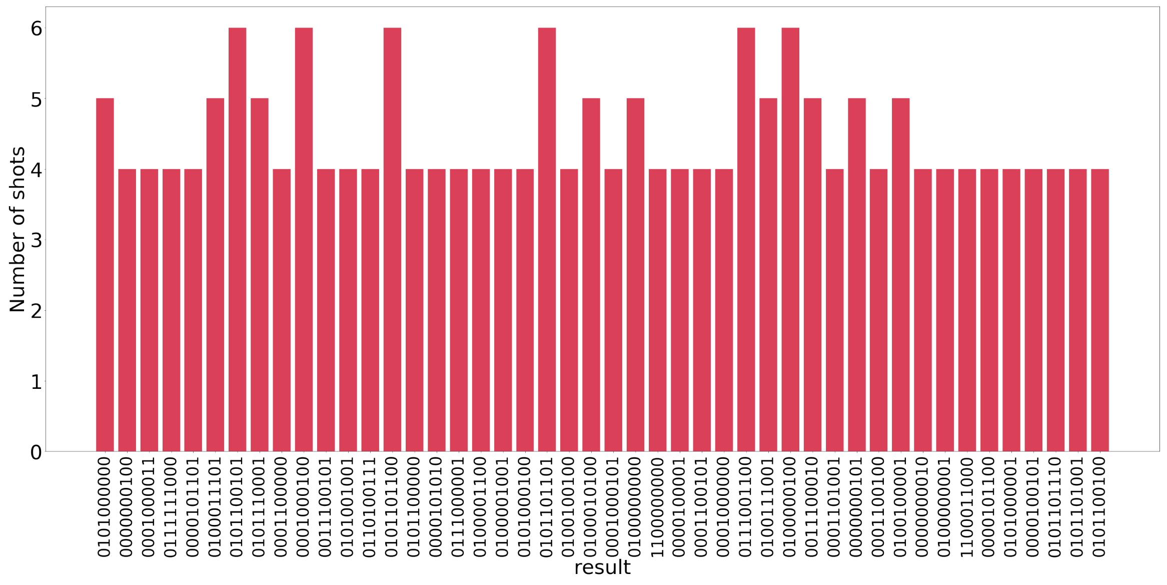

Figure 6.14: Experiment 2.4: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 2000 with shots number set to 1000. Olny results with shots number above 3 are presented.

Summary

After the change of machine usage constraint, the received results are much better. The results received from qasm_simulator got almost 80% correctness for maximal optimizer iterations set to 1000. It could be seen that for the simulator the next two highest results after the optimal one are very similar to the optimal solution as they differ only by one slack variable value. Increasing the optimizer iterations to 2000 increased the percentage of optimal solutions by 10 percent. For the quantum computer, the correct results got less than 1%.

6.4.3 Experiment set no. 3

The third set of experiments was run with the use of Ising Hamiltonian created with the method in detail presented in section 6.2.3. Table 6.9 shows a summary of performed experiments with execution parameters and received results for this Ising Hamiltonian. The histogram diagrams with received results are presented in 6.15.

Table 6.9: The table presents execution parameters and results for the testing problem with the use of Ising Hamiltonian generated with method described in section 6.2.3. The optimal, correct and incorrect solutions and correct configs column labels are described in section 6.3.3.

experiment number	backend	optimizer iterations	optimal solutions	correct solutions	incorrect solutions	correct configs
3.1	qasm_simulator	1000	97,3%	97,3%	2,7%	14,29%
3.2	qasm_simulator	2000	98,1%	98,1%	1,9%	14,29%
3.3	ibmq_16_melbourne	1000	0,1%	0,7%	99,3%	57,14%
3.4	ibmq_16_melbourne	2000	0%	0,8%	99,2%	57,14%

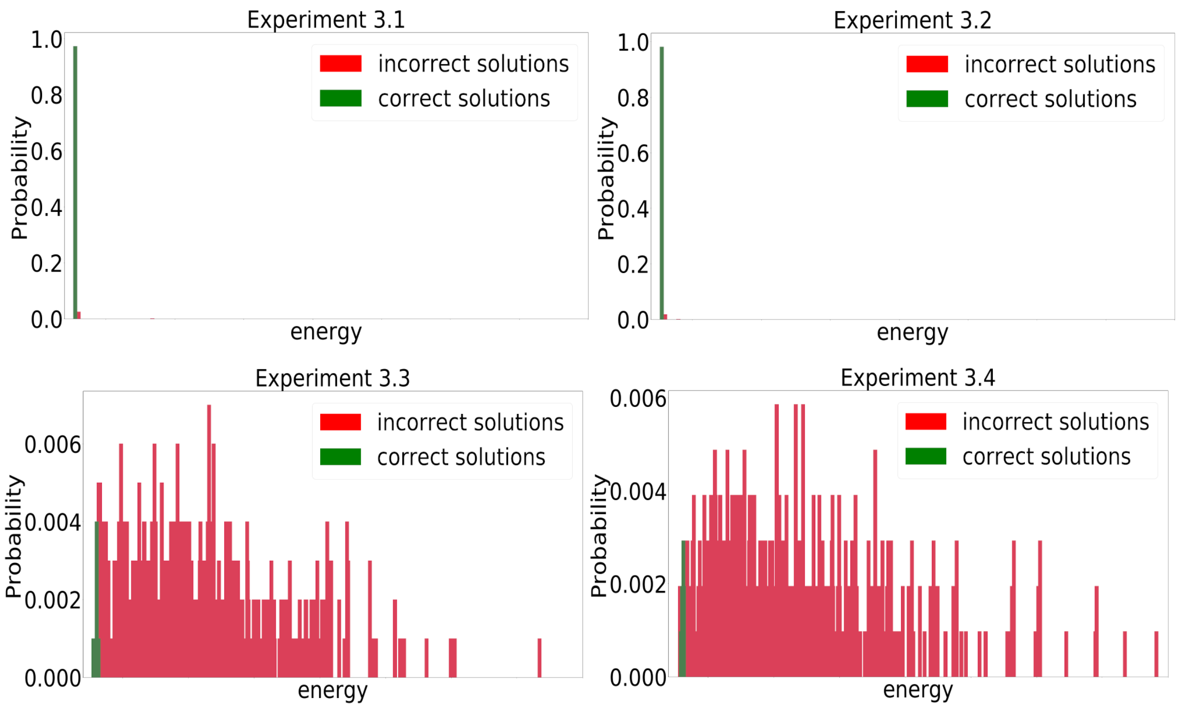

Figure 6.15: Probability density from energy diagrams for experiments described in table 6.9

Results from simulator

Figures 6.16 and 6.17 present exact results from the simulations that were performed on qasm_simulator. The y-axis represents the number of returned results for the specific configuration, where the total number of shots was set to 1000 in both experiments. In the 3.1 simulation, the correct and at the same time optimal result was reached in 97,3% of tests. In total the experiments returned 4 different configurations. 1 configuration was correct. In experiment 3.2 the optimal result was reached 98,1%. In total there were 3 configurations returned, one of them was correct. Table 6.10 presents 2 the highest reached results received from simulator.

Table 6.10: Two the most frequently returned configurations for experiment 3.1 and experiment 3.2.

result	experiment 3.1	experiment 3.2	energy	correct
1010100000	97,3%	98,1 %	-144,5	yes
1010100010	2,5%	1,76%	-139,5	no

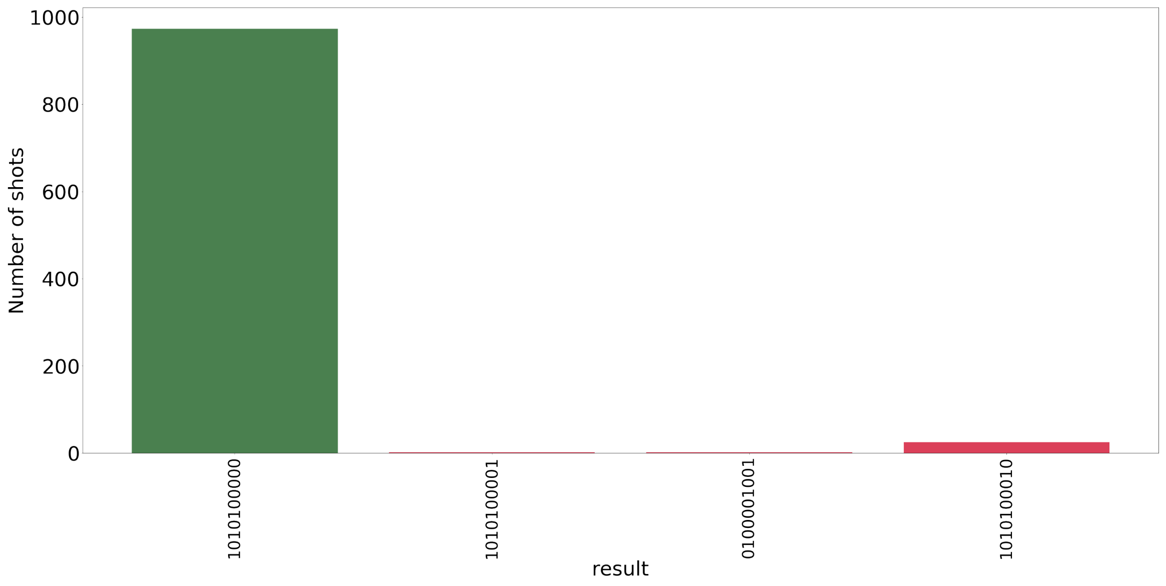

Figure 6.16: Experiment 3.1: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. The correct configuration is marked with green.

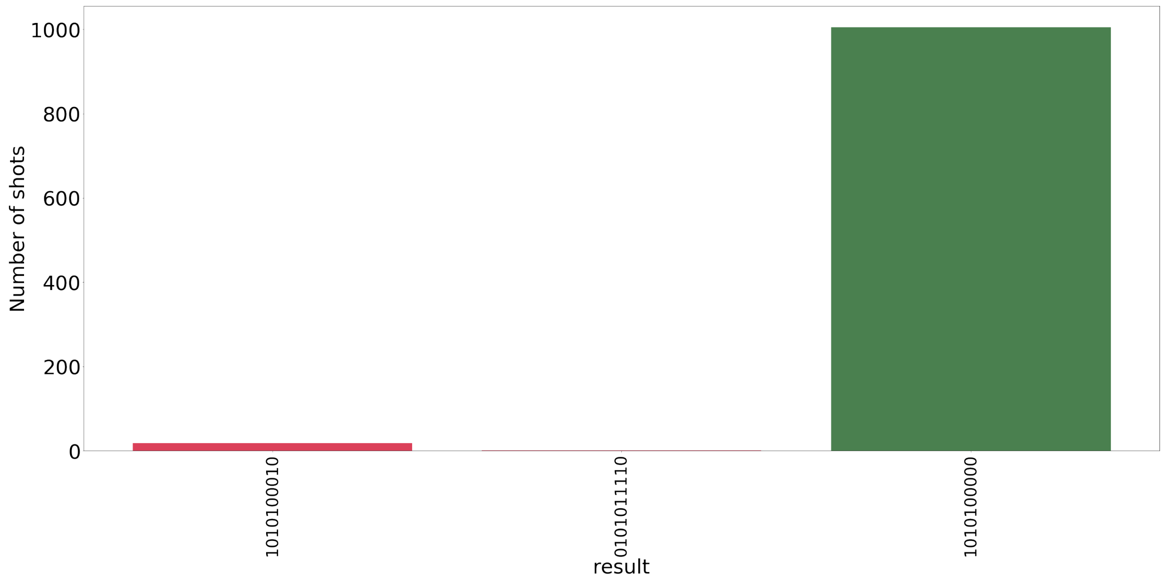

Figure 6.17: Experiment 3.2: exact results received from running the testing problem on qasm_simulator, with the maximum number of optimizer iterations set to 2000 with shots number set to 1000. The correct configuration is marked with green.

Results from quantum computer

Figures 6.18 and 6.19 present exact results from the experiments that were performed on ibmq_16_melbourne. The y-axis represents the number of returned results for the specific configuration, where the total number of shots was set to 1000. Experiment 3.3 only once returned the optimal result. In total the experiments returned 574 different configurations. 4 configurations were correct. In experiment 3.4 on ibmq_16_melbourne, the optimal solution was not reached. In total 648 different configurations were returned, of which 4 are correct.

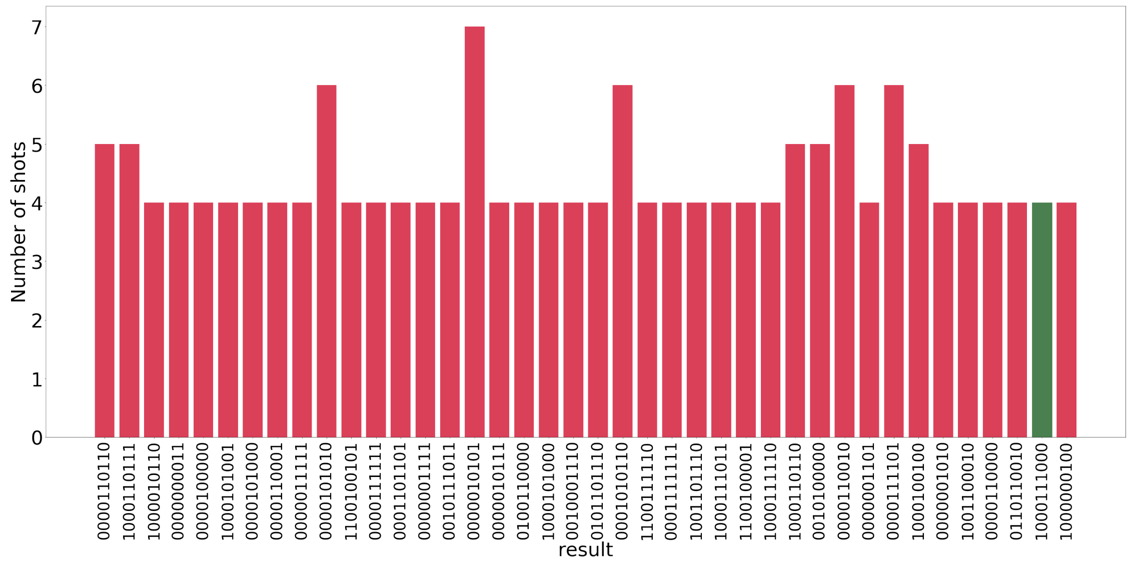

Figure 6.18: Experiment 3.3: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 1000 with shots number set to 1000. Only results with shots above 3 are presented. The correct solutions are marked with green.

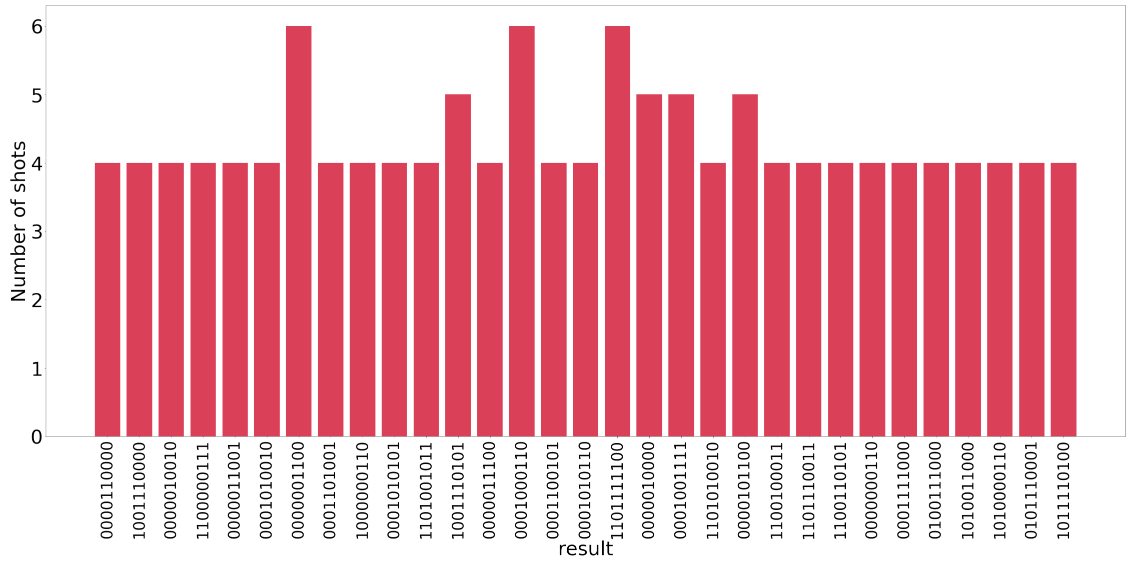

Figure 6.19: Experiment 3.4: exact results received from running the testing problem on ibmq_16_melbourne, with the maximum number of optimizer iterations set to 2000 with shots number set to 1000. Only results with shots above 3 are presented.

Summary

It can be observed, that increasing the number of iterations of the optimizer for the experiment performed on the simulator does not significantly improve the result. This indicates that the maximum number of iterations for the optimizer set to 1000 is sufficient. Getting the optimal solution at 97% on the simulator is a very good result. The small fluctuations that appear on the simulator are the effect of finite space sampling of the result vector.

Results received from the quantum computer are slightly better in comparison to results achieved in previous sections, although correct results are still less than 1% of all results. Increasing the number of optimizer iterations does not improve greatly percentage of correct results, adding 1000 interactions improved the outcome by 0.1%.

6.5 Summary

For the first set of experiments, although the created Ising Hamiltonian is correct the received results are not satisfactory since the optimal result was not reached by the noise-free simulator. The second set of experiments returned better results and for the maximal optimizer iterations set to 2000 got almost 90% optimal solutions. The Ising Hamiltonian with manually set weights returned the best results because over 97% of results from the simulator for the maximal optimizer iterations set to 1000 were optimal. This proves that using Ising Hamiltonian for solving workflow scheduling optimization problem is possible and the used method is correct. For now, results from the quantum computer are very noisy. For every Ising Hamiltonian increasing number of classical optimizer iterations does not significantly improve results. It is possible that after greatly increasing the number of this optimizer iterations the results could be better, but the number must be enormous which is impossible to test as it would require a lot of execution time.

Chapter 7

Conclusions

Quantum computers can be used to solve optimization problems. The very relevant and common problem – workflow scheduling problem was solved with the use of the Qiskit Aqua framework and run on simulators as well as the IBM quantum computer.

7.1 Achieved goals and observations

- Workflow scheduling is a complicated problem that for solving requires a complex quantum circuit, consisting of many gates and other elements. In the current generation of quantum computers, each gate introduces a bit of imperfection and noise, therefore if the circuit is very complex the imperfections add to each other and the circuit does not work correctly. Because of that running workflow scheduling on a real quantum computer yields much worse results than on a quantum circuit simulator.

- Getting correct results from simulators proves that it is possible to solve the workflow scheduling problem with the quantum approach using appropriate weights, however, quantum computers need to improve so that they are less noisy.

- The Qiskit Aqua framework is a very useful tool for working with quantum algorithms. It is easy to use and has detailed and helpful documentation.

- IBM Quantum Experience gives public access to IBM quantum computers. Thanks to its integration with the Qiskit framework it is easy to use.
- Thanks to the hybrid VQE algorithm it is possible to use quantum computers for solving many optimization problems as it decreases the number of qubits required for solving a problem and modern quantum computers have very limited qubit count.
- The DOcplex module allows its users to automatically generate Ising Hamiltonian for a given problem which is very useful for solving optimization problems and makes the whole process much easier. Results achieved from models generated by DOcplex are slightly worse than the models generated manually.

7.2 Future work

In the future, when more powerful quantum computers are available it should be possible to use them for solving workflow scheduling problems. Results that could be found in this work show, that this problem is solvable using the quantum approach, however right now it is not profitable. It may be beneficial to test if quantum computers provided by IBM Quantum Experience could be used for solving other optimization problems. They turned out to be useful but not very practical for workflow scheduling, but maybe other, simpler problems would work much better on them.

7.3 Summary

Quantum computers have big potential and may have many usages in the future. One of them is solving optimization problems. The workflow scheduling problem is solvable with a quantum approach, as it was demonstrated on a simulator, although so far not on IBM quantum computer. The ibmq_16_melbourne is very noisy and such complex problems as workflow scheduling do not work well with it.

List of Figures

List of Tables

Bibliography

[1] Arute F., Arya K., Babbush R., Bacon D., Bardin J., Barends R., Biswas R., Boixo S., Brandao F., Buell D., Burkett B., Chen Y., Chen Z., Chiaro B., Collins R., Courtney W., Dunsworth A., Farhi E., Foxen B., Martinis J.: Quantum supremacy using a programmable superconducting processor. In: *Nature*, vol. 574, pp. 505–510, 2019. URL http://dx.doi.org/10.1038/s41586-019-1666-5.

[2] Berriman B., Deelman E., Good J., Jacob J., Katz D.S., Kesselman C., Laity A., Prince T., Singh G., Su M.H.: Montage: A grid enabled engine for delivering custom science-grade mosaics on demand. In: *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5493, 2004. URL http://dx.doi.org/10.1117/12.550551.

[3] D-Wave. URL https://www.dwavesys.com/.

[4] Doaa M. Abdelkader F.O.: Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. In: *Egyptian Informatics Journal*, vol. 13(2), pp. 135 – 145, 2012. ISSN 1110-8665. URL http://dx.doi.org/https://doi.org/10.1016/j.eij.2012.04.001.

[5] DOcplex. URL https://qiskit.org/documentation/stubs/qiskit.optimization.applications.ising.docplex.html.

[6] DOcplex Model. URL http://ibmdecisionoptimization.github.io/docplex-doc/mp/docplex.mp.model.html.

[7] Dongmin K., Hariri S.: *Task-Scheduling Algorithm*, pp. 79–86. Springer US, Boston, MA, 2001. ISBN 978-1-4615-1553-1. URL `http://dx.doi.org/10.1007/978-1-4615-1553-1_7`.

[8] Foster I., Zhao Y., Raicu I., Lu S.: Cloud Computing and Grid Computing 360-Degree Compared. In: *2008 Grid Computing Environments Workshop*, pp. 1–10. 2008.

[9] Gidney C., Ekerå M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, 2019.

[10] Gisin N., Ribordy G., Tittel W., Zbinden H.: Quantum cryptography. In: *Reviews of Modern Physics*, vol. 74(1), p. 145–195, 2002. ISSN 1539-0756. URL `http://dx.doi.org/10.1103/revmodphys.74.145`.

[11] Gossett S.: 8 QUANTUM COMPUTING APPLICATIONS EXAMPLES, Far from commercially scalable but more than mere fantasy, quantum computing will one day be a transformative reality., 2019. URL `https://builtin.com/hardware/quantum-computing-applications?fbclid=IwAR3r9gaVlG1VNTsMlSNC5yjQZKVzifMIb-D8yqck9h7CIFNjxaF_cgdaqbE`.

[12] Graves R., Jordan T.H., Callaghan S., Deelman E., Field E., Juve G., Kesselman C., Maechling P., Mehta G., Milner K., Okaya D., Small P., Vahi K.: CyberShake: A Physics-Based Seismic Hazard Model for Southern California. In: *Pure and Applied Geophysics*, vol. 168(3), pp. 367–381, 2011. ISSN 1420-9136. URL `http://dx.doi.org/10.1007/s00024-010-0161-6`.

[13] Grover L.K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, 1996. URL `http://dx.doi.org/10.1145/237814.237866`.

[14] Horodecki R., Horodecki P., Horodecki M., Horodecki K.: Quantum entanglement. In: *Reviews of Modern Physics*, vol. 81(2), p. 865–942, 2009. ISSN 1539-0756. URL `http://dx.doi.org/10.1103/revmodphys.81.865`.

[15] The IBM Quantum Experience. URL `https://quantum-computing.ibm.com/`.

[16] Kennedy J., Eberhart R.: Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4. 1995.

[17] Kwok Y.K., Ahmad I.: Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. In: *ACM Comput. Surv.*, vol. 31(4), p. 406–471, 1999. ISSN 0360-0300. URL `http://dx.doi.org/10.1145/344588.344618`.

[18] Leap. URL `https://www.dwavesys.com/take-leap`.

[19] Marco Pistoia J.G.: Qiskit Aqua — A Library of Quantum Algorithms and Applications. URL `https://medium.com/qiskit/qiskit-aqua-a-library-of-quantum-algorithms-and-applications-33ecf3b36008`.

[20] NumPy. URL `https://numpy.org/`.

[21] NumPyEigensolver. URL `https://qiskit.org/documentation/stubs/qiskit.aqua.algorithms.NumPyEigensolver.html`.

[22] Peruzzo A., McClean J., Shadbolt P., Yung M.H., Zhou X.Q., Love P.J., Aspuru-Guzik A., O'Brien J.L.: A variational eigenvalue solver on a photonic quantum processor. In: *Nature Communications*, vol. 5(1), 2014. ISSN 2041-1723. URL `http://dx.doi.org/10.1038/ncomms5213`.

[23] Peruzzo A. M.J.S.P.e.a.: The Variational Quantum Eigensolver: An unsung hero of approximate quantum computing. In: , 2014.

[24] Prashant: A Study on the basics of Quantum Computing. In: , pp. 10–12, 2005.

[25] Qiskit. URL `https://qiskit.org/`.

[26] Qiskit Aqua Documnetation. URL `https://qiskit.org/documentation/apidoc/aqua/aqua.html`.

[27] Qiskit Aqua Algorithms Documentation. URL `https://qiskit.org/documentation/apidoc/aqua/algorithms/algorithms.html`.

[28] Qiskit Aqua Grover Documentation. URL https://qiskit.org/documentation/stubs/qiskit.aqua.algorithms.Grover.html.

[29] Qiskit Optimizers Documentation. URL https://qiskit.org/documentation/apidoc/qiskit.aqua.components.optimizers.html?highlight=optimizer#module-qiskit.aqua.components.optimizers.

[30] Qiskit Aqua QAOA Algorithm Documentation. URL https://qiskit.org/documentation/apidoc/aqua/algorithms/algorithms.html?highlight=qaoa#qiskit.aqua.algorithms.QAOA.

[31] Qiskit Aqua Shor Documentation. URL https://qiskit.org/documentation/stubs/qiskit.aqua.algorithms.Shor.html.

[32] Simulating Molecules using VQE, Qiskit Textbook. URL https://qiskit.org/textbook/ch-applications/vqe-molecules.html#Simulating-Molecules-using-VQE.

[33] Qiskit Aqua VQE Algorithm Documentation. URL https://qiskit.org/documentation/api/qiskit.aqua.algorithms.VQE.html.

[34] IBM 53 qubit quantum computer announcement. URL https://newsroom.ibm.com/2019-09-18-IBM-Opens-Quantum-Computation-Center-in-New-York-Brings-Worlds-Larg

[35] RealAmplitudes Documentation. URL https://qiskit.org/documentation/stubs/qiskit.circuit.library.RealAmplitudes.html.

[36] Rigetti. URL https://rigetti.com/.

[37] Samadi Y., Zbakh M., Tadonki C.: *Workflow Scheduling Issues and Techniques in Cloud Computing: A Systematic Literature Review*, pp. 241–263. 2019. ISBN 978-3-319-97718-8. URL http://dx.doi.org/10.1007/978-3-319-97719-5_16.

[38] Schlosshauer M.: Quantum decoherence. In: *Physics Reports*, vol. 831, p. 1–57, 2019. ISSN 0370-1573. URL http://dx.doi.org/10.1016/j.physrep.2019.10.001.

[39] Shaydulin R., Ushijima-Mwesigwa H., Negre C.F.A., Safro I., Mniszewski S.M., Alexeev Y.: A Hybrid Approach for Solving Optimization Problems on Small Quantum Computers. In: *Computer*, vol. 52(6), pp. 18–26, 2019.

[40] Shor P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134. 1994.

[41] Spall J.: An Overview of the Simultaneous Perturbation Method for Efficient Optimization. In: , 2001.

[42] Sutor B.: Scientists Prove a Quantum Computing Advantage over Classical. In: , 2018.

[43] Tomasiewicz D., Pawlik M., Malawski M., Rycerz K.: Foundations for Workflow Application Scheduling on D-Wave System. In: , 2020.

[44] Variational-Quantum-Eigensolver (VQE) Grove Documentation. URL `https://grove-docs.readthedocs.io/en/latest/vqe.html`.