AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Analysis of D'Wave 2000Q Applicability for Job Scheduling Problems

Analiza możliwości zastosowania komputera D'Wave 2000Q dla problemów szeregowania zadań

Autor: *Dawid Tomasiewicz*
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

Quantum computing has recently gained a lot of popularity due to the rapid development of quantum hardware, which allow already existing quantum algorithms to be implemented and tested. The phenomena used in such machines promise substantial speedup against the traditional transistor based computers. The hardware development includes both the universal quantum computers and quantum annealers. Task scheduling is a problem class, for members of which finding the quantum solution could significantly improve their execution time as many of them belong to the NP class. The goal of this work is to check the applicability of D-Wave 2000Q quantum annealer for task scheduling problems. In particular, in this thesis an abstraction over workflow scheduling problems is discussed. The author provides its transformation in a form possible to operate with the use of the quantum annealer, as well as the discussion of the experiments' results. The work shows that it is possible to use the D-Wave 2000Q for workflow scheduling problems with limitations, it also discusses the current state of the art of other task scheduling problems.

Streszczenie

Obliczenia kwantowe zyskują na popularności ze względu na szybki rozwój komputerów kwantowych, które pozwalają na implementację i testowanie istniejących kwantowych algorytmów. Zjawiska występujące w takich komputerach dają szansę na znacznie szybsze działanie niż tradycyjne komputery oparte o tranzystory. Rozwój sprzętu dotyczy zarówno uniwersalnych komputerów kwantowych jak i tych opartych o kwantowe wyżarzanie. Szeregowanie zadań to klasa problemów, dla których znalezienie kwantowego rozwiązania znacząco poprawiłoby czas wykonania, gdyż wiele z nich należy do klasy NP. Celem tej pracy jest sprawdzenie możliwości zastosowania kwantowego wyżarzacza D-Wave 2000Q dla problemów szeregowania zadań. W szczególności, w tej pracy dyskutowana jest abstrakcja problemów szeregowania zadań w aplikacji typu "workflow". Autor przedstawia jej transformację do formy wykonywalnej na kwantowym wyżarzaczu jak również dyskusję wyników eksperymentów. Praca pokazuje, że z pewnymi ograniczeniami jest możliwe zastosowanie komputera D-Wave 2000Q dla problemu szeregowania zadań w aplikacji typu "workflow" i bada obecny stan wiedzy na temat innych problemów szeregowania zadań.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, dr inż. Katarzyna Rycerz, for her invaluable support, showing the direction, in which the work could be done, continuous exchange of ideas and lots of remarks that helped me accomplish this work. I would also like to thank dr inż. Maciej Malawski and Maciej Pawlik for suggesting problems to be solved with the use of new methods, Kraków Quantum Informatics Seminar organizers for the possibility of presenting results of my work, dr Piotr Gawron for helpful remarks and Nathan Williams for proofreading. Last but not least I would like to thank my fiancée Magdalena for her continuous support and love while I was writing this thesis.

Contents

1. Introduction

This chapter provides a brief introduction to problems tackled in this thesis. Two main parts are considered: task scheduling and quantum annealing. The goals and motivation for this thesis are then stated.

1.1. Task scheduling overview

Scheduling is a process of assigning pieces of work, called tasks, to the available resources. Such mapping from tasks to resources is called the schedule. Scheduling is encountered in many domains of computer science, examples of which are:

- operation systems - all the processes request the processor time and memory, therefore an entity which assigns these resources to tasks - a scheduler - must appear in such system, the work [41] presents a lot of information in this field
- job scheduling in grid systems - system such as PL-Grid [27] need to be set in an execution order considering CPU's, memory and time usage on the resources available
- network devices - scheduling is one of the ways to ensure the proper QoS in networks

There are lots of algorithms and heuristics that can solve this problem, which include FIFO (First In - First Out - execution in order of appearance), priority scheduling (assigning a metric to each task and scheduling according to it), round robin scheduling (assigning a small amount of time to each task repeatedly until finished) and many others. The work [9] contains the detailed description of scheduling problems and solution algorithms. A standard scheduling problem answers the question: can the jobs be scheduled in any order? There is a specific group of scheduling problems, which are optimization problems as well. Algorithms for these problems not only create a schedule, but also optimize a value specified e.g. minimize makespan or cost. An example of such problem is job shop scheduling problem (JSSP) [32], which schedules a set of jobs on machines, where each job contains one or more operations which have to be executed in the correct order, the goal is to find a minimum makespan schedule.

1.2. Quantum computing overview

Quantum computing is a computational model, which relies on two phenomena that have been discovered in quantum mechanics:

- superposition - unlike in classical computational models where a bit can have only one of the two states, quantum bits (qubits) can have multiple states at once
- entanglement - the situation in which the state of the group of qubits can be described, but it is impossible to describe the state of each qubit separately

Thanks to the superposition phenomenon it is possible for the system of n qubits to process 2^n states at once. Therefore this model and it's physical implementation have theoretical potential to improve the existing solutions exponentially. The complete use of these features could result in solving NP-complete problems with the time complexity defined as BQP (Bounded-error Quantum Polynomial) [35]. Practical polynomialness of this class could make a workaround for these problems in case $P \neq NP$.

The theoretical models and algorithms on quantum computing have been developed for many years. The beginning of the gate-model related works is dated to 1981 when Richard Feynman stated that quantum physics should be simulated using quantum simulators and models rather than classical [17]. After this time works like Deutsch-Jozsa algorithm [16] or Shor algorithm [45] appeared showing the potential usability of such an approach. Also the model important for quantum annealing, Ising model, is not a new problem - it was described by Lenz in 1920 [30] and solved by Ising in 1925 [22]. However, in recent years there is a rapid growth of interest in this matter due to the development of the actual computers (e.g. IBM-Q [1], Rigetti computing [2] or D-Wave [3]) which allow to test the theoretical predictions and verify their real-world usability. There are two main types of quantum computers:

- universal quantum computers - based on quantum gates model, the model and algorithms for this type are widely described in [33]
- quantum annealing computers - minimizing the energy of the physical state [46]

This thesis focuses on the quantum annealing model as it is implemented by the D-Wave 2000Q. The discussion on the quantumness of the D-Wave 2000Q is presented in the work [44].

1.3. Motivation

The advantages of quantum phenomena based computing over classical transistor based computing gives new opportunities for solving the challenges related to scheduling. Finding a method of solving such problems with the use of quantum computers would significantly improve their execution time as many of them belong to the NP complexity class [29]. Although not all the problems from this class can be solved polynomially on quantum computers [35], there exists a class of problems called BQP (bounded-error quantum polynomial time). It is possible that there are problems from NP class that fit to BQP class as well. As the BQP class coverage is not known, it is desirable to test whether scheduling problems (many of which belong to NP class) can be successfully applied and executed for quantum

[1] https://quantum-computing.ibm.com/
[2] https://www.rigetti.com/
[3] https://www.dwavesys.com/

computers. Regardless of computing classes, unless $P = NP$ it will not be possible to solve scheduling problems quickly with the use of classical computers. Analysis of the scheduling problems applicability for quantum annealers and creating a solution for the issue constitute the reasons for preparing this work.

1.4. Goals

The main goal of this work is to examine the possibilities for solving scheduling problems with the use of the D-Wave 2000Q machine. Achieving this constitutes the following steps:

- analysis of the actual possibilities of the D-Wave quantum computer - the machine has its limitations and specifications that have to be considered when preparing the solution
- reproducing the currently available attempts for the issue
- choosing the scheduling problem type and preparing a precise definition of the problem that is actually solved in this thesis
- formulating the problem in the form possible to run on the D-Wave 2000Q quantum annealer
- running the problem on the machine, gathering results and analysis of their usefulness for real world problems

1.5. Structure of work

This work consists of 7 chapters. Chapter 2 provides the current state of the art in context of tasks scheduling and problems possible to solve with the use D-Wave 2000Q quantum annealer. Chapter 3 presents the overview of mathematical and computational definitions and processes that are used in quantum annealer computing. It also provides the necessary background for the actual solution in the next parts of the work. Chapter 4 presents the precise definition of the problem that has been chosen to solve in this thesis. Chapter 5 provides the transformation of the mentioned problem into the D-Wave 2000Q runnable form. Chapter 6 presents the results of running the transformed problem on the D-Wave 2000Q quantum annealer. In Chapter 7 the discussion of the solution is provided as well as possible future works.

2. Related work

The chapter presents the works related to the topic of this thesis. Task scheduling methods and quantum annealing works are presented. The optimization and scheduling problems solved on such machines are described in detail.

2.1. Task scheduling heuristics

Tasks scheduling is not a precisely defined one. It has many versions and applications. It usually contains tasks and machines, for which the problem is to assign tasks to machines with various conditions and requirements. One of the most popular problem representations is DAG (Directed Acyclic Graph) in which task order is described as such structure, especially as PDG (Program Dependence Graph) in which tasks and edges have their weights (for tasks it is execution time, for edges communication cost). The work [25] presents the chosen scheduling methods and compares them. The heuristics described in the paper include:

- critical path heuristics, which define weight for each vertex and edge in DAG, use them for finding the worst (potentially longest lasting) path called the critical path and trying to remove them e.g. by combining adjacent tasks into a grain
- list scheduling heuristics - tasks are formed into a priority list and executed according to it, the priority may be based e.g. on the weight of edges incident to the task
- graph decomposition methods - divides the problem graph into subgraphs with groups of tasks treated as grains and the relationships between these tasks grouped

The paper provides a wide analysis of the methods, tests them and discusses their applicability for various instance types of the scheduling problems. Another type of solution is presented in the work [36]. The problem considered is similar (scheduling and mapping of the precedence-constrained task graph to the processors). However, the solution provided is based on genetic algorithms. Two such algorithms are considered. The first of them (Critical Path Genetic Algorithm) counts two fitness functions, one of which minimizes the makespan (schedule length), whereas the second function provides the load balancing features. The second algorithm is based on task duplication (Task Duplication Genetic Algorithm). It's main idea is to allocate critical tasks redundantly to processors so that they can use the idle time slots in which the processor is waiting for the beginning of another task.

The work [8] presents a set of heuristics for solving the scheduling problem on public cloud providers as well as on private infrastructure. The authors consider lots of factors algorithmically to decide which cloud vendor can better fit the problem.
The aforementioned works as well as many others, like ant algorithm [19] or simulated annealing [18], show that the problem is commonly known and various solutions attempts have been made. It's "considered one of the most crucial NP-complete problems in the parallel and distributed computing systems" [36]. An example of scheduling problem is workflow scheduling. Workflow is a paradigm for describing and preserving complex scientific processes and applications [15] usually represented as DAG (Directed Acyclic Graph). Workflow scheduling is assigning the resource to each task from workflow so that the execution fulfils the constraints specified, further described in [51].

2.2. Workflow problems

For the purpose of this work an abstraction over workflow scheduling problems is considered as the problem solved on the D-Wave computer. Workflows themselves have become a standard for description for lots of processes and applications.[15]. A typical representation of a workflow consists of a directed acyclic graph (DAG) [31]. Such simple and standardized representation significantly increases the portability of solutions created. As soon as the problem is transformed to the workflow form all the algorithms and methods created for it can be applied. On the other hand: algorithms and methods need to be created for this specific form only, which makes their development more specialized. Examples of applications implemented as workflows are: Montage [5] - a grid enabled image mosaic service ; Software used by LIGO collaboration, created for processing data about gravitational waves [2] ; Software used for earthquake modelling in California [31].

2.3. General problems solved with the use of D-Wave 2000Q

Overview
Quantum algorithms have been developed for many years. Shor algorithm [45] is a polynomial method for solving an NP-complete problem - factorization. Grover algorithm [21] provides a method of unsorted database search in $O(\sqrt{N})$ time complexity where N is the size of a database. The work [24] provides an overview of current status of theoretical quantum algorithms. These theoretical calculations promise exponential speedup against non-quantum methods of solving similar problems. In recent years quantum computing has gained a lot of popularity due to the rapid development of quantum hardware, which includes both universal quantum computers and quantum annealers. Problems solved with the use of quantum annealers, an example of which is D-Wave 2000Q, must be provided as Quadratic Unconstrained Binary Optimization (QUBO) or Ising model solution (for further description of these forms see Section 3).
Example problems
There are lots of works discussing solving different problems with QUBO/Ising as a problem formulation and D-Wave 2000Q quantum annealer as the solving machine. The work [38] presents the translation of

the **travelling salesman problem** with time windows into QUBO. The model is complete and can be directly used as an input for the quantum annealer. However, the problem is not executed on such a machine, there are no results of using the QUBO model presented there in practice. In the work [12] another NP-hard problem - finding a **maximum clique** in a graph, which is one of the most fundamental problems that belong to this class. The authors provide the translation of the problem into QUBO. Different sizes of problems are considered: sizes under 45 vertices that can fit directly into D-Wave2X and larger problems that need to be decomposed in order to be run on the D-Wave2X machine. Gurobi and a third-party clique finding heuristics is used for a comparison of the quantum annealer's performance with the known classical methods. The paper shows that for the general case there is no quantum speedup for this problem, however for instances that fit well the D-Wave machine's architecture the speedup can be noticed.

Large instances

The real world applicability of the solution method requires solving the large instances of problems, not only small proof of concepts.The work [39] is an example of such a solution. Similarly to [12] the problem is translated into QUBO and solved on the quantum annealer, but the authors also present methods of decomposition of a large instance into smaller ones so that they can be run subsequently on the D-Wave machine. Heuristics are applied including upper and lower bound, vertex and edge extraction, generally pruning subproblems that do not contribute to the solution. The same authors use similar techniques in solving the **minimum vertex cover** problem, which also belongs to the NP-hard class in the work [40].

2.4. Optimization and scheduling on D-Wave 2000Q

The actual problem solved by the D-Wave quantum annealer in a form of QUBO is an optimization problem. Therefore optimization problems are a natural application for this machine. The three different examples of such problems are presented below.

Traffic flow optimization

The paper [34] can be considered as a very important one in terms of D-Wave solutions development. Authors from Volkswagen and D-Wave Systems present a solution of a traffic flow optimization problem, which is solved with the use of iterative hybrid quantum/classical approach. The classical part consists of preprocessing data, identifying traffic congestions, determining valid routes for cars, QUBO formulation and redistribution of cars based on the results. The quantum part using the D-Wave machine is the actual optimization - finding the assignment of cars to routes so that the congestion is minimized. The iterativeness of the solution is based on the repetition of all mentioned steps until the solution meets the requirements, e.g. there is no large congestion. The authors present all the tools created by D-Wave that are necessary for problem solving. The results provided look very promising. However it must be mentioned that the graph of the problem discussed in [34] is very sparse which can significantly improve the quality of solutions returned by the quantum annealer. This can make the methods presented unusable in case of some other problems. The obstacle of limited size of the D-Wave quantum annealer is overcome by using a qbsolv library implementing the algorithm described in [6].

Satellite scheduling

The work [47] presents the attempts and results of running the satellite scheduling problem on the quantum annealer. The main goal is to create the schedule for the satellite flight so that it collects as much data as possible. The factors that must be considered are: battery level - satellite can be charged in flight (e.g. by solar panels), but data gathering is limited then and memory limitations. Using the QUBO formulation from [43] the authors perform the experiment on the D-Wave 2X system. The results point that small instances of the problem can be solved on the machine, however it is hard to assess the absolute performance and scalability of quantum annealers because of the limited size of problems that can be executed on them.

Job shop scheduling

Another work that can be considered important in terms of solving the problem stated in the topic of this thesis is the solution of the job shop scheduling problem presented in the work [50]. The QUBO formulation provided in this paper includes three dimensions of the problem: machines number, operations number and time. The high dimensionality significantly increases the size of the translated problem. The full translation is provided as well as several techniques of pruning the number of variables; the authors are aware that this size can be a crucial obstacle to overcome. The work also describes the results of running on the D-Wave machine. The results of the work are quite good. This work is the closest to the topic of this thesis as it covers the optimization and scheduling solving on quantum annealers

2.5. Summary

The works presented in this chapter show that the topic of task scheduling is widely discussed as well as it being possible to solve common problems with the use of the D-Wave 2000Q. Attempts to perform scheduling on a quantum annealer are also presented. Based on the current state of the art represented by the mentioned papers, a DAG based abstraction over workflow problems has been chosen as the problem to be solved with the use of the D-Wave 2000Q quantum annealer in this thesis. Chapter 4 states the precise definition of that problem.

3. D-Wave computing overview

This chapter provides the overview of mathematical and computational definitions and processes that are used in quantum annealer computing. It also provides the necessary background for the description of the problem solution (provided in the next chapters).

3.1. Preliminaries

This section provides a theoretical background useful in the description of phenomena and information processing related to programming the D-Wave 2000Q quantum annealer.

Hamiltonian

The Hamiltonian of a system in classical mechanics is defined to be the sum of the kinetic and potential energies expressed as a function of positions and their conjugate momenta [1], denoted as H. It's equivalent for quantum mechanics is a Hamiltonian operator denoted as $\hat{H}$. The following intuition can be connected with this operator: it transforms the state of the particular system into the energy of this system. In the classical systems it is possible to find the energy of any possible state. For the quantum systems only particular states may have their energies calculated. These particular states are called *eigenstates* and the energies for these states are called *eigenenergies*. For any state other than eigenstate the energy of the state is undefined.

Ising model

Classical Ising model [2] is a mathematical model. It consists of an infinite d-dimensional lattice of sites. There is a discrete variable defining each site's state κ_k such that $\kappa_k \in \{+1, -1\}$. An example configuration for the 4×4 part of the two dimensional lattice is presented in the Fig. 3.1. The model also includes two types of interactions. Each site j has an external field value h_j. For each pair of adjacent sites i, j there is an interaction value $J_{i,j}$. The state of the whole lattice is denoted as κ. The Ising model can be a description of the physical state in which:

- sites are magnets, the state of each site is a particular spin of the material: -1 if the spin points down, 1 otherwise

[1] http://www.nyu.edu/classes/tuckerman/stat.mech/lectures/lecture_1/node4.html
[2] http://stanford.edu/ jeffjar/statmech2/intro4.html

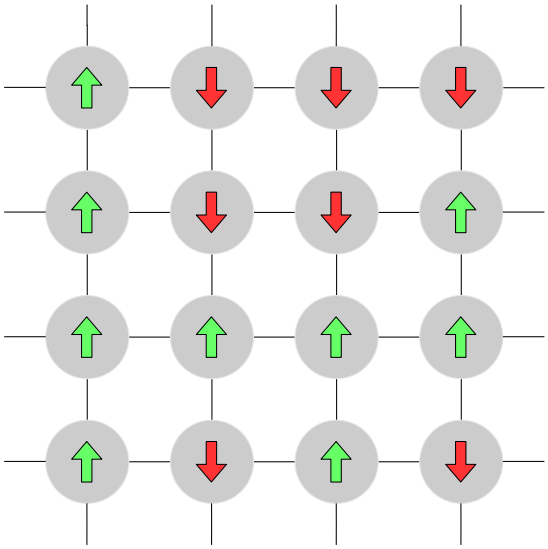

Figure 3.1: An example configuration for the 4×4 part of the two dimensional Ising model lattice

- external field is an external magnetic field applied to each magnet, it's size tells how much higher in energy one spin is than the other
- interaction is the magnetic influence of magnets that want to align or anti-align, the value of this interaction tells how strong this influence is

The Ising model energy function can be written [1] as presented in equation (3.1).

$$E(s|h,J) = \{\sum_{i=1}^{N} h_i s_i + \sum_{i<j}^{N} J_{i,j} s_i s_j\} \qquad s_i \in \{-1,+1\} \tag{3.1}$$

Quantum computing

Qubit, superposition

Qubit is a basic unit of information in quantum computing [33]. The name is a short for *quantum bit*. Qubit can be in one of two states, which are labelled $|0\rangle$ and $|1\rangle$. The fundamental difference between the classical bit and qubit is the possibility of qubit to exist not only in the two states $|0\rangle$ or $|1\rangle$ states but also in a superposition state which means that its state is unknown, it's in both states at once. When the qubit in such a state is measured it loses the superposition property and falls into one of two possible states. If the state of qubit is labelled as $|\Psi\rangle$, a superposition state is written as

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ and α^2, β^2 are probabilities of falling into $|0\rangle$ and $|1\rangle$ states respectively after measurement.

Entanglement

The qubits forming the quantum system can be entangled. For the simplest two quantum systems case, where systems are denoted as A and B it means that the values of certain properties of system A are correlated with the values that those properties will assume for system B even if the two systems are

spatially separated [33]. Thanks to such influence it is possible to bind qubits so that they are more likely to end up in the same or in opposite state.

Pauli matrices

Pauli matrices are a set of matrices commonly used in quantum computing, which are:

$$\begin{aligned} \sigma_1 = \sigma_x &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \sigma_2 = \sigma_y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ \sigma_3 = \sigma_z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned} \tag{3.2}$$

Linear Programming

The definitions in these sections are based on the work [37].

The general definition of linear programming (LP) can be written as follows: "*Given an $m \times n$ integer matrix A with rows a_i let M be the set of row indices corresponding to equality constraints and let $\bar{M}$ be those corresponding to inequality constraints. Similarly, let $x \in R^n$ and let N be the column indices corresponding to constrained variables and $\bar{N}$ those corresponding to unconstrained variables. Then an instance of the general linear programming is defined as in(3.3) (b is an m-vector of integers and c an n-vector of integers, c^T is a transposition of vector c)*" [37]

$$\begin{aligned} & min\ c^T x \\ & a_i x = b_i \quad i \in M \\ & a_i x \geq b_i \quad i \in \bar{M} \\ & x_j \geq 0 \quad j \in N \\ & x_j \neq 0 \quad j \in \bar{N} \end{aligned} \tag{3.3}$$

This definition is called *general form*. The definition from equation (3.4) is called *standard form*.

$$\begin{aligned} & min\ c^T X \\ & AX = b \\ & x \geq 0 \end{aligned} \tag{3.4}$$

Both forms are equivalent [37]. If the x values from the standard form of LP formulation are limited to be integers, such problem is called Integer Linear Programming (ILP), moreover if these values are limited to the set $\{0, 1\}$, such problem is called Binary Integer Linear Programming (BILP). Therefore the definition of BILP can be written in the standard form as provided in equation 3.5.

$$\begin{aligned} & min\ c^T X \\ & AX = b \\ & \forall x \in X\ x \in \{0, 1\} \end{aligned} \tag{3.5}$$

In this (3.5) formulation c and b are vectors, A is a matrix and X is the vector of binary variables providing a solution space.

The goal is to find the minimal value of $c^T X$ with constraints $AX = b$ for vector X as a variable.

3.2. D-Wave 2000Q as the quantum annealer

The D-Wave 2000Q computer is a quantum annealer, which means that quantum annealing is the basic process making the machine possible to work.

3.2.1. Quantum annealing

In terms of computer science annealing is described as the process of searching the solutions space with the temperature represented as probability of moving from one solution to another (simulated annealing [26]). The general idea of quantum annealing [46] is similar. However the whole process happens in the real physical environment. The temperature is the actual SI unit measured in Kelvin degrees. The particles (in metallurgical annealing) or bits (simulated annealing) are replaced by quantum bits (qubits).

Quantum phenomena

The qubits of D-Wave 2000Q machine are superconducting loops [1], which can be in two lowest-energy states (marked as "0" and "1") as well as in a superposition of the "0" and "1" states, which have higher energies. The superposition qubits is one of the features that makes this computer quantum in contrast to the traditional transistor based machines, in which bits can be in two states only. If the quantum annealing finishes without any external obstructions, the probability of falling into each of the states is 50%. However, it is possible to change it by applying the external magnetic field. The strength of such a field is described by a bias parameter. It is also possible to bind qubits with each other (to couple them). The strength of such coupling is another parameter of initial state when solving a problem. Binding two qubits with each other also leads to the second quantum phenomenon, which is entanglement. In terms of quantum annealing it can be defined as the ability to consider all the 2^N possible states (where N is number of qubits) at once during quantum annealing.

Physical background

Quantum annealing in terms of physics is a process of cooling the system of qubits so that the minimum energy of Hamiltonian describing a process is reached. For the D-Wave 2000Q an **objective function** can be defined, which expresses the problem solved. Such function corresponds with the Hamiltonian so that the eigenstate with the lowest eigenvalue is also the state with the lowest value of the objective function. For the D-Wave 2000Q quantum annealer the Hamiltonian takes the form presented in (3.6)[1]. The first part of (3.6) is an initial Hamiltonian (measuring state at the beginning of the annealing process)

$$H = -\frac{A(s)}{2}(\sum_i \hat{\sigma}_x^{(i)}) + \frac{B(s)}{2}(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}) \tag{3.6}$$

Equation 3.6: Hamiltonian for the D-Wave system, $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices, h_i are coupling biases, $J_{i,j}$ are coupling strengths. The first term of this sum is the initial Hamiltonian, the second term is the problem Hamiltonian. $A(s)$ and $B(s)$ are energy scaling functions, which evolve so that at the beginning only the initial Hamiltonian is considered, and at the end only the problem Hamiltonian.

and the second part is the final Hamiltonian (describing the actual problem). In terms of such description quantum annealing can be described as a process of moving from the initial Hamiltonian state to the

final Hamiltonian state. The solution of the problem described by the final Hamiltonian is it's minimum energy ; the goal of quantum annealing is to find this minimum energy state.

3.2.2. The quantumness of the D-Wave 2000Q

Regardless of the commercial hype for the D-Wave quantum annealers, there is also some criticism for this technology. The authors of the work [42] discuss the methods of benchmarking the quantum annealers. They also test the older D-Wave quantum annealer version showing it provides no speedup against classical machines. The work [44] provides the fully classical model that achieves excellent correlation with the theoretical model of adiabatic quantum computing and very good correlation with D-Wave annealer's results. All these results do not deny the actual quantumness, however pose an important question for further investigation of this fact.

3.2.3. Access to the D-Wave 2000Q machine

The D-Wave Systems company provides access to the quantum annealer through the cloud service [3]. A few plans for access are available: comercial, educational and free developer access (used in this thesis). It is possible to program the machine using Python language. It requires configuration [4], especially requires providing the private API token. After such configuration the quantum annealer is available with the use of D-Wave Ocean software stack. Another possibility is using the web IDE provided by D-Wave Systems.

3.3. Quadratic Unconstrained Binary Optimization (QUBO)

The second Hamiltonian in equation (3.6) is strongly related to the Ising model in physics (see Section 3.1). The form presented in equation (3.1) is by simple transformation ($s = 2x - 1$) isomorphic with the form (3.7). The Ising model uses values $\{-1, 1\}$, which is related to the physical phenomena it describes (e.g. spin values) whereas QUBO uses the common in computing binary values $\{0, 1\}$. There-

$$f(x) = \sum_i Q_{i,i}x_i + \sum_{i<j} Q_{i,j}x_ix_j \qquad x_i \in \{0, 1\} \tag{3.7}$$

Equation 3.7: QUBO formulation, X is a vector representing the solutions space, Q is an upper-diagonal $|X| \times |X|$ matrix

fore it is possible to write the problem Hamiltonian as the form (3.7) and minimization it's energy can be written as Quadratic Unconstrained Binary Optimization problem which is presented in equation (3.8) or in the simpler matrix and vector multiplication form (3.9)

$$\min_{x\in\{0,1\}^N} f(x) = \sum_i Q_{i,i}x_i + \sum_{i<j} Q_{i,j}x_ix_j \tag{3.8}$$

[3] https://www.dwavesys.com/take-leap

[4] https://docs.ocean.dwavesys.com/en/stable/overview/sapi.html

$$\min_{x \in \{0,1\}^N} f(x) = x^T Q x \tag{3.9}$$

Such equivalence gives the possibility to solve the problems with the use of quantum annealer by stating the actual problems as QUBO, however this fact is restricted by the machine's architecture (see Section 3.5). It is worth mentioning that the only input necessary to run on the quantum annealer is the Q matrix (later also called the QUBO matrix). The diagonal values are the bias values for each qubit, and the non-diagonal values (upper triangular) are couplings values between qubits. As a matrix is a natural graph description, QUBO matrix (which is a square matrix) can also be described as QUBO graph, where:

- each vertex represents an index of the column/row in the QUBO matrix, a vertex weight is a diagonal value for this index ($Q_{i,i}$ for index i), in terms of quantum annealing it is a bias value
- edges represent non-diagonal values in a matrix, the edge weight between indices i and j is a $Q_{i,j}$ value in matrix, in quantum annealing it is a coupling strength between two qubits

In the general case the QUBO matrix is not triangular. QUBO matrices created for the purpose of quantum annealing should be upper triangular because the quantum hardware architecture is an undirected graph. However, it is possible to translate any QUBO matrix into an upper triangular matrix without loss of generality by adding values under the main diagonal to the respective values above the main diagonal and then zeroing everything under the main diagonal [20]. An example transformation of a non-triangular QUBO matrix into the undirected graph is presented in the Fig. 3.2.

	0	**1**	**2**	**3**
0	5	10	0	7
1	2	8	5	2
2	0	1	1	4
3	9	10	4	3

	0	**1**	**2**	**3**
0	5	12	0	16
1	0	8	6	12
2	0	0	1	8
3	0	0	0	3

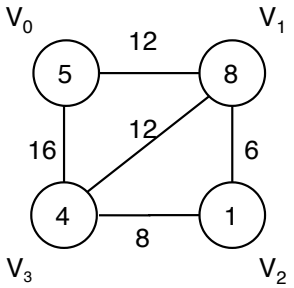

Figure 3.2: An example transformation of a non-triangular QUBO matrix (the leftmost item) into a triangular matrix (the middle item) and finally into an undirected graph (the rightmost item)

It is important to notice that for binary variables $x^2 = x$.

3.4. Problems' transformation to QUBO

3.4.1. Constraints in QUBO

In terms of stating problems as QUBO there are two important classes of optimization problems necessary to be considered:

- unconstrained - simple optimization with the objective function defined and no further conditions

Constraint	Penalty formulation
$x + y \leq 1$	$P \cdot (x \cdot y)$
$x + y \geq 1$	$P \cdot (1 - x - y - xy)$
$x + y = 1$	$P \cdot (1 - x - y + 2xy)$
$x \leq y$	$P \cdot (x - xy)$
$x = y$	$P \cdot (x + y - 2xy)$

Table 3.1: Table of a few known penalties transformation from the form of inequality appearing in lots of problems to the form of $P \cdot f(x_1, x_2...x_n)$ form perfectly suitable for QUBO. These penalties are designed so that the lowest value is equal zero and the constraint is fulfilled if and only if this zero value is reached (P is a positive scalar).

- constrained - solutions must be optimal for the objective function, but they also have to fulfil the predefined constraints

QUBO (Quadratic **Unconstrained** Binary Optimization) is unconstrained by definition. Therefore the unconstrained problems can usually be easier transformed into QUBO as they do not require additional factors included into it. For the constrained problems the constraints must be transformed into the unconstrained form and put into QUBO so that the original optimization problem is preserved. For this purpose the concept of **penalty** is created. Including constraint into QUBO requires preparing QUBO so that the objective function minimized in this problem has low values only for solutions fulfilling constraints. Solutions breaking constraints should have higher values of the objective function.

3.4.2. General single penalty formulation

It is possible to write the constraints for the following problem [20]:

$$min\ y = f(X)\ subject\ to\ the\ constraint.$$

The example constraints are presented in Tab. 3.1. If the penalty function is marked as $e(X)$, the minimization problem can be written as

$$min\ f(x) + P \cdot e(X).$$

For example the problem

$$min\ 2x - y\ such\ that\ x \leq y$$

can be written as

$$min\ 2x - y + P \cdot (x - xy) \quad \Longleftrightarrow \quad min\ x \cdot (2 + P) + y \cdot (-1) + xy \cdot (-P)).$$

The last form is a QUBO formulation possible to write as 2×2 matrix presented in the equation (3.10).

$$Q = \begin{bmatrix} 2 + P & -P \\ 0 & -1 \end{bmatrix} \tag{3.10}$$

The proper value of parameter P is required for this formulation. It does not result from any previous calculations. The P value is a penalty strength coefficient that must be tuned for each QUBO to set up the proper strength of penalty.

3.4.3. Example problem transformations

Many problems have already been transformed into QUBO. This section presents a few examples of problems with their transformation to QUBO based on the work [20].

XNOR

One of the simplest constrained problems possible to map into QUBO is XNOR. It can be defined in the following way: *Given two binary variables x_0 and x_1 provide the solutions, in which $x = y$ constraint is fulfilled.* The objective function that is minimized in the QUBO problem for two binary variables takes the form (3.11).

$$f(X) = f(x_0, x_1) = Q_{0,0} \cdot x_0 + Q_{1,1} \cdot x_1 + Q_{0,1} \cdot x_0 x_1 \tag{3.11}$$

Coefficients $Q_{a,b}$ are elements of the QUBO matrix necessary as an input for the quantum annealer. As $f(0,0) = 0$ and the solution $[0,0]$ is correct, the energy of correct solutions in this problem should be equal to 0. Solutions $[0,1]$ and $[1,0]$ should be penalized equally, $f(1,0) = Q_{0,0}$, $f(0,1) = Q_{1,1}$. Let's introduce a positive real number k and assume that $f(1,0) = f(1,0) = k$. The $f(1,1)$ value should equal 0 as this is the correct solution such as $(0,0)$. As $f(1,1) = Q_{0,0} + Q_{1,1} + Q_{0,1} = 2k + Q_{0,1} = 0$, therefore $Q_{0,1} = -2k$. For the purpose of the QUBO formulation k can be any positive real number. For the purpose of running the QUBO on the quantum annealer the values should not extend the maximum range for the diagonal values and the non-diagonal values (so called h values and J values respetively), however it is not important for the developer as D-Wave software scales the problem automatically if the values extend or don't reach the maximum ranges possible. Let's assume that $k = 0.5$. In this case the objective function (3.12) can be also presented as a matrix (3.13).

$$f(x_0, x_1) = 0.5x_0 + 0.5x_1 - x_0 x_1 \tag{3.12}$$

$$Q = \begin{bmatrix} 0.5 & 1 \\ 0 & 0.5 \end{bmatrix} \tag{3.13}$$

The Number Partitioning Problem

The number partitioning problem is to find a partition of a set of numbers into two subsets so that the sums of these subsets are as close as possible. For the set of numbers $S = \{s_1, s_2, ...s_m\}$ let $x_j = 1$ if S_j is assigned to subset 1; 0 otherwise. Then the sum for subset 1 is given by $sum_1 = \sum_{j=1}^{m} s_j xj$ and the sum for subset 2 is given by $sum_2 = \sum_{j=1}^{m} s_j - \sum_{j=1}^{m} s_j x_j$. The difference of these sums is then:

$$diff = \sum_{j=1}^{m} s_j - 2\sum_{j=1}^{m} s_j x_j = c - 2\sum_{j=1}^{m} s_j x_j$$

The minimization of such difference can be achieved by minimizing it's square (so that it does not matter which subset sum is larger, note: $x_j^2 = x_j asx \in \{0,1\}$):

$$diff^2 = \left(c - 2\sum_{j=1}^{m} s_j x_j\right)^2 = c^2 - 4c\sum_{j=1}^{m} s_j x_j + 4\sum_{j=1}^{m} s_j^2 x_j^2 + 4\sum_{i=1}^{m}\sum_{j=1}^{m} s_i s_j x_i x_j$$

$$= c^2 + 4\sum_{j=1}^{m}(cs_j x_j + s_j^2 x_j) + 4\sum_{i<j}^{m} s_i s_j x_i x_j$$

$$= c^2 + 4\sum_{j=1}^{m} x_j(s_j(s_j - c)) + 4\sum_{i<j}^{m} s_i s_j x_i x_j$$

If the QUBO matrix coefficients are written as $q_{i,i} = s_i(s_i - c)$ and $q_{i,j} = q_{j,i} = s_i s_j$, the above equation can be written as:

$$c^2 + 4\left(\sum_{i=1}^{m} q_{i,i} x_i + \sum_{i<j} q_{i,j} x_i x_j\right) = c^2 + 4x^T Q x$$

As the optimization problem is considered, the constants c and 4 can be dropped and the exact $x^T Q x$ form of QUBO is received.

Binary Integer Linear Programming

Binary Integer Linear Programming (BILP) problem is the commonly known linear programming problem, formulated and described in Section 3.1. It can be translated with the use of matrix transformations in the following way [20] (c is an additive constant, it can be dropped):

$$y = x^T Cx + P(Ax - b)^T(Ax - b) = x^T Cx + x^T Dx + const = x^T Qx + const \tag{3.14}$$

This transformation is used in this thesis as a part of transformation of the workflow problem into QUBO.

Job shop scheduling problem

The job shop scheduling problem (JSSP) is the most similar problem to what is solved in this thesis that has already been solved with the use of the D-Wave quantum annealer. The work [50] presents the solution as well as results for this problem.

Problem statement

Two main objects in the job shop scheduling problem definition are jobs and machines. Each job has an ordered list of one or many operations. Each operation has it's execution time assigned (zero or more). Each machine can run zero or one operation at the given point of time. The order of operations inside the job must be followed (the operation must not start until the preceding operation finishes). Each operation starts only once in the whole process. A machine can only run one process at one moment in time. The goal of JSSP is to create a schedule for operations and machines so that it matches the aforementioned constraints and the total execution time of such a schedule (called makespan) is the lowest. An example schedule for the job shop scheduling problem is presented in the Fig. 3.3

JSSP transformation to QUBO

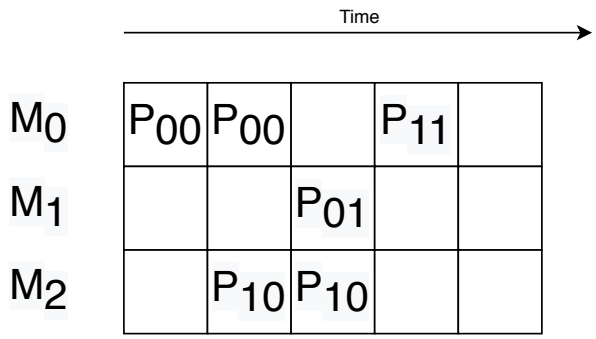

Figure 3.3: An example schedule for job shop problem with three machines (M_0, M_1, M_2), two jobs containing two operations each ($J_0 = [P_{0,0}, P_{0,1}]$, $J_1 = [P_{1,0}, P_{1,1})$

The operations are collected into one list and numbered from 1 to N_{op}. The time is discrete. The binary variables formulation is presented in the equation (3.15).

$$x_{i,t} = \begin{cases} 1 & \text{operation } O_i \text{ starts at time t,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

The constraints: starting operations once and only once (1), one job running at one machine at the given point of time (2) and operations order (3) are transformed in the way presented in the next paragraphs.

Starting operations only once

Constraint in the presented formulation is described in the equation (3.16).

$$\left(\sum_t x_{i,t} = 1 \text{ for each } i \right) \tag{3.16}$$

The equation (3.17) presents the penalty function for QUBO resulting from this constraint.

$$\sum_i \left(\sum_t x_{i,t} - 1 \right)^2 \tag{3.17}$$

One job at one machine

The constraint is presented in equation (3.18)

$$\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} = 0 \text{ for each } m, \tag{3.18}$$

where $R_m = A_m \bigcup B_m$ and

$$\begin{aligned} &A_m = (i,t,k,t') : (i,k) \in I_m \times I_m, \\ &i \neq k, 0 \leq t, t' \leq T, 0 < t' - t < p_i \\ &B_m = (i,t,k,t') : (i,k) \in I_m \times I_m, \\ &i < k, t' = t, p_i > 0, p_j > 0, \end{aligned} \tag{3.19}$$

which results in the penalty in equation (3.20)

$$\sum_m \left(\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} \right), \tag{3.20}$$

Operations order

The order of operations is enforced by the following constraint:

$$\sum_{\substack{k_{n-1}<i<k_n \\ t+p_i>t'}} x_{i,t}x_{i+1,t'} \text{ for each } n, \tag{3.21}$$

which leads to penalty:

$$\sum_n \left(\sum_{\substack{k_{n-1}<i<k_n \\ t+p_i>t'}} x_{i,t}x_{i+1,t'} \right). \tag{3.22}$$

3.5. Minor embedding

Note: for the purpose of this section terms "QUBO matrix" and "QUBO graph" will be used interchangeably, where "QUBO graph" is a graph, representation of which is a QUBO matrix.

The problem represented by a QUBO matrix can only be run on the quantum annealer if the QUBO graph is a minor of the hardware graph (which defines connectivity between physical qubits in graph). It is possible if there exists such a minor. For most QUBO matrices it is not true. Therefore it is necessary to connect qubits of the annealer into chains so that they represent one value in the QUBO matrix.

3.5.1. D-Wave 2000Q architecture

The qubits of the D-Wave 2000Q quantum annealer are organized in the graph pattern called Chimera. The graph consists of a lattice of cells. The single cell, which is a K_4^4 graph (complete bipartite graph with four qubits in each part), presented in Fig. 3.4. Fig. 3.5 presents the 3×3 Chimera

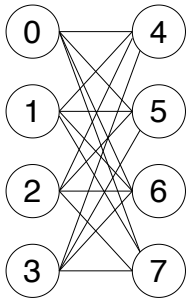

Figure 3.4: A single cell of the Chimera graph

lattice. The actual D-Wave 2000Q architecture is a 16×16 lattice created based on this pattern. The properties of this graph that are worth emphasizing in context of programming the machine are:

- degree value of the graph, which is 6 - it limits the number of non-zero values in each row of QUBO matrix to 6 if the programming is done without chains
- lack of cycles of odd length - graphs with such cycles cannot be embedded in the architecture (without chains)

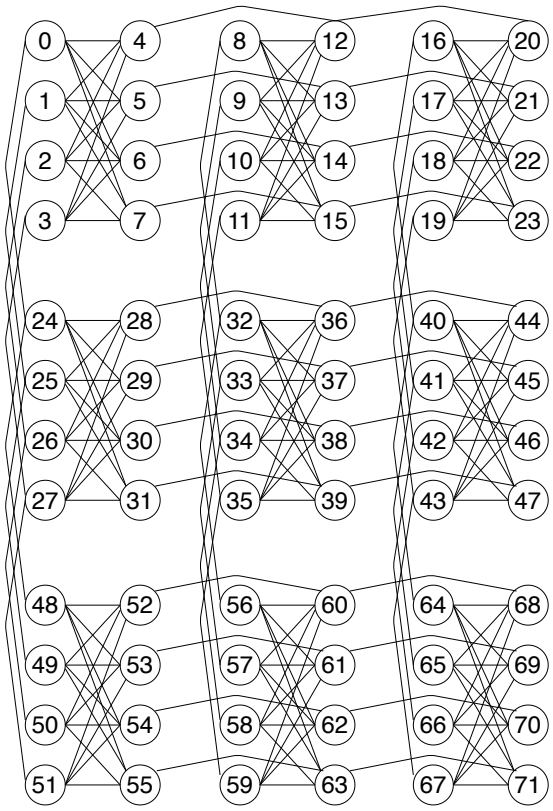

Figure 3.5: A 3×3 Chimera lattice

It is worth mentioning that the new architecture - Pegasus - is to be released in 2020. It has degree 15 and contains 5640 qubits. It will significantly improve the performance for the existing problems, however the programming flow and general problems stay the same regardless of the architecture.

3.5.2. Connecting qubits using chains

Because of the limitations mentioned lots of graphs cannot be directly embedded into the Chimera graph (they are not Chimera's minor). In order to overcome this obstacle, qubits of the quantum annealer are connected into chains (sets of physical qubits, which represent the same value in the original problem). Therefore to minor embed the QUBO is to add a new equality constraint to it and apply it as penalty into the new QUBO instance (as described in Sections 3.4.1, 3.4.2).

A very simple example of a directly not embeddable graph is the triangle: graph with three vertices and three edges creating one cycle of length three. It is presented in the Fig. 3.6 As there is no such minor in the Chimera graph, it is necessary to treat two vertices of it as one vertex of a triangle. Fig. 3.7 presents a simple embedding of such triangle into a part of the Chimera cell (numbers of qubits in the Chimera cell are coherent with Fig. 3.4). Qubits "0" and "5" of Chimera represent the same qubit "0" of the triangle. It is obvious to achieve with the constrained model (state that $q_0 = q_5$, see Section 3.4.1), however in the QUBO model, which is unconstrained, it is necessary to introduce a new parameter: chain strength. It is the coupling strength between two connected physical qubits, that has to be strong enough to overcome the couplings between the two logical qubits, but weak enough not to let the annealer ignore

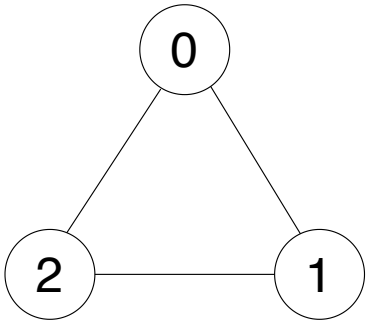

Figure 3.6: A triangle graph

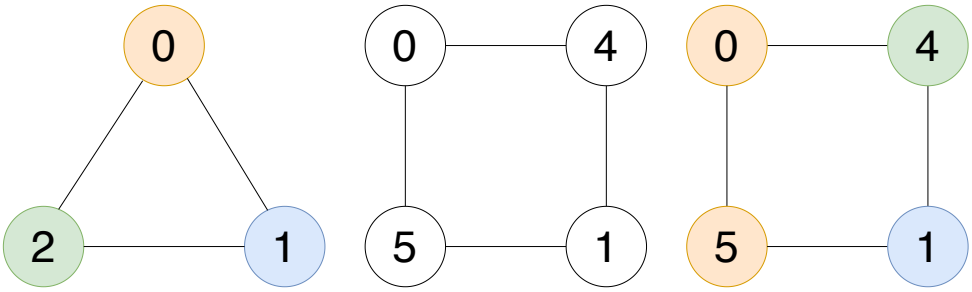

Figure 3.7: An embedding of a triangle graph into a part of Chimera cell

the actual problem solved.

3.5.3. Embedding more complex graphs

Fig. 3.8 presents the embedding of a little larger example of graph into Chimera. Vertices "0", "2" and "6" of the initial graph are represented as two vertices in the Chimera graph. As the Chimera graph is

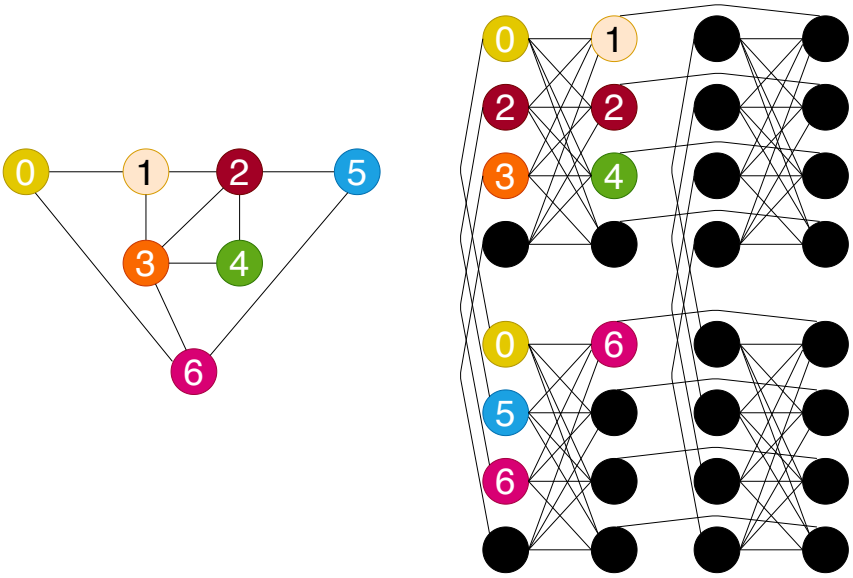

Figure 3.8: An example of a larger graph and it's embedding into 2x2 Chimera lattice. The vertices of Chimera that are not used in embedding are marked black.

quite sparse (degree value is 6) finding a minor is a difficult problem. There are lots of works discussing this problem ([3], [10], [7]). An efficient algorithm for solving the minor-embedding is presented in [14], its authors provide a proof outperformance of their algorithm against the D-Wave provided algorithm. The technique, which is universal but also expensive in terms of physical qubits use and chains length, is treating the initial graph as a minor of a complete graph with the same number of vertices and embedding the complete graph into Chimera. Fig. 3.9 presents such embedding for K_9 graph within the 2×2 Chimera

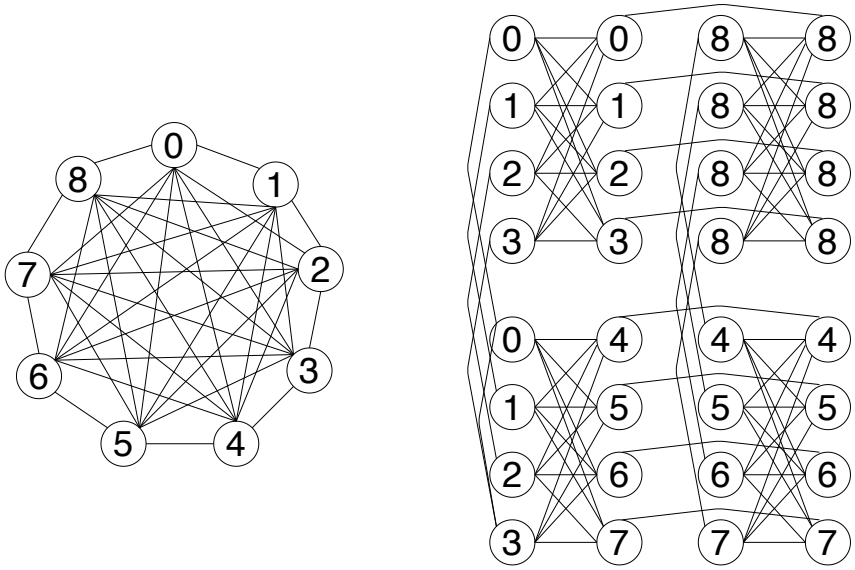

Figure 3.9: A K_9 graph embedding into Chimera part.

lattice. The method of embedding larger complete graphs on larger lattices is presented in the Fig. 3.10. The lattice can be theoretically infinite. The D-Wave 2000Q contains 16×16 lattice. On the $N \times N$ lattice it is possible to embed a $K_{(4N+1)\times(4N+1)}$ graph, therefore the largest complete graph possible to embed in the D-Wave 2000Q quantum annealer contains 65 vertices.

3.5.4. Obstacles for D-Wave 2000Q

Chains

Embeddings described in the previous sections require using chains. Their length ranges from 2 to infinity (17 in case of 16×16 Chimera lattice). The long chains raise the following problems:

- weak chains cause a tendency to break, and what follows, qubits that are assumed to have the same value, have different, which makes it immensely difficult to solve the actual problem
- strong chains are less likely to break, but the embedded QUBO problem with such chains covers the actual problem, which means, that the minima of this QUBO show only that the chains is not broken and do not reflect the energy minima of the problem solved (they compress the problem scale, in an extreme case the results are practically random)

Therefore the chain strength must be balanced between these two problems. There is no method for finding the perfect chain strength. It is necessary to find the proper strength by testing it for the QUBO

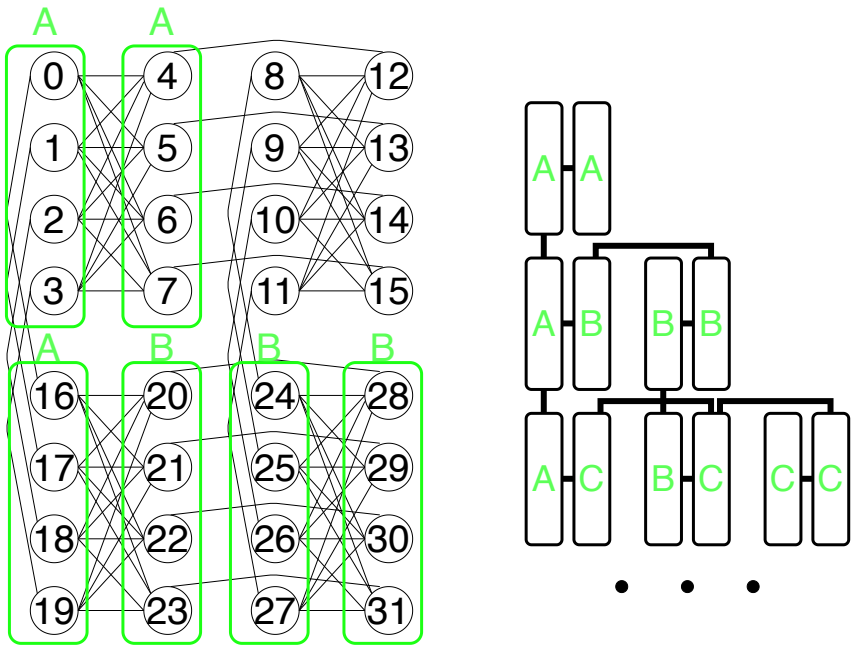

Figure 3.10: Symbolic presentation of an embedding of a complete graph. This schema can be continued infinitely.

.

tested. Usually the best method is to start from the value, which is close to the largest coupler or bias value in the QUBO and then increase, if necessary, until the amount of chains broken is large enough.

DAC resolution

Another factor that can significantly lower the quality of solutions retrieved from the D-Wave 2000Q computer is low resolution of DAC (Digital to Analog Converter). It is the part of the machine which converts values in the QUBO provided into the real magnetic fields and qubits couplings values. The limited number of bits used in such quantization [5] for QUBOs with lots of different values may have an effect of translating different values in QUBO into the same value on the machine, which obviously makes the results at least less accurate. These obstacles are not present in the Fujitsu Digital Annealer [6]. It is a special purpose computer with the complete graph with 8192 vertices as a base architecture and 64 bits of DAC resolution. However, Fujitsu Digital Annealer is not quantum.

3.6. D-Wave programming flow

All the previous sections present the parts of the programming flow necessary to gather the results of the problem solution with the use D-Wave 2000Q quantum annealer requires it's preprocessing and postprocessing. The whole programming flow consists of the following steps:

- stating the problem as QUBO (as presented in Section 3.4)
- minor-embedding the QUBO so that it matches the computer architecture

[5] https://docs.dwavesys.com/docs/latest/c_qpu_1.html#dac-quantization-ice-3

[6] https://www.fujitsu.com/global/services/business-services/digital-annealer/what-is-digital-annealer/index.html

- actual running on the machine
- reverse embedding and stating - transformation of the minor-embedded QUBO results into the problem solution

Summary

Mathematical and computational definitions as well as basics of quantum annealer computing have been described in this chapter. The definitions and methods presented here form a background for the precise definition and the solution of the problem solved in this thesis, which are presented in the next two chapters.

4. Problem definition

This chapter presents the formal description of the scheduling problem chosen to be solved in this thesis, which is an abstraction over workflow scheduling problems (for a general description of this group of problems see Section 2.2). It is a simple form of workflow scheduling based on the serverless architecture [4]. This description is the basis for the translation of the workflow scheduling problem into the form processable by the D-Wave 2000Q quantum annealer. To the best knowledge of the author of this thesis this set of problems has not yet been tested previously on a quantum annealer .

4.1. Formulation

The serverless architecture usually allows tasks to be run on lots of types of machines. Each machine has a different cost depending on it's type. To describe an abstraction over such model the following entities are included in the description:

- machines with the predefined cost per time unit
- tasks with the predefined execution time on each machine

It is assumed that the number of machines is infinite, but the number of machine types is limited. The workflow problem instance types considered in this thesis can be represented as directed acyclic graphs (DAGs). Vertices of the graph represent tasks (each vertex represents exactly one task, each task is represented by exactly one vertex). Directed edges represent the precedence order of tasks relatively to each other. If there is an edge from the vertex representing task A to the vertex representing task B then the task A must be finished before task B starts. If there is no edge, no such constraint occurs. The model presented here can be applied to any DAG. The formal definition of the problem requires the following input parameters:

- a times matrix $T = [\tau_{i,j}]_{t \times m}$ where t is number of tasks, m is number of machines and $\tau_{i,j}$ stores time of the task with number i being run on machine with number j.
- a machines cost per time unit vector $K = [k_i]_m$ measured in money units per time unit. This matrix can be used to calculate the cost of running the workflow
- a deadline d (an integer, limiting this value is crucial for the size of the resulting QUBO problem)
- Φ - a list of paths from the vertex representing first task to the vertex representing last task (both inclusive) in DAG representing a problem

The solutions must meet the following constraints:

- machine usage constraint - one task must have exactly one machine assigned
- deadline constraint - all tasks must finish before the designated time

The goal is to assign the tasks to machines so that this assignment meets the constraints mentioned above and the total cost of execution of all tasks is as low as possible.

4.2. Example problem instance

An example instance of the problem includes 8 tasks and 3 machines. It can be represented as the DAG presented on the figure 4.1 This graph can be written as one of the parameters aforementioned:

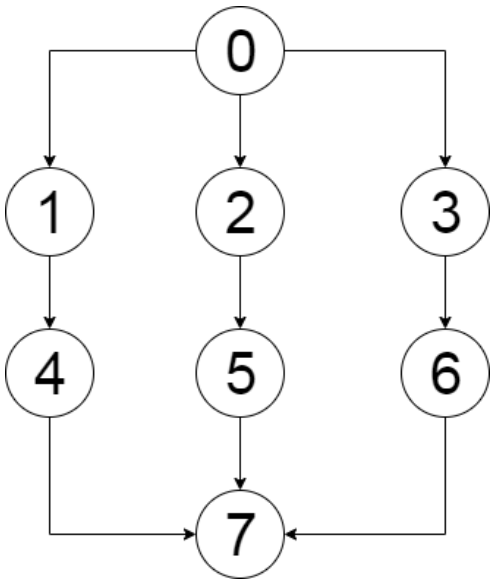

Figure 4.1: Graph representation of the example workflow problem.

$$\Phi = [[0, 1, 4, 7], [0, 2, 5, 7], [0, 3, 6, 7]]$$

The times matrix T can we written as follows:

$$T = \begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 6 & 12 & 24 \\ 4 & 2 & 14 & 6 & 10 & 2 & 4 & 8 \\ 8 & 4 & 28 & 12 & 20 & 4 & 8 & 16 \end{bmatrix}$$

For example: the execution time of the task with number 5 on the machine with number 1 equals 2 (all indexes start from 0). A machines cost vector in this example has the value:

$$K = \begin{bmatrix} 10 & 18 & 6 \end{bmatrix}$$

It means that one time unit of calculation on the machine with number 1 costs 18 units of money. The deadline value for the example problem is $D = 45$.

Summary

The scheduling problem chosen to be solved in this thesis has been formally described in this chapter. It will be used in the Chapter 5 to present the transformation of the problem into QUBO.

5. Problem transformation to the D-Wave-runnable form

This chapter presents the transformation of the problem stated in Chapter 4 to the form possible to run directly on the quantum annealer. The three-step transformation as well as the discussion of the transformed problem size in comparison to the original one is provided. Each step is supported by the example introduced in Section 4.2.

5.1. Transformation idea

As described in Section 3.6, running a problem instance on the D-Wave quantum annealer requires preparing an instance of a QUBO problem, which is represented by a matrix (or by a directed graph - interchangeably). As the QUBO problem can be without the loss of generality transformed into the upper triangle matrix (undirected graph) [20] this form will be used in the further description. The QUBO instance must also match the architecture of the hardware, so it has to be minor-embedded on it. This chapter presents the workflow problem transformation into such QUBO with the following steps:

- transformation into the commonly known BILP problem
- transformation of BILP problem into QUBO presented in the work [20]
- minor embedding the resulting QUBO onto the hardware graph

The resulting minor-embedded QUBO matrix is with no further processing the input for the quantum annealer. The remaining sections of this chapter present these steps in detail. Each of them is described for the general case as well as for the example problem from Section 4.2.

5.2. Workflow to BILP

The first step of the problem conversion presented in this chapter is transformation of the problem defined in Chapter 4 to BILP problem (described in Section 3.1).

5.2.1. Binary variables

In the problem transformed, the BILP binary variables are defined as $x = [x_i]_n$ where $n = t \cdot m$. Variable x_i is set to 1 if the task with number $(i \, mod \, t)$ runs on a machine with number $(i \, div \, t)$ where

Table 5.1: The binary variables $X = [x_0, x_1, x_2...x_{23}]$ for BILP formulation of the sample workflow problem. Task count is $t = 8$ and machine count is $m = 3$. Binary variable x_i is set to 1 when task with number $(i \, mod \, t)$ runs on a machine with number $(i \, div \, t)$ and to 0 otherwise.

		Task							
		1	2	3	4	5	6	7	8
Machine	0	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
	1	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
	2	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}

div is an integer division. Tab. 5.1 contains the example problem instance (Section 4.2) mapping of x_i parameters to task and machine combinations.

According to this indexing the matrix T is also used in a row-by-row vectorized form λ such that $\lambda_{i+t \cdot j} = T_{i,j}$. For the example problem instance, in which matrix T takes the value:

$$T = \begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 6 & 12 & 24 \\ 4 & 2 & 14 & 6 & 10 & 2 & 4 & 8 \\ 8 & 4 & 28 & 12 & 20 & 4 & 8 & 16 \end{bmatrix}$$

the λ vector is

$$\lambda = \begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 6 & 12 & 24 & 4 & 2 & 14 & 6 & 10 & 2 & 4 & 8 & 8 & 4 & 28 & 12 & 20 & 4 & 8 & 16 \end{bmatrix}$$

5.2.2. Costs vector transformation

In order to calculate a matrix γ of costs of tasks for each machine two input variables from the formulation in Section 4.1 are necessary:

- a times matrix $T = [\tau_{i,j}]_{t \times m}$
- a machines cost per time unit vector $K = [k_i]_m$

The cost matrix is created by multiplying the times matrix elements by the cost per time unit of the machine corresponding to this element:

$$\gamma_{i,j} = \tau_{i,j} \cdot k_j \tag{5.1}$$

A cost vector c is then obtained by row-by-row vectorization:

$$c_{i+t \cdot j} = \gamma_{i,j} \tag{5.2}$$

In the example from Section 4.2 costs matrix created by performing the aforementioned operations on matrix:

$$T = \begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 6 & 12 & 24 \\ 4 & 2 & 14 & 6 & 10 & 2 & 4 & 8 \\ 8 & 4 & 28 & 12 & 20 & 4 & 8 & 16 \end{bmatrix} \tag{5.3}$$

and vector $K = \begin{bmatrix} 10 & 18 & 6 \end{bmatrix}$ has the following form:

$$\gamma = \begin{bmatrix} 120 & 60 & 420 & 180 & 300 & 60 & 120 & 240 \\ 72 & 36 & 252 & 108 & 180 & 36 & 72 & 144 \\ 48 & 24 & 468 & 72 & 120 & 24 & 48 & 96 \end{bmatrix} \tag{5.4}$$

The flattened costs vector is:

$$c = [120, 60, 420, 180, 300, 60, 120, 240, 72, 36, 252, 108, 180, 36, 72, 144, 48, 24, 468, 72, 120, 24, 48, 96] \tag{5.5}$$

Such preparation of vector c makes $c^t X = \sum_{i=0}^{|X|-1} c_i x_i$ value a total cost of running. For example: if the solution is $x = [1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]$ the total cost of running the workflow is $1 \cdot 120 + 0 \cdot 60 + ... + 1 \cdot 96 = 1008$. Therefore the costs vector c can be also treated as c vector in BILP formulation provided.

5.2.3. Constraints transformation

All the constraints on the problem must be included in matrix A and vector b or BILP. Two types of constraints are considered:

- deadline constraint - all tasks must finish not later than the deadline specified
- machine usage constraint - one task can have only one machine assigned

Deadline constraint

Deadline constraint can be defined as follows: *for each path ϕ from list Φ the summary execution time of tasks included in this path must be less than or equal to the specified deadline d* . For the purpose of including the deadline constraint into BILP a matrix $R = [r_{i,j}]_{|X| \times |\Phi|}$ (the dimensions: $|X|$ - number of binary variables, $|\Phi|$ - number of all possible paths in the workflow DAG). R stores the time for each machine-based task assignment $i \in [0..n-1]$, but only if the task belongs to the path $\phi \in [0, |\Phi| - 1]$ as follows:

$$r_{i,\phi} = \begin{cases} t_i & \text{if path } \phi \text{ contains task number i mod t (t - task count),} \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

For the example problem R is a 24×3 matrix with the following values:

$$R = \begin{bmatrix} 12 & 6 & 0 & 0 & 30 & 0 & 0 & 24 & 4 & 2 & 0 & 0 & 10 & 0 & 0 & 8 & 8 & 4 & 0 & 0 & 20 & 0 & 0 & 16 \\ 12 & 0 & 42 & 0 & 0 & 6 & 0 & 24 & 4 & 0 & 14 & 0 & 0 & 2 & 0 & 8 & 8 & 0 & 28 & 0 & 0 & 4 & 0 & 16 \\ 12 & 0 & 0 & 18 & 0 & 0 & 12 & 24 & 4 & 0 & 0 & 6 & 0 & 0 & 4 & 8 & 8 & 0 & 0 & 12 & 0 & 0 & 8 & 16 \end{bmatrix} \tag{5.7}$$

The next step is to formulate a set of constraints corresponding to each path ϕ:

$$\sum_{i=0}^{|X|-1} r_{i,\phi} x_i \leq d, \tag{5.8}$$

which can be rewritten for $D = [d]_n$ as

$$Rx \leq D. \tag{5.9}$$

Path	Minimum execution time	d - (min. ex. time)	Binary variables' number
[0,1,4,7]	24	21	5
[0,2,5,7]	28	17	5
[0,3,6,7]	22	23	5

Table 5.2:

For the example problem the constraints can be written as follows:

$$\begin{cases} 12x_0 + 6x_1 + 30x_4 + 24x_7 + 4x_8 + 2x_9 + 10x_{12} + 8x_{15} + 8x_{16} + 4x_{17} + 20x_{20} + 16x_{23} \leq 45 \\ 12x_0 + 42x_2 + 6x_5 + 24x_7 + 4x_8 + 14x_{10} + 2x_{13} + 8x_{15} + 8x_{16} + 28x_{18} + 4x_{21} + 16x_{23} \leq 45 \\ 12x_0 + 18x_3 + 12x_6 + 24x_7 + 4x_8 + 6x_{11} + 4x_{14} + 8x_{15} + 8x_{16} + 12x_{19} + 8x_{22} + 16x_{23} \leq 45 \end{cases} \tag{5.10}$$

x_i values can be equal to 0 or 1. For example if $x_0 = x_{12} = x_{20} = 1$ and rest x_i values are 0 then the first constraint is fulfilled. The BILP formulation (3.5) includes only equalities whereas (5.8) forms the system of inequalities. Therefore it is necessary to transform these inequalities into equalities. It is made with the use of slack variables. The transformation is relying on the fact that there always exist a slack variable s such that for natural numbers

$$a \in \mathbb{N}, Z \in \mathbb{N}, a \leq Z \implies \exists s \in N : a + s = Z.$$

It is necessary to extend the matrix R with additional columns that represent all necessary slack variables [20]. Generally a slack variable value s is a natural number. As the variables in BILP are binary, it is necessary to represent them using standard binary expansion. If

$$\sum_{i;task(i)\in\phi} \min_j T_{i,j} \tag{5.11}$$

is indicated as the minimum time of running tasks in path ϕ, then the number of binary variables b for storing slack variables for each path ϕ equals

$$b(\phi) = log_2(|d - \sum_{i;task(i)\in\phi} \min_j T_{i,j}|). \tag{5.12}$$

For the example problem Tab. 5.2 presents the numbers of variables necessary to add for each path. Every path must have it's own set of slack variables.

Considering the slack variables, the constraints in this example takes the following form:

$$12x_0 + 6x_1 + 30x_4 + ... + 16x_{23} + \sum_{i=0}^{4} 2^i s_{i,0} = 45$$

$$12x_0 + 42x_2 + 6x_5 + ... + 16x_{23} + \sum_{i=0}^{4} 2^i s_{i,1} = 45$$

$$12x_0 + 18x_3 + 12x_6 + ... + 16x_{23} + \sum_{i=0}^{4} 2^i s_{i,2} = 45$$

where $s_{i,j}$ denotes a slack variable for the j-th path (j-th row of matrix R) responsible for including or not the bit representing the value 2^i. It is worth noticing that in every equation only one set of slack variables is included, the rest of them have zero value and as such zero are represented in the matrix R.

Machine usage constraint

This category of constraints enforces the fact that during the whole process one task can be assigned to one machine only, assigning to multiple machines would impose ambiguity in terms of calculating time and cost for this task. A matrix $U = [u_{i,j}]_{|X|,t}$ is defined (t - task count) such that for each task $s \in [0, t-1]$ and its machine-based position $i \in [0, n-1]$

$$u_{i,s} = \begin{cases} 1 & \text{if } i \bmod t = \text{s} \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

Therefore each row of U indicates all machine-based positions of a task coresponding to that row. To assure that only one machine is assigned to each task, the following constraint must be fulfilled:

$$\sum_{i=0}^{n-1} u_{i,s} x_i = 1 \tag{5.14}$$

The matrix U also must be extended with bits corresponding to slack variables described in deadline constraint paragraph. They are all set to 0. For the vector $E = [1]_n$ it is possible to write this constraint as

$$Ux = E \tag{5.15}$$

For the example problem it can be written as provided in equation (5.16)

$$\begin{cases} x_0 + x_8 + x_{16} = 1 \\ x_1 + x_7 + x_{17} = 1 \\ \qquad ... \\ x_7 + x_{15} + x_{23} = 1 \end{cases} \tag{5.16}$$

For this constraint a part of matrix U can be written as follows:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ ... \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

As mentioned before, the matrix U must be extended with the zeros representing slack variables. The first row such extended matrix takes the following form:
$U_0 = [1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$.

Final constraints formulation

The two mentioned type of constraints are connected together into the matrix A and vector b from BILP

formulation. If s is the total number of slack variables, the matrix A is defined as $A = [a_{i,j}]_{(|X|+s)\times(|\Phi|+t)}$ and combines the two constraint matrices as follows:

$$a_{i,j} = \begin{cases} r_{i,j} & 0 \leq j < r \\ u_{i,j-r} & r \leq j < r + t, \end{cases} \tag{5.18}$$

which is simply putting two matrices together binding with the row (matrix R above matrix U). Both matrices have the same number of columns equal $n + s$, because matrix U was intentionally extended with zeros without meaning for the machine usage constraint (as described in the previous paragraph). The vector b is defined similarly as putting the vector D above the vector E, which can be formally written as

$$b_i = \begin{cases} d & 0 \leq j < r \\ 1 & r \leq j < r + t. \end{cases} \tag{5.19}$$

Both constraints are now connected together in a matrix A and vector b, though they practically separate as they have different rows assigned. Thanks to such denotation it is possible to write all these constraints as

$$Ax = b, \tag{5.20}$$

which is the exact constraints' BILP formulation as described in Section 3.1. For the example problem the matrix A takes the form from equation (5.21).

$$A = \begin{bmatrix}
12 & 6 & 0 & 0 & 30 & 0 & 0 & 24 & 4 & 2 & 0 & 0 & 10 & 0 & 0 & 8 & 8 & 4 & 0 & 0 & 20 & 0 & 0 & 16 & 16 & 8 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 42 & 0 & 0 & 6 & 0 & 24 & 4 & 0 & 14 & 0 & 0 & 2 & 0 & 8 & 8 & 0 & 28 & 0 & 0 & 4 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 16 & 8 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 18 & 0 & 0 & 12 & 24 & 4 & 0 & 0 & 6 & 0 & 0 & 4 & 8 & 8 & 0 & 0 & 12 & 0 & 0 & 8 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 8 & 4 & 2 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{5.21}$$

The vector b takes the form such that $b^t = [45, 45, 45, 1, 1, 1, 1, 1, 1, 1, 1]$.

5.3. BILP to QUBO

BILP is a constrained problem in opposite to the QUBO - Quadratic **Unconstrained** Binary Optimization (for the difference between models see Section 3.4.1). Therefore the matrix A formulation from Section 5.2.3 for two constraints is sufficient for BILP, however when applying such constraints it is necessary to weigh them so that neither of them is covered by the other one. It is achieved by adding

an S parameter to the matrix U, which is the part of the matrix A from BILP formulation. Therefore the machine usage constraint takes the form the matrix U takes then the form

$$\sum_{i=0}^{n-1} S u_{i,s} x_i = S, \tag{5.22}$$

which does not change it mathematically, but raises it's relative weight to the deadline constraint. In the example problem the matrix U is changed into

$$U = \begin{bmatrix} S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & & & & & & & & & & & & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S \end{bmatrix} \tag{5.23}$$

and the whole matrix A has the form as in equation (5.24).

$$A = \begin{bmatrix} 12 & 6 & 0 & 0 & 30 & 0 & 0 & 24 & 4 & 2 & 0 & 0 & 10 & 0 & 0 & 8 & 8 & 4 & 0 & 0 & 20 & 0 & 0 & 16 & 16 & 8 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 42 & 0 & 0 & 6 & 0 & 24 & 4 & 0 & 14 & 0 & 0 & 2 & 0 & 8 & 8 & 0 & 28 & 0 & 0 & 4 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 16 & 8 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 18 & 0 & 0 & 12 & 24 & 4 & 0 & 0 & 6 & 0 & 0 & 4 & 8 & 8 & 0 & 0 & 12 & 0 & 0 & 8 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 8 & 4 & 2 & 1 \\ S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.24}$$

A matrix C for (5.25) must also be introduced for the purpose of this transformation. It is a matrix with non-zero values on the diagonal and zero values everywhere else such that $C_{i,i} = c_i$, where c is the vector from BILP formulation. Considering the modified matrix A, matrix C and vector b a QUBO matrix can be calculated using the formula (5.25) described in [20]):

$$y = x^T C x + P \cdot (Ax - b)^T (Ax - b) = x^T C x + x^T D x + c = x^T Q x + c. \tag{5.25}$$

The solved problem is an optimization problem, so the additive constant c can be dropped and the exact QUBO formulation (minimizing $x^T Q x$) can be formulated. It includes two scalar parameters:

- S - the machine usage constraint relative strength mentioned above
- P - all constraints penalty (as described in Section 3.4.1)

The way of finding these parameters' value is described in Section 5.5.

5.4. Minor embedding

For the purpose of this thesis the complete graph embedding into Chimera graph was used according to the method presented in Section 3.5.3, Fig. 3.10. For this method a new scalar parameter must be introduced: chain strength, as described in Section 3.5.2.

5.5. Finding scalar parameters' values

Three scalar parameters have been introduced in previous sections: S, P and chain strength. The value of all these parameters must be found for each problem separately. No exact method for this issue has been found. These parameters' values can be found with the use of the metaheuristic (it is used in this thesis). First, (1), an initial value of P is found based on [20]. The potential initial value of parameter S should be similar to values in vector T in order to achieve proper balance between constraints. Then, (2), the discrete space of possible pairs is searched (P, S) for values relatively close to the initial values, using a classical solver - Gurobi[1]. Next, (3), the chain strength between physical qubits is set for the minor-embedded problem basing on the D-Wave guidebook's [13] suggestion that it should be large enough to keep chains and small enough not to cover the actual problem. It has been found that it should approximate the largest values of the Q matrix. Finally, (4), the selected chain strength is used as an input for the dwave.embedding library, which performs actual minor embedding.

5.6. Discussion

There are a few problems with the solution provided that need to be discussed.

Size of the translated solution

If t is task count and m is machine count then the original problem could be represented by $t \cdot m$ bits ($= 24$ for the example problem). After adding slack variables to BILP formulation for s as total slack variables the number of bits grew up to $t \cdot m + s$ ($s = 15 -> t \cdot m + s = 39$ for the example problem). Minor embedding the clique generally requires using $1 + \lceil N/4 \rceil$ groups marked with letters in the Fig. 3.10 (where N is the number of bits of the original QUBO, 4 comes from the K_4^4 graph as described in Section 3.5.1). The same is the length of chains - number of physical qubits representing one logical qubit. Therefore the total number of qubits necessary to represent the problem is $(1+\lceil N/4 \rceil) \cdot (t \cdot m + s)$. For the example problem this number is equal to $(1 + \lceil 9.75 \rceil) \cdot (8 \cdot 3 + 15) = 11 \cdot 39 = 429$. This means that the problem solution must use 429 out of 2048 qubits of the D-Wave quantum annealer, which significantly decreases the scalability of this solution. Two main reasons for such state are (1) slack variables representation - the number of qubits grows logarithmically with the deadline value and (2) minor embedding - the number of physical qubits grows quadratically with the number of logical qubits.

Finding parameters' values

Another issue related to the solution proposed is the necessity to find empirically the values of parameters P,S and chain strength for each problem instance separately. For a proof of concept type of work, which

[1] http://www.gurobi.com

this thesis provides, it is acceptable, however considering the scalability of the solution (the goal of such work is to be able to solve problems that are impossible to solve with the methods using traditional computers) finding an automated and reliable method of finding these parameters would be highly usable. The method proposed in this thesis cannot be applied to large problem instances as it assumes solving the problem before applying it into quantum computer

Summary

This chapter proved that it is possible to translate the workflow scheduling problem to the form runnable with the use of the D-Wave 2000Q quantum annealer in spite of several limitations. Experiments have been made with the use of the web-accessible quantum annealer, results of which are presented in the next chapter.

6. Results analysis and comparison to classical methods.

This chapter provides the results of execution of the problem presented in the Chapter 5 on the D-Wave 2000Q quantum annealer as well as the description processing of such results.

6.1. Input and output format

Executing the problem with the use of D-Wave 2000Q requires passing the input data to it, which is a QUBO matrix representing the problem. This matrix must match the architecture of the computer - non zero values are only allowed in cells representing connections that really exist in the annealer, otherwise such matrix is not processed by the computer. After submitting such the QUBO, the return value is a list of samples. Each sample contains:

- the mapping of each qubit to its final state ("0" or "1"), which is the actual sample
- energy of the sample
- number of occurrences of this sample during the process of sampling

6.2. Correctness of returned samples

The results returned by D-Wave 2000Q quantum annealer need to be translated into the original form - the translation presented in Chapter 5 must be reversed.

Unembedding the problem into the initial QUBO

The first step of reversing the transformation is reversing the minor-embedding process, which is transforming the embedded QUBO (denoted as eQUBO) problem into the initial QUBO (denoted as iQUBO). In this process it is necessary to check whether the chains have not been broken. It is true if all the variables in eQUBO representing one variable in iQUBO have the same value. Only samples with not broken chains are proceeded to be checked as potential correct solutions in terms of the other constraints.

BILP constraints

If no chains from minor-embedding are broken, the initial QUBO represents the original WSP problem. It is necessary then to check the BILP constraints included in this QUBO, which are:

- machine usage constraint

- deadline constraint
- BILP equations correctness considering slack variables

The constraints are verified according to their definitions in Section 5.2.3.

Correctness example study

Let's consider four result vectors:

$$X_1 = [1,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,1,0,0,0]$$
$$X_2 = [1,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,0,1,0,0,0,0,1,1,0]$$
$$X_3 = [1,0,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,1,0]$$
$$X_4 = [1,0,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,0,0,1,0,1,0,1,0,1,1]$$

To make them easier to read the following notation of solutions is introduced:

$$X_1 = [x_0, x_2, x_5, x_{11}, x_{15}, x_{17}, x_{18}, x_{22}, x_{26}, x_{28}, x_{29}, x_{33}, x_{36}]$$
$$X_2 = [x_0, x_2, x_5, x_{11}, x_{15}, x_{17}, x_{20}, x_{22}, x_{28}, x_{29}, x_{31}, x_{36}, x_{37}]$$
$$X_3 = [x_0, x_5, x_{10}x_{11}, x_{15}, x_{17}, x_{20}, x_{22}, x_{28}, x_{31}, x_{36}, x_{37}]$$
$$X_4 = [x_0, x_5, x_{10}x_{11}, x_{15}, x_{17}, x_{20}, x_{22}, x_{28}, x_{31}, x_{33}, x_{35}, x_{37}, x_{38}]$$

In this notation only variables with value "1" are listed, the other are ignored. The variables with indices between 0 and 23 are the actual WSP variables from tasks and machines matrix, the variables with indices between 24 and 38 are the slack variables' binary expansions. The solution X_1 does not meet the machine usage constraint as

$$x_2 + x_{10} + x_{18} = 1 + 0 + 1 = 2 \neq 1$$
$$x_4 + x_{12} + x_{20} = 0 + 0 + 0 = 0 \neq 1$$

In such case it makes no sense to check the deadline constraint as well as the slack variables' correctness (in the extreme case all the binary variables could be equal to 0, and such solution would still meet the deadline constraint).
The solution X_2 does meet the machine usage constraint, but the deadline constraint is not met, because for the path $[0, 2, 5, 7]$ the execution time equals

$$time = x_0t_0 + x_2t_2 + x_{15}t_{15} = 12 + 42 + 6 + 8 = 68 > 45 = D$$

Therefore it is pointless to check if the slack variables are set correctly as for the exceeded deadline there is no such natural number, which added to the execution time would result in the deadline value.
The solution X_3 meets the machine usage constraint as well as deadline constraint (for paths [[0,1,4,7],[0,2,5,7],[0,3,6,7]] execution times are 44,40 and 34 respectively. For the deadline value 45 three slack variables represented by binary variables x_{24} to x_{28}, x_{29} to x_{33}, x_{34} to x_{38} should have values 1,5 and 11 respectively. The first slack variable has value $16{\cdot}x_{24}+8{\cdot}x_{25}+4{\cdot}x_{26}+2{\cdot}x_{27}+1{\cdot}x_{28} = 1$,

Table 6.1: The tested instances of the workflow problem (for definitions of parameters, see Section 4.1)

No.	Binary variable count	T	K	paths
1	8	$\begin{bmatrix} 6 & 3 & 12 & 9 \\ 2 & 1 & 4 & 3 \end{bmatrix}$	[1,4]	[[0,1,3],[0,2,3]]
2	10	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \end{bmatrix}$	[1,4]	[[0,1,4],[0,2,4],[0,3,4]]
3	15	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \\ 4 & 2 & 8 & 6 & 4 \end{bmatrix}$	[1,5,2]	[[0,1,4],[0,2,4],[0,3,4]]
4	18	$\begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 24 \\ 4 & 2 & 14 & 6 & 10 & 8 \\ 8 & 4 & 28 & 12 & 20 & 16 \end{bmatrix}$	K=[8,18,6]	[[0,1,3,5],[0,1,4,5],[0,2,4,5]]

so it is correct. However the second variable's value is 4 and the third's is 6. Therefore this solution also cannot be marked as correct, though it was very close.
Finally the solution X_4 meets all the constraints. It has the same values for binary variables from x_0 to x_{23}, so it meets the machine usage and deadline constraints. The slack variables' values are 1,5 and 11, so the left side of the equality with slack variables sums up to the right side, which is the deadline value.

Definition of correct and wrong solution

Based on the criteria mentioned above it can be defined the **correct solution** is the solution that has no chains broken, meets machine usage constraint, deadline constraint and the BILP equalities with slack variables match. If at least one of these conditions is not fulfilled, the solution is described as **wrong solution**. Only correct solutions can be considered when comparing their objective function, which is the total cost of execution. **The best solution** (or global optimum) is the correct solution having the lowest possible value of execution cost.

6.3. Problem instances solved

The instances of the abstraction over workflow scheduling problems solved in this thesis are presented in the Tab. 6.1. The Directed Acyclic Graphs representing these problems are presented in the Fig. 6.1.

6.4. Reference methods

The problems instances discussed in this thesis are small enough to be solved using classical methods. The following four classical methods have been used to verify D-Wave machine results:

- A brute-force method using initial problem formulation. It was used to calculate exact results, along with minimal energy, through direct use of the objective function (3.7),

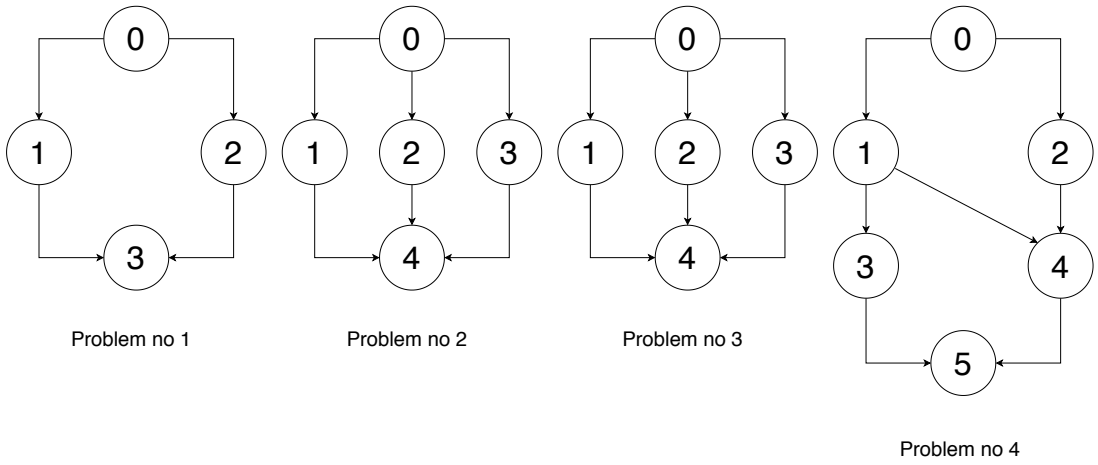

Figure 6.1: Graph representations of the tested workflow problems.

- GNU Linear Programming Kit[1] library for solving BILP problem prior to its translation to QUBO.
- Gurobi[2] sampler for the QUBO problem without minor-embedding. (requires two parameters: P, S)
- Gurobi sampler for QUBO problem with minor-embedding, requires three parameters (P, S, chain strength).

6.5. Results

This section presents the results of solving the workflow scheduling problem with the use of D-Wave 2000Q quantum annealer, which include:

- values of parameters P,S and chain strength values (for parameters' description see Section 5.3)
- results of problem execution on the quantum annealer

6.5.1. Values of transformation parameters

Values of parameters P,S and chain strength are dependent on the problem instance. Therefore a part of the solution is finding their values so that the problem has a chance to be properly solved on the quantum annealer using the method described in Section 5.5. For the four problem instances solved in this work the Tab. 6.2. There is no guarantee that these values are optimal, however they are the best found with the compliance to the method mentioned. Deadline value is not a result, however it's value has a significant impact on the parameters' values. The percentage of correct solutions is the probability of finding the correct solution for random choice from the uniform distribution.

[1] https://www.gnu.org/software/glpk/
[2] http://www.gurobi.com

Table 6.2: Values of parameters P, S and chain strength found for each size of the workflow problem for the given deadline (which is predefined, but has an impact on the parameters' values). The rightmost column represents the percentage of correct solutions in relation to all possible solutions.

No.	Binary variable number	Deadline	P	S	Chain strength	Percentage of correct solutions
1	8	19	8	10	1200	62,5%
2	10	19	14	25	6650	56,25%
3	15	17	11	10	2800	54,3%
4	18	70	6	40	18000	46,91%

Table 6.3: Summary of D-Wave 2000Q results. The fourth column presents the number of D-Wave unique correct solutions in relation to the total number of brute force correct solutions. The fifth column indicates whether the global optimum was found, listing the absolute energy error between the lowest D-Wave solution and the brute-force global optimum. The two rightmost columns refer to the execution time and cost of the lowest D-Wave solution respectively (min means global optimum cost).

No.	Binary variables number	Correct solutions samples (from 2000 samples)	Unique correct solutions	Global optimum found	Time	Cost
1	8	98	10 (100%)	YES	19(d=19)	34(min=34)
2	10	27	14 (77.8%)	YES	16(d=19)	40(min=40)
3	15	4	4 (3.03%)	NO (6)	16(d=17)	45(min=40)
4	18	0	0 (0%)	NO (65862)	N/A	N/A

6.5.2. Results of running on D-Wave 2000Q

The experiments were performed using the D-Wave 2000Q 5.0 machine. Each problem instance was sampled 2000 times with annealing time set to $8\mu s$. Fig. 6.2 shows the energy distribution of the actual results. For Problem 1 and Problem 2, the minimal energy corresponds to the global optimum, while for Problem 3 it indicates a correct solution, but not the best possible one. All energies found for the most complex Problem 4 correspond to wrong solutions. It can be noticed that the count of wrong solutions grows along with the size of the solution space. In general, the obtained characteristics of energies remain similar to [23].

Details of the results are described in Tab. 6.3, where the number of correct solutions found by D-Wave and their relation to the total number of correct solutions is presented. The results are compared to the global optima obtained by the brute force method.

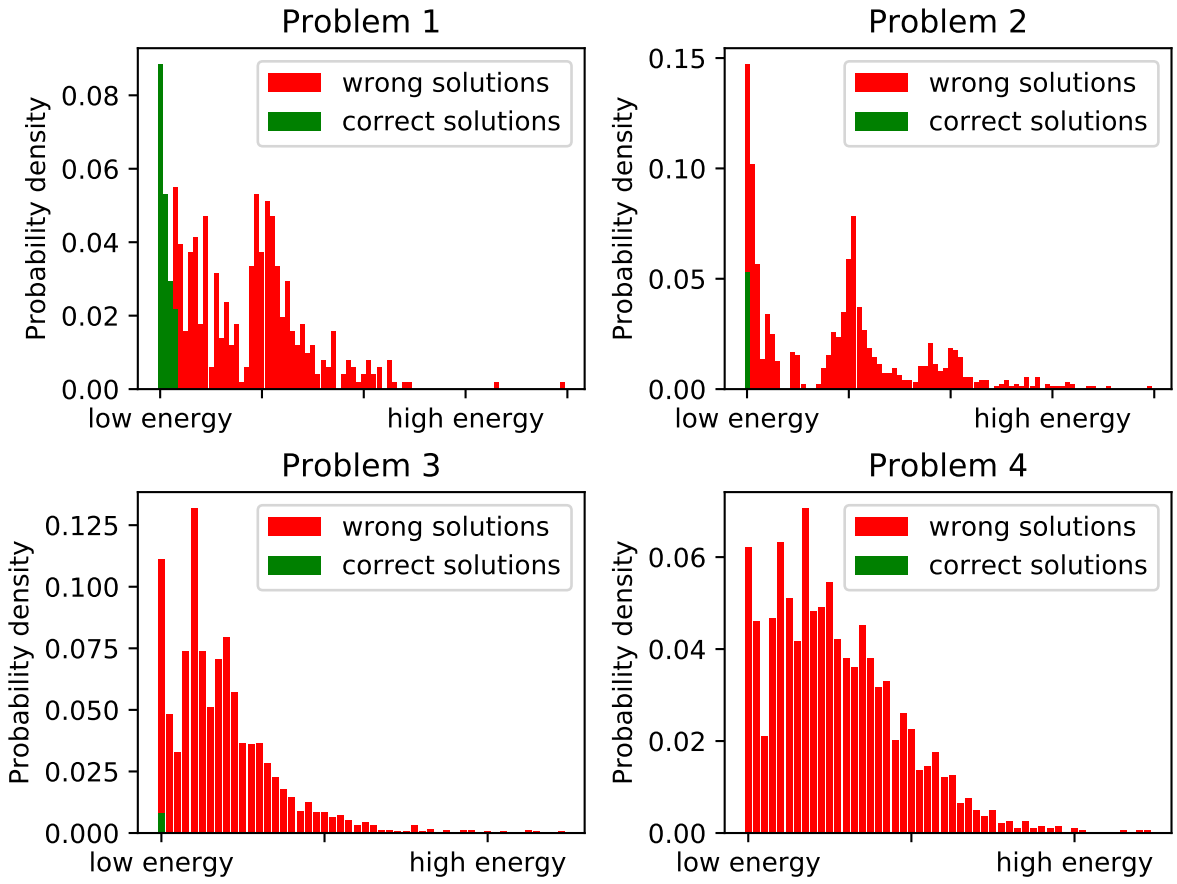

Figure 6.2: Histograms of results for the four types of workflow problems tested. The x axis represents values of energies equal to the value of minimized objective function (3.7). The y axis represents the probability density. For Problems 1-3 the correct solutions found are shown on the left-hand side of the spectrum.

6.6. Comparison to non-quantum methods

Three out of four methods returned the global optima for all the four problems' instances solved in this thesis:

– brute force method for the workflow scheduling problem

– GNU Linear Programming Kit for the BILP problem

– Gurobi sampler for the QUBO problem without minor-embedding

The problems' instances were small enough to perform such calculations. The fourth method - Gurobi sampler for minor-embedded QUBO did not always succeed in finding the best solution. The summary of results gathered with the use of this method is presented in 6.4. It can be noted that D-Wave results are comparable to Gurobi results. In the first two problems, workflow optimization was solved exactly and the global optimum was found. In problem 3, the Gurobi solution was still optimal, while the D-Wave

Table 6.4: Gurobi solutions table. The rightmost column shows whether the global optimum was found, with the absolute energy error between the lowest Gurobi solution and the brute force global optimum.

No.	Binary variables number	Global optimum found
1	8	YES
2	10	YES
3	15	YES
4	18	NO, wrong (22831)

solution was correct, but not the best ($\tilde{1}2\%$ worse in terms of cost; 33th out of 132 correct solutions). For problem 4 neither sampler found the global optimum.

7. Summary, conclusions and future work

This chapter provides the summary of the work. The solution is discussed, its advantages and disadvantages are pointed as well as the possible improvements.

7.1. Applicability of D-Wave 2000Q for task scheduling problems

The workflow scheduling problem is an example of task scheduling problem, checking the applicability of which was the main goal of this work. In this thesis it has been shown that it is possible to translate the workflow scheduling problem into a QUBO problem, execute it on a quantum annealer and achieve not only correct, but also globally optimal results for some of the analyzed problem instances. This fact as well the results of works like [50], [34] or [47] lead to the positive answer to the question of applicability of the D-Wave 2000Q quantum annealer for task scheduling problems.

7.2. Discussion of the quantum annealing based workflow scheduling problem solution

The transformation from Chapter 5 and the results presented in Chapter 6 prove that it is possible to solve the workflow scheduling problem on the quantum annealer. However there are some limitations and disadvantages to this approach.

7.3. Limitations and disadvantages

Transformation issues

As described in Section 5.6, the problem size increases significantly with the deadline value and number of paths from the beginning vertices to the ending vertices of the DAG representing the problem. This growth quantum annealer's qubits necessary to represent the problem has a negative impact on the problem's scalability. It also reduces the size of the problem that can be directly solved as the D-Wave 2000Q has 2048 qubits available. Another issue mentioned in the Section 5.6 is the necessity to find P, S and $chain\ strength$ parameters. The best achieved way of finding them requires the actual solving of many instances of the problem, therefore for larger instances such approach reduces usability of the solution.

High connectivity of the problem graph

The graph representing the transformed QUBO (after translating from BILP, before minor embedding) has quite a high connectivity. In the example problem described in Chapter 5 variables representing tasks 0 and 7 are present in all the equations responsible for the deadline constraint, therefore they are connected with all the variables. Obviously in the general case the density of QUBO is dependent on the problem instance, however high connectivity in the transformation presented in this thesis is unavoidable. Due to this fact it is necessary to use complete graph embedding during the minor-embedding process, which is quite a qubit-consuming method.

7.4. Possible improvements

The following improvements can be proposed as the future work on the topic of this thesis:

- proposing a better problem translation which would reduce the number of logical qubits and connectivity of graph e.g. using of the domain wall encoding proposed in the work [11], lower connectivity would also allow to use minor-embedding heuristics better than complete graph embedding,
- using heuristics reducing the size of problems applied into quantum annealer like time windows [28], which allow to run large problem instances with the use of quantum annealer, but may accumulate errors appearing in each step of execution,
- using the new D-Wave quantum annealer architecture - Pegasus - when it is released, it contains more qubits with higher connectivity (graph degree is 15 instead of 6 in Chimera).

It would also be useful to test and compare the solution with a non-quantum Fujitsu digital annealer [49] as both machines are designed to solve problems with a similar approach but different hardware.

Bibliography

[1] Getting Started with the D-Wave System, D-Wave User Manual (2018)

[2] Abbott, B.P., Abbott, R., Abbott, T., Abernathy, M., Acernese, F., Ackley, K., Adams, C., Adams, T., Addesso, P., Adhikari, R., et al.: Observation of gravitational waves from a binary black hole merger. Physical review letters **116**(6), 061102 (2016)

[3] Adachi, S.H., Adair, T.J., BOERKOEL, J.C., Brent, T.W., Campbell, D.S., ORNSTEIN, J.R., et al.: Heuristic graph embedding methods for adiabatic quantum computation optimization (May 7 2019), uS Patent 10,282,674

[4] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al.: Serverless computing: Current trends and open problems. In: Research Advances in Cloud Computing, pp. 1–20. Springer (2017)

[5] Berriman, G., Good, J., Laity, A., Bergou, A., Jacob, J., Katz, D., Deelman, E., Kesselman, C., Singh, G., Su, M.H., et al.: Montage: A grid enabled image mosaic service for the national virtual observatory. In: Astronomical Data Analysis Software and Systems (ADASS) XIII. vol. 314, p. 593 (2004)

[6] Booth, M., Reinhardt, S.P., Roy, A.: Partitioning optimization problems for hybrid classical. quantum execution. Technical Report pp. 01–09 (2017)

[7] Boothby, T., King, A.D., Roy, A.: Fast clique minor generation in Chimera qubit connectivity graphs. Quantum Information Processing **15**(1), 495–508 (2016)

[8] Van den Bossche, R., Vanmechelen, K., Broeckhove, J.: Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. In: 2011 IEEE third international conference on cloud computing technology and science. pp. 320–327. IEEE (2011)

[9] Brucker, P., Brucker, P.: Scheduling algorithms, vol. 3. Springer (2007)

[10] Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. arXiv preprint arXiv:1406.2741 (2014)

[11] Chancellor, N.: Domain wall encoding of discrete variables for quantum annealing and QAOA. Quantum Science and Technology **4**(4), 045004 (2019)

[12] Chapuis, G., Djidjev, H., Hahn, G., Rizk, G.: Finding maximum cliques on the d-wave quantum annealer. Journal of Signal Processing Systems **91**(3-4), 363–377 (2019)

[13] D-Wave Systems Inc.: D'wave problem solving handbook. https://docs.dwavesys.com/docs/latest/_downloads/09-1171A-A_Developer_Guide_Problem_Solving_Handbook.pdf

[14] Date, P., Patton, R., Schuman, C., Potok, T.: Efficiently embedding QUBO problems on adiabatic quantum computers. Quantum Information Processing **18**(4), 117 (2019)

[15] Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5), 528–540 (2009)

[16] Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**(1907), 553–558 (1992)

[17] Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**, 467–488 (1982). https://doi.org/10.1007/BF02650179

[18] Fidanova, S.: Simulated annealing for grid scheduling problem. In: IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06). pp. 41–45. IEEE (2006)

[19] Fidanova, S., Durchova, M.: Ant algorithm for grid scheduling problem. In: International Conference on Large-Scale Scientific Computing. pp. 405–412. Springer (2005)

[20] Glover, F., Kochenberger, G., Du, Y.: A Tutorial on Formulating and Using QUBO Models (2019)

[21] Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 212–219. STOC '96, ACM, New York, NY, USA (1996). https://doi.org/10.1145/237814.237866, http://doi.acm.org/10.1145/237814.237866

[22] Ising, E.: Beitrag zur theorie des ferromagnetismus. Zeitschrift für Physik **31**(1), 253–258 (1925)

[23] Jałowiecki, K., Więckowski, A., Gawron, P., Gardas, B.: Parallel in time dynamics with quantum annealers. arXiv preprint arXiv:1909.0429

[24] Jordan, S.: Quantum algorithms zoo web page. https://quantumalgorithmzoo.org/

[25] Khan, A., McCreary, C.L., Jones, M.S.: A comparison of multiprocessor scheduling heuristics. In: 1994 Internatonal Conference on Parallel Processing Vol. 2. vol. 2, pp. 243–250. IEEE (1994)

[26] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)

[27] Kitowski, J., Turała, M., Wiatr, K., Dutka, Ł.: PL-Grid: foundations and perspectives of national computing infrastructure. In: Building a National Distributed e-Infrastructure–PL-Grid, pp. 1–14. Springer (2012)

[28] Kurowski, K., et al.: Hybrid quantum annealing heuristic method for solving Job Shop Scheduling Problem. ICCS (2020). https://doi.org/0.1007/978-3-030-50433-5_39

[29] Lenstra, J.K., Kan, A.R., Brucker, P.: Complexity of machine scheduling problems

[30] Lenz, W.: Beitrag zum Verständnis der magnetischen Erscheinungen in festen Körpern. Z. Phys. **21**, 613–615 (1920)

[31] Maechling, P., Chalupsky, H., Dougherty, M., Deelman, E., Gil, Y., Gullapalli, S., Gupta, V., Kesselman, C., Kim, J., Mehta, G., et al.: Simplifying construction of complex workflows for non-expert users of the Southern California Earthquake Center Community Modeling Environment. ACM SIGMOD Record **34**(3), 24–30 (2005)

[32] Manne, A.S.: On the job-shop scheduling problem. Operations Research **8**(2), 219–223 (1960)

[33] McMahon, D.: Quantum computing explained. John Wiley & Sons (2007)

[34] Neukart, F., Compostella, G., Seidel, C., Von Dollen, D., Yarkoni, S., Parney, B.: Traffic flow optimization using a quantum annealer. Frontiers in ICT **4**, 29 (2017)

[35] Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Phys. Today **54**, 60–2 (2001)

[36] Omara, F.A., Arafa, M.M.: Genetic algorithms for task scheduling problem. In: Foundations of Computational Intelligence Volume 3, pp. 479–507. Springer (2009)

[37] Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Corporation (1998)

[38] Papalitsas, C., Andronikos, T., Giannakis, K., Theocharopoulou, G., Fanarioti, S.: A qubo model for the traveling salesman problem with time windows. Algorithms **12**(11), 224 (2019)

[39] Pelofske, E., Hahn, G., Djidjev, H.: Solving large Maximum Clique problems on a quantum annealer. In: International Workshop on Quantum Technology and Optimization Problems. pp. 123–135. Springer (2019)

[40] Pelofske, E., Hahn, G., Djidjev, H.: Solving large Minimum Vertex Cover problems on a quantum annealer. In: Proceedings of the 16th ACM International Conference on Computing Frontiers. pp. 76–84. ACM (2019)

[41] Ramamritham, K., Stankovic, J.A.: Scheduling algorithms and operating systems support for real-time systems. Proceedings of the IEEE **82**(1), 55–67 (1994)

[42] Rønnow, T.F., Wang, Z., Job, J., Boixo, S., Isakov, S.V., Wecker, D., Martinis, J.M., Lidar, D.A., Troyer, M.: Defining and detecting quantum speedup. Science **345**(6195), 420–424 (2014)

[43] Schaus, V., Fischer, P.M., Lüdtke, D., Tiede, M., Gerndt, A.: A continuous verification process in concurrent engineering. In: AIAA SPACE 2013 Conference and Exposition. p. 5429 (2013)

[44] Shin, Seung Woo and Smith, Graeme and Smolin, John A and Vazirani, Umesh: How" quantum" is the D-Wave machine? arXiv preprint arXiv:1401.7087 (2014)

[45] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)

[46] Steiger, D.S., Heim, B., Rønnow, T.F., Troyer, M.: Performance of quantum annealing hardware. In: Electro-Optical and Infrared Systems: Technology and Applications XII; and Quantum Information Science and Technology. vol. 9648, p. 964816. International Society for Optics and Photonics (2015)

[47] Stollenwerk, T., Basermann, A.: Experiences with scheduling problems on adiabatic quantum computers. In: Proceedings of the 1st International Workshop on Post-Moore Era Supercomputing (PMES). pp. 45–46. Future Technologies Group Technical Report FTGTR-2016-11 (2016)

[48] Tomasiewicz, D., Pawlik, M., Malawski, M., Rycerz, K.: Foundations for workflow application scheduling on d-wave system. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science – ICCS 2020. pp. 516–530. Springer International Publishing, Cham (2020)

[49] Tsukamoto, S., Takatsu, M., Matsubara, S., Tamura, H.: An accelerator architecture for combinatorial optimization problems. Fujitsu Sci. Tech. J **53**(5), 8–13 (2017)

[50] Venturelli, D., Marchand, D., Rojo, G.: Job shop scheduling solver based on quantum annealing. In: Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS). pp. 25–34 (2016)

[51] Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. In: Metaheuristics for scheduling in distributed computing environments, pp. 173–214. Springer (2008)

A. The paper for the Quantum Computing Workshop thematic track on International Conference on Computational Science 2020

Based on the results obtained in this thesis a paper has been prepared and published on the Quantum Computing Workshop thematic track on International Conference on Computational Science 2020 [48].

Foundations for Workflow Application Scheduling on D-Wave System

Dawid Tomasiewicz[1], Maciej Pawlik[1], Maciej Malawski[1], and Katarzyna Rycerz[1]

Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
tomasiewicz.dawid@gmail.com, {mapawlik,malawski,kzajac}@agh.edu.pl

Abstract. Many scientific processes and applications can be represented in the standardized form of workflows. One of the key challenges related to managing and executing workflows is scheduling. As an NP-hard problem with exponential complexity it imposes limitations on the size of practically solvable problems. In this paper, we present a solution to the challenge of scheduling workflow applications with the help of the D-Wave quantum annealer. To the best of our knowledge, there is no other work directly addressing workflow scheduling using quantum computing. Our solution includes transformation into a Quadratic Unconstrained Binary Optimization (QUBO) problem and discussion of experimental results, as well as possible applications of the solution. For our experiments we choose four problem instances small enough to fit into the annealer's architecture. For two of our instances the quantum annealer finds the global optimum for scheduling. We thus show that it is possible to solve such problems with the help of the D-Wave machine and discuss the limitations of this approach.

Keywords: D-Wave, QUBO, workflow scheduling, serverless

1 Introduction

The paradigm of workflows is commonly used for describing and preserving complex scientific processes and applications [13]. Workflows are usually represented as Directed Acyclic Graphs (DAG) [23]. Each vertex represents a task, while edges designate dependencies or data transfers between tasks. By using such a general representation it is possible to improve application portability and reusability. The abstract graph representation enables decoupling the application from a specific infrastructure and easily extract parts of the process with clear understanding of what the extracted part does and what its dependencies and outputs are. Some examples of scientific applications implemented as workflows include: Montage [4] – image mosaic software used to construct human-perceptible images of sky features from multiple images captured by telescopes; software used by the LIGO collaboration, designed to process data related to detecting gravitational waves [1]; software designed to predict the occurrence and effects of earthquakes based on geological data [23].

One of the key challenges related to managing and executing scientific workflows is scheduling. The general goal of scheduling is to create a plan of execution with respect to given parameters such as deadline, budget and computing resources. For the scope of this paper we assumed a simple form of workflow scheduling, where we operate on the serverless infrastructure discussed in [3]. In this case the serverless infrastructure is implemented as a set of cloud functions, provided by major cloud service providers, such as Google, Amazon, Azure and IBM. Cloud functions have the potential to be used for compute-intensive tasks, as proposed in [27], with particular emphasis on challenges which require high levels of infrastructure elasticity. Scheduling workflows for serverless infrastructures can be summarized as defining the runtime parameters of a function based on user-supplied deadline and budget. In this paper we assume that the serverless infrastructure consists of an infinite number of machine instances, while the number of machine types is finite, with each machine type associated with specific cost and performance.

In recent years, quantum computing has become popular due to its potential ability to solve problems that are beyond the capabilities of classical computing infrastructures. The research is still in its early stage, however there are many theoretical quantum algorithms already available, such as famous polynomial time Shor factorization [26] or $O(\sqrt{N})$ complexity Grover search in unsorted databases [16]. A good overview of the current status of theoretical quantum algorithms can be found in [18]. There are also various attempts to implement algorithms on the available quantum hardware: IBM-Q[1], Rigetti computing[2] or D-Wave[3]. Scheduling problems are assumed to be one of the challenges which might be efficiently solved by this new approach.

In this paper, we present a solution for the workflow scheduling problem with the use of the D-Wave quantum annealer. In particular, we propose a method of reformulating the problem as Quadratic Unconstrained Binary Optimization (QUBO) [21] required by D-Wave. To achieve this, we developed a Binary Integer Linear Programming (BILP) [19] formulation of the problem, which is then translated to QUBO in a similar way as shown in [14]. Finally, we discuss results obtained on the annealer. We attempt to find the optimal solution for selected instances of workflow scheduling problems, which are constrained by the deadline and have the lowest cost possible.

This paper is organized as follows. In Section 3 we provide the overview of quantum computation with the use of the quantum annealer. In Section 4 we present the precise formulation of the problem. In Section 5 we provide a complete description of transforming the workflow scheduling problem into a QUBO problem. First, the transformation to BILP is presented, which includes all the constraints necessary to be translated. Next comes BILP to QUBO problem translation, using methods from [14]. Finally, Sections 6 and 7 present full re-

[1] https://quantum-computing.ibm.com/
[2] https://www.rigetti.com/
[3] https://www.dwavesys.com/

sults with a detailed commentary, as well as discussion and suggestions for future work in this matter.

2 Related work

There is a significant body of knowledge available concerning the topic of scheduling workflows on cloud infrastructures. Due to its widespread adoption, most scheduling solutions operate on Infrastructure as a Service (IaaS) services. To the best of our knowledge, there is no other work directly addressing the problem of workflow scheduling on serverless infrastructures with help of quantum computing. In the presented case, the serverless infrastructure is represented by a Function as a Service (FaaS) type of service. For example, in [2] Arabnejad et al. propose a heuristic list scheduling algorithm with low computational complexity, designed for running workflows on IaaS. Work presented in this paper aims to provide similar features, albeit for FaaS and with help from quantum computing. In particular, we propose a method to create the final execution plan upfront in a short time and at low cost. One of the factors included in the presented algorithm is the sole cost of executing workflow tasks in the cloud. This problem was discussed in more detail in [30]. Zhou et al. addressed the problem of performance offered by cloud services with specific focus on the use case of running workflow applications. The matter of performance offered by serverless infrastructures was studied in [24], with focus on potential parallelism and application of FaaS as an infrastructure for large-scale scientific workflows. The specific topic of scheduling workflows on serverless infrastructures was addressed by Kijak et al. in [20], where they proposed a heuristic algorithm for scheduling workflows on cloud functions.

Although solving scheduling problems with quantum computers is a novelty, some examples of such work are available. In particular, a promising approach is to use the D-Wave quantum annealer based on a chimera graph [5]. A possible method of solving a shop job scheduling problem (JSP) [15] using D-Wave is shown in [29], however the proposed three-dimensional structure of the problem description is a strong limitation for scalability. In general, many NP-hard problems can be formulated as Ising problems [22] which can then be solved using a quantum annealer. For example [8] discusses the problem of finding maximum cliques in a graph. An important factor for solving problems with the use of quantum annealers is small machine size. In [25] the authors discuss heuristics for solving large-scale problems described in [8].

3 D-Wave computing overview

D-Wave 2000Q [5] is an adiabatic quantum computer that, unlike its universal counterparts (e.g. IBM-Q or Rigetti), cannot run algorithms implemented with the use of general quantum circuits. Therefore, in spite of its architecture offering a relatively large number of qubits, its usage is limited to certain types of optimisation problems. It is a fully analog machine, the result of which depends on the

value of the magnetic field applied to each qubit and the value of the connection between qubits (coupler). Programming the D-Wave 2000Q computer involves determining values of fields and associations. The aim of quantum annealing is to find the minimal energy state for the problem defined by a programmer. Annealing begins with an initial well-known quantum system for which the minimal energy state is known. Then it slowly switches to the quantum system related to the search problem. Ideally, during the whole process of switching, the system remains in its minimal energy state, so it should end up in the minimal energy state of the intended problem.

From a programmer's perspective, the problems to be solved must be formulated as an objective function using the Ising model [22] or, alternatively, a QUBO problem description [21]. Both approaches are isomorphic and can be transformed into each other in polynomial time. In this study, we choose to use QUBO formulation of the problem. In general, a QUBO problem consists of a matrix Q of size NxN, where N is the number of used binary variables and the size of the vector of binary variables that constitute the searched state. It can be converted without loss of generality to the upper triangular matrix as described in [14]. The actual problem solved by D-Wave is to find the final state of n binary variables $X = (x_1, x_2...x_n)$ that minimizes the objective function defined as

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i,j,i<j} Q_{i,j} x_i x_j. \tag{1}$$

Elements with the same indexes ($Q_{i,i}$) are responsible for "bias", i.e. the initial value of the magnetic field applied to the qubit, and elements with unequal indexes are responsible for the links between qubits (couplers). Minimizing such function is an NP-hard problem which makes it well suited for solving with help from the D-Wave 2000Q quantum computer.

The problem description does not limit the number of connections between qubits. However, the actual hardware is, in fact, a chimera graph (16x16 board of $K(4,4)$ graphs, degree 6); thus, the problem graph must be mapped onto that architecture, which is achieved by minor-embedding. It is a process of mapping each logical binary variable present in a problem description to a group (called a chain) of physical qubits on the actual machine so all required connections are realized. For dense matrices there is probably no better method than minor-embedding based on complete graph embedding described in [11]. For sparse matrices it is reasonable to attempt heuristics to lower the qubit chain length (e.g. as described in [6]).

4 Workflow problem formulation

In its general form, the workflow application can be represented as a Directed Acyclic Graph, where each task is represented by a vertex. An example graph is depicted in Fig. 1. We assume that the serverless infrastructure consists of a finite number of machine types, albeit with an infinite number of instances for

each type. Each machine type has an associated cost. Each task has an associated running time for each machine type. Our model can be applied to any DAG.

The goal is to assign a machine type to each task with minimal total cost, while respecting the deadline. For simplicity, we will use the term *machine* instead of *machine type* from now on.

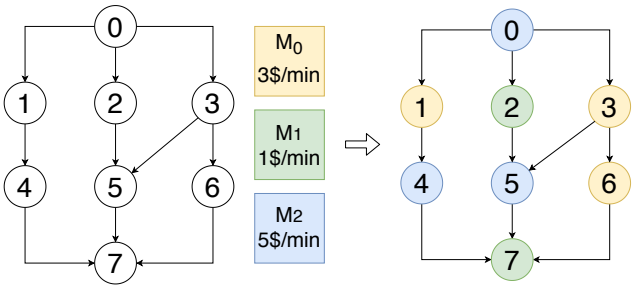

Fig. 1: Example workflow problem as a graph with task count t=8. We search for an assignment of each task to one of the machine types. For example, with machine count m=3 vector $K = [k_0, k_1, k_2]$ describes the cost of time unit execution on each machine while matrix $T = [\tau_{i,j}]_{8\times3}$ describes the execution time of task i running on machine j

The formal definition of the problem addressed in this paper consists of:

- a time matrix $T = [\tau_{i,j}]_{t\times m}$ where t is the number of tasks, m is the number of machines and $\tau_{i,j}$ expresses the execution time of task with number i on the machine with number j;
- a machine cost per time unit vector $K = [k_i]_m$ measured in currency units per time unit. This matrix can be used to calculate the cost of running the workflow;
- a deadline d (an integer, limiting this value is crucial for the size of the resulting QUBO problem);
- a list Θ of paths from the vertex representing the first task to the vertex representing the final task (both inclusive) in a DAG representing the problem.

For the purpose of describing results, we apply the following terminology: a **correct** result meets all the constraints, the **(global) optimum** is a correct result which has the lowest cost possible for the problem instance, while a **wrong** result fails to meet at least one constraint.

5 Transformation to QUBO problem

The problem addressed in this paper is NP-hard [9]. We provide its translation to a QUBO problem which consists of two steps. First, we propose a transformation

of the problem to BILP, which is described in this section. The second step shows how to translate BILP to a QUBO problem using [14].

The general goal of solving a BILP problem (also called "0-1 Integer Programming" [19]) is to find, for a given vector $C = [c_i]_n$, n binary numbers $X = [x_1, ..., x_n]$, for which the function

$$f(x_1, ..., x_n) = C^T X = \sum_{j=1}^{n} c_j x_j \tag{2}$$

is minimal, subject to constraints indicated by the following linear equation:

$$Ax = b, \tag{3}$$

where $A = [a_{i,j}]_{n \times w}$ stores w constraints for n binary numbers.

Translation from initial formulation to BILP
In our problem, the BILP binary variables are defined as $x = [x_i]_n$ where $n = t \cdot m$ (see Section 4). Variable x_i is set to 1 if the task with number $(i \, mod \, t)$ runs on a machine with number $(i \, div \, t)$ where div is an integer division. Tab. 1 contains an example mapping of x_i parameters to task and machine combinations.

Table 1: The binary variables $X = [x_0, x_1, x_2 ... x_{31}]$ for BILP formulation of the sample workflow problem. Task count is $t = 8$ and machine count is $m = 4$. Binary variable x_i is set to 1 when task with number $(i \, mod \, t)$ runs on a machine with number $(i \, div \, t)$ and to 0 otherwise.

		Task							
		1	2	3	4	5	6	7	8
Machine	0	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
	1	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
	2	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}
	3	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x_{31}

The BILP minimization function can subsequently be defined as the total cost of running tasks on selected machines

$$f(X) = \sum_{i=1}^{n} c_i x_i \ , \tag{4}$$

where $C = [c_i]_n$ is the cost vector and c_i indicates the cost of running task with number $(i \, mod \, t)$ on the machine with number $(i \, div \, t)$.

The cost vector needed for BILP formulation of the problem is obtained from the problem definition (Section 4) as follows: first, the cost matrix $[\gamma_{i,j}]_{t \times m}$ is created by multiplying $\gamma_{i,j} = \tau_{i,j} \cdot k_j$. Next, the cost vector C is obtained by row-by-row vectorization $[\tau_{i,j}]$ using $c_{i+t \cdot j} = \tau_{i,j}$. Finally, the time vector $T = [t_i]_n$

is obtained in an analogous manner by calculating $t_{i+t\cdot j} = \gamma_{i,j}$.

In addition to the minimization function, there are also two types of constraints, the deadline and machine usage, where it is necessary to ensure that only a single machine will be selected for a given task.

Deadline constraints. We define a matrix $R = [r_{i,j}]_{n\times r}$, where r is the number of all possible paths in the workflow DAG. R stores the time for each machine-based task assignment $i \in [0..n-1]$, but only if the task belongs to the path $\theta \in [0, r-1]$ as follows:

$$r_{i,\theta} = \begin{cases} t_i & \text{if path } \theta \text{ contains task number } i \, mod \, t \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Next, we formulate a set of constraints corresponding to each path θ

$$\sum_{i=0}^{n-1} r_{i,\theta} x_i \leq d \, , \tag{6}$$

which we can rewrite as (for $D = [d]_n$)

$$Rx \leq D. \tag{7}$$

To describe constraints as part of BILP according to (3) we need to transform an inequality (7) into an equality. To achieve this, we rely on the fact that there always exists a slack variable s such that for natural numbers

$$a \in \mathbb{N}, Z \in \mathbb{N}, a \leq Z \implies \exists s \in N : a + s = Z.$$

Therefore it is necessary to extend R with additional columns that represent all necessary slack variables [14]. Generally a slack value s is a natural number, so we represent it using standard binary expansion. If we indicate

$$\sum_{i;task(i)\in\theta} \min_j \tau_{i,j} \tag{8}$$

as the minimum time of running tasks in path θ, then the number of binary variables b for storing slack variables for each path θ equals:

$$b(p) = log_2(|d - \sum_{i;task(i)\in\theta} \min_j \tau_{i,j}|). \tag{9}$$

Machine usage constraints. The next category of constraints comprises "one machine per task only" constraints. We define $U = [u_{i,j}]_{n\times t}$ such that for each task $s \in [0, t-1]$ and its machine-based position $i \in [0, n-1]$

$$u_{i,s} = \begin{cases} 1 & \text{if } i \, mod \, n = s \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Therefore, each row of U indicates all machine-based positions of a task corresponding to that row. Then, to assure that only one machine is assigned to each task, the following constraint must be fulfilled:

$$\sum_{i=0}^{n-1} u_{i,t} x_i = 1. \tag{11}$$

If we indicate s as the total number of slack variables, the final BILP constraints matrix (3) is defined as $A = [a_{i,j}]_{(n+s)\times(r+t)}$ and combines all constraints together as follows:

$$a_{i,j} = \begin{cases} r_{i,j} & 0 \leq j < r \\ u_{i,j-r} & r \leq j < r+t. \end{cases} \tag{12}$$

To perform this operation, we need matrices of compatible sizes. Therefore, the matrix U must also be extended with slack variables, which are set to 0.

Vector b from (3) is defined in a similar manner:

$$b_i = \begin{cases} d & 0 \leq j < r \\ 1 & r \leq j < r+t. \end{cases} \tag{13}$$

Translation from BILP to QUBO problem.
Given matrices A, C and vector b a QUBO matrix can be calculated using the following formula (from [14]):

$$y = x^T Cx + P \cdot (Ax - b)^T (Ax - b) = x^T Cx + x^T Dx + c = x^T Qx + c. \tag{14}$$

By dropping the additive constant c, the exact QUBO problem form, which is minimizing $x^T Qx$, can be formulated. However it is necessary to introduce two additional scalar parameters:

- P - relative strength of all constraints in relation to the objective function, see (14)
- S - weight required for balancing R and U values so that the constraints they represent are efficiently included in the QUBO problem. Namely, it replaces constraints defined by (11) with

$$\sum_{i=0}^{n-1} S u_{i,t} x_i = S. \tag{15}$$

Both mentioned parameters, along with the minor embedding chain strength, need to be balanced, to make sure that (1) the solution meets constraints and (2) the objective function is minimized. Finding proper values for these parameters is not a trivial task. It is important to mention that the resulting QUBO problem might have high resolution, which can result in errors due to the limited resolution of the annealer's Digital to Analog Converter (DAC), so the hardware conversion from QUBO to analog values may not be accurate enough.

6 Results

In this section we describe the experimental results obtained using D-Wave 2000Q, compared with selected classical reference methods. In order to match the limitations of the existing quantum annealer, we considered four sample instances of the problem. Their DAG representation is shown in Fig. 2 and parameters in Tab. 2.

Table 2: The tested instances of the workflow problem (for definitions of parameters, see Section 4)

No.	Binary variable count	T	K	paths
1	8	$\begin{bmatrix} 6 & 3 & 12 & 9 \\ 2 & 1 & 4 & 3 \end{bmatrix}$	[1,4]	[[0,1,3],[0,2,3]]
2	10	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \end{bmatrix}$	[1,4]	[[0,1,4],[0,2,4],[0,3,4]]
3	15	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \\ 4 & 2 & 8 & 6 & 4 \end{bmatrix}$	[1,5,2]	[[0,1,4],[0,2,4],[0,3,4]]
4	18	$\begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 24 \\ 4 & 2 & 14 & 6 & 10 & 8 \\ 8 & 4 & 28 & 12 & 20 & 16 \end{bmatrix}$	K=[8,18,6]	[[0,1,3,5],[0,1,4,5],[0,2,4,5]]

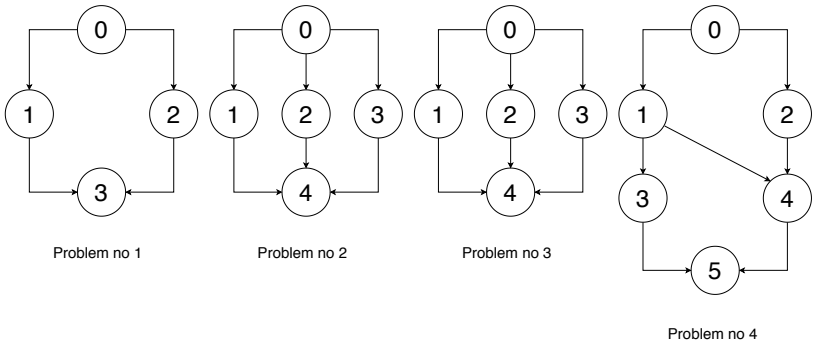

Fig. 2: Graph representations of the tested workflow problems.

Finding parameters: P, S and chain strength. Parameters needed for QUBO problem formulation are dependent on the specific problem instance. For the purpose of this research, parameters were obtained using a metaheuristic. First, (1), we find an initial value of P basing on [14]. The potential initial

value of parameter S should be similar to values in vector T in order to achieve proper balance between constraints. Then, (2), we search the discrete space of possible pairs (P, S) for values relatively close to the initial values, using a classical solver – Gurobi[4]. Next, (3), we set the chain strength between physical qubits for the minor-embedded problem basing on the D-Wave guidebook's [11] suggestion that it should be large enough to keep chains and small enough not to cover the actual problem. We have found that it should approximate the largest values of the Q matrix. Finally, (4), the selected chain strength is used as an input for the dwave.embedding library, which performs actual minor embedding. The final parameter values are presented in Tab. 3. Deadline values for all four problems were selected in such a way that they yield similar percentages of correct solutions.

Table 3: Values of parameters P, S and chain strength found for each size of the workflow problem for the given deadline (which is predefined, but has an impact on the parameters' values). The rightmost column represents the percentage of correct solutions in relation to all possible solutions.

No.	Binary variable number	Deadline	P	S	Chain strength	Percentage of correct solutions
1	8	19	8	10	1200	62,5%
2	10	19	14	25	6650	56,25%
3	15	17	11	10	2800	54,3%
4	18	70	6	40	18000	46,91%

Reference methods. The problems discussed in this paper are small enough to be solved using classical methods. The following four classical methods have been used to verify D-Wave machine results:

- A brute-force method using initial problem formulation. It was used to calculate exact results, along with minimal energy, through direct use of the objective function. (1),
- GNU Linear Programming Kit[5] library for solving BILP problem prior to its translation to QUBO. This method always finds the global optima.
- Gurobi sampler for the QUBO problem without minor-embedding. This method always finds the global optima, provided that parameters P and S are set up properly.
- Gurobi sampler for QUBO problem with minor-embedding, requiring three parameters (P, S, chain strength). The summary of the results is presented in Tab. 4.

Experiments with D-Wave. We perform experiments using the D-Wave 2000Q 5.0 machine sampled 2000 times with annealing time set to $8\mu s$. Fig. 3 shows

[4] http://www.gurobi.com
[5] https://www.gnu.org/software/glpk/

Table 4: Gurobi solutions table. The rightmost column shows whether the global optimum was found, with the absolute energy error between the lowest Gurobi solution and the brute force global optimum.

No.	Binary variables number	Global optimum found
1	8	YES
2	10	YES
3	15	YES
4	18	NO, wrong (22831)

the energy distribution of the actual results. For Problem 1 and Problem 2, the minimal energy corresponds to the global optimum, while for Problem 3 it indicates a correct solution, but not the best possible one. All energies found for the most complex Problem 4 correspond to wrong solutions. It can be noticed that the count of wrong solutions grows along with the size of the solution space. In general, the obtained characteristics of energies remain similar to [17].

Details of the results are described in Tab. 5, where the number of correct solutions found by D-Wave and their relation to the total number of correct solutions is presented. The results are compared to the global optima obtained by the brute force method. It can be noted that D-Wave results are comparable to Gurobi results. In the first two problems, workflow optimization was solved exactly and the global optimum was found. In problem 3, the Gurobi solution was still optimal, while the D-Wave solution was correct, but not the best ($\tilde{1}2\%$ worse in terms of cost; 33th out of 132 correct solutions). For problem 4 neither sampler found the global optimum.

Table 5: Summary of D-Wave 2000Q results. The fourth column presents the number of D-Wave unique correct solutions in relation to the total number of brute force correct solutions. The fifth column indicates whether the global optimum was found, listing the absolute energy error between the lowest D-Wave solution and the brute-force global optimum. The two rightmost columns refer to the execution time and cost of the lowest D-Wave solution respectively (*min* means global optimum cost).

No.	Binary variables number	Correct solutions samples (from 2000 samples)	Unique correct solutions	Global optimum found	Time	Cost
1	8	98	10 (100%)	YES	19(d=19)	34(min=34)
2	10	27	14 (77.8%)	YES	16(d=19)	40(min=40)
3	15	4	4 (3.03%)	NO (6)	16(d=17)	45(min=40)
4	18	0	0 (0%)	NO (65862)	N/A	N/A

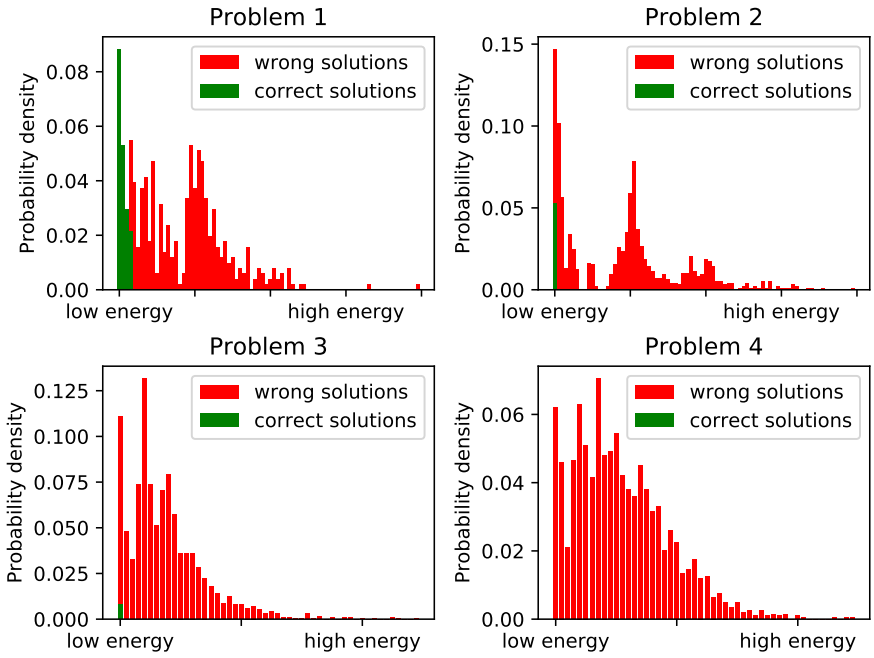

Fig. 3: Histograms of results for the four types of workflow problems tested. The x axis represent values of energies equal to the value of minimized objective function (1). The y axis represents the probability density. For Problems 1-3 the correct solutions found are shown on the left-hand side of the spectrum.

7 Conclusions and future work

In this paper we showed that it is possible to translate the workflow scheduling problem into a QUBO problem, execute it on a quantum annealer and achieve not only correct, but also globally optimal results for some of the analyzed problem instances. However, the presented method of adapting the scheduling challenge for D-Wave significantly increases the size of the problem. For example, the 18-binary variable problem required each of 39 QUBO problem variables to be represented by 11 physical qubits; thus the initial problem with a solution space size of $= 3^6 = 729$ ($m^t, m = 3, t = 6$, see Section 4) was converted into a problem with size $2^{429} \sim 10^{143}$ (QUBO problem with 39 variables, 11 qubits each, $39 \cdot 11 = 429$). For such a large QUBO it is difficult for the quantum annealer to find the lowest energy solution. Therefore, in the future we will focus on solutions such as domain wall encoding [7] to reduce the number of binary variables.

However, it is worth noting that for larger problems the brute force method would no longer be usable because of its exponential complexity. This leaves a

space for experimenting with a quantum annealer for larger instances, as the annealing process is very fast.

To sum up, this work proved that it is possible to solve workflow scheduling problems with the use of currently existing quantum annealers. Future work might involve finding a better translation of the problem to a QUBO problem, making the Q matrix more sparse and using minor-embedding heuristics. Additionally, the problem solution could be tested on the Pegasus machine [12] upon its release. It may also be possible to divide the problem into smaller pieces – this means dividing the original problem or dividing the QUBO problem (or even using both methods in parallel) with tools such as D-Wave QBSolv [10]. Another interesting direction of research would be to compare the performance of the presented problem on the D-Wave 2000Q machine with a non-quantum Fujitsu digital annealer [28] as both machines are designed to solve problems with a similar approach but different hardware.

Acknowledgements. The research presented in this paper has been partially supported by the National Science Centre, Poland, Grant no. 2016/21/B/ST6/01497. The authors also thank Piotr Gawron, Bartłomiej Gardas, Andy Mason and Piotr Nowakowski for helpful remarks.

References

1. Abbott, B.P., Abbott, R., Abbott, T., et al.: Observation of gravitational waves from a binary black hole merger. Physical review letters **116**(6), 061102 (2016)
2. Arabnejad, H., Barbosa, J.G., Prodan, R.: Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. Future Generation Computer Systems **55**, 29–40 (2016)
3. Baldini, I., Castro, P., Chang, K., Cheng, P., et al.: Serverless computing: Current trends and open problems. In: Research Advances in Cloud Computing, pp. 1–20. Springer (2017)
4. Berriman, G., Good, J., Laity, A., et al.: Montage: A grid enabled image mosaic service for the national virtual observatory. In: Astronomical Data Analysis Software and Systems (ADASS) XIII. vol. 314, p. 593 (2004)
5. Bian, Z., Chudak, F., Macready, W.G., Rose, G.: The Ising model: teaching an old problem new tricks. D-wave systems **2** (2010)
6. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. arXiv preprint arXiv:1406.2741 (2014)
7. Chancellor, N.: Domain wall encoding of discrete variables for quantum annealing and QAOA. Quantum Science and Technology **4**(4) (2019)
8. Chapuis, G., Djidjev, H., Hahn, G., Rizk, G.: Finding maximum cliques on the d-wave quantum annealer. Journal of Signal Processing Systems **91**(3), 363–377
9. Coffman, E.G., Bruno, J.L.: Computer and job-shop scheduling theory. John Wiley & Sons (1976)
10. D-Wave Systems: D-Wave Initiates Open Quantum Software Environment (2017), https://www.dwavesys.com/press-releases/d-wave-initiates-open-quantum-software-environment
11. D-Wave Systems Inc.: D'wave problem solving handbook. https://docs.dwavesys.com/docs/latest/_downloads/09-1171A-A_Developer_Guide_Problem_Solving_Handbook.pdf

12. Dattani, N., Szalay, S., Chancellor, N.: Pegasus: The second connectivity graph for large-scale quantum annealing hardware. arXiv preprint arXiv:1901.07636 (2019)
13. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5), 528–540 (2009)
14. Glover, F., Kochenberger, G., Du, Y.: A Tutorial on Formulating and Using QUBO Models. arXiv preprint arXiv:1811.11538 (2018)
15. Graham, R.L.: Bounds for certain multiprocessing anomalies. The Bell System Technical Journal **45**(9), 1563–1581 (Nov 1966). https://doi.org/10.1002/j.1538-7305.1966.tb01709.x
16. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 212–219. STOC '96 (1996)
17. Jałowiecki, K., Więckowski, A., Gawron, P., Gardas, B.: Parallel in time dynamics with quantum annealers. arXiv preprint arXiv:1909.0429
18. Jordan, S.: Quantum algorithms zoo web page. https://quantumalgorithmzoo.org/
19. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972), https://doi.org/10.1007/978-1-4684-2001-2_9
20. Kijak, J., Martyna, P., Pawlik, M., Balis, B., Malawski, M.: Challenges for scheduling scientific workflows on cloud functions. In: 11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018. pp. 460–467. IEEE Computer Society (2018). https://doi.org/10.1109/CLOUD.2018.00065
21. Lewis, M., Glover, F.: Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis. Networks **70**(2), 79–97 (2017)
22. Lucas, A.: Ising formulations of many NP problems. Frontiers in Physics **2**, 5 (2014). https://doi.org/10.3389/fphy.2014.00005, https://www.frontiersin.org/article/10.3389/fphy.2014.00005
23. Maechling, P., Chalupsky, H., Dougherty, M., et al.: Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment. ACM SIGMOD Record **34**(3), 24–30 (2005)
24. Pawlik, M., Figiela, K., Malawski, M.: Performance considerations on execution of large scale workflow applications on cloud functions. arXiv preprint arXiv:1909.03555 (2019)
25. Pelofske, E., Hahn, G., Djidjev, H.: Solving Large Maximum Clique Problems on a Quantum Annealer. In: Feld, S., Linnhoff-Popien, C. (eds.) Quantum Technology and Optimization Problems. pp. 123–135. Springer International Publishing, Cham
26. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
27. Spillner, J., Mateos, C., Monge, D.A.: FaaSter, Better, Cheaper: The Prospect of Serverless Scientific Computing and HPC. In: Latin American High Performance Computing Conference. pp. 154–168. Springer (2017)
28. Tsukamoto, S., Takatsu, M., Matsubara, S., Tamura, H.: An accelerator architecture for combinatorial optimization problems. Fujitsu Sci. Tech. J **53**(5), 8–13 (2017)
29. Venturelli, D., M.D.J.J., Rojo, G.: Quantum Annealing Implementation of Job-Shop Scheduling. arXiv:1506.08479 (2015)
30. Zhou, A.C., He, B., Liu, C.: Monetary cost optimizations for hosting workflow-as-a-service in iaas clouds. IEEE Transactions on Cloud Computing **4**(1), 34–48 (2015)