Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

Instytut Informatyki

PRACA DYPLOMOWA

Selected aspects of adapting the D-Wave annealer solutions to Workflow Management Systems

Wybrane aspekty dostosowania rozwiązań dla wyżarzacza D-Wave do systemów zarządzania aplikacjami typu workflow

Autor: **Mateusz Hurbol**
Kierunek studiów: **Informatyka**
Opiekun pracy: **dr inż. Katarzyna Rycerz**

Kraków, 2022

Abstract

This thesis presents the problem of optimizing the execution of workflow problems in order to reduce the costs of research and scientific calculations. For this purpose, the family of D-Wave solutions built around quantum annealing devices was used. The presented solutions uses well known workflow execution format provided by WfCommons framework. In order to perform the experiments, a hybrid (quantum-classical) CQM optimizer was used to optimize functions in the form of constrained binary models. The results from the CQM were compared with the results returned from the classical optimizer Gurobi optimizer. The conducted experiments confirm that it is possible to optimize the actual workflow type problems using solutions based on quantum annealing, while at this point the optimization process using the above-mentioned annealer is much slower than when using the Gurobi optimizer.

Streszczenie

W poniższej pracy przedstawiony jest problem optymalizacji wykonania problemów typu "workflow" (naukowy przepływ pracy) w celu zmniejszenia kosztów przeprowadzania badań i obliczeń naukowych. W tym celu została wykorzystana rodzina rozwiązań firmy D-Wave zbudowana wokół wyżarzaczy kwantowych. W pracy problemy zostały przedstawione w formacie WfFormat pochodzącym z projektu WfCommons mającego na celu integracje rozwiązań dla systemów typu workflow. W pracy w celu wykonania eksperymentów został użyty hybrydowy (kwantowo-klasyczny) optymalizator CQM pozwalający minimalizować funkcje w postaci ograniczonych modeli binarnych. Rezultaty pochodzące z CQM zostały porównane z rezultatami zwróconymi przez klasyczny optymalizator Gurobi. Przeprowadzone eksperymenty potwierdzają że jest możliwe zoptymalizowanie faktycznych problemów typu "Workflow" przy użyciu rozwiązań opartych na wyżarzaczu kwantowym, natomiast w tym momencie proces optymalizacji przy użyciu ww. wyżarzacza jest znacznie wolniejszy niż przy użyciu optymalizatora Gurobi.

Acknowledgments

I would like to thank my thesis supervisor dr inż. Katarzyna Rycerz for her ongoing help with the thesis, patience, and for introducing me to the world of quantum computing during her classes and later while writing this thesis. I would also want to thank Justyna Zawalska for introducing me to the world of workflows and providing some initial resources on how to work with them.

Contents

1 Introduction

Quantum computing, a new and rapidly developing branch of computer science, could potentially lead to faster algorithms allowing us to solve many problems science currently does not have enough conventional computing power [3]. Although this field is still in its infancy, as the hardware required for quantum computation is still highly experimental (but viable commercial products are brought to market), We can find many well-researched algorithms that are being run on quantum machines [24].

1.1 Background

1.1.1 Quantum computing directions

Currently, there are two main directions in which researchers and engineers are taking the quantum computers: sing universal quantum gates and quantum annealing. The quantum gates are similar to classical gates known from Boolean logic when you can create an algorithm by chaining basic types of gates into a more complex circuit. Quantum annealing on the other hand tries to minimize the energy of a function provided as an input for the computer. Quantum annealers are only capable of solving a subset of problems that gate computers can. On the other hand, quantum annealers produced by the D-Wave company[1] have a much larger number of qubits (5760 qubits released in 2020 [4]) compared to IBM Eagle (127 qubits released November 2021[2]).

1.1.2 Quantum annealers

Usage of quantum computers for running quantum algorithms allows us to solve many problems in much less time. Most known examples of quantum algorithms include Grover's Algorithm [13], which allows us to search an unsorted database of size N for a

[1] https://www.dwavesys.com/
[2] https://research.ibm.com/blog/eagle-quantum-processor-performance

specific key in time $O\left(\sqrt{N}\right)$ and also Shor's Algorithm [29] that allows us very quickly to find prime factors of a number which in turn can be used to break many modern encryption schemes. However, those algorithms were designed to run on the gate-based quantum computers and as such are not well suited to run on quantum annealers.

Quantum annealers are a type of computational device using a quantum effect that is capable of solving a very specific family of problems and are not general quantum computers. They are only capable of solving problems reducible to Quadratic Unconstrained Binary Optimization (QUBO) [22]. Quantum Annealers are a subtype of a much more powerful family of quantum devices namely the adiabatic quantum computers which are equivalent in capabilities to full-blown gate computer [1]. However quantum annealers seem to provide wide enough problem space to seem useful, as many problems can be transformed into QUBO form [12]. We observe that more and more problems are being solved using D-Wave infrastructure e.g. Stock Portfolio Optimisation [23].

1.1.3 Workflow management systems

Scientific workflows are currently used as a way to describe research processes. Workflow usually consists of tasks or jobs that need to be done, nodes or machines that are capable of performing jobs, and connections between them. Workflows are connected in the Directed Acyclic Graph (DAG) where each task can have some dependent jobs and be a dependency of another one [26]. Recently, there has been a push to standardize workflow formats to enable easy interoperability of workflow management systems (WMS). The result of such efforts is WfCommons standard [6]. WfCommons is an open-source format data format that is represented as a JSON file. Its simple structure allows for ease of generation and validation of workflows. WfCommons contains a database of real and synthetic workflows[3] on their GitHub repository.

1.1.4 Job scheduling on quantum annealer

Optimization problems that can be transformed to a form that can be inputted to quantum annealer could massively benefit from the additional power of quantum

[3] https://github.com/wfcommons/pegasus-instances

processing. One of such problems is scientific workflow scheduling with a deadline, which is the subject of work and analysis in this thesis.

Recently there has been some progress regarding job scheduling on quantum annealers. The two master's theses by Dawid Tomasiewicz [32] and Michał Rudnik [27] are focused on the research technologies provided by D-Wave company. Dawid focused on the solution using pure quantum annealing directly on the D-Wave 2000Q Quantum computer. On the other hand, Michał researched the Leap hybrid solver, which besides quantum annealing uses different heuristics and classical algorithms to find the optimal solution for a given function. With hybrid solvers, it was possible to solve some very simple and basic job scheduling problems.

1.2 Motivation

As the scientific calculations can be costly and take a long time it would be beneficial to provide a way to take existing workflow and given the time limit, and return the least costly configuration of hardware resources, that still meet the deadline. We would like to check the viability of quantum optimization for such problems using the realistic scientific workflows on the D-Wave quantum infrastructure. The realistic scientific workflows will be defined in WfCommons own WfFormat which are supported by many workflow management systems, making this optimizer more universal.

1.3 Goals

The main goal is to assess the viability of existing quantum annealers for solving real-world problems and to create a complete solution for optimizing workflows using a D-Wave Quantum Annealer.

1.4 Methodology

The work started with an analysis of both the quantum annealer and the selected workflow format (WfCommons), then in the next steps, the work proceeded with the following steps.

- Transforming the problem as the quantum unconstrained binary optimization (QUBO) problem,
- Creating s translation layer from the workflow data into a form acceptable by D-Wave API,

- Analyzing the results coming from the quantum annealer,
- Exploring the hybrid solver provided by D-Wave,
- Comparing the created solution and run times with a reference solution (Gurobi solver).

1.5 Structure of work

This thesis consists of six chapters.

- 1-st chapter is an introduction to the thesis and contains short overview of the current state of workflow optimization on quantum computers,
- 2-nd chapter is an introduction to the D-Wave quantum annealers and mathematical programming,
- 3-rd chapter describes the scientific workflows in general and WfCommons project,
- 4-th chapter describes the transformation of the input workflow into an optimization problem understood by the D-Wave optimizer and presents the algorithm used by the software created for this thesis "Workflow optimizer",
- 5-th chapter contains experiment performed on the "Workflow optimizer" with the results,
- 6-th chapter is the summary of the results and it contains ideas for possible future improvements.

2 Theoretical introduction to D-Wave annealer

In this chapter, we provide the basic theoretical background required for understanding the process quantum annealers use for computation [16]. Then, we describe the functionality provided by D-Wave for solving optimization problems.

2.1 Simulated annealing

Annealing is a metallurgical process of heating the metal and then slowly cooling the material to achieve the internal balance of forces and minimize the potential energy of the system. This process usually is used to make the metal more malleable and to reduce its hardness. Mmetallurgical annealing is a basis for simulated annealing [15] a probabilistic numerical method for finding the global minimum/maximum of a given function. This is achieved by defining the examined function as energy of the simulation. In the process of finding the global minimum, we start by randomly selecting solution s from the function domain. During the whole process, the algorithm maintains the parameter T called temperature which is slowly decreased in each step of the simulation. Temperature is used in each step to determine the radius of the domain search space for the next guess s'. The new guess is accepted according to the probability function that usually is defined as follows

$$p = e^{\frac{\overbrace{E(s)}^{\text{current state}} - \overbrace{E(s')}^{\text{next guess}}}{T}} \tag{2.1}$$

which is analogous to the Boltzmann distribution when dealing with metallurgical annealing and real temperatures. This process at first have high variability of the solution to move out of the local minimums/maximums and then the guess is getting more and more refined. The most important parameter for the process of simulated annealing is the amount that the temperature is reduced in each iteration. Too large

value would cause the likelihood of the algorithm missing the global minimum much higher while too small would make the execution take much longer time.

2.2 Quantum computing

In classical computing, the most fundamental unit of information is a single bit that can have one of two states **0** or **1**

$$b = 0 \vee b = 1. \tag{2.2}$$

In the quantum world, the single value can be at the same time either of the states with different probabilities. This quantum version of a bit is called a qubit [21]. Usually, the qubit is denoted as

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.3}$$

Wwere

$$\begin{aligned} \alpha^2 + \beta^2 &= 1 \\ \alpha, \beta &\in \mathbb{C}, \end{aligned} \tag{2.4}$$

The values of states can be represented as two-element vectors

$$\begin{aligned} |0\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ |1\rangle &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{aligned} \tag{2.5}$$

Representing qubit states in such a manner allows us to easily create multi-qubit states by using tensor product

$$|\psi\rangle \otimes |\theta\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a * c \\ a * d \\ b * c \\ b * d \end{bmatrix} \tag{2.6}$$

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1*0 \\ 1*1 \\ 0*0 \\ 0*1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle \tag{2.7}$$

This representation has a particular advantage. For each pure state the index on which 1 is located in the vector is the binary value of the state [21].

2.2.1 Operators

In boolean logic defines sets of operations that when chained together in specific ways can create all the possible operations on bits. One of such sets of operation include ***And***, ***Or***, ***Not***. You can achive this by using just a single operation ***Nor*** or ***Nand***. The equivalent transformation in the quantum world is called Pauli's matrices [21] which are unitary matrices meaning that the length of the vector after applying the matrix is not changed. Together with the 2x2 identity matrix Pauli's matrices create a spanning basis of the linear space of all quantum states.

$$I = \sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.8}$$

$$X = \sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.9}$$

$$Y = \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.10}$$

$$Z = \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.11}$$

Just like with values the Matrices can be extended to use multiple qubits by using tensor product

$$I \otimes X = \begin{bmatrix} 1*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & 0*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ 0*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & 1*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.12}$$

2.2.2 Measurment

The quantum state by itself is not that useful. In order to get the information we must measure it to get one classical bit out of each qubit. Unfortunately, the measurement operation destroys the quantum superposition collapsing the state

$$|\psi\rangle = \alpha * |0\rangle + \beta * |1\rangle \tag{2.13}$$

into either $|0\rangle$ or $|1\rangle$ with the probabilities $|\alpha|^2$ and $|\beta|^2$ respectively.

2.3 Quantum annealing

The basic building block of the D-Wave quantum computer is a superconducting loop that represents a single qubit. This qubit can represent a value of $|1\rangle$ or $|0\rangle$ corresponding to the opposite direction of the current flow in the superconducting loop [9]. However qubit also has the ability to represent a state that is the superposition of both $|1\rangle$ and $|0\rangle$. Each qubit in its default state is in equal superposition meaning it has the same probability of being in either state when measured. However this can be changed by carefully applying bias the qubit by using a magnetic field that encourages current flow in one direction and inhibits it in the other. The qubits in D-Wave QPUs

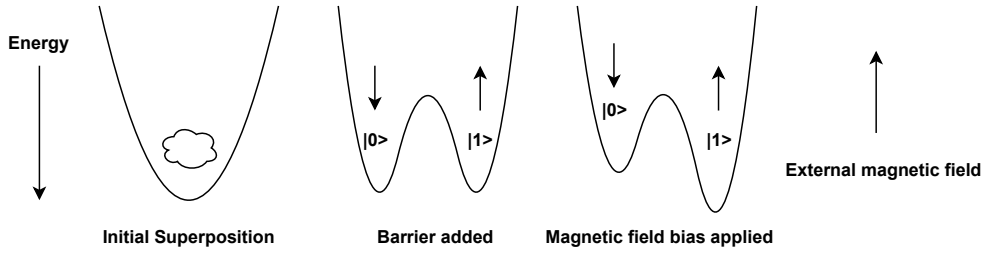

Figure 2.1: Schematic of energy levels for states of single qubit.

are arranged in lattices where some qubits can be connected using couplers, which are devices designed to force qubits to end up in the same final state or the opposite force them to have a different state. This device uses quantum entanglement as a mechanism to provide this functionality [9].

2.3.1 Hamiltonian of D-Wave annealer

Hamilton Operator or Hamiltonian $\mathcal{H}$ in quantum mechanics is a measure of a system's total energy including both kinetic and potential energies. This is a value that D-Wave quantum annealers try to minimize [9], and is given by the expression:

$$\mathcal{H}(s) = \underbrace{-\frac{A(s)}{2}\left(\sum_i \sigma_1^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i\sigma_3^{(i)} + \sum_{i>j} J_{i,j}\sigma_3^{(i)}\sigma_3^{(j)}\right)}_{\text{Final Hamiltonian}}, \tag{2.14}$$

where

$$\begin{aligned} A(s) &- \text{Tunneling energy and at s} \\ B(s) &- \text{Hamiltonian energy of problem at s} \\ h_i &- \text{Bias strength for quibt i} \\ J_{i,j} &- \text{Coupling strength for qubits i,j.} \end{aligned}$$

The s parameter is usually defined as $s = \frac{t}{t_{max}}$ where t is current time and t_{max} is total time for annealing. At the end of the annealing the $A(s) >> B(s)$

2.3.2 Execution on D-Wave computer

At the start of the process, the initial state of the system is the eigenstate (a state with a defined energy value) that has the lowest Hamiltonian energy provided and has all of the energy in the system. Then, as it slowly transitions its wave function mostly stays at the lowest energy region and in the end it is the most probable to be found at the global minimum [16]. The process of annealing goes as follows:

1. The quantum computer I/O system translates the problem data into magnetic field strength and qubits pairs to entangle,
2. The qubits are initialized to their base eigenstate with the lowest energy,
3. The qubits are coupled with their partners,
4. The energetic barrier separating future states $|0\rangle$ and $|1\rangle$ is slowly raised,
5. The biases are slowly applied to the qubit lattice as the system transitions from the initial Hamiltonian A to problem the Hamiltonian B,

6. The qubits are measured to provide with some probability low energy solution to the given function.

2.3.3 Noise problem

The measurement of a quantum variable to read its value is an inherently noisy process and can often provide the measurement apparatus with a theoretically unexpected value. To overcome this issue, the quantum computation can be performed in multiple instances or be repeated. This can be used to enhance the signal and reduce noise. The D-Wave computers run the same calculation on multiple physical qubits that are presented to the end-user as a single logical qubit [8]. This reduces the random noise for each logical qubit but lowers the number of qubits that are available for usage.

2.3.4 Quadratic unconstrained binary optimization

Problems submitted to D-Wave quantum computers usually are translated into a form called Quadratic Unconstrained Binary Optimization (QUBO) problem [17], which is a common form used for many combinatorial problems. This form is quite easy to map to quantum annealer's physical layout and therefore quite suitable for an input format for the D-Wave quantum annealer.

2.3.5 Penalty model for QUBO

The problem in the form of QUBO as the name suggest is only useful for minimizing the energy of the problem, and any additional constraints must be encoded into QUBO itself. This can be achieved with the penalty model that adds additional data to QUBO in order to force the solutions that violate the constraints on the problem to have higher energy than the ones that correctly satisfy the constraints [32][27]. This approach however greatly increases the amount of variables needed when constraints are inequalities [27] or the value of the constraint is large compared to the number of variables in base function. The big issue with penalty model is fine tuning the penalty coefficient for each problem as too small value can lead to many of the returned solutions violating the constraints, while too large value could lead the annealer to focus on satisfying the constraints and failing on optimizing the base function.

2.3.6 Constrained Quadratic Model

Recently D-Wave has introduced a new hybrid (using a mix of either classical and quantum algorithms and heuristics) solver that in addition to accepting problems in

the QUBO form also accepts the problems in the Constrained Quadratic Model (CQM) form [20]. The CQM hybrid solver that takes care of translating the problem from the constrained form into the one that can be run on the quantum computer. hybrid solvers is also responsible for splitting the problem into smaller parts that can be easily run on the quantum annealer. This new solver is much more capable of solving the problems with constraints than the unconstrained solver with constraints manually translated into penalties [30].

2.4 Classical alternative - Gurobi optimizer

In this thesis, the Gurobi optimizer [14] has been used as a classical alternative and a reference method for the D-Wave CQM hybrid solver. The Gurobi optimizer is a software capable of mathematical optimization (also called mathematical programming) that can be used for solving a wide range of different mathematical models, including the Quadratically Constrained Programming (QCP) models that are analogous to the D-Wave CQM solver input model.

2.5 Conclusion

The processes of simulated and quantum annealing described in this chapter can be used to potentially solve various optimization problems. The hybrid solvers provided by the D-Wave systems are used for the **Workflow Optimizer**. The reference implementation is using the Gurobi optimizer.

3 Scientific workflow management systems and WfCommons

In this chapter the first section provides the basic presentation of workflow management systems is provided. The second section describes the WfCommons format that is used for this thesis as a way to read and store optimized workflows.

3.1 Scientific workflow management systems

Workflow management systems (WMS) have been used in various areas of science as a way to easily share and reproduce scientific research and processes. Workflows have become a standard way of describing a lot of applications and data-flows. Workflows are usually represented as a Directed Acyclic Graph (DAG) of task and data dependencies between various programs, machines, and scripts. Example data-flow in form of DAG is presented in Fig. 3.1.

There exists a lot of systems utilizing workflow management software to produce valuable results and insights like:

- "California Earthquake Center Community Modeling Environment" uses workflows to process data to minimize the effects of the devastating earthquakes in California [18],
- "The Event Horizon Telescope" data processing uses workflow pipelines to process data from different radio-telescopes to produce the images of distant black hole [31],
- "The International Genome Sample Resource" uses workflows to process and collect genetic data into the database of human DNA as part of 1000 Genome project [11].

Over the years many Diffrent WMS have been designed like general perpouse Apache Airavata [19] more recent MakeFlow [2] or Pegasus [10]. There are also some

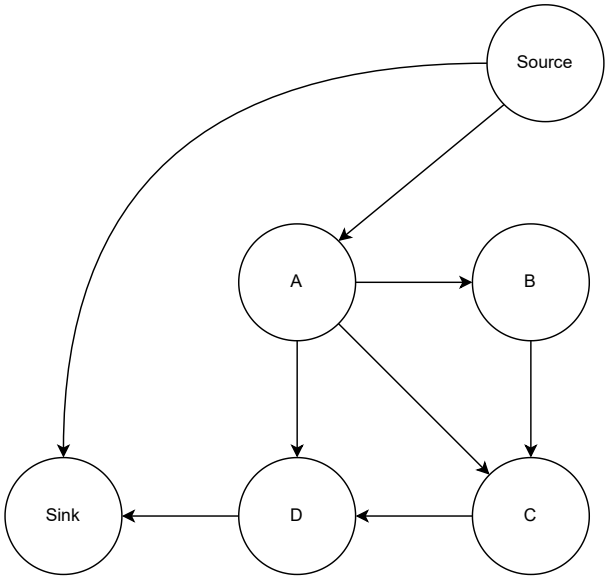

Figure 3.1: Example of directed acyclic graph with 6 nodes and 8 edges. The source and the sink of the graph have been marked.

more specialized systems like GenePattern [25] used to access hundreds of genomic analysis programs.

3.2 WfCommons

The WfCommons project is an initiative enabling scientific workflow research and development providing many tools for generating workflow recipes or totally synthetic but realistically looking instances of workflow problems [7]. The tools provided by the initiative can be used for researching new algorithms and approaches for finding new and efficient ways of running and improving upon existing workflows. WfCommons project targets the need for more and more complex workflows in a modern distributed cloud environment. The cloud presents new challenges for running scientific computations but also provides new opportunities for optimally using resources of various scales and prices. At the time of writing this thesis, the basic workflow development life cycle is presented in Fig. 3.2.

The different elements of the WfCommons project [7] as seen in Fig. 3.2 are described below:

- **WfInstances** - a collection of open access production workflow executions from

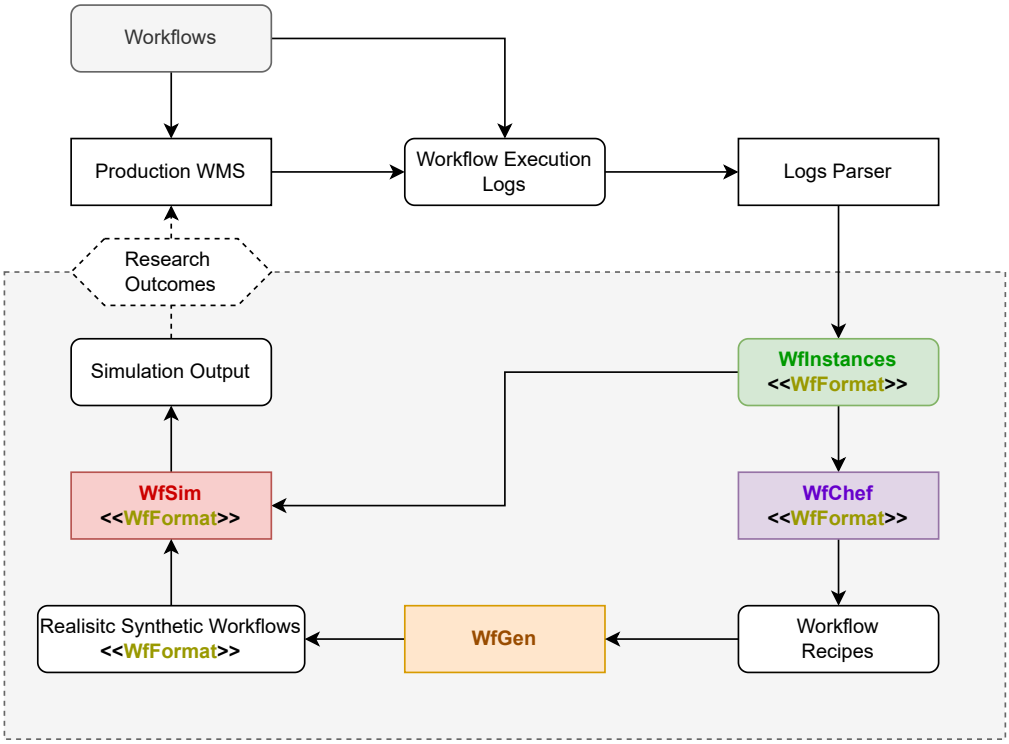

Figure 3.2: Workflow research life cycle process integrating the components of the WfCommons project [6]

programs from various scientific disciplines,

- **WfChef** - a framework that automates the construction of synthetic workflow generators. It can create a workflow generator based on a set of input workflow instances from any scientific discipline [5],

- **WfGen** - a workflow generator that uses recipes of workflows and creates different synthetic workflows based on the distribution of workflow parameters like input/output files runtime, and file sizes. The generated workflow is created in WfFormat,

- **WfSim** - a workflow execution simulator that can be used to test different behaviors of the workflows as a part of the scientific workflow research process. It was designed to replace the need for running the real-world scientific experiments using the still in the development workflows.

3.2.1 WfFormat

WfFormat is a common format used by the WfCommons tools and some external WMSs. A workflow in this format is a JSON file that follows the schema that can be found on GitHub repository[1] together with the schema validator script. The current version of this schema is 1.3. The main components of this format are the "machines" field which contains the definition of machines that the workflow was run on and the jobs for and the "tasks" field which contains all the jobs that are part of this workflow. Each task contains information on which machine it was being run and the runtime of the job and how much memory is used. Each machine contains information about machine CPU speed and available memory. That information has been used as a basis for the **Workflow Optimizer** described later (in Chapter 4 of this thesis).

3.3 Workflow cost and optimization

Each scientific workflow is, in fact, a set of instructions on how to correctly execute some collection of tasks to arrive at the expected result. When a new workflow is created during the process of some research it usually will be created and run on machines available to the researcher. The approach to assigning the task to machines may be random or done in a not optimal way. Currently, the approach of running the computations in cloud environments or on-demand highly scalable computer clusters is gaining more attraction. The cost of running a task on the cloud is proportional to what has been ordered and provisioned [28]. This fact suggests that optimizing the assignment of jobs to machines can potentially reduce the monetary cost of running scientific workflow.

3.3.1 Structure of the workflow

The goal of the workflow problem is to find the optimal **allocation of computing machines** to each of the tasks in the problem's DAG (example in Fig. 3.1) in terms of the cost, considering their causal relationship. The problem also contains **the deadline** value before each task must finish processing, described in more detail in Subsection 3.3.3. The value of **processing time** and **cost** for each job and machine are being calculated based on the data from workflow where speed and price of each machine is considered. This calculation is better described in Subsection 3.3.2 and Section 4.3.

[1] https://github.com/wfcommons/wfformat

3.3.2 Data sources for optimization

The optimization problem we wanted to solve is based on the problem that can be found in theses of Dawid Tomasiewicz [32] and Michał Rudnik [27]. In both works, the optimization problem that was being run on the D-Wave solvers has their values of cost and runtime of each task artificially defined. Those values are required for the optimization problem. In this work, we wanted to calculate those values from the data included in WfCommons workflow files. To calculate the cost of using each machine additional information about the machine price must be provided. The optimization problem requires the value of the deadline which is described in more detail in Subsection 3.3.3. This problem is only one of many in the similar category and the solver can be easily modified for slight variations of the problem.

3.3.3 Workflow runtime and Deadline

While using only the cheapest option on the cloud is a possibility, the amount of time needed to execute a workflow will be quite larger than the amount if we used faster machines. Running all of the workflow task on the least expensive hardware can be a blocking factor for this strategy. This is a reason why for this optimization problem we introduce a deadline D which is workflow execution time limit that we are willing to accept for a given workflow. This means that we require that *all* tasks in the workflow finish before the deadline. This can potentially invalidate some of the cheapest machine assignments when slower machines cannot execute the workflow before the deadline.

3.4 Conclusion

In this chapter, the brief overview of the scientific workflow management systems had been provided. Additionally introduction to the WfCommons project has been presented. The chapter also contained the exact workflow optimization problem we wanted to solve had been presented.

4 Workflow optimizer - system architecture

This chapter presents a **Workflow Optimizer** system with support for optimization of user-provided workflow definition in WfCommons format. The result of this optimization can be then used to potentially reduce hardware costs for the system.

4.1 Architecture Overview

This section describes the data-flow and functionalities of the **Workflow Optimizer** application. The user of the **Workflow Optimizer** is required to provide the workflow

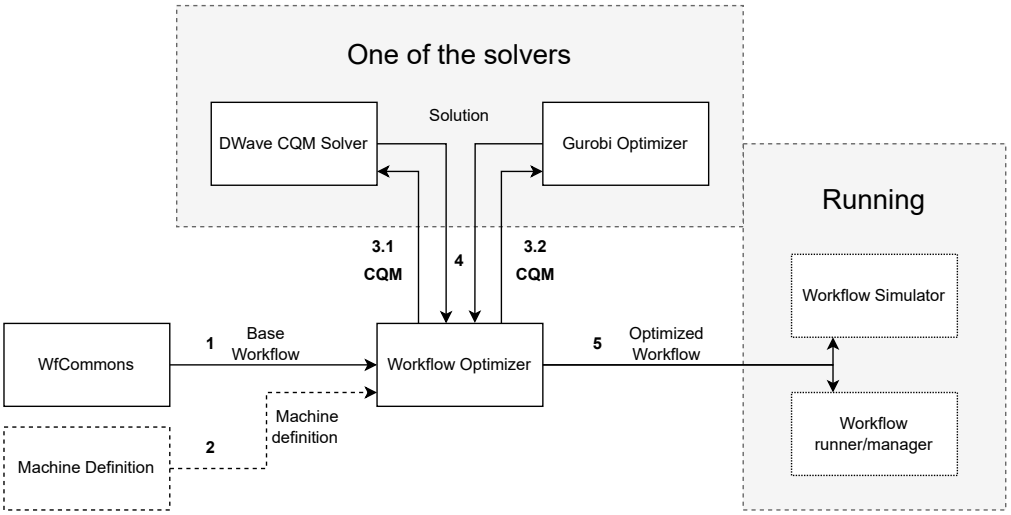

Figure 4.1: Dataflow using the **Workflow Optimizer**

to be processed (Step 1 in Fig. 4.1). User can optionally provide custom machine definition that will override the machines defined in the source workflow (Step 2). This data is then transformed into a Constrained Quadratic Model for Job scheduling with deadline problems. Depending on the user's choice it is then fed into either the

D-Wave hybrid CQM solver (Step 3.1) or the Gurobi optimizer instance (Step 3.2). The response from the solver is a list of potentially binary variable values that could be an optimal solution for the given CQM (Step 4). The Application then checks for each solution if the constraints from CQM are satisfied and then returns the optimal one from the set of correct solutions (Step 5). The energy returned by the D-Wave hybrid solver is the energy of the minimized function i.e. it does not include any potential penalties that constraints could have generated.

4.2 Input data

The following input data are needed for the system to optimize the workflow

- Deadline - provided by the user (is used to generate constraints)
- List of jobs - It is taken from WfCommons description of a workflow, the most important value is the job runtime and reference to the machine on which the job was run. The secondary data from this source is memory used by the job (provides f_j c_j and $t_{0,j}$ used in Equation 4.1 and mem_j used in Equation 4.3),
- List of machines - It can be taken from the workflow, as well as provided separately. It contains information about processor frequency and core count as well as available memory on the machine (provides f_m and c_m used in Equation 4.2),
- List of machine prices - It is taken from workflow with the WfFormat extension (see Section 4.5), or provided separately together with machine list. The machine prices need to be defined in the same source as machines. It contains prices for 1 unit of processing time on each machine (provides d_m and δ_m used in Equation 4.3).

The details about the input data can be found in the WfCommons schema in GitHub repository [1].

4.3 Transforming the problem into CQM

Given the input workflow the application proceeds with normalizing the data by estimating the work that is needed to complete each job. The process of calculating cost and runtime values described in this section is written in psudocode in Listing 1.

[1] https://github.com/wfcommons/wfformat/blob/master/wfcommons-schema.json

```
INPUT -> jobs_list, machine_list
OUTPUT <- runtimes_matrix, cost_matrix

work_amount_list = []
for job in jobs_list:
    # calculate_work as in equation 4.1
    work_amount_list += [calculate_work(job)]

work_amount_list = normalize_work(work_list)

runtimes_matrix = []
cost_matrix = []
for machine in machine_list:
    for work_amount in work_amount_list:
        # calculate_runtime as in equation 4.2
        runtime = calculate_runtime(machine, work_amount)
        runtimes_matrix += [runtime]
        # calculate_cost as in equation 4.3
        cost_matrix += [calculate_cost(runtime, machine)]
```

Listing 1: Psudocode for calculating runtime and cost values for workflow.

$$\mathcal{W}_j \approx t_{0,j} * f_j * c_j, \tag{4.1}$$

where

$\mathcal{W}_j$ – Estimated amount of work required for the job j to finish

$t_{0,j}$ – Time measured for the job j to finish on machine m

f_j – CPU Frequency of machine m used by the job j

c_j – Number of CPU cores on the machine m used by the job j.

The estimated amount of work for each job are then normalized (the easiest job has work value of 1). Then for each (machine, job) pair we calculate the expected run time values

$$t(j,m) = \frac{\mathcal{W}_j}{f_m * c_m}, \tag{4.2}$$

where

$$\begin{aligned} \mathcal{W}_j &- \text{Estimated amount of work required for the job } j \text{ to finish} \\ t(j,m) &- \text{Time estimated for the job } j \text{ to finish on machine } m \\ f_m &- \text{CPU Frequency of machine } m \\ c_m &- \text{Number of CPU cores on the machine } m. \end{aligned}$$

With the estimated time for each job on each machine we calculate cost of running job j on machine m, the following cost function has been used in application.

$$f(j,m) = t(j,m) * p_m * (1 + \delta_m)^{max(0,\lceil mem_j \rceil - 1)}, \tag{4.3}$$

where

$$\begin{aligned} t(j,m) &- \text{Time estimated for the job } j \text{ to finish on machine } m \\ p_m &- \text{relative price of machine } m \text{ per unit of time} \\ \delta_m &- \text{value of "memory\_cost\_multiplayer" field for machine } m \\ mem_j &- \text{Memory required by job } j \text{ in GiB .} \end{aligned}$$

With those values it is possible to define the optimisation objective for the CQM given that for each (machine, job) pair we create a variable binary $x_{j,m}$ which decides if the job j will be run on the machine m

$$F = \sum_j \sum_m f(j,m) * x_{j,m} \tag{4.4}$$

As we require that each job must be run on exactly one machine we add a following constraint to the model

$$\forall_j \left[\left(\sum_m x_{j,m} \right) = 1 \right] \tag{4.5}$$

Another requirement of the problem is that all the jobs finish before a deadline D. As jobs can have parent jobs that must finish before the next starts we can represent the jobs as a DAG. By adding dummy jobs as the source and sink to this graph (by adding source as a parent to all the parentless jobs and adding all childless jobs as parent to the sink job) we can find set P of all the paths from source to sink. Then we

can formulate a constraint

$$\forall_{p \in P} \left[\left(\sum_{j \in p} \sum_{m} t(j,m) * x_{j,m} \right) <= D \right] \tag{4.6}$$

With those constraints defined the model is ready to be sent to either the Leap hybrid CQM solver or the Gurobi optimizer.

4.4 Transforming the solution back to WfCommons

After a solution has been returned by solver, it is being checked if all constraints are satisfied. Then the best solution is used to generate new assignment of jobs to machines and whole workflow can be printed to a json file.

4.5 WfCommons schema extension

In the effort to include the information regarding the cost of usage of particular machine into the calculation it has been decided that WfCommons schema used by the optimizer application has been extended with the following two fields for machine presented in Listing 2. If the fields are missing the default value is used (provided in Listing 2). The

```
{
    "workflow": {
        "machines": [{
            "price": 1,
            "memory_cost_multiplayer": 0,
        }],
}}
```

Listing 2: Additional fields in schema recognized by the optimizer.

application can also be provided with custom machine definition, and the custom data must be part of the WfFormat that is related to machines definition.

4.6 Implementation

The **Workflow Optimizer** has been created using Python 3.8.10, and is available on GitHub repository [2] under the MIT license.

4.7 Conclusion

In this chapter the architecture of the **Workflow Optimizer** and transformation process from input data to constrained quadratic model. The additional information required to optimize the workflow was described. The process described in this chapter is a basis for the experiments described in Chapter 5.

[2] https://github.com/matix522/workflows-dwave

5 Experiments and results

In this chapter we present the optimization experiments run with the system described in Chapter 4. In first section the assumptions used for all experiments are provided. Each subsequent section focuses on three different aspect of the researched problem:

- Correctness of the solution,
- Scalability of the model,
- Different deadline restrictions.

5.1 Assumptions and preconditions

Two different solvers have been used in the experiments. Local installation of the Gurobi optimizer[1] with Academic Licence and D-Wave's hybrid CQM solver in the Leap cloud[2]. The local machine running the Gurobi optimizer has been Dell Inspiron 15 7590 with Intel i7-9750H processor and 16 GB of RAM.

5.2 Experiment 1 - comparing D-Wave and Gurobi solvers

The main goal of the first experiment was to find if the solution returned by both the Gurobi optimizer and D-Wave CQM solver are equivalent solutions to a given problem. It is assumed that data shared between jobs can be accessed from all machine types with similar speed and therefore it is not included in cost calculations. The developer license used for running problems on the D-Wave CQM solver is limited. This fact, unfortunately, reduced the number of experiments that could be performed as there also is a minimal processing time required for each problem (5 seconds). If not stated otherwise the minimal amount of processing time (5 seconds) on the D-Wave solver has been used.

The secondary goal of the experiment was measuring the time required to get the solution using both approaches.

[1] https://www.gurobi.com/downloads/gurobi-optimizer-eula/
[2] https://cloud.dwavesys.com/leap/

5.2.1 Workflow to optimize

The workflow chosen for this experiment is ”SRA search” workflow coming from the WfCommons repository[3]. This workflow contains 22 separate tasks that were run on a single machine. The run time for various jobs ranges from 0.115 to 900 seconds. The task in this workflow are grouped into 3 layers, as shown in Fig. 5.1.

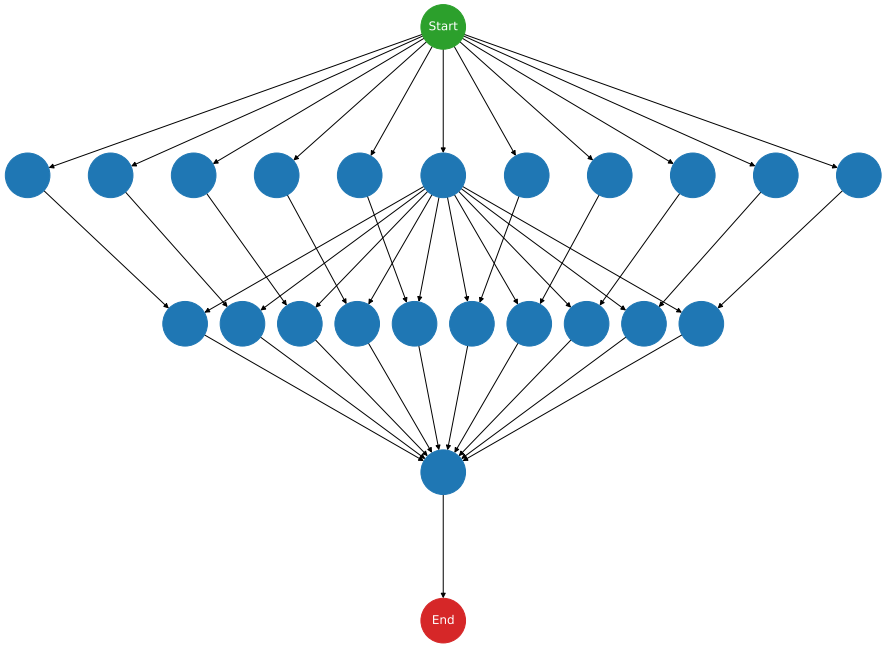

Figure 5.1: The graph representing dependencies between task in workflow from Experiment 1, with additional 2 nodes for sink (red) and source (green) of the graph added.

This figure include two artificial nodes representing start and finish of the workflow. This structure generate 20 different paths one can reach the sink from source.

5.2.2 Machine setup

The machines used for this experiment have properties defined in Table 5.1.
This setup is giving us ability to choose either the fast machine in cases where we

[3] https://raw.githubusercontent.com/wfcommons/pegasus-instances/master/srasearch/chameleon-cloud/srasearch-chameleon-10a-001.json

Table 5.1: Machines defined for first experiment

Name	CPU Count	CPU Speed	Price
OneCPU	1	1.0	1
FourCPU	4	1.0	5
SixteenCPU	16	1.0	25

would go over the deadline, but this advantage is going to increase the price and energy of the CQM solution. In this case we assume each machine has the same memory available for processing the tasks.

5.2.3 Deadline

The deadline for this experiment was chosen as a value that forces the optimizer to assign a mix of cheap and expensive machines. The value for this problem has been chosen as 4000.

5.2.4 Size of the model

Given the workflow with 22 tasks and 3 possible machines it generates 66 binary variables. Each task is required to have one machine assigned creating 22 constraints. The 20 possible paths generate 20 additional constraints to the objective.

5.2.5 Results

Both the Gurobi optimizer and the CQM solver returned the same best solution for this problem. The execution time was presented in Table 5.2.

Table 5.2: Time used for calculation in Experiment 1

Gurobi optimizer	D-Wave CQM solver
0.00742 [s]	14.6235 [s]

Unfortunately the quantum solution is slower by three orders of magnitude. This is partially caused by the D-Wave API which requires minimal time for processing the problem (in this case 5 seconds). D-Wave also provides multiple potential solutions to the problem. The discrepancy between the minimal processing time and actually measured value can be explained with iteration over the results from the solver, as the first solution is available after 5 seconds but each solution needed to be fetched from D-Wave cloud API and this took some time.

5.3 Experiment 2 - scalability

In this experiment the goal is to test system scalabilty with regard to size of the workflow. We used the D-Wave CQM solver for this experiment. It focuses on the distribution of the solutions energy for each test scenario.

5.3.1 Workflow to optimize

This experiment uses four workflow files. The details of the workflow in this experiment are included in Table 5.3.

Table 5.3: Job size and unique path count in each workflow.

Name	Task Count	Path Count
Genome_54	54	308
Genome_156	156	924
Genome_492	492	4368
Genome_902	902	8008

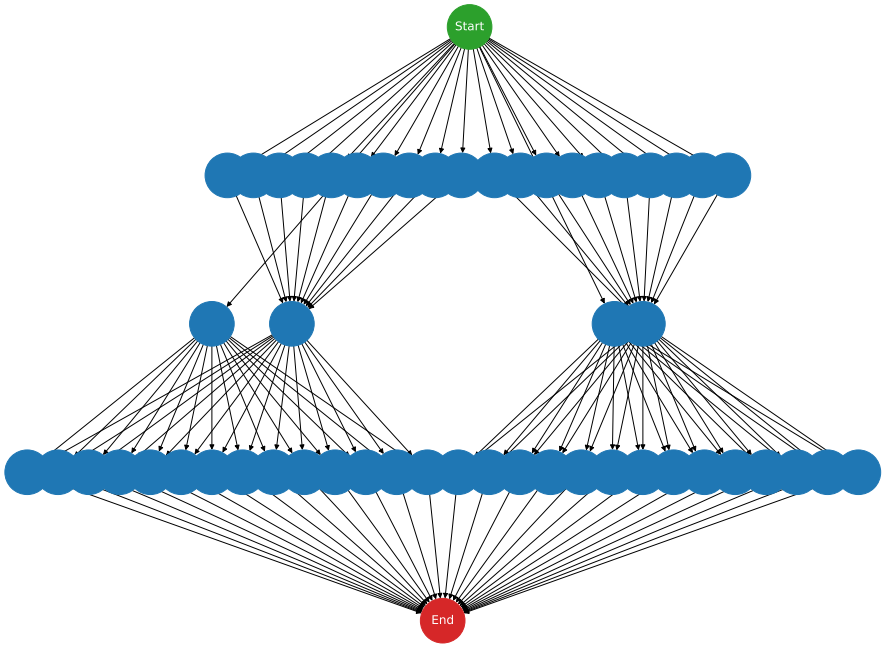

Figure 5.2: The graph representing dependencies between task in workflow "Genome_54" from Experiment 2, with additional 2 nodes for sink (red) and source (green) of the graph added.

Each workflow groups jobs into three layers of tasks of similar structure only scaled horizontally. The graph for Workflow "Genome_54" is presented in Fig. 5.2. Workflows "Genome_54", "Genome_156", "Genome_492", "Genome_902" are workflows related to 1000 Genome project a catalog for human genetic diversity. The WfCommons format workflow files can be found in the GitHub repository[4].

5.3.2 Machine setup

For this experiment two different machines groups have been prepared. The basic group assumes the machines have infinite memory and the details are presented in Table 5.4.

The second group is based on the machines available in Academic Computer Center

[4] https://github.com/wfcommons/pegasus-instances/tree/master/1000genome/chameleon-cloud

Table 5.4: Machines defined as a part of the basic test group

Name	CPU Count	CPU Speed	Price
OneCPU	1	1.0	1
FourCPU	4	1.0	5
SixteenCPU	16	1.0	25

’"Cyfronet"’ on AGH Univesity of Science and Technology. The details for the machines are presented in Table 5.5.

Table 5.5: Machines defined as a part of the Cyfronet group.

Name	Processor Count	Processor Speed	Price	Memory
AresCpu	1	1.6	0.08	8 [GiB]
AresGpu	100	1.6	5.0	8 [GiB]
PrometheusCpu	1	1.0	0.08	4 [GiB]
PrometheusGpu	100	1.0	5.0	4 [GiB]
ZeusCpu	1	0.25	0.08	2 [GiB]

5.3.3 Deadline

The deadline for this experiment was chosen as the value that forces the optimizer to assign mix of cheap and expensive machines. The values for this problem has presented in Table 5.6.

Table 5.6: Deadline chosen for each machine in Experiment 2.

x	basic_test	Cyfronet
Deadline	150	600

5.3.4 Size of the model

The constrained quadratic model for this experiment is significantly larger than for the Experiment 1. The number of binary variables for this model have been presented in Table 5.8

Table 5.7: Number of binary variables for each machines/workflow pair in Experiment 2.

x	basic_test	Cyfronet
Genome_54	162	270
Genome_156	468	780
Genome_492	1476	2460
Genome_902	2706	4510

Number of constraints required to be satisfied by the solution is given in Table 5.7.

Table 5.8: Number of constraints for each workflow in Experiment 2.

x	Meet deadline	Use one machine	Total
Genome_54	308	54	362
Genome_156	924	156	1080
Genome_492	4368	492	4860
Genome_902	8008	902	8910

5.3.5 Results

In this subsection the result of the experiment are covered. The analysis is split in two subsections considering results for each of the machine groups. The solution has been categorized by the its compliance with model constraint. The solutions on the histograms has been grouped int 3 categories:

- The red color represent the situation when the solution is violating the constraints for machine assignment, meaning that there is more than one machine assigned to a task or there is no machine assigned at all,
- The orange color represents the situation when solution is violating the constraints for meeting deadline on at least one path,
- The blue solutions are the one where all constraints are met.

The energy of the solution only takes into account the model objective, meaning constraints satisfaction does not change energy unlike the penalty model described in Subsection 2.3.5. Note that the charts in this section have variable scale on both axis.

Results for the basic_test machine group

Each workflow that has been optimized for "basic_test" had the solutions returned by the D-Wave solver presented in the form of a histogram.

In Fig. 5.3 representing the workflow "Genome_54", it can be seen that most of the solution returned by the solver have been put into a single bucket, this shows that the solver has high confidence for this problem solution. As most of the returned solutions are correct it can be assumed with relatively high confidence the best solution from this run is quite close to the optimal arrangement of machines.

In Fig. 5.4 representing the workflow "Genome_156", it can be seen that the correct solutions are a little more spread out but this can be explained by lower range of the maximal and minimal range of solutions which caused each bucket to contain a smaller range of solutions. We can also assume the same as in the previous case that the solver solutions have high confidence of being close to optimal.

In Fig. 5.5 representing the workflow "Genome_492", it can be seen that the resulta are much more spread out possibly indicating either different solver algorithm used

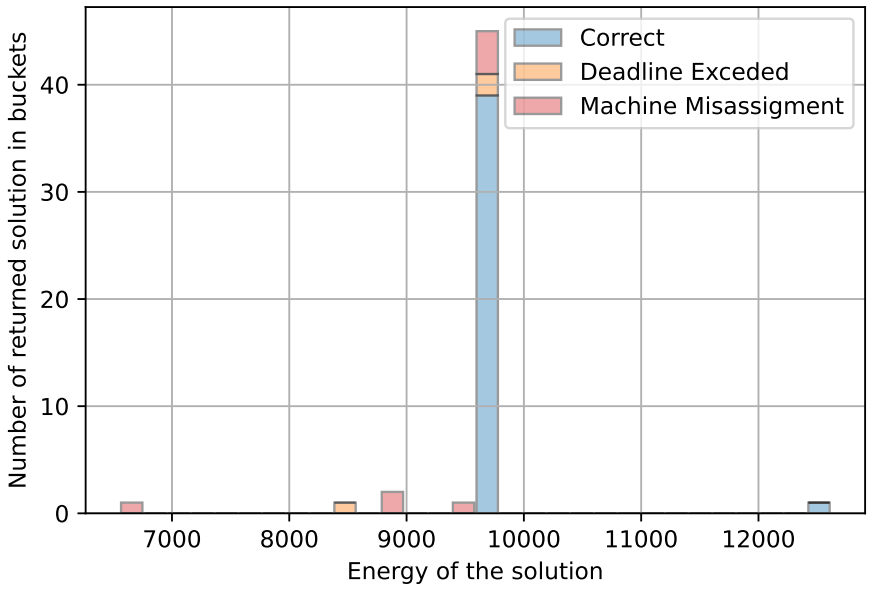

Figure 5.3: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_54 Workflow" and "basic_test" machines, with information about compliance with model constraints. (Gurobi energy value 9718.)

by D-Wave CQM or not enough time was given for processing the model of this size. The solver returns much more incorrect results, mostly using incorrect number of machines per task. We can observe the dividing line on the bucket starting at 27000 that splits the mainly correct results on the right to the incorrect results on the left, unfortunately the solver returned only 2 solutions in the best bucket, showing the solver did not find this solution with high confidence.

In Fig. 5.6 representing the workflow "Genome_902", it can be seen that the result are in line with the previous data in Fig. 5.5 but with even more incorrect solutions we can assume that the larger size of the workflow indeed has an impact quality of the CQM solution. The dividing line can be found around 67000 mark.

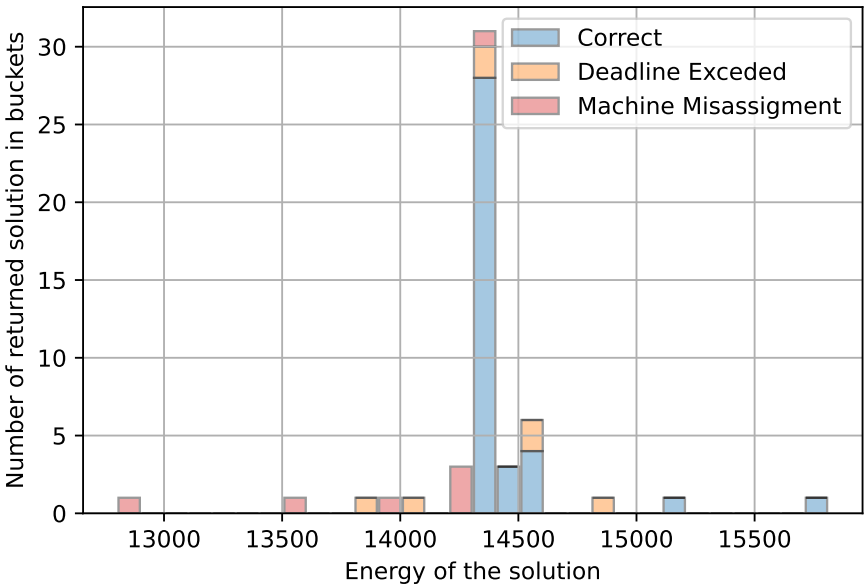

Figure 5.4: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_156 Workflow" and "basic_test" machines, with information about compliance with model constraints. (Gurobi energy value 14252.)

Results for the Cyfronet machine group

Each workflow that has been optimized for "Cyfronet" had the solutions returned by the D-Wave solver presented in the form of a histogram. The energy values between this group and the previous cannot be directly comparable.

In Fig. 5.7 representing the workflow "Genome_54", it can be seen that most of the solution returned by the solver have been put into a single bucket, similarly like for "basic_test" group test. The solver returned solutions also show high confidence int the solution at 310 energy line.

In Fig. 5.8 representing the workflow "Genome_156", it can be seen that unlike in the previous machine group there are not any confident results for this run, but most of the solutions returned are correct and there is no dividing line between the correct and incorrect results. This may suggest that there is some room for the solver to find a better solution.

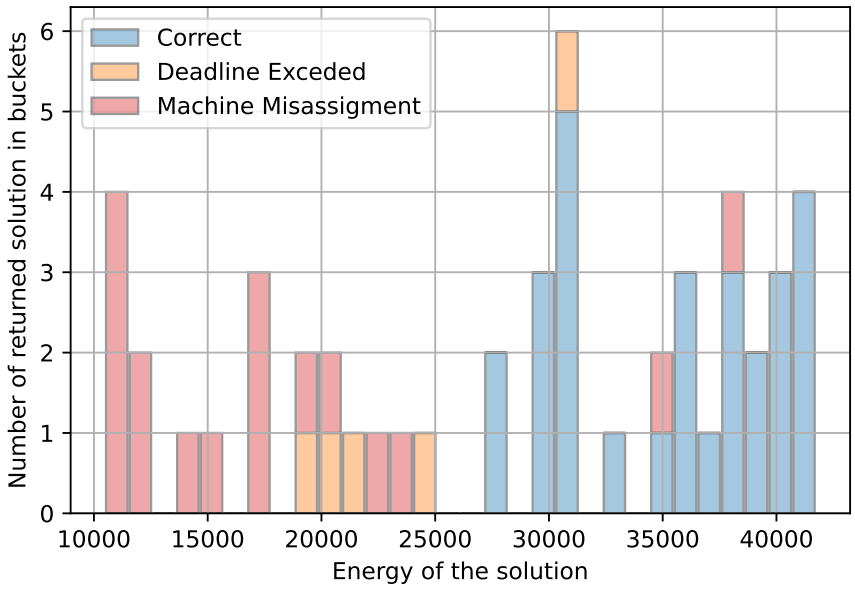

Figure 5.5: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_492 Workflow" and "basic_test" machines, with information about compliance with model constraints. (Gurobi energy value 24917.)

In Fig. 5.9 the top chart is representing the workflow "Genome_492" and in Fig. 5.10 the top bottom chart is representing the workflow "Genome_902". On both charts the distribution is much more spread out and more over almost all returned solutions are considered correct with regards to constraints. It is quite possible that those solutions are quite far from the optimal solution.

5.3.6 Conclusion

Given the results in the previous subsection we can conclude that in fact the size of the workflow with set computation time limit is affecting the solution quality. The exact relationship is quite difficult to pin down as the different workflows slightly differ in structure not only in task number.

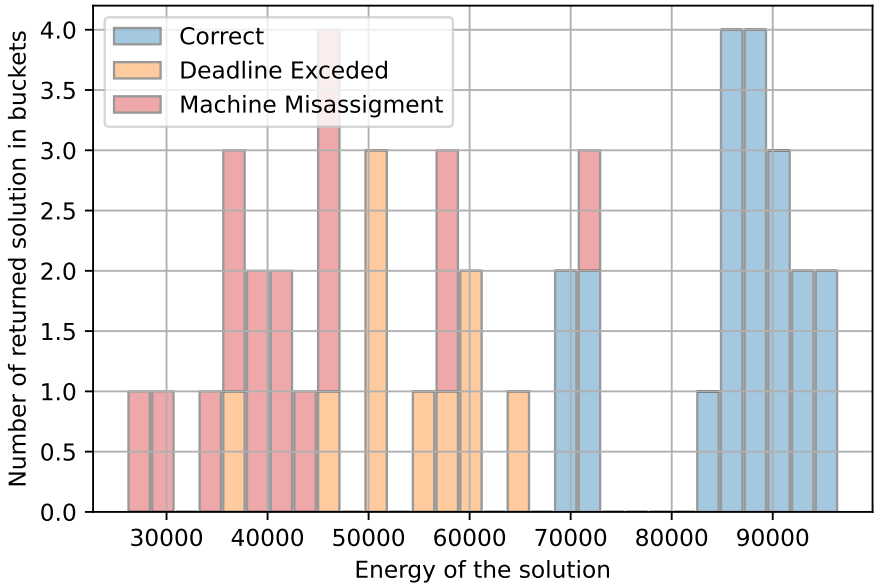

Figure 5.6: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_902 Workflow" and "basic_test" machines, with information about compliance with model constraints. (Gurobi energy value 60536.)

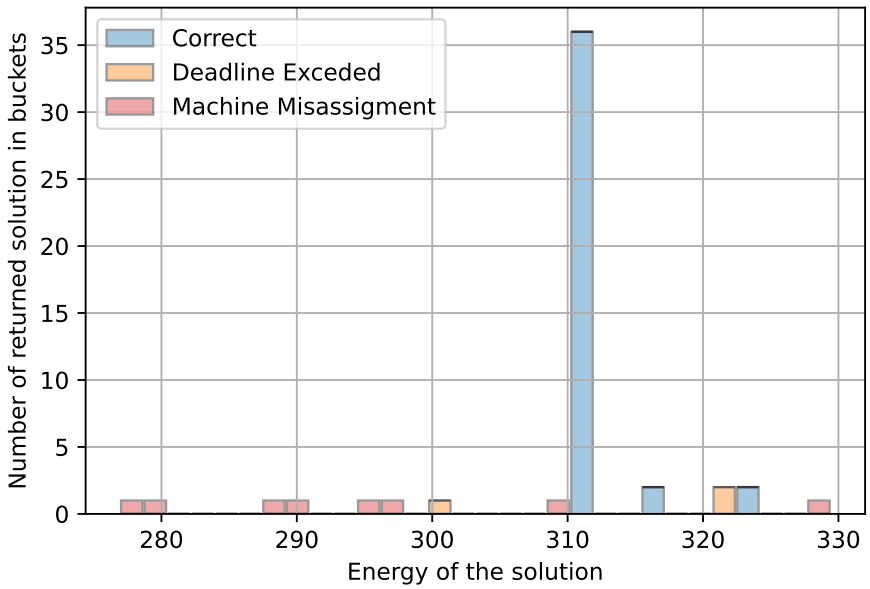

Figure 5.7: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_54 Workflow" and "Cyfronet" machines, with information about compliance with model constraints. (Gurobi energy value 311.)

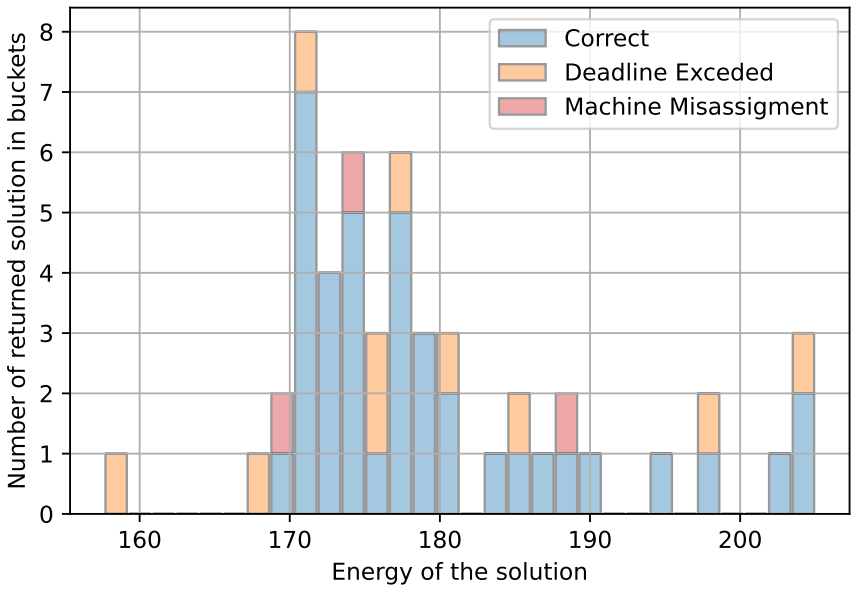

Figure 5.8: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_156 Workflow" and "Cyfronet" machines, with information about compliance with model constraints. (Gurobi energy value 169.)

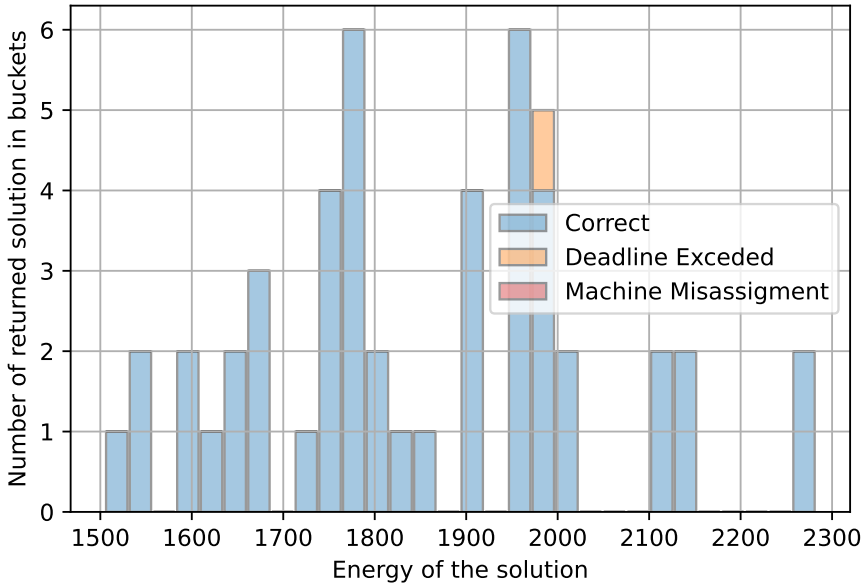

Figure 5.9: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_492 Workflow" and "Cyfronet" machines, with information about compliance with model constraints. (Gurobi energy value 728.)

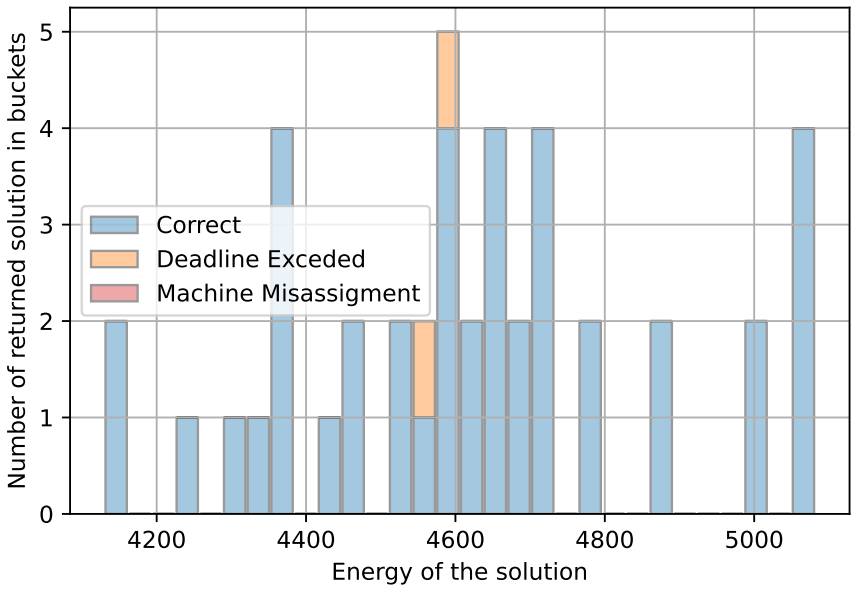

Figure 5.10: Histogram of energies of solutions returned by the D-Wave CQM solver for "Genome_902 Workflow" and "Cyfronet" machines, with information about compliance with model constraints. (Gurobi energy value 1868.)

5.4 Experiment 3 - deadline restriction

In this experiment the goal is to test how solution returned by the system changes with different values of the deadline given for the model. We used both the D-Wave CQM solver and Gurobi optimizer for this experiment. It focuses on the distribution of the solutions energy for test scenarios and response times from the solvers.

5.4.1 Workflow to optimize

For this experiment the workflow used in the previous Experiment "Genome_492" has been chosen. The details has been presented in Table 5.9.

Table 5.9: Job size and unique paths count in the "Genome_492" workflow in Experiment 3.

Name	Task Count	Path Count
Genome_492	492	4368

5.4.2 Machine setup

The machine group for this experiment has been previously used "basic_test". The parameters of the machines in the group has been presented in Table 5.4.

5.4.3 Deadline

Given the machine and workflow for this experiment the following values of the deadline have been generated to cover the whole range of the possible best solutions from fastest to slowest machines. The exact values are presented in Table 5.10

Table 5.10: Deadline values for experiment 3.

Deadline	18.3	20	60	100	140	180	220	260	300	340

5.4.4 Time limit

Time limit is only applicable for the D-Wave CQM solver and represents the actual time of computation performed on Leap Cloud solution using CQM solver. Two values have been used 5 and 10 second time limits. The 5 seconds is the minimal time the D-Wave Cloud API allows.

5.4.5 Size of the model

The number of binary variables and constraints for this model have been presented in Table 5.11

Table 5.11: Number of binary variables for each Experiment 3

x	Variables	Constraints
Genome_492	1476	4860

5.4.6 Results

The solution provided by the Gurobi solver in each test case is the solution with lowest energy as shown in Fig. 5.11. The Gurobi solver provides us with a unique solution, but the D-Wave CQM provides multiple solutions, therefore the one with the best energy is used in case of the D-Wave solver.

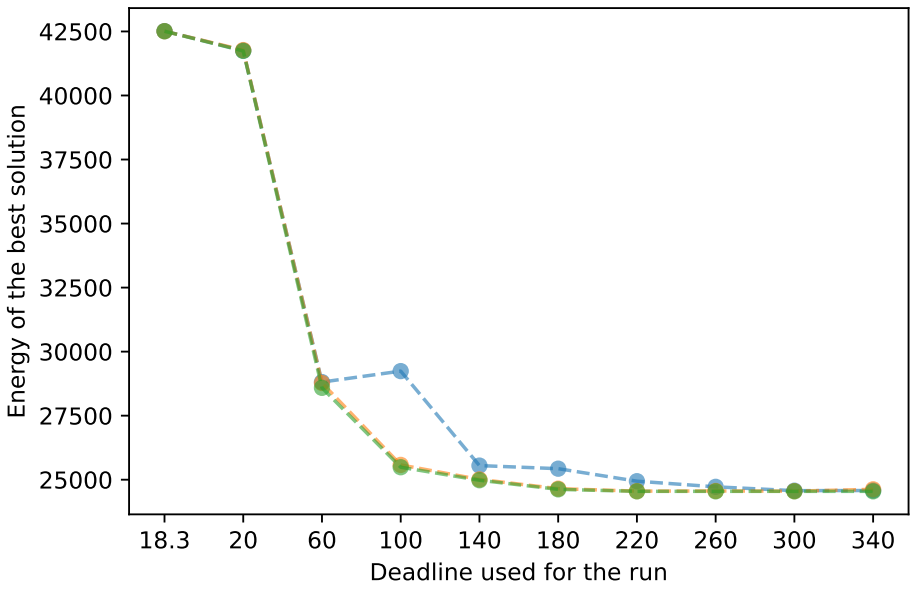

Figure 5.11: Energy of the best solution for optimization model for given time limit and solver vs the values of deadline used for each run of Experiment 3.

The differences on the chart in many places are relatively small compared to total value of energy, therefore in Fig. 5.12 and in Fig. 5.13 it is shown the relative difference between results from the D-Wave CQM solver and baseline from the Gurobi solver.

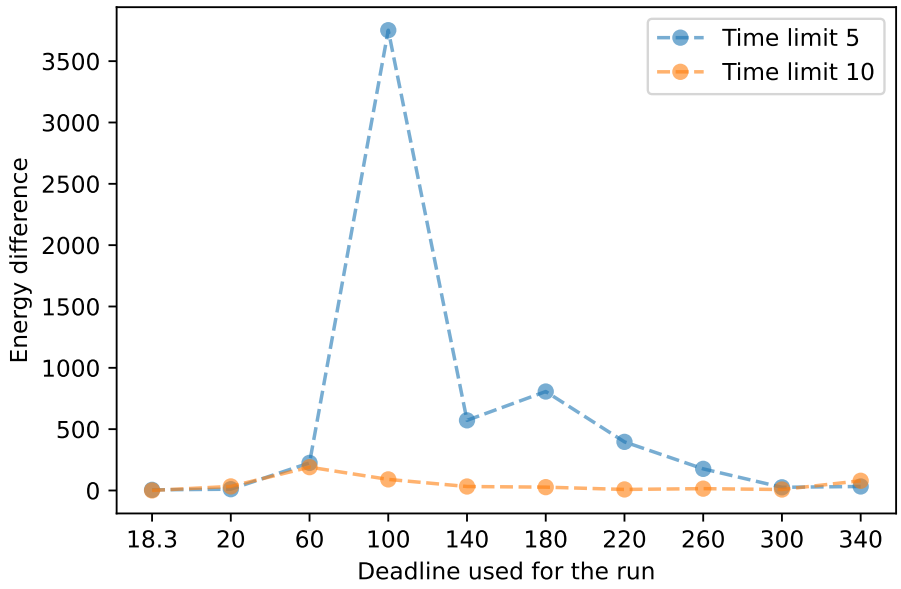

Figure 5.12: Difference in energy of the best solution for optimization model and Gurobi solver solution vs the values of deadline used for each run of Experiment 3.

As we can see providing more time to the CQM solver helped with finding better solutions in some cases, but in each case it still has isn't better than Gurobi solver. We can see in Fig. 5.14 that when 5 seconds time limit were given to the solver for the middle value of deadline constraint, energy values are more scattered than the ones on the provided by the solver with 10 second time limit located in Fig. 5.15. On both charts the most restrictive deadlines are quite visibly shifted towards greater energies showing that in fact the more restrictive the deadline, the smaller the potential solutions space is.

The Gurobi solver has been much faster in providing the result than the CQM regardless of the time limit which is shown in Fig. 5.16. The difference is quite substantial, the Gurobi solver is 3 to 4 orders of magnitude faster than the full response time from the

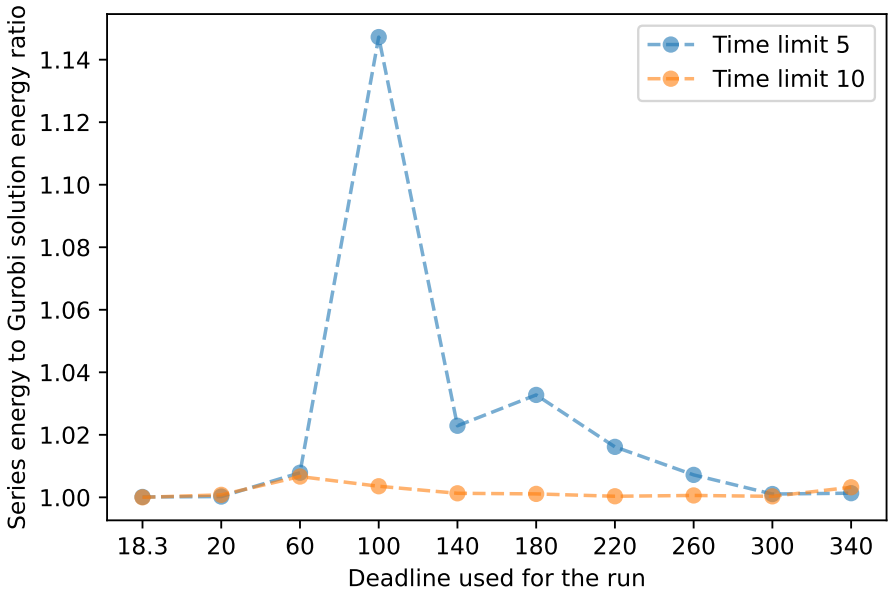

Figure 5.13: Ratio of energy of the best solution for optimization model and Gurobi solver solution vs the values of deadline used for each run of Experiment 3.

D-Wave solver. The behaviour of the D-Wave solver mandated further investigation. The experiment measured the time from requesting calculation on the Leap cloud to receiving first solution from D-Wave solver and also time from request to final solution arriving. The results have been presented in Fig. 5.17 and in Fig. 5.18 for 5 and 10 seconds limits respectively. As the inner workings of the CQM solver are being run on the D-Wave cloud the additional time usage can be attributed to:

- Network delay for communication with the solver,
- Awaiting in the queue for processing on the cloud,
- Each solution returned by solver being a separate http call.

5.4.7 Conclusion

The CQM solver is both much slower and returns worse results for bigger and more complex problems, this makes its usage impractical, however it is able to provide

solutions that are close to the optimal (or at least close to the one that is purely classical).

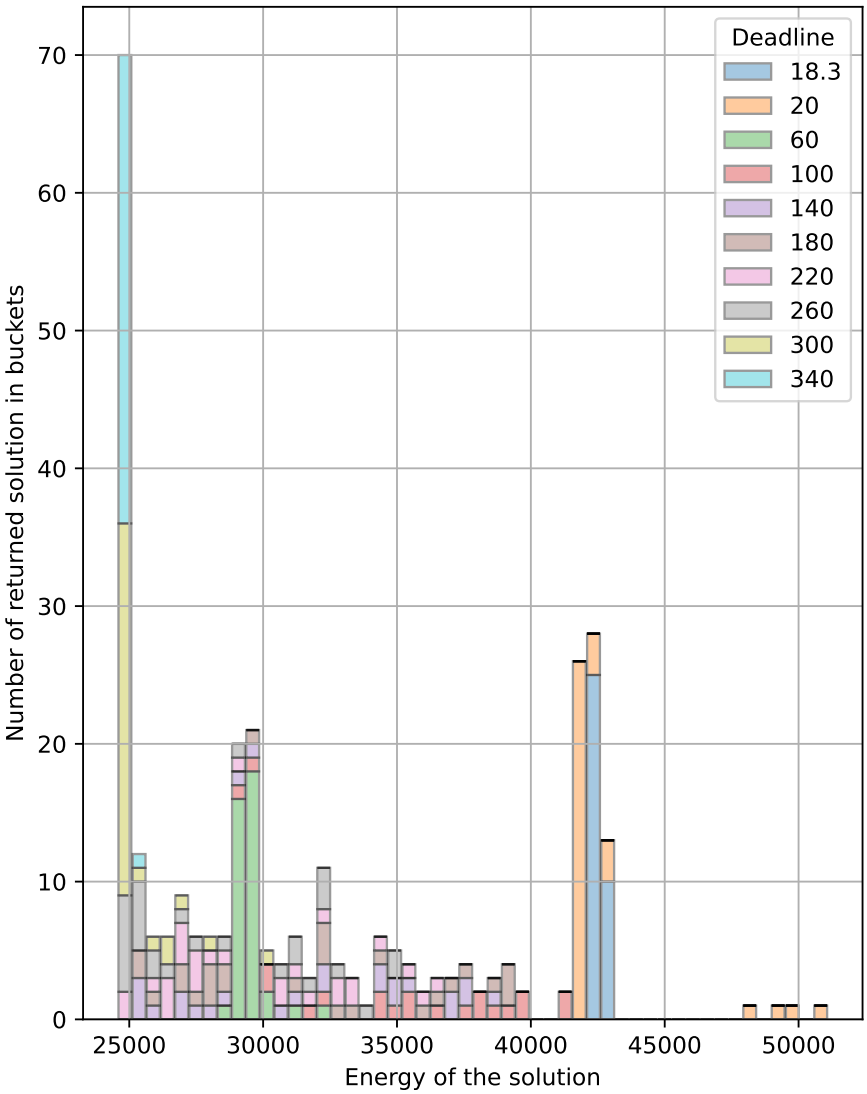

Figure 5.14: Histogram of energy values for correct solutions from the D-Wave CQM solver given 5 second time limit, grouped by the value of deadline parameter for the model.

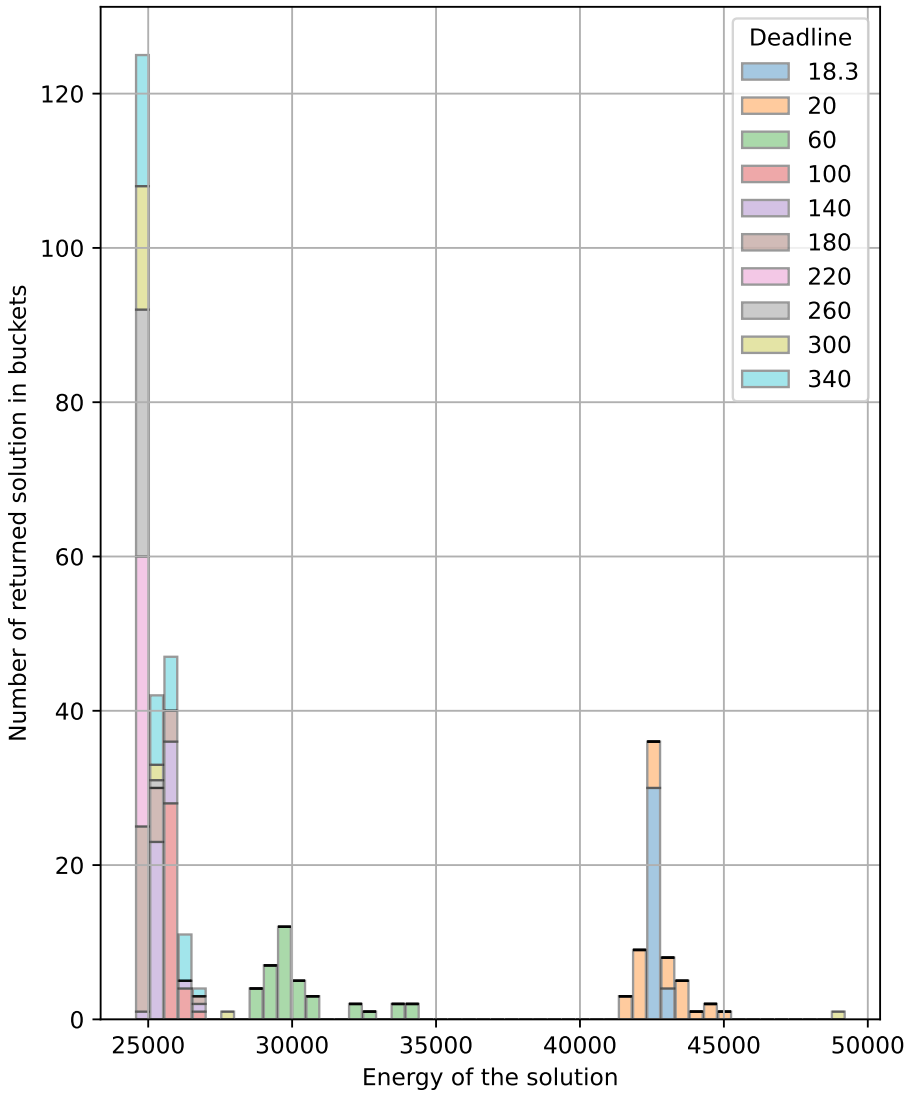

Figure 5.15: Histogram of energy values for correct solutions from the D-Wave CQM solver given 10 second time limit, grouped by the value of deadline parameter for the model.

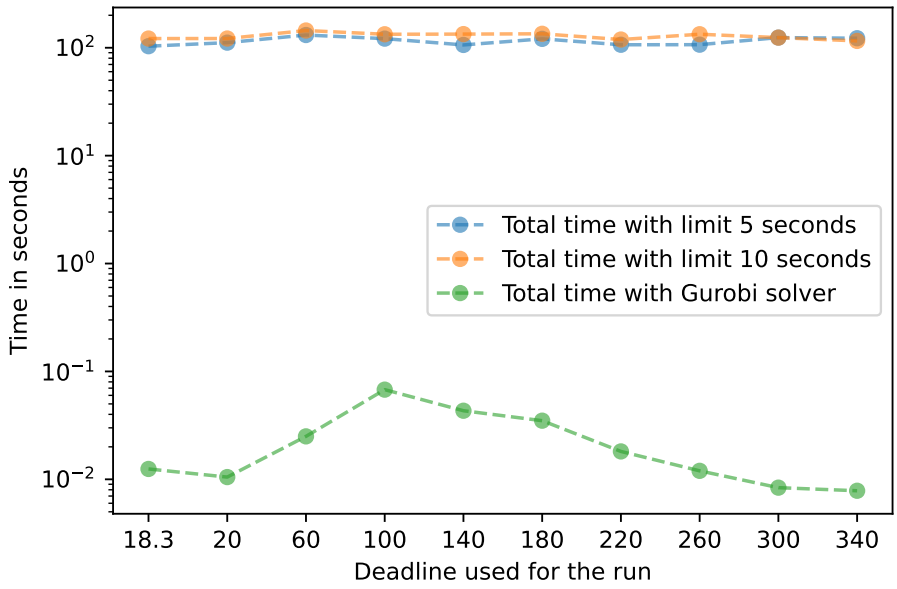

Figure 5.16: Response time from solvers.

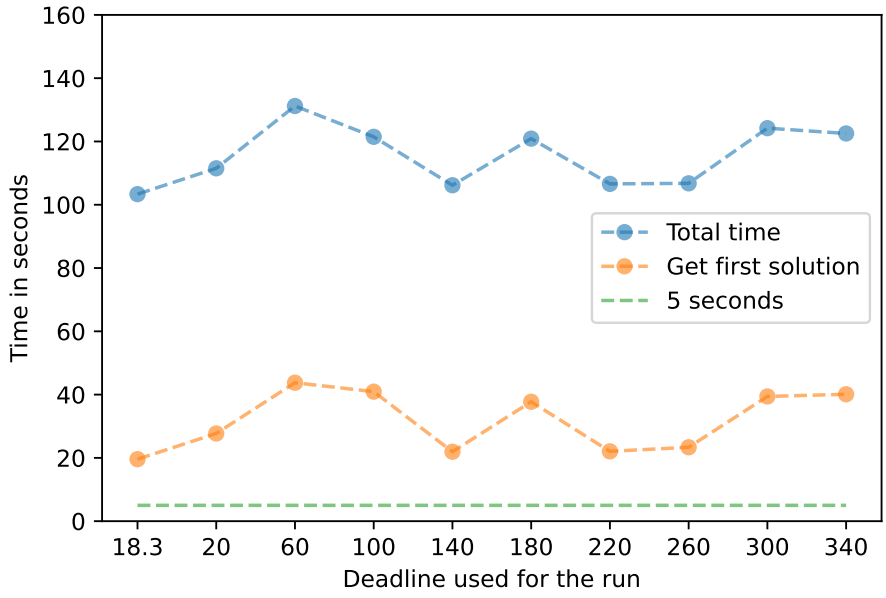

Figure 5.17: Timings of the CQM solver with 5 seconds time limit.

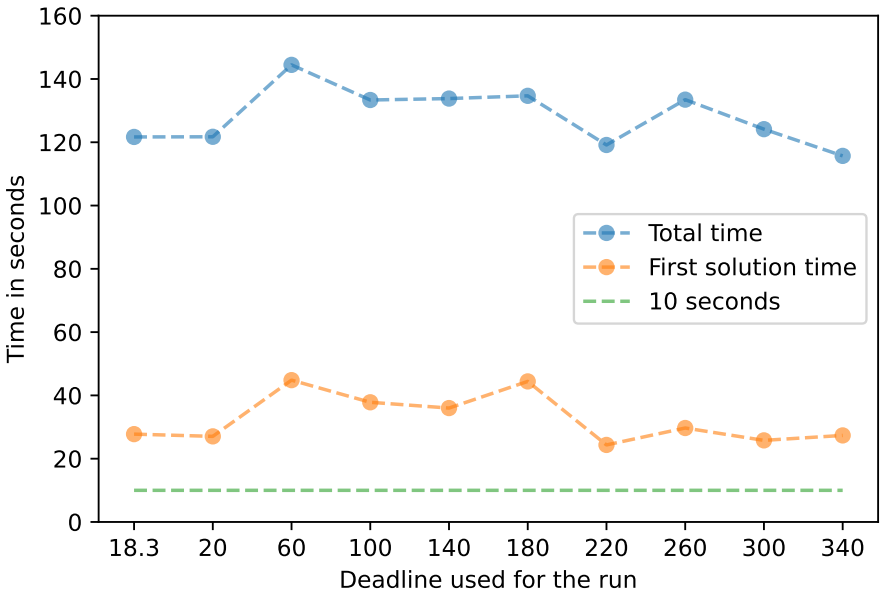

Figure 5.18: Timings of the CQM solver with 10 seconds time limit.

6 Summary and future works

In this chapter, the summary of the thesis is provided as well as suggestions for future improvements. The advantages and disadvantages of using the D-Wave CQM solver for Workflow Scheduling problems are discussed.

6.1 Translation of Workflow Scheduling problem to CQM model

In Chapter 4 of this thesis, it has been demonstrated that it is possible to easily transform the Workflow Scheduling problem into a Constrained Quadratic Model which then can be solved by a specialized solver. This has been confirmed by the experiments in Chapter 5 that the solution for the Workflow scheduling problem can be obtained using both the Gurobi optimizer as well as the D-Wave CQM.

6.2 Practicality of using D-Wave CQM solver

In the experiments in Chapter 5, it has been shown that the D-Wave CQM dolver can provide a solution in a much larger amount of time than the Gurobi optimizer. The D-Wave solver provides its user with multiple solutions for each call. Some of the returned solutions are not satisfying the constraints of the problem definition. The minimal time of 5 seconds for processing time is comparatively very large when taking into account that the same problem can be solved by the Gurobi optimizer in milliseconds. Taking all this into account the D-Wave CQM solver at this point is too slow for practical usage.

6.3 Potential Future Works and improvements

There still exists the possibility of future research and improvements on the solution and discoveries described in this thesis, mainly:

- Investigation into the inner workings of the D-Wave's CQM hybrid solver to redefine the problem in a way that better suite the internal algorithm,
- Define a custom hybrid solver using the D-Wave hybrid solver API to better control the heuristic environment in which the problem is solved,
- Try different translations of workflow scheduling problem into CQM. The new translations could take more input data into account like file access time, network configuration of machines etc.

This potential future research subjects could make the D-Wave hybrid annealer more practical to use.

Bibliography

[1] Tameem Albash and Daniel A. Lidar. "Adiabatic Quantum Computing". In: *Reviews of Modern Physics* 90.1 (Jan. 2018). arXiv: 1611.04471, p. 015002. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.90.015002.

[2] Michael Albrecht et al. "Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids". en. In: *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies - SWEET '12*. Scottsdale, Arizona: ACM Press, 2012, pp. 1–13. ISBN: 9781450318761. DOI: 10.1145/2443416.2443417.

[3] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". en. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1666-5.

[4] Kelly Boothby et al. "Next-Generation Topology of D-Wave Quantum Processors". In: *arXiv:2003.00133 [quant-ph]* (Feb. 2020). arXiv: 2003.00133.

[5] Tainã Coleman, Henri Casanova, and Rafael Ferreira da Silva. "WfChef: Automated Generation of Accurate Scientific Workflow Generators". In: *arXiv:2105.00129 [cs]* (Apr. 2021). arXiv: 2105.00129.

[6] Tainã Coleman et al. "WfCommons: A Framework for Enabling Scientific Workflow Research and Development". In: *arXiv:2105.14352 [cs]* (May 2021). arXiv: 2105.14352.

[7] Tainã Coleman et al. "WfCommons: A framework for enabling scientific workflow research and development". en. In: *Future Generation Computer Systems* 128 (Mar. 2022), pp. 16–27. ISSN: 0167739X. DOI: 10.1016/j.future.2021.09.043.

[8] D-Wave. *D-Wave QPU Architecture: Topologies*.

[9] D-Wave. *What is Quantum Annealing?*

[10] Ewa Deelman et al. "Pegasus, a workflow management system for science automation". en. In: *Future Generation Computer Systems* 46 (May 2015), pp. 17–35. ISSN: 0167739X. DOI: 10.1016/j.future.2014.10.008.

[11] Susan Fairley et al. “The International Genome Sample Resource (IGSR) collection of open human genomic variation resources”. en. In: *Nucleic Acids Research* 48.D1 (Jan. 2020), pp. D941–D947. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkz836.

[12] Fred Glover, Gary Kochenberger, and Yu Du. “A Tutorial on Formulating and Using QUBO Models”. In: *arXiv:1811.11538 [quant-ph]* (Nov. 2019). arXiv: 1811.11538.

[13] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In: *arXiv:quant-ph/9605043* (Nov. 1996). arXiv: quant-ph/9605043.

[14] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2022.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. en. In: *Science* 220.4598 (May 1983), pp. 671–680. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.220.4598.671.

[16] Yaroslav Koshka and M. A. Novotny. “Comparison of D-Wave Quantum Annealing and Classical Simulated Annealing for Local Minima Determination”. In: *arXiv:1911.03338 [quant-ph]* (Nov. 2019). arXiv: 1911.03338.

[17] Mark Lewis and Fred Glover. “Quadratic Unconstrained Binary Optimization Problem Preprocessing: Theory and Empirical Analysis”. In: *arXiv:1705.09844 [cs]* (May 2017). arXiv: 1705.09844.

[18] Philip Maechling et al. “Simplifying construction of complex workflows for non-expert users of the Southern California Earthquake Center Community Modeling Environment”. en. In: *ACM SIGMOD Record* 34.3 (Sept. 2005), pp. 24–30. ISSN: 0163-5808. DOI: 10.1145/1084805.1084811.

[19] Suresh Marru et al. “Apache airavata: a framework for distributed applications and computational workflows”. en. In: *Proceedings of the 2011 ACM workshop on Gateway computing environments - GCE ’11*. Seattle, Washington, USA: ACM Press, 2011, p. 21. ISBN: 9781450311236. DOI: 10.1145/2110486.2110490.

[20] *New Hybrid Solver: Constrained Quadratic Model*. 2021.

[21] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 1st ed. Cambridge University Press, June 2012. ISBN: 9781107002173 9780511976667. DOI: 10.1017/CBO9780511976667.

[22] Masayuki Ohzeki. “Breaking limitation of quantum annealer in solving optimization problems under constraints”. en. In: *Scientific Reports* 10.1 (Feb. 2020), p. 3126. ISSN: 2045-2322. DOI: 10.1038/s41598-020-60022-5.

[23] Frank Phillipson and Harshil Singh Bhatia. "Portfolio Optimisation Using the D-Wave Quantum Annealer". In: *arXiv:2012.01121 [quant-ph, q-fin]* (Nov. 2020). arXiv: 2012.01121.

[24] Matej Pivoluska and Martin Plesch. "Implementation of quantum compression on IBM quantum computers". en. In: *Scientific Reports* 12.1 (Dec. 2022), p. 5841. ISSN: 2045-2322. DOI: 10.1038/s41598-022-09881-8.

[25] Michael Reich et al. "GenePattern 2.0". en. In: *Nature Genetics* 38.5 (May 2006), pp. 500–501. ISSN: 1061-4036, 1546-1718. DOI: 10.1038/ng0506-500.

[26] Hajo A. Reijers and Wil M.P. van der Aalst. "The effectiveness of workflow management systems: Predictions and lessons learned". en. In: *International Journal of Information Management* 25.5 (Oct. 2005), pp. 458–472. ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2005.06.008.

[27] Michał Kamil Rudnik. "D'Wave Hybrid algorithms for workflow scheduling". PhD thesis. 2021.

[28] Jaydip Sen et al. "Cloud Computing - Architecture and Applications". In: *arXiv:1707.09488 [cs]* (June 2017). arXiv: 1707.09488. DOI: 10.5772/62794.

[29] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172.

[30] D'Wave Systems. "Hybrid Solver for Constrained Quadratic Models". In: (2021).

[31] The Event Horizon Telescope Collaboration. "First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole". In: *arXiv:1906.11238 [astro-ph, physics:gr-qc]* (June 2019). arXiv: 1906.11238. DOI: 10.3847/2041-8213/ab0ec7.

[32] Dawid Tomasiewicz. "Analysis of D'Wave 2000Q Applicability for Job Scheduling Problems". PhD thesis. 2020.

List of Figures

List of Tables