UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

TOWARDS SCALABLE CIRCUIT PARTITIONING FOR MULTI-CORE QUANTUM ARCHITECTURES WITH DEEP REINFORCEMENT LEARNING

ARNAU PASTOR LACUEVA

Thesis supervisor: SERGI ABADAL CAVALLÉ (Department of Computer Architecture)

Degree: Bachelor Degree in Informatics Engineering (Computing)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

27/06/2023

Abstract

Quantum computing holds immense potential for solving classically intractable problems by leveraging the unique properties of qubits. However, the scalability of quantum architectures remains a significant challenge. To address this issue, multi-core quantum architectures are proposed. Yet, the realization of such multi-core architectures poses multiple challenges in hardware, algorithms, and the interface between them. In particular, one of these challenges is how to optimally partition the algorithms to fit within the cores of a multi-core quantum computer. This thesis presents a novel approach for scalable circuit partitioning on multi-core quantum architectures using Deep Reinforcement Learning. The objective is to surpass existing meta-heuristic algorithms, such as FGP-rOEE's partitioning algorithm, in terms of accuracy and scalability. This research contributes to the advancement of both quantum computing and graph partitioning techniques, offering new insights into the optimization of quantum systems. By addressing the challenges associated with scaling quantum computers, we pave the way for their practical implementation in solving computationally challenging problems.

Keywords: Scalable circuit partitioning, multi-core quantum architectures, deep reinforcement learning

Resumen

La computación cuántica tiene un inmenso potencial para resolver problemas clásicamente intratables aprovechando las propiedades únicas de los cúbits. Sin embargo, la escalabilidad de las arquitecturas cuánticas sigue siendo un desafío significativo. Para abordar este problema, se proponen arquitecturas cuánticas de múltiples núcleos. No obstante, la realización de dichas arquitecturas plantea múltiples desafíos en hardware, algoritmos y la interfaz entre ellos. En particular, uno de estos desafíos es cómo particionar de manera óptima los algoritmos para que se ajusten dentro de los múltiples núcleos. Esta tesis presenta un enfoque novedoso para la partición escalable de circuitos en arquitecturas cuánticas de múltiples núcleos utilizando Aprendizaje Profundo Reforzado. El objetivo es superar a los algoritmos metaheurísticos existentes, como el algoritmo de particionamiento de FGP-rOEE, en términos de precisión y escalabilidad. Esta investigación contribuye al avance tanto de la computación cuántica como de las técnicas de particionamiento de gráficos, ofreciendo nuevos conocimientos sobre la optimización de los sistemas cuánticos. Al abordar los desafíos asociados con la escalabilidad de las computadoras cuánticas, abrimos el camino para su implementación práctica en la resolución de problemas computacionalmente desafiantes.

Resum

La computació quàntica té un enorme potencial per resoldre problemes clàssicament intractables aprofitant les propietats úniques dels cúbits. No obstant això, l'escalabilitat de les arquitectures quàntiques continua sent un gran desafiament. Per abordar aquesta qüestió, es proposen arquitectures quàntiques amb múltiples nuclis. Tanmateix, la realització d'aquestes arquitectures planteja diversos reptes en hardware, algorismes i la interfície entre ells. En particular, un d'aquests reptes és com particionar òptimament els algorismes per ajustar-los dins dels múltiples nuclis. Aquesta tesi presenta un enfocament innovador per a la partició escalable de circuits en arquitectures quàntiques amb múltiples nuclis mitjançant Aprenentatge Profund per Reforç. L'objectiu és superar els algorismes metaheurístics existents, com l'algorisme de particionament de FGP-rOEE, en termes d'exactitud i escalabilitat. Aquesta recerca contribueix a l'avançament tant de la computació quàntica com de les tècniques de particionament de gràfics, oferint nous coneixements sobre l'optimització dels sistemes quàntics. En abordar els reptes associats amb l'escalabilitat dels ordinadors quàntics, obrim el camí per a la seva implementació pràctica en la resolució de problemes computacionalment desafiants.

Acknowledgment

First and foremost, I would like to express my gratitude to my thesis supervisor, Sergi Abadal Cavallé. His guidance and support have been invaluable throughout this journey. I am grateful to Sergi for suggesting the topic of this thesis and for providing essential insights and direction at every stage of the project.

I would also like to express my deep appreciation to Pau Escofet i Majoral and Pere Barlet Ros, who generously shared their field expertise and provided consistent feedback throughout the development of this project. Their constructive criticism and insightful suggestions have greatly enriched the quality and depth of this thesis. Their willingness to invest their time and effort in reviewing and offering valuable input has been truly appreciated.

Finally, I would like to thank all the teachers and staff at FIB who have put dedication and commitment into my academic growth. Their tireless efforts and support have played a crucial role in shaping my educational journey and have contributed significantly to my development as a person.

Contents

List of Figures

List of Tables

Chapter 1

Context and scope

Quantum computing is a rapidly evolving field that has the potential to revolutionize computation by making certain types of classically intractable problems solvable. Unlike traditional computing, which uses bits as the most basic unit of information, quantum computing relies on qubits or quantum bits. Qubits have the unique property of being able to exist in multiple states simultaneously, known as superposition, and can also become entangled with other qubits which allows much faster computation.

One of the primary reasons why quantum computation is such a promising field is the ability to perform certain computations exponentially faster. For example, Shor's algorithm [23], a quantum algorithm, is used for finding the prime factors of an integer much faster than any known classical algorithm, which has massive implications for cryptography and computer security. Additionally, quantum computing can also be used to solve classical NP-hard problems, that for now were intractable, such as the travelling salesman problem or the knapsack problem.

1.1 Motivation

Despite the fact that quantum computing seems like a great alternative for solving some problems that classical computing has been stagnant for some time, it has encountered a significant roadblock with regard to scalability in terms of the number of qubits a single machine can have.

Currently, scaling-up (increasing the computational power of the machine, usually by increasing the number of qubits) these devices with low error rates has proven extremely challenging for numerous reasons [12], like wiring and crosstalk problems [20]. IBM, a frontrunner in the field of quantum computing, recently unveiled a new computer boasting over 400 qubits [10] and has published its development roadmap, Figure 1.1, which outlines the specific goals IBM is working towards (including their scaling objectives). However, this number of qubits is not yet large enough to efficiently solve the aforementioned problems.

Consequently, when it comes to scaling quantum computers, for now, it seems better to use a scale-out strategy (combining multiple computers together and connecting them via a quantum coherent interconnect) rather than a scale-up one. Using a modular approach can overcome the limitations of the total number of qubits used, but, nevertheless, it brings to the table new problems to overcome. It is thought by experts that, optimistically, with the current technological advancements, performing non-local (between different machines) communication operations, using the quantum coherent interconnect, has a latency somewhere between 5-100x higher than in-cluster (within the same machine) communication [4].

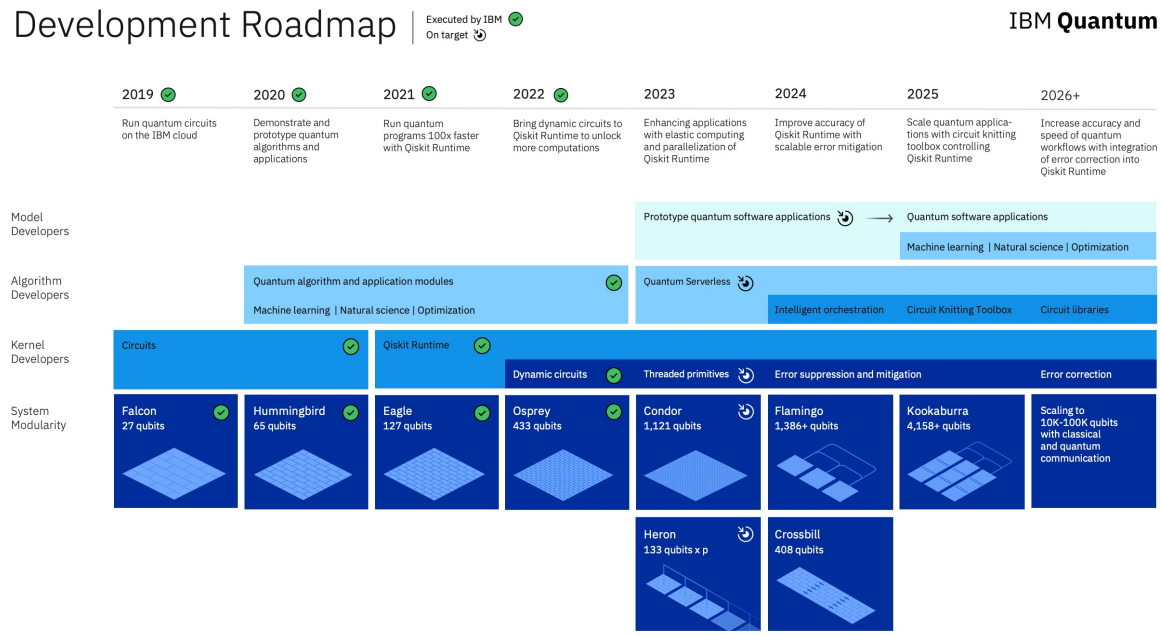

Figure 1.1: IBM quantum development roadmap (Source: [9])

Hence, since all interactions between qubits are known at compile time, it is the compiler's job to distribute the program across multiple machines to reduce the non-local communication operations at its minimum, thus reducing the overall program execution time.

Thus far, very few research works have addressed the problem of partitioning quantum algorithms. One of these works is the FGP-rOEE algorithm [4], a state-of-the-art algorithm that utilizes a heuristic procedure to optimize the mapping of qubits into quantum cores across the execution of the quantum algorithm. This algorithm is widely employed; however, it suffers from a serious scalability problem that greatly restricts its utilization in large circuits.

1.1.1 Stakeholders

The success of this thesis would benefit a broad number of stakeholders. A fair share of them would be the quantum computing hardware manufacturers such as IBM, Microsoft, Google, Intel and many others who could benefit from the model built to reduce the communication latency between their machines, thus focusing on maybe developing efficient physical clusters of quantum computers. Also, in the scenario of reducing the latency of non-local communication near to in-cluster one, from the perspective of an outsider, it would create the illusion that a set of computers is operating as a single entity, apart from giving these companies the possibility to offer more powerful and reliable options, making them more attractive to potential costumers. In a similar way to the quantum hardware manufacturers, other companies that would also be benefited are the cloud providers which also provide cloud-based quantum computing, such as AWS and Microsoft Azure.

Other important stakeholders would be research institutions and quantum algorithm developers. This accomplishment would enable the development of more efficient and powerful multicore quantum algorithms.

Overall, any improvement in the quantum computing space can lead to a significant impact on society, as it has the potential to revolutionize critical industries such as finance, healthcare, logistics, and more.

1.2 Justification

1.2.1 Previous studies and existing solutions

As previously discussed, movements between cores in quantum multi-core architectures can carry a high latency associated and must be considered when optimizing mapping. To date, only one mapping algorithm for multicore quantum computing architectures has been proposed, FGP-rOEE, which is based on the relaxed Overall Extreme Exchange (rOEE) [4].

While the rOEE algorithm is a promising mapping strategy, it has only been tested on a limited range of quantum circuits used in common algorithms, such as Shor's algorithm [23], using between 50 to 100 qubits. In terms of hardware, this algorithm was tested with 10 machines each containing 10 qubits. It is important to note that testing a mapping algorithm on a small number of machines and qubits can limit the generalizability of the results to larger and more complex quantum systems.

Another interesting resource is the master's thesis written by Anabel Ovide González [15], where, she replicates FGP-rOEE's algorithm and changed different architectural parameters, such as the number of cores and qubits per core, as well as algorithm characteristics like gate density. Moreover, Ovide conducted a scalability study of the rOEE algorithm.

The mentioned resources represent the "rival to beat," as the primary goal of this thesis is to build an RL model that outperforms them. In line with this objective, there is Ruizhe Yu Xia's master's thesis [24], which focuses on creating a Graph Neural Network (GNN) capable of reproducing conventional partitioning algorithms such as FGP-rOEE.

1.2.2 Problem definition

This thesis aims to develop an RL model that endeavours to approximate the optimal graph partition, thereby minimizing non-local communications and surpassing the performance of FGP-rOEE's algorithm. Section 1.1 highlights that non-local communication operations between machines can exhibit a latency 5-100 times higher than in-cluster communication. The objective is to leverage this knowledge to construct a robust RL model that achieves superior results compared to FGP-rOEE's algorithm, explained in both the paper [4] and Chapter 3.

1.2.3 Use of Reinforcement Learning

Reinforcement Learning is a type of machine learning technique that enables an agent to learn through experimentation using feedback from its own actions and experiences. It involves an agent that interacts with an environment, learns from feedback in the form of rewards, and takes actions to maximize cumulative rewards over time.

The problem addressed in this thesis can be reduced to a graph partitioning problem, where achieving a balanced partition of nodes across multiple machines with the minimum cut becomes the optimal solution.

Existing solutions for this problem, like FGP-rOEE's algorithm, typically use meta-heuristics to provide approximate solutions. However, these approaches suffer from scalability issues, especially when dealing with large circuits, which is the case in most real-world scenarios.

Given the recent advancements and significant improvements in the Deep Learning industry, it is natural to consider leveraging this technology to obtain a better solution in terms of accuracy and scalability.

The graph partitioning problem is a well-known NP-hard problem, making the use of Supervised Learning impractical since computing the optimal solution is infeasible. Another potential approach using Supervised Learning involves considering FGP-rOEE's results as the ground truth that the algorithm aims to predict. However, this approach limits the maximum accuracy the model can achieve to FGP-rOEE's accuracy, whereas the overall objective is to surpass it

Reinforcement Learning, on the other hand, appears to be a suitable technique to tackle this problem. Given the widespread use of RL in solving video games, the approach taken was to "gamify" the problem, assigning positive rewards when the agent takes actions that bring it closer to an optimal solution, and negative rewards otherwise. Through this approach, the agent is guided towards an almost optimal solution.

Once the model is trained, making predictions only involves matrix multiplications, which leads to efficient execution without scalability issues. It is worth noting that the learning time for large circuits might be longer, but the scalability problems lie in the execution phase rather than the training phase.

1.3 Scope

1.3.1 General objectives

The primary objective of this thesis is to **design a RL environment and model capable of achieving maximum accuracy in graph partitioning tasks**. The graphs under consideration will be derived from quantum systems, which are known to exhibit unique and often complex patterns.

Given the inherent challenges associated with quantum systems, the development of RL environments and models tailored to these systems has the potential to unlock new insights in the field of quantum computing and machine learning.

1.3.2 Sub-objectives

Given the aforementioned general objective, the following sub-objectives can be formulated:

- Comprehend and dive deep into the state-of-the-art partitioning techniques for quantum algorithms.
- Design and implement various reinforcement learning architectures with the aim of minimizing qubit reassignment during circuit execution.
- Evaluate the performance of the proposed models on a diverse range of quantum computing benchmark datasets.
- Conduct a scalability comparison of the proposed architecture with higher accuracy and FGP-rOEE's algorithm [4].

Among all the above sub-objectives, the second one, which involves designing and implementing various RL architectures, will likely be the most challenging. This is due to the considerable effort required in designing and testing multiple architectures to identify the most effective one.

1.3.3 Requirements

The requirements this thesis has to fulfil in order to be considered a success are the following:

- Develop a RL model capable of accurately making valid partitions with an accuracy of at least 95%.
- Outperform FGP-rOEE's algorithm [4] in terms of execution time and performance.

Chapter 2

Background

2.1 Concepts

- Compiler: software that transforms a collection of statements written in a high-level programming language into a lower-level representation.
- Graph partitioning problem: given a graph $G = (V, E)$ and a fixed number of partitions k, assign the vertices V into k partitions such that the weight of the edges between vertices in different partitions is minimized. Finding the optimal assignment in NP-hard.
- Quantum algorithm: a step-by-step procedure for solving a problem, where each step or instruction can be performed on a classical computer.
- Quantum computer: a type of computer that uses the principles of quantum mechanics to perform calculations. It uses the qubit or quantum bit as its basic unit of quantum information.
- Qubit: basic unit of quantum information. It can be in a state of superposition, which can be described as:

  $$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

  Where α and β are the probability amplitudes, meaning that $|\alpha|^2$ is the probability of outcome $|0\rangle$ and $|\beta|^2$ is the probability of outcome $|1\rangle$ when we measure it.
- Hamming distance: a measurement function that, given two vectors, returns the number of positions where they have different values.
- Quantum circuit: sequence of quantum logical gates and operations that manipulate qubits to perform a specific computational task.
- Time-slice: segment of a quantum circuit that contains a set of consecutive quantum logic gates that can be executed simultaneously, with each qubit involved in at most one gate.

2.2 Deep Reinforcement Learning (DRL)

Reinforcement Learning (RL) is a branch of machine learning that tackles sequential decision-making problems [1]. It involves an agent interacting with an environment, learning from feed-

back in the form of rewards or penalties, and striving to maximize cumulative rewards over time. Through experimentation and experience, the agent develops optimal policies and strategies to achieve its objectives.

Deep Reinforcement Learning (DRL) is an area of machine learning that combines deep learning techniques with reinforcement learning algorithms to enable agents to learn and make decisions in complex environments. Specifically, neural networks are used to approximate the Policy Network and the Value Function. It has gained significant attention in recent years due to its ability to tackle a wide range of challenging problems, including game playing and robotics [13].

The key components of DRL include:

- **Environment**: the system with which the agent interacts. It defines the set of possible states, rules, and rewards. The agent extracts information from the environment, takes actions based on its current state, and subsequently observes the resulting state and associated reward.
- **State**: represents the current observation or input to the agent.
- **Agent**: the learning entity that interacts with the environment with the aim of solving a given task.
- **Action**: choices available to the agent in a given state.
- **Reward**: scalar value that serves as immediate feedback or reinforcement received by the agent after taking an action in a particular state. It quantifies the desirability or quality of the agent's actions, and the goal is to maximize the cumulative reward over time.

Policy Network (π)

A policy is a function that maps observations from the environment to actions, guiding the agent's actions in different states to maximize cumulative rewards over time. In DRL, the policy network is typically implemented as a neural network, where observations serve as inputs and the outputs represent a probability distribution over the actions. This distribution indicates the likelihood or confidence of each action being the optimal choice given a particular state.

The advantage of using a neural network as the policy network is that it allows for the representation of complex mappings between observations and actions. Through training, the policy network learns to generalize from observed states and actions, enabling it to make predictions of unobserved data. This adaptability enables the agent to perform well in complex environments characterized by high-dimensional state spaces.

Value Function

The value function is a mapping from an environment observation (or observation-action pair) to the expected cumulative reward. It quantifies the expected return that the agent can achieve from that state onwards. By quantifying the value of different states, the value function helps the agent assess the long-term desirability of taking certain actions or transitioning to specific states.

In DRL the value function is estimated using a neural network. The network takes an observation or observation-action pair as input and outputs an estimate of the expected cumulative reward. This estimation is based on the agent's acquired knowledge and experiences, enabling it to make predictions regarding the long-term value of various states or state-action pairs.

Exploration vs Exploitation

Exploration vs exploitation represents the trade-off between gathering new information and utilizing existing knowledge. Achieving the right balance is crucial for effective learning and the discovery of optimal policies.

Exploration refers to the act of seeking out new and unfamiliar states or actions in the environment, allowing the agent to gather valuable information and explore the dynamics of the system. By exploring different parts of the action space, the agent can gain insights into the environment's behavior and identify strategies that lead to higher cumulative rewards. Adequate exploration is necessary during the early stages of learning to ensure the agent does not miss out on potentially rewarding opportunities and to facilitate the discovery of optimal policies.

Exploitation, on the other hand, involves utilizing existing knowledge and experience gained through exploration to maximize rewards. By leveraging the learned policy or value estimates, the agent selects actions that have previously yielded high rewards or are known to be effective. Exploitation allows the agent to make decisions that are likely to result in short-term rewards and take advantage of established successful strategies.

Finding the right balance between exploration and exploitation is challenging. If the agent focuses solely on exploration, it may spend excessive time exploring unproductive areas of the state-action space, resulting in slower learning and lower immediate rewards. Conversely, excessive exploitation may cause the agent to get stuck in suboptimal strategies and miss out on discovering potentially better policies.

Actor - Critic

The Actor-Critic architecture is a widely employed approach in DRL that combines the strengths of policy-based and value-based methods, effectively addressing the challenges of learning an optimal policy and estimating state values simultaneously.

The Actor plays the role of the policy network, responsible for selecting actions based on the current state of the environment. On the other hand, the Critic's responsibility is to estimate the value function.

The Actor-Critic architecture leverages the interaction between the Actor and the Critic to improve both the policy and value estimates. During training, the Actor selects actions based on the current policy, while the Critic provides feedback on the quality of those actions. This feedback is then used to update the policy and value estimates, enabling the agent to learn and improve its decision-making capabilities over time.

2.2.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization is a widely-used algorithm developed by OpenAI [21] in the field of DRL. It is designed to provide stable and reliable learning by using a trust region approach that constrains policy updates. This ensures that the policy changes are not too large, preventing instability during the learning process. PPO is also known for its sample efficiency, as it efficiently reuses and reweights collected samples to make the most of the available data.

This algorithm belongs to the family of Policy Gradient Methods.

Policy Gradient Methods

This family of methods aims to optimize the policy of an agent directly by employing gradient ascent to update it [18].

The general loss function is

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.1}$$

where π_θ represents the policy, which, given an observed state, provides a probability distribution over the actions, and $\hat{A}_t$ is the advantage function.

In essence, these methods aim to maximize the expectation of the probability of selecting an action in a specific state, multiplied by the advantage function, which indicates the value of the action taken.

Advantage function

The advantage function estimates the relative value of the selected action in the current state. It determines whether the action taken was better or worse than expected.

The advantage function $\hat{A}_t$ is computed as

$$\hat{A}_t = G_t - b(s_t) \tag{2.2}$$

where G_t is discounted reward and $b(s_t)$ the baseline estimate.

The discounted reward follows the formula

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

where $\gamma \in [0, 1]$ is the discount factor. When it is close to 0, it provides a myopic evaluation, but when close to 1, it values future rewards more. This discounted reward is computed using the known values of the old policy.

The baseline estimate is essentially the value function, which aims to predict the cumulative reward given a state. The value function is continuously updated with the discounted reward once the real cumulative rewards are known (supervised learning problem).

Ultimately, if the estimate of the loss function L^{PG} is positive, the likelihood of selecting the corresponding action given the state will increase. Conversely, if it is negative, the probability of selecting the action decreases.

The problem with the classic Policy Gradient Methods is that, if gradient ascent is used on one batch of collected experience, there is the possibility that the updated parameters in the network can end up far outside the range where the data of the batch was collected. This can lead to the advantage function (more precisely the value function, which is a prediction) performing very poorly, and its predictions becoming very far-fetched from reality, ultimately destroying the policy.

To solve this issue, multiple approaches have emerged, such as Trust Region Policy Optimization (TRPO) [22], which aims to restrict the size of policy updates. However, the introduction of this constraint also adds an additional overhead to the optimization process, which can result in undesired training behaviour.

PPO is introduced with the objective of ensuring that the updated policy does not deviate too far from the current policy while maintaining a relatively simple loss function. To achieve this, it introduces the concept of the textbfClipped Surrogate Objective, whose objective function is

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\,\hat{A}_t,\ clip(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\,\hat{A}_t] \tag{2.4}$$

where ϵ is a hyperparameter, and $r_t(\theta)$, the probability ratio, is defined as

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, \; r(\theta_{old}) = 1 \tag{2.5}$$

The objective of the function $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$ is to modify the objective function so that the value of r_t lies within the range $[1-\epsilon, 1+\epsilon]$. This effectively prevents doing large updates.

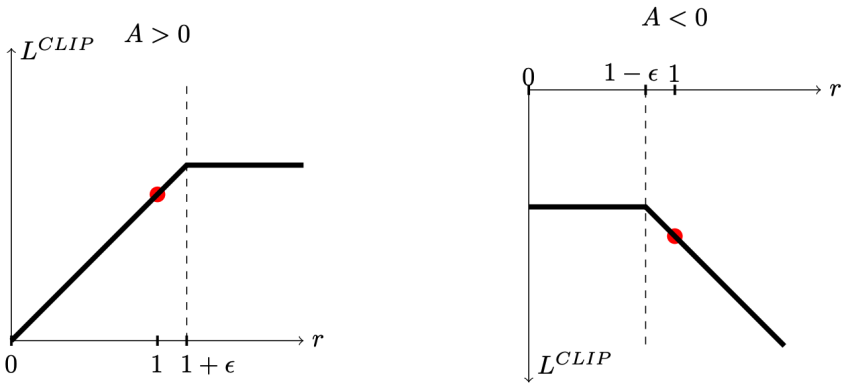

Figure 2.1: Plots representing the function L^{CLIP} under different conditions of A (Source: [21])

In Figure 2.1, the right plot demonstrates how the maximum update is flattened out when the advantage function is positive and r_t becomes too high. This limits the effect of the gradient update. Similarly, when the advantage function is negative, it flattens out as r_t approaches 0. This avoids excessively reducing the likelihood of committing the action.

Finally, the objective function of PPO combines the objective function L^{CLIP}, the loss function of the value function, and an entropy term that controls the level of exploration the agent undertakes. The overall objective function is defined as follows

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{2.6}$$

where c_1, c_2 are coefficients, and S represents the entropy bonus.

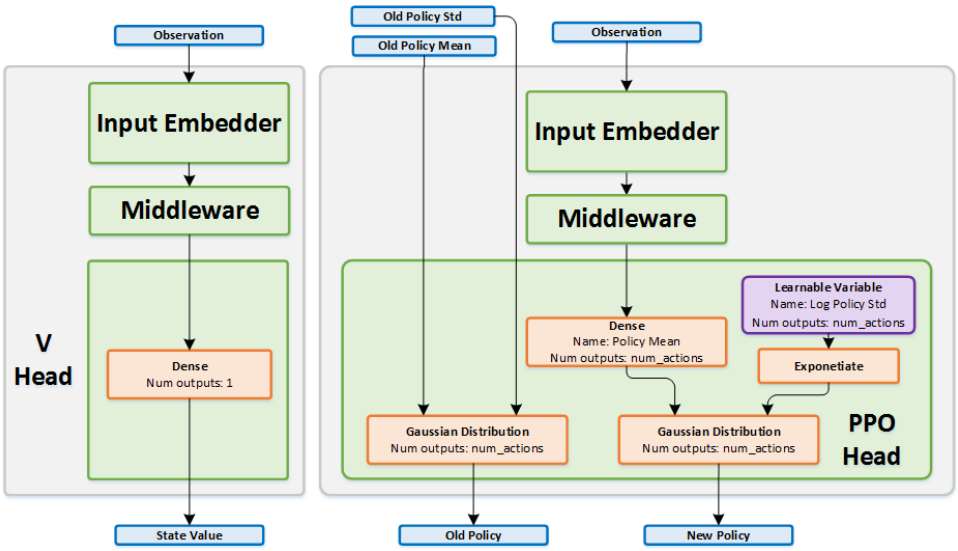

Figure 2.2: Diagram representing the workflow of PPO algorithm (Source: [11])

MaskablePPO

MaskablePPO is a novel variant of the Proximal Policy Optimization (PPO) algorithm that introduces a masking mechanism to enhance its capabilities. This variant extends the basic PPO algorithm by incorporating a mask that selectively determines which parts of the policy should be updated during training. By doing so, MaskablePPO provides finer control over policy updates.

The underlying mechanism of MaskablePPO is straightforward yet effective (for a detailed explanation, refer to [8]). As previously mentioned, the policy network is implemented as a neural network that takes an observation as input and produces a probability distribution as output. Typically, the final layer of this network employs a Softmax function to generate the distribution. In the context of MaskablePPO, when faced with an invalid action, such as a_2 in state s, the softmax value corresponding to a_2 is replaced with a large negative number M, effectively driving the probability of selecting a_2 towards zero. To achieve this, a masking function $mask : \mathbb{R} \to \mathbb{R}$ is applied to the network's output distribution $\pi_\theta(\cdot|s)$. Thus, the process can be described as follows

$$\begin{aligned} \pi_\theta(\cdot|s) = softmax(mask([l_0, l_1, l_2, l_3])) = softmax([l_0, l_1, M, l_3]) = \\ = [\pi_\theta(a_0|s), \pi_\theta(a_1|s), 0, \pi_\theta(a_3|s)] \end{aligned} \tag{2.7}$$

The introduction of the masking mechanism in MaskablePPO allows for improved adaptability and control during the training process, as well as faster convergence. By selectively suppressing certain actions based on their validity in specific states, the algorithm can learn policies that align more closely with desired behaviors or constraints. This level of granularity provides valuable flexibility when dealing with complex environments.

In conclusion, MaskablePPO presents a valuable enhancement to the PPO algorithm, enabling precise policy updates through the integration of a masking mechanism.

Chapter 3

State of the Art: FGP-rOEE

This chapter will focus on exploring the state-of-the-art algorithm known as FGP-rOEE [4], which stands for Fine Grained Partitioning using relaxed Overall Extreme Exchange. FGP-rOEE is an advanced quantum circuit partitioning algorithm designed to optimize mappings for each time-slice. The algorithm utilizes a lookahead function to minimize the expenses associated with reassignments in future time-slices. By analyzing the key components of FGP-rOEE, such as Overall Extreme Exchange (OEE) and the utilization of lookahead weights, valuable insights can be gained into the cutting-edge techniques employed for circuit partitioning. Furthermore, the limitations of FGP-rOEE are discussed, particularly its scalability, and how the approach taken in this thesis can solve them.

3.1 Overall Extreme Exchange (OEE)

The Overall Extreme Exchange (OEE) algorithm is a heuristic procedure for solving the k-way graph partitioning problem [17], which consists of allocating the nodes in the graph into k partitions equally. It works by consecutively selecting two nodes to exchange that maximizes the reduced exchange cost. It can also find a sequence of pairs of vertices to exchange and make as many exchanges as possible to obtain an overall benefit. The algorithm continues as long as the exchange reduces the partitioning cost.

The general goal of the algorithm is to find the optimal partitioning that minimizes the cut between the machines. It is widely known that the graph partitioning problem is NP-hard, and therefore this approach to solving it is also considered NP-hard. Considering this information, FGP-rOEE utilizes what they call a relaxed version of the algorithm.

Relaxing the Partitioning Algorithm

In the relaxed version of OEE, instead of running the algorithm until finding the optimal assignment, they run it until the partition becomes valid for the time-slice (all interacting qubits are in the same partition), and then they make no more exchanges.

Making this change speeds up OEE, making it possible to execute it in cases where the execution time of OEE would be infeasible.

3.2 Lookahead weights

A key feature of the FGP-rOEE algorithm is the incorporation of lookahead weights. Lookahead weights are assigned to the edges of the graph based on a lookahead strategy. These weights capture the anticipated impact of exchanging a qubit in future time-slices and guide the swapping decisions.

More specifically, Baker et al. emphasize the importance of lookaheads in [4], stating:

> It is critically important we make our exchange choices using lookahead weights applied to the time slice graphs. Choosing without information about the upcoming circuit provides no insight into which qubits are beneficial to exchange.

The lookahead scheme exhibits such a remarkable conceptual appeal and will therefore be used in this thesis. Consequently, the agent, apart from observing the interacting qubits for the current time-slice, can also see future interactions.

3.3 Graph formation

Now that the concept of lookahead has been introduced, this section contains a brief and simplified explanation of how each time-slice graph is generated for the FGP-rOEE algorithm and the approach taken in this thesis.

The way the graphs are formed for each time-slice, given a quantum circuit using lookahead, is as follows:

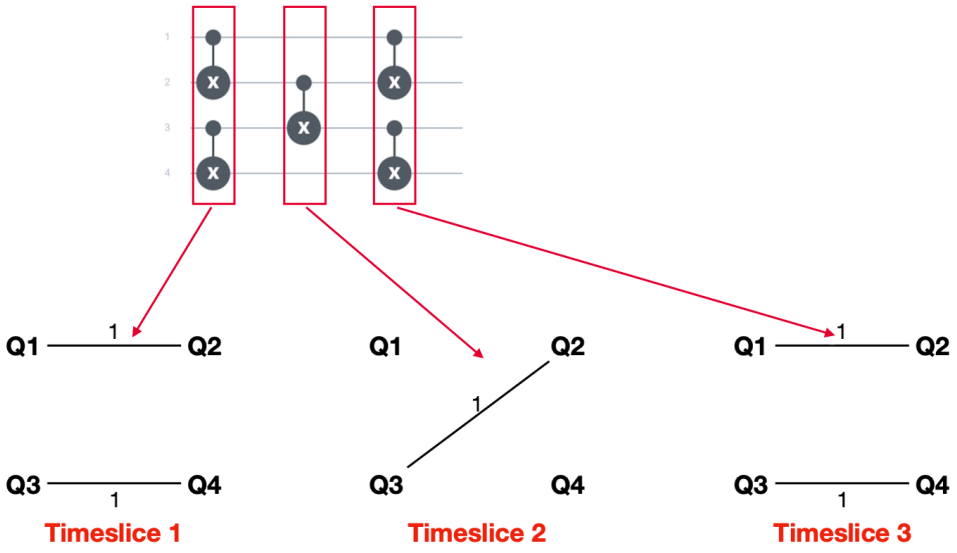

Figure 3.1: Illustration of the graph formation for each time-slice of a quantum circuit

Figure 3.1 shows, at the top, the quantum circuit with its logic gate interactions, where each horizontal line represents a qubit. Then the circuit is segmented into time-slices (each red box), and for each of them, a graph is created. Each node represents a qubit that will have an edge with another one if they are involved in the same logic gate in the circuit for that specific time-slice.

Once all the graphs are formed for each time-slice, the lookahead scheme is applied so that each time-slice has information about the behaviour of the qubits in future slices.

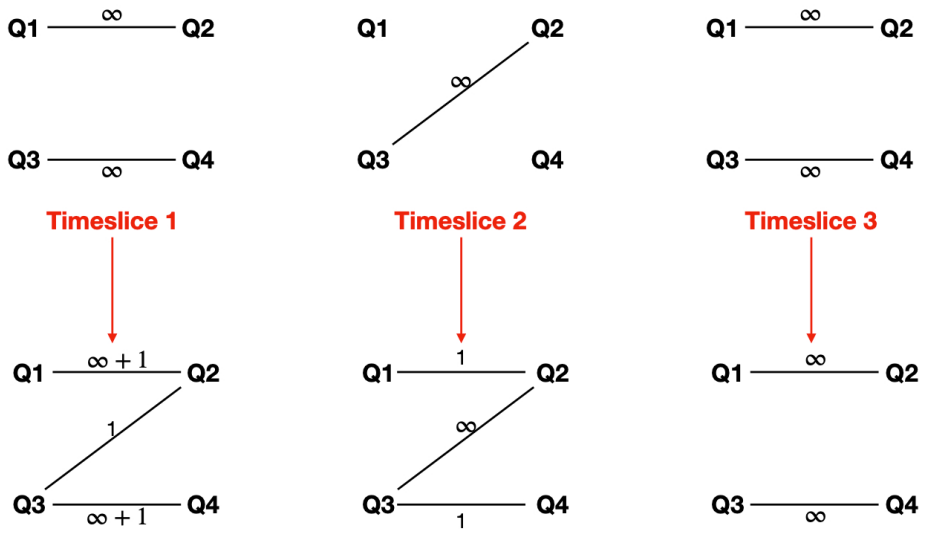

Figure 3.2: Integration of lookahead function in each time-slice

As shown in Figure 3.2, first, all the edges of the original graphs change weights to infinity to emphasize that those qubits need to be in the same machine. Then, for each time-slice, the lookahead function is applied, and new edges are created based on the subsequent slices. The value of the weight of these new edges will depend on the variation of the lookahead function, depending on how much relevance is given to future slices.

3.4 Limitations

While the FGP-rOEE algorithm has demonstrated significant advancements in quantum circuit partitioning, it is not without limitations. One major limitation is scalability. As the size and complexity of quantum circuits increase, the algorithm may face challenges in efficiently partitioning large circuits within a reasonable execution time.

Scalability is a crucial aspect to consider in quantum circuit partitioning algorithms as it directly impacts the practicality of implementing these algorithms in real-world quantum computing systems.

The objective of this thesis is to overcome these limitations by incorporating similar ideas to the FGP-rOEE algorithm, such as using swaps as actions and utilizing lookahead graphs. However, the proposed model aims to address scalability issues by leveraging the benefits of deep learning, where once the model is trained, making predictions becomes a matter of performing simple and easily parallelizable operations.

Chapter 4

Experimental Setup

4.1 Dataset Description

In the entire process of training the model, the dataset used consisted of randomly generated circuits tailored to each stage of development in terms of qubits and complexity. The decision to employ random circuits in the dataset can be attributed to the unavailability of a sufficiently large number of known circuits with the required number of qubits. While known circuits could potentially offer specific insights and patterns, the scarcity of such circuits poses a limitation in terms of generating a diverse and extensive dataset for training and evaluation purposes. Furthermore, the use of random circuits allows us to explore the generalizability of our approach across various circuit configurations and complexity levels.

Nevertheless, the possible limitations of this choice are acknowledged. Since random circuits lack the targeted design and intentionality of known ones, it may be challenging to gain a deep understanding of the underlying patterns and interactions that real circuits have.

4.2 Observation and Action Space

Obervation

The observation will have a critical impact on the performance of the agent, as it is the input to the policy and value neural networks, as explained in Section 2.2.1.

During the exploration of different approaches, various elements of the observation have been analyzed. However, certain elements have remained constant as they provide essential information necessary for solving the problem satisfactorily. These are:

- **Lookahead matrix**: described in Section 3.2, provides great insights into the constrained problem of having the qubits involved in the same logical gate on the same machine. It also offers a future perspective for the upcoming time-slices, which is very useful when the ultimate objective is to minimize the overall changes across all the time-slices.
- **Current assignment**: describes the allocation of each qubit to a specific machine.

Some other information that has been added to certain approaches, resulting in significant improvements, includes:

- **Validitiy of current assignment**: a binary value, with value 1 if the current assignment was valid[1], 0 otherwise.
- **The assignment at the start of the time-slice**: describes the allocation of each qubit at the end of the previous time-slice and at the beginning of the current one.

Since all these elements were used as the input for neural networks, their structures and values were flattened and concatenated together.

Action

The actions are also crucial as they allow the agent to interact with the environment, and choosing the right actions leads to successful agent performance.

When designing the OpenAI Gym environment, different types of actions were considered. Rather than opting for a highly complex action, such as assigning each qubit to a machine per time-slice, it was considered preferable to simplify the actions as much as possible. Following the approach of FGP-rOEE [4], it was decided that the most promising action was to perform swaps between two qubits. This action, with a balanced initialization, satisfied the constraint of maintaining an equal number of qubits in each machine.

Now a mechanism was required to indicate the completion of swaps and progress to the next time-slice. Given that RL is commonly used for creating agents to play video games, it was chosen to gamify the problem and treat it as a game. Each time-slice was treated as a level, allowing the agent to perform as many actions as desired. Once the partition was valid and the agent considered the current solution to be satisfactory, it took action to advance to the next level, which in this case represented the following time-slice.

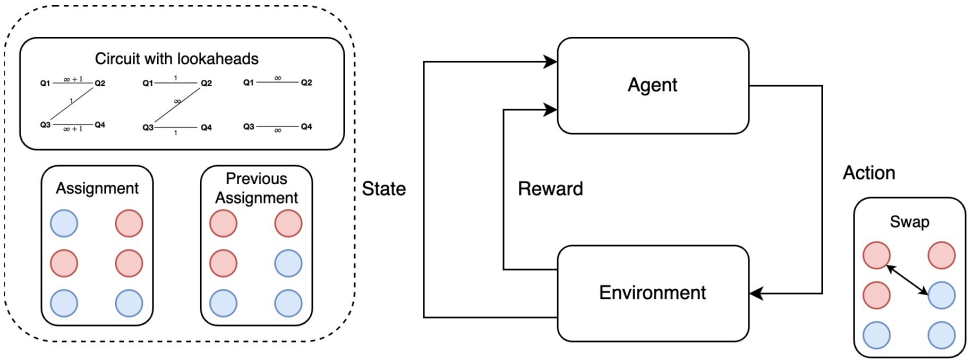

Figure 4.1: Block diagram of the DRL model

4.3 Evaluation Metrics

In order to compare different approaches and quantify the value of each one, evaluation metrics are necessary. In this case, two metrics indicate the degree of the success of a model.

The first metric is binary and indicates, for each episode, whether the agent found a valid partition for each time-slice, also considered **game finished**. It is assigned a value of 1 if a valid partition is achieved, and 0 if the agent becomes stuck in a time-slice where it cannot find a valid assignment and reaches the maximum permitted actions per time-slice (explained

[1] A valid assignment means that all the qubits involved in the same logical gate for the current time-slice are on the same machine.

in Section 4.4). By collecting this metric for a large number of episodes, the true mean of successfully completed episodes can be computed. This metric holds significant importance in measuring the performance of each model, as the absence of a valid assignment for any time-slice makes the partial assignments meaningless.

The second most important metric is the **mean Hamming distance per time-slice** for each episode. This metric provides valuable insights into the agent's performance, aligning with the primary objective of this thesis, which is to minimize qubit movements between machines during the execution of a circuit in order to reduce overall latency. By reducing the mean Hamming distance, the aim is to achieve a more efficient allocation of qubits, optimizing their placement and reducing the need for excessive movements throughout the computation process. It is important to note that the metric is only produced when a valid assignment has been provided for all time-slices, as having this metric with an invalid assignment wouldn't make sense.

Additionally, another important metric used to evaluate the success of a DRL model is the **episode mean reward**. While the aforementioned metrics are crucial, the episode mean reward serves as an indicator of whether the incentives are aligned and the agent is effectively learning. By analyzing the rewards obtained over multiple episodes, it can be assessed the learning progress and the effectiveness of the chosen approach. This metric complements the previously mentioned metrics, offering a comprehensive evaluation of the model's overall performance.

4.4 Finishing mechanisms

In RL, episodes can be terminated or finished in various ways based on specific conditions. In the gamified version of the problem, aforementioned in the action choice in Section 4.2, there are two ways an episode can end.

The first mechanism involves the successful completion of all time-slices with valid assignments. When the agent achieves a valid assignment for every time-slice and chooses the action to progress to the next time-slice in the last one, the episode is considered successfully finished. This mechanism ensures that the final solution obtained by the agent is valid and satisfies all the constraints of the problem.

The second mechanism involves setting a maximum number of actions per time-slice. This approach is inspired by the concept of limiting the maximum time per episode, as discussed in the paper by Pardo [16]. By imposing a maximum number of actions, the episode terminates when the agent reaches the predefined limit for a time-slice. This maximum number of actions permitted is restarted every time-slice. The maximum number of actions is calculated as follows:

$$\begin{aligned} MAX\_ACTIONS &= \frac{n}{2} \times C_m^n, \\ C_m^n = \binom{n}{m} &= \frac{n!}{m!\,(n-m)!}, \end{aligned} \tag{4.1}$$

where n is the number of qubits in the circuit and m is the number of machines.

Equation (4.1) indicates the worst-case number of swaps needed to be performed in one time-slice given a specific configuration of qubits and machines. C_m^n represents the number of different possible assignments, and it is multiplied by half the number of qubits because, by definition, in each slice, each qubit is involved with at most one two-qubit gate. Therefore, in the worst-case scenario, every qubit is involved in one, resulting in a maximum of $\frac{n}{2}$ two-qubit gates.

4.5 Hyperparameter tuning

In RL and in deep learning in general, hyperparameters play a crucial role in the performance and convergence of models. A proper selection and tuning of hyperparameters is essential to achieve optimal model performance. In this case, it is important to take into account general hyperparameters as well as PPO-specific ones.

Some of the most important hyperparameters are:

- **Learning Rate**: determines the step size at which the model's parameters are updated based on the computed gradients. It controls the speed of learning and can significantly impact the convergence and stability of the training process.
- **Discount Factor (Gamma)**: determines the importance of future rewards relative to immediate rewards. A high discount factor gives more weight to long-term rewards, while a low discount factor prioritizes immediate rewards. The choice of gamma depends on the task and the agent's preference for short-term or long-term rewards.
- **Value Function Coefficient (VF Coef)**: balances the importance of the value function loss in the overall objective function. It controls how much the policy and value networks influence each other during training. Tuning this coefficient affects the trade-off between exploration and exploitation.
- **Entropy Coefficient (Entropy Coef)**: encourages exploration by adding an entropy term to the objective function. It controls the level of exploration, with higher values promoting more exploration. The entropy coefficient affects the agent's ability to explore new actions and potentially discover better policies.
- **Clip Range**: implements a clip on the surrogate objective function to prevent large policy updates. It restricts the policy update to a certain range, ensuring stability during training. The choice of clip range influences the balance between exploiting the current policy and exploring new actions.

The concrete process of hyperparameter tuning involved varying the values of the hyperparameters and evaluating the performance of the model for each combination. The process consisted of iterative experimentation, analyzing the results, and refining the hyperparameter values to achieve the best possible performance and convergence in the given task.

After running multiple tests, the only hyperparameter that really made a significant difference when modifying from the default values, presented in the original paper introducing the algorithm [21], was the learning rate. Instead of using a fixed value, a scheduler was implemented. This scheduler gradually decreased the learning rate over time, which proved beneficial in avoiding local minima during the initial stages of training and enabled more cautious adjustments as training progressed.

4.6 Desired Behaviour

The ideal behaviour the agent should have to fulfil the objectives set in this thesis is as follows, sorted by priority:

1. Learn to find a valid assignment for all the time slices, which is considered as finishing the game.

2. Learn to minimize the overall difference in the assignment of consecutive time-slice (reduce the Hamming distance between time-slices).

The primary focus is on the agent's capacity to find valid assignments, ensuring that all slices meet the constraints. This capability is crucial, and if it is not achieved, the second objective of the agent becomes irrelevant. Minimizing the overall Hamming distance doesn't provide any value if not all the assignments are valid.

Chapter 5

Comparative Analysis

This chapter will discuss the most significant approaches explored in the thesis, including the initial hypotheses upon their design, a description of their specific details, and the results obtained. These approaches will be evaluated in relation to the desired objectives stated in Section 4.6.

5.1 Swap betweeen all qubits penalizing useless actions

A **useless action** is defined as a set of actions that result in no change to the observation. This includes actions such as swapping a qubit with itself, swapping qubits that are already assigned to the same machine, and choosing to pass to the next time-slice when the current assignment for that time-slice is invalid.

Hypothesis

The hypothesis for this approach is that penalizing useless actions will incentivize the agent to avoid them. Additionally, providing the agent with a high degree of freedom in its actions will enable it to extract deeper patterns and optimize the allocation of qubits.

Description

In this approach, the agent is allowed to perform swaps between all qubits as part of its action space. However, a penalty in the form of a small negative reward is imposed when the agent commits a useless action. By incorporating this penalty, it is expected that the agent will learn to optimize its actions, avoiding the useless ones, and instead focusing on actions that have an impact on the observation.

Results

By examining both Figure 5.1 and Figure 5.2, a quick conclusion can be drawn. The incentives of the agent and the thesis' goal are not aligned. This is deduced by looking at the upward reward curve in Figure 5.2, which indicates that the agent is learning since it consistently earns bigger rewards. However, Figure 5.1 clearly indicates the failure of the agent to find a valid assignment for all the time slices.

The result of this approach was not the expected one. Instead of learning to avoid useless actions, the agent exhibited a tendency to prioritize terminating the episode as quickly as possible. As a result, it repeatedly performed the same action until reaching the maximum number of allowed actions per time-slice, leading to the termination of the episode. This behaviour was likely

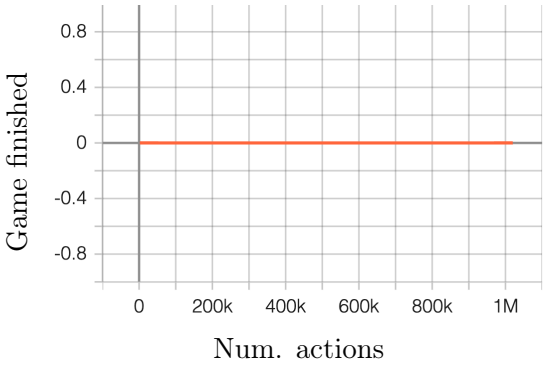

Figure 5.1: Game finished metric for the *Swap between all qubits penalizing useless actions* approach

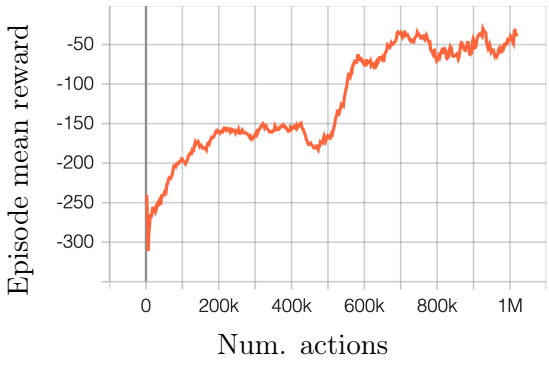

Figure 5.2: Episode mean reward metric for the *Swap between all qubits penalizing useless actions* approach

influenced by the penalty imposed on useless actions, leading the agent to adopt a strategy of early termination rather than focusing on using only performing valid actions.

5.2 Swap between all qubits with no penalizing actions

Hypothesis

The hypothesis behind this approach is to provide the maximum freedom to the agent, allowing it to perform swaps between all qubits without penalizing useless actions. The idea is to observe how the agent explores the action space without facing any negative consequences for useless actions. Additionally, by rewarding the agent solely for successfully passing to the next time-slice, it is expected to encourage the agent to prioritize advancing through the circuit.

Description

This approach represents a natural progression from the previous one. Since the success of the model is not affected by performing useless actions, the agent will only receive a reward for the action that truly matters, i.e., successfully passing to the next time-slice. This incentivizes the agent to prioritize advancing through the circuit rather than being overly concerned about the validity of each individual action.

By providing the agent with this level of freedom, it is expected that it will explore a wider range of possibilities, potentially discovering more efficient ways to allocate qubits and reduce the mean Hamming distance. However, it is important to note that the lack of penalization for useless actions can also lead to suboptimal behaviour, as the agent may perform a large number of unnecessary swaps without consequence.

Results

The results of this approach, shown in Figure 5.4, demonstrate that the agent is capable of learning and adapting its behaviour based on the given rewards. Additionally, Figure 5.3, shows that the agent achieves a high success rate in finding valid assignments for all the slices, with approximately 96% success.

However, a notable issue arises from Figure 5.5, which illustrates the reduction of the mean Hamming distance metric over time. While it initially decreases, it reaches a plateau around

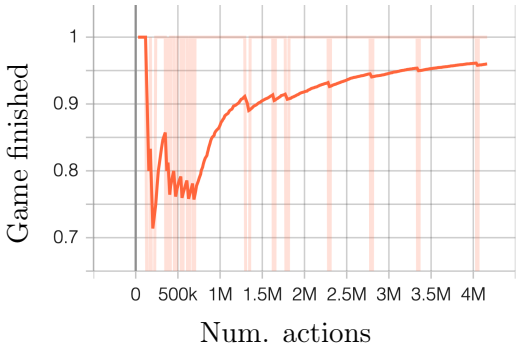

Figure 5.3: Game finished metric for the *Swap between all qubits with no penalizing actions* approach

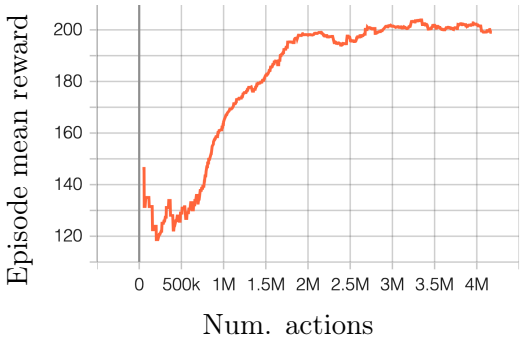

Figure 5.4: Episode mean reward metric for the *Swap between all qubits with no penalizing actions* approach

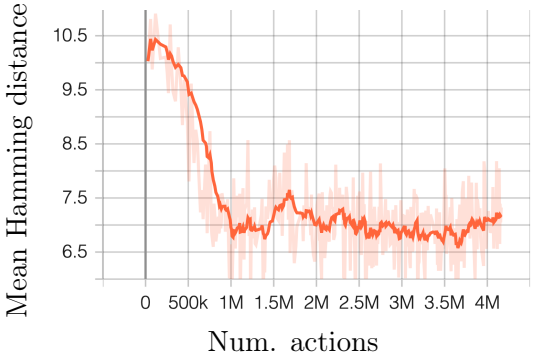

Figure 5.5: Mean Hamming distance per time-slice metric for the *Swap between all qubits with no penalizing actions* approach

the value of 7, suggesting that the agent struggles to further optimize the allocation of qubits. This stagnation in the learning process highlights the necessity for additional analysis and improvements to enhance the agent's performance.

5.3 Loosely restricted mask: no swaps that do not have an impact on the observation

Hypothesis

Noting that agents with a large set of available actions may not perform optimally, the hypothesis of this approach is that by introducing a mask that disables a set of actions, the model will converge earlier and exhibit improved performance.

Description

In this approach, the goal is to prevent the agent from performing useless actions by implementing a mask that restricts them. Specifically, a variation of the Proximal Policy Optimization (PPO) algorithm called MaskablePPO [8] is employed (see Section 2.2.1 for details on the algorithm).

The mask is designed to disable the subset of actions that have no impact on the observation. By removing these actions from the agent's available action space, it is expected that the agent

will converge faster and exhibit more efficient behaviour.

Results

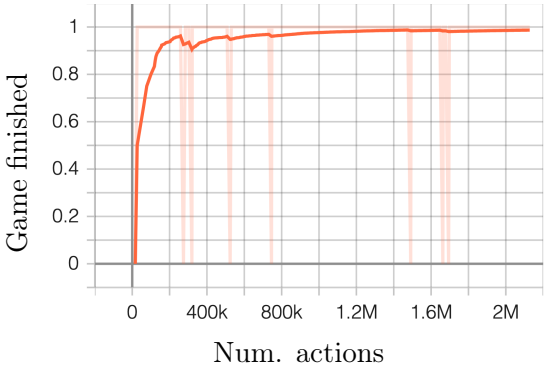

Figure 5.6: Game finished metric for the *Loosely restricted mask: no swaps that do not have an impact on the observation* approach

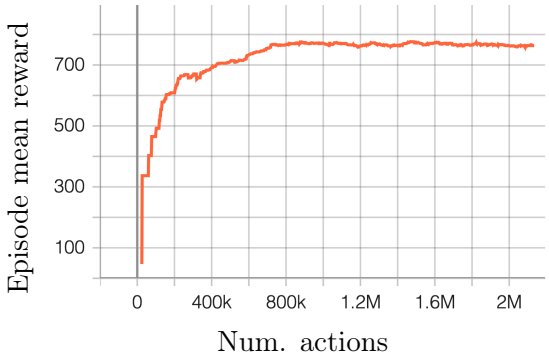

Figure 5.7: Episode mean reward metric for the *Loosely restricted mask: no swaps that do not have an impact on the observation* approach

The results of this approach demonstrate that the agent exhibits a significantly improved performance compared to previous approaches. Figure 5.6, exposes a huge success in finishing the "game", with a 98% of games finished.

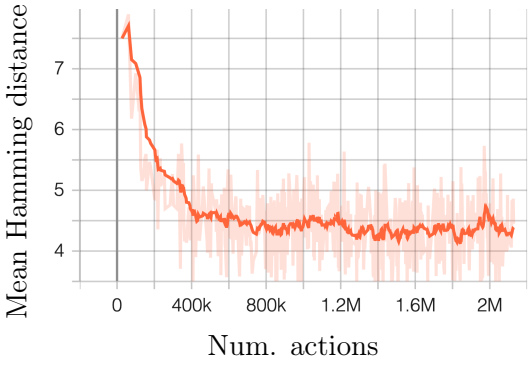

Figure 5.8: Mean Hamming distance per time-slice metric for the *Loosely restricted mask: no swaps that do not have an impact on the observation* approach

In Figure 5.7, it is observable that the reward shows a steep upward tendency until the 800k timesteps, indicating significant learning improvement by the agent. A very good sign that the incentives of the agent and the goal of the thesis are aligned is when the plot of rewards (Figure 5.7) and the plot of the mean Hamming distance (Figure 5.8) have almost the same shape but are inverted.

Finally, Figure 5.8 shows that the agent learns to navigate the action space effectively and converges to a satisfactory solution, with a mean Hamming distance per time-slice that is almost half of the value of the previous approach (Section 5.2). However, a limitation observed is that the agent still gets stuck at the specific mean Hamming distance of 4.5, indicating its struggle to achieve further optimization.

5.4 Restrictive mask

Hypothesis

The hypothesis behind this approach is that by implementing a more restrictive mask on the agent's action space and guiding it towards a good solution, the model will be able to find sub-optimal assignments while still ensuring a valid assignment for each time-slice.

Description

In this approach, a more restrictive mask is applied to the agent's action space to focus its search on specific actions that are crucial for finding optimal solutions. The mask is designed to solve the most general problem this thesis wants to solve.

The goal of the thesis is to have the minimum mean Hamming distance possible for all the time-slice. However, the key challenge lies in efficiently reallocating the qubits being swapped when putting in the same machine the constrained pair of qubits. The strategic selection of which qubits should be moved to other machines can greatly impact the number of future changes required in the following time-slices.

To address this challenge, the following restrictions were applied to the mask:

- **Useless actions**: swapping a qubit with itself, swapping qubits already assigned to the same machine, and choosing to pass to the next time-slice when the current assignment for that time-slice is invalid.
- **Qubits already in the same machine**: actions involving constrained qubits that are already in the same machine.
- **Moving constrained qubits to different machines**: when there is enough space in at least one of the constrained qubits' machines, only swaps involving one of the constrained qubits and the qubits allocated in that specific machine are permitted.

These restrictions significantly reduced the number of possible actions given the state of the environment, allowing the agent to focus on the most relevant and impactful actions.

Results

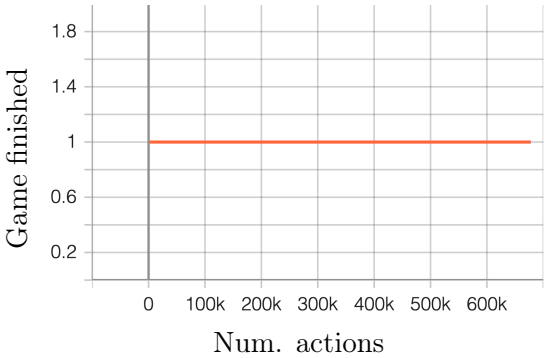

Figure 5.9: Game finished metric for the *Restrictive mask* approach

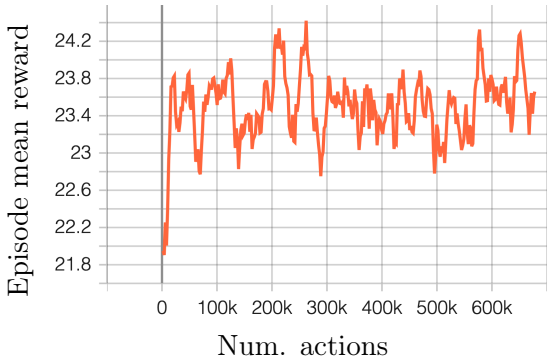

Figure 5.10: Episode mean reward metric for the *Restrictive mask* approach

The results obtained from this approach were highly promising, as seen in Figure 5.11. The agent consistently found valid assignments for all time-slices (Figure 5.9), and the achieved mean

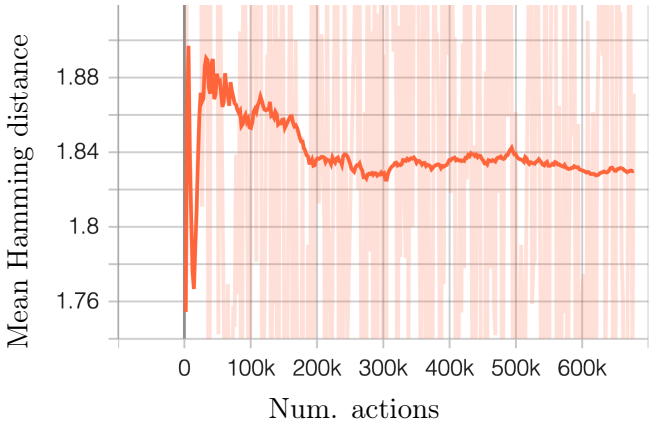

Figure 5.11: Mean Hamming distance per time-slice metric for the *Restrictive mask* approach

Hamming distance was significantly low. At first glance, the approach appears to yield similar results to FGP-rOEE, a state-of-the-art algorithm (see Chapter 3). These positive outcomes suggest that the restrictive mask approach effectively balances the trade-off between finding valid assignments and minimizing the Hamming distance.

On the other hand, when examining the learning curve of the agent, represented by the episode mean reward in Figure 5.10, it becomes apparent that the agent is not experiencing significant learning. With fair degree of certainty, this phenomenon can be attributed to the limited number of actions permitted, which directs the agent towards a very good solution but restricts its ability to learn deeper underlying patterns.

Chapter 6

Results and Discussion

In this chapter, an examination of the results and discussion of this thesis will be presented. The main focus revolves around providing a comprehensive comparison among the four approaches outlined in Chapter 5. Additionally, the influence of the reward function on the agent's behaviour will be explored, accompanied by a detailed summary of all attempted modifications and their respective outcomes.

To conclude, a comparison will be made between the approach yielding the best results and the state of the art algorithm FGP-rOEE, utilizing real circuits. This comparative analysis aims to offer valuable insights and evaluate the effectiveness of the proposed approach.

6.1 Performance Comparision of Different Approaches

This section presents a comparison among the four different approaches for solving the problem introduced in Chapter 5.

For a more comprehensive and clearer comparison, the approaches will be labelled following the order in which they were presented:

- **Approach 1**: Swap between all qubits penalizing useless actions (Section 5.1)
- **Approach 2**: Swap between all qubits with no penalizing actions (Section 5.2)
- **Approach 3**: Loosely restricted mask: no swaps that do not have an impact on the observation (Section 5.3)
- **Approach 4**: Restrictive mask (Section 5.4)

For all the approaches, the absolute values of the metrics *Game finished* and *Mean Hamming distance per time-slice* will be compared. This comparison is possible because all the approaches have been trained with random circuits that have the same characteristics (16 qubits and 4 machines). However, when it comes to the metric *Episode mean reward*, comparing absolute values becomes problematic. This is due to the fact that different reward functions have been employed for each approach. As a result, the analysis of this metric will focus on identifying underlying tendencies rather than direct numerical comparisons.

By analyzing all the approaches and considering the metrics of *Game finished* and *Mean Hamming distance per time-slice*, it can be determined that the objectively best approach is Approach 4 (refer to Section 5.4). Similar to Approach 3, it always finds a valid assignment of all the slices,

but additionally, it achieves a mean Hamming distance of approximately 1.83, which is 59% better than the results of Approach 3 and 74% better than Approach 2.

It is clear that the best approach, considering the objective of the thesis and for later comparison with the state-of-the-art in Section 6.4, is the fourth one. However, if further optimization is desired, this approach would not be the best starting point. As stated in its respective Section 5.4, the agent has a very limited will, which leads to excellent results but restricts its ability to learn and discover better strategies. A better starting point would be Approach 3 (Section 5.3) because it masks actions that have no impact (useless actions), and the agent's metrics demonstrate that it is indeed learning and finding notable strategies.

6.2 Analysis of Reward Function Modifications

The reward function plays a vital role in reinforcement learning by providing feedback to the agent based on its actions and the state of the environment. It serves as a signal to reinforce desired behaviours and discourage undesirable ones. However, designing an effective reward function can be a challenging task, as it requires a careful balance between providing enough information for the agent to learn while avoiding the introduction of unintended biases or incentives.

As described in Section 5.1, the approach of *Swap between all qubits penalizing useless actions* reveals that an initial idea that seemed logical, can lead to unintended behaviour. Specifically, in this case, the agent observed that the more it lived, the more it engaged in actions that resulted in penalties, resulting in diminished rewards. Consequently, the agent devised a strategy to terminate the game quickly by repeatedly executing a non-penalizing action until the maximum number of actions per time-slice was reached, thereby ending the game.

After extensively experimenting with various modifications of the reward system, including different approaches to immediate and final rewards, as well as determining the optimal combination of variables and their respective weights to achieve the desired goal, it becomes evident that simplicity is key. A complex reward system often leads to situations where the agent's incentives diverge from the intended objectives. Therefore, it can be concluded that a simpler reward system is generally more effective in ensuring alignment between the agent's actions and the desired outcomes.

However, the temporal aspect of rewards is another consideration. In some cases, relying solely on final rewards can be challenging, as it may take a long time for the agent to receive feedback on its behaviour. This delayed feedback can lead to slow learning or difficulty in establishing the connection between specific actions and their consequences. To mitigate this, a combination of immediate and final rewards can be used, where immediate rewards provide guiding feedback, and final rewards provide a more comprehensive evaluation.

Another key aspect of the reward mechanism is the difference in magnitude of the different rewards. Even though it is important to significantly weigh and emphasize the rewards that are more directly related to the ultimate goal of the agent, such as the final one, it is important to take into account that highly weighting some rewards diminishes the value of rewards with smaller values.

6.3 Modification Analysis

In the process of developing this thesis, an extensive modification analysis was conducted on the agent in order to enhance its performance. A wide range of diverse modifications were implemented, aiming to maximize the agent's improvement and effectiveness. These modifications encompassed various aspects of the agent's observation and reward system.

However, it is important to note that not all modifications yielded favourable results. As expected in any iterative development process, some modifications did not have the desired impact and, in certain cases, even had a negative effect on the agent's performance. This outcome emphasizes the necessity of rigorous analysis and evaluation of each modification before its implementation.

Furthermore, there were modifications that demonstrated mixed results, with varying degrees of success. These modifications showed improvements in certain aspects of the agent's performance while highlighting areas that still required attention or further refinement. The analysis of these mixed results helped in subsequent iterations and guided the development process towards more effective modifications.

The modifications were approached with a systematic methodology, taking into consideration the specific objectives of the thesis. This involved conducting extensive experimentation and testing to measure the agent's performance under different scenarios and conditions. Quantitative metrics, such as the evolution of the episode mean reward, the percentage of games won, the mean Hamming distance per time-slice and the execution time, were used to objectively evaluate the impact of each modification.

In addition to quantitative metrics, qualitative feedback from the thesis supervisor, Sergi Abadal Cavallé, and domain experts like Pau Escofet i Majoral and Pere Barlet Ros was also solicited to gather insights regarding the agent's performance after each modification. This feedback played a crucial role in identifying potential issues or areas for further improvement that may not have been captured by the quantitative metrics alone.

Concept	Description	Impact
Negative rewards useless op.	Prevent the agent from doing useless actions by giving them a negative reward	Negative
Immediate reward on valid swaps	Give an immediate reward when performing a swap based on the difference value of the graph cut	None
Only finishing episode when game won	Mechanisms that force that the only way of finishing an episode is by winning it	Negative
Penalising the agent when it "died"	If the episode finishes by reaching the max. number of actions permited, penalise the agent	None
Normalise the rewards	Normalise the reward system, so in each step, the maximum reward obtainable is 1 and the minimum zero	Low
Only final reward	The reward is only given when finishing the game, based on the results	Negative
Reward for moving to the next time-slice	Give a positive reward when finding a valid assignment and moving on to the next one, based on the Hamm. distance obtained	High
Add the coverage of the cut as part of the reward	Add the percentage of the weights covered as a reward when passing a time-slice	Medium
Add mask	Use MaskablePPO to create a mask to guide the agent	High
Add the last assignment as part of the observation	Add the assignment given at the start of the time-slice as part of the observation of the agent	Medium
Add the current Hamming distance as observation	Include the Hamming distance with the current assignment as part of the observation	Low
Add the validity of the assignment as observation	Add one bit to the observation that will be one in case the current assignment is valid, 0 otherwise	Medium

Table 6.1: Modification analysis

Table 6.1 presents a comprehensive summary of the significant modifications and their impact. It should be noted that modifications with a negative impact were obviously discarded, as were those without any impact, in order to simplify and minimize the agent's complexity. All modifications with a positive impact were implemented to some extent, with certain adjustments made to accommodate others.

6.4 Comparison with FGP-rOEE

To evaluate the effectiveness of the proposed approach, a comparison will be conducted with the state-of-the-art algorithm FGP-rOEE (explained in Chapter 3). This comparative analysis will involve three benchmark circuits: Random, Cuccaro, and QAOA. The performance of the proposed approach and FGP-rOEE on these circuits will be examined to assess their respective capabilities and performance. This comparison aims to provide valuable insights into the strengths and limitations of both algorithms in the context of different circuit types and architecture configurations.

The metric chosen for evaluation is the mean Hamming distance per time-slice. This choice is based on the findings in Section 6.1, which demonstrated that the approach with the most

restrictive mask (described in Section 5.4) consistently achieves a valid assignment for all slices in all cases. Therefore, considering a valid assignment as the baseline, the aim is to compare the maximum reduction in movement between cores.

Id. config	Num. qubits	Num. machines
1	8	2
2	16	4
3	32	8
4	64	16
5	128	32

Table 6.2: Benchmark configurations for a number of qubits and machines

The different configurations utilized for benchmarking all three circuits are displayed in Table 6.2. In each configuration, four qubits are allocated to each machine, which exemplifies a case of horizontal scaling. This implies that the aim is to test the scenario where the processing power is enhanced by adding more machines to the system's architecture.

To ensure a fair comparison and accurately determine which approach yields better results, each approach is executed using exactly the same circuits, starting with identical initial random assignments. Since the initial assignments are random and may have varying effects, each circuit is executed multiple times, and the final result is obtained by calculating the mean of all executions. This approach allows for a more comprehensive evaluation and reduces the influence of random variations on the overall assessment.

Random

The randomly generated circuits are created using the same methodology as the one used to generate the training dataset. However, to ensure fairness and maintain the integrity of the tests, these circuits were not included in it.

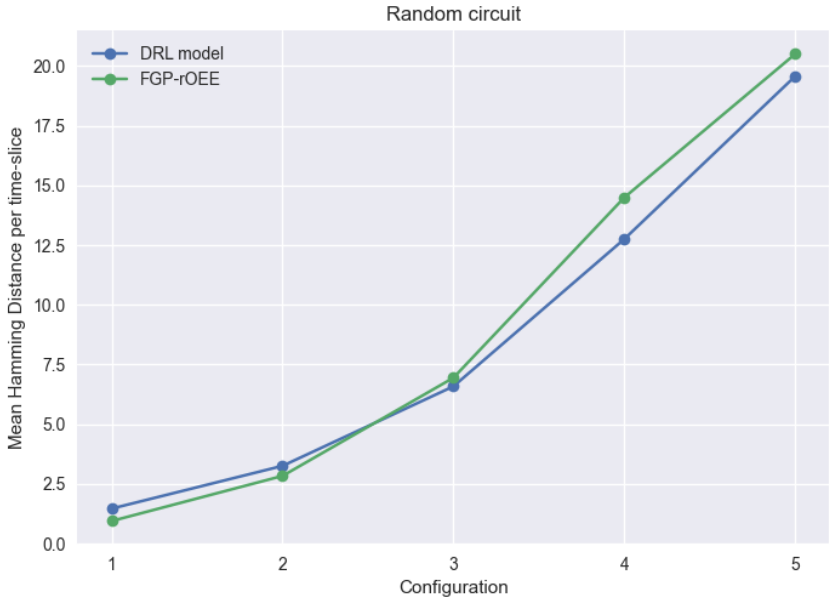

Figure 6.1: Benchmark using random circuit

The benchmark results for the random circuit, as presented in Figure 6.1, demonstrate that FGP-rOEE achieves better results for smaller configurations. However, these advantages diminish as the configuration size increases.

Num. qubtis	Num. machines	DRL model	FGP-rOEE	Improvement
8	2	1.46	0.94	-55 %
16	4	3.25	2.83	-15 %
32	8	6.58	6.94	5 %
64	16	12.75	14.49	12 %
128	32	19.57	20.50	5 %

Table 6.3: Results and improvement for random circuit comparison

When examining the values in the "Improvement" column of Table 6.3, which indicates the improvement achieved with the DRL approach compared to FGP-rOEE, an upward trend can be observed. This trend is correlated with the increase in the number of qubits and machines.

Cuccaro

Cuccaro is a quantum circuit for addition proposed by Steven Cuccaro et al. in 2004 [5]. This circuit was generated multiple times to fit the different specifications of each configuration described in Table 6.2.

The characteristic of this circuit is that it is not very extensive in terms of two-gate operations. This means that in most cases, there is no need to perform a large number of actions to find a valid assignment. This can be easily deduced by examining the mean Hamming distance values for both the DRL model and the FGP-rOEE algorithm in Table 6.4.

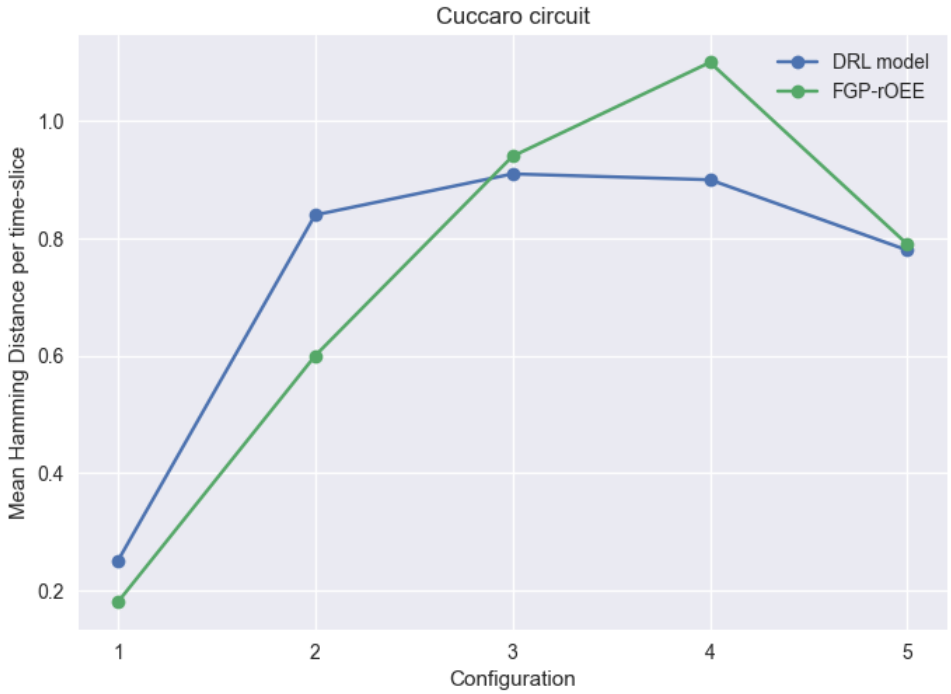

Figure 6.2: Benchmark using Cuccaro circuit

The plot in Figure 6.2 exhibits a similar behaviour to the plot in Figure 6.1 obtained with a random circuit, but with more pronounced differences. The figure demonstrates that, for the first two configurations, the FGP-rOEE model had a significant distance that is surpassed by a substantial margin in configuration 4 as the size increases.

Num. qubtis	Num. machines	DRL model	FGP-rOEE	Improvement
8	2	0.25	0.18	-39 %
16	4	0.84	0.60	-40 %
32	8	0.91	0.94	4 %
64	16	0.90	1.10	18 %
128	32	0.78	0.79	1 %

Table 6.4: Results and improvement for the Cuccaro circuit comparison

The tendency shown by the improvement percentage in Table 6.4 is also very similar to the obtained with random circuits.

Delving deep into the specific results of each approach, it appears that the strategy of the agent in the DRL model was to make minimal changes in the assignment in each time-slice to achieve a valid assignment. Conversely, FGP-rOEE, in most cases, performs a significant reassignment that allows it to avoid any further changes in future slices.

QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm that can be used to find approximate solutions to combinatorial optimization problems [7].

Following the same methodology, multiple versions of the circuit were generated to match the exact specifications as the ones described in Table 6.2.

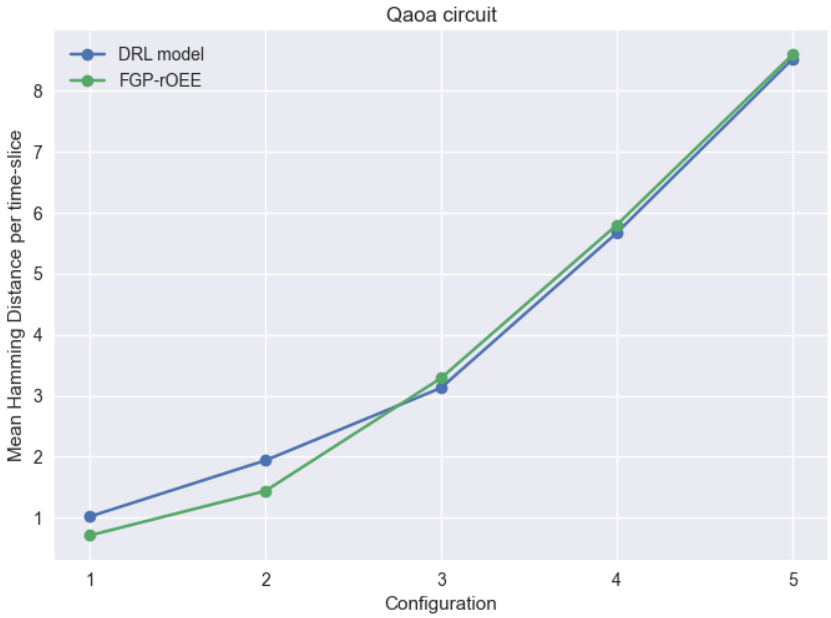

Figure 6.3: Benchmark QAOA circuit

Figure 6.3 illustrates that for this circuit, the performance of both algorithms was considerably similar. Once again, in the first two configurations, FGP-rOEE demonstrated a small advantage. However, beyond that point, the results of both approaches were practically indistinguishable, with the DRL model exhibiting a slight edge.

Num. qubtis	**Num. machines**	**DRL model**	**FGP-rOEE**	**Improvement**
8	2	1.02	0.71	-44 %
16	4	1.94	1.44	-37 %
32	8	3.13	3.29	5 %
64	16	5.67	5.80	3 %
128	32	8.52	8.59	1 %

Table 6.5: Results and improvement for the QAOA circuit comparison

The improvement percentage in Table 6.5 confirms the aforementioned analysis, showing a similar tendency to the random and Cuccaro circuits, but with a much-reduced disparity between them.

Chapter 7

Conclusions

Reviewing the Requirements Subsection 1.3.3, which outlined the necessary criteria for the thesis to be considered successful, it can be concluded that the objectives have been achieved. Firstly, the development of a reinforcement learning model capable of accurately generating valid partitions with a minimum accuracy of 95% has not only been met but exceeded. The model consistently produced valid partitions for all time-slices in every experiment conducted, covering a diverse range of circuits and configurations. Secondly, the requirement to outperform FGP-rOEE's algorithm [4] in terms of reducing qubit movement between cores and scalability was also met. As detailed in Section 6.4, the model showcased superior performance compared to FGP-rOEE for medium-sized circuits in terms of qubits. Through our research, invaluable insights have been gained about the practical application of RL techniques in the field of partitioning algorithms.

In conclusion, this thesis presents a novel approach that contributes to the field of partitioning algorithms for quantum applications by successfully developing an RL model that achieves high accuracy in making valid partitions and outperforms existing algorithms in terms of execution time and scalability. By addressing the identified limitations, Section 7.1, and exploring future research directions, Section 7.2, the performance, efficiency, and applicability of RL-based partitioning models for quantum applications can be further improved. The use of this novel technique opens up a lot of opportunities for their utilization in a wide range of real-world scenarios.

7.1 Limitations

The project encountered several limitations that affected the performance and scalability of the proposed approach. These limitations revolved around the trade-off between the amount of freedom the agent has and learning performance, as well as the computational power required for training the reinforcement learning model with a large number of qubits.

One of the observed limitations was the decrease in the amount of learning the agent achieved when its actions were strictly limited. It was noticed that allowing the agent to have fewer restrictions in its actions resulted in better learning curves. However, those approaches came at the cost of lower performance. On the other hand, imposing more restrictions on the agent's actions improved its effectiveness but limited the learning process.

Another significant limitation was the computational power needed for training the model with a large number of qubits. The project faced road-blocking difficulties when attempting to train

with datasets of circuits with more than 128 qubits due to limited computational resources. This limitation directly impacted the scalability of our approach, as it prevented us from exploring larger problem instances. Overcoming this limitation would require leveraging more powerful computing infrastructure or utilizing specialized hardware designed for training deep learning models, such as Tensor Processing Units (TPUs).

7.2 Future work

Improving the performance of the RL model is an area that requires further attention and enhancement in future work. Various possibilities can be explored to achieve this objective. Firstly, it is essential to investigate more RL algorithms, as well as, be on the lookout for new ones since RL is an evolving technology that is changing day by day. It is possible that newer algorithms may provide better performance and help identify approaches that could be more effective in addressing the partitioning problem. Through experimentation with alternative algorithms, valuable insights can be obtained leading to superior performance outcomes.

Moreover, obtaining similar results with more pronounced learning curves is crucial to ensure that the agent is indeed learning. This process may involve adjusting the reward system, refining the observation methods, or expanding the range of permissible actions. Additionally, trying alternative neural network architectures and performing better hyperparameter tuning can contribute to fine-tuning these aspects of the RL model, leading to more optimized and near-optimal results.

Finally, to continue with the research, it would be very interesting to train the RL model on larger circuits using more powerful computing resources. This approach enables testing the model's scalability and performance under more demanding conditions. By subjecting the model to larger and more complex circuits, its ability to handle increased computational requirements can be evaluated, ensuring its viability in real-world scenarios.

Bibliography

[1] Kai Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240.

[2] *Average Early-Career Machine Learning Engineer Salary in Spain*. URL: https://www.payscale.com/research/ES/Job=Machine_Learning_Engineer/Salary/c8bbe6e7/Early-Career. (accessed: 12.03.2023).

[3] *Average Project Manager, Information Technology (IT) Salary in Spain*. URL: https://www.payscale.com/research/ES/Job=Project_Manager%5C%2C_Information_Technology_(IT)/Salary. (accessed: 12.03.2023).

[4] Jonathan M. Baker et al. "Time-sliced quantum circuit partitioning for modular architectures". In: *Proceedings of the 17th ACM International Conference on Computing Frontiers*. ACM, May 2020. DOI: 10.1145/3387902.3392617. URL: https://doi.org/10.1145%2F3387902.3392617.

[5] Steven A. Cuccaro et al. "A new quantum ripple-carry addition circuit". In: (2004). arXiv: quant-ph/0410184 [quant-ph].

[6] Eric. *How Long Does a MacBook Pro Last?* URL: https://www.macbookproslow.com/how-long-macbook-pro-last/. (accessed: 14.01.2013).

[7] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[8] Shengyi Huang and Santiago Ontañón. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms". In: *CoRR* abs/2006.14171 (2020). arXiv: 2006.14171. URL: https://arxiv.org/abs/2006.14171.

[9] IBM. *IBM Quantum Computing: Roadmap*. Oct. 2015. URL: https://www.ibm.com/quantum/roadmap.

[10] IBM. *IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two*. URL: https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two. (accessed: 03.03.2023).

[11] Intellabs. URL: https://intellabs.github.io/coach/components/agents/policy_optimization/ppo.html.

[12] Nicholas LaRacuente et al. "Modeling Short-Range Microwave Networks to Scale Superconducting Quantum Computation". In: (2022). DOI: 10.48550/ARXIV.2201.08825. URL: https://arxiv.org/abs/2201.08825.

[13] Yuxi Li. "Deep Reinforcement Learning: An Overview". In: *CoRR* abs/1701.07274 (2017). arXiv: 1701.07274. URL: http://arxiv.org/abs/1701.07274.

[14] *MacBook Pro*. URL: https://www.apple.com/es/macbook-pro-14-and-16/specs/#footnote-3. (accessed: 12.03.2023).

[15] Anabel Ovide González. "Multi-core Quantum Computers: Analyzing the mapping of quantum algorithms". MA thesis. University of Tartu, 2022.

[16] Fabio Pardo et al. "Time Limits in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4045–4054. URL: https://proceedings.mlr.press/v80/pardo18a.html.

[17] Taehoon Park and Chae Y. Lee. "Algorithms for partitioning a graph". In: *Computers Industrial Engineering* 28.4 (1995), pp. 899–909. ISSN: 0360-8352. DOI: https://doi.org/10.1016/0360-8352(95)00003-J. URL: https://www.sciencedirect.com/science/article/pii/036083529500003J.

[18] J. Peters. "Policy gradient methods". In: *Scholarpedia* 5.11 (2010). revision #137199, p. 3698. DOI: 10.4249/scholarpedia.3698.

[19] *Precio de la luz por horas*. URL: https://tarifaluzhora.es/. (accessed: 12.03.2023).

[20] Mohan Sarovar et al. "Detecting crosstalk errors in quantum information processors". In: *Quantum* 4 (Sept. 2020), p. 321. DOI: 10.22331/q-2020-09-11-321. URL: https://doi.org/10.22331%2Fq-2020-09-11-321.

[21] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[22] John Schulman et al. "Trust Region Policy Optimization". In: *CoRR* abs/1502.05477 (2015). arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477.

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

[24] Ruizhe Yu Xia. "GNN for Time-Sliced Quantum Circuit Partitioning". MA thesis. Universitat Politècnica de Catalunya, 2022.

Appendix A

Time planning

To ensure the production of a high-quality thesis, a significant level of dedication is required. This thesis is equivalent to 18 ECTS, with each ECTS requiring an approximate workload of 30 hours. Thus, it is expected that a total of 540 hours of effort will be invested in this project.

This project started the 28/02/2020 and the expected date of ending is 26/06/2023. As such, there are 17 weeks to complete it.

A.1 Potential obstacles and risks

- Limited computing resources: deep learning models often require a substantial amount of computational resources. A shortage of resources may result in longer training times and may limit the number of different model architectures to experiment with.
- RL model not learning: a general potential risk when building machine learning models is that the model may not learn effectively, which can result in poor performance and inaccurate predictions.
- RL model overfits: similar to the aforementioned potential risk, but in the opposite direction, it can end up in a case of overfitting. Overfitting occurs when the model is trained too closely to the training data, resulting in poor generalization and performance on new data. This risk increase with the complexity of the model.
- Skill level: quantum computing is a highly complex field. Building a RL model that effectively fulfils its purpose requires a good understanding of both quantum computing and deep learning. This risk can be minimized with the support of researchers currently working on related topics at the Barcelona Neural Networking Center (BNN).

A.2 Description of tasks

A.2.1 Project management

T1 **Context and scope**: contextualise the thesis and describe its scope.

T2 **Time planning**: state general data of development of the thesis such as start and end date, task description and duration and overall temporal planning.

T3 **Budget and sustainability**: evaluate the project's economic aspect and its sustainable elements to ensure that the project is financially feasible and environmentally responsible.

T4 **Meetings**: weekly meeting with the thesis supervisor and monthly with the research group associated (BNN) to review progress, discuss challenges and receive feedback.

T5 **Documentation**: create a detailed and well-formatted document that includes all project-related information.

A.2.2 Review of the state of the art

T6 **Understanding existing graph partition algorithms**: conduct a comprehensive review of existing graph partition algorithms, including their strengths, limitations, and applications.

T7 **Study of different RL algorithms**: identify and evaluate various RL algorithms as well as environments, considering their suitability for the problem of this thesis and their potential.

T8 **Acquiring deeper knowledge in quantum computing**: research on the principles, applications, and limitations of quantum computing to gain a deeper understanding of the field. Additionally, understand how quantum algorithms can be translated into an environment that can be explored by an RL agent.

T9 **Study of deep learning best practices**: review the best practices when developing a large deep learning project, including regularization techniques, hyperparameter initialisation and optimization methods, helping to avoid common pitfalls and improve the likelihood of success.

A.2.3 Design and implementation

T10 **DRL model design**: design multiple DRL architectures and environments that address the problem statement.

T11 **Implementation of the proposed models**: implement the most promising envrionment, algorithms and policy architecture(s) using Stable Baselines 3 and the OpenAI Gym.

A.2.4 Experimentation and performance evaluation

T12 **Research of a set of representative datasets**: identify and collect a set of representative quantum systems of varying sizes and shapes.

T13 **Benchmark the proposed RL models**: evaluate the performance of the proposed RL models on collected datasets and benchmark them with respect the accuracy and scalability.

A.2.5 Scalability comparison

T14 **Replication of the results of FGP-rOEE's algorithm**: replicate the results of FGP-rOEE's algorithm with the help of these resources [4, 15]

T15 **Comparison between results of the proposed model vs. FGP-rOEE's algorithm**: compare the performance of the proposed RL model with FGP-rOEE's algorithm in terms of accuracy, speed and scalability to determine the effectiveness of the proposed RL model.

A.2.6 Conclusions and presentation

T16 **Conclusions**: after completing all the tasks aforementioned, see if we could meet the objective set and if so by how much. In the other case, see what have we missed and what could have been improved.

T17 **Preparation of the oral defence**: summarize the most relevant part of the thesis and write the presentation script.

A.3 Summary of the tasks

Now that all the tasks involved in this thesis have been stated and defined, Table A.1 presents the tasks along with the number of hours required for each task and their respective dependencies.

Id	Task	Time (h)	Dependencies
T1	Context and scope	8	-
T2	Time planning	5	-
T3	Budget and sustainability	4	-
T4	Meetings	8	-
T5	Documentation	10	-
T6	Understanding existing graph partition algorithms	35	-
T7	Study of different RL algorithms	30	-
T8	Acquiring deeper knowledge in quantum computing	25	-
T9	Study of deep learning best practices	25	-
T10	DRL model design	80	-
T11	Implementation of the proposed models	80	T10
T12	Research of a set of representative datasets	40	-
T13	Benchmark the proposed RL models	40	T12
T14	Replication of the results of FGP-rOEE's algorithm	60	-
T15	Comparison between results of the proposed architecture vs. FGP-rOEE's algorithm	40	T14
T16	Conclusions	25	All
T17	Preparation of the oral defence	25	All

Table A.1: Summary of the defined tasks, showing the expected duration in hours and dependencies

A.4 Resources

The resources needed to accomplish the goals of this thesis are the following:

Human resources

There are several human resources that can smooth much more all the process. The main one is my thesis supervisor, Sergi Abadal Cavallé, who will guide me and provide me with constant feedback on the project's development. It is also noteworthy to mention the UPC professor and Deep Learning expert Pere Barlet Ros, who will also be an advisor, and Pau Escofet, who is researching FGP-rOEE's algorithm and have a lot of knowledge on this subject which can be very valuable.

Material resources

As my research involves extensive investigation, the most useful and frequently used materials are likely to be numerous papers and articles. For the experimental and documentation part, I predict the main resources will be my computer, which will serve as the primary device for conducting experiments, a server to run the training of the deep learning models, Slack for communication with my supervisor and some other key contributors and finally Overleaf, which is the platform used to write the documentation.

A.5 Gantt chart

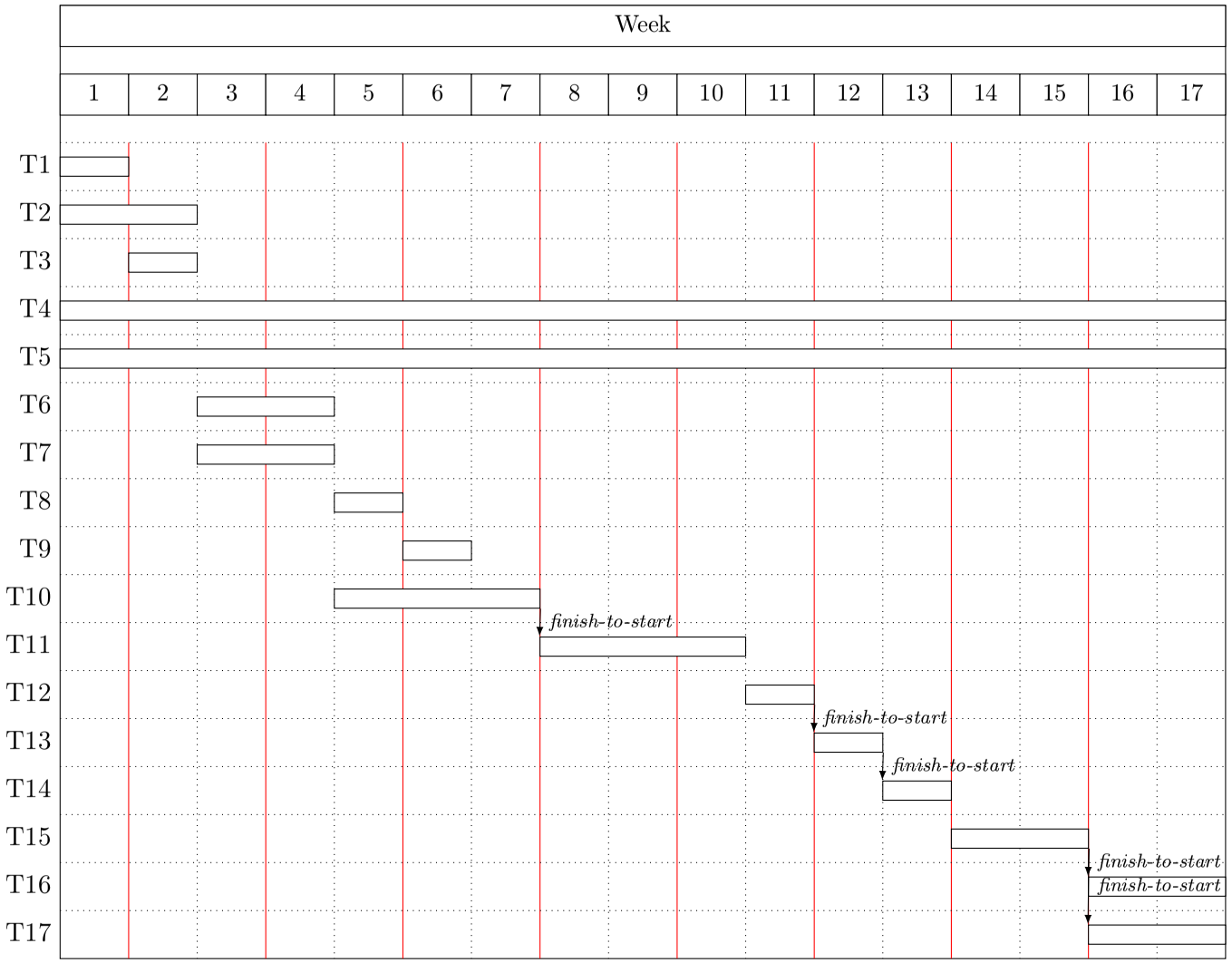

Figure A.1: Gantt chart representing the project's timeline

A.6 Risk management: alternative plans and obstacles

When developing this type of project, usually, in one way or another, there are always setbacks. That is why it is important to try to predict the risks and look for alternatives to be as prepared as possible.

A.6.1 Limited computing resources

Training deep learning models often requires a big computational power. A shortage of them may result in longer training times and may limit the number of different model architectures to experiment with.

- **Impact**: High
- **Proposed solution**: have good planning so even when the model is training, there is more work that can be done concurrently. In case, the training time become unfeasible, look for alternatives with more computational power like Google Colab (free) or even some cloud computing services like AWS (paid).

In case of encountering this obstacle, 10 more hours of dedication should be put to choose and set up a new environment to train the model.

A.6.2 RL model not learning

One common risk that may arise when building deep learning models is the possibility of the model failing to learn effectively, which can result in poor performance and inaccurate predictions.

- **Impact**: Medium
- **Proposed solution**: various techniques can be applied in case this problem is encountered, like using an alternative loss function, changing the hyperparameters, tweaking the environment and rewards or even trying alternative architectures.

This situation would require an additional 20 hours to identify the root cause of the problem and propose an appropriate solution.

A.6.3 RL model overfits

This problem can occur when the model is trained too closely to the training data. This can lead to poor generalization of unseen data. The risk of overfitting increases as the complexity of the model increases.

- **Impact**: Medium
- **Proposed solution**: use regularization techniques and early stopping. It is also important to simplify the architecture.

To overcome this problem it is estimated an additional 20 hours from what was initially estimated. This additional time could be used to find the best method to stop overfitting or to propose alternative architecture.

A.6.4 Skill level

The quantum computing field along with deep learning can be extremely complex. Given that quantum computing is a new domain for me and that I have a moderate knowledge of deep learning, this can become a problem.

- **Impact**: Low
- **Proposed solution**: put in extra hours to mitigate the lack of knowledge and also use the knowledge of the researchers of BNN involved in similar topics to leverage their expertise.

To address this problem, more research hours will be needed. It has been estimated an extra 30 hours will be required to thoroughly dive deep into the necessary topics.

Appendix B

Budget and sustainability

B.1 Budget

This section aims to provide a comprehensive overview of the budget and potential costs associated with conducting this thesis project.

B.1.1 Human resources

The development of this project can be done by two main roles, one project supervisor and a machine learning engineer.

The successful development of this project will require two main roles, a project supervisor and a machine learning engineer. In addition to these two roles, other roles like project advisors and domain experts may be used, but the time they will invest in the project is very little so their cost is negligible.

Role	**Salary(€)/year**
Project Manager	48,452
Machine Learning Engineer	35,512

Table B.1: Annual salaries of the different roles needed. Data from [3, 2]

Once the gross salaries are clear, they have to be multiplied by 1.35 to include the cost of social security payments. Since those salaries are computed on full-time jobs, they can be divided by the total number of working hours in a year to obtain the hourly salary.:

$$40\ hours/week \times 52\ weeks/year = 2,080\ hours/year$$

Once, all the values are clear, it can be computed the total cost per

Id	Concept	Time (h)	Cost (€/h)	Total cost
T1	Salary (project manager)	8	31.45	251.6
T2	Salary (project manager)	5	31.45	157.25
T3	Salary (project manager)	4	31.45	125.8
T4	Salary (project manager)	8	31.45	251.6
T5	Salary (project manager)	10	31.45	314.5
T6	Salary (machine learning engineer)	35	23.05	806.75
T7	Salary (machine learning engineer)	30	23.05	691.5
T8	Salary (machine learning engineer)	25	23.05	576.25
T9	Salary (machine learning engineer)	25	23.05	576.25
T10	Salary (machine learning engineer)	80	23.05	1844
T11	Salary (machine learning engineer)	80	23.05	1844
T12	Salary (machine learning engineer)	40	23.05	922
T13	Salary (machine learning engineer)	40	23.05	922
T14	Salary (machine learning engineer)	60	23.05	1383
T15	Salary (machine learning engineer)	40	23.05	922
T16	Salary (project manager)	25	31.45	786.25
T17	Salary (project manager)	25	31.45	786.25
	TOTAL			**13,161**

Table B.2: Derived human costs of each activity in the Gantt chart

Adding everything up, it can be seen in Table B.2 that the total human resources add up to 13,161€.

B.1.2 Generic costs

Amortizations

The main physical resource needed for the development of this project will be a laptop. In this case, the laptop is a brand new M2 MacBook Pro, which costs 2,799 € and will have an expected life of 7 years [6]. Thus, the amortization cost of this project, which will take place over four months, will be:

$$\frac{2799\ euro}{7\ year} \times \frac{1\ year}{12\ month} \times 4\ month = 133.28\ euro$$

Commuting cost

The author of the project does not live in the same Barcelona, where the UPC is, so to attend meetings, there will be a total of two commutes from Terrassa, the origin city to Barcelona and back. These cities are 35 kilometres apart, so the total cost would come down to:

$$17\ week \times \frac{2\ trip}{1\ week} \times 35\ km \times \frac{6.7\ l}{100\ km} \times \frac{1.8\ euro}{1\ l} = 143.51\ euro$$

Electricity cost

The M2 MacBook Pro consumes an average of 69.6 watts per hour [14]. Having an energy price of 0.12921€/kwh [19], and having estimated that the total duration of the project will be 540 hours, the total energy consumption will be:

$$69.9\ wh \times \frac{1\ kwh}{1000\ wh} \times \frac{0.12921}{1\ kwh} = 0.009031779\ euro$$

Obtaining such a small number makes this cost negligible.

Final budget

By putting together the human resources and all the general costs, it can be observed in Table B.3 the total cost of the project.

Description	**Cost (€)**
Human resources	13161
Amortizations	133.28
Commuting cost	143.51
TOTAL	**13437.79**

Table B.3: Total budget of human resources plus generic costs

B.1.3 Contingency

The aforementioned budget is an estimation and is prone to change as unforeseen events happen. For this reason, it is a great idea to have a contingency fund in order to minimize the effect of possible drawbacks.

For this thesis, a contingency fund rate of 10% will be established, as it is a commonly used rate in many IT projects. Based on the total estimated budget in Table B.3, this fund would be approximately 1,343.78€.

B.1.4 Incidentals

In the case, some of the possible obstacles occur, this would also have an effect on the budget.

- **Limited computing resources**: in the case, there is a lack of computational resources, a good option may be to use cloud computing services like AWS. The price derived from this would entirely depend on the amount of computational power needed and the number of hours needed. This event has a likelihood of 10% to happen and would require an additional 10 hours of work.
- **RL model not learning** & **RL model overfits**: in the case, these risks become a reality, it would mean that an extra of more or less 20 hours would be needed. These hours would be for the role of machine learning engineer. The probability of one of those events happening is 5%.

- **Skill level**: in case the skill level of the machine learning engineer becomes a problem, this would imply an extra 30 hours of study the most difficult fields. This event has a probability of happening of 8%.

Description	Estimated cost	Likelihood (%)	**Cost (€)**
Limited computing resources	230.5	10	23.05
RL model not learning	461	5	23.05
RL model overfits	461	5	23.05
Skill level	691.5	8	55.32
TOTAL			**124.47**

Table B.4: Incidental costs. The estimated cost column was computed by multiplying the estimated extra hours in Subsection B.1.4 by the salary of the respective role, as stated in Table B.1.

B.1.5 Management control

To manage the spending of the budget along the way of developing this thesis, the weekly meetings with the thesis supervisor are essential to ensure that everything is running smoothly and that the budget is being followed.

Furthermore, to calculate any deviations from the estimated cost, once a task is finished, its real cost will be computed and then subtracted from its estimated cost to determine the deviation. In the event of overspending, an investigation will be conducted to determine the cause and the amount of variance.

The total cost of a task can be computed as follows:

$$Total\ cost = direct\ costs + indirect\ costs + contingency + incidentals$$

Then the deviation can be computed as:

$$Deviation = total\ cost - estimated\ cost$$

These deviations from individual tasks will serve as an indicator of how much the real cost varies from its initial estimation.

B.2 Sustainability

The survey from the EDINSOST project was designed to analyze the level of sustainability education among students in the Spanish university system. The form uses a definition of sustainability which encompasses social, environmental, and economic fields as dimensions of the concept. The consideration of these dimensions is key to achieving sustainable solutions.

After filling out the form, I have gained a bigger insight into my knowledge and flaws in this topic. This form made me realize that even though I have a decent knowledge of the theoretical field, I am unaware of the tools and mechanisms necessary to ensure that sustainable principles are implemented. This highlights the need for continuous education I need on this topic, as well as, the focus on implementing sustainable practices every day.

I believe that even though the focus of this thesis is primarily on software, and that it's hard to reduce its sustainability impact in a significant way, it is crucial to be conscientious about it and to make an active effort to mitigate most of its negative effects.

B.2.1 Environmental impact

Have you quantified the environmental impact of undertaking the project? What measures have you taken to reduce the impact? Have you quantified this reduction?

The development of this software project doesn't have a huge environmental impact compared to physical projects. The primary environmental impact of this project is the power consumption during the training of the deep learning models. The measures taken to reduce the impact involved closely monitoring the training process to avoid training models for longer than necessary. The exact impact has not been quantified, but since it only affects the energy consumption of training relatively small DL models, the impact can be considered minimal.

If you carried out the project again, could you use fewer resources?

No, the computational resources used during the development of the thesis were the necessary ones. No computational resources were wasted.

What resources do you estimate will be used during the useful life of the project? What will be the environmental impact of these resources?

Only computational resources will be used throughout the useful life of the project, but the environmental impact will be beneficial. The success of this thesis lies in replacing current algorithms that are much more computationally expensive. The implementation of this thesis would improve the speed of those algorithms, thus significantly reducing the power consumption required to run them, and consequently reducing the environmental impact.

Will the project enable a reduction in the use of other resources? Overall, does the use of the project improve or worsen the ecological footprint?

This project presents a deep learning model that outperforms traditional algorithms used for quantum partitioning circuits, while being computationally less expensive. Therefore, by efficiently solving computationally challenging problems, the project has the potential to reduce the excessive computational resources required by current traditional approaches, such as the FGP-rOEE algorithm [4]. In general, the utilization of this project can enhance the environemental footprint by enabling more sustainable and efficient computing methods.

The success of this project will directly and indirectly impact various stakeholders across multiple industries (see Section 1.1.1). Consequently, if the thesis achieves its objectives, it could have a significant impact on power savings.

Could situations occur that could increase the project's ecological footprint?

There are no situations that could lead to an increase in the project's ecological footprint. In the worst-case scenario, if the approach proposed in this thesis is not used, the ecological footprint would remain the same as if it did not exist.

B.2.2 Economic impact

Have you quantified the cost (human and material resources) of undertaking the project? What decisions have you taken to reduce the cost? Have you quantified these savings?

In terms of the economic impact, in the previous Section B.1, all expenses related to the development of the project were taken into account, including salaries for the various roles involved and the amortization of devices.

Is the expected cost similar to the final cost? Have you justified any differences (lessons learned)?

The expected cost and the final cost have been very similar, as the initial budget was meticulously prepared, taking into account potential risks and making significant efforts to anticipate them.

What cost do you estimate the project will have during its useful life? Could this cost be reduced to increase viability?

The cost of the project during its lifespan will depend on the computational resources required by the hardware used for execution. Utilizing modern and specialized hardware can help reduce the overall cost. Additionally, the maintenance cost of this project is nearly negligible, as once the model is developed, it can be widely used without significant concerns about potential breaks or maintenance requirements.

Have you considered the cost of adaptations/updates/repairs during the useful life of the project?

Since this project is a software project, specifically resulting in a deep learning model, there is no need for repairs or updates. As a result, it can be widely used without concerns about the additional costs associated with adaptations, updates, or repairs.

Could situations occur that are detrimental to the project's viability?

The viability of the project could be affected if new approaches in quantum circuit partitioning were to emerge that outperformed the model presented in this thesis. Quantum computing is a rapidly evolving and extensively researched field, making it plausible to believe that such a situation could happen. However, it is important to note that the project still holds value since it presents a novel approach that can serve as a foundation for developing further optimized models. These models could potentially have an even more positive effect on the environmental footprint.

B.2.3 Social impact

Has undertaking this project led to meaningful reflections at the personal, professional or ethical level among the people involved?

Directly, the development of this thesis may not have led to a significant reflection. However, indirectly, by improving the accuracy and speed of quantum algorithm compilation, it can accelerate the production and adoption of quantum computers. This, in turn, can contribute to the creation of new applications that have the potential to generate substantial reflections and impacts at all levels, personal, professional and ethical.

Who will benefit from the use of the project? Could any group be adversely affected by the project? To what extent?

As stated in the stakeholders Section 1.1.1, quantum computing has the potential to bring huge benefits to multiple industries and positively impact people's lives. Like any technological advancement, it is possible for quantum computing to be misused and have adverse effects on various communities. However, a priori, it does not have any inherently negative effects.

To what extent does the project solve the problem that was established initially?

This thesis has fulfilled all the requirements established at the beginning of its development, which were necessary for it to be considered a success (more information in the conclusions Chapter 7).

Could situations occur in which the project adversely affects a specific population segment?

It is unlikely that the project would adversely affect specific communities. On the contrary, the development of this thesis could potentially lead to the creation of new applications that have the potential to revolutionize numerous industries, improving the lives of many people. Additionally, it could create new job opportunities in the field and even lead the development of new products and services, contributing to economic growth. A priori, it is expected that nobody will be negatively affected by the release of this project.

Could the project create any kind of dependency that puts users in a weak position?

It is highly improbable that this project would create a dependency that puts users in a weak position.

Appendix C

Status Report

This section provides an update on the project's status with respect the initial planning. It contains all changes made related to context and scope, time planning, methodology and rigour, analysis of alternatives, knowledge integration and identification of laws and regulations.

Initially, the project approach involved solving the problem using Graph Neural Networks (GNN). However, after careful consideration and expert advice in the field, it was determined that Reinforcement Learning was better suited for addressing this kind of problem. As a result of this change, the previous sections have been modified to align with the RL approach instead of the GNN one.

C.1 Context and Scope

Apart from modifying the parts where it was previously stated that GNN would be used to solve the problem, the most significant changes have been made in the Concepts 2.1 subsection, where concepts related to GNN have been removed and key concepts related to RL have been added. Another key change is the new subsection "Use of Reinforcement Learning 1.2.3", where the justification for using RL to accomplish the goal of this thesis is provided.

The risks defined in the section "Potential obstacles and risks A.1" have resulted as follows:

- Limited computing resources: although it is true that there were initial limitations in computing resources, it was quickly decided to test the validity of the approach with smaller circuits, which would drasticlly reduce the training time. Moreover, it was granted access to a server, allowing background execution without interrupting the work on the main computer.
- RL model not learning: this has been the major problem during the devlopment of the thesis. The resulting envionment is challenging, making it difficult for the agent to learn underlying patterns required to reach the goal and achieve a satistactory cummulative reward.
- RL model overfits: this is not currently a problem nor will it be since the dataset used to train the model is randomly generated, allowing to generate as many circuits as needed.
- Skill level: as anicipated, the level of knowledge about Reinforcement Learning has sometimes resulted in longer debugging times for certain problems that could have been quickly resolved. This can be atytributed to a lock of expertise and experience in the field.

C.2 Time Planning

Taking a look to the Gantt Diagram A.1, all the initial planning was followed until the task $T10 - DRL\ model\ design$. From task $T10$ onwards, until task
$T13 - Benchmark\ the\ proposed\ RL\ models$ it was a completly iterative process, deviating from the initially planned linear progression. This process involved designing an envionment, testing different algorithms and hyperparmeters, and extracting the best-performing approaches.

As it was mentioned in the previous section C.1, the intially indentified risk of the RL models not learning properly end up happening, which resulted in an additional two weeks compared to the initial plan. Consequently, the task
$T14 - Replication\ of\ the\ results\ of\ FGP - rOEE's\ algorithm$ has been reschueduled from week 13 to week 15, allowing it to be done concurrently with task
$T15 - Comparison\ between\ results\ of\ the\ proposed\ architecture\ vs.\ FGP - rOEE's\ algorithm$. Since FGP-rOEE's algorithm is already well-known and implemented in certain libraries, its replication can be completed quickly, freeing up most of week 15 to focus on task $T15$.

Since the total duration of the project remain inteact, the budged planned initially is still valid.

The Gantt Diagram figure C.1 shows the time planning after the rescheduling.

C.3 Revised Gantt Chart

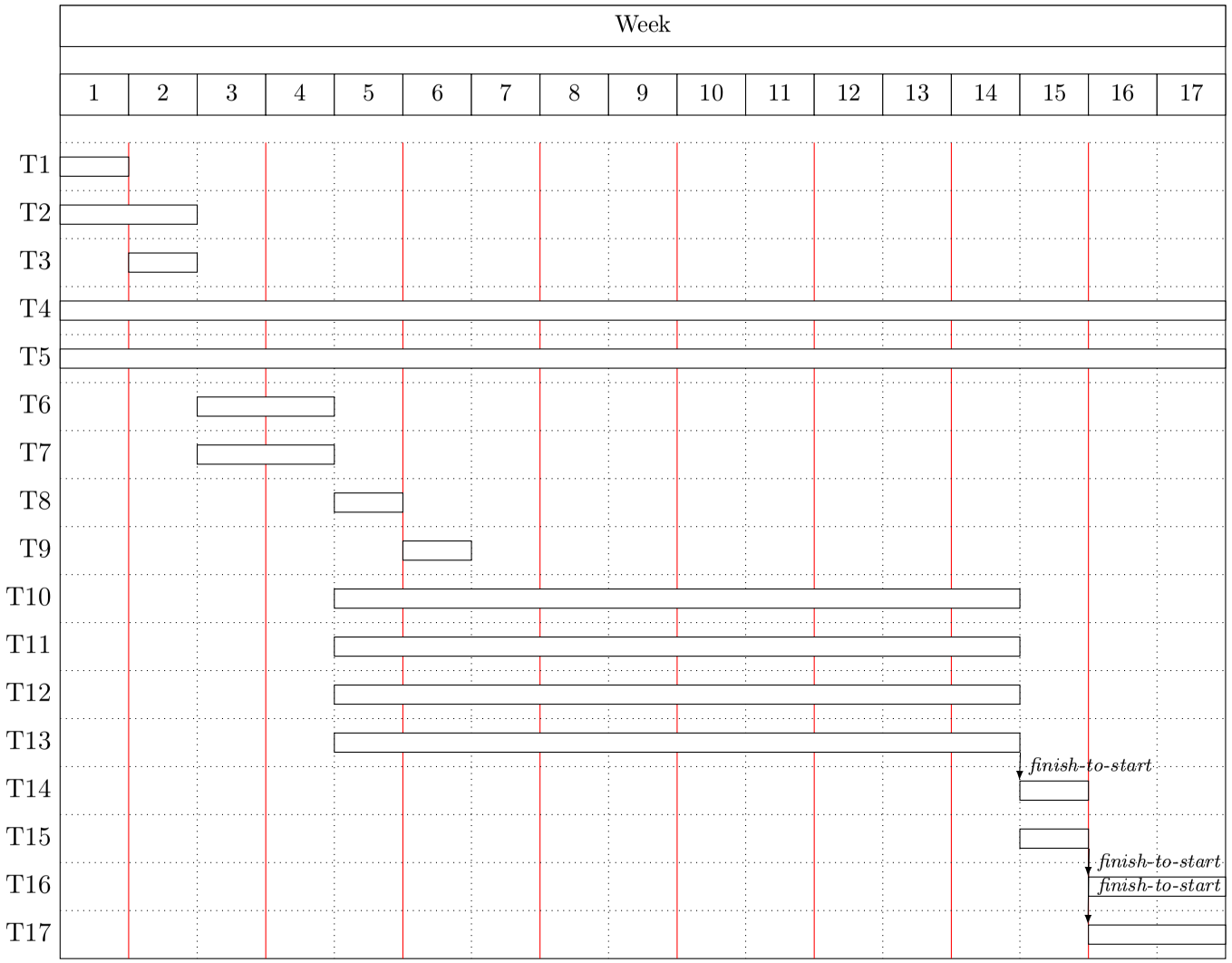

Figure C.1: Gantt chart modified after the status report with the actual plannig

C.4 Methodology and rigour

The working methodology has remained unchanged during the development of this project. Now that its finalization its comming closer, Agile continues to be a suitable approach. Breaking down the thesis into small tasks has proven to make the project more managable, especially considering its ambitious goal.

Regarding the monitoring tools, a weekly meeting was scheduled with the supervisor of this thesis Sergi Abadal Cavallé, as well as other experts in fields related to the project, who provided guidance and advice throughout the development process.

C.5 Analysis of alternatives

Apart from the initial analysis of alternatives conducted in the section "Analysis of alternatives 1.2.1", an extensive research was carried out during the realization of the project to explore papers focusing on solving similar problems, like using Reinforcement Learning to solve graph partitioing problems. While several interesting papers were discovered during this research, none of them provided a definitive solution or a "golden nugget" that could be directly applied or easily transformed to solve the problem addressed in this thesis.

C.6 Knowledge integration

Knowledge from various sources has been integrated for the resolution of this project. A significant part of knowledge comes from subjects taught at FIB, such as PRO1, PRO2, EDA, and A, which provided a solid foundation in algorithms and data structure knowledge. Additionally, knowledge acquired during the completion of subjects like Machine Learning at TUM University, as part of the Erasmus program, proved to be highly useful.

However, it is important to note that Reinforcement Learning was not covered during my college education. Therefore, a substantial portion of the knowledge was self-taught, acquired through numerous internet resources.

C.7 Identification of laws and regulations

There are no laws relevant to the scope of this project.

Appendix D

Methodology and rigour

D.1 Work methodology

In terms of working methodology for software development projects, there are several options, such as Waterfall, Scrum, Agile, and Rapid application development, among others and there is not a one-size-fits-all solution. Due to the characteristics of this thesis, Agile seems to be the most suitable methodology since it is known for its adaptability and iterative development, which is ideal for complex deep learning projects that may encounter situations that change some of the sub-objectives and goals.

The core of Agile methodology is its iterative approach, which involves breaking a large project into smaller, more manageable chunks called sprints. In this context, each task defined in Section A.2 will be considered a sprint.

In addition to Agile, Kanban will be used as a framework to manage the task workflow and improve the overall efficiency of the project development. Incorporating Kanban into our Agile approach can help visualize the tasks and track their progress through each stage of the workflow. Kanban provides a board that divides tasks into three different stages in the workflow: "To Do", "In progress", and "Done". This framework can be very useful in identifying and addressing possible delays and bottlenecks, as well as controlling the number of concurrent tasks opened at the same time.

D.2 Monitoring tools

To monitor the correct development of the project, we have set a weekly meeting with the supervisor of this bachelor's thesis Sergi Abadal Cavallé, in them, it will be discussed the state of the project and possible obstacles and how to overcome them.

Moreover, once a month is scheduled a meeting with the group responsible for working with quantum computing inside the Barcelona Neural Networking Center (BNN). There will be an opportunity to discuss topics related to this thesis with other researchers.