Thesis Dissertation

Implementation of a Minimum Vertex Cover Quantum Decomposition Algorithm

Konstantinos Larkou

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

May 2023

UNIVERSITY OF CYPRYS

COMPUTER SCIENCE DEPARTMENT

Implementation of a Minimum Vertex Cover Quantum Decomposition Algorithm

Konstantinos Larkou

Supervisor

Anna Philippou

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science at University of Cyprus

May 2023

Acknowledgements

I would like to express my heartfelt acknowledgements to the following individuals who have played significant roles in my academic journey:

First and foremost, I am immensely grateful to Dr. Anna Philippou, my esteemed professor and supervisor. Your guidance, expertise, and unwavering support have been instrumental in shaping my research and academic growth. Your mentorship has been invaluable, and I am truly fortunate to have had the opportunity to work under your supervision.

I would also like to extend my sincere appreciation to Eleftheria Kouppari, my PhD supervisor. Your continuous encouragement, insightful feedback, and dedication to my progress have been truly motivating. Your expertise and guidance have been pivotal in shaping the direction of my research, and I am grateful for the knowledge and experience I gained under your mentorship.

I would like to express my deepest gratitude to my beloved parents for their unwavering support and encouragement throughout my academic journey. Your unconditional love, understanding, and belief in my abilities have been a constant source of strength. I am forever indebted to you for the sacrifices you have made to provide me with the opportunities and resources needed to pursue my dreams.

To my dear friends, thank you for tolerating my late nights, endless discussions, and occasional academic rants. Your friendship, camaraderie, and moral support have made this journey more enjoyable and memorable. Your presence has provided a much-needed balance in my life, and I am grateful for the shared experiences and laughter we have had together.

A special mention goes to my study buddies, Andrianni Kakoulli and Demetris Vereis, with whom I have spent countless nights in the library. Your dedication, focus, and mutual encouragement have created an environment conducive to productivity and growth. The shared determination and countless hours spent together have made these study sessions both memorable and productive.

To all the individuals who have directly or indirectly contributed to my academic and personal development, thank you. Your belief in my potential, encouragement, and support have been indispensable throughout this journey.

Abstract

The Minimum Vertex Cover, one of Karp's 21 NP-complete problems, is useful in fields like network security, operations research and biochemistry, but it is challenging to find the optimal solution using classical computers. Quantum annealers can solve such problems, but it is difficult to map them onto the hardware due to their limitations (number of qubits, noise). This thesis implements a suggested algorithm for solving the Minimum Vertex Cover (MVC) problem using a decomposition approach. The proposed algorithm allows for the solving of larger MVC instances on D-Wave quantum computers, which was previously not possible due to the limited number of available qubits. The decomposition algorithm recursively breaks down the original MVC problem into smaller subproblems until the generated subproblems are small enough to be solved directly using a quantum annealer or a classical solver. It is important to note that the algorithm is designed to provide exact solutions, meaning that the optimality of the solution is guaranteed as long as all subproblems are solved exactly. Also, various techniques are suggested to speed up the process. The performances of a classical algorithm, the decomposition algorithm with a classical solver, and the decomposition algorithm with two quantum solvers using different topologies (Chimera and Pegasus) are evaluated and compared.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Quantum computing has emerged as a promising field due to its potential to solve problems that are infeasible for classical computers. The power of quantum parallelism and non-deterministic nature of quantum mechanics offers new opportunities for computation. In particular, quantum computing offers the potential to solve complex optimization problems such as the Minimum Vertex Cover (MVC) problem.

The MVC problem is an important problem in graph theory and has many practical applications, such as in scheduling, transportation and communication networks, and database management. Despite its simplicity, this problem is known to be NP-hard, which means that it is computationally intractable on a classical computer for large graphs. Specifically, the problem is believed to be non-polynomial time (NP)-complete, which means that there is no known algorithm that can solve the problem in polynomial time, and it is unlikely that such an algorithm exists.

The reason why the minimum vertex cover problem is hard to solve on a classical computer is that the number of possible vertex covers grows exponentially with the size of the graph. To see why, consider a graph with n nodes. For each node, there are two possible choices: either include it in the vertex cover or exclude it. Therefore, the total number of possible vertex covers is 2^n, which is an exponential function of n. This means that for large graphs, the number of possible vertex covers quickly becomes too large to enumerate all of them in a reasonable amount of time.

At the same time, solving large MVC problems on a quantum annealer using a direct approach can be challenging due to the limited number of qubits available in current hardware.

Fortunately, decomposition algorithms have emerged which allow us to solve larger MVC problems by recursively splitting a given instance of MVC into smaller subproblems until they can be solved directly on a quantum annealer. This approach not only overcomes the limitations of the available hardware but it also claims to offer potential speedup over classical methods.

Additionally, the emergence of decomposition algorithms opens up the possibility of solving even larger MVC problems, which can have a profound impact on industries such as logistics, manufacturing, and finance.

Therefore, the motivation for this thesis is to explore the potential benefits of using decomposition algorithms on a quantum annealer to solve large MVC problems.

1.2 Work Purpose

The purpose of this work is to implement a quantum algorithm for solving the Minimum Vertex Cover (MVC) problem using decomposition techniques. The proposed algorithm, developed by Elijah Pelofske, Georg Hahn, and Hristo Djidjev [37] decomposes the original graph into smaller subgraphs until they can be solved efficiently by a quantum annealer. The algorithm then utilizes a classical solver to find the solution for the MVC problem. The classical solver is used when the subgraph size is below a certain threshold.

The main contribution of this work is the implementation of a quantum solver for this algorithm on a quantum annealing and to study its performance in different setups. Furthermore, we compare its performance with a classical solution.

1.3 Work Methodology

In this study, we aimed to implement the decomposition algorithm proposed in the paper "Decomposition Algorithms for Scalable Quantum Annealing" [37] for solving the Minimum Vertex Cover (MVC) problem using quantum annealing. Specifically, I used a classical solver described in the paper using the NetworkX package in Python. For the quantum solver, I converted the input graph to a Constrained Quadratic Model and used D-Wave samplers to solve the resulting optimization problem.

In order to evaluate the performance of the proposed algorithm, I conducted experiments on a range of graphs with 20 to 120 nodes and densities ranging from 0.1 to 0.9. Due to the computational requirements of the experiments, I requested a

computer from university with more processing power than my personal laptop as it could not execute the algorithms for larger graphs.

To measure the performance of the algorithm, I compared the execution times of the decomposition algorithm with quantum and classical solvers to a fully classical algorithm for each graph instance. To compare the quality of the solutions produced by each algorithm I measure the execution time of each algorithm while ensuring that the size of the minimum vertex cover was minimum.

Overall, the methodology used in this study involves implementing and testing the decomposition algorithm with both classical and quantum solvers, conducting experiments on a range of graph instances, and comparing the results to evaluate the effectiveness of the proposed approach.

1.4 Thesis Structure

Chapter 1: Introduction

The Introduction Chapter provides an overview of the research, including the motivation behind the study, the purpose of the work, and the methodology employed. It sets the context for the research and outlines the objectives that guide the entire thesis.

Chapter 2: Literature Review

The Literature Review chapter conducts an in-depth analysis of existing research, theories, and approaches relevant to the Minimum Vertex Cover (MVC) problem and quantum computing. It reviews scholarly articles, books, and other relevant sources to establish the theoretical foundation for the study. This chapter aims to identify gaps in the literature and lay the groundwork for the proposed research.

Chapter 3: Implementation

The Implementation Chapter delves into the technical details of the proposed algorithm for solving the MVC problem using a decomposition approach. It describes the design and development of the algorithm, highlighting the key components and methodologies employed. This chapter presents the conceptual framework and provides insights into the practical implementation of the algorithm.

Chapter 4: Experimental Evaluation

The Experimental Evaluation chapter presents the results of the empirical study conducted to assess the performance and effectiveness of the proposed algorithm. It describes the research methodology, data collection procedures, and experimental setup. The chapter presents and analyzes the findings, discussing the performance

metrics, comparisons with existing algorithms, and any significant observations or patterns observed during the evaluation.

Chapter 5: Conclusion

The Conclusion Chapter summarizes the main findings and contributions of the research. It addresses the research objectives and hypotheses posed at the beginning of the thesis, providing a critical evaluation of the results. This chapter discusses the implications of the findings, highlights the limitations of the study, and suggests avenues for future research. It serves as a final reflection on the research process and offers closing remarks.

Chapter 2

Literature Review

2.1 Quantum Theory & Computing

> "The universe is a quantum computer. Since you can simulate any set of particle interactions with a quantum computer made of the same number of particles, then there's no practical difference between the universe and a quantum computer simulating the universe."
>
> **David Walton, Supersymmetry**

2.1.1 Quantum Theory

Quantum theory explains the behavior of the smallest particles that make up the universe, such as atoms, photons, and electrons. Unlike classical physics, which is based on deterministic laws, quantum mechanics introduces the concept of probability and uncertainty in the behavior of particles. The principles of quantum mechanics have been tested extensively and have been shown to accurately describe the behavior of these particles [1]. The field of quantum theory has led to many technological advancements, including the development of transistors, lasers, and MRIs, which have transformed various industries [2]. Quantum theory has been extensively tested and validated through a variety of experiments over many decades. One of the most famous experiments is the double-slit experiment [3] [36].

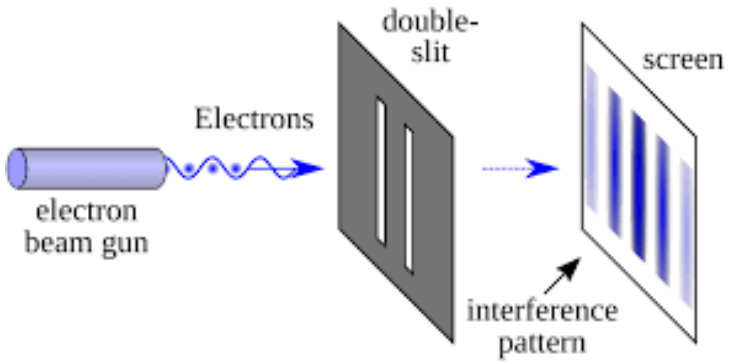

Figure 2.1 The double-slit experiment [36]

Figure 2.1 demonstrates the double-slit experiment which investigates the wave-particle duality of light and matter. When particles or light pass through two slits, an interference pattern is observed on a screen, suggesting that particles behave like waves. This experiment reveals the dual nature of particles, displaying both wave-like and particle-like properties.

Another experiment is the Stern-Gerlach experiment [4], which demonstrated the quantization of angular momentum and the existence of intrinsic spin. These experiments, and many others, have consistently shown that the predictions of quantum mechanics are accurate and reliable, and they have led to the development of many technologies that rely on quantum mechanics principles. The study of quantum mechanics has also sparked interest in the development of quantum computing, which has the potential to revolutionize the way we solve complex computational problems.

2.1.2 Quantum Computing

Quantum computing is a new and highly researched area of computer science and physics. It revolutionizes computation as it is conventionally defined, with the aid of Quantum Theory. The basic difference of a quantum computer from a classical computer, is that it operates on qubits, which can encode the values of traditional bits (0 and 1), but all the values in the complex space between them as well. Quantum computers can also harness several phenomena that are derived from quantum mechanics, such as entanglement. This advantage of quantum computers enables us to execute a new class of algorithms that are more efficient and powerful. August Stern published a paper in 1970 titled "Thinking as a Quantum-Mechanical Process," [5] which presented the concept of Quantum Computing (QC). However, it was not until

1980 that theoretical physicist Richard Feynman popularized the idea of using quantum effects to improve computational efficiency [6].

2.1.2.1 Quantum phenomena

Superposition is a fundamental concept in quantum mechanics that describes the ability of quantum particles to exist in multiple states simultaneously. In classical physics, an object can only exist in one state at a time, but in quantum theory, a particle can exist in a superposition of many possible states. For example, an electron can be in a superposition of spin-up and spin-down states at the same time until it is measured, at which point it collapses into a definite state [7].

Entanglement is a quantum phenomenon that occurs when two or more particles are connected in such a way that the state of one particle is dependent on the state of the other particle, even if they are separated by a large distance. This means that measuring the state of one particle instantly affects the state of the other particle, regardless of the distance between them. Entanglement is a key feature of quantum mechanics and has been shown to have potential applications in fields such as quantum computing and cryptography [8].

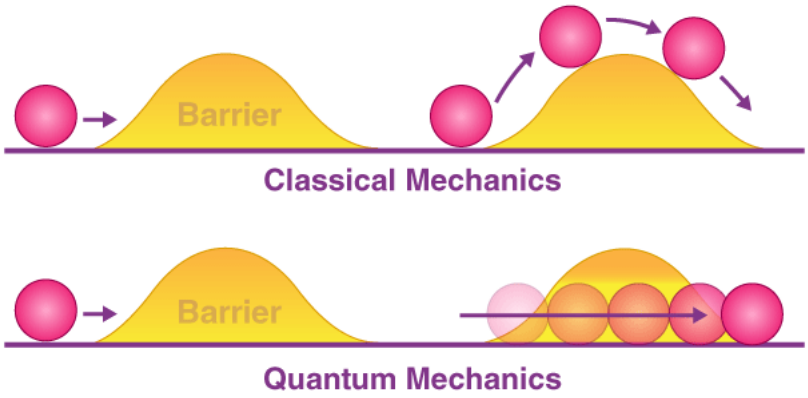

Figure 2.2 Tunneling [6]

Figure 2.2 demonstrates tunneling, a quantum mechanical phenomenon where a particle can pass through a potential barrier that it does not have enough energy to overcome classically. This is possible due to the wave-like nature of particles, which allows them to exist in regions that would be forbidden by classical mechanics. Tunneling is important in many areas of quantum mechanics, including nuclear physics, solid-state

physics, and quantum computing. It is also a crucial factor in many chemical reactions and biological processes [9].

2.1.3 Quantum Annealing

Quantum annealing is a computational technique used to solve optimization problems in a faster and more efficient way than classical computers. It is based on the principles of quantum theory, which allow for the creation and manipulation of quantum states that can be used to represent and process information [10].

The idea behind quantum annealing is similar to simulated annealing, a classical optimization algorithm inspired by nature's thermal annealing process. In simulated annealing, a material is heated and then slowly cooled to reach a state of lowest energy [11]. The energy landscape of a problem is represented by its cost function, and the algorithm tries to find the lowest energy point on the landscape, which is the global minimum.

Quantum annealing uses a similar approach, but instead of probabilistically exploring the search space, it uses quantum mechanics to travel along the energy cost function. This is made possible by quantum properties like superposition and entanglement, which enable new ways of exploring the energy landscape that are not accessible through classical annealing [12].

Superposition allows for the probabilistic representation of all states of the problem with only N qubits, while entanglement enables the alteration of superposition of states rather than qubit by qubit. Under certain conditions, entanglement and superposition create a phenomenon known as tunneling, which allows quantum annealing to jump through hills and escape local minima unlike the classical approach.

In quantum annealing, the qubits are initialized in a state of superposition, which means that each qubit represents all possible states simultaneously. During the annealing process, the qubits evolve and interact with each other through a Hamiltonian that is designed to minimize the energy of the system. Figure 2.3 demonstrates that as the annealing progresses, the qubits gradually lose their superposition and entanglement properties and become classical bits, where each bit represents a single state. At the end of the annealing process, the qubits are measured and the resulting classical binary string corresponds to the solution of the optimization problem being solved [13].

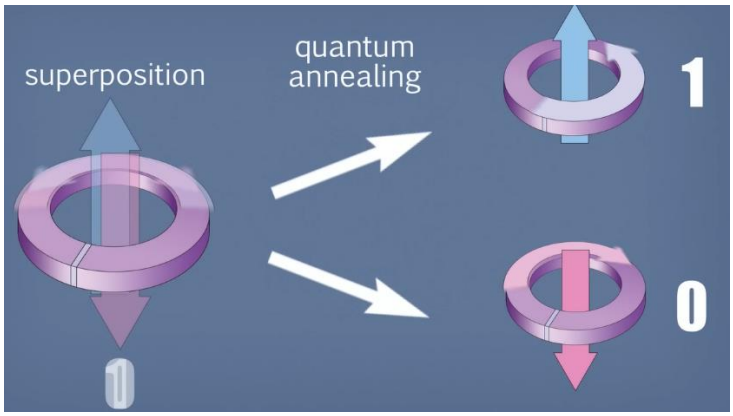

Figure 2.3 Process of quantum annealing [48]

The cost function that quantum annealing travels along is derived from the definition of the problem we are trying to solve. This optimization problem has a number of variables and constraints. Figures 2.4 exhibits a bias which is a constraint that only affects one variable, while constraints between two variables will be called couplers.

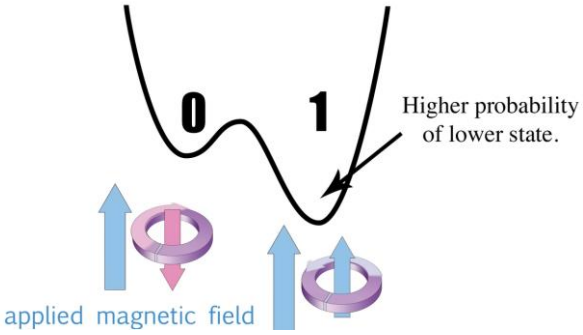

Figure 2.4 A bias [48]

Figure 2.5 shows the phenomenon of entanglement in which two or more particles are correlated in such a way that the state of one particle cannot be described independently of the state of the other. In the context of quantum annealing, entanglement between qubits means that the state of one qubit depends on the state of the other and this can be achieved using a coupler. Specifically, if two or more qubits are entangled, measuring the state of one qubit will determine the state of the other qubit, even if they are separated by large distances.

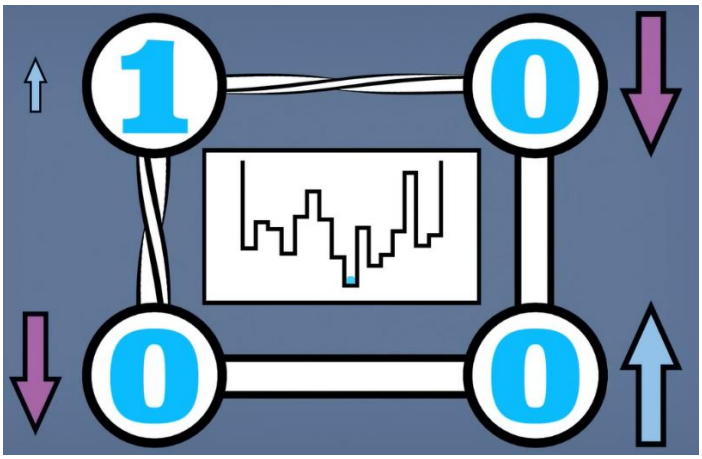

Figure 2.5 Entanglement [48]

In order to understand quantum annealing, it is helpful to examine the equation of quantum annealing and how it changes during the process. It is also important to understand how the architecture of the system evolves when embedded on the quantum annealer, and to observe how the energy of the problem progresses at each step of the annealing process.

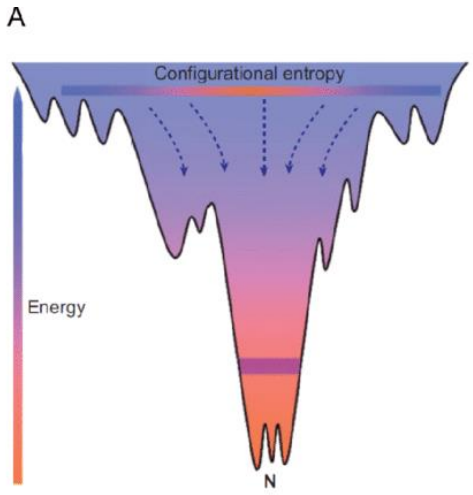

Figure 2.6A Two-dimensional representation of the energy landscape [47]

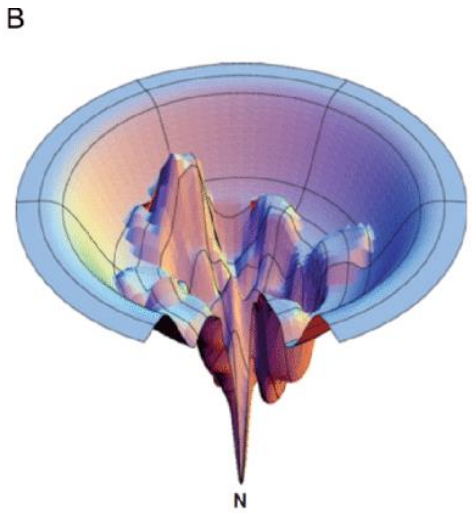

Figure 2.6B Three-dimensional representation of the energy landscape [47]

The landscape shown in Figure 2.6A and Figure 2.6B are characterized by hills and valleys, with the minimum energy point denoted by N and referred to as the global minimum. The annealing process involves traversing this energy landscape to find the global minimum, which is where the problem is optimally solved.

Many optimization problems can be formulated as finding the minimum of a cost function, where the cost function represents the objective to be optimized. In such cases, finding the minimum energy point of the corresponding energy landscape is equivalent to finding the optimal solution to the optimization problem.

Quantum annealing has many potential applications, including optimization problems in a variety of fields such as finance, logistics, and chemistry. However, it is still a developing field and there are many challenges that need to be overcome before it can be widely adopted. These challenges include improving the reliability and scalability of quantum annealers, developing new algorithms and techniques to optimize the annealing process, and integrating quantum annealing with classical computing systems.

2.1.4 The Hamiltonian and Eigenspectrum

In quantum annealing, the Hamiltonian and its eigenspectrum play a crucial role in determining the behavior of the system during the annealing process.

The Hamiltonian, shown in Figure 2.7, is the operator that represents the total energy of the system in quantum mechanics. In quantum annealing, the Hamiltonian is typically expressed as a sum of two terms: the problem Hamiltonian and the driver Hamiltonian.

The problem Hamiltonian encodes the optimization problem that is being solved, while the driver Hamiltonian is responsible for driving the system towards the lowest energy state [14].

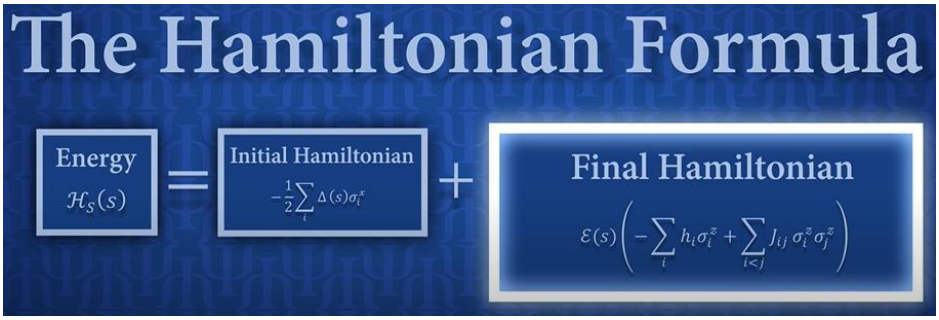

Figure 2.7 Annealing process equation [15]

During the annealing process, the Hamiltonian is gradually changed from the driver Hamiltonian to the problem Hamiltonian, allowing the system to explore the energy landscape and eventually settle into the ground state, which corresponds to the solution of the optimization problem.

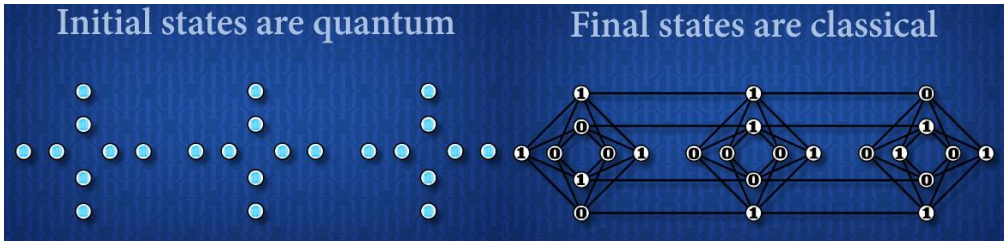

Figure 2.8 States of Hamiltonian [15]

The Figure 2.8 above illustrates the process of quantum annealing. The initial state of the qubits is a quantum state, represented by the initial Hamiltonian and the final state of qubits is a classical state, represented by the final Hamiltonian. The final Hamiltonian incorporates the biases and couplers of the problem being solved, which are essential for finding the optimal solution.

The eigenspectrum of the Hamiltonian is the set of all possible energy levels of the system. In quantum annealing, the eigenspectrum determines the probability of the system being in a particular energy state at any given point in time. The ground state of the problem Hamiltonian corresponds to the solution of the optimization problem, while the excited states correspond to suboptimal solutions.

The eigenspectrum of the Hamiltonian is also important for determining the success probability of the annealing process [16] [17] [18]. In general, the larger the energy gap between the ground state and the first excited state, the higher the success probability

[19] [20]. This is because the larger gap makes it less likely for the system to become trapped in an excited state during the annealing process, and more likely to settle into the ground state [21].

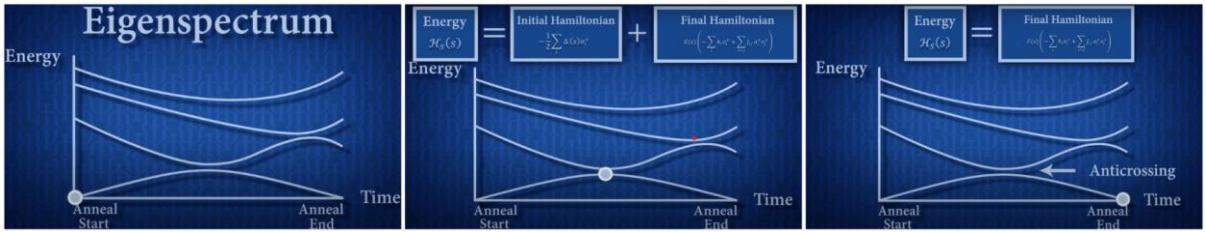

Figure 2.9 Eigenspectrum [15]

The Figure 2.9 presented above depicts the energy states of the system at three different stages of the annealing process. The first state represents the initial energy of the system, while the second state shows the energy during the annealing process. At some point during the process, the system reaches a stage where its energy is almost the same as that of an excited state, which is referred to as an anti-crossing. The third and final state represents the energy of the system after the annealing process, which should be the optimal state if the process was conducted correctly.

2.2 DWave Quantum Annealer

The D-Wave quantum computer system [22] utilizes a Quantum Processing Unit (QPU) that operates under the principles of quantum mechanics, requiring it to be maintained at a temperature close to absolute zero and shielded from electromagnetic interference. To meet these requirements, the system uses a closed-loop cryogenic dilution refrigerator system to achieve cryogenic temperatures below 15 mK [23], an RF-shielded enclosure, and a magnetic shielding subsystem. The QPU itself is made up of tiny metal loops functioning as qubits or couplers, which become superconductors and exhibit quantum-mechanical effects at temperatures below 9.2 kelvin.

The couplers that connect the qubits together are made of niobium and have less control circuitry. DWave2000Q [24], the current Chimera topology implementation [25], is robust against many types of noise, and the superconducting flux qubit used allows for run-time programming of the qubits. D-Wave Advantage is the latest Pegasus topology quantum computer. It is a 5th-generation quantum annealer and has 5640 superconducting qubits interconnected by 15 connections each. It features improved qubit yield and coherence times, making it more reliable and stable for practical use cases.

The Control unit is responsible for programming the QPU and inputting the instance into the QPU and reading the results produced to present the solution to the user. Digital-to-analog controllers (DACs) are used to carry the biases signal from the control to the processor to load the problem onto the QPU chip. The initialization process requires the DAC to turn up H1 and completely turn down Hp to set the qubits into a superposition. The annealing process can then begin.

The development of different topologies is necessary to embed larger and more complicated problems in the QPU. The QUBO formulation of a problem has a linear and a quadratic part, with the linear part being represented by the qubits in the host graph, while the quadratic part is matched to the couplers. In practical applications, it is not possible to directly match the qubits, so chains are used, which limit the number of remaining available qubits as well as the connectivity structure. Allowing higher degree will help to solve larger problems, while having a more complicated coupling structure will enable more difficult problems to be embedded, such as those with more constraints.

2.2.1 DWave Topologies

2.2.1.1 Chimera Topology

In the D-Wave 2000Q systems, the qubits on the Quantum Processing Unit (QPU) are arranged in a vertical or horizontal orientation, as depicted in Figure 2.11 [26] [49].

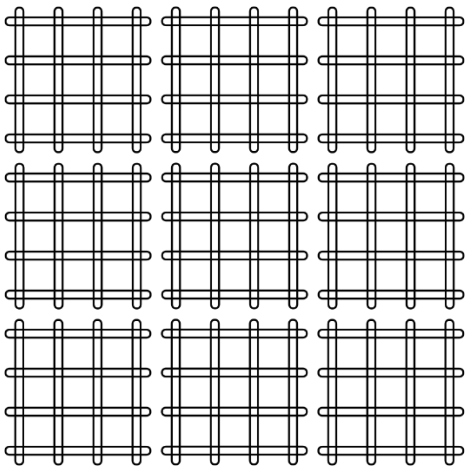

Figure 2.11 Representation of qubits as loops [49]

Figure 2.11 illustrates qubits as loops where there are three rows of 12 vertical qubits and three columns of 12 horizontal qubits, totaling 72 qubits with 36 vertical and 36 horizontal orientations.

For QPUs utilizing the Chimera topology, it is conceptually useful to classify couplers into two categories.

Internal couplers: These couplers connect pairs of orthogonal qubits with opposite orientations, as shown in Figure 2.12. The Chimera topology consists of a recurring structure known as a unit cell, which comprises four horizontal qubits coupled to four vertical qubits in a bipartite graph called $K_{4,4}$ [49].

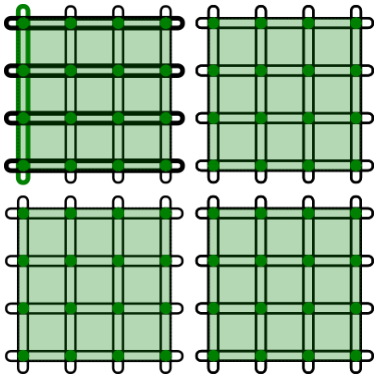

Figure 2.12 Visual representation of internal couplers [49]

Figure 12 illustrates internal couplers denoted by green circles at the intersections of qubits. For example, the upper leftmost vertical qubit, highlighted in green, is internally coupled to four horizontal qubits, shown in bold. The translucent green squares illustrate the recurring structure of the topology, representing the division of couplings into unit cells of the $K_{4,4}$ bipartite graphs.

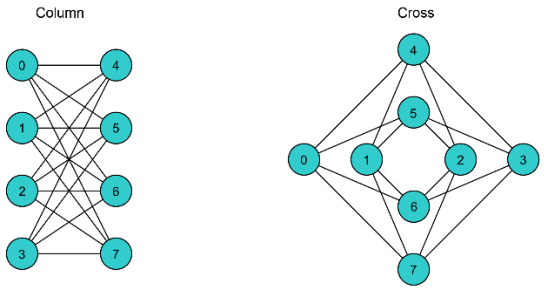

Figure 2.13 The unit cell of a Chimera structure [49]

Figure 2.13 presents the cell structure of a Chimera unit cell which consists of two groups, each containing four qubits. Every qubit in one group is connected to all the

qubits in the other group, while having no connections to qubits within its own group. This configuration forms a $K_{4,4}$ graph. For instance, the green qubit labeled as 0 is connected to the bolded qubits 4 to 7.

External Couplers: These couplers connect pairs of colinear qubits, meaning parallel qubits in the same row or column, as depicted in Figure 2.14. Vertical qubits are coupled to adjacent vertical qubits, and horizontal qubits are coupled to adjacent horizontal qubits. For example, the green horizontal qubit in the center couples to the two blue horizontal qubits in adjacent unit cells. In addition to the external couplers, the green qubit is also internally coupled to the bolded qubits within its own unit cell.

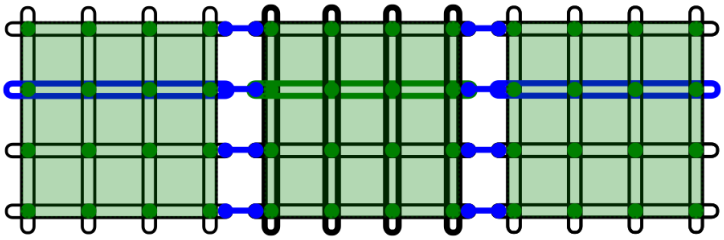

Figure 2.14 Illustration of external couplers as connected blue circles [49]

The internal couplers form the $K_{4,4}$ bipartite graphs in unit cells, and these unit cells are interconnected by external couplers to create the Chimera topology.

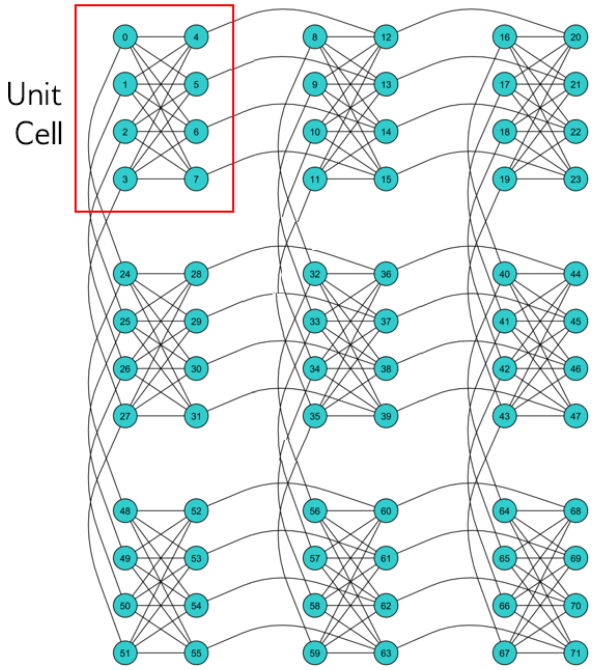

Figure 2.15 A cropped view of unit cells of a Chimera graph [49]

Figure 2.15 illustrates unit cells of a Chimera graph where qubits are arranged in 9-unit cells (green circles) interconnected by external couplers (black lines).

In terms of characteristics, Chimera qubits are considered to have a nominal length of 4, as each qubit is connected to four orthogonal qubits through internal couplers. Furthermore, the qubits have a degree of 6, as each qubit is coupled to six different qubits.

The notation C_n refers to a Chimera graph consisting of an *N x N* grid of unit cells. In the case of the D-Wave 2000Q QPU, it supports a C_{16} Chimera graph, where its 2048 qubits are logically mapped into a *16 x 16* matrix of unit cells, each containing 8 qubits [49].

2.2.1.2 Pegasus Topology

In Advantage quantum processing units (QPUs), qubits are arranged in a vertical or horizontal orientation similar to the Chimera topology [26] [49]. However, in the Pegasus topology, the aligned qubits are also shifted. This arrangement is illustrated in Figure 2.16, which provides a partial view of the Pegasus topology, shows approximately three rows of 12 vertical qubits and three columns of 12 horizontal qubits, resulting in a total of 72 qubits (36 vertical and 36 horizontal) [49].

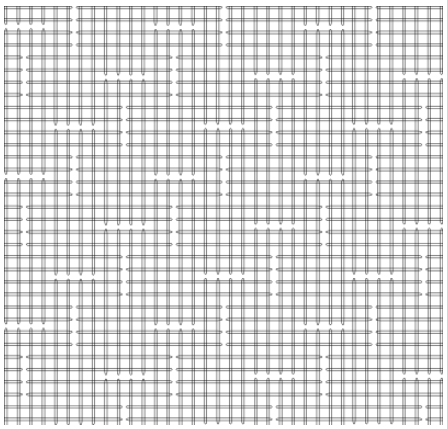

Figure 2.16 Pegasus qubits [49]

When working with QPUs utilizing the Pegasus topology, it is conceptually useful to classify couplers as internal, external, or odd. Figure 2.17 and Figure 2.18 present two different perspectives of how qubits are coupled in this topology.

Figure 2.17 shows the coupling of qubits represented as horizontal and vertical loops. The central horizontal qubit, depicted in red and labeled as 1, is internally connected to pairs of vertical qubits (3 through 8), with each pair and its odd coupler displayed in a distinct color. Additionally, it is externally connected to horizontal qubits 2 and 9, each depicted in a different color.

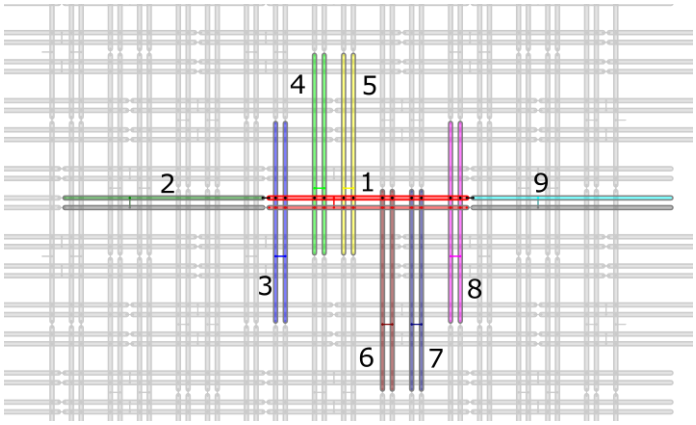

Figure 2.17 Pegasus Couplers [49]

Figure 2.18 showcases a graphical representation of the coupled qubits, with the qubits represented as dots and the couplers as lines. The upper center qubit, highlighted in red and labeled as 1, is oddly coupled to the red qubit directly beneath it. It is also internally connected to vertical qubits in pairs (3 through 8), with each pair and its odd coupler shown in a different color. Furthermore, it is externally connected to horizontal qubits 2 and 9, each displayed in a different color.

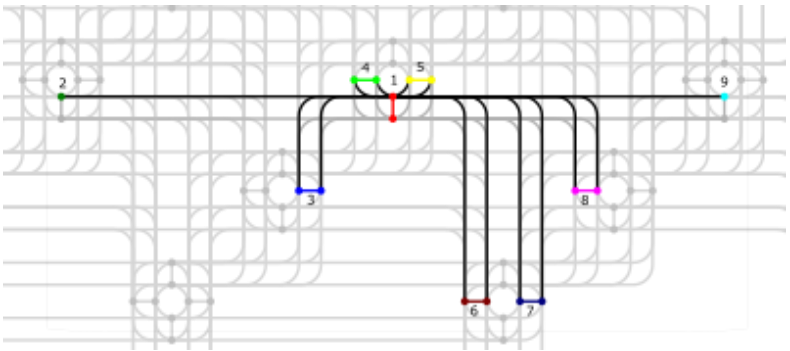

Figure 2.18 Coupled Qubits [49]

Internal couplers: Internal couplers establish connections between pairs of qubits with orthogonal orientations, as depicted in Figure 2.19. Each qubit is internally coupled to 12 other qubits [49].

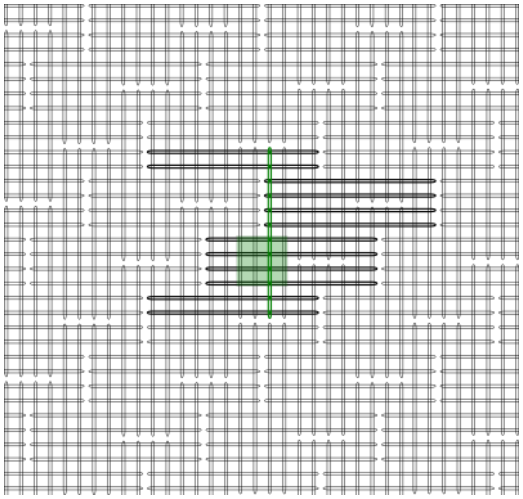

Figure 2.19 Pegasus Internal Couplers [49]

Figure 2.19 represents the internal couplers through junctions of horizontal and vertical loops. For instance, the green vertical qubit is coupled to 12 horizontal qubits, which are highlighted. The translucent green square illustrates a Chimera unit cell structure, which corresponds to a bipartite graph of internal couplings (K4,4).

External **couplers**: External couplers establish connections between adjacent qubits of the same alignment, as shown in Figure 2.20.

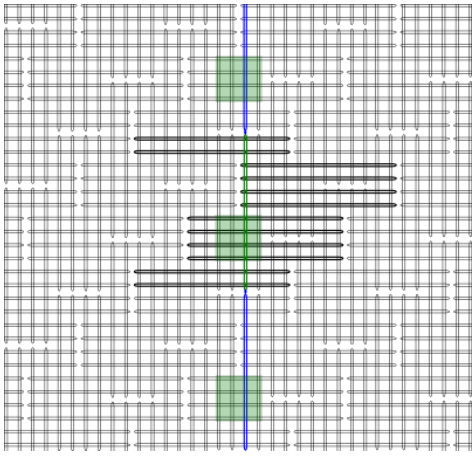

Figure 2.20 Pegasus External Couplers [49]

Figure 2.20 demonstrates how external couplers connect similarly aligned adjacent qubits. For example, the green vertical qubit is coupled to two adjacent vertical qubits, highlighted in blue.

Odd couplers: Odd couplers establish connections between pairs of qubits with the same alignment, as depicted in Figure 2.21.

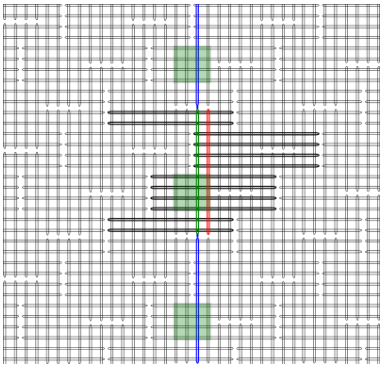

Figure 2.21 Odd couplers [49]

Figure 2.21 displays how odd couplers connect similarly aligned pairs of qubits. For instance, the green vertical qubit is connected to the red vertical qubit through an odd coupler.

The Pegasus topology features qubits with a degree of 15 and incorporates native K_4 and $K_{6,6}$ subgraphs. Pegasus qubits are considered to have a nominal length of 12, indicating that each qubit is connected to 12 orthogonal qubits through internal couplers. Additionally, each qubit has a degree of 15, implying that it is coupled to 15 different qubits.

PN refers to instances of Pegasus topologies; for example, P3

is a graph with 144 nodes. A Pegasus unit cell contains twenty-four qubits, with each qubit coupled to one similarly aligned qubit in the cell and two similarly aligned qubits in adjacent cells, as shown in Figure 22. An Advantage QPU is a lattice of 16x16

such unit cells, denoted as a P16

Pegasus graph.

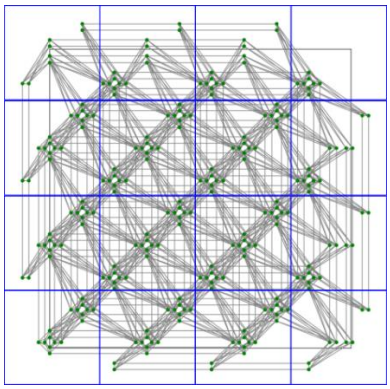

Figure 2.22 Pegasus P_4 graph [49]

Figure 2.22 visualizes Pegasus unit cells in a P4 graph, where qubits are represented as green dots, and couplers are represented as gray lines.

In a more formal representation, the Pegasus unit cell is depicted as 48 halves of qubits, which are split between adjacent unit cells. Figure 2.23 presents the Pegasus unit cell by illustrating 48 halves of qubits from adjacent unit cells. The qubits are represented as truncated loops (double lines), while the internal couplers are represented as dots. The external and odd couplers are depicted as dots connected by short lines [49].

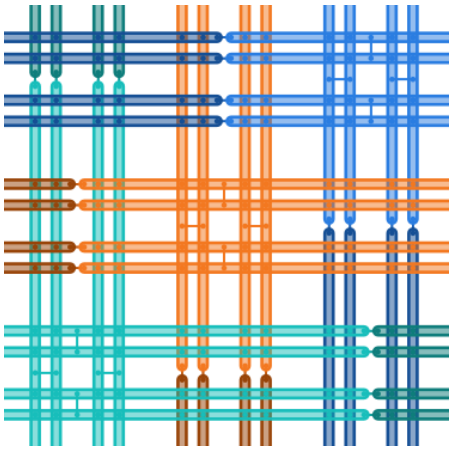

Figure 2.23 Pegasus unit cell [49]

2.2.2 Embedding QUBO on DWave Quantum Annealer

In order to solve problems using a Quantum Annealer (QA), the input must be in the form of Ising or Quadratic Unconstrained Binary Optimization (QUBO) models. These models are equivalent and return a cost function that needs to be minimized. This cost function can be represented as a graph, where the nodes of the graph represent the variables of the problem and the edges represent the constraints between the variables.

This graph is then matched onto the graph of the QA, which is referred to as the host graph.

Sometimes, it is impossible to find a direct (D) mapping between the logical variables and the physical variables of the system. In these cases, an embedding of the problem graph onto the host graph is performed [44]. The nodes that represent the same variable are grouped together to form a chain, and the chains also have a chain-strength parameter.

The QA machine has a specific qubit structure or topology, and often, it is not possible to directly map the variables of the problem graph to the annealer's topology. This is where minor embedding comes into play, where a representation of the problem graph within the topology of the annealer is found. In this process, a set of rules must be followed, including mapping a logical variable to a set of connected physical qubits, ensuring the coupling strength between them is $-\infty$, and distributing the bias of the logical variable to each qubit in the chain.

Embedding is a process used to solve problems on a Quantum Annealer when there is no direct mapping between the problem graph and the host graph. Since the Quantum Annealer has a limited size, the goal of the embedding algorithm is to use as few qubits as possible. It is also important to minimize the maximum chain length in the mapping, as experiments have shown that the performance of the annealer decreases with larger chain sizes.

2.3 Minimum Vertex Cover & Solution Approaches

A vertex cover is a fundamental concept in graph theory that has numerous practical applications in fields such as computer science, operations research, and network design. In a graph, a vertex cover is a subset of the vertices such that every edge in the graph is incident to at least one vertex in the subset. In other words, a vertex cover is a set of vertices that "covers" all the edges of the graph [28].

Formally, we are given an undirected graph $G = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$. A subset $V' \subseteq V$ is called a vertex cover if every edge in E has at least one endpoint in V', that is, if for every $e = (u, v) \in E$ it holds true that $u \in V'$ or $v \in V'$. A minimum vertex cover is a vertex cover of minimum size.

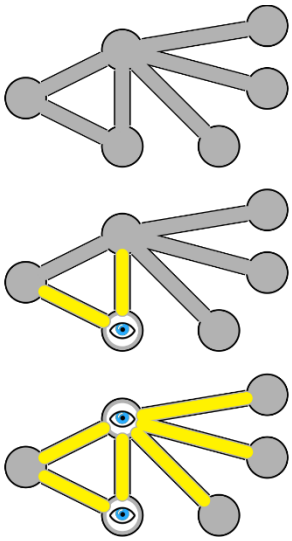

Figure 2.24 Example Minimum Vertex Cover [50]

Figure 2.24 shows an example graph that has a vertex cover comprising 2 vertices (bottom), but none with fewer.

The minimum vertex cover problem is known to be NP-hard, that as the size of the graph increases, the computational complexity grows exponentially. However, recent advances in quantum computing have opened up the possibility of using quantum annealing to solve NP-hard optimization problems such as the minimum vertex cover problem.

The MVC problem is a well-studied optimization problem, and there have been various approaches proposed to solve it. One of the earliest and most well-known approaches is the brute-force algorithm, which involves checking all possible combinations of vertices to determine the minimum vertex cover. However, this approach is inefficient for large graphs due to the exponential growth of the search space.

Another approach is a heuristic algorithm, which is a non-deterministic algorithm that does not guarantee an optimal solution. One example of a heuristic algorithm for the MVC problem is the Greedy algorithm, which iteratively selects a vertex with the highest degree and adds it to the vertex cover until all edges are covered. This approach is fast and simple, but it can produce suboptimal solutions [29].

Approximation algorithms are also commonly used to solve the MVC problem. One popular approximation algorithm is the 2-approximation algorithm, which is guaranteed to produce a vertex cover that is at most twice the size of the minimum vertex cover [30]. Another approximation algorithm is the LP rounding algorithm, which solves a linear programming relaxation of the problem and rounds the resulting fractional solution to an integer solution [31].

Recently, there has been growing interest in using quantum computing to solve optimization problems, including the MVC problem. Quantum annealing, in particular, has been shown to be a promising approach for solving the MVC problem. Various quantum-inspired heuristic algorithms have also been proposed, such as the Quantum-inspired Tabu Search (QTS) algorithm [32] and the Quantum-inspired Genetic Algorithm (QGA).

2.4 Decomposition Methods

The current state of decomposition methods for solving large Minimum Vertex Cover (MVC) problems on a quantum annealer has seen significant progress in recent years. Several decomposition algorithms have been proposed, each with its unique approach to solving the problem. These algorithms aim to overcome the limitations of quantum annealers' limited qubit connectivity and capacity by breaking down the original problem into smaller subproblems that can be solved on the quantum annealer.

One of the earliest decomposition algorithms proposed is the Recursive-Edge-Contraction (REC) method [50]. The REC method breaks down the original graph into smaller subgraphs by recursively removing edges and contracting the vertices. The generated subgraphs can then be solved on the quantum annealer, and the solutions are merged to obtain the final solution. This method has shown promising results in solving large-scale MVC problems, but it suffers from generating a large number of subproblems, which increases the runtime.

Another approach to MVC decomposition is the Branch-and-Bound (BnB) method [52]. The BnB method uses a divide-and-conquer strategy to break down the original problem into smaller subproblems. It starts by solving a small subproblem using a classical solver and obtains a lower bound on the solution. The algorithm then branches into several subproblems, each of which is a fraction of the original problem, and solves them recursively. The solutions obtained from each subproblem are then combined, and the best solution is selected. This method has been shown to produce high-quality solutions and is well-suited for large-scale MVC problems.

Recently, the Hybrid Recursive-Edge-Contraction and Branch-and-Bound (HREC-BnB) method [27] has been proposed, which combines the advantages of both the REC and BnB methods. The HREC-BnB method generates subproblems using the REC method and solves them using the BnB method. This method has shown to produce

high-quality solutions while reducing the number of subproblems generated, thus decreasing the runtime.

In summary, the current state of decomposition methods for solving large MVC problems on a quantum annealer has seen significant progress. These methods have shown to overcome the limitations of quantum annealers and enable the solution of larger problems. The emergence of novel decomposition algorithms has provided new opportunities to explore the potential of quantum computing in solving real-world problems.

2.5 Jgrapht

The jgrapht library [34] is a Java-based library that provides a wide range of tools for graph analysis and manipulation. It supports various types of graphs, including directed and undirected graphs, weighted and unweighted graphs, and graphs with self-loops and parallel edges. The library also includes algorithms for various graph problems, such as shortest path, maximum flow, and minimum spanning tree.

For the minimum vertex cover problem, jgrapht provides **jgrapht.algorithms.vertexcover.exact**, a package that contains implementations of exact algorithms for solving the vertex cover problem. The exact package in JGraphT provides implementations of algorithms that can solve the vertex cover problem exactly, meaning they are guaranteed to find the optimal solution. These algorithms include brute-force algorithms, algorithms based on linear programming, and algorithms based on branch and bound techniques. By providing these exact algorithms, JGraphT enables users to find the optimal solution to the vertex cover problem on graphs of any size, albeit at the expense of potentially long computation times for large graphs. To use the jgrapht library for the minimum vertex cover problem, one needs to create a graph object and add vertices and edges to it. The library provides various constructors for creating different types of graphs, as well as methods for adding and removing vertices and edges. The library also provides a VertexCoverAlgorithm interface for computing the minimum vertex cover from the maximal cliques.

Although JGraphT is written in Java, it can be used in Python through the Py4J library. The Python interface to JGraphT provides all of the functionality of the Java version, including algorithms for graph manipulation and analysis. The Python bindings of JGraphT is a pure python/native package having no dependency on the JVM. During

the build process the backend JGraphT library is compiled as a shared library and bundled inside the python package.

Overall, the jgrapht library is a versatile and efficient tool for graph analysis and manipulation in Java. It will be used in this study as a fully classical solver for the minimum vertex cover problem. This choice allows for a comparison between the performance of the classical algorithm and the DBR algorithm with classical solver, as well as the two quantum solvers. The jgrapht library provides an exact algorithm for minimum vertex cover, making it a suitable candidate for comparison with the quantum and hybrid quantum/classical approaches. By evaluating the performance of the jgrapht solver against the quantum and hybrid solvers, we can assess the potential advantages and limitations of quantum computing for solving combinatorial optimization problems. Such comparisons are essential for identifying the strengths and limitations of different solvers and for informing the development of new quantum computing algorithms.

2.6 NetworkX

The NetworkX library [35] is a popular and extensively used Python library for graph and network analysis. It provides tools for creating, manipulating, and analyzing graphs, as well as a wide variety of algorithms for various graph problems. In my work, I utilized two functions from the library for solving the maximum clique and minimum vertex cover problems on graphs.

The first function, maximum_clique_exact_solve_np_hard(G_in), is used to solve the maximum clique problem for a given graph. The maximum clique is the largest subset of nodes in the graph that are all adjacent to each other. The function uses the graph_clique_number function from NetworkX to calculate the size of the largest clique, and then uses the find_cliques function to generate all cliques in the graph. Finally, it returns the largest clique found in the graph.

The second function, minimum_vertex_cover_exact_solve_np_hard(G), is used to solve the minimum vertex cover problem for a given graph. The minimum vertex cover is the smallest subset of nodes in the graph such that each edge in the graph is adjacent to at least one node in the subset. The function first creates the complement graph of the input graph using the complement function, then calculates the maximum clique in the complement graph using the maximum_clique_exact_solve_np_hard function. The

minimum vertex cover is then calculated by subtracting the maximum clique from the set of nodes in the input graph.

In addition, I used the gnp_random_graph(num_nodes, num_density) function from NetworkX to generate random graphs for testing and benchmarking the different solvers. The function creates a random graph with num_nodes nodes and num_density edge probability. The resulting graph can be used for testing and comparing the performance of different algorithms on graphs of varying sizes and densities.

Chapter 3

Implementation

3.1 Decomposition method for Minimum Vertex Cover

To solve the Minimum Vertex Cover (MVC) problem for a given graph G = (V, E), the paper "Solving large Minimum Vertex Cover problems on a quantum annealer" [37] suggests a decomposition algorithm. This algorithm splits the problem into two subproblems by considering two cases for each vertex v ∈ V, resulting in subproblems of reduced size. The goal is to find the Minimum Vertex Cover V' ⊆ V.

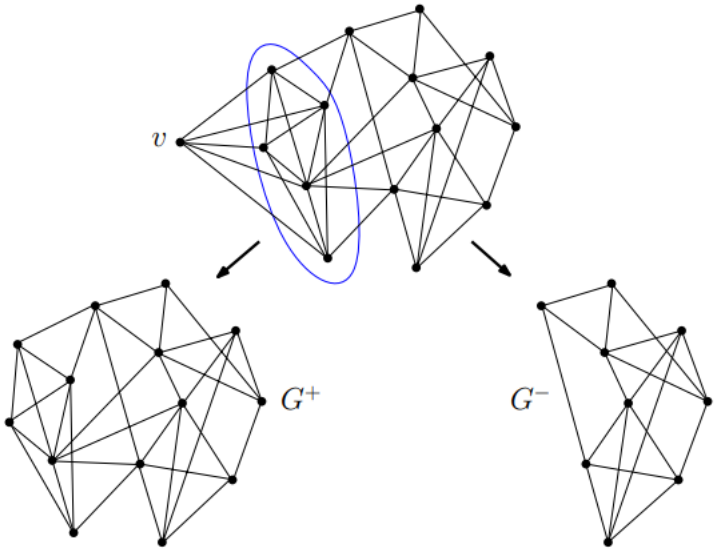

Figure 3.1 Demonstration of the process of splitting vertices at a vertex v.

In the first case, if vertex v belongs to the Minimum Vertex Cover (V'), it is added to the set of vertices representing the MVC. Afterwards, v and all its adjacent edges can be removed since they are already covered by v's inclusion in the MVC. This is illustrated in Figure 3.1 using the subgraph G+.

In the second case, if v does not belong to the MVC ($v \notin V'$), we observe that for all edges with v as an endpoint (i.e., $(v, u) \in E$), it is necessary for u to belong to V'. This is because if v is not part of the MVC, those edges must still be covered by their other endpoint u in the MVC. Additionally, v can be removed from the graph G since we know it is not in the MVC. Similarly, all vertices u with $(u, v) \in E$ and their adjacent edges can be removed, as they are known to be part of the MVC.

In Figure 3.1, assuming that v is not part of the Minimum Vertex Cover ($v \notin V'$), all vertices enclosed within the blue circle must be included in the MVC. After removing v, its adjacent edges, and assigning the enclosed vertices to the MVC, the algorithm obtains the subgraph G-.

By considering the exhaustive cases of v belonging to V' and v not belonging to V' ($v \in V'$ and $v \notin V'$), the graph G can be split into the subgraphs G+ and G-. The algorithm then continues computing the MVC on both subgraphs. If either subgraph is still too large to compute an exact MVC solution, the decomposition technique can be recursively applied. In each level of recursion, proper bookkeeping is necessary to track the current set of cover vertices for each generated subgraph.

The algorithm ensures the termination by reducing the graph size by at least one in each recursion level, as neither the subgraph G+ nor G- contains the vertex v. The described algorithm provides the exact Minimum Vertex Cover (MVC) solution when an exact method is used to solve MVC on the leaf-level subgraphs. However, when employing a solver like D-Wave for graphs of up to 15 vertices, an exact solution is not guaranteed as our experimental evaluation has shown. Nevertheless, the algorithm can be applied probabilistically. If the solver used on the leaf-level subgraphs finds optimal solutions with probability p, the decomposition algorithm will report the correct MVC for the original graph with the same probability p, because the probabilistic nature of the solver used in the leaf-level subgraphs affects the overall accuracy of the decomposition algorithm.

3.2 DBR Algorithm

The DBR algorithm employs several techniques, including decomposition, bounds and reduction to solve the Minimum Vertex Cover (MVC) problem. The bounds below are employed during the recursion to prune those subproblems which cannot contain vertices belonging to the MVC of the input graph. In addition to using upper and lower

bounds, three reduction techniques are also used that allow to figure out a partial solution to the MVC by deciding for a subset of the vertices of G whether they belong to V 0 or not.

3.2.1 High Degree Vertex Selection

The selection of the vertex for decomposition during the computation significantly affects the number of generated subproblems and, consequently, the overall running time. When solving the Minimum Vertex Cover (MVC) problems, and selecting the vertex with the highest degree has shown to be the most effective [37].

3.2.2 Deterministic Lower Bounds

Firstly, the "min weighted vertex cover" function from the NetworkX [35] package is utilized [38]. This function employs the Bar-Yehuda and Even algorithm [39] to compute an approximate vertex cover size, which is at most twice the size of the optimal cover. Dividing this result by two provides a lower bound on the MVC size.

Secondly, the algorithm applies the matrix rank upper bound proposed by Budinich [40] to each generated subgraph. This upper bound is based on the order of the maximum independent set. Since the complement of any independent set corresponds to a vertex cover, an upper bound on the maximum independent set size serves as a lower bound on the MVC size.

Lastly, the algorithm utilizes the easily computed minimum degree bound presented by Willis [41].

By utilizing these methods, the algorithm can obtain lower bounds for the MVC size, which assist in the analysis and optimization of the graph.

3.2.3 Chromatic Number Heuristic Upper Bound

An upper bound on the chromatic number can be obtained through any graph coloring. This is because in a maximum clique, all vertices must have distinct colors assigned to them. Consequently, the chromatic number also serves as an upper bound for the clique number. By extension, it provides an upper bound for the size of the maximum independent set in the complement graph. This upper bound can be utilized as described in the previous point. To compute a graph coloring, we utilize the greedy color heuristic function from the NetworkX package [46], which is applied to the complement graph G.

3.2.4 Reduction Techniques

In addition to utilizing upper and lower bounds, the algorithm incorporates three reduction techniques to obtain a partial solution for the Minimum Vertex Cover (MVC) problem by making decisions about whether certain vertices in graph G belong to V' or not. These reduction techniques are as follows:

The first technique, called neighbor-based vertex removal, involves identifying and removing triangles, vertices of degree 1, and vertices of degree zero in any subgraph. Since any two vertices within a triangle of degree 2 belong to the MVC, the algorithm adds a contribution of 2 to the MVC size and eliminate the triangle. Similarly, vertices of degree 1 are automatically part of the MVC, so the algorithm remove them along with their only neighbor after adding a contribution of 1 to the MVC size. Vertices of degree zero can be removed without further processing.

Another reduction technique operates on the QUBO formulation of the MVC problem instead of directly on the graph. The algorithm generates the corresponding MVC Hamiltonian and analyze it using general-purpose preprocessing techniques that identify binary relations or persistencies in QUBO or Ising Hamiltonians. If a variable xv for vertex v is assigned a specific value in the persistency analysis (e.g., xv = 1), the algorithm adds v to the current vertex cover, remove v and its adjacent edges from the subgraph, and update the MVC size accordingly. If xv = 0, the algorithm can remove v and its adjacent edges without additional processing. The persistency analysis is performed using the QPBO Python bindings by Rother et al [42]

Reduction methods from the vertex cover-master Java package by Akiba and Iwata [43] are also amplied, which includes various reduction techniques used in the theoretical study of exponential-complexity branch-and-reduce algorithms for the MVC problem. These techniques involve degree-one reductions, decomposition, dominance rules, unconfined vertex reduction, LP- and packing reductions, as well as folding-, twin-, funnel-, and desk reductions. By repeatedly applying these reduction methods, vertex cover-master returns a superset of the MVC. Consequently, any vertex not included in the output can be confidently removed from consideration.

3.3 Quantum Solvers

There are two Python modules that are used as solver for the DBR algorithm. We use the D-Wave quantum annealer to solve the minimum vertex cover problem. One is on the Chimera topology and the other is on the Pegasus.

The solvers define a binary variable "x" for each node in the graph, where x[i] = 1 if node[i] is in the minimum vertex cover, and 0 otherwise, as shown in Pseudocode Section 3.1 line 1.

Pseudocode Section 3.1

```
    # x[i] = 1 if node[i] is in minimum vertex cover
    # x[i] = 0 otherwise
1   x = {n: Binary(n) for n in G.nodes}
```

We then set up a Constrained Quadratic Model (CQM) that minimizes the size of the cover while ensuring that each edge in the graph has at least one node in the cover, as shown in Pseudocode Section 3.2 line 2 and line 3.

Pseudocode Section 3.2

```
   # Create a constrained quadratic model
2  cqm = ConstrainedQuadraticModel()
   # Set the objective to minimize the size of the cover
3  cqm.set_objective(sum(x[i] for i in G.nodes))

   # Add constraint that each edge has at least a node in the cover
4   for (i,j) in G.edges:
5       cqm.add_constraint(x[i] + x[j] >= 1, label = f'edge{i}_{j}')
```

The Constrained Quadratic Model is created to model the minimum vertex cover problem. The objective is to minimize the size of the minimum vertex cover, which is the sum of all x[i] values.

To add constraints to the CQM, a linear inequality is created for each edge in the input graph G. If an edge connects nodes i and j, the inequality x[i] + x[j] >= 1 is added to ensure that at least one of the two nodes is included in the minimum vertex cover, as shown in Pseudocode Section 3.2 line 5.

The values of the constraints were chosen to enforce the condition that each edge has at least one node in the minimum vertex cover. The objective function is designed to minimize the size of the minimum vertex cover.

Pseudocode Section 3.3

```
# Convert the CQM to BQM
6   bqm, invert = dimod.cqm_to_bqm(cqm)
```

The Constrained Quadratic Model is converted into a binary quadratic model (BQM) using the "dimod.cqm_to_bqm()" function, as shown in Pseudocode 3.3 line 6. The reason for this conversion is to transform the constrained problem represented by the CQM into an unconstrained binary quadratic problem that can be solved by the D-Wave system. On the other hand, the BQM is the standard form of an optimization problem that is compatible with the D-Wave system. It represents an unconstrained Ising or quadratic binary optimization problem. The BQM consists of binary variables (x[i]) and quadratic terms (interaction between variables) that define the objective function.

Pseudocode Section 3.4

```
7    sampler = DWaveSampler(solver={'topology__type': 'chimera'})
8   embedding_sampler = AutoEmbeddingComposite(sampler)
9   sampleset = embedding_sampler.sample(bqm,num_reads = 100)
```

In Pseudocode Section 3.4 line 7 we create an instance of the DWaveSampler class, which represents a connection to a D-Wave quantum processing unit (QPU). The solver parameter can be set to either 'chimera' or ‘pegasus’, which specifies the topology type of the QPU.

In Section 3.4 line 8, we create an instance of the AutoEmbeddingComposite class, which is a composite sampler that automatically selects an embedding for the problem. The sampler object created in the previous step is passed as an argument to the AutoEmbeddingComposite constructor.

Section 3.4 line 9 uses the embedding_sampler to sample from the provided Binary Quadratic Model (BQM). The sample method is called with BQM and num_reads arguments to specify the number of samples to be generated (100 in this case). The returned sampleset object contains the results of the sampling process.

Pseudocode Section 3.5

```
11    for smpl, energy in sampleset.data(['sample','energy']):
12       minimum_vertex_cover = [n for n, value in smpl.items() if value
             == 1 and not str(n).startswith('slack')]
13    return minimum_vertex_cover
```

We finally retrieve the solution from the result and store the minimum vertex cover in a list called "minimum_vertex_cover", as shown in Pseudocode Section 3.5 lines 11, line 12 and line 13.

Chapter 4

Experimental evaluation

4.1 Introduction

The experimental evaluation of the proposed decomposition algorithm is a crucial component in assessing its performance and effectiveness in solving large-scale graph optimization problems. In this chapter, we present the results and analysis of our experiments conducted on a diverse set of graphs, ranging from 20 to 120 nodes, with densities spanning from 0.1 to 0.9.

The decomposition algorithm, initially introduced in the paper "Decomposition Algorithms for Scalable Quantum Annealing," served as the foundation for our implementation. We utilized a classical solver, as indicated earlier, which leveraged the NetworkX package in Python. Additionally, to explore the benefits of quantum computing, we transformed the input graph into a Constrained Quadratic Model and employed D-Wave samplers to solve the resulting optimization problem using quantum annealing techniques.

To ensure a comprehensive evaluation, we selected a wide range of graph instances, varying both in size and density. The inclusion of graphs with different node counts and density levels allows us to examine the algorithm's performance under varying levels of complexity and problem structures. However, it is important to note that due to the computational demands of these experiments, we sought access to a university-provided computer with enhanced processing power, as our personal laptop was insufficient for executing the algorithms on larger graphs.

In addition to evaluating the proposed decomposition algorithm, we conducted a comparative analysis by running a fully classical algorithm on the same set of graph

instances. This comparison provides insights into the potential advantages and trade-offs offered by our algorithm in comparison to conventional classical approaches.

By conducting these experiments and analyzing the results, we aim to assess the efficiency, scalability, and performance characteristics of the proposed decomposition algorithm. The findings from this evaluation will enable us to gain a deeper understanding of the algorithm's strengths and limitations, as well as provide insights into its applicability to real-world graph optimization problems.

In the following sections, we will present the experimental setup, describe the graphs used, discuss the computational resources employed, and present the results obtained from running both the proposed decomposition algorithm and the fully classical algorithm. Through a thorough analysis and interpretation of these results, we will extract meaningful insights and draw conclusions regarding the performance and efficacy of our algorithm.

4.2 Experimental Setup

For our experimental evaluation, we established a well-defined setup that encompassed the following key components:

4.2.1 Dataset and Input Data

We utilized a diverse set of graph instances as our input data. These graphs were generated synthetically to ensure controlled experimentation. The graphs varied in size, ranging from 20 to 120 nodes, and were constructed with densities spanning from 0.1 to 0.9. The inclusion of graphs with different sizes and densities allowed us to evaluate the algorithm's performance across various problem complexities.

```
for i in range(19): #20 - 110
    for j in range(9): #0.1 - 0.9
        num_nodes = 20 + i * 5
        num_density = 0.1 + j * 0.1
        G = nx.gnp_random_graph(num_nodes, num_density)
```

gnp_random_graph #

gnp_random_graph(n, p, seed=None, directed=False) [source]

Returns a $G_{n,p}$ random graph, also known as an Erdős-Rényi graph or a binomial graph.

The $G_{n,p}$ model chooses each of the possible edges with probability p.

Parameters:

n : int
The number of nodes.

p : float
Probability for edge creation.

4.2.2 Preprocessing:

Prior to running the algorithms, we performed necessary preprocessing steps on the input graphs. These steps involved ensuring the graphs were connected and removing any redundant or irrelevant information.

4.2.3 Hardware Configuration:

To execute our experiments, we utilized a computer with enhanced processing capabilities, which was provided by our university. This system was equipped with a high-performance multi-core processor (Intel(R) Core (TM) i5-8365U CPU @ 1.60GHz 1.90 GHz), ample RAM (8,00 GB, 7,72 GB usable), and a dedicated GPU. The additional computational resources were necessary to handle the computational demands of solving optimization problems for larger graphs.

4.2.4 Software Configuration:

We implemented the algorithms in Python, leveraging various libraries and frameworks. Specifically, for the classical solver, we utilized the NetworkX package, a powerful graph manipulation library in Python. For the quantum annealing solver, we converted the input graph into a Constrained Quadratic Model and employed D-Wave samplers. The D-Wave Ocean software suite, including the D-Wave Python API, provided the necessary tools for interacting with the D-Wave quantum annealing hardware.

4.3 Results Presentation

We will present graphs that display the execution time comparing the four different approaches. Specifically, the blue line represents the results obtained using a fully classical solver from the jgrapht package in Python. On the other hand, the other three lines correspond to the algorithm proposed in the paper "Decomposition Algorithms for Scalable Quantum Annealing," utilizing three different solvers. The red line represents the outcomes obtained using a classical solver from the NetworkX package in Python. The grey line represents the results obtained using a quantum solver based on the Pegasus topology quantum computer. Lastly, the yellow line represents the outcomes achieved using a quantum solver based on the Chimera topology quantum computer.

For each graph size varying from 20 to 120 nodes, we will present three graphs illustrating the execution time for varying density sizes ranging from 0.1 to 0.9. These graphs will demonstrate the performance of the algorithm under three different decomposition limits. As mentioned earlier, decomposition limit is the subproblem in which the solver is called, because the graph size is small enough to be able to find the Minimum-Vertex Cover.

4.3.1 Section 1

In first section focusing on graph sizes ranging from 20 to 70, it is observed that both the classical algorithm and the DBR algorithm successfully solve the MVC problem instantaneously when using a classical solver. However, when employing the two quantum solvers, a communication overhead is encountered, resulting in an execution time of approximately 5 to 6 seconds, as depicted in Figures 4.3 to 4.5

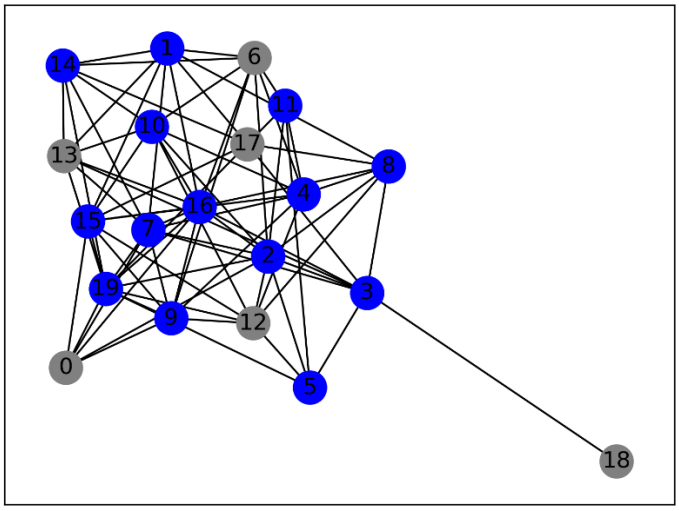

Figure 4.1 Illustration of Minimum Vertex Cover (MVC) obtained by the jgrapht algorithm on a graph consisting of 20 nodes with a density of 0.4.

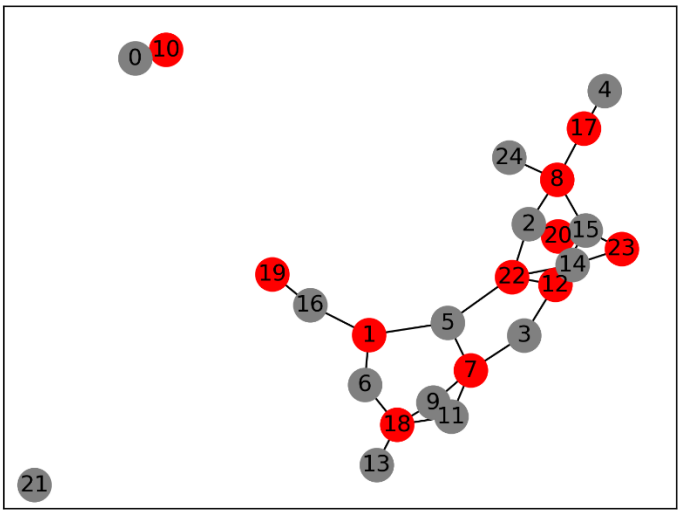

Figure 4.2 Illustration of Minimum Vertex Cover (MVC) obtained by the DBR algorithm with the NetworkX solver on a graph consisting of 25 nodes with a density of 0.1.

Figure 4.1 and Figure 4.2 present, utilizing the NetworkX Python library, graphical representations of the solutions returned by the DBR algorithm. It is worth noting that even though Figure 4.1 depicts a small-sized graph, the Minimum Vertex Cover (MVC) is larger due to the higher density of the graph compared to the graph in Figure 4.2. This

relationship between graph density and MVC size is based on the definition of the MVC problem and the concept of vertex covers. When a graph is denser, there are more edges that need to be covered. Each edge requires at least one vertex from the vertex cover to ensure that all edges are incident to a vertex in the cover. Therefore, as the number of edges increases, the size of the vertex cover also tends to increase.

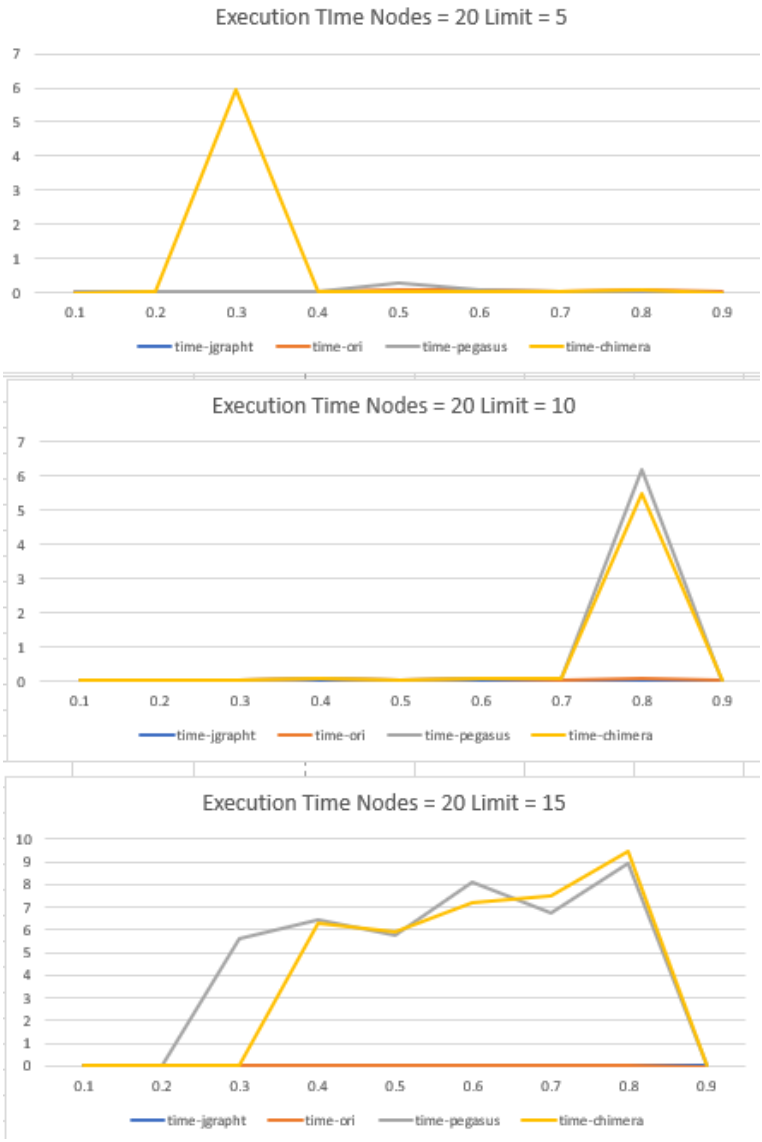

Figure 4.3 Execution time for graphs of size 20

In Figure 4.3, it can be observed that the execution time for both the classical algorithm and the DBR algorithm approaches zero. However, when utilizing the quantum solvers, a significant increase in execution time is evident. This observation is further illustrated in Figure 4.4, which presents a comparison between the partitioning time and the time taken by the solver. The solver consumes the majority of the total time due to the overhead associated with communication.

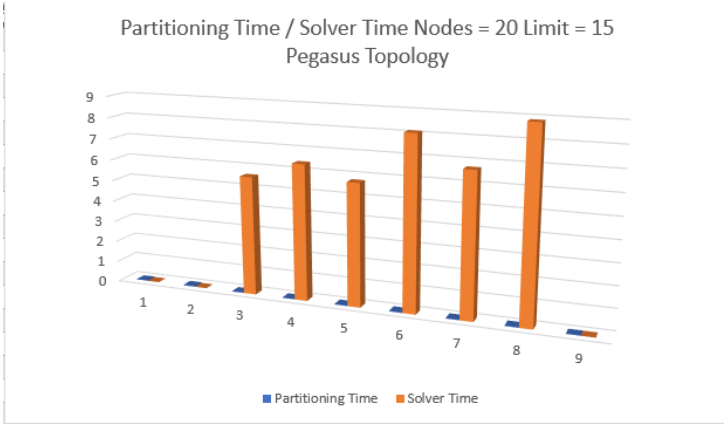

Figure 4.4 Comparative Evaluation of Partitioning Time and Solver Time for the DBR Algorithm Utilizing the Pegasus Topology Solver (Nodes = 20, Limit = 15)

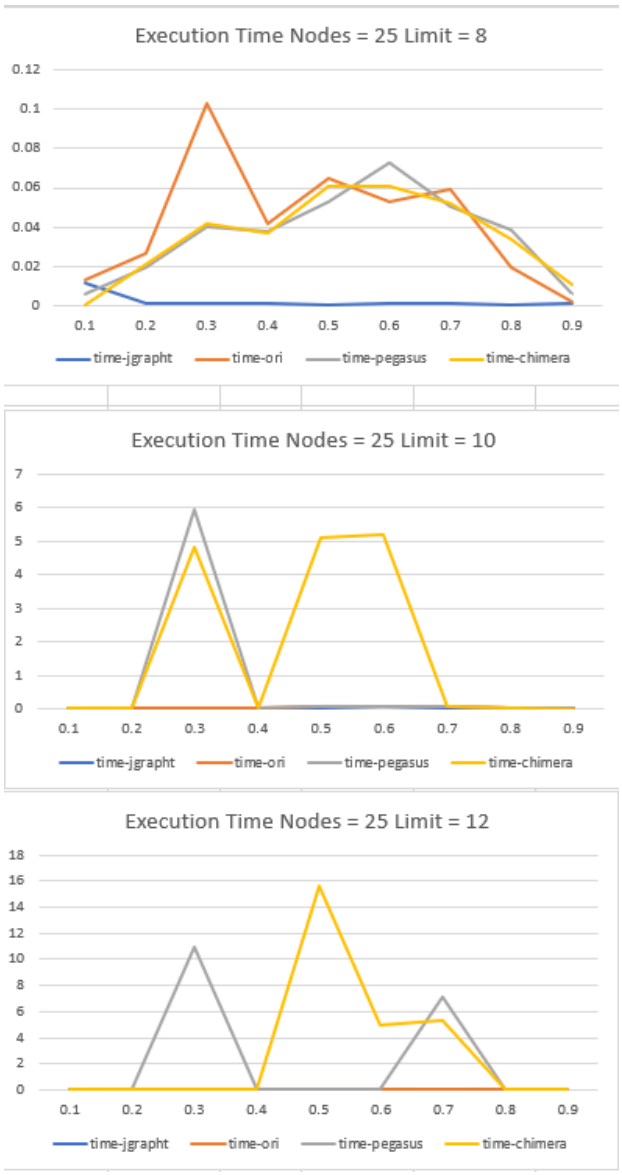

Figure 4.5 Execution time for graphs of size 25

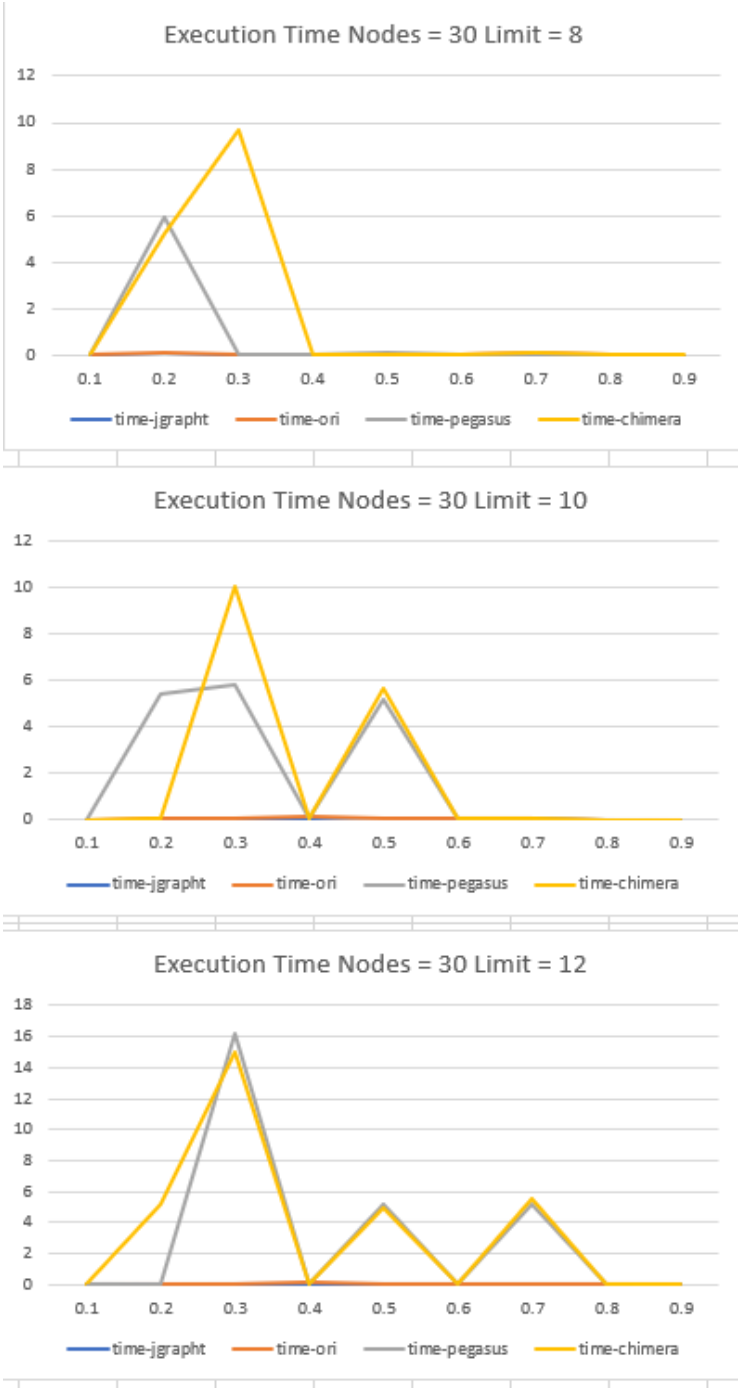

Figure 4.6 Execution time for graphs of size 30

Figure 4.5 and Figure 4.6 exhibis a similar trend as observed in Figure 4.3, but the execution time of the DBR algorithm utilizing quantum solvers increases even further. This is attributed to the increased number of times the quantum solvers are invoked, due to larger graph sizes, leading to a cumulative effect on the overall execution time.

4.3.2 Section 2

In the second section we are focusing on graph sizes ranging from 75 to 85.

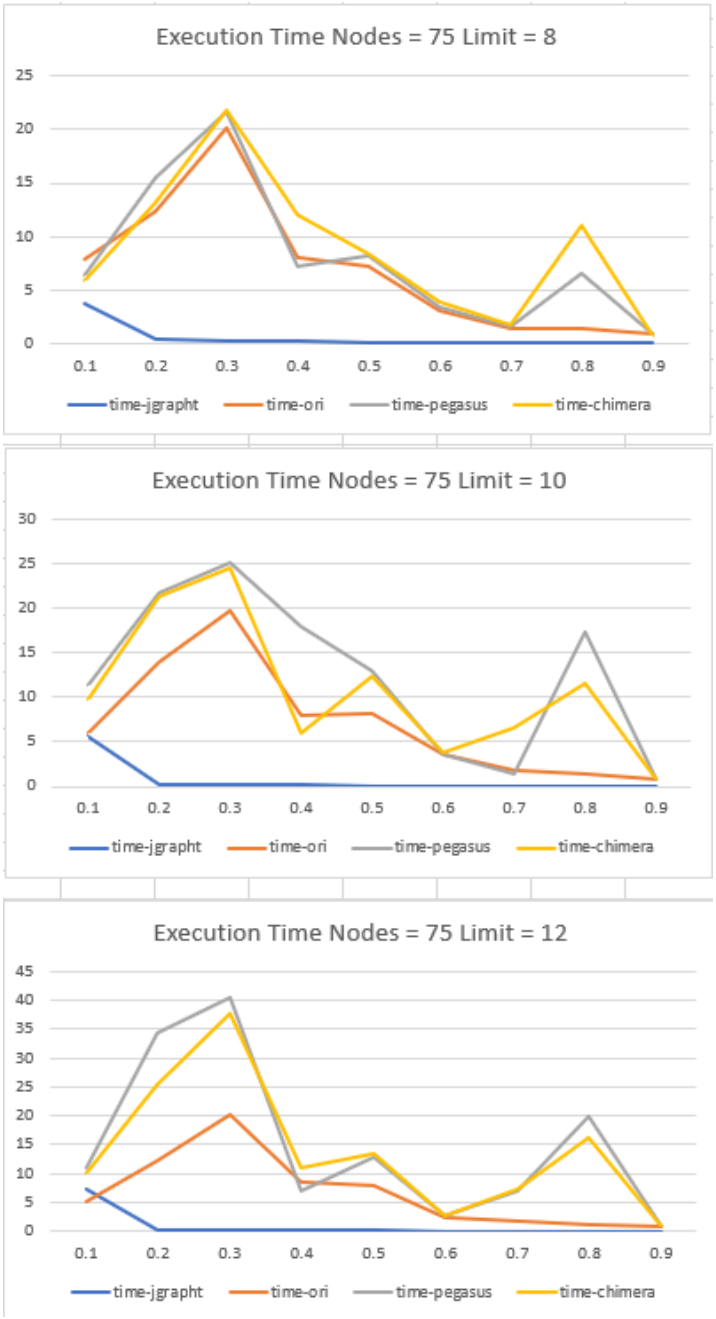

Figure 4.9 Execution time for graphs of size 75

Figure 4.9, Figure 4.10 and Figure 4.11 illustrate that the classical algorithm demonstrates instantaneous solutions for the MVC problem across all densities, except for a density of 0.1, which is resolved in approximately 5, 15 and 20 seconds accordingly.

Figure 4.10 Execution time for graphs of size 80

On the other hand, the DBR algorithm exhibits a slower performance for every density. Lower density graphs require more time compared to higher density graphs, with densities of 0.2 to 0.3 being the most demanding in terms of execution time, as depicted in Figures 4.8 to 4.10.

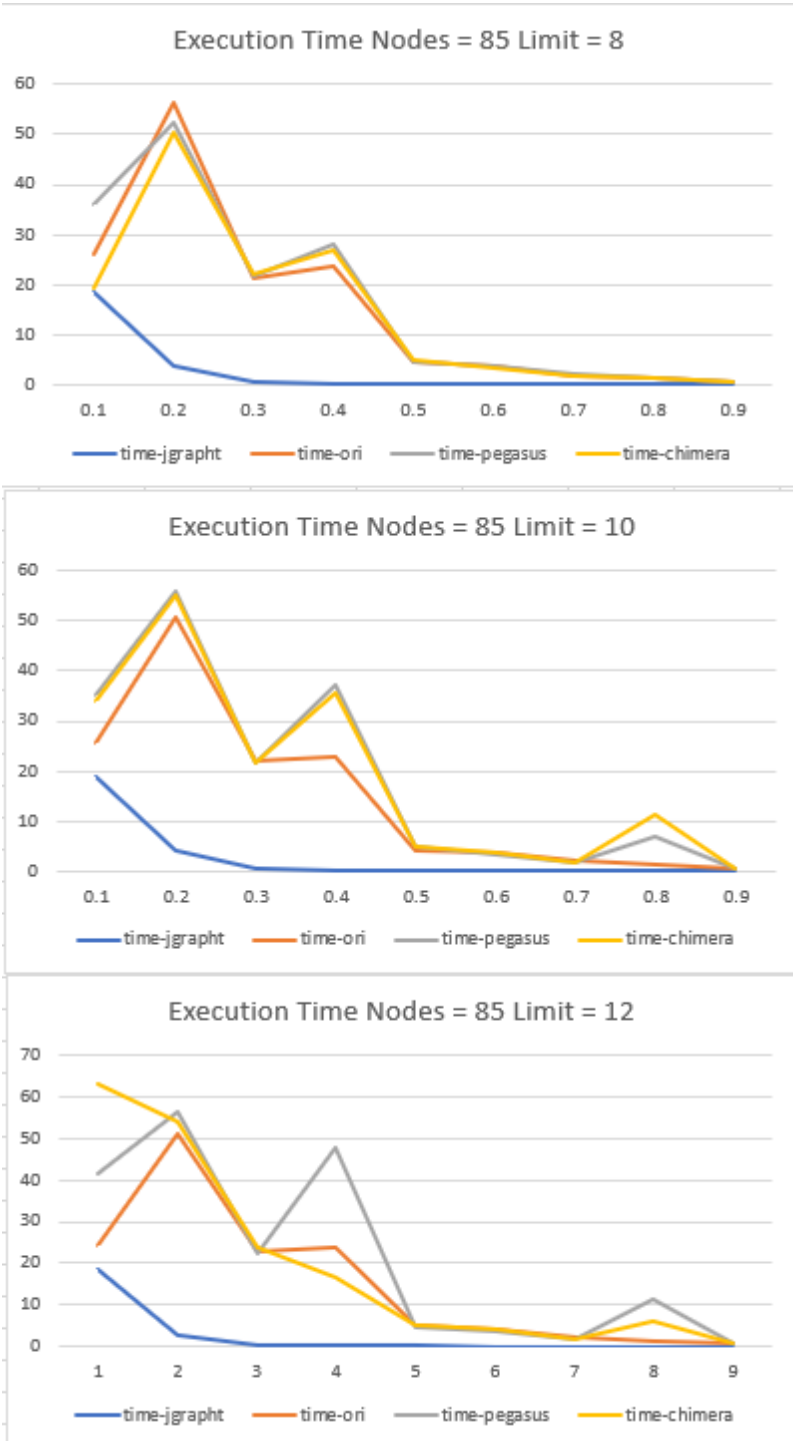

Figure 4.11 Execution time for graphs of size

Moreover, it is noteworthy to discuss that, at this particular graph sizes, the quantum solvers failed to return optimal solutions for decomposition limits greater than or equal to 15. Consequently, we were compelled to reduce the maximum decomposition size of our testing to 12 in order to obtain reliable results. This limitation indicates the difficulty faced by the quantum solvers in effectively handling larger decomposition limits for this specific graph size.

4.3.3 Section 3

In third section we are focusing on graph sizes ranging from 90 to 105. Notably, the peak execution time is observed for graphs with density of 0.2.

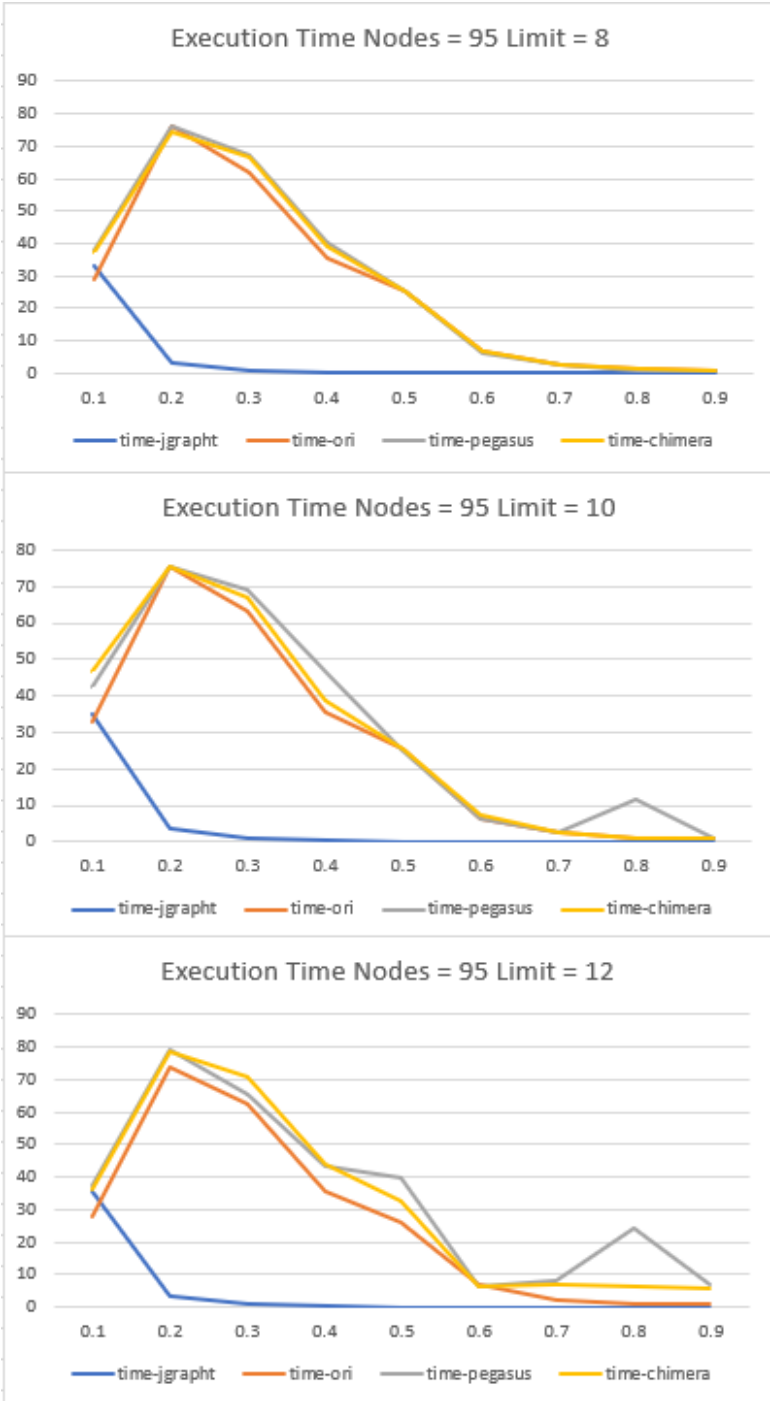

Figure 4.13 Execution time for graphs of size 90

Figures 4.13 and 4.14 illustrate that the classical algorithm demonstrates rapid solving of the MVC problem for all densities except for a density of 0.1, which exhibits similar or slower solving times compared to the DBR algorithm. The DBR algorithm exhibits

slower solving times for all other densities, with lower densities being more computationally demanding compared to higher density graphs

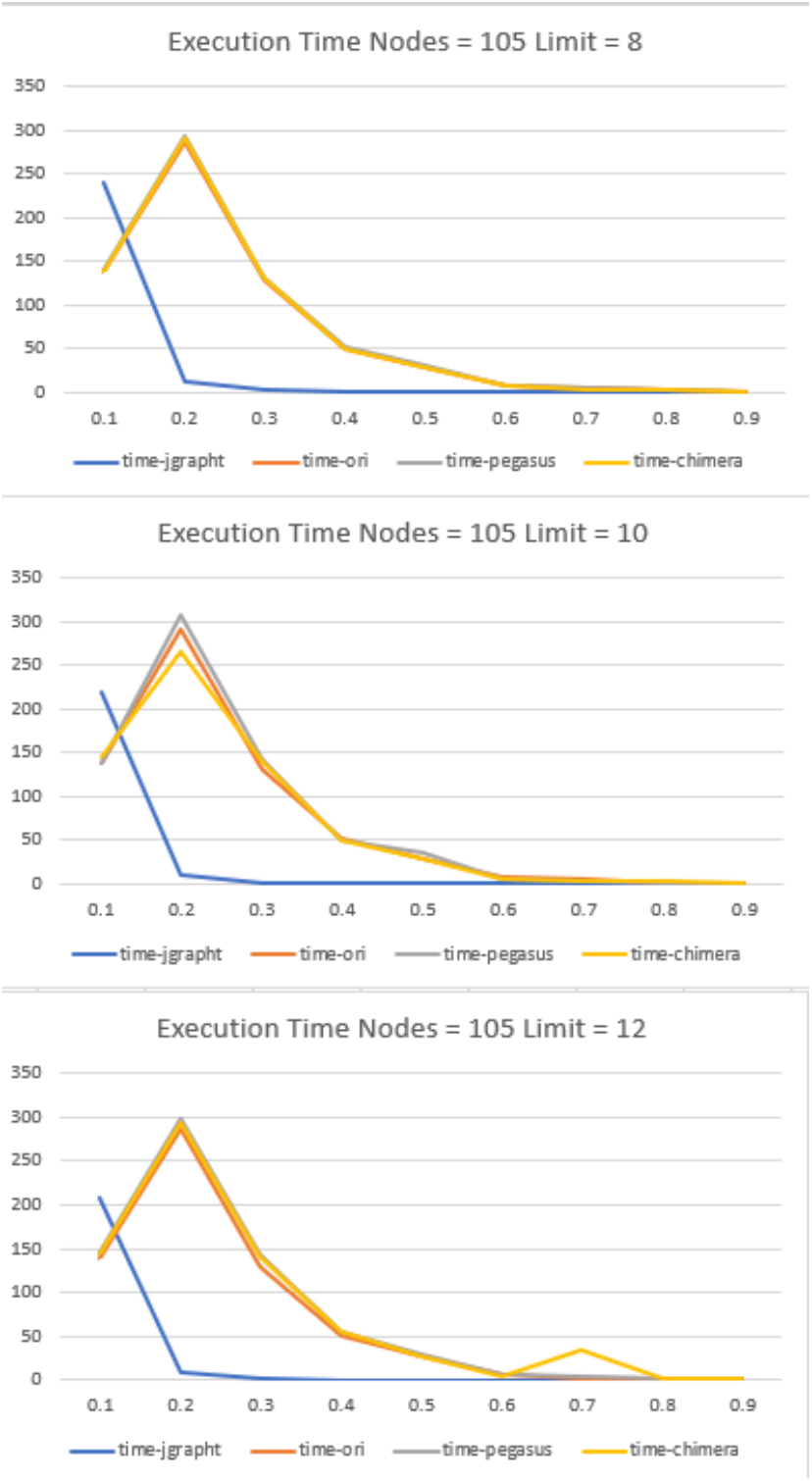

Figure 4.14 Execution time for graphs of size 105

Furthermore, DBR solution with the classical solver performs comparable with two approaches utilizing quantum solvers, with the communication overhead being negligible, as depicted in Figures 4.12 to 4.14.

4.3.4 Section 4

In the fourth section, we are focusing on graph sizes ranging from 110 to 120

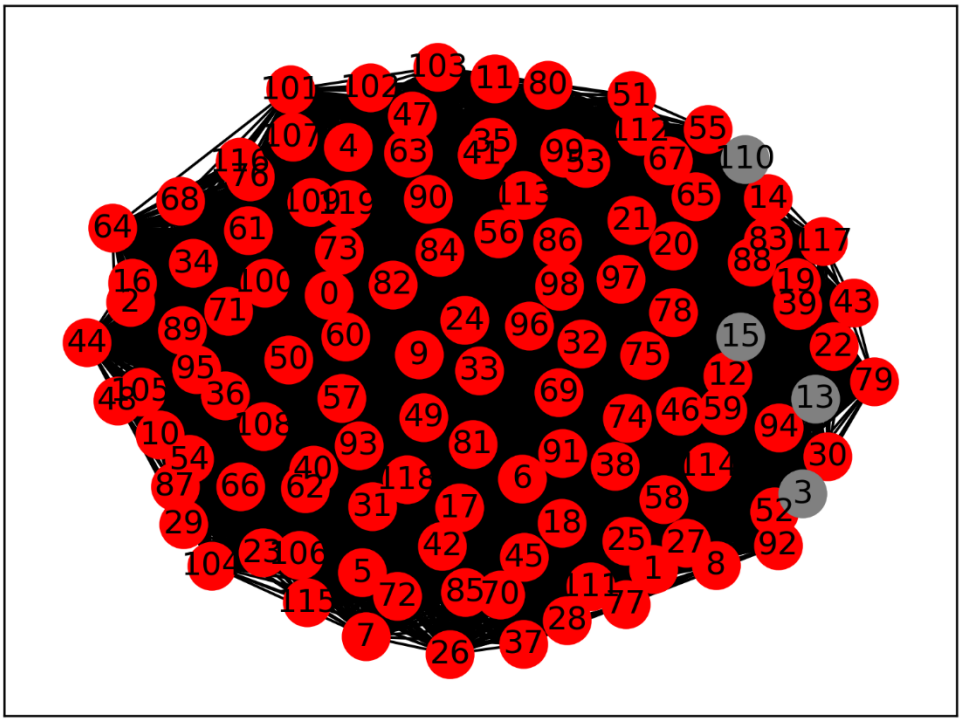

Figure 4.15 Illustration of Minimum Vertex Cover (MVC) obtained by the DBR algorithm with the NetworkX solver on a graph consisting of 120 nodes with a density of 0.9.

Figure 4.15 presents, utilizing the NetworkX Python library, a graphical representation of the solutions returned by the DBR algorithm. It is evident from the graph with 120 nodes and an edge density of 0.9 that nearly every node is included in the Minimum Vertex Cover (MVC). This observation arises due to the significant number of edges present in the graph. In order to ensure that all edges are covered, at least one node from each edge must be included in the MVC. Consequently, the MVC size becomes larger as the number of edges increases, resulting in the inclusion of a greater number of nodes in the cover.

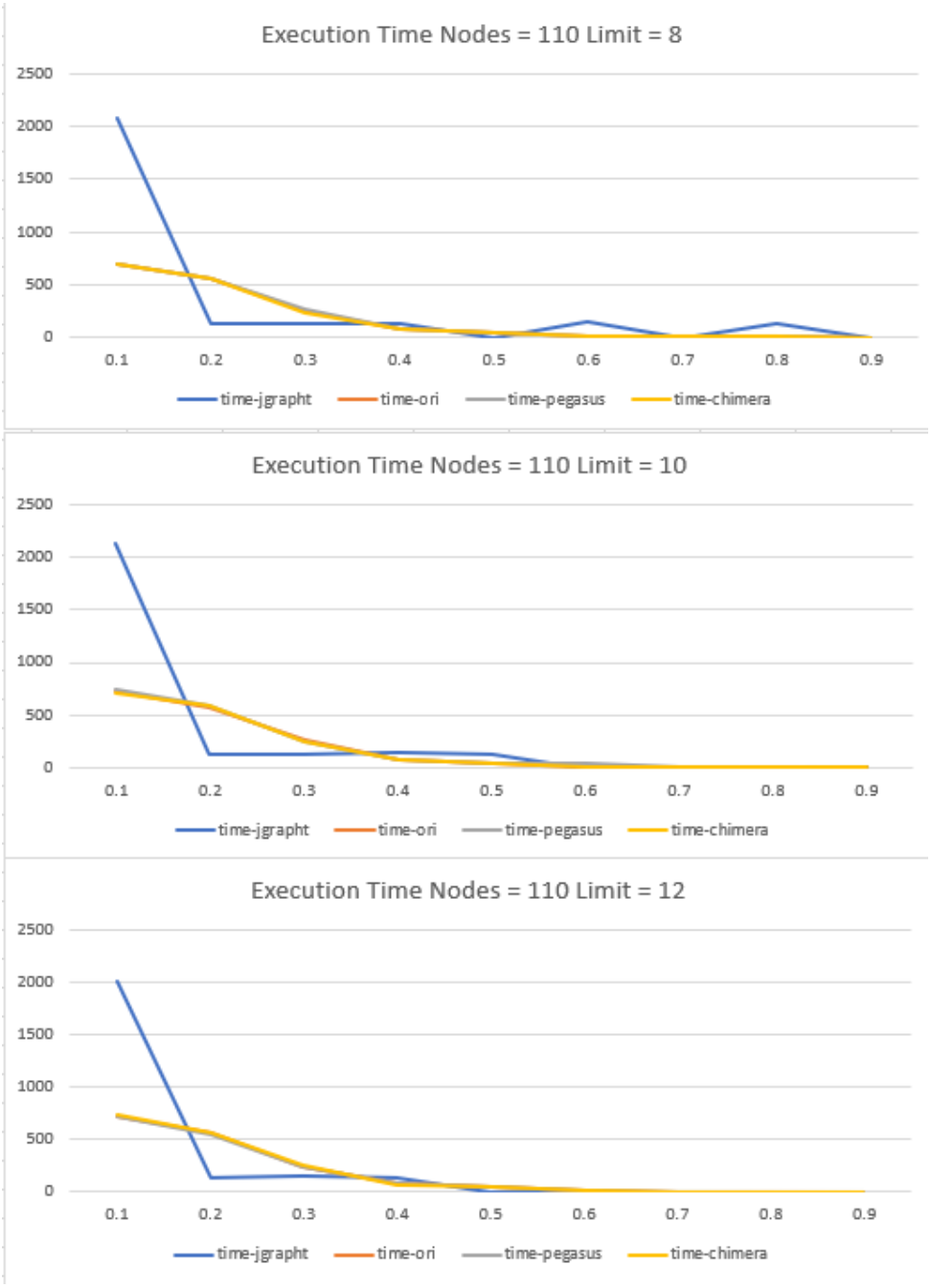

Figure 4.16 Execution time for graphs of size 110

Figure 4.16 and Figure 4.17 present notable differences in solving times between the classical algorithm and the DBR algorithm for various densities. Specifically, the classical algorithm exhibits slower solving times compared to the DBR algorithm for all densities except 0.2 and 0.3. It is worth mentioning that for the density of 0.1, the classical algorithm encountered difficulties and was unable to solve the MVC problem on the graph, resulting in an exception being thrown by the jgrapht Python library after 2250 seconds.

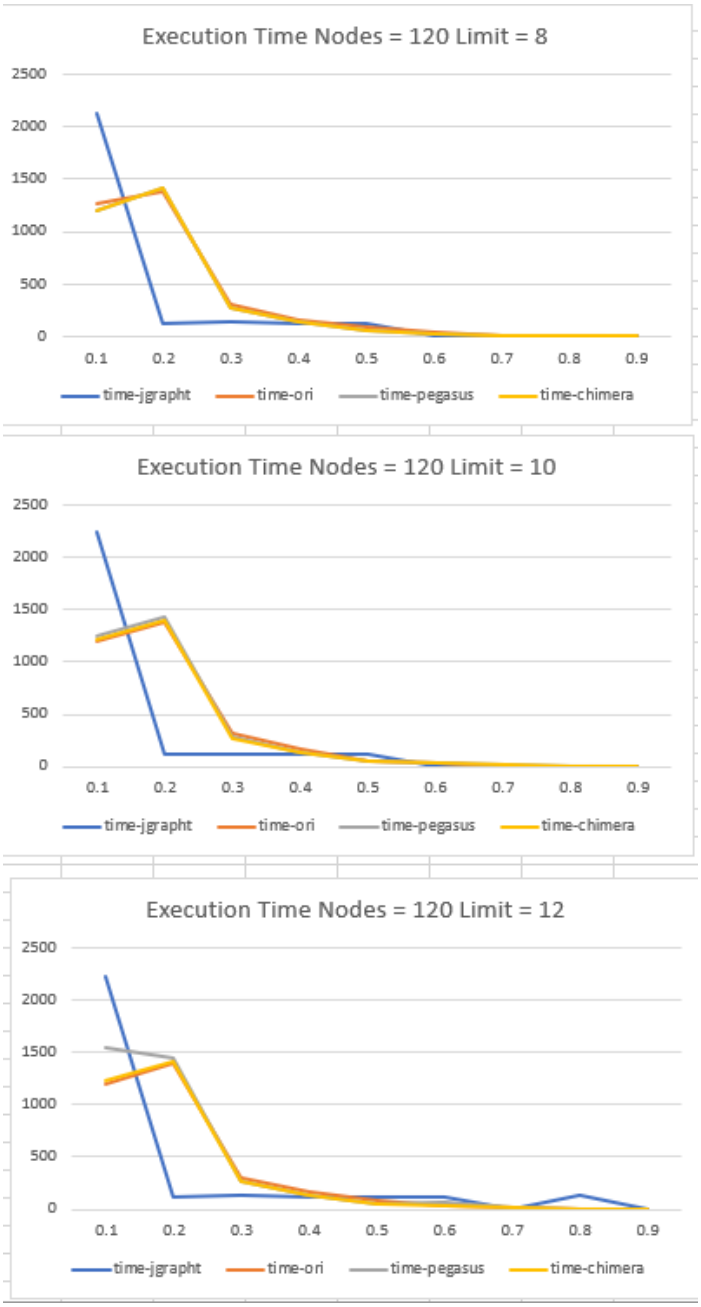

Figure 4.17 Execution time for graphs of size 120

On the other hand, the DBR algorithm successfully solved every problem, with lower densities posing greater computational demands than higher density graphs. Notably, the peak execution time was observed for graphs with a density of 0.2.

The DBR solution with the classical solver performed on par with the two approaches utilizing quantum solvers, with the communication overhead being negligible, as depicted in Figures 4.16 and 4.17.

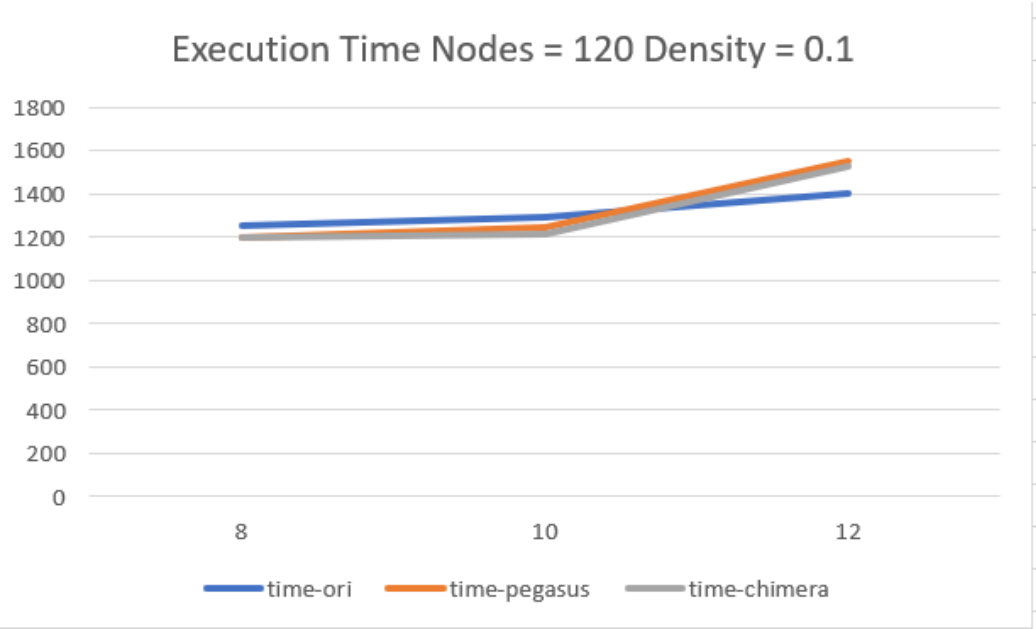

Figure 4.18 Relationship between the execution time and the decomposition limit for graphs of size 120 and density 0.1. The x-axis represents the decomposition limit, while the y-axis represents the corresponding execution time.

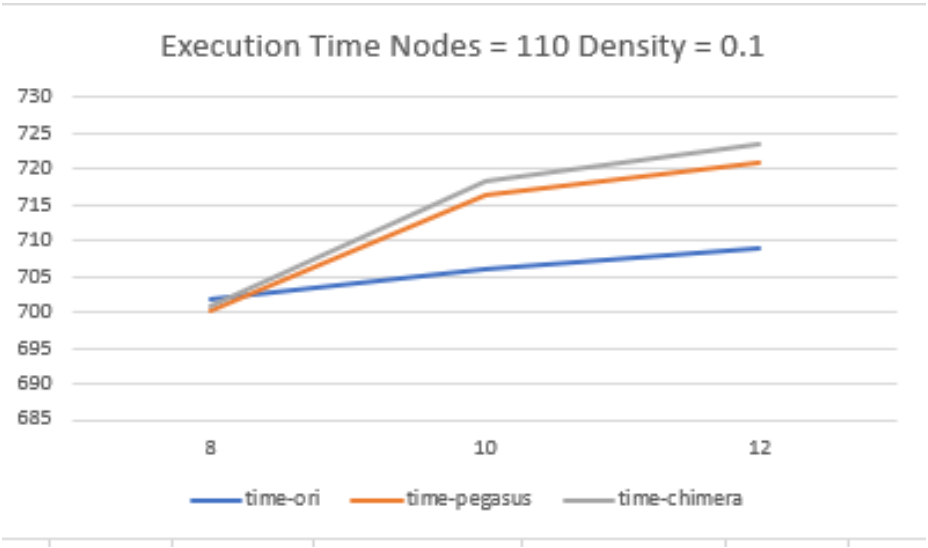

Figure 4.19 Relationship between the execution time and the decomposition limit for graphs of size 110 and density 0.1. The x-axis represents the decomposition limit, while the y-axis represents the corresponding execution time.

The results in Figure 4.18 and Figure 4.19 demonstrate an interesting trend: as the decomposition limit increases, the algorithm's performance slows down. This observation may initially seem counter-intuitive, as one might expect that a smaller limit

would result in more subproblems. However, this behaviour can be attributed to the nature of the quantum annealer and the classical solver, which requires additional time to generate solutions for larger problems. Moreover, in some cases, the quantum solvers may even produce incorrect results due to chain breaks. It is important to note that for decomposition limits equal to or greater than 15, the DBR algorithm with quantum solvers failed to return correct solutions, as the minimum vertex cover obtained from the solvers did not meet the desired criteria.

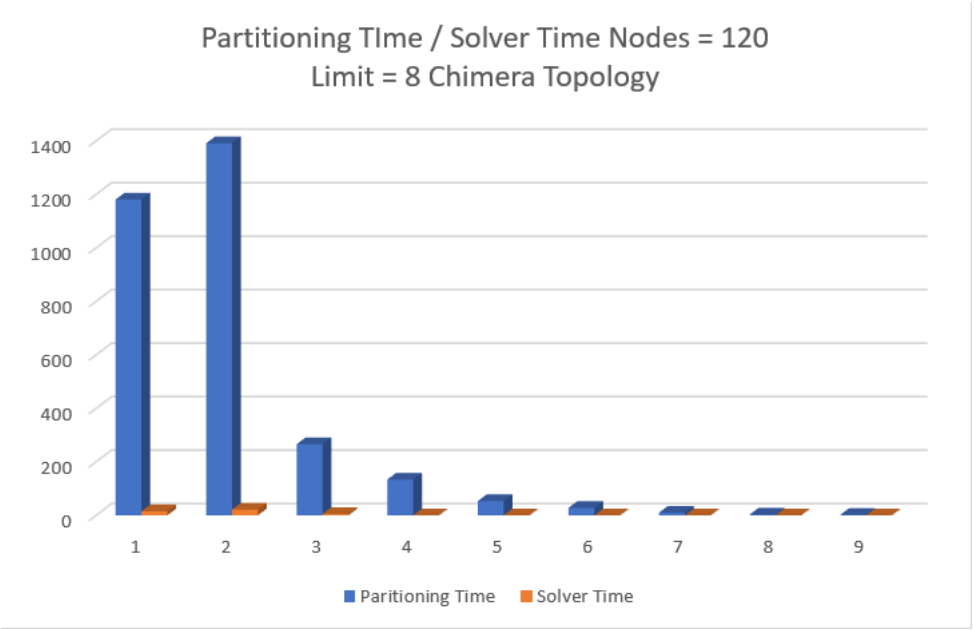

Figure 4.20 Comparative Evaluation of Partitioning Time and Solver Time for the DBR Algorithm Utilizing the Chimera Topology Solver (Nodes = 120, Limit = 8)

Figure 4.20 presents a comparison between the partitioning time and the time taken by the solver. The partitioning time consumes the majority of the total time due to the overhead associated with communication, in contrast to Figure 4.4. This shows that the solver does not have a major effect, but it is the decomposition method that provides better execution time and enables solving larger graphs compared to the classical algorithm which failed to solve the MVC with density 0.1 for graph sizes equal or larger than 110 nodes.

4.4 Discussion of Results

The results obtained from the experiments on smaller graphs did not demonstrate promising outcomes. The execution time of the DBR algorithm was slower compared to the classical algorithm. However, as we scaled up to larger graphs, we observed that the decomposition algorithm exhibited improved performance across a majority of graph densities. Additionally, it successfully solved graphs with a density of 0.1, which the classical algorithm was unable to handle.

Interestingly, in all experiments, we noticed a counter-intuitive trend: as the graph density increased, the running time decreased. This unexpected observation can be attributed to the characteristics of the decomposition algorithm, which benefits from vertices with higher degrees. In Figure 3.1, it can be observed that the larger the degree of the vertex v, the smaller the size of the graph G-. This results in decomposition trees with shorter branches for dense graphs, leading to nearly linear recursions in some cases. Conversely, for sparse graphs, the decomposition trees are more balanced and can be of exponential size.

To explore the decomposition limit as a subproblem in the DBR algorithm, we conducted additional tests. Surprisingly, the results revealed that with a larger decomposition limit, the algorithm exhibited slower performance. This may appear counter-intuitive, but it can be explained by the fact that the quantum annealer requires more time to generate a solution for larger problems or even returns incorrect results due to chain breaks.

Furthermore, the experimental evaluates suggest that using quantum solvers did not provide the expected benefits. Instead, running the quantum solver resulted in delays when executing the DBR algorithm. Although in larger graphs where most of the execution time was spent on graph decomposition, the lack of benefit was less noticeable, it was still present. We were unable to increase the reduction limit of the DBR algorithm, which was expected to be more efficient than a classical solver, because the quantum solvers returned incorrect results for graphs with more than 15 nodes. Consequently, in the comparison between solvers in the base cases, the quantum solvers did not show any advantages, as the classical solver performed faster.

Finally, we aimed to investigate whether one quantum computer topology (Pegasus or Chimera) was more beneficial than the other. However, the results indicated that the choice of topology did not significantly impact the algorithm's performance.

Altogether, the findings suggest that the DBR algorithm's advantages are more pronounced for larger graphs, showcasing improved execution time and the ability to handle challenging instances that the classical algorithm struggles with. The counter-intuitive trends observed highlight the intricacies of the decomposition algorithm and the interplay between graph density, vertex degrees, and the resulting decomposition trees.

Chapter 5

Conclusion

5.1 Conclusion

The main objective of this thesis was to implement a decomposition-based algorithm for solving the Minimum Vertex Cover (MVC) problem and harness the benefits of quantum computers to overcome the limitations imposed by a small number of qubits. The goal was to explore the potential of quantum computing in improving the efficiency and accuracy of solving combinatorial optimization problems.

The research methodology employed in this study involved the implementation of the Decomposition-Based Reduction (DBR) algorithm. The DBR algorithm was implemented using three different solvers: a classical solver (NetworkX), a quantum solver utilizing the Pegasus topology quantum computer, and another quantum solver utilizing the Chimera topology quantum computer. The performance of the DBR algorithm was compared with a classical algorithm (jgrapht) commonly used for solving the MVC problem.

The findings of this study indicate that the performance of the DBR algorithm is influenced by the size of the graph. For smaller graphs, the DBR algorithm initially appeared slower compared to the classical algorithm, and the benefits of quantum phenomena were not prominently observed. However, for larger graphs with more than 110 nodes, the DBR algorithm demonstrated faster execution times for the majority of graph densities. Furthermore, the DBR algorithm successfully solved graphs with a density of 0.1, which the classical algorithm was unable to solve. The comparison between the DBR algorithm and the classical algorithm has provided insights into the performance differences and advantages of utilizing decomposition algorithms. The results demonstrate that the DBR algorithm, especially when applied to larger graphs,

offers improved efficiency and the ability to solve challenging instances that the classical algorithm struggles with.

However, we did not observe any of the quantum benefits we were expecting. Running the quantum solver only added a delay in executing the DBR algorithm. In larger graphs where the majority of the execution time was spent on decomposing the graph, this lack of benefit was not apparent, but it was still present. We were unable to run the DBR algorithm with a higher reduction limit, which we expect would be more efficient than a classical solver, due to the limitation that the results returned by the quantum solvers were incorrect for graphs with more than 15 nodes. Therefore, in the base cases where the solvers were compared, the quantum solvers showed no benefit as the classical solver was faster.

It is important to acknowledge the limitations and constraints that may have influenced the outcomes of this study. One significant limitation is the size of the qubits in the quantum solvers, which proved to be limiting when dealing with larger problem instances. The results indicate that the quantum solvers faced difficulties returning correct solutions due to chain breaks, particularly for larger reduction limits. Additionally, other factors such as computational resources and solver-specific parameters may have influenced the validity and generalizability of the results. These limitations should be taken into consideration when interpreting the findings and considering the practical implications of the research.

The findings have demonstrated that the DBR algorithm, when utilized with different solvers including classical and quantum approaches, exhibits varying performance characteristics. While the DBR algorithm initially appeared slower for smaller graphs compared to the classical algorithm, it showcased significant improvements for larger graphs.

The practical implications of this research lie in the potential application of the DBR algorithm for solving the MVC problem in real-world scenarios. The algorithm's ability to leverage quantum computing capabilities, despite the limitations of qubit size, opens up possibilities for more efficient and accurate solutions. Furthermore, the DBR algorithm's success in solving graphs with a density of 0.1, which the classical algorithm struggled with, highlights the algorithm's superiority in certain scenarios. These findings contribute to the advancement of algorithms for tackling combinatorial optimization problems and provide valuable insights into the benefits and challenges of quantum computing in solving such problems.

Theoretical implications of this research are rooted in the exploration of decomposition-based approaches for combinatorial optimization problems. The DBR algorithm represents a novel and effective strategy for breaking down the MVC problem into subproblems, allowing for more efficient computation. By comparing different solvers, including classical and quantum alternatives, this study contributes to the understanding of the strengths and limitations of various solution approaches. The findings expand our knowledge of decomposition algorithms and shed light on the trade-offs between classical and quantum solvers in solving combinatorial optimization problems.

Throughout this research, several challenges and lessons were encountered. The limitations imposed by the size of qubits in the quantum solvers became apparent when dealing with larger decomposition limits, leading to incorrect results due to chain breaks. This highlights the importance of considering hardware constraints when developing quantum algorithms. Additionally, the implementation and comparison of multiple solvers required careful consideration of various factors such as computational resources, solver-specific parameters, and compatibility with the problem formulation. These challenges served as learning experiences, guiding the refinement of the research approach and providing insights for future investigations.

In conclusion, this thesis has advanced our understanding of the DBR algorithm and its utilization in solving the MVC problem. The findings pave the way for further research and exploration of quantum computing approaches for solving combinatorial optimization problems.

5.2 Future Work

While this study has provided valuable insights into the application of the Decomposition-Bounds-Reduction (DBR) algorithm for solving the Minimum Vertex Cover (MVC) problem using quantum computing, there are several avenues for future research and improvement. The following suggestions outline potential directions for further investigation:

Exploring Larger Qubit Architectures: One of the key limitations encountered in this study was the restricted size of the qubit architectures in the quantum solvers. Future work could focus on exploring larger qubit architectures, such as those proposed in emerging quantum technologies, to handle more complex problem instances.

Investigating the effectiveness of these architectures in improving the accuracy and scalability of the DBR algorithm would be a fruitful area of research.

Optimizing Quantum Solver Parameters: To enhance the performance of the DBR algorithm, it is essential to delve deeper into the optimization of quantum solver parameters. Fine-tuning parameters such as the annealing schedule, chain strength, and embedding techniques could potentially mitigate chain breaks and improve the reliability and efficiency of the quantum solvers. In our research, we utilized autoembedding when solving the Minimum Vertex Cover (MVC) problem on a quantum annealer. However, it is worth investigating whether using a specific embedding tailored to the problem instance could yield even greater benefits. By exploring different embedding techniques and selecting appropriate embeddings for specific problem classes, we can potentially improve the success rate and solution quality of the quantum solvers. Further investigations into these parameter optimizations, including exploring specific embeddings, are warranted in order to unlock the full potential of quantum solvers within the DBR algorithm.

Developing Hybrid Classical-Quantum Approaches: Hybrid classical-quantum algorithms have shown promise in combining the strengths of classical and quantum computing. Future work could explore the development of hybrid approaches that leverage the strengths of classical solvers and quantum solvers. This could involve integrating classical preprocessing techniques, such as reduction methods or heuristics, with the DBR algorithm to improve its overall performance and scalability.

Benchmarking and Comparative Studies: Conducting extensive benchmarking studies to evaluate the performance of the DBR algorithm against other state-of-the-art classical and quantum algorithms would provide a comprehensive understanding of its strengths and weaknesses. Comparative studies can help identify scenarios where the DBR algorithm outperforms existing methods and determine its applicability to a wider range of problem instances.

Real-world Application and Problem Variants: Applying the DBR algorithm to real-world datasets and exploring its effectiveness in solving practical optimization problems would be a valuable area of future work. Additionally, investigating the adaptability of the DBR algorithm to problem variants related to vertex cover, such as weighted vertex cover or dynamic vertex cover, could further expand its practical utility and contribute to the field of combinatorial optimization.

By focusing on these avenues for future work, researchers can build upon the findings of this thesis and contribute to the advancement of quantum computing techniques for solving the Minimum Vertex Cover problem, as well as other combinatorial optimization problems.

References

[1] D. Bouwmeester and A. Zeilinger. The Physics of Quantum Information: Basic Concepts, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. doi:10.1007/978-3-662-04209-0_1.

[2] Greg Byrd Frank Mueller. Programming Quantum Computers: A Primer. NC State University, Electricaland Computer Engineering, 2020.

[3] Thomas Young. The bakerian lecture: Experiments and calculations relative to physical optics. Philosophical Transactions of the Royal Society of London, 94:1–16, 1804. URL: http://www.jstor.org/stable/107135.

[4] Franklin, Allan and Slobodan Perovic, "Experiment in Physics", *The Stanford Encyclopedia of Philosophy* (Summer 2023 Edition), Edward N. Zalta & Uri Nodelman (eds.), forthcoming URL: https://plato.stanford.edu/archives/sum2023/entries/physics-experiment/.

[5] Stern, August (1994). The quantum brain: theory and implications. New York: North-Holland/Elsevier.

[6] H. D. Zeh www.zeh-hd.de Eur. Phys. J. H, Feynman's Interpretation of Quantum Theory1, doi:10.1140/epjh/e2011-10035-2 - arxiv:0804.3348v6

[7] Folkerts, N., Putzer, J., Villalba-Chávez, S., & Müller, C. (2023). Electron-positron pair creation in the superposition of two oscillating electric field pulses with largely different frequency, duration and relative positioning. arXiv preprint arXiv:2303.02350 [physics.atom-ph]

[8] Jennewein, T. et al. Quantum cryptography with entangled photons. Phys. Rev. Lett. 84, 4729–4732 (2000).

[9] J. Am. Chem. Soc. 2017, 139, 43, 15276–15283, Tunneling Control of Chemical Reactions: The Third Reactivity Paradigm (2017), doi: https://doi.org/10.1021/jacs.7b06035

[10] Philipp Hauke, Helmut G Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D Oliver. Perspectives of quantum annealing: methods and implementations. Reports on Progress in Physics, 83(5):054401, may 2020. doi:10.1088/1361-6633/ab85b8

[11] Diego de Falco and Dario Tamascelli. An introduction to quantum annealing. RAIRO - Theoretical Informatics and Applications, 45(1):99–116, jan 2011. URL: https://doi.org/10.1051%2Fita%2F2011013, doi:10.1051/ita/2011013.

[12] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. Finding maximum cliques on the d-wave quantum annealer. Journal of Signal Processing Systems, 91(3):363–377, 2019. doi:10.1007/s11265-018-1357-8.

[13] Erio Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. Journal of Physics A: Mathematical and Theoretical, 41:209801, 05 2008. doi:10.1088/1751-8113/41/20/209801.

[14] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. Journal of Mathematical Physics, 49(12):125210, dec 2008. URL: https://doi.org/10.1063%2F1.2995837, doi:10.1063/1.2995837

[15] D-Wave Systems. Physics of Quantum Annealing - Hamiltonian and Eigenspectrum. 2022. URL: https://www.youtube.com/watch?v=tnikftltqE0&list=PLPvKnT7dgEsvVQwGgrlUVXBa2J6PAW8a4&index=3.

[16] Andrew M. Childs, Edward Farhi, and John Preskill. Robustness of adiabatic quantum computation. Physical Review A, 65(1), 2001. URL: https://doi.org/10.1103%2Fphysreva.65.012322, doi: 10.1103/physreva.65.012322.

[17] Hidetoshi Nishimori, Junichi Tsuda, and Sergey Knysh. Comparative study of the performance of quantum annealing and simulated annealing. Physical Review E, 91(1), 2015. URL: https://doi.org/10.1103%2Fphysreve.91.012104, doi:10.1103/physreve.91.012104.

[18] Andrew D. King and Catherine C. McGeoch. Algorithm engineering for a quantum annealing platform. ArXiv, abs/1410.2628, 2014.

[19] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. arXiv: Quantum Physics, 2000.

[20] Davide Venturelli, Salvatore Mandra , Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. Physical Review X, 5(3), sep 2015. URL: https://doi.org/10.1103%2Fphysrevx.5.031040, doi:10.1103/physrevx.5.031040.

[21] Diego de Falco and Dario Tamascelli. An introduction to quantum annealing. RAIRO - Theoretical Informatics and Applications, 45(1):99–116, jan 2011. URL: https://doi.org/10.1051%2Fita%2F2011013, doi:10.1051/ita/2011013

[22] D-Wave Systems. Dwave. https://www.dwavesys.com/.

[23] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. Finding maximum cliques on the d-wave quantum annealer. Journal of Signal Processing Systems, 91(3):363–377, 2019. doi:10.1007/s11265-018-1357-8.

[24] N G Dickson, M W Johnson, M H Amin, R. Harris, F. Altomare, A J Berkley, P. Bunyk, J. Cai, E M Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J P Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M C Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A B Wilson, Z. Merali, and G. Rose. Thermally assisted quantum annealing of a 16-qubit problem. Nature Communications, 4(1):1903, 2013. doi:10.1038/ncomms2920

[25] R. Harris, M. William Johnson, Trevor Lanting, Andrew J. Berkley, J. Johansson, Paul I. Bunyk, Elena Tolkacheva, Eric Ladizinsky, Nicolas Ladizinsky, Travis Oh, Florentin Cioata, I. G. Perminov, P. Spear, C. Enderud, Chris Rich, Sergey Uchaikin, Murray C. Thom, E. M. Chapple, J. Wang, Brendan J Wilson, Mohammad H. S. Amin, Neil G. Dickson, Kamran Karimi, Bill Macready, C. J. S. Truncik, and Geordie Rose. Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. Physical Review B, 82, 2010.

[26] Embedding algorithms for quantum annealers with chimera and pegasus connection topologies. In Ponnuswamy Sadayappan, Bradford L. Chamberlain, Guido Juckeland, and Hatem Ltaief, editors, High Performance Computing, pages 187–206, Cham, 2020. Springer International Publishing.

[27] Zhang, H., Wang, L., & Jiang, L. (2010). A Hybrid Graph Representation for Recursive Backtracking Algorithms. In Frontiers in Algorithmics, 4th International Workshop, FAW 2010, Wuhan, China, August 11-13, 2010. Proceedings (pp. 160-171). DOI: 10.1007/978-3-642-14553-7_15.

[28] Pemmaraju, S. and Skiena, S. "Minimum Vertex Cover." §7.5.2 in Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Cambridge, England: Cambridge University Press, p. 317, 2003.

[29] Xie Q., Li Y., Hu S., Zhu Y., Wang H. Two heuristic algorithms for the minimum weighted connected vertex cover problem under greedy strategy IEEE Access, 10 (2022), Article 116467-116472

[30] Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. Annals of Discrete Mathematics 25, 27–46 (1985)

[31] N. Garg, Saving an epsilon: a 2-approximation for the k-mst problem in graphs Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC'05, ACM, New York, NY, USA (2005), pp. 396-402, 10.1145/1060590.1060650
[32] Chiang, Hua-Pei, Yao-Hsin Chou, Chia-Hui Chiu, Sy-Yen Kuo, and Yueh-Min Huang. "A quantum-inspired Tabu search algorithm for solving combinatorial optimization problems." Soft Computing 18 (2013): 1771-1781. doi: 10.1007/s00500-013-1203-7.
[33] Saad, H.M.H., Chakrabortty, R.K., Elsayed, S., & Ryan, M.J. (2021). Quantum-Inspired Genetic Algorithm for Resource-Constrained Project-Scheduling. IEEE Access, vol. 9, pp. 35082-35093, doi: 10.1109/ACCESS.2021.3062790.
[34] Michail, D., Kinable, J., Naveh, B., & Sichi, J.V. (2020). JGraphT -- A Java library for graph data structures and algorithms. arXiv preprint arXiv:1904.08355 [cs.DS].
[35] NetworkX. https://networkx.org/documentation/latest/
[36] The double slit experiment. https://en.wikipedia.org/wiki/Double-slit_experiment.
[37] Pelofske, E., Hahn, G., & Djidjev, H. N. (2019). Solving large minimum vertex cover problems on a quantum annealer. In F. Palumbo, M. Becchi, M. Schulz, & K. Sato (Eds.), Proceedings of the 16th ACM International Conference on Computing Frontiers, CF 2019, Alghero, Italy, April 30 - May 2, 2019 (pp. 76-84). ACM. doi:10.1145/3310273.3321562
[38] Hagberg, A., Schult, D., and Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. Proceedings of SciPy2008, pages 11–15.
[39] Bar-Yehuda, R. and Even, S. (1985). A local-ratio theorem for approximating the weighted vertex cover problem. Annals of Discrete Mathematics, 25:27–46
[40] Budinich, M. (2003). Exact bounds on the order of the maximum clique of a graph. Discrete Applied Mathematics, 127(3):535–543.
[41] Willis, W. (2011). Bounds for the independence number of a graph. Master's thesis, Virginia Commonwealth University. https://scholarscompass.vcu.edu/etd/2575.
[42] Rother, C., Kolmogorov, V., Lempitsky, V., and Szummer, M. (2007). Optimizing binary mrfs via extended roof duality.
[43] Akiba, T. and Iwata, Y. (2015a). Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover. 2015 Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments (ALENEX), pages 1–12.
[44] Salvador E. Venegas-Andraca, William Cruz-Santos, Catherine McGeoch, and Marco Lanzagorta. A cross-disciplinary introduction to quantum annealing-based

algorithms. Contemporary Physics, 59(2):174–197, 2018. doi: https://doi.org/10.1080/00107514.2018.1450720, doi:10.1080/00107514. 2018.1450720.

[45] Hagberg, A., Schult, D., and Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. Proceedings of SciPy2008, pages 11–15.

[46] "A breakthrough 'speed test' in quantum tunnelling" The Conversation, 19-Mar-2019. [Online]. https://theconversation.com/we-did-a-breakthrough-speed-test-in-quantum-tunnelling-and-heres-why-thats-exciting-113761.

[47] Tracking chaperone-mediated folding using force spectroscopy. ttps://www.researchgate.net/figure/Illustration-of-a-protein-folding-energy-landscape A-Two-dimensional-representation_fig1_299537797, 2022.

[48] D-WAVE Systems. How the Quantum Annealing Process Works. 2022. URL: https://www.youtube.com/watch?v=UV_RlCAc5Zs&list=PLPvKnT7dgEsvVQwGgrlUVXBa2J6PAW8a4&index=2.

[49] D-Wave Systems. D-Wave QPU Architecture: Topologies. https://docs.dwavesys.com/docs/latest/c_gs_4.html

[50] Vertex Cover. https://en.wikipedia.org/wiki/Vertex_cover

[51] Lima, P. T., dos Santos, V. F., Sau, I., Souza, U. S., & Tale, P. (2022). Reducing the Vertex Cover Number via Edge Contractions. arXiv preprint arXiv:2202.03322 [cs.DS]

[52] Wang, L., Li, C.-M., Zhou, J., Jin, B., & Yin, M. (2019). An Exact Algorithm for Minimum Weight Vertex Cover Problem in Large Graphs. arXiv preprint arXiv:1903.05948 [cs.DS].

Appendix A

```
import dimod
import networkx as nx
from dimod import ConstrainedQuadraticModel
from dimod import Binary
from dwave.system import DWaveSampler,AutoEmbeddingComposite
from itertools import combinations
#import dwave.inspector

def minimum_vertex_cover_chimera(G):

    # x[i] = 1 if node[i] is in minimum vertex cover
    # x[i] = 0 otherwise
    x = {n: Binary(n) for n in G.nodes}

    cqm = ConstrainedQuadraticModel()

    # Set the objective to minimize the size of the cover
    cqm.set_objective(sum(x[i] for i in G.nodes))

    # Add constraint that each edge has at least a node in the cover
    for (i,j) in G.edges:
        cqm.add_constraint(x[i] + x[j] >= 1, label = f'edge{i}_{j}')

    bqm, invert = dimod.cqm_to_bqm(cqm)

    #sampler = DWaveSampler()
    sampler = DWaveSampler(solver={'topology__type': 'chimera'})
    embedding_sampler = AutoEmbeddingComposite(sampler)
    sampleset = embedding_sampler.sample(bqm,num_reads = 100)

    #dwave.inspector.show(sampleset)


    for smpl, energy in sampleset.data(['sample','energy']):
        minimum_vertex_cover = [n for n, value in smpl.items() if value
== 1 and not str(n).startswith('slack')]
        mvc_size = len(minimum_vertex_cover)
        break

    print("=== DWave done ===")
    return minimum_vertex_cover
```