

Application of Quantum Annealing in Factory Layout Optimization

A Thesis

submitted to

Indian Institute of Science Education and Research Pune

in partial fulfillment of the requirements for the

BS-MS Dual Degree Programme

by

Raghav Kaul



Indian Institute of Science Education and Research Pune

Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

April, 2024

Supervisor: Dr. Aniruddha Pant

© Raghav Kaul 2024

All rights reserved

Certificate

This is to certify that this dissertation entitled Application of Quantum Annealing in Factory Layout Optimization towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Raghav Kaul at Indian Institute of Science Education and Research under the supervision of Dr. Aniruddha Pant , Department of Physics, during the academic year 2023-2024.



Dr. Aniruddha Pant

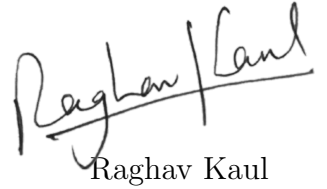
Committee:

Dr. Aniruddha Pant

Dr. MS Santhanam

Declaration

I hereby declare that the matter embodied in the report entitled Application of Quantum Annealing in Factory Layout Optimization are the results of the work carried out by me at the Department of Physics, Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Aniruddha Pant , and the same has not been submitted elsewhere for any other degree.

A handwritten signature in black ink, appearing to read 'Raghav Kaul', written over a horizontal line.

Raghav Kaul

Acknowledgments

I want to acknowledge Dr. Aniruddha Pant and his wife, Mrs Kanchan Pant, as without their supervision and manufacturing insights, this project would not be possible. I would also like to thank Dr. M.S Santhanam for his generosity and for extending a helping hand by agreeing to be the expert for this thesis while already being swamped with multiple other students.

I would also like to thank my parents for their unwavering support, which allowed me to focus completely on my work. I admire my parents for their selflessness and dedication to their own work and I strive to achieve the same excellence throughout my career. Both of them have put in immense amounts of hard work for me to be able to pursue my passions, and I wish to one day have the same level of respect and appreciation from my colleagues as my parents have built in their own respective fields.

I would also like to thank the friends I have made in IISER, they have been a vital part of my life until this point and have been the key to my happiest memories in IISER. Finally, I would like to thank IISER Pune for providing me with the opportunity and flexibility to approach physics and research in my own way.

Abstract

Quantum computing offers a revolutionary approach to computation by leveraging quantum bits or qubits. Quantum computers can perform certain calculations exponentially faster than classical computers. In the context of factory layout planning, quantum annealing, a specific quantum computing approach, can efficiently optimize complex problems by exploring multiple solutions simultaneously and quickly finding the optimal configuration for factory layouts. By harnessing quantum annealing's ability to navigate vast solution spaces rapidly, factory planners can enhance efficiency, reduce costs, and improve overall productivity in designing optimal factory layouts. This thesis aims to prove that not only factory layout planning is achievable using quantum annealing, but it is also many folds faster than classical Monte-Carlo simulation techniques. This thesis proves that quantum annealing can be used to generate multiple optimal or near-optimal layouts which can help in the early stages of factory layout planning. By utilizing D'Wave's QUBO formulation, a factory can be divided into a number of positions and, along with the functional units, can be described as a graph, with nodes mapping to qubits and the edges mapping to couplers between the qubits in D'Wave's QPUs. By qualitative and quantitative analysis, architects and engineers can create multiple optimal layouts which can be used to streamline the creation of digital doubles of the factory. The model created is flexible, as it takes no specialized data and just requires the rate of transportation of product between functional units in a pipeline, and the distance between each position with every other position. This makes this method be applicable to any sort of factory, as long as it follows pipeline based production style.

Contents

| | |
|---|-----------|
| Abstract | ix |
| 1 Introduction | 1 |
| 1.1 A Brief Overview of Quantum Mechanics | 3 |
| 1.2 Qubits | 5 |
| 1.3 Bits vs Qubits | 8 |
| 1.4 Factory Layout Planning | 10 |
| 1.5 Scope | 13 |
| 2 Theory | 15 |
| 2.1 Two types of Quantum Computing | 15 |
| 2.2 Annealing | 16 |
| 2.3 Quantum Annealing by D’Wave | 18 |
| 2.4 Model Formulation | 23 |
| 3 Results | 31 |
| 3.1 Simulated Annealing | 33 |
| 3.2 Quantum Annealing | 44 |

| | | |
|----------|--|-----------|
| 4 | Discussion | 49 |
| 4.1 | Results | 49 |
| 4.2 | Why Use a Hybrid Solver | 50 |
| 5 | Conclusion and Future Prospects | 51 |
| 5.1 | Future Improvements | 51 |
| 5.2 | Conclusion | 52 |

Chapter 1

Introduction

Quantum computing is pushing the frontier in technological advancements, leveraging the principles of quantum mechanics to tackle problems beyond classical computers’ capabilities. The concept was introduced by Richard Feynman (and independently by Yuri Manin) in his paper *Simulating Physics with computers, 1982*^[3], who highlighted classical systems’ limitations in simulating quantum behaviour. A typical example of showcasing this difference, aptly explained by IBM^[17]; is that a classical computer might be great at rigorous tasks like sorting through a vast database of molecules and their features. However, when it comes to simulating the behaviour of those molecules, it falls short.

Determining how a molecule behaves entails synthesization, real-world experimentation, and documentation. If one wishes to tweak the molecule, one must go through the expensive task of re-synthesizing, experimenting, and re-documenting the results, especially in industries like drug^[2] and semiconductor design^[4]. A classical supercomputer can try to simulate every permutation of every part of a molecule using brute force, but, except for the most elementary particles, no current computer has a large enough working memory or processing power to handle all the possibilities by utilizing any known methods.

These complex problems are where a quantum computer shines. Quantum computers take advantage of the inherent probabilistic nature of quantum mechanics by creating multi-dimensional computing spaces. Any tweak one might want to make on a molecule will already be encoded within the simulation, allowing simultaneous computation of all permutations of the molecule.

Quantum computing uses qubits, the quantum equivalent of classical bits, enabling new ways to process information. Key features like superposition and entanglement distinguish quantum systems from classical ones. Superposition allows qubits to represent multiple configurations simultaneously, enhancing parallel computational power. Entanglement correlates the behaviour of qubits, enabling interconnected operations that lead to advanced computations.

On a separate note, manufacturing is pivotal in today's global economy, serving as a linchpin for economic development, innovation, and societal progress. Its significance emanates from various interrelated dimensions, each contributing to its importance. It serves as a fundamental driver of economic growth and productivity. Creating goods and services generates employment opportunities across diverse skill levels, fostering income generation and reducing unemployment rates. Its activities lead to the development of other sectors, such as transportation, logistics, and services, which can amplify economic activity and enhance overall prosperity.

Quantum computing can revolutionize manufacturing by addressing complex optimization challenges in factory layout design, supply chain management, quality control, failure predictions, and materials science. Its unparalleled computational power allows for efficient analysis of vast datasets and also provides robust cybersecurity measures and safeguarding sensitive manufacturing data in an increasingly digitized environment.

The motivation for this thesis arose from the already extensive study of simulating real-world physical systems using computers. These digital simulations of physical systems are cost-effective, as they eliminate the need to create an experimental setup to study them, which is often time-consuming. It offers granular control and repeatability, allowing researchers to systematically study the problem and verify and generate trends and patterns. Due to our giant leaps in computational power, simulations have become accessible to a broad range of students and researchers. It has enabled us to simulate larger and larger systems over longer time scales, which is usually impossible or impractical to investigate experimentally. Quantum computing is the next leap that will open up a whole new dimension and allow us to simulate quantum mechanical systems, as well as high-complexity commercial problems with speed, accuracy and precision, which even the most advanced supercomputers are nowhere near achieving.

Factory layout planning (FLP) is one of the first steps in designing a factory or a produc-

tion pipeline. It is the systematic planning of the layout of machines, facilities, stations, etc., to minimize internal transportation of materials, manage storage and manpower, increase overheads, and increase overall profits by minimizing production costs. Even considering all our species' innovations since the Industrial Revolution, this process is still long and arduous, requiring significant amounts of money, data, manpower, and computation power.

This thesis aims to prove that when used properly, quantum computing can be an effective tool to help generate optimized layouts quickly and efficiently, reducing the amount of preliminary manual work required.

1.1 A Brief Overview of Quantum Mechanics

Quantum states, represented by Dirac's ket $|\psi\rangle$, in time according to the Schrödinger equation:

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = \hat{H}(t) |\psi(t)\rangle \quad (1.1)$$

And we know that the time evolution of the Schrödinger equation is described by unitary operators. So for a unitary operator $\hat{U}(t)$, replacing $|\psi\rangle$ with $\hat{U}(t)|\psi\rangle$ in Eq.1.1, we get

$$i\hbar \frac{d}{dt} \hat{U}(t) = \hat{H}(t) \hat{U}(t) \quad (1.2)$$

Where $|\psi\rangle$ is the quantum state or wavefunction and \hat{H} is the Hamiltonian

This has been extensively tested and verified by experiments and is probabilistic in nature. This implies that the value, on measurement, of a state is not deterministic. Even after repeating an identical evolution, measurement will not always yield the same value, even though the wavefunction would be the same.

Due to the linearity of the Schrödinger equation, quantum states also evolve linearly, i.e., linear combinations of solutions are also solutions. This is what we call *Superposition of States*. Upon measurement, the state can collapse into any of the eigenstates, each with a

probability equal to the square of the probability amplitude. In general,

$$\psi = \alpha_1\psi_1 + \alpha_2\psi_2 + \dots + \alpha_n\psi_n \quad (1.3)$$

is a solution, where $\psi_1, \psi_2, \dots, \psi_n$ are solutions of the equation, and $\alpha_1^2 + \alpha_2^2 + \dots + \alpha_n^2 = 1$

This fundamental property of quantum mechanics is the basis of all quantum computers.

1.1.1 A Conceptual Explanation of Superposition

Before going into specific details on quantum superposition, it is helpful to explain how the term “superposition” is used in different contexts in both classical and quantum physics^[8].

Classical Superposition

In classical mechanics, the concept of superposition is used to describe when two physical quantities are added together to make another third physical quantity that is entirely different from the original two. An example of this can be seen while studying waves. Two pulses travelling on a string which pass through each other will interfere following the principle shown in Fig.1.1, waves in the same phase interfere constructively, and waves in the opposite phase interfere destructively. Noise-cancelling headphones work on this principle, recording the surrounding sound and playing the same wave but with the opposite polarity into our ears, causing destructive interference and negating noise. Another example is the vector sum of two forces acting on a body.

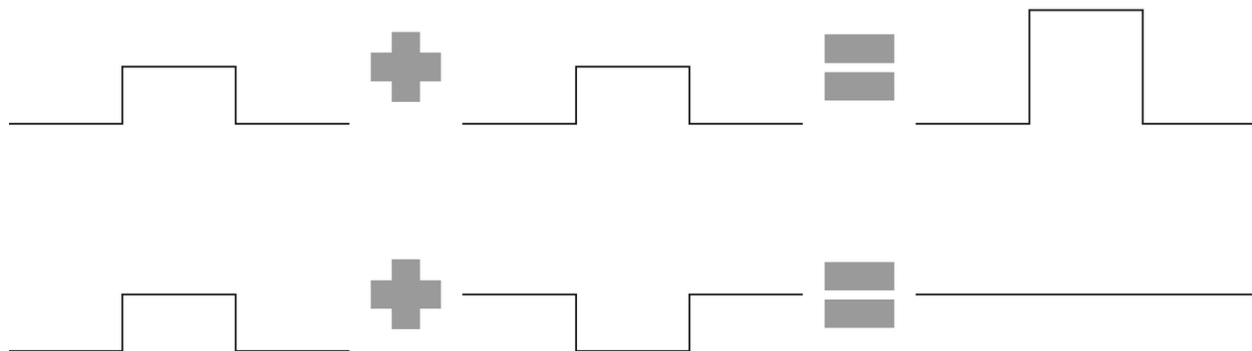


Figure 1.1: Interference of waves

Quantum Superposition

Quantum superposition occurs when non-classical particles, which have a high wave-particle duality (nuclei, protons, electrons, photons, etc.), interact with one another. The most famous thought experiment that makes quantum superposition easier to understand is Schrodinger's Cat. This experiment presents a situation with a cat placed in a closed box along with a radioactive nucleus that will emit a deadly amount of radiation once it decays. There is no way to predict when that decay happens, as it is a spontaneous process, so there is no way to know whether the cat is alive or dead unless we open the box to check. It can be said that the cat is both alive and dead with some non-zero probability. In other words, the cat is in a superposition state until we open the box and measure it. Upon measurement, the cat is either alive or dead, and the superposition state has collapsed into a definite, non-superposition eigenstate.

As stated before, a linear combination of solutions of the Schrödinger equation are also solutions. So, the system in a superposition state, when measured, can decay into any of the individual solutions. This loss of superposition is called *decoherence* and is one of the biggest hurdles to overcome to create commercially viable quantum computers.

1.2 Qubits

The smallest unit of information on a classical computer is called a bit, and can have the value 0 or 1. Quantum computers, similarly, also have a basic unit of information called quantum bits, or "qubits". Qubits represent the eigenstate of the wavefunction in the Schrödinger equation. Like its classical counterpart, qubits can be in an "off" state (denoted by $|0\rangle$), and an "on" state (denoted by $|1\rangle$). But that is where the similarities end. If we recall Eq.1.3, it is evident that any linear combination of $|0\rangle$ and $|1\rangle$ is also a valid description of the wave function. So, the general state of a qubit is of the form

$$\alpha_0|0\rangle + \alpha_1|1\rangle \tag{1.4}$$

where α_0 and α_1 are complex numbers and are subject to the normalization constraint:

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \quad (1.5)$$

and $|0\rangle$ and $|1\rangle$ are orthonormal basis vectors of the 2 dimensional Hilbert space of the qubit.

1.2.1 Many-Qubit Systems

The state of a system with two or more qubits can be represented as a tensor product of all the eigenstates of an individual qubit. For example, a 2-qubit system can have the states as described by the tensor product of the individual states of each qubit:

$$\begin{aligned} |0\rangle \otimes |0\rangle &= |00\rangle \\ |0\rangle \otimes |1\rangle &= |01\rangle \\ |1\rangle \otimes |0\rangle &= |10\rangle \\ |1\rangle \otimes |1\rangle &= |11\rangle \end{aligned} \quad (1.6)$$

An equally weighted superposition of all these states would be:

$$\Psi = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \quad (1.7)$$

Where Ψ represents the state of the system. Note how the coefficients follow the constraint in Eq.1.5. Upon measurement, the system will decay into any of the four eigenstates (as listed in Eq.1.6). Similarly, an equally weighted system with 3 qubits will be of the form:

$$\Psi = \frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle + \frac{1}{\sqrt{8}}|010\rangle + \dots + \frac{1}{\sqrt{8}}|110\rangle + \frac{1}{\sqrt{8}}|111\rangle \quad (1.8)$$

A basic example of interpreting the output of the system described by Eq.1.7 is that the probability of the 2^{nd} qubit having value 1 after measurement is $(\frac{1}{2})^2 + (\frac{1}{2})^2 = \frac{1}{2}$.

1.2.2 Quantum Entanglement

Let's assume that the state of a 2-qubit system is described as:

$$\Psi = \sqrt{\frac{3}{10}}|00\rangle + \sqrt{\frac{5}{10}}|01\rangle + \sqrt{\frac{1}{10}}|10\rangle + \sqrt{\frac{1}{10}}|11\rangle \quad (1.9)$$

Let's ask ourselves what the probability of the system to be in the state $|00\rangle$ is upon measurement. Obviously, it is $(\sqrt{\frac{3}{10}})^2 = 0.3$. But what if we ask, what is the probability of the 1st qubit being $|0\rangle$? To find that out, we can simply take both states where the 1st qubit is in the state $|0\rangle$, which there are two of because of the state of the 2nd qubit, and add their probabilities, which gives us 0.8.

While the answer of the second question might be a bit anticlimactic, the purpose of the question is to realise the fact that the 1st qubit can be $|0\rangle$ irrespective of whether the 2nd qubit is $|0\rangle$ or $|1\rangle$. Similarly in Eq.1.8, each of the three qubits can collapse to $|0\rangle$ or $|1\rangle$ independently of the other two, giving us a total of eight possible eigenstates. The values of each qubit are independent of the other qubits in the system. If we now take a similar system but described by the state

$$\Psi = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad (1.10)$$

we can see that the probability of the 1st qubit to be $|0\rangle$ is 0.5. But in this case the 2nd qubit can *only* be in the $|0\rangle$ state. Similarly if the 1st qubit is in the $|1\rangle$ state, the only possible state for the 2nd qubit is $|1\rangle$. It can be said that the two qubits are *entangled*.

In formal terms, a pair or a group of particles is entangled when the quantum state of each particle cannot be described independently of the other particle, while the entire system as a whole can be described^[9]. For example, a system of two particles is created in such a way that the total spin is zero. That is, their state can be written as:

$$\Psi = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \quad (1.11)$$

If we measure one particle and see the spin is clockwise (let's say $|1\rangle$), the other will always have the opposite spin, i.e., the measurement of one qubit will cause the other qubit to collapse automatically. In fact, any change to a qubit will *instantly* adjust the other qubit

to be consistent with the quantum mechanical rules. We might be tempted to think that an entangled qubit can "feel" that a measurement is performed and "knows" what the outcome should be, but this is not the case. This phenomenon happens without any information transfer; the particles could be billions of miles away from each other, and the entanglement would still be present.

While this can raise questions about faster-than-light information transfer, that is untrue. Suppose Alice and Bob each have a qubit from a pair of entangled qubits in the state $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$. If Alice measures her qubit and gets $|0\rangle$, Alice now knows that Bob will measure the state of his qubit to be $|0\rangle$, too. But Bob has no way of knowing the result Alice got. To Bob, measuring his qubit provides him with no information on whether Alice measured her qubit or not. It is mathematically provable that there is no test that Bob can do that will let him know the same. Essentially, Bob's measurement, to Bob, is random. The only way for Bob to know Alice's result of measurement is if Alice sends her result to Bob through a classical channel, which is limited by the speed of light. This is called the **No Communication Theorem**^[13]. The mathematical proof, however, is outside the scope of this thesis.

1.3 Bits vs Qubits

Now that we know the basics of what qubits are and how they work, let's compare them to the more familiar classical bits. This section focuses on the benefits qubits possess over classical bits.

1.3.1 Data Representation

For classical bits, we know:

- Bits have a value of either 0 or 1. Strings of bits are interpreted as binary encoding, where each bit represents a power of 2
- Bits are independent; each bit can always exist in its state without any inherent correlation with other bits

- They are deterministic; the value of a bit can be precisely determined at any time. They indefinitely exist in that state unless explicitly changed.

And if we compare to the properties of qubits that we have discussed till now,

- Qubits can have a value of $|0\rangle$ or $|1\rangle$ and any linear combination of the two basis vectors.
- Qubits can be independent, but they can also be entangled, where the state of a qubit is inherently correlated to another single/group of qubits.
- The outcome of measuring a qubit is probabilistic. When measured, a qubit can collapse into any of its basis states ($|0\rangle$ or $|1\rangle$), with probability equal to the square of its coefficients in the superposition state.

1.3.2 Parallelism

Due to the ability of qubits to exist in superposition states, qubits can compute operations on a vast number of permutations simultaneously. For example, take the task of searching through an unsorted database. For a classical computer with N entries in the database, it will have to search through each entry one by one, so the time required is proportional to N , or linear time ($O(N)$). Comparing that to quantum computers, Grover's Algorithm^[6] can search through a dataset of size N in roughly $O(\sqrt{N})$ time, offering a quadratic speedup.

Another example is the factorization problems of large integers. The most efficient classical algorithm, the General Number Field Sieve^[14] (GNFS) algorithm, has a subexponential time complexity; it is higher than polynomial time but lower than exponential time. The exact complexity, however, depends on various factors. Shor's Algorithm^[15], on the other hand, works in polynomial time, more specifically $O((\log N)^3)$, which is of course, going to be much faster than the GNFS algorithm.

1.3.3 Data Storage and Manipulations

In classical computers, the computation power or ability to represent larger data rises linearly with an increase in bits; in order to store an 8-bit binary number, a classical computer needs 8 bits. The addition of two 4-bit binary numbers requires 8 bits to store the numbers and 5 more bits to store the sum (assuming the largest possible sum). Similarly, the addition of two 8-bit numbers requires 16 bits total for the addends and another 9 bits for the sum.

In contrast, in quantum computers, computation power doubles with every added qubit. A singular qubit can only represent 2 bits, but adding another bit to the system allows the system to store 4 bits of data (see Eq.1.6), while three qubits can store 8 bits of data (see Eq.1.8).

In general, a classical computer needs N bits to store and manipulate an N -bit number. Whereas a quantum computer with N qubits can store and manipulate a 2^N -bit number. To put this in perspective, consider: in order to double the computation power of an N -bit classical computer, we would require $2N$ bits, and in order to double the power of a quantum computer with N qubits, we would only require $N + 1$ qubits.

This difference between quantum and classical computing can be expanded into operations on vast arrays of numbers. One math operation on 2^N numbers with an N -bit classical computer will require 2^N steps if done sequentially, or 2^N parallel processors. Doing the same operation on 2^N numbers on a quantum computer with N qubits requires only ONE step, as all the possible numbers are represented simultaneously due to superposition. This phenomenon alone is the source of the massive advantage quantum computers have over classical computers in parallelised computing. Table1.1 consolidates all the differences discussed till now.

1.4 Factory Layout Planning

Factory layout planning involves designing the optimal layout for a manufacturing facility to ensure efficiency, productivity, and safety. It is a critical aspect that can lead to increased productivity, reduced costs, and improved safety. Understanding the value stream and requirements, optimizing material flow, incorporating flexibility into the design, and placing

| Classical bit | Qubit |
|--|---|
| Two discrete states, 0 and 1 | Any linear combination of $ 0\rangle$ and $ 1\rangle$ |
| Are completely independent of other bits | Can be independent or quantumly entangled with other qubits |
| Can be measured completely | Can be measured partially with given probability |
| Are not affected by measurement | Are changed by measurement |
| Can be copied | Cannot be copied |
| Can be erased | Cannot be erased |
| Computation power rises linearly with increase in number of bits | Computation power rises exponentially with increase in qubits |

Table 1.1: Difference between bits and qubits

equipment in appropriate locations are key principles of production layout planning. Factory planning focuses on the physical properties needed for a facility to fulfil its function, including specifying equipment and modifications needed during operation. Layout planning is more focused on arranging what goes where within the facility. Different types of facility layouts include process layout, product layout, fixed-position layout, and cellular layout, each suitable for different production processes and industries. Let's explore these layout ideologies further

Process Based Layout

The process layout, also known as the functional layout, organizes equipment and tools into different groups based on production functions. In this layout, resources of a similar nature or function are grouped together, making it suitable for customized, low-volume products with varying processing requirements and sequences of operations. Examples of facilities using process layouts include machine shops, hospitals, banks, auto repair shops, libraries, and universities. The process layout aims to process goods or provide services that involve diverse processing requirements and offer flexibility to handle various routes and process needs efficiently. The design process for a process layout typically involves gathering information, developing a block plan or schematic of the layout, and creating a detailed layout. This type of layout is beneficial for minimizing transportation costs, distances, or times within a facility

Product Based Layout

The product layout is a production system where workstations and equipment are arranged along the line of production, commonly seen in assembly lines. Work units move along the line, and each workstation performs small amounts of work tailored to specific tasks, leading to higher production rates. Product layouts are suitable for large-scale industries with continuous processes like manufacturing units in industries such as sugar, paper, cement, automobiles, and electronic appliances like printers and refrigerators. This layout is characterized by a structured arrangement of machines focused on producing a single final product, where the output of one machine becomes the input for the next machine in a sequential manner. Product layouts require significant investment and space but are efficient for high-volume, standardized products that require repetitive processes.

Fixed-Position Based Layout

A fixed-position layout is a production system where the primary components of a project remain stationary while all other machinery, materials, and workforce come to the project. This layout is commonly used in industries where the essential equipment or components for operations are too heavy, intricate, or delicate to move, or when the project must be completed in a specific location. Examples of industries and facilities that use fixed-position layouts include airplane manufacturing, shipbuilding, boiler plants, turbines, construction sites for bridges, dams, and properties, and even hospital operating rooms. The advantages of a fixed-position layout include lower costs due to reduced product movement, easier customization for specific projects, and reduced product damage from shipping and transportation. However, there are also disadvantages, such as higher equipment movement costs, the need for skilled workers, space limitations, and the potential for longer completion times and project delays.

Cellular Based Layout

A cellular layout in manufacturing groups machines and equipment needed to produce a specific product or process together, improving efficiency and communication. It involves organizing workstations into self-contained cells, enhancing material flow, flexibility, quality

control, and reducing costs. Successful implementation requires identifying suitable products, emphasizing communication, investing in training, and monitoring performance for continuous improvement.

1.5 Scope

Let's focus on the product-based ideology for designing factory layouts. The methodology used in this thesis is applied on a factory with a similar ideology but can be adapted to be applied to any of the other ideologies. There are multiple steps required when designing a factory layout. A comprehensive list is given below, along with an estimate of how long each step takes.

1. Planning and Preparation (1-2 weeks): In this stage, pertinent information on the facility's existing condition—such as process flows, material handling protocols, inventory levels, and production volumes—must be gathered. Value stream mapping^[12] exercises are another tool stakeholders can use to find waste and inefficiencies in the current design.
2. Process Mapping and Segmentation (2-4 weeks): For every product or product family, create process maps or flowcharts, classifying them into groups according to comparable requirements or manufacturing processes. Determining the production flow and resource requirements for each section may also be part of this phase.
3. Stakeholder Engagement (2-4 weeks): Consult with engineers, operators, production managers, and other relevant parties for their opinions. Make sure all company goals, objectives, and financial restrictions are met. To increase support and buy-in for the layout design, address objections and take suggestions into consideration.
4. Layout Design (4-8 weeks): Provide layout solutions that satisfy every product segment's production needs. Organize resources, workstations, and equipment according to the production flow. Make the best use of available space and include flexibility in the layout plan.
5. Simulation and Analysis (2-4 weeks): Provide layout solutions that satisfy every product segment's production needs. Organize resources, workstations, and equipment

according to the production flow. Make the best use of available space and include flexibility in the layout plan.

6. Implementation (6-12 months): Carry out the implementation strategy, moving workstations, rearranging equipment, and upgrading infrastructure. In order to reduce production disruptions during the changeover, collaborate with the appropriate teams. Keep close tabs on developments and deal with any problems or difficulties that crop up throughout implementation.

Of course, the time required for each step can vary wildly because of numerous factors like the industry, geographic location, supply availability, scale of the factory, etc., which are completely different for different factories, plus other unforeseen circumstances like resource constraints, natural disasters, supply chain failures, equipment failures, which are impossible to predict beforehand with any sort of reliability.

This Thesis focuses on improving upon the "Layout Design" and "Simulation and Analysis" steps highlighted above. It aims to provide a new way of optimizing factory layouts on the basis of the rate of flow of products between machines or facilities (will be referred to as functional units from now on) and the size and shape of the factory, along with the size of functional units. It also aims to reduce the amount of manual labour of gathering copious amounts of data, consequently reducing the amount of data dumped onto the engineers and architects when simulating a digital twin of the factory^[1].

Currently, technological limitations on our implementation of physical quantum computers and solving algorithms impede our ability to generate granular, exact layouts. The layouts that are currently computationally feasible to generate offer a starting point for further qualitative analysis and fine-tuning done by engineers, to account for real-world constraints that will not always be mathematically modellable. This takes considerably less data and takes up much less computing time, and can give preliminary knowledge that can be used to analyse and pitch layouts much faster. Future technological innovations will allow the layouts to be much more exact while following the same methods, allowing further flexibility by being able to prioritize speed over accuracy or vice versa.

Chapter 2

Theory

Now that we have a basic understanding of quantum computing and factory layout planning, this subsequent chapter will focus on the physics and implementation of the quantum computer and the mathematical model used for this project.

There are many methods of engineering quantum computers, like Nuclear Magnetic Resonance Quantum Computers (NMRQC)^[16]; which use the spin states of nuclei in molecules as qubits, Trapped Ion qubits; which use the electron's different excitation states in an optically trapped ion as qubits, photonic qubits; where the presence of photons in a cavity is used to denote the states. However, the most reliable form of qubits are created using superconductors, as they provide an exceptional degree of control in manipulation, preparation, and efficient measurement, together with the possibility to tailor circuit properties and implement tunable qubit frequencies and coupling strengths.

2.1 Two types of Quantum Computing

Gate-based quantum computers and annealing-based quantum computers represent two different approaches to quantum computing.

2.1.1 Gate-Based Quantum Computers:

1. Gate-based models are more flexible and general than quantum annealing, as they can encode and solve any problem that can be expressed as a quantum circuit.
2. Universal gate quantum computing relies on building reliable qubits where basic quantum circuit operations can be put together to create any sequence, running increasingly complex algorithms.
3. Gate-based models can implement a wider range of quantum algorithms than quantum annealing, such as Grover's search, Shor's factorization, and quantum machine learning algorithms.

2.1.2 Annealing-Based Quantum Computers:

1. Quantum annealing seeks to utilize effects known as quantum fluctuations to find the best possible solution for the problem being solved.
2. Quantum annealing works by finding the minimum energy in the given problem's energy landscape
3. Quantum annealing works best on problems with many potential solutions where finding a "good enough" solution is sufficient, making it suitable for certain optimization tasks like faster flight or aerospace material research.

In conclusion, gate-based quantum computers offer more flexibility and broader applicability, while annealing-based quantum computers excel in specific optimization tasks. The choice between these two approaches is clear when considering our problem; finding an optimal or near-optimal layout for a given factory.

2.2 Annealing

Before we delve into the process of quantum annealing, Let us understand what the process of annealing is. This will help us to gain an understanding of the thought process behind quantum annealers, and how they came to be.

In metallurgy and materials science, annealing is a heat treatment technique used to change a material's physical and occasionally chemical properties to make it easier to work with. In this procedure, a material is heated above its recrystallization temperature, kept at that temperature for a specific duration of time, and then cooled slowly. Changes in ductility and hardness result from atoms migrating inside the crystal lattice during annealing, which lowers the amount of dislocations. The cooling process allows the material to recrystallize. Annealing is essential for reducing internal stresses, boosting ductility, and regulating grain size and phase composition. It can be used to enhance the formability, machinability, and general mechanical qualities of a variety of metals, including copper, silver, brass, aluminium, and steel. This manufacturing technique was so influential, that it gave birth to a new variation of the Monte Carlo algorithm; Simulated Annealing.

2.2.1 Simulated Annealing

Simulated Annealing, which was first theorised by Kirkpatrick et al., 1983^[11], can be used for heuristic optimization or approximate Boltzmann sampling, as it is essentially a metropolis algorithm. Here is how it works:

1. An initial configuration is chosen at random from the solution space of the system.
2. A change is proposed to the current configuration in the form of a new configuration that is slightly different from the initial one.
3. The new configuration is evaluated by measuring the change in energy associated with the update. Let's call this change in energy ΔE
4. If $\Delta E < 0$, i.e. the new configuration has a lower energy than the previous configuration, it is immediately accepted. If $\Delta E > 0$, the change is accepted with a probability. The probability is given by

$$P(\text{accept}) = e^{-\beta \Delta E} \tag{2.1}$$

$$\text{and } \beta = \frac{1}{k_B T} \tag{2.2}$$

where k_B is the Boltzmann constant and T is the temperature of the system. This is called a Metropolis-Hastings update.

5. If the update is accepted, the system is updated to the new configuration, and if rejected the system is kept as is. The temperature of the system, is then decreased according to a β -schedule, which is a sequence of temperatures that the system will update itself through. Each metropolis update will be called a *sweep*.
6. Repeat steps 2 to 5 until a predefined number of sweeps are completed, or the system reaches the final temperature in the β -schedule. Each completed run of the metropolis algorithm will be called a *read*.

When β is large, the target distribution concentrates, at equilibrium, over the ground states of the system. β -schedules can map any array of temperatures, but since solutions are guaranteed to match the equilibrium for long and smooth β schedules, they are usually linear, geometric, or logarithmic decreases in temperatures. Since problems can have a lot of local minimas, it is advisable to simulate multiple reads with a large number of sweeps to ensure the robustness of the solution.

2.3 Quantum Annealing by D’Wave

This study utilizes D’Wave’s quantum annealers. D’Wave uses superconducting loops as qubits, embedded in a 2-dimensional lattice the size of a thumbnail. These Quantum Processing Units, or QPUs, as coined by D’Wave, are maintained at around $12mK$ using a dilution fridge to minimize external interference and to maintain superconductive properties. The latest and greatest annealers, their 5th generation Advantage systems, contain more than 5000 qubits per qpu, giving users the ability to solve large problems with ease. D’Waves complete documentation for their quantum computers can be found here^[5].

2.3.1 How Quantum Annealing Works in D’Wave QPUs

The qubits in QPUs are the lowest energy states of the superconducting loops that make up the D-Wave QPU. These states have a circulating current and a corresponding magnetic field, as shown in Fig.2.1. As we know, these qubits can also exist in a superposition state of the two states. At the end of the quantum annealing process, each qubit collapses from a superposition state into either $|0\rangle$ or $|1\rangle$.

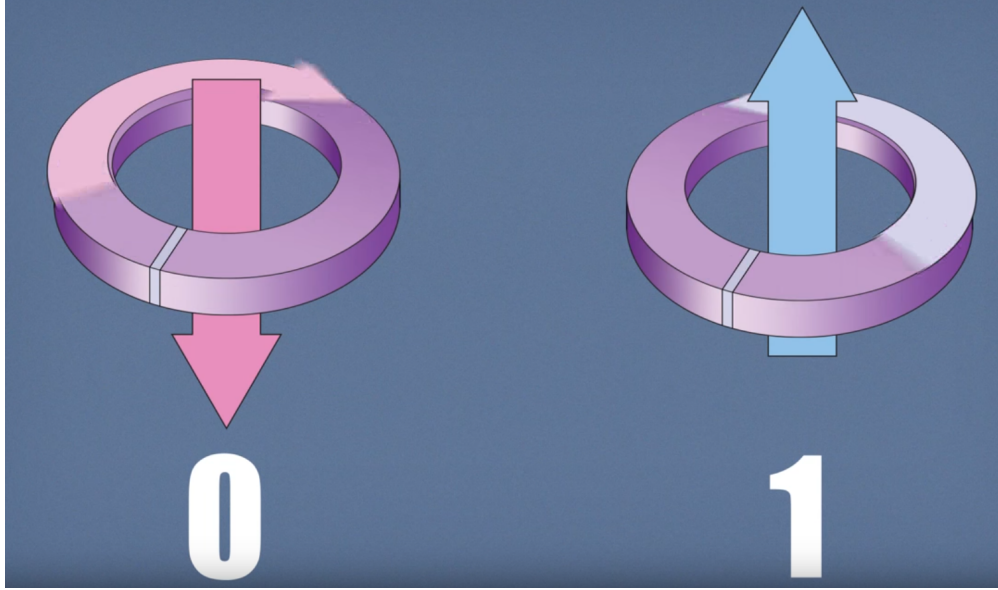


Figure 2.1: Qubit in a QPU

An energy diagram, like the one in Fig.2.2, can be used to illustrate the physics of this process. The diagram's evolution is depicted in (a), (b), and (c). There is a single minimum or valley (a) to start. When the barrier is raised due to the quantum annealing process, the energy diagram is transformed into a double-well potential (b). In this instance, the 0 state corresponds to the low point of the left valley, and the 1 state corresponds to the low point of the right valley. At the end of the anneal, the qubit finds itself in one of these valleys.

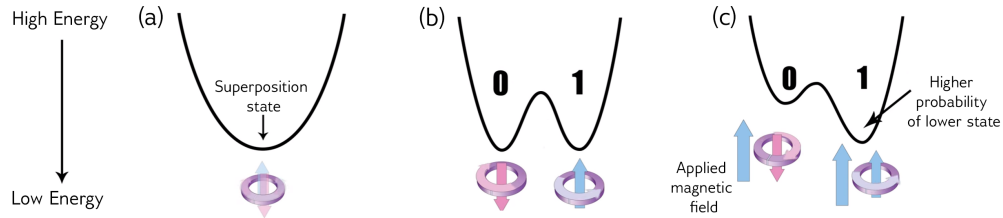


Figure 2.2: change in the annealing diagram as the annealing process runs

Assuming all other factors remain constant, there is a 50% chance that the qubit will end up in the 0 or 1 state. However, we may manipulate the likelihood that it will enter the 0 or 1 state by subjecting the qubit (c) to an external magnetic field. The double-well potential is tilted by this field, which raises the likelihood that the qubit will fall into the bottom well. A bias is a programmable quantity that regulates the external magnetic field; while the bias is present, the qubit minimizes its energy.

Still, the bias by itself is not helpful. When you connect qubits so they may affect one

another, or in other words, become entangled, that's when their true potential is revealed. A tool known as a coupler is used to accomplish this. Two qubits can be made to lean toward opposite states via a coupler, or they can tend toward the same state—both 0 or both 1. The correlation weights between connected qubits can be programmed by coupling strength, just like a qubit bias. In the D-Wave quantum computer, configurable biases and weights work together to define a problem.

Two qubits can be viewed as a single entity with four potential states when they are entangled. This concept is demonstrated in Fig.2.3 by a potential with four states: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, each of which represents a distinct combination of the two qubits(Eq.1.6). The coupling between qubits and their biases determine the relative energy of each state. The qubit states are potentially delocalized in this landscape during the anneal, and at the end of the anneal, they settle into $|11\rangle$. Users set the biases and couplers' values while defining the problem. The quantum computer determines the minimum energy of an energy landscape that is defined by the biases and couplings.

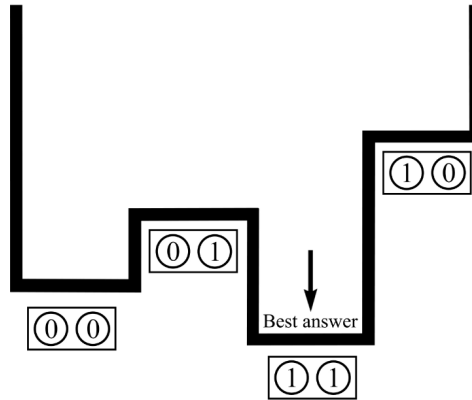


Figure 2.3: Energy Diagram showing the best answer

In summary, the system begins with a set of qubits all in the superposition states of 0 and 1. They are not entangled yet. The couplers and biases are added, and the qubits get entangled when the annealing process starts. The system is now in a state of entanglement with numerous potential solutions. Each qubit is in a classical state by the end of the anneal, which either reflects the problem's least energy state or one that is quite near to it.

2.3.2 Underlying Physics

For the D'Wave quantum computer, the Hamiltonian is described as

$$\mathcal{H}_{ising} = \underbrace{-\frac{A(T)}{2} \left(\sum_i \hat{\sigma}_x^{(i)} \right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(T)}{2} \left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \right)}_{\text{Final Hamiltonian}} \quad (2.3)$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on a qubit q_i , and h_i and $J_{i,j}$ are the qubit biases and the coupling strengths. We can see that the Hamiltonian is the sum of two terms, the *Initial Hamiltonian* and the *Final Hamiltonian*. Both Hamiltonians have a scaling factor attached to them, $A(T)$ and $B(T)$, and these are called the energy scales.

- The first term, or the initial Hamiltonian's lowest energy is when all qubits are in a superposition state of $|0\rangle$ and $|1\rangle$. We will call it the tunnelling Hamiltonian.
- The second term, or the final Hamiltonian's lowest energy is the solution to the problem we are trying to solve. Thus we will call it the problem Hamiltonian.

Quantum annealing occurs between the time $t = 0$ and t_f , which for simplicity, is parameterized as T ranging from 0 to 1. This is called the normalized anneal fraction. The system begins at $t = 0$ in the ground state of the initial Hamiltonian (each spin s_i is in a superposition state $s_i = \pm 1$), that is, $A(0) \gg B(0)$. As the annealing proceeds, the problem Hamiltonian is slowly introduced to the system by increasing B , which contains all the biases and couplers, while the initial Hamiltonian's influence is slowly reduced by reducing A . At the end of an annealing cycle ($t = t_f$ or $T = 1$), that is, $B(T) \gg A(T)$, ideally, the system will have stayed in the ground state throughout the annealing process so that by the end of the cycle the system is in the ground state of the problem Hamiltonian, which will be the solution to our problem. Each qubit will have also decohered into a classical object by the end of this cycle, as it would've collapsed into one of its two states. At this point, $\hat{\sigma}_z^{(i)}$ can be replaced by classical spin variables $s_i = \pm 1$, and so the Hamiltonian is described by the classical Ising spin system in Eq2.4, such that the spin states represent the low energy solution.

$$E_{ising} = \sum_i h_i s_i + \sum_{i>j} J_{i,j} s_i s_j \quad (2.4)$$

A clear advantage to note is that quantum annealing does not have to deal with the issue of potentially getting stuck inside a local minima. As depicted in Fig.2.4, annealing would require a lot of "lucky" iterations or a supply of extra "energy" to the system to overcome the peaks around the minimas, but quantum annealing exploits quantum tunnelling and superposition to go "through" the peak while traversing the energy landscape to escape from the local minima and potentially find the global minima.

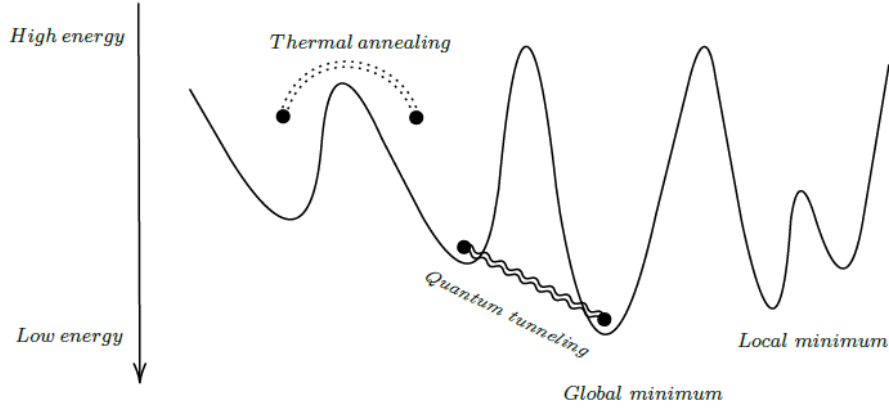


Figure 2.4: Visual representation of the traversal through the energy landscape by quantum and simulated annealing

2.3.3 Hardware and Energy Scales

The physical construction of the QPUs is a network of radio frequency superconducting quantum-interference devices (rf-SQUID)^[7]. The physical Hamiltonian is expressible as

$$\begin{aligned}
 H = & -\frac{1}{2} \sum_i [\Delta_q(\Phi_{\text{CCJJ}}(T)) \hat{\sigma}_x^{(i)} - 2h_i |I_p(\Phi_{\text{CCJJ}}(T))| \Phi_i^x(T) \hat{\sigma}_z^{(i)}] \\
 & + \sum_{i>j} J_{i,j} M_{\text{AFM}} I_p(\Phi_{\text{CCJJ}}(T))^2 \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)}
 \end{aligned} \tag{2.5}$$

Here,

- Δ_q is the energy difference between the two eigenstates ($\frac{|0\rangle \pm |1\rangle}{\sqrt{2}}$) of the qubit with no externally applied flux. this energy difference is the contribution of coherent tunnelling between the two energy wells.
- I_p is the magnitude of current flowing through the rf-SQUID qubits.

- M_{AFM} is the maximum mutual inductance generated between the qubits by the couplers.
- $\Phi_i^x(T)$ is an external flux applied to the qubits.
- $\Phi_{CCJJ}(T)$ is an external flux applied to all Josephson-junction^[10] structures of the qubits

If in Eq.2.5, we set $\Phi_i^x(T) = M_{AFM}|I_p(T)|$ we can see that as Φ_{CCJJ} changes during the anneal, $\Phi_i^x(T)$ changes so as to the ratio of energy between the h_i and $J_{i,j}$ terms constant. Physically, this means that as the annealing progresses, the magnitude of external flux needed to maintain a steady h value increases. Now, Eq.2.5 can be mapped to Eq.2.3. Thus we get the following expression for the energy scales $A(T)$ and $B(T)$:

$$A(s) = \Delta_q(\Phi_{CCJJ}(s)) \quad (2.6)$$

$$B(s) = 2M_{AFM}|I_p(\Phi_{CCJJ}(s))|^2 \quad (2.7)$$

This shows the problem Hamiltonian's energy scale grows quadratically, as seen in Fig.2.5

2.4 Model Formulation

To solve problems using the Q'Wave QPUs, we need to formulate the problem Hamiltonian (for simplicity, the problem Hamiltonian will be referred to just as the Hamiltonian). This can be done using quadratic models.

2.4.1 Quadratic Models

The Ocean SDK provided by D'Wave has three quadratic models

- **Binary Quadratic Models (BQM):** These are unconstrained and use binary decision variables, which can either be true (yes) or false (no).

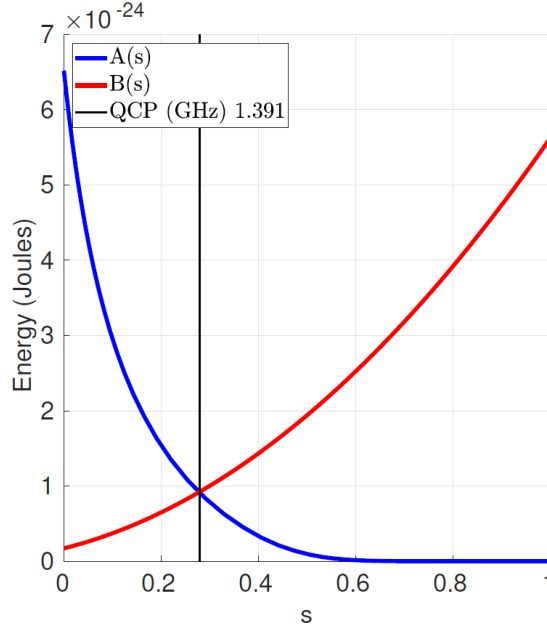


Figure 2.5: Energy scales versus parameterized time s in D’Waves Advantage QPUs. The quantum critical point (QCP) is the point in the anneal where $A(s)$ and $B(s)$ have the same magnitude. The system is very sensitive to quantum fluctuations at this point.

- **Constrained Quadratic Models (CQM):** These can be constrained and can use integer and binary decision variables.
- **Discrete Quadratic Models (DQM):** These are unconstrained and have discrete decision variables.

For our FLP problems, we are going to be using a BQM. This class of quadratic models can use either Ising formulation, where the decision variables are ± 1 or Quadratic Unconstrained Binary Optimization (QUBO) formulation, where the decision variables are either 0 or 1. Both formulations work the same way, in fact, QUBO models are converted to Ising models anyway when submitting a problem to the QPUs. But depending on the problem at hand, choosing one of the two can simplify the Hamiltonian for our use. We will choose the QUBO formulation for our problem.

The general QUBO model is an objective function (the Hamiltonian) on N binary variables represented as a real-valued, $N \times N$ upper-triangular matrix \mathcal{Q} . The diagonal terms ($\mathcal{Q}_{i,i}$) are the linear coefficients (biases) and the non-zero off-diagonal terms ($\mathcal{Q}_{i,j}$) are the quadratic coefficients (coupling strengths). The Energy of the QUBO model is given by the

expression

$$E(x) = \sum_{i \leq j} x_i Q_{i,j} x_j \quad (2.8)$$

where x_i is our binary decision variable ($N = [x_1, x_2, x_3, \dots, x_N]$), and $x_i \in \{0, 1\}$. In scalar notation, the objective can be expressed as:

$$E_{qubo} = \sum_i a_i q_i + \sum_{i \leq j} b_{i,j} q_i q_j \quad (2.9)$$

2.4.2 Building the model

To start, a qualitative analysis of the size of the factory and the size of the individual functional units is done to choose a certain unit area A . This unit area is then used to divide the factory into p positions $\mathcal{P} = \{p_1, p_2, p_3, \dots, p_p\}$ which are assigned to one of f functional units $\mathcal{F} = \{f_1, f_2, f_3, \dots, f_f\}$. The transportation intensity or rate of flow of product from functional unit f to f' with $f, f' \in F$ is stored in a square transportation matrix denoted by T with dimensions $f \times f$. Similarly, the distance between any two positions $p, p' \in P$ is stored in a square distance matrix denoted by D with dimensions $p \times p$. The matrix to be optimized is denoted by X with dimensions $f \times p$. The value of A affects the size of the problem. A smaller A will increase the number of positions offering a more granular and precise solution, but will also increase the complexity and time of the computation. Taking larger values of A will result in a small problem that is easier to solve but will compromise the accuracy of the solution. It is imperative to choose an appropriate value for A based on our priorities.

A is also used to calculate the relative size of the functional units by dividing the actual area of the functional units by A and rounding off to the nearest integer. The relative sizes are denoted by S ($\mathcal{S} = \{S_1, S_2, S_3, \dots, S_f\}$.)

Now we can define the Hamiltonian for our problem:

$$H_1 = \min \left(\sum_{p,p'} \sum_{f,f'} x_{f,p} \times x_{f',p'} (D_{p,p'} \times T_{f,f'}) \right) \quad (2.10)$$

where $x_{f,p} \in X$, and,

$$x_{f,p} = \begin{cases} 1 & \text{if functional unit } f \text{ is in position } p, \\ 0 & \text{otherwise.} \end{cases} \quad (2.11)$$

We can quickly verify the validity of this Hamiltonian with an example. If $x_{f,p} = 1$ and $x_{f',p'} = 1$, the energy of these two qubits is the product of the transportation intensity $T_{f,f'}$, and the distance $D_{p,p'}$ between the two positions they are placed in. If one of the decision variables is 0, that is, the position p is empty, the entire expression becomes 0, irrespective of the transportation intensity and distance. When summed over all positions and functional units, the energy will be minimized when functional units with high transportation intensity are placed closer together. This is the basis of the minimization of energy of the Hamiltonian. If the problem is submitted as is, the solver will just return a configuration where all machines are placed on top of each other. This means we need to introduce some constraints into our model.

As the QUBO model needs to be unconstrained, setting hard limitations on the Hamiltonian is not possible. However, we can apply penalties in the form of additional addends to the Hamiltonian defined in Eq.2.10, which artificially inflate the energy for unfavourable configurations.

Our model requires two penalties which can be modelled as

$$H_2 = \sum_p \sum_{f,f'} x_{f,p} \times x_{f',p} \quad (2.12)$$

which enforces that the number of functional units assigned to a position p cannot exceed 1.

And,

$$H_3 = \sum_f \left(\sum_p x_{f,p} - S_f \right)^2 \quad (2.13)$$

which enforces that the number of positions assigned to a functional unit f must equal the

relative size S_f of the functional unit.

The final Hamiltonian of our problem would be,

$$H = \alpha H_1 + \beta H_2 + \gamma H_3 \quad (2.14)$$

where α , β , and γ are hyperparameters. Here is where we will realise the benefit of using a QUBO model. It is obvious that the Hamiltonian is only valid if we use 0 and 1 decision variables, but another property is that on expanding the Hamiltonian, all the $x_{f,p}^2$ terms can be replaced with $x_{f,p}$ without losing any information.

2.4.3 Inputs for the Model

All the data about the factory was taken from Ojaghi et al., 2015. According to the data, the following values are chosen/calculated.

- The number of functional units $f = 13$.
- Unit area A is chosen to be 5m^2 . Using this, the relative area of each functional unit is calculated. The table of functional units, along with their size, is given in Table.2.1
- Number of positions $p = 81$. The factory floor is divided into a 9×9 matrix.
- The transportation matrix is created by using the REL matrix given in the paper. Since there isn't actual transportation data provided, the pipeline given by the paper is analyzed, and the elements in the transportation matrix are deleted(set to 0) if the two functional units don't have any product flow between them. The diagonal elements of this matrix are what we will call *internal flow*, a non-physical quantity, as a functional unit's output cannot be its own input. Its only use is to incentivize the solver to assign adjacent positions to the same functional unit. This value is set higher than every other value to prevent "breakage" of the functional units and is functionally a hyperparameter. Finally, all values of the matrix are cubed.
- The distance matrix is created by taking the cartesian distance between the centres of all positions.
- The hyperparameters that yielded the best results are $\alpha = 1$, $\beta = 200$, $\gamma = 300$

| S. No. | Department | Size(m2) | Relative Size |
|--------|----------------------------|----------|---------------|
| 1 | Receiving Department | 48 | 10 |
| 2 | Raw Material Sotrage | 20 | 4 |
| 3 | Crushing Department | 34.8 | 7 |
| 4 | Peeling Department | 21 | 4 |
| 5 | Chopping-Mixing Department | 9.45 | 2 |
| 6 | Chopping Department | 5 | 1 |
| 7 | Forming-Cooking Department | 70 | 14 |
| 8 | Cooking-Mixing Department | 7.5 | 2 |
| 9 | Blasting Department | 67.5 | 14 |
| 10 | Packaging Department | 16.32 | 3 |
| 11 | Filling Department | 5 | 1 |
| 12 | Food Court | 15 | 3 |
| 13 | Finished Goods | 10 | 2 |

Table 2.1: Functional units and their sizes

$$T = \begin{bmatrix} 10 & 6 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 6 & 10 & 6 & 6 & 6 & 3 & 3 & 4 & 2 & 4 & 4 & 3 & 4 \\ 2 & 6 & 10 & 4 & 5 & 3 & 2 & 2 & 2 & 2 & 2 & -4 & 2 \\ 2 & 6 & 4 & 10 & 3 & 5 & 2 & 4 & 2 & 2 & 2 & -4 & 2 \\ 2 & 6 & 5 & 3 & 10 & 4 & 5 & 3 & 2 & 2 & 2 & -4 & 2 \\ 2 & 3 & 3 & 5 & 4 & 10 & 3 & 5 & 2 & 3 & 2 & -4 & 2 \\ 2 & 3 & 2 & 2 & 5 & 3 & 10 & 4 & 5 & 3 & 2 & 2 & 2 \\ 2 & 4 & 2 & 4 & 3 & 5 & 4 & 10 & 5 & 3 & 3 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 5 & 5 & 10 & 5 & 5 & 3 & 3 \\ 2 & 4 & 2 & 2 & 2 & 3 & 3 & 3 & 5 & 10 & 4 & 3 & 4 \\ 2 & 4 & 2 & 2 & 2 & 2 & 2 & 3 & 5 & 4 & 10 & 3 & 4 \\ 2 & 3 & -4 & -4 & -4 & -4 & 2 & 2 & 3 & 3 & 3 & 10 & 3 \\ 2 & 4 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 4 & 4 & 3 & 10 \end{bmatrix} \quad (2.15)$$

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 & 1.41 & 2.24 & 3.16 & \dots \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 1.41 & 1 & 1.41 & 2.24 & \dots \\ 2 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 2.24 & 1.41 & 1 & 1.41 & \dots \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 3.16 & 2.24 & 1.41 & 1 & \dots \\ 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 4.12 & 3.16 & 2.24 & 1.41 & \dots \\ 5 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 5.1 & 4.12 & 3.16 & 2.24 & \dots \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 6.08 & 5.1 & 4.12 & 3.16 & \dots \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 7.07 & 6.08 & 5.1 & 4.12 & \dots \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8.06 & 7.07 & 6.08 & 5.1 & \dots \\ 1 & 1.41 & 2.24 & 3.16 & 4.12 & 5.1 & 6.08 & 7.07 & 8.06 & 0 & 1 & 2 & 3 & \dots \\ 1.41 & 1 & 1.41 & 2.24 & 3.16 & 4.12 & 5.1 & 6.08 & 7.07 & 1 & 0 & 1 & 2 & \dots \\ 2.24 & 1.41 & 1 & 1.41 & 2.24 & 3.16 & 4.12 & 5.1 & 6.08 & 2 & 1 & 0 & 1 & \dots \\ 3.16 & 2.24 & 1.41 & 1 & 1.41 & 2.24 & 3.16 & 4.12 & 5.1 & 3 & 2 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (2.16)$$

The transportation and distance matrices are given in Eq.2.15 and Eq.2.16 respectively. The matrices are square symmetric. The distance matrix D is incompletely shown as the true matrix is too big to fit onto the page, but it is easy to visualize what the matrix will look like. Each position is labelled in the order of left to right of each row. Fig.2.6 depicts the labelling of positions.

| Position Labels | | | | | | | | |
|-----------------|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 |
| 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 |
| 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 |
| 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |

Figure 2.6: Labels of the positions of the factory floor.

Chapter 3

Results

The problem was solved through simulated annealing and quantum annealing. The quantum solver used was D’Wave’s hybrid solver, as the problem was too large to be directly embedded onto the QPUs. The hybrid solver uses a proprietary algorithm to break the problem into smaller parts using classical means, solve each part on the QPU separately and then consolidate all the parts again using a classical computer. Both methods return a matrix whose elements are the values of the binary decision variables. An example is depicted by Fig.3.1.

This matrix is then post-processed and represented as a layout of the factory. The layout will be interpreted as a top-down view of the factory floor, allowing us to visualize where each facility is placed (Fig.3.2).

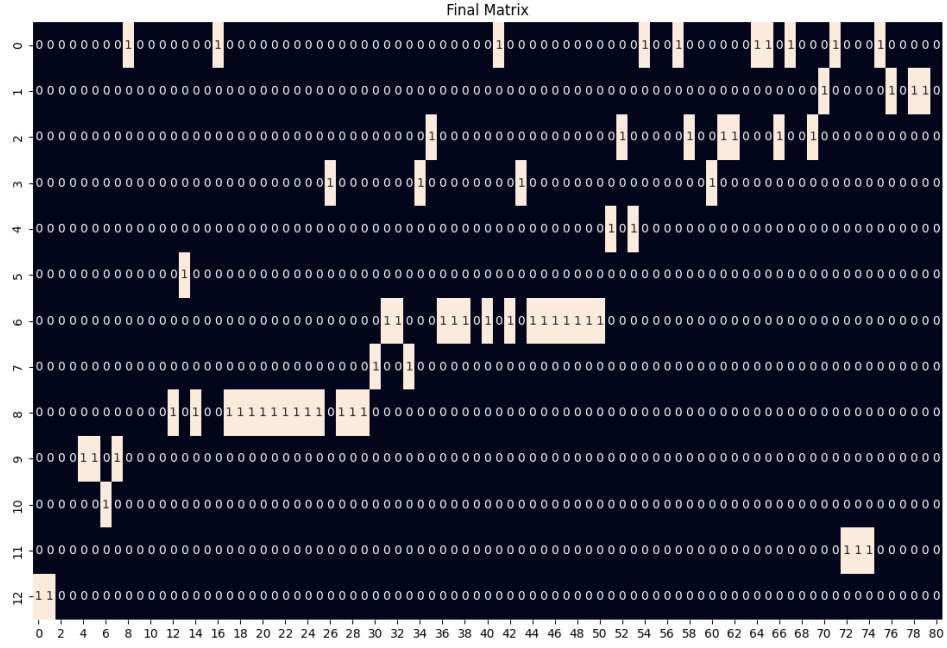


Figure 3.1: A solution generated by the Simulated Annealing Solver

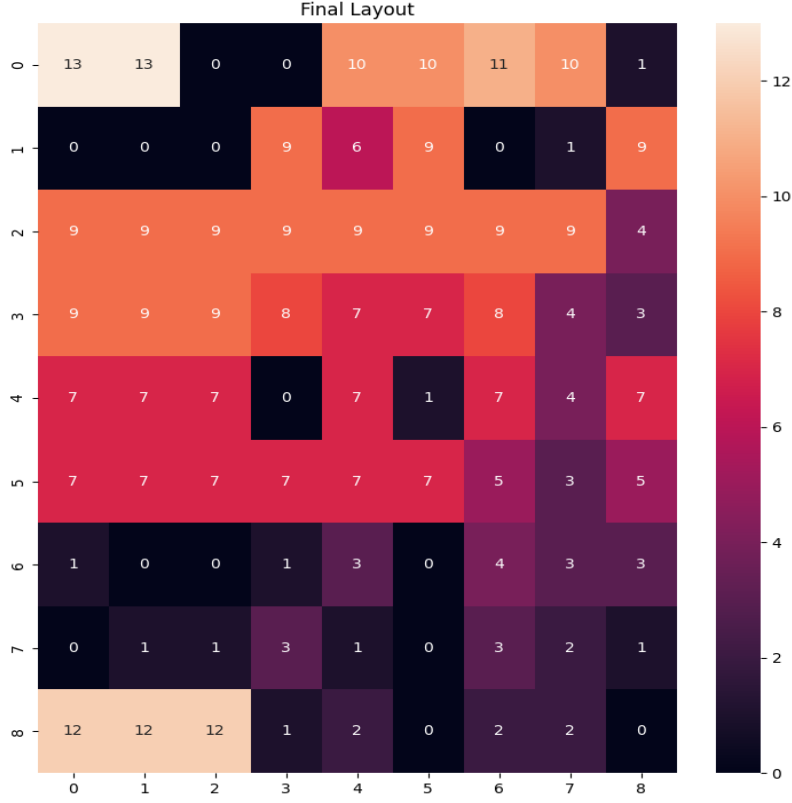


Figure 3.2: The post-processed solution

3.1 Simulated Annealing

Since simulated annealing is probabilistic, and there are multiple low-energy configurations that are not physically possible due to breakage, multiple runs were statistically averaged to highlight the most common spots for each functional unit to be placed.

Each run was initialized in a random configuration, and the number of sweeps was set to 400,000, while the reads were set to 1000. So a single simulated annealing process of 400,000 iterations was repeated 1000 times, and the best result from those 1000 runs was chosen, and its energy was noted. We will call this a "sample". This set of 1000 simulations or samples was repeated 80 times, and the statistical averaging of the samples was conducted. This was done by counting the positions where each functional unit occurred.

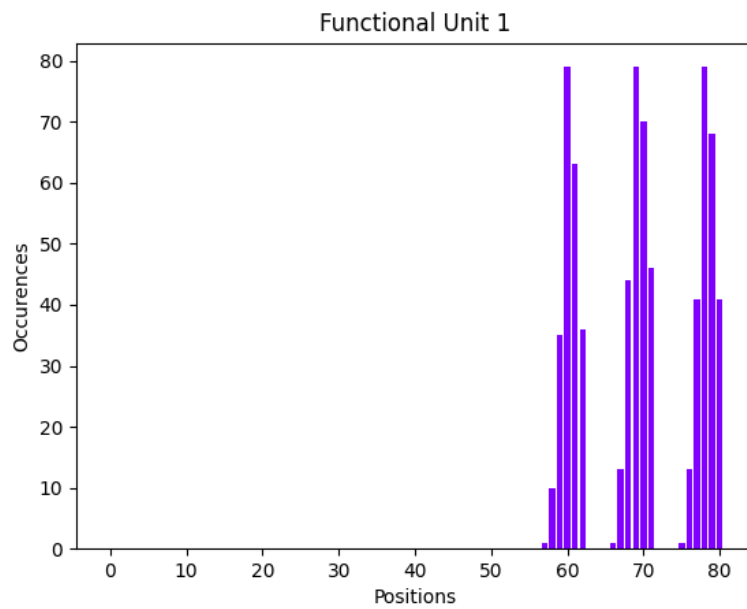


Figure 3.3: Occurrence vs Position for functional unit 1

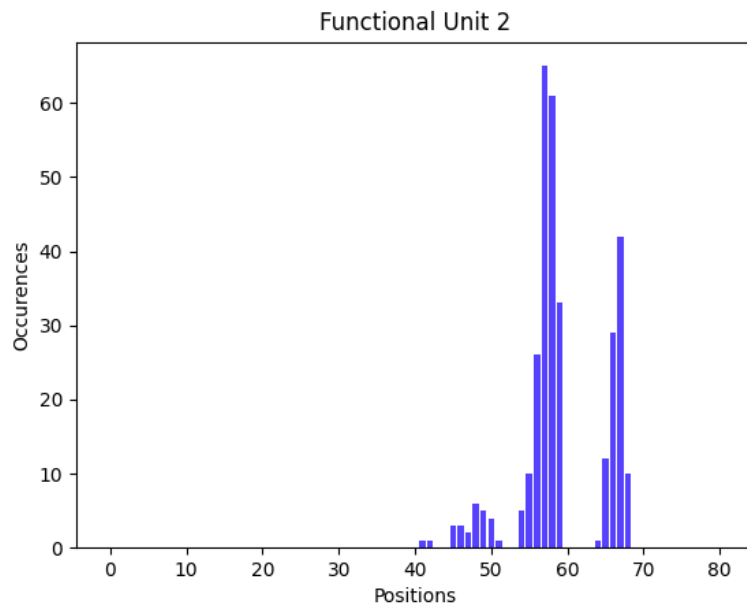


Figure 3.4: Occurrence vs Position for functional unit 2

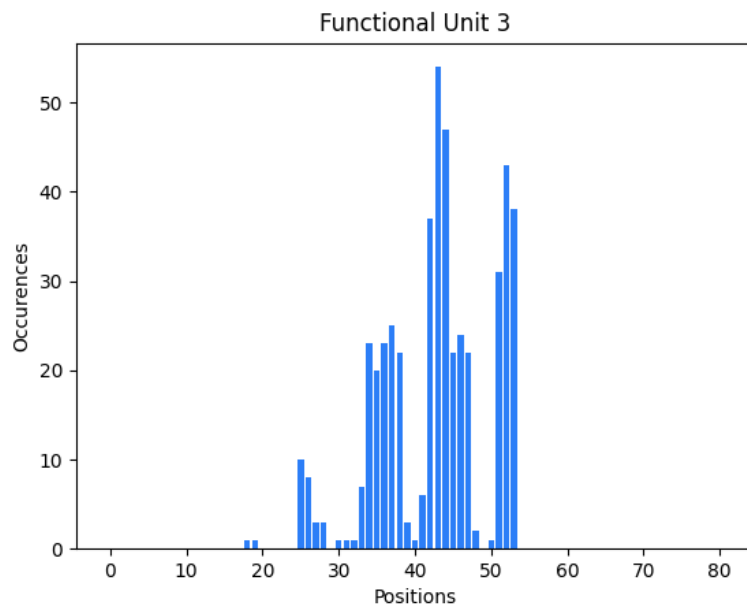


Figure 3.5: Occurrence vs Position for functional unit 3

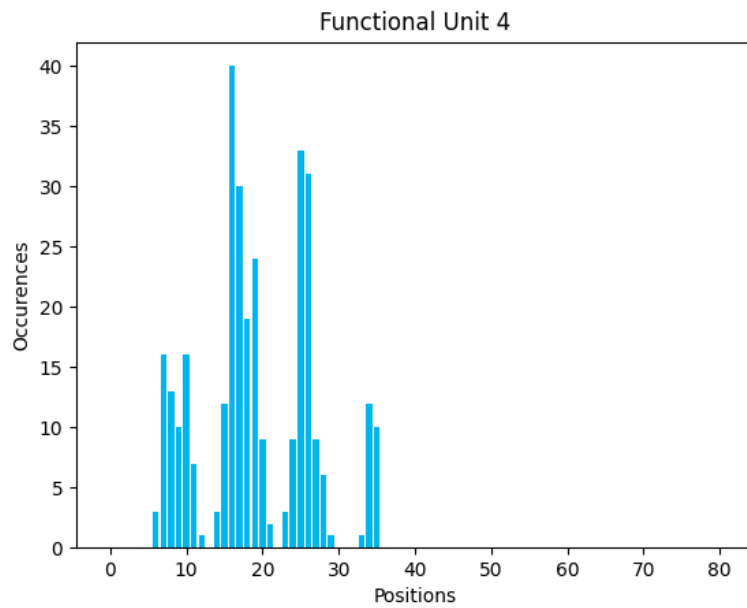


Figure 3.6: Occurrence vs Position for functional unit 4

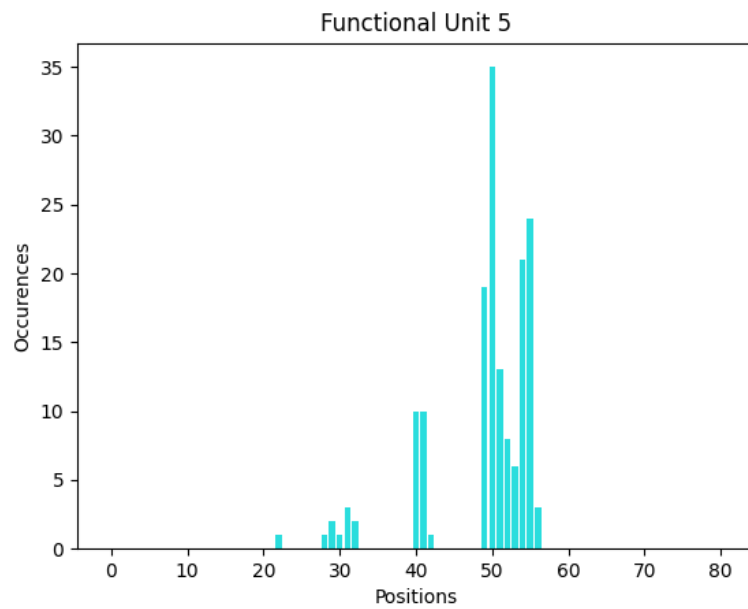


Figure 3.7: Occurrence vs Position for functional unit 5

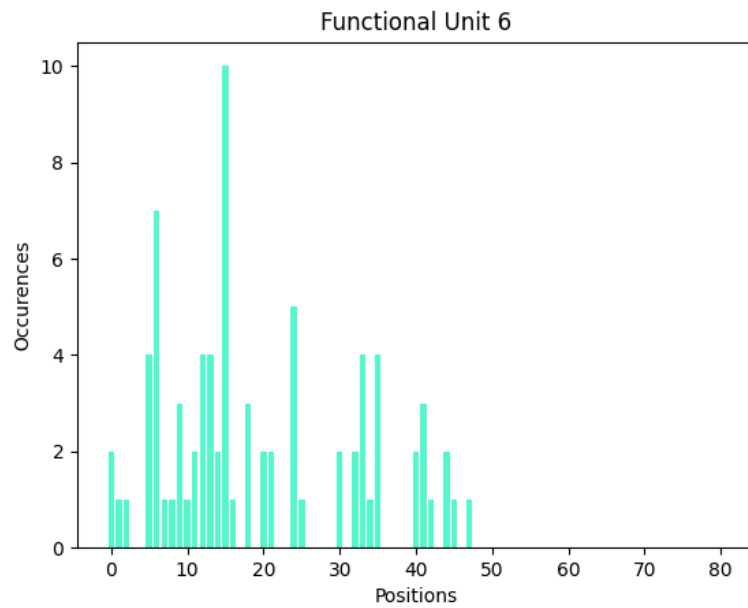


Figure 3.8: Occurrence vs Position for functional unit 6

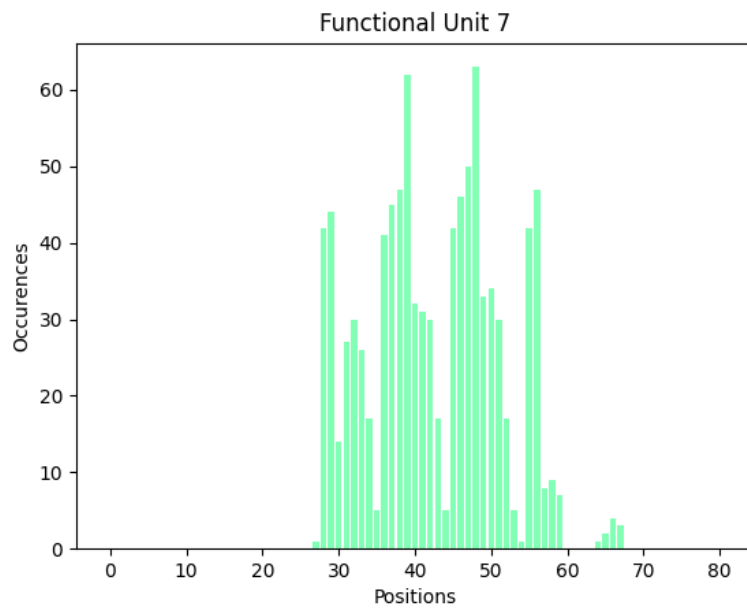


Figure 3.9: Occurrence vs Position for functional unit 7

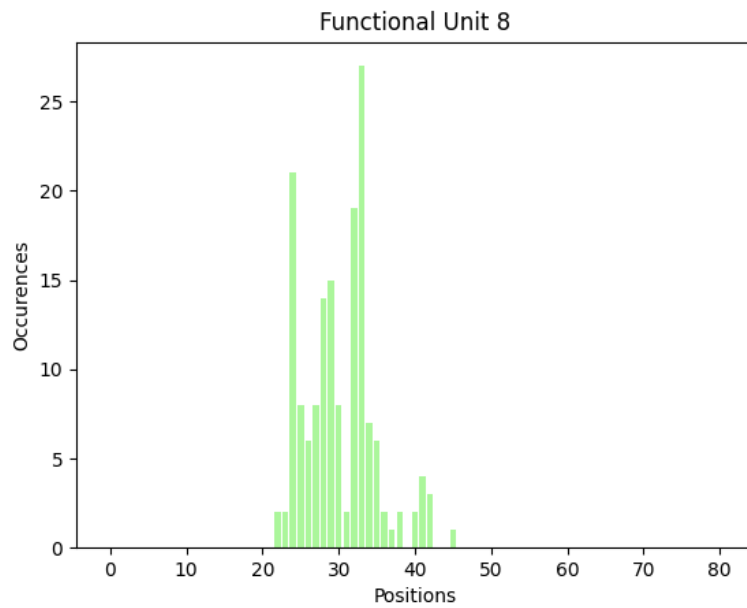


Figure 3.10: Occurrence vs Position for functional unit 8

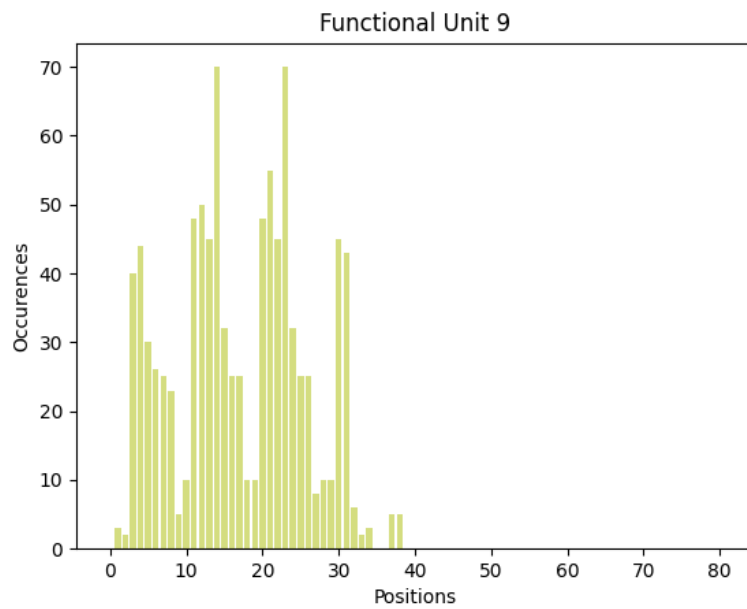


Figure 3.11: Occurrence vs Position for functional unit 9

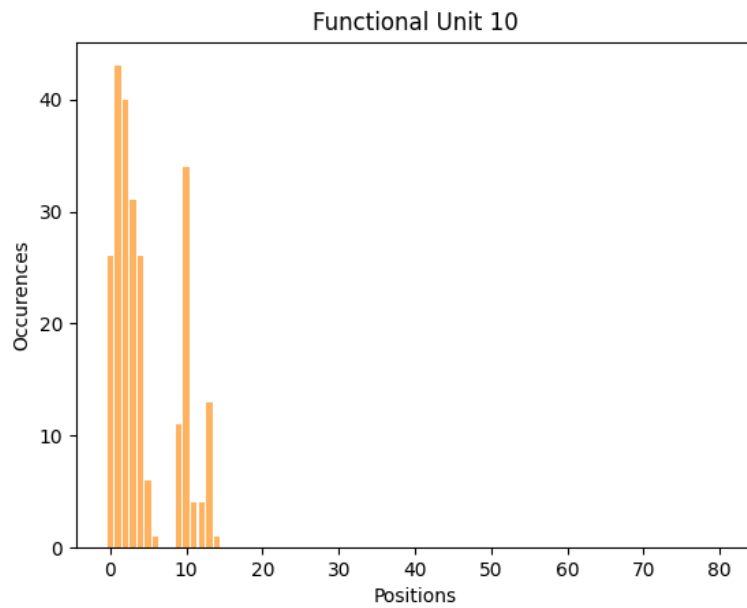


Figure 3.12: Occurrence vs Position for functional unit 10

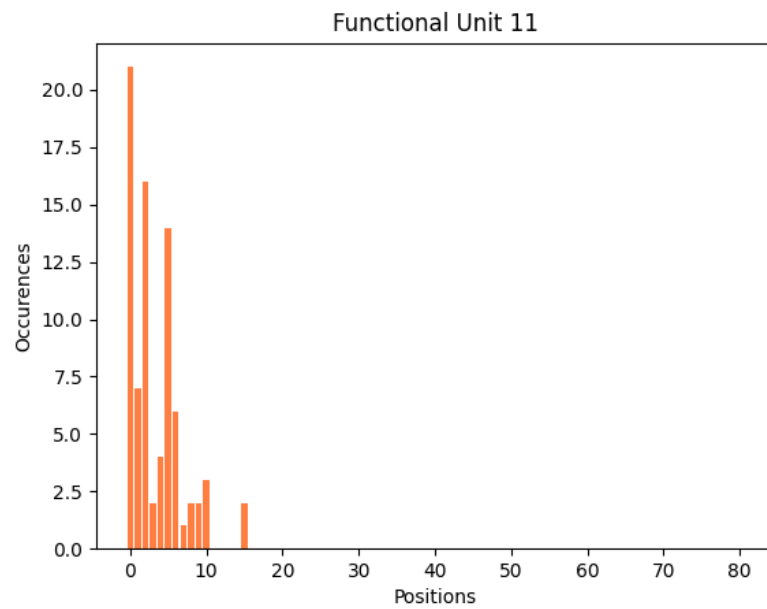


Figure 3.13: Occurrence vs Position for functional unit 11

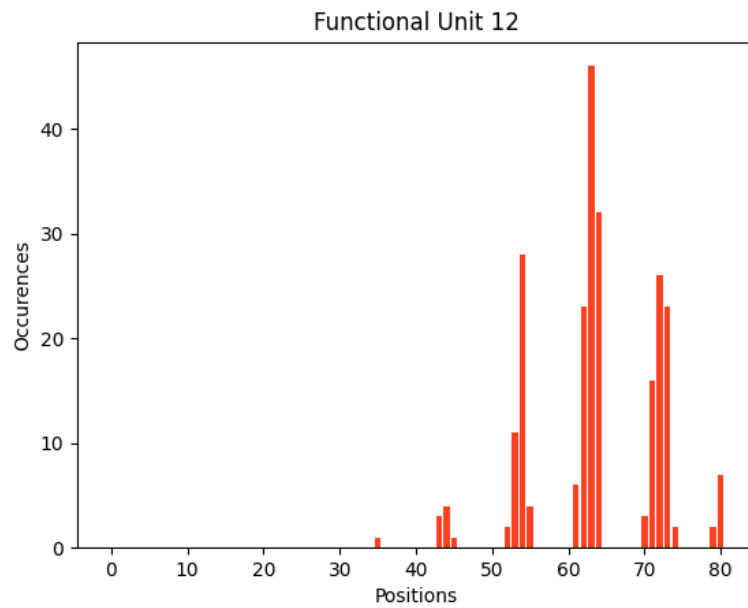


Figure 3.14: Occurrence vs Position for functional unit 12

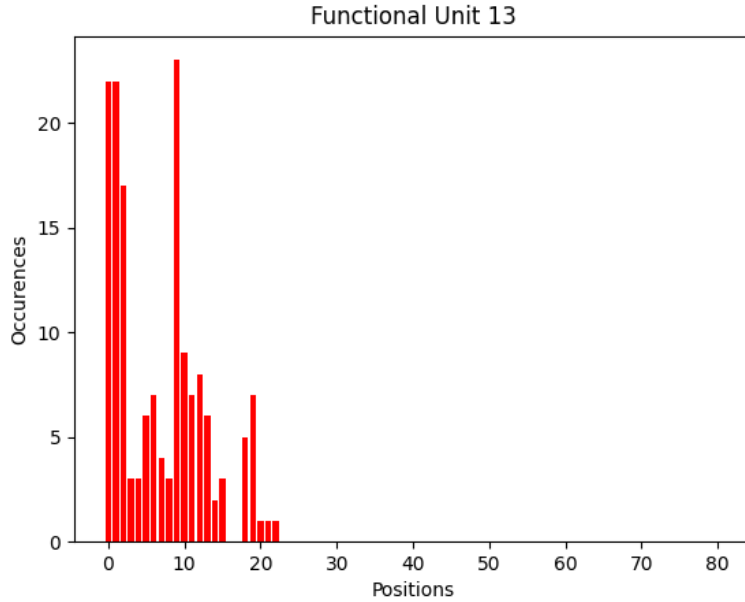


Figure 3.15: Occurrence vs Position for functional unit 13

From Figs.3.3-3.15 we can see that most facilities peaks exist at different positions. While some plots look like they contain separated high peaks, that is not the case. Vertical neighbours of any cells, while far apart in labels (multiples of 9 apart), are geometrically very close to each other. On closer inspection of the above figures where there are multiple dominant peaks, it is visible that the peaks are ≈ 9 positions apart, which makes them immediate, or near immediate vertical neighbours.

The same data can be depicted as a heatmap per functional unit, too; it will make the visualization - of separate peaks not necessarily being geometrically far apart - easier.

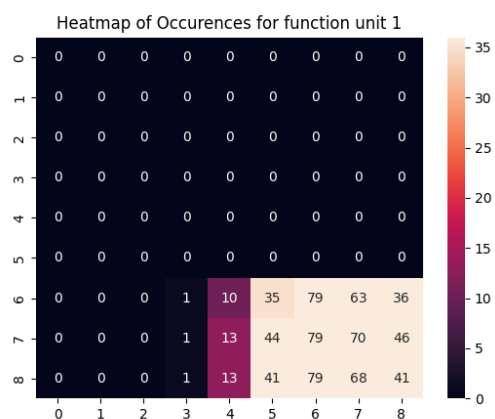


Figure 3.16: Heatmap of functional unit 1

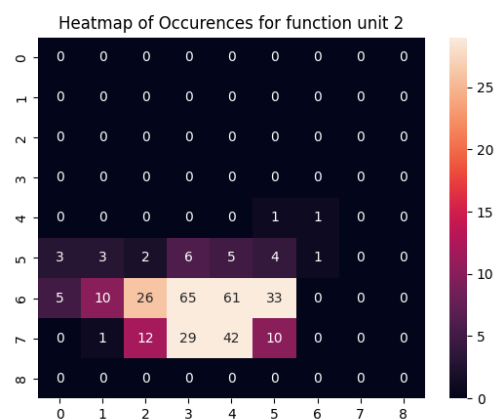


Figure 3.17: Heatmap of functional unit 2

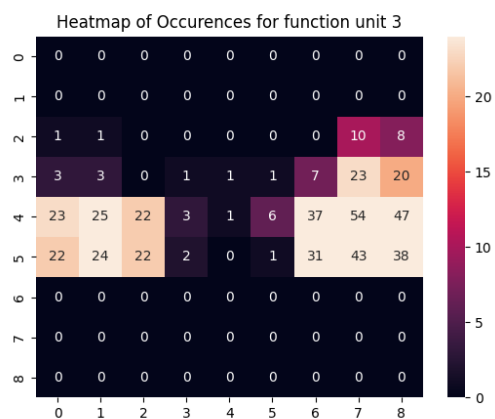


Figure 3.18: Heatmap of functional unit 3

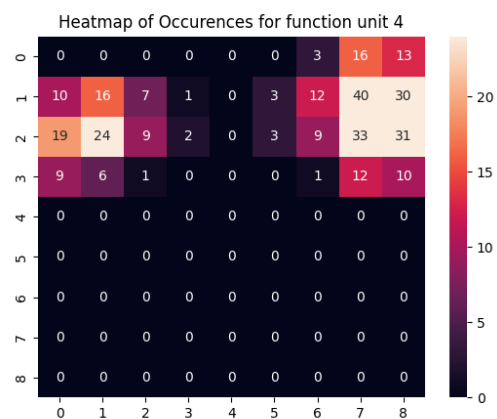


Figure 3.19: Heatmap of functional unit 4

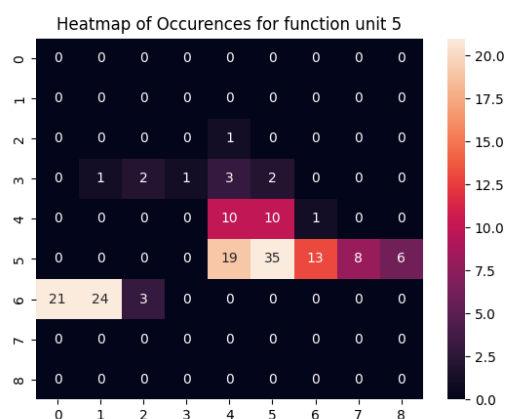


Figure 3.20: Heatmap of functional unit 5

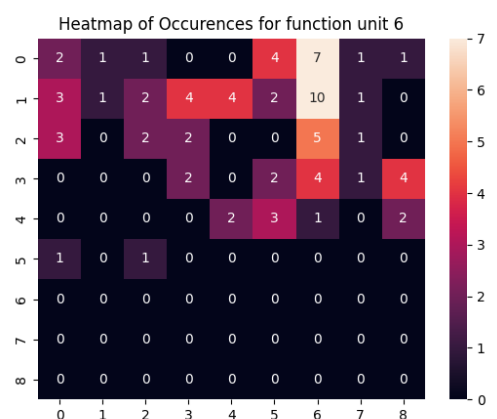


Figure 3.21: Heatmap of functional unit 6

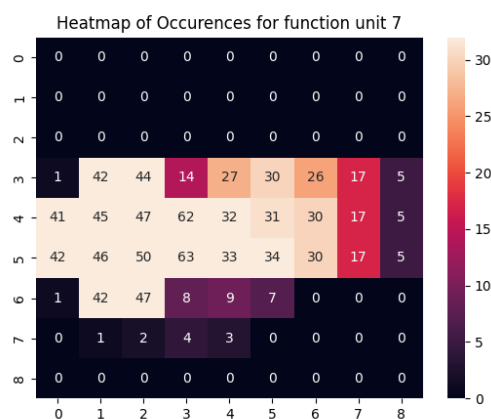


Figure 3.22: Heatmap of functional unit 7

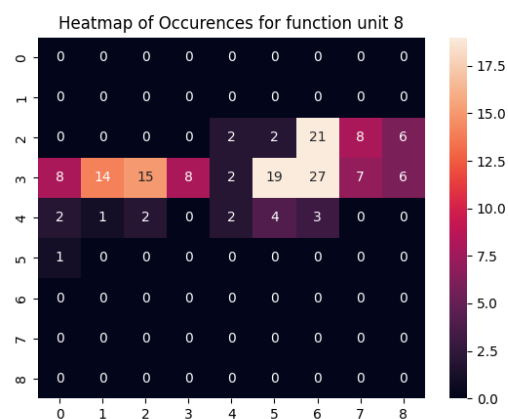


Figure 3.23: Heatmap of functional unit 8

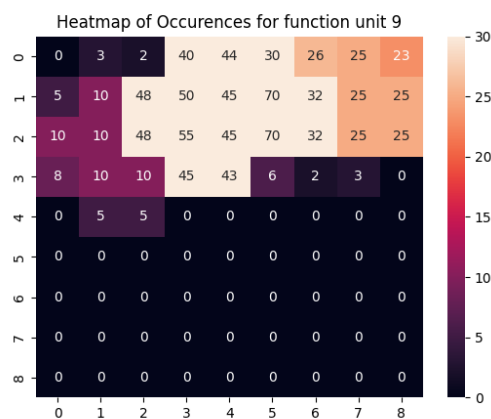


Figure 3.24: Heatmap of functional unit 9

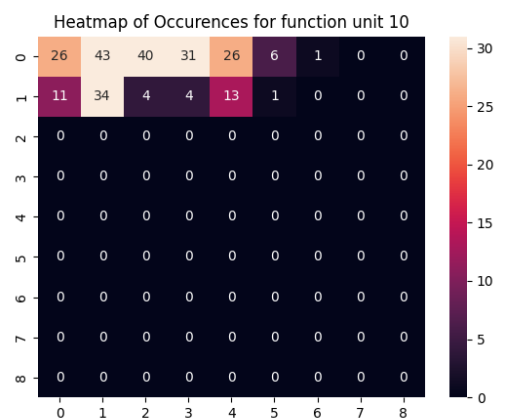


Figure 3.25: Heatmap of functional unit 10

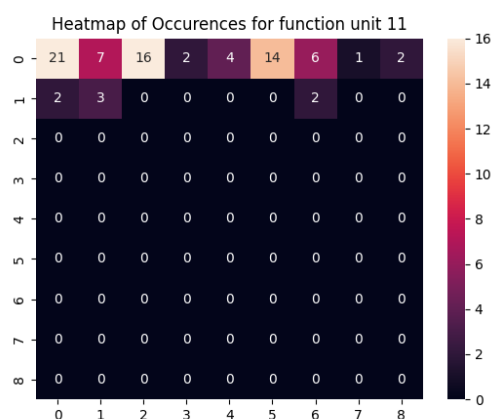


Figure 3.26: Heatmap of functional unit 11

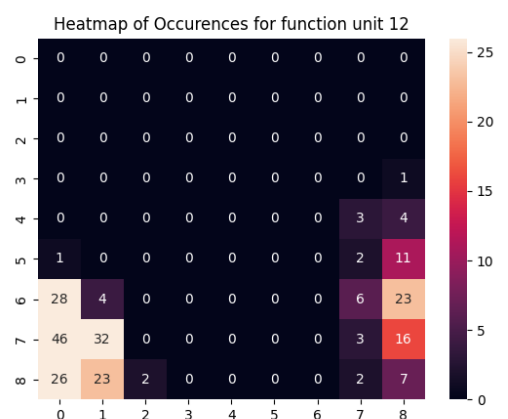


Figure 3.27: Heatmap of functional unit 12

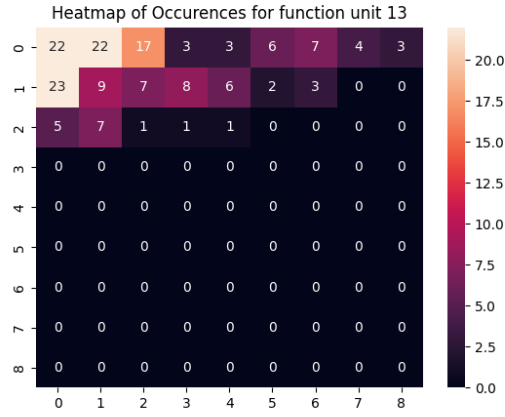


Figure 3.28: Heatmap of functional unit 13

When analysing the energies of the samples, the ratio of the minimum energy to the maximum energy was 1.008, negating the need to weigh the occurrences against their energies. This also proves that the current number of sweeps is optimal for consistent layouts with minimal variance. A quick analysis of the energies of the 80 samples is given below:

- Average computation time per sample = 85,458.88s which is approximately 23.7 hours.
- Minimum and maximum energy in the sampleset = 16,329.64 & 16,470.98. The difference between the two is 141.34, which is a difference of 0.87% relative to the minimum energy
- This negligible variance in energy proves that the results are consistent and repeatable.
- The hyperparameter values that generated the best results are $\alpha = 1$, $\beta = 200$, $\gamma = 300$

The sample with the lowest energy generated by simulated annealing is given in Fig.3.29.

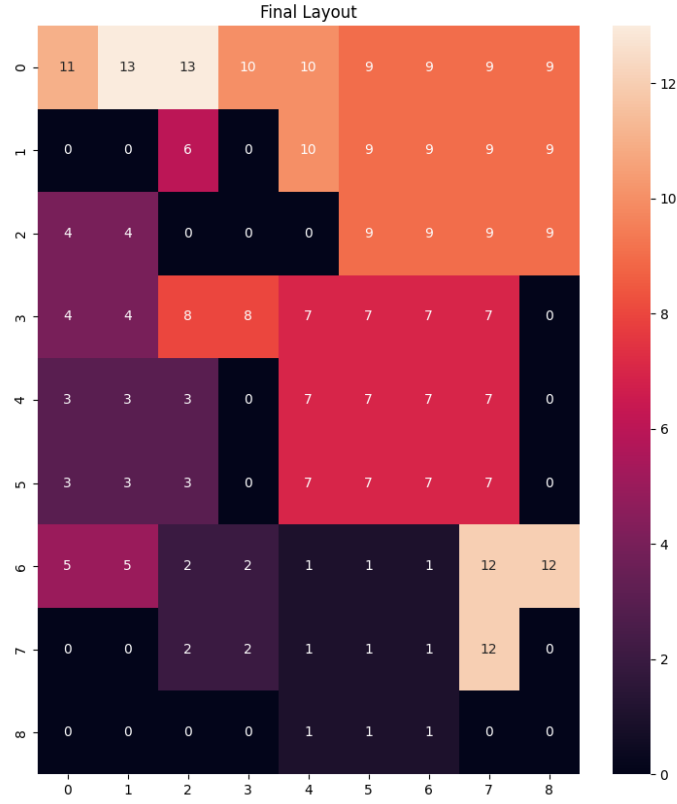


Figure 3.29: best solution generated by the simulated annealing solver

3.2 Quantum Annealing

D’Wave’s hybrid solver service allows us to set a time limit on it. As it uses a hybrid of classical and quantum algorithms, a certain time limit set is the total time taken for the classical and quantum process. It was found that 30 seconds of total hybrid solver time amounts to 1.5 seconds of total QPU usage with a variation of 1-2 milliseconds. The best solutions generated by the Hybrid Solvers for 3 time limits (20s, 60s, 90s) are given below.

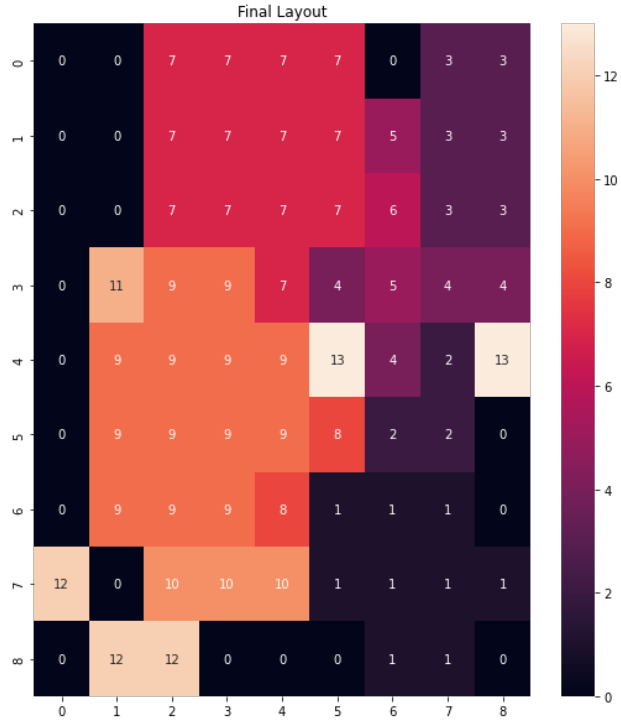


Figure 3.30: A solution produced by the hybrid solver with time limit = 20s

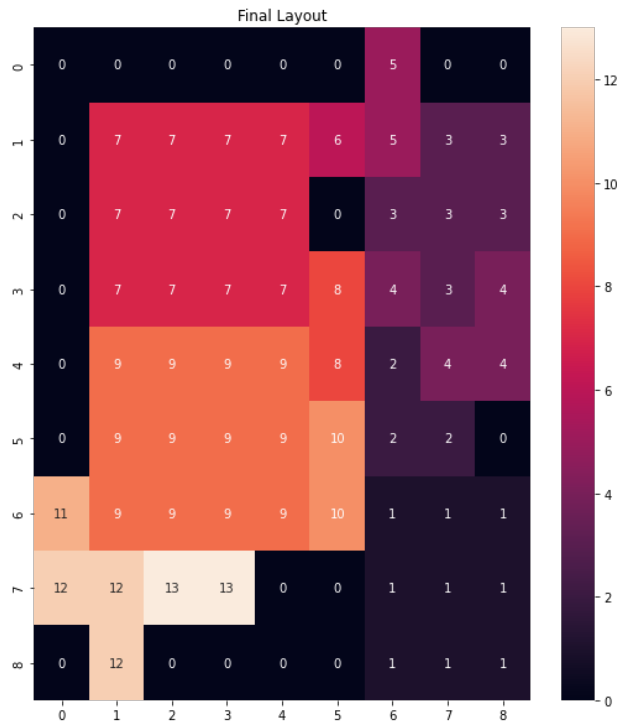


Figure 3.31: A solution produced by the hybrid solver with time limit = 60s

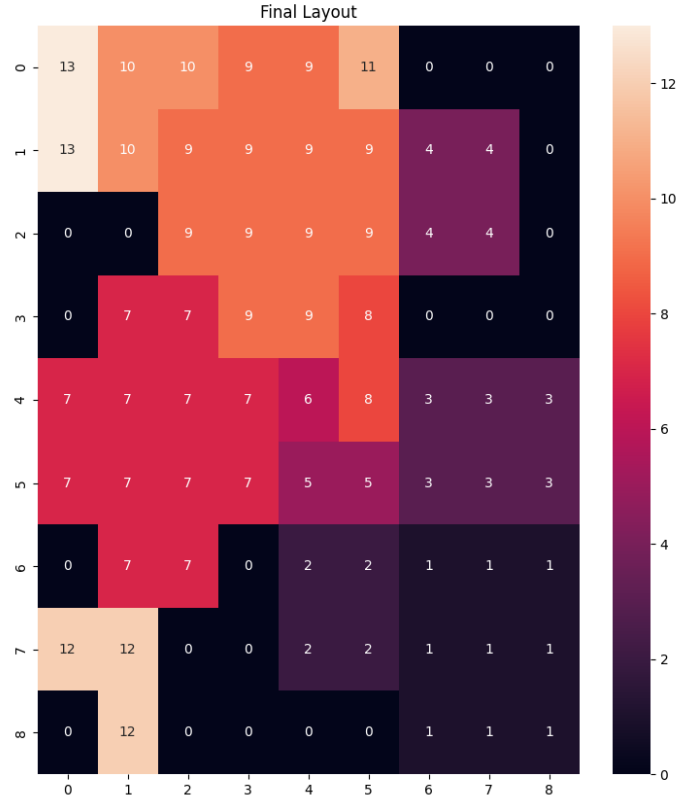


Figure 3.32: A solution produced by the hybrid solver with time limit = 90s

Due to access limitations, enough runs could not be submitted on the hybrid solvers for statistical averaging to have any meaningful impact. It is important not to set the time limit very high, as there is a possibility of the qubits decohering. An example of a layout produced when the QPU has decohered is shown in Fig.3.33.

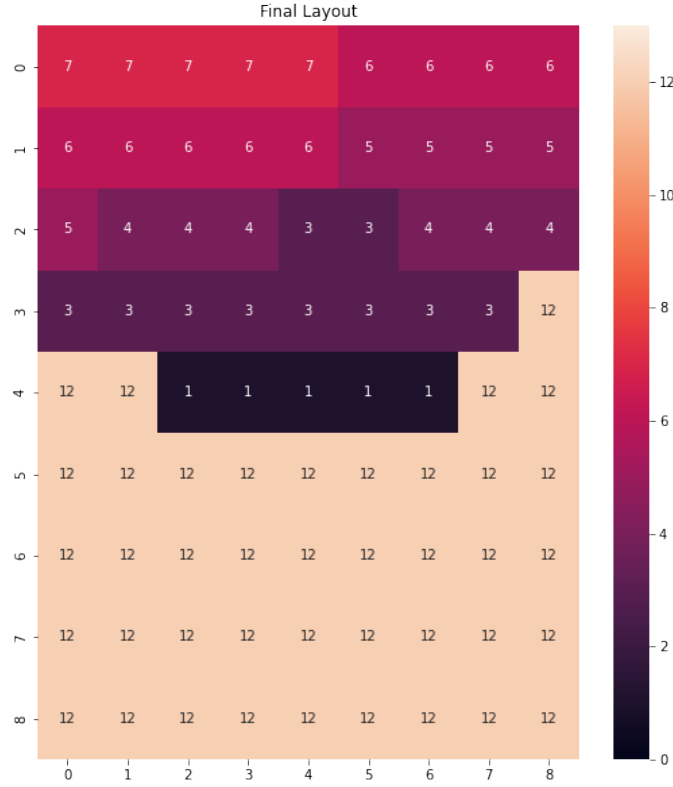


Figure 3.33: A solution produced by the hybrid solver with time limit = 240s

It can be seen that a 20s or 60s time limit does not always produce a viable solution, but 90 seconds produces a layout that closely follows the heatmaps generated by the simulated annealing solvers. The total QPU time taken for 20, 60, and 90 seconds of hybrid solver is 1, 3, and 4.5 seconds respectively, which is unequivocally better than the time taken for the simulated annealing run to take place. It is to be noted that the solutions generated by the hybrid solvers are not always the same. A few examples of the different (but viable) solutions generated by the quantum annealing solver (time limit = 90) are given in Fig.3.34 and Fig.3.35. These do not follow the layout described the simulated annealing solver, but they highlight the probabilistic nature of quantum annealing. Each of these rough layouts can be refined further to produce a valid and optimized layout of the factory.

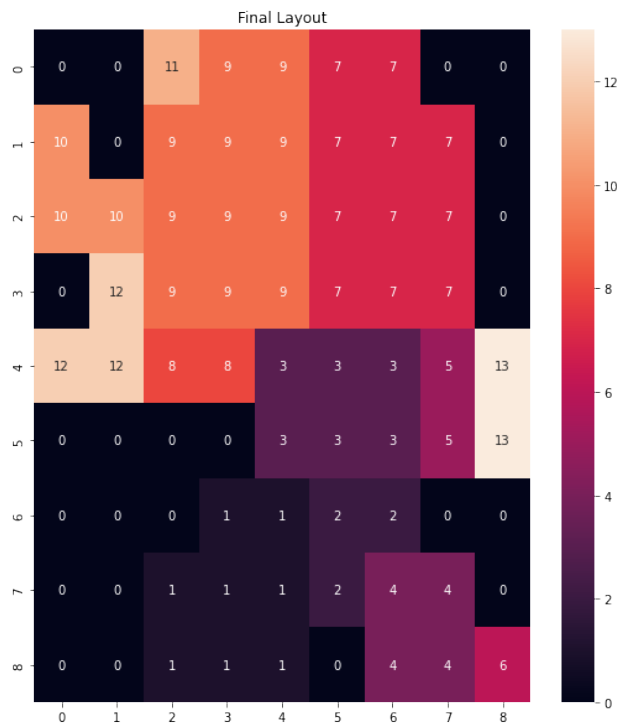


Figure 3.34: Alternate solution produced by Quantum annealing

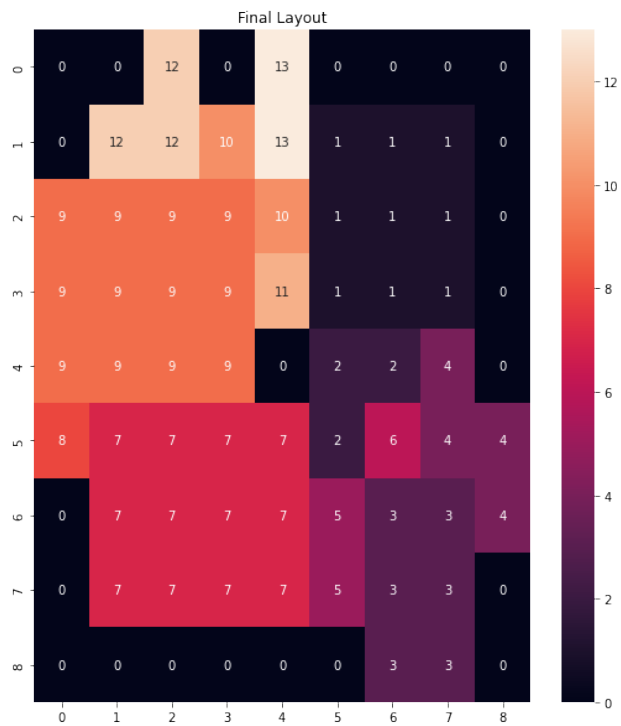


Figure 3.35: Alternate solution produced by Quantum annealing

Chapter 4

Discussion

4.1 Results

It is clear to see that the layouts generated by the hybrid solver follow the same pattern as the layouts generated by simulated annealing. The difference lies in the time taken for processing. The average time taken per sample for simulated annealing is roughly 85,500s seconds or around 23.7 hours. On the contrary, quantum annealing can produce layouts of a similar or even greater quality than simulated annealing in 90 seconds. Out of the 90-second total time limit on the hybrid solver, the QPU was active for only 4.5 seconds.

While the layouts generated by both simulated annealing and quantum annealing are not always physically feasible, it is obvious that functional unit 7 in Fig.3.2, for example, cannot be broken into the little pieces like the layout suggests. However, it is obvious that this layout intends to place functional unit 7 in the rectangle enclosed between positions 36 and 51. This breakage of functional units occurs due to the degeneracy of the Hamiltonian. A layout with functional unit 7 being broken by a single position and functional unit 9 being intact, will have the same energy with the breakage switched to functional unit 9 with 7 being intact. While both will have higher energy than a layout with no breakage, the probabilistic nature of quantum annealing and the physical limitation of qubits not being 100 isolated from their surroundings lead to the Hamiltonian sometimes ending up in a higher energy state. The statistical analysis of the layouts generated by the simulated annealing solver can be used as verification to confirm the results we get from the quantum annealer.

This is why qualitative analysis and further refining of the layout are necessary. This situational analysis is used to create a digital double of the factory.

4.2 Why Use a Hybrid Solver

In order to understand the physical limitations we currently face, we can look at a problem as a graph: where each decision variable, or qubit, is a node on the graph with a bias, and the edge strength when connecting two nodes is the coupling strength. When mapping - or "embedding" - a graph onto the QPU, it is not always enough to assign a singular qubit to each node. On a QPU, every qubit is not coupled to every other qubit. D'Wave refers to the number of other qubits a particular qubit is connected to, as the qubit's "degree". In D'Wave's Advantage QPUs, each qubit has a degree of only 15. This means that there is a possibility that there is no singular qubit that can be assigned to a node that has all the physical couplings that a node requires for a problem. This is tackled by assigning multiple qubits to a single node, essentially making the multiple qubits act as a single qubit. A problem such as ours, where each node of the graph is connected to every other node, is called a fully connected graph.

For our particular problem, where we have $81 * 13 = 1053$ decision variables or "logical qubits" in a fully connected graph, we have $^{1053}C_2 = 553878$ total connections. Our current technology comes nowhere near to being able to embed an entire graph of this size onto a QPU. This is why a hybrid solver is used, it uses classical algorithms to chop the problem into smaller parts and solve them on the QPU separately. The hybrid solver can solve problems with over 12,000 logical qubits.

Chapter 5

Conclusion and Future Prospects

5.1 Future Improvements

Currently, the biggest drawback of this model is that there is no way for the model to know what the actual shape of the functional unit is. Two functional units of the shape $4m \times 10m$ and $8m \times 5m$ will both have an area of $40m^2$. The model currently prioritises arranging all the pieces of the functional unit in a circle. Future improvements can incorporate the shape of the functional unit as well, by including the diagonal length by penalising any piece that is further than the diagonal distance away from the other distances. Or instead of using a quadratic Hamiltonian, a cubic Hamiltonian can be used to check if the piece in question is within the rectangle defined by the two edge pieces, and penalise the model if not.

Another possible improvement can be the creation of a better distance matrix. Instead of just using the geometric distance between the positions, we can measure the distance using the path that the model will actually be following, be it cranes, conveyor belts or any other mode of transportation a factory might use.

This work utilized a BQM model, but this problem could also be tackled by using a DQM model. A DQM model, or a discrete quadratic model, uses a discrete, finite set of real values as the decision variables. Instead of the combinations of functional units and positions as logical qubits, we assign a logical qubit to each position, and each position can collapse to one of 14 discrete values (13 functional units and an empty space). This reduces the

dimensionality of our X matrix, drastically reducing the number of logical qubits required. This does come at a cost, however, as now "empty space" is treated as a functional unit by the model and requires to have a rate of flow defined. A logical qubit can be used as a discrete variable by superposing multiple physical qubits; for example, our problem with 14 discrete values can be encoded in 4 qubits which provides 16 possible values with 2 values to spare. However, this again causes a stronger limitation on the size of the problem we can compute as it is added to the previous coupling limitations. However all these can simply be overcome using better quantum computers as our technology improves.

5.2 Conclusion

This work proves that quantum computing can provide significant computational benefits over classical methods, and that the same techniques that are used for the simulation of physical problems can be applied to real-world problems. In the coming years, as our technologies are refined and improved upon, and we are able to create quantum computers with even more qubits with longer decoherence times, we will unlock the potential to solve even larger problems in a time frame which is negligible compared to the time current classical simulations take.

It is obvious that a lot of factors that are going to be needed to create an accurate layout are ignored. For example, a factory that manufactures metallic parts almost certainly has a CNC machine, and it requires a constant supply of water during its operation. It is vital for the machine to be placed in a place where there is direct access to water. Realistically, there are then going to be only a few positions where it is viable to have the machine be placed. Unfortunately, the model is not capable of taking into account such intricacies. While this issue, in particular, can be tackled by simply removing the machine and the positions it will take up, from the model while the rest of the pipeline is optimized normally, it is not always going to be so easy. But, with some qualitative analysis and consulting with experts, quantum annealing can provide multiple different variations of layouts that we are sure to be near optimal. It is then going to be extremely easy to create a more refined digital double of the factory.

Another benefit that this model provides is that it does not care what the product being manufactured is, nor does it care about the size or shape of the factory. It takes minimal,

non-identifiable data, which further protects the manufacturer's trade secrets. This means that the same Hamiltonian has the flexibility to be applied directly to a multitude of factories, both manned and unmanned, and still be guaranteed to generate optimal layouts.

Bibliography

- [1] Malcolm Beaverstock, Allen Greenwood, and William Nordgren. *Applied Simulation: Modeling and Analysis Using FlexSim*. Publisher, Year. Chap. 3.2.
- [2] Yudong Cao, Jhonathan Romero, and Alán Aspuru-Guzik. “Potential of quantum computing for drug discovery”. In: *IBM Journal of Research and Development* 62.6 (2018), pp. 6–1.
- [3] Richard P Feynman. “Simulating physics with computers”. In: *Feynman and computation*. CRC Press, 2018, pp. 133–153.
- [4] Xujiao Gao et al. “Quantum computer aided design simulation and optimization of semiconductor quantum dots”. In: *Journal of Applied Physics* 114.16 (2013).
- [5] *Getting Started with D-Wave Solvers*. Accessed on 2023-03-20. 2024. URL: https://docs.dwavesys.com/docs/latest/doc_getting_started.html.
- [6] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: <https://doi.org/10.1145/237814.237866>.
- [7] R. Harris and et al. “Experimental investigation of an eight qubit unit cell in a superconducting optimization processor”. In: *Physical Review B* 82.2 (2010), p. 024511. DOI: 10.1103/PhysRevB.82.024511.
- [8] Ciaran Hughes et al. “Introduction to Superposition”. In: *Quantum Computing for the Quantum Curious*. Cham: Springer International Publishing, 2021, pp. 1–5.
- [9] Quantum Inspire. *Superposition and Entanglement*. n.d. URL: <https://www.quantum-inspire.com/kbase/superposition-and-entanglement/>.

- [10] Brian D. Josephson. *The Discovery of Tunneling Supercurrents*. Nobel Lecture. Dec. 1973.
- [11] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.
- [12] Tony Manos. “Value Stream Mapping—an Introduction”. In: *Quality Progress* (2006). via University of Washington, p. 64.
- [13] Michael A. Nielsen and Isaac L. Chuang. “Quantum Computing and Quantum Information”. In: 10th Anniversary Edition. Cambridge University Press, 2000. Chap. 1.
- [14] Gireesh Pandey and S.K. Pal. “Polynomial selection in number field sieve for integer factorization”. In: *Perspectives in Science* 8 (2016). Recent Trends in Engineering and Material Sciences, pp. 101–103. ISSN: 2213-0209. DOI: <https://doi.org/10.1016/j.pisc.2016.04.007>. URL: <https://www.sciencedirect.com/science/article/pii/S2213020916300210>.
- [15] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/S0097539795293172. URL: <http://dx.doi.org/10.1137/S0097539795293172>.
- [16] M. Steffen. *NMRQC - Nuclear Magnetic Resonance Quantum Computing*. <https://web.physics.ucsb.edu/~msteffen/nmrqc.htm>. Accessed: March 2024.
- [17] *What is Quantum Computing? — IBM*. <https://www.ibm.com/topics/quantum-computing>.