Design Science

Comparing Gate and Annealing-based Quantum Computing Approaches for Engineering Design Applications

Oliver Schiffmann[1], James Gopsill[1], and Ben Hicks[1]

[1] *Design and Manufacturing Futures Lab, University of Bristol, Bristol, UK*

Abstract

The computing capabilities of classical systems are approaching their limits, especially when confronted with increasingly intricate engineering design challenges. Conventional methods, such as search heuristics, encounter difficulties as design spaces expand in complexity and scale. This study explores the potential of quantum computing, focusing on the gate-based Grover's algorithm and Quantum Annealing, to overcome these limitations. Using a generalised tile placement problem, the research compares the benefits of each quantum computing approach, considering their implementation on real quantum hardware. Results indicate that Quantum Annealing produces usable solutions efficiently, while the Grover's algorithm approach is heavily impacted by noise.

Abstract diverges from new work here. Should include that QA can outperform a simple brute force approach at some problem scales investigated Both quantum approaches are surpassed by a simple classical brute force method.

Back to recent work here The findings suggest that gate-based approaches may not yield practical results in the near term, emphasising the promise of Quantum Annealing for engineers today. The paper's significance lies in the comparison of two quantum approaches, evaluating their physical implementation on hardware, outlining the process, and offering examples of problem reformulation for engineering designers utilising quantum computing **Perhaps another contribution has been made now?**. The study provides valuable insights and recommendations for the near-term utilisation of quantum computing in engineering design.

Corresponding author
Oliver Schiffmann
oliver.schiffmann@bristol.ac.uk

Des. Sci.
journals.cambridge.org/dsj
DOI

Introduction

50 years ago, Engineering Design relied heavily on manual calculations, physical prototypes, and intuitive decision-making. The design process was time-consuming, limited in scope, and heavily dependent on the expertise and intuition of the designers.

Today, the reliance on analytical hand-calculated solutions has significantly diminished as tools such as computational fluid dynamics (CFD) and finite element analysis (FEA) software have become standard practices. These digital tools allow engineers to visualise, manipulate, and iterate on designs in a virtual environment, saving time, resources, and reducing the need for extensive physical testing. Moreover, engineers now have access to powerful algorithms, simulations, and modelling techniques that enable them to explore a vast design space and optimise solutions with unprecedented precision. Examples include: thermomechanical analysis using FEA Belhocine and Abdullah [2020], generative design for structural optimisation [Gonzalez-Delgado et al., 2023], and cost-benefit analysis of intelligent supply chains [Schiffmann et al., 2023]. Computational methods have become indispensable tools and have revolutionised the process of designing and optimising solutions. A consequence of this evolution is that the design process can now be characterised as a data-driven decision-making process.

All these methods enable designers to discover key information about their problem and its potential solutions. This information ranges from how sensitive a design is to changes in design and/or scenario parameters, what are the number of theoretical solutions, and how optimal is the current solution? The role of the designer is therefore shifting from the generation of solutions to the definition/constraint of design spaces that can then be explored computationally more quickly and to a much fuller extent Biskjaer et al. [2014]. However, no matter how well our computational methods perform, designers are always looking to increase the fidelity and expanse of the design space they are exploring. Never satisfied, the primary objective remains to reduce uncertainty and increase confidence in the design taken to production.

This increasing complexity stems from the desire to capture a more holistic and realistic view of the design space, incorporating system-level interactions and trade-offs between competing design objectives. This enhanced fidelity is essential for optimising designs with complex performance, safety, cost, societal and environmental criteria. This growing complexity is exemplified in the design of modern integrated circuits. Integrated circuits used to contain a handful of transistors, and their design could be handled manually using relatively simple computational tools. However, today's integrated circuits can contain trillions of transistors [Lécuyer, 2022], and their design requires sophisticated Computer-Aided Design (CAD) software. This increase in transistor count has also led to a dramatic increase in the number of design variables, constraints, and objectives that must be considered.

As problem complexity increases the advantage classical computation methods provide plateaus. This limitation primarily stems from the inherent complexity in computationally representing, resolving, and exploring design spaces. This increasing complexity becomes a limitation as we reach the upper end of our manufacturing process limits for classical processors. Despite increasing numbers of transistors the clock speed of classical computers is capped at around 5Ghz [Sutter et al., 2005]. This indicates we have almost exhausted our vertical scaling potential. As we work in this processing-speed-limited world, understanding and addressing the challenges with problem complexity is crucial to enhance the design process. Key challenges relevant engineering design include:

- **Optimisation in High-Dimensional Spaces:** Design problems often involve a vast number of variables and constraints, resulting in high-dimensional solution spaces. Navigating and optimising these spaces efficiently is a complex task, requiring advanced algorithms and computational power [Gopsill et al., 2022].

- **Trade-offs and Multi-Objective Optimisation:** In many engineering design scenarios, there are conflicting objectives that must be carefully balanced. Achieving the optimal trade-offs between parameters such as cost, performance, reliability, and sustainability presents a significant challenge [Sun et al., 2018].

- **Exploration of Design Space:** Traditional design approaches often rely on human intuition and experience, limiting the exploration of alternative design options. Expanding the search space to consider a wider and more diverse range of solutions is essential to discovering globally optimal designs.
- **Computational Bottlenecks:** As design complexity increases, so does the computational cost and time required to perform detailed analyses, simulations, and optimisations. This leads to longer design cycles and potentially hampers the ability to iterate and explore design alternatives.
- **Incorporating Uncertainty:** Engineering design is subject to various sources of uncertainty, including material properties, environmental conditions, and manufacturing variations. Effectively addressing and quantifying uncertainty is vital for robust design decisions [Schiffmann et al., 2023].

To illustrate, a simple 3-stage gearbox design problem is considered by Gopsill et al. [2022]. The design parameters for this system are as follows:

- 17 gear options for the 6 gears,
- 5 materials for the 4 shafts,
- 9 bearing options for the 8 bearings, and
- 8 electric motors for the motor.

While only a relatively small number of options exist for each component choice, the resulting solution space features 5.18×10^{18} possible design options. Evaluating all these solutions would require 164 years if each design option were able to be evaluated in a single clock cycle on a 1GHz processor. This demonstrates the coupling between computational complexity and the scale of design space although there are some subtleties in that not every design option taking the same time to compute and whether one needs to compute every option to identify the optimum.

Further, having evaluated a set of options, it may be be useful to store the results for later analysis or selection. Assuming 5 bits are required to store a gear option, 3 bits for the material, 4 for the bearings, and 8 for the motors then a total of 15 bits or 2 bytes is required to store a single design option. Multiplying this but the number of possible options yields the storage required to capture the entire design space – 1.036×10^7 TB. With modern hard disk drives storage capacity in the 10s of TBs we are quickly diverging from what is capable with classical computers.

The gearbox selection task exemplifies the challenge of navigating vast design spaces, even when the computation of a single design options remains trivial. There is also the challenge of the computational time to evaluate a single design option. Work that exemplifies this problem is concerned with the design of winding strands in electrical machines - specifically their lay and how that affects AC losses [Mellor et al., 2021, Hoole et al., 2021]. Mellor et al. [2021], Hoole

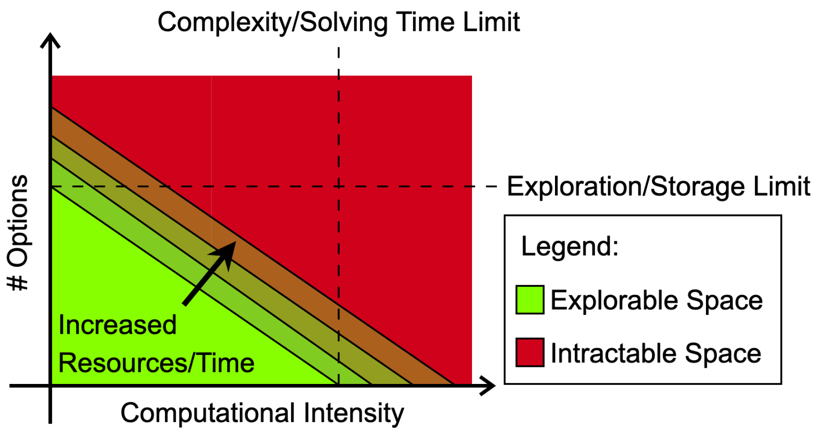


Figure 1: A figure showing how the type of problems we can tackle using classical computation are bounded by the number of options we can compute and the complexity of evaluating a single option.

et al. [2021] discuss the development of a conductor placement algorithm. For the algorithm to fill a single slot with approximately 130 conductors a run time of around 3 hours is required. This is for a single slot of which there will be multiple in an electric machine. Further, this is for a single set of design parameters. Should the designer wish to explore even a small number of design options this task quickly becomes intractable.

Together these form the space of problems we can tackle with classical computation. This is shown in fig. 1 where the green space represents the problem space we are bound to by the limits of classical computation.

These challenges are well known and often encountered in Engineering Design. As a result, research has developed a variety of classical methods to help optimise the exploration of design spaces. Some examples are gradient descent methods, evolutionary algorithms, particle swarm optimisation, and generative approaches. However, these each face their own issues as design spaces become increasingly complex. For example gradient descent struggles with convergence to local optima and traversing discontinuous design spaces [Gopsill et al., 2021].

When it comes to computationally intense single problems, classical computation faces several challenges. These are exemplified in the context of finite element solvers. Mesh optimisation is the process of refining or adapting a computational mesh to better capture the geometric and physical features of a problem domain. As the complexity of the problem increases, the number of elements and therefore the number of degrees of freedom in the mesh grows exponentially, resulting in a substantial increase in computational requirements [Zandsalimy and Ollivier-Gooch, 2022]. This scaling limits the ability of our computational tools designed for evaluating a single complex scenario such as FEA. This scaling phenomena is discussed in [Filipovic and Selberherr, 2020].

In both cases deep-domain understanding is required to to achieve the quality of results expected today. For example, knowledge of the appropriate mesh elements, their size, and locations where increased fidelity is need is important when using FEA. For evaluating many options, awareness of the different particle swarm, evolutionary, or slime mould derivative algorithms is needed. This understanding restricts their general use.

The approach being examined in this paper is Quantum Computing (QC), which has emerged as a promising method that overcomes some of the barriers faced by classical methods. The field of QC remains an evolving field, within which there are a variety of techniques being developed. Quantum computers are computers that use quantum phenomena to represent information. The smallest unit of information represented in a quantum computer is a quantum bit - often referred to as a qubit. While classical computers process information in binary states (0 or 1), qubits can be in a state of 0, 1, or a superposition of both. Superposition enables quantum computers to perform computations in parallel (within the same qubits) and explore multiple solutions simultaneously. This superposition is represented mathematically as

$$|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha |0\rangle + \beta |1\rangle \tag{1}$$

where $|\phi\rangle$ represents the superposition state created by superimposing the two binary states $|0\rangle$ and $|1\rangle$, α and β are their linear coefficients, respectively. Operations can then be performed on a state, $|\phi\rangle$, as part of a computational process. $|\phi\rangle$ can eventually be measured to extract an answer from the quantum computer. This answer would be one of the basis states $|0\rangle$ and $|1\rangle$ and the probability of each being returned would be $|\alpha|^2$ or $|\beta|^2$, respectively [Yingchareonthawornchai et al., 2012].

Taking advantage of superposition, as well as other QC features such as entanglement, has resulted in a number of quantum algorithms for tackling generic computational problems. These play a crucial role in the current landscape of quantum computing research, offering the prospect of exponential speedup compared to classical counterparts. Among these, Grover's Algorithm for unstructured search stands out for its quadratic acceleration in database searching. Similarly, Shor's Algorithm, designed for integer factorisation, holds significant implications for cryptography [Nielsen and Chuang, 2001].

These quantum algorithms are primarily designed within the framework of gate-based quantum computation. In this paradigm, quantum bits, or qubits, are manipulated through quantum gates, analogous to classical bits and logic gates in classical computing. An alternative to gate-based quantum computation is Quantum Annealing (QA), an approach that leverages quantum entanglement and potentially another quantum phenomena, quantum tunnelling [Hauke et al., 2020, Venegas-Andraca et al., 2018]. In QA, an energy landscape is defined through the use of penalty functions. In this landscape, more optimal solutions have lower energies (less optimal ones experience higher positive penalties). QA can be thought of as an analogue form of computation [Yang et al., 2023] as it involves running a physical process that evolves over time and is described by a Hamiltonian. A Hamiltonian is a mathematical construct that represents the time evolution of quantum states. The Hamiltonian, $H(t)$, that represents the energy landscape during QA is shown in Eq.

$$H(t) = A(t)H_0 + B(t)H_1 \tag{2}$$

where $A(t)$ and $B(t)$ are values varied from 0 to 1 and 1 to 0 over the course of the computation, respectively, and H_1 is some initial Hamiltonian in its ground state and H_0 is the Hamiltonian whose ground state corresponds to an optimal solution (lowest energy in the energy landscape) [Hauke et al., 2020]. If this evolution is slow enough, or adiabatic, such that no energy is added then the final solution (encoded in H_0) should be the global optimum.

QA is intended to explore solution spaces for combinatorial optimisation problems. These types of combinatorial design problems (like the gearbox example provided above) are present across a variety of engineering sectors. As such, QA is an interesting avenue of exploration for engineering designers seeking to overcome the limitations of classical hardware - despite the non-universality of appropriate problems.

Further, QA systems have matured considerably in recent years [Hauke et al., 2020]. This hardware advancement affirms QA as a suitable area for exploration in this Noisy Intermediate-Scale Quantum (NISQ) era. NISQ devices are those devices who, whilst being able to outperform some classical approaches, suffer from substantial noise in their results [Dasgupta and Humble, 2020].

An important feature of quantum computing research is the recognition that both gate-based and annealing methods hold promise in the near and long term. This realisation underpins the importance of exploring all avenues until a dominant method emerges such that we can take full advantage when/if it does. The field is still in its infancy and development of hardware is rapid. Many approaches exist, including those just discussed, as well as hybrid versions - approaches that combine classical and quantum processing. For engineers a central challenge emerges: how to represent problems effectively to exploit the unique strengths of both gate-based and annealing approaches. Further, a gap exists in the current research that specifically addresses how to implement the developing approaches using the existing hardware.

The contribution of this work are insights into the readiness of the different hardware options for an abstracted engineering design problem. This insight could be used to eliminate the investigation of certain options for unsuitable problem architectures, as well as provide examples for how problems can be refactored to be quantum-solvable. In addition a methodology is laid out for applying two QC methods to an engineering design problem. To achieve this aim several objectives were laid out:

- Develop a case study problem that can be used to evaluate the advantages and disadvantages of each approach.
- Build on author's previous work by running a Grover's approach to tiling problem on a real QPU.
- Explore the potential for QA by constructing the problem for a D-wave device and obtaining results
- Compare the approaches based on difficulty formulating the problem, time to solution, noise level of results, and the quality of results.
- Draw conclusions about the usefulness of each technique for engineering designers.
- Identify directions for further research.

Related Work

In this section the existing literature related to this work is discussed. The value of this discussion is the demonstration of the gaps existing in the field of QC and its applications, as well as a justification as to why this is the correct time to begin investigating QC for engineering design. These related works show that there is a lot of theoretical work demonstrating the potential for QC, and a few recent examples of real quantum achieved results. Further, these works help to show the main directions that are being explored and which options went unexplored as part of this work.

QC and particularly its applications remains a young yet rapidly maturing field [Gill et al., 2022, Mahmoudi et al., 2022] with the earliest works dating 2001 [Williams, 2001]. The majority of research relating to QC and its applications fit into one of four categories: dedicated gate-based, dedicated QA based, QML based, or a hybrid/variational approach combining classical computation with one of the other categories.

Recent summary papers have looked at how this range of techniques can be applied to different fields or industries. Ullah et al. [2022] discuss the various methods through which QC might help manage the increasing computational complexity of smart grids. This increasing complexity arises as a result of our shift away from fossil fuels to more Distributed Energy Resources (DERs) such as photovoltaic cells, wind turbines, and electric vehicles. The non-linearity and uncertain nature of these DERs increases the complexity of decision making at every level of smart grids. The algorithms discussed include dedicated gate-based approaches such as the quantum HHL algorithm (for tackling systems of linear equations), hybrid approaches such as the Variational Quantum Eigensolver algorithm (a hybrid quantum-classical computational approach for finding the ground state of a Hamiltonian), QA approaches for minimising objective equations, and Quantum Machine Learning (QML).

Each of the aforementioned categories has seen varying degrees of individual exploration and implementation. For example, Mahasinghe [2019] report a gate-based version of the HHL algorithm for solving the equation of motion of an undamped and nonelastic rotating system. The algorithm would reportedly provide an exponential time saving when compared to classical approaches thereby enabling designers to either evaluate a greater number of options, or more complex scenarios, for the same resource cost. The work demonstrates the potential for QC to help overcome limits in classical computation and it does so for an engineering context. However, it does not discuss practical implementations and therefore how the expected speedup may vary when implemented on a real quantum device is unknown.

Examples of work exploring the use of simulated QML include [Correa-Jullian et al., 2022, Li et al., 2022]. Both were looking at fault diagnosis in wind turbines and roller bearings, respectively. Their QML approaches improve the ability of machine learning algorithms to process large amounts of data. Li et al. [2022] state that QC hardware with sufficient coherent times are not available at present, and so explore a simulated approach to demonstrate the feasibility of their Quantum Support Vector Machine (QSVM) approach. Correa-Jullian et al. [2022] also explore a simulated quantum approach and compare that to modern a classical approach implemented on modern GPU hardware.

Ray et al. [2022] are one of the exceptions who discuss practical imple-

Table 1: A table summarising each of the related works discussed in this section.

Ref.	Engineering Problem	Approach
Ullah et al. [2022]	Increased complexity in smart grids due to DERs	Summary of available QC techniques
Mahasinghe [2019]	Solving the EoM of ratating systems	Theoretical gate-based HHl algorithm
Correa-Jullian et al. [2022]	Fault detection in wind turbines	Simulated QML
Li et al. [2022]	Fault detection in roller bearings	Simulated QML
Ray et al. [2022]	1-D fluid flow	QA implemented on D-wave device
Zhang et al. [2022]	Welded beam, spring, and pressure vessel design	Variational SMA

mentation of QC. They investigated using QA on a D-wave device to solve a 1-dimensional fluid flow problem. In their paper, they present real quantum computed results and compare their accuracy against classical results. This may indicate the annealing devices are more mature as of today and provide a better option for gaining quantum advantage in the near-term [Leymann and Barzen, 2020].

Zhang et al. [2022] has explored variational algorithms to tackle specifically engineering design problems. They investigate performance on three typical engineering design problems: welded beam design, spring design, and pressure vessel design. They discuss improving the well known Slime Mould Algorithm (SMA) by augmenting it with quantum computation – specifically a dynamic quantum rotation gate. What this means is that part of the algorithm involves using some early results and classical processing to tune the parameters of a quantum gate in between iterations, based on those results. This allows for an iterative algorithm set-up that improves its ability to find good solutions as the problem is explored. They claim that this has allowed them to produce an algorithm that outperforms the original, classical version of the SMA in global search capability.

From the examples presented in this section, it can be seen that there are a multitude of promising quantum techniques applicable to the field of engineering design. The approaches explored are summarised in Table 1.

Experimental Setup

The components steps of the experimental setup can be seen in Fig 2. Step 1 established a trial engineering design problem on which the performance of the QC approaches could be judged.

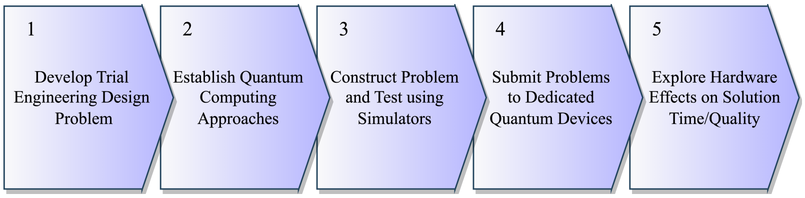

1
Develop Trial Engineering Design Problem
2
Establish Quantum Computing Approaches
3
Construct Problem and Test using Simulators
4
Submit Problems to Dedicated Quantum Devices
5
Explore Hardware Effects on Solution Time/Quality

Figure 2: A Fig showing the major steps to be executed as part of the experimental process.

Ideally this problem would be a generalised version of many different engineering design problems such that any leanings presented during this paper can be relevant to as many different cases as possible. For this work, the authors chose to continue using a constraint–based tile placement problem first presented in Gopsill et al. [2021] and then developed in Gopsill et al. [2022]. The problem is analogous to many engineering design problems, including:

- **VLSI Chip Design:** In Very Large Scale Integration (VLSI) chip design, engineers need to place numerous electronic components like transistors, logic gates, and interconnects efficiently on a silicon wafer. The objective is to minimise the size, power consumption and maximise the performance of the chip [Held et al., 2011].
- **Factory Floor Layout:** In manufacturing, optimising the layout of machines and workstations on the factory floor is crucial for efficiency and productivity. Engineers need to consider factors such as workflow, material handling, and safety [Ripon and Torresen, 2014].
- **Optical Fibre Network Design:** In telecommunications, engineers design optical fibre networks by deciding where to lay fibre optic cables to connect cities or regions efficiently. This involves minimising the total cable length while considering geography [Ranaweera et al., 2019].
- **Placement of Sensors in Environmental Monitoring:** In environmental monitoring, the placement of sensors (e.g., for air quality or weather) is critical. Engineers need to determine the best locations to obtain accurate data while minimising costs [Al-Turjman et al., 2013].

From these some common themes that can be extracted to help us form realistic constraints and objectives. These commonly include but are not limited to:

- Certain positions are out of bounds for placement (a component or feature already exists there in the chip or factory, some prohibitive geography prevents laying cable or sensing is not allowed).
- Things cannot occupy the same space (components, features, facilities cannot occupy the same physical space or use the same resources).

We can therefore define our tiling problem as shown in Fig 3. The problem has two constraints. 1. Both tiles must be placed on the eastern wall of the grid. 2. The tiles cannot be placed in the same position on the grid. This problem setup results in $64^2 = 4096$ potential solutions with 56 unique valid solutions. Note that the tiles are not considered to be identical, so if they switch places that would be another unique solution. Step 4 of Fig 2 can be executed using the 8x8 version of the problem shown in Fig 3. However, to see how performance versus a classical approach varies at different scales this problem will be investigated at increased grid sizes - 16x16, 32x32, and 64x64.

In this paper the authors perform an investigation into two promising avenues for QC's implementation into engineering design. Namely, these are QA and some gate-based algorithm (Fig 2 step 2). QA performs well when tackling combinatorial optimisation problems such as this tiling problem. For the gate-based approach, the authors chose to develop the approach presented in [Gopsill et al., 2022] that uses Grover's algorithm for unstructured search. These two approaches can then be compared against a classical method, such that

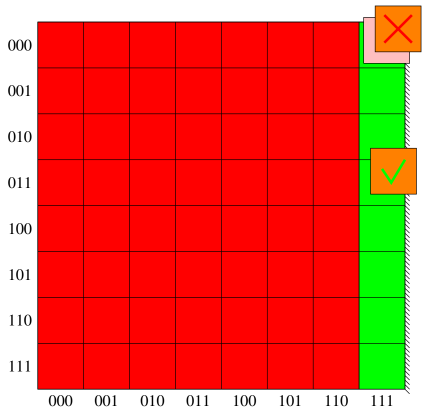


Figure 3: A figure showing the combinatorial tiling problem with two tiles (pink and orange) and their valid positions.

conclusions about the readiness of QC for use in engineering design, as well as the suitability of the different methods for this type of problem, can be drawn.

Two quantum approaches are investigated as part of this work, but results are collected for five approaches in total. These are a simulated gate-based approach, a real-quantum gate-based approach, a simulated QA approach (different from the classical Simulated Annealing (SA) algorithm), a real-QA approach, and a classical brute-force approach. The simulated quantum approaches will allow us to identify if anything in the results is a result of hardware or problem formulation (Fig 2 step 3). The brute-force approach works by simply iterating through every solution and checking if it meets the two constraints. This way it can be certain the classical approach would have found every valid solution.

Once the problem's construction for each approach has been validated using the simulators it can be submitted to the respective dedicated devices (Fig 2 step 4). This will provide preliminary results to determine whether the approach shows promise for near-term use. Should the results returned be usable (meaning they can be used to determine valid solutions that meet the constraints) then further investigation into how the quantum methods perform at different problem scales can be conducted.

The performance of the two QC approaches will be determined by looking at three factors: time-to-solution, error levels in the results, and the closeness of the valid results to a uniform distribution. Time to solution is important as a major barrier faced by classical computation is a limitation on the number or in the complexity of solutions that can be explored in a given time. The time to solution will be made up of two distinct components: processing time on the QPU or CPU (for quantum or classical approaches, respectively) and queue time. These can then be summed to find the total time for job completion. These timings can be found using the job managers provided on IBM's quantum platform [IBM, 2023a] and D-wave's Leap Dashboard [D-wave, 2023c]. The CPU times for the locally run classical approaches were achieved using an M1 pro chip (approximate clock speed of 3.2GHz).

Jobs used to investigate time-to-solution, error, and uniformity will be repeated 50 times, and mean values reported. This is to reduce the impact of the inherent variability in quantum achieved results on any observations made. The classical approach will also be repeated 50 times and averaged to find a mean time.

For this paper, error is defined as the percentage of solutions returned that do not meet the constraints. The number of solutions violating each constraint can be divided by the total number of returned solutions. This will give an error level for each constraint, which can then be combined to obtain the total error. Error levels provide an insight into the readiness of the chosen hardware for the scale and type of problem considered in this paper. High levels of error would indicate that the delicate quantum state used to represent information is being affected too much by the environment.

Finally, the distribution of valid results (those in which both tiles are placed on the eastern wall) will be compared to a uniform distribution. This is done using the coefficient of variation. This is calculated by dividing the standard deviation of the frequencies for valid positions by the mean. A lower value indicates that the results are closer to uniformly distributed. Understanding how close the results returned by a quantum computer are to a uniform distribution is useful for several reasons. In quantum computing, jobs submitted to these devices often undergo multiple executions due to inherent noise and errors in the hardware. Examining the closeness to a uniform distribution enables us to assess whether there are any systematic biases present within the quantum computer. By analysing this closeness, we gain insights into the reliability and accuracy of the quantum device for specific problem scales. Additionally, it helps us determine the number of times we may need to sample the results to effectively mitigate these biases.

A Gate-based Approach using Grover's Algorithm

Grover's algorithm presents a quantum search through an unstructured database with potential operations reduced from $O(N)$ to $O(\sqrt{N})$ [Nielsen and Chuang, 2001]. An example of the impressive potential improvement offered by a quantum search algorithm is given inNielsen and Chuang [2001]. If you were given a map of many cities and wished to find the shortest route passing through all of them, one option would be to search all possible routes through every city whilst maintaining a record of the shortest route so far. On a classical computer, if there are N possible routes then it would take $O(N)$ operations. Grover's algorithm requires only $O(\sqrt{N})$ operations.

The approach is typically applied using gate-based quantum computers. Operations in such devices can be depicted using circuit diagrams, which are graphical models that depict the evolution of qubits through a series of quantum gates. An example can be seen in Fig 4. A quantum gate is a fundamental operation that manipulates the state of a qubit, analogous to classical logic gates. For instance, the Hadamard gate, $\boxed{H}$, creates quantum superpositions, and the Controlled NOT (CNOT) gate, $\boxed{\oplus}$, enables entanglement between qubits. The circuit combines these gates, resulting in a procedure that transforms the initial state (of qubits) into a final state, which encodes a potential solution to the problem Nielsen and Chuang [2001].

The circuit reads from left to right and each "wire" represents a qubit as it evolves during the execution of the algorithm. The circuit shown in fig. 4 Nielsen and Chuang [2001] shows how a quantum state, represented by ψ can be moved or "teleported" from the top qubit to the bottom qubit. At the end of the circuit

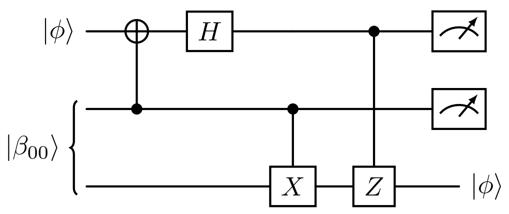


Figure 4: A Fig showing the quantum teleportation circuit as demonstration of the quantum circuit model. **It would be good to add some labels to this diagram i think**

two measurement gates can be seen, which collapse the qubits into classical bits. In Fig 4 two additional gates, $\boxed{X}$ and $\boxed{Z}$, can be seen. These are the Pauli-X and Z gates respectively. Pauli-X is the quantum equivalent of the NOT logic gate and Pauli-Z is often described as a phase-flip. In Fig 4 these are actually controlled versions of the Pauli gates - meaning they only apply certain control qubits are in specific states. Note that the CNOT gate is the same as a controlled Pauli-X.

The implementation of an algorithm such as Grover's is the process of representing your problem using a number of qubits in an array of states. Then the correct sequence of gates should be applied to modify the state of those qubits so that when they are eventually measured, they exist in a state with a high probability of collapsing into a state that encodes an optimal answer to your problem. Note that there is only a high probability (not a certainty) of obtaining this optimal solution.

Fig 5 presents a high-level representation of Grover's algorithm. The algorithm makes use of a register of n qubits (problem qubits) and a second register of computational bits (or the oracle workspace). The problem qubits are used to represent the problem variables and encode the solution to our problem when measured. The computational bits allow computations to be performed on the problem qubits. The circuit begins with all the problem qubits in the ground state $|0\rangle$ then applies a Hadamard gate to each wire, or qubit. This places each qubit in a state of equal superposition between the states $|0\rangle$ and $|1\rangle$, shown in eq. (3).

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{3}$$

The square of the coefficients of the states represent the probability of that state being returned $(1/\sqrt{2})^2 = 1/2$). Once in this state of superposition, the circuit undergoes its first iteration of Grover's algorithm, G. This iteration can be broken up into four steps [Nielsen and Chuang, 2001]:

1. Apply the Oracle. The Oracle is another combination of more elementary gates whose purpose is to mark or recognise a solution to the problem. This marking is carried out by by flipping a qubit in the Oracle workspace (or in the computational qubit register) from $|0\rangle$ to $|1\rangle$ if a solution is detected.

2. Apply a Hadamard to each qubit in the problem qubits register.

3. Perform a "phase shift" which has the effect of increasing the amplitude of

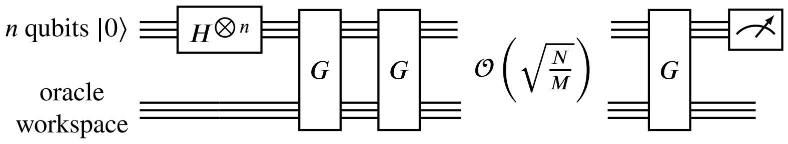


Figure 5: A figure showing the schematic circuit for Grover's algorithm [Nielsen and Chuang, 2001].

the target state, or the probability of it being returned.

4. Apply a Hadamard to each qubit in the problem qubits register.

This G can be repeated many times, and the optimum number is a function of problem size. For problems where there are M valid solutions in a space of size N the Oracle, O, should be applied $O\sqrt{N/M}$ times. For our problem that is $\sqrt{4096/56} = 8.5 \approx 9$ times.

Implementation

The algorithm was implemented using the Python Software Development Kit (SDK) Qiskit [Qiskit, 2023]. Qiskit is an open-source toolkit for developing and compiling quantum circuits. These circuits can then be sent as jobs to be executed using the compute resources supported by the IBM Quantum Platform [IBM, 2023a]. Other SDKs exist (cirq, Quantum Devlopemnt Kit (QDK), TKET...) and there are other suppliers of quantum computing resources (Google, Microsoft, IonQ...) [Ullah et al., 2022]. The approach adopted for this work was chosen due to the author's prior experience with python, Qiskit, and IBM's quantum experience.

Before the construction of any quantum circuit, the representation of the problem variables must be decided. For the tiling problem discussed in Section the variables are the x and y coordinates of each tile. Since quantum gates in IBM's devices operate on the computational basis $|0\rangle$ and $|1\rangle$, which collapse to the binary digits 0 and 1, respectively, we can represent the problem variables in terms of binary digits. Since our placement grid begins as 8 cells long in each dimension we can represent each coordinate of a tile with 3 binary bits ($2^3 = 8$). With 3 digits for each coordinate, and two coordinates per tile, and two tiles total we need a problem register of 12 qubits. Therefore, a solution to the tiling problem will take the form

$$\underbrace{q_1, q_2, q_3,}_{x_1}\underbrace{q_4, q_5, q_6,}_{y_1}\underbrace{q_7, q_8, q_9,}_{x_2}\underbrace{q_{10}, q_{11}, q_{12}}_{y_2} \tag{4}$$

where q_n is the measured value from the nth qubit in the problem qubit register, capable of taking the value 0 or 1.

Now that the representation of variables using qubits has been decided, the circuit that will implement Grover's algorithm can be constructed. This circuit construction was taken from the tutorials provided by Qiskit [Tapia, 2023] and is discussed in Gopsill et al. [2022].

Once the quantum circuit has been created it can be sent to a device for execution. IBM offers two fundamentally different options - the simulator and the dedicated device. For this work the IBMQ_QASM_Simulator was chosen to obtain simulated (classically achieved) results and ensure that the quantum circuit was correct. This simulator is described by IBM as their general purpose simulator and it has access to 32 simulated qubits. For real quantum achieved results the IBMQ_Brisbane device was used. This device has access to 127 qubits and was selected as it was, at the time, the only device large enough to run Grover's algorithm available for free on the IBM Quantum Platform. This is because although only 12 qubits are required to represent the solutions to the tiling problem 6 additional computational qubits and one Oracle qubit are required bringing the total to 19. At the time of writing, two additional devices with 127 qubits have become available - IBMQ_Osaka and IBMQ_Kyoto. These devices differ in their values for Error Per Layered Gate (EPLG) IBM [2023b].

Devices are often referred to as samplers in the SDK documentation. A sampler is something that runs a solver multiple times and returns the distribution of results. Hence, the number of runs for the entire quantum circuit, not just G, should be chosen. 1000 runs was chosen as this kept the problem below the maximum allowable estimated run time, kept classical simulations of reasonable length, and provided a sufficient number of results to visualise the distribution - as there are only 56 valid solutions for the 8x8 problem scale.

Quantum Annealing Approach

QA is a heuristic quantum optimisation algorithm for solving complex combinatorial optimisation problems. While gate-based quantum computers are still in their early stages of development, QA devices have already reached an increased level of maturity, with the largest quantum information processing devices currently available being annealing devices [Albash and Lidar, 2018].

This capability makes QA a promising option for tackling real-world problems in engineering design. QA has shown promise in practical applications, with several successful implementations in industry reported by D-Wave Systems [D-wave, 2023e]. These successes suggest that QA is a viable tool for solving complex engineering problems, even in the absence of a demonstrated quantum speedup. The relative ease of use of QA algorithms, which do not require deep-domain expertise in the classical algorithms they aim to outperform [Hauke et al., 2020], further enhances their appeal for engineering applications.

Adiabatic quantum computation (AQC) is a theoretical framework for quantum computing that utilises the adiabatic theorem [Amin, 2009] from quantum mechanics to perform computations. This theorem states that if a quantum system starts in its ground state and the Hamiltonian of the system is changed slowly enough, the system will remain in its ground state at all times - Equation . This property of adiabatic evolution allows AQC to solve optimisation problems by transforming the initial Hamiltonian, which represents the starting problem, into a final Hamiltonian, which encodes the solution.

QA is a specific implementation of AQC, it is the generic name for quantum algorithms that use quantum mechanical fluctuations (quantum tunnelling) to search for the solution of an optimisation problem [Morita and Nishimori, 2008]. QA's classical counterpart, Simulated Annealing (SA), can be used to help understand the process. "Quantum tunnelling between different classical states replaces thermal hopping in SA." [Morita and Nishimori, 2008].

The main challenge for QA is evolving the system slowly enough (adiabatically enough) such that it does not jump from the lowest energy state to a higher energy state. The minimum disparity between the lowest energy state and the next lowest one is called the energy gap. Issues arise when this gap becomes so small such that the time required to avoid crossing it becomes in-feasibly long [Hastings, 2021]. There is not a general predicted speed up for QA as there is for Grover's algorithm, however, the required annealing time does scale inversely with energy gap size [Hauke et al., 2020]. As the complexity of the problem increases (more constraints, objectives, and variables) then the energy landscape can become more "rugged", the energy gap will shrink and the annealing time will need to increase, or more samples will need to be taken.

Problem formulation for QA using D-wave devices

With an understanding of how QA differs from gate-based approaches problem formulation for QA devices can be discussed. QA can be considered as a form of analogue computation and so the problem formulation can be dependant on the hardware chosen [Yang et al., 2023]. For this work, the QA devices offered by the company D-wave were chosen. D-wave provide instructional guides on how to use, formulate problems, and submit jobs to their devices. This implementation can be done in Python using their Ocean SDK [D-wave, 2023d].

There are three classes of problem that can be tackled when considering the whole suite of D-wave devices. These are Binary Quadratic Models (BQM), Discrete Quadratic Models (DQM), and Constrained Quadratic Models (CQM). Each of these can handle a different class of variable, binary, discrete, and integer, respectively [Developers, 2022]. However, to make use of a dedicated device you must formulate the problem as BQM.

BQMs are made up of two classes. These are quadratic unconstrained binary optimisation (QUBO) problems and Ising models. These are mathematically equivalent but some problems may see better results with one option. For this work the QUBO approach was selected. It is also possible to convert between the approaches once the problem has been formulated. The QUBO approach followed for this work was taken from the resources provided by D-wave in their "Problem Formulation Guide" [D-wave, 2023d]. It should be noted that problem formulation is restricted to QUBOs instead of higher-order Polynomial Unconstrained Binary Optimisation problems (PUBO) due to experimental devices only being able to handle 2-local interactions [Hauke et al., 2020]. The general form for a QUBO can be seen in Equation 5

$$min\left(\sum_{i} a_i x_i + \sum_{i}\sum_{j>i} b_{i,j} x_i x_j + c\right) \tag{5}$$

where a_i and $b_{i,j}$ are constants that are chosen to define the problem, x_i and x_j are the binary variables for the problem, and c is a constant term. Note that Equation 5 is minimised because annealing devices are always trying to find the lowest energy state.

The formulation of a problem as a QUBO is needed to program the problem onto a D-wave device. The D-Wave Quantum Processing Unit (QPU) is a lattice of interconnected qubits. These qubits are connected via couplers - although the qubits are not fully connected. To program a D-Wave quantum computer is to set values for its qubit biases and coupler strengths. The coefficients for the linear terms in Equation 5 set the values for the bias and the quadratic terms set the values for the coupler strengths [D-wave, 2023b].

Implementation

The Ocean SDK in Python was used to construct the problem for D-wave devices [D-wave, 2023a]. To submit a problem to a dedicated D-wave device, you must formulate a BQM as either a QUBO or Ising model. The following is a coverage of the key components for creating a BQM using the QUBO approach and Ocean SDK (the repository that stores the code executed for this work can be found at the following address: **omitted for review**):

1. The implementation begins with the creation of an empty BQM as a binary model: ”bqm = BinaryQuadraticModel(’BINARY’)”. This sets up a model where variables (q_{1-12} in Equation 4) will be binary (0 or 1).

2. Variables must then be added into the empty BQM. This is initially done with 0 as the linear coefficient (a_i in Equation 5): ”bqm.add_variable(qi, 0)”. Where ”qi” is q_{1-12} from Equation 4.

3. The east wall placement constraint was then added. This was included as an objective equation by modifying the linear coefficients for certain variables (those that determine the x coordinate of both tiles). ”bqm.add_variable(qi, -east_wall_weight)” was used to modify the linear coefficient of the x coordinate variables to a value equal to the negative of the variable ”east_wall_weight”. The negative is used to help promote (decrease the energy of) solutions that are on the east wall. Choosing this value is a matter of balancing the relative importance of each constraint. A higher value will prioritise solutions which meet this constraint over the other. A value of 3 was found to work well through simulator and dedicated device testing.

4. The no-overlap constraint was incorporated indirectly by defining quadratic interactions between variables. For instance, ”bqm.add_interaction(qi, qj, penalty)” adds a quadratic term to the model. The value of ”penalty” determines the weighting of this no-overlap constraint. The major challenge with representing this problem as a BQM was implementing the no-overlap constraint. This is because it requires the consideration of more than 2 variables at once (for example the tiles can share their 2 digits that make up their y coordinate so long as they do not share the third). Remember that due to manufacturing constraints problems must be formulated as QUBOs instead of HUBOs. To overcome this, an ancillary variable can be used. An ancillary variable z with a large negative bias to encourage $z = 1$ is used in conjunction with other variables to model these more complex constraints.

5. The BQM can then be solved using D-wave’s resources. For this work both a simulated and dedicated approach was utilised. As with the Grover’s approach, this required specifying the number of runs. 1000

was chosen for this approach as well. Ocean supports the "SimulatedAnnealingSampler()" for simulating results locally, and "EmbeddingComposite(DwaveSampler())" was used for obtaining quantum achieved results.

An important note is the concept of embedding. The lattice of qubits is not a fully connected lattice and hence the BQM must undergo a process of embedding where qubits are grouped using different topologies so that the interactions between them can be represented. "EmbeddingComposite()" is the default option offered by D-wave and was sufficient for a problem of the complexity considered here.

Results

In this section, the results obtained through the use of simulated and real quantum devices for both the Grover's algorithm and the QA approach to solving the tiling problem are presented. These are the results of executing steps 3, 4, and 5 in Fig 2. These results consist of 3-D histograms to show the frequency with which each tile is placed in each position for the simulated and quantum computations of both approaches - Figures 7 and 8. Note that these results consider only 8x8 grids and were used to confirm if further investigation was warranted. This is then followed by Table 2 which contains the remaining preliminary (step 4) results describing the levels of error returned and Table 3 which shows a breakdown of the time to solution for each approach.

It can be seen from these initial results that the Grover's algorithm approach was dominated by error when run on a dedicated device. As such, the results shown in Fig 9 and 10 show how only QA achieved results vary at increasing problem scales. Table 4 contains results showing how the time-to-solution varies for classical and QA approaches as grid size increases, as well as the error and closeness to a uniform distribution for QA results.

Timing results for the Grover's approaches were taken from the job manager on the IBM quantum platform where Qiskit runtime usage was taken as the QPU time and CPU time was worked out by subtracting queue time from total time. Simulated QA times were obtained by timing script execution. Real QA timings were obtained by extracting the timings returned as part of the results. QPU time is taken as QPU_access_time. An explanation of annealing time components can be seen in Figure 6 which is taken from the Operation and Timing section of D-wave [2023b].

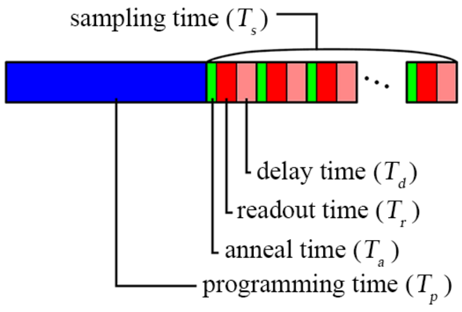
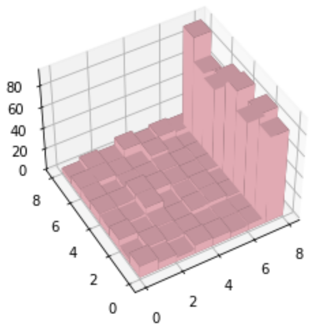
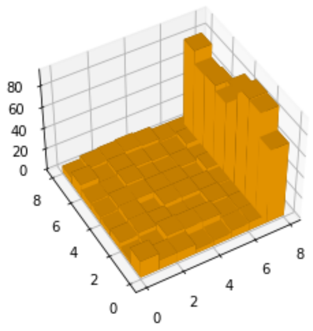


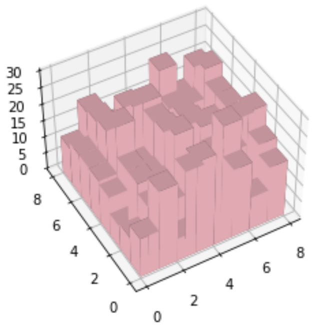
Figure 6: A figure showing



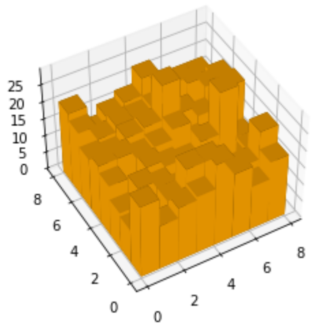
(a) T1 (IBMQ_qasm_sim)



(b) T2 (IBMQ_qasm_sim)



(c) T1 (IBMQ_Brisbane)



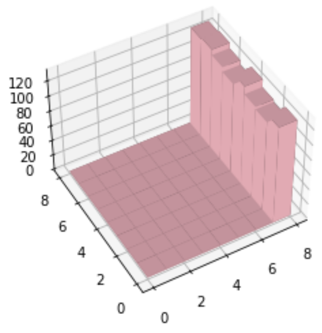
(d) T2 (IBMQ_Brisbane)

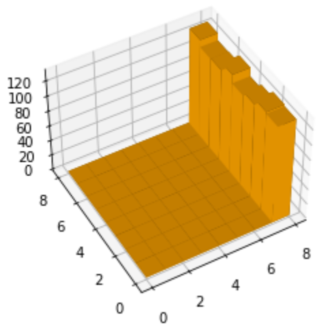
Figure 7: Figures showing the results of both simulated and real Grover's Algorithm approaches to the tiling problem. The frequency of tile placement at each position on the grid is indicated by the height of the bar at that coordinate.

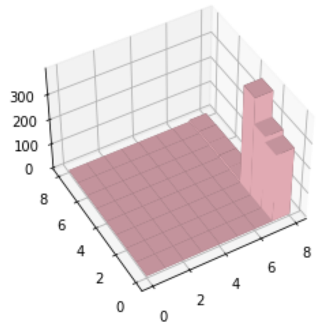
Discussion

This work set out to investigate two different QC approaches for tackling a generalised engineering design problem. The aim of this investigation was to determine if either of the approaches were suitable for use by engineering designers. The results collected provide insight into the quality of results through the percentage error, distribution of valid solutions, and the time-to-solution of the respective approaches. Further, as a result of executing the methodology observations about the ease of implementation for the respective approaches can also be made.

Results were collected in two stages, preliminary results for assessing the feasibility of each approach and more detailed results concerning performance at different problem scales. Preliminary results for both approaches are presented. The combination of the 3-D histograms for Grover's approach, shown in Figure 7, and the error levels, shown in Table 2, show that a simulated approach obtains good results. It is clear that the approach is more likely to return a result that meets our constraints and that there is a roughly uniform distribution of results across the 8 valid tile positions. This provides assurance that the quantum circuit for Grover's algorithm has been constructed appropriately for our problem. Whilst not getting the potential advantages of QC, the opportunity to model problems using QC principles with the view to one day using a real-world


(a) T1 (Annealing Sim)


(b) T2 (Annealing Sim)


(c) T1 (Annealing Real)

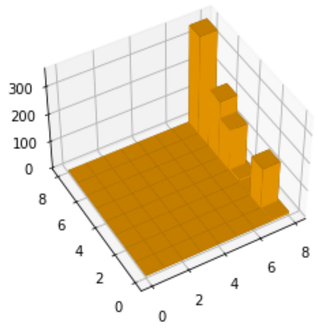

(d) T2 (Annealing Real)

Figure 8: Figures showing the results of both simulated and real QA approaches to the tiling problem. The frequency of tile placement at each position on the grid is indicated by the height of the bar at that coordinate.

device is beneficial. However, when looking at the results obtained from one of IBM's dedicated quantum devices it is clear that the results become dominated by noise.

The IBM_Brisbane device has recently had it Circuit Layer Operations Per Seconds (CLOPS) published as 2700 [IBM, 2023a], and the depth (number of operations/gates applied in sequence) of the Grover's circuit with 9 Oracle repetitions is 289. This means it would take ≈ 0.107s to complete the circuit. As of 2022 IBM claim they have measured coherence times of up to 0.0004s. This indicates that the qubits inside the IBM devices may well be decohering (having their quantum state disrupted by environmental factors) before useful results can be obtained. Further, NISQ devices suffer from other forms of error, such as gate errors caused by poor calibration [Yang et al., 2023]. The depth of the circuit could be reduced by decreasing the number of applied oracles, at the expense of reducing the probability of returning a correct solution. This negative may be offset by increasing the number of runs. A small experiment was run that investigated a single oracle version for 8000 runs. However, this was found to be just as dominated by error and so the Grover's approach was not investigated further as part of step 5 in Fig 2.

The 3-D histograms in Fig 8 and error levels in Table 2 for the annealing approach show improved performance over the Grover's approach. The simulated results show that the problem has been formulated correctly. Further, the results

Table 2: A Table containing the percentage error present in both the simulated and real quantum results obtained for Grover's and QA approaches, broken up as they apply to each constraint.

	No overlap [%]	Eastern Wall [%]	Total [%]
Grover's Simulated	1.5	42.0	43.5
Grover's Quantum	1.6	97.8	99.4
QA simulated	0.0	0.0	0.0
QA real	0.0	0.0	0.0

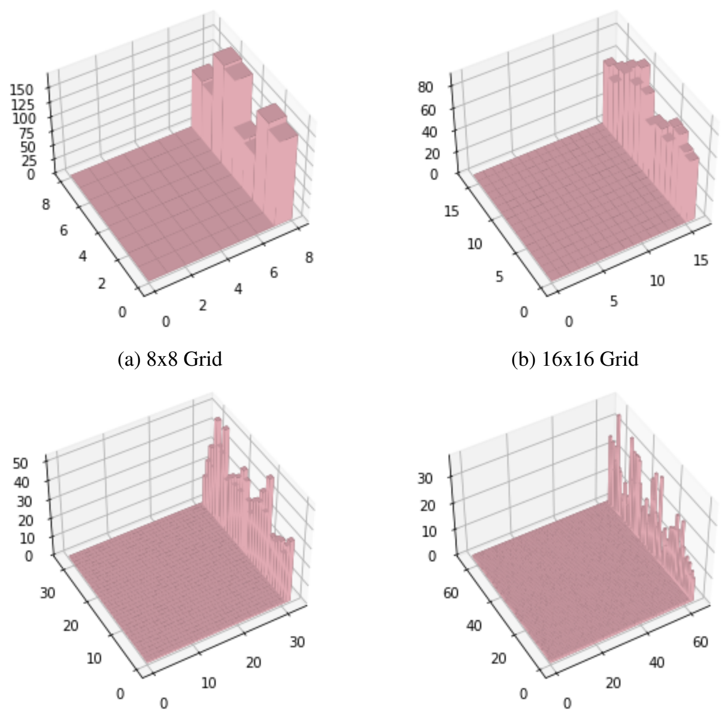
Table 3: A Table containing the times required for obtaining the results presented in Figures 7 and 8. Note that a queue time of N/A is given for locally run approaches.

	QPU/CPU time	Queue time	Total time
Grover's Simulated	≈5m 45s	less than 1s	5m 45s
Grover's Quantum	1m 27s	2h 17m 17.5s	3h 0m 25.3s
QA Simulated	0.243s	N/A	0.243s
QA Real	0.126s	0.136	0.262s
Brute Force	0.00367s	N/A	0.00367s

achieved using a dedicated D-wave device suffer from 0% error, as oppose to the 99.4% error when using IBM devices. This is promising as it implies that D-wave devices can be used to obtain useful results. However, these histograms also show the inherent bias that is known to affect D-wave's devices [Hauke et al., 2020]. This means that whilst the annealing approach does return valid results, its solution space exploration capacity might be limited. Further, real-world problems are likely to contain many more constraints and objectives which will complicate the energy landscape. As such, increased error would be expected to appear in results obtained via annealing.

table 3 contains time-to-solution values for the preliminary results. The classical brute force approach is 2 orders of magnitude faster than QA and 7 orders faster than the quantum implemented Grover's approach. A quantum speedup has not been demonstrated, however, this relatively small problem may not fully leverage the promised scaling benefits of QC. Comparing QA and Grover's approach in table 3, QA proves faster, in both queue time and QPU time. Note that these timings are for single jobs and were not averaged over multiple jobs.

These preliminary results show that only QA has produced feasible results for the chosen problem. It was also of interest how this would change at increased problem scales. Fig 9 and 10 show the results returned from a single job for each gird size. Considering these figures alongside the data presented in Table 4 shows how QA performance changes at increased problem scales. The solutions remain usable at all scales, meaning that almost all solutions returned are valid. However, the coefficient of variation does increase with grid size. This is to be



(a) 8x8 Grid (b) 16x16 Grid

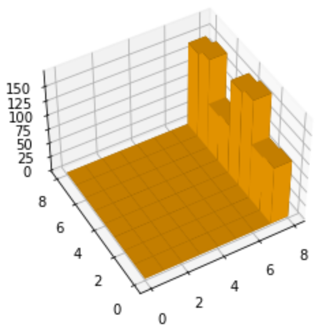
(c) 32x32 Grid (d) 64x64 Grid

Figure 9: Figures showing the results for Tile 1 obtained using Dwave's Advantage_system4.1 at varying grid sizes.

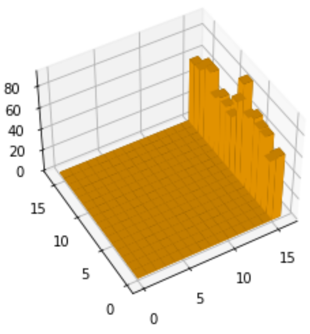
expected as QA is a sampling method and so when there are more valid solutions the results on the eastern wall are less uniformly distributed. The QPU time and percentage error exhibit small increases with problem scale. Since annealing time is specified as a problem parameter there would be no expected increase in QPU access time aside from an increased time to program the problem onto the annealing device. The most significant observation from these results is that QA's usability remains relatively constant at increased scales, but the classical brute force approach suffers significantly. Between a 16x16 and 32x32 grid QA overtakes the brute force approach in time-to-solution.

It is important to note that there are classical algorithms superior to the brute force approach used here. As such, this is not a demonstration of quantum speed-up. However, these QA annealing results were obtained without using the problem specific knowledge that would be needed to choose and make use of the appropriate classical algorithm. Instead, knowledge of the QA procedure and how to formulate the problem as a BQM was needed.

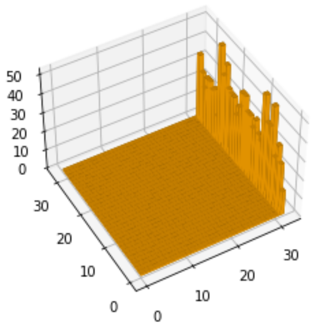
The ease of implementation for the hardware used in each approach can be discussed subjectively as a result of completing this methodology. The classical analogy of logic gates for the gate-based Grover's approach could be argued to aid understanding of the algorithms operation. However, the annealing approach does not require the construction of a circuit. Instead it requires the selection of coefficients in Equation 5. Whilst this is a more abstract approach it is an approach that remains constant, even for other problem



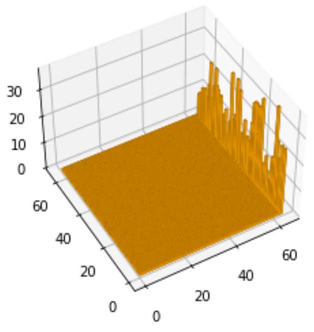
(a) 8x8 Grid



(b) 16x16 Grid



(c) 32x32 Grid



(d) 64x64 Grid

Figure 10: Figures showing the results for Tile 2 obtained using Dwave's Advantage_system4.1 at varying grid sizes. **Could combine these into a 2x8 figure??**

formulations. Further, the energy landscape considered throughout the annealing approach is analogous to the solution space engineering designers are used to exploring. The documentation supporting Qiskit as well as the job manager provided by IBM appeared more developed than D-wave's counterparts. In the authors' opinion, the background theory required for the gate-based approach was more demanding, whilst the problem formulation for annealing was more complex. Considering all these points, the authors' would consider the barrier to implementation lower for annealing methods.

Observations drawn from this work regarding gate-based versus QA methods for engineering design applications suggest that reducing circuit depth in gate-based approaches may enhance usability. D-wave's QA approach can yield rapid and usable results compared to Grover's method, but users should be cautious of inherent biases. In more complex cases, as with increasing scale and a more intricate energy landscape, error in QA results may increase due to a reduced energy gap. Bias drawbacks may intensify if valid solutions represent a smaller part of the solution space. However, the scaling benefits of QC reduce the time disparity between QA and classical approaches. Notably, most of the QPU time for QA was spent on problem programming, suggesting potential for increases in annealing time and more runs without significantly affecting time-to-solution. The number of runs should be increased as problem scale increases to maintain even coverage of equally optimal solutions.

Table 4: A Table containing data comparing the time to solution for the classical brute force approach (CPU time) and the QA approach, as well as error and coefficient of variation values for QA, at different problem scales. Note that a lower value for coefficient of variation indicates less deviation from a uniform distribution.

Grid size	CPU time / s	QPU time / s	Error / %	Coefficient of Variation
8x8	0.00292	0.125	0.01	0.248
16x16	0.04512	0.131	0.006	0.291
32x32	0.72754	0.133	0.002	0.370
64x64	14.7632	0.137	0.002	0.463

Perhaps the most significant observation made during this work concerns the formulation of problem QUBOs for D-wave's annealing devices.

The dedicated QA devices at D-wave accept BQM problems in the QUBO or Ising format. A QUBO formulation was chosen for this work as it appeared more intuitive. To formulate a problem as a QUBO, an equation with all the binary variables needed to represent the problem must be created. The coefficients for those variables and their quadratic combinations in the QUBO equation must also be selected. This should be done such that the equation would reach a minimum value when the variables take values that represent the optimal solution. However, when dealing with many constraints and objectives it is not obvious what the values of these coefficients should be, particularly when considering the weighting of each constraint. Fortunately, each constraint/objective can be considered individually, and their respective QUBOs summed to form the overall BQM [D-wave, 2022]. However, when constraints start to concern more than two variables, ancilla variables are required so that no terms become of a higher order than quadratic.

The example seen in D-wave [2022] is for a small two variable problem. It uses a constraint satisfaction table to generate a system of linear equations which can then be solved for exact QUBO coefficient values representing the constraint. If this method was followed for the problem discussed in this paper, then the number of equations exceeded the number of variables and the system became over-constrained. Instead, the bqm.add_variable() method was used. This method sets the bias value for a variable, meaning it determines the degree to which a variable tends to a particular outcome [D-wave, 2023d]. Essentially, it rewards when the variable takes the value 1 and penalises when it takes the value 0 when the bias is negative (or vise versa if the bias is positive). This method when combined with the binary string problem formulation shown in Equation 4 makes it difficult to create certain constraints and objectives. For example, imagine that the objective was to place a tile as close to the centre of the grid as possible. The number of 1's and 0's in the binary string dictating the solution is roughly equal. How would you penalise a solution closer to the eastern wall? Penalising more 1's would results in the optimum solution being placed on the western wall. Rewarding more 1's would move the optimum towards the eastern wall. Additionally, this binary string formulation makes it hard to create smoothly varying constraints. This is best seen in Figure 11. This figure is a visual representation of the energy landscape for tile 1's position, where a higher energy represents a more probable returned solution. It can be seen that solutions on the eastern wall are best, and hence they are returned most often (seen in the results as part of Figure 8 and 9). However, it can also be seen (most clearly at x=5) that not every solution performs better than its westerly neighbour. This is due to the way that counting in binary works, and that as the integer value the string represents increases the number of 1's within the string does not always increase (011 → 100).

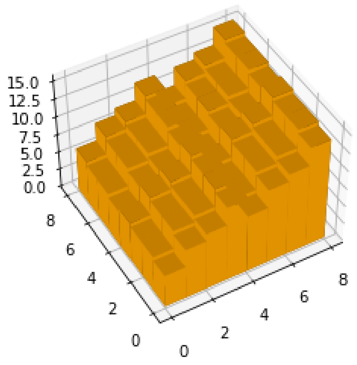


Figure 11: A figure showing how the difficulty in creating a smoothly varying energy landscape using the binary string solution representation. Note that for visual clarity the sign of all energies have been changed to make them positive.

The impact of this is that engineering designers must consider the types of constraints and objectives they will be creating during the formulation of the problem. Whilst the number of variables (qubits) required for solution representation scales well with problem size for this approach, it is less flexible when creating constraints and objectives. An alternative problem formulation may be better. An example could be one variable per tile per space for each axis. So for an 8x8 grid there would be 8 variable for the x-coordinate of tile 1, 8 for the y-coordinate of tile 1, and 16 more for tile 2. The position of a tile could then be indicated by a single variable taking the value 1 for each coordinate. This would scale less well with problem space and would come with the added constraint that not all value combinations for the variables constitute a possible solution (when more than one variable is 1 for example). The scaling problem is less of a concern since D-wave devices have many more qubits than were used in this study. However, it is important to remember that not all qubits can be used for variables as some are needed to connect qubits within the device.

Future Work

The areas identified for further investigation include exploring various types of engineering design problem currently addressed with classical computation, providing insights into potential areas for QC. This could involve a review of the literature looking for areas that suit the characteristics of QC approaches. This could involve searching for extremely broad problems but those where checking the validity of a solution remains simple. This aligns well with the identified limitations of gate-based approaches. Additionally, investigating other gate-based algorithms that lead to shallower but wider circuits as they will be more resistant to noise.

Another avenue for exploration is other gate-based QC hardware options. IBM uses superconducting qubits, but other options with longer coherence times are available. This could involve exploring what options are offered by other hardware developing companies such as Microsoft that use Nuclear Magnetic Resonance (NMR) QC or IonQ that use trapped ion QC [Hassija et al., 2020, Cheng et al., 2023] as these are said to have long coherence times.

Conclusion

In conclusion, D-Wave stands out as an appropriate quantum computing platform for near-term use. The robustness of simulators is crucial in building experience that will enable quantum computing principles for engineering problem solving to be leveraged. An important future direction involves determining the optimal size of design problems for quantum computing applications, considering scalability and computational limits. This work demonstrates the relatively straightforward representation of a generic engineering design problem for both QA and gate-based approaches, with QA already yielding usable results. It has also been shown that engineering designers should pay particular attention to the type of constraints and objectives that they will be using when deciding their problem formulation. Expanding research in quantum computing offers hope for improved hardware, including longer coherence times, reduced bias in annealing devices, increased CLOPS, and more qubits. As a result, this work provides a foundation for engineering designers to consider how they can adapt their problems to leverage potential quantum speedup in the future.

References

F. M. Al-Turjman, H. S. Hassanein, and M. A. Ibnkahla. Efficient deployment of wireless sensor networks targeting environment monitoring applications. *Computer Communications*, 36(2):135–148, 2013.

T. Albash and D. A. Lidar. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Physical Review X*, 8(3):031016, 2018.

M. H. Amin. Consistency of the adiabatic theorem. *Physical review letters*, 102(22): 220401, 2009.

A. Belhocine and O. I. Abdullah. Thermomechanical model for the analysis of disc brake using the finite element method in frictional contact. *Multiscale Science and Engineering*, 2:27–41, 2020.

M. M. Biskjaer, P. Dalsgaard, and K. Halskov. A constraint-based understanding of design spaces. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 453–462, 2014.

B. Cheng, X.-H. Deng, X. Gu, Y. He, G. Hu, P. Huang, J. Li, B.-C. Lin, D. Lu, Y. Lu, et al. Noisy intermediate-scale quantum computers. *Frontiers of Physics*, 18(2): 21308, 2023.

C. Correa-Jullian, S. Cofre-Martel, G. San Martin, E. Lopez Droguett, G. de Novaes Pires Leite, and A. Costa. Exploring quantum machine learning and feature reduction techniques for wind turbine pitch fault detection. *Energies*, 15(8):2792, 2022.

D-wave. D-wave ocean software documentation, 2023a. URL https://docs.ocean.dwavesys.com/en/stable/.

I. D-wave. Problem formulation guide, 2022. URL https://www.dwavesys.com/media/bu0lh5ee/problem-formulation-guide-2022-01-10.pdf.

I. D-wave. Getting started with d-wave solvers, 2023b. URL https://docs.dwavesys.com/docs/latest/doc_getting_started.html.

I. D-wave. D-wave leap log-in, 2023c. URL https://cloud.dwavesys.com/leap/.

I. D-wave. Resource library, 2023d. URL https://www.dwavesys.com/learn/resource-library/.

I. D-wave. Real-world quantum applications at business scale, 2023e. URL https://www.dwavesys.com/learn/customer-success-stories/.

S. Dasgupta and T. S. Humble. Characterizing the stability of nisq devices. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 419–429. IEEE, 2020.

D.-W. Developers. Measuring performance of the leap constrained quadratic model solver. Technical report, Tech. Rep. 14-1065A-A, D-Wave Systems Inc, 2022.

L. Filipovic and S. Selberherr. Microstructure and granularity effects in electromigration. *IEEE Journal of the Electron Devices Society*, 9:476–483, 2020.

S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya. Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience*, 52(1):66–114, 2022.

D. Gonzalez-Delgado, P. Jaen-Sola, and E. Oterkus. Design and optimization of multi-mw offshore direct-drive wind turbine electrical generator structures using generative design techniques. *Ocean Engineering*, 280:114417, 2023.

J. Gopsill, G. Johns, and B. Hicks. Quantum combinatorial design. *Proceedings of the design society*, 1:2511–2520, 2021.

J. Gopsill, O. Schiffmann, and B. Hicks. Research questions in applying quantum computing to systems design. In J. S. Gero, editor, *Design Computing and Cognition'22*, pages 735–745, Cham, 2022. Springer International Publishing. ISBN 978-3-031-20418-0. doi: 10.1007/978-3-031-20418-0_43,.

V. Hassija, V. Chamola, V. Saxena, V. Chanana, P. Parashari, S. Mumtaz, and M. Guizani. Present landscape of quantum computing. *IET Quantum Communication*, 1(2): 42–48, 2020.

M. B. Hastings. The power of adiabatic quantum computation with no sign problem. *Quantum*, 5:597, 2021.

P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver. Perspectives of quantum annealing: Methods and implementations. *Reports on Progress in Physics*, 83(5):054401, 2020.

S. Held, B. Korte, D. Rautenbach, and J. Vygen. Combinatorial optimization in vlsi design. *Combinatorial Optimization*, pages 33–96, 2011.

J. Hoole, P. H. Mellor, N. Simpson, and D. North. Statistical simulation of conductor lay and ac losses in multi-strand stator windings. In *2021 IEEE International Electric Machines & Drives Conference (IEMDC)*, pages 1–8. IEEE, 2021.

IBM. Compute resources, 2023a. URL https://quantum-computing.ibm.com/services/resources.

IBM. Updating how we measure quantum quality and speed, 2023b. URL https://research.ibm.com/blog/quantum-metric-layer-fidelity.

C. Lécuyer. Driving semiconductor innovation: Moore's law at fairchild and intel. *Enterprise & Society*, 23(1):133–163, 2022.

F. Leymann and J. Barzen. The bitter truth about gate-based quantum algorithms in the nisq era. *Quantum Science and Technology*, 5(4):044007, 2020.

Y. Li, L. Song, Q. Sun, H. Xu, X. Li, Z. Fang, and W. Yao. Rolling bearing fault diagnosis based on quantum ls-svm. *EPJ Quantum Technology*, 9(1):1–15, 2022.

A. Mahasinghe. Vibration analysis of cyclic symmetrical systems by quantum algorithms. *Mathematical Problems in Engineering*, 2019, 2019.

Y. Mahmoudi, N. Zioui, H. Belbachir, M. Tadjine, and A. Rezgui. A brief review on mathematical tools applicable to quantum computing for modelling and optimization problems in engineering. *Emerging Science Journal*, 7(1):289–312, 2022.

P. Mellor, J. Hoole, and N. Simpson. Computationally efficient prediction of statistical variance in the ac losses of multi-stranded windings. In *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 3887–3894. IEEE, 2021.

S. Morita and H. Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12), 2008.

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

Qiskit. Qiskit homepage, 2023. URL https://qiskit.org/.

C. Ranaweera, P. Monti, B. Skubic, E. Wong, M. Furdek, L. Wosinska, C. M. Machuca, A. Nirmalathas, and C. Lim. Optical transport network design for 5g fixed wireless access. *Journal of Lightwave Technology*, 37(16):3893–3901, 2019.

N. Ray, T. Banerjee, B. Nadiga, and S. Karra. On the viability of quantum annealers to solve fluid flows. *Frontiers in Mechanical Engineering*, 8:906696, 2022.

K. S. N. Ripon and J. Torresen. Integrated job shop scheduling and layout planning: A hybrid evolutionary method for optimizing multiple objectives. *Evolving Systems*, 5: 121–132, 2014.

O. Schiffmann, B. Hicks, A. Nassehi, J. Gopsill, and M. Valero. A cost–benefit analysis simulation for the digitalisation of cold supply chains. *Sensors*, 23(8), 2023. ISSN 1424-8220. doi: 10.3390/s23084147.

G. Sun, H. Zhang, J. Fang, G. Li, and Q. Li. A new multi-objective discrete robust optimization algorithm for engineering design. *Applied Mathematical Modelling*, 53: 602–621, 2018.

H. Sutter et al. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210, 2005.

E. P. Tapia. qiskit-algorithms, 2023. URL https://github.com/qiskit-community/qiskit-algorithms/blob/stable/0.2/docs/tutorials/06_grover.ipynb.

M. H. Ullah, R. Eskandarpour, H. Zheng, and A. Khodaei. Quantum computing for smart grid applications. *IET Generation, Transmission & Distribution*, 16(21):4239–4257, 2022.

S. E. Venegas-Andraca, W. Cruz-Santos, C. McGeoch, and M. Lanzagorta. A cross-disciplinary introduction to quantum annealing-based algorithms. *Contemporary Physics*, 59(2):174–197, 2018.

C. P. Williams. Quantum search algorithms in science and engineering. *Computing in science & engineering*, 3(2):44–51, 2001.

Z. Yang, M. Zolanvari, and R. Jain. A survey of important issues in quantum computing and communications. *IEEE Communications Surveys & Tutorials*, 2023.

S. Yingchareonthawornchai, C. Aporntewan, and P. Chongstitvatana. An implementation of compact genetic algorithm on a quantum computer. In *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*, pages 131–135. IEEE, 2012.

M. Zandsalimy and C. Ollivier-Gooch. A novel approach to mesh optimization to stabilize unstructured finite volume simulations. *Journal of Computational Physics*, 453:110959, 2022.

Y. Zhang, S. Du, and Q. Zhang. Improved slime mold algorithm with dynamic quantum rotation gate and opposition-based learning for global optimization and engineering design problems. *Algorithms*, 15(9):317, 2022.