Certified Quantum Random Generators and QUBO Solutions

by

Nan Huang

A thesis submitted in partial fulfilment of the requirements for the degree of PhD in Computer Science, the University of Auckland, 2023.

ABSTRACT

This thesis has two main parts.

In the first part, we develop tools for distinguishing quantum random strings generated by quantum experiments certified by the Kochen-Specker theorem from random strings generated by classic algorithms. This analysis is important and essential because the theoretical certification has to be validated experimentally to be physically relevant. Instead of standard randomness tests that check the specific properties of strings of bits, our tests focus on indirectly identifying evidence of incomputability, which distinguishes quantum random sequences from pseudo ones. Variants of Solovay-Strassen primality tests based on the Chaitin-Schwartz theorem are implemented. Even though some of the tests are capable of showing differences, conclusive differences have not been identified. This indicates that further study is needed to overcome the difficulty of observing incomputability in quantum random strings. The results in this part have been published in [10].

The aim of the second part is to develop quantum algorithms that solve the maximum common subgraph isomorphism problems. Three quantum annealing algorithms have been developed, proved correct, and analysed based on different properties to be maximised. Tests have been performed on a D-Wave machine (which is a quantum annealing type of quantum computer). The tests showed that the current quantum machine could solve toy problems with high enough accuracy. More experiments with new versions of the quantum machine and the use of proposed quantum algorithms in practical applications are interesting problems to be further studied. The results in this part have been published in [93, 92].

ACKNOWLEDGMENTS

I am grateful to my supervisors, Prof. C. Calude and Dr. M. Dinneen, for their guidance not only scientific but also beyond.
I thank my co-authors A. Abbott, C. Calude, M. Dinneen and D. Roje for a fruitful collaboration.
I also thank Richard Hua for many discussions.
Finally, I thank my family, who supported me on this journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents an overview of the quantum theory necessary for the main results included in this thesis.

1.1 Quantum computing

Quantum computing was first introduced by Paul Benioff and Yuri I. Manin in 1980 and Richard Feynman in 1982 and intensively researched afterwards.

The origins of quantum computing can be traced back to the proposals made by Benioff for quantum Turing machines [30] and Feynman's ideas for overcoming the challenge of simulating quantum mechanics using classical computers [80]. These proposals eventually paved the way for Deutsch's proposition of a universal quantum circuit and the gate model for quantum computing [73].

1.1.1 Quantum Bits

All computers rely on the ability to store and manipulate information. Classical computers use bits which are physically represented by two-state classical systems (two positions of an electrical switch, two distinct voltage or current levels allowed by a circuit).

Quantum computers use quantum physical systems which can also be in two states, conventionally named *quantum bits*, or *qubits*, and denoted by $|0\rangle$ and $|1\rangle$, [52, 118].

Quantum bits are described by quantum states (i.e. abstract, in Hilbert space) which are physically implemented by quantum systems (e.g. an atom) which are in one of two definite states. For example, the state of a spin-$\frac{1}{2}$ particle, when measured, is always found to be in one of two possible states, represented—using Dirac bra-ket notation— as

$$\left|+\frac{1}{2}\right\rangle \text{ (spin-up) or } \left|-\frac{1}{2}\right\rangle \text{ (spin-down).}$$

In a classical system, a bit is in one state or the other. However, quantum mechanics allows the qubit to be in a superposition of both states simultaneously. A classical analogue is a coin spinning through the air before it lands on a table. Superpositions can be entangled with those of other objects, meaning their outcomes will be "related" even if we don't know yet what they are.

Unlike the intermediate states of a classical bit (e.g. any voltages between the "standard" representations of 0 and 1) which can be distinguished from 0 and 1, but do not exist from an informational point of view, quantum intermediate states cannot be reliably distinguished, even in principle, from the basis states, but do have an informational "existence".

Formally, a *qubit* is a unit vector in the complex space $\mathbf{C}^2$, so for each qubit $|x\rangle$, there are two (complex) numbers $a, b \in \mathbf{C}$ such that

$$|x\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \tag{1.1}$$

where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and $|a|^2 + |b|^2 = 1$.

We can perform a measurement that projects the qubit onto the basis $\{|0\rangle, |1\rangle\}$. Then we will obtain the outcome $|1\rangle$ with probability $|b|^2$, and the outcome $|0\rangle$ with probability $|a|^2$. *After* performing the measurement, the qubit has been prepared in a known state (either $|0\rangle$ or $|1\rangle$); this state is typically different from the previous state.

With the exception of limit cases $a = 0$ and $b = 0$, the *measurement irrevocably disturbs the state*:

> *If the value of the qubit is initially unknown, then there is no way to determine a and b with any conceivable measurement.*

The above facts point out an important difference between qubits and classical bits. There is no problem in measuring a classical bit without disturbing it, so we can decode all of the information that it encodes. If we have a classical bit with a fixed but unknown value (0 or 1), then we can only say that there is a probability that the bit has the value 0, and a probability that the bit has the value 1, and these two probabilities add up to 1. When we measure the bit, we acquire additional information; after measurement, we will know *completely* the value of the bit.

The ability of quantum systems to exist in a “blend" of all their allowed states simultaneously is known as the *Principle of Superposition*.

Even though a qubit can be put in a superposition (1.1), it contains no more information than a classical bit, in spite of its having infinitely many states. The reason is that information can be extracted only by measurement. But, as we have argued, for any measurement of a qubit with respect to a given orthonormal basis, there are only two possible classical results, corresponding to the two vectors of the basis.

It is not possible to capture more information by measuring on two different bases because the *measurement changes the state*. Even worse, we have the No-cloning Theorem:

> *Quantum states cannot be cloned: it is impossible to create an independent and identical copy of an arbitrary unknown quantum state.*

This result has profound implications in quantum computing. For example, it prevents the use of certain classical error correction techniques on quantum states: backup copies of a state in the middle of a quantum computation cannot be created and used for correcting subsequent errors. In 1995, Shor and Steane independently devised the first quantum error correcting codes, which circumvent the No-cloning Theorem.

Quantum algorithms are probabilistic and give the correct answer with some probability; the probability of failure can be decreased by repeating the algorithm. Even if the correct answer is obtained with probability 1, the result is not deterministically correct.

Here are some achievements of quantum computing and algorithmics:

- In 1985 Deutsch constructed the first model of quantum computer by quantisation of the universal Turing machine.
- In 1994 Shor designed a quantum algorithm for factoring integers in polynomial (quantum) time in the size of the input; the problem whether there is a classical polynomial algorithm for factoring is still open.
- Two years later Grover discovered a quantum algorithm for searching an unsorted N-entry database in $\mathrm{O}(\sqrt{N})$ time and $\mathrm{O}(\log N)$ space: this algorithm is optimal within the quantum computing model for black box oracles.

1.1.2 The quantum gate model

The *quantum evolution* of a qubit is described by a "unitary operator", that is, an operator induced by a unitary matrix.[1]

Any unitary operator $U : \mathbf{C}^2 \to \mathbf{C}^2$ can be viewed as a single *qubit gate*. Considering the basis $\{|0\rangle, |1\rangle\}$, the transformation is fully specified by its effect on the basis vectors. In order to obtain the associated matrix of an operator U, we put the coordinates of $U|0\rangle$ in the first column and the coordinates of $U|1\rangle$ in the second one.

The quantum gate model is the most studied model of quantum computing.

The general form of a transformation that acts on a single qubit is a 2×2 matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

which transforms the qubit state $\alpha|0\rangle+\beta|1\rangle$ into the state $(\alpha a+\beta b)|0\rangle+(c\alpha+d\beta)|1\rangle$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha a + \beta b \\ c\alpha + d\beta \end{pmatrix}.$$

The following gate flips the state of its input,

$$\text{NOT } |0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle,$$

and

$$\text{NOT } |1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

We may think of logic gates as transformations. For example, the NOT transformation which interchanges the vectors $|0\rangle$ and $|1\rangle$, is the matrix

$$\text{NOT } = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The *square-root of* NOT (introduced by Deutsch) is the transformation

$$\sqrt{\text{NOT}} : \begin{array}{rcl} |0\rangle & \to & \frac{1}{2}(1+i)|0\rangle + \frac{1}{2}(1-i)|1\rangle, \\ |1\rangle & \to & \frac{1}{2}(1-i)|0\rangle + \frac{1}{2}(1+i)|1\rangle, \end{array}$$

[1] A quadratic matrix A of order n over $\mathbf{C}$ is *unitary* if $AA^\dagger = I$ (the identity $n \times n$ matrix); $A^\dagger$ is the transposed conjugate matrix of A.

$$\sqrt{\text{NOT}} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}.$$

The square-root of NOT is a typical "quantum" gate in the sense that it is impossible to have a single-input/single-output classical binary logic gate that satisfies (1.2). Indeed, any classical binary

$$\sqrt{\text{NOT}}_{\text{classical}}$$

gate is going to output a 0 or a 1 for each possible input 0/1. Assume that we have such a classical square-root of NOT gate acting as a pair of transformations

$$\sqrt{\text{NOT}}_{\text{classical}}(0) = 1,\ \sqrt{\text{NOT}}_{\text{classical}}(1) = 0.$$

Then, two consecutive applications of it will *not* flip the input!

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}} = \text{NOT}\ , \tag{1.2}$$

and

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}^{\dagger}$$
$$= \frac{1}{4}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix} = I.$$

Finally, we consider the Hadamard transformation H is defined by

$$H: \begin{array}{rcl} |0\rangle & \to & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle & \to & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array},$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

This transformation has a number of important applications. When applied to $|0\rangle$, H creates a superposition state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

Recall (1.1):

$$|x\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix},$$

where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and $|a|^2 + |b|^2 = 1$.

A measurement that projects $|x\rangle$ onto the basis $\{|0\rangle, |1\rangle\}$ produces $|1\rangle$ with probability $|b|^2$ and $|0\rangle$ with probability $|a|^2$.

Let $|x\rangle = a|0\rangle + b|1\rangle$ and $|y\rangle = c|0\rangle + d|1\rangle$ with $|a|^2 + |b|^2 = |c|^2 + |d|^2 = 1$.

What about $|x\rangle|y\rangle$? We are $\mathbf{C}^4$ so the basis is

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The pair of qubits

$$|x\rangle|y\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}.$$

This is correct because

$$(|ac|)^2 + (|ad|)^2 + (|bc|)^2 + (|bd|)^2 = |a|^2(|c|^2 + |d|^2) + |b|^2(|c|^2 + |d|^2) = 1,$$

so the probability that a measurement of $|x\rangle|y\rangle$ produces

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

is $(|ac|)^2$ and so on.

In the quantum gate model, a quantum gate array decomposes computation into a sequence quantum gates. A quantum computation can be described as a network of quantum logic gates and measurements. A choice of gate family that enables this construction is known as a universal gate set, since a computer that can run such circuits is a universal quantum computer. One common such set includes all single-qubit gates as well as the CNOT gate. This means any quantum computation can be performed by executing a sequence of single-qubit gates together with CNOT gates. Though this gate set is infinite, it can be replaced with a finite gate set using the Solovay-Kitaev Theorem [99, 70].

1.1.3 Quantum annealing

The adiabatic quantum computing (AQC) model is an alternative to the gate model of quantum computing. In AQC, the computation process commences with an initial Hamiltonian whose ground state is simple to prepare, and concludes with a final Hamiltonian that encodes the solution of the computational problem in its ground state. If the system evolves slowly enough, the Adiabatic Theorem guarantees that the system will remain in its ground state throughout this process. Consequently, we can extract the solution from the final Hamiltonian.

Quantum annealing is one method of engineering an AQC machine, which is believed to be capable of assisting in the resolution of specific combinatorial optimisation problems [98].

More general references about AQC and D-Wave machine are found in [116, 48, 68]. A presentation of practical QA used by D-Wave machines—which are quantum computers designed to solve optimisation problems—is in [117].

Results by [79, 20] "suggest" that the two models of quantum computing are polynomially equivalent. One distinction between the two models is their discrete versus analogue natures: the quantum gate mode is discrete while ADC is a continuum.

Adiabatic in AQC refers to a process in which there is no transfer of energy between the system and its environment. This a thermal (not quantum) property; its name is used here metaphorically. AQC uses the propensity of physical systems—classical or quantum—to minimise their free energy. *Quantum annealing* is free energy minimisation in a quantum system. An AQC algorithm computes an exact or approximate solution of an optimisation problem encoded in the ground state—its lowest-energy state—of a Hamiltonian (the operator corresponding to the total energy of the system).

1.1.4 Quantum annealing with D-Wave machines

The D-Wave machines, such as the D-Wave 2X that we utilise in subsequent chapters for our experiments, are quantum annealing devices specifically designed for solving discrete optimisation problems. In this subsection, we will provide a concise overview of the methods for formulating problems on D-Wave machines, the operational cycles of these machines, the connectivity graph of qubits, and the post-processing stage.

Discrete optimisation, alternatively referred to as combinatorial optimisation, entails optimising objective functions within search spaces composed of a finite set of objects. This section delves into two distinct classes of optimisation objectives.

Ising model

The D-Wave Quantum Processing Unit (QPU) can be seen as a heuristic approach to minimising Ising objective functions through the implementation of physical quantum annealing. The Ising objective function, defined for N variables denoted as $\mathbf{s} = [s_1, ..., s_N]$ where s_i takes values in $\{+1, -1\}$, is given by:

$$E_{ising}(\mathbf{s}) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j. \tag{1.3}$$

Each variable s_i corresponds to a physical Ising spin that can be in a $+1$ or -1 state with a local applied field on each spin that causes it to prefer either the $+1$ or -1 state. The sign and magnitude of this preference—that is, the value of the local field—is denoted h_i. There may also be couplings between spins i and j such that the system prefers the pair of spins to be in either of the two sets defined by $s_i = s_j$ (ferromagnetic coupling) or $s_i = -s_j$ (antiferromagnetic coupling). The sign and magnitude of this preference is denoted $J_{i,j}$.

In a single-spin system, when h_1 has a large positive value, the lowest energy state is achieved when s_1 is set to -1. Likewise, in a two-spin system with $h_1 = h_2 = 0$ and $J_{1,2} = -1$, there are two equally favourable lowest energy states: one with $s_1 = -1$ and $s_2 = -1$, and the other with $s_1 = +1$ and $s_2 = +1$.

The D-Wave QPU is constructed using a physical lattice of qubits and couplers known as the Chimera architecture (see Section 1.1.4 for more details). The total number of spins is contingent upon the quantity of qubits within the QPU. As an illustration, the D-Wave 2000 QPU encompasses more than 2000 spins s_i, each of which is connected to a maximum of six other spins. The permissible ranges for h and J values for a specific system can be accessed by querying the solver properties, either through D-Wave's Qubist or the client libraries.

Fomulation of the Ising problem as a QUBO

Frequently, it is more practical to work with variables that take values of 0 or 1, rather than the Ising variables that take values of -1 or 1. As an illustration, consider a maximum independent set (MIS) problem. In this problem, the objective is to choose the most extensive subset of vertices (referred to as the independent set) from a graph $G = (V, E)$, in such a way that no two selected vertices are connected by an edge. Let $x_i = 1$ if vertex i is part of the independent set, and $x_i = 0$ otherwise.

The restriction that no two selected vertices can be linked by an edge can be straightforwardly expressed through penalties of the form $M_{i,j} > 1$ for all $(i, j) \in E$. The maximal independent set is established by minimising $-\sum_{i \in V} x_i$,

thereby defining the overall objective to be minimised as follows:

$$\mathrm{E}_{qubo}(\mathbf{x}) = -\sum_{i \in V} x_i + \sum_{(i,j) \in E} M_{i,j} x_i x_j.$$

In the $-1/+1$ representation, where $s_i = +1$ if i is in the independent set and $s_i = -1$, otherwise, the same penalty is expressed less intuitively as $(s_i s_j + s_i + s_j + 1)/4$. The $0/1$ variables offer a more natural and concise representation for this problem.

QUBOs are commonly represented using matrix notation, typically taking the form:

$$\mathbf{x}^* = \mathrm{argmin}_{\mathbf{x}}\, E(\mathbf{x}|\mathbf{Q}) = \mathrm{argmin}_{\mathbf{x}} \sum_{i>j} x_i Q_{i,j} x_j = \mathrm{argmin}_{\mathbf{x}} \langle \mathbf{x}, \mathbf{Q}\mathbf{x} \rangle$$

Given $x_i \in \{0, 1\}$, the linear terms in x_i stem from the diagonal elements of Q because $Q_{i,i} x_i^2 = Q_{i,i} x_i$ since $x_i^2 = x_i$ for $x_i = 0$ or 1. Let us assume that Q is upper-triangular. In [101] and related papers, QUBOs have been demonstrated to be an effective representation for modelling and solving a diverse range of discrete optimisation problems.

The Ising and QUBO models are connected through the straightforward transformation $\mathbf{s} = 2\mathbf{x} - \mathbf{1}$, leading to the relationship:

$$\langle \mathbf{x}, \mathbf{Q}\mathbf{x} \rangle = \frac{1}{4} \langle \mathbf{1}, \mathbf{Q}\mathbf{1} \rangle + \langle \overbrace{\mathbf{Q}\mathbf{1}/2}^{h}, \mathbf{s} \rangle + \langle \mathbf{s}, \overbrace{\mathbf{Q}/4}^{J} \mathbf{s} \rangle$$

where $\mathbf{1}$ is the vector all of whose components are 1. Therefore, $E(\mathbf{x}|\mathbf{Q}) = \langle \mathbf{1}|\mathbf{Q}\mathbf{1} \rangle + E(\mathbf{s}|\mathbf{Q}\mathbf{1}/2, \mathbf{Q}/4)$. This allows for seamless translation between the Ising and QUBO representations, depending on convenience. The client libraries accessible on Qubist offer code for implementing this translation process.

The Ising and QUBO problems have been established as NP-hard according to [25], a fact evident when representing the maximum independent set as a QUBO and applying the arithmetic transformation mentioned above. This NP-hardness remains true even when the problem is confined to the Chimera architecture and within the specific range of h_i and $J_{i,j}$ values for the QPU. While there exist several manageable subclasses of Ising/QUBO problems [35], more detailed information can be found in references such as [102], [133], or [62]. For insights into the mapping of NP problems onto the Chimera architecture, refer to Section 1.1.4.

Programming cycle

When a set of h and J values is provided for an Ising problem, the D-Wave system transmits these values to the Digital-to-Analog Converters (DACs) situated on the QPU. Room-temperature electronics generate the initial signals, which are

then transmitted through wires into the refrigerator to configure the DACs. Subsequently, the DACs administer static magnetic-control signals directly to the qubits and couplers. This constitutes the programming cycle of the QPU. Following this, the QPU is allowed to undergo a post-programming thermalisation period, typically lasting 1 millisecond. The entire duration spent on programming the QPU, including the post-programming thermalisation time, is recorded and reported as qpu_programming_time.

Anneal-read cycle

Following the programming cycle, the system transitions into the annealing phase. During this phase, the QPU undergoes repeated cycles of annealing and readout. Annealing is executed using the analog lines for a duration set by the user as annealing_time, which is then reported by the QPU as qpu_anneal_time_per_sample. Subsequently, the digital readout system of the QPU retrieves and provides the spin states of the qubits. The system is then allowed to cool for a period specified by the QPU as qpu_delay_time_per_sample. This interval comprises a fixed value along with any additional time optionally set by the user through the readout_thermalization parameter.

This sequence of annealing and readout is also referred to as a "sample". The process is repeated for a number of samples determined by the user through the num_reads parameter, yielding one solution per sample. The total time required to complete the requested number of samples is provided by the QPU as qpu_sampling_time.

Minor embedding

In graph theory, an undirected graph H earns the title of being a minor of the graph G if it can be derived from G through a series of operations including edge and vertex deletions, as well as edge contractions. Edge contraction involves the removal of an edge while simultaneously merging the two vertices it connected. When an undirected graph H can be transformed from G by contracting specific edges, deleting edges, and removing vertices, we establish that H is a minor of G. The sequence and order in which these contractions and deletions are executed on G do not alter the resultant graph H. If H is indeed a minor of G, we say that H is embedded within G.

Definition 1. *Let $H = (V_1, E_1)$ and $G = (V_2, E_2)$ be two graphs. A **minor embedding** of H onto G is a function $f : V_1 \to V_2$ such that:*

1. *For all $v \in V_1$, the set of vertices v maps to under f are disjoint.*

2. *For all $v \in V_1$, there is a subset of edges $E' \subset E_2$ such that $G' = (f(v), E')$ is connected.*

3. *If $\{u, v\} \in E_1$, then there exist $u, v' \in V_2$ such that $u' \in f(u), v' \in f(v)$ and $\{u', v'\}$ is an edge in E_2.*

Given a specific graph H, there exist algorithms that can efficiently find a minor-embedding of H within G, with polynomial time complexity relative to the size of G. These algorithms range from the pioneering $O(|V(G)|^3)$ time algorithm by Robertson and Seymour [129]. It is important to note, however, that these algorithms are tailored for a fixed H, and their run-time grows exponentially with the size of H.

When dealing with D-Wave machines, the challenge of minor-embedding arises in the task of finding a suitable minor-embedding for a given graph H, while holding G fixed. Heuristic polynomial algorithms [36, 43, 153] are employed to tackle the minor-embedding problem; in particular, the D-Wave software API utilizes similar heuristic solvers for this purpose [67].

Chimera

The D-Wave 2X machines employ the Chimera graph architecture, as illustrated in Figure 1.1. This architecture is not a complete graph. Thus, we used the default minor-embedding algorithm provided by the D-Wave company in their user interface [66]. This algorithm assists in identifying a subgraph within the hardware graph of the machine, ensuring that the QUBO graph is a minor of it.

Available methods for post-processing

The D-Wave system offers users the capability to apply post-processing optimisation and sampling algorithms to solutions obtained from the QPU. post-processing allows for local refinements of these solutions with minimal additional computational load. When submitting a problem to the QPU, users have the following options to choose from:

- No post-processing (default for hardware solvers)
- Optimisation post-processing
- Sampling post-processing (default for VFYC solvers)

For optimisation problems, the objective is to identify the state vector with the lowest energy. Conversely, for sampling problems, the objective is to generate samples from a specific probability distribution. In both scenarios, a logical

graph structure is defined and integrated into the QPU's Chimera topology. post-processing techniques are then applied to solutions defined within this logical graph structure.

Trade-offs for post-processing

To map most real-world problems onto a Chimera graph, it's necessary to enhance the connectivity on the QPU. This is achieved by introducing what are known as "chains" - groups of qubits tightly linked together to collectively represent a single problem variable. In an ideal scenario, where there is infinite precision on the h and J values, one could compel the qubits on the chain to assume the same spin by assigning a sufficiently large (problem-specific) coupling strength between them. However, in reality, chains can and often do break. When this occurs, the corresponding sample can either be disregarded or mapped to a nearby feasible state that doesn't have a breakage. The first option results in sample wastage, while the second involves some additional processing overhead. D-Wave's post-processing algorithms employ a majority voting approach on the chain to map any broken chains to their closest feasible state.

Additionally, in optimisation problems where the goal is to find the global optimum or at least good local optima, states that are not locally optimal are not of immediate interest prior to further post-processing. The most straightforward post-processing method to transform a non-locally optimal state into a potential candidate solution is to conduct a local search to locate a nearby local optimum state. In practice, some of the samples provided by the QPU may not be locally optimal. Similar to handling broken chains, we are faced with two options for dealing with them: either discard them or initiate a local search to rectify them. Once again, the decision involves a trade-off between investing time in generating new samples or allocating time to perform a local search on such samples.

Sample D-Wave code

We present explicit Python code, using the parameters mentioned above, for our generated QUBOs for the maximum common subgraph problem in Appendix A.

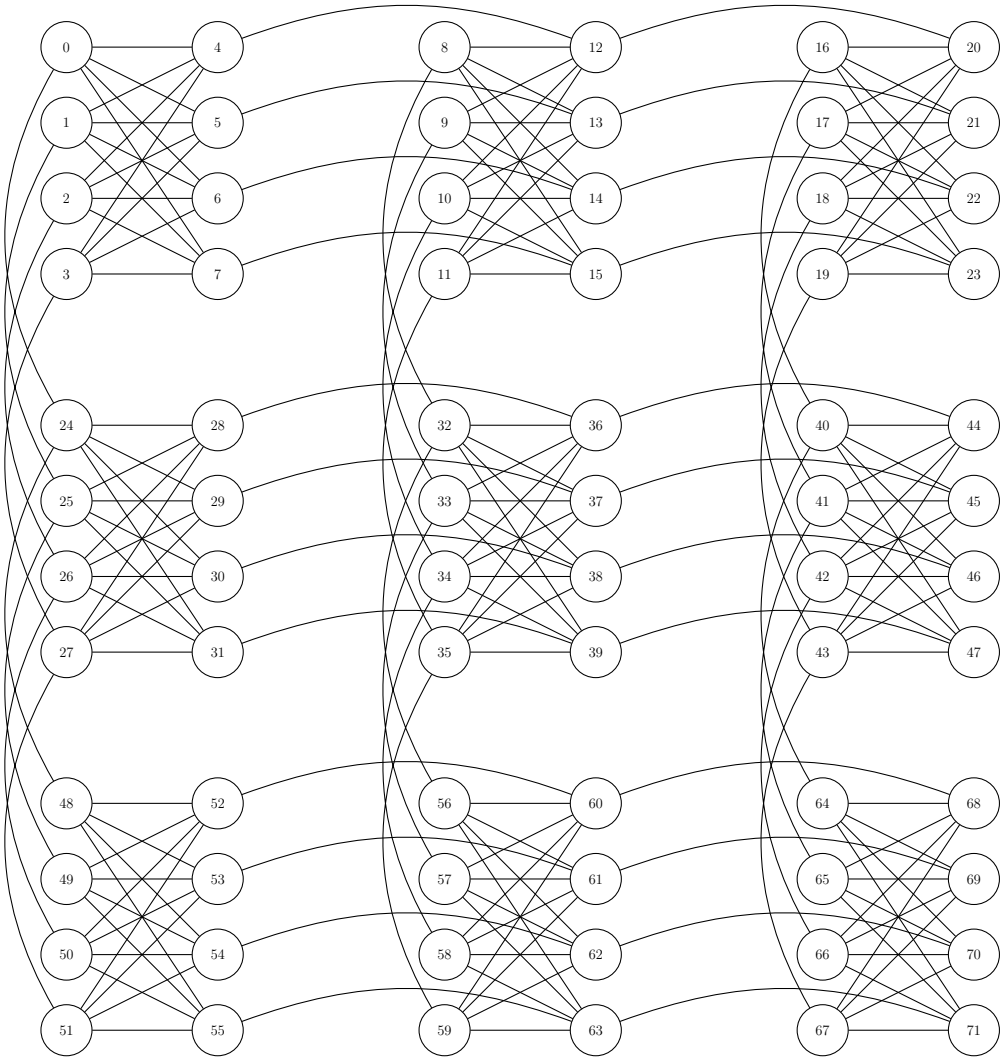


Figure 1.1: Chimera architecture example.

1.1.5 Quantum advantage

Quantum computing cannot compute all partial functions a universal Turing machine can calculate because only total functions can be computed by quantum circuits [21]. Consequently, quantum computing's potential advantages could come only from faster than classical computations.

While Shor's algorithm, Deutsch-Jozsa algorithm and various others in the "black-box" paradigm – where access to a quantum black-box or "oracle" with certain structural properties is assumed – are believed to provide an exponential speedup over classical computers, this is far from the case in general. We use the word "believed" because the superiority of Shor's quantum algorithm over classical ones is still an open problem and various techniques allowing efficient classical simulation of quantum algorithms have been successfully developed [87, 42, 2] even for some "black-box" quantum ones [45, 1, 96, 97].

The quantum computational advantage for simulating quantum systems was first stated by Feynman in 1982, in one of the pioneering papers in quantum computing [81] (the other one was Manin [109]). What is the justification of Feynman's insight? According to the data processing inequality [63, 28], (classical) post-processing cannot increase information. This suggests that to run an accurate classical simulation of a quantum system one must know a lot about the system before the simulation is started [22]. Manin [109] and Feynman [81] have argued that a quantum computer might not need to have so much knowledge. Deutsch [72] proposed

> **The postulate of quantum computation**: Computational devices based on quantum mechanics will be computationally superior compared to digital computers.

Shor's groundbreaking 1994 polynomial quantum algorithm factoring [136] provided remarkable support for this postulate, even though the question of whether factoring is solvable in polynomial time, denoted as P, remained and still remains an open problem. The belief that factoring integers is computationally hard is crucial for the foundations of contemporary cryptography and computational security. For results pointing to the opposite assumption see [87, 42, 2, 122, 33]. Notably, in 2002, Hemaspaandra, Hemaspaandra and Zimand [89], improving results in [137, 34], showed that there are tasks on which polynomial-time quantum machines are exponentially faster almost everywhere than any classical – even bounded-error probabilistic – machine.

In 2011 the syntagm "quantum supremacy" was coined and discussed by J. Preskill in his Rapporteur talk "Quantum Entanglement and Quantum Computing" [128] at the 25th *Solvay Conference on Physics* (Brussels, Belgium, 19–22 October 2011):

> We therefore hope to hasten the onset of the era of quantum supremacy, when we will be able to perform tasks with controlled quantum systems going beyond what can be achieved with ordinary digital computers.

More recently, quantum supremacy was described in [38] as follows:

> Quantum supremacy is achieved when a formal computational task is performed with an existing quantum device which cannot be performed using any known algorithm running on an existing classical supercomputer in a reasonable amount of time.

A quantum computational supremacy experiment has to prove both a lower bound and an upper bound. In Google's proposed experiment [121], the upper bound is given by a quantum algorithm running on a quantum computer with 49 qubits. There are many ways to build qubits, but not all qubits are equal. The magic number 49 (or 50) refers to qubits in the quantum circuit model which are more difficult to control than the qubits used by the D-Wave machine [48]; the lower bound is necessary for proving that no current classical computer can simulate the sampling in a reasonable time from the output distributions of pseudo-random quantum circuits.

All attempts to achieve quantum supremacy, re-named quantum advantage in the last years, succeeded in showing the upper bound, but failed to provide a mathematical proof for the lower bound [47].

1.2 Randomness

Randomness is an important resource in diverse domains: it is used in science, statistics, cryptography, gambling, and even in art and politics. In many of these domains, it is crucial that the randomness is of high quality. This is clearly seen in cryptography, where good randomness is vital to the security of data and communication, but is equally, albeit more subtly, true in other areas such as politics, where decisions of consequence may be made based on scientific and statistical studies relying crucially on randomness.

For a long time, people have predominantly relied on pseudo-random number generators (PRNGs)—that is, computer algorithms designed to simulate randomness—to serve such needs. Problems with various PRNGs, often only uncovered when it is too late, are all too common and can have serious consequences.[2] This has

[2] An example is the discovery in 2012 of a weakness in the encryption system used worldwide for online shopping, banking and email; the flaw was traced to the numbers a PRNG had produced [105].

driven a recent surge of interest in RNGs exploiting physical phenomena, and more particularly in quantum RNGs (QRNGs) that utilise the inherent randomness in quantum mechanics [143, 141, 127, 31]. QRNGs are generally considered to be, by their very nature, better than classical RNGs (such as PRNGs), but how (or can) one test this in practice?

1.2.1 Randomness and algorithmic information theory

In order to develop tests of randomness for QRNGs, it is important to understand what randomness is.

Historically, the quest to develop a formal understanding of randomness focused on the problem of determining whether a given (finite) string or (infinite) sequence of bits is random. One of the first attempts to formalise such a notion of randomness is due to E. Borel, who defined the concept of *Borel normality* for infinite sequences [40]. Borel normality formalises the notion that bits should be evenly and equally distributed within a sequence. Although this captures one of the most intuitive features of randomness, it does not alone capture fully the desired concept. For example, the *Champernowne sequence* $0\,1\,00\,01\,10\,11\,000\,001\,011\,100\ldots$ [56] contains every string of length k with the same limiting frequency of 2^{-k}, and yet the sequence has a simple description: concatenate the binary representation of strings of length k in lexicographical order for $k = 1, 2, \ldots$. Given this description, it is clear that the Champernowne sequence is not random, but highly ordered.

The study of algorithmic information theory, developed in the 1960s by Solomonoff, Kolmogorov and Chaitin, provides more robust and acceptable definitions of a random sequence. In this framework, random strings and sequences are those that are incompressible [54]. The incompressibility of strings depends on the choice of the universal Turing machine; this significant shortcoming disappears when the definition is extended to infinite sequences [44, 75]. The notions of randomness—both for finite strings and infinite sequences—defined in terms of incompressibility are generically called *algorithmic randomness*.

Let us briefly give some technical details useful later in this chapter; we refer the reader to [44] for further details. Consider Turing machines operating on binary strings. A Turing machine U is universal if for every Turing machine M there exists a prefix p (depending only on U and M) such that $U(px) = M(x)$, for every program x. The *Kolmogorov* (or *algorithmic*) *complexity* of a Turing machine M is defined by

$$K_M(x) = \inf\{|s| : M(s) = x\},$$

, where by $|s|$ we denote the length of the string s. We can see that U is universal if and only if for every Turing machine M there exists a constant c such that

$$K_U(x) \leq K_M(x) + c,$$

for every string x. For this notion of complexity, the running time and the amount of storage required for computation are irrelevant. One can prove that for every M the maximum value of $K_M(x)$ over all strings x of a fixed length $|x| = n$ is $n+\mathrm{O}(1)$. Furthermore, the overwhelming majority of strings x of length n have $K_M(x)$ very close to n. This means that almost all strings of length n are incompressible by M: more formally, very few such strings have $K_M(x) < n$ (i.e., are compressible). If U is a universal Turing machine, then the condition $K_M(x) < |x|$ means that $K_U(x) < |x| - c$ for some constant $c > 0$, that is, x is *c-incompressible* (or *c-Kolmogorov random*). These incompressible strings are highly random, patternless and typical. For example, it is easy to prove that less than 2^{n-c} strings of length n are not c-incompressible.

An infinite sequence $\mathbf{x}$ is called *Martin-Löf random* if there exists a constant C such that infinitely many prefixes of $\mathbf{x}$ are C-Kolmogorov random. This definition is equivalent to the condition that $\mathbf{x}$ passes all Martin-Löf tests of randomness [113]; see Section 2.3.2 for more details.

While algorithmic information theory provides a sound notion of randomness for strings and sequences, two important points must be mentioned. Firstly, it is not effectively decidable whether a string or sequence is random, so the notion does not provide a practical way to test the randomness of a finite or infinite sequence of bits. Secondly, it is possible to define ever stronger notions of randomness: from an algorithmic perspective, no notion of "true", "perfect" or "absolute" randomness exists, only degrees of randomness [44, 46, 86]. This should temper any desire to verify the randomness of an RNG only with tests on its output. Instead, we can only hope to compare the quality of strings produced.

As interest in *generating* random numbers soared, the concept of randomness received increased philosophical attention and it became clearer that the algorithmic notion of randomness fails to capture aspects of randomness important for RNGs [3]. Indeed, as von Neumann noted, "there is no such thing as a random number—there are only methods to produce random numbers" [149]. The insight of von Neumann is not that the algorithmic notion of randomness is problematic—indeed, it is highly satisfactory as a notion of random *objects*—but that there is a dual concept of randomness, that of random *processes* [78, 3, 138]. Such a concept has historically received little attention, but the most convincing attempts to make it rigorous are perhaps those who define it as a form of maximal unpredictability: the outcomes of such a process should be unpredictable for any physical observer [77, 14].

The randomness of a process is often quantified in terms of entropy, but it is important to note that, entropy being a function of the probability distribution associated to a process, such a quantification requires i) knowing that the process is indeed unpredictable, and ii) knowing the probability distribution modelling

its behaviour. Although one can empirically estimate the distribution from the output of an RNG, the entropy calculated from such data can only be interpreted as a measure of randomness if (i) is satisfied, and this cannot be directly verified from the empirical data alone.

There are thus two legitimate notions of randomness to be reconciled: that of *process randomness* (which is applicable to RNGs—viewed as processes— themselves), and that of *product randomness* (which is applicable to the strings—i.e. objects—obtained from RNGs). The distinction between these notions is important for understanding tests of randomness.

1.2.2 Random number generators (RNGs)

An ideal random number generator is normally taken to be a random process producing the same probability distribution as the ideal (but unphysical) unbiased coin. It thus produces bits sequentially, thereby generating a sequence $\mathbf{x} = x_1 x_2 \ldots$ with each bit x_i being equiprobable, i.e. $p(x_i = 0) = p(x_i = 1) = 1/2$, and with successive bits produced independently. Hence, all strings x of length k have probability $p(x) = 2^{-k}$ and, in the infinite limit, one obtains the Lebesgue measure over all infinite sequences [44]. This type of ideal source has maximal entropy. It is important to recognise that this conception of an ideal RNG embodies the notion of random processes, not products, and concerns the distribution produced by said process and not its output.

If one tries to implement such a device in practice, two issues immediately become apparent.

Firstly, how is one to know that the process exploited is really random and actually produces the expected ideal distribution? This issue touches on the interpretation of probability [88] (although this is beyond the scope of this thesis). For example, a physical process thought to be represented by the uniform distribution might only exhibit epistemic randomness, and a more precise, deterministic model of the process might be possible which reveals its non-randomness. The most direct way to avoid such possibilities is to harness an indeterministic process to ensure its unpredictability [14].

Secondly, how does one test or verify the randomness of a RNG given that one only has access to (finite) strings produced by it? Although the concepts of process and product randomness are indeed distinct, they are nonetheless related: long enough strings produced by an ideal RNG will, with high probability, be incompressible, while in the infinite limit the sequences produced will be Martin-Löf random (and thus also incomputable) with probability 1 *but not with certainty*: an ideal coin can in principle produce non-random or even computable sequences. However, as mentioned earlier, the randomness of sequences is already an incomputable property. Thus, one can do no better than verifying finitely many

properties of randomness to gain confidence in a RNG.

1.2.3 Pseudo-Random Generators (PRNGs)

The predominant approach to generating randomness is to use algorithms to produce "pseudo-randomness", and such PRNGs are ubiquitous as a result of their practicality and speed. However, the very fact that such devices use computational methods to produce their outcomes distinguishes them from ideal RNGs. PRNGs typically use a short string from an external source—generally assumed to be random—as an initial "seed" for an algorithm [84]. Thus, PRNGs can only produce computable sequences, whereas such sequences should be produced only with probability 0 by an ideal RNG. Instead, effort is made to make PRNGs difficult to distinguish from an ideal RNG given limited (typically polynomial time) computational resources [85], so that the PRNG appears to be a high entropy source. This provides a degree of security against cryptographic attacks, even if the resulting distribution (induced by the distribution over the initial seeds) is far from uniform in reality.

PRNGs generally produce sequences that satisfy many intuitive aspects of randomness—such as the equidistribution of the bits produced—and pass most standard statistical tests of randomness despite their computability. Nonetheless, deficiencies resulting from the non-randomness of PRNGs are regularly exploited (see, e.g., [32]) and much of the interest in quantum randomness has been driven by the potential to avoid the shortcomings of PRNGs.

1.2.4 Quantum randomness

For some time now, quantum mechanics has garnered interest as a potential source of randomness for RNGs. Such interest stems from the fact that certain quantum phenomena, such as the radioactive decay of an atom or the detection of a photon having passed through a beamsplitter, are generally taken to be "intrinsically random" under the standard interpretation of quantum mechanics [18].

The simplest way to generate quantum random bits is via the standard beam splitter which can be realized with the Hadamard transformation H: When applied to $|0\rangle$, H creates a superposition state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

The probability to obtain via measurement 0 is equal to the probability to obtain 1 and equal to $1/2$.

We will first discuss these claims in a more detail—since it is important to base the randomness of QRNGs on more formal grounds rather than simply assuming

such randomness—before discussing one approach to the generation of quantum randomness in more detail.

Claims about quantum randomness originate with the fact that, as a formal theory, quantum mechanics differs fundamentally from classical physics in that not all observable properties are simultaneously defined with arbitrary precision. Instead, quantum mechanics, via the Born rule, only specifies the probabilities with which individual measurement outcomes occur for the measurement of a physical quantity—i.e., a quantum *observable*. Formally, if a system is in a quantum state $|\psi\rangle$ and one measures an observable A with spectral decomposition $A = \sum_i a_i P_i$, where we adopt the notation $P_i = |i\rangle\langle i|$ for rank-1 projection observables, then one obtains outcome a_i with probability

$$P(a_i|\psi) = |\langle i|\psi\rangle|^2. \tag{1.4}$$

Thus, whereas randomness in classical physics is due to incomplete knowledge of the precise initial conditions of a system (e.g., as in chaotic systems) [107], in quantum mechanics it is intrinsic to the standard interpretation of the formal theory.

Nonetheless, the Born rule is a purely formal statement, and interpreting the probability distribution specified by the Born rule remains the subject of ongoing debate. The orthodox interpretation, however, is that the distribution should be understood ontically as representing an indeterministic phenomenon [18]. Crucially, this interpretation is more than a mere assumption: several well-known no-go theorems rule out classical statistical interpretations of quantum randomness.

Bell's Theorem [29] is the most well-known of these results, and shows that a classical, local hidden variable theory cannot reproduce the statistics of quantum correlations that are observed [23] between entangled particles. The Kochen-Specker Theorem [100], although perhaps lesser known, pinpoints this breakdown in determinism in a more precise way: it shows that, for any quantum system with more than 2 dimensions, it is logically impossible to predetermine all measurement outcomes prior to measurement in a noncontextual fashion (i.e., in a way which is independent of other compatible—and thus non-disturbing—measurements one can perform).

More recently, this theorem has been refined to show that the only observables that can be predetermined in a noncontextual way are those for which the Born rule assigns the probability 1 to a particular outcome [11, 17]. More precisely, we say that an observable A is *value definite* for a system prepared in a state $|\psi\rangle$ if it has a predetermined measurement outcome $v_\psi(A)$. The stronger result shows that for systems of more than 2 dimensions, if we assume that any such value definite observables should be noncontextual, then A is value definite if and only if $|\psi\rangle$ is an eigenstate of A; all other observables must be *value indefinite*.

This result makes the extent of quantum value indefiniteness—and thus indeterminism—clear and pinpoints which measurements are protected by such formal results. This not only allows some QRNGs to be based more rigorously on physical principles but also to clarify the link between quantum randomness and indeterminism. Crucially, this result also allows one to show that the measurement of such value indefinite observables satisfies a strong form of unpredictability [16], proving that one really cannot provide better predictions than the Born rule specifies, and thus giving a stronger theoretical grounding to claims about the form of quantum randomness proposed for QRNGs.

Below are more details.

We assume the following premises:

- **Admissibility.** This assumption guarantees agreement with quantum mechanics predictions. Fix a set O of one-dimensional projection observables on $\mathbb{C}^n$ and the value assignment function $v : O \to \{0, 1\}$. Then v is *admissible* if for every context C of O, we have that $\sum_{P \in C} v(P) = 1$. Accordingly, only one projection observable in a context can be assigned the value 1.

- **Non-contextuality of definite values.** Every outcome obtained by measuring a value definite observable is *non-contextual*, i.e. it does not depend on other compatible observables which may be measured alongside it.

- **Eigenstate principle.** If a quantum system is prepared in the state $|\psi\rangle$, then the projection observable P_ψ is value definite.

Theorem 2 (Localised Kochen-Specker Theorem [12, 15, 104, 19])**.** *Assume a quantum system prepared in the state $|\psi\rangle$ in a dimension $n \geq 3$ Hilbert space $\mathbf{C}^n$, and let $|\phi\rangle$ be any quantum state such that $0 < |\langle\psi|\phi\rangle| < 1$. If the following three conditions are satisfied: i) admissibility, ii) non-contextuality and iii) eigenstate principle, then the projection observable P_ψ is value indefinite.*

Theorem 2 states that, under the given assumptions, any quantum state $|\phi\rangle$ that is neither orthogonal nor parallel to $|\psi\rangle$ is *value indefinite*. This result has two major consequences:

1. it shows how to construct a value indefinite observable effectively,

2. it guarantees that the status of "value-indefiniteness" is invariant under minor errors in measurements: this is a significant property as no measurement is exact.

How "good" is such a 3D-QRNG, i.e. what randomness properties can be *certified* for their outcomes? For example, can we prove that the outcomes of the

3D-QRNG are "better" than the outcomes produced by *any* pseudo-random number generator (PRNG)?

For certification, we use the following assumption, which is motivated by the fact that a computable sequence is the strongest form of "deterministic hidden variable":

- **epr principle**: If a repetition of measurements of an observable generates a computable sequence, then these observables are value definite.

Based on the Eigenstate and epr principles, we can prove that the answer to the last question is affirmative: *Any infinite repetition of the experiment measuring a quantum value indefinite observable generates an incomputable infinite sequence* $x_1x_2\ldots$*: no PRNG has this randomness property.*

A stronger result is true. A sequence $\mathbf{x}$ is bi-immune if no algorithm can generate infinitely many correct values of its elements (pairs, (i, x_i)).

Theorem 3 ([6, 19])**.** *Assume the Eigenstate and epr principles. An infinite repetition of the experiment measuring a quantum value indefinite observable in* $\mathbb{C}^b$ *always generates a b-bi-immune sequence* $\mathbf{x} \in A_2^\omega$*, for every* $b \geq 2$*.*

In particular, every sequence generated by the 3D-QRNG is 3-bi-immune.

Theorem 4 ([19])**.** *Assume the epr and Eigenstate principles. Let* $\mathbf{x}$ *be an infinite sequence obtained by measuring a quantum value indefinite observable in* $\mathbb{C}^b$ *in an infinite repetition of the experiment* E*. Then, no single bit* x_i *can be predicted.*

Chapter 2

Quantum Randomness

2.1 Quantum Random Generators (QRNGs)

The properties of quantum measurements make them an ideal candidate for random number generation: if one measures an observable for which the Born rule predicts a uniform distribution, then the QRNG embodies a perfect coin. Moreover, the results discussed above show that—subject to very reasonable physical assumptions about how classical objects should behave—this distribution cannot be given an epistemic interpretation and the corresponding measurement outcomes are thus truly of indeterministic origin. The attractiveness of QRNGs is further enhanced by the possibility of obtaining high bitrates and the simplicity of their physical models. This is in contrast to RNGs based on classical physics, such as chaotic systems.

Early QRNGs relied on features such as radioactive decay [132], but simpler systems based, for example, on measuring the polarisation [95, 141, 135] or detection times [142] of photons, have become the norm due to the practical advantages they provide. Such approaches have led to the development of commercial QRNGs, such as ID Quantique's Quantis [94].

Many successful QRNGs exploit two-dimensional systems to generate randomness (e.g. Quantis uses the polarisation of photons). This greatly simplifies the design and production of such devices but neither Bell's Theorem (which requires entanglement) nor the Kochen-Specker Theorem (which requires at least 3-dimensional systems) are applicable, and these QRNGs thus lack the rigorous theoretical certification that quantum mechanics can provide, even if it may be reasonable to think that the measurements they exploit should still be indeterministic.

The most direct approach to overcoming this shortcoming is to use higher dimensional systems for which the value indefiniteness of measurement outcomes

is, via the Kochen-Specker Theorem in the version of Theeorem 2, provable [13, 7] to certify a QRNG. Such certification is necessarily "device dependent"—that is, it relies on knowledge of the functioning of the QRNG—but nonetheless allows the randomness of the device to be more formally grounded. A simple example of a QRNG certified in this way was proposed in [13, 7] for spin-1 particles, but its principle is applicable to any 3-dimensional system (i.e., an implementation of a qutrit). The approach proposed was to prepare a qutrit in the state $|0\rangle$ before measuring the observable $A = a_0|0'\rangle\langle 0'| + a_1|1'\rangle\langle 1'| + a_2|2'\rangle\langle 2'|$ for which the orthonormal basis $\{|0'\rangle, |1'\rangle, |2'\rangle\}$ is chosen such that $\langle 0|0'\rangle = 0$ and $\langle 0|1'\rangle = \langle 0|2'\rangle = \frac{1}{\sqrt{2}}$ (see Figure 2.1 below). Since the state $|0\rangle$ is thus an eigenstate of the projection observable $P_{0'} = |0'\rangle\langle 0'|$, this observable is value definite with value $v(P_{0'}) = 0$—that is, the measurement outcome a_0 never occurs.[1] However, by the results of [7, 17], both $P_{1'} = |1'\rangle\langle 1'|$ and $P_{2'} = |2'\rangle\langle 2'|$ are value indefinite and, moreover, both outcomes a_1 and a_2 occur with probability 1/2 according to the Born rule 1.4. Thus, the QRNG operates as an ideal coin certified by value indefiniteness.

A QRNG based on this proposal has recently been implemented experimentally [104], not with spin-1 particles but by exploiting a superconducting transmon coupled to a microwave cavity as a qutrit. Figure 2.1 shows a schematic of the QRNG proposed in [13, 7] based on the implementation used by Kulikov et al. [104]. This implementation was used to generate a large number of bits, and in the subsequent sections we will analyse sample sequences produced by this QRNG implementation. In particular, we will look to detect differences between such sequences and pseudo-random sequences arising from algorithmic properties of the sequences.

This approach to certifying a QRNG via value indefiniteness implies some additional interesting algorithmic properties of the output sequences of the device if one is willing to accept slightly stronger physical assumptions (in particular, about whether being able to compute properties in advance implies well-defined physical properties). Specifically, it was shown in [7] that such a device, if used repeatedly *ad infinitum* to generate an infinite sequence $\mathbf{x}$ of bits, will produce a sequence that is strongly incomputable (technically, "bi-immune" [75]) not just with probability 1, but *with certainty*. Although such a result will not alone lead to observable advantages for finite strings – recall that, from the Born rule, an ideal QRNG will produce an incomputable sequence with probability 1—this nonetheless highlights the differences between pseudo and quantum randomness in relation to computability.

More recently there has also been growing interest in a different type of QRNG

[1] This is, of course, only true in the ideal case. In the non ideal scenario, any such outcomes can simply be discarded.

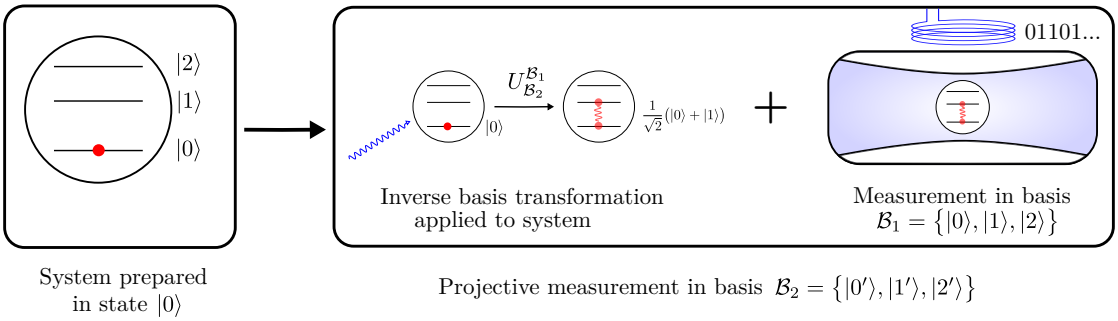


Figure 2.1: Schematic showing the QRNG based on the Kochen-Specker Theorem as implemented in [104]. A transmon qutrit system is initially prepared in the state $|0\rangle$ (with respect to the computational basis $\mathcal{B}_1 = \{|0\rangle, |1\rangle, |2\rangle\}$ by thermal cooling. The system is then measured in the basis $\mathcal{B}_2 = \{|0'\rangle, |1'\rangle, |2'\rangle\}$ with $\langle 0|0'\rangle = 0$ and $\langle 0|1'\rangle = \langle 0|2'\rangle = \frac{1}{\sqrt{2}}$. In practice, this measurement is performed by first performing the inverse basis transformation on the system and measuring in the basis $\mathcal{B}_1$. Since $|\langle 0|0'\rangle|^2 = 0$, this outcome never occurs in an ideal implementation, so the outcomes a_1 and a_2 corresponding to $|1'\rangle\langle 1'|$ and $|2'\rangle\langle 2'|$ are mapped to a binary sequence.

which can provide a stronger form of certification but requires initial random seeds as input. (Such devices are thus technically randomness expansion devices, rather than RNGs.) Typically, such devices rely on violating a Bell inequality, which allows one to certify that the QRNG indeed uses a value indefinite system without assuming *a priori* anything about the workings of the device [127, 58, 108]. Such certification is thus termed "device independent", and allows one to place lower bounds on the entropy of the source; it is particularly important in cryptographic settings, where one perhaps does not wish to trust the workings of a given RNG. Such schemes are very costly, however: not only is an initial random seed required, but one also must separate the QRNG into two space-like separated (or at least isolated) components and the stringent requirements of loophole-free Bell tests reduce the obtainable bitrate by several orders of magnitude compared to "standard" QRNGs [127].

Other related randomness expansion schemes have also been proposed which are less experimentally demanding but require additional physical assumptions [90]. In particular, we note that "noncontextuality inequalities" obtainable from proofs of the Kochen-Specker Theorem can be used to provide such a certification [145, 71, 119]. In doing so, rather than trusting outright that the QRNG uses a system in which the Kochen-Specker Theorem applies, one actively verifies this under a weaker physical assumption about the workings of the device. Nonetheless, such

schemes are still significantly more demanding than that described in Fig. 2.1. Here we thus focus on the similar device-dependent type scheme described in detail above. Indeed, our focus is on testing the algorithmic properties of individual strings produced by a QRNG; such tests are complementary to those aimed at certifying the indeterministic nature of the process itself, and the simplicity of this scheme, along with its high bitrate, facilities such an analysis.

2.2 Testing RNGs

While it is crucial to have a good theoretical understanding of any RNG, there are several reasons why testing experimentally their outputs is nonetheless crucial. Firstly, one can never be sure that the implementation of a RNG matches its theoretical description, a fact that is equally as true for hardware RNGs as for software RNGs. Indeed, in the extreme limit, one might not wish to trust any theoretical claims about a given RNG, and thus confidence in the RNG can only be gained from performing carefully selected tests. Secondly, thorough testing gives one the opportunity to detect any issues with assumptions made in the theoretical analysis of a device or in its practical deployment (e.g., if the distribution of seeds does not match that assumed theoretically the performance of a RNG might be compromised).

It is nonetheless important to recognise that experimental testing can never allow one to perfectly characterise a device. Instead, with access to only finite strings produced by it and the ability to perform a finite number of tests, one can only ever gain increasing confidence in the operation of the device. One can never be certain, for instance, that the output obtained was not a simply atypical behaviour obtained purely by chance. This is doubly true since, as we discussed earlier, randomness is characterised by an infinity of properties, so one must carefully choose the tests one performs.

The issues arising when testing the outputs of RNGs can be illustrated pointedly with an example. Imagine a device which deterministically outputs the digits of the binary expansion of $\pi = \pi_1\pi_2\pi_3\ldots$ starting from the 10^{10}th bit. If we are unaware of the behaviour of this device and believe it to be a RNG, its output will appear extremely random to us; indeed, π passes all standard statistical tests of randomness [111] despite the fact that it is not even known to be Borel normal [151, 24]. Nevertheless, the sequence produced by this box would be computable and thus not random at all. Similarly, any attempt to estimate the entropy of the source from the empirically observed distribution would lead one to believe the source to be a highly random process, despite the fact it is completely deterministic.

Standard statistical tests of randomness focus on properties of the distribution

of bits or bit strings within sequences, properties more closely related to Borel normality than algorithmic complexity. Many such tests were developed with the aim of testing PRNGs, where reproducing such statistical predictions is a primary issue, particularly since failing to do so may leak information about the seed and thus break the security of the PRNG [105]. QRNGs[2] have generally been tested against similar tests, such as the NIST [130] and DIEHARD [112] batteries, and generally perform well. For example, Quantis is officially certified as passing these tests on 1000 samples of 1 million bits [94]. Such tests, however, far from confirm the randomness of the device; indeed, analysis of longer sequences (of 2^{32} bits) revealed (albeit it very small) bias and correlation amongst the output bits [4].

Such statistical non-uniformity is, however, to be expected in RNGs exploiting physical phenomena due to experimental imperfections and instability [13]. Inasmuch as this form of non-uniformity is small enough for the required application, this is not necessarily problematic as long as a QRNG remains certified by value indefiniteness: unlike for PRNGs, where non-equidistribution is often a symptom of deeper issues, the unpredictability of QRNGs is a result of the indeterministic nature of the device, and is thus assured even if the resulting distribution is biased [16]. Moreover, bias can be reduced by careful post-processing [149, 125, 5], allowing quantum indeterminism to still be exploited sufficiently. Although testing such properties is crucial in order to ensure any bias remains tolerably low, such tests do not directly probe crucial advantages of quantum randomness, such as the degree of algorithmic randomness or incomputability of their output.

Some authors have also looked at the compressibility of quantum random sequences using standard compression algorithms [103] as a proxy for direct tests of Kolmogorov complexity. In practice, however, just like the aforementioned tests, this approach fundamentally relies on statistical properties of the sequence and suffers from similar problems as the above tests (such as being fooled by computable sequences). Indeed, it is not possible to directly compute the Kolmogorov complexity since it is an incomputable quantity. Nevertheless, one may still ask whether there are useful tests that indirectly probe this to try and differentiate PRNGS—which always produce computable sequences—from QRNGs [49]. In the following sections we investigate more closely this question.

[2] As we discussed in the previous section, we restrict our discussion henceforth to standard QRNGs, rather than device-independent randomness expansion schemes. These devices remain the standard approach to QRNGs in practice, and developing tests for their output remains a crucial problem despite the increased interest in device-independent QRNGs.

2.3 Testing incomputability and algorithmic randomness

In this section we describe several tests based on algorithmic properties which we use to study random bits obtained from both PRNGs and the QRNG detailed in Figure 2.1. We tested 80 sequences of 2^{26} bits[3] obtained from each of the following six sources: the initial bits of the binary representation of π (which can be seen as a form of pseudo-randomness [24]), the PRNG used by Python v3.5.4 (a Mersenne Twister algorithm) [115], Random123 v1.09 [131], PCG v0.98 [123], xoroshiro128+ [110], and the QRNG described in Section 2.1 (see [104]).

2.3.1 Tests of Borel normality

As mentioned earlier, the notion of Borel normality was the first mathematical definition of algorithmic randomness [40], and although, like many standard tests of randomness, it focuses on the distribution of bits within a sequence, it is nonetheless worth looking at in its own right.

An infinite sequence $\mathbf{x} \in \{0,1\}^\infty$ is (Borel) normal if every binary string appears in the sequence with the right frequency (which is 2^{-n} for a string of length n). Every Martin-Löf random infinite sequence is Borel normal [44], but the converse implication is not true: there exist computable normal sequences, such as Champernowne's sequence mentioned earlier. Normality is invariant under finite variations: adding, removing, or changing a finite number of bits in any normal sequence leaves it normal.

The notion of normality was subsequently transposed from infinite sequences to (finite) strings [44]. In doing so, one has to replace limits with inequalities, and one obtains the following definition. For any fixed integer $m > 1$, consider the alphabet $B_m = \{0,1\}^m$ consisting of all binary strings of length m, and for every $1 \leq i \leq 2^m$ denote by N_i^m the number of occurrences of the lexicographical ith binary string of length m in the string x (considered over the alphabet B_m). By $|x|_m$ we denote the length of x over B_m; $|x|_1 = |x|$. A string $x \in B_m$ is *Borel normal* (*with accuracy* $\frac{1}{\log_2 |x|}$) if for every integer $1 \leq m \leq \log_2 \log_2 |x|$ and each $1 \leq j \leq 2^m$ we have:

$$\left| \frac{N_j^m(x)}{|x|_m} - 2^{-m} \right| \leq \frac{1}{\log_2 |x|}. \tag{2.1}$$

Almost all algorithmic random strings are Borel normal with accuracy $\frac{1}{\log_2 |x|}$ [44]; in particular, they have approximately the same number of 0s and 1s. Furthermore,

[3] The sequences were obtained from 10 longer sequences of 2^{29} bits, each obtained during separate experimental runs. We split them further into smaller sequences in order to provide a more detailed statistical analysis.

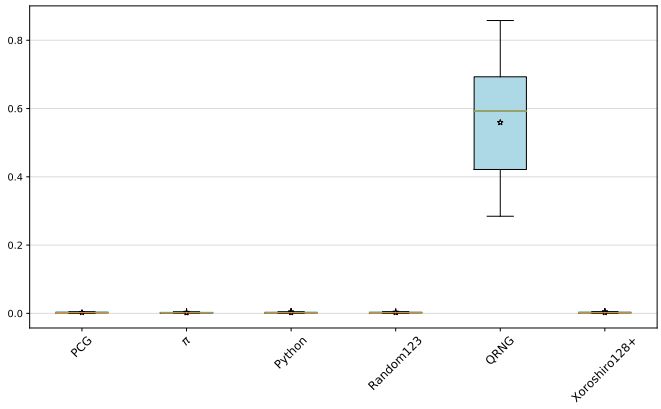


Figure 2.2: Borel normality test: Box-plot showing the distribution of the quantity $\max\left(\left|\frac{N_j^m(x)}{|x|_m} - 2^{-m}\right|\right) \log_2 |x|$ for the 80 strings of length $|x| = 2^{26}$ bits produced by each the six RNGs tested.

if all prefixes of a sequence are Borel normal, then the sequence itself is also Borel normal.

The fact that Borel normality for finite strings is only defined up to the accuracy function arises from the fact that the definition is well behaved (and converges to the definition for sequences in the limit) if the right-hand-side of Eq. 2.1 is replaced by any decreasing computable real function in $|x|$ converging to 0. Fixing a specific accuracy function allows one to test explicitly the normality of finite sequences (and such tests have previously been performed on strings produced by QRNGs [49, 139, 114]), but such a choice of accuracy function is necessarily somewhat arbitrary. However, the relative normality of strings can be tested by comparing the values of a metric based on 2.1; a reasonable choice of such a metric is the quantity $\max\left(\left|\frac{N_j^m(x)}{|x|_m} - 2^{-m}\right|\right) \log_2 |x|$ over the values $m = 1, \ldots, \lfloor \log_2 \log_2 |x| \rfloor$ and each $1 \leq j \leq 2^m$. We recorded this metric for the six sources of random bits under consideration, and the resulting distributions are shown in Figure 2.2.

The results show clearly that the bits produced by the QRNG are significantly less normal than those produced by the other sources. This is, however, not surprising, since the experiment implementing the QRNG was known to exhibit bias due to experimental imperfections [104]. Although it is possible to use normalisation procedures to unbias a source, simple techniques significantly reduce the length of the output strings (and thus the obtainable statistical power) and can alter various computability theoretic properties of a sequences [5]; conversely, the consequences of more complicated techniques on such properties of the output are poorly understood, so we opted against performing any such normalisation. Nonetheless, as

discussed at the end of Section 2.2, a sufficiently small bias may be less problematic in practical applications for QRNGs than for traditional PRNGs.

While examining the normality of sequences produced by any RNG is important, this algorithmic property fails to test properties of algorithmic randomness or incomputability in the way we aim to do. The example of Champernowne's sequence again testifies to this. To probe such behaviour of QRNGs we thus need to delve further into algorithmic properties of randomness.

2.3.2 A Martin-Löf test of incomputability

Is it possible to give formally a test which rejects every computable sequence as nonrandom? Martin-Löf randomness is an important, if not the most important, form of algorithmic randomness and is based on the notion of Martin-Löf test of randomness. A test of randomness is defined by a uniformly computably enumerable shrinking sequence of constructive open sets in Cantor space (the components of the test) whose intersection is a constructive null set (with respect to Lebesgue measure); see [44] for more details. A sequence passes the test if it is not contained in this null set. A sequence is Martin-Löf random if it passes all Martin-Löf tests. There exist countably many such tests: some test normality, others test the law of large numbers, etc. The answer to the question above is affirmative: such a Martin-Löf test exists.

Testing incomputability rather that randomness directly is an important initial step: indeed, all algorithmically random sequences are incomputable and it is this property of randomness that PRNGs fail most starkly, since they necessarily produce computable sequences. Moreover, the robustness of incomputability to bias in a sequences makes such tests potentially more robust in practice. To specify a Martin-Löf test for computability, we must define the sequences contained in its nth component for all integers $n > 0$. To do so, one can take the nth component to be the union of all $\sigma\{0,1\}^*\{0,1\}^\infty$ for which there is an e such that $\sigma(0) = \varphi_e(0), \ldots, \sigma(e+n+1) = \varphi_e(e+n+1)$ and $\sigma \in \{0,1\}^*$. This is an open computably enumerable class that contains all computable sets, as each computable set has a computable characteristic function φ_e. Furthermore, the measure of the nth component is bounded from above by $\sum_e 2^{-n-e-2}$, which in turn is bounded from above by 2^{-n-1}, as the string σ derived from φ_e has length $e+n+2$ and is a prefix of the set for which φ_e computes the characteristic function.

It is not difficult to see that the above test for computability depends on the enumeration (φ_e), and there is no obvious "natural" choice. Furthermore, invariance under finite variations renders the test unsuitable for finite experiments. As a result, it is necessary to consider more indirect methods to test the incomputability of sequences produced by RNGs.

2.3.3 Chaitin-Schwartz-Solovay-Strassen tests

In this section we propose and carry out several related tests based on a rather different property of random sequences: their ability to de-randomise the Solovay-Strassen probabilistic test of primality [140]. In contrast with most standard tests of randomness which check specific properties of strings of bits, these tests are based on the behaviour of the strings with respect to certain "secondary" tasks. We first briefly describe the Solovay-Strassen primality test and the advantage offered in this task by random strings, before presenting the tests themselves.

The Solovay-Strassen test checks the primality of a positive integer n: take k natural numbers uniformly distributed between 1 and $n-1$, inclusive, and, for each $i(= i_1, \ldots, i_k)$, check whether a certain, easy to compute, predicate $W(i, n)$ holds (W is called the Solovay-Strassen predicate). If $W(i, n)$ is true then "i is a witness of n's compositeness", hence n is composite. If $W(i, n)$ holds for at least one i then n is composite; otherwise, the test is inconclusive, but in this case the probability that n is prime is greater than $1 - 2^{-k}$. This is due to the fact that *at least half* the i's between 1 and $n-1$ satisfy $W(i, n)$ if n is composite, and *none* of them satisfy $W(i, n)$ if n is prime [140].

Chaitin and Schwartz [55] proved that, if c is a large enough positive integer and s is a long enough c-Kolmogorov random binary string, then n is prime if and only if $Z(s, n)$ is true, where Z is a predicate constructed directly from $O(\log n)$ conjunctions of negations of W predicates (see Section 2.3.3 below for more details). The crucial fact is that the set of c-Kolmogorov random strings is highly incomputable: technically the set is immune, that is, it contains no infinite computably enumerable subset [44]. As a consequence, de-randomisation is thus non-constructive, and thus without practical value.

Drawing on this result, we propose several tests that operationalise it in order to test the randomness of a sequence based on whether certain numbers obtained from RNGs succeed in witnessing the compositeness of well chosen targets. We will make particular use of Carmichael numbers as these target composites. A Carmichael number is a composite positive integer n satisfying the congruence $b^{n-1} \equiv 1 \pmod{n}$ for all integers b relatively prime to n. Although Carmichael numbers are composite, they are difficult to factorise and thus are "very similar" to primes; they are sometimes called pseudo-primes. Many Carmichael numbers can pass Fermat's primality test, but less of them pass the Solovay-Strassen test. Increasingly Carmichael numbers become "rare".[4]

In what follows we thus present four different tests based on the Chaitin-Schwartz Theorem and the Solovay-Strassen test. Since the proposed tests rely directly on the algorithmic randomness of a string, they can potentially give direct

[4] There are 1,401,644 Carmichael numbers in the interval $[1, 10^{18}]$.

empirical evidence of incomputability, in stark contrast to most tests of randomness. For example, the Borel normality test discussed previously is unable to do so: the normality of Champernowne's sequence mentioned earlier is evidence of this.

Since the Chaitin-Schwartz Theorem relies on the Kolmogorov randomness of the sequences it involves, these tests also go beyond the previous one in not only probing incomputability, but also algorithmic complexity more generally. Indeed, an ideal QRNG should produce c-Kolmogorov random strings with very high probability, while PRNGs produces strings of very low Kolmogorov complexity (since, in the limit, they are computable). Nonetheless, we focus on probing the incomputability of strings from QRNGs rather than their Kolmogorov complexity or randomness, a doubly motivated choice. Firstly, the fact that incomputability is a weaker property than Kolmogorov randomness and less affected by bias means that any difference between pseudo and quantum randomness will potentially be easier to observe. Secondly, as mentioned earlier, subject to an additional physical assumption, QRNGs can be shown to produce incomputable sequences with certainty, and not just probability one [7].

As in [49], we conduct various statistical tests to determine whether any observed difference is statistically significant or not. If a difference is found to be significant, we then look at whether this really provides evidence of incomputability or not. As it is not *a priori* clear what distribution the various test metrics we employ should follow, we utilise the non-parametric and distribution free Kolmogorov-Smirnov test for two samples [59] to determine whether two datasets differ significantly. This test returns a p-value[5] indicating the probability, given the observed test statistic, that the observed distributions were indeed drawn from the same distribution. We conclude that "the difference between the two datasets is statistically significant" if the p-value is less than 0.005. We choose this relatively strict p-value to lower the chance of false positives arising from the fact that we will perform several tests between several different data sources: the probability of observing a spurious difference (simply by chance) on at least one of the many tests is much higher than the critical p-value of 0.005 of obtaining such a spurious result on any single test. A higher critical p-value (such as the commonly used 0.05) would mean such false positives would be highly probable.

When no significant difference is found by the Kolmogorov-Smirnov test, we additionally check whether the test metric distribution is consistent with a normal distribution by performing a Shapiro-Wilk test [134];[6] if it is,[7] we then use the

[5] Exact p-values are only available for the two-sided two-sample tests with no ties.

[6] More precisely, the Shapiro-Wilk test examines the null hypothesis that the samples $z_1, \ldots, z_n$ come from a normally distributed population. This test is appropriate for small samples, since it is not an asymptotic test.

[7] Here we consider evidence for non-normality to be a p-value below 0.05.

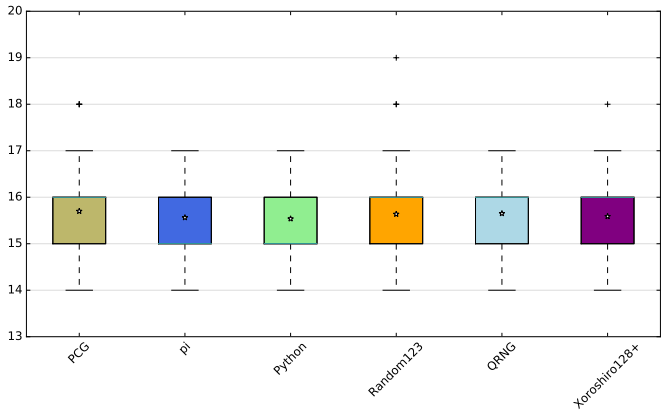


Figure 2.3: First Chaitin-Schwartz-Solovay-Strassen test on 80 samples: Box-plot showing the distribution in the minimum number of witnesses needed to verify the compositeness of all Carmichael numbers of at most 16 digits.

(parametric) Welch t-test [152], which is a version of Student's test, to determine whether there is a significant difference between the means of the test statistics for the different RNGs under the assumption of normally distributed test metrics.

First Chaitin-Schwartz-Solovay-Strassen test

The first test we look at, which was previously used in [49], probes directly the efficacy of a set of random bits in simulations (in our case for checking primality).

We performed this test on all of the 246,683 Carmichael numbers n with at most 16 digits as computed in [126], using strings of bits from each random source to specify the numbers tested as potential witnesses of compositeness. More precisely, for a fixed k (see below) and each Carmichael number n we take k strings of $\lceil \log_2 n \rceil$ bits from the source string and reject and resample those which specify the binary representation of a number greater than $n-1$. These k strings, interpreted as the binary representation of k numbers $i_1, \ldots, i_k$, serve as the witnesses to test the primality of n (i.e., the i in $W(i, n)$). Initially we take $k = 1$ and increase k until all the Carmichael numbers are correctly determined to be composite.

The metric for the test is taken to be the smallest k such that at most k witness numbers were required to obtain a verdict of non-primality for all of the Carmichael numbers. For each k, new bits are read from the sample string for each Carmichael number to be tested; we only restart reading from the start of the string (and thus recycling bits) when there was a need to try a larger value of k to pass this test.

Figure 2.3 shows the performance of the 80 bit strings from each RNG (i.e., the

same ones as tested for Borel normality in Section 2.3.1) using the metric described above.

The full results of the statistical analysis of this test (as well as the following) are given in the Appendix. The Kolmogorov-Smirnov tests found no statistical significant difference between any of the sources of randomness (see Table 2.1). The Shapiro-Wilk tests showed that the distribution of test statistics were not normally distributed (see Table 2.2), so further parametric tests were not performed. This test therefore did not provide any evidence of significant differences between the RNGs, let alone evidence of incomputability of the QRNG.

Table 2.1: Kolmogorov-Smirnov tests for the first Chaitin-Schwartz-Solovay-Strassen test with the metric that records the minimum number of witnesses needed to verify the compositeness of all Carmichael numbers of at most 16 digits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.8186	0.8186	1	1	1
π		0.9976	0.9976	0.5596	1
Python			0.9976	0.5596	0.9976
Random123				0.9976	1
QRNG					0.9780

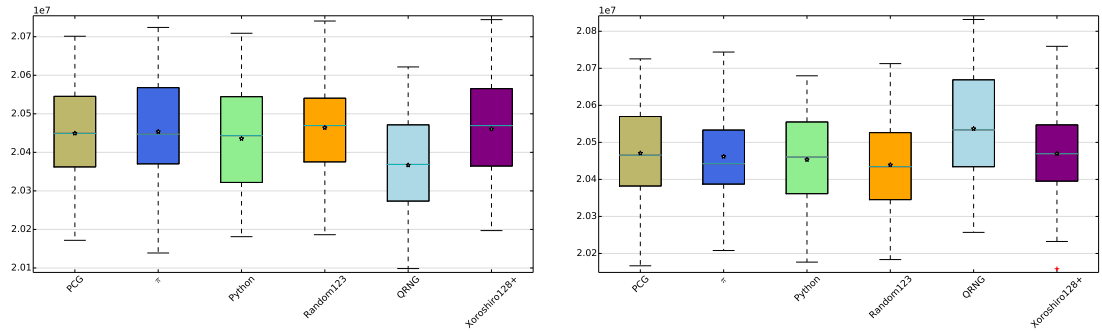
Table 2.2: Shapiro-Wilk tests of normality for the first Chaitin-Schwartz-Solovay-Strassen test with the metric that records the minimum number of witnesses needed to verify the compositeness of all Carmichael numbers of at most 16 digits.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	$< \mathbf{10^{-4}}$	$< \mathbf{10^{-4}}$	$< \mathbf{10^{-4}}$	$< \mathbf{10^{-4}}$	$< \mathbf{10^{-4}}$	$< \mathbf{10^{-4}}$

Second Chaitin-Schwartz-Solovay-Strassen test

We next consider a closely related (and similarly motivated) test with a slightly different metric. For each Carmichael number n, we repeatedly obtain a witness from the string being tested (in the same manner as in the first test and using new bits for each Carmichael number) until the compositeness of n is successfully witnessed. For this test metric we take the total number of bits used (for a given string to test) to confirm the compositeness of all 16 digit Carmichael numbers.

We calculate this as the sum, over all such Carmichael numbers n, of $\lceil \log_2 n \rceil$ times the number of Solovay-Strassen trials needed to witness the compositeness of n. (In this way, bits that are read but then rejected because they give a witness larger than n do not contribute to the metric.)



(a) (b)

Figure 2.4: Second Chaitin-Schwartz-Solovay-Strassen test: total number of bits required to verify the compositeness of all Carmichael numbers of at most 16 digits using (a) the 80 strings from each RNG, and (b) the complement of these strings.

Figure 2.4(a) shows a boxplot of the results for the 80 strings from each RNG being tested. The visible difference between the QRNG and the other sources is confirmed by the Kolmogorov-Smirnov tests (see Table 2.3), which showed a statistically significant difference between the QRNG and π, Random123 and xoroshiro128+. There is not, however, a general trend of normality for the test metric across all sources (in particular, there is weak evidence to reject normality of the distribution for the Python strings; see Table 2.4), so it is not appropriate to use Welch's t-test to look for a difference between the QRNG and Python.

Although a significant difference was found between the QRNG and most the other sources, this is not necessarily a result of the incomputability we wish to test. Indeed, we have already seen from the Borel normality test that the QRNG has a small statistical bias, so we should thus verify that the difference seen here is not also a result of this bias. As mentioned earlier, we opted against trying to normalise the data to see if the bias is indeed to origin of the effect, not only because, with the amount of data available to us, this would markedly reduce our statistical power, but also because the effect of normalisation on the algorithmic complexity of the sequence is not entirely understood. Instead, a simple way to test whether bias is behind the observed differences is to perform the same test on the complement of the strings we have tested (i.e., exchanging 0 and 1). Since this transformation preserves randomness and incomputability, if the difference observed is evidence

of such properties it should not be affected by such a transformation.

Figure 2.4(b) shows the result of the test on the complemented sequences. Here we see that again there is an apparent difference between the QRNG and some of the other sources. This is confirmed by the Kolmogorov-Smirnov tests (see Table 2.5) to be the case between the QRNG and π, Python and Random123. In this case, the test metric is consistent with being normally distributed (see Table 2.6), so it is reasonable to use Welch's t-test to try and confirm this difference further under an assumption of normality. Doing so (see Table 2.7) shows that there is indeed a statistically significant difference between the QRNG and all the other sources on the complemented strings.

Table 2.3: Kolmogorov-Smirnov tests for the second Chaitin-Schwartz-Solovay-Strassen test with the "bit counting" metric on the non-complemented (i.e., original) bits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.6953	0.4383	0.922	0.0132	0.6953
π		0.4383	0.8219	**0.0045**	0.9794
Python			0.0814	0.0537	0.5625
Random123				**0.0014**	0.5625
QRNG					**0.0026**

Table 2.4: Shapiro-Wilk tests of normality for the second Chaitin-Schwartz-Solovay-Strassen test with the "bit counting" metric on the non-complemented (i.e., original) bits.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	0.4892	0.2003	**0.04867**	0.5951	0.1669	0.0808

However, as is clear from Figure 2.4(b), this difference is in the opposite direction to (and of the same magnitude as) that in Figure 2.4(a): in the latter the QRNG appears to perform better, while in the former, it performs worse. It thus appears that this difference was indeed due to the bias of the QRNG rather than incomputability. Nonetheless, we note that it is strange that biased sequences (in particular, biased towards having more zeroes) perform better in proving the compositeness of Carmichael numbers; we are not aware of any number theoretic explanation for this.

Table 2.5: Kolmogorov-Smirnov tests for the second Chaitin-Schwartz-Solovay-Strassen test with the "bit counting" metric on the complemented bits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.4383	0.3307	0.2424	0.05372	0.5625
π		0.4383	0.1202	**0.0045**	0.5625
Python			0.5625	**0.0026**	0.8219
Random123				**0.0014**	0.2424
QRNG					0.0132

Table 2.6: Shapiro-Wilk tests of normality for the second Chaitin-Schwartz-Solovay-Strassen test with the "bit counting" metric on the complemented bits.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	0.199	0.2433	0.0754	0.4401	0.0518	0.9673

To conclude, this test shows that the QRNG behaves significantly differently from almost all the other sources on this test (whether we use either the original bits or the complemented bits), but that this difference is likely due to the bias of the QRNG. Understanding better why this bias makes such a difference would nonetheless be interesting.

Third Chaitin-Schwartz-Solovay-Strassen test

While the above tests are inspired by the Chaitin-Schwartz Theorem [55], they do not directly test the predicate $Z(s, n)$ appearing therein that we mentioned earlier. A key difference between these tests and the previous ones is the method they use to convert strings of random bits into potential witnesses to test.

Consider $s = s_0 \dots s_{m-1}$ a binary string (of length m) and n an integer greater than 2. Let k be the smallest integer such that $(n-1)^{k+1} > 2^m - 1$; we can thus rewrite the number whose binary representation is s into base $n-1$ and obtain the unique string $d_k d_{k-1} \dots d_0$ over the alphabet $\{0, 1, \dots, n-2\}$, that is,

$$\sum_{i=0}^{k} d_i (n-1)^i = \sum_{t=0}^{m-1} s_t 2^t.$$

The predicate $Z(s, n)$ is defined by

$$Z(s, J) = \neg W(1 + d_0, n) \wedge \dots \wedge \neg W(1 + d_{k-1}, n), \tag{2.2}$$

Table 2.7: Welch t-tests for the second Chaitin-Schwartz-Solovay-Strassen test with the "bit counting" metric on the complemented bits.

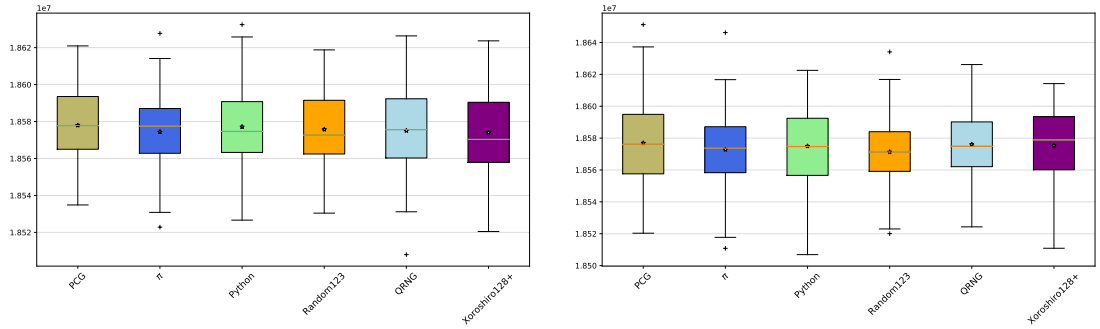
p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.6422	0.3796	0.1265	**0.0034**	0.9454
π		0.6343	0.2287	**0.0004**	0.6795
Python			0.4683	**0.0001**	0.3964
Random123				$< \mathbf{10^{-4}}$	0.1271
QRNG					**0.0020**

where W is the Solovay-Strassen predicate from Section 2.3.3. The digits of s (rewritten in base $n-1$) define the witnesses used to test the primality of n.

The main result from [55] is:

Theorem 5. *For all sufficiently large c, if s is a c-Kolmogorov random string of length $\ell(\ell+2c)$ and n is an integer whose binary representation is ℓ bits long, then $Z(s,n)$ is true if and only if n is prime.*

In order to carry out these tests we first fix c. For each Carmichael number n (with an ℓ-bit binary representation) we take $c = \ell - 1$.[8]



(a) (b)

Figure 2.5: Third Chaitin-Schwartz-Solovay-Strassen test: Box-plot showing the distribution of total number of bits used to identify all 16-digit Carmichael numbers as composite by (a) the 80 strings from each RNG, and (b) the complement of these strings.

[8] This is somewhat arbitrary; other choices could of course be made, but would make little difference to our test.

The metric of the third test has some similarities to that used in the second test. For each such n we take $\ell\,(\ell+2c)$ bits. Rewriting s in base $n-1$ as described above, we then compute $W(1+d_j, n)$ for $0 \leq j \leq k$ until the first j is found such that $W(1+d_j, n)$ holds (and the compositeness of n is thus witnessed). The metric itself is then taken as the sum (over all 16-digit Carmichael numbers n tested) of $j \times \lceil \log_2(n-1) \rceil$. Note that, if no first $j \leq k$ is found such that $W(1+d_j, n)$ holds (which occurs very rarely), then we simply count all the bits used when testing that Carmichael number, i.e., $\ell\,(\ell+2c)$. Figures 2.5(a) shows the performance of the 80 strings from each of the six sources according to this metric. In order to be able to do decouple any potential difference between the QRNG and the other sources due to algorithmic randomness from those resulting from the bias of the QRNG, we similarly perform the same test on the complement of each of the strings, the results of which are shown in Figure 2.5(b).

The results of the Kolmogorov-Smirnov tests on the data shown in Figures 2.5(a) and 2.5(b) are given in Tables 2.8 and 2.11, respectively. No statistically significant differences between any of the sources were found, reinforcing the impression given by Figure 2.5 that the RNGs all give similar results. The Shapiro-Wilk test shows (see Tables 2.9 and 2.12) that there is no strong evidence against the normality of test metric for the non-complemented strings (but there was weak evidence against it for the complemented ones), so we were able to use Welch's t-test to look for any further evidence of differences between the sources on these strings (see Table 2.10). No significant differences between the sources were found by these tests either. We therefore conclude that the third Chaitin-Schwartz-Solovay-Strassen test with this metric, which counts the total number of bits required to verify the compositeness of all Carmichael numbers of at most 16 digits, failed to find significant differences between the QRNG and the PRNGs tested.

Table 2.8: Kolmogorov-Smirnov tests for the third Chaitin-Schwartz-Solovay-Strassen test with the "bit-counting" metric for the non-complemented (i.e., original) bits for all Carmichael numbers of at most 16 digits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.2694	0.4821	0.2988	0.4013	0.1054
π		0.6953	0.4383	0.3307	0.4383
Python			0.8186	0.5625	0.5625
Random123				0.9794	0.8219
QRNG					0.8219

Table 2.9: Shapiro-Wilk tests of normality for the third Chaitin-Schwartz-Solovay-Strassen test with the "bit-counting" metric for the non-complemented (i.e., original) bits for all Carmichael numbers of at most 16 digits.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	0.2076	0.4921	0.3337	0.1956	0.7608	0.1347

Table 2.10: Welch t-tests for the third Chaitin-Schwartz-Solovay-Strassen test with the "bit-counting" metric for the non-complemented (i.e., original) bits for all Carmichael numbers of at most 16 digits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.2838	0.81	0.5227	0.4335	0.2437
π		0.4186	0.6833	0.8401	0.911
Python			0.6956	0.584	0.3653
Random123				0.8585	0.6096
QRNG					0.7629

Table 2.11: Kolmogorov-Smirnov tests for the third Chaitin-Schwartz-Solovay-Strassen test with the "bit-counting" metric for the complemented bits for all Carmichael numbers of at most 16 digits.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.5596	0.9794	0.173	0.9794	0.3307
π		0.922	0.8219	0.8219	0.6953
Python			0.5625	0.9194	0.6953
Random123				0.4383	0.1201
QRNG					0.8219

Fourth Chaitin-Schwartz-Solovay-Strassen test

The final test is based more closely on the Chaitin-Schwartz Theorem out of the tests we consider. Rather than looking at how many witnesses need to be tested until a Carmichael number's compositeness is verified, we look directly at the ability of the entire *set* of witnesses evaluated in 2.2 to verify the compositeness of a number. In other words, we look for direct violations of the Chaitin-Schwartz Theorem: a violation appears when for all $j = 0, \ldots, k-1$, $W(1+d_j, n)$ are false;

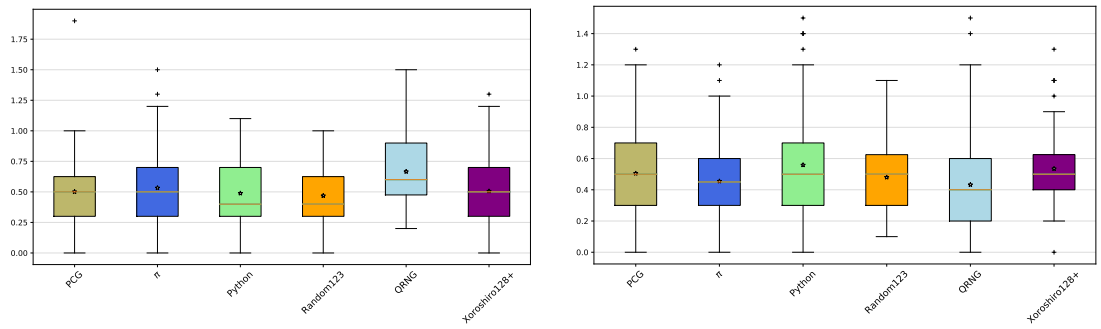
Table 2.12: Shapiro-Wilk tests of normality for the third Chaitin-Schwartz-Solovay-Strassen test with the "bit-counting" metric for the complemented bits for all Carmichael numbers of at most 16 digits.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	0.4616	0.6708	0.6067	0.94	0.9355	**0.0239**

that is, all tests wrongly conclude that n is "probably prime".

However, as the Solovay-Strassen test guarantees that $W(1+d_j, n)$ is true with probability at least one half when n is a composite number, it quickly becomes difficult, in practice, to observe such violations for even the smallest Carmichael numbers used in the previous tests. In order to observe some violations with the length of random strings (and time) we have access to, we have to severely restrict ourselves and be content with testing the performance of the strings on only the odd composite numbers less than 50: $9, 15, 21, 25, 27, 33, 35, 39, 45, 49$. For these numbers, we compute $Z(s, n)$ by reading $\ell(\ell + 2c)$ bits and following the same procedure as in the third test. When $Z(s, n) = 1$, a violation of the Chaitin-Schwartz Theorem is thus observed. Since testing this predicate a single time on the ten numbers above would give insufficient statistics to observe any difference between the sources, we then repeated the above procedure reading from then 2nd bit of each string, then the 3rd, etc., until all the random bits have been used. The metric is thereby taken as the average number of violations observed for the 10 composites tested (where the average is taken over all the repetitions). Figures 2.6(a) and 2.6(b) show the results of this test for the 80 strings of each of the six sources used in the previous tests: again, the tests in the former figure use the original strings from each source while the tests in the latter use the complemented strings.

We apply the same statistical tests to determine whether there are any statistically significant differences in performance between the different RNGs. The results of the Kolmogorov-Smirnov tests for the data in Figures 2.6(a) and 2.6(b) are given in Tables 2.13 and 2.15, respectively. Unlike the results for the previous metrics, the QRNG exhibits significantly different behaviour on the original (i.e., non-complemented) strings from the PCG, Python and Random123 PRNGs. However, no significant difference is found on any of the complemented strings. The Shapiro-Wilk tests (see Tables 2.14 and 2.16) find strong evidence against the normality of the distribution of the test metric, so Welch's t-test was not applied to see if further evidence of significant differences was present.



(a) (b)

Figure 2.6: Fourth Chaitin-Schwartz-Solovay-Strassen test: Box-plot showing the distribution of the average count of violations of the Chaitin-Schwartz Theorem for all odd composite numbers less than 50 by (a) the 80 strings from each RNG, and (b) the complement of these strings.

Table 2.13: Kolmogorov-Smirnov tests for the fourth Chaitin-Schwartz-Solovay-Strassen test with the "violation-count" metric for non-complemented (i.e., original) bits for all odd composite numbers that are less than 50.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.318	0.2414	0.692	**0.0027**	0.9976
π		0.692	0.8186	0.05397	0.9976
Python			0.9194	**0.0004**	0.8186
Random123				**0.0047**	0.8186
QRNG					0.0348

Table 2.14: Shapiro-Wilk tests of normality for the fourth Chaitin-Schwartz-Solovay-Strassen test with the "violation-count" metric for non-complemented (i.e., original) bits for all odd composite numbers that are less than 50.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	$< \mathbf{10^{-4}}$	**0.0040**	**0.0002**	**0.0056**	**0.0115**	**0.0148**

Table 2.15: Kolmogorov-Smirnov tests for the fourth Chaitin-Schwartz-Solovay-Strassen test with the "violation-count" metric for the complemented bits for all odd composite numbers that are less than 50.

p-values	π	Python	Random123	QRNG	xoroshiro128+
PCG	0.692	0.9194	0.9194	0.1725	0.5596
π		0.5596	0.9976	0.692	0.2414
Python			0.692	0.1725	0.8186
Random123				0.5596	0.5596
QRNG					0.0135

Table 2.16: Shapiro-Wilk tests of normality for the fourth Chaitin-Schwartz-Solovay-Strassen test with the "violation-count" metric for the complemented bits for all odd composite numbers that are less than 50.

	PCG	π	Python	Random123	QRNG	xoroshiro128+
p-value	0.06601	**0.02957**	$\mathbf{< 10^{-4}}$	**0.0080**	$\mathbf{< 10^{-4}}$	**0.0017**

Again, the reason for the apparently significant differences in performance between the QRNG and some of the sources (at least for the non-complemented strings) is unclear, and further investigation is required. The fact that only very small composite numbers were able to be tested means that, in the absence of strong evidence of differences between the sources, the results should be interpreted cautiously. Indeed, the Chaitin-Schwartz Theorem is an asymptotic result, and a significant difference on larger composites (ideally Carmichael numbers), would be preferable. We thus cautiously conclude that the fourth Chaitin-Schwartz-Solovay-Strassen test with the violation-count metric potentially identifies differences between QRNGs and the other sources, but that further testing and study is needed to confirm the robustness of the initial results observed here.

2.4 Conclusions

In this chapter, we study the ability to formulate tests that probed the incomputability and algorithmic randomness of strings produced by QRNGs. Standard tests used to assess the quality of strings produced by PRNGs and QRNGs alike focus on simple statistical properties of the sequences, yet the most marked differences between QRNGs and PRNGs are the algorithmic properties of strings pro-

duced by such devices. Such tests thus provide an important and novel approach to evaluating the performance of QRNGs. This type of test, which probes the randomness of *outputs* of QRNGs, is complementary to the certification of a QRNG as exploiting random *processes*, either via theoretical analysis of the device [13, 7] or the use of device-independent randomness expansion schemes [127, 58].

The properties of incomputability (and, consequently, of algorithmic randomness too) mean that one must resort to indirect tests of incomputability in practice, and we discussed several such approaches. We considered testing the Borel normality of sequences—a necessary property of algorithmic randomness—which probes the bias of a sequence rather than its incomputability *per se*. This served as a useful preliminary probe for the analysis of later tests. We then focused on a different approach based around the Chaitin-Schwartz Theorem, which shows a practical consequence of algorithmic randomness in probabilistic primality testing algorithms. We proposed four different tests based on this result which, in principle, could exhibit advantages due to the incomputability—as well as the algorithmic randomness—of sequences from QRNGs over PRNGs.

To assess the practical utility of these tests, we applied them to long sequences generated by various RNGs: a QRNG (described in Section 2.1), and several different PRNGs. Two of the tests (the first and the third) failed to find any significant differences between the QRNG and the PRNGs. A significant difference was, on the other hand, observed, for the second test. However, we were able to show that the difference was due to a small bias present in the strings produced by the QRNG rather than a result of any incomputability. Indeed, this highlighted a key challenge: the need to decouple the incomputability from the bias within the test results, since the tests can in general be affected by both these elements. To this end, we examined the performance of tests on the complement of the strings as well as the strings themselves, but conclude that care should be taken to formulate tests that are not affected by the bias of a sequence. This task is complicated, however, by the fact that the effect of using a biased distribution in probabilistic primality testing is not well understood theoretically. For future studies, it would thus be desirable to have sufficiently long strings to analyse from a certified QRNG for which the bias is sufficiently small so as to not be a limiting factor for the tests. Conversely, one should also study further the effect of normalisation procedures [149, 5] on such metrics, so that such tests can be properly analysed when applied to normalised, rather than raw, sequences of bits.

Our fourth test, which was designed to follow more faithfully the Chaitin-Schwartz Theorem and to be potentially more robust to bias (but, unfortunately, more demanding to apply in practice), produced ambiguous results. In particular, significant differences were found only on the non-complemented strings, but it was not clear whether these differences were entirely due to bias, as one would expect

the complemented strings to show a similar difference in the opposite direction, which was not observed. Due to the practical limitations of this test and small numbers tested, further testing (and, probably, refinements of the test itself) on more data are needed to understand this effect better.

While our tests failed to find any conclusive experimental evidence of incomputability of quantum randomness, they provide an important study for the development of new types of tests aimed at probing algorithmic properties of quantum randomness. Indeed, being based on the Chaitin-Schwartz Theorem, the tests in fact probe the stronger property of c-Kolmogorov randomness, and this fact potentially contributes to the difficulty in observing indirect effects of incomputability. The development of further tests to this end, as well as additional experimental studies, are therefore merited.

We finish by reiterating that tests of the output of QRNGs, such as we describe here, complement, rather than replace, the certification by value indefiniteness of a QRNG. Indeed, just as with standard statistical tests of RNGs, even if no difference between QRNGs and PRNGs is found on the tests, it is important that QRNGs are verified to pass such tests. QRNGs certified by the Kochen-Specker Theorem, such as the one used to provide the data for this paper [104], can also be used to perform device-independent randomness expansion [145, 71, 119], and it would be interesting to combine algorithmic tests like we develop here with such an approach, although producing sufficient amounts of data for this to be possible remain a challenge.

All the test data (i.e., random strings), programs and results are available online in [9].

Chapter 3

Solving the Maximum Common Subgraph Isomorphism Problem via Quantum Annealing

3.1 Introduction

Quantum computing is a field of increasing interest to computer scientists that involves taking advantage of quantum effects to perform computations that differ significantly from those of classical computers. Certain types of problems, such as simulations of quantum systems, which are considered to be too hard for classical computers, may be solved by quantum computers within a reasonable time. Even though it is still debatable if exponential speed-up on certain types of problems can be achieved by quantum computers, the possibility of a huge advantage still drives people to develop new types of quantum machines. In this area, developing quantum algorithms is essential.

D-Wave machines are based on a process known as quantum annealing. These quantum computers solve discrete optimisations problems like Ising problem or the Quadratic Unconstrained Binary Optimisation (QUBO) problem. D-Wave machines take advantage of quantum tunnelling to find the global minimum of objective functions. Various problems can be reformulated as QUBOs, so they can be solved by these machines [37, 120, 124]. This method has the potential to solve hard graph problems, such as the graph isomorphism problem [155], graph partitioning [146] and many others [57, 74, 144].

Subgraph isomorphism encompasses both the maximum clique problem and the task of determining the presence of a Hamiltonian cycle in a graph. As a result, it falls into the category of NP-complete problems. Cook's seminal paper from 1971 [61], which established the Cook-Levin theorem, also demonstrated the

NP-completeness of subgraph isomorphism. This was accomplished by employing a reduction from 3-SAT that involved cliques. Since finding a subgraph isomorphism is NP-complete, and the MCS problem is a more complex version of this, it inherits the NP-completeness property [83]. The maximum common subgraph isomorphism problem has a wide impact in bioinformatics [39], alignment of chemical structures [76], computer vision and pattern matching [60, 69], etc. By solving this problem, we get the largest common subgraph of the two input graphs. There are two variations of this problem that have been studied a lot [39, 76]: the maximum common induced subgraph isomorphism (MCISI) problem and the maximum common edge subgraph isomorphism (MCESI) problem.

However, both MCISI and MCESI do not consider the ratio of the number of common edges to common vertices. We introduce restrictions on how dense the common subgraphs are (a graph is denser when it has more edges with the same number of vertices). By maximising the density of the common subgraph, we can obtain a common subgraph that is more closely connected. The k-densest subgraph problem aims to identify the subgraph with the highest density, precisely on k vertices. This problem serves as a generalization of the clique problem and, consequently, is NP-hard in arbitrary graphs. Our problem of k-densest common subgraph isomorphism is at least as hard as the k-densest subgraph problem, since we are seeking the k-densest common subgraph shared by the two input graphs.

In this chapter we first propose QUBO formulations for the problems MCISI, MCESI and k-densest maximum common subgraph isomorphism. We first give their definitions and their QUBO formulations. Then we prove the correctness of these formulations. Finally, we run MCISI and MCESI algorithms on the D-Wave 2X machine instances of graphs that represent some small organic molecules and analyse the experimental data. We also give a short discussion for the reason why k-densest maximum common subgraph isomorphism algorithm is not suitable to run on D-Wave 2X machines.

3.2 QUBO formulations for the maximum common subgraph isomorphism problem

In [50] QUBO formulations for the subgraph isomorphism problem and the induced subgraph isomorphism problem are provided. An improved QUBO formulation that reduces the number of variables was proposed in [154]. Solving the graph isomorphism problem decides whether the two graphs are isomorphic. Solving the subgraph (induced subgraph) isomorphism problem decides whether one graph is isomorphic to a subgraph (induced subgraph) of the other graph. A simple example of these three problems are shown in Figure 3.1. In Figure 3.1(a), the two graphs

are isomorphic to each other. In Figure 3.1(b), there is an isomorphism between the left graph and an induced subgraph of the right graph. In Figure 3.1(c), the left graph is isomorphic to a subgraph of the right graph. However, the solutions of all three problems cannot show the common substructure shared by two graphs of arbitrary sizes. In order to obtain this piece of information, we can solve the maximum common subgraph isomorphism problem to determine the largest shared sub-structure. We will build our QUBO formulation for the maximum common subgraph isomorphism problem by extending the idea in [50]. Before we show our QUBO formulation, we first describe the notation that we will use, then we give a detailed definition of the problems.

3.3 Definition and notation

We denote the integers modulo n by $\mathbb{Z}_n = \{0, \ldots, n-1\}$ and a repeated cross product by

$$\underbrace{\mathbb{Z}_n \times \cdots \times \mathbb{Z}_n}_{m \text{ times}} = \mathbb{Z}_n^m.$$

The domain of f, denoted $\mathrm{Dom}(f)$, is all $x \in X$ where $f(x)$ is defined and the image of f, denoted $\mathrm{Im}(f)$, is the set of $f(x)$ for all $x \in \mathrm{Dom}(f)$. A partial function $f : X \dashrightarrow Y$ is called a partial (total) function if $\mathrm{Dom}(f) \subseteq A$ ($\mathrm{Dom}(f) = A$).

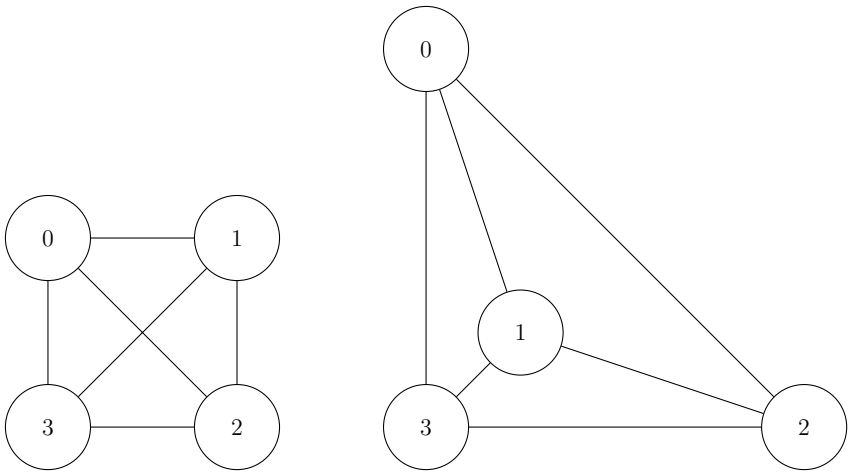
Definition 6. *A labeled graph (adding weights to a simple graph) can be represented by a four-tuple $G = (V, E, \alpha, \beta)$, where*

- *V is the set of vertices,*
- *$E \subseteq V \times V$ is the set of edges,*
- *$\alpha : V \to L_V$ is a function assigning labels to vertices,*
- *$\beta : E \to L_E$ is a function assigning labels to edges.*

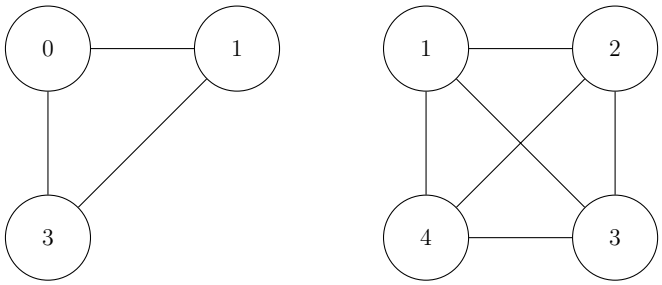
If $u, v \in V$ then we denote an edge connecting u and v by uv. Alternatively, we say $(u, v) \in E$.

Definition 7. *Given a graph $G = (V, E, \alpha, \beta)$, a subgraph of G is a graph $S = (V_S, E_S, \alpha_S, \beta_S)$, where*

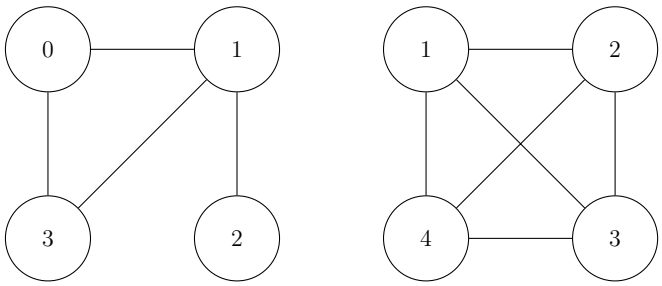
- *$V_S \subseteq V$,*
- *$E_S \subseteq E \cap (V_S \times V_S)$,*
- *$\alpha_S = \alpha\,|_{V_S}$,*



(a) Isomorphic graphs



(b) Graphs that shares an isomorphic induced subgraph



(c) Graphs that shares an isomorphic subgraph

Figure 3.1: Examples of isomorphic graphs, isomorphic induced subgraph and isomorphic subgraph

- $\beta_S = \beta \mid_{E_S}$.

Definition 8. *Given a graph $G = (V, E, \alpha, \beta)$, an induced subgraph of G is a graph $I = (V_I, E_I, \alpha_I, \beta_I)$, where*

- $V_I \subseteq V$,
- $E_I = E \cap (V_I \times V_I)$,
- $\alpha_I = \alpha \mid_{V_I}$,
- $\beta_I = \beta \mid_{E_I}$.

Definition 9. *A bijective function $\psi : V_1 \to V_2$ is a graph isomorphism between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if*

- *$\alpha_1(v_1) = \alpha_2(\psi(v_1))$ for all $v_1 \in V_1$ and $\alpha_2(v_2) = \alpha_1(\psi^{-1}(v_2))$ for all $v_2 \in V_2$,*
- *for any edge $e_1 = (v_1, v_1') \in E_1$, there exists an edge $e_2 = (\psi(v_1), \psi(v_1')) \in E_2$, such that $\beta_1(e_1) = \beta_2(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_2$, there exists an edge $e_1 = (\psi^{-1}(v_2), \psi^{-1}(v_2')) \in E_1$, such that $\beta_1(e_1) = \beta_2(e_2)$*

Definition 10. *An injective function $\phi : V_1 \to V_2$ is a subgraph isomorphism between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if there exists a subgraph S of the graph G_2, such that ϕ is a graph isomorphism between G_1 and S.*

Definition 11. *An injective function $\Phi : V_1 \to V_2$ is an induced subgraph isomorphism between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if there exists an induced subgraph I of the graph G_2, such that Φ is a graph isomorphism between G_1 and I.*

Definition 12. *Given graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, S is a common subgraph of both G_1 and G_2 if there exists a subgraph isomorphism between S and G_1, and a subgraph isomorphism between S and G_2.*

Definition 13. *Given graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, I is a common induced subgraph of both G_1 and G_2 if there exists an induced subgraph isomorphism Φ between I and G_1, and an induced subgraph isomorphism between I and G_2.*

Next, we give the definition for the maximum common subgraph isomorphism. Since it is both meaningful to find the maximum common induced subgraph and the maximum common subgraph, we give the definitions for both choices. For the maximum common induced subgraph isomorphism, it is enough to maximise the number of vertices. For the maximum common subgraph isomorphism, it is more meaningful to maximise the number of edges, otherwise we are likely to end up with a solution that contains only isolated vertices.

Definition 14. *The maximum common induced subgraph (MCIS) is the common induced subgraph between* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *and* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *with the largest number of vertices compared to the other common induced subgraphs of* G_1 *and* G_2*.*

Definition 15. *The maximum common edge subgraph (MCES) is the common subgraph between* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *and* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *with the largest number of edges compared to the other common subgraphs of* G_1 *and* G_2*.*

Observation 16. *If the graphs* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *and* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *have a maximum common induced subgraph* $I = (V_I, E_I, \alpha_I, \beta_I)$*,* Φ_1 *is an induced subgraph isomorphism from* I *to* G_1 *and* Φ_2 *is an induced subgraph isomorphism from* I *to* G_2*, then the maximum common induced subgraph isomorphism (MCISI) is a bijective partial function* $\gamma : V_1 \dashrightarrow V_2$*. For every* $v_1 \in V_1$*, we have*

$$\gamma(v_1) = \begin{cases} \Phi_2(\Phi_1^{-1}(v_1)), & \text{if } v_1 \in \mathrm{Im}(\Phi_1), \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{3.1}$$

Observation 17. *If the graphs* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *and* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *have a maximum common edge subgraph* $S = (V_S, E_S, \alpha_S, \beta_S)$*,* ϕ_1 *is a subgraph isomorphism from* S *to* G_1 *and* ϕ_2 *is a subgraph isomorphism from* S *to* G_2*, then the maximum common edge subgraph isomorphism (MCESI) is a bijective partial function* $\gamma : V_1 \to V_2$*. For every* $v_1 \in V_1$*, we have*

$$\gamma(v_1) = \begin{cases} \phi_2(\phi_1^{-1}(v_1)), & \text{if } v_1 \in \mathrm{Im}(\phi_1), \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{3.2}$$

Next we give some examples and explain when MCISI or MCESI should be used. In Figure 3.2 we have two input graphs, so solving the MCISI may give us a mapping from 0 to a, 1 to b and 3 to d, while solving the MCESI may give us a mapping from 0 to a, 1 to b, 2 to c and 3 to d. Note that the above solutions are not unique. The solutions are different because MCISI requests the common subgraph to be an induced subgraph of both input graphs and MCESI does not. MCISI has a tighter restriction due to the requirement of being induced subgraph. MCISI can be used when we are looking for exactly the same sub-structure while we will use MCESI for detecting a similar sub-structure.

Definition 1. *Given graphs* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *and* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$*,* S *is a common subgraph of both* G_1 *and* G_2 *if there exists a subgraph isomorphism between* S *and* G_1*, and a subgraph isomorphism between* S *and* G_2*.*

Before we show our QUBO formulation, we first describe the notation that we will use, then we give a detailed definition of the problems.

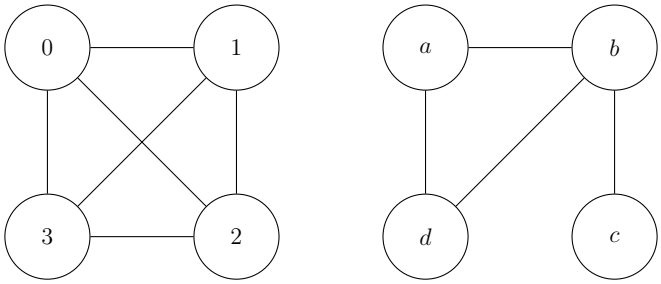


Figure 3.2: An example to show the difference between MCISI and MCESI.

Definition 2. *The density of the graph* $G = (V, E)$ *is defined by*

$$\mathrm{DENS}(G) = \frac{|E|}{|V| \cdot (|V| - 1)/2}.$$

A QUBO objective function is a quadratic binary function. We can extract a QUBO matrix from the objective function by setting the binary variables as row and column of the matrix and the coefficient of each quadratic term as the corresponding entry. The QUBO matrix can be view as an adjacency matrix for a graph. The variables are vertices while the products of variables are edges. Since the D-Wave systems use the Chimera graph as their hardware graph (host graph), a minor-embedding is needed to be done in order to map the QUBO graph (guest graph) into the host graph. Details for minor-embedding can be found in [65].

3.4 Classical algorithms to solve the maximum common subgraph isomorphism problem

Since the maximum common subgraph isomorphism problem can be reduced to the maximal clique problem, a lot of work has been done via this reduction. A clique of a graph is a subgraph of that graph, such that all pairs of vertices in the subgraph have an edge to connect them. A maximal clique is a clique of the graph that has the largest number of vertices. In order to reduce the maximum common subgraph isomorphism to the maximal clique problem, a compatibility graph needs to be constructed first. A compatibility graph is a graph generated from two input graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The vertex set is the set $V_1 \times V_2$. Two vertices (v_1, u_1), (v_2, u_2), $v_1, u_1 \in V_1, v_2, u_2 \in V_2$ are adjacent when $v_1u_1 \in E_1$ and $v_2u_2 \in E_2$, or $v_1u_1 \notin E_1$ and $v_2u_2 \notin E_2$. Then by solving the maximal clique problem of the compatibility graph gives us the solution for the maximum common subgraph isomorphism problem [27, 106].

Another class of algorithms uses backtracking algorithms to solve the problem. Since the isomorphism can be represented by vertex mappings, if we build up the

mapping by adding one vertex at a time, then a tree structure can be generated that backtracking algorithms help reduce the time of traversal. Algorithms based on backtracking are discussed in [53, 148].

The works above are exact solvers for the maximum common subgraph problem. There are also heuristic solvers that are faster but only give approximate results. Genetic algorithms are one important class in this paradigm. A genetic algorithm uses tools such as crossover, mutation, clone, etc. to generate a new population from the old ones. A fitness function is defined to make sure the new population has higher fitness. The speed and performance of this approach depend on the design of the fitness function. Different fitness functions are developed in [150, 41, 147]. Funabiki and Kitamichi [82] have used another combinatorial optimisation method that they call two-Stage Discrete Optimisation Method. They have used a greedy algorithm to find candidates and then a randomised discrete method to refine the result. Simulated annealing has been used by Barakat and Dean in [26].

3.5 QUBO formulation for MCISI

Starting with the definition of MCISI, we will design penalty terms in the QUBO objective function according to the features of MCISI. Let us work with the two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$. We denote the MCIS of G_1 and G_2 as $I = (V_I, E_I, \alpha_I, \beta_I)$ and the MCISI from G_1 to G_2 as γ. That means there exists an induced subgraph $I_1 = (V_{I_1}, E_{I_1}, \alpha_{I_1}, \beta_{I_1})$ of G_1 and an induced subgraph $I_2 = (V_{I_2}, E_{I_2}, \alpha_{I_2}, \beta_{I_2})$ of G_2, such that:

- $\alpha_{I_1}(v_1) = \alpha_{I_2}(\gamma(v_1))$ for all $v_1 \in V_{I_1}$, and $\alpha_{I_2}(v_2) = \alpha_{I_1}(\gamma^{-1}(v_2))$ for all $v_2 \in V_{I_2}$ (injective);
- for any edge $e_1 = (v_1, v_1') \in E_{I_1}$, there exists an edge $e_2 = (\gamma(v_1), \gamma(v_1')) \in E_{I_2}$, such that $\beta_{I_1}(e_1) = \beta_{I_2}(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_{I_2}$, there exists an edge $e_1 = (\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \in E_{I_1}$, such that $\beta_{I_1}(e_1) = \beta_{I_2}(e_2)$ (edge-preserving);
- if any pair of vertices $(v_1, v_1') \notin E_{I_1}$, then $(\gamma(v_1), \gamma(v_1')) \notin E_{I_2}$, and if any pair of vertices $(v_2, v_2') \notin E_{I_2}$, then $(\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \notin E_{I_1}$ (non-edge-preserving).

We then build the QUBO objective function of MCISI by translating those three features into penalty terms, which are terms with positive coefficients. We design penalty terms so that any assignments that do not fit into these features will add a positive value into the final result; they will not be picked as the solution of the problem since a D-Wave machine will choose the assignments that give

the minimum value of the objective function. Let $V_1 = \{u_0, u_1, \ldots, u_{n_1-1}\}$ and $V_2 = \{v_1, v_2, v_{n_2-1}\}$. For every $u \in V_1$, $v \in V_2$, a variable $x_{u,v}$ is used to represent the possible vertex mapping between u and v. Let $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ with

$$\mathbf{x} = (x_{u_0,v_0}, x_{u_0,v_1}, \ldots, x_{u_0,v_{n_2-1}}, x_{u_1,v_0}, \ldots, x_{u_{n_1-1},v_{n_2-2}}, x_{u_{n_1-1},v_{n_2-1}}).$$

Then we define a function $\tau : V_1 \times V_1 \times V_2 \times V_2 \to \mathbb{Z}$ to identify the possible mapping between an edge in G_1 to an edge in G_2 with respect to the vertex labels and edge labels. Let $e_1 = (u, u')$, $u, u' \in V_1$ and $e_2 = (v, v')$, $v, v' \in V_2$,

$$\tau(u, u', v, v') = \begin{cases} 0, & \alpha_1(u) = \alpha_2(v), \alpha_1(u') = \alpha_2(v') \text{ and } \beta_1(e_1) = \beta_2(e_2), \\ 1, & \text{otherwise.} \end{cases} \tag{3.3}$$

The pre-calculated function τ can reduce the density of the QUBO matrix concerning the structure of the input graphs. This is important for solving the problem on the D-Wave 2X machine.

We define the objective function by

$$F : \mathbb{Z}_2^{n_1 n_2} \to \mathbb{R},$$

$$F(\mathbf{x}) = A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] - BM(\mathbf{x}), \quad A, B \in \mathbb{R}^+, \tag{3.4}$$

where

$$H(\mathbf{x}) = \sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) + \sum_{v \in V_2} \sum_{u \in V_1} \left(x_{u,v} \sum_{\substack{u' \in V_1 \\ u' \neq u}} x_{u',v} \right), \tag{3.5}$$

$$P(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \tau(u, u', v, v') \right), \tag{3.6}$$

$$N(\mathbf{x}) = \sum_{uu' \notin E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} e_{v,v'} \right), \tag{3.7}$$

$$M(\mathbf{x}) = \sum_{u \in V_1} \sum_{v \in V_2} x_{u,v}. \tag{3.8}$$

We request B to be sufficiently smaller than A, which is needed in later proofs. Here $e_{v,v'}$ can be calculated by $e_{v,v'} = 1$ if $vv' \in E_2$ and $e_{v,v'} = 0$ otherwise.

The function $H(\mathbf{x})$ is designed to ensure that the mapping is injective. $P(\mathbf{x})$ ensures that the mapping is edge-preserving. Penalties are only applied to impossible-edge mappings with respect to both the vertex label and the edge label. With the help of τ, some of the entries of the QUBO matrix may be updated to zero. $N(\mathbf{x})$

ensures that the mapping is non-edge-preserving. $M(\mathbf{x})$ ensures that the number of vertices is maximised. All these claims will be shown in later proofs.

The following part is to show that this objective function indeed solves the maximum common induced subgraph isomorphism problem. We define $\mathcal{F}$ to be the set of all common subgraph isomorphisms between G_1 and G_2. We now specify a decoder function

$$D : \mathbb{Z}_2^{n_1 n_2} \dashrightarrow \mathcal{F},$$

the purpose of which is to interpret the isomorphism encoded in $\mathbf{x}$ so that we can obtain a vertex mapping. The domain of D contains all vectors $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ that can be 'decoded' into such functions. That is

$$\mathrm{Dom}(D) = \left\{ \mathbf{x} \in \mathbb{Z}_2^{n_1 n_2} \;\middle|\; \sum_{v \in V_2} x_{u,v} \leq 1, \text{ for all } u \in V_1 \right. \\ \left. \text{and } \sum_{u \in V_1} x_{u,v} \leq 1, \text{ for all } v \in V_2 \right\}$$

and

$$D(\mathbf{x}) = \begin{cases} \psi, & \text{if } \mathbf{x} \in \mathrm{Dom}(D), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Here $\psi : V_1 \dashrightarrow V_2$ is a partial function that for every $u \in V_1$, $\psi(u) = v$, $u \in V_1$ and $v \in V_2$, if and only if $x_{u,v} = 1$.

Lemma 18. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective.*

Proof. We first show that if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

Since $H(\mathbf{x})$ only contains the sum of products of two binary variables, which are non-negative, $H(\mathbf{x}) = 0$ if and only if all products have value zero.

The first part of $H(\mathbf{x})$, $\sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$ if and only if all products have value zero. For every fixed u, the products can only be in the following three cases:

1. all elements of $\{x_{u,v} | v \in V_2\}$ have value zero;
2. exactly one element of $\{x_{u,v} | v \in V_2\}$ has value one;
3. more than one element of $\{x_{u,v} | v \in V_2\}$ have value one.

In the first case: $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$, since all elements $\{x_{u,v} \mid v \in V_2\}$ have value zero. In the second case: since $v' \neq v$ and there is only one $x_{u,v}$

can have value one, $x_{u,v}$ and $x_{u,v'}$ cannot simultaneously have value one. That means every product in $\sum_{v\in V_2}\left(x_{u,v}\sum_{\substack{v'\in V_2\\ v'\neq v}} x_{u,v'}\right)$ has value zero. In the third case: since at least two elements of $\{x_{u,v} \mid v \in V_2\}$ have value one, there exists an $x_{u,v} = 1$ and an $x_{u,v'} = 1$ while $v \neq v'$. We then have at least one product in $\sum_{v\in V_2}\left(x_{u,v}\sum_{\substack{v'\in V_2\\ v'\neq v}} x_{u,v'}\right)$ has value one.

In order to have $H(\mathbf{x}) = 0$, for every $u \in V_1$, the variables' assignment should be in either the first case or the second one. If it is in the first case, then that u is mapped to no vertex in V_2. If it is in the second case, then that u is mapped to exactly one vertex in V_2.

Similarly, we have the second part of $H(\mathbf{x}) = 0$ if and only if no two elements of V_1 is mapped to the same element in V_2 by $D(\mathbf{x})$. That means $\mathbf{x} \in \mathrm{Dom}(D)$. Therefore, we have if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

We then show that if $D(\mathbf{x})$ is injective, then $H(\mathbf{x}) = 0$.

Let us assume $D(\mathbf{x})$ is injective. Then for each $u \in V_1$, the value of all elements of $\{x_{u,v}|v \in V_2\}$ will fall into the first two cases. Hence, the sum of all these products are zero. That means $H(\mathbf{x}) = 0$.

Therefore, $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective.

□

Lemma 19. *If $H(\mathbf{x}) = 0$ then $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving.*

Proof. Assume $H(\mathbf{x}) = 0$ and let $\psi = D(\mathbf{x})$, we know that ψ is an injective function by Lemma 18. For brevity, define

$$P_{u,u'}(\mathbf{x}) = \sum_{v\in V_2}\left(x_{u,v}\sum_{v'\in V_2} x_{u',v'}\tau(u,u',v,v')\right).$$

Then

$$P(\mathbf{x}) = \sum_{uu'\in E_1} P_{u,u'}(\mathbf{x}).$$

Now $\tau(u,u',v,v')$ is a pre-computed constant that is either 0 or 1. Therefore $P_{u,u'}(\mathbf{x})$ is a linear combination of non-negative terms and is thus non-negative. Then it follows that

$$P(\mathbf{x}) = 0 \text{ if and only if } P_{u,u'}(\mathbf{x}) = 0, \text{ for all } uu' \in E_1.$$

Let us assume that $P(\mathbf{x}) = 0$. We know that $P_{u,u'}(\mathbf{x}) = 0$ and we can expand

$P_{u,u'}(\mathbf{x})$ to obtain

$$\begin{aligned} P_{u,u'}(\mathbf{x}) = \sum_{v \in V_2} x_{u,v} \Big(& x_{u',v_0}\tau(u,u',v,v_0) + x_{u',v_1}\tau(u,u',v,v_1) + \cdots \\ & + x_{u,v_{n_2-1}}\tau(u,u',v,v_{n_2-1}) \Big) \\ = 0. \end{aligned}$$

Because ψ is injective we have two cases:

1. $x_{u,v} = 0$ for all $v \in V_2$, so uu' is not an edge in the common subgraph.
2. there exists a unique element $x^*_{u,v}$ with value one in $\{x_{u,v} \mid v \in V_2\}$ and similarly we have $x^*_{u',v'}$ in $\{x_{u',v'} \mid v' \in V_2\}$.

It then follows that we have

$$\begin{aligned} P_{u,u'}(\mathbf{x}) &= x^*_{u,v} \Big(x_{u',v_0}\tau(u,u',v,v_0) + x_{u',v_1}\tau(u,u',v,v_1) + \cdots \\ &\qquad + x_{u,v_{n_2-1}}\tau(u,u',v,v_{n_2-1}) \Big) \\ &= x^*_{u,v} \Big(x^*_{u',v'}\tau(u,u',v,v')) \Big) \\ &= \tau(u,u',v,v'). \end{aligned}$$

We know $P_{u,u'}(\mathbf{x}) = 0$, thus $\tau(u,u',v,v') = 0$. From the definition of $\tau(u,u',v,v') = 0$, we know a possible edge mapping exists between uu' and vv' for all $uu' \in E_1$. Similarly, a possible edge mapping exists between uu' and vv' for all $vv' \in E_2$. Thus $D(x)$ is edge-preserving.

Conversely, assume that $D(x)$ is edge-preserving and $H(\mathbf{x}) = 0$ while $P(\mathbf{x}) \neq 0$. Since $H(\mathbf{x}) = 0$, ψ is injective and we have the same two cases as above. There then exists a uu' that is an edge of the common subgraph, such that $P_{u,u'}(\mathbf{x}) \neq 0$. Because it is in the second case, we have

$$P_{u,u'}(\mathbf{x}) \Leftrightarrow x^*_{u,v}x^*_{u',v'}\tau(u,u',v,v') \neq 0 \Leftrightarrow \tau(u,u',v,v') \neq 0.$$

It follows that there exists no $vv' \in E_2$, such that a possible edge mapping exists between uu' and vv'. Hence ψ is not edge-preserving. A contradiction arises.

Therefore we have $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. □

Lemma 20. *If $H(\mathbf{x}) = 0$ then $N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is non-edge-preserving.*

Proof. An analogue of the proof of Lemma 19 proves this lemma. □

Corollary 21. *For all* $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ *if and only if* $D(\mathbf{x})$ *is an injective, edge-invariant function.*

Proof. All three components, $H(\mathbf{x}), P(\mathbf{x}), N(\mathbf{x})$ are non-negative. We also know from Lemma 18 and 19 that $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective and $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. Lemma 20 shows $N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is non-edge-preserving. Hence for all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-invariant function. □

Lemma 22. $\min(\mathrm{Im}(F)) \leq 0$.

Proof. If $\mathbf{x} = (0, \ldots, 0)$, then $F(\mathbf{x}) = 0$. Hence $\min(\mathrm{Im}(F)) \leq 0$. □

Lemma 23. *If* $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ *then* $D(\mathbf{x})$ *is an edge-invariant injection.*

Proof. Assume $F(\mathbf{x})$ is the minimum value of F. By Lemma 22, we have $F(\mathbf{x}) \leq 0$. Suppose that $D(\mathbf{x})$ is not an injective, edge-invariant function. That is, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] > 0$. For B sufficiently smaller than A we have $F(\mathbf{x}) > 0$. A contradiction arises. Therefore, we have $D(\mathbf{x})$ is an edge-invariant injection. □

Theorem 24. *If* $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ *then* $D(\mathbf{x})$ *is a solution to MCISI.*

Proof. By Lemma 23 we know that $D(\mathbf{x})$ is an injective, edge-invariant function. Suppose that $D(\mathbf{x})$ is not a solution to MCISI and instead we have an $\mathbf{x}'$ such that $D(\mathbf{x}')$ is. Since $D(\mathbf{x}')$ is a solution of MCISI, it must be an injective, edge-invariant function. Hence $A[H(\mathbf{x}') + P(\mathbf{x}') + N(\mathbf{x}')] = 0$. However, $M(\mathbf{x}') > M(\mathbf{x})$ as $D(\mathbf{x}')$ is a solution to MCISI and thus the corresponding common induced subgraph of $D(\mathbf{x})'$ has more vertices than the corresponding common induced subgraph of $D(\mathbf{x})$. So we obtain $F(\mathbf{x}') < F(\mathbf{x})$. However, this is not possible, since $F(\mathbf{x})$ is the minimal value of F. Hence $D(\mathbf{x})$ is a solution to MCISI. □

3.6 QUBO formulation for MCESI

Following a similar path, we will design penalty terms in the QUBO objective function according to the features of MCESI. Let us work with the two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$. We denote the MCES of G_1 and G_2 as $S = (V_S, E_S, \alpha_S, \beta_S)$ and the MCESI from G_1 to G_2 as γ. This means there exists a subgraph $S_1 = (V_{S_1}, E_{S_1}, \alpha_{S_1}, \beta_{S_1})$ of G_1 and a subgraph $S_2 = (S_{S_2}, E_{S_2}, \alpha_{S_2}, \beta_{S_2})$ of G_2, such that:

- $\alpha_{S_1}(v_1) = \alpha_{S_2}(\gamma(v_1))$ for all $v_1 \in V_{S_1}$, and $\alpha_{S_2}(v_2) = \alpha_{S_1}(\gamma^{-1}(v_2))$ for all $v_2 \in V_{S_2}$ (injective);

- for any edge $e_1 = (v_1, v_1') \in E_{S_1}$, there exists an edge $e_2 = (\gamma(v_1), \gamma(v_1')) \in E_{S_2}$, such that $\beta_{S_1}(e_1) = \beta_{S_2}(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_{S_2}$, there exists an edge $e_1 = (\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \in E_{S_1}$, such that $\beta_{S_1}(e_1) = \beta_{S_2}(e_2)$ (edge-preserve).

Then we use the same variable system and function τ to encode these features into the QUBO objective function. We define the objective function of MCESI as

$$F : \mathbb{Z}_2^{n_1 n_2} \to \mathbb{R},$$
$$F(\mathbf{x}) = A[H(\mathbf{x}) + P(\mathbf{x})] - BM(\mathbf{x}), A, B \in \mathbb{R}^+, \tag{3.9}$$

by

$$H(\mathbf{x}) = \sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) + \sum_{v \in V_2} \sum_{u \in V_1} \left(x_{u,v} \sum_{\substack{u' \in V_1 \\ u' \neq u}} x_{u',v} \right), \tag{3.10}$$

$$P(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \tau(u, u', v, v') \right), \tag{3.11}$$

$$M(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{vv' \in E_2} \left(x_{u,v} x_{u',v'} \right). \tag{3.12}$$

We request B to be sufficiently smaller than A.

The function $H(\mathbf{x})$ is designed to add penalty when the mapping is not injective. $P(\mathbf{x})$ adds penalty when the mapping is not edge-preserving. $M(\mathbf{x})$ adds penalty when the number of edges is not maximised.

We then prove the correctness of this objective function.

Corollary 25. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function.*

Proof. Both $H(\mathbf{x})$ and $P(\mathbf{x})$ are non-negative functions. We also know from Lemma 18 and 19 that $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective and $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. Hence it follows that $A[H(\mathbf{x}) + P(\mathbf{x})] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function. □

Lemma 26. *If $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ then $D(\mathbf{x})$ is an edge-preserving injection.*

Proof. Assume $F(\mathbf{x})$ is the minimum value of F. By Lemma 22, we have $F(\mathbf{x}) \leq 0$. Suppose that $D(\mathbf{x})$ is not an injective, edge-preserving function. That is, $A[H(\mathbf{x}) + P(\mathbf{x})] > 0$. For B sufficiently smaller than A we have $F(\mathbf{x}) > 0$. A contradiction arises. Therefore, we have $D(\mathbf{x})$ is an edge-preserving injection.

□

Theorem 27. *If* $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ *then* $D(\mathbf{x})$ *is a solution to MCESI.*

Proof. By Lemma 26 we know that $D(\mathbf{x})$ is an injective, edge-preserving function. Suppose that $D(\mathbf{x})$ is not a solution to MCESI and instead we have an $\mathbf{x}'$ such that $D(\mathbf{x}')$ is. Since $D(\mathbf{x}')$ is a solution of MCESI, it must be an injective, edge-preserving function. Hence $A[H(\mathbf{x}') + P(\mathbf{x}')] = 0$. However, $M(\mathbf{x}') > M(\mathbf{x})$ as $D(\mathbf{x}')$ is a solution to MCESI and thus the corresponding common edge subgraph of $D(\mathbf{x}')$ has more edges than the corresponding common edge subgraph of $D(\mathbf{x})$. So we obtain $F(\mathbf{x}') < F(\mathbf{x})$. However, this is not possible, since $F(\mathbf{x})$ is the minimal value of F. Hence $D(\mathbf{x})$ is a solution to MCESI. □

3.7 QUBO formulation for the *k*-densest common subgraph isomorphism

We first give the definition of the problem:
***k*-densest common subgraph**:

Instance: Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $n_1 = |V_1|$, $n_2 = |V_2|$, a positive integer $k \leq \min(n_1, n_2)$.
Question: Find a common subgraph $G' = (V', E')$ of G_1 and G_2 with order k, such that $\mathrm{DENS}(G'(V')) = \max\{\mathrm{DENS}(G''(V'')) \mid G'' = (V'', E'')$ is a common subgraph of G_1 and G_2 with order $k\}$.

Unlike the standard maximum common subgraph isomorphism problem, which maximise the number of vertices or edges of the common subgraph, in the k-densest common subgraph isomorphism problem we maximise the density of the common subgraph with a fixed order k. The solution usually implies a more complex substructure shared by the two input graphs compared to low density solutions. In Figure 3.3, we show an example of the 4-densest common subgraph isomorphism between two graphs. We use A and B to denote different vertex label. We use subscriptions to distinguish different atoms. In Figure 3.3, one of the possible solution for the 4-densest subgraph isomorphism problem is: A_2 is mapping to A_a, A_2 to A_b, B_1 to B_a, B_2 to B_b.

We will design penalty terms in the QUBO objective function according to the features of the above definition of the k-densest common subgraph. Let us work with two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$. We denote the k-densest common subgraph isomorphism from G_1 to G_2 as γ. That means there exists a subgraph $G_1' = (V_{G_1'}, E_{G_1'}, \alpha_{G_1'}, \beta_{G_1'})$ of G_1 and a subgraph $G_2' = (V_{G_2'}, E_{G_2'}, \alpha_{G_2'}, \beta_{G_2'})$ of G_2, such that:

- $\alpha_{G_1'}(v_1) = \alpha_{G_2'}(\gamma(v_1))$ for all $v_1 \in V_{G_1'}$, and $\alpha_{G_2'}(v_2) = \alpha_{G_1'}(\gamma^{-1}(v_2))$ for all $v_2 \in V_{G_2'}$ (injective);

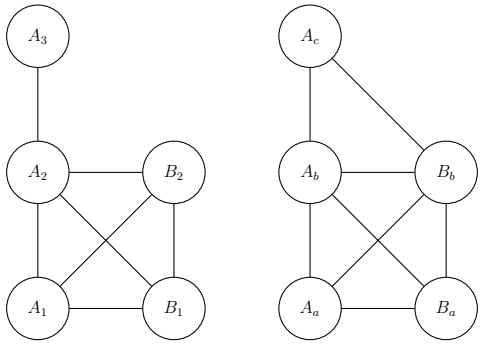


Figure 3.3: Example for two graphs that shares a 4-densest common subgraph

- for any edge $e_1 = (v_1, v_1') \in E_{G_1'}$, there exists an edge $e_2 = (\gamma(v_1), \gamma(v_1')) \in E_{G_2'}$, such that $\beta_{G_1'}(e_1) = \beta_{G_2'}(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_{G_2'}$, there exists an edge $e_1 = (\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \in E_{G_1'}$, such that $\beta_{G_1'}(e_1) = \beta_{G_2'}(e_2)$ (edge-preserving);
- $|V_{G_1'}| = |V_{G_2'}| = k$.

We build the QUBO objective function of the k-densest common subgraph by translating those three features into penalty terms, which are terms with positive coefficients. Any assignments that do not fit into these features will add a positive value into the solution of the objective function. Thus, these cases will not be picked as potential solutions of the problem since the D-Wave machines will pick up the assignments that result the minimum value of the objective function. Let $V_1 = \{u_0, u_1, \ldots, u_{n_1-1}\}$ and $V_2 = \{v_1, v_2, v_{n_2-1}\}$. For every $u \in V_1$, $v \in V_2$, a variable $x_{u,v}$ is used to represent the possible vertex mapping between u and v. Let $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ with

$$\mathbf{x} = (x_{u_0,v_0}, x_{u_0,v_1}, \ldots, x_{u_0,v_{n_2-1}}, x_{u_1,v_0}, \ldots, x_{u_{n_1-1},v_{n_2-2}}, x_{u_{n_1-1},v_{n_2-1}}).$$

We define a function $\tau : V_1 \times V_1 \times V_2 \times V_2 \to \mathbb{Z}$ to identify the possible mapping between an edge in G_1 to an edge in G_2 with respect to the vertex labels and edge labels. Let $e_1 = (u, u')$ for $u, u' \in V_1$ and $e_2 = (v, v')$ for $v, v' \in V_2$,

$$\tau(u, u', v, v') = \begin{cases} 0, & \alpha_1(u) = \alpha_2(v), \\ & \alpha_1(u') = \alpha_2(v') \\ & \text{and } \beta_1(e_1) = \beta_2(e_2), \\ 1, & \text{otherwise.} \end{cases} \tag{3.13}$$

We define the objective function by

$$F : \mathbb{Z}_2^{n_1 n_2} \to \mathbb{R},$$

$$F(\mathbf{x}) = A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] - BM(\mathbf{x}), \tag{3.14}$$

where

$$H(\mathbf{x}) = \sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) \tag{3.15}$$

$$+ \sum_{v \in V_2} \sum_{u \in V_1} \left(x_{u,v} \sum_{\substack{u' \in V_1 \\ u' \neq u}} x_{u',v} \right), \tag{3.16}$$

$$P(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \right. \tag{3.17}$$

$$\left. \tau(u, u', v, v') \right), \tag{3.18}$$

$$N(\mathbf{x}) = \left(\sum_{u \in V_1} \sum_{v \in V_2} x_{u,v} - k \right)^2, \tag{3.19}$$

$$M(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \right). \tag{3.20}$$

$A, B \in \mathbb{R}^+$. We request B to be sufficiently smaller than A, which is needed in later proofs. Here $e_{v,v'}$ can be calculated by $e_{v,v'} = 1$ if $vv' \in E_2$ and $e_{v,v'} = 0$ otherwise.

The function $H(\mathbf{x})$ is designed to ensure that the mapping is injective. $P(\mathbf{x})$ ensures that the mapping is edge-preserving. Penalties are only applied to impossible-edge mappings with respect to both the vertex label and the edge label. With the help of τ, some entries of the QUBO matrix may be updated to zero. $N(\mathbf{x})$ ensures that the common subgraph has order k. $M(\mathbf{x})$ ensures that the density of the common subgraph is maximised. All these claims will be showed in later proofs.

After the QUBO objective function is in place, now show that this objective function indeed solves the maximum common induced subgraph isomorphism problem. We define $\mathcal{F}$ to be the set of all common subgraph isomorphisms between G_1 and G_2. We specify a decoder function

$$D(\mathbf{x}) : \mathbb{Z}_2^{n_1 n_2} \to \mathcal{F}.$$

The purpose of which is to interpret the isomorphism encoded in $\mathbf{x}$ so that we can obtain a vertex mapping. The domain of D contains all vectors $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ that

can be 'decoded' into such functions. That is

$$\mathrm{Dom}(D) = \Bigg\{ \mathbf{x} \in \mathbb{Z}_2^{n_1 n_2} \;\Bigg|\; \sum_{v \in V_2} x_{u,v} \leq 1, \forall u \in V_1 \text{ and } \sum_{u \in V_1} x_{u,v} \leq 1, \forall v \in V_2 \Bigg\}$$

and

$$D(\mathbf{x}) = \begin{cases} \psi, & \text{if } \mathbf{x} \in \mathrm{Dom}(D), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Here $\psi : V_1 \to V_2$ is a partial function that for every $u \in V_1$, $\psi(u) = v$, $u \in V_1$ and $v \in V_2$, if and only if $x_{u,v} = 1$.

Lemma 28. *For all* $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $H(\mathbf{x}) = 0$ *if and only if* $D(\mathbf{x})$ *is injective.*

Proof. We first show that if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

Since $H(\mathbf{x})$ only contains sum of products of two binary variables, which are non-negative, $H(\mathbf{x}) = 0$ if and only if all products have value zero.

The first part of $H(\mathbf{x})$, $\sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$ if and only if all products have value zero. For every fixed u, the products can only be in the following three cases:

1. all elements of $\{x_{u,v} \mid v \in V_2\}$ have value zero;
2. exactly one element of $\{x_{u,v} \mid v \in V_2\}$ has value one;
3. more than one element of $\{x_{u,v} \mid v \in V_2\}$ have value one.

In the first case: $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$, since all elements $\{x_{u,v} \mid v \in V_2\}$ have value zero. In the second cases: since $v' \neq v$ and there is only one $x_{u,v}$ can have value one, $x_{u,v}$ and $x_{u,v'}$ cannot simultaneously have value one. That means every product in $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right)$ has value zero. In the third case: since at least two elements of $\{x_{u,v} \mid v \in V_2\}$ have value one, there exists an $x_{u,v} = 1$ and an $x_{u,v'} = 1$ while $v \neq v'$. We then have at least one product in $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right)$ that has value one.

In order to have $H(\mathbf{x}) = 0$, for every $u \in V_1$, the variables' assignment should be in either the first case or the second one. If it is in the first case, then that u

is mapped to no vertex in V_2. If it is in the second case, then that u is mapped to exactly one vertex in V_2.

Similarly, we have the second part of $H(\mathbf{x}) = 0$ if and only if no two elements of V_1 is mapped to the same element in V_2 by $D(\mathbf{x})$. That means $\mathbf{x} \in \mathrm{Dom}(D)$. Therefore, we have if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

We show that if $D(\mathbf{x})$ is injective, then $H(\mathbf{x}) = 0$.

Let us assume $D(\mathbf{x})$ is injective. Then for each $u \in V_1$, the value of all elements of $\{x_{u,v} \mid v \in V_2\}$ will fall into the first two cases. Hence, the sum of all these products are zero. That means $H(\mathbf{x}) = 0$.

Therefore, $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective.

□

Lemma 29. *If $H(\mathbf{x}) = 0$ then $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving.*

Proof. Assume $H(\mathbf{x}) = 0$ and let $\psi = D(\mathbf{x})$, we know that ψ is an injective function by Lemma 28. For brevity, define

$$P_{u,u'}(\mathbf{x}) = \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \tau(u, u', v, v') \right).$$

Then

$$P(\mathbf{x}) = \sum_{uu' \in E_1} P_{u,u'}(\mathbf{x}).$$

Now $\tau(u, u', v, v')$ is a pre-computed constant that is either 0 or 1. Therefore, $P_{u,u'}(\mathbf{x})$ is a linear combination of non-negative terms and is thus non-negative. Then it follows that

$$P(\mathbf{x}) = 0 \text{ if and only if } P_{u,u'}(\mathbf{x}) = 0, \forall uu' \in E_1.$$

Let us assume that $P(\mathbf{x}) = 0$. We know that $P_{u,u'}(\mathbf{x}) = 0$, and we can expand $P_{u,u'}(\mathbf{x})$ to obtain

$$\begin{aligned} P_{u,u'}(\mathbf{x}) = \sum_{v \in V_2} x_{u,v} \Big(& x_{u',v_0} \tau(u, u', v, v_0) \\ & + x_{u',v_1} \tau(u, u', v, v_1) + \cdots \\ & + x_{u,v_{n_2-1}} \tau(u, u', v, v_{n_2-1}) \Big) \\ =& 0. \end{aligned}$$

Because ψ is injective we have two cases:

1. $x_{u,v} = 0$ for all $v \in V_2$, so uu' is not an edge in the common subgraph.

2. There exists a unique element $x^*_{u,v}$ with value one in $\{x_{u,v} \mid v \in V_2\}$, and similarly we have $x^*_{u',v'}$ in $\{x_{u',v'} \mid v' \in V_2\}$.

It then follows that we have

$$\begin{aligned} P_{u,u'}(\mathbf{x}) &= x^*_{u,v}\Big(x_{u',v_0}\tau(u,u',v,v_0) \\ &\quad + x_{u',v_1}\tau(u,u',v,v_1) + \cdots \\ &\quad + x_{u,v_{n_2-1}}\tau(u,u',v,v_{n_2-1})\Big) \\ &= x^*_{u,v}\Big(x^*_{u',v'}\tau(u,u',v,v')\Big) \\ &= \tau(u,u',v,v'). \end{aligned}$$

We know $P_{u,u'}(\mathbf{x}) = 0$, thus $\tau(u,u',v,v') = 0$. Hence, a possible edge mapping exists between uu' and vv' for all $uu' \in E_1$. Similarly, a possible edge mapping exists between uu' and vv' for all $vv' \in E_2$. Thus, $D(x)$ is edge-preserving.

Conversely, assume that $D(x)$ is edge-preserving and $H(\mathbf{x}) = 0$ while $P(\mathbf{x}) \neq 0$. Since $H(\mathbf{x}) = 0$, ψ is injective, and we have the same two cases as above. There then exists a uu' that is an edge of the common subgraph, such that $P_{u,u'}(\mathbf{x}) \neq 0$. Because it is in the second case, we have

$$\begin{aligned} P_{u,u'}(\mathbf{x}) &\Leftrightarrow x^*_{u,v}x^*_{u',v'}\tau(u,u',v,v') \neq 0 \\ &\Leftrightarrow \tau(u,u',v,v') \neq 0. \end{aligned}$$

It follows that there exists no $vv' \in E_2$, such that a possible edge mapping exists between uu' and vv'. Hence, ψ is not edge-preserving. A contradiction arises.

Therefore, we have $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. □

Lemma 30. *$N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ maps exactly k vertices in G_1 to k vertices in G_2.*

Proof. If $N(\mathbf{x}) = 0$, then $\sum_{u\in V_1} \sum_{v\in V_2} x_{u,v} = k$. That means the common subgraph has order k.

If $D(\mathbf{x})$ maps exactly k vertices in G_1 to k vertices in G_2, then there are exactly k variables has value 1. Thus, $\sum_{u\in V_1} \sum_{v\in V_2} x_{u,v} = k$ and $N(\mathbf{x}) = 0$. □

Corollary 31. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function that maps exactly k vertices in G_1 to k vertices in G_2.*

Proof. All three components, $H(\mathbf{x}), P(\mathbf{x})$ and $N(\mathbf{x})$ are non-negative. We also know from Lemmas 28 and 29 that $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective and $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. Lemma 30 shows $N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ maps exactly k vertices in G_1 to k vertices in G_2. Hence, for all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function that maps exactly k vertices in G_1 to k vertices in G_2. □

Lemma 32. $\min(\mathrm{Im}(F)) \leq 0$.

Proof. If $\mathbf{x} = (\underbrace{0, \ldots, 0}_{n_1 n_2 \text{ times}})$, then $F(\mathbf{x}) = 0$. Hence, $\min(\mathrm{Im}(F)) \leq 0$. □

Lemma 33. *If* $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ *then* $D(\mathbf{x})$ *is an edge-preserving injection that maps exactly* k *vertices in* G_1 *to* k *vertices in* G_2.

Proof. Assume $F(\mathbf{x})$ is the minimum value of F. By Lemma 32, we have $F(\mathbf{x}) \leq 0$. Suppose that $D(\mathbf{x})$ is not an injective, edge-preserving function that maps exactly k vertices in G_1 to k vertices in G_2. That is, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] > 0$. For B sufficiently smaller than A we have $F(\mathbf{x}) > 0$. A contradiction arises. Therefore, we have $D(\mathbf{x})$ is an edge-invariant injection. □

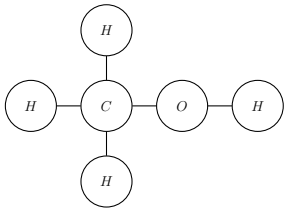
Theorem 34. *If* $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ *then* $D(\mathbf{x})$ *is a solution to the* k*-densest common subgraph isomorphism problem.*

Proof. By Lemma 33 we know that $D(\mathbf{x})$ is an injective, edge-preserving function that maps exactly k vertices in G_1 to k vertices in G_2. Suppose that $D(\mathbf{x})$ is not a solution to the k-densest common subgraph isomorphism problem, and instead we have an $\mathbf{x}'$ such that $D(\mathbf{x}')$ is. Since $D(\mathbf{x}')$ is a solution of the k-densest common subgraph isomorphism problem, it must be an injective, edge-preserving function that maps exactly k vertices in G_1 to k vertices in G_2. Hence, $A[H(\mathbf{x}') + P(\mathbf{x}') + N(\mathbf{x}')] = 0$. However, $M(\mathbf{x}') > M(\mathbf{x})$ as $D(\mathbf{x}')$ is a solution to the k-densest common subgraph isomorphism problem and thus the corresponding common subgraph of $D(\mathbf{x})'$ has more vertices than the corresponding common subgraph of $D(\mathbf{x})$. So we obtain $F(\mathbf{x}') < F(\mathbf{x})$. However, this is not possible, since $F(\mathbf{x})$ is the minimal value of F. Hence, $D(\mathbf{x})$ is a solution to the k-densest common subgraph isomorphism problem. □

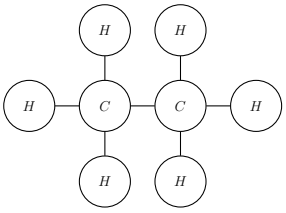
3.8 Experiments on a D-Wave 2X machine

We have used a D-Wave 2X machine to test the performance of our QUBO formulations. We first used a classical solver to find the optimal answers for the QUBO matrices and decode them into the solutions of corresponding maximum

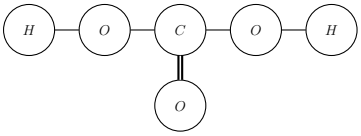
common subgraph isomorphism problems. Then we compared them with the solutions from a classical maximum common subgraph isomorphism solver. After these checks, we run these instances on a D-Wave 2X machine and collect data for further analysis. Since our QUBO formulation for both MCISI and MCESI needs to have a logical variable for each possible vertex mapping and the QUBO matrix is quite dense, we do not have enough resources (qubits) to run for large graphs. Hence, only small graphs that represent some small organic molecules are used to generate instances for the D-Wave 2X machine. We represent some simple molecules as graphs in Figure 3.4. All these molecules are very small; they contain no more than nine atoms. Different atoms are given different labels, such as C, O and H. Different covalent bonds are labelled as single lines and double lines.



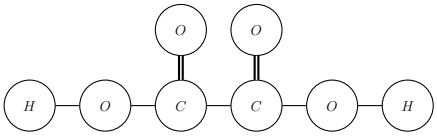
(a) Methanol



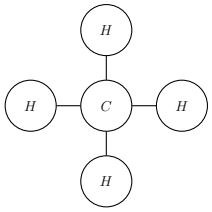
(b) Ethane



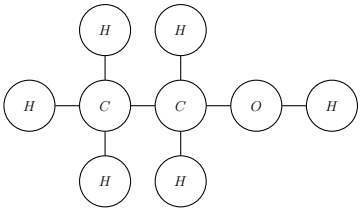
(c) Carbonic acid



(d) Oxalic acid



(e) Methane



(f) Ethanol

Figure 3.4: Input graphs

We have used the QUBO formulations in Section 3.2 to generate the corresponding QUBO matrices for MCISI and MCESI corresponding to the chosen graphs. Then the QUBO matrices were used by the default embedding program from D-Wave to generate the corresponding embedding. We repeated this process

for all possible pairs of graphs in the list in Figure 3.4. Finally, we have sent the batch of problems to the D-Wave machine for processing. The following table shows the performance of the D-Wave 2X runs.

In the experiments, we used the majority voting feature that the D-Wave API provides. When a broken chain appears, it will force the qubits in the same chain to have the same value as that of more than half the qubits. The tables show how effective the algorithm works on the D-Wave 2X machine. The first column shows which pair of molecule graphs was used as input. The second column shows the length of the max chain after we have performed minor embedding. The third column shows the average chain length of all logical variables. The fourth column is the count of correct solutions out of 5000 rounds with post-processing. The fifth column is the count of correct solutions out of 5000 rounds without post-processing. The correct solutions, here, refer to the solution obtained from a brute force classical solver that generates all subgraphs of the input graphs and then finds the maximum common subgraphs between them. We count the solutions that match the results from the classical solver we mentioned earlier. Figure 3.5 and Figure 3.6 show that when both the maximum chain length and average chain length are small enough, the performance is quite good: a majority of the graph pairs have a high count of correct answers. However, when chain length grows, the performance drops significantly. This is consistent with other experiments [91].

The D-Wave API provides a post-process function that classically optimises the quantum machine's output [64]. According to this document, a D-Wave machine will break the embedded graph into several low tree-width subgraphs, then it will run local search for those subgraphs and combine the results to update the global result. Since the embedded graph contains all constraints for the problem, while the subgraphs only contain part of the constraints, it is possible that the post-processing may give a wrong answer. We run the same examples with the post-processing on and off to see whether this explanation makes sense. Figure 3.5 and Figure 3.6 show that when the chain length is long but not too long, without post-optimisation, we may get a bit more correct answers. However, when the chain length is short, the post-processing improves the quality of the answers. These facts give some support to the explanation. More importantly, one has to be careful about when to turn the post-processing on. For the current problem, verifying answers is as difficult as finding them. Therefore, even when we know that some of the correct answers may be discarded by the post-processing, we may still want to apply the post-processing, as it is too time consuming to investigate when the post-processing makes mistakes. However, for the problems like integer factoring, where it is hard to find the answers, but very easy to verify them, the situation is different. For such cases it is advisable to turn the post-processing off to avoid discarding correct answers.

input pair of graphs	max chain length	average chain length	count of correct answers with post-process	count of correct answers without post-process
CarbonicAcid_CarbonicAcid	4	2.57	4855	3070
CarbonicAcid_Ethane	4	3.14	0	532
CarbonicAcid_Ethanol	6	4.47	0	1
CarbonicAcid_Methane	3	2.44	0	11
CarbonicAcid_Methanol	3	2.41	5000	4505
CarbonicAcid_OxalicAcid	6	4.61	0	0
Ethane_CarbonicAcid	5	3.64	1	12
Ethane_Ethane	16	12.05	0	0
Ethane_Ethanol	14	10.77	0	0
Ethane_Methane	8	6.5	402	255
Ethane_Methanol	9	7.03	0	0
Ethane_OxalicAcid	5	3.75	2820	533
Ethanol_CarbonicAcid	6	4.35	0	0
Ethanol_Ethane	18	12.17	0	0
Ethanol_Ethanol	18	13.41	0	0
Ethanol_Methane	9	7.19	53	86
Ethanol_Methanol	12	7.74	0	0
Ethanol_OxalicAcid	6	5.3	6	0
Methane_CarbonicAcid	3	2.44	4955	4662
Methane_Ethane	9	6.42	0	0
Methane_Ethanol	9	7.11	0	0
Methane_Methane	5	3.58	1371	545
Methane_Methanol	5	3.94	499	506
Methane_OxalicAcid	4	3.1	4466	3744
Methanol_CarbonicAcid	5	3.25	5000	3423
Methanol_Ethane	9	7.11	0	0
Methanol_Ethanol	10	7.51	0	0
Methanol_Methane	5	3.70	179	55
Methanol_Methanol	5	4.05	12	14
Methanol_OxalicAcid	5	4.0	4952	3763
OxalicAcid_CarbonicAcid	7	5.0	0	0
OxalicAcid_Ethane	5	3.81	7	21
OxalicAcid_Ethanol	6	5.05	1	3
OxalicAcid_Methane	3	2.7	0	28
OxalicAcid_Methanol	6	4.35	3609	1132
OxalicAcid_OxalicAcid	9	7.25	0	0

Figure 3.5: Maximum common induced subgraph isomorphism results.

input pair of graphs	max chain length	average chain length	count of correct answers with post-process	count of correct answers without post-process
CarbonicAcid_CarbonicAcid	6	3.28	308	217
CarbonicAcid_Ethane	3	2.71	5000	4998
CarbonicAcid_Ethanol	4	3.11	0	8
CarbonicAcid_Methane	2	1.66	5000	4978
CarbonicAcid_Methanol	3	2.25	24	1196
CarbonicAcid_OxalicAcid	7	4.22	0	0
Ethane_CarbonicAcid	4	3.64	4004	3513
Ethane_Ethane	19	11.67	0	0
Ethane_Ethanol	14	10.32	0	0
Ethane_Methane	10	7.53	0	0
Ethane_Methanol	8	6.61	0	0
Ethane_OxalicAcid	5	4.43	1821	1170
Ethanol_CarbonicAcid	5	4.23	58	1
Ethanol_Ethane	14	10.82	0	0
Ethanol_Ethanol	18	11.0	0	0
Ethanol_Methane	10	6.34	26	10
Ethanol_Methanol	10	7.59	0	0
Ethanol_OxalicAcid	7	4.55	0	0
Methane_CarbonicAcid	4	2.55	4995	4966
Methane_Ethane	10	8.03	0	0
Methane_Ethanol	9	7.42	0	0
Methane_Methane	6	4.70	167	20
Methane_Methanol	6	4.58	99	287
Methane_OxalicAcid	4	2.7	5000	4978
Methanol_CarbonicAcid	4	2.91	3559	147
Methanol_Ethane	10	6.92	0	0
Methanol_Ethanol	10	7.66	0	0
Methanol_Methane	6	4.64	34	253
Methanol_Methanol	5	4.27	22	2
Methanol_OxalicAcid	6	3.42	1012	53
OxalicAcid_CarbonicAcid	5	3.83	62	3
OxalicAcid_Ethane	10	3.0	5000	3259
OxalicAcid_Ethanol	8	3.15	0	0
OxalicAcid_Methane	2	1.6	5000	4963
OxalicAcid_Methanol	5	2.57	1384	550
OxalicAcid_OxalicAcid	9	5.70	0	0

Figure 3.6: Maximum common edge subgraph isomorphism results.

It would be interesting to compare the run times obtained with the quantum solution with those obtained with the state of art heuristic classical solvers. However, the comparison would not make much sense unless larger scale problems could be solved on the quantum annealing machine (the heuristic classical solvers mentioned in Section 3.4 are capable of solving much larger scale problems).

3.9 Conclusions and future work

We have proposed QUBO formulations for the problems MCISI and MCESI, proved their correctness in Section 3.2. Then we solved instances for some small organic molecules on a real D-Wave machine and analysed the experimental data, Section 3.8. Very small scale problems can be solved by the D-Wave 2X machine with high enough success probabilities; however, for larger scale problems, the accurate rate drops significantly. With 5,000+ qubits and a 15-way qubit connectivity for its Pegasus architecture, D-Wave Advantage can embed larger scale problems and may have a better performance than its predecessors. It will be interesting to run more experiments for these problems with the new machine.

In the current version of our algorithm, we have set no restrictions on the connectivity of the common subgraphs. The density of a graph can be defined as the ratio of the size of the graph (number of edges) and the size of the complete graph with the same order (number of vertices). Since we are investigating the common sub-structure shared by two graphs, a denser common subgraph may have a better chance to give the relevant information. Therefore, one possible future step is to add in some constraints that will force the answer to be denser. We could also investigate methods that require fewer logical variables and/or fewer interactions among the logical variables. The former may help to extract the information of interest and the latter may improve chances to solve larger problems on the same machine. Investigation of quantum-classic hybrid methods, which break down the problem classically and solve the smaller scale problems on the quantum machines, is one of our future directions. We can start with Qbsolve and Hybrid Solver System (HSS). If larger scale problems could be solved on future quantum annealing machines, then meaningful comparisons between the proposed quantum algorithms and the state of art heuristic classical ones would be possible.

We have proved that our QUBO formulation can solve the k-densest common subgraph isomorphism problem. However, the associated graph will be complete for our guest graph, because of $N(\mathbf{x})$ in Eq 3.15. After embedding onto D-Wave host graph we would expect very long chains. The chain length will greatly affect the rate to get correct answers due to the current hardware's unreliability for long chains. In [93], a sparser input graphs pair results a lower density QUBO matrix. Thus, after embedding, fewer number of variables are used. However, the density

of input graphs will not affect the density of the QUBO graphs in our proposed algorithm. Note that in [93], the rate of getting a correct solution is not good enough, therefore, we expect even worse results if we try inputs of a similar size on a D-Wave machine. Hence, this paper's results are purely theoretical.

Unlike the algorithm for MCISI and MCESI, we introduced restrictions to the density of the common subgraph. That means it is possible that the algorithm will give output graphs with low connectivity. This implies relatively simple structure. In this paper, we require the density to be maximised. Even though full connectivity among vertices of the output graph is not guaranteed, it is very unlikely to have isolated vertices or subset of the vertices which are connected with each other but not to the complement of this subset.

Due to the fact, we can re-run the problem with a different k, we can retrieve a series of common subgraphs with different values of k. Among these graphs, we can pick the most interesting one to perform further analysis. We can run the problem with all possible values of k of the QUBO objective function and then pick the common subgraph with maximum density. However, since when $k = 2$, the density of the densest subgraph is always 1 (unless there is no edges in one or both of the input graphs). Therefore, it is more reasonable to set up a lower bound for k when we search for the densest common subgraph.

The only place in the objective function that k affects is $N(\mathbf{x})$. The value of $\sum_{u \in V_1} \sum_{v \in V_2} x_{u,v}$ changes when k changes. This is the only negative term in the objective function. It is possible that after changing k, we have a non-complete QUBO graph. However, it is very likely that this QUBO graph only has a few edges missing compared to the complete graph of the same order. Therefore, if we use complete graph embedding all the time. We do not need to compute embedding at all. It is another example of hybrid quantum-classical computing as proposed in [8].

Combining the results in [93] and [51], we proposed a QUBO formulation for solving the k-densest common subgraph isomorphism problem. We can use this QUBO formulation to solve the densest common subgraph isomorphism problem with a lower bound for the order of the common subgraph by running all possible k and picking the output that has the largest density.

Chapter 4

Final Remarks

The results in this thesis fall into two categories:

1. Experimental testing of the quality of quantum randomness.
2. Quantum annealing fast solutions for two important computational NP-hard problems:
   (a) the maximum common subgraph isomorphism problem,
   (b) the k-densest common subgraph isomorphism problem.

The quality of quantum randomness assessed via the analysis of very long strings of outputs has been studied in detail. These new tests have been cited and used in a few articles, including

- Martínez, A. C., Solis, A., Díaz Hernández Rojas, R., U'Ren, A. B., Hirsch, J. G., & Pérez Castillo, I. (2018). Advanced statistical testing of quantum random number generators. *Entropy*, 20, no. 11 (2018): 886. https://www.mdpi.com/1099-4300/20/11/886.
- Martínez, A.C., Solís, A., Rojas, R.D.H., U'Ren, A.B., Hirsch, J.G. and Castillo, I.P., 2018. Testing randomness in quantum mechanics. arXiv preprint arXiv:1810.08718.

New results in this direction have been recently obtained in

- Agüero Trejo, J. M. and Calude, C. S., and Dinneen, M. J. and Fedorov, A. and Kulikov, A. A. and Navarathna, R. and Svozil, K. How Real Is Incomputability in Physics? Report CDMTCS-572, https://www.cs.auckland.ac.nz/research/groups/CDMTCS/researchreports/download.php?selected-id=872.

Both problems in the second part of the thesis have been reformulated in an equivalent mathematical forms as QUBO problems and the correctness of these formulations was proved. The QUBO formulation for solving the the k-densest common subgraph isomorphism problem can be used to solve the densest common subgraph isomorphism problem with a lower bound for the order of the common subgraph by running all possible k and picking the output that has the largest density. The quality of these solutions was tested experimentally on the D-Wave 2X machine. The results show the advantage of quantum annealong solutions over the classical ones.

These results, particularly the mathematical ones, will be more useful wwhen the technology of D-Wave will advance.

Bibliography

[1] A. A. Abbott. The Deutsch-Jozsa problem: De-quantisation and entanglement. *Natural Computing*, 11(1):3–11, 2011.

[2] A. A. Abbott. De-quantisation of the quantum Fourier transform. *Applied Mathematics and Computation*, 291(1):3–13, 2012.

[3] A. A. Abbott. *Value Indefiniteness, Randomness and Unpredictability in Quantum Foundations*. PhD thesis, University of Auckland; École Normale Supérieure de Paris, 2015.

[4] A. A. Abbott, L. Bienvenu, and G. Senno. Non-uniformity in the Quantis random number generator. *CDMTCS Research Report Series 472*, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, 2014.

[5] A. A. Abbott and C. S. Calude. Von Neumann normalisation of a quantum random number generator. *Computability*, 1:59–83, 2012.

[6] A. A. Abbott, C. S. Calude, J. Conder, and K. Svozil. Strong Kochen-Specker theorem and incomputability of quantum randomness. *Physical Review A*, 86(062109), Dec 2012.

[7] A. A. Abbott, C. S. Calude, J. Conder, and K. Svozil. Strong Kochen-Specker theorem and incomputability of quantum randomness. *Physical Review A*, 86:062109, 2012.

[8] A. A. Abbott, C. S. Calude, M. J. Dinneen, and R. Hua. A hybrid quantum-classical paradigm to mitigate embedding costs in quantum annealing. *International Journal of Quantum Information*, 17(05):1950042, 2019.

[9] A. A. Abbott, C. S. Calude, M. J. Dinneen, and N. Huang. Experimental probing of the incomputability of quantum randomness. *CDMTCS Research Report Series 515v2*, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, 2018. https://www.cs.auckland.ac.nz/research/groups/CDMTCS/export/80_random_seqs/.

[10] A. A. Abbott, C. S. Calude, M. J. Dinneen, and N. Huang. Experimentally probing the algorithmic randomness and incomputability of quantum randomness. *Physica Scripta*, 94(4):045103, Feb 2019.

[11] A. A. Abbott, C. S. Calude, and K. Svozil. Value-indefinite observables are almost everywhere. *Physical Review A*, 89(032109), 2013.

[12] A. A. Abbott, C. S. Calude, and K. Svozil. A quantum random number generator certified by value indefiniteness. *Mathematical Structures in Computer Science*, 24:e240303, 6 2014.

[13] A. A. Abbott, C. S. Calude, and K. Svozil. A quantum random number generator certified by value indefiniteness. *Mathematical Structures in Computer Science*, 24(3):e240303, 2014.

[14] A. A. Abbott, C. S. Calude, and K. Svozil. A non-probabilistic model of relativised predictability in physics. *Information*, 6(4):773–789, 2015.

[15] A. A. Abbott, C. S. Calude, and K. Svozil. A non-probabilistic model of relativised predictability in physics. *Information*, 6(4):773–789, 2015.

[16] A. A. Abbott, C. S. Calude, and K. Svozil. On the unpredictability of individual quantum measurement outcomes. In L. D. Beklemishev, A. Blass, N. Dershowitz, B. Finkbeiner, and W. Schulte, editors, *Fields of Logic and Computation II – Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 69–86. Springer International, Switzerland, 2015.

[17] A. A. Abbott, C. S. Calude, and K. Svozil. A variant of the Kochen-Specker theorem localising value indefiniteness. *Journal of Mathematical Physics*, 56:102201, 2015.

[18] A. Acín. True quantum randomness. In A. Suarez and P. Adams, editors, *Is Science Compatible with Free Will?: Exploring Free Will and Consciousness in the Light of Quantum Physics and Neuroscience*, chapter 2, pages 7–22. Springer, 2013.

[19] J. M. Agüero Trejo and C. S. Calude. A new quantum random number generator certified by value indefiniteness. *Theoretical Computer Science*, 862:3–13, Mar. 2021.

[20] D. Aharonov, W. v. Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. arXiv:quant-ph/0405098, March 2005.

[21] E. H. Allen and C. S. Calude. Quassical computing. *International Journal of Unconventional Computing*, 14:43–57, 2018.

[22] N. Allen. Email to C. S. Calude. 19 November 2017.

[23] A. Aspect, P. Grangier, and G. Roger. Experimental realization of Einstein-Podolsky-Rosen-Bohm *Gedankenexperiment*: A new violation of Bell's inequalities. *Physical Review Letters*, 49(2):91–94, 1982.

[24] D. H. Bailey, J. M. Borwein, C. S. Calude, M. J. Dinneen, M. Dumitrescu, and A. Yee. An empirical approach to the normality of π. *Experimental Mathematics*, 21(4):375–384, 2012.

[25] F. Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[26] M. T. Barakat and P. M. Dean. Molecular structure matching by simulated annealing. iii. the incorporation of null correspondences into the matching problem. *Journal of computer-aided molecular design*, 5(2):107–117, 1991.

[27] H. G. Barrow and B. RM. Subgraph isomorphism, matching relational structures and maximal cliques. 1976.

[28] N. J. Beaudry and R. Renner. An intuitive proof of the data processing inequality. *Quantum Info. Comput.*, 12(5-6):432–441, May 2012.

[29] J. S. Bell. On the Eistein-Podolsky-Rosen paradox. *Physics*, 1(3):195–200, 1964.

[30] P. Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22:563–591, 1980.

[31] M. N. Bera, A. Acín, M. Kuś, M. Mitchell, and M. Lewenstein. Randomness in quantum mechanics: Philosophy, physics and technology. *Reports on Progress in Physics*, 80:124001, 2017.

[32] D. J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, T. Lange, and N. van Someren. Factoring RSA keys from certified smart cards: Coppersmith in the wild. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, pages 341–360, Berling, 2013. Springer.

[33] D. J. Bernstein, N. Heninger, P. Lou, and L. Valenta. Post-quantum RSA. cr.yp.to/papers/pqrsa-20170419.pdf, 2017.

[34] A. Berthiaume and G. Brassard. Oracle quantum computing. *Journal of Modern Optics*, 41:195–199, 1992.

[35] D. Bertsimas, C.-P. Teo, and R. Vohra. On dependent randomized rounding algorithms. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 330–344. Springer, 1996.

[36] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy. Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, 2:56, 2014.

[37] Z. Bian, F. Chudak, W. Macready, A. Roy, R. Sebastiani, and S. Varotti. Solving sat (and maxsat) with a quantum annealer: Foundations, encodings, and preliminary results. *Information and Computation*, page 104609, 2020.

[38] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven. Characterizing quantum supremacy in near-term devices. arXiv:1608.00263 [quant-ph], April 2017.

[39] V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(7):S13, 2013.

[40] É. Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo*, 27:247–271, 1909.

[41] R. D. Brown, G. Jones, P. Willett, and R. C. Glen. Matching two-dimensional chemical graphs using genetic algorithms. *Journal of Chemical Information and Computer Sciences*, 34(1):63–70, 1994.

[42] D. E. Browne. Efficient classical simulation of the quantum Fourier transform. *New Journal of Physics*, 9(5):146, May 2007.

[43] J. Cai, W. G. Macready, and A. Roy. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741*, 2014.

[44] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*. Springer-Verlag, Berlin, second edition, 2002.

[45] C. S. Calude. De-quantizing the solution of Deutsch's problem. *International Journal of Quantum Information*, 5(3):409–415, 2007.

[46] C. S. Calude. Quantum randomness: From practice to theory and back. In S. B. Cooper and M. Soskova, editors, *The Incomputable: Journeys Beyond the Turing Barrier*, pages 169–181. Springer, 2017.

[47] C. S. Calude and E. Calude. The road to quantum computational supremacy. In D. H. Bailey, N. S. Borwein, R. P. Brent, R. S. Burachik, J. H. Osborn, B. Sims, and Q. J. Zhu, editors, *From Analysis to Vizualization*, pages 249–367. Springer, 2020.

[48] C. S. Calude, E. Calude, and M. J. Dinneen. Adiabatic quantum computing challenges. *ACM SIGACT News*, 46(1):40–61, March 2015.

[49] C. S. Calude, M. J. Dinneen, M. Dumitrescu, and K. Svozil. Experimental evidence of quantum randomness incomputability. *Physical Review A*, 82:022102, 2010.

[50] C. S. Calude, M. J. Dinneen, and R. Hua. Qubo formulations for the graph isomorphism problem and related problems. *Theoretical Computer Science*, 701:54–69, 2017.

[51] C. S. Calude, M. J. Dinneen, and R. Hua. Quantum solutions for densest k-subgraph problems. *Journal of Membrane Computing*, 2(1):26–41, 2020.

[52] C. S. Calude and G. Păun. *Computing with Cells and Atoms*. Taylor & Francis Group, London and New York, 2001.

[53] Y. Cao, T. Jiang, and T. Girke. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics*, 24(13):i366–i374, 2008.

[54] G. J. Chaitin. Algorithmic information theory. *IBM Journal of Research and Development*, 21(4):350–359, 1977.

[55] G. J. Chaitin and J. T. Schwartz. A note on Monte Carlo primality tests and algorithmic information theory. *Communications on Pure and Applied Mathematics*, 31(4):521–527, 1978.

[56] D. G. Champernowne. The construction of decimals normal in the scale of ten. *Journal of the London Mathematical Society*, 8:254–260, 1933.

[57] G. Chapuis, H. Djidjev, G. Hahn, and G. Rizk. Finding maximum cliques on the d-wave quantum annealer. *Journal of Signal Processing Systems*, 91(3-4):363–377, 2019.

[58] R. Colbeck and A. Kent. Private randomness expansion with untrusted devices. *Journal of Physics A: Mathematical and General*, 44:095305, 2011.

[59] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, 1999.

[60] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[61] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[62] J. Coughlan. A tutorial introduction to belief propagation. *The Smith-Kettlewell Eye Research Institute*, 2009.

[63] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[64] D-Wave. Postprocessing methods.

[65] D-Wave. Problem-solving handbook. https://docs.dwavesys.com/docs/latest/doc_handbook.html.

[66] D-Wave. Developer's guide. *D-Wave system Inc.*, 2016.

[67] D-Wave. Developer guide for python. Technical report, 2018.

[68] D-Wave Systems, www.dwavesys.com. *D-Wave Problem-Solving Handbook. User Manual*, March 2018.

[69] G. Damiand, C. Solnon, C. De La Higuera, J.-C. Janodet, and É. Samuel. Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Computer Vision and Image Understanding*, 115(7):996–1010, 2011.

[70] C. M. Dawson and M. A. Nielsen. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030*, 2005.

[71] D.-L. Deng, C. Zu, X.-Y. Chang, P.-Y. Hou, H.-X. Yang, Y.-X. Wang, and L.-M. Duan. Exploring quantum contextuality to generate true random numbers. 2013.

[72] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[73] D. E. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[74] M. J. Dinneen, M. R. Hooshmandasl, and R. Hua. Formulating mixed dominating set problems for adiabatic quantum computers. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 2017.

[75] R. Downey and D. Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer, Berlin, 2010.

[76] E. Duesbury, J. Holliday, and P. Willett. Comparison of maximum common subgraph isomorphism algorithms for the alignment of 2d chemical structures. *ChemMedChem*, 13(6):588–598, 2018.

[77] A. Eagle. Randomness is unpredictability. *British Journal for the Philosophy of Science*, 56(4):749–790, 2005.

[78] A. Eagle. Chance versus randomness. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, Spring 2014 edition, 2014.

[79] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106, January 2000.

[80] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7), 1982.

[81] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[82] N. Funabiki and J. Kitamichi. A two-stage discrete optimization method for largest common subgraph problems. *IEICE TRANSACTIONS on Information and Systems*, 82(8):1145–1153, 1999.

[83] M. R. Garey and D. S. Johnson. Computers and intractability: a guide to the theory of np-completeness. 1980.

[84] J. E. Gentle. *Random Number Generations and Monte Carlo Methods*. Springer-Verlag, New York, 2 edition, 2003.

[85] O. Goldreich. *Foundations of cryptography I: Basic Tools*. Cambridge University Press, Cambridge, 2001.

[86] R. Graham and J. H. Spencer. Ramsey theory. *Scientific American*, 262:112–117, Sept. 1990.

[87] R. Griffiths and C. Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76(17):3228–3231, January 1996.

[88] A. Hájek. Interpretations of probability. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, Winter 2012 edition, 2014.

[89] E. Hemaspaandra, L. A. Hemaspaandra, and M. Zimand. Almost-everywhere superiority for quantum polynomial time. *Information and Computation*, 175(2):171 – 181, 2002.

[90] T. V. Himbeeck, E. Woodhead, N. J. Cerf, R. García-Patón, and S. Pironio. Semi-device-independent framework based on natural physical assumptions. *Quantum*, 1:33, 2017.

[91] R. Hua and M. J. Dinneen. Improved qubo formulation of the graph isomorphism problem. *SN Computer Science*, 1(1):19, Sep 2019.

[92] N. Huang. A qubo formulation for the k-densest common subgraph isomorphism problem via quantum annealing. In *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pages 1–7, 2020.

[93] N. Huang and D. Roje. A new qubo objective function for solving the maximum common subgraph isomorphism problem via quantum annealing. *SN Computer Science*, 2(3):186, 2021.

[94] ID Quantique. Quantis QRNG. https://www.idquantique.com/random-number-generation/.

[95] T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71:1675–1680, 2000.

[96] N. Johansson and J.-Å. Larsson. Efficient classical simulation of the Deutsch–Jozsa and Simon's algorithms. *Quantum Information Processing*, 16(9):233, Aug 2017.

[97] N. Johansson and J.-Å. Larsson. Quantum simulation logic, oracles, and the quantum advantage. *Entropy*, 21(8):1–76, 2019.

[98] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–8, May 12 2011.

[99] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.

[100] S. B. Kochen and E. Specker. The problem of hidden variables in quantum mechanics. *Journal of Mathematics and Mechanics (now Indiana University Mathematics Journal)*, 17(1):59–87, 1967.

[101] G. A. Kochenberger, F. Glover, B. Alidaee, and C. Rego. A unified modeling and solution framework for combinatorial optimization problems. *OR Spectrum*, 26(2):237–250, 2004.

[102] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004.

[103] M. G. Kovalsky, A. A. Hnilo, and M. B. Agüero. Kolmogorov complexity of sequences of random numbers generated in Bell's experiments. 2018.

[104] A. Kulikov, M. Jerger, A. Potočnik, A. Wallraff, and A. Fedorov. Realization of a quantum random generator certified with the Kochen-Specker theorem. *Phys. Rev. Lett.*, 119:240501, Dec 2017.

[105] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, Whit is right. Santa Barbara: IACR: 17, eprint.iacr.org/2012/064.pdf, 2012.

[106] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4):341, 1973.

[107] G. Longo and T. Paul. The mathematics of computing between logic and physics. In S. B. Cooper and A. Sorbi, editors, *Computability in Context: Computation and Logic in the Real World*, chapter 7, pages 243–274. Imperial College Press/World Scientific, London, 2008.

[108] X. Ma, X. Yuan, Z. Cao, B. Qi, and Z. Zhang. Quantum random number generation. *npj Quantum Information*, 2:16021, 2016.

[109] Y. I. Manin. Vychislimoe i nevychislimoe [Computable and Noncomputable] (in Russian). Sov. Radio. pp. 13–15. 1980.

[110] G. Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 8(14), 2003.

[111] G. Marsaglia. On the randomness of Pi and other decimal expansions. *Interstat*, 10(5):1–17, 2005.

[112] G. Marsaglia and A. Zaman. Towards a universal random number generator. *Statistics & Probability Letters*, 9(1):35–39, 1990. http://www.stat.fsu.edu/pub/diehard/.

[113] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9(6):602–619, 1966.

[114] A. C. Martínez, A. Solís, R. D. H. Rojas, A. B. U'Ren, J. G. Hirsch, and I. P. Castillo. Testing randomness in quantum mechanics. 2018.

[115] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[116] C. McGeoch. *Adiabatic Quantum Computation and Quantum Annealing. Theory and Practice*. Morgan & Claypool Publishers, 2014.

[117] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk. Practical annealing-based quantum computing. *Computer, www.computer.org/computer*, pages 38–46, 2019.

[118] D. N. Mermin. *Quantum Computer Science*. Cambridge University Press, Cambridge, 2007.

[119] C. A. Miller and Y. Shi. Universal security for randomness expansion from the spot-checking protocol. *SIAM Journal on Computing*, 46(4):1304, 2017.

[120] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu. Solving a higgs optimization problem with quantum annealing for machine learning. *Nature*, 550(7676):375–379, 2017.

[121] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, R. Barends, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, B. Foxen, R. Graff, E. Jeffrey, J. Kelly, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis. A blueprint for demonstrating quantum supremacy with superconducting qubits. arXiv:1709.06678 [quant-ph].

[122] M. J. Nene and G. Upadhyay. Shor's algorithm for quantum factoring. In R. K. Choudhary, J. K. Mandal, N. Auluck, and H. A. Nagarajaram, editors, *Advanced Computing and Communication Technologies: Proceedings of the 9th ICACCT, 2015*, pages 325–331, Singapore, 2016. Springer Singapore.

[123] M. E. O'Neill. Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation. Technical Report HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, Sep 2014.

[124] D. O'Malley, V. V. Vesselinov, B. S. Alexandrov, and L. B. Alexandrov. Nonnegative/binary matrix factorization with a d-wave quantum annealer. *PloS one*, 13(12):e0206653, 2018.

[125] Y. Peres. Iterating von Neumann's procedure for extracting random bits. *The Annals of Statistics*, 20(1):590–597, 1992.

[126] R. G. Pinch. The Carmichael numbers up to 10^{21}. In A.-M. Ernvall-Hytönen, M. Jutila, J. Karhumäki, and A. Lepistö, editors, *Proceedings of Conference on Algorithmic Number Theory 2007. TUCS General Publication No 46*, pages 129–131, Turku, Finland, 2007. Turku Centre for Computer Science.

[127] S. Pironio, A. Acín, S. Massar, A. B. de la Giroday, D. N. Matsukevich, P. Maunz, S. Olmchenk, D. Hayes, L. Luo, T. A. Manning, and C. Monroe. Random numbers certified by Bell's Theorem. *Nature*, 464:09008, 2010.

[128] J. Preskill. Quantum computing and the entanglement frontier. In D. Gross, M. Henneaux, and A. Sevrin, editors, *The Theory of the Quantum World*, pages 63–80, Singapore, November 10 2012. World Scientific Publishing. arXiv:1203.5813 [quant-ph].

[129] N. Robertson and P. D. Seymour. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

[130] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Special Publication 800-22, NIST, 2010.

[131] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw. Parallel random numbers: As easy as 1, 2, 3. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC11)*, New York, 2011. ACM.

[132] H. Schmidt. Quantum-mechanical random-number generator. *Journal of Applied Physics*, 41(2):462–468, 1970.

[133] N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar ising models. *Advances in Neural Information Processing Systems*, 21, 2008.

[134] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 2005.

[135] Y. Shen, L. Tian, and H. Zou. Practical quantum random nubmer generator based on measuring the shot noise of vacuum states. *Physical Review A*, 81(063814), 2010.

[136] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium of on Foundations of Computer Science, Santa Fe, NM, Nov. 20-22, 1994*. IEEE Computer Society Press, November 1994.

[137] D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

[138] A. Solis and J. G. Hirsch. Quantumness, randomness and computability. *Journal of Physics: Conference Series*, 624:012001, 2015.

[139] A. Solis, A. M. A. Martínez, R. R. Alarcón, H. C. Ramírez, A. B. U'Ren, and J. G. Hirsch. How random are random numbers generated using photons? *Physica Scripta*, 90:074034, 2015.

[140] R. M. Solovay. On random r.e. sets. In A. I. Arruda, N. C. A. da Costa, and R. Chuaqui, editors, *Non-Classical Logics, Model Theory, and Computability*, pages 283–307. North-Holland, Amsterdam, 1977.

[141] A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, and H. Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598, 2000.

[142] M. Stipčević and B. M. Rogina. Quantum random number generator based on photonic emission in semiconductors. *Review of Scientific Instruments*, 78(4):045104, 2007.

[143] K. Svozil. The quantum coin toss – testing microphysical undecidability. *Physics Letters A*, 143(9):433–437, 1990.

[144] O. Titiloye and A. Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, 2011.

[145] M. Um, X. Zhang, J. Zhang, Y. Wang, S. Yangchao, D.-L. Deng, L.-M. Duan, and K. Kim. Experimental certification of random numbers via quantum contextuality. *Scientific Reports*, 3:1627, 2013.

[146] H. Ushijima-Mwesigwa, C. F. Negre, and S. M. Mniszewski. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pages 22–29. ACM, 2017.

[147] C. Valenti. A genetic approach to the maximum common subgraph problem. In *Proceedings of the 20th International Conference on Computer Systems and Technologies*, pages 98–104, 2019.

[148] R. J. Van Berlo, W. Winterbach, M. J. De Groot, A. Bender, P. J. Verheijen, M. J. Reinders, and D. De Ridder. Efficient calculation of compound similarity based on maximum common subgraphs and its application to prediction of gene transcript levels. *International journal of bioinformatics research and applications*, 9(4):407–432, 2013.

[149] J. von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards Applied Math Series*, **12** (1951), 36–38. In A. H. Traub, editor, *John von Neumann, Collected Works*, pages 768–770. MacMillan, New York, 1963.

[150] M. Wagener and J. Gasteiger. The determination of maximum common substructures by a genetic algorithm: Application in synthesis design and for the structural analysis of biological activity. *Angewandte Chemie International Edition in English*, 33(11):1189–1192, 1994.

[151] S. Wagon. Is π normal? In J. L. Berggren, J. M. Borwein, and P. B. Borwein, editors, *Pi: A Source Book*, pages 557–559. Springer-Verlag, New York, 2004.

[152] B. L. Welch. The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 34(1-2), 1947.

[153] Z. Yang and M. J. Dinneen. Graph minor embeddings for d-wave computer architecture. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 2016.

[154] N. Yoshimura, M. Tawada, S. Tanaka, J. Arai, S. Yagi, H. Uchiyama, and N. Togawa. Efficient ising model mapping for induced subgraph isomorphism problems using ising machines. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 227–232. IEEE, 2019.

[155] K. M. Zick, O. Shehab, and M. French. Experimental quantum annealing: case study involving the graph isomorphism problem. *Scientific reports*, 5:11168, 2015.

Appendix A

Solving QUBOs on a D-Wave Machine

We use the following Python code to run the Maximum Common Subgraph Problem QUBOs on D-Wave.

```
#!/usr/bin/env python
# QUBO (with embedding) -> Ising -> DWave
# scale Ising in range [-.8 to 1]


import sys, time, math, traceback

from dwave_sapi2.remote import RemoteConnection
from dwave_sapi2.util import get_hardware_adjacency
from dwave_sapi2.embedding import embed_problem,
   unembed_answer
from dwave_sapi2.util import qubo_to_ising,
   ising_to_qubo
from dwave_sapi2.core import solve_ising

from sys import exc_info

# read input

line=sys.stdin.readline().strip().split()
print('header:',line)
n=int(line[0])
```

```
#Q = defaultdict(int)
Q = {}
for i in range(n):
    line=sys.stdin.readline().strip().split()
    for j in range(n):
        t = float(line[j])
        if t==0: continue
        if i <= j: Q[(i,j)]=Q.setdefault((i,j),0)+t
        else:      Q[(j,i)]=Q.setdefault((j,i),0)+t

print('Q=',Q)

(H,J,ising_offset) = qubo_to_ising(Q)

embedding=eval(sys.stdin.readline())
print 'embedding=', embedding
qubits = sum(len(embed) for embed in embedding)
print 'Physical qubits used= %s' % qubits

# create a remote connection using url and token and
   connect to solver
#
url = "https://affi.qcc.isi.edu/sapi"
token = "secrete_token_key"
solver_name = "DW2X"

print('Attempting to connect to network...')
try:
    remote_connection = RemoteConnection(url, token)
    solver = remote_connection.get_solver(solver_name)
except:
    print('Error: %s %s %s' % sys.exc_info()[0:3])
    traceback.print_exc()

#print('Solver properties:\n%s\n' % solver.properties)
A = get_hardware_adjacency(solver)

# Embed problem into hardware
(h0, j0, jc, new_emb) = embed_problem(H, J, embedding, A
   )
```

```
assert new_emb==embedding

# compute scale s so in range [-.8,1]

maxH=0.0
if len(h0): maxH=max(abs(min(h0)),abs(max(h0)))
maxJ=max(abs(min(j0.values())),abs(max(j0.values())))
maxV=max(maxH/2.0,maxJ)
s = 0.8/maxV

h1= [val*s for val in h0]
j1 = {}
for (key, val) in j0.iteritems():
    j1[key]=val*s

assert max(h1) <= 2.01
assert min(h1) >= -2.01
assert max(j1.values()) <= 1.01
assert min(j1.values()) >= -0.81

j1.update(jc)

# call the solver

print 'batch 1: with post-process on and programming
   time 1000'
for spins in [2]: # [[2,4,8,16]:
  #break
  annealT,progT,readT=20,1000,100 # had progT=100 on
   first runs
  print 'annealT=',annealT,'progT=',progT,'readT=',readT
   ,'spins=',spins,'post=optimize'
  result = solve_ising(solver, h1, j1, num_reads=5000,
   annealing_time=annealT,\
      programming_thermalization=progT,
   readout_thermalization=readT, postprocess='
   optimization',\
      num_spin_reversal_transforms=spins,auto_scale=False
   )
```

```
  print 'result:', result

  #newresult = unembed_answer(result['solutions'],
   new_emb, broken_chains='discard', h=H, j=J)
  newresult = unembed_answer(result['solutions'],
   new_emb, broken_chains='vote', h=H, j=J)
  print 'newresult:', newresult

print 'batch 2: with post-process off and programming
   time 100'
for spins in [2]: # [[2,4,8,16]:
  #break
  annealT,progT,readT=20,100,100 # had progT=100 on
   first runs
  print 'annealT=',annealT,'progT=',progT,'readT=',readT
   ,'spins=',spins,'post=None'
  result = solve_ising(solver, h1, j1, num_reads=5000,
   annealing_time=annealT,\
     programming_thermalization=progT,
   readout_thermalization=readT, postprocess=None,\
     num_spin_reversal_transforms=spins,auto_scale=False
   )
  print 'result:', result

  #newresult = unembed_answer(result['solutions'],
   new_emb, broken_chains='discard', h=H, j=J)
  newresult = unembed_answer(result['solutions'],
   new_emb, broken_chains='vote', h=H, j=J)
  print 'newresult:', newresult

print 'batch 3: with post-process off and programming
   time 1000'
for spins in [2]: # [[2,4,8,16]:
  #break
  annealT,progT,readT=20,1000,100 # had progT=100 on
   first runs
  print 'annealT=',annealT,'progT=',progT,'readT=',readT
   ,'spins=',spins,'post=None'
  result = solve_ising(solver, h1, j1, num_reads=5000,
   annealing_time=annealT,\
```

```
    programming_thermalization=progT,
  readout_thermalization=readT, postprocess=None,\
    num_spin_reversal_transforms=spins,auto_scale=False
  )
 print 'result:', result

 #newresult = unembed_answer(result['solutions'],
  new_emb, broken_chains='discard', h=H, j=J)
 newresult = unembed_answer(result['solutions'],
  new_emb, broken_chains='vote', h=H, j=J)
 print 'newresult:', newresult
```

preembed_Nan_Subgraph.py

We use the following BASH script to enumerate through several test cases.

```
#!/bin/bash

for n in Subgraph_Isomorphism/MCISI/QUBO/*.H; do
    echo; echo 'testing', $n
    time ./preembed_Nan_Subgraph.py < $n  | tee ${n%H}d.
  out
    sleep 1
done

for n in Subgraph_Isomorphism/MCESI/QUBO/*.H; do
    echo; echo 'testing', $n
    time ./preembed_Nan_Subgraph.py < $n  | tee ${n%H}d.
  out
    sleep 1
done
```

run_Nan_QUBOs_on_Dwave.sh