IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Generation of Truly Random Numbers on a Quantum Annealer

HARSHIL BHATIA[1,2], EDITH TRETSCHK[2], CHRISTIAN THEOBALT[2], and VLADISLAV GOLYANIK[2]

[1]Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur. (email: bhatia.2@iitj.ac.in)
[2]Visual Computing and Artificial Intelligence, Max Planck Institute for Informatics, Saarbrücken, Germany. (email: hbhatia, tretschk, theobalt, golyanik @mpi-inf.mpg.de)

Corresponding author: Vladislav Golyanik (e-mail: golyanik@mpi-inf.mpg.de.)

Partial financial support was received from the ERC Grant 4DReply (770784).

ABSTRACT This study investigates how qubits of modern quantum annealers (QA) such as D-Wave can be applied for generating truly random numbers. We show how a QA can be initialised and how the annealing schedule can be set so that after the annealing, thousands of truly random binary numbers are measured in parallel. Those can then be converted to uniformly distributed natural or real numbers in desired ranges, either biased or unbiased. We discuss the observed qubits' properties and their influence on the random number generation and consider various physical factors that influence the performance of our generator, *i.e.,* digital-to-analogue quantisation errors, flux errors, temperature errors and spin bath polarisation. The numbers generated by the proposed algorithm successfully pass various tests on randomness from the NIST test suite. Our source code and large sets of truly random numbers will be made publicly available on our project web page.

INDEX TERMS Random numbers, true randomness test, superconducting flux qubits, quantum annealing, QUBO

I. INTRODUCTION

RANDOM numbers have a wide variety of applications in computer science ranging from Monte Carlo simulation [46], to randomised sampling [26], and cryptography [33], [55].

Ideal, non-deterministic generators output unpredictable values obeying a certain (marginal or expected) probability distribution, also known as true random numbers [54]. Unfortunately, classical computers operate deterministically and cannot be used to generate truly random numbers. Instead, they rely on *pseudo random number generators* (PRNG) that produce statistical approximations of random numbers. A PRNG is a classical algorithm with a secret internal state initialised from a pre-defined seed. It then produces a sequence of numbers that emulates statistical properties of a truly random sequence [28]. Since its internal state evolves deterministically, it is possible to predict the number sequence if the seed is known.

However, while some applications may tolerate replacing truly random numbers by pseudo-random ones, others may lose crucial properties, *e.g.,* true unpredictability in security contexts [55]. Building a *true random number generator* (TRNG) is therefore crucial, but is difficult because it usually relies on an external environment with non-deterministic effects, *e.g.,* thermal noise or radioactive decay [2]. Unlike these classical phenomena, certain aspects of quantum systems are not just practically unpredictable but also fundamentally non-deterministic, providing completely random behaviour. While specialised devices tailored to generating random numbers based on quantum effects exist, they are difficult to access or modify and hard to integrate into quantum algorithms. We instead focus on general-purpose quantum computers, whose current-generation Noisy Intermediate State Quantum (NISQ) [51] technology can be found in quantum chips by Google [5], IBM [1] and D-Wave Systems [45]. In recent years, a gate-based system (IBM Q) and an adiabatic quantum annealer (D-Wave System) have become available for research and commercial purposes. Both gate-based models [38], [40] and annealers [8], [23], [52], [53] have recently been employed for pattern recognition.

In this work, we propose a new method for generating truly random numbers using *Radio Frequency Superconducting Quantum Interference Device* (RF-SQUID) qubits. The availability of current-generation quantum hardware coupled

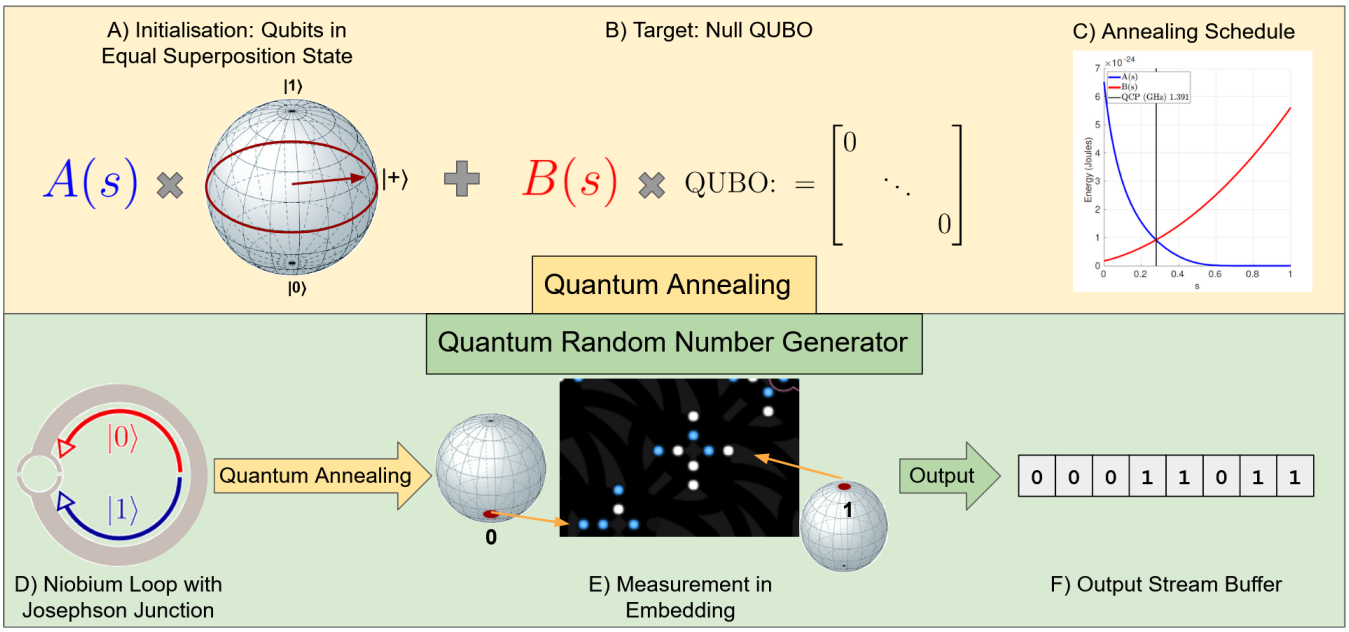


FIGURE 1: We use the quantum annealers D-Wave Advantage 4.1 and 2000Q to generate truly random numbers. (Top) Quantum Annealing with a Null QUBO: We anneal from (A) the initial equal-superposition $|+\rangle$ state to (B) the target null QUBO, following (C) the annealing schedule [13]. (Bottom) To generate random bits on (D) real quantum hardware, we perform quantum annealing. Afterwards, (E) we measure the qubits on the hardware as $|0\rangle$ or $|1\rangle$, (F) generating a stream of truly random bits. To generate an arbitrary number of random bits, we repeat this process over multiple anneals. For an overview of the method refer to Sec. IV-A. Niobium loop adapted from [53].

with the ease of implementation of our method allows the user to generate truly random numbers without acquiring any special hardware. Unlike prior work by Li *et al.* [39], which uses a gate-based system to generate random numbers, our method uses D-Wave's quantum annealer, the only quantum annealer that is easily accessible via the cloud at the time of writing. Gate-based and annealing systems differ substantially from each other. D-Wave offers a *quantum processing unit* (QPU) in its quantum annealer with $>5k$ qubits, a substantially larger number than the 127 qubits present in IBM's gate-based quantum processor. We can thus measure more bits in parallel on the quantum annealer than on the gate-based machine. Furthermore, quantum-annealing methods are already used to solve real-world problems (*e.g.,* permutation synchronisation [8], RNA folding [20]), which makes it desirable to have a directly compatible RNG.

Unfortunately, adiabatic quantum computers are limited to solving *quadratic unconstrained binary optimisation* (QUBO) problems, and do not offer the flexibility to measure qubits easily without performing an anneal. Performing an anneal, inevitably, adds additional noise to the system which leads to noise in the results. Still, one significant advantage of the quantum annealer is its reduced sensitivity to noise and decoherence. Taking these challenges into account, we propose a formulation for generating random bits that ensures independence among the qubits, while minimising the *output bias*, *i.e.,* the bias of the generated bits. Note that the generated bits can be easily turned into uniformly distributed integers or real numbers $\in \mathcal{U}(0,1)$ via interval halving, and from there into any other distribution (given its CDF) via inverse transform sampling. To summarise, the primary contributions of this article are as follows:

- A new generator of true random numbers based on the D-Wave quantum annealer, crucially taking into account the sources of error.
- A variant of our generator that allows to generate biased random numbers.
- An analysis how different sources of hardware errors (digital-to-analogue quantisation errors, flux errors, integrated control errors, temperature errors and spin batch polarisation) influence the performance of the proposed algorithm.

This is the first comprehensive and detailed study of the random number generation capabilities of the D-Wave quantum annealer. We show how to initialise the QPU, set the annealing schedule, choose a minor embedding, and post-process the measured bit sequence. We also show how temperature needs to be taken into account in order to generate *biased* random numbers. In a broader sense, this article explores one of the fundamental properties of quantum machines: qubit randomness. How well it is realised in hardware affects all computational steps of a quantum algorithm, either on gate-based or quantum-annealing machines. We thus investigate the scenarios in which random properties of qubits are preserved and in which deviations from the expected behaviour are to be expected.

We perform extensive experiments with our random number generator on D-Wave's 2000Q and Advantage. Experiments with the NIST test suite show that the D-Wave QPUs are well suitable for the generation of truly random numbers with the proposed algorithm. Moreover, we conclude that the Advantage is better suited for unbiased random number generation than the 2000Q.

The remainder of the paper is structured as follows: Sec. II gives a brief literature review. The preliminaries are introduced in Sec. III. We present our proposed methodology and the design choices in Sec. IV. In Sec. V, we compare different hardware choices for unbiased and biased RNG and test our proposed method. Sec. VI provides an analysis of the results, limitations and possible future work. Finally, we draw conclusions in Sec. VII.

II. RELATED WORKS

In the literature, various random numbers generators (RNGs) have been developed in order to obtain random binary sequences. They fall into two categories: pseudo and true random number generators. The TRNGs use non-deterministic (physical systems) entropy sources as their building blocks. TRNGs are unpredictable and non-reproducable (*i.e.,* have no period). However, TRNGs have significant drawbacks such as higher computation time, high costs and usually require special hardware. As a consequence of the high cost of TRNGs, PRNGs are employed in everyday settings. PRNGs are cheap, easily implemented and do not require any special hardware. For cryptographic purposes, it is more suitable to use a hybrid random RNG [61] , where the generator consists of a deterministic part and an additive input from a strong TRNG. Finally, given the implications of random sequences, it is important to gauge the performance of an RNG. A strong RNG ideally passes several test suites such as NIST [7], Diehard [42], or TEstUI [37].

a: Pseudo Random Number Generators

PNRGs are based on complex mathematical algorithms, with the most famous and widely used being the Mersenne Twister [43]. This PRNG is used by Python, Julia, Linux and others. Due to the sensitivity of chaotic systems, they have been used as PRNG for more than two decades [34], [47], [48]. The recent advent of machine learning has led to several neural-network-based PNRGs [17], [58], which demonstrate a strong performance in statistical randomness testing. While these approaches use neural networks as part of a more complex algorithm, end-to-end neural PRNG was introduced by Bernardi *et al.* [16] and has been further improved since then [49].

b: True Random Number Generators

Research on producing truly random numbers has been carried out for decades, with investigations into several classical entropy sources like jitter in field-programmable gate array (FGPA) integrated circuits [6], ring oscillators [18], or metastability [59]. Non-classical quantum effects are another source of entropy for true random number generation. A traditional example of quantum randomness is radioactive decay [2]: Although each atom has a certain probability of decaying in a each time interval, the exact moment of decay is unknowable beforehand. More recently, optical quantum RNGs have gained immense popularity due to the inherent randomness present in many parameters of the quantum states of light. This has given rise to a plethora of different implementations: One of the first implementation was proposed by Jennewein *et al.* [29], which splits a beam of photons on a beam splitter. Many other methods involving, for example, Lazer Phase noise [24], [30] or shot-noise measurement [21], [56] have been proposed. There is a need for quantum random numbers to perform, for example, various cryptographic computations. To facilitate this, Huang *et al.* [27] integrate four different types of quantum RNGs on Alibaba's cloud services, performing an XOR operation on their output for practical security. We also refer to a comprehensive review of quantum RNGs [25]. Unlike these methods, our approach does not depend on devices specialised in generating random numbers but rather uses a general-purpose quantum computer. Since this QPU is easily accessible via the cloud, our method has the potential to democratise the usage of TRNGs.

There is only little prior work on using quantum computers to generate random numbers. Tamura and Shikano [57] use statistical RNG tests for randomness to test the condition and stability of various qubits present in IBM's gate-based hardware. They ultimately find that the qubits present in the IBM's hardware are unsuitable for random number generation. Recently, Li *et al.* [39] have proposed a protocol that successfully generates random numbers on a gate-based quantum computer while simulatenously accounting and estimating the state preparation error present in the equal superposition state Unlike prior works, we do not use a gate-based model, but rather our method utilises the random nature of RF-SQUID qubits present in the D-Wave quantum annealer, in order to generate uniformly distributed random numbers. *This is the first systematic and in-depth study of random number generation capabilities of modern general-purpose quantum annealers.*

c: Quantum Annealing

Quantum annealing [32] has been used primarily to solve $\mathcal{NP}$-hard optimisation objectives formulated as Quadratic Unconstrained Binary Optimisation (QUBO) problems. Given the commercial impact of quantum annealing, several organisations have made heavy investments into quantum computers using superconducting qubits [4], [31]. Thus, the modern-day D-Wave Advantage offers more than 5000 qubits [45]. In contrast to existing works, ours is the first comprehensive and detailed study of random number generation on D-Wave QPUs. However, the current generation annealers are far from perfect and have several error sources (see also Sec. III-A4). Krauss *et al.* [35] perform an in-depth investigation into the single qubit biases present in

the D-Wave system. They conclude that individual qubits show a time varying qubit bias, although the measurable bias is insignificant when considered in a large population. Our proposed method accounts for the error sources and qubit bias by design.

III. BACKGROUND AND NOTATIONS

A. ADIABATIC QUANTUM COMPUTING

1) Physical Implementation

Current generation quantum annealing hardware produced by D-Wave is based on experimental results of condensed matter physics. Specifically, it exploits quantum tunnelling to search for the ground state of spin-glass systems. The spin-glass system is implemented as a superconducting integrated circuit made up of pair-wise coupled CCJJ rf-SQUIDs (compound Josephson junction, radio frequency, superconducting quantum interference devices), which needs to be kept cooled at cryogenic temperatures close to absolute zero ($\sim 15\mu$K). Each physical qubit in this circuit is made of niobium (Nb) in the shape of a double ring interrupted and controlled by two Josephson junctions [44]. An external magnetic field is used to modify the flow of the current through the double ring. The direction of flow is the quantum state of the qubit: clockwise, anticlockwise, or in both directions at once when the qubit is in quantum superposition.

2) Quantum Annealing

To specify an optimisation problem for the quantum computer, we use the Hamiltonian operator, which outputs the energy of the state of a quantum system. (Specifically, the eigenstates of a time-dependent Hamiltonian are the stationary energy states, whose corresponding eigenvalues give the energy of the system.) The D-Wave system we use is based on the spin-glass Ising formulation for the Hamiltonian. In this formulation, each qubit i of the system has a qubit bias $h_i \in \mathbb{R}$ towards $|0\rangle$ or $|1\rangle$. Furthermore, each qubit i can be coupled with each other qubit j with a strength $J_{i,j} \in \mathbb{R}$, which encourages (anti-)correlation of the qubits' states if $J_{i,j} \neq 0$. Quantum annealing [32] is an optimisation technique that takes as input a *problem Hamiltonian* and then seeks to find the quantum state that minimises that Hamiltonian. To that end, the qubits are first initialised to a known minimum energy state of a generic initial Hamiltonian, namely to the equal superposition state $|+\rangle^{\otimes n}$. The adiabatic theorem [3] states that, if the Hamiltonian is changed slowly enough, a quantum system that minimises its current Hamiltonian will evolve to stay in a minimising state. Thus, slowly transforming (*annealing*) the initial Hamiltonian into the problem Hamiltonian will put the qubits in a state that minimises the problem Hamiltonian and hence is the solution to our optimisation problem. The overall runtime of the annealing process is denoted by τ and for simplicity we consider the rescaled time $s = \frac{t}{\tau} \in [0, 1]$ where t is time. Then, the Hamiltonian over the course of the anneal is given by:

$$\mathcal{H}_{\text{Ising}} = \frac{A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right) + \frac{B(s)}{2}\left(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right), \tag{1}$$

where $A(s)$ and $B(s)$ are the anneal schedules of the initial and problem Hamiltonian, respectively, and σ_x, σ_z are the Pauli-X and Pauli-Z matrices.

3) Solution Distribution

Ideally, measuring the qubits at the end of the annealing process would lead the qubits to only collapse to states that minimise the final Hamiltonian. However, in practice, the D-Wave system returns solutions under a distribution close to a Boltzmann distribution of the final Hamiltonian, but at a temperature T of the QPU midway through the anneal, which is called the freeze-out point. After this freez-out of the quantum system, the spin-states do not change appreciably while the Ising spin Hamiltonian finishes its evolution. The probability of obtaining a solution with energy E under the final Hamiltonian is given by $\exp(-E/k_B T)$, where k_B is Boltzman's constant. For example, consider a single qubit system with $E = h_i \cdot s_i$, where s_i is the Ising-spin state of the qubit. The energy is calculated at the scaled time s of the freeze-out, using the amplitude of the anneal function $B(s)$. Hence, the probability of the obtaining the state $|1\rangle$ is given by:

$$\begin{aligned} P_t(h_i, T)_{|1\rangle} &= \frac{\exp\left(\frac{B(s)\cdot h_i}{k_B \cdot T}\right)}{\exp\left(\frac{B(s)\cdot h_i}{k_B \cdot T}\right) + \exp\left(-\frac{B(s)\cdot h_i}{k_B \cdot T}\right)} \\ &= \frac{1}{1 + \exp\left(-\frac{2B(s)\cdot h_i}{k_B \cdot T}\right)}. \end{aligned} \tag{2}$$

4) QPU Error Sources

The hardware of D-Wave's QPU is bound to several errors [10], [11]. In this section, we first describe the *integrated control errors* (ICE), which are errors that prevent a perfection translation of the mathematical problem (qubit biases h_i and couplings $J_{i,j}$) into the physical QPU. We also discuss the temperature error and spin bath polarisation, which have undesirable impacts on the optimisation.

Background Susceptibility Noise: Background susceptibility is one of the more prominent errors. Physically close rf-SQUID qubits become weakly coupled. The resulting interactions between neighbouring qubits include undesirable h_i qubit bias leakage from a qubit to its neighbours, and spill-over coupling from qubit i to k if i and j, and j and k are coupled via $J_{i,j}$ and $J_{j,k}$.

Flux Error: The qubit wiring causes magnetic field fluctuations called *1/f flux noise*. This introduces an independent (yet time-dependent) error term for each h_i. The flux error is corrected each hour by D-Wave in order to bound its impact. This error leads to an approximate relative error of $\Delta h \approx 0.01$.

DAC Quantisation Error: The Ising spins are analogue in nature and are regulated by spatially local magnetic fields. There is a finite quantisation step that exists when converting digital to analogue, which is the *Digital-to-Analogue (DAC) quantisation error*. Physically, a zero value for the qubit bias also needs to be embedded onto the QPU using the magnetic fields, with the typical quantisation step giving a worst case error of $\Delta h \approx 0.008$ for $h = 0$ and $\Delta J \approx 0.002$ for $J = 0$.

Temperature Error: As discussed earlier, the temperature T of the QPU impacts the optimisation. T depends on the specific physical instance of the QPU architecture and is specified in the documentation [12]. However, while running, small amounts of heat are transferred in the QPU. Depending on the frequency of anneals, this can raise the effective temperature of the qubits, resulting in a fluctuation of a few mK.

Spin Bath Polarisation: In addition to the flux error, current passing through the wires can cause polarisation of the qubit spins, which ultimately biases the qubits. The magnitude of this effect is directly proportional to the annealing time, with longer anneal schedules resulting in increased polarisation and a change in the qubit bias. Anneal-to-anneal correlations may occur because of the partially polarised qubits, which can be avoided by allowing the spin bath to depolarise between anneals.

B. INPUT FORMULATION

The optimisation problem described by the final Hamiltonian can be formulated as a Quadratic Unconstrained Binary Optimisation problem (QUBO) [22]:

$$\text{QUBO: } \underset{x\in\{0,1\}^n}{\arg\min}\, x^t Q x, \tag{3}$$

where $x \in \{0,1\}^n$ are the decision variables and $Q \in \mathbb{R}^{n\times n}$, *i.e.,* a square matrix whose diagonal accounts for per-qubit biases and whose off-diagonals are inter-qubit coupling weights. QUBO problems belong to the class of $\mathcal{NP}$-hard problems, see Lucas *et al.* [41] for examples. D-Wave provides an API for its QPU, which takes as input a QUBO and then applies the corresponding Hamiltonian to the physical qubits of the QPU. The mapping from the decision variables to the physical qubits is called an *embedding*. To *embed* a problem on the QPU is to initialise the physical qubits according to a QUBO and its embedding.

C. RANDOM NUMBER GENERATORS

a: χ^2 Test

The χ^2 test is used as the most simple statistical hypothesis test to check for randomness in a bit sequence. It assumes that n random samples are divided into k mutually exclusive classes—in our case of random bit sequences, there are two classes: 0 and 1 bits. The null hypothesis gives a theoretically expected value E_i for each class. Pearson [50] proposed that for validating the hypothesis, as $n \to \infty$, the following quantity follows a χ^2 distribution with $k-1$ degrees of freedom:

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}. \tag{4}$$

In the context of random numbers, it tests whether there are statistically significant differences between the expected and observed frequencies.

b: National Institute of Standards and Technology (NIST) Test Suite

The NIST statistical test suite for random number generators [7] is the most popular application to test the randomness of sequences of bits. It is designed to probe three main properties of a random number generator:

1) Investigate the distribution of 0s and 1s (independent of the order);
2) Analyse harmonics of the sequence using spectral methods;
3) Detect patterns based on probability theory.

To test these properties, the suite uses the χ^2 test and 14 other tests. For example, the serial test is based on the fact that every bit pattern of length m has the same probability of occurrence in the sequence, while the Discrete Fourier Transform (DFT) test is designed to identify any periodic features in the sequence. Each test produces a *p-value*, *i.e.,* the probability of obtaining the measured result if the null hypothesis is true. The null hypothesis we seek to test is perfect randomness, *i.e.,* $P(b_i = 1|\{b_j\}_{j\neq i}) = 0.5$. It is rejected with a confidence of $1-\alpha$ if the p-value lies below a significance level of $\alpha = 0.01$. (The confidence level for randomness has been adapted from the NIST Test Suite [7].) Otherwise, we fail to reject the null hypothesis. Since the *random excursions* and the *random excursion variant* test produce several p-values, we combine these into a single p-value for each of these tests using Fisher's method (discussed next).

c: Fisher's Method

Fisher's combined probability test [19] is designed to aggregate results from several independent tests while performing hypothesis testing. The test combines k p-values into a test statistic, X^2, which follows a χ^2 distribution with $2k$ degrees of freedom:

$$X^2 = -2\sum_{i=1}^{k} \log(p_i), \tag{5}$$

where p_i is the p-value for the ith statistical test. The final p-value is obtained for the X^2 test statistic.

IV. METHOD

We present a true random number generator based on non-deterministic quantum effects. We first provide an overview of the algorithm and then move to describing the algorithm in Sec. IV-B and the post-processing in Sec. IV-C. We then discuss how errors propagate from the physical qubits to

the final measurements in Sec. IV-D. Finally, we turn to generating biased random numbers in Sec. IV-E and discuss expected errors in Sec. IV-F.

A. OVERVIEW

We use the quantum annealers D-Wave Advantage 4.1 and 2000Q to generate truly random numbers. We initialise the annealer to the equal superposition ($|+\rangle$ state), which has, in theory, an equal probability for each qubit being measured as $|0\rangle$ or $|1\rangle$. The qubits that are implemented in hardware as niobium loops cannot be directly measured; rather, we have to anneal them first. During an anneal, the initial state, minimising the initial Hamiltonian (weighted by $A(s)$), is slowly transformed into a state that minimises the problem Hamiltonian (weighted by $B(s)$). $A(s)$ and $B(s)$ are parameters of the anneal schedule and are varied as depicted in Fig. 1The Hamiltonian varies during the anneal following (1). To prevent the qubits from leaving their initial $|+\rangle$ state, we use the *null* QUBO, which the API turns into the problem Hamiltonian. Crucially, we take into account and mitigate several sources of errors that occur when embedding the problem Hamiltonian on the physical qubits and during the anneal. After the anneal, the physical qubits on the QPU are measured as either $|0\rangle$ or $|1\rangle$, generating a stream of truly random bits. This entire process is repeated over multiple anneals, re-initialising the QPU and generating a new embedding before each anneal. A single anneal generates random bits (scaling with the size of the QPU) and multiple anneals allow us to generate multiplicatively many random bits.

B. ALGORITHM

We design the following method to generate random numbers, while taking into account the error sources described earlier. We consider N qubits on the QPU and perform A anneals to generate a bit sequence of length AN.

1) RNG Initialisation

At the core of our method lies the measurement of an equal superposition quantum state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The probability of measuring $|0\rangle$ or $|1\rangle$ as the outcome is equal, *i.e.,* $= 0.5$. D-Wave's QPU allows to put the physical qubits into this quantum state by initialising the quantum annealer with any embedding. Ideally, we would then directly measure the qubits. Unfortunately, this is infeasible on D-Wave's QPU because a minimum annealing time before measurement always exists. We seek to circumvent this issue by keeping the system in its initial quantum state for the duration of the anneal time. We choose the minimum anneal time both for higher throughput and to reduce the undesirable effects of spin bath polarisation. However, naïvely providing an empty QUBO to the QPU's API would result in an empty embedding, which would not invoke any physical qubits. Instead, we pass a *null* QUBO to the API:

$$\sum_{i=0}^{N} x_i - \sum_{i=0}^{N} x_i, \tag{6}$$

where x is a vector representing the states of the QUBO variables.

2) Choosing Problem Size

The number of qubits considered per anneal greatly affects the parallel nature of our RNG and, subsequently, the throughput. On an ideal (hypothetical) QPU, taking many anneals—each with a small number of qubits and with independent embeddings—would be equivalent to taking only a few anneals but with a larger number of qubits per anneal. However, the D-Wave QPU has several error sources (see Sec. III-A4 for details) that could potentially make the two settings not equivalent.

In particular, the DAC quantisation errors are distributed over all physical qubits. While the magnitude of their influence on any specific h_i or $J_{i,j}$ is independent of the overall number of qubits that are used in the embedding, the DAC quantisation error can cause (undesirable) residual couplings $J_{i,j} \neq 0$ between qubits. When a higher number of qubits per anneal is used, each qubit i has a higher number of other qubits j with which residual couplings can happen. This then would lead to more (anti-)correlations between qubits and hence between generated bits, when more qubits are used.

The other error sources do not depend on the problem size.

We perform experiments (see *e.g.,* Fig. 2 and Sec. V) to understand the variation of bias in the output sequence due to these disturbing effects. Surprisingly, we empirically find that the output bias is independent of the number of qubits used in the anneal. Hence, we use all 4950 qubits at once which allows us to measure thousands of truly random binary numbers in parallel. This, in turn, leads to a high throughput of our RNG.

3) Choosing Embeddings

Due to the nature of our generator, our problem Hamiltonian does not have any qubit bias or coupling present, which implies minuscule levels of bias leakage and hence negligible background susceptibility noise. However, the impact of the DAC quantisation and "1/f flux" errors depend on whether the problem gets re-embedded on the QPU. If we do not re-embed the problem for every anneal, the value of the DAC quantisation error of a physical qubit would remain the same across anneals. Further, there are fluctuations in the flux noise that have lower frequency than the typical inverse annealing time, so problems solved in quick succession have highly correlated contributions from flux noise. Since re-embedding takes time, we can circumvent the negative impact of the flux noise. Note that, in the literature, designing the right embedding has an influence on the optimisation and the optimality of the empirical solutions in problems which show high graph connectivity [9]. Since our QUBO does not have couplings, this is not an issue for our problem.

C. POST-PROCESSING

The previous section addressed the issue of correlations in sequences of generated bits, where the unconditional prob-

ability of the next bit, $P(b_{I+1} = 1)$, can differ from the probability conditioned on the previously generated bits, $P(b_{I+1} = 1|\{b_i\}_{i=1}^{I})$. As an additional issue, the *raw* output, *i.e.,* before post-processing, of most truly random number generators [25], including ours, tends to be slightly biased towards 0 or 1: $P(b_i = 1) \neq 0.5$. To address this problem, we employ a naive post-processing technique by von Neumann [60] for improving the entropy of our produced bits by eliminating any output bias present in the non-post-processed bit sequence. Note that although the post-processing removes any bias present in the sequence, it does not convert a non-random sequence into a random sequence, *i.e.,* it does not eliminate potential already present patterns in the sequence. The von Neumann corrector splits the bit stream $\{b_i\}_{i\in[0,1,2,\ldots,K-1]}$ of length K into successive pairs $\{(b_i, b_{i+1}\}_{i\in[0,2,4,\ldots,K-1]}$ and discards pairs that contain identical bits ($b_i = b_{i+1}$). For pairs where the bits are distinct, the corrector keeps the first bit and discards the second bit. For an RNG, whose output is biased by e, *i.e.,* the probability of getting $|0\rangle$ is $0.5 + e$, this scheme reduces the number of bits to $(25 - 100e^2)\%$ of the unprocessed number of bits.

D. PROPAGATION OF ERROR IN PROBABILITY

For our later analysis, we need to understand how the probability of obtaining 0 or 1 varies with the error δh of the qubit bias h (due to the ICE errors). Specifically, the error propagation δP into the probability P defined in (2) is:

$$\delta P = \pm \left(\frac{2B(s)}{k_B T} \cdot \frac{e^{-\beta \cdot 2B(s)h}}{\left(1 + e^{-\beta \cdot 2B(s)h}\right)^2} \right) \delta h, \qquad (7)$$

where $\beta = 1/(k_B T)$ is the inverse Boltzmann temperature and the value of $B(s)$ at freezeout temperature remains constant. Unfortunately, the QPU does not allow us to measure the exact experimental value of the qubit bias h that has been embedded onto the QPU, but D-Wave provides a range of δh values depending on the various errors (see Sec. III-A4). This is sufficient for us to use this equation to develop an empirical understanding when comparing the errors present in two different D-Wave architectures in the experiments (Sec. V-A).

E. GENERATING BIASED RANDOM NUMBERS

Biased random bits have a different probability of getting $|1\rangle$ than getting $|0\rangle$. Ideally, to generate qubits that are biased to a certain amount, we would be able to specify the right non-zero qubit bias value h for all qubits using the probability transformation given by (2). But Eq (2) depends on the effective temperatures T of the qubits, which can fluctuate depending on the programming cycles. We thus need to first determine the current temperature of the QPU before we can generate random numbers with a certain probability P of obtaining $|0\rangle$. Ultimately, if we know the empirical bias towards 0 for some h, we can use (2) to determine T. Since each physical qubit i can react slightly differently to a specific qubit bias h, we first fix an embedding. We seek a robust estimate of T and thus determine the empirical probabilities $P_{i,h}$ of obtaining $|0\rangle$ for H equally spaced qubit bias values $h \in [-1, +1]$. (Note that we use the same qubit bias for all physical qubits.) However, applying a qubit bias h naively would lead the QPU to automatically rescale the biases of all qubits into the range $[-2, +2]$ before applying them to the physical qubits, which makes it impossible to apply different values of h to them. Thus, we crucially remove the automatic scaling and stay within the physically implementable range of qubit bias values, namely $[-1, +1]$. Over any such qubit bias, we perform A reads[1] to determine $P_{i,h}$. We next remove any per-qubit imperfection. Specifically, for qubit i, the average probability of obtaining $|0\rangle$ across all considered qubit bias values ($u_i = \frac{1}{H}\sum_h P_{i,h}$) should be 50%. We thus correct $P_{i,h}$: $P'_{i,h} = P_{i,h} - (u_i - 0.5)$. Next, we average across qubits to obtain the probability $P_h = \frac{1}{N}\sum_i P'_{i,h}$ for a qubit bias h and fit the probability curve given by (2) to $\{P_h\}_h$ by minimising the mean squared error. We thus obtain the temperature T.

Knowing T finally enables us to compute the qubit bias h for a desired probability P via the probability transformation given in (2). The QUBO to generate biased random numbers is then:

$$QUBO : \min \sum hx_i. \qquad (8)$$

F. ERROR ACROSS PHYSICAL QUBITS

Due to errors (Sec. III-A4), annealing with a certain qubit bias h will result in varying empirical probabilities $P'_{i,h}$. We investigate how this variation in $P'_{i,h}$ differs from the theoretically expected variation and how it differs between QPU architectures. We consider the standard deviation $\sigma = \sigma(\{P'_{i,h}\}_i)$. Since QPU time is expensive, the number of bits generated per qubit is quite small. This results in a theoretical standard deviation σ_t which would be present even on an ideal QPU. The probability of obtaining $|1\rangle$ from an ideal qubit with a qubit bias h at temperature T is denoted by $P_t(h, T)$. Then, the probability of getting the state $|1\rangle$ s times when annealing an ideal qubit A times is given by:

$$P_{freq}(s) = \frac{P_t(h,T)^s (1 - P_t(h,T))^{A-s} \binom{A}{s}}{\sum_{s'=0}^{A} P_t(h,T)^{s'} (1 - P_t(h,T))^{A-s'} \binom{A}{s'}} \qquad (9)$$

We get the theoretically expected number of ideal qubits that have probability $\frac{s}{A}$ as $N \cdot P_{freq}(s)$, from which we compute σ_t.

V. RESULTS

This section describes the experimental results. All experiments are done on an Intel Core i7-8565U CPU with 8GB RAM using Python 3.9. The QPUs, namely the Advantage System 4.1 (Pegasus architecture [15]) and the DW_2000Q_6 (Chimera architecture [15]), were accessed using Leap 2 [14].

[1]We use *read* to refer to an anneal in a series of anneals where the embedding and biases remain the same across anneals, and the QPU is not properly reset in between anneals.

A. HARDWARE COMPARISON

We first determine which D-Wave architecture is more suitable for our random number generator: the Advantage or the 2000Q. The Advantage is the newer QPU, having more than 5000 qubits, but the 2000Q, which has 2000 qubits, is the system with less noise in δh [45]. To compare between them, we analyse sequences of length $A'N'$ with $A' = 200$ anneals of $N' = 4950$ (2000) qubits for the Advantage (2000Q), without post-processing. We generate ten sequences on each architecture. First, we calculate each sequence's output bias e. Fig. 2b shows the results. We see that the output bias present in the 2000Q is greater than in the Advantage. This is surprising as the 2000Q is considered to have less errors. We next compute the standard deviation of the output bias across the physical qubits, following the methodology detailed in Sec. IV-F. The results are in Fig. 7. As we are currently considering unbiased random numbers, the standard deviation at a qubit bias of 0 is of importance, which is higher for the 2000Q than for the Advantage. This is again unexpected, as the 2000Q is considered the less noisy system.

In order to understand these counter-intuitive results, we use the propagation of error in the probability, see Sec. IV-D. For our special case of an unbiased qubit ($h_i = 0$) and no couplings ($J_{i,j} = 0$), the propagation of error is $\delta P = \frac{2B(s)}{k_B T}\delta h$. This depends on $B(s)$ and T, which we determine next before we can proceed with our analysis. The 2000Q and the Advantage have different single qubit freezeouts: The single qubit freezeout of the 2000Q is at $0.719\mu s$ with $B(s) = 6.54GHz$, while the Advantage has a single qubit freezeout time of $0.612\mu s$, giving $B(s) = 3.92GHz$. As mentioned in Sec. III-A4, depending on the frequency of anneals on the QPU, the temperature T may vary by a few mK. To calculate the operational temperature, we follow Sec. IV-E and use 4950 (2000) qubits for the Advantage (2000Q) over $A = 1000$ anneals each for $H = 101$ equally spaced qubit bias values $h \in [-1, +1]$. The temperature of the Advantage and the 2000Q are $18.9mK$ and $11.9mK$, respectively, which gives the graph in Fig. 3. Overall, for our case of unbiased qubits without couplings, we thus find a higher value of $\frac{B(s)}{k_b T}$ for the 2000Q than for the Advantage. This increases the impact of errors δh on the output for the 2000Q, which leads to a higher standard deviation across the qubits and a higher bias in the output sequence for the 2000Q than for the Advantage. We thus focus more on the Advantage rather than the 2000Q for our RNG.

B. TESTING UNBIASED RANDOM NUMBER GENERATION

a: Evaluation Methodology

We use the NIST Test Suite [7] (Sec. III-C0b) to evaluate the generated bit sequences. It contains 15 tests. However, for sequences shorter than $387,840$ bits, the *universal* test cannot be performed, and for sequences shorter than 10^6 bits, the *random excursions*, *variational random excursions*, *linear complexity*, and *overlapping patterns* tests cannot be performed. Unfortunately, quantum compute remains expensive and the hardware has limited number of qubits, which makes generating multiple sufficiently long sequences for these tests not computationally feasible within a reasonable budget. We hence omit these tests from the results. We consider the log-sum of the p-values according to Fisher's method as summary statistic over the different tests.

b: Generating Test Sequences

Since the NIST test suite is usually evaluated on multiple sequences , we generate several sequences of random bits. Although the Advantage has more than 5000 qubits, the number of qubits that can be utilised is limited due to the need to run the QPU at cryogenic temperatures in a low–magnetic-field environment. Hence, at any given point only a subset of them are available to the users. For our experiments, we were able to consistently utilise 4950 qubits, allowing us to produce 4950 bits in parallel. To test our RNG for stastical properties of randomness, we generate ten sequences using 4950 qubits for 200 anneals. This results in sequences of length $990k$ bits before post-processing and approximately $247k$ after post-processing.

c: Results

As Fig. 2b shows, the sequences exhibits output biases if no post-processing is applied. This leads to several tests failing on almost all of the ten sequences, as the left side of Table 1 shows. The raw bit sequences are thus not random. However, the post-processing eliminates the output bias and hence increases the p-value on average (see Fig. 4 and right side of Table 1), which lets the post-processed sequences pass all applicable tests.

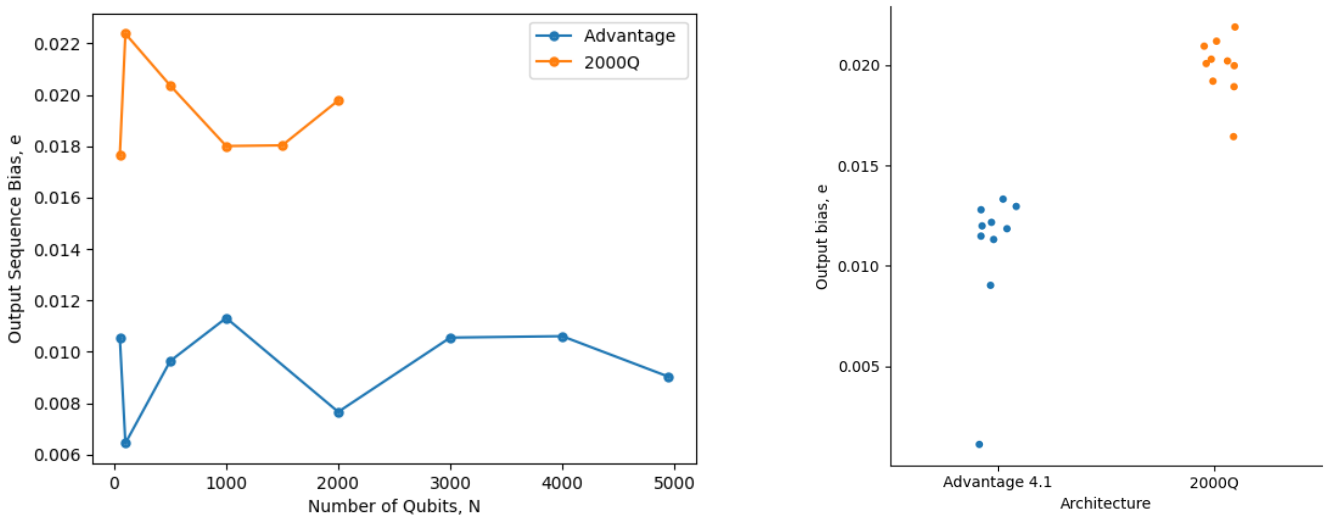
d: Single Long Sequence

Since we pass the NIST test suite for multiple, shorter sequences but lack the computational resources to generate multiple large sequences, we next evaluate a single long sequence. This allows us to run several additional tests that require long inputs. We anneal 2000 times to obtain a sequence containing 9.9 million bits, which reduces to $2,474,434$ bits after post-processing. The p-values for each test in the NIST test suite is given in Fig. 5, with a cumulative Fisher value of 0.18. The long sequence passes all tests.

C. TESTING BIASED RANDOM NUMBER GENERATION

We finally evaluate how well our method generates biased numbers. Since the NIST test suite only works for unbiased numbers, we use the χ^2 test (Sec. III-C0a) to assess the results. Note that we do not apply von Neuman-post-processing since it would eliminate the bias.

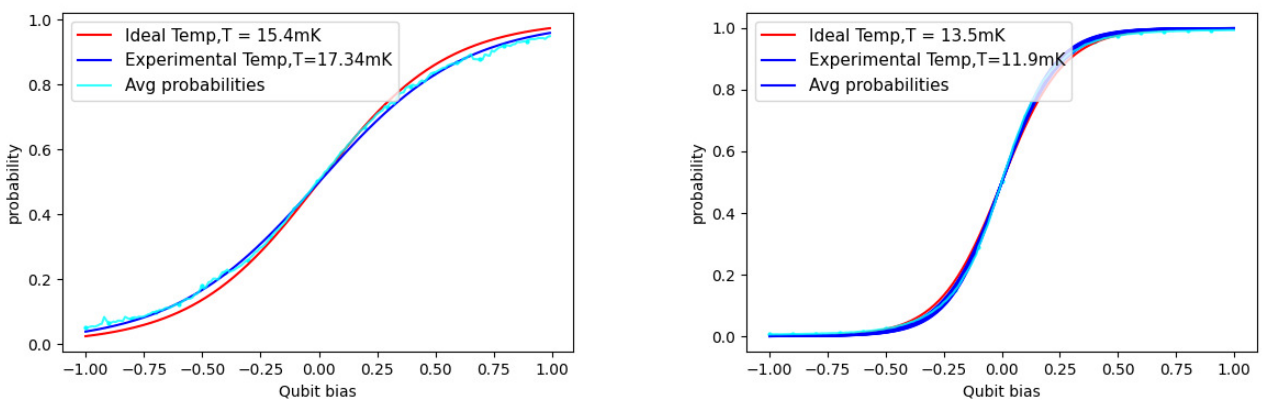
We first determine which architecture is better suited. For *unbiased* numbers, we found the Advantage to be preferable due to, among other reasons, its lower standard deviation of probabilities across physical qubits (see Fig. 7). In fact, the Advantage exhibits smaller standard deviations for all qubit



(a) Output bias variation with problem size

(b) Output bias variation across runs

FIGURE 2: *Output Bias Variation.* We investigate the output bias e (without post-processing). The output bias is the value by which 0.5 is exceed in the probability of a given quantum state. a) First, we vary the number of qubits, perform 200 anneals each, and plot the output bias. The Advantage has consistently lower bias. b) Next, we generate 10 sequences by using the maximum number of usable qubits per architecture (2000 and 4950 respectively) for 200 anneals each giving us sequences of length $400k$ and $990k$ bits. The mean output bias for the 2000Q is 0.0199 with standard deviation 0.0015. Similarly, the mean output bias for the Advantage is 0.010 with standard deviation 0.0036. The Advantage again has consistently lower bias. Note that the higher standard deviation of the Advantage is due to an outlier having significantly lower output bias.



(a) Advantage

(b) 2000Q

FIGURE 3: *QPU Temperature Determination.* We follow Sec. IV-E to determine the experimental QPU temperature T. For equally spaced qubit biases $h \in [-1, +1]$, we anneal the QUBO given in (8) and compute the corresponding average output biases over all qubits (4950 for the Advantage, 2000 for the 2000Q). The cyan plot shows these average probabilities. Least-squares fitting to the probability curve (2) yields the blue plot, allowing us to compute the average QPU temperature of $18.9mK$. The red line shows the expected probabilities under the ideal temperature, which is specified in the QPU-specific properties documentation [12]. *Using the experimental temperature instead of the ideal one fits the observations significantly better.* The mean squared error after least-squares fitting for the Advantage and the 2000Q are $1.5 \cdot 10^{-4}$ and $6.9 \cdot 10^{-5}$ respectively. We thus find that the 2000Q follows the error model better, which makes it easier to use (2) for biased RNG than the Advantage.

TABLE 1: *P-values for Each Test.* We compare NIST test suite results of ten different sequences generated by the Advantage, before and after post-processing. For each test, we report the average and worst value across the sequences, as well as the number of sequences that pass each test.

Test	Before Post-Processing			After Von-Neumann Post-Processing		
	Average p-value	Worst p-value	Tests Passed	Average p-value	Worst p-value	Tests Passed
monobit	0.0027	0	1	0.6271	0.322	10
block frequency	0.0431	0	1	0.3592	0.0388	10
runs test	0.0007	0	0	0.408	0.1077	10
longest one block	0.1722	$2.9 \cdot 10^{-7}$	7	0.4465	0.1304	10
binary matrix rank	0.3906	0.0819	9	0.6705	0.1356	10
spectral	0.5877	0.0002	8	0.4847	0.0303	10
non overlapping	0.0725	0	1	0.4643	0.0956	10
approximate entropy	0.0512	0	1	0.4378	0.0568	10
cumulative sums (forward)	0.0035	0	1	0.5744	0.2542	10
cumulative sums (backward)	0.0045	0	1	0.6103	0.1333	10

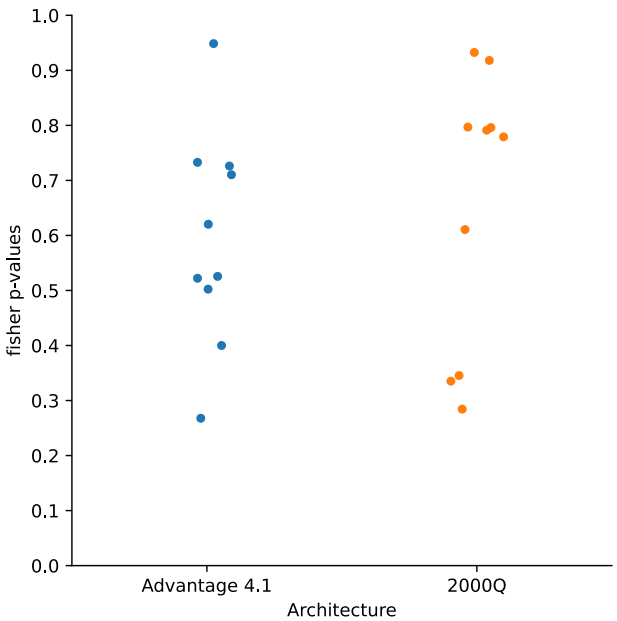


FIGURE 4: *Fisher p-values* Combined Fisher's values for ten sequences that are generated via 200 anneals of the maximum number of usable qubits on the Advantage and the 2000Q, 4950 and 2000, respectively. Both architectures pass the test suite and we find no significant difference in terms of summary p-values.

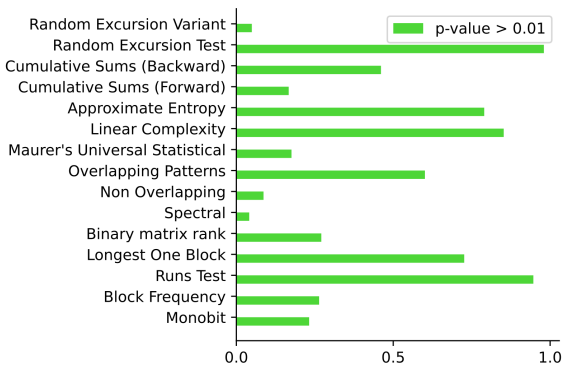


FIGURE 5: *P-values for Each Test.* A single long sequence is generated by using 4950 qubits for 2000 anneals, which gives 2.47 million bits after post-processing. The plot shows the p-values of each test, which are all passed. The cumulative Fisher value of the sequence is 0.18.

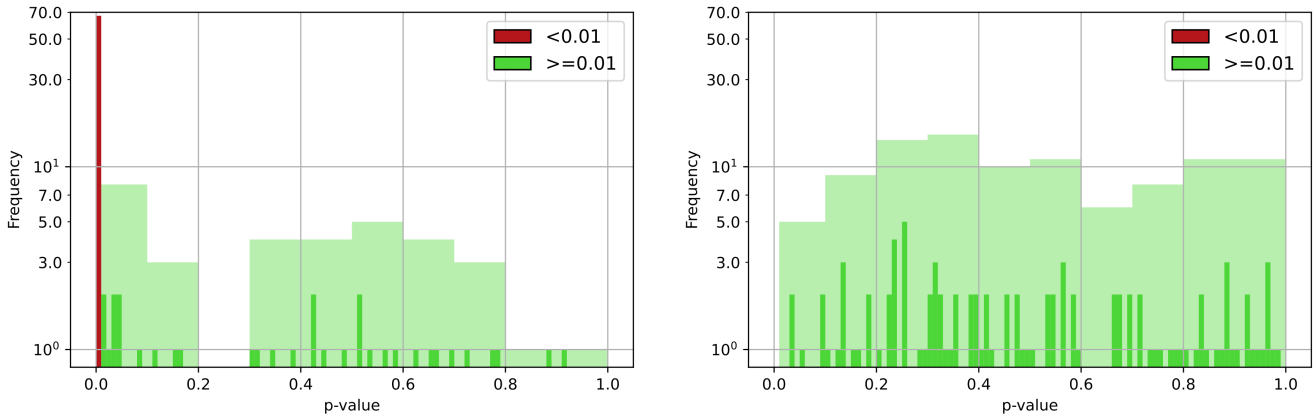
biases close to 0. However, we want to also obtain bits that are more than just slightly biased. For these cases, we require larger qubit biases. The 2000Q has lower standard deviations for such larger qubit biases and is thus preferable. In addition, Fig. 3a shows that, when using the determined temperature, the 2000Q has a lower mean-squared error to the empirical output biases than the Advantage. That means that the 2000Q follows (2) (which we use to determine the appropriate qubit bias from the desired output bias) more closely than the Advantage. The 2000Q is thus more fit for generating *biased* random numbers than the Advantage, which is also supported by the Advantage failing the χ^2 tests. We thus focus on the 2000Q for biased number generation.

We next ablate whether determining the experimental temperature is necessary or if the ideal temperature could be used instead. We find that directly using the ideal temperature results in the failure of the χ^2 test.

Using the experimental temperature, we perform 200 anneals for each qubit bias. All of the resulting sequences pass the χ^2 test, which depict the capability of the 2000Q to produce biased random numbers.

VI. DISCUSSION

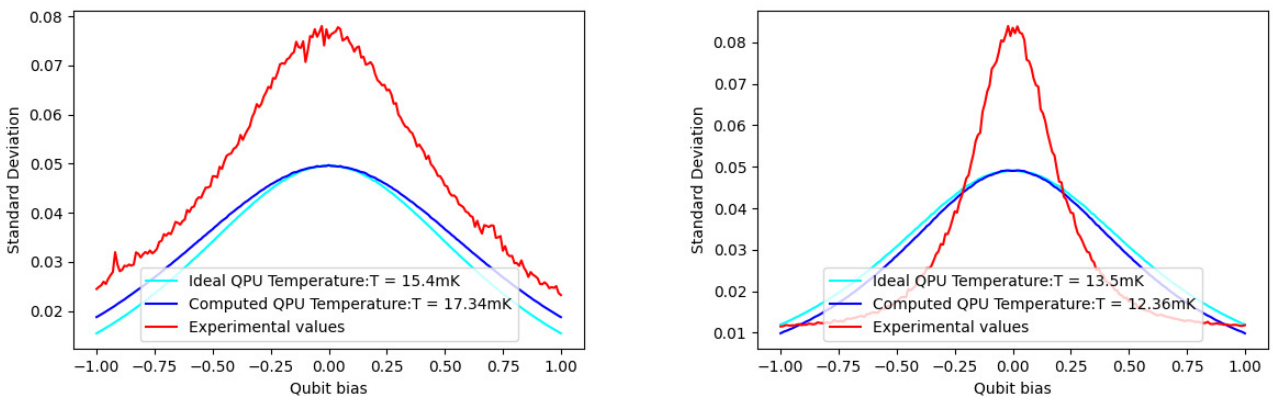
As expected, there are discrepancies between the theoretically predicted behaviour and the empirical observations. The presence of noise prevents the quantum annealer from producing perfectly unbiased random numbers. In general, the 2000Q is the lower-noise system compared to the Advantage. However, our experiments (unbiased qubits without any couplings) show that differences in the annealing parameters and QPU temperature lead errors (δh) to have a stronger impact on outcomes (δP) on the 2000Q than the Advantage.



(a) P-values before post-processing

(b) P-values after post-processing

FIGURE 6: *P-values of All Tests.* Each of the ten sequences generated on the Advantage gives a p-value for each test. Here, we show a histogram over the p-values before and after post-processing, with coarse (bright green) and fine (dark green) binning. The frequencies in the plot are $\log_{10}$-scaled. We see that all sequences pass all tests after post-processing.



(a) Advantage

(b) 2000Q

FIGURE 7: *Standard Deviation of Probabilities Across Physical Qubits.* When computing the temperature (Fig. 3), we anneal with several, equally spaced qubit biases (x-axis). For each such bias, we plot the standard deviation (across the physical qubits) of the per-qubit probabilities. In particular, the experimentally determined standard deviation for each qubit bias is shown in red. In addition, the expected standard deviation, derived from (9), is shown for the measured QPU temperature (in blue) and for the ideal QPU temperature (in cyan).

Because of that, we find that the quantum annealer's ability to generate random numbers is not entirely reflective of the errors (δh) on the QPU and, hence, should not be used as a metric of the quality of the physical qubits. For example, Fig. 7a shows that the standard deviations on the Advantage are less symmetric about $h = 0$ than the 2000Q. Still, we recommend to use the Advantage for our method, which also offers higher parallelisation due to having more qubits.

Careful method design and post-processing enables unbiased random number generation on a quantum annealer. Even though the Advantage is more suited for unbiased random number generation, its deviation from the ideal probability curve diminishes its ability to generate biased random numbers. However, the 2000Q is able to produce biased numbers.

Following the proposed methodology, we can use 4950 qubits to generate 4950 bits in parallel. While the annealing time is only $1\mu s$ per anneal, there are overheads involved which lead to the substantially higher QPU time of approximately $8.7ms$ per anneal (also called *QPU access time*). The latter gives our RNG a QPU throughput of 0.56 megabits per second. We find the QPU access time to be independent of the number of qubits used. The throughput of our RNG is therefore limited by the current overheads present in the QPU.

A. SOCIAL IMPACT

As mentioned in Sec. I, random numbers have many applications, and a way to generate and access provably true random numbers can influence many areas. Thus, cryptographic systems can be made more resilient to attacks if they use truly random numbers, improving security and privacy.

B. LIMITATIONS

Note that the tests in the test suite are not necessarily independent (a requirement for Fisher's method) since failing one test might increase the conditional likelihood of failing another one as well. However, using Fisher's method as a summary statistic makes comparison between different bit sequences more intuitive than only considering the worst p-value.

The von Neumann post-processing is simple but has a fairly low output rate. If higher output rates are desired, other post-processing methods can be used in future [36].

Directly generating biased random numbers is time-consuming due to the need to determine the QPU temperature beforehand. It also follows that any change in temperature or high flux errors might lead to negative results. Hence, we recommend generating unbiased random numbers on the QPU and then applying a subsequent inverse transformation using the CDF.

Although the 2000Q can generate biased random numbers, the Advantage is unable to do so sufficiently well to pass the χ^2 test.

C. FUTURE DIRECTIONS

D-Wave continues to add qubits and to improve their connectivity patterns. This comes at the cost of a more erroneous system, which amplifies the effect of external noise on the annealer. Hence, instead of trying to mitigate the influence of noise as we do, a method could try to harness the random distribution of noise in the hardware as a source of randomness for generating bit streams of random numbers. Unfortunately, the internal sensors of the QPU are not publicly accessible.

We focused on single bits in this first work of using a quantum annealer as an RNG. Directly generating more complicated distributions on the QPU with dependencies between multiple bits might be possible by finding appropriate QUBOs that make certain bit patterns more likely than others.

VII. CONCLUSION

For the first time in the literature, our work exploits random properties of a quantum annealer for the construction of an RNG. Our experiments show that the proposed method can extract truly random numbers from the Advantage quantum annealer. Counter-intuitively, our experiments indicate that the Advantage is better suited for this task than the less noisy 2000Q. We would like to stress that this does not imply that the Advantage possesses superior or less erroneous physical qubits than the 2000Q. Our discussion rather shows how this is a consequence of how qubit bias errors propagate into the output bias. Furthermore, the 2000Q is better suited than the Advantage for generating *biased* random numbers. We hope that our work inspires follow-ups on exploiting general-purpose quantum computers for randomness.

REFERENCES

[1] Ibm quantum, 2021. online; accessed on 2022-01-12.
[2] Ammar Alkassar, Thomas Nicolay, and Markus Rohe. Obtaining true-random binary numbers from a weak radioactive source. In International Conference on Computational Science and Its Applications, 2005.
[3] M. H. S. Amin. Consistency of the adiabatic theorem. Phys. Rev. Lett., 102:220401, 2009.
[4] Mohammad H. S. Amin, Peter J. Love, and C. J. S. Truncik. Thermally assisted adiabatic quantum computation. Physical review letters, 100 6:060503, 2008.
[5] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, and et al. Quantum supremacy using a programmable superconducting processor. Nature, 574:505–510, 2019.
[6] Mohammed Bakiri, Christophe Guyeux, Jean-François Couchot, and Abdelkrim Kamel Oudjida. Survey on hardware implementation of random number generators on fpga: Theory and experimental analyses. Computer Science Review, 27:135–153, 2018.
[7] Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, M Levenson, M Vangel, Nathanael Heckert, and D Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010.
[8] Tolga Birdal, Vladislav Golyanik, Christian Theobalt, and Leonidas Guibas. Quantum permutation synchronization. In Computer Vision and Pattern Recognition (CVPR), 2021.
[9] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. Finding maximum cliques on the d-wave quantum annealer. Journal of Signal Processing Systems, 91:363–377, 2019.
[10] D-Wave Systems, Inc. Error souces for problem representation. online; accessed on 2022-01-12.
[11] D-Wave Systems, Inc. Other error sources. online; accessed on 2022-01-12.
[12] D-Wave Systems, Inc. Qpu-specific characteristics. online; accessed on 2022-01-12.
[13] D-Wave Systems, Inc. Quantum annealing schedule. online; accessed on 2022-09-20.
[14] D-Wave Systems, Inc. Leap, 2022. online; accessed on the 21.01.2022.
[15] Nike Dattani, Szilard Szalay, and Nick Chancellor. Pegasus: The second connectivity graph for large-scale quantum annealing hardware. arXiv e-prints, 2019.
[16] Marcello De Bernardi, MHR Khouzani, and Pasquale Malacaria. Pseudo-random number generation using generative adversarial networks. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 191–200, 2018.
[17] Veena Desai, V. B. Deshmukh, and D. H. Rao. Pseudo random number generator using elman neural network. In IEEE Recent Advances in Intelligent Computational Systems, 2011.
[18] Markus Dichtl and Jovan Dj. Golić. High-speed true random number generation with logic gates only. In Cryptographic Hardware and Embedded Systems(CHES), pages 45–62, 2007.
[19] R. A. Fisher. Statistical Methods for Research Workers. 1992.
[20] Dillion M. Fox, Christopher M Macdermaid, Andrea M. A. Schreij, Magdalena Zwierzyna, and Ross C. Walker. Rna folding using quantum computers. bioRxiv, 2021.
[21] Christian Gabriel, Christoffer Wittmann, Denis Sych, Ruifang Dong, Wolfgang Mauerer, Ulrik L. Andersen, Christoph Marquardt, and Gerd Leuchs. A generator for unique quantum random numbers based on vacuum states. Nature Photonics, 4:711–715, 2010.
[22] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models. arXiv e-prints, 2018.
[23] Vladislav Golyanik and Christian Theobalt. A quantum computational approach to correspondence problems on point sets. In Computer Vision and Pattern Recognition (CVPR), 2020.
[24] Hong Guo, Wenzhuo Tang, Yu Liu, and Wei Wei. Truly random number generation based on measurement of phase noise of a laser. Phys. Rev. E, 81:051137, 2010.
[25] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. Rev. Mod. Phys., 89:015004, 2017.
[26] Felix J Herrmann. Randomized sampling and sparsity: Getting more information from fewer samples. Geophysics, 75:173–187, 2010.

[27] Leilei Huang, Hongyi Zhou, Kai Feng, and Chongjin Xie. Quantum random number cloud platform. npj Quantum Information, 7:107, 2021.
[28] Frederick James and Lorenzo Moneta. Review of high-quality random number generators. Computing and Software for Big Science, 4:2, 2020.
[29] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, Anton Zeilinger University of Vienna, University of Innsbruck, and University of Munich. A fast and compact quantum random number generator. Review of Scientific Instruments, 71:1675–1680, 2000.
[30] M. Jofre, M. Curty, F. Steinlechner, G. Anzolin, J. P. Torres, M. W. Mitchell, and V. Pruneri. True random numbers from amplified quantum vacuum. Opt. Express, 19:20665–20672, 2011.
[31] M. William Johnson, Mohammed Amin, S. Gildert, Trevor Lanting, Firas Hamze, Neil G. Dickson, R. Harris, Andrew J. Berkley, J. Johansson, and et al. Quantum annealing with manufactured spins. Nature, 473:194–198, 2011.
[32] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. Physical Review E, 58:5355, 1998.
[33] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Fast Software Encryption, pages 168–188. Springer Berlin Heidelberg, 1998.
[34] H. Konno and T. Kondo. Iterative chaotic map as random number generator. Annals of Nuclear Energy, 24:1183–1188, 1997.
[35] Thomas Krauss, Alexander Giffen, Phillip Truppelli, and Alan Michaels. Statistical bias in d-wave qubits. Journal of Physics: Conference Series, 1936:012010, 2021.
[36] Patrick Lacharme. Post-processing functions for a biased physical random number generator. In International Workshop on Fast Software Encryption, 2008.
[37] Pierre L'Ecuyer and Richard Simard. Testu01: A c library for empirical testing of random number generators. ACM Transactions on Mathematical Software, 33, 2007.
[38] Guangxi Li, Zhixin Song, and Xin Wang. Vsql: Variational shadow quantum learning for classification. In AAAI Conference on Artificial Intelligence, 2021.
[39] Yuanhao Li, Yangyang Fei, Weilong Wang, Xiangdong Meng, Hong Wang, Qianheng Duan, and Zhi Ma. Quantum random number generator using a cloud superconducting quantum computer based on source-independent protocol. Scientific Reports, 11:23873, 2021.
[40] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. In AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2020.
[41] Andrew Lucas. Ising formulations of many np problems. Frontiers in Physics, 2:5, 2014.
[42] George Marsaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness. Florida State University, 1995.
[43] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul., 8:3–30, 1998.
[44] Catherine Mcgeoch. Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice, volume 5. 2014.
[45] Catherine McGeoch and Pau Farre. The advantage system: Performance update. Technical report, D-Wave Systems, 2021.
[46] Nicholas Metropolis and S. Ulam. The monte carlo method. Journal of the American Statistical Association, 44:335–341, 1949.
[47] Hamid Nejati, Ahmad Beirami, and Warsame Ali. Discrete-time chaotic-map truly random number generators: design, implementation, and variability analysis of the zigzag map. Analog Integrated Circuits and Signal Processing, 73:363–374, 2012.
[48] Narendra K. Pareek, Vinod Patidar, and Krishan K. Sud. A random bit generator using chaotic maps. Int. J. Netw. Secur., 10:32–38, 2010.
[49] Luca Pasqualini and Maurizio Parton. Pseudo random number generation: a reinforcement learning approach. Procedia Computer Science, 170:1122–1127, 2020.
[50] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302):157–175, 1900.
[51] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, 2018.
[52] Marcel Seelbach Benkner, Vladislav Golyanik, Christian Theobalt, and Michael Moeller. Adiabatic quantum graph matching with permutation matrix constraints. In International Conference on 3D Vision (3DV), 2020.
[53] Marcel Seelbach Benkner, Zorah Lähner, Vladislav Golyanik, Christof Wunderlich, Christian Theobalt, and Michael Moeller. Q-match: Iterative shape matching via quantum annealing. In International Conference on Computer Vision (ICCV), 2021.
[54] Mario Stipevic and Çetin Kaya Koç. True random number generators. In Open Problems in Mathematics and Computational Science, 2014.
[55] Berk Sunar. True random number generators for cryptography. Cryptographic Engineering, pages 55–73, 2009.
[56] Thomas Symul, Syed Muhamad Assad, and Ping Koy Lam. Real time demonstration of high bitrate quantum random number generation with coherent laser light. Applied Physics Letters, 98:231103, 2011.
[57] Kentaro Tamura and Yutaka Shikano. Quantum random numbers generated by a cloud superconducting quantum computer. International Symposium on Mathematics, Quantum Theory, and Cryptography, page 17–37, 2019.
[58] Kayvan Tirdad and Alireza Sadeghian. Hopfield neural networks as pseudo random number generators. In Annual Meeting of the North American Fuzzy Information Processing Society, 2010.
[59] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital trng based on metastable ring oscillator. In Cryptographic Hardware and Embedded Systems(CHES), 2008.
[60] John Von Neumann. Various techniques used in connection with random digits. John von Neumann, Collected Works, 5:768–770, 1963.
[61] Selman Yakut, Taner Tuncer, and Ahmet Bedri Ozer. Secure and efficient hybrid random number generator based on sponge constructions for cryptographic applications. Elektronika ir Elektrotechnika, pages 41–45, 2019.

HARSHIL BHATIA is currently a final-year computer science undergraduate student at the Indian Institute of Technology (IIT), Jodhpur, India. He is also a research intern at the Visual Computing and Artificial Intelligence Department of the Max Planck Institute for Informatics (MPII), Saarbrücken, Germany. His primary research interests are matching problems on point sets, quantum computer vision and adiabatic quantum computing.

.png

EDITH TRETSCHK is a fourth-year Ph.D. student in the Visual Computing and Artificial Intelligence department at the Max Planck Institute for Informatics in Saarbrücken, Germany. She received her B.Sc. in Computer Science from Saarland University. Her research interests are at the intersection of computer vision, computer graphics, and machine learning. Her focus is on 3D reconstruction problems, especially general non-rigid objects from image or video inputs.

CHRISTIAN THEOBALT is a Professor of Computer Science and the Director of the Visual Computing and Artificial Intelligence Department at the Max Planck Institute for Informatics, Saarbruecken, Germany. He is also a professor at Saarland University. His research lies at the Intersection of Computer Graphics, Computer Vision and Machine Learning. For instance, he works on virtual humans, 3D and 4D scene reconstruction, neural rendering and neural scene representations, marker-less motion and performance capture, machine learning for graphics and vision, and new sensors for 3D acquisition. Christian received several awards, for instance the Otto Hahn Medal of the Max-Planck Society (2007), the EUROGRAPHICS Young Researcher Award (2009), the German Pattern Recognition Award (2012), the EURIGRAPHICS Outstanding Technical Contributions Award (2020), an ERC Starting Grant (2013) and an ERC Consolidator Grant (2017). He is a co-founder of theCaptury (www.thecaptury.com).

VLADISLAV GOLYANIK leads the 4D and Quantum Vision research group at the Visual Computing and Artificial Intelligence Department of the Max Planck Institute for Informatics (MPII), Saarbrücken, Germany. The primary research interests of his team include 3D reconstruction and analysis of deformable scenes, matching problems on point sets and graphs, neural rendering, quantum computer vision and event-based vision. He received a doctoral degree in Informatics from the University of Kaiserslautern in 2019, advised by Didier Stricker. Prior to joining MPII as a post-doctoral researcher, Vladislav was a visiting fellow at NVIDIA (San José, USA), and Institute of Robotics and Industrial Informatics (Barcelona, Spain). He is the recipient of the WACV'16 best paper award and the 2020's yearly dissertation award of the German Association for Pattern Recognition (DAGM).