IEEE Transactions on Quantum Engineering

Received March 19, 2021; revised June 9, 2021; accepted June 23, 2021; date of publication July 14, 2021; date of current version August 12, 2021.

Digital Object Identifier 10.1109/TQE.2021.3094280

Performance of Domain-Wall Encoding for Quantum Annealing

JIE CHEN[1], TOBIAS STOLLENWERK[2], AND NICHOLAS CHANCELLOR[1]

[1]Department of Physics, Joint Quantum Centre Durham-Newcastle, Durham University, Durham DH1 3LE, U.K.
[2]German Aerospace Center, 51147 Cologne, Germany

Corresponding author: Jie Chen. (cencen_cj2015@yahoo.com)

The work of Jie Chen and Nicholas Chancellor was supported by the Engineering and Physical Sciences Research Council, UK Research and Innovation, under Grant EP/S00114X/1, and QPU access for early experiments was supported by impact acceleration funding associated with Grant EP/L022303/1, although none of these data are directly presented in the manuscript. The authors gratefully acknowledge the Jülich Supercomputing Centre for funding this project by providing computing time through the Jülich Unified Infrastructure for Quantum Computing and the D-Wave quantum annealer.

ABSTRACT In this article, we experimentally test the performance of the recently proposed domain-wall encoding of discrete variables Chancellor, 2019, on Ising model flux qubit quantum annealers. We compare this encoding with the traditional one-hot methods and find that they outperform the one-hot encoding for three different problems at different sizes of both the problem and the variables. From these results, we conclude that the domain-wall encoding yields superior performance against a variety of metrics furthermore; we do not find a single metric by which one hot performs better. We even find that a 2000Q quantum annealer with a drastically less connected hardware graph but using the domain-wall encoding can outperform the next-generation Advantage processor if that processor uses one-hot encoding.

INDEX TERMS Analogue computer, optimization methods, quantum computing.

I. INTRODUCTION

Quantum annealing is a subject of much recent interest, because of recent advances in both theory and experimental implementations. After the initial numerical studies, which pointed to quantum annealing as a potential tool for optimization [1], focus was mostly on relatively simple closed systems in the adiabatic limit [2], [3]. However, a wide variety of advances have now taken place, for example, better understanding of the role noise plays [4], more rapid quenches [5]–[8], and how to incorporate quantum annealing into hybrid protocols [9]–[12]. Experimentally, this field is exciting because it allows for large-scale experiments on superconducting hardware designed to solve difficult optimization problems. Proof-of-concept studies have taken place on a diverse range of topics, including aerospace problems [13], [14], hydrology [15], radar waveform design [16], scheduling [17]–[19], and traffic flow optimization [20], [21]. While, to our knowledge, a scaling advantage for optimization has yet to be seen, signs of a potential advantage have been observed in recent quantum simulation experiments [22].

The problems which these devices solve are encoded as energy minimization with respect to quadratic unconstrained binary (QUBO) Hamiltonians (aka. penalty functions) of the form

$$H_{\mathrm{QUBO}} = \sum_{i,j\in\chi} Q_{ij} b_i b_j \tag{1}$$

where $b_i \in \{0, 1\}$ encodes the value of qubit i and Q is the QUBO matrix, which defines the problem.[1] Similarly, one could use the equivalent Ising formulation by substituting $b_i = (1 - z_i)/2$, $z_i \in \{1, -1\}$. The interactions are constrained to a graph χ, but as long as the graph obeys certain structural constraints, arbitrary connectivity can be mapped using a technique known as minor embedding [23], [24], where variables within a graph minor are joined using strong ferromagnetic (negative) interactions. These joined variables are referred to as chains, and the strength of the interactions is referred to as chain strength. An alternative approach to mapping is to use parity constraints [25]–[28], but we use minor embedding methods for this article because they are more commonly used, and because quantum Monte Carlo studies have suggested that this is a better method for the kind of devices we study here [29].

Since solving a QUBO is known to be NP-hard, all other optimization problems can be mapped to them with only a polynomial overhead. One particular mapping, which is

[1]Note that $b_i^2 = b_i$.

common, is a discrete-to-binary mapping, where discrete variables with greater than two values are mapped to binary variables. The traditional way to do this is to use a kind of constraint known as a one-hot constraint, which requires that only one of a set of qubits can be in the $|1\rangle$ configuration.

To construct a quantum algorithm to solve a QUBO problem, quantum annealing is performed; the Hamiltonian that describes this process includes Pauli X terms (X_i), which introduce quantum mechanical (qu) bit flips

$$H(A, B) = -A(t)\sum_i X_i + B(t)H_{\text{QUBO}} \tag{2}$$

where the protocol starts out in an equal positive superposition of all possible solutions with $\frac{A(t=0)}{B(t=0)} \gg 1$ and ends with $\frac{B(t=t_f)}{A(t=t_f)} \gg 1$. Note that the first term, which is called *driver*, can be replaced by other operators, which do not commute with the second operator. While we are not concerned with the detailed physics of how these devices operate in the present study, it is worth remarking that the devices we study here operate in a highly dissipative regime, where interactions with a low-temperature environment play an important role in the dynamics.

Recently, it has been demonstrated in [30] that a different way of encoding discrete variables, known as domain-wall encoding, can lead to problem structures, which make minor embedding more efficient and use fewer variables than one-hot encodings while still allowing arbitrary interactions between the variables. This study was purely numerical and theoretical and, furthermore, has pointed out that the qubit flips explore the solution space in fundamentally different ways for one-hot versus domain-wall encoded problems. The domain-wall encoding has found use in quantum simulations of quantum field theories [31], [32] and has been used in proof-of-concept experiments for using quantum fluctuations to guide searches on annealers [33]. To our knowledge, however, there has never been a direct experimental test of the relative performance between domain-wall and one-hot encodings.

Since the solution space is not explored in the same way for the two encodings, it is not *a priori* clear that the more efficient embedding will translate to improved problem solving abilities. The search could be less effective in a way that negates the gains from improved embedding. However, when we perform the experiments on several examples, we find that the domain-wall encoding does indeed lead to an improvement over many different metrics. These experiments are performed on two different quantum processing units (QPUs) manufactured by D-Wave Systems, Inc., which have different allowed interaction graphs, an older less connected generation (2000Q), and a newer more connected one (Advantage). We find that at least by some metrics, the use of the more sophisticated domain-wall encoding can make more of a difference to the ability to solve problems than the re-engineered hardware graphs. Although not the primary goal of this article, we also compare between the two QPU architectures (and between all QPU–encoding combinations); this allows us to compare the gains from using the domain-wall encoding to those attained by using a more connected architecture.

II. DISCRETE QUADRATIC MODELS

While the native models for quantum annealers are QUBO optimization problems, many real-world problems are most naturally expressed in a way that still involves pairwise interactions between terms; these are referred to as discrete quadratic models (DQMs). A DQM can be described by a set of discrete variables d_i, $i \in [n-1]$, as well as arbitrary pairwise interactions between these variables. Note that the elements in d_i do not need to be integers or even a number; it can be any discrete set, for example, colors in a coloring problem, but are indexed in order by the index i. A DQM Hamiltonian can be written as an extension of a QUBO Hamiltonian with an additional index denoting the variable value

$$H_{\text{DQM}} = \sum_{i,j}\sum_{\alpha,\beta} D_{(i,j,\alpha,\beta)} x_{i,\alpha} x_{j,\beta} \tag{3}$$

where

$$x_{i,\alpha} = \begin{cases} 1, & \text{variable } d_i \text{ takes value } \alpha \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and $D_{(i,j,\alpha,\beta)}$ defines the pairwise interactions between the variables. For the sake of simplicity, we further constrain the values of the discrete variables to consecutive integers $\alpha \in [m-1]$ in this article. An example for such discrete variables is the colors α of vertex i in graph coloring problems. The extension to arbitrary sets of discrete values is straightforward. See [14] as an example.

A. DOMAIN-WALL AND ONE-HOT ENCODINGS

The discrete variable d_i is encoded in multiple binary variables $x_{i\alpha}$. We now quickly review the encoding methods, which are commonly used for these discrete variables. The traditional method is known as one-hot encoding. Here, each qubit corresponds to one possible value of the discrete variable and a constraint, which specifies that the variable only takes one value has to be imposed

$$\forall i \sum_{\alpha} x_{i,\alpha} = 1\,. \tag{5}$$

This constraint can be enforced by adding a quadratic penalty term to the Hamiltonian

$$H_{\text{one hot}} = H_{\text{DQM}} + \lambda \sum_i \left(\sum_{\alpha=0}^{m-1} x_{i,\alpha} - 1\right)^2. \tag{6}$$

Hence, we can write the discrete variables as $d_i = \sum_{\alpha=0}^{m} \alpha x_{i,\alpha}$. However, from the perspective of physically implementation on a real device, it has the undesirable property

IEEE Transactions on Quantum Engineering

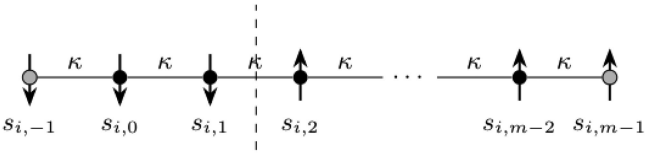

FIG. 1. Domain-wall encoding scheme. The value of the discrete variable d_i is given by the position $\alpha \in [m-1]$ of the domain-wall (indicated by dashed line) in this Ising chain with fixed outer spins $s_{i,-1} = -1$ and $s_{i,m-1} = 1$.

TABLE I Representative Comparison of Domain-Wall and One-Hot Encodings on Four Qubits

value	one-hot	domain-wall	binary
0	N/A	0000	0
1	1000	1000	1
2	0100	1100	10
3	0010	1110	11
4	0001	1111	100

Note that, while convenient in this example, the convention of starting the domain-wall encoding at 0 and one-hot at 1 is not used elsewhere in the article

that all qubits used to encode a variable must be able to interact with all others used to encode that same variables.

It was shown in [30] that instead using a "domain-wall" encoding strategy can encode discrete variables with one fewer qubit per variable and does not require interactions between all qubits to implement the constraint. This article found that minor embedding was more efficient when the domain-wall encoding was used. While a full description of this encoding strategy can be found in [30], and Python code to implement the domain-wall encoding can be found in [34], we review the key details here in the interest of making this manuscript self-contained. The underlying principle of the domain-wall encoding is to use the degeneracy of domain-wall positions on a segment of a frustrated Ising spin chain, as shown in Fig. 1. For each discrete variable d_i, we have $m-1$ spin variables and two fixed spins at the edges of the chain $s_{i,-1} = -1$ and $s_{i,m-1} = 1$. The variable d_i is correctly encoded if there is a single domain wall in the Ising chain. This is enforced by the penalty Hamiltonian describing a ferromagnetic coupling between the Ising spin variables

$$H_{\text{chain}} = -\kappa \left(\sum_{\alpha=-1}^{m-2} s_{i,\alpha} s_{i,\alpha+1} \right) \tag{7}$$

where κ is a coupling large enough to enforce a single domain wall in the ground state (cf. [30]). A concrete example of one-hot and domain-wall encodings ($m = 4$ for one-hot encoding and $m = 4$ for domain-wall encoding) is provided in Table I. Then, the binary variable $x_{i,\alpha}$ depends on the values of the spin variable according to

$$x_{i,\alpha} = \frac{1}{2}\left(s_{i,\alpha} - s_{i,\alpha-1}\right) \quad \forall i, \alpha \in [n-1] \times [m-1] \tag{8}$$

and (5) is fulfilled. Note that only $m-1$ variables (the inner spins) are needed to encode a variable with m different values. This is one less than in the one-hot-encoding case. Hence, the total Hamiltonian for the DQM reads

$$H_{\text{domain wall}} = H_{\text{DQM}} + H_{\text{chain}}. \tag{9}$$

Note that this is a function purely of the $(m-1)n$ (*inner*) spin variables $\{s_{i,\alpha} \mid i \in [n-1], \alpha \in [m-2]\}$. The conversion back to a QUBO is straightforward.

B. BINARY ENCODING

While the main purpose of this article is to compare domain-wall and one-hot encodings, it is also worth comparing to direct binary encodings. Naively, these encodings appear that they should always be more efficient, since to store a variable of size m only requires $\lceil \log_2 m \rceil$ binary variables as opposed to m or $m-1$. However, this does not tell the whole story, as currently available annealing hardware only has linear and quadratic interactions available. It is not necessarily true that the interactions between two binary variables can be written only in terms of these types of interactions; in fact, except for special cases such as objective functions involving multiplication of variables [30], [35], it is not going to be true. This can be rectified by engineering effective higher order interactions [27], [28], but each of these will require at least one auxiliary variable to engineer (although if this variable instantiated as a physical qubit, it may have less stringent requirements than the one used for computation [28]). A fair accounting of the binary variables needed for an encoding would also have to track these auxiliary variables as well.

A lower bound of the number of auxiliary qubits for a given interaction can be obtained by a degree-of-freedom counting argument. For an arbitrary interaction of two variables both of size m, we require m^2 degrees of freedom, to independently assign energies to each configuration. On the other hand, for $2\lceil \log_2 m \rceil$ qubits, there will be $2\lceil \log_2 m \rceil \times (1 + 2\lceil \log_2 m \rceil)$ linear or quadratic degrees of freedom. The total number of higher than quadratic interactions needed will then be the difference between the number of degrees of freedom needed and the total number of linear or quadratic degrees of freedom available along with an additional -1 for the irrelevant energy offset degree of freedom

$$n_{\text{high}}(m) = m^2 - \lceil \log_2 m \rceil (1 + 2\lceil \log_2 m \rceil) - 1. \tag{10}$$

From the previous formula, we first see that for $m = 2$, there are exactly the needed number of linear and quadratic degrees of freedom, which is a good check that this formula is sensible, since higher order terms are not possible in a system involving only two qubits. For $m = 3$, there are more linear or quadratic terms than are necessary to encode all of the degrees of freedom, so auxiliary variables are not necessary for binary encoding. This is unsurprising, because the $m = 3$ domain-wall encoding can be thought of as a special case of binary encoding, where the domain-wall interactions are used to eliminate one of the logical states and encode on the three remaining ones.

However, for $m = 4$, we find that there are five fewer linear or quadratic degrees of freedom than are needed, and therefore, $n_{\text{high}}(4) = 5$, indicating that each interaction between

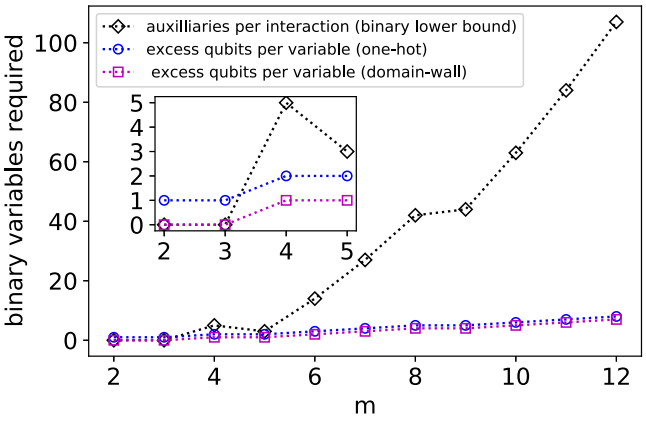

FIG. 2. Number of binary variables requires, per interaction in the binary case, and the number of extra (when compared to binary) per higher variable in the domain-wall and one-hot cases, versus *m*. The inset is the same, but zoomed in to show detail at low values of *m*.

variables of size $m = 4$ in a binary encoding will require at least five auxiliary variable to engineer higher order interactions and, in practice, may need more; for example, the encoding of higher order interactions used in [36] requires l auxiliary variables for an l-order interaction for any case where $l > 3$. For simplicity, we construct a lower bound assuming that each higher order interaction can be engineered using only one auxiliary. For a general problem with q variables of size m and q_{int} interactions between the variables, we, therefore, lower-bound the number of variables, which are needed for the binary encoding as

$$n_{\text{bin}}(m, q, q_{\text{int}}) = q_{\text{int}} n_{\text{high}}(m) + q\lceil \log_2(m) \rceil. \tag{11}$$

While we examine the specific problems used in this study later, but for now, let us consider the general question of when a binary encoding would be superior. Each higher-than-binary interaction will require at least one auxiliary binary variable; additionally, there have to be at least as many interactions as variables for the interaction graph of a problem to be connected. It is, therefore, reasonable to infer that if the number of auxiliary binary variables per interaction is greater than the number of excess binary variables to encode each higher variable when comparing domain-wall or one-hot to binary encoding, then the binary encoding will not be favorable. As shown in Fig. 2, the number of auxiliaries per interaction is always higher, indicating that at least in terms of the number of binary variables required to encode the problem, domain-wall and one-hot encodings will be superior to direct binary encoding. Recall that this analysis only applies to arbitrary interactions and binary encoding may still be more efficient for specific kinds of interactions such as multiplication of numbers; however, we are not aware of any such structure for the coloring problems studied here. This result is counterintuitive, but, somewhat unsurprising given that [30], has previously argued that the domain-wall encoding is maximally efficient for encoding arbitrary interactions, assuming that only linear and quadratic terms are directly accessible.

The present somewhat counterintuitive picture where an encoding that is effectively unary is the most efficient way to represent arbitrary interactions depends entirely on the limitation in the types of coupling available (although we do note that this does meet the requirement of having a robust tensor product structure, which is required for efficient quantum computing [30], [37]). This could change in the future if hardware with native higher order interactions and more exotic driver Hamiltonians (for example, using the strategies of [27], [28], and [38]) became available.

C. k-COLORING

One type of problem we use to contrast the performance of different QPU–encoding combinations is maximum coloring problems. Formally, these are max-k-colorable subgraph [39] problems or equivalently max-k-cut problems [40], [41]. However, since we do not consider any other types of coloring problems in this article, we refer to these problems as k-coloring problems without fear of ambiguity. These problems consist of finding ways to color a graph with q nodes using k colors such that nodes of the same color are in contact with each other in as few places as possible. For graphs that are colorable, this reduces to finding a coloring of the graph. We use randomly generated Erdös–Rényi graphs [42], [43], where for each pair of nodes, the presence or absence of an edge is independently and randomly decided with a given probability.

In the one-hot encoding scheme for the max-k-colorable subgraph problem [39], we have $k = m$ decision variables $x_{i\alpha}$ for each node i in the graph and every color $\alpha \in [m - 1]$, which is 1 if the node i has the color α. The Hamiltonian for graph $G = ([n - 1], E)$ reads

$$H_{\text{DQM}} = \sum_{\alpha=0}^{m-1} \sum_{(i,j)\in E} x_{i\alpha} x_{j\alpha}\,. \tag{12}$$

Following the procedure in [30], we consider two classes of k-coloring problems. For the first, we fix $k = 3$ and vary q; we refer to these as maximum three-coloring problems or simply three-coloring problems. For these problems, we use graphs with an edge probability of 0.5.

The second class of coloring problems we consider are k-coloring problems, where both k and q are varied; with $q = 2k$ and an edge probability of 0.75, we refer to these as maximum k-coloring problems or simply k-coloring problems.

For the maximum k-coloring problems, the bound on the number of qubits needed from (11) will be $n_{\text{bin}}(k, 2\,k, 0.75 \times 4 \times k^2)$. Throwing away nonleading-order terms, we find that, asymptotically, the number of variables required for the binary encoding scale as k^4 as opposed to k^2 for both domain-wall and one-hot encodings. Fig. 3 shows that even for $k = 4$, binary encoding will require an impractically large

IEEE Transactions on Quantum Engineering

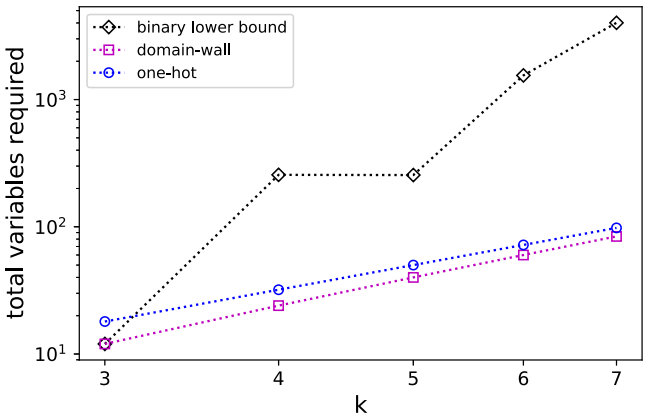

FIG. 3. Average number of binary variables required for domain-wall and binary encodings as well as a lower bound for binary encoding for *k*-coloring problems at sizes relevant to our experiments.

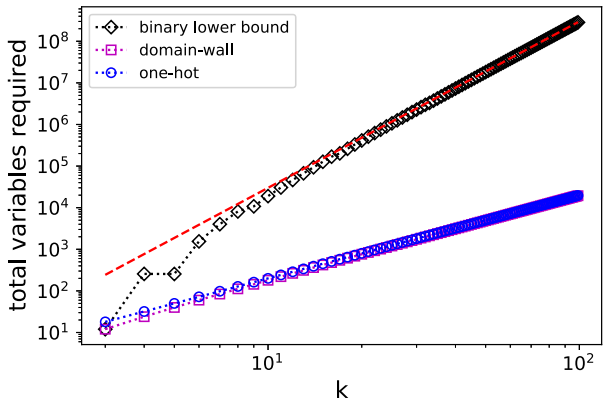

FIG. 4. Average number of binary variables required for domain-wall and binary encodings as well as a lower bound for binary encoding for *k*-coloring problems at sizes useful for showing approach to the asymptotic limit. The dashed line is the leading order term from the binary lower bound, $3k^4$.

number of variables. Recall that for size $k = 3$, the domain-wall encoding is a special case of binary encoding.

We further extend the variable number calculation to higher k. As shown in Fig. 4, the binary encoding reaches a scaling, which strongly resembles the final asymptotic scaling we expect and is far inferior to the domain-wall and even one-hot encoding.

D. FLIGHT GATE ASSIGNMENT

The flight gate assignment problem was already investigated for Quantum Approximate Optimization Algorithm (QAOA) [44] and quantum annealing in the one-hot encoding scheme [13] . We want to assign n flights to m gates using the decision variable

$$x_{i\alpha} = \begin{cases} 1, & \text{if flight } i \text{ is assigned to gate } \alpha \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

for $i \in [n-1]$ and $\alpha \in [m-1]$. The cost function Hamiltonian calculates the total transit time of all passengers at the airport. It reads

$$H_{\mathrm{DQM}} = \sum_{i\alpha} \left(n_i^d t_\alpha^d + n_i^a t_\alpha^a \right) x_{i\alpha} + \sum_{ij\alpha\beta} n_{ij} t_{\alpha\beta}\, x_{i\alpha}\, x_{j\beta} \tag{14}$$

where the various problem parameters are listed in Table II.

TABLE II Flight Gate Assignment Problem Parameters

Symbol	Meaning
n_i^{d}	# of passengers departing with flight i
n_i^{a}	# of passengers arriving with flight i
n_{ij}	# of transfer passengers between flights i and j
t_i^{in}	Arrival time of flight i
t_i^{out}	Departure time of flight i
t_α^{a}	Transfer time from gate α to baggage claim
t_α^{d}	Transfer time from check-in to gate α
$t_{\alpha\beta}$	Transfer time from gate α to gate β
t^{buf}	Buffer time between two flights at the same gate

There are two hard constraints in the problem. First, a flight should be assigned to exactly one gate. This is represented by (5). Second, flights with temporal overlap are not allowed to be assigned to the same gate

$$\forall\alpha, \forall(i,j) \in E, \qquad x_{i,\alpha} \cdot x_{j,\alpha} = 0 \tag{15}$$

where

$$E = \left\{ (i,j) \mid (t_i^{\mathrm{in}} - t_j^{\mathrm{out}} < t^{\mathrm{buf}}) \wedge (t_j^{\mathrm{in}} - t_i^{\mathrm{out}} < t^{\mathrm{buf}}) \right\} \tag{16}$$

is the set of forbidden flight pairs. This second constraint is enforced by adding the following:

$$H_{\mathrm{temp}} = \mu \sum_{\alpha} \sum_{(i,j)\in E} x_{i\alpha} x_{j\alpha} \tag{17}$$

to the total Hamiltonians (6) and (7), respectively. Again, the penalty weight μ must be sufficiently large to ensure the constraint satisfaction in the ground state (see [13] for details). Note that both constraints are equivalent to the proper coloring of a graph with edged E. This relation to graph coloring is extensively discussed, e.g., in [44].

Due to precision problems of the D-Wave quantum annealer, in [13], it was found that the largest number of instances with nonvanishing success probability was 29 with $n = 7$ flights and $m = 2$. We used these instances for our study. Given that no variable in the flight gate assignment problems studied here is larger than $m = 3$, all domain-wall encodings used for these problems are also special cases of binary encoding.

III. EXPERIMENTAL SETUP AND PERFORMANCE MEASURES

In this section, we discuss the experimental setup and measures for comparing the performance of domain-wall and one-hot encodings.

A. CHAIN AND CONSTRAINT STRENGTHS

To experimentally study these problems, we need to decide upon a method to choose the strength of both embedding chains and the constraints used to restrict the number of

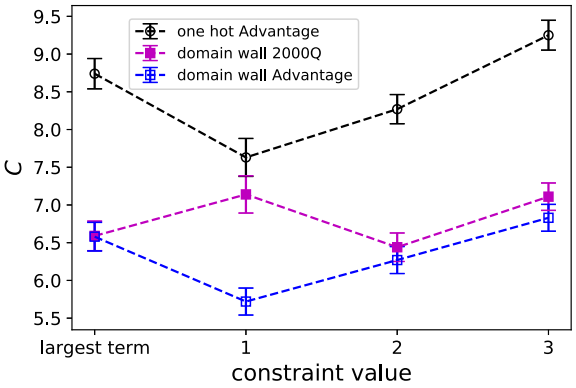

FIG. 5. **Cost function *C*, which is equal to the total number of places with the same color touch for the three-coloring problem with 15 nodes, averaged over 100 instances versus the constraint strength.**

domain walls to one or enforce the one-hot constraint [e.g., λ, μ, and κ in (6), (7), and (17)]. Since the goal of this article is to compare rather than develop an absolute benchmark, it is not crucial that these choices be completely optimal, but for these comparisons to be relevant to real calculations, we should still ensure that we are operating in a regime where the behavior is likely to be similar to the optimal choices.

For the chain strength, we use the "uniform torque compensation" [45] feature, which is available through the D-Wave ocean software repository [46]. On the other hand, no such feature exists for finding the strength of the constraints. However, a practical approach is to choose the strength of the constraint parameters λ, κ, and μ equal to the magnitude of the largest single field or coupler in the problem definition before embedding. Both of these choices have the advantage of being "automatic," the sense of adjusting based on the structure of the problem at hand. Since the goal of this article is to make a comparison of performance on the same footing, rather than to test the optimal performance of the device, we do not need to show that our parameter choices are the best possible, but that they perform reasonably well. To verify that these choices are sensible, we compare different constraint choices for a size-15 three-coloring problem in Fig. 5. We find that our strategy for choosing the constraint strength is roughly optimal for 2000Q; it is slightly suboptimal for Advantage. While this performs well enough for the purposes of the analysis we do here, it is worth keeping in mind that there is potential room for further gains within our Advantage data by finding a better performing parameter setting heuristic.

B. EXPERIMENTAL PROCEDURES

All experiments reported here were performed in the autumn of 2020. They were performed using the default anneal time of 5 μs, and 100 anneals were performed in each run using a single embedding (we found that the performance variation between different embeddings was negligible). All embeddings were performed using the minorminer embedding software provided by D-Wave systems, Inc. [47]. If an embedding failed, we attempt it twice more (for a total of three attempts) to verify that this failure was not an anomaly. We found that for all problem classes (for example, particular size of graph for three coloring or number of colors for k-coloring), for any given QPU/embedding combination, either all problem embeddings failed or they all succeeded. Except when explicitly stated otherwise in this article, broken chain decoding was performed using the majority vote decoding tool included in the software package. Spin reversal transforms (sometimes referred to as gauge averaging) were not used. Experimental data are available in a public repository [48].

Numerical analysis and plotting were performed using MATLAB and the python programming language [49]; in particular, heavy use was made of the numpy [50], [51] and matplotlib [52] packages, as well as jupyter notebooks [53], [54].

C. HYPOTHESIS TESTING

One method to compare between pairs of QPU–encoding combinations is to run them on the same problems and see how many times each can outperform the others, ignoring cases where they can each find equally good solutions. The immediate question then becomes how statistically significant a given sample is. In other words, how likely are we to see a result, which is at least as favorable by random chance? To quantify this significance, we perform hypothesis testing.

To this end, we partition the solutions, where one QPU–encoding combination outperforms the other into the two classes. First, consider the number of cases where the combination we expect to do better (the "expected winner") has performed better than the combination that is being compared to n_b. Second, consider the number of cases where the "expected winner" performs worse n_w. This allows us to calculate the statistical significance

$$p = \frac{1}{2^{n_b+n_w}} \sum_{k=n_w}^{n_b+n_w} \binom{n_b + n_w}{k}. \tag{18}$$

This is effectively the probability that the expected winner could perform better at least as many times if the better performing QPU–encoding combination were chosen at random with 50% probability (our null hypothesis). Effectively, it is the probability that the result (or an even more favorable one) could happen by chance if both performed equally well.

By convention, $p < 0.05$ is considered to be a statistically significant result rejecting the null hypothesis [55] and, therefore, confirming that the expected winner does indeed perform better; by symmetry, $p > 0.95$ is also a statistically significant result, but rejecting an alternate hypothesis that the expected winner was chosen correctly and, therefore,

IEEE Transactions on Quantum Engineering

showing that we chose the expected winner incorrectly. A value of $0.05 < p < 0.95$ is not statistically significant and indicates that an insufficient number of samples have been taken to draw any conclusions based on our hypothesis testing strategy.

D. PERFORMANCE MEASURES

To understand the effect of encoding and QPU choice on the ability of the device to solve the problem, we choose to analyze four performance metrics. The first is the fraction of raw solutions, which do not contain any broken embedding chains, R_{chain}. This measure allows us to get a sense of how faithfully the device is able to represent the problem, since solutions with broken chains no longer correspond to valid solutions.

The second quantity we examine is the rate of solutions, where, after majority vote decoding is performed on any broken chains, the solutions satisfy all of the one-hot or domain-wall constraints. We call this rate R_{enc}.

We next define the cost function C, which is the quantity which we are attempting to minimize subject to our constraints. By convention, if an annealing run has not returned any solutions where all the constraints are satisfied, then we define the effective value of C to be infinite, since no valid solution is returned; therefore, as we have defined it, the average cost function is infinite even if a single problem does not yield a valid solution.

Finally, we define the success probability P, which is defined as the fraction of problems for which the optimal value of C was found. We only report this measure for problem sizes, which are small enough that the optimal solution can be definitively found using exhaustive search. Recall that we only perform 100 anneals per run, so it is likely that higher success probabilities could be found by increased sampling, potentially also using spin reversal transforms, or even more advanced tricks such as reverse annealing [9], [11], [56], pausing [57], sample persistence [58], anneal offsets [59], or extended coupling range [60]. The goal of this study is a relative comparison between QPUs and encoding methods rather than to benchmark the best possible performance, which can be attained using these devices, and for this reason, we have decided to keep our experiments simple at the expense of cutting-edge performance.

In this section, we present our results by showing the four performance measures for the flight gate assignment problem, the three-coloring problem, and the k-coloring problem. Also, we show our results on the hypothesis testing for the three-coloring and k-coloring problems.

E. FLIGHT GATE ASSIGNMENT

As discussed in Section II-D, all studied flight gate assignment instances have $m = 2$. Since the encoding of a discrete variable of size 2 into a domain-wall encoding reduces to a direct binary encoding, it is not mathematically possible for the domain-wall constraint to be violated in these cases. On the other hand, we find that the one-hot constraint is only satisfied in 71% and 65% of solutions on average for 2000Q and Advantage, respectively. We do find some chain breaks in all cases, but they are so rare that it is not possible to reliably differentiate which encoding performs better based on our data, although we have observed that the domain-wall encoding seems to perform slightly better. We observe that the problem is solved after 100 reads in all cases except for three using the one-hot encoding on Advantage.

To study what effect chain breaks would have on larger systems, we set embedding chain strengths to intentionally suboptimal values as opposed to using the uniform torque compensation tool, which performs well in all cases. The results are shown in Fig. 6. We find that the domain-wall encoding performs equally or better than one-hot against all metrics, regardless of the QPU used. What is particularly striking is that the domain-wall encoding was able to solve all instances at all chain strengths for both processors, while the one-hot encoding was only able to achieve this for the uniform torque compensation scheme.

F. THREE COLORING

For further demonstration of the effect of the different encodings, we consider 100 random instances of three color problems, as studied in [30]. For these problems, each variable d_i has three possible values, corresponding to each color, since the domain-wall encoding consists of two qubits in this case; it is mathematically possible for the single domain-wall constraint to not be satisfied. For each of these instances, we run both domain-wall and one-hot encodings each on the Advantage and 2000Q QPU.

Fig. 7 shows the four performance measures, as discussed in Section III-D. For the same QPU, the domain-wall encoding performs consistently better than one hot. Also, for the same encoding, the Advantage QPU performs consistently better than the 2000Q QPU. When comparing the domain-wall encoding on the 2000Q QPU with one-hot encoding on the Advantage QPU, we can show that the average solution quality appears either to be comparable or to favor the domain wall on the less advanced processor. On the other hand, even using the one-hot encoding, the Advantage processor has fewer chain breaks. In essence, using a more connected QPU and using domain wall as opposed to one hot seem to reduce chain breaks, and furthermore, the higher connectivity seems to be the more decisive factor.

However, it seems to be the case that (at least for majority vote decoding) broken chains in the domain-wall encoding are more likely to be decoded to correct solutions. For the cost function as well, there is a visible difference between the performance of both the one-hot and domain-wall encodings. Finally, we observe that, while the domain-wall encoding makes a large difference in the probability of finding the optimal solution within 100 reads, the difference between the two types of QPUs is within error bars for all sizes we test. Aside from the fact that larger problems can be

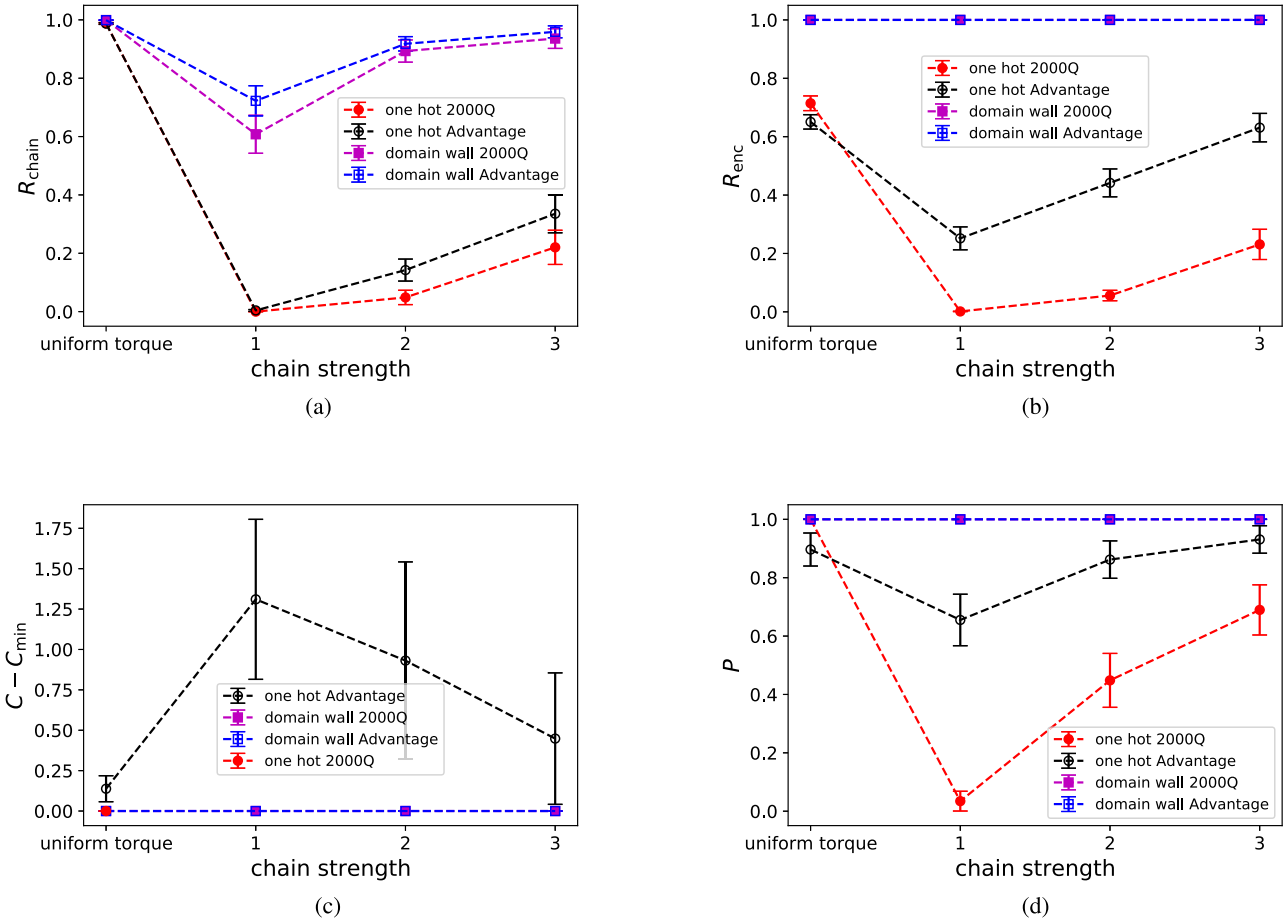

FIG. 6. **Mean of the four performance measures against the chain strength for the 29 instances of the flight gate assignment problem for all four QPU–encoding combinations. Note that the cost function is normalized by subtracting the optimal value C_{min} for more meaningful comparison across problems. In part (c), the one-hot 2000Q point only appears for uniform torque compensation since the data for all other chain strengths contained at least one problem, where none of the 100 reads produced a solution, which satisfied all one-hot constraints. All bars in this plot are standard error. (a) Rate of unbroken chains in physical solutions. (b) Rate of correctly encoded logical solutions. (c) Cost function of the problem from logical solutions. (d) Success probability.**

encoded on the Advantage QPU, there is not a significant difference between the performance of the two QPUs for the same encoding for graph sizes less than about 25; however, as we demonstrate later, the performance differences can be better understood using analysis based on hypothesis testing.

Since it is difficult to visually distinguish the average cost functions, we apply a different technique to compare performance of solutions against the cost function. We examine how many cases each processor–encoding combination does better or worse than any of the others. We further perform hypothesis testing to see which of the differences are statistically significant, as described in Section III-C. The results are shown in Table III. We can see that except for at the smallest size (five nodes), there are differences in how the processor–encoding combinations perform. We find that except for at this smallest size, the domain-wall encoding always performs better than the one-hot encoding, even when comparing the domain-wall encoding on a 2000Q to one-hot encoding on an Advantage. We further find that all statistically significant results point toward Advantage performing better than 2000Q, but at smaller sizes, the differences are not statistically significant.

A particularly striking result here is that, at least up to the size where the problems can no longer be embedded on the 2000Q, using a domain-wall encoding rather than one-hot encoding makes a bigger difference to solution quality than using the more advanced processor. This underscores the importance of encoding methods to obtaining high-quality solutions over just waiting for hardware improvements. An astute reader may question whether this result is simply because the majority vote decoding seems to perform better on domain-wall encoded problems than one-hot [as can be seen by comparing Fig. 7(a) to (b)]. To answer this question, we consider an alternate way of processing the data, in which solutions with broken chains are discarded rather than decoded by majority vote. We find that this approach does not affect the qualitative result that the domain-wall encoding on

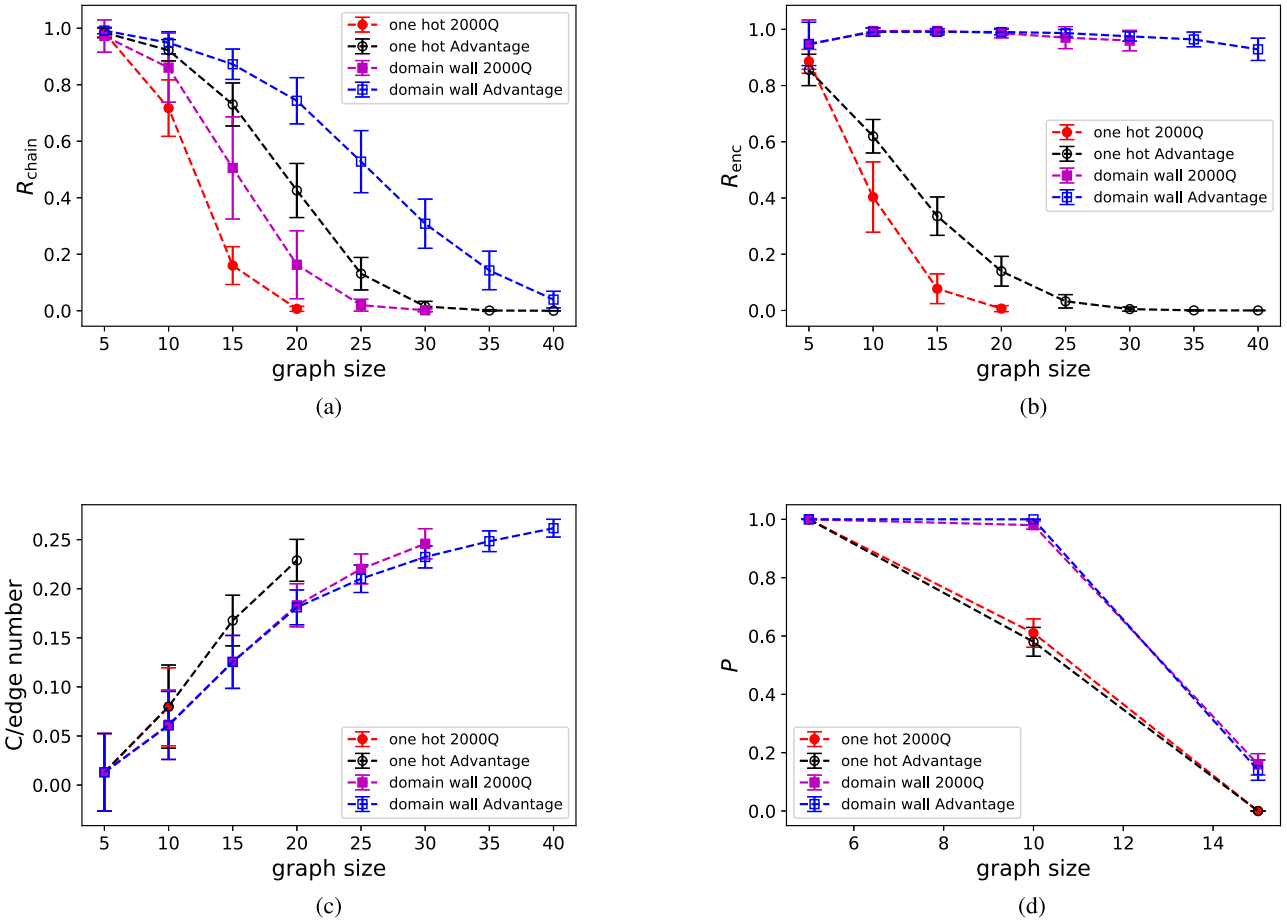

FIG. 7. **Mean of the four performance measures against the problem size for the three-coloring problem for all four QPU–encoding combinations. The cost function in part (c) has been normalized by edge number to allow a more direct comparison at different sizes. The bars for parts (a), (b), and (c) are the standard deviation of the distribution, rather than the standard error. Since all data points are based on 100 samples, the standard error is ten times smaller than what is depicted by these bars. Bars for (d) are standard error. (a) Rate of unbroken chains in physical solutions. (b) Rate of correctly encoded logical solutions. (c) Cost function of the problem from logical solutions. (d) Success probability.**

TABLE III **Hypothesis Testing Results for All Six Possible Comparisons of QPU–Encoding Combinations for Three Color Problems of Different Sizes**

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
5 node (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
5 node p												
10 node (b,w)	42	0	37	0	2	0	19	21	39	0	40	0
10 node p	2.27×10^{-13}		7.28×10^{-12}		2.50×10^{-1}		6.82×10^{-1}		1.82×10^{-12}		9.09×10^{-13}	
15 node (b,w)	85	2	95	3	32	34	70	22	94	1	91	2
15 node p	2.47×10^{-23}		4.95×10^{-25}		6.44×10^{-1}		2.67×10^{-7}		2.42×10^{-27}		4.41×10^{-25}	
20 node (b,w)	99	0	100	0	43	41	94	3	100	0	93	2
20 node p	1.58×10^{-30}		7.89×10^{-31}		4.57×10^{-1}		9.60×10^{-25}		7.89×10^{-31}		1.15×10^{-25}	
25 node (b,w)	100	0		FAIL	66	20		FAIL		FAIL	98	2
25 node p	7.89×10^{-31}				3.33×10^{-7}						3.98×10^{-27}	
30 node (b,w)	100	0		FAIL	72	20		FAIL		FAIL	97	2
30 node p	7.89×10^{-31}				2.30×10^{-8}						7.81×10^{-27}	
35 node (b,w)	100	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
35 node p	7.89×10^{-31}											
40 node(b,w)	100	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
40 node p	7.89×10^{-31}											

For each comparison, the expected winner is listed first. For each size, the count of cases where the expected winner (written first at the top of the column) performs better n_b (left) and worse n_w (right) is listed. Below is listed the value of p as calculated by (18). In cases where either both combinations perform the same on all problems or one or both fail to embed, statistical significance cannot be calculated. In case where the expected winner failed to embed, we write "FAIL" in the left column, and likewise if the embedding fails for the QPU–encoding combination described by the right column. These comparisons are performed for the single best solution found out of all 100 samples, using majority vote decoding for broken chains. If none of the samples decode to valid solutions, then the cost function is treated as being "infinite" and any finite value is considered to be better. Color coding used as a guide to the eye: green indicates a statistically significant rejection of the null hypothesis, while yellow indicates a result that is not statistically significant.

TABLE IV Hypothesis Testing Results for All Six Possible Comparisons of QPU–Encoding Combinations for *k*-Color Problems With Different Numbers of Colors

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
3 color (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
3 color p												
4 color (b,w)	34	1	37	2	11	3	26	16	44	1	33	7
4 color p	1.05×10^{-9}		1.42×10^{-9}		2.87×10^{-2}		8.21×10^{-2}		1.31×10^{-12}		2.11×10^{-5}	
5 color (b,w)	91	1	78	1	34	18	23	59	88	1	91	1
5 color p	1.88×10^{-26}		1.32×10^{-22}		1.82×10^{-2}		≈ 1		1.45×10^{-25}		1.88×10^{-26}	
6 color(b,w)	99	0		FAIL	59	15		FAIL		FAIL	99	0
6 color p	1.58×10^{-30}				1.28×10^{-7}						1.58×10^{-30}	
7 color(b,w)	92	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
7 color p	2.02×10^{-28}											

For each comparison, the expected winner is listed first. For each size, the count of cases where the expected winner (written first at the top of the column) performs better n_b (left) and worse n_w (right) is listed. Below is listed the value of p as calculated by (18). In cases where either both combinations perform the same on all problems, or one or both fail to embed, statistical significance cannot be calculated. In case where the expected winner failed to embed, we write "FAIL" in the left column, and likewise if the embedding fails for the QPU–encoding combination described by the right column. These comparisons are performed for the single best solution found out of all 100 samples, using majority vote decoding for broken chains. If none of the samples decode to valid solutions, then the cost function is treated as being "infinite" and any finite value is considered to be better. Color coding used as a guide to the eye: green indicates a statistically significant rejection of the null hypothesis, while yellow indicates a result that is not statistically significant. Red indicates a statistically significant result that rejects the alternate hypothesis.

TABLE V Hypothesis Testing Results for All Six Possible Comparisons of QPU–Encoding Combinations for Three Color Problems of Different Sizes

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
5 node (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
5 node p												
10 node (b,w)	42	0	38	0	2	0	22	21	40	0	40	0
10 node p	2.27×10^{-13}		3.64×10^{-12}		2.50×10^{-1}		5.00×10^{-1}		9.09×10^{-13}		9.09×10^{-13}	
15 node (b,w)	88	2	96	3	35	28	78	16	96	0	91	3
15 node p	3.31×10^{-24}		2.55×10^{-25}		2.25×10^{-1}		2.89×10^{-11}		1.26×10^{-29}		6.99×10^{-24}	
20 node (b,w)	99	1	96	0	60	33	98	0	100	0	82	15
20 node p	7.97×10^{-29}		1.26×10^{-29}		3.35×10^{-3}		3.16×10^{-30}		7.89×10^{-31}		1.19×10^{-12}	
25 node (b,w)	100	0		FAIL	93	6		FAIL		FAIL	61	24
25 node p	7.89×10^{-31}				1.89×10^{-21}						3.70×10^{-5}	
30 node (b,w)	100	0		FAIL	100	0		FAIL		FAIL	14	3
30 node p	7.89×10^{-31}				7.89×10^{-31}						6.36×10^{-3}	
35 node (b,w)	100	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
35 node p	7.89×10^{-31}											
40 node(b,w)	88	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
40 node p	3.23×10^{-27}											

For each comparison, the expected winner is listed first. For each size, the count of cases where the expected winner performs better n_b (left) and worse n_w (right) are listed. Below is listed the value of p as calculated by (18). In cases where either both combinations perform the same on all problems or one or both fail to embed, statistical significance cannot be calculated. In case where the expected winner failed to embed, we write "FAIL" in the left column, and likewise if the embedding fails for the QPU–encoding combination described by the right column. These comparisons are performed for the single best solution found out of all 100 samples, where samples with broken chains are treated as being invalid. If none of the samples decode to valid solutions, then the cost function is treated as being "infinite" and any finite value is considered to be better. Color coding used as a guide to the eye: green indicates a statistically significant rejection of the null hypothesis, while yellow indicates a result that is not statistically significant.

a 2000Q performs better in a statistically significant way. For completeness, these results are shown in Table V in the Appendix. While the domain-wall encoding does always make a bigger performance difference in cases where the problem can be embedded on both QPUs, it is possible to embed larger problems on the Advantage QPU; even with the domain-wall encoding, 30 nodes is the largest size we are able to embed on an 2000Q, whereas Advantage can embed a problem of at least size 40.

G. *k*-COLORING

Now that we have demonstrated that the domain-wall encoding leads to significantly better performance in encoding discrete variables with both two and three possible values, the next natural question is what happens at higher values, particularly because Chancellor [30] found that these were the cases where the structure had the most effect on embedding efficiency. To do this, we examine maximum k-coloring problems, for which the value of k scales with the number of nodes. Fig. 8 shows the results. We see a similar pattern as before, with Fig. 8(a) showing that the QPU structure makes a bigger difference in terms of chain breaks; however, at least for the Advantage QPU, usage of the domain-wall encoding can also significantly reduce the number of breaks. Also, as seen in the three-coloring case, the encoding type is the dominant factor in determining the number of solutions, which are decoded correctly. Unlike the three-coloring case however, R_{enc} decreases toward zero as the number of colors and, therefore, the problem size are increased. This is likely because for more colors, there are more possible ways to violate the constraint. The final cost function likewise shows encoding being the dominant factor in determining performance and being a more significant factor than QPU type. We further see this trend in probability of finding the most optimal solution, although the performance difference between the Advantage and 2000Q by this metric is larger than

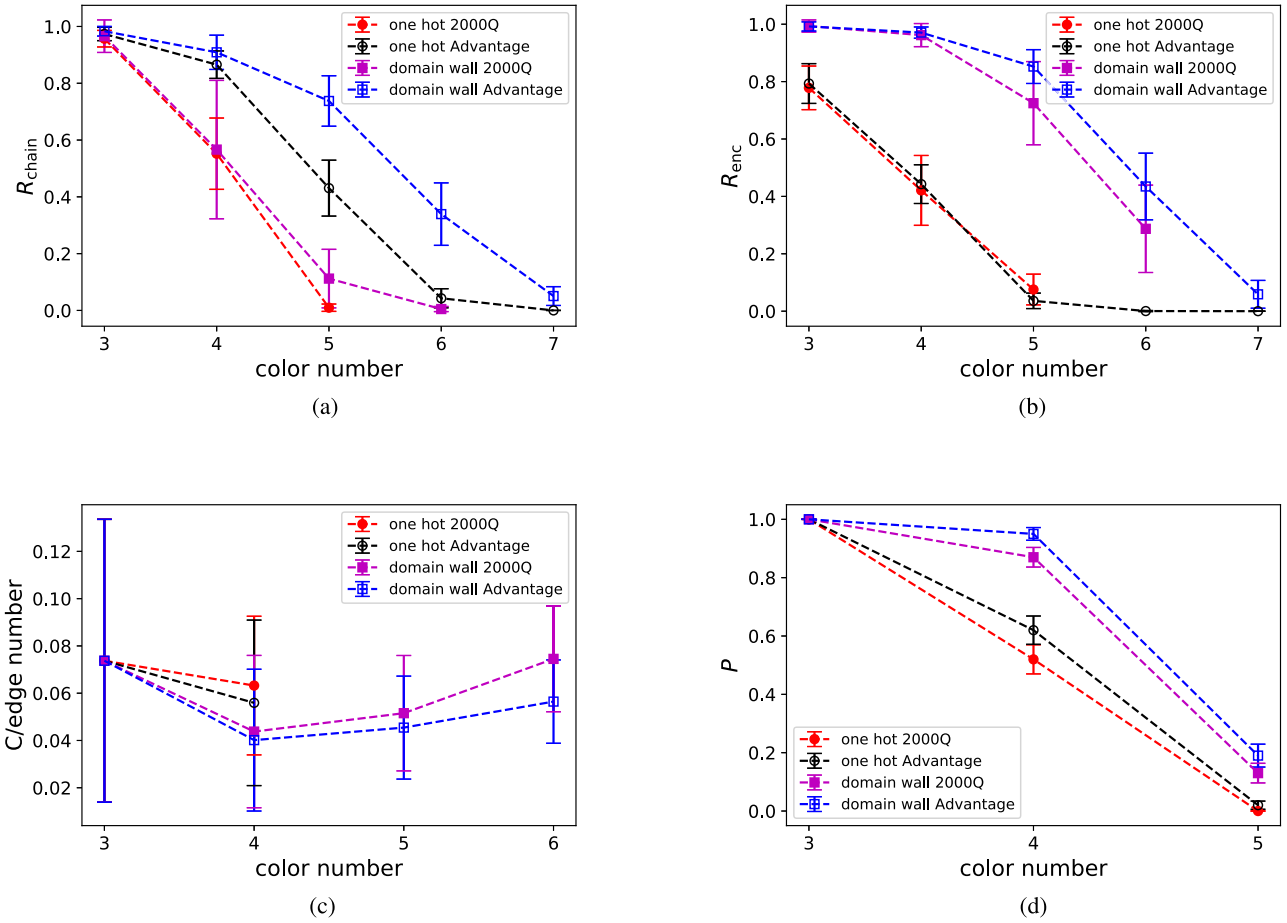

FIG. 8. Mean of the four performance measures against the problem size for the *k*-coloring problem for various numbers of colors and comparing all four QPU–encoding combinations. The cost function in part (c) has been normalized by edge number to allow a more direct comparison at different sizes. The bars for parts (a)–(c) are the standard deviation of the distribution, rather than the standard error. Since all data points are based on 100 samples, the standard error is ten times smaller than what is depicted by these bars. Bars for (d) are standard error. Note that the *x*-axis values are not the same on each graph; this is due to the fact that for subfigure (c), no QPU–encoding combination returned valid solutions for all problems, and for subfigure (d), our exhaustive method of finding solutions failed beyond size 5, and we felt the trend was clear enough from these points. (a) Rate of unbroken chains in physical solutions. (b) Rate of correctly encoded logical solutions. (c) Cost function of the problem from logical solutions. (d) Success probability.

the three-coloring case, suggesting that the slightly better performance from Advantage is a real effect; as we will see later, a different method of comparison actually favors the older version of the processor, the 2000Q.

H. HYPOTHESIS TESTING FOR k-COLORING

As we have done before, we perform hypothesis-testing-based analysis on the best cost function values returned by all QPU–encoding combinations. As with the three-coloring case, the domain-wall encoding leads to a statistically significant improvement at all but the smallest size. Even when we perform a cross comparison, we see that the domain-wall encoding on a 2000Q outperforms the one-hot encoding on the Advantage. We do find one surprising result, not in terms of comparison between the two encodings, but in terms of comparisons between the two QPUs. For the one-hot encoding of the maximum five-coloring problem, we find a highly statistically significant result that the 2000Q actually outperforms the Advantage processor. This is the opposite trend to what is seen when the domain-wall encoding is used and very unusual since a more highly connected device should perform better than a less connected one. To understand the root cause of this effect, we perform the same analysis, but discard broken chain solutions, rather than performing majority vote decoding. As shown in Table VI, the effect goes away when we change the decoding strategy, indicating that this is an artifact of the strategy. Since the primary purpose of this article is to compare the encoding strategy, rather than QPU performance, we have elected not to probe this effect further, although doing so could potentially yield interesting results.

IV. DISCUSSION AND CONCLUSION

We have performed the first (to our knowledge) experimental tests of the domain-wall encoding proposed in [30] on quantum annealing processors. We find that for problems with

TABLE VI Hypothesis Testing Results for All Six Possible Comparisons of QPU–Encoding Combinations for *k*-Color Problems of Different Number of Colors

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
3 color (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
3 color p												
4 color (b,w)	39	1	42	3	17	2	33	14	53	1	32	9
4 color p	3.73×10^{-11}		4.33×10^{-10}		3.64×10^{-4}		3.97×10^{-3}		3.05×10^{-15}		2.15×10^{-4}	
5 color (b,w)	97	1	89	2	66	7	70	7	100	0	80	9
5 color p	3.12×10^{-28}		1.69×10^{-24}		1.92×10^{-13}		1.76×10^{-14}		7.89×10^{-31}		1.15×10^{-15}	
6 color(b,w)	100	0		FAIL	98	0		FAIL		FAIL	26	0
6 color p	7.89×10^{-31}				3.16×10^{-30}						1.49×10^{-8}	
7 color(b,w)	38	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
7 color p	3.64×10^{-12}											

For each comparison, the expected winner is listed first. For each size, the count of cases where the expected winner performs better n_b (left) and worse n_w (right) are listed. Below is listed the value of p as calculated by (18). In cases where either both combinations perform the same on all problems or one or both fail to embed, statistical significance cannot be calculated. In case where the expected winner failed to embed, we write "FAIL" in the left column, and likewise if the embedding fails for the QPU–encoding combination described by the right column. These comparisons are performed for the single best solution found out of all 100 samples, where samples with broken chains are treated as being invalid. If none of the samples decode to valid solutions, then the cost function is treated as being "infinite" and any finite value is considered to be better. Color coding used as a guide to the eye: green indicates a statistically significant rejection of the null hypothesis.

variables up to size 7, the domain-wall encoding outperforms the traditional one-hot encoding in all but the smallest cases, for which their performance is roughly equal due to the ease of the problems. We further find that the domain-wall encoding generally reduces the number of broken minor embedding chains and increases the proportion of solutions, which decode correctly. Crucially, for every problem we look at, we do not find a single metric for which the domain-wall encoding performs worse on average than the one-hot encoding on the same QPU, suggesting that the domain-wall encoding should be the method of choice.

Dramatically, when we perform a cross comparison of performance between domain-wall encoding on the older 2000Q QPU versus one-hot encoding on the newer Advantage QPU, we find that the encoding makes a bigger difference in solution quality than the QPU architecture (at least on problems that can be embedded into both QPUs; embedding fails at a smaller size for domain wall on a 2000Q than it does for one hot on Advantage). This underscores the importance of "software" advances like better encoding in parallel to hardware advances. This is particularly true given that, while developing new hardware can be a very expensive endeavor, using a different encoding can be done at almost no cost. Given the recent trend for the programming of these devices to be done at an increasingly higher level, the end user does not necessarily need to even "see" the encoding steps at all, for example, the D-Wave ocean repository currently has a DQM solver [61], which uses one-hot encoding to solve optimization problems [62]. Changing the underlying encoding used by this solver is likely to improve performance, but would have no other effect on the way the end users interact with this solver and would not require them to understand how this strategy works. The German Aerospace Center is developing a library, which is more hardware agnostic (including gate-based quantum computers) with overlapping functionality. It is planned to integrate different encodings, including domain-wall encoding, and publish the software under an open-source license.

Although not the focus of the present study, it would be very illuminating to examine the underlying cause of the drop in success probability as size increases. In particular, it has been observed, for example, in [63] that even for an isolated domain-wall chain without a programmed potential, the probability of finding a domain wall at different locations is far from uniform, especially if spin reversal transforms are not performed. In that paper, the underlying cause was analog noise on the device causing biases toward some configurations; these biases may have been enhanced by the fact that the dynamics of 1-D chains tend to freeze very late in the anneal [64].

While it being not the main purpose of our study, we have also compared the Advantage and 2000Q QPUs. We have found that minor embedding chains break less frequently on the Advantage QPU, as what should be expected on a more connected graph. We found that the Advantage QPU also performed better (or for small problems no statistically significant difference could be found) at solving problems in all but one example on a five-coloring problem. This result is highly statistically significant, so it is unlikely to be a statistical anomaly. It also goes away when we do not perform majority vote decoding on the broken chains. While we have not investigated this effect further since it is far from our main purpose, it is likely that a more complete investigation could be fruitful in finding improved broken chain decoding strategies. It is also worth remembering that there was room for improvement in how the constraint strength was chosen for Advantage; a more optimal strategy here would change the comparison between Advantage and 2000Q (further) in favor of Advantage.

While our work gives compelling evidence that the domain-wall strategy is superior for currently available superconducting flux qubit QPUs, there are still many unanswered questions with regard to the encoding, which are beyond the scope of our current work. It would be illuminating to test these strategies on larger problems and more connected hardware graphs using quantum Monte Carlo.

This approach has previously been used to compare embedding strategies [29]. It could further be interesting to test whether the advantages seen in annealing carry over to gate model optimization algorithms. Preliminary work [30] gives some theoretical suggestions for this because our domain-wall encoding approach will require fewer interactions between distant qubits. However, an experimental test would be enlightening.

In this Appendix, we provide versions of Tables III and IV, but where solutions with broken chains are discarded rather than decoded by majority vote, the results appear in Tables V and VI, respectively.

REFERENCES

[1] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov. 1998, doi: 10.1103/PhysRevE.58.5355

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

[3] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Rev. Modern Phys.*, vol. 90, no. 1, Jan. 2018, Art. no. 015002, doi: 10.1103/RevModPhys.90.015002

[4] L. C. Venuti, T. Albash, D. A. Lidar, and P. Zanardi, "Adiabaticity in open quantum systems," *Phys. Rev. A*, vol. 93, no. 3, Mar. 2016, Art. no. 032118, doi: 10.1103/PhysRevA.93.032118

[5] J. G. Morley, N. Chancellor, S. Bose, and V. Kendon, "Quantum search with hybrid adiabatic-quantum-walk algorithms and realistic noise," *Phys. Rev. A*, vol. 99, no. 2, Feb. 2019, Art. no. 022339, doi: 10.1103/PhysRevA.99.022339

[6] M. B. Hastings, "Duality in quantum quenches and classical approximation algorithms: Pretty good or very bad," *Quantum,* vol. 3, Nov. 2019, Art. no. 201, doi: 10.22331/q-2019-11-11-201

[7] A. Callison, M. Festenstein, J. Chen, L. Nita, V. Kendon, and N. Chancellor, "An energetic perspective on rapid quenches in quantum annealing," 2020, *arXiv:2007.11599.*

[8] E. J. Crosson and D. A. Lidar, "Prospects for quantum enhancement with diabatic quantum annealing," 2020, *arXiv:2008.09913.*

[9] A. Perdomo-Ortiz, S. E. Venegas-Andraca, and A. Aspuru-Guzik, "A study of heuristic guesses for adiabatic quantum computation," *Quantum Inf. Process.*, vol. 10, no. 1, pp. 33–52, Feb. 2011, doi: 10.1007/s11128-010-0168-z

[10] Q.-H. Duan, S. Zhang, W. Wu, and P.-X. Chen, "An alternative approach to construct the initial hamiltonian of the adiabatic quantum computation," *Chin. Phys. Lett.*, vol. 30, no. 1, 2013, Art. no. 010302, doi: 10.1088/0256-307x/30/1/010302

[11] N. Chancellor, "Modernizing quantum annealing using local searches," *New J. Phys.*, vol. 19, no. 2, 2017, Art. no. 023024, doi: 10.1088/1367-2630/aa59c4

[12] T. Graß, "Quantum annealing with longitudinal bias fields," *Phys. Rev. Lett.*, vol. 123, Sep. 2019, Art. no. 120501, doi: 10.1103/PhysRevLett.123.120501

[13] T. Stollenwerk, E. Lobe, and M. Jung, "Flight gate assignment with a quantum annealer," in *Proc. 1st Int. Workshop Quantum Technol. Optim. Problems*, Mar. 2019, pp. 99–110, doi: 10.1007/978-3-030-14082-3_9

[14] T. Stollenwerk *et al.*, "Quantum annealing applied to de-conflicting optimal trajectories for air traffic management," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 285–297, Jan. 2020, doi: 10.1109/TITS.2019.2891235

[15] D. O'Malley, "An approach to quantum-computational hydrologic inverse analysis," *Sci. Rep.*, vol. 8, no. 1, 2018, Art. no. 6919, doi: 10.1038/s41598-018-25206-0

[16] G. E. Coxson, C. R. Hill, and J. C. Russo, "Adiabatic quantum computing for finding low-peak-sidelobe codes," *Proc. IEEE High Perform. Extreme Comput. Conf.*, 2014, pp. 1–6, doi: 10.1109/HPEC.2014.7040953

[17] A. Crispin and A. Syrichas, "Quantum annealing algorithm for vehicle scheduling," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3523–3528, doi: 10.1109/SMC.2013.601

[18] D. Venturelli, D. J. J. Marchand, and G. Rojo, "Quantum annealing implementation of job-shop scheduling," 2000, *arXiv:1506.08479.*

[19] T. T. Tran *et al.*, "A hybrid quantum-classical approach to solving scheduling problems," in *Proc. 9th Annu. Symp. Combinatorial Search*, 2016, pp. 98–106.

[20] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Front. ICT*, vol. 4, 2017, Art. no. 29, doi: 10.3389/fict.2017.00029

[21] S. Yarkoni *et al.*, "Quantum Shuttle: Traffic Navigation With Quantum Computing," in *Proc. 1st ACM SIGSOFT Int. Workshop Architectures and Paradigms for Eng. Quantum Softw. (APEQS 2020)*, 2020, pp. 22–30, doi: 10.1145/3412451.3428500

[22] A. D. King *et al.*, "Scaling advantage in quantum simulation of geometrically frustrated magnets," 2019, *arXiv:1911.03446*.

[23] V. Choi, "Minor-embedding in adiabatic quantum computation: I. The parameter setting problem," *Quantum Inf. Process.*, vol. 7, no. 5, pp. 193–209, 2008, doi: 10.1007/s11128-008-0082-9

[24] V. Choi, "Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 343–353, 2011, doi: 10.1007/s11128-010-0200-3

[25] W. Lechner, P. Hauke, and P. Zoller, "A quantum annealing architecture with all-to-all connectivity from local interactions," *Sci. Adv.*, vol. 1, no. 9, 2015, Art. no. e1500838, doi: 10.1126/sciadv.1500838

[26] A. Rocchetto, S. C. Benjamin, and Y. Li, "Stabilisers as a design tool for new forms of Lechner-Hauke-Zoller annealer," *Sci. Adv.*, no. 2, 2016, Art. no. e1601246, doi: 10.1126/sciadv.1601246

[27] M. Leib, P. Zoller, and W. Lechner, "A transmon quantum annealer: Decomposing many-body Ising constraints into pair interactions," *Quantum Sci. Technol.*, vol. 1, no. 1, 2016, Art. no. 015008, doi: 10.1088/2058-9565/1/1/015008

[28] N. Chancellor, S. Zohren, and P. A. Warburton, "Circuit design for multi-body interactions in superconducting quantum annealing systems with applications to a scalable architecture," *npj Quantum Inf.*, vol. 3, 2017, Art. no. 21, doi: 10.1038/s41534-017-0022-6

[29] T. Albash, W. Vinci, and D. A. Lidar, "Simulated quantum annealing with two all-to-all connectivity schemes," *Phys. Rev. A*, vol. 94, no. 2, 2016, Art. no. 022327, doi: 10.1103/PhysRevA.94.022327

[30] N. Chancellor, "Domain wall encoding of discrete variables for quantum annealing and QAOA," *Quantum Sci. Technol.*, vol. 4, no. 4, Aug. 2019, Art. no. 045004, doi: 10.1088/2058-9565/ab33c2

[31] S. Abel, N. Chancellor, and M. Spannowsky, "Quantum computing for quantum tunnelling," 2020, *arXiv:2003.07374.*

[32] S. Abel and M. Spannowsky, "Observing the fate of the false vacuum with a quantum laboratory," 2020, *arXiv:2006.06003*.

[33] N. Chancellor, "Fluctuation-guided search in quantum annealing," *Phys. Rev. A*, vol. 102, no. 6, Dec. 2020, Art. no. 062606, doi: 10.1103/PhysRevA.102.062606

[34] N. Chancellor, "Code associated with: Domain wall encoding of integer variables for quantum annealing and QAOA," doi: 10.15128/r27d278t029

[35] D. Joseph, A. Callison, C. Ling, and F. Mintert, "Two quantum Ising algorithms for the shortest-vector problem," *Phys. Rev. A*, vol. 103, no. 3, Mar. 2021, Art. no. 032433, doi: 10.1103/PhysRevA.103.032433

[36] N. Chancellor, S. Zohren, P. A. Warburton, S. C. Benjamin, and S. Roberts, "A direct mapping of max k-SAT and high order parity checks to a chimera graph," *Sci. Rep.*, vol. 6, no. 1, Nov. 2016, Art. no. 37107, doi: 10.1038/srep37107

[37] R. Blume-Kohout, C. M. Caves, and I. H. Deutsch, "Climbing mount scalable: Physical resource requirements for a scalable quantum computer," *Found. Phys.*, vol. 32, no. 11, pp. 1641–1670, Nov. 2002, doi: 10.1023/A:1021471621587

[38] I. Hen and M. S. Sarandy, "Driver hamiltonians for constrained optimization in quantum annealing," *Phys. Rev. A*, vol. 93, no. 6, Jun. 2016, Art. no. 062312, doi: 10.1103/PhysRevA.93.062312

[39] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms,* vol. 12, no. 2, 2019, Art. no. 34, doi: 10.3390/a12020034

[40] A. Frieze and M. Jerrum, "Improved approximation algorithms for MAXk-CUT and max bisection," *Algorithmica*, vol. 18, no. 1, pp. 67–81, May 1997, doi: 10.1007/BF02523688

[41] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, "Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?," *SIAM J. Comput.*, vol. 37, no. 1, pp. 319–357, 2007, doi: 10.1137/S0097539705447372

[42] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[43] P. Erdös and A. Rényi, "On the evolution of random graphs," *Bull. Inst. Int. Statist.*, vol. 38, pp. 343–347, 1960, doi: 10.1515/9781400841356.38

[44] T. Stollenwerk, S. Hadfield, and Z. Wang, "Toward quantum gate-model heuristics for real-world planning problems," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art no. 3101816, doi: 10.1109/TQE.2020.3030609

[45] "Documentation on d-wave uniform torque compensation tool," Accessed: Jan. 19, 2021. [Online]. Available: https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/generated/dwave.embedding.chain_strength.uniform_torque_compensation.html

[46] "D-wave ocean software documentation," Accessed: Jan. 19, 2021. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/

[47] "minorminer," Accessed: Jan. 19, 2021. [Online]. Available: https://github.com/dwavesystems/minorminer

[48] J. Chen, T. Stollenwerk, and N. Chancellor, "Performance of domain-wall encoding for quantum annealing [dataset]," 2021, doi:10.15128/r11j92g748d. [Online]. Available: http://doi.org/10.15128/r11j92g748d

[49] G. Van Rossum and F. L. Drake, *Python Language Reference Manual*. Godalming, U.K.: Network Theory, 2003.

[50] "Numpy1.11.1." Accessed: Aug. 10, 2016. [Online]. Available: http://www.numpy.org/

[51] T. E. Oliphant, *A Guide to NumPy*, vol. 1. USA: Trelgol Publishing, 2006.

[52] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55

[53] T. Kluyver *et al.*, "Jupyter notebooks—A publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. Amsterdam, The Netherlands: IOS Press, 2016, pp. 87–90, doi: 10.3233/978-1-61499-649-1-87.

[54] F. Pérez and B. E. Granger, "IPython: A system for interactive scientific computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, 2007, doi: 10.1109/MCSE.2007.53

[55] E. L. Lehmann, "The fisher, Neyman-Pearson theories of testing hypotheses: One theory or two?," *J. Amer. Statist. Assoc.*, vol. 88, no. 424, pp. 1242–1249, 1993, doi: 10.2307/2291263

[56] D. Venturelli and A. Kondratyev, "Reverse quantum annealing approach to portfolio optimization problems," *Quantum Mach. Intell.*, vol. 1, no. 1, pp. 17–30, May 2019, doi: 10.1007/s42484-019-00001-w

[57] J. Marshall, D. Venturelli, I. Hen, and E. G. Rieffel, "Power of pausing: Advancing understanding of thermalization in experimental quantum annealers," *Phys. Rev. Appl.*, vol. 11, Apr. 2019, Art. no. 044083, doi: 10.1103/PhysRevApplied.11.044083

[58] H. Karimi, G. Rosenberg, and H. G. Katzgraber, "Effective optimization using sample persistence: A case study on quantum annealers and various Monte Carlo optimization methods," *Phys. Rev. E*, vol. 96, Oct. 2017, Art. no. 043312, doi: 10.1103/PhysRevE.96.043312

[59] S. H. Yarkoni, A. Wang Plaat, and T. Bäck, "Boosting quantum annealing performance using evolution strategies for annealing offsets tuning," *Quantum Technology and Optimization Problems*, S. Feld and C. Linnhoff-Popien, Eds. Cham, Switzerland: Springer, 2019, pp. 157–168, doi: 10.1007/978-3-030-14082-3_14.

[60] "Source code for dwave.system.composites.virtual_graph," Accessed: Jan. 19, 2021. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/_modules/dwave/system/composites/virtual_graph.html

[61] "Discrete quadratic models," Accessed: Jan. 19, 2021. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/concepts/dqm.html

[62] M. Tanvi, "Forum response to: Does the discrete quadratic model function use a one-hot encoding?," D-Wave user forums. Dec. 8, 2020. [Online]. Available: https://support.dwavesys.com/hc/en-us/community/posts/360051436394/comments/360015043293

[63] N. Chancellor *et al.*, "Disorder-induced entropic potential in a flux qubit quantum annealer," 2020, *arXiv:2006.07685.*

[64] Z. G. Izquierdo, T. Albash, and I. Hen, "Testing a quantum annealer as a quantum thermal sampler," *ACM Trans. Quantum Comput.*, vol. 2, 2021, Art. no. 7, doi: 10.1145/3464456