First results solving arbitrarily structured Maximum Independent Set problems using quantum annealing

Sheir Yarkoni*[1,2], Aske Plaat[2], and Thomas Bäck[2]

[1]D-Wave Systems Inc., Burnaby, Canada
[2]LIACS, Leiden University, Netherlands

Abstract

Commercial quantum processing units (QPUs) such as those made by D-Wave Systems are being increasingly used for solving complex combinatorial optimization problems. In this paper, we review a canonical NP-hard problem, the Maximum Independent Set (MIS) problem. We show how to map MIS problems to quadratic unconstrained binary optimization (QUBO) problems, and use a D-Wave 2000Q QPU to solve them. We compare the results from the D-Wave system to classical algorithms such as simulated thermal annealing and the graphical networks package NetworkX. To our knowledge, these are the first results of experiments involving arbitrarily-structured MIS inputs using a D-Wave QPU. We find that the QPU can be used as a heuristic optimizer for randomly generated inputs, but due to physical control errors, can be outperformed by simulated thermal annealing.

I. Introduction

Current-generation quantum processing units (QPUs), like the D-Wave 2000Q, are used as metaheuristics for solving binary optimization and sampling problems. The D-Wave QPUs are physical implementations of the Ising model, using superconducting loops of niobium metal as the quantum bits, or qubits [1]. At the beginning of the computation, each qubit starts in a state of superposition as both 1 and 0. As the system evolves, an energy landscape is introduced while simultaneously lowering the strength of the transverse field, allowing qubits to tunnel dynamically through energy barriers. How these quantum effects contribute to solving computationally challenging optimization and sampling problems is well documented in literature [2]–[4]. The D-Wave QPUs are designed to minimize the following objective function:

$$\mathrm{Obj}(x, Q) = x^T \cdot Q \cdot x, \tag{1}$$

where Q is a real-valued $N \times N$ matrix, and x is a bit-string vector. These problems are known as quadratic unconstrained binary optimization (QUBO) problems.

The maximum independent set (MIS) problem is defined as follows: Given an undirected graph G, with vertices V and edges E, find the maximum set of vertices $V' \in V$, such that no two vertices in V' share an edge in E. The decision version of this problem is NP-complete, and finding the maximum such set is NP-hard. MIS can be transformed to QUBO form in the following way: For every vertex $v \in V$, assign a binary variable x_i, where $x_i = 1$ denotes that x_i is in the candidate solution, and $x_i = 0$ denotes it is not. Let Q be the matrix representing the MIS problem in QUBO form. Assign a weight of -1 to every diagonal element Q_{ii}. For every edge in $e \in E$ between nodes v_i and v_j (represented by binary variables x_i and x_j), assign the off-diagonal term Q_{ij} a value of 2. Thus, the minimum of this QUBO is when no two adjacent nodes are selected, while maximizing the number of nodes in the set. Now the set of selected nodes is both independent and the size is maximized, as required. A similar transformation of the MIS problem to the Ising model is shown in [5]. The Ising model is equivalent to QUBO (up to a constant offset), using a simple transformation of variables. Given an Ising spin s (in the $\{-1, 1\}$ basis), and a QUBO binary variable x (in the $\{0, 1\}$ basis), a simple transformation is given by $s = 2x - 1$. A full derivation of an Ising form of MIS (as well as many other NP-complete problems) is shown in [5].

* Corresponding author: syarkoni@dwavesys.com

II. Previous works

Solving general MIS problems using quantum annealing was originally investigated in [6] to show that exponentially small energy gaps between the global minimum and first excited states did not necessarily prevent quantum annealing from failing to solve NP-complete problems. However, this result is a theoretical proof, and a follow-up paper with experimental results used only quantum Monte Carlo simulations to solve these problems [7].

Practical results solving the MIS problem are shown in [8], however these problems were generated on the D-Wave QPU's native structure which is, by construction, bipartite. Therefore, these instances have solutions which can be found in polynomial time. Other previous reports testing arbitrarily-structured MIS problem using D-Wave QPUs have focused on the Weighted Maximum Independent Set problem rather than the unweighted version considered here [9].

III. Problem testbed

In this paper, we generate random graphs, and empirically determine the point of maximum difficulty. This is measured by finding the minimum probability of the QPU to find the presumed MIS as a function of edge probability p in the random graph. We find that for the QPU, the point of maximum difficulty is around $p = 0.2$, as shown in Figure 1. This point of maximum difficulty is similar to that found in previous studies for random graphs [10].

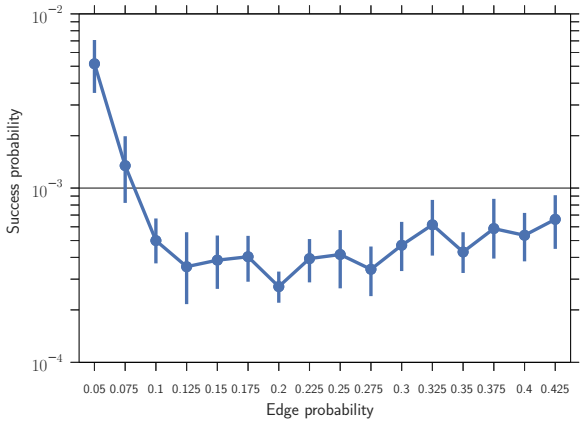

Fig. 1. Plot showing the performance of the D-Wave 2000Q as a function of edge probability (p) in random graphs. The point with the lowest success probability (lowest in the graph) is presumed to be maximally difficult for the QPU; this is observed to be at $p = 0.2$.

For every point in Figure 1, we generated 50 random graph instances testing edge probabilities $p \in [0.05, 0.425]$. We then submitted the problem directly to the QPU, and collected 100,000 sample solutions per instance. Mean probability of success over problem instances is shown in Figure 1; error bars are bootstrapped 95% CI.

IV. Methods and solvers

The classical software chosen to compare against the D-Wave QPU were selected to fulfill two criteria. The first point is

to address the fact that this paper is *not* concerned with the physical aspects of quantum annealing. The point of interest is the performance of the QPU on a canonical NP-hard problem, something closer to an application than a physics simulation. Meaning, algorithms such as quantum Monte Carlo were not considered: while these algorithms are adept at simulating the physical dynamics of the QPU, they are not necessarily relevant to solving the MIS problem.

Currently, when solving problems directly using current D-Wave QPUs, problems *must* be posed in QUBO form. Obviously, when solving the MIS problem in practice, this is not the case. Therefore, we chose one classical algorithm that provides a quick polynomial approximation to the MIS problem natively, and another heuristic algorithm that solves the QUBO problem natively. This allows a fair comparison across both the "native" problem space of MIS, and the "specialized" problem space of QUBO. Here we review the classical software used in this study.

A. Simulated thermal annealing

Simulated thermal annealing (STA) is a heuristic optimizer, inspired by the physical annealing of metals [11]. The system is initialized to a random binary vector, mimicking infinite temperature, and is manipulated through a "cooling" schedule to reach a candidate solution. The simulated thermal annealing procedure in this paper uses Gibbs sampling, a special case of the Metropolis-Hastings algorithm [12]. At every step of the procedure, we iterate through the variables of the binary vector, and propose possible bit-flips based on the following rule:

$$P = \max\left\{1, \exp(-\beta \Delta E)\right\}, \tag{2}$$

where P is the probability of accepting the proposal, ΔE is the difference in energy resulting from the proposed bit-flip, and β is the inverse temperature parameter ($\beta = 1/T$). The schedule is the sequence in which the temperature is decremented, or "cooled". The cooling schedule we used was a linear interpolation between $\beta_{\text{start}} = 0.01$ and $\beta_{\text{end}} = 3$[1]. At each $\beta_i \in [\beta_{\text{start}}, \beta_{\text{end}}]$, we propose a flip as per Equation 2 to all variables in the problem. Updating all the variables at a single β_i is called a sweep. We perform exactly one sweep per interpolated β_i in the schedule, meaning the number of β_i is N_{sweeps}.

The distance between successive β_i in the schedule, dictated by the number of sweeps, control the rate of change in dynamics within the proposed solution. The distribution of solutions from STA is expected to converge to the Boltzmann distribution as the number of sweeps are increased. In our tests, we ran the STA routine with increasing number of sweeps ($N_{\text{sweeps}} \in \{10, 100, 1000\}$), and selected the version with the fastest average time-to-solution. This was always the $N_{\text{sweeps}} = 10$ version, and is the version used in the comparison tests. Our STA algorithm was executed on a single threaded CPU.

B. NetworkX

NetworkX is an open-source package in Python for working with and visualizing graph structures[2]. There are various algorithms implemented in this package to calculate graph properties; among these is a heuristic to estimate the maximum independent set. This heuristic is a polynomial-time approximation that returns a solution with quality bounded by $O\left(|V|/\log\left(|V^2|\right)\right)$ in the worst case [13].

[1]We normalized the inputs' Q matrix to be between 0 and 1 so that the same schedule can be used for all inputs.

[2]GitHub source: https://networkx.github.io

C. The D-Wave QPU

To program a D-Wave QPU, one specifies the matrix Q, and the QPU attempts to find the configuration of bits, x, that minimizes Equation 1. Finding this minimizing vector x is equivalent to finding the minimum of an Ising model, and is an NP-hard problem [5]. It is conjectured that, should D-Wave QPUs provide a computational speedup, this may be extended to many important and well-known applications [14], [15].

1) Submitting queries to the QPU: Currently, D-Wave QPUs only accept problems to be solved in QUBO form (or trivially transformed to an Ising Hamiltonian). While this is possible for all NP-complete problems [5], a polynomial transformation can often result in more problem variables than available qubits. Communicating directly with the D-Wave QPU is done using a client library, SAPI (Solver API), provided by D-Wave Systems. This library has implementations in C, Python, and Matlab, and has an assortment of tools to construct both QUBOs and Ising models, tuning them, and submitting them to the QPU. While there are other methods of solving complex optimization problems using the QPU (often using hybrid classical/quantum software), for simplicity we chose to use a purely quantum approach and submit problems directly using SAPI.

The topology of all current-generation D-Wave QPUs is called Chimera. This structure is a 2D lattice of unit cells, with 8 qubits in each cell, arranged as a $K_{4,4}$ bipartite graph[3]. One half of the bipartition is connected to left/right adjacent cells, and the other half of the bipartition is connected to adjacent cells on the top/bottom (connected via couplers, allowing qubits' quantum states to influence each other). This results in a relatively sparse topology, with qubits in the inner tiles having degree 6, edge tiles' qubits having degree 5. A visualization of the QPU used in this paper is shown in Figure 2. Due to this structure, arbitrarily-structured problems, like the random problems considered in this paper, must be mapped to fit this graph. This process, called minor embedding of graphs, assigns multiple variables in the target graph (in this case, multiple physical qubits on the D-Wave 2000Q QPU) to represent logical variables in the problem graph. Finding an optimal (i.e. minimum) embedding is an NP-hard problem in itself, so heuristics are used in practice. The heuristic used in this study is the `find_embedding` function provided by D-Wave in the SAPI library. This function implements an iterative algorithm that embeds one graph into another, while (heuristically) minimizing the distance between variables, i.e., the number of variables used to represent each logical variable. A full explanation of the algorithm is provided in [16].

Although generating an embedding is time-consuming, it has been shown in literature that embeddings can have strong effects on performance [3]. Therefore, in this work we investigated the possibility of using multiple embeddings for the same problem. During a test trial of 50 randomly-generated instances of 50 variables (at the point of maximum difficulty, $p = 0.2$), we generated 25 different embeddings for each problem, and submitted the embedded problem to the QPU. We measured the performance of the QPU by comparing the size of the largest independent set found by each embedding to the record maximum independent set found for each problem. We then performed linear regression to check correlation between the size of the independent set with the size of the embedding (number of qubits in the embedding). The slope of this regression line is of interest, as it

[3]Bipartite graphs, typically denoted as $K_{N,M}$ with size $N + M$ variables, are graphs with the property that all nodes can be separated into two distinct groups, with no edges within a group.

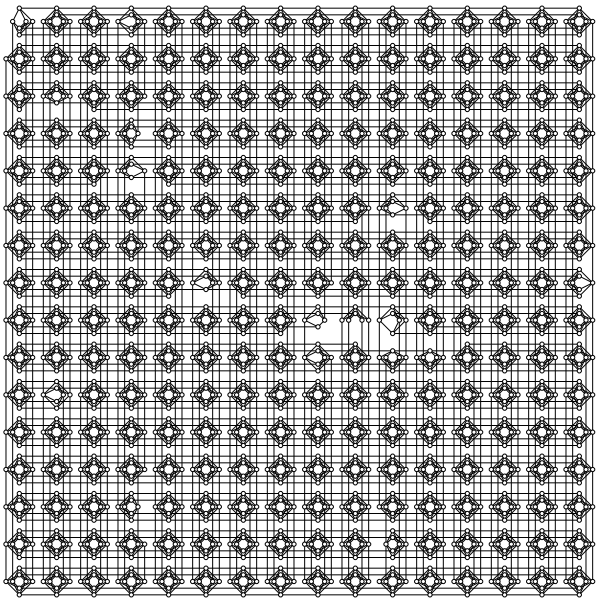

Fig. 2. Visualization of the D-Wave 2000Q QPU used in these studies. Qubits are represented as vertices and couplers are shown as edges. The maximum number of qubits on a D-Wave 2000Q QPU is 2048, however due to fabrication issues, only 2023 qubits were usable on this QPU. The number of such active qubits varies between QPUs [17], [18].

shows the marginal degradation of results with increasing embedding size. A box plot with the full suite of results is shown in Figure 3.

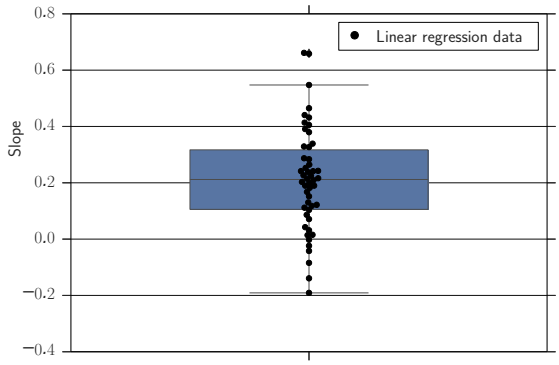

Fig. 3. Box plot showing distribution of degradation of performance as a function of increasing embedding size. Plotted are the slopes of linear regression performed for 50 unique problems (random graphs, $p = 0.2$) with 25 embeddings each. Slopes greater than zero (higher in the plot) imply smaller independent sets found by the D-Wave 2000Q with increasing embedding size.

Results show that in most cases, the size of the independent set found by QPU shrinks as the size of the embedding grows. Therefore, we find it useful to allocate some time to pre-computing embeddings, and selectively choose a "best candidate" in order to solve the problem using the QPU. The number of qubits in the embedding is demonstrated to be a sufficient criterion for performance.

Given that D-Wave QPUs are analog systems, another issue in using the QPU directly (embedding a problem) is the precision required to encode the problem. For each qubit and coupler, there is analog noise that perturb the values used to program the QUBO problem [19]. As the precision required to encode the problem grows, the analog noise becomes comparable to the gap between energy levels in the QUBO. Thus, it becomes increasingly difficult for the QPU to find the global optimum. While the MIS QUBO formulation as presented in Section I is low precision, embedding problems complicates matters. As mentioned previously, multiple qubits can be chained together to act as a single logical variable when embedding a problem. This works by applying a "chain strength", essentially an additional constraint forcing the chains to act as a single continuous qubit. However, as the size of the chain grows (i.e., the degrees of the variables in the problem), the chain strength must be increased in order to preserve the quantum properties of the qubits.

One common mitigation technique for the analog noise is the use of spin-reversal transforms, which have been shown to boost performance [4]. This method uses the generation of random strings of $\{-1, 1\}^N$ (length N) to reverse the signs of some terms in the problem submitted to the QPU, leaving the underlying structure of the problem invariant[4]. It is trivial to transform the results obtained from the QPU back to solutions of the original problem, and averaging over many such random strings averages over the analog noise. However, each spin-reversal transform requires reprogramming the QPU for a new random string, and has diminishing returns the more they are used. A rule-of-thumb for using spin-reversal transforms is suggested in [4].

2) Timing the QPU: Since the D-Wave QPU is an analog system, there are various sources to consider when timing the results. Total wall-clock time is a misrepresentation of the properties of the underlying problem, given that a remote connection to the D-Wave system needs to be established. Even then, once the problem is submitted, there are various sources of time that are uncontrollable by the user, such as time spent in the submission queue, time to convert the QUBO to analog signals, etc. Thus, we use the "total computation time" metric, established previously in literature [17]:

$$t_{\text{compute}} = N_{\text{reads}} \cdot t_{\text{anneal}} \tag{3}$$

where t_{compute} is the total *computation time*, N_{reads} is the number of solutions returned by the QPU, and t_{anneal} is the amount of time required to generate a single solution. This represents the total time required by the QPU to perform the core of the calculation, and is independent of engineering issues such as reprogramming, I/O, and latency. In the experiments presented here, we use the default value of $t_{\text{anneal}} = 20\ \mu s$. We explicitly do not include programming time as part of the computation time, despite the fact that spin-reversal transforms are used to improve the performance of the QPU. As the point of this paper is to quantify performance of the QPU in solving a well-known NP-hard problem, and not to measure end-to-end application performance (where timing is critical from a user perspective), we consider this an appropriate timing scheme.

3) Tuning QPU parameters: There are many different parameters that a user can tune when using a D-Wave QPU, such as: anneal time, anneal offsets[5], and number of spin-reversal transforms. For some of these parameters (like spin-reversal transforms) it is understood how they affect performance, and what their usage should be. However, for many other parameters it is not obvious how to set them *a priori*,

[4]As of version 2.4 of D-Wave's SAPI library, the number of spin-reversal transforms can be specified as a parameter as part of submitting a problem, which are then performed on the server side.

[5]The anneal offset features are new to the D-Wave 2000Q QPU, and allow a user to delay or advance the annealing procedure for qubits independently. This has been shown to boost performance of the QPU on some problems by up to 1000x [20]. Given the independence of each qubit, the search space for optimal anneal offset settings is exponential, so exhaustive search is impossible. Setting these parameters is out of the scope of this paper and the default values (no offset) are used.

so an exhaustive approach is used. In this section, we discuss the various QPU parameters that have been tuned in our experiments.

Although not explicitly a QPU parameter, the chain strength used in embedding a problem directly onto a QPU can strongly affect performance. Chain strength is the magnitude of couplers within a chain of qubits in an embedding. The magnitude must be high enough to enforce the chained qubits to act as a single device, but not too high as to push the logical problem into the noise range of the QPU. To test the performance of the QPU with respect to chain strength, we submitted random graph instances ($p = 0.2$, 50 random instances per problem size) with varying chain strength, and measure the success probability (probability of finding the presumed optimal MIS). The results of this experiment are shown in Figure 4.

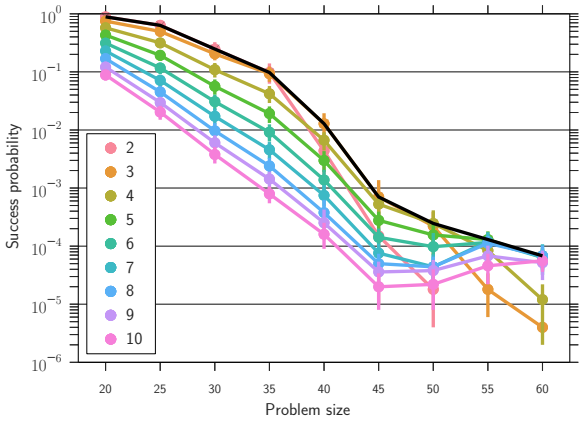

Fig. 4. Performance of the QPU as a function of the size of the random graph (edge probability $p = 0.2$). The black line is the upper envelope of the success probability (higher being better success probability), and is viewed as optimized at each problem size with respect to chain strength.

From these results, we confirm that as we increase the problem size, stronger chains result in better QPU performance, as has been shown in the past [21]. Additionally, high magnitude chain strengths at small problem sizes hurt QPU performance. However, one surprise is how low the chain strength is relative to the number of variables in the problem. The weakest chain strength tested (2) was the highest performing chain strength of all strengths tested for problem sizes $\in [20, 35]$. To understand why this is the case, we looked at the distribution of chain lengths as a function of logical problem size, shown in Figure 5. From these results, it seems that a chain strength of (2) is sufficient for chain lengths of up to 6, as this was the chain strength with the highest success probability in that region. This is likely due to two key factors. First, because the QUBO formulation of the MIS, as presented here, is relatively low precision. Second, because the edge probability in these random graphs is $p = 0.2$, the graphs that are generated are relatively sparse. This means that the chains don't experience as much torque as they would in a dense graph, making them less likely to break.

We can use the results in Figures 4 and 5 as a basic guide to choosing a chain strength given an embedding. While this is obviously only optimized for this application (MIS in random graphs with edge probability $p = 0.2$), given the density of the graph and an embedding, it is possible to make an educated initial guess for a suitable chain strength. In this study, we used the optimal chain strengths as shown in Figure 4.

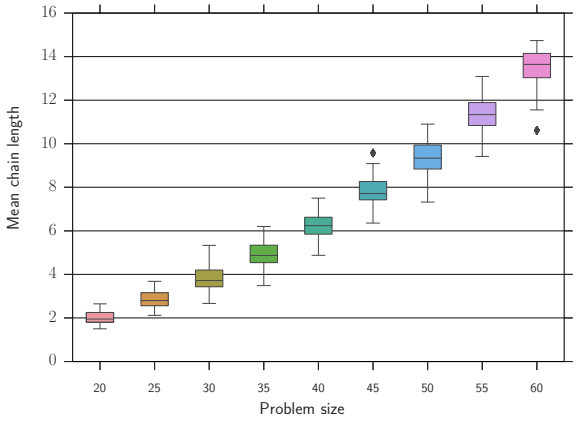

Fig. 5. Box plot showing the distribution of chain lengths as a function of number of logical variables in the problems from Figure 4. Small chain lengths (low points in the plot) are more desirable.

For the comparison tests, we submitted each MIS problem to the QPU with the parameters as presented above, and requested 10,000 samples with 100 spin-reversal transforms (the point of diminishing returns as per [4]). Therefore, the maximum runtime t_{compute} per problem for the QPU was 0.2 seconds.

V. Results

When using heuristics to solve a problem, there are three main criteria that are always in tension: quality of solution, runtime to obtain solution, and input problem size. Ideally, a good heuristic will provide optimal solutions to large problems with short runtimes. In practice, these three criteria often limit each other: a better result in one criterion will often worsen results in another (for example, better results can be obtained by running certain heuristics longer). In order to compare the solvers in this paper fairly, we present Figures 6 and 7. These show the performance of each algorithm with respect to runtime[6] and solution quality, for increasing problem sizes.

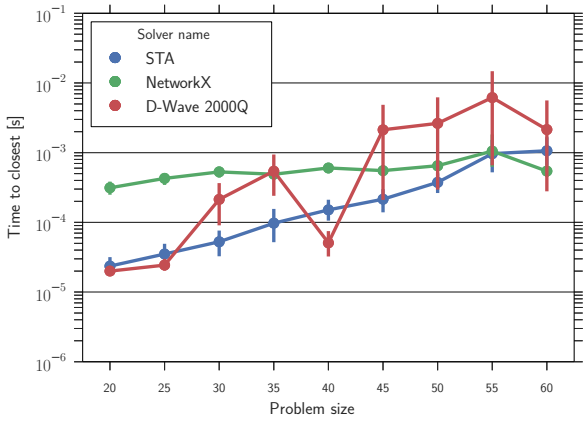

Fig. 6. Plot showing mean runtimes for the different algorithms to reach their respective presumed optimum (not the global optimum) for 50 random instances per problem size. Shorter times-to-solution (lower in the graph) is better, implying a more successful solver.

From the results in Figure 6, we can see that, especially for small problem sizes (up to $n = 40$), the D-Wave QPU finds its

[6] As stated in Section IV-C2, we compare only the computation time between the algorithms. For the classical algorithms, this means we explicitly remove the initialization time.

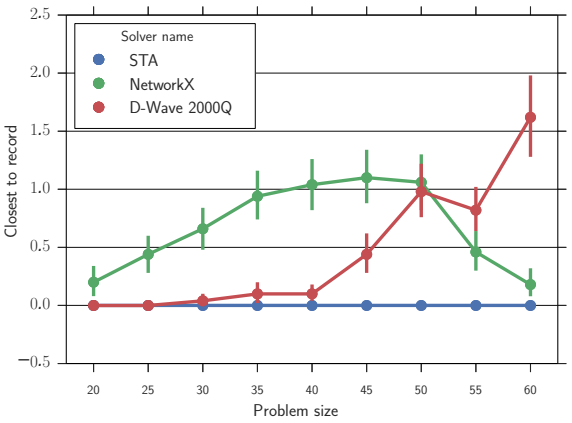

Fig. 7. Plot showing mean relative proximity of the presumed optimum to the global optimum for 50 random instances per problem size. Lower points in the graph show a solver's ability to find (putative) global optima.

presumed optimum extremely efficiently, often needing fewer than 10 samples. Additionally, these presumed optima are (on average) indeed the global optima to the random problems, as shown in Figure 7. However, as problem sizes increase, it appears the polynomial-time approximation provided by NetworkX performs increasingly better, without a significant increase in runtime. This indicates that the randomly generated inputs (with edge probability $p = 0.2$) become easier to approximate and solve classically with increasing problem sizes. That the QPU does not experience a similar boost in performance is explained by referring to Figure 5. It has been well-documented that the quantum dynamics of qubits in chains experience a "freeze-out" effect, meaning the classical states of the qubits are determined earlier in the anneal the longer the chain (as is the case with the larger inputs tested). This limits the QPU's ability to effectively solve problems, independent of the problem's combinatorial difficulty. As mentioned, tuning the anneal offset parameter designed to mitigate these effects is beyond the scope of this paper.

The best-performing algorithm in these tests was simulated thermal annealing, which always found the putative optimum, and was always the fastest in doing so for problem sizes of 30 and greater. This is additional evidence that the problem testbed was particularly easy, as a quick STA routine (10 sweeps per sample) was enough to solve these problems. That the QPU tested here was slowest in finding optima is not necessarily surprising, since the problems were generated at the point of maximum difficulty for the QPU, as stated in Section III.

VI. Conclusions

In this paper, we have introduced a method to generate arbitrarily-structured maximum independent set problems, and prepare them for submission to a D-Wave QPU. We have also shown that it is possible to use a D-Wave QPU as a heuristic to provide fast approximate solutions to these problems, evidenced by Figures 6 and 7. We observed a regime (up to input sizes of $n = 50$) where the QPU provides quick and mostly optimal results to the MIS problem. Beyond these sizes ($n > 50$) the QPU provided mostly near-optimal solutions. This was despite the evidence that the performance of the QPU was limited by physical effects, independent of the classical difficulty of the input problems. These physical effects, namely the need to use long chains to embed dense problems leading to high numerical precision, proved to be detrimental to the QPU's performance. Alternative topologies with higher density graphs (and thus lower precision) should improve performance drastically.

It is important to note that the specific inputs used in this study were synthetic and randomly generated. Additionally, in retrospect, they were easy to solve using common heuristics. This is evidenced by the fact that a short STA routine (as explained in Section IV-A) could quickly find the putative optima, usually faster than the QPU. Previous studies that compared STA and D-Wave QPUs have found that, on native instances generated on the QPU's Chimera topology, STA can often struggle to find global optima efficiently [17], [18]. This further lends credence to the notion that the QPU suffers greatly from the process of embedding problems. The specific inputs used in this study are also a single slice in the domain of randomly-generated MIS problems, as only size is varied and edge probability (p) is held constant. Thus, the results presented here should not be interpreted as general benchmarks of the QPU on the MIS problem.

A limiting factor in this study was the number of QPU parameters that need to be tuned to optimize performance. The QPU has many parameters, a subset of which were tuned in these experiments (chain strength and embedding size). It is important to stress that the QPU has additional parameters that were not tuned at these experiments, and could improve results. However, given the number of parameters of the QPU and their associated search space, it is impractical to tune all parameters exhaustively. A better solution could be to write software to automatically tune QPU parameters with an iterative approach. This could allow for proper testing of larger inputs, extending the results presented here. Given that the MIS problem is interesting both from an academic perspective as well as in applications, the MIS problem would be a good candidate for further investigation in this direction.

The focus of future research will be to incorporate iterative parameter tuning, as well as finding additional problem classes (like the MIS) that are both classically difficult and low precision in QUBO form. This will allow fair comparison of the QPU performance to classical algorithms, and highlight the use cases of quantum annealing in practical applications.

REFERENCES

[1] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, pp. 194–198, May 2011.

[2] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?," *Phys. Rev. X*, vol. 6, p. 031015, Aug 2016.

[3] D. Venturelli, D. J. J. Marchand, and G. Rojo, "Quantum annealing implementation of job-shop scheduling." arXiv:1506.08479, 2015.

[4] J. Raymond, S. Yarkoni, and E. Andriyash, "Global warming: Temperature estimation in annealers," *Frontiers in ICT*, vol. 3, p. 23, 2016.

[5] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.

[6] N. G. Dickson and M. H. S. Amin, "Does adiabatic quantum optimization fail for np-complete problems?," *Phys. Rev. Lett.*, vol. 106, p. 050502, Feb 2011.

[7] N. G. Dickson and M. H. Amin, "Algorithmic approach to adiabatic quantum optimization," *Phys. Rev. A*, vol. 85, p. 032303, Mar 2012.

[8] O. Parekh, J. Wendt, L. Shulenburger, A. Landahl, J. Moussa, and J. Aidun, "Benchmarking adiabatic quantum optimization for complex network analysis." arXiv:1604.00319, 2016.

[9] C. Wang, H. Chen, and E. Jonckheere, "Quantum versus simulated annealing in wireless interference network optimization," *Sci Rep*, vol. 6, p. 25797, May 2016.

[10] T. Back and S. Khuri, "An evolutionary heuristic for the maximum independent set problem," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, IEEE, 1994.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[12] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge, UK: Cambridge University Press, 2nd ed., 2005.

[13] R. Boppana and M. M. Halldórsson, "Approximating maximum independent sets by excluding subgraphs," *BIT*, vol. 32, pp. 180–196, May 1992.

[14] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[15] T. Albash and D. A. Lidar, "Evidence for a limited quantum speedup on a quantum annealer." arXiv:1705.07452, 2017.

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

[17] J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, "Benchmarking a quantum annealing processor with the time-to-target metric." arXiv:1508.05087, 2015.

[18] J. King, S. Yarkoni, J. Raymond, I. Ozfidan, A. D. King, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, "Quantum annealing amid local ruggedness and global frustration." arXiv:1701.04579, 2017.

[19] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, "Computational multiqubit tunnelling in programmable quantum annealers," vol. 7, p. 10327 EP, Jan 2016.

[20] E. Andriyash, Z. Bian, F. Chudak, M. Drew-Brook, A. D. King, W. G. Macready, and A. Roy, "Boosting integer factoring performance via quantum annealing offsets." https://www.dwavesys.com/resources/publications, 2016.

[21] Z. Bian, F. Chudak, R. B. Israel, B. Lackey, W. G. Macready, and A. Roy, "Mapping constrained optimization problems to quantum annealing with application to fault diagnosis," *Frontiers in ICT*, vol. 3, p. 14, 2016.