US ARO final report: Search for a quantum speedup

(Dated: September 26, 2019)

Quantum annealing promises solutions to certain computational problems that are beyond non-quantum capability, and companies such as D-Wave Systems aim to develop such devices. Although practical quantum annealers (QAs) have shown success in many fields including optimization, machine learning and material science, they are yet to prove quantum speedup over existing classical algorithms. The aim of our work is to find computational problems that demonstrate quantum speedup on a D-Wave QA. To this end, we developed an approach to search intelligently for computational problems on a D-Wave QA. Our work is important as it provides a structured method to test the quantum speedup of D-Wave QA, which is vital for future research in the direction of practical quantum annealing. The novelty of our work is to formulate a search task for computational problems and cast it as an unsupervised learning problem.

I. INTRODUCTION

Quantum annealing is a heuristic optimization procedure that finds the global minimum of an objective function by using properties of quantum physics [1]. Physically, quantum annealing is described by the Ising Hamiltonian, corresponding to the NP-Hard problem maximum weighted 2-satisfiability (MAX W2SAT). MAX W2SAT can be expressed as a quadratic unconstrained binary optimization (QUBO) problem over Boolean variable, and maximization is over an objective function that can be expressed as a bilinear form over a bit string [2]. Quantum annealing could be advantageous for some languages in MAX W2SAT, but a quantum speedup is not yet proven.

Empirically discovering whether quantum annealing offers a quantum speedup relies on two methods. The first method formulates a promising computational problem as an Ising or QUBO problem with problem instances corresponding to changing the input size measured in number of bits. One attribute of a promising problem is the potential for quantum tunnelling through barriers for quantum-enhanced optimization. The other required method is assessing whether a quantum speedup is present and this method involves seeing how computational resources scale with increasing instance sizes. Searching for quantum speedup depends on users trialling problems in a rather subjective way based on beliefs concerning what kinds of problems are more promising.

Our aim is to devise an autonomous approach to constructing problems to test by quantum annealing. Our algorithm would determine trial problems, test them and then move onto the next problem. Through this autonomous approach, a set of problems will be methodically searched, and a set of tests will be performed in each case by varying instance size and testing against various classical algorithms such as simulated annealing (SA) and quantum Monte Carlo (QMC). We plan to make this test "intelligent" in that we devise a reward function for choosing problems so that the autonomous agent learns from experience and thus rapidly finding promising problems.

Our approach to seeking quantum speedup for quantum annealing comes with daunting, fascinating, tantalizing challenges. One challenge is constructing the set of Ising problems in which the agent searches. Second, the set of problems could could have structure or be devoid of structure, and finding structure, or, alternatively, determining unstructured machine learning approaches, meaning that the learning model is wide open. A third challenge is deciding which types of quantum speedup to incorporate into the reward function. A fourth challenge is the coding needed to automate the search for problems exhibiting quantum speedup, thereby removing humans as much as possible from the process.

Enabling autonomous study of quantum speedup is significant for two reasons. One reason is that making the process autonomous can lead to a thorough methodical search for speedup and could reveal valuable cases where quantum annealing is superior to classical computing. Another valuable aspect of our autonomous approach is that casting the problem into this framework will shine a new light onto a hard program of investigation and could lead to new clear thinking about how quantum annealing could speed up computing.

The report is organized as follows: In Sec. II, we give the necessary background to our work. Sec. III describes in detail our methods and results obtained so far. We provide an overview of the roadblocks and plan in Sec. IV.

II. BACKGROUND

Twenty years since the dawn of the quantum-annealing field in 1998 [1], companies such as D-Wave Systems, Inc. have made great strides towards practical implementations of quantum annealing, refining concepts regarding the nature of a potential quantum speedup [3] including the value of finite-range quantum tunneling [4], and several tests have been performed on D-Wave QAs in the hope of finding a quantum speedup [5–8].

A D-Wave QA uses the following time-dependent quantum annealing Hamiltonian:

$$H_{\mathrm{QA}}/\mathrm{h} = \underbrace{-A(t/t_{\mathrm{f}})\sum_i \sigma_i^X}_{\text{initial Hamiltonian}} + \underbrace{B(t/t_{\mathrm{f}})\left(\sum_i h_i\sigma_i^Z + \sum_{i>j} J_{ij}\sigma_i^Z\sigma_j^Z\right)}_{\text{final Hamiltonian}}, \quad (1)$$

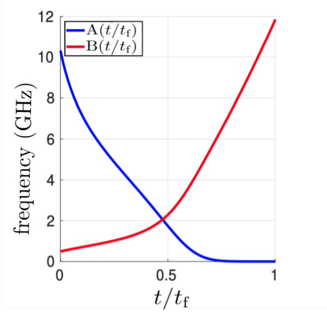


FIG. 1: Annealing schedule for a D-Wave QA (DW2000Q_5) [9].

where $t_{\rm f}$ is the annealing time, σ_i^X and σ_i^Z are the Pauli matrices acting on qubit i, and h_i and J_{ij} are the qubit biases and coupling strengths, respectively (take $\hbar = 1$). The optimization problem is represented in the form of the final Ising Hamiltonian [Eq. (1)]. In the beginning of an annealing run, the system starts in the ground state of the initial Hamiltonian [Eq. (1)], where all qubits are in a superposition of $|0\rangle$ and $|1\rangle$. During annealing the Hamiltonian of the system slowly changes from the initial to the final Hamiltonian following a preset annealing schedule given by $A(t/t_{\rm f})$ and $B(t/t_{\rm f})$ [Fig. 1]. At the end of the anneal, the system is ideally in the ground state of the final Hamiltonian, which encodes the solution of the given optimization problem.

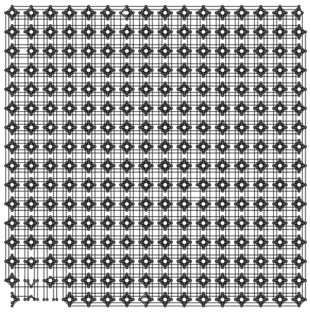
D-Wave QA solves optimization problems by converting them to a particular graph structure pertinent to the underlying connected-qubit hardware. This underlying graph is called Chimera [Fig. 2] and its properties are quite relevant to assessing spatial overhead for solving particular problems, with the number of qubits required for encoding a problem scaling anywhere from linearly to quadratically. The native connectivity of the D-Wave 2000Q has 2048 vertices (qubits) and 6016 edges (couplers), but a working chip has usually lesser number of qubits and couplers due to various technical issues. In this work we are using the system DW2000Q_5 with 2030 active qubits and 5909 active couplers which operates at temperature 13.5 mK.

The performance of a D-Wave QA is typically quantified in terms of the time-to-solution (TTS) metric [3], which is defined as the time required by the QA to find the ground state of the problem Hamiltonian at least once with probability 0.99,

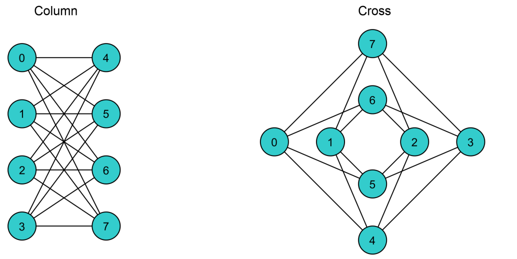
$$\mathrm{TTS} := t_{\rm f}\,\frac{\log(1-0.99)}{\log(1-p_{\rm s})} = \frac{2t_{\rm f}}{-\log\left(1-p_{\rm s}\right)}. \qquad (2)$$

In Eq. (2), $n_{\rm a}$ and $p_{\rm s}$ denotes the number of annealing runs and the probability of finding the ground state, respectively. The exact solutions of Ising type problems can be obtained using dynamic programming and exact belief propagation algorithm [10].

It has been about a decade since D-Wave announced its first QA, but yet it couldn't establish a significant speedup over classical algorithms. Google undertook a comprehensive study [4] of the performance of a D-Wave QA (D-Wave 2X) compared to SA and QMC methods on a single-core processor and the Hamze-de Freitas-Selby (HFS) algorithm for a particular problem called the weak-strong cluster problem [11]. This analysis showed that for instances with 945 variables, D-Wave 2X is $\sim 10^8$ times faster than optimized SA, $\sim 10^8$ times faster than optimized QMC and comparable to HFS algorithm.



(a) DW2000Q_5 Chimera graph



(b) Unit cell of a Chimera graph in two orientations

FIG. 2: Architecture of a D-Wave QA [9].

In most experiments involving D-Wave QA, the quantum annealing time is kept fixed for all problem instances. It was first shown in [7] that D-Wave QA also exhibits an optimal annealing time for crafted problem instances upto more than 2000 qubits. This paper also found a scaling advantage of D-Wave QA over SA but failed to find any speedup over QMC.

III. METHODS AND RESULTS

In this section we explain the rules to generate random instances of different Ising minimization problem classes following a normal distribution. We develop an algorithm that generates instances for increasing problem size and provide some examples here. We also provide a flowchart that demonstrates the process of evaluating TTS for these randomly generated problem instances. A method that uses machine learning algorithms to search for computational problems with quantum speedup is proposed, without any relevant results.

We define the Ising problem $H_{n\ell}^{(k)}$ as

$$H_{n\ell}^{(k)} := \sum_i (h_i)_{n\ell}^{(k)} Z_i + \sum_{i>j} (J_{ij})_{n\ell}^{(k)} Z_i Z_j \qquad (3)$$

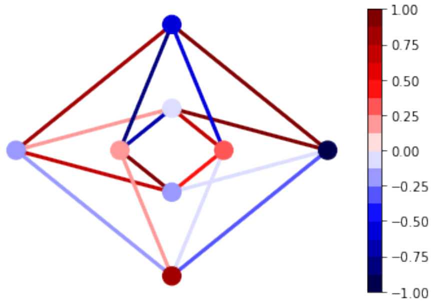


FIG. 3: A random problem designed on a Chimera unit cell with $k = 1$ and $n = n_{\mathrm{b}} = 8$.

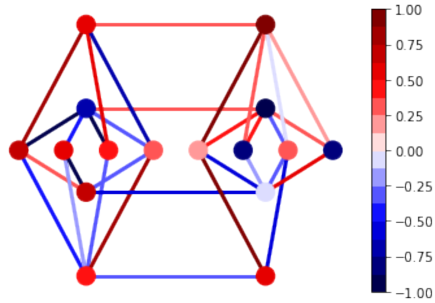
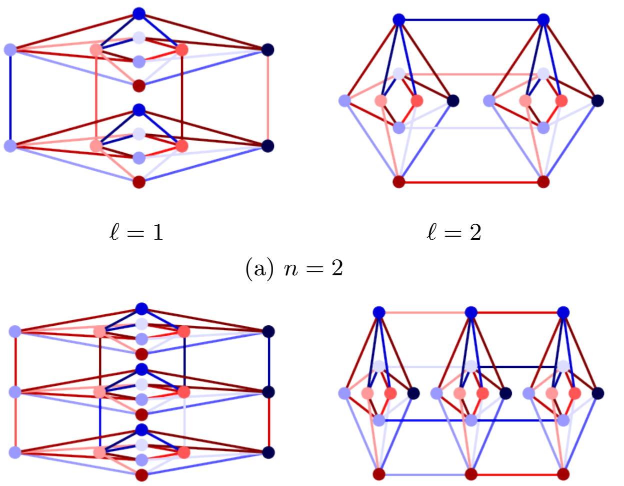


FIG. 5: A random problem designed on two adjacent Chimera unit cells with $k = 2$ and $n = n_{\mathrm{b}} = 16$.



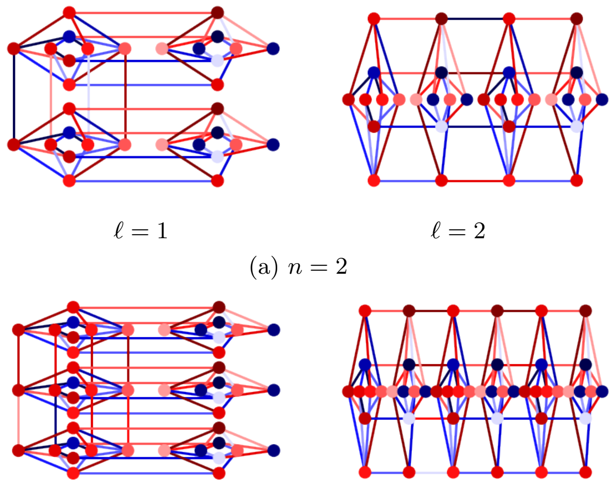
$\ell = 1$ $\ell = 2$

(a) $n = 2$

$\ell = 1$ $\ell = 2$

(b) $n = 3$

FIG. 4: Higher size problem instances of the problem depicted in Fig. 3.



$\ell = 1$ $\ell = 2$

(a) $n = 2$

$\ell = 1$ $\ell = 2$

(b) $n = 3$

FIG. 6: Higher size problem instances of the problem depicted in Fig. 5.

where k, n and l are the problem label, instance size and case label respectively. $Z_{i,j}$ denotes a spin state and thus takes a value in $\{-1, 1\}$. We construct $(h_i)_{n\ell}^{(k)}$ and $(J_{ij})_{n\ell}^{(k)}$ from $\boldsymbol{J}_{n\ell}^{(k)} \in \mathrm{sym}_{n\times n}(\mathbb{F})$: $\mathbb{F} = \mathrm{GF}\left(2^4\right)$, $(\boldsymbol{J}_{ii})_{n\ell}^{(k)} \equiv (h_i)_{n\ell}^{(k)}$ and $(\boldsymbol{J}_{ij})_{n\ell}^{(k)} \equiv (J_{ij})_{n\ell}^{(k)}$. In order to make the best use of quantum resources, we generate $\boldsymbol{J}_{n\ell}^{(k)}$ matrices restricted to the Chimera architecture of the latest D-Wave QA.

As an example, let us define a problem class with label $k = 1$ and generate some random instances with increasing size n. We first pick a building block of size n_{b} ($= 8\mathbb{N}\backslash\{0\}$ and ≤ 2048), which denotes the number of qubits for the smallest possible instance of the problem class. For $n_{\mathrm{b}} = 8$, there are 16 possible J_{ij} and 8 possible h_i, and thus $(2^4)^{16} \times (2^4)^8$ possible problem classes. We choose one of them randomly following a normal distribution and label it as $k = 1$. The smallest possible problem instance in this class has $n = n_{\mathrm{b}} = 8$ and is shown in Fig. 3. The rule for increasing instance size is to connect adjacent building blocks together with random inter-cell couplings. As the Chimera graph is in 2D, we can either add blocks horizontally or vertically, producing instances with different ℓ labels [Fig. 4] but with similar performances (TTS) [Eq. (2)]. For instances with $n > n_{\mathrm{b}}$, we calculate $\langle\mathrm{TTS}\rangle$ as the median TTS of an ensemble of 100 random problem instances of equal n.

To further clarify our rules for generating problem classes, we give another example of a problem class with $n_{\mathrm{b}} = 16$ and label it as $k = 2$ [Fig. 5]. Higher size instances with different ℓs are shown in Fig. 6. For each k, we can have multiple cases (ℓs) for $n > 2n_{\mathrm{b}}$. In the examples we only show two such cases for simplicity.

Investigating the performance of D-Wave QA on all the problem classes is intractable. Therefore, our learning agent starts initially by choosing a sample of problem classes at random which we call a training dataset [Fig. 7]. We developed a Python code to build the training dataset for the machine learning algorithm. The dataset consists

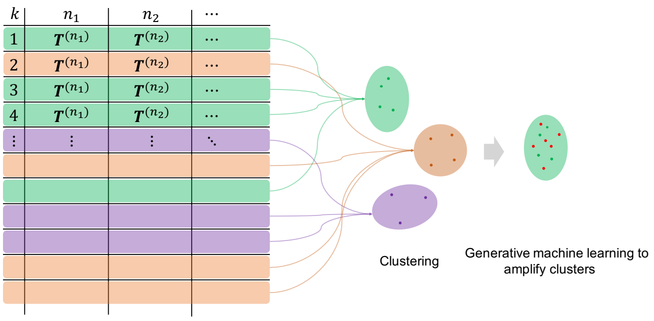


FIG. 7: Proposal of using machine learning to search for problem class(es) with quantum speedup: the table shows the dataset, following clustering and generative learning.

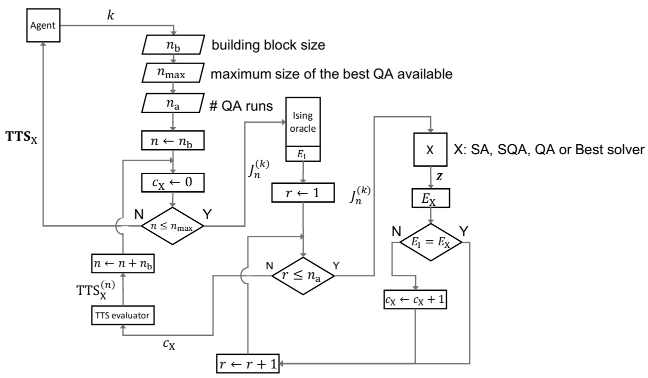


FIG. 8: Flowchart for evaluating TTS of a problem with increasing size in a problem class k.

of the description of the building block and a TTS list for each instance size, for different problem classes. The TTS list holds the $\langle \text{TTS} \rangle$ values for QA, SA and SQA, and Fig. 8 shows a flowchart depicting the algorithm to calculate TTS for any instance of a problem class with label k.

The agent then clusters problem classes in the training dataset based on the performance of the D-Wave QA for each problem class [3] employing the unsupervised method K-means. In the next step the agent generates a new set of problem classes by employing a generative adversarial network to amplify the determined clusters and intelligently searches for a class (or classes) of problems having better scaling on D-Wave QA than classical algorithms among the newly generated problem classes. We have developed implementations of both the K-means algorithm and the generative adversarial network in Python and tested the agent's performance on a synthetic dataset.

IV. CONCLUSION

We have developed an approach for intelligent searching for computational problems demonstrating a quantum speedup on a practical QA. We represent computational problems as pseudo-code rules for specifying Chimera graphs at all sizes. We have implemented an unsupervised clustering algorithm called K-means to cluster problems according to their performances (TTS), and a generative machine learning algorithm to amplify good clusters. Although our machine learning codes work well on a synthetic dataset, we couldn't perform any actual experiment with them for our objective of search for quantum speedup due to lack of an actual dataset [Fig. 7].

Acknowledgements

We acknowledge the efforts of Austin Nhung and Nolan McMahon, undergraduate students of the University of Calgary, in developing most of the Python codes used in this project. We thank Dr. Radhakrishnan Balu for his insightful comments and other help. This work is funded by US ARO grant W911NF-18-1-0203.

[1] T. Kadowaki and H. Nishimori, Physical Review E **58**, 5355 (1998).
[2] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, D-wave systems **2** (2010).
[3] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Science **345**, 420 (2014).
[4] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, Physical Review X **6**, 031015 (2016).
[5] S. Santra, G. Quiroz, G. Ver Steeg, and D. A. Lidar, New Journal of Physics **16**, 045006 (2014).
[6] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Nature Physics **10**, 218 (2014).
[7] T. Albash and D. A. Lidar, Physical Review X **8**, 031016 (2018).
[8] J. Job and D. Lidar, Quantum Science and Technology **3**, 030501 (2018).
[9] *QPU-Specific Physical Properties: DW2000Q_5*, D-Wave Systems (2019).
[10] R. Dechter, Artificial Intelligence **113**, 41 (1999).
[11] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Physical Review A **94**, 022337 (2016).