DOI: 10.14529/jsfi200403

Improving Quantum Annealing Performance on Embedded Problems

Michael R. Zielewski[1], *Mulya Agung*[2], *Ryusuke Egawa*[3], *Hiroyuki Takizawa*[1,2]

© The Authors 2020. This paper is published with open access at SuperFri.org

Recently, many researchers have been investigating quantum annealing as a solver for real-world combinatorial optimization problems. However, due to the format of problems that quantum annealing solves and the structure of the physical annealer, these problems often require additional setup prior to solving. We study how these setup steps affect performance and provide insight into the interplay among them using the job-shop scheduling problem for our evaluation. We show that the empirical probability of success is highly sensitive to problem setup, and that excess variables and large embeddings reduce performance. We then show that certain problem instances are unable to be solved without the use of additional post-processing methods. Finally, we investigate the effect of pausing during the anneal. Our results show that pausing within a certain time window can improve the probability of success, which is consistent with other work. However, we also show that the performance improvement due to pausing can be masked depending on properties of the embedding, and thus, special care must be taken for embedded problems.

Keywords: quantum annealing, quantum computer, job-shop scheduling, combinatorial optimization.

Introduction

As quantum computing devices are increasing in size and availability, quantum computing is experiencing a surge of interest as a method to efficiently solve problems that are otherwise impractical for classical computers or for which even small speedups are desirable. This interest is justified by existing quantum algorithms that improve upon classical ones, such as Shor's algorithm for integer factorization [31] and Grover's search algorithm [14]. One type of quantum computing that excels at solving combinatorial optimization problems is quantum annealing (QA). By exploiting fundamental quantum mechanical properties, such as tunneling and superposition, QA has high potential to efficiently find high quality solutions [18]. While early results [15, 20] may not have seemed promising for QA over classical solvers, later results [2, 12, 24] were more promising, and indicated that harder benchmark problems reveal the power of QA over classical solvers. QA achieves computation by preparing a system in an initial quantum configuration, and slowly evolving the system toward a final configuration, which represents the target problem. According to the adiabatic theorem, if the system is evolved slowly enough, it will remain in the ground state and the final qubit values represent a solution to the target problem [13]. However, due to problem properties, thermal excitations, noise, and other factors, the system does not always remain in the ground state; thus, solutions may not be found. Performance and usability are further impacted by preparatory steps required not only by QA, but also by the current physical implementation of QA.

One of the major challenges in using QA is converting problems to quadratic unconstrained binary optimization (QUBO) form, which is the native problem that QA solves. QUBO form requires that all variables take binary values, and that all variable terms are of degree two or less.

[1]Graduate School of Information Sciences, Tohoku University, Sendai, Japan
[2]Cyberscience Center, Tohoku University, Sendai, Japan
[3]Department of Information and Communication Engineering, Tokyo Denki University, Tokyo, Japan

For some types of problems, such as integer programming problems, conversion of non-binary variables to binary variables can be achieved through techniques such as one-hot encoding, yet will result in a significantly higher number of variables in the QUBO. For other types of problems that already use binary variables, such as the satisfiability problem, additional variables may be introduced when reducing terms having a degree greater than two. Therefore, it is clear that conversion to QUBO form often results in a problem containing significantly more variables than the original representation did.

A second challenge associated with current QA arises from the physical quantum processing unit (QPU) that implements QA. Commercially available QPUs have limited connectivity between qubits, and impose similar constraints on qubits in QUBOs. Limited connectivity can be remedied through minor embedding, which is a process that produces a mapping of a QUBO to the QPU, while allowing one logical qubit to be represented by a "chain" of multiple physical qubits. A problem that requires embedding is known as an embedded problem. As many applications, such as the one considered in this work, contain multiple variables that must be represented by qubit chains, the number of qubits in an embedding can be significantly larger than in its corresponding QUBO. This effectively reduces the maximum sizes of problems that can be solved by the QPU, especially for embedded problems with variables that share many constraints and have high connectivity.

The effects that problem formulation and embedding have on performance are not well studied. The purpose of this work is to investigate these effects, determine methods through which performance can be maximized, and investigate the interplay between them. We select scheduling problems for our evaluation. Scheduling is one of the most ubiquitous types of problems in optimization and has applications in many fields. The standard form of these problems is finding an assignment of tasks to resources that satisfies problem constraints [6]; however, real world applications require domain specific variants [16, 29, 32, 35]. In this work, we select the standard $n \times m$ job-shop scheduling problem (JSP) to evaluate the performance of QA. One characteristic of the JSP that makes it a suitable candidate for evaluating the performance of QA is that it is NP-hard, and often requires the use of heuristic solvers. Furthermore, binarization of the JSP results in extra variables, which can have a significant impact on performance. Additionally, the structure and connectivity of the JSP necessitates embedding, which allows us to evaluate the performance of a multitude of unique embeddings. Seeking methods to improve performance, we also evaluate post-processing techniques and modified anneal schedules that include a pause. Finally, we conclude with a suggestion of how the considered methods can be combined to address the above challenges and achieve high performance with QA on embedded problems. The main contributions of this work can be summarized as follows:

- We investigate the impacts of problem formulation and embedding on performance;
- We provide methods to improve performance and show the interplay between them;
- We show that the effects of anneal pausing can be masked with a poor selection of embedding, and provide an explanation as to why this occurs; and
- We suggest a combined approach to achieve high performance on embedded problems.

The rest of this paper is organized as follows. Section 1 provides background on QA and its current implementation, and reviews a JSP formulation for QA. Then, in Section 2, we describe in detail our methods for variable reduction, embedding, post-processing, and anneal schedule modification. Section 3 contains our evaluation setup, results, and analysis. Related work is discussed in Section 4. The final section contains our conclusions and potential future direction.

1. Background

In this section, we first describe QA in detail. We then introduce our target problem, the JSP, and provide details on how it can be formulated for QA.

1.1. Quantum Annealing

QA is a quantum mechanical metaheuristic for minimizing combinatorial optimization problems. System evolution follows a time-dependent Hamiltonian of the form

$$H(s) = A(s)H_I + B(s)H_F, \tag{1}$$

where A and B are time-dependent coefficients, H_I is the initial Hamiltonian, H_F is the final or problem Hamiltonian, and s is the re-scaled annealing time taking values in the range $[0, 1]$. In a system of N qubits, H_I and H_F are given by

$$H_I = \sum_{i=1}^{N} \sigma_x^i, \tag{2}$$

and

$$H_F = \sum_{i}^{N} h_i \sigma_z^i + \sum_{i}^{N} \sum_{j=i+1}^{N} J_{i,j} \sigma_z^i \sigma_z^j, \tag{3}$$

where σ_z^i and σ_z^j are the Pauli matrices operating on qubits i and j, and h and J are local biases and coupling strengths that encode the problem to be solved.

System evolution starts at $s = 0$ in the ground state of H_I, where A is large and $B = 0$. Tunneling strength, which is determined by A, is initially strong and analogous to the temperature term in the classical metaheuristic simulated annealing [21]. As system evolution progresses, A monotonically decreases to 0 and B monotonically increases, introducing H_F. According to the adiabatic theorem, the system will remain in the ground state provided that evolution is sufficiently slow with respect to the minimum gap, that is, the difference in energy between the two lowest energy states of the system [13].

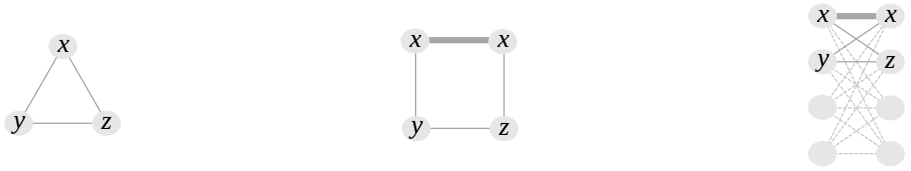
When system evolution ceases at $s = 1$, the system is in a classical state described by the Ising model

$$E(s) = \sum_{i}^{N} h_i s_i + \sum_{i}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j, \tag{4}$$

where $s_i \in \{\pm 1\}$ are spin variables. Similar to the Ising model, but having binary variables $x_i \in \{0, 1\}$, is the QUBO problem of the form

$$E(x) = \sum_{i}^{N} \sum_{j=i}^{N} Q_{i,j} x_i x_j, \tag{5}$$

where $Q_{i,j}$ is a matrix containing local and quadratic biases. Note that QUBO variables can be converted to Ising variables with the mapping $s_i = 2x_i - 1$. Due to the binary values of the QUBO matching those most frequently used in traditional computing, we use QUBO form throughout the rest of this work.



(a) Original representation (b) Including a chain (c) Embedded into a unit cell

Figure 1. An example embedding for a problem consisting of three variables

In QA, there are no limitations placed on the connectivity of qubits; however, in current QPUs implementing QA, qubit connectivity is limited. In the QPU used in this work, the layout of qubits in the QPU follows the C_{16} Chimera graph, which is a 16×16 grid of $K_{4,4}$ bipartite graphs, termed unit cells. In this graph, vertices, and thus qubits, have a maximum connectivity of six. To solve QUBOs with variables having higher connectivity, or QUBOs that have a structure that does not match that of the QPU, a technique called minor embedding must first be performed. In minor embedding, connectivity between logical variables in the QUBO is achieved by allowing the variables to be represented by a "chain" of multiple physical qubits in the QPU. To ensure that all qubits of a chain take the same value, a strong negative bias is used for intra-chain couplings. Problems that require chaining often use far more qubits in the QPU than variables in their QUBO, which effectively reduces the maximum sizes of problems that can be solved directly on the QPU.

We provide an example problem that requires embedding, as well as a potential embedding for it, in Fig. 1. The problem contains three fully connected variables, and can be represented graphically by a triangle, as shown in Fig. 1a. Note that in the unit cell in Fig. 1c, no three vertices can be used to represent this system. Only by chaining a variable, as is done in Fig. 1b, is it possible to embed the system into the unit cell as shown in Fig. 1c.

1.2. Job-shop Scheduling Problem

In this work, we consider the standard $n \times m$ JSP in which a set of n jobs J is to be scheduled on a set of m machines M [4]. Each job $j \in J$ must be processed exactly once by each machine $r \in M$. The processing of a job on a machine is called an operation, denoted by $o_i \in O = \{o_1, \ldots, o_{n \times m}\}$, where every o_i corresponds to the machine r on which the operation is to be executed. Each operation has a corresponding positive integer processing duration $d_i \in D = \{d_1, \ldots, d_{n \times m}\}$. The i-th set of operations and processing durations corresponds to job j_i.

We seek a schedule of J on M that respects the following three constraints. First, operations of a job must take place in the order defined in O, and no operation can start until all preceding operations have completed. Second, machines can only execute one operation having a nonzero processing duration at any given time. Lastly, all operations must be scheduled with exactly one starting time and cannot be interrupted during processing. A schedule is valid if none of these constraints are violated.

We define the shortest time in which all operations complete, the optimal makespan of an instance, with T. In the optimization version of the JSP, we aim to minimize the makespan of a schedule in such a way that the makespan is nearest to the instance's optimal makespan. In this work, we consider the decision variant of the JSP, in which we only seek a valid schedule.

Before solving any problem with QA, its representation must be made compatible with QUBO form. Typical problem representations may be composed of non-binary variable types, such as integers, which are not directly compatible with QUBO form and require transformation [23]. One common method of integer variable binarization is to use one-hot encoding [26]. In this method, given an integer variable whose values span N integers, we define $P = (p_1, \ldots, p_N)$ as the sequence of those integer values. A vector of binary variables $\mathbf{x} = [x_1, ..., x_N]$, $x_i \in \{0, 1\}$, is used to represent P in such a way that if $x_i = 1$, the i-th element of P is selected. In order to maintain a coherent representation by ensuring that exactly one value is selected, the following constraint is introduced:

$$\sum_{i}^{N} x_i = 1. \tag{6}$$

This constraint can then be modeled as the following penalty

$$\left(\sum_{i}^{N} x_i - 1 \right)^2. \tag{7}$$

One method of binarizing the JSP is for a binary variable $x_{i,t}$ to represent the execution of operation i at time t [6]. In this work, we use a formulation that was introduced in [36] that has already been used with QA and in which binarization is achieved through variables $x_{i,t}$ representing the *starting* of operation i at time t. One limitation of these methods is that they require $\tilde{T}$, an estimation of T used for limiting the number of binary variables created. An estimate that is less than T results in a QUBO that cannot solve the JSP instance. On the other hand, overestimating T produces an excess of binary variables. The penalties applied to these variables, representing their constraints, that are used to construct the QUBO are:

$$\sum_{i} \left(\sum_{t} x_{i,t} - 1 \right)^2, \tag{8}$$

$$\sum_{n} \left(\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{i,t} x_{i+1,t'} \right), \tag{9}$$

$$\sum_{m} \left(\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} \right). \tag{10}$$

These penalties can be summarized as follows: Equation (8) penalizes configurations in which operations do not start exactly once, Equation (9) penalizes out of order execution for operations within a job, and Equation (10) penalizes configurations that violate the capacity of a machine. For additional details, the reader is directed to [36].

2. Methods

In this section, we introduce the methods that are required for preparing a problem for QA as well as those that will be used to improve the performance of QA on the JSP.

2.1. Variable Reduction

To investigate the impacts of the formulation process on the performance of QA, we employ two commonly used software packages for creating QUBOs. The first of these packages is *D-Wave Binary CSP* (DBC). This tool is used to convert constraint satisfaction problems to QUBO form and is developed by D-Wave Systems [8]. The second package is *PyQUBO* (PYQ), developed by Recruit Communications [28]. While the object of this work is not to directly compare these two tools, we observe that the output QUBOs can differ, despite both being constructed from the same penalties and capable of producing correct schedules. For small size problems, DBC and PYQ often output identical QUBOs; however, as problem size increases, DBC outputs larger QUBOs than PYQ. The source of this discrepancy is auxiliary variables, which are only found in QUBOs constructed by DBC. While auxiliary variables are typically introduced to reduce polynomial terms with a degree higher than two, none of the terms in Equations (8), (9), or (10) contain such terms. Thus, the creation of auxiliary variables can be considered a software error that can be corrected. However, as QUBOs produced by DBC are able to successfully solve JSP instances, a comparison of results from the QUBOs of the two tools will help to emphasize the impact that formulation efficiency has on performance.

A second method of variable reduction is variable pruning, which is a formulation dependent method for removing variables that cannot be set in valid solutions. We adopt the variable pruning method from [36]. This method first examines the duration and order of operations within a job. Then, variables are removed if they allow an operation to execute at a time that would result in an invalid schedule. The specific conditions for which a variable is pruned are if it allows an operation to start earlier than the sum of the processing durations of its predecessors, or if it allows an operation to start later than the sum of the processing durations of itself and its successor operations subtracted from $\tilde{T}$. Therefore, by identifying the variables which meet these conditions, a set of variables can be pruned.

2.2. Re-Embedding

We generate embeddings for QUBOs with the heuristic minor embedding algorithm described in [7], which is the standard algorithm for problems of arbitrary structure. Owing to its heuristic nature, the minor embedding algorithm generates embeddings over a range of sizes where the largest often require double the amount of qubits than the smallest do. The significance of this becomes apparent when considering the number of qubits available in the QPU; if a QUBO is especially large or its variables have connectivity that is difficult to achieve on the QPU, the embedding algorithm may fail to find any embedding. Additionally, we generally seek embeddings that minimize either the total number of qubits required or maximum chain lengths in order to avoid factors that degrade performance, such as early freezeout [3, 7]. In order to find smaller embeddings for QUBOs and to evaluate performance at different embedding sizes, we repeat the embedding process for each QUBO.

2.3. Post-Processing

An additional post-processing step is required for problems that require chains of physical qubits. During post-processing, chains are examined to determine the value of the corresponding logical qubit. This step is necessary because despite the strong coupling bias between qubits in a chain, which compels the qubits to take the same value, the values within a chain do not always

match at the end of an anneal. In the case where all qubits within a chain take the same value, the logical qubit also takes that value. However, if the values within a chain span both binary values, the chain is "broken", and additional logic is required to determine which value the logical qubit should take. We consider two post-processing methods for resolving broken chains: majority voting (MV), which is provided in [9]; and minimizing energy (ME), which is provided in [10]. In the first method, MV, the logical qubit takes on the value that most frequently occurs in the chain. The second method, ME, implements a greedy algorithm that assigns the value resulting in the lowest energy penalty based on linear and quadratic biases associated with the variable. Related work has shown that MV can reduce the penalty of a broken chain, but leads to higher penalties from constraints of neighboring qubits [29]. We expect that ME will result in higher performance since it accounts for neighboring qubits when selecting the value for a broken chain.

2.4. Anneal Schedule Modifications

The D-Wave 2000Q provides annealing controls, which are special features that provide additional control over the anneal schedule, beyond specifying the annealing time. These controls have been used to great effect in improving performance [16, 17, 22, 25, 38]. In this work, we focus on modifying the anneal schedule so that it includes a pause, a period of time during which A and B remain constant. Marshall et al. [25] have shown that pausing at the right time allows the system to relax to the ground state, and improves success probability by orders of magnitude. A follow-up study performed by Izquierdo et al. [17] evaluated the performance improvement from pausing on embedded problems. However, there is, as yet, no evaluation into the effect that embedding size has on the performance improvement from pausing. In this work, we seek to extend the understanding of pausing and explain the effect of embedding size on pausing. To accomplish this, we perform additional anneals with modified annealing schedules that include a pause, similar to what is done in [17, 25].

3. Evaluation and Discussion

This section presents our evaluation configuration, evaluation results, and ends with a discussion of our findings.

3.1. Configuration

We solve JSP instances on the D-Wave 2000Q quantum computer. The instances on which we evaluate QA are randomly generated with $n = m \in \{3, 4, 5, 6, 7\}$ with operation processing durations, $d_i \in D$, selected from a subset of $\{0, 1, 2\}$. For each instance, T is found by solving the instance with a classical solver. The formulation described in Section 1.2 is implemented, with $\tilde{T} = T$, and a QUBO is generated with both DBC and PYQ. Initial results showed that variable pruning was a necessary procedure for obtaining any valid schedules, regardless of instance size. Thus, variable pruning is performed on each instance and no results are shown for QUBOs whose variables have not been pruned. The minor embedding algorithm was repeated one million times in order to collect embeddings of different sizes for each QUBO. Starting from 102 random JSP instances, QA was performed on a total of $36,805$ embeddings. For all samples taken with the annealer, the anneal time was set to $20\,\mu s$, and 300 anneals were performed, with all other

parameters set to their default values. In the final experiment, in which we introduce a pause into the anneal schedule, the pause duration is set to 100 µs. Pause locations were limited to the range $[0.25, 0.75]$, after initial testing revealed that the optimal pause location consistently occurred between these values, and pause locations were selected at intervals of 0.01. We also applied ten spin-reversal transforms, a procedure that is used to reduce the effect of hardware biases on results, and has been shown to be a critical step in analyzing the effects of pausing [17]. A limited set of instances was selected for evaluation, and embeddings for each instance were selected uniformly, based on the number of qubits required for the embedding.

We evaluate the performance of QA using $P_{success}$, the observed probability of successfully finding a valid schedule, that is, one that violates no constraints and is the ground state of H_F. This is a standard metric [16, 17] for evaluating the performance of QA and is defined as

$$P_{success} = \frac{number\ of\ valid\ schedules}{number\ of\ anneals}. \tag{11}$$

3.2. Results

We first show the effect of variable reduction on QUBO size and the embedding process. Figure 2 shows the number of variables in QUBOs generated by DBC and PYQ for JSP instances where $n = m = 4$, and T ranges from 5 to 13. This figure shows the effect described in Section 2.1; DBC and PYQ output QUBOs of equal size for small JSP instances, yet for larger ones, DBC introduces unnecessary auxiliary variables, resulting in larger QUBOs. For the largest instance in this figure — where $T = 13$ — auxiliary variables comprise 22% of the variables in the DBC QUBO. The effects that this has on the embedding process are shown in Fig. 3, which compares the frequency distributions of embedding sizes, which are approximately normal, for the JSP instances where $T = 9$ and $T = 13$ in Fig. 2. When $T = 9$, the small amount of auxiliary variables produced by DBC have a relatively small effect on the embedding size distribution, resulting in the distribution being shifted right and thus embeddings being larger. On the other hand, when $T = 13$, the significant number of auxiliary variables has a considerable effect on embedding size. Furthermore, the increased number of variables in the QUBO also affects the failure rate of the embedding algorithm. In this figure, the failure rate of the algorithm is visualized by redistributing the percentage of failed embeddings to each embedding size proportionally, based on the percentage of successful embeddings generated for that size. We see that the embedding failure rate is only 25% for the PYQ QUBO, but is as high as 96% for the DBC QUBO. These results show that an efficient formulation is necessary to minimize failed embeddings and embedding size.

Next, we evaluate the performance implications of embedding size on $P_{success}$. In Fig. 4, $P_{success}$ is shown for JSP instances where $n = m = 4$, yet having different operation processing times than the instances used for Fig. 2 and Fig. 3. For this series of data, the default post-processing method MV is used. These figures show the effect that unnecessary variables have on $P_{success}$; annealing using embeddings generated from the DBC QUBO produces no valid schedules. Figure 4a also shows that $P_{success}$ is moderately negatively correlated with embedding size ($r = -0.69$), and thus, fewer valid schedules are found with larger embeddings. This effect is also visible in Fig. 4b, where only the smallest embeddings of the PYQ QUBO have a nonzero $P_{success}$. These results emphasize not only the advantages of using smaller embeddings but also the disadvantages of using larger embeddings.

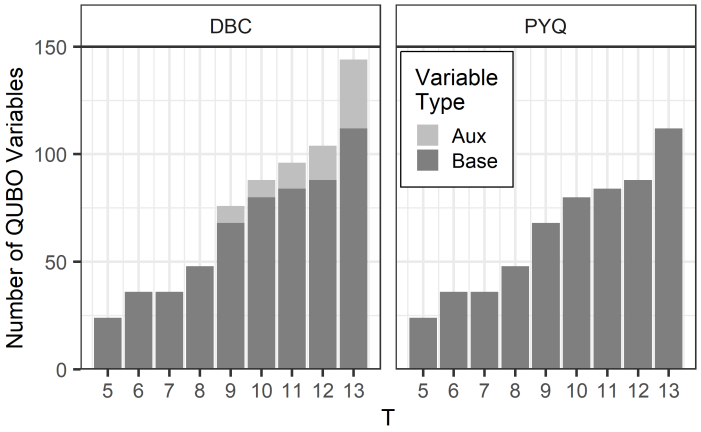
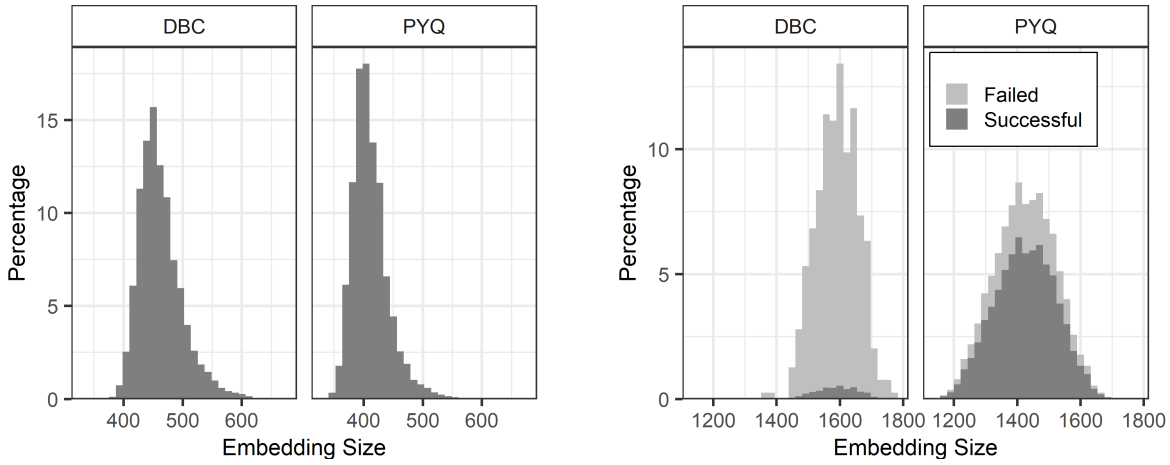
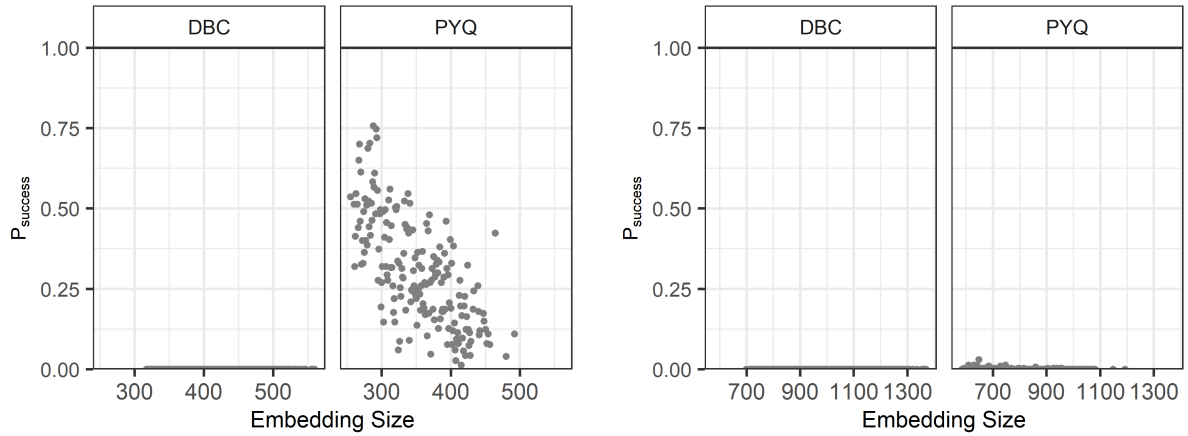


Figure 2. Comparison of the number of variables in QUBOs by tool



(a) $T = 9$ (b) $T = 13$

Figure 3. Distributions of embedding sizes for two JSP instances



(a) $T = 6$ (b) $T = 10$

Figure 4. Downward trend in $P_{success}$

We provide additional statistics on the embedding process in Fig. 5 and Fig. 6, where we show the median embedding time and embedding failure rate, which both increase with QUBO size. We see that for relatively small QUBOs, the cost of re-embedding is low due to short

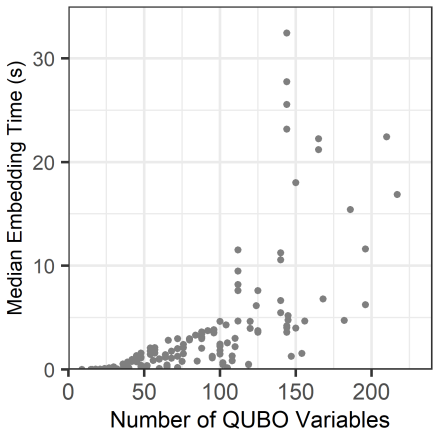


Figure 5. Median embedding times

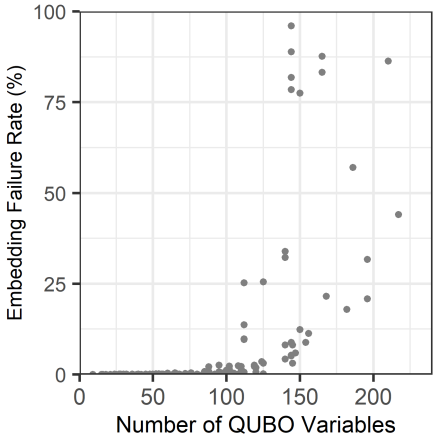


Figure 6. Embedding failure rates

Table 1. Summary of post-processing methods

$P_{success}$ comparison	Percentage
MV $<$ ME	33.32
MV $>$ ME	$<$0.01
MV $=$ ME $= 0$	56.80
MV $=$ ME $\neq 0$	9.87

embedding time and low embedding failure rate. On the other hand, for larger QUBOs, the cost of re-embedding is high due to both the long embedding time and high embedding failure rate. Considering these costs alone may suggest that re-embedding is a poor choice for large QUBOs; however, the benefit of re-embedding is highest for large QUBOs. This is shown in the results from the embeddings for PYQ QUBOs in Fig. 4a and Fig. 4b, corresponding to a small and large QUBO, respectively. In the case of the small QUBO, re-embedding found smaller embeddings that resulted in a higher $P_{success}$, but any embedding solved the instance as each resulted in a $P_{success}$ that was greater than zero. For the case of the large QUBO, the smallest embedding found through re-embedding also resulted in a higher $P_{success}$ than larger embeddings, but most larger embeddings resulted in a $P_{success}$ of zero. Therefore, despite the increasing cost of re-embedding with QUBO size, the largest QUBOs are the ones that most benefit from re-embedding, as it may be required to solve problem instances.

Next, in Fig. 7, we show a comparison of post-processing methods on PYQ QUBOs only. Here, we see that ME consistently recovers a higher number of valid schedules than MV does. As shown where $T = 9$, ME can even recover valid schedules when MV often produces none. Table **??** shows a broader comparison of post-processing methods for all embeddings used in this work. Excluding the cases where neither post-processing method resulted in a positive $P_{success}$, which often corresponds to the largest embeddings, ME resulted in a higher $P_{success}$ 77% of the time. For the majority of the remainder of the cases with a positive success rate, ME and MV returned the same number of valid schedules. The percentage of cases in which MV outperformed ME is less than 0.01%. Therefore, to maximize $P_{success}$ on the JSP, ME should be used during post-processing.

Finally, we evaluate the impacts of embedding size on an altered anneal schedule that includes a pause. Figure 8 shows our results from a set of ten embeddings of a PYQ QUBO where

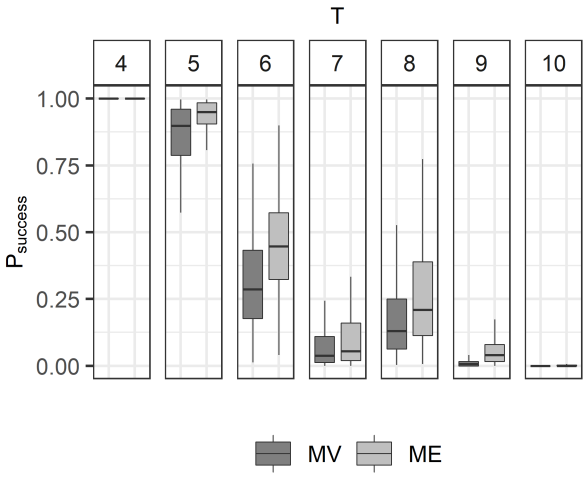


Figure 7. Comparison post-processing methods

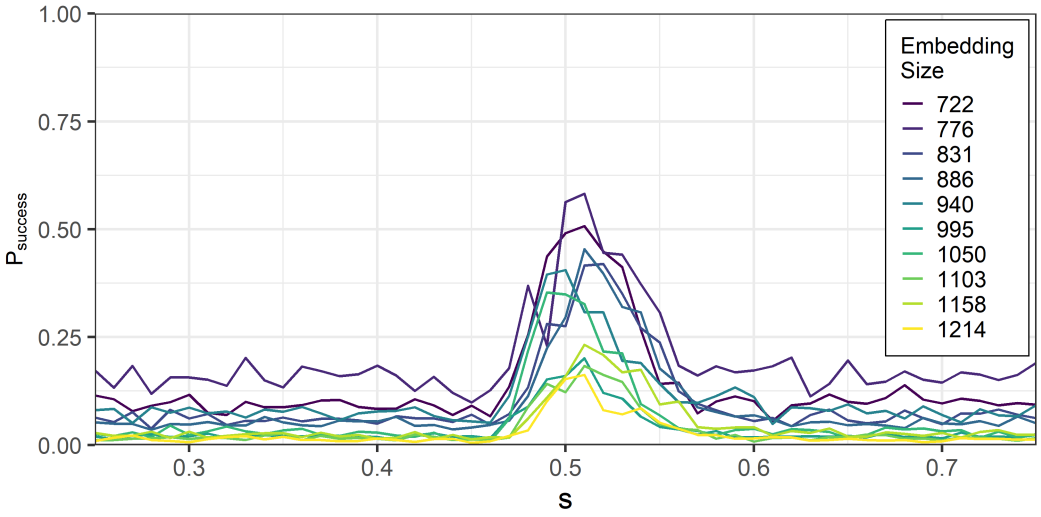


Figure 8. Effect of embedding size on pausing

ME is used during post-processing and is representative of results obtained for embeddings of other instances. These results are consistent with those in [17, 25]; only within a narrow region of s does pausing result in a higher $P_{success}$, and outside this region, $P_{success}$ remains relatively constant. In this figure, we see that $P_{success}$ decreases as embedding size increases, which was also observed in Fig. 4a. However, this figure also shows that the performance improvement from pausing near the optimal pause location also decreases as embedding size increases. We attribute this effect to the increased cost associated with flipping larger chains. That is, as the number of qubits within a chain increases, changing the value of all chained qubits comes at a higher energy cost. Thus, for embeddings with longer chains, thermal relaxation will not be as effective as it is for shorter chains. Lastly, we see that while embedding size affects $P_{success}$, it does not significantly affect the optimal pause location. In summary, we see that in order to maximize performance when pausing near the optimal pause location, embedding size should be minimized.

3.3. Discussion

As previously mentioned in the introduction to this paper, two major challenges need to be addressed to improve QA performance. Our evaluation shows how the four methods in Section 2 can address these challenges and further improve QA performance. We address the first challenge through an efficient problem formulation — which we represent with the PYQ tool — and through variable pruning. Our results show the impacts of this through smaller QUBOs, smaller embedding sizes, and reduced embedding failure rates. While the second challenge is also addressed through our variable reduction techniques, we primarily use re-embedding to find smaller embeddings that result in higher performance. The necessity of this step is shown in Fig. 4b, where the embeddings that are most likely to be found by the embedding algorithm are unable to solve the JSP instance. Despite performing both variable reduction and re-embedding, $P_{success}$ can still be low for larger QUBOs. We show that this can be remedied through proper selection of a post-processing method and insertion of a pause near the optimal pause location in an anneal schedule. However, to make the most of pausing, our results show that re-embedding needs to be performed to find smaller embeddings with relatively short chains, since the performance benefits from pausing decrease as chain length increases. Lastly, our results show that while embedding size affects the performance improvement from pausing, it does not significantly change the optimal pause location. In summary, to achieve the highest performance with QA, all the considered methods must be performed together.

4. Related Work

One of the first steps in QA is problem formulation. While this is a problem dependent step, Lucas [23] provides Ising formulations of many NP-hard problems and shows that some share fundamental components. One of the major considerations in this step is the number of variables used, as these are represented by qubits on the QPU, which are highly limited in number. The desire to minimize the number of variables can be clearly seen in the work presented by Venturelli et al. [36] through the use of variable pruning methods. However, even when using this method, only relatively small problems are able to be solved. Stollenwerk et al. [32] show effects of tunable resolution in discretization of a variable. Fine resolution results in high quality solutions, but comes at the cost of an increased number of variables. Conversely, a lower resolution can reduce the number of variables required, but results in lower quality solutions.

In order to enable solving large scale problems with current QA hardware, various hybrid quantum-classical methods have been proposed. These range from methods that partition and modify problems over many anneals to strategies for decomposing problems into subproblems to be solved by classical and quantum computers. Karimi and Rosenberg [19] present a method to iteratively set the values of variables that nearly always take the same value across many annealing runs. Through the use of this method, the authors note that not only does problem size decrease, but success rate increases, and outperforms QA when the method is not used. Another iterative method that targets problems too large to be fully embedded on the QPU is given by Rosenberg et al. [30]. In this method, a large problem is solved by iteratively setting variables, and solving the remaining subproblem. In contrast to these iterative methods are hybrid methods that employ both classical and quantum computers to solve subproblems of a decomposed problem. Such a method is presented by Tran et al. in [35], in which a battery capacity constraint for a Mars lander is offloaded to classical computers. A similar work applies this method to multiple scheduling problems, and provides the decomposition for each [34]. Yet

another hybrid quantum-classical method is to use classical computation resources as search managers for problems. In this method, a binary search tree is constructed for a QUBO, and is then used to direct the search based on results from the annealer by setting the values of certain variables. Not only does setting variable values make use of previous results, but it also provides means to reduce problem size. This method can be found in many works [34–36]. Similar to the methods in our work, hybrid methods can successfully reduce problem and embedding size, and improve performance. Additionally, hybrid quantum-classical methods can also enable guided searches of the solution space. Therefore, development of hybrid quantum-classical methods is a promising area of research for QA.

Another step required by QA that also has an effect on the size of the problem submitted to the QPU, is embedding. While embedding does not influence the size of the original problem, it does determine the size of the embedded problem, which is what will be solved on the QPU. Thus, embedding algorithms that result in small embeddings are of great interest. Currently, the embedding tool developed by D-Wave, *minorminer*, is commonly used [7]. This method employs a heuristic, resulting in embeddings of different sizes being found for the same QUBO. However, embedding size is not the only metric by which embedding algorithms are judged; two other metrics of interest are the time required to generate an embedding, and the resulting performance of annealing with a specific embedding. Therefore, embedding algorithms that can improve any of these three metrics are useful.

Pudenz et al. [27] show that as chain length increases, performance decreases. This implies that two of the metrics for embedding, embedding size and performance, may be related. However, using chain length alone to measure embedding size may be misleading, as shortening one chain may result in lengthening other chains. Abbott et al. [1] show that embedding time can eliminate any computational speedup resulting from using QA. While they do not present a new embedding algorithm, they note that for their type of problem, the dynamically weighted maximum-weight independent set problem, the embedding cost for any number of weight configurations of a graph can be reduced to the embedding cost for one configuration by reusing the embedding with altered weights. One embedding algorithm that is able to improve all three metrics, by considering the structure of the QPU, is presented by Date et al. [11]. However, their algorithm assumes the working graph of the QPU has a 100% yield, that is, no qubits on the QPU are turned off. Furthermore, the algorithm only applies to Chimera architecture QPUs, and will not apply to Pegasus architecture QPUs. In [37], Yarkoni et al. note that a parameter related to embedding, chain strength, has a strong effect on performance. These works show that the embedding process can have a significant impact on the performance of QA and affect the computational speedup obtained from using QA. These results also show that there is room for improvement in current embedding methods.

Another way of improving performance of QA is through the use of anneal controls. However, as they are only available on the latest Chimera architecture QPU, research implementing them is relatively limited. Lanting et al. [22] present one of the first works using anneal controls to improve performance by applying non-uniform driver Hamiltonians, through the use of anneal offsets, to mitigate perturbative anticrossings. Marshall et al. [25] explore the use of anneal pausing and reverse annealing, and are able to improve performance by orders of magnitude. Izquierdo et al. [17] extend this work to include embedded problems, and study the effect of chain strength. In [38], Ikeda et al. perform forward annealing, and use the results as input to reverse annealing. They show that reverse annealing can improve performance, and that selecting the lowest energy configurations from forward annealing results in higher performance than randomly selecting a configuration. However, they only explore limited schedule configurations

for reversed annealing. In these works, it is clear that anneal controls have a large configuration space, and an exhaustive search would be costly in terms of QPU access time. Yarkoni et al. [38] address this by proposing and successfully implementing an evolutionary algorithm to tune anneal offsets on a per-qubit basis. Overall, these works show that annealing controls enable performance improvements, yet finding the optimal configuration is challenging.

Conclusions and Future Work

In this work, we evaluated QA on the JSP. We first converted each JSP instance to use binary variables and determined a set of excess variables that could be pruned. We then used two tools to create QUBOs of different sizes that each produced valid schedules for the original JSP instance. Next, for each QUBO, we repeated the embedding process to find embeddings over a large range of embedding sizes. We retrieved samples from the QPU using each embedding, and applied two post-processing techniques to recover incomplete configurations. Finally, we performed annealing a second time, using a modified annealing schedule containing a pause.

Our results show that for the JSP, QA performance is sensitive to problem formulation, embedding, and post-processing. Furthermore, when studying the effects of anneal controls, such as anneal schedule modifications, special care must be taken when preparing a problem for annealing so as not to mask the effect of the control on performance. We have shown that embedding size can have a significant impact on performance, and that the embeddings that are most commonly found by embedding tools do not result in the highest performance. We then proposed re-embedding as a method to improve $P_{success}$. While re-embedding introduces a time-performance trade-off, we showed that it can be necessary for solving larger instances. We have also shown that the size of a QUBO affects the sizes of embeddings found, and consequently, QUBO size also impacts performance. Thus, minimization of QUBO size through an efficient formulation and variable pruning techniques are critical to high performance. We then showed that a higher number of valid schedules can be found through use of a post-processing method that considers the penalties associated with different qubit values. Finally, we showed that pausing can improve $P_{success}$, yet the magnitude of improvement lowers as embedding size increases, due to the increased energy cost associated with changing the state of an entire qubit chain. In summary, we have shown that due to the sequential nature of and the interplay between the steps used to solve the JSP, they must be considered together in order to achieve high performance. Furthermore, as these steps are not specific to the JSP, they can be applied to other embedded problems to improve performance.

Some main limitations of the current QA hardware are the low number of qubits and low connectivity between qubits. While Pegasus architecture QPUs [5] will increase both the number of qubits and qubit connectivity, embedded problems may still be challenging, as they often require numerous variables due to one-hot encoding of integer variables. We expect that quantum-classical hybrid methods, such as those of the related work, will become increasingly important, as they can decompose problems so that fewer resources are required for solving. Additionally, as many problems require variable chaining, embedding algorithms that find smaller, higher performing embeddings, in less time than current methods, will also be of interest for QA. Lastly, since anneal schedule modifications can significantly improve QA performance, yet many real-world applications will require embedding, which as we have shown limits the effectiveness of anneal schedule modifications, additional work is required to determine which modifications result in the highest performance.

Acknowledgements

The authors thank Professor Samy Baladram for his comments on the manuscript. We also thank Professor Masayuki Ohzeki and Professor Masamichi Miyama for their useful discussions.

This work is partially supported by MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program "R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications", Grant-in-Aid for Scientific Research(B) #16H02822 and #17H01706.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Abbott, A.A., Calude, C.S., Dinneen, M.J., et al.: A hybrid quantum-classical paradigm to mitigate embedding costs in quantum annealing. International Journal of Quantum Information 17(05), 1950042 (2019), DOI: 10.1142/S0219749919500424

2. Albash, T., Lidar, D.A.: Demonstration of a scaling advantage for a quantum annealer over simulated annealing. Phys. Rev. X 8(3), 031016 (2018), DOI: 10.1103/PhysRevX.8.031016

3. Amin, M.H.: Searching for quantum speedup in quasistatic quantum annealers. Phys. Rev. A 92(5), 052323 (2015), DOI: 10.1103/PhysRevA.92.052323

4. Applegate, D.L., Cook, W.J.: A computational study of the job-shop scheduling problem. INFORMS J. Comput. 3(2), 149–156 (1991), DOI: 10.1287/ijoc.3.2.149

5. Boothby, K., Bunyk, P., Raymond, J., et al.: Next-generation topology of D-Wave quantum processors. `https://www.dwavesys.com/sites/default/files/14-1026A-C_Next-Generation-Topology-of-DW-Quantum-Processors.pdf` (2019), accessed: 2020-11-18

6. Bowman, E.H.: The schedule-sequencing problem. Operations Research 7(5), 621–624 (1959), DOI: 10.1287/opre.7.5.621

7. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. `https://arxiv.org/abs/1406.2741` (2014), accessed: 2020-11-18

8. D-Wave Systems: D-Wave Binary CSP. `https://github.com/dwavesystems/dwavebinarycsp`, accessed: 2020-07-10

9. D-Wave Systems: dimod. `https://github.com/dwavesystems/dimod`, accessed: 2020-07-10

10. D-Wave Systems: dwave-system. `https://github.com/dwavesystems/dwave-system`, accessed: 2020-07-10

11. Date, P., Patton, R.M., Schuman, C.D., et al.: Efficiently embedding QUBO problems on adiabatic quantum computers. Quantum Inf. Process. 18(4), 117 (2019), DOI: 10.1007/s11128-019-2236-3

12. Denchev, V.S., Boixo, S., Isakov, S.V., et al.: What is the computational value of finite-range tunneling? Phys. Rev. X 6(3), 031015 (2016), DOI: 10.1103/PhysRevX.6.031015

13. Farhi, E., Goldstone, J., Gutmann, S., et al.: Quantum computation by adiabatic evolution. https://arxiv.org/abs/quant-ph/0001106 (2000), accessed: 2020-11-18

14. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 22-24 May 1996, Philadelphia, Pennsylvania, USA. pp. 212–219. ACM (1996), DOI: 10.1145/237814.237866

15. Hen, I., Job, J., Albash, T., et al.: Probing for quantum speedup in spin-glass problems with planted solutions. Phys. Rev. A 92(4), 042325 (2015), DOI: 10.1103/PhysRevA.92.042325

16. Ikeda, K., Nakamura, Y., Humble, T.S.: Application of quantum annealing to nurse scheduling problem. Scientific Reports 9(1), 12837 (2019), DOI: 10.1038/s41598-019-49172-3

17. Izquierdo, Z.G., Grabbe, S., Hadfield, S., et al.: Ferromagnetically shifting the power of pausing. https://arxiv.org/abs/2006.08526 (2020), accessed: 2020-11-18

18. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse Ising model. Phys. Rev. E 58(5), 5355–5363 (1998), DOI: 10.1103/PhysRevE.58.5355

19. Karimi, H., Rosenberg, G.: Boosting quantum annealer performance via sample persistence. Quantum Inf. Process. 16(7), 166 (2017), DOI: 10.1007/s11128-017-1615-x

20. Katzgraber, H.G., Hamze, F., Zhu, Z., et al.: Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. Phys. Rev. X 5(3), 031026 (2015), DOI: 10.1103/PhysRevX.5.031026

21. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983), DOI: 10.1126/science.220.4598.671

22. Lanting, T., King, A.D., Evert, B., et al.: Experimental demonstration of perturbative anticrossing mitigation using nonuniform driver hamiltonians. Phys. Rev. A 96(4), 042322 (2017), DOI: 10.1103/PhysRevA.96.042322

23. Lucas, A.: Ising formulations of many NP problems. Frontiers in Physics 2, 5 (2014), DOI: 10.3389/fphy.2014.00005

24. Mandrà, S., Zhu, Z., Wang, W., et al.: Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches. Phys. Rev. A 94(2), 022337 (2016), DOI: 10.1103/PhysRevA.94.022337

25. Marshall, J., Venturelli, D., Hen, I., et al.: Power of pausing: Advancing understanding of thermalization in experimental quantum annealers. Phys. Rev. Applied 11(4), 044083 (2019), DOI: 10.1103/PhysRevApplied.11.044083

26. Okada, S., Ohzeki, M., Taguchi, S.: Efficient partition of integer optimization problems with one-hot encoding. Scientific Reports 9(1), 13036 (2019), DOI: 10.1038/s41598-019-49539-6

27. Pudenz, K.L., Albash, T., Lidar, D.A.: Error-corrected quantum annealing with hundreds of qubits. Nature Communications 5(1), 3243 (2014), DOI: 10.1038/ncomms4243

28. Recruit Communications: PyQUBO. https://github.com/recruit-communications/pyqubo, accessed: 2020-07-10

29. Rieffel, E.G., Venturelli, D., OGorman, B., et al.: A case study in programming a quantum annealer for hard operational planning problems. Quantum Information Processing 14(1), 1–36 (2015), DOI: 10.1007/s11128-014-0892-x

30. Rosenberg, G., Vazifeh, M., Woods, B., et al.: Building an iterative heuristic solver for a quantum annealer. Comput. Optim. Appl. 65(3), 845–869 (2016), DOI: 10.1007/s10589-016-9844-y

31. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. 41(2), 303–332 (1999), DOI: 10.1137/S0036144598347011

32. Stollenwerk, T., OGorman, B., Venturelli, D., et al.: Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. IEEE Transactions on Intelligent Transportation Systems 21(1), 285–297 (2020), DOI: 10.1109/TITS.2019.2891235

33. Tanahashi, K., Takayanagi, S., Motohashi, T., et al.: Application of Ising machines and a software development for Ising machines. Journal of the Physical Society of Japan 88(6), 061010 (2019), DOI: 10.7566/JPSJ.88.061010

34. Tran, T.T., Do, M., Rieffel, E.G., et al.: A hybrid quantum-classical approach to solving scheduling problems. In: Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, 6-8 July 2016, Tarrytown, NY, USA. pp. 98–106. AAAI Press (2016), http://aaai.org/ocs/index.php/SOCS/SOCS16/paper/view/13958

35. Tran, T.T., Wang, Z., Do, M., et al.: Explorations of quantum-classical approaches to scheduling a Mars lander activity problem. In: Planning for Hybrid Systems, Papers from the 2016 AAAI Workshop, 13 February 2016, Phoenix, Arizona, USA. AAAI Workshops, vol. WS-16-12. AAAI Press (2016), http://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12664

36. Venturelli, D., Marchand, D.J.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling. In: Proceedings of the Eleventh Workshop on Constraint Satisfaction Techniques for Planning and Scheduling, COPLAS 2016, 13-14 June 2016, London, UK. pp. 25–34 (2016)

37. Yarkoni, S., Plaat, A., Bäck, T.: First results solving arbitrarily structured maximum independent set problems using quantum annealing. In: 2018 IEEE Congress on Evolutionary Computation, CEC 2018, 8-13 July 2018, Rio de Janeiro, Brazil. pp. 1–6. IEEE (2018), DOI: 10.1109/CEC.2018.8477865

38. Yarkoni, S., Wang, H., Plaat, A., et al.: Boosting quantum annealing performance using evolution strategies for annealing offsets tuning. In: Feld, S., Linnhoff-Popien, C. (eds.) Quantum Technology and Optimization Problems - First International Workshop, QTOP@NetSys 2019, 18 March 2019, Munich, Germany, Proceedings. Lecture Notes in Computer Science, vol. 11413, pp. 157–168. Springer (2017), DOI: 10.1007/978-3-030-14082-3_14