

Job Shop Scheduling Solver based on Quantum Annealing

Davide Venturelli^{1,2}, Dominic J.J. Marchand³, Galo Rojo³

¹*Quantum Artificial Intelligence Laboratory (QuAIL), NASA Ames*

²*U.S.R.A. Research Institute for Advanced Computer Science (RIACS)*

³*1QB Information Technologies (1QBit)*

Quantum annealing is emerging as a promising near-term quantum computing approach to solving combinatorial optimization problems. A solver for the job-shop scheduling problem that makes use of a quantum annealer is presented in detail. Inspired by methods used for constraint satisfaction problem (CSP) formulations, we first define the makespan-minimization problem as a series of decision instances before casting each instance into a time-indexed quadratic unconstrained binary optimization. Several pre-processing and graph-embedding strategies are employed to compile optimally parametrized families of problems for scheduling instances on the D-Wave Systems' Vesuvius quantum annealer (D-Wave Two). Problem simplifications and partitioning algorithms, including variable pruning, are discussed and the results from the processor are compared against classical global-optimum solvers.

I. I. INTRODUCTION

The commercialization and independent benchmarking [1–4] of quantum annealers based on superconducting qubits has sparked a surge of interest for near-term practical applications of quantum analog computation in the optimization research community. Many of the early proposals for running useful problems arising in space science [5] have been adapted and have seen small-scale testing on the D-Wave Two processor [6]. The best procedure for comparison of quantum analog performance with traditional digital methods is still under debate [3, 7, 8] and remains mostly speculative due to the limited number of qubits on the currently available hardware. While waiting for the technology to scale up to more significant sizes, there is an increasing interest in the identification of small problems which are nevertheless computationally challenging and useful. One approach in this direction has been pursued in [9], and consisted in identifying parametrized ensembles of random instances of operational planning problems of increasing sizes that can be shown to be on the verge of a solvable-unsolvable phase transition. This condition should be sufficient to observe an asymptotic exponential scaling of runtimes, even for instances of relatively small size, potentially testable on current- or next-generation D-Wave hardware. An empirical takeaway from [6] (validated also by experimental results in [10, 11]) was that the established programming and program running techniques for quantum annealers seem to be particularly amenable to scheduling problems, allowing for an efficient mapping and good performance compared to other applied problem classes like automated navigation and Bayesian-network structure learning [12].

Motivated by these first results, and with the intention to challenge current technologies on hard problems of practical value, we herein formulate a quantum annealing version of the job-shop scheduling problem (JSP). The JSP is essentially a general paradigmatic constraint satisfaction problem (CSP) framework for the problem

of optimizing the allocation of resources required for the execution of sequences of operations with constraints on location and time. We provide compilation and running strategies for this problem using original and traditional techniques for parametrizing ensembles of instances. Results from the D-Wave Two are compared with classical exact solvers. The JSP has earned a reputation for being especially intractable, a claim supported by the fact that the best general-purpose solvers (CPLEX, Gurobi Optimizer, SCIP) struggle with instances as small as 10 machines and 10 jobs (10×10) [13]. Indeed, some known 20×15 instances often used for benchmarking still have not been solved to optimality even by the best special-purpose solvers [14], and 20×20 instances are typically completely intractable. We note that this early work constitutes a wide-ranging survey of possible techniques and research directions and leave a more in-depth exploration of these topics for future work.

A. Problem definition and conventions

Typically the JSP consists of a set of jobs $\mathcal{J} = \{\mathbf{j}_1, \dots, \mathbf{j}_N\}$ that must be scheduled on a set of machines $\mathcal{M} = \{\mathbf{m}_1, \dots, \mathbf{m}_M\}$. Each job consists of a sequence of operations that must be performed in a predefined order

$$\mathbf{j}_n = \{O_{n1} \rightarrow O_{n2} \rightarrow \dots \rightarrow O_{nL_n}\}.$$

Job \mathbf{j}_n is assumed to have L_n operations. Each operation O_{nj} has an integer execution time p_{nj} (a value of zero is allowed) and has to be executed by an assigned machine $\mathbf{m}_{q_{nj}} \in \mathcal{M}$, where q_{nj} is the index of the assigned machine. There can only be one operation running on any given machine at any given point in time and each operation of a job needs to complete before the following one can start. The usual objective is to schedule all operations in a valid sequence while minimizing the makespan (i.e., the completion time of the last running job), although other objective functions can be used. In what follows, we will denote with \mathcal{T} the minimum possible makespan associated with a given JSP instance.

As defined above, the JSP variant we consider is denoted $\mathbf{JM}|p_{nj} \in [p_{\min}, \dots, p_{\max}]|C_{\max}$ in the well-known $\alpha|\beta|\gamma$ notation, where p_{\min} and p_{\max} are the smallest and largest execution times allowed, respectively. In this notation, \mathbf{JM} stands for job-shop type on M machines, and C_{\max} means we are optimizing the makespan.

For notational convenience, we enumerate the operations in a lexicographical order in such a way that

$$\begin{aligned} \mathbf{j}_1 &= \{O_1 \rightarrow \dots \rightarrow O_{k_1}\}, \\ \mathbf{j}_2 &= \{O_{k_1+1} \rightarrow \dots \rightarrow O_{k_2}\}, \\ &\dots \\ \mathbf{j}_N &= \{O_{k_{N-1}+1} \rightarrow \dots \rightarrow O_{k_N}\}. \end{aligned} \quad (1)$$

Given the running index over all operations $i \in \{1, \dots, k_N\}$, we let q_i be the index of the machine \mathbf{m}_{q_i} responsible for executing operation O_i . We define I_m to be the set of indices of all of the operations that have to be executed on machine \mathbf{m}_m , i.e., $I_m = \{i : q_i = m\}$. The execution time of operation O_i is now simply denoted p_i .

A priori, a job can use the same machine more than once, or use only a fraction of the M available machines. For benchmarking purposes, it is customary to restrict a study to the problems of a specific family. In this work, we define a ratio θ that specifies the fraction of the total number of machines that is used by each job, assuming no repetition when $\theta \leq 1$. For example, a ratio of 0.5 means that each job uses only $0.5M$ distinct machines.

B. Quantum annealing formulation

In this work, we seek a suitable formulation of the JSP for a quantum annealing optimizer (such as the D-Wave Two). The optimizer is best described as an oracle that solves an Ising problem with a given probability [15]. This Ising problem is equivalent to a quadratic unconstrained binary optimization (QUBO) problem [10]. The binary polynomial associated with a QUBO problem can be depicted as a graph, with nodes representing variables and values attached to nodes and edges representing linear and quadratic terms, respectively. The QUBO solver can similarly be represented as a graph where nodes represent qubits and edges represent the allowed connectivity. The optimizer is expected to find the global minimum with some probability which itself depends on the problem and the device's parameters. The device is not an ideal oracle: its limitations, with regard to precision, connectivity, and number of variables, must be considered to achieve the best possible results. As is customary, we rely on the classical procedure known as embedding to adapt the connectivity of the solver to the problem at hand. This procedure is described in a number of quantum annealing papers [6, 11]. During this procedure, two or more variables can be forced to take on the same value by including additional constraints in the model. In the underlying Ising model, this is achieved by introducing a large ferromagnetic (negative) coupling

J_F between two spins. The embedding process modifies the QUBO problem accordingly and one should not confuse the *logical* QUBO problem value, which depends on the QUBO problem and the state considered, with the Ising problem energy seen by the optimizer (which additionally depends on the extra constraints and the solver's parameters, such as J_F).

We distinguish between the *optimization* version of the JSP, in which we seek a valid schedule with a minimal makespan, and the *decision* version, which is limited to validating whether or not a solution exists with a makespan smaller than or equal to a user-specified timespan T . We focus exclusively on the decision version and later describe how to implement a full optimization version based on a binary search. We note that the decision formulation where jobs are constrained to fixed time windows is sometimes referred in the literature as the job-shop CSP formulation [16, 17], and our study will refer to those instances where the jobs share a common deadline T .

II. QUBO PROBLEM FORMULATION

While there are several ways the JSP can be formulated, such as the rank-based formulation [18] or the disjunctive formulation [19], our formulation is based on a straightforward time-indexed representation particularly amenable to quantum annealers (a comparative study of mappings for planning and scheduling problems can be found in [10]). We assign a set of binary variables for each operation, corresponding to the various possible discrete starting times the operation can have:

$$x_{i,t} = \begin{cases} 1 & : \text{operation } O_i \text{ starts at time } t, \\ 0 & : \text{otherwise.} \end{cases} \quad (2)$$

Here t is bounded from above by the timespan T , which represents the maximum time we allow for the jobs to complete. The timespan itself is bounded from above by the total work of the problem, that is, the sum of the execution times of all operations.

A. Constraints

We account for the various constraints by adding penalty terms to the QUBO problem. For example, an operation must start once and only once, leading to the constraint and associated penalty function

$$\left(\sum_t x_{i,t} = 1 \text{ for each } i \right) \rightarrow \sum_i \left(\sum_t x_{i,t} - 1 \right)^2. \quad (3)$$

There can only be one job running on each machine at any given point in time, which expressed as quadratic constraints yields

$$\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} = 0 \text{ for each } m, \quad (4)$$

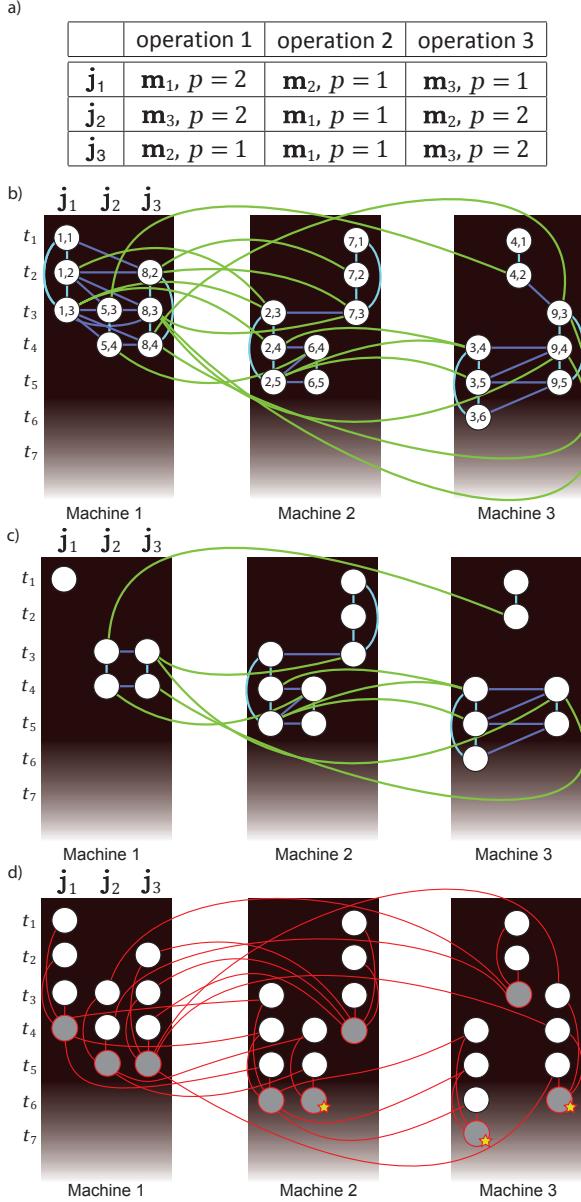


FIG. 1: a) Table representation of an example 3×3 instance whose execution times have been randomly selected to be either 1 or 2 time units. b) Pictorial view of the QUBO mapping of the above example for $H_T=6$. Green, purple, and cyan edges refer respectively to h_1 , h_2 , and h_3 quadratic coupling terms (Eqs. 7–9). Each circle represents a bit with its i, t index as in Eq. 2. c) The same QUBO problem as in (b) after the variable pruning procedure detailed in the section on QUBO formulation refinements. Isolated qubits are bits with fixed assignments that can be eliminated from the final QUBO problem. d) The same QUBO problem as in (b) for $H_T=7$. Previously displayed edges in the above figure are omitted. Red edges/circles represent the variations with respect to $H_T=6$. Yellow stars indicate the bits which are penalized with local fields for timespan discrimination.

where $R_m = A_m \cup B_m$ and

$$\begin{aligned} A_m &= \{(i, t, k, t') : (i, k) \in I_m \times I_m, \\ &\quad i \neq k, 0 \leq t, t' \leq T, 0 < t' - t < p_i\}, \\ B_m &= \{(i, t, k, t') : (i, k) \in I_m \times I_m, \\ &\quad i < k, t' = t, p_i > 0, p_j > 0\}. \end{aligned}$$

The set A_m is defined so that the constraint forbids operation O_j from starting at t' if there is another operation O_i still running, which happens if O_i started at time t and $t' - t$ is less than p_i . The set B_m is defined so that two jobs cannot start at the same time, unless at least one of them has an execution time equal to zero. Finally, the order of the operations within a job are enforced with

$$\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{i,t} x_{i+1,t'} \quad \text{for each } n, \quad (5)$$

which counts the number of precedence violations between consecutive operations only.

The resulting classical objective function (Hamiltonian) is given by

$$H_T(\bar{x}) = \eta h_1(\bar{x}) + \alpha h_2(\bar{x}) + \beta h_3(\bar{x}), \quad (6)$$

where

$$h_1(\bar{x}) = \sum_n \left(\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{i,t} x_{i+1,t'} \right), \quad (7)$$

$$h_2(\bar{x}) = \sum_m \left(\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} \right), \quad (8)$$

$$h_3(\bar{x}) = \sum_i \left(\sum_t x_{i,t} - 1 \right)^2, \quad (9)$$

and the penalty constants η , α , and β are required to be larger than 0 to ensure that unfeasible solutions do not have a lower energy than the ground state(s). As expected for a decision problem, we note that the minimum of H_T is 0 and it is only reached if a schedule satisfies all of the constraints. The index of H_T explicitly shows the dependence of the Hamiltonian on the timespan T , which affects the number of variables involved. Figure 1-b illustrates the QUBO problem mapping for $H_T=6$ for a particular 3×3 example (Figure 1-a).

B. Simple variable pruning

Figure 1-b also reveals that a significant number of the NMT binary variables required for the mapping can be pruned by applying simple restrictions on the time index t (whose computation is polynomial as the system size increases and therefore trivial here). Namely, we can define

an effective release time for each operation corresponding to the sum of the execution times of the preceding operations in the same job. A similar upper bound corresponding to the timespan minus all of the execution times of the subsequent operations of the same job can be set. The bits corresponding to these invalid starting times can be eliminated from the QUBO problem altogether since any valid solution would require them to be strictly zero. This simplification eliminates an estimated number of variables equal to $NM(M\langle p \rangle - 1)$, where $\langle p \rangle$ represents the average execution time of the operations. This result can be generalized to consider the previously defined ratio θ , such that the total number of variables required after this simple QUBO problem pre-processing is $\theta NM[T - \theta M\langle p \rangle + 1]$.

III. III. QUBO FORMULATION REFINEMENTS

Although the above formulation proves sufficient for running JSPs on the D-Wave machine, we explore a few potential refinements. The first pushes the limit of simple variable pruning by considering more advanced criteria for reducing the possible execution window of each task. The second refinement proposes a compromise between the decision version of the JSP and a full optimization version.

A. Window shaving

In the time-index formalism, reducing the execution windows of operations (i.e., shaving) [20], or in the disjunctive approach, adjusting the *heads* and *tails* of operations [21, 22], or more generally, by applying constraints propagation techniques (e.g. [23]), together constitute the basis for a number of classical approaches to solving the JSP. Shaving is sometimes used as a pre-processing step or as a way to obtain a lower bound on the makespan before applying other methods. The interest from our perspective is to showcase how such classical techniques remain relevant, without straying from our quantum annealing approach, when applied to the problem of pruning as many variables as possible. This enables larger problems to be considered and improves the success rate of embeddability in general (see Figure 3), without significantly affecting the order of magnitude of the overall time to solution in the asymptotic regime. Further immediate advantages of reducing the required number of qubits become apparent during the compilation of JSP instances for the D-Wave device due to the associated embedding overhead that depends directly on the number of variables. The shaving process is typically handled by a classical algorithm whose worst-case complexity remains polynomial. While this does not negatively impact the fundamental complexity of solving JSP instances, for *pragmatic* benchmarking the execution time needs to be taken into account and added to the quantum annealing

runtime to properly report the time to solution of the whole algorithm.

Different elimination rules can be applied. We focus herein on the iterated Carlier and Pinson (ICP) procedure [21] reviewed in the appendix with worst-case complexity given by $\mathcal{O}(N^2 M^2 T \log(N))$. Instead of looking at the one-job sub-problems and their constraints to eliminate variables, as we did for the simple pruning, we look at the one-machine sub-problems and their associated constraints to further prune variables. An example of the resulting QUBO problem is presented in Figure 1-c.

B. Timespan discrimination

We explore a method of extracting more information regarding the actual optimal makespan of a problem within a single call to the solver by breaking the degeneracy of the ground states and spreading them over some finite energy scale, distinguishing the energy of valid schedules on the basis of their makespan. Taken to the extreme, this approach would amount to solving the full optimization problem. We find that the resulting QUBO problem is poorly suited to a solver with limited precision, so a balance must be struck between extra information and the precision requirement. A systematic study of how best to balance the amount of information obtained versus the extra cost will be the subject of future work.

We propose to add a number of linear terms, or local fields, to the QUBO problem to slightly penalize valid solutions with larger makespans. We do this by adding a cost to the last operation of each job, that is, the set $\{O_{k_1}, \dots, O_{k_N}\}$. At the same time, we require that the new range of energy over which the feasible solutions are spread stays within the minimum logical QUBO problem's gap given by $\Delta E = \min\{\eta, \alpha, \beta\}$. If the solver's precision can accommodate K distinguishable energy bins, then makespans within $[T - K, T]$ can be immediately identified from their energy values. The procedure is illustrated in Figure 1-d and some implications are discussed in the appendix.

IV. IV. ENSEMBLE PRE-CHARACTERIZATION AND COMPILED

We now turn to a few important elements of our computational strategy for solving JSP instances. We first show how a careful pre-characterization of classes of random JSP instances, representative of the problems to be run on the quantum optimizer, provides very useful information regarding the shape of the search space for \mathcal{T} . We then describe how instances are compiled to run on the actual hardware.

A. Makespan Estimation

In Figure 2, we show the distributions of the optimal makespans \mathcal{T} for different ensembles of instances parametrized by their size $N = M$, by the possible values of task durations $\mathcal{P}_p = \{p_{\min}, \dots, p_{\max}\}$, and by the ratio $\theta \leq 1$ of the number of machines used by each job. Instances are generated randomly by selecting θM distinct machines for each job and assigning an execution time to each operation randomly. For each set of parameters, we can compute solutions with a classical exhaustive solver in order to identify the median of the distribution $\langle \mathcal{T} \rangle(N, \mathcal{P}_p, \theta)$ as well as the other quantiles. These could also be inferred from previously solved instances with the proposed annealing solver. The resulting information can be used to guide the binary search required to solve the optimization problem. Figure 2 indicates that a normal distribution is an adequate approximation, so we need only to estimate its average $\langle \mathcal{T} \rangle$ and variance σ^2 . Interestingly, from the characterization of the families of instances up to $N = 10$ we find that, at least in the region explored, the average minimum makespan $\langle \mathcal{T} \rangle$ is proportional to the average execution time of a job $\langle p \rangle \theta N$, where $\langle p \rangle$ is the mean of \mathcal{P}_p . This linear ansatz allows for the extrapolation of approximate resource requirements for classes of problems which have not yet been pre-characterized, and it constitutes an educated guess for classes of problems which cannot be pre-characterized due to their difficulty or size. The accuracy of these functional forms was verified by computing the relative error of the prediction versus the fit of the makespan distribution of each parametrized family up to $N = M = 9$ and $p_{\max} = 20$ using 200 instances to compute the makespan histogram. The prediction for $\langle \mathcal{T} \rangle$ results are consistently at least 95% accurate, while the one for σ has at worst a 30% error margin, a very approximate but sufficient model for the current purpose of guiding the binary search.

B. Compilation

The graph-minor embedding technique (abbreviated simply “embedding”) represents the de facto method of recasting the Ising problems to a form compatible with the layout of the annealer’s architecture [24, 25], which for the D-Wave Two is a Chimera graph [1]. Formally, we seek an isomorphism between the problem’s QUBO graph and a graph minor of the solver. This procedure has become a standard in solving applied problems using quantum annealing [6, 11] and can be thought of as the analogue of compilation in a digital computer programming framework during which variables are assigned to hardware registers and memory locations. This process is covered in more details in the appendix. An example of embedding for a 5×5 JSP instance with $\theta = 1$ and $T = 7$ is shown in Figure 3-a, where the 72 logical variables of the QUBO problem are embedded using 257 qubits of the

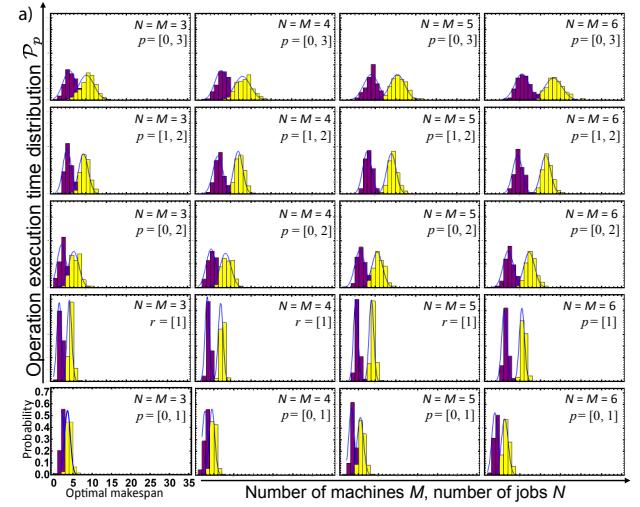


FIG. 2: a) Normalized histograms of optimal makespans \mathcal{T} for parametrized families of JSP instances with $N = M$, \mathcal{P}_p on the y-axis, $\theta = 1$ (yellow), and $\theta = 0.5$ (purple). The distributions are histograms of occurrences for 1000 random instances, fitted with a Gaussian function of mean $\langle \mathcal{T} \rangle$. We note that the width of the distributions increases as the range of the execution times \mathcal{P}_p increases, for fixed $\langle p \rangle$. The mean and the variance are well fitted respectively by $\langle \mathcal{T} \rangle = A_{\mathcal{T}} N p_{\min} + B_{\mathcal{T}} N p_{\max}$ and $\sigma = \sigma_0 + C_{\sigma} \langle \mathcal{T} \rangle + A_{\sigma} p_{\min} + B_{\sigma} p_{\max}$, with $A_{\mathcal{T}} = 0.67$, $B_{\mathcal{T}} = 0.82$, $\sigma_0 = 0.7$, $A_{\sigma} = -0.03$, $B_{\sigma} = 0.43$, and $C_{\sigma} = 0.003$.

Chimera graph. Finding the optimal tiling that uses the fewest qubits is NP-hard [26], and the standard approach is to employ heuristic algorithms [27]. In general, for the embedding of time-indexed mixed-integer programming QUBO problems of size N into a graph of degree k , one should expect a quadratic overhead in the number of binary variables on the order of aN^2 , with $a \leq (k-2)^{-1}$ depending on the embedding algorithm and the hardware connectivity [11]. This quadratic scaling is apparent in Figure 3-b where we report on the compilation attempts using the algorithm in [27]. Results are presented for the D-Wave chip installed at NASA Ames at the time of this study, for a larger chip with the same size of Chimera block and connectivity pattern (like the latest chip currently being manufactured by D-Wave Systems), and for a speculative yet-larger chip where the Chimera block is twice as large. We deem a JSP instance embeddable when the respective $H_{T=\mathcal{T}}$ is embeddable, so the decrease in probability of embedding with increasing system size is closely related to the shift and spreading of the optimal makespan distributions for ensembles of increasing size (see Figure 2). What we observe is that, with the available algorithms, the current architecture admits embedded JSP instances whose total execution time $N M \theta \langle p \rangle$ is around 20 time units, while near-future (we estimate 2 years) D-Wave chip architectures could potentially double that. As noted in similar studies (e.g., [6]), graph connectivity has a much more dramatic impact on embeddability than qubit count.

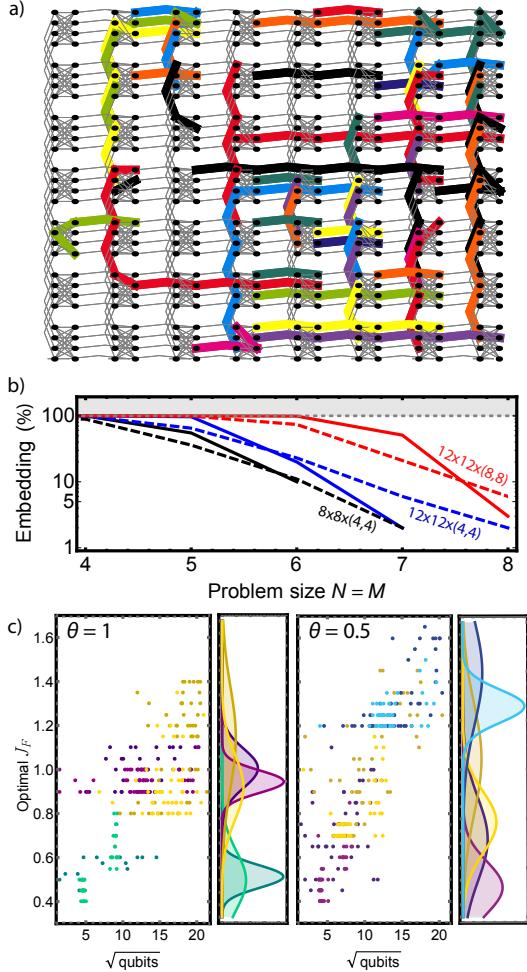


FIG. 3: a) Example of an embedded JSP instance on NASA’s D-Wave Two chip. Each chain of qubits is colored to represent a logical binary variable determined by the embedding. For clarity, active connections between the qubits are not shown. b) Embedding probability as a function of $N = M$ for $\theta = 1$ (similar results are observed for $\theta = 0.5$). Solid lines refer to $\mathcal{P}_p = [1, 1]$ and dashed lines refer to $\mathcal{P}_p = [0, 2]$. 1000 random instances have been generated for each point, and a cutoff of 2 minutes has been set for the heuristic algorithm to find a valid topological embedding. Results for different sizes of Chimera are shown. c) Optimal parameter-setting analysis for the ensembles of JSP instances we studied. Each point corresponds to the number of qubits and the optimal J_F (see main text) of a random instance, and each color represents a parametrized ensemble (green: 3×3 , purple: 4×4 , yellow: 5×5 , blue: 6×6 ; darker colors represent ensembles with $\mathcal{P}_p = [1, 1]$ as opposed to lighter colors which indicate $\mathcal{P}_p = [0, 2]$). Distributions on the right of scatter plots represent Gaussian fits of the histogram of the optimal J_F for each ensemble. Runtime results are averaged over an ungauged run and 4 additional runs with random gauges [28].

Once the topological aspect of embedding has been solved, we set the ferromagnetic interactions needed to adapt the connectivity of the solver to the problem at hand. For the purpose of this work, this should be re-

garded as a technicality necessary to tune the performance of the experimental analog device and we include the results for completeness. Introductory details about the procedure can be found in [6, 11]. In Figure 3-c we show a characterization of the ensemble of JSP instances (parametrized by N , M , θ , and \mathcal{P}_p , as described at the beginning of this section). We present the best ferromagnetic couplings found by runs on the D-Wave machine under the simplification of a uniform ferromagnetic coupling by solving the embedded problems with values of J_F from 0.4 to 1.8 in relative energy units of the largest coupling of the original Ising problem. The run parameters used to determine the best J_F are the same as we report in the following sections, and the problem sets tested correspond to Hamiltonians whose timespan is equal to the sought makespan $H_{T=\mathcal{T}}$. This parameter-setting approach is similar to the one followed in [6] for operational planning problems, where the instance ensembles were classified by problem size before compilation. What emerges from this preliminary analysis is that each parametrized ensemble can be associated to a distribution of optimal J_F that can be quite wide, especially for the ensembles with $p_{\min} = 0$ and large p_{\max} . This spread might discourage the use of the mean value of such a distribution as a predictor of the best J_F to use for the embedding of new untested instances. However, the results from this predictor appear to be better than the more intuitive prediction obtained by correlating the number of qubits after compilation with the optimal J_F . This means that for the D-Wave machine to achieve optimal performance on structured problems, it seems to be beneficial to use the information contained in the structure of the logical problem to determine the best parameters. We note that this “offline” parameter-setting could be used in combination with “online” performance estimation methods such as the ones described in [28] in order to reach the best possible instance-specific J_F with a series of quick experimental runs. The application of these techniques, together with the testing of alternative offline predictors, will be the subject of future work.

V. V. RESULTS OF TEST RUNS AND DISCUSSION

A complete quantum annealing JSP solver designed to solve an instance to optimality using our proposed formulation will require the independent solution of several embedded instances $\{H_T\}$, each corresponding to a different timespan T . Assuming that the embedding time, the machine setup time, and the latency between subsequent operations can all be neglected, due to their being non-fundamental, the running time T of the approach for a specific JSP instance reduces to the expected *total* annealing time necessary to find the optimal solution of each embedded instance with a specified minimum target probability $\simeq 1$. The probability of ending the annealing cycle in a desired ground state depends, in an essen-

tially unknown way, on the embedded Ising Hamiltonian spectrum, the relaxation properties of the environment, the effect of noise, and the annealing profile. Understanding through an ab initio approach what is the best computational strategy appears to be a formidable undertaking that would require theoretical breakthroughs in the understanding of open-system quantum annealing [29, 30], as well as a tailored algorithmic analysis that could take advantage of the problem structure that the annealer needs to solve. For the time being, and for the purposes of this work, it seems much more practical to limit these early investigations to the most relevant instances, and to lay out empirical procedures that work under some general assumptions. More specifically, we focus on solving the CSP version of JSP, not the full optimization problem, and we therefore only benchmark the Hamiltonians with $\mathcal{T} = T$ with the D-Wave machine. We note however that a full optimization solver can be realized by leveraging data analysis of past results on parametrized ensembles and by implementing an adaptive binary search. Full details can be found in a longer version of this work [31].

On the quantum annealer installed at NASA Ames (it has 509 working qubits; details are presented in [32]), we run hundreds of instances, sampling the ensembles $N = M \in \{3, 4, 5, 6\}$, $\theta \in \{0.5, 1\}$, and $P_p \in \{[1, 1], [0, 2]\}$. For each instance, we report results, such as runtimes, at the most optimal J_F among those tested, assuming the application of an optimized parameter-setting procedure along the lines of that described in the previous section. Figure 4-a displays the total annealing repetitions required to achieve a 99% probability of success on the ground state of $H_{\mathcal{T}}$, with each repetition lasting $t_A = 20$ μ s, as a function of the number of qubits in the embedded (and pruned) Hamiltonian. We observe an exponential increase in complexity with increasing Hamiltonian size, for both classes of problems studied. This likely means that while the problems tested are small, the analog optimization procedure intrinsic to the D-Wave device's operation is already subject to the fundamental complexity bottlenecks of the JSP. It is, however, premature to draw conclusions about performance scaling of the technology given the current constraints on calibration procedures, annealing time, etc. Many of these problems are expected to be either overcome or nearly so with the next generation of D-Wave chip, at which point more extensive experimentation will be warranted.

In Figure 4-b, we compare the performance of the D-Wave device to two exhaustive classical algorithms in order to gain insight on how current quantum annealing technology compares with paradigmatic classical optimization methods. Leaving the performance of approximate solutions for future work, we chose not to explore the plethora of possible heuristic methods as we operate the D-Wave machine, seeking the global optimum.

The first algorithm, B, detailed in [35], exploits the disjunctive graph representation and a branch-and-bound strategy that very effectively combines a branching

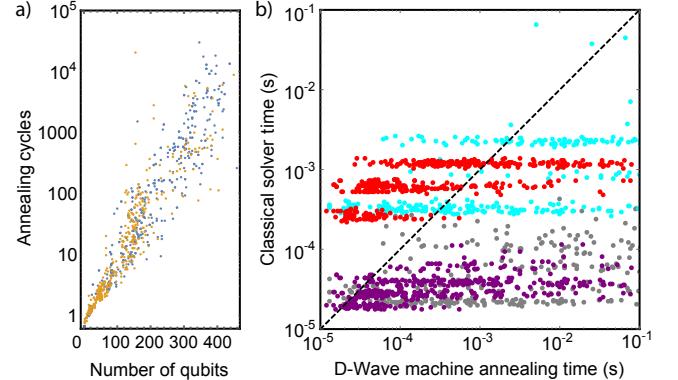


FIG. 4: a) Number of repetitions required to solve $H_{\mathcal{T}}$ with the D-Wave Two with a 99% probability of success (see the appendix). The blue points indicate instances with $\theta = 1$ and yellow points correspond to $\theta = 0.5$ (they are the same instances and runtimes used for Figure 3-c). The number of qubits on the x-axis represents the qubits used after embedding. b) Correlation plot between classical solvers and the D-Wave optimizer. Gray and violet points represent runtimes compared with algorithm B, and cyan and red are compared to the MS algorithm, respectively, with $\theta = 1$ and $\theta = 0.5$. All results presented correspond to the best out of 5 gauges selected randomly for every instance. In case the machine returns embedding components whose values are discordant, we apply a majority voting rule to recover a solution within the logical subspace [6, 11, 28, 33, 34]. We observe a deviation of about an order of magnitude on the annealing time if we average over 5 gauges instead of picking the best one, indicating that there is considerable room for improvement if we were to apply more-advanced calibration techniques [32].

scheme based on selecting the direction of a single disjunctive edge (according to some single-machine constraints), and a technique introduced in [36] for fixing additional disjunctions (based on a preemptive relaxation). It has publicly available code and is considered a well-performing complete solver for the small instances currently accessible to us, while remaining competitive for larger ones even if other classical approaches become more favorable [37]. B has been used in [38] to discuss the possibility of a phase transition in the JSP, demonstrating that the random instances with $N = M$ are particularly hard families of problems, not unlike what is observed for the quantum annealing implementation of planning problems based on graph vertex coloring [9].

The second algorithm, MS, introduced in [20], proposes a time-based branching scheme where a decision is made at each node to either schedule or delay one of the available operations at the current time. The authors then rely on a series of shaving procedures such as those proposed by [21] to determine the new bound and whether the choice leads to valid schedules. This algorithm is a natural comparison with the present quantum annealing approach as it solves the decision version of the JSP in a very similar fashion to the time-indexed formulation we have implemented on the D-Wave machine, and it makes

use of the same shaving technique that we adapted as a pre-processing step for variable pruning. However, we should mention that the variable pruning that we implemented to simplify H_T is employed at each node of the classical branch and bound algorithm, so the overall computational time of MS is usually much more important than our one-pass pre-processing step, and in general its runtime does not scale polynomially with the problem size.

What is apparent from the correlation plot in Figure 4-b is that the D-Wave machine is easily outperformed by a classical algorithm run on a modern single-core processor, and that the problem sizes tested in this study are still too small for the asymptotic behavior of the classical algorithms to be clearly demonstrated and measured. The comparison between the D-Wave machine’s solution time for H_T and the full optimization provided by B is confronting two very different algorithms, and shows that B solves all of the full optimization problems that have been tested within milliseconds, whereas D-Wave’s machine can sometimes take tenths of a second (before applying the multiplier factor $\simeq 2$, due to the binary search; see the appendix). When larger chips become available, however, it will be interesting to compare B to a quantum annealing solver for sizes considered B-intractable due to increasing memory and time requirements.

The comparison with the MS method has a promising signature even now, with roughly half of the instances being solved by D-Wave’s hardware faster than the MS algorithm (with the caveat that our straightforward implementation is not fully optimized). Interestingly, the different parametrized ensembles of problems have distinctively different computational complexity characterized by well-recognizable average computational time to solution for MS (i.e., the points are “stacked around horizontal lines” in Figure 4-b), whereas the D-Wave machine’s complexity seems to be sensitive mostly to the total qubit count (see Figure 4-a) irrespective of the problem class. We emphasize again that conclusions on speedup and asymptotic advantage still cannot be confirmed until improved hardware with more qubits and less noise becomes available for empirical testing.

VI. CONCLUSIONS

Although it is probable that the quantum annealing-based JSP solver proposed herein will not prove competitive until the arrival of an annealer a few generations away, the implementation of a provably tough application from top to bottom was missing in the literature, and our work has led to noteworthy outcomes we expect will pave the way for more advanced applications of quantum annealing. Whereas part of the attraction of quantum annealing is the possibility of applying the method irrespective of the structure of the QUBO problem, we have shown how to design a quantum annealing

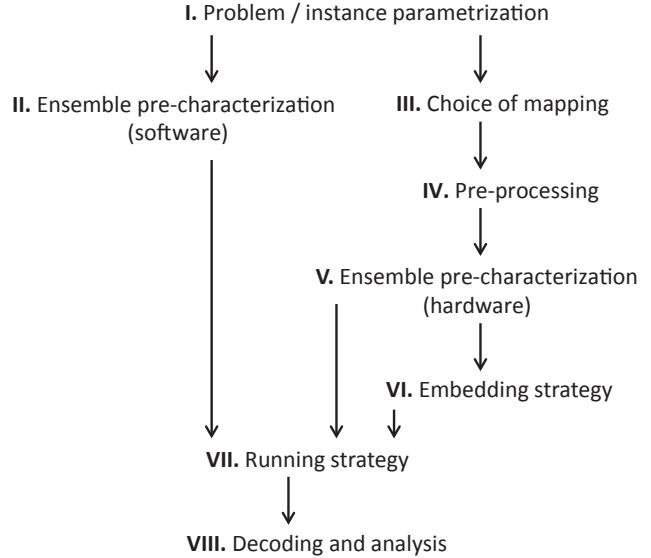


FIG. 5: I-II) Appropriate choice of benchmarking and classical simulations is discussed in Section IV. III-IV) Mapping to QUBO problems is discussed in Sections II and III. V-VI) Pre-characterization for parameter setting is described in Section VI. VII) Structured run strategies adapted to specific problems have not to our knowledge been discussed before. We discuss a prescription in the appendix. VIII) The only decoding required in our work is majority voting within embedding components to recover error-free logical solutions. The time-indexed formulation then provides QUBO problem solutions that can straightforwardly be represented as Gantt charts of the schedules.

solver, mindful of many of the peculiarities of the annealing hardware and the problem at hand, for improved performance. Figure 5 shows a schematic view of the streamlined solving process describing a full JSP optimization solver. The pictured scheme is not intended to be complete, for example, the solving framework can benefit from other concepts such as performance tuning techniques [28] and error-correction repetition lattices [39]. The use of the decision version of the problem can be combined with a properly designed search strategy (the simplest being a binary search) in order to be able to seek the minimum value of the common deadline of feasible schedules. The proposed timespan discrimination further provides an adjustable compromise between the full optimization and decision formulations of the problems, allowing for instant benefits from future improvements in precision without the need for a new formulation or additional binary variables to implement the makespan minimization as a term in the objective function. As will be explored further in future work, we found that instance pre-characterization performed to fine tune the solver parameters can also be used to improve the search strategy, and that it constitutes a tool whose use we expect to become common practice in problems amenable to CSP formulations as the ones proposed for the JSP. Additionally, we have shown that there is great potential

in adapting classical algorithms with favorable polynomial scaling as pre-processing techniques to either prune variables or reduce the search space. Hybrid approaches and metaheuristics are already fruitful areas of research and ones that are likely to see promising developments with the advent of these new quantum heuristics algorithms.

A. Acknowledgements

The authors would like to thank J. Frank, M. Do, E.G. Rieffel, B. O’Gorman, M. Bucyk, P. Haghnegahdar, and other researchers at QuAIL and 1QBit for useful input and discussions. This research was supported by 1QBit, Mitacs, NASA (Sponsor Award Number NNX12AK33A), and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IAA 145483 and by the AFRL Information Directorate under grant F4HBKC4162G001.

Appendix A: JSP and QUBO formulation

In this appendix we expand on the penalty form used for the constraints and the alternative reward-base formulation as well as the timespan discrimination scheme.

1. Penalties versus rewards formulation

The encoding of constraints as terms in a QUBO problem can either reward the respecting of these constraints or penalize their violation. Although the distinction may at first seem artificial, the actual QUBO problem generated differs and can lead to different performance on an imperfect annealer. We present one such alternative formulation where the precedence constraint (7) is instead encoded as a reward for correct ordering by replacing $+\eta h_1(\bar{x})$ with $-\eta' h'_1(\bar{x})$, where

$$h'_1(\bar{x}) = \sum_n \left(\sum_{\substack{k_{n-1} < i < k_n \\ t+p_i \leq t'}} x_{i,t} x_{i+1,t'} \right). \quad (\text{A1})$$

The new Hamiltonian is

$$H'_T(\bar{x}) = -\eta' h'_1(\bar{x}) + \alpha h_2(\bar{x}) + \beta h_3(\bar{x}). \quad (\text{A2})$$

The reward attributed to a solution is equal to η' times the number of satisfied precedence constraints. A feasible solution, where all constraints are satisfied, will have energy equal to $-\eta'(k_N - N)$.

The functions h_1 and h'_1 differ only by the range of t' : in the rewards version we have

$$t' - t \geq p_i,$$

and in the penalties version we have

$$t' - t < p_i.$$

The fact that we are allowing equality in the rewards version means that h'_1 will always have more quadratic terms than h_1 regardless of variable pruning, leading to a more connected QUBO graph and therefore a harder problem to embed.

Another important disadvantage is revealed when choosing the coefficients η' , α , and β in H'_T to guarantee that no unfeasible solution has energy less than $-\eta'(k_N - N)$. This can happen if the penalty that we gain from breaking constraints h_2 or h_3 is less than the potential reward we get from h'_1 . The penalty-based formulation simply requires that η , α , and β be larger than 0. The following lemma summarizes the equivalent condition for the reward-based case.

Lemma 1. If $\beta/\eta' \geq 3$ and $\alpha > 0$, then

$$H'_T(\bar{x}) \geq -(k_N - N), \quad (\text{A3})$$

for all \bar{x} , with equality if and only if \bar{x} represents a feasible schedule.

We also found examples that show that these bounds on the coefficients are tight.

The fact that β/η' must be greater than or equal to 3 is a clear disadvantage because of the issues with precision of the current hardware. In H_T we can set all of the penalty coefficients (and hence all non-zero couplers) to be equal, which is the best possible case from the point of view of precision.

2. Timespan discrimination

The timespan discrimination that we propose is a specification to strike a compromise between the information obtained from each solver call, and the required precision for this information to be accurate and obtained efficiently. Specifically, we want this extra information to help identify the optimal makespan by looking at the energy of the solutions. This means breaking the degeneracy of the ground states (i.e., the valid solutions) and assigning different energy sectors to different makespans. To prevent collisions with invalid solutions, these energy sectors have to fit within the logical QUBO problem’s gap given by $\Delta E = \min\{\eta, \alpha, \beta\}$. We note that this will affect the actual gap (as seen by the hardware) of the embedded Ising model.

Since the binary variables we have defined in the proposed formulation are not sufficient to write a simple expression for the makespan of a given solution, additional auxiliary variables and associated constraints would need to be introduced. Instead, a simple way to implement this feature in our QUBO formulation is to add a number of local fields to the binary variables corresponding to the last operation of each job, $\{O_{k_1}, \dots, O_{k_N}\}$. Since

the makespan depends on the completion time of the last operation, the precedence constraint guarantees that the makespan of a valid solution will be equal to the completion time of one of those operations. We can then select the local field appropriately as a function of the time index t to penalize a fixed number K of the larger makespans ranging from $T - K + 1$ to T . Within a sector assigned to the time step \mathcal{T} , we need to further divide $\Delta E_{\mathcal{T}}$ by the maximum number of operations that can complete at \mathcal{T} to obtain the largest value we can use as the local field $h_{\mathcal{T}}$, i.e., the number of distinct machines used by at least one operation in the set of operations $\{O_{k_1}, \dots, O_{k_N}\}$, denoted by M_{final} . If K is larger than 1, we also need to ensure that contributions from various sectors can be differentiated. The objective is to assign a distinct T -dependent energy range to all valid schedules with makespans within $[T - K, T]$. More precisely, we relate the local fields for various sectors with the recursive relation

$$h_{\mathcal{T}-1} = \frac{h_{\mathcal{T}}}{M_{\text{final}}} + \epsilon, \quad (\text{A4})$$

where ϵ is the minimum logical energy resolvable by the annealer. Considering that ϵ is also the minimum local field we can use for $h_{\mathcal{T}-K+1}$ and that the maximum total penalty we can assign through this time-discrimination procedure is $\Delta E - \epsilon$, it is easy to see that the energy resolution should scale as $\Delta E/(M_{\text{final}}^K)$. An example of the use of local fields for timespan discrimination is shown in Figure 1-d of the main text for the case $K = 1$.

Appendix B: Computational strategy

This appendix elaborates on the compilation methods and the quantum annealing implementation of a full optimization solver based on the decision version and a binary search as outlined in the main text.

1. Compilation

The process of compiling, or *embedding*, an instance for a specific target architecture is a crucial step given the locality of the programmable interactions on current quantum annealer architectures. During the graph-minor topological embedding, each vertex of the problem graph is mapped to a subset of connected vertices, or subgraph, of the hardware graph. These assignments must be such that the edges in the problem graph have at least one corresponding edge between the associated subgraphs in the hardware graph. Formally, the classical Hamiltonian Eq. (6) is mapped to a quantum annealing Ising Hamiltonian on the hardware graph using the set of equations that follows. The spin operators $s\vec{\sigma}_i$ are defined by setting $s = 1$ and using the Pauli matrices to write $\vec{\sigma}_i = (\sigma_i^x, \sigma_i^y, \sigma_i^z)$. The resulting spin variables $\sigma_i^z = \pm 1$,

our qubits, are easily converted to the usual binary variables $x_i = 0, 1$ with $\sigma_i^z = 2x_i - 1$. The Ising Hamiltonian is given by

$$H = A(t)[H_Q + H_E] + B(t)H_D, \quad (\text{B1})$$

$$H_Q = \sum_{ij} J_{ij} \sigma_{\alpha_i}^z \sigma_{\beta_j}^z |_{(\alpha_i, \beta_j) \in E(i, j)} + \sum_{\substack{i \\ k \in V(i)}} \frac{h_i}{N_{V(i)}} \sigma_k^z, \quad (\text{B2})$$

$$H_E = - \sum_{\substack{i \\ (k, k') \in E(i, i)}} J_{i,k,k'}^F \sigma_k^z \sigma_{k'}^z, \quad (\text{B3})$$

$$H_D = \sum_{\substack{i \\ k \in V(i)}} \sigma_k^x, \quad (\text{B4})$$

where for each logical variable index i we have a corresponding ensemble of qubits given by the set of vertices $V(i)$ in the hardware graph with $|V(i)| = N_{V(i)}$. Edges between logical variables are denoted $E(i, j)$ and edges within the subgraph of $V(i)$ are denoted $E(i, i)$. The couplings J_{ij} and local fields h_i represent the logical terms obtained after applying the linear QUBO-Ising transformation to Eq. (6). $J_{i,k,k'}^F$ are embedding parameters for vertex $V(i)$ and $(k, k') \in E(i, i)$ (see discussion below on the ferromagnetic coupling). The equation above assumes that a local field h_i is distributed uniformly between the vertices $V(i)$ and the coupling $J_{i,j}$ is attributed to a single randomly selected edge (α_i, β_j) among the available couplers $E(i, j)$, but other distributions can be chosen. In the actual hardware implementation we rescale the Hamiltonian by dividing by J_F , which is the value assigned to all $J_{i,k,k'}^F$, as explained below. This is due to the limited range of J_{ij} and h_i allowed by the machine [11].

Once a valid embedding is chosen, the ferromagnetic interactions $J_{i,k,k'}^F$ in Eq. (B3) need to be set appropriately. While the purpose of these couplings is to penalize states for which $\langle \sigma_k^z \rangle \neq \langle \sigma_{k'}^z \rangle$ for $k, k' \in V(i)$, setting them to a large value negatively affects the performance of the annealer due to the finite energy resolution of the machine (given that all parameters must later be rescaled to the actual limited parameter range of the solver) and the slowing down of the dynamics of the quantum system associated with the introduction of small energy gaps. There is guidance from research in physics [11, 40] and mathematics [41] on which values could represent the optimal $J_{i,k,k'}^F$ settings, but for application problems it is customary to employ empirical prescriptions based on pre-characterization [6] or estimation techniques of performance [28].

Despite embedding being a time-consuming classical computational procedure, it is usually not considered part of the computation and its runtime is not measured in determining algorithmic complexity. This is because we can assume that for parametrized families of problems one could create and make available a portfolio of embeddings that are compatible with all instances belonging to a given family. The existence of a such a library would reduce the computational cost to a simple query in a lookup

table, but this could come at the price of the available embedding not being fully optimized for the particular problem instance.

2. Quantum annealing optimization solver

We now detail our approach to solving individual JSP instances. We shall assume the instance at hand can be identified as belonging to a pre-characterized family of instances for minimal computational cost. This can involve identifying N , M , and θ , as well as the approximate distribution of execution times for the operations. The pre-characterization is assumed to include a statistical distribution of optimal makespans as well as the appropriate solver parameters (J_F , optimal annealing time, etc.). Using this information, we need to build an ensemble of queries $\mathcal{Q} = \{\mathbf{q}\}$ to be submitted to the D-Wave quantum annealer to solve a problem H . Each element of \mathcal{Q} is a triple (t_A, R, T) indicating that the query considers R identical annealings of the embedded Hamiltonian H_T for a single annealing time t_A . To determine the elements in \mathcal{Q} we first make some assumptions, namely, i) *sufficient statistics*: for each query, R is sufficiently large to sample appropriately the ensembles defined in Eqs. (B7)–(B9); ii) *generalized adiabaticity*: t_A is optimized (over the window of available annealing times) for the best annealing performance in finding a ground state of H_T (i.e., the annealing procedure is such that the total expected annealing time $t_A R^*$ required to evolve to a ground state is as small as possible compared to the time required to evolve to an excited state, with the same probability). Both of these conditions can be achieved in principle by measuring the appropriate optimal parameters $R^*(\mathbf{q})$ and $t_A^*(\mathbf{q})$ through extensive test runs over random ensembles of instances. However, we note that verifying these assumptions experimentally is currently beyond the operational limits of the D-Wave Two device since the optimal t_A for generalized adiabaticity is expected to be smaller than the minimum programmable value [3]. Furthermore, we deemed the considerable machine time required for such a large-scale study too onerous in the context of this initial foray. Fortunately, the first limitation is expected to be lifted with the next generation of chip, at which point nothing would prevent the proper determination of a family-specific choice of R^* and t_A^* . Given a specified annealing time, the number of anneals is determined by specifying r_0 , the target probability of success for queries or confidence level, and measuring $r_{\mathbf{q}}$, the rate of occurrence of the ground state per repetition for the following query:

$$R^* = \frac{\log[1 - r_0]}{\log[1 - r_{\mathbf{q}}]}. \quad (\text{B5})$$

The rate $r_{\mathbf{q}}$ depends on the family, T , and the other parameters. The minimum observed during pre-characterization should be used to guarantee the ground state is found with at least the specified r_0 . Formally,

the estimated time to solution of a problem is then given by

$$T = \sum_{\mathbf{q} \in \mathcal{Q}} t_A \left(\frac{\log[1 - r_0]}{\log[1 - r_{\mathbf{q}}]} \right). \quad (\text{B6})$$

The total probability of success of solving the problem in time T will consequently be $\prod_{\mathbf{q}} r_{\mathbf{q}}$. For the results presented in this paper, we used $R^* = 500\,000$ and $t_A^* = \min(t_A) = 20\ \mu\text{s}$.

We can define three different sets of qubit configurations that can be returned when the annealer is queried with \mathbf{q} . \mathcal{E} is the set of configurations whose energy is larger than ΔE as defined in Section III of the paper. These configurations represent invalid schedules. \mathcal{V} is the set of solutions with zero energy, i.e., the solutions whose makespan \mathcal{T} is small enough ($\mathcal{T} \leq T - K$) that they have not been discriminated by the procedure described in the subsection on timespan discrimination. Finally, \mathcal{S} is the set of valid solutions that can be discriminated ($\mathcal{T} \in (T - K, T]$). Depending on what the timespan T of the problem Hamiltonian H_T and the optimal makespan \mathcal{T} are, the quantum annealer samples the following configuration space (reporting R samples per query):

$$T < \mathcal{T} \rightarrow \mathcal{V}, \mathcal{S} = \emptyset \rightarrow E_0 > \Delta E, \quad (\text{B7})$$

$$\mathcal{T} \in (T - K, T] \rightarrow \mathcal{V} = \emptyset \rightarrow E_0 \in (0, \Delta E], \quad (\text{B8})$$

$$\mathcal{T} \leq T - K \rightarrow \mathcal{E}, \mathcal{V}, \mathcal{S} \neq \emptyset \rightarrow E_0 = 0. \quad (\text{B9})$$

Condition (B8) is the desired case where the ground state of H_T with energy E_0 corresponds to a valid schedule with the optimal makespan we seek. The ground states corresponding to conditions (B7) and (B9) are instead, respectively, invalid schedules and valid schedules whose makespan could correspond to a global minimum (to be determined by subsequent calls). The above-described assumption ii) is essential to justify aborting the search when case (B8) is encountered. If R and t_A are appropriately chosen, the ground state will be preferentially found instead of all other solutions such that one can stop annealing reasonably soon (i.e., after a number of reads on the order of R^*) in the absence of the appearance of a zero-energy solution. We can then declare this minimum-energy configuration, found within $(0, \Delta E]$, to be the ground state and the associated makespan and schedule to be the optimal solution of the optimization problem. On the other hand, we note that if $K = 0$, a minimum of two calls are required to solve the problem to optimality, one to show that no valid solution is found for $T = \mathcal{T} - 1$ and one to show that a zero-energy configuration is found for $T = \mathcal{T}$. While for cases (B8) and (B9) the appearance of an energy less than or equal to ΔE heuristically determines the trigger that stops the annealing of H_T , for case (B7), we need to have a prescription, based on pre-characterization, on how long to anneal in order to be confident that $T < \mathcal{T}$. While optimizing these times is a research program on its own

that requires extensive testing, we expect that the characteristic time for achieving condition (B8) when $T = \mathcal{T}$ will be on the same order of magnitude for this unknown runtime.

3. Search strategy

The final important component of the computational strategy consists in determining the sequence of timespans of the calls (i.e., the ensemble \mathcal{Q}). Here we propose to select the queries based on an optimized binary search that makes informed dichotomic decisions based on the pre-characterization of the distribution of optimal makespans of the parametrized ensembles as described in the previous sections. More specifically, the search is designed based on the assumption that the JSP instance at hand belongs to a family whose makespan distribution has a normal form with average makespan $\langle \mathcal{T} \rangle$ and variance σ^2 . This fitted distribution is the same \mathcal{P}_p described in Figure 2-a of the main text whose tails have been cut off at locations corresponding to an instance-dependent upper bound T_{\max} and strict lower bound T_{\min} (see the following section on bounds).

Once the initial T_{\min} and T_{\max} are set, the binary search proceeds as follows. To ensure a logarithmic scaling for the search, we need to take into account the normal distribution of makespans by attempting to bisect the range $(T_{\min}, T_{\max}]$ such that the probability of finding the optimal makespan on either side is roughly equal. In other words, T should be selected by solving the following equation and rounding to the nearest integer:

$$\begin{aligned} \operatorname{erf}\left(\frac{T_{\max} + \frac{1}{2} - \langle \mathcal{T} \rangle}{\sigma\sqrt{2}}\right) + \operatorname{erf}\left(\frac{T_{\min} + \frac{1}{2} - \langle \mathcal{T} \rangle}{\sigma\sqrt{2}}\right) = \\ \operatorname{erf}\left(\frac{T + \frac{1}{2} - \langle \mathcal{T} \rangle}{\sigma\sqrt{2}}\right) + \operatorname{erf}\left(\frac{T - \max(1, K) + \frac{1}{2} - \langle \mathcal{T} \rangle}{\sigma\sqrt{2}}\right), \end{aligned} \quad (\text{B10})$$

where $\operatorname{erf}(x)$ is the error function. For our current purpose, an inexpensive approximation of the error function is sufficient. In most cases this condition means initializing the search at $T = \langle \mathcal{T} \rangle$. We produce a query q_0 for the annealing of H_T . If no schedule is found (condition (B7)) we simply let $T_{\min} = T$. If condition (B9) is verified, on the other hand, we update T_{\max} to be the makespan \mathcal{T} of the valid found solution (which is equal to $T - \max(1, K) + 1$ in the worst case) for the determination of the next query q_1 . The third condition (B8), only reachable if $K > 0$, indicates both that the search can stop and the problem has been solved to optimality. The search proceeds in this manner by updating the bounds and bisecting the new range at each step and stops either with condition (B8) or when $\mathcal{T} = T_{\max} = T_{\min} + 1$. Figure 6-a shows an example of such a binary search in practice. The reason for using this guided search is that the average number of calls to find the optimal makespan is dramatically reduced with respect to a linear search on the range

$(T_{\min}, T_{\max}]$. For a small variance this optimized search is equivalent to a linear search that starts near $T = \langle \mathcal{T} \rangle$. A more spread-out distribution, on the other hand, will see a clear advantage due to the logarithmic, instead of linear, scaling of the search. In Figure 6-b, we compute this average number of calls as a function of N , θ , and K for $N = M$ instances generated such that an operation's average execution time also scales with N . This last condition ensures that the variance of the makespan grows linearly with N as well, ensuring that the logarithmic behavior becomes evident for larger instances. For this calculation we use the worst case when updating T_{\max} due to condition (B9) being met. We find that for the experimentally testable instances with the D-Wave Two device (see Figure 3-b of the main text), the expected number of calls to solve the problem is less than three (in the absence of pre-characterization it would be twice that), while for larger instances the size of \mathcal{Q} scales logarithmically, as expected.

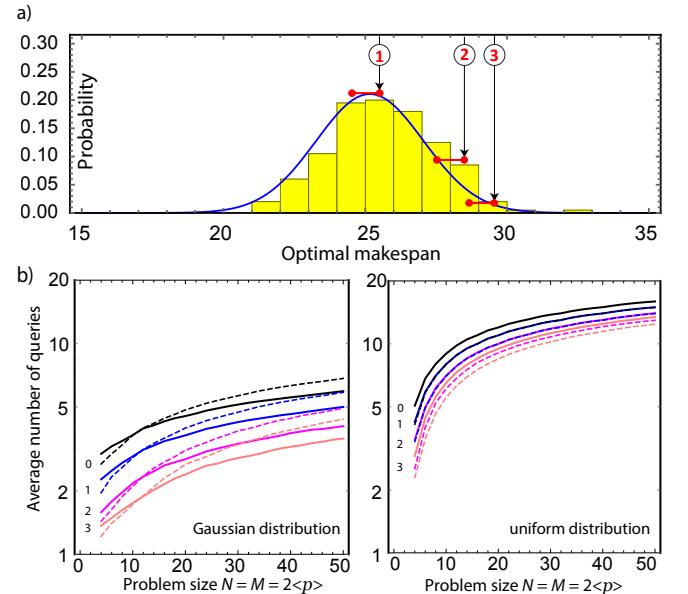


FIG. 6: a) View of a guided binary search required to identify the minimum makespan over a distribution. The fitted example distribution corresponds to $N = M = 8$, fitted to a Gaussian distribution as described in the main text. We assume $K = 1$. The first attempt queries H_{26} , the second H_{29} , and the third H_{30} (the final call), following Eq. (B10). b) Average number of calls to the quantum annealer required by the binary search assuming Eq. (B10) (left panel) or assuming a uniform distribution of minimum makespans between trivial upper and lower bounds. Thick and dashed lines correspond to $\theta = 1$ and $\theta = 0.5$, respectively, and the numeric values associated with each color in the figure correspond to different values of K . The operations' execution times are distributed uniformly with $P_p = \{0, \dots, N/2\}$.

4. JSP bounds

The described binary search assumes that a lower bound T_{\min} and an upper bound T_{\max} are readily available. We cover their calculation for the sake of completeness. The simplest lower bounds are the *job* bound and the *machine* bound. The job bound is calculated by finding the total execution time of each job and keeping the largest one of them, put simply

$$\max_{n \in \{1, \dots, N\}} \sum_{i=k_{n-1}}^{k_n} p_i, \quad (\text{B11})$$

where we use the lexicographic index i for operations and where $k_0 = 1$. Similarly, we can define the machine bound as

$$\max_{m \in \{1, \dots, M\}} \sum_{i \in I_m} p_i, \quad (\text{B12})$$

where I_m is the set of indices of all operations that need to run on machine \mathbf{m}_m . Since these bounds are inexpensive to calculate, we can take the larger of the two. An even better lower bound can be obtained using the iterated Carlier and Pinson (ICP) procedure described in the window shaving subsection of Section III of the main text. We mentioned that the shaving procedure can show that a timespan does not admit a solution if a window closes completely. Using shaving for different timespans and performing a binary search, we can obtain the ICP lower bound in $\mathcal{O}(N^2 \log(N) M^2 T_{\max} \log_2(T_{\max} - T_{\min}))$, where T_{\min} and T_{\max} are some trivial lower and upper bound, respectively, such as the ones described in this section. Given that the search assumes a strict bound, we need to decrease whichever bound we chose here by one.

As for the upper bound, an excellent choice is provided by another classical algorithm developed by Applegate and Cook [42] for some finite computational effort. The straightforward alternative is given by the total work of the problem

$$\sum_{i \in \{1, \dots, k_N\}} p_i. \quad (\text{B13})$$

The solver's limitations can also serve to establish practical bounds for the search. For a given family of problems, if instances of a specific size can only be embedded with some acceptable probability for timespans smaller than T_{\max}^{embed} , the search can be set with this limit, and failure to find a solution will result in T_{\max}^{embed} , at which point the solver will need to report a failure or switch to another classical approach.

Appendix C: Classical algorithms

When designing a quantum annealing solver, a survey of classical methods provides much more than a

benchmark for comparison and performance. Classical algorithms can sometimes be repurposed as useful pre-processing techniques as demonstrated with variable pruning. We provide a quick review of the classical methods we use for this work as well as some details on the classical solvers to which we compare.

1. Variable pruning

Eliminating superfluous variables can greatly help mitigate the constraints on the number of qubits available. Several elimination rules are available and we explain below in more detail the procedure we used for our tests.

The first step in reducing the processing windows is to eliminate unneeded variables by considering the precedence constraints between the operations in a job, something we refer to as simple variable pruning. We define r_i as the sum of the execution times of all operations preceding operation O_i . Similarly, we define q_i as the sum of the execution times of all operations following O_i . The numbers r_i and q_i are referred to as the *head* and *tail* of operation O_i , respectively. An operation cannot start before its head and must leave enough time after finishing to fit its tail, so the window of possible start times, the *processing window*, for operation O_i is $[r_i, T - p_i - q_i]$.

If we consider the one-machine subproblems induced on each machine separately, we can update the heads and tails of each operation and reduce the processing windows further. For example, recalling that I_j is the set of indices of operations that have to run on machine M_j , we suppose that $a, b \in I_j$ are such that

$$r_a + p_a + p_b + q_b > T.$$

Then O_a must be run after O_b . This means that we can update r_a with

$$r_a = \max\{r_a, r_b + p_b\}.$$

We can apply similar updates to the tails because of the symmetry between heads and tails. These updates are known in the literature as *immediate selections*.

Better updates can be performed by using *ascendant sets*, introduced by Carlier and Pinson in [36]. A subset $X \subset I_j$ is an ascendant set of $c \in I_j$ if $c \notin X$ and

$$\min_{a \in X \cup \{c\}} r_a + \sum_{a \in X \cup \{c\}} p_a + \min_{a \in X} q_a > T.$$

If X is an ascendant set of c , then we can update r_c with

$$r_c = \max \left\{ r_c, \max_{X' \subset X} \left[\min_{a \in X'} r_a + \sum_{a \in X'} p_j \right] \right\}.$$

Once again, the symmetry implies that similar updates can be applied to the tails.

Carlier and Pinson in [21] provide an algorithm to perform all of the ascendant-set updates on M_j in

$\mathcal{O}(N \log(N))$, where $N = |I_j|$. After these updates have been carried out on a per-machine basis, we propagate the new heads and tails using the precedence of the operation by setting

$$r_{i+1} = \max \{r_{i+1}, r_i + p_i\}, \quad (\text{C1})$$

$$q_i = \max \{q_i, q_{i+1} + p_{i+1}\}, \quad (\text{C2})$$

for every pair of operations O_i and O_{i+1} that belong to the same job.

After propagating the updates, we check again if any descendant-set updates can be made, and repeat the cycle until no more updates are found. In our tests, we use an implementation similar to the one described in [21] to do the descendant-set updates.

Algorithm 1 is pseudocode that describes the shaving procedure. Here, the procedure `UpdateMachine(i)` updates heads and tails for machine i in $\mathcal{O}(N \log(N))$, as described by Carlier and Pinson in [21]. It returns True if any updates were made, and False otherwise. `PropagateWindows()` is a procedure that iterates over the tasks and checks that Eqs. (C1) and (C2) are satisfied, in $\mathcal{O}(NM)$.

Algorithm 1 Shaving algorithm

```

1: procedure ICP_SHAVE
2:   updated  $\leftarrow$  True
3:   while updated do
4:     updated  $\leftarrow$  False
5:     for  $i \in \text{machines}$  do
6:       updated  $\leftarrow \text{UpdateMachine}(i) \vee \text{updated}$ 
7:     if updated then PropagateWindows()

```

For each repetition of the outermost loop of Algorithm 1, we know that there is an update on the windows, which means that we have removed at least one variable. Since there are at most NMT variables, the loop will run at most this many times. The internal **for** loop runs exactly M times and does work in $\mathcal{O}(N \log(N))$. Putting all of this together, the final complexity of the shaving procedure is $\mathcal{O}(N^2 M^2 T \log(N))$.

2. Classical algorithm implementation

Brucker et al.'s branch and bound method [35] remains widely used due to its state-of-the-art status on smaller JSP instances and its competitive performance on larger ones [43]. Furthermore, the original code is freely available through ORSEP [44]. No attempt was made at optimizing this code and changes were only made to properly interface with our own code and time the results.

Martin and Shmoys' time-based approach [20] is less clearly defined in the sense that no publicly available standard code could be found and because a number of variants for both the shaving and the branch and bound strategy are described in the paper. As covered in the section on shaving, we have chosen the $\mathcal{O}(n \log(n))$ variants of heads and tails adjustments, the most efficient choice available. On the other hand, we have restricted our branch and bound implementation to the simplest strategy proposed, where each node branches between scheduling the next available operation (an operation that was not yet assigned a starting time) immediately or delaying it. Although technically correct, the same schedule can sometimes appear in both branches because the search is not restricted to *active* schedules, and unwarranted idle times are sometimes considered. According to Martin and Shmoys, the search strategy can be modified to prevent such occurrences, but these changes are only summarily described and we did not attempt to implement them. Other branching schemes are also proposed which we did not consider for this work. One should be careful when surveying the literature for runtimes of a full-optimization version based on this decision-version solver. What is usually reported assumes the use of a good upper bound such as the one provided by Applegate and Cook [42]. The runtime to obtain such bounds must be taken into account as well. It would be interesting to benchmark this decision solver in combination with our proposed optimized search, but this benchmarking we also leave for future work.

Benchmarking of classical methods was performed on an off-the-shelf Intel Core i7-930 processor clocked at 2.8 GHz.

-
- [1] M. W. Johnson, P. Bunyk, F. Maibaum, E. Tolokacheva, A. J. Berkley, E. M. Chapple, R. Harris, J. Johansson, T. Lanting, I. Perminov, et al., *Superconductor Science and Technology* **23**, 065004 (2010), URL <http://stacks.iop.org/0953-2048/23/i=6/a=065004>.
 - [2] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, *Nature Physics* **10**, 218 (2014), ISSN 1745-2473, URL <http://dx.doi.org/10.1038/nphys2900>.
 - [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), <http://www.sciencemag.org/content/345/6195/420.full.pdf>, URL <http://www.sciencemag.org/content/345/6195/420.abstract>.
 - [4] C. C. McGeoch and C. Wang, in *Proceedings of the ACM International Conference on Computing Frontiers* (ACM, New York, NY, USA, 2013), CF '13, pp. 23:1–23:11, ISBN 978-1-4503-2053-5, URL <http://doi.acm.org/10.1145/2482767.2482797>.
 - [5] V. N. Smelyanskiy, E. G. Rieffel, S. I. Knysh, C. P. Williams, M. W. Johnson, M. C. Thom, W. G. Macready, and K. L. Pudenz, arXiv:1204.2821 [quant-ph] (2012), 1204.2821.
 - [6] E. G. Rieffel, D. Venturelli, B. O'Gorman, M. B. Do, E. M. Prystay, and V. N. Smelyanskiy, Quantum Infor-

- mation Processing **14**, 1 (2015), ISSN 1570-0755, URL <http://dx.doi.org/10.1007/s11128-014-0892-x>.
- [7] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. Lidar, arXiv:1502.01663 [quant-ph] (2015), 1502.01663.
- [8] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, Physical Review X **5**, 031026 (2015), URL <http://link.aps.org/doi/10.1103/PhysRevX.5.031026>.
- [9] E. Rieffel, D. Venturelli, M. Do, I. Hen, and F. J., *Parametrized families of hard planning problems from phase transitions* (2014).
- [10] B. O’Gorman, E. G. Rieffel, D. Minh, and D. Venturelli, in *ICAPS 2015, Research Workshop Constraint Satisfaction Techniques for Planning and Scheduling Problems* (2015).
- [11] D. Venturelli, S. Mandrà, S. Knysh, B. O’Gorman, R. Biswas, and V. Smelyanskiy, Physical Review X **5**, 031040 (2015), 1406.7553.
- [12] B. O’Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy, The European Physical Journal Special Topics **224**, 163 (2015), ISSN 1951-6355, URL <http://dx.doi.org/10.1140/epjst/e2015-02349-9>.
- [13] W.-Y. Ku and J. C. Beck (2014).
- [14] A. Jain and S. Meeran, European Journal of Operational Research **113**, 390 (1999), ISSN 0377-2217, URL <http://www.sciencedirect.com/science/article/pii/S0377221798001131>.
- [15] E. Boros and P. L. Hammer, Discrete Applied Mathematics **123**, 155 (2002), ISSN 0166-218X, URL [http://dx.doi.org/10.1016/S0166-218X\(01\)00341-9](http://dx.doi.org/10.1016/S0166-218X(01)00341-9).
- [16] C.-C. Cheng and S. F. Smith, Annals of Operations Research **70**, 327 (1997).
- [17] A. Garrido, M. A. Salido, F. Barber, and M. López, in *Proc. ECAI-2000 Workshop on New Results in Planning, Scheduling and Design (PuK2000)* (2000), pp. 44–49.
- [18] H. M. Wagner, Naval Research Logistics Quarterly **6**, 131 (1959), ISSN 1931-9193, URL <http://dx.doi.org/10.1002/nav.3800060205>.
- [19] A. S. Manne, Operations Research **8**, 219 (1960), ISSN 0030364X, 15265463, URL <http://www.jstor.org/stable/167204>.
- [20] P. Martin and D. B. Shmoys, in *Integer Programming and Combinatorial Optimization*, edited by W. H. Cunningham, S. McCormick, and M. Queyranne (Springer Berlin Heidelberg, 1996), vol. 1084 of *Lecture Notes in Computer Science*, pp. 389–403, ISBN 978-3-540-61310-7, URL http://dx.doi.org/10.1007/3-540-61310-2_29.
- [21] J. Carlier and P. E., European Journal of Operational Research **78**, 146 (1994), ISSN 0377-2217, URL <http://www.sciencedirect.com/science/article/pii/0377221794903794>.
- [22] L. Péridy and D. Rivreau, European Journal of Operational Research **164**, 24 (2005), ISSN 0377-2217, URL <http://www.sciencedirect.com/science/article/pii/S0377221703009019>.
- [23] Y. Caseau and F. Laburthe, in *Proc. of the 11th International Conference on Logic Programming* (1994).
- [24] W. M. Kaminsky and S. Lloyd, in *Quantum Computing and Quantum Bits in Mesoscopic Systems*, edited by A. J. Leggett, B. Ruggiero, and P. Silvestrini (Springer US, 2004), pp. 229–236, ISBN 978-1-4613-4791-0, URL http://dx.doi.org/10.1007/978-1-4419-9092-1_25.
- [25] V. Choi, Quantum Information Processing **10**, 343 (2011), ISSN 1570-0755, URL <http://dx.doi.org/10.1007/s11128-010-0200-3>.
- [26] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos, in *Algorithm Theory - SWAT 2010*, edited by H. Kaplan (Springer Berlin Heidelberg, 2010), vol. 6139 of *Lecture Notes in Computer Science*, pp. 322–333, ISBN 978-3-642-13730-3, URL http://dx.doi.org/10.1007/978-3-642-13731-0_31.
- [27] J. Cai, W. G. Macready, and A. Roy, arXiv:1406.2741 [quant-ph] (2014), 1406.2741.
- [28] A. Perdomo-Ortiz, J. Fluegemann, R. Biswas, and V. N. Smelyanskiy, arXiv:1503.01083 [quant-ph] (2015), 1503.01083.
- [29] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. Amin, A. Smirnov, M. Mohseni, and H. Neven, arXiv:1411.4036 [quant-ph] (2014), 1411.4036.
- [30] V. N. Smelyanskiy, D. Venturelli, A. Perdomo-Ortiz, S. Knysh, and M. I. Dykman, arXiv:1511.02581 [quant-ph] (2015), 1511.02581.
- [31] D. Venturelli, D. J. J. Marchand, and G. Rojo, arXiv:1506.08479 [quant-ph] (2015), 1506.08479.
- [32] A. Perdomo-Ortiz, B. O’Gorman, J. Fluegemann, R. Biswas, and V. N. Smelyanskiy, arXiv:1503.05679 [quant-ph] (2015), 1503.05679.
- [33] A. D. King and C. C. McGeoch, arXiv:1410.2628 [cs.DS] (2014), 1410.2628.
- [34] K. L. Pudenz, T. Albash, and D. A. Lidar, Nature Communications **5** (2014), URL <http://dx.doi.org/10.1038/ncomms4243>.
- [35] P. Brucker, B. Jurisch, and B. Sievers, Discrete Applied Mathematics **49**, 107 (1994), ISSN 0166-218X, URL <http://www.sciencedirect.com/science/article/pii/0166218X94902046>.
- [36] J. Carlier and E. Pinson, Annals of Operations Research **26**, 269 (1991), ISSN 0254-5330, URL <http://dl.acm.org/citation.cfm?id=115185.115198>.
- [37] J. C. Beck, T. K. Feng, and J.-P. Watson, INFORMS Journal on Computing **23**, 1 (2011).
- [38] M. J. Streeter and S. F. Smith, Journal of Artificial Intelligence Research **26**, 247 (2006).
- [39] W. Vinci, T. Albash, G. Paz-Silva, I. Hen, and D. A. Lidar, Physical Review A **92**, 042310 (2015).
- [40] A. D. King, T. Lanting, and R. Harris, arXiv:1502.02098 [quant-ph] (2015), 1502.02098.
- [41] V. Choi, Quantum Information Processing **7**, 193 (2008), ISSN 1570-0755, URL <http://dx.doi.org/10.1007/s11128-008-0082-9>.
- [42] A. D. and C. W., ORSA J. Comput. **3**, 149 (1991).
- [43] W. Brinkkötter and P. Brucker, Journal of Scheduling **4**, 53 (2001), ISSN 1099-1425, URL [http://dx.doi.org/10.1002/1099-1425\(200101/02\)4:1<53::AID-JOS59>3.0.CO;2-Y](http://dx.doi.org/10.1002/1099-1425(200101/02)4:1<53::AID-JOS59>3.0.CO;2-Y).
- [44] B. P., J. B., and S. B., European Journal of Operational Research **57**, 132 (1992).