Job Shop Scheduling Solver based on Quantum Annealing

Abstract

Quantum annealing is emerging as a promising near-term quantum computing approach to solve combinatorial optimization problems. An method to solve the makespan-minimization job-shop scheduling problem that makes use of a quantum annealer is presented in detail. After formulating the problem as a time-indexed quadratic unconstrained binary optimization, several pre-processing and graph-embedding strategies are employed to compile optimally parametrized families of the problem for scheduling instances of up to six jobs and six machines on the D-Wave Systems Vesuvius processor. Problem simplifications and partitioning algorithms, including variable pruning and running strategies that consider tailored binary searches, are discussed and the results from the processor are compared against state-of-the-art global-optimum solvers.

I. Introduction

The commercialization and independent benchmarking (Johnson et al. (2010); Boixo et al. (2014a); Rønnow et al. (2014); McGeoch and Wang (2013)) 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 (Smelyanskiy et al. (2012)) have been adapted and have seen small-scale testing on the D-Wave Two™ "Vesuvius" processor (Rieffel et al. (2015)). The best procedure for comparison of quantum analog performance with traditional digital methods is still under debate (Rønnow et al. (2014); Hen et al. (2015); Katzgraber et al. (2015)) 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 Rieffel et al. (2014), 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 sizes, potentially testable on current- or next-generation D-Wave hardware. An empirical takeaway from Rieffel et al. (2015) (validated also by experimental results in O'Gorman et al. (2015b); Venturelli et al. (2015)) 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 (O'Gorman et al. (2015a)).

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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 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 state-of-the-art 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 x 10) (**?**). Indeed, some known 20 x 15 instances often used for benchmarking still have not been solved to optimality even by the best special-purpose solvers (Jain and Meeran (1999)), and 20 x 20 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.

Problem definition and conventions

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

$$\mathbf{j}_n = \{O_{n1} \to O_{n2} \to \cdots \to O_{nL_n}\}.$$

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; see the supplemental material for more details) 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 $\mathbf{J}M\big|p_{nj}\in[p_{\min},\ldots,p_{\max}]\big|C_{\max}$ in the well-known $\alpha|\beta|\gamma$ notation, where $p_{\min}$ and $p_{\max}$ are the smallest and largest execution time allowed, respectively. In this notation, $\mathbf{J}M$ 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

$$\begin{aligned}\mathbf{j}_1 &= \{O_1 \to \cdots \to O_{k_1}\},\\ \mathbf{j}_2 &= \{O_{k_1+1} \to \cdots \to O_{k_2}\},\\ &\cdots\\ \mathbf{j}_N &= \{O_{k_{N-1}+1} \to \cdots \to O_{k_N}\}. \end{aligned} \tag{1}$$

Given the running index over all operations $i \in \{1,\ldots,k_N\}$, we let q_i be the index of the machine $\mathbf{m}_{q_i}$ responsible for executing operation O_i. We let I_m 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.

Quantum annealing formulation

In this work, we seek a suitable formulation of the JSP for a quantum annealing optimizer (such as the flux-qubit-based D-Wave Two). The optimizer is best described as an oracle that solves quadratic unconstrained binary optimization (QUBO) problems (Boros and Hammer (2002)). The binary polynomial associated with a QUBO problem or the QUBO solver can be depicted as a graph, with nodes representing variables and values attached to nodes and edges representing linear and quadratic terms, respectively. The optimizer is expected to find the global minimum with some probability which itself depends on the problem and the device's parameters, yet 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 customarily done, we rely on the classical procedure known as embedding to adapt the connectivity of the solver to the problem at hand. During this procedure, two or more variables can be forced to take on the same value by including additional constraints in the model. As will be detailed in the following sections, in the underlying Ising model, which is equivalent to a QUBO problem (O'Gorman et al. (2015b)), this is achieved by introducing a large ferromagnetic (negative) coupling J_{F} between two spins. 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 its 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.

II. QUBO formulation

While there are several ways the JSP can be formulated as a mixed-integer programming problem, such as the rank-based formulation (Wagner (1959)) or the disjunctive formulation (Manne (1960)), our formulation is based on a straightforward time-indexed representation particularly amenable to quantum annealers. We assign a set of binary variables for each operation, corresponding to the various possible discrete starting times the operation can have:

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

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.

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

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

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

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

where $R_m = A_m \cup B_m$ and

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

a)

	operation 1	operation 2	operation 3
$\mathbf{j}_1$	$\mathbf{m}_1$, $p=1$	$\mathbf{m}_2$, $p=1$	$\mathbf{m}_3$, $p=1$
$\mathbf{j}_2$	$\mathbf{m}_3$, $p=2$	$\mathbf{m}_1$, $p=1$	$\mathbf{m}_2$, $p=2$
$\mathbf{j}_3$	$\mathbf{m}_2$, $p=1$	$\mathbf{m}_1$, $p=1$	$\mathbf{m}_3$, $p=2$

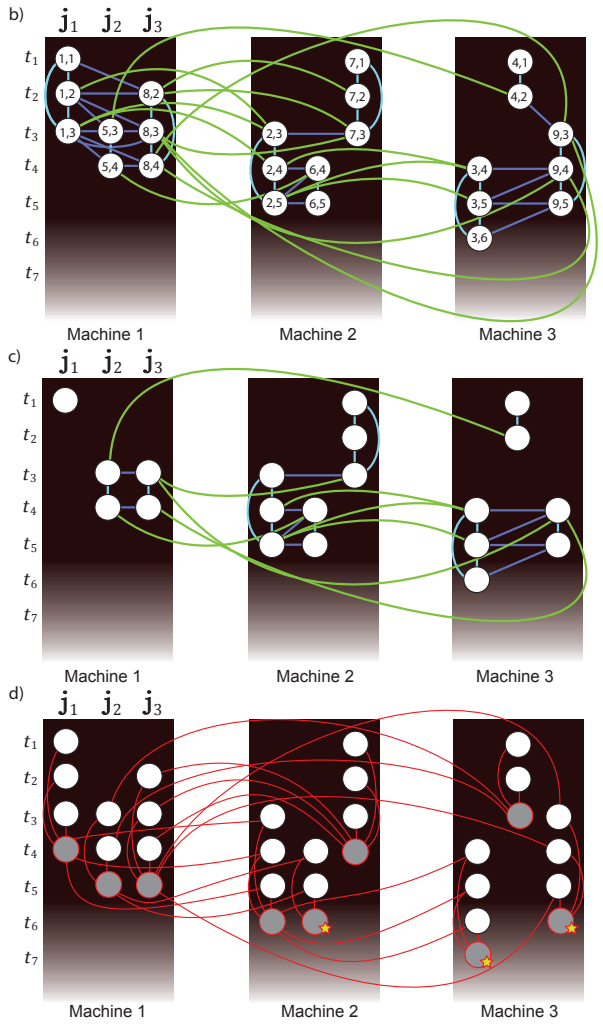

Figure 1: a) Table representation of an example 3 x 3 instance whose execution times have been randomly selected to be either 1 or 2 time units. b) Pictorial view of the QUBO mapping of the above example for $H_{T=6}$. Green, purple, and cyan edges refer respectively to h_1, h_2, and h_3 quadratic coupling 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.

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

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

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

The resulting classical objective function (Hamiltonian) is given by

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

where

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

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

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

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 x 3 example (Figure 1-a).

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 trivially polynomial as the system size increases). 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 null. This simplification eliminates an estimated number of variables equal to $NM\left(M\left\langle p\right\rangle - 1\right)$, 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. 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.

Window shaving

In the time-index formalism, reducing the execution windows of operations (i.e., shaving) (Martin and Shmoys (1996)), or in the disjunctive approach, adjusting the *heads* and *tails* of operations (Carlier and E. (1994); Péridy and Rivreau (2005)), constitutes 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 prune as many variables as possible, thus enabling larger problems to be considered and improving 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 (Carlier and E. (1994)) reviewed in the supplemental material with worst-case complexity given by $\mathcal{O}(N^2M^2T\log(N))$. Instead of looking at the one-job subproblems and their constraints to eliminate variables, as we did for the simple pruning, we look at the one-machine subproblems and associated constraints to further prune variables. An example of the resulting QUBO problem is presented in Figure 1-c.

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 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 $\{O_{k_1}, \ldots, O_{k_N}\}$. 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 available precision can accomodate 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 supplemental material.

IV. Ensemble pre-characterization and compilation

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.

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 $\mathcal{P}_p = \{p_{\min}, \ldots, p_{\max}\}$, 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. As we discuss in the supplemental material, 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.

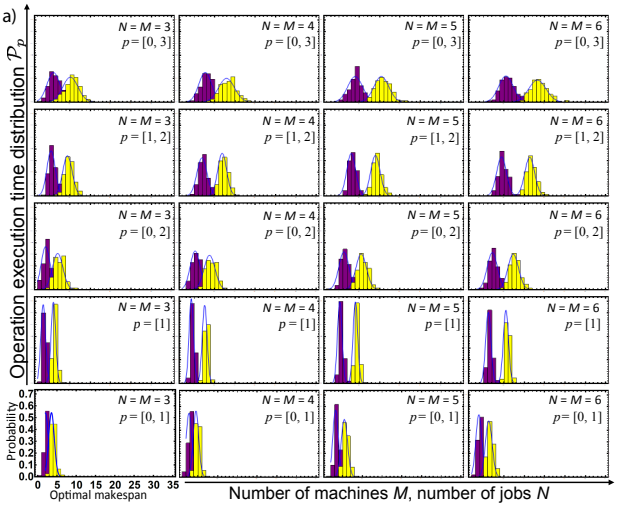

Figure 2: a) Normalized histograms of optimal makespans $\mathcal{T}$ for parametrized families of JSP instances with $N = M$, $\mathcal{P}_p$ on the y-axis, $\theta = 1$ (yellow), and $\theta = 0.5$ (purple). The distributions are histograms of occurrences for 1000 random instances, fitted with a Gaussian 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 $C_\sigma = 0.003$.

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 (Kaminsky and Lloyd (2004); Choi (2011)), which for the D-Wave Two is a Chimera graph (Johnson et al. (2010)). Formally, we seek an isomorphism between the problem's QUBO graph and a graph minor of the solver. This procedure 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. The supplemental material covers this process in more detail. An example of embedding for a 5 x 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 Chimera graph. Finding the optimal tiling that uses the fewest qubits is NP-hard (Adler et al. (2010)), and the standard approach is to employ heuristic algorithms (Cai, Macready, and Roy (2014)). 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 (Venturelli et al. (2015)). This quadratic scaling is apparent in Figure 3-b where we report on the compilation attempts using the algorithm in Cai, Macready, and Roy (2014). 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 next-generation chip currently being manufactured by D-Wave Systems), and for a speculative yet-larger chip where the Chimera block is double in size. 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 $NM\theta\langle p \rangle$ 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 Rieffel et al. (2015), graph connectivity has a much more dramatic impact than qubit count on embeddability.

Once the topological aspect of embedding is solved, we set the ferromagnetic interactions needed to adapt the connectivity of the solver to the problem at hand. 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 Rieffel et al. (2015) 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 Perdomo-Ortiz et al. (2015a) in order to reach the best possible instance-specific J_F with a series

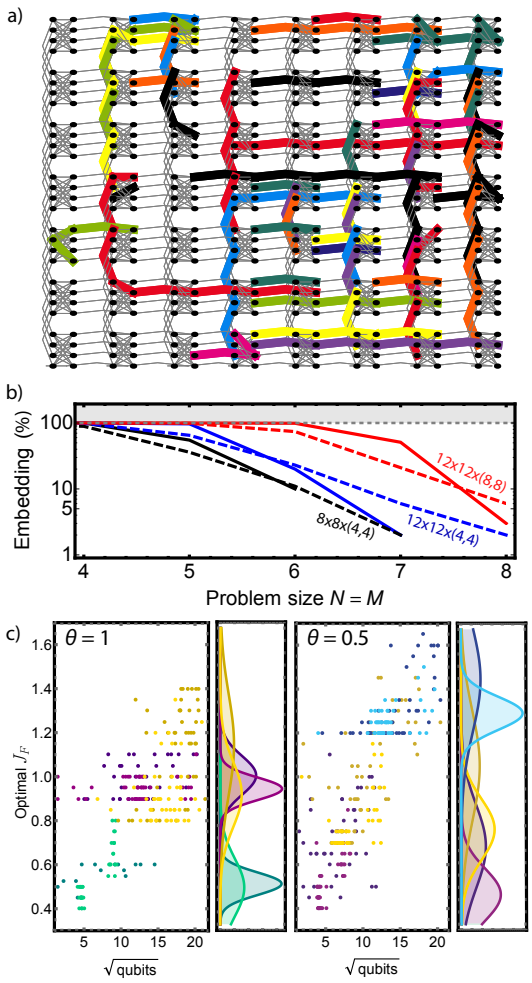

Figure 3: a) Example of an embedded JSP instance on NASA's D-Wave Two chip. Each colored chain of qubits represents a logical binary variable determined by the embedding. For clarity, active connections between the qubits are not shown. b) Embedding probability as a function of $N = M$ for $\theta = 1$ (similar results are observed for $\theta = 0.5$). Solid lines refer to $\mathcal{P}_p = [1, 1]$ while 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 x 3, purple: 4 x 4, yellow: 5 x 5, blue: 6 x 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 (Perdomo-Ortiz et al. (2015a)).

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. 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 $\mathtt{T}$ 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 anneal in a desired ground state depends, in an essentially 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 (Boixo et al. (2014b); Smelyanskiy et al. (2015)), 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 benchmarking only the Hamiltonians with $\mathcal{T} = T$ with the D-Wave machine, but in the supplemental material we present a prescription on how to operate the machine in the general case by leveraging data analysis of past results on parametrized ensembles.

On the device installed at NASA Ames (it has 509 working qubits; details are presented in Perdomo-Ortiz et al. (2015b)), 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 99% probability of success on the ground state of $H_{\mathcal{T}}$ (i.e., $R^\star$ following the notation in the supplementary information on computational strategy, each annealing cycle lasting $t_A = 20$ μs) 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, the annealing time, etc. Many of these problems are

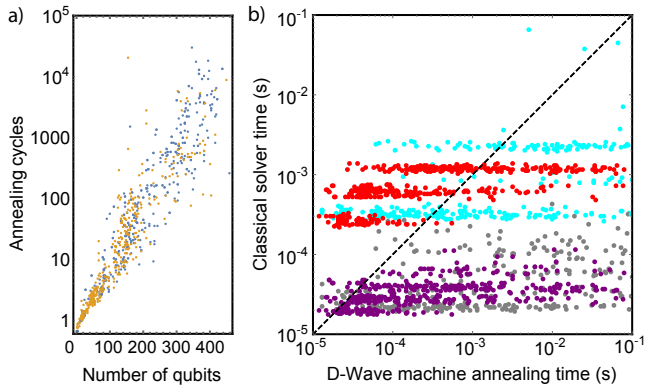

Figure 4: a) Number of repetitions required to solve $H_{\mathcal{T}}$ with the D-Wave Two with 99% probability of success (see the supplemental material). The blue points indicate instances with $\theta = 1$ while yellow points correspond to $\theta = 0.5$ (they are the same instances and runtimes used 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 D-Wave optimizer. Gray and violet points represent runtimes compared with algorithm "B" (see Brucker et al. in main text) while cyan and red are compared to the MS algorithm (see Martin and Shmoys in text), 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 Venturelli et al. (2015); Rieffel et al. (2015); Perdomo-Ortiz et al. (2015a); King and McGeoch (2014); Pudenz, Albash, and Lidar (2014). 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 Perdomo-Ortiz et al. (2015b).

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 state-of-the-art exhaustive classical algorithms in order to gain insight on how current quantum annealing technology compares with paradigmatic classical optimization methods. Leaving the performance on 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 Brucker, Jurisch, and Sievers (1994), exploits the disjunctive graph representation and a branch and bound strategy that very effectively combines a branching scheme based on selecting the direction of a single disjunctive edge (according to some single-machine constraints), and a technique introduced in Carlier and Pinson (1991) for fixing additional disjunctions (based on a preemptive relaxation). It has publicly available code and is considered a state-of-the-art complete solver for the small instances currently accessible to us, while remaining competitive for larger ones even if other classical approaches become more favorable (Beck, Feng, and Watson (2011)). B has been used in Streeter and Smith (2006) 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 (Rieffel et al. (2014)).

The second algorithm, "MS", introduced in Martin and Shmoys (1996), 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 Carlier and E. (1994) 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 (more details about our implementation of MS are in the supplemental material), 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 state-of-the-art classical algorithm run on a single-core modern processor, and that the problem sizes tested in this study are still too small for the asymptotic behavior of the classical software to be clearly demonstrated and measured. The comparison between the D-Wave machine's solution time for $H_{\mathcal{T}}$ 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, while 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 supplemental material). It will, however, be interesting to compare B to a quantum annealing solver for sizes considered B-intractable due to increasing memory and time requirements when larger chips become available.

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; see the supplemental material for details). 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), while the D-Wave machine's complexity seems to be sensitive mostly to the

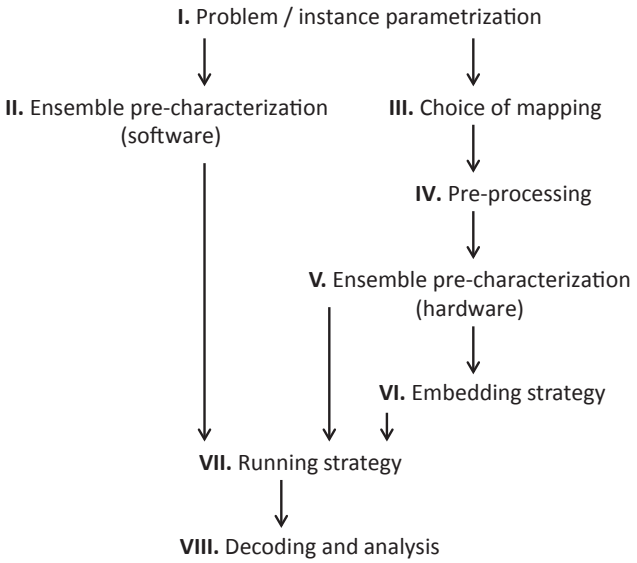

Figure 5: I–II) Appropriate choice of benchmarking and classical simulations is discussed in Section IV. III–IV) Mapping to QUBO problems is discussed in Sections II and III. V–VI) Pre-characterization for parameter setting is described in Section VI. VII) Structured run strategies adapted to specific problems have not to our knowledge been discussed before. We discuss a prescription in the supplementary information. 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.

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 extensive 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 problem 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. While 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 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 we describe in the previous sections. The pictured scheme is not intended to be complete, e.g., the solving framework can benefit from other concepts such as performance-tuning techniques (Perdomo-Ortiz et al. (2015a)) and error-correction repetition lattices (**?**). The use of the decision version of the problem combined with a properly designed search strategy is a first example of a powerful partitioning scheme. The proposed timespan discrimination further provides an adjustable compromise between the full optimization and decision versions, 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. Instance pre-characterization to fine-tune the solver parameters has also been used to improve the search strategy and it constitutes a tool whose use we expect to become common practice in application problems amenable to similar 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 a fruitful area of research and one that is likely to see promising developments with the advent of these new quantum heuristics algorithms.

Acknowledgements

The authors would like to thank

Supplemental material at the *ICAPS 2016* website.

References

Adler, I.; Dorn, F.; Fomin, F. V.; Sau, I.; and Thilikos, D. M. 2010. Faster parameterized algorithms for minor containment. In Kaplan, H., ed., *Algorithm Theory - SWAT 2010*, volume 6139 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 322–333.

Beck, J. C.; Feng, T. K.; and Watson, J.-P. 2011. Combining constraint programming and local search for job-shop scheduling. *INFORMS Journal on Computing* 23(1):1–14.

Boixo, S.; Ronnow, T. F.; Isakov, S. V.; Wang, Z.; Wecker, D.; Lidar, D. A.; Martinis, J. M.; and Troyer, M. 2014a. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics* 10(3):218–224.

Boixo, S.; Smelyanskiy, V. N.; Shabani, A.; Isakov, S. V.; Dykman, M.; Denchev, V. S.; Amin, M.; Smirnov, A.; Mohseni, M.; and Neven, H. 2014b. Computational role of collective tunneling in a quantum annealer. *arXiv:1411.4036 [quant-ph]*.

Boros, E., and Hammer, P. L. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics* 123(1-3):155–225.

Brucker, P.; Jurisch, B.; and Sievers, B. 1994. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics* 49:107 – 127.

Cai, J.; Macready, W. G.; and Roy, A. 2014. A practical heuristic for finding graph minors. *arXiv:1406.2741 [quant-ph]*.

Carlier, J., and E., P. 1994. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research* 78(2):146 – 161.

Carlier, J., and Pinson, E. 1991. A practical use of jackson's preemptive schedule for solving the job shop problem. *Annals of Operations Research* 26(1-4):269–287.

Choi, V. 2011. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing* 10(3):343–353.

Hen, I.; Job, J.; Albash, T.; Rønnow, T. F.; Troyer, M.; and Lidar, D. 2015. Probing for quantum speedup in spin glass problems with planted solutions. *arXiv:1502.01663 [quant-ph]*.

Jain, A., and Meeran, S. 1999. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research* 113(2):390 – 434.

Johnson, M. W.; Bunyk, P.; Maibaum, F.; Tolkacheva, E.; Berkley, A. J.; Chapple, E. M.; Harris, R.; Johansson, J.; Lanting, T.; Perminov, I.; Ladizinsky, E.; Oh, T.; and Rose, G. 2010. A scalable control system for a superconducting adiabatic quantum optimization processor. *Superconductor Science and Technology* 23(6):065004.

Kaminsky, W. M., and Lloyd, S. 2004. Scalable architecture for adiabatic quantum computing of np-hard problems. In Leggett, A. J.; Ruggiero, B.; and Silvestrini, P., eds., *Quantum Computing and Quantum Bits in Mesoscopic Systems*. Springer US. 229–236.

Katzgraber, H. G.; Hamze, F.; Zhu, Z.; Ochoa, A. J.; and Munoz-Bauza, H. 2015. Seeking quantum speedup through spin glasses: The good, the bad, and the ugly*. *Physical Review X* 5:031026.

King, A. D., and McGeoch, C. C. 2014. Algorithm engineering for a quantum annealing platform. *arXiv:1410.2628 [cs.DS]*.

Manne, A. S. 1960. On the job-shop scheduling problem. *Operations Research* 8(2):219–223.

Martin, P., and Shmoys, D. B. 1996. A new approach to computing optimal schedules for the job-shop scheduling problem. In Cunningham, W. H.; McCormick, S.; and Queyranne, M., eds., *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 389–403.

McGeoch, C. C., and Wang, C. 2013. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '13, 23:1–23:11. New York, NY, USA: ACM.

O'Gorman, B.; Babbush, R.; Perdomo-Ortiz, A.; Aspuru-Guzik, A.; and Smelyanskiy, V. 2015a. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics* 224(1):163–188.

O'Gorman, B.; Rieffel, E. G.; Minh, D.; and Venturelli, D. 2015b. Compiling planning into quantum optimization problems: a comparative study. In *ICAPS 2015, Research Workshop Constraint Satisfaction Techniques for Planning and Scheduling Problems*.

Perdomo-Ortiz, A.; Fluegemann, J.; Biswas, R.; and Smelyanskiy, V. N. 2015a. A Performance Estimator for Quantum Annealers: Gauge selection and Parameter Setting. *arXiv:1503.01083 [quant-ph]*.

Perdomo-Ortiz, A.; O'Gorman, B.; Fluegemann, J.; Biswas, R.; and Smelyanskiy, V. N. 2015b. Determination and correction of persistent biases in quantum annealers. *arXiv:1503.05679 [quant-ph]*.

Péridy, L., and Rivreau, D. 2005. Local adjustments: A general algorithm. *European Journal of Operational Research* 164(1):24 – 38.

Pudenz, K. L.; Albash, T.; and Lidar, D. A. 2014. Error-corrected quantum annealing with hundreds of qubits. *Nature Communications* 5.

Rieffel, E.; Venturelli, D.; Do, M.; Hen, I.; and J., F. 2014. Parametrized families of hard planning problems from phase transitions.

Rieffel, E. G.; Venturelli, D.; O'Gorman, B.; Do, M. B.; Prystay, E. M.; and Smelyanskiy, V. N. 2015. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing* 14(1):1–36.

Rønnow, T. F.; Wang, Z.; Job, J.; Boixo, S.; Isakov, S. V.; Wecker, D.; Martinis, J. M.; Lidar, D. A.; and Troyer, M. 2014. Defining and detecting quantum speedup. *Science* 345(6195):420–424.

Smelyanskiy, V. N.; Rieffel, E. G.; Knysh, S. I.; Williams, C. P.; Johnson, M. W.; Thom, M. C.; Macready, W. G.; and Pudenz, K. L. 2012. A Near-Term Quantum Computing Approach for Hard Computational Problems in Space Exploration. *arXiv:1204.2821 [quant-ph]*.

Smelyanskiy, V. N.; Venturelli, D.; Perdomo-Ortiz, A.; Knysh, S.; and Dykman, M. I. 2015. Quantum annealing via environment-mediated quantum diffusion. *arXiv:1511.02581 [quant-ph]*.

Streeter, M. J., and Smith, S. F. 2006. How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines. *Journal of Artificial Intelligence Research* 26:247–287.

Venturelli, D.; Mandrà, S.; Knysh, S.; O'Gorman, B.; Biswas, R.; and Smelyanskiy, V. 2015. Quantum Optimization of Fully Connected Spin Glasses. *Physical Review X* 5(3):031040.

Wagner, H. M. 1959. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly* 6(2):131–140.