Challenges in Solving Scheduling Problems with the D-Wave Quantum Annealer

Mike Zielewski, Mulya Agung, Ryusuke Egawa, Hiroyuki Takizawa
Tohoku University, Sendai, Japan
Email: mziele1@dc.tohoku.ac.jp

Introduction

Quantum Annealing (QA)

QA is a metaheuristic method for minimizing functions in quadratic unconstrained binary optimization (QUBO) form.

$$\sum_i Q_{i,i}x_i + \sum_{i<j} Q_{i,j}x_ix_j$$

where Q is an upper diagonal matrix of weights and $\boldsymbol{x}$ is a vector of binary variables.

By exploiting principles of quantum physics, such as superposition and tunneling, QA has the potential to solve problems faster than modern classical computers.

Motivation

To utilize the D-Wave 2000Q quantum annealer, problems must first go through a series of preparatory steps, such as:

- **Formulation**—Redefine a problem to use binary variables and be compatible with QUBO format.
- **Embedding**—Find a mapping of QUBO variables to hardware qubits.

These steps incur time and computational overheads which can offset potential acceleration. Furthermore, each of these steps can have a significant impact on the performance of QA.

Objectives

Evaluate the challenges and overheads in all phases of the QA process and determine how each is affected by problem size and complexity.

Target Problem

The problem selected for exploring the challenges of the QA process is the job-shop scheduling problem (JSP). The JSP is a classical NP-Hard optimization problem in which given a set of $\boldsymbol{N}$ jobs, each containing a set of $\boldsymbol{O}$ operations, and a set of $\boldsymbol{M}$ machines**,** find an arrangement of operations for which job order and machine capacity constraints are not violated.

Common objective variations are the decision version (no constraints violated) and the optimization version (minimize the makespan, the time from start to finish).

Formulation

Time-indexed JSP

To prepare the JSP as a QUBO, a formulation using binary variables is required. We use the time-indexed JSP formulation introduced in [1]. The time-indexed JSP formulation has the following properties:

- Binary transformation— variables to indicate the start of every operation at every time interval
- Execution upper bound = $\boldsymbol{T}$
- Extra constraint—operations start exactly once

This formulation requires $\boldsymbol{N} \times \boldsymbol{M} \times \boldsymbol{T}$ variables, however, variable pruning techniques based on operation precedence are used to prune variables leading to impossible schedules.

Instance Parameters

Random JSP instances were generated having the following properties:

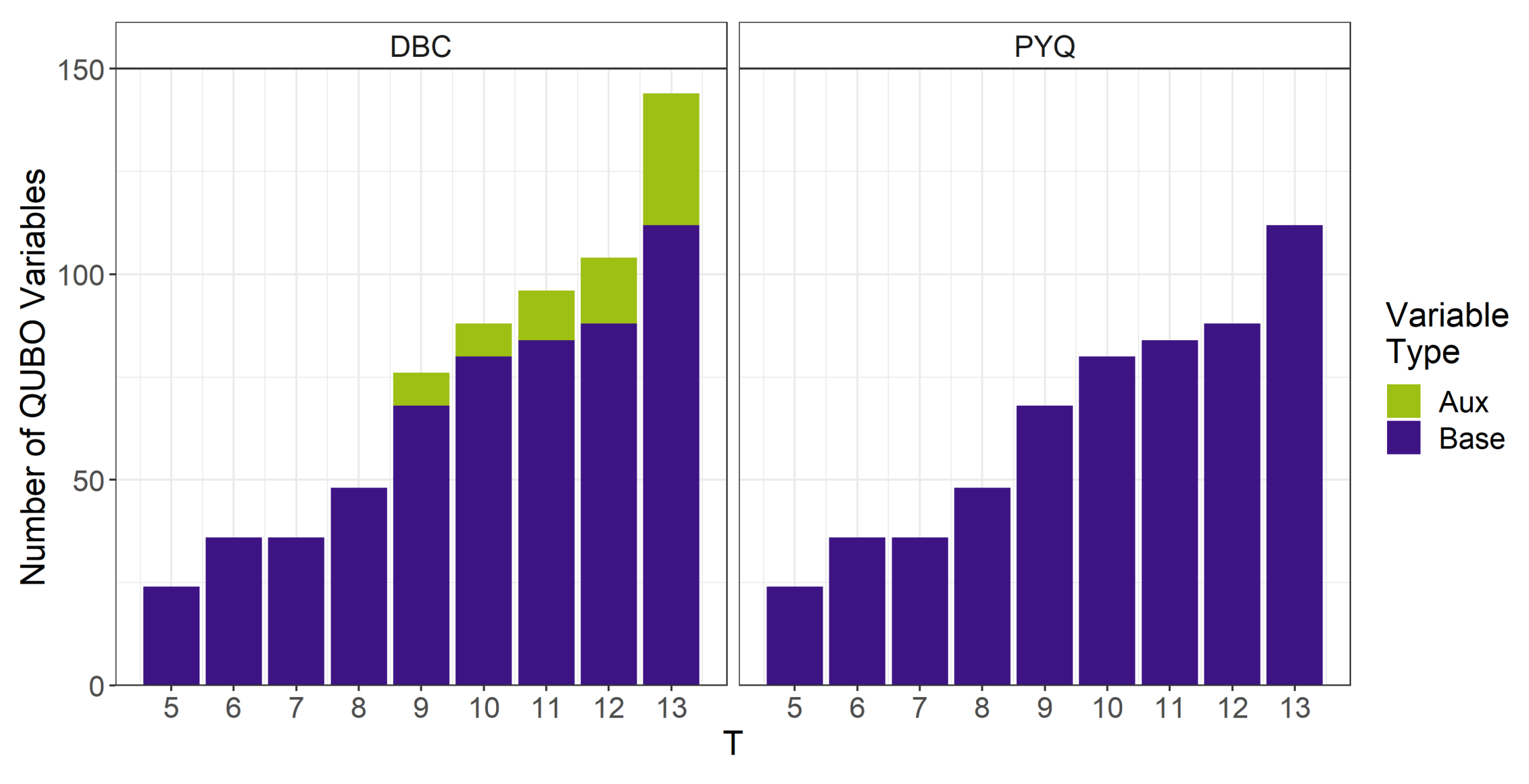
- $\boldsymbol{M}$, $\boldsymbol{N}$— range from 3 to 7
- Operation processing times— subset of [0, 1, 2]
- $\boldsymbol{T}$— the optimal makespan for the instance, calculated using classical tools.

Instances were discarded if the resulting QUBO was too large to be embedded onto the annealer.

QUBO Tools

D-Wave Binary CSP (DBC)
- Developed by D-Wave.

PyQUBO (PYQ)
- Developed by Recruit Communications.



Both tools use the same number of QUBO variables, but DBC introduces **additional auxiliary variables**. Additionally, PYQ compilation was observed to be faster than that of DBC.

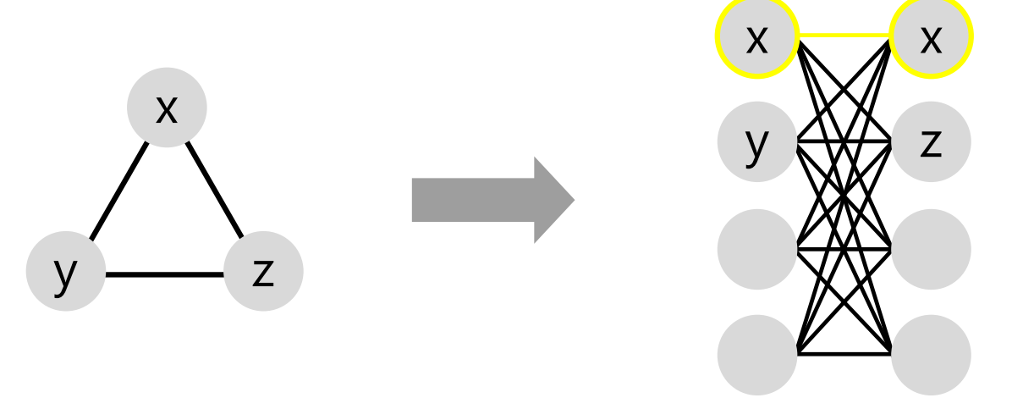
Embedding

Quantum Processing Unit (QPU)

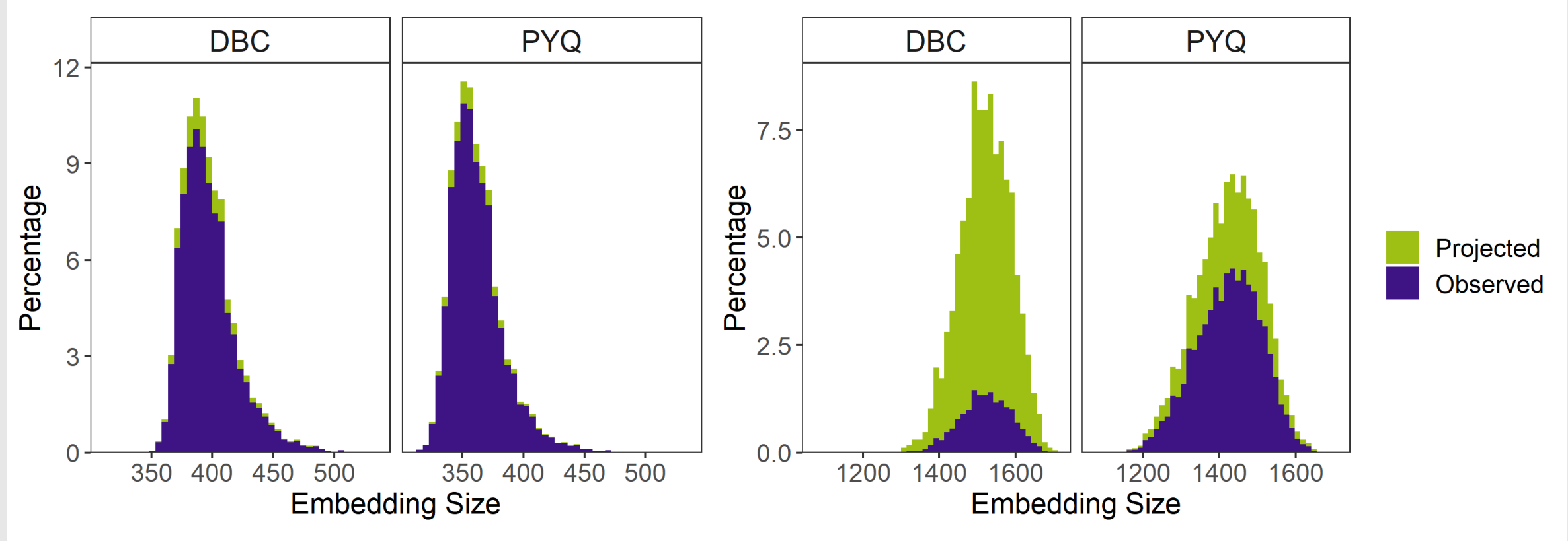
The D-Wave 2000Q QPU hardware graph consists of 2048 qubits arranged in a 16x16 grid of complete bipartite K(4,4) unit cells. In this architecture, qubits have a maximum connectivity of six.

Graph Minor

In order to utilize QA, a mapping of variables to qubits must be found such that the QUBO is a graph minor of the QPU hardware graph. Due to the **limited connectivity** of the QPU, multiple physical qubits may need to be chained together into one logical qubit to increase connectivity.

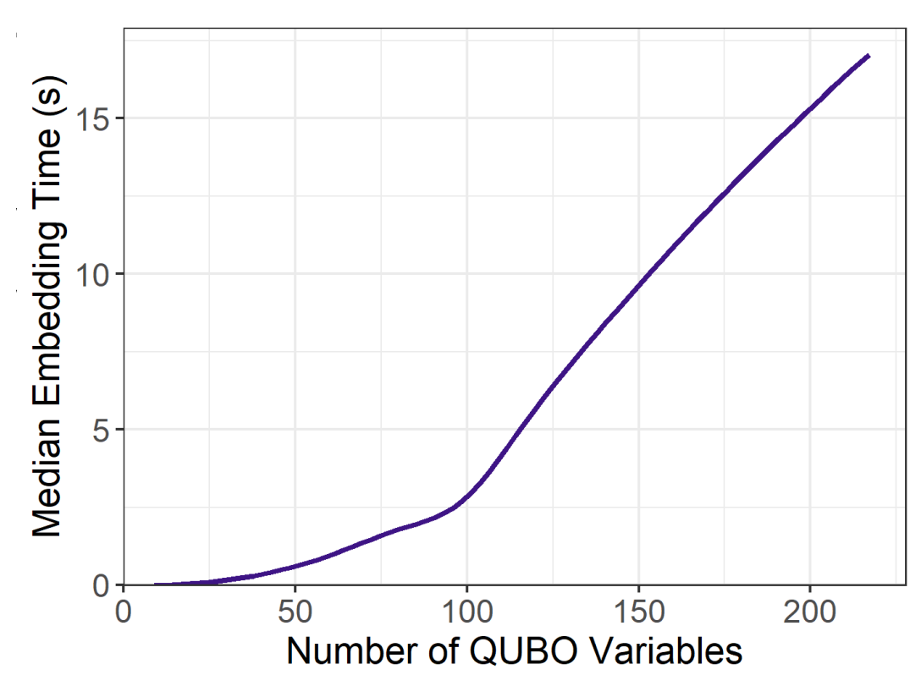


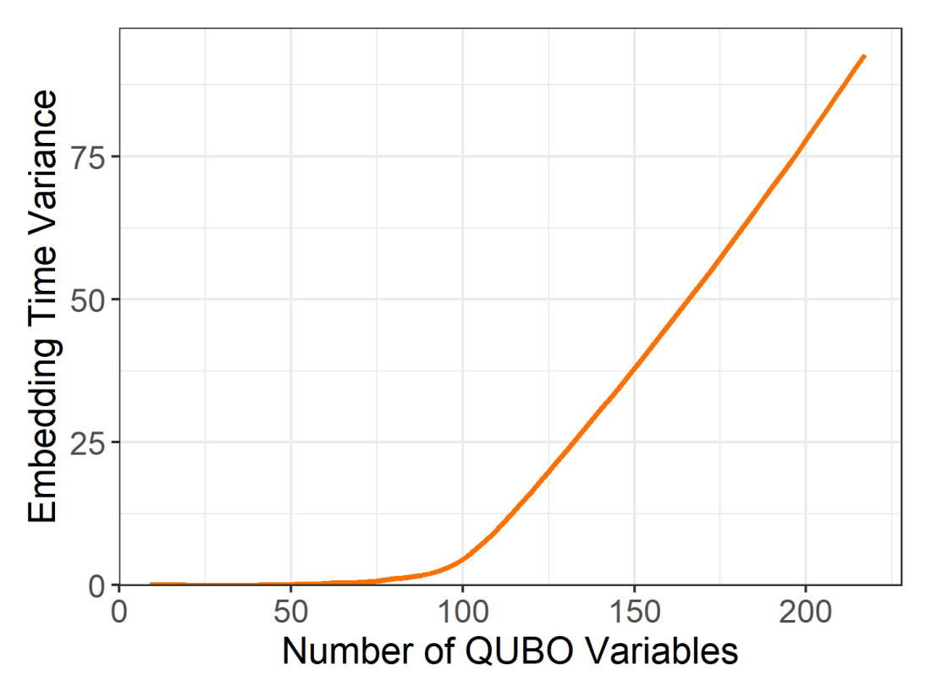
D-Wave provides *minorminer*, a heuristic tool for finding embeddings. Owing to its heuristic nature, embeddings with varying chain lengths and total qubit usage will be found; however, the algorithm also can fail and return no embedding. Generally, smaller embeddings and chains are preferred to their larger counterparts which suffer from early freezeout; the time in the anneal after which qubit values cease to change.



The above figures visualize the observed distributions of embeddings for a small and large JSP instance as well as the projected frequency if no embedding failures occurred.

For a small instance, there are not many embedding failures and the range of embedding sizes is small compared to larger instances. The DBC formulation of the larger instance has a wider range of embedding sizes and suffers from a **83% failure rate** despite the mean embedding size taking up 73% of the QPU.





Summary

- Embedding size and time grow with the number of variables in a QUBO.
- QUBOs with many variables can experience high embedding failure rates.
- High variable connectivity requires qubit chaining, which can also increase embedding failure rate.

Unembedding

Chain Unembedding

QUBO variables take on the value of their chain after annealing. Chains having differing values require a method to determine which value to assign the variable. We compare two methods:

- Majority vote—Select the most frequently occurring value.
- Minimize energy—Select the value that results in the lowest local energy penalty.

The majority vote unembedding method does not use any information from the problem in determining the value of a chain. On the other hand, the minimize energy unembedding method uses local and neighboring qubit biases to determine which configuration results in the lowest energy. Because of this, the minimize energy unembedding method may result in more high quality configurations.

Acknowledgments

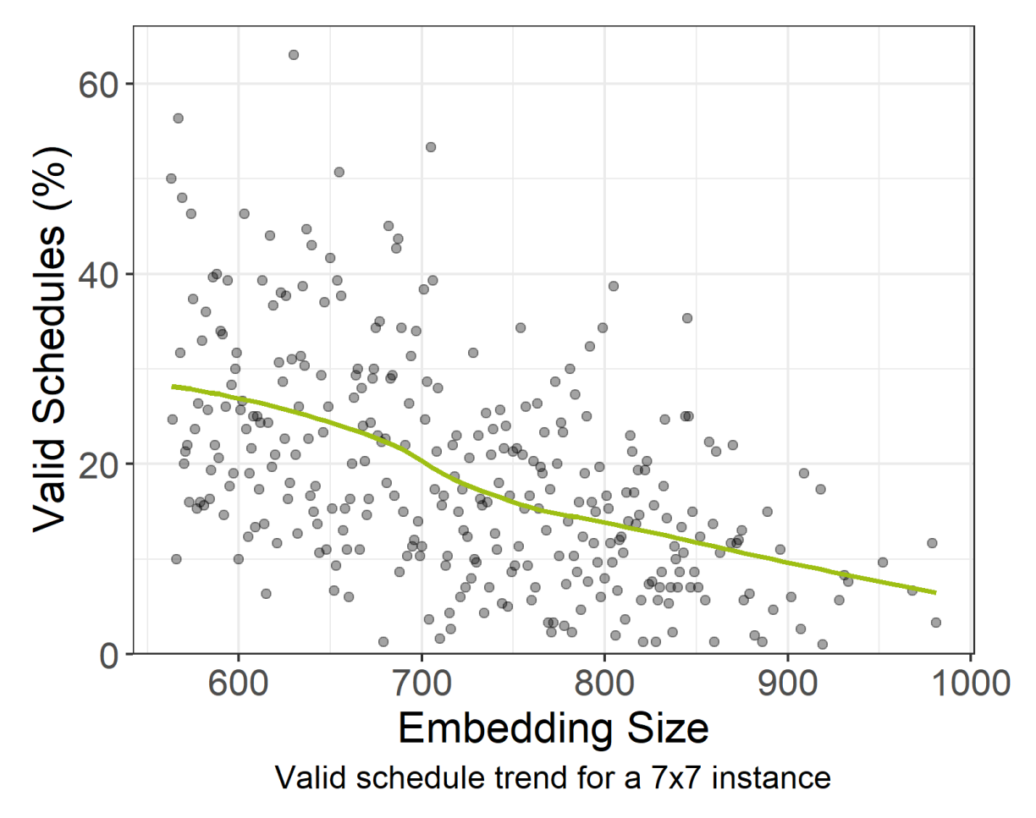
This work is partially supported by MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program ``R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications,'' Grant-in-Aid for Scientific Research(B) #16H02822 and #17H01706.

References

[1] D. Venturelli, D.J. Marchand, G. Rojo, Quantum annealing implementation of job-shop scheduling, 2015 (arXiv:1506.08479).

Results

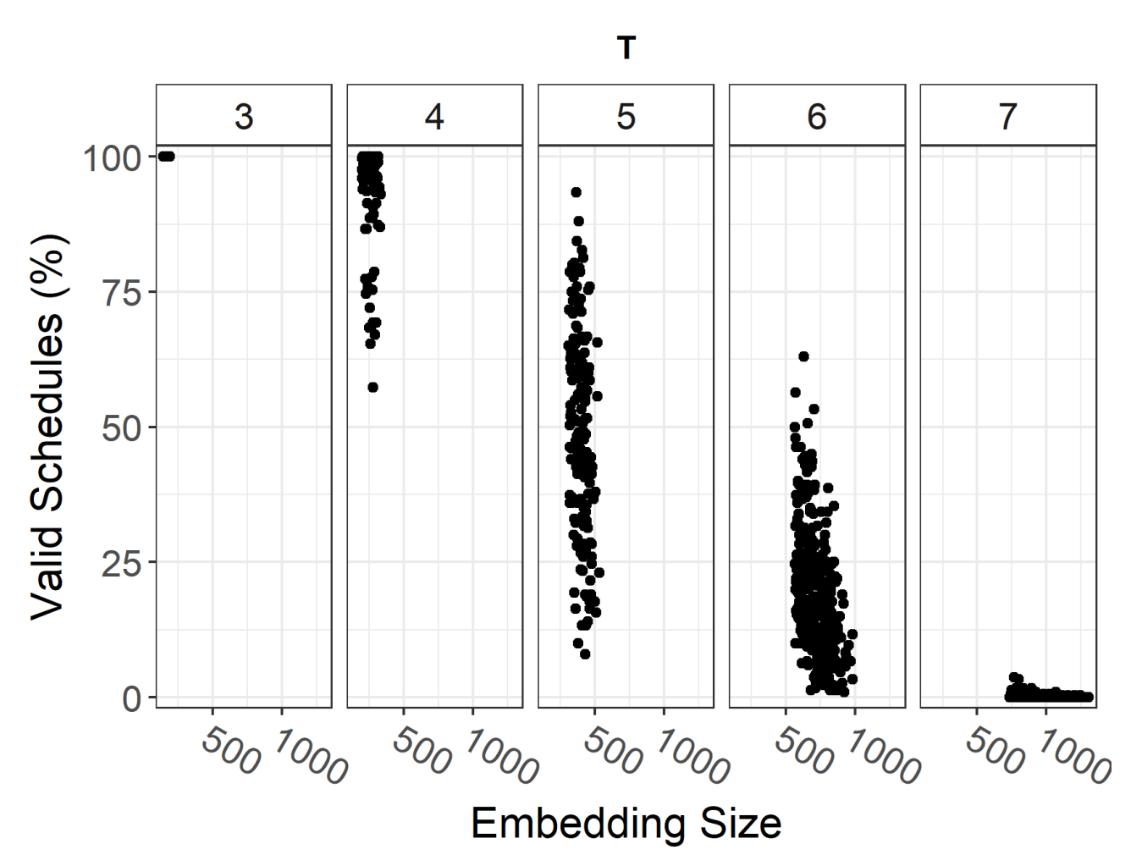
Our performance metric is the percentage of valid schedules returned from the annealer. A valid schedule breaks no constraints.



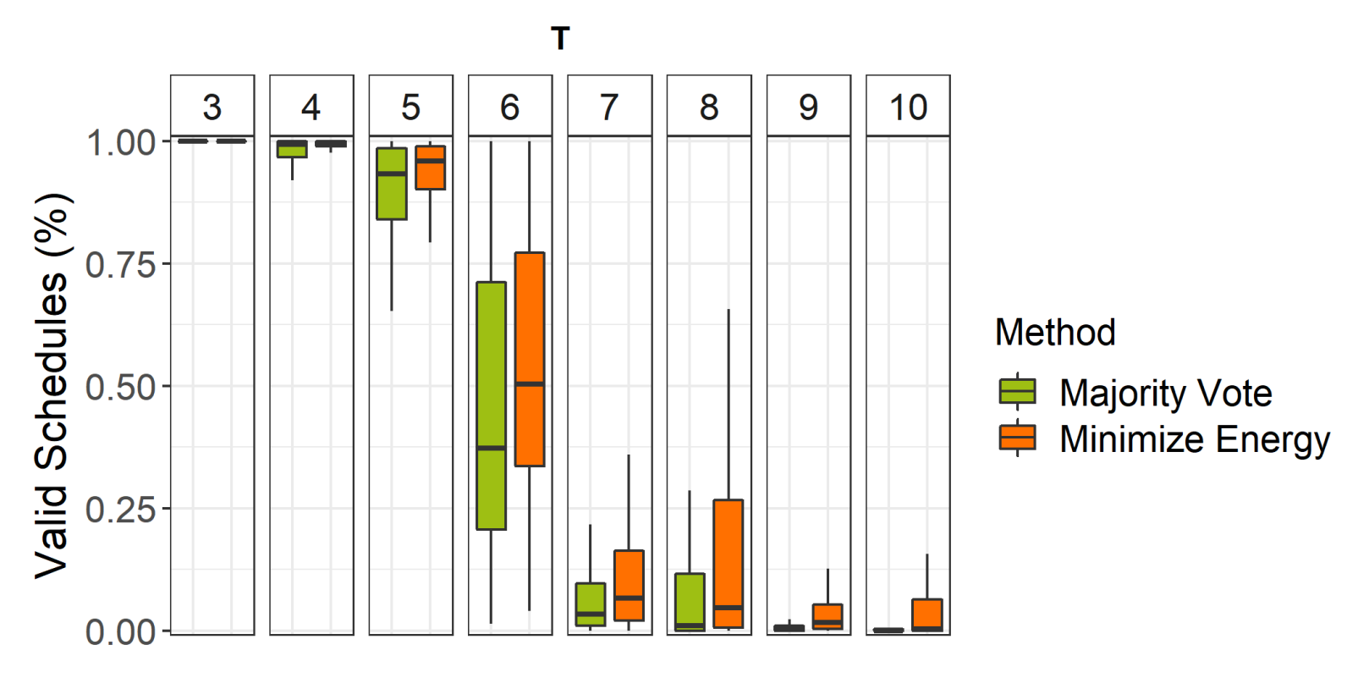
This figure shows a **negative correlation** between embedding size and the percentage of valid schedules. This indicates that the embeddings with the highest performance are found in the lower tail of the distribution of embeddings from *minorminer*.

Two main factors are responsible for this decreasing trend:

- Chain breaks
- Early freezeout



This figure shows the results for instances with increasing $\boldsymbol{T}$ values. In general, performance decreases as problem complexity increases.



The results from the unembedding methods are shown in the figure above. This clearly demonstrates that the minimize energy method consistently results in more valid schedules than majority vote method.

Conclusions

QA has shown that it has the ability to solve NP-Hard problems, but is as yet practical for certain types of problems due to various overheads in preparing a problem for QA.

Despite having only used QA on the time-indexed JSP, similar trends to those observed are expected for problems with similar properties. Problems with inefficient formulations will require many variables and not scale well for current QA hardware. Problems with densely connected variables will require qubit chaining during embedding. As the number of variables increases, embedding size, embedding size range, embedding time, and embedding failure rate also increase. For fewer variables, the embedding failure rate can be considered negligible; however, the failure rate can exceed 80% for problems with many variables.

In terms of performance, the smaller infrequently found embeddings are preferred considering the negative correlation between performance and size. Furthermore, embeddings with shorter chains are preferred as they will suffer less from early freezeout. Considering all of these factors, the best performing embeddings come at a high computational cost from repeated embedding, making a cost-performance tradeoff apparent.

Until QA hardware contains sufficient qubits to support problem sizes that are intractable for classical computers, QA will be limited in practicality for some problems.

Future Direction

One of the largest challenges currently facing QA is found in the embedding process. While next generation hardware having more qubits with higher connectivity will alleviate problems faced in embedding, there are other avenues to address these challenges and improve performance.

First is problem formulation. Problem formulation has a significant impact on variable usage and connectivity and problem dynamics. Using different formulations may provide the means to lessen some overhead (e.g. a formulation that uses fewer variables may reduce embedding time and size). Another promising area is **quantum-classical hybrid systems**. For example, a problem can be decomposed so that both classical and quantum systems have components to solve. A different application involves using QA as a high quality solution sampler that is guided by a classical search algorithm, such as a binary search tree [1].