Hybrid quantum annealing heuristic method for solving Job Shop Scheduling Problem

Krzysztof Kurowski[1], Jan Węglarz[2], Marek Subocz[1], Rafał Różycki[2], and Grzegorz Waligóra[2]

[1] Poznań Supercomputing and Networking Center affiliated to IBCH PAS, Poznań, Poland
{krzysztof.kurowski, marsub}@man.poznan.pl

[2] Poznań University of Technology, Poznań, Poland
{jan.weglarz,rafal.rozycki,grzegorz.waligora}@cs.put.poznan.pl

Abstract. Scheduling problems have attracted the attention of researchers and practitioners for several decades. The quality of different methods developed to solve these problems on classical computers have been collected and compared in various benchmark repositories. Recently, quantum annealing has appeared as promising approach to solve some scheduling problems. The goal of this paper is to check experimentally if and how this approach can be applied for solving a well-known benchmark of the classical Job Shop Scheduling Problem. We present the existing capabilities provided by the D-Wave 2000Q quantum annealing system in the light of this benchmark. We have tested the quantum annealing system features experimentally, and proposed a new heuristic method as a proof-of-concept. In our approach we decompose the considered scheduling problem into a set of smaller optimization problems which fit better into a limited quantum hardware capacity. We have tuned experimentally various parameters of limited fully-connected graphs of qubits available in the quantum annealing system for the heuristic. We also indicate how new improvements in the upcoming D-Wave quantum processor might potentially impact the performance of our approach.

Keywords: Quantum annealing · Job Shop Scheduling Problem · Heuristic.

1 Introduction

Quantum computing has attracted the attention of many researchers and provides a new challenge for solving certain classes of combinatorial problems more efficiently than on classical computers. Moreover, quantum computers are starting to approach the limit of classical simulation, and consequently entering an era of unprecedented ways to explore quantum algorithms [1]. There are many ongoing efforts to run new experiments and benchmarks with quantum computing to discover new applications and solve real-world problems. Many researchers have been trying to find efficient ways of approaching NP-hard scheduling problems over the last decades. A comprehensive study to the theory and applications of scheduling in advanced planning and computer systems as well as various practical industrial, real-time engineering, management science, business administration and information systems use cases were presented in [3]. Recently, there has been much interest in the possibility of using quantum annealing, which is a derivative of adiabatic quantum optimization, e.g. in [4], [5], [7], [9]. In general, the main assumption in adiabatic quantum optimization is that there is a quantum Hamiltonian H_P whose ground state encodes the solution to a considered problem, and another Hamiltonian H_0, whose ground state is easy-to-implement. We first prepare a quantum system to be in the ground state of a known H_0, and then adiabatically change the Hamiltonian for a time T by the following formula:

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_P \tag{1}$$

Next, if T is large enough, and H_0 and H_P do not interchange, the quantum system will remain in the ground state for all times by the adiabatic theorem of quantum mechanics. At time T, measuring the quantum state will return a solution of a considered problem.

The emerging quantum technologies support mostly 2-local interactions, thus the problem Hamiltonian H_P, containing only 2-local terms between the qubits, can be expressed by the formula:

$$H(\sigma) = \sum_{i=0}^{N-1} h_i \sigma_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{ij} \sigma_i \sigma_j \tag{2}$$

The aim is to minimize the energy of a 2-local Ising Hamiltonian function, where $h_i \in \mathbb{R}, J_{ij} \in \mathbb{R}$ and $\sigma_i \in \{-1, +1\}$. This physics formula version is often called in short Ising. Various strategies together with useful techniques for mapping a wide variety of NP-hard problems to Ising formulations to benefit from adiabatic quantum optimization have already been demonstrated, e.g., in [8]. In fact, an alternative problem formulation when translated to an objective function, the 2-local condition on the problem Hamiltonian means that the objective function can also be expressed in the form used in Operations Research community:

$$Obj(x) = \sum_{i=0}^{N-1} a_i x_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} b_{ij} x_i x_j \tag{3}$$

Thus, the problem is to minimize quadratic pseudo-Boolean objective function which is known as the Quadratic Unconstrained Binary Optimisation (QUBO), where $a_i \in \mathbb{R}, b_{ij} \in \mathbb{R}$, and $x_i \in \{1, 0\}$. In a nutshell, to program a quantum annealer we have to provide an appropriate list of a_i and b_{ij} values. One should also note that the conversion between these two formula versions requires only a linear transformation, as $x_i = \frac{(\sigma_i + 1)}{2}$.

Several questions are still open in the field of quantum computing in practice. Technical challenges include the preparation of a quantum physical system and its operation at temperatures close to absolute zero isolated from the surrounding environment in order to behave quantum mechanically. There are also various limits on the quantum system controllability, noise and imperfections. However, in the context of this paper, we address the question what scheduling problems can be solved today using the existing quantum hardware.

We believe that one of the most important applications of quantum annealing is in the category of scheduling. In this paper, however, we have limited our experiments to solve only a certain class of scheduling problems. We wanted to validate experimentally a new quantum annealing heuristic applied for solving a well-known benchmark of the classical Job Shop Scheduling Problem (JSSP). In terms of computational complexity, the JSSP is NP-hard in the strong sense [3]. Consequently, in practice optimal solutions can not be found within a reasonable time. Thus, several polynomial-time heuristics have been developed for finding suboptimal solutions and tested experimentally. Various heuristics have been proposed based on traditional models of computing to find the best-known solutions for JSSP benchmark instances. According to the recent comprehensive literature study in [10], half of them have been based on tabu search algorithms, followed by local search, shifting bottleneck, and branch and bound, and also simulated annealing techniques. The simulated annealing metaheuristic is a probabilistic technique for approximating the global optimum of a given function, successfully applied for solving various scheduling problems, including the JSSP [12] and the MRCPSP (Multi-mode Resource-Constrained Project Scheduling Problem) [13]. In principle, the classical simulated annealing metaheuristic has much in common with a physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy. There is also a mathematical analogy to the adiabatic theorem of quantum mechanics adopted in quantum annealing processes implemented in the existing quantum hardware.

The rest of this paper is organized as follows. In Section 2 we formulate the considered scheduling problem. The procedure of mapping the JSSP to the QUBO formula together with variable pruning techniques are presented in Section 3. Our heuristic approach is briefly explained in Section 4. The obtained results are presented in Section 5. We conclude our paper and present future work in Section 6.

2 Problem formulation

The JSSP can be described by a set of jobs $J = \{\mathbf{j}_1, \ldots, \mathbf{j}_N\}$ that must be scheduled on a set of machines $M = \{\mathbf{m}_1, \ldots, \mathbf{m}_R\}$. Each job $\mathbf{j}_n$ consists of a sequence of L_n operations that have to be performed in a predefined order:

$$\mathbf{j}_n = \{O_{n1} \rightarrow O_{n2} \rightarrow \cdots \rightarrow O_{nL_n}\} \qquad (4)$$

To simplify the notation, we enumerate the operations of all jobs in a lexicographical order, in such a way that:

$$\begin{aligned}\mathbf{j}_1 &= \{O_1 \to O_2 \to \cdots \to O_{k_1}\}\\ \mathbf{j}_2 &= \{O_{k_1+1} \to O_2 \to \cdots \to O_{k_2}\}\\ &\cdots\\ \mathbf{j}_N &= \{O_{k_{N-1}+1} \to O_2 \to \cdots \to O_{k_N}\}\end{aligned} \tag{5}$$

The processing time of an operation $O_i, i = 1, 2, \ldots, k_N$, is p_i which is a positive integer. Moreover, each operation requires for its processing a particular machine. Let q_i be the index of the machine $\mathbf{m}_{q_i}$ on which operation O_i is to be executed. By I_m we will denote the set of indices of all of the operations that are to be executed on machine $\mathbf{m}_m$, i.e., $I_m = \{i : q_i = m\}$. 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 be completed before the following one can start. The objective is to schedule all operations in a valid sequence in order to minimize the schedule length (or the makespan), which is the completion time of the last running job.

The JSSP is usually formulated as an optimization problem. However, it can be easily transformed into a decision problem, which is limited to decide whether there exists a feasible solution (schedule) with a makespan smaller than or equal to a given time. The JSSP can be easily mapped to a constraint satisfaction problem (CSP), and the existence of a solution to a CSP can be viewed as a decision problem. The decision version of the JSSP, together with its formulation suitable for a quantum annealing solver was presented in detail in [14]. The obtained results have encouraged us to perform further empirical investigations. We have followed the proposed time-indexed decision version of the JSSP to implement basic steps in our quantum annealing heuristic. However, we have added new conditions and extensions to be able to potentially apply it for practical use. Among many existing JSSP test instances we have decided to focus on the first benchmark set proposed in [15] to analyse various capabilities and properties supported by the D-Wave 2000Q quantum chip. In our preliminary studies, as a proof-of-concept, we have selected a small JSSP test instance denoted as *ft06* (6 jobs and 6 machines), but large enough to investigate several schedule variables and properties relevant in a quantum annealing process aiming at the next generation of D-Wave QPU.

Any JSSP instance can be represented as the disjunctive graph $G = (V, C \cup D)$, where V is the set of nodes, representing the operations of the jobs. Each node i has a weight which is equal to the processing time p_i of the corresponding operation O_i, and there are two special nodes, a source 0 and a sink $*$, whereby p_0 and p_* are equal to 0. C is the set of conjunctive arcs which reflect the job-order of all the operations, and the set off these arcs is denoted by D, see Fig. 1 for the selected *ft06* JSSP test instance where bidirectional connections representing operations executed on the same machine are depicted as nodes with the same colour for better reading. The scheduling decision is to define ordering among all those operations which have to be processed on the same machine. In the disjunctive graph representation it is done by turning undirected arcs into directed ones. A selection S defines a feasible schedule if and only if every undirected arc has been fixed, and the resulting graph $G(S) = (V, C \cup S)$ is acyclic, where S is called a complete selection. For a complete selection S the makespan is equal to the length of the longest weighted path (i.e., critical path) from the source 0 to the sink $*$ in the acyclic graph $G(S) = (V, C \cup S)$.

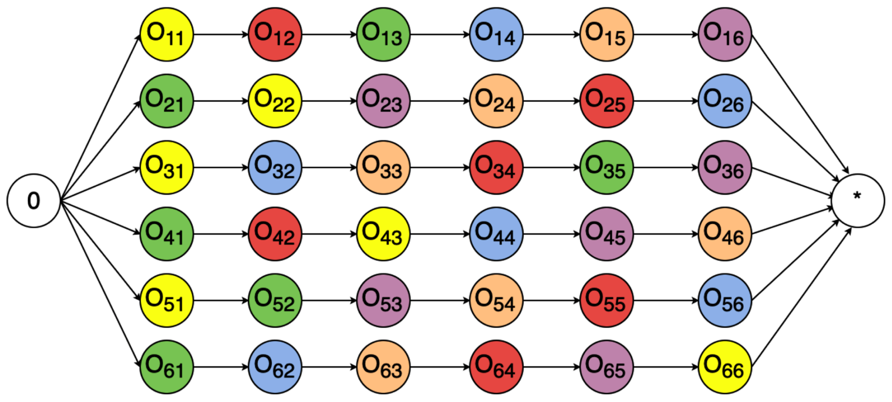

Fig. 1. The example *ft06* (6 jobs, and 6 machines) JSSP test instance represented as the disjunctive graph.

3 JSSP mapping to QUBO

In this section we present how the considered JSSP can be mapped to QUBO, and then how all the QUBO variables are embedded into a specific quantum hardware, in our case - the D-Wave 2000Q quantum processing unit (QPU). From the QUBO problem formulation perspective, it is essential to note that the QPU is a hardware implementation of an undirected graph with qubits as vertices and couplers as edges among them. For instance, in the D-Wave 2000Q QPU there are 2048 qubits logically mapped into a 16x16 matrix of unit cells of eight qubits. Each qubit is connected to at most six other qubits. The D-Wave 2000Q has been designed to solve QUBO problems, where each qubit represents a variable, and couplers between qubits represent the cost associated with qubit pairs. The limited qubits connectivity in the D-Wave 2000Q QPU is a fundamental quantum hardware property which affected the quality of obtained results and in various experiments reduced scheduling problem instances solved by quantum annealing based heuristic methods. However, according to recent publicly available technical specifications, the limited qubits connectivity will be significantly improved in the upcoming D-Wave Pegasus QPU. Therefore, while designing our new heuristic quantum annealing heuristic we have taken this upcoming feature into account.

Following the example JSSP time-indexed QUBO formulation, we define n_O*T binary variables for the JSSP, where n_O is a total number of operations and T is the upper bound for a given JSSP instance. During the initial QUBO mapping phase, we have to assign a set of binary variables for each operation and its various possible discrete starting times:

$$x_{it} = \begin{cases} 1 : \text{operation } O_i \text{ starts at time } t \\ 0 : \text{otherwise} \end{cases} \tag{6}$$

The upper bound T for the JSSP can be simply calculated as the sum of the execution times of all operations, where t is smaller than or equal to T.

3.1 Quadratic constraints and the objective function

Let us introduce a set of penalty functions and corresponding constraints expressed as quadratic constraints, as proposed in [14]. To force the order of operations within a job, the following formula was proposed to count the number of precedence violations among consecutive operations:

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

Then, there can be only one job running on each machine at any time:

$$h_2(x) = \sum_m \Big(\sum_{(i,t,k,t') \in R_m} x_{it} x_{kt'} \Big) \tag{8}$$

where $R_m = A_m \cup B_m$, and $A_m = \{(i,t,k,t') : (i,k) \in I_m \times I_m, i \neq k, 0 \leq t, t' \leq T, 0 < t'-t < p_i\}$, $B_m = \{(i,t,k,t') : (i,k) \in I_m \times I_m, i < k, t' = t, p_i > 0, p_j > 0\}$. To prevent operation O_j from starting at t' if there is another operation O_i started at time t and $t'-t < p_i$, the set A_m was defined. Then, the set B_m was defined so that two jobs can not start at the same time unless at least one of their execution time is zero.

The last penalty function expressed as the quadratic constraint was defined to force that an operation must start once and only once:

$$h_3(x) = \sum_i \Big(\sum_t x_{it} - 1 \Big)^2 \tag{9}$$

The considered JSSP Hamiltonian as the objective function can be expressed simply as the sum of the quadratic constraints defined above:

$$H_T(x) = \eta h_1(x) + \alpha h_2(x) + \beta h_3(x) \tag{10}$$

The penalty constants η, α, and β must be larger than 0 and in practice setup experimentally to ensure that unfeasible solutions do not have a lower energy than ground states. One should also note that the index of H_T indicates the strong dependence of the Hamiltonian on the timespan T, which affects the number of variables, and is one of the critical challenges for running JSSP time-indexed QUBO formulations for reference JSSP benchmark instances on the D-Wave 2000Q QPU.

3.2 Variable pruning

One of the challenging tasks in our experimental studies was to eliminate as many as possible variables called the variable pruning process. We disabled all of the variables $0 \leq x_{ij} < S$, where S is the sum of execution times of all the operations prior to the considered one in the same job. Then, we also disable all of the variables $T - S \leq x_{ij} < T$, where S is the sum of execution times of all the operations after the considered one in the same job. Finally, in our public available source code, we introduced a set of parameters for selecting any point in schedule time and corresponding variable or removing a certain time-frame in a schedule.

4 Heuristic

In order to eliminate variables in the JSSP time-indexed QUBO formulation, and consequently be able to run bigger problem instances on the limited number of qubits available, we proposed a new hybrid heuristic method which extends the basic variable pruning techniques. The main idea behind the heuristic is to define a processing window and move it in time till the end of a schedule, so only a limited number of operations is considered. In other words, we iterate the processing window in time, and check all the operations if they fit into one of three categories, where:

- s_i is the start time of operation i;
- w_{begin} is the start time of the processing window;
- w_{end} is the end time of the processing window;
- p_i is the execution time of operation i.

A schematic view of the processing window used in our heuristic is presented in Fig. 2. The example operations filled with the red color are reaching out of the processing window, and therefore they are treated differently. Given that operations A and B belong to the same job, the operation A will not be scheduled when operation B occurs, even though operation B will be removed from the schedule.

A: Inside the processing window

$$\begin{cases} s_i \geq w_{begin} \\ s_i + p_i < w_{end} \end{cases} \tag{11}$$

We include those operations in a new dictionary of jobs, creating a smaller instance.

B: Reaching out of the processing window from the left side

$$\begin{cases} s_i < w_{begin} \\ s_i + p_i < w_{end} \end{cases} \tag{12}$$

Thanks to our modular heuristic implementation we were able to prune additionally variables corresponding to the machines when those operations occupied them and prevent the next operation in the same job from starting before the one ends.

C: Reaching out of the processing window from the right side

$$s_i + p_i \geq w_{end} \tag{13}$$

Respectively to **B**, we were able to disable a machine when those operations take place and prevent the previous operation in the same job from interfering with this one.

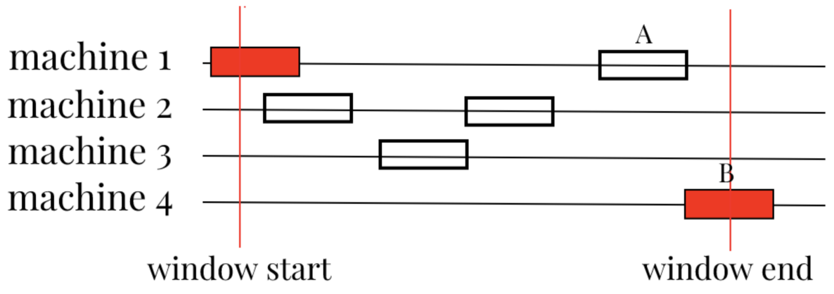

Fig. 2. Dividing a schedule into a set of processing windows for many machines and job operations.

5 Results

5.1 Jobs vs. operations

First, we wanted to check which parameters are especially crucial for the quality of generated solutions. In order to ensure that no other parameters interfered with the obtained results, we designed the second simple experiment. During the experiment, we compared schedules of one job consisting of many operations with many jobs consisting of only one operation. Each operation execution time was equal to 1, and it was performed on the same machine, see Fig. 3.

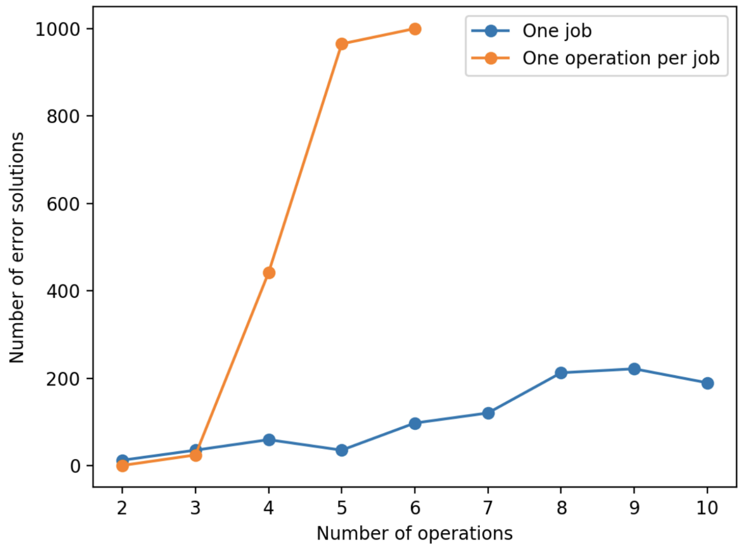

Fig. 3. The quality of solutions for a different number of jobs and operations.

Based on the performed parameters tuning experiments, we discovered that the maximum processing window was around 14 time-units to be able to run the JSSP *ft06* instance on the D-Wave 2000Q QPU. Additionally, during the pre-processing step we defined three classes of job operation length, namely *short*, *medium*, *long*, to reduce a number of time-indexed variables and the processing window size down to 5 time-units. The example initial step in our heuristic improving the scheduling solution by turning undirected arcs into directed ones within the given processing window are presented in Fig. 4.

5.2 Embedding and qubits chain strength

In Section 3 we represented the JSSP in the QUBO form as a theoretical graph of qubits and corresponding variables. However, the next challenge in our ex-

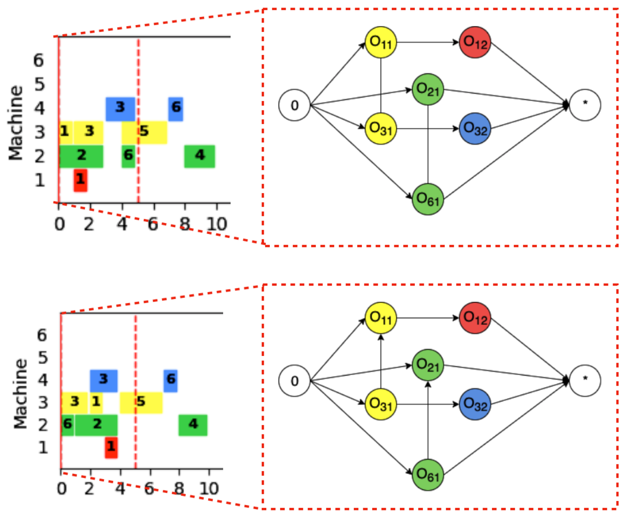

Fig. 4. The initial disjunctive graph representation of the feasible scheduling solution within a processing window with the undirected arcs O_{11}, O_{31} and O_{21}, O_{61} turned by the heuristic into directed arcs during the optimization process.

perimental studies was the embedding procedure of QUBO structures onto the quantum hardware with limited fully-connected graphs of qubits. The graph configuration implemented in the existing D-Wave hardware is called a Chimera structure. In fact, the tested D-Wave 2000Q QPU is a lattice of interconnected qubits. While some qubits connect to others via couplers, the D-Wave QPU is not fully connected, and qubits are arranged in sparsely connected groups of at most six other qubits. Currently, D-Wave provides a set of automated programming tools and APIs to find and perform an embedding automatically. Nevertheless, many quits must be chained together, so the chain is used as a single qubit. The chain strength value c must be set up carefully, as there is no methodology for choosing an optimal value [18]. To evaluate the embedding procedure we designed the following experiment: we randomly generated a set of schedules consisting of 4 jobs, each job consisted of 4 operations, and we assigned relatively short execution times to all operations, so the makespan was $T \leq 7$, and we solved the JSSP problem 100 times. Note that long schedules impose a large number of variables for the JSSP time-indexed QUBO formulation, and then the embedding procedure. Thus, we used different values of the chain strength c to discover what is optimal for our problem, see Fig. 5. Based on the obtained results, we were able to assign the strength value $c = 3$, but we observed a significant impact of its different values on the number of error solutions generated by the D-Wave quantum annealer. To quantify the quality of a generated solution, we simply counted the number of feasible schedules after the quantum optimization procedure.

Additionally, we have performed various experiments with our heuristic to make sure that all the relevant controlling parameters and configurations were used efficiently. In particular, we have selected the $minimum\_classical\_gap = 2.0$ value experimentally to provide enough energy to differentiate the ground state from other states in the QPU. Naturally, we can reduce this value and try to squeeze more problem information onto the QPU, but consequently it will also reduce the accuracy of the considered problem. In our case, the biggest constraint is a number of possible variables within a processing window. Therefore, by decreasing the $minimum\_classical\_gap = 1.0$ we had an opportunity to increase the processing window to 6 time-units, and further decreasing of the $minimum\_classical\_gap = 0.5$ value increased the processing window to 7 time-units. However, *minimum_classical_gap* values below 2.0 gave more and more unacceptable solutions and had a negative impact onto the overall quality of obtained solutions.

The existing Chimera D-Wave QPU architecture and available APIs give developers a lot of flexibility to implement and improve strategies by increasing the gap between ground and excited states during the quantum annealing process. A set of interesting error suppression techniques using quantum annealing correction with auxiliary qubits and the energy gap were discussed for instance in [11]. Technically speaking, the D-Wave QPU minimizes the energy of an Ising spin configuration whose pairwise interactions lie on the edges of the Chimera graph. To automatically minor-embeds our problem into a structured sampler we used the *EmbeddingComposite* method. However, there are other *dimod composites* with various parameters during pre- and post-processing phases worth to consider in our future work. One should note, that the range of coupling strengths available in the D-Wave QPU is finite, so chaining is typically accomplished by setting the coupling strength to the largest allowed negative value and scaling down the input couplings relative to that. Thus, we have also checked various values of the another controlling *extended_j_range* parameter to increase the strength of minor embedding coupling. Typically, using the available larger negative values of J increases the dynamic J range. According to technical specifications of the existing D-Wave 2000Q QPU, strong negative couplings can bias a chain and therefore flux-bias offsets must be applied to recalibrate it to compensate for this effect. However, we have not noticed a significant impact on the quality of obtained solutions by changing the *extended_j_range* parameter. Nevertheless, we believe that additional tests and techniques will be required, in particular the spin-reversal transform can improve results by reducing the impact of possible analog and systematic errors. We are planning to compare all the above mentioned controlling parameters on the new Pegasus QPU architecture and explore more precisely the JSSP problem structure and solution space. We will apply new techniques for encoding discrete variables into Ising model qubits, e.g. [6], and try to take advantage of new and more connected Pegasus QPU graph for more efficient embedding of the considered problem.

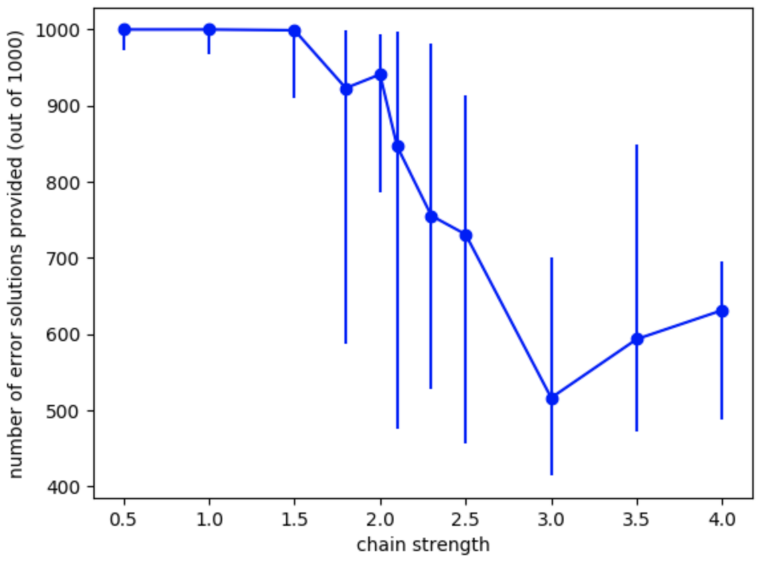

Fig. 5. The chain strength c bars based on the standard deviation and their impact on the JSSP solutions quality.

Finally, our hybrid heuristic method improved the quality of solutions by reducing the makespan of randomly generated schedules down to 60, see Fig. 6. Nevertheless, it is clear that even simple classical heuristics proposed for the JSSP outperform our quantum annealing-based approach reaching a set of many schedules with optimal makespan 55. We expect to use a much bigger processing windows in our heuristic tailored for the upcoming D-Wave Pegasus QPU architecture. The Pegasus graph will allow each qubit to couple to 15 other qubits instead of 6 qubits, so we expect to run the same heuristic successfully for bigger JSSP instances. Our proof-of-concept implementation, including the heuristic source code, has been published in the GitHub repository for reusing and external testing [19].

6 Conclusions

We presented a new quantum annealing heuristic for solving the Job Shop Scheduling Problem (JSSP) on publicly available D-Wave 2000Q QPU. Due to a limited number of available qubits and couplers among qubits implemented in a specific topology, we decomposed the JSSP into a set of smaller optimization problems as window processing slices. We estimated the number of feasible solutions during experimental studies on the well-known scheduling JSSP test instance *ft06*. We also tuned experimentally QUBO parameters proposed for the

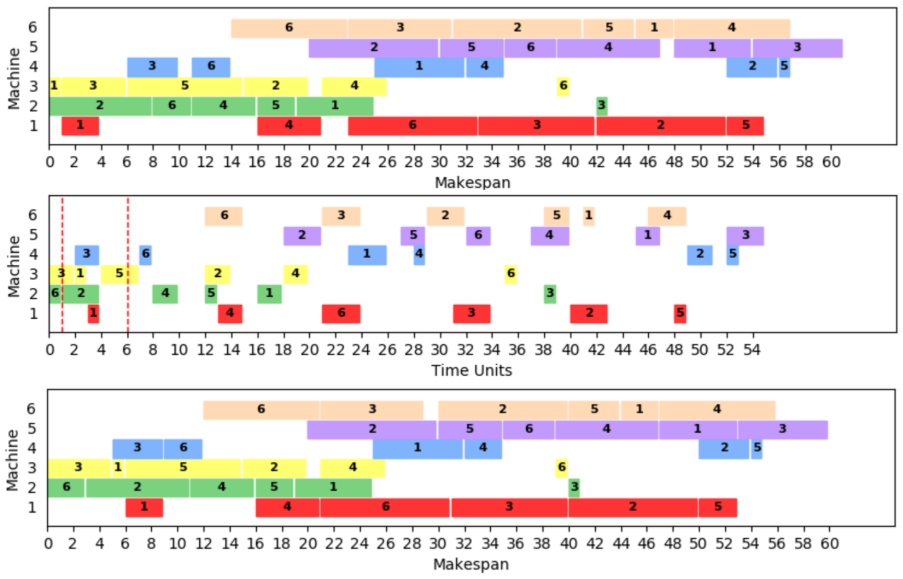

Fig. 6. The incremental optimization process of improving schedule quality within the given processing window generated by the hybrid quantum annealing heuristic for the selected JSSP *ft06* instance.

time-indexed decision version of the JSSP, and compared the obtained results with optimal solutions generated by classical heuristic methods for the well-known JSSP benchmark. Our approach can be easily extended, modified and applied to other scheduling problems by researchers, as we have released the source code of our heuristic.

In our future work, we intend to consider other scheduling problems, and test our hybrid heuristic to divide large problem domains into small subproblems. The existing heuristic has been designed to be modular, open and extensible source code, so we plan to incorporate additional heuristic techniques, and add improved variable pruning and selection algorithms. It will also be interesting to explore the design of such extensions within the existing D-Wave quantum annealers limits and upcoming Pegasus QPU improvements.

7 Acknowledgements

This research has been partially supported by the statutory funds of Poznan University of Technology and Poznan Supercomputing Networking Center affiliated to Institute of Bioorganic Chemistry Polish Academy of Sciences, grant PRACE-LAB2 POIR.04.02.00-00-C003/19.

References

1. Arute F. et al., *Quantum supremacy using a programmable superconducting processor*, Nature, 574 (2019), 505–510.
2. Błażewicz J., Lenstra J.K., Rinnooy Kan A.H.G., *Scheduling projects subject to resource constraints: classification and complexity*, Discrete Applied Mathematics, 5 (1983), 11-24.
3. Błażewicz J., Ecker K.H., Pesch E., Schmidt G., Sterna M., Węglarz J., *Handbook on Scheduling: From Theory to Practice*, Cham: Springer, 2019.
4. Kazuki I., Yuma N., Travis H.S., *Application of Quantum Annealing to Nurse Scheduling Problem*, Scientific Reports, 9, 12837 (2019), doi: 10.1038/s41598-019-49172-3.
5. Albash T., Lidar D.A., *Adiabatic quantum computation*, Reviews of Modern Physics, 90, 015002 (2018).
6. Chancellor N., *Domain wall encoding of discrete variables for quantum annealing and* $\{QAOA\}$, Quantum Science and Technology, 4:4 (2019), doi: 10.1088/2058-9565/ab33c2.
7. Humble T.S. et al. *An integrated programming and development environment for adiabatic quantum optimization*, Computational Science & Discovery, 7, 015006 (2014).
8. Lucas A., *Ising formulations of many NP problems*, Frontiers in Physics, 2:5 (2014), 1-15.
9. Boixo S., Albash T., Spedalieri F. et al. *Experimental signature of programmable quantum annealing*, Nature Communications, 4, 2067 (2013), doi: 10.1038/ncomms3067.
10. van Hoorn J.J., *The Current state of bounds on benchmark instances of the job-shop scheduling problem*, Journal of Scheduling, 21 (2018), 127-128.
11. Pudenz K.L., Albash T., Lidar D.A., *Quantum annealing correction for random Ising problems*, Physical Review A 91:4, 042302 (2015).
12. van Laarhoven P.J.M., Aarts E.H.L., Lenstra J.K., *Job Shop Scheduling by Simulated Annealing*, Operations Research, 40 (1992), 113-125.
13. Mika M., Waligóra G., Węglarz J., *Overview and State of the Art.* In C. Schwindt and J. Zimmermann (Eds.), Handbook on Project Management and Scheduling Vol. 1 (2015), International Handbooks on Information Systems, chapter 25, pp. 445-490, Cham: Springer.
14. Venturelli D., Marchand D.J.J., Rojo G., *Job Shop Scheduling Solver based on Quantum Annealing*, Quantum Artificial Intelligence Laboratory, NASA Ames U.S.R.A. Research Institute for Advanced Computer Science, 1QB Information Technologies, (2016), https://arxiv.org/pdf/1506.08479v2.pdf.
15. Fisher H., Thompson G.L., *Probabilistic learning combinations of local job-shop scheduling rules.* In J.F. Muth & G.L. Thompson (Eds.), Industrial scheduling (1963), chapter 15, pp. 225–251, Englewood Cliffs: Prentice Hall.
16. Brucker P., Jurisch B., Sievers B., *A branch and bound algorithm for the job-shop scheduling problem*, Discrete Applied Mathematics, 49 (1994), 107-127.
17. JSSP benchmark instance ft06: http://jobshop.jjvh.nl/instance.php?instance_id=6.
18. Coffrin C.J., *Challenges with Chains: Testing the Limits of a D-Wave Quantum Annealer for Discrete Optimization*, United States: N.p., (2019), doi: 10.2172/1498001.
19. Quantum annealing hybrid heuristic for JSSP: https://github.com/mareksubocz/QuantumJSP.