Comparison of D-Wave Quantum Computing Environment Solvers for a Two-Machine Jobs Scheduling Problem

Wojciech Bożejko[1(✉)], Sergii Trotskyi[1], Mariusz Uchroński[1,2], and Mieczysław Wodecki[3]

[1] Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
{wojciech.bozejko,sergii.trotskyi,mariusz.uchronski}@pwr.edu.pl
[2] Wroclaw Centre for Networking and Supercomputing, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
[3] Department of Telecommunications and Teleinformatics, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
mieczyslaw.wodecki@pwr.edu.pl

Abstract. Solving discrete optimization problems on D-Wave's quantum platform requires formulating the problem in the form of quadratic programming. In this work we consider an NP-hard two-machine flow problem with minimization of the sum of jobs weighted tardinesses. We present two formulations of the Constrained Quadratic Model (CQM) for solving the considered problem, the transformation method to QUBO, and the results of computational experiments performed in three environments: LeapHybridSampler, DWaveSampler, and Gurobi. We propose a novel approach to constructing hybrid quantum annealing algorithms generating solutions very close to or equal to the optimal ones.

Keywords: quantum annealing · quantum computing · scheduling · discrete optimization · flow shop

1 Introduction

Quantum computing represents an attractive new field of intensive research. Despite the growing capabilities of quantum machines, they are not yet a real competitor to classical approaches based on massive supercomputing calculations. This is especially visible in the case of NP-hard combinatorial optimization problems - we are actually able to solve problems with sizes of several dozen or several dozen integer variables or several hundred binary variables on quantum computers. However, classical computing systems are currently able to accurately solve tasks with sizes of tens of thousands of variables (e.g. Concorde solver for traveling salesman problem, Gurobi, CPLEX, etc.).

H. Quintián et al. (Eds.): SOCO 2024, LNNS 888, pp. 68–76, 2024.
https://doi.org/10.1007/978-3-031-75013-7_7

Among the solutions in the field of quantum computing, two approaches can be distinguished: general quantum computing, implemented by circuits with quantum gates (IBM, Google, Amazon, Intel, Alibaba Group), and quantum annealers (D-Wave, NEC, Qilimanjaro, A Star Quantum, JiJ). In this work, we use the second approach as it fits the problem of discrete optimization exceptionally well. The work is a continuation of research on the construction of accurate quantum optimization algorithms for NP-hard problems (see [4,5]). An important limitation of running calculations in the quantum annealer environment is the formulation of the task in the form of quadratic or linear programming without constraints. In practice, this requires a precise formulation of the task without the use of recursion (very useful in the current practice of model construction), and the constraints - in order to "build" them into the objective function, must be at most linear. In this work, we formulate this type of model for a certain task scheduling problem.

We consider a two-machine flow problem with minimizing the sum of late costs (*total weighted tardiness minimization in two-machine flow shop problem*), which in the literature is denoted by $F2||\sum w_iT_i$. Each of n jobs must be completed successively on the first one and then on the second one machine. The data includes job execution times and the latest dates for their completion (on the second machine). Exceeding this deadline will result in a penalty being charged, depending on the extent of the delay and a constant penalty factor. You need to determine the order in which the jobs should be performed (the same on both machines). minimizes the sum of penalties. A similar problem with minimizing the deadline for completing jobs (criterion $C_{\max}$) is polynomial (Johnson's algorithm with complexity $O(n \ln n)$, where n is the number of jobs). In the case of a larger (than two) number of machines, the problem with the $C_{\max}$ criterion belongs to the class of the most difficult ones, *NP-hard*, and is one of the classic discrete optimization problems today. Many of its specific properties have been proven (e.g., blocks from the critical path), which are successfully used in the construction of very effective exact and approximate algorithms.

In this paper, we propose using the quantum computing environment to solve the NP-hard task scheduling problem. The main disadvantage of calculations on quantum computers is their non-determinism. Therefore, there is no guarantee of optimality of solutions to discrete optimization problems determined on a quantum computer. We propose an innovative approach to constructing hybrid algorithms that alternately use calculations performed on a classical computer and a quantum computer, guaranteeing the optimality of the solution.

2 Related Works

Problem $F2||\sum w_iT_i$ is a generalization of the *NP-hard*, the single-machine problem with the same criterion, i.e., minimization sum of late penalties $\sum w_iT_i$. The single-machine problem has a long history, over 50 years old. Over the years, many exact and approximate algorithms have been published. Its description, specific properties elimination, and a very effective algorithm based on the tabu

search method can be found in the work of Bożejko, Grabowski, and Wodecki [3].

There are relatively few works devoted to examining the $F2||\sum w_iT_i$ problem in terms of specific properties and methods of solving it. Ruiz and Stützle [12] presented a greedy algorithm, Liao et al. [9] method-based algorithm searching with taboos. Some theoretical and good results approximation algorithms are presented in the works: Lin [10] and Bulfin i Hallah [6]. There are much more works devoted to two-machine problems with criteria other than those considered in this work. Algorithms optimal (based on the B&B scheme) for certain variants problems with sum costs criteria are described in the works: Moukrim et al. [11] and Hamdi et al. [8]. As the authors write, within a reasonable time, it is possible solve examples with the number of jobs not exceeding 50. There is much more work on approximate algorithms: construction, metaheuristics and their hybrids. Especially interesting works published in the last few years: Ahmadi et al. [1], Cheng et al. [7], and also Ardakan et al. [2].

In this paper, we present a description of the problem under consideration and formulate two of its mathematical models as constrained quadratic programming tasks. This is a natural way of formulating tasks for the D-Wave quantum platform. We also present the results of the computational experiments performed.

3 Description of the Problem and Its Mathematical Model

Two-machine permutation flow problem with minimizing the total costs of delays can be formulated as follows:

TT2FS Problem: Given a set of jobs

$$\mathcal{J} = \{1, 2, \ldots, n\},$$

Job and the set of machines

$$\mathcal{M} = \{1, 2\}.$$

Job $j \in \mathcal{J}$ consists of two operations O_{j1}, O_{j2}. The operation O_{jk} corresponds to the action executing job j on machine k. For job $i \in \mathcal{J}$ we define:

p_{ik} - *execution time* of operation O_{ik},
d_i - *requested term endings*,
w_i - *missing penalty coefficient*.

Each job must be performed on both machines, and they must be fulfilled the following limitations:

(a) each job must be performed on first, and then on the second machine,
(b) the execution of the job cannot be interrupted,

(c) job can only be performed on one machine at a time,
(d) the machine cannot execute more than that at the same time than one job,
(e) order of job execution, on both machines, it must be the same.

Any order of job execution that satisfies constraints (a)–(e) may be represented by some permutation of elements (jobs) from the set $\mathcal{J}$. Let Φ be the set of all such permutations.

For permutation $\pi \in \Phi$ (a fixed order of execution of jobs on machines), let $S_{\pi(i),j}$ will be the starting point of the operation $O_{\pi(i),j}$ ($i \in \mathcal{J},\ j = 1, 2$). From constraints (b) and (c) it follows that $C_{\pi(i),j} = S_{\pi(i),j} + p_{\pi(i),j}$ is the moment of termination of the operation $O_{\pi(i),j}$. These moments can be determined from the following relationships recursive:

$$C_{\pi(i),j} = \max\{C_{\pi(i-1),j},\ C_{\pi(i),j-1}\} + p_{\pi(i),j}, \quad i = 1, 2, \ldots, n,\ \ j = 1, 2, \quad (1)$$

with initial conditions:

$$C_{\pi(0),j} = 0, \quad j = 1, 2 \text{ and } C_{\pi(i),0} = 0,\ \ i = 1, 2, \ldots, n, \quad (2)$$

or non-recursive

$$\mathcal{C}_k(\pi) = C_{\pi(k),2} = \max\left\{\sum_{i=1}^{k-s} p_{\pi(i),1} + \sum_{i=k-s}^{k} p_{\pi(i),2} : s = 0.1, \ldots, k-1\right\}, \quad (3)$$

for $k = 1, 2, \ldots, n$. By $\mathcal{C}_k(\pi) = C_{\pi(i),2}$ we denote the deadline for completing the job $\pi(i)$, i.e. operation $O_{\pi(i),2}$. Then

$$T_{\pi(i)} = \max\{0, \mathcal{C}_{\pi(i)} - d_{\pi(i)}\} \quad (4)$$

is the *tardiness* of job $\pi(i) = w_{\pi(i)} \cdot T_{\pi(i)}$ *penalty* for being late (in other words - *cost of execution jobs*). If $T_{\pi(i),} = 0$, then the job is called *timely*, and otherwise - *late*. *Penalty* for late completion of jobs (abbreviated as *solution cost*)

$$\mathcal{T}(\pi) = \sum_{i=1}^{n} w_{\pi(i)} \cdot T_{\pi(i)}. \quad (5)$$

In the problem under consideration, the order of execution should be determined jobs that minimizes the sum of penalties for late jobs, i.e. permutation optimal $\pi^* \in \Phi$ for which

$$\mathcal{T}^* = \mathcal{T}(\pi^*) = \min\{\mathcal{T}(\pi) :\ \pi \in \Phi\}. \quad (6)$$

4 Constrained Quadratic Model

In this chapter, we propose two constrained quadratic programming models for the considered problem. They allow the solution (or its upper bound if optimization is not exact) to be determined by a quantum annealer.

4.1 First Model with Completion Times

In this case, we use a binary matrix x to remember the order of jobs (i.e. permutations π) and auxiliary variables $\mathcal{C}_k$ representing the completion times of the jobs on the second machine.

Goal to minimize:

$$\min_{\mathbf{x},\mathcal{C},T} \sum_{i=1}^{n}\sum_{j=1}^{n} w_j x_{ij} T_i \tag{7}$$

with constraints (subject to):

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\ldots,n, \tag{8}$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,2,\ldots,n, \tag{9}$$

$$\mathcal{C}_k \geq \sum_{i=1}^{k-s}\sum_{j=1}^{n} p_{j,1}x_{ij} + \sum_{i=k-s}^{k}\sum_{j=1}^{n} p_{j,2}x_{ij},\ \ k = 1,2,\ldots,n,\ \ s = k-1,k-2,\ldots,0, \tag{10}$$

$$T_k \geq \mathcal{C}_k - \sum_{j=1}^{n} x_{kj}d_j,\ k = 1,2,\ldots,n, \tag{11}$$

$$T_k \geq 0,\ k = 1,2,\ldots,n, \tag{12}$$

$$x_{ij} \in \{0,1\},\ i,j = 1,2,\ldots,n, \tag{13}$$

$$\mathcal{C}_k \in \mathbb{N} \cup \{0\},\ k = 1,2,\ldots,n. \tag{14}$$

The matrix $\mathbf{x} = [x_{ij}]_{n\times n}$ is interpreted as follows: if $x_{ij} = 1$, then a job j is on the position i in the schedule (zero otherwise).

4.2 Second Model

In this model, we propose to use fewer variables by reducing the use of $\mathcal{C}_k$ variables, while maintaining x and $\mathcal{T}_k$.

Goal to minimize:

$$\min_{\mathbf{x},T} \sum_{i=1}^{n}\sum_{j=1}^{n} w_j x_{ij} T_i \tag{15}$$

with constraints (subject to):

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\ldots,n, \tag{16}$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,2,\ldots,n, \tag{17}$$

$$T_k \geq \sum_{i=1}^{k-s}\sum_{j=1}^{n} p_{j,1}x_{ij} + \sum_{i=k-s}^{k}\sum_{j=1}^{n} p_{j,2}x_{ij} - \sum_{j=1}^{n} x_{kj}d_j,$$

$$k = 1, 2, \ldots, n,\ s = 0, 1, \ldots, k-1, \tag{18}$$

$$T_k \geq 0,\ k = 1, 2, \ldots, n, \tag{19}$$

$$x_{ij} \in \{0, 1\},\ i, j = 1, 2, \ldots, n. \tag{20}$$

The matrix $\mathbf{x} = [x_{ij}]_{n\times n}$ is interpreted as follows: if $x_{ij} = 1$, then a job j is on the position i in the schedule (zero otherwise).

5 Computational Experiments

A quantum machine and a computing environment provided by the Canadian company D-Wave were used to carry out the calculations. The proposed algorithm was launched in two versions: (i) in the LeapHybridCQMSampler environment using the CPU, GPU as well as QPU (with automatic defining tasks for the QPU, which performs quantum annealing) and (ii) in the D-WaveSampler environment (with native use of quantum annealing). A discussion of the QPU usage in LeapHybridSampler can be found in the work of Stogiannos et al. [13]. To compare the quality of the obtained results, the optimal algorithm was run in the Gurobi environment, determining the value of $\mathcal{T}^*$.

UB (Upper Bound) formulation is given as CQM – Constrained Quadratic Model. Now, we have two possibilities to compute them on D-Wave:

(i) run them in *LeapHybridSolver* environment (on CPUs, GPUs and QPU), or
(ii) convert them to BQM (Binary Qaudratic Model) and QUBO and run as hardware Quantum Annealing procedure.

The second approach we can achieve by using `dimod.cqm_to_bqm` procedure of D-Wave Ocean SDK:

```
cqm_to_bqm(cqm: ConstrainedQuadraticModel, lagrange_multiplier:
float | floating | integer | None = None) -> Tuple
[BinaryQuadraticModel, CQMToBQMInverter]}
```

The calculations were performed in the D-Wave Leap environment using the following:

(i) `hybrid_constrained_quadratic_model_version1p` solver – `LeapHybridCQM`
`Sampler` class from `dimod` package,
(ii) `Advantage_system6.4` solver – `DwaveSampler` class from `dimod` package,

and run on a machine using *Pegasus* topology type. The number of active couplers for the D-Wave solver was 40297, and the number of active qubits 5627.

Table 1 contains the results of computational experiments on test data for the considered two-machine problem. Optimal solutions are marked in bold (the

Table 1. Results of computational experiments of upper bounds calculation

run	*n*	LeapHybridCQMSampler			DWaveSampler				Gurobi
		UB	t_{QPU} [ms]	t_{RUN} [s]	UB	t_{QPU} [ms]	No.int	No.var	$\mathcal{T}^*$
1	3	**62**	31.96	5.14	1005	16.13	1464	93	**62**
2	4	**77**	32.02	5.21	2879	15.93	2963	142	**77**
3	5	**163**	32.04	5.26	7065	16.01	6245	224	**163**
4	6	**320**	16.04	5.04	*	*	12315	336	**320**
5	7	**197**	32.00	5.08	*	*	20104	443	**197**
6	8	**191**	16.02	5.00	*	*	31209	564	**191**
7	9	558	31.93	5.35	*	*	46506	699	**426**
8	10	654	16.01	5.02	*	*	73349	912	**516**

* embedding onto QPU not possible

reference model in Gurobi was determined by optima - the far right column of the table). The following columns contain: *run* run number; example size n; upper bound value UB for the LeapHybridCQMSampler solver (implementing hybrid computations combining metaheuristics on the CPU and quantum annealing on the QPU); running time of this solver on QPU (QPU_ACCESS_TIME); total running time of this solver (RUN_TIME); upper bound value UB for the native DWaveSampler solver (implementing only quantum annealing); running time of this solver on QPU (QPU_ACCESS_TIME); number of interactions of BQM model (No.int); number of binary variables (No.var) of BQM model. Due to the complexity of the representation, relatively small examples were selected for testing ($n = 3, 4, \ldots, 10$), which, however, already for $n \geq 6$ the model was too big for embedding on the QPU quantum processor – the number of interactions between variables for $n = 6$ is already 12315 and the number of binary variables is 336, which exceeds the possibilities of embedding this model on a quantum processor – given the current state of quantum hardware capabilities.

The main goal of the computational experiments performed was to compare the effectiveness of the LeapHybridCQMSampler and DWaveSampler models. For sizes from $n = 3$ to 8, the results obtained by LeapHybridCQMSampler are optimal (equal to Gurobi's results). For sizes $n = 7$ and 8 only approximate solutions were obtained. The lack of optimal solutions results from the fact that the number of variables and interactions between them (see No.var and No.int columns in Table 1) exceeds the possibilities of embedding the D-Wave quantum processor in the *Pegasus* topology. The average PRD error for LeapHybridCQMSampler's UB results is 7%. The gun time was practically constant and amounted to about 5 s.

In turn, DWaveSampler, which performs native quantum annealing, was only able to recalculate examples with sizes $n = 3.4$ and 5, with a large error. It is worth emphasizing the very short operating time of this model, around 16 milliseconds.

The Fig. 1 shows the Percentage Relative Devatyions (PRD) of LeapHybrid-CQMSamper compared to the Gurobi package solutions for different input data sizes $n = 3, 4, \ldots, 10$ (marked with a solid line). Additionally, the running times t_{RUN} of the algorithm are marked with a dashed line.

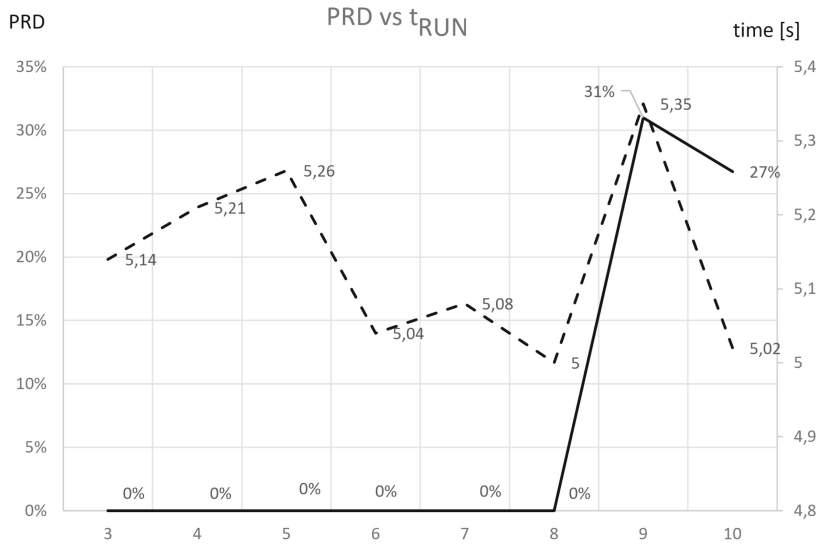

Fig. 1. Comparison of LeapHybridCQMSampler's PRD relative errors (solid line) and t_{RUN} runtimes (dotted line).

6 Conclusions

Solving discrete optimization problems on the D-Wave quantum platform requires formulating them as a constrained quadratic programming problem. For many *NP-hard* problems this is not easy, because it is necessary to use an additional large number of variables and constraints. Therefore, the size of the task increases significantly, making its solution more difficult. Despite, performing calculations on a quantum computer creates new possibilities in constructing new, effective optimal algorithms. The work considered the *NP-hard* problem of scheduling jobs on two machines. Models of the problem were formulated, enabling calculations on the D-Wave quantum platform, and the results of the conducted computational experiments.

Future research directions are planned to further investigate the model's effectiveness and efficiency across various variables representation, taking into account the possibility of embedding larger problems on the QPU. Additionally, integration with other advanced computational techniques (such as reducing the

degree of binary polynomials to QUBO) are planned to enhance the algorithm's overall efficiency and performance.

Acknowledgment. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2024/017186. Calculations have been partially carried out using resources provided by Wroclaw Centre for Networking and Supercomputing (http://wcss.pl), grant No. 96.

References

1. Ahmadi-Darani, M.-H., Moslehi, G., Reisi-Nafchi, M.: A two-agent scheduling problem in a two-machine flowshop. Int. J. Ind. Eng. Comput. **9**(3), 289–306 (2018)
2. Ardakan, M.A., Beheshti, A.K., Mirmohammadi, S.H., Ardakani, H.D.: A hybrid meta-heuristic algorithm to minimize the number of tardy jobs in a dynamic two-machine flow shop problem: numerical. Algebra Control Optim. **7**(4), 465–480 (2017)
3. Bożejko, W., Grabowski, J., Wodecki, M.: Block approach tabu search algorithm for single machine total weighted tardiness problem. Comput. Ind. Eng. **50**(1–2), 1–14 (2006)
4. Bożejko W., Burduk A., Pempera J., Uchroński M., Wodecki M.: Optimal solving of a binary knapsack problem on a D-Wave quantum machine and its implementation in production systems. Ann. Oper. Res. (2024, in press). https://doi.org/10.1007/s10479-024-06025-1
5. Bożejko W., Pempera J., Uchroński M., Wodecki M.: Quantum annealing-driven branch and bound for the single machine total weighted number of tardy jobs scheduling problem. Future Gener. Comput. Syst. **155** 245–255 (2024). https://doi.org/10.1016/j.future.2024.02.016
6. Bulfin, R.L., Hallah, R.: Minimizing the weighted number of tardy jobs on two-machineflow shop. Comput. Oper. Res. **30**, 1887–1900 (2003)
7. Cheng, S.-R., Yunqiang, Y., Wen, Ch.-H., Lin, W.-CH., Wu, Ch.-Ch., Liu, J.: A two-machine flowshop scheduling problem with precedence constraint on two jobs. Soft Comput. Fusion Found. Methodol. Appl. **21**(8), 2091–2103 (2017)
8. Hamdi, I., Oulamara, A., Loukil, T.: A branch and bound algorithm to minimise the total tardiness in the two-machine permutation flowshop scheduling problem with minimal time lags. Int. J. Oper. Res. **23**(4), 387–405 (2015)
9. Liao, C.J., Liao, L.M., Tseng, C.T.: A performance evaluation of permutation vs. non-permutation schedules in a flowshop. Int. J. Prod. Res. **44**(20), 4297–4309 (2006)
10. Lin, B.M.T.: Scheduling in the two-machine flowshop with due date constraints. Int. J. Prod. Econ. **70**, 117–123 (2001)
11. Moukrim, A., Rebaine, D., Serairi, M.: A branch and bound algorithm for the two-machine flowshop problem with unit-time operations and time delays. RAIRO-Oper. Res. **48**, 235–254 (2014)
12. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **177**(3), 2033–2049 (2007)
13. Stogiannos, E., Papalitsas, C., Andronikos, T.: Experimental analysis of quantum annealers and hybrid solvers using benchmark optimization problems. Mathematics **10**(8), 1294 (2022)