An Exact Quantum Annealing-Driven Branch and Bound Algorithm for Maximizing the Total Weighted Number of on-Time Jobs on a Single Machine

Wojciech Bożejko[1(✉)], Jarosław Pempera[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,jaroslaw.pempera,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. This paper considers the problem of maximizing the total weighted number of on-time jobs on a single machine. Using the problem as a case study, we present a new approach to solving *NP*-hard discrete optimization problems using D-Wave's QPU quantum processor architecture implementing quantum annealing. Although optimization on a quantum machine does not guarantee optimality, the hybrid method of construction of the partitioning and constraint algorithm proposed in this paper, using together CPU and QPU as well as Lagrange relaxation for upper bounds determination, makes it possible to determine the exact, optimal solution.

Keywords: Scheduling · Quantum machine · Optimization

1 Introduction

The effective use of today's advanced computing tools, including quantum machines, is associated with some specific limitations and requires a personalized approach. Solving practical problems on a quantum machine is a major challenge today due to the small scale of solvable problems and the lack of a high-level approach, adequate to the high-level languages in programming classical computers.

Currently available quantum computers represent several quantum computing concepts. Among those offered are quantum computers based on the quantum

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
M. Pawelczyk et al. (Eds.): PCC 2023, LNNS 709, pp. 79–89, 2023.
https://doi.org/10.1007/978-3-031-35173-0_8

gate model (e.g. developed by IBM and Google). Another approach from quantum computing is to use the process of quantum annealing. D-Wave Systems provides quantum computers that enable computations that are admittedly limited to the use of quantum annealing, but which fit perfectly with the needs of the discipline of operations research. In this case, instead of expressing the algorithm for solving the problem under study in terms of quantum gates, the user presents it as a binary quadratic programming problem. Typically, this is a non-trivial task. In this paper, we present an exact hybrid algorithm combining quantum solving of an NP-hard job scheduling problem, whose construction is based on the *Branch and Bound* (B&B) method. We propose a new approach by determining the upper and lower bounds of the objective function on a quantum computer. For the determination of the upper bound, we formulate a binary quadratic programming problem with constraints. In contrast, for the determination of the lower bound, we use the Lagrange relaxation method. Although calculations on a D-Wave machine do not guarantee optimality, they can be successfully used in the construction of optimal algorithms.

2 Problem Formulation

The paper considers one of the simplest to formulate (although NP-hard) job scheduling problems on a single machine, which can be formulated as follows. **Maximise the total weight of on-time jobs** (abbreviated as TWOT). There is a *set of jobs* $\mathcal{J} = \{1, 2, ..., n\}$ given which should be executed on a single machine. We can assume that:

(i) execution of the jobs starts at time 0,
(ii) at any time one machine performs at most one job,
(iii) performance of the job cannot be interrupted.

The following are associated with each job $i \in \mathcal{J}$: *execution time* p_i, *due date* d_i, and *weight* (value) w_i. For a fixed order of execution of jobs on a machine (satisfying the constraints (i)-(iii)), let C_i be the ending time of execution of job $i \in \mathcal{J}$. Let U_i =1 when $C_i > d_i$ or 0 otherwise. If U_i =1, then the job is *delayed* (tardy), and when $1 - U_i$ =1 (i.e. U_i =0) it is *on-time*. Then $w_i(1 - U_i)$ is the profit for completing the i-th job on time. The problem of maximizing the total profit (i.e., the sum of the weights of the on-time jobs) considered in this paper is to determine such a sequence of jobs that maximizes the sum of $\sum_{i=1}^{n} w_i(1 - U_i)$. In the literature it is denoted by $1||\max \sum w_i(1 - U_i)$ and belongs to the class of *NP*-hard problems (Karp [5]). It is one of a number of single-machine scheduling problems.

Problems of jobs scheduling on a single machine are among the oldest considered in the literature (Rinnoy Kan et al. [8]). One of the most frequently studied, due to the simplicity of the formulation, is the problem of minimizing the sum of the weights of tardy jobs $1||\min \sum w_i U_i$. The algorithms for solving it are usually adaptations of algorithms for solving the much more common $1||\min \sum w_i T_i$ problem (e.g. quantum optimization approach Bożejko et al. [2]).

Published research mainly deals with variants of the TWOT problem. In the work of Gerstl and Mosheiov [3], a problem is considered in which the moment of completion of a job depends on its position in a permutation. Mor and Mosheiov [7], on the other hand, consider the problem of selecting a subset of jobs to be executed with the maximization of the sum of time jobs weights. In addition, it is assumed that the sum of the weights of rejected jobs cannot exceed a certain value. Problems with time windows are also considered. The work by Koulamas [6] presents a problem decomposition method that was used in the construction of a heuristic algorithm. An extensive review of the literature on scheduling problems with time windows is presented in Shabtay and Steiner [9]. An important generalization of the problem $\sum_{i=1}^{n} w_i(1-U_i)$ is considered in the paper by Atsmony and Mosheiov [1] regarding the scheduling problem on parallel identical machines. Despite its long history, the different aspects of scheduling jobs on one machine are still interesting. For example, Hejl et al. [4] investigates the impact of correlations between task parameters, e.g. between execution times and weights and the difficulty of solving them. New possibilities related to performing calculations on quantum machines are an inspiration for research on exact algorithms for solving large examples of NP-hard problems.

3 Solution Method

Let Φ be the set of permutations of elements (jobs) from the set $\mathcal{J}$. For a permutation $\pi \in \Phi$, let $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(i)}$, $i = 1, 2, \ldots, n$ be the sequence of jobs completion moments. Because $U_{\pi(i)}$ depends on $C_{\pi(i)}$, thus the solutions to the problem under consideration can be represented only by permutations (instead of all completions times) of the elements of the set $\mathcal{J}$. *Profit* (goal function) of a permutation $\pi \in \Phi$ is

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)}(1 - U_{\pi(i)}). \tag{1}$$

The considered problem of maximization of the sum of profits of on-time jobs is therefore reduced to determining the optimal permutation $\pi^* \in \Phi$ such that $F(\pi^*) = \min\{F(\beta) : \beta \in \Phi\}$. To determine permutation π^* we will use an algorithm based on the Branch and Bound method.

The process of browsing the elements of the solution set Φ will be represented by means of a *solution tree* $\mathcal{H}$. The root of the tree (the only vertex at the zero level) is any (starting) permutation Φ. From the root, one can generate n new vertices (permutations) at level 1. Each of these is created by fixing at the n-th position one job. Similarly, from any first-level permutation can be generated, by fixing the job of the $n-1$-th position, $n-1$ new permutations forming level 2 of the tree. The full tree thus has n levels.

If a permutation β has been generated from π by fixing a certain assignment, then the pair (π, β) forms an arc in the tree $\mathcal{H}$. We then say that β was generated directly from π. By $\mathcal{H}(\pi)$ let us denote a subtree in $\mathcal{H}$ whose root of which is

the vertex π. Any vertex π of the h-th level ($h = 0, 1, 2, \ldots, n$) in the tree $\mathcal{H}$ is characterized by the *sequences of free jobs* (respectively *free positions*)

$$\pi^{\mathcal{B}} = (\pi(1), \pi(2), \ldots, \pi(n-h)) \tag{2}$$

and *fixed jobs* (*fixed positions*)

$$\pi^{\mathcal{E}} = (\pi(n-h+1), \pi(n-h+2), \ldots, \pi(n)). \tag{3}$$

By $\mathcal{Y}(\pi) = \{\pi^{\mathcal{B}}(1), \pi^{\mathcal{B}}(2), \ldots, \pi^{\mathcal{B}}(n-h)\}$ we denote *the set of free jobs* in π.

Let h be some level of the $\mathcal{H}$ tree. For simplicity of notation, we assume $t = n-h$ (this is the last free position in π). Generating from π) a new permutation of β (a vertex at the (h+1)-th level of the tree) consists in determining, at position t in β, one free job from the set $\mathcal{Y}(\pi)$. It is inserted at position t by performing an appropriate move of the *swap*-type. If we generate the β by fixing the free job $\pi(k)$ at position t, then to the $\mathcal{H}$ tree we add an arc (π, β). In each successor β (permutation of $\mathcal{H}(\beta)$) the job $\pi(k)$ is fixed at position t. Immediately after the generation of a new vertex β in the tree in $\mathcal{H}$, the values of the lower and upper bounds of the objective function (1) for the vertices (solutions) from the subtree $\mathcal{H}(\beta)$ are determined. The set of jobs *candidates* to determine we denote as $\mathcal{K}(\beta)$. If we backtrack from vertex β to π, then from the set $\mathcal{K}(\pi)$ we remove the job that was determined by generating β. We perform the backtracking operation from vertex β if the set of candidates $\mathcal{K}(\beta)$ is empty or in the subtree $\mathcal{Y}(\beta)$ there is no better solution than the best one found so far. The solution to the problem TWOT thus boils down to determining the vertex in the tree of $\mathcal{H}$ (permutation) with the maximum value of the profit function.

The essential elements of the B&B algorithm for solving the TWOT problem are the procedures for determining the lower and upper bound values of the objective function. To determine these, we use the solver *LeapHybridCQMSampler* from D-Wave implementing hardware quantum annealing. This requires the formulation of the problem under consideration as a quadratic discrete optimization task with constraints (CQM, *Constrained Quadratic Model*).

3.1 Upper Bound on the D-Wave Computer

For simplicity of notation, let us assume that at the vertex $\mathcal{H}(\pi)$ the free jobs $\pi(1), \pi(2), \ldots, \pi(t)$ are numbered consecutively from 1 to t such that $d_i \leq d_{i+1}$, $i = 1, 2, \ldots, t-1$. Then the problem $n|1|\sum w_i(1-U_i)$ can be formulated equivalently as the problem of maximizing the weighted number of jobs executed on time from among t of free jobs from the set $\mathcal{Y}(\pi)$ at the vertex π of the tree $\mathcal{H}$:

$$\max \sum_{i=1}^{t} w_i x_i, \tag{4}$$

with constraints (subject to):

$$\begin{aligned} p_1x_1 &\le d_1,\\ p_1x_1 + p_2x_2 &\le d_2,\\ p_1x_1 + p_2x_2 + p_3x_3 &\le d_3,\\ &\vdots\\ p_1x_1 + p_2x_2 + \ldots + p_tx_t &\le d_t,\\ x_i \in \{0,1\},\ i = 1,2,\ldots,t. & \end{aligned} \tag{5}$$

Due to the requirements of a quantum computer (minimization of functions), the task (4), (5) will be considered as a minimization task with the criterion:

$$-\min\left(-\sum_{i=1}^{t} w_i x_i\right). \tag{6}$$

We use the Lagrange relaxation method to compute the value of the upper bound of the function $\sum w_i(1-U_i)$ at a node of the tree $\mathcal{H}$, expressed by the objective function (4), denoted by $UB_{on\ time}$. The Lagrange function with multipliers $u = (u_1, u_2, \ldots, u_t)$ for $x = (x_1, x_2, \ldots, x_t)$ takes the form:

$$\begin{aligned} L(x,u) = \sum_{i=1}^{t} w_i x_i + u_1(p_1x_1 - d_1) + u_2(p_1x_1 + p_2x_2 - d_2) + \\ \ldots + u_t(p_1x_1 + p_2x_2 + \ldots + p_tx_t - d_t) = \\ = x_1(w_1 + u_1p_1 + u_2p_1 + \ldots + u_tp_1) + x_2(w_2 + u_2p_2 + u_3p_2 + \ldots + u_tp_2) + \\ \ldots + x_t(w_t + u_tp_t) - \sum_{i=1}^{t} u_i d_i = \sum_{i=1}^{t} x_i \left(w_i + p_i \sum_{j=i}^{t} u_j\right) - \sum_{i=1}^{t} u_i d_i. \end{aligned}$$

Let $L_i(x_i,u) = x_i\left(w_i + p_i\sum_{j=i}^{t} u_j\right)$, then $L(x,u) = \sum_{i=1}^{t} L_i(x_i,u) - \sum_{i=1}^{t} u_i d_i$, whereby maximization of $L(x,u)$ against individual variables x_i, for fixed values of u_i, can be performed independently.

Let us note that for *any* $u_i \le 0$, $i = 1,2,\ldots,t$ and the optimum x^* being a solution to the objective function problem (4) there is an inequality

$$\begin{aligned} \sum_{i=1}^{t} w_i x_i^* = \max_x \sum_{i=1}^{t} w_i x_i \le \min_u \max_x L(x,u) = \\ \min_u \left(\sum_{i=1}^{t} \max_x L_i(x_i,u)\right) - \sum_{i=1}^{t} u_i d_i = UB_{on\ time}. \end{aligned} \tag{7}$$

In turn, to calculate the upper bound of the Lagrange function $UB_{on\ time}$ on the D-Wave PC, we determine the values of the vector u using quantum annealing by solving the following CQM task:

$$UB_{on\ time} = \min_u \left(\sum_{i=1}^{t} L_i(x_i,u)\right) - \sum_{i=1}^{t} u_i d_i, \tag{8}$$

with constraints

$$L_i(x_i, u) \geq L_i(0, u) \text{ and } L_i(x_i, u) \geq L_i(1, u),$$

for each $i = 1, 2, \ldots, t$ (the constraints are a total of $2t$).

Calculating the value of $UB_{on\ time}$ according to the inequality (8) requires determining the optimal values of the variables $x_i, i = 1, 2, \ldots, t$. For each $L_i(x_i, u)$ the optimization against x_i can be performed independently by checking two values of the variables x_i, i.e. 0 or 1. The optimization against u can be approximated. Ultimately

$$QuantumLagrangeUB(\pi) = F(\pi^{\mathcal{E}}) + UB_{on\ time}. \tag{9}$$

The $F(\pi^{\mathcal{E}}) = \sum_{j=t+1}^{n} w_{\pi(j)}(1 - U_{\pi(j)})$ is the sum of the costs of the fixed jobs.

3.2 Lower Bound on the D-Wave Computer

Let

$$QuantumAnnealingLB(\pi) = F(\pi^{\mathcal{E}}) + QA(\pi^{\mathcal{B}}), \tag{10}$$

where $QA(\pi^{\mathcal{B}})$ is some approximate solution of the NP-hard TWOT problem for the set of free jobs $\mathcal{Y}(\pi) = \{\pi^{\mathcal{E}}(1), \pi^{\mathcal{E}}(2), \ldots, \pi^{\mathcal{E}}(t)\}$ in vertex π of the tree $\mathcal{H}$. We determine these by applying the (hardware) metaheuristics of quantum annealing $QA(\pi^{\mathcal{B}})$, for the solution of the $\pi^{\mathcal{B}}$.

4 Exact Algorithm Controlled by Quantum Annealing

To solve the optimization problem under consideration, we propose an optimization algorithm based on the B&B method in which the key elements – i.e., lower and upper bounds – are calculated using quantum annealing on a D-Wave computer. The algorithm uses the strategy of selecting nodes for expansion *best-first*, with the value of the objective function for the best solution generated by the quantum algorithm determining the lower estimate as the evaluation function. This strategy is implemented using a heap (variable *Heap*) whose vertices are remembered fours: (*permutation, number of free jobs, value of objective function, upper bound*). The pseudo-code of the `B&B_CPU_QPU` algorithm is shown below.

5 Case Study

To see the differences between the performance of the proposed hybrid quantum `B&B_CPU_QPU` and classical B&B algorithm running on a silicon computer, let us consider a problem instance consisting of $n = 6$ jobs. The parameters are presented in Table 1. Figure 1 illustrates the run of the B&B algorithm,

in which the selection of nodes is done with respect to the smallest value of the upper estimate determined by the Lagrange method on a silicon computer.

Hybrid `B&B_CPU_QPU` **Algorithm:**

Input π^0 – starting solution;
Output π^* – optimal solution;

Heap: priority queue of partial solutions sorted in non-decreasing order by their upper bounds UB;
$\pi^* := \pi^0$; Put(*Heap*,π^*,n,$F(\pi^*)$, $F(\pi^*)$);
while *Heap*$\neq \emptyset$ **do**
1 Get(*Heap*,(π, t, $F(\pi)$, UB_π));
2 **if** ($\lfloor UB_\pi \rfloor \leq F(\pi^*)$) **then continue**;
3 Find a set of K_π – jobs that can be set at t-th position;
4 **for** $j \in K_\pi$ **do**
4.1 Insert$(\pi, i = ps(j), t)$
4.2 $UB^{QA} := QuantumLagrangeUB(\pi^{\mathcal{B}})$;
4.3 $\pi^{QA} := \arg(QuantumAnnealingLB(\pi^{\mathcal{B}}))$; $\pi' := \pi^{QA}\pi^{\epsilon}$;
4.4 **if** ($F(\pi') > F(\pi^*)$) **then** $\pi^* := \pi'$;
4.5 $UB := \lfloor UB^{QA} \rfloor + \sum_{i=t}^{n} w_{\pi(i)} \cdot (1 - U_{\pi(i)})$;
4.6 **if** ($UB \geq F(\pi^*) + 1$) **then** Put(*Heap*,π',$t-1$,UB, $F(\pi')$);
end while

Each node in the tree shows the number of the iteration in which the node is considered (top left), the value of the upper estimate (top right), the permutation (in the middle), and the value of the objective function (bottom right). Determined items are bolded in the description of permutations. The descendant nodes are ordered according to the non-increasing value of UB.

Table 1. An instance `wt6_016` of $n = 6$ size.

i	1	2	3	4	5	6
p_i	13	81	19	45	53	23
w_i	1	7	6	1	2	4
d_i	23	37	41	42	51	62

During its run, the B&B algorithm generated 20 nodes and considered 11 of them. The optimal solution $(1, 3, 6, 2, 4, 5)$ with an objective function value of 11 was generated when considering the descendant nodes of the 10th considered node. The nodes not considered are those with an upper bound value greater than the value of the objective function of the best solution found or having no descendant nodes, e.g. the node (2,3,4,5,6,1). The value of the objective function of the best-found solution is 7 when the descendants of the first node are generated and changes to a value of 11 when the descendants of the 10th node are generated. Let us note that if the optimal value was known when considering node 7, the descendants of this node would not be generated.

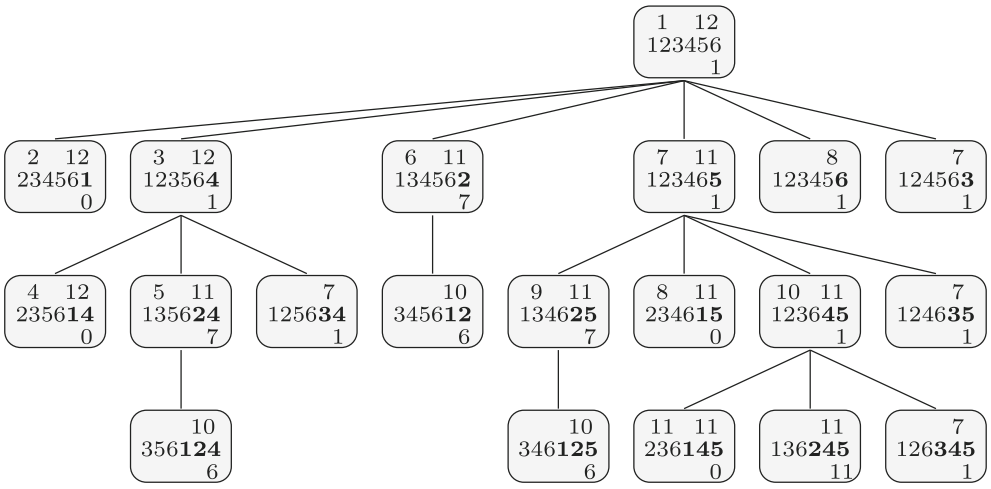

Fig. 1. .

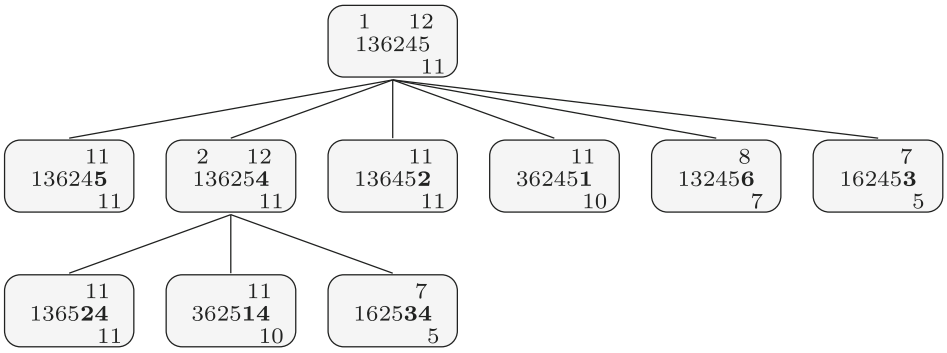

Fig. 2. .

Figure 2 shows the run of the B&B_CPU_QPU algorithm. During the run, the algorithm generated 10 nodes, i.e. half as many as the B&B algorithm, and performed only 2 iterations, i.e. almost 6 times fewer than the B&B algorithm. With the optimal solution already found in the first iteration, the remaining iterations were performed to prove the optimality of this solution.

6 Computational Experiments

The aim of the computer experiment is to compare the quality of solutions generated by the hybrid quantum algorithm with those generated by the GUROBI software package on test instances of large size. The computer experiment was carried out on instances consisting of between 200 and 1,000 jobs available at [10]. The execution of the quantum algorithm was limited to performing only one iteration, i.e. determining the lower and upper bounds for the full set of jobs.

The Table 2 shows the experimental results. In particular, in the case of B&B_CPU_QPU the value of the lower (value of the objective function of the best

Table 2. Results of experimental research.

instance	QAdB&B			Gurobi	
	UB	$F(\pi)$	QPU [ms]	$F(\pi)$	CPU [ms]
wt200_011	919.55	888	16.003	890	25.15
wt200_012	905.81	**873**	16.027	873	24.19
wt200_013	890.16	855	16.035	856	25.97
wt200_014	899.76	**865**	16.048	865	25.30
wt200_015	896.71	862	16.049	865	26.49
wt200_016	659.77	**647**	16.049	647	27.71
wt200_017	689.68	677	16.050	678	23.48
wt200_018	674.75	666	16.051	667	24.28
wt200_019	710.37	**699**	16.051	699	32.94
wt200_020	636.47	623	16.052	625	44.10
wt500_011	2312.90	2033	16.009	2066	181.35
wt500_012	2644.19	2174	16.036	2211	191.39
wt500_013	2544.50	2070	16.022	2097	208.56
wt500_014	4588.66	2157	16.023	2186	178.87
wt500_015	2496.14	2108	16.025	2143	177.85
wt500_016	1851.54	1718	16.049	1744	240.51
wt500_017	2089.65	1688	16.045	1715	241.02
wt500_018	2270.54	1679	15.998	1720	209.72
wt500_019	1862.21	1651	16.001	1682	234.91
wt500_020	2873.79	1715	16.042	1727	205.35
wt1000_011	5192.06	4146	16.003	4176	1080.34
wt1000_012	5462.87	4350	16.030	4391	1166.82
wt1000_013	5555.24	4339	16.033	4392	1044.92
wt1000_014	5523.50	4403	15.916	4455	1003.97
wt1000_015	5547.13	4385	16.043	4456	1251.22
wt1000_016	4371.40	3376	16.003	3432	1018.88
wt1000_017	4143.56	3329	15.923	3371	1054.50
wt1000_018	3999.50	3369	15.980	3416	1231.53
wt1000_019	4037.61	3346	16.019	3416	1000.03
wt1000_020	4019.61	3315	16.031	3362	1018.01

solution found) and upper bound, and the total computation time on the quantum annealer expressed in [ms] are given, while in the case of the GUROBI package the value of the objective function for the best solution generated by the package and the computation time (all solutions generated by this package are optimal).

The analysis of experimental results shows that the B&B_CPU_QPU algorithm found the optimal solution (marked in bold) in the first iteration in 4 cases. The average relative error of the algorithm running this way is $\delta_{\pi^{QA}} = \frac{F(\pi^{QA})-F(\pi^*)}{F(\ pi^*)} \cdot 100\%$, where π^{QA} is the solution generated by the B&B_CPU_QPU algorithm, and π^*) is the optimal solution, slightly increases with the increase in the number of jobs and amounts to $ave(\delta_{\pi^{QA}}) = 0.13\%$ for $n = 200$, $ave(\delta_{\pi^{QA}}) = 1.55\%$ for n =500, and $ave(\delta_{\pi^{QA}}) = 1.33\%$ for $n = 1000$.

The calculation time of the QAdB&B quantum algorithm does not change with the increase in the number of jobs and is just over 15 ms. It is comparable to the calculation time of the silicon machine for n =200 and on average 12 times shorter for instances consisting of $n = 500$ jobs and as much as 67 times shorter for the largest instances, i.e. consisting of n =1000 jobs.

7 Summary

Performing computations on a quantum computer creates new opportunities for solving NP-hard discrete optimization problems. Although the quantum annealing algorithm does not guarantee the optimality of the determined solution, it can be successfully used in the construction of an exact algorithm. This is because it significantly improves its efficiency by (non-deterministically) controlling the process of searching the solution space. This paper presents the concept of a quantum hybrid branch and bound algorithm for solving a single-machine scheduling problem with minimization of a weighted number of delayed jobs. The computational results of the quantum annealing algorithm are used to determine the computational trajectory. The presented algorithm can be adapted to solve other NP-hard problems. Experiments were performed on a D-Wave quantum machine.

References

1. Atsmony, M., Mosheiov, G.: Scheduling to maximize the weighted number of on-time jobs on parallel machines with bounded job-rejection. J. Sched. (2022). https://doi.org/10.1007/s10951-022-00745-7
2. Bożejko, W., Pempera, J., Uchroński, M., Wodecki, M.: Distributed Quantum Annealing on D-Wave for the Single Machine Total Weighted Tardiness Scheduling Problem. In: Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science - ICCS 2022, vol. 13353. LNCS, pp. 171–178. Springer (2022). https://doi.org/10.1007/978-3-031-08760-8_15
3. Gerstl, E., Mosheiov, G.: Single machine scheduling to maximize the number of on-time jobs with generalized due-dates. J. Sched. **23**(3), 289–299 (2020)
4. Hejl L., Šůcha P., Novák A., Hanzálek Z.: Minimizing the weighted number of tardy jobs on a single machine. Euro. J. Operat. Res. **298**(2), 413–424 (2022)
5. Karp R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.)Complexity of Computer Computation, New York, pp. 85–104 (1972)

6. Koulamas, C.: Maximizing the weighted number of on-time jobs in single machine scheduling with time windows. Math. Comput. Model. **25**(10), 57–62 (1997)
7. Mor, B., Mosheiov, G.: Single machine scheduling to maximize the weighted number of on-time jobs with job-rejection. Oper. Res. Int. Journal **22**(3), 2707–2719 (2022)
8. Rinnoy Kan A.H.G., Lageweg B.J., Lenstra J.K.: Minimizing total costs in one-machine scheduling. Oper. Res. **25**, 908–927 (1975)
9. Shabtay, D., Steiner, G.: Scheduling to maximize the number of just-in-time jobs: A survey. Springer Optimiz. Appli. **60**, 3–20 (2012). https://doi.org/10.1007/978-1-4614-1123-9-1
10. Uchroński M.: Benchmark instances files. https://data.e-science.pl/51561/1WiTiTestData.zip. (Accessed 19 Feb 2023)