Future Generation Computer Systems 155 (2024) 245–255

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Quantum annealing-driven branch and bound for the single machine total weighted number of tardy jobs scheduling problem

Wojciech Bożejko [a,*], Jarosław Pempera [a], Mariusz Uchroński [b], Mieczysław Wodecki [c]

[a] Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, Janiszewskiego 11/17, 50-372 Wrocław, Poland
[b] Wroclaw Centre for Networking and Supercomputing, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
[c] Department of Telecommunications and Teleinformatics, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

ARTICLE INFO

Keywords:
Optimization
Quantum computing
Scheduling
Optimal algorithm
Quantum annealing
Lagrange relaxation

ABSTRACT

In the paper we present a new approach to solving *NP*-hard problems of discrete optimization adapted to the architecture of quantum processors (QPU, Quantum Processor Unit) implementing hardware quantum annealing. This approach is based on the use of the quantum annealing metaheuristic in the exact branch and bound algorithm to compute the lower and upper bounds of the objective function. To determine the lower bound, a new method of defining the Lagrange function for the dual problem (the generalized discrete knapsack problem) was used, the value of which is calculated on the QPU of a quantum machine. In turn, to determine the upper bound, we formulate an appropriate task in the form of binary quadratic programming with constraints.

Despite the fact that the results generated by the quantum machine are probabilistic, the hybrid method of algorithm construction proposed in the paper, using alternately a CPU and QPU, guarantees the optimal solution. As a case study we consider the *NP*-hard single machine scheduling problem with minimizing the weighted number of tardy jobs. The performed computational experiments showed that optimal solutions were already obtained in the root of the solution tree, and the values of the lower and upper bounds differ by only a few percent.

1. Introduction

The concept of quantum computing originates from the 1980s. Currently, quantum computers are available that represent one of two approaches to quantum computing. The first, offered by companies such as Google, Honeywell, IBM and Intel, are quantum computers with quantum gate models (e.g., Hadamard and Toffoli). Unlike many classical logic gates, quantum logic gates are reversible. Programming in this model of quantum computing is still a great challenge due to the small scale of solvable problems and the lack of a high-level approach, adequate to high-level languages in the programming of classic silicon computers. The second approach, called *quantum annealing*, by using effects known as quantum fluctuations and quantum tunneling, determines the best possible solution to the optimization problem, see e.g., (Bozejko et al. [1]). D-Wave Systems provides quantum computers, proposing an approach to computation limited only to the use of quantum annealing; however, it is perfectly suited to the needs of the discipline of operations research. In this case, instead of expressing the algorithm solving a given problem in the form of quantum gates, the user presents it as a binary quadratic programming problem. However, Aharonov et al. [2] show that quantum annealing (as an adiabatic quantum computation) is equivalent to the standard quantum-gate model of a quantum computer.

There are several attempts to apply the methodology of quantum computations to operational research problems. Regarding branch and bound quantum modifications, Montanaro [3] describes a quantum algorithm that can accelerate classical branch-and-bound algorithms near-quadratically in a general setting. The author considers a spin model addressed using branch-and-bound, which is the Sherrington–Kirkpatrick model, which is the family of classical Hamiltonians $H(x) = \sum_{1\le i<j\le n} a_{ij}x_ix_j$ where $x \in \{\pm 1\}$ and a_{ij} are distributed according to the normal distribution. Finding the lowest-energy state for such a Hamiltonian can be achieved in time $O(2^{0.5n}poly(n))$ using Grover's algorithm, which is less efficient than the approach proposed in [3] with the time $O(2^{0.226n})$.

The paper of Markevich and Trushechkin [4] shows the theoretical branch and bound for a quantum-gates based quantum computer (without experiments).

* Corresponding author.
E-mail addresses: wojciech.bozejko@pwr.edu.pl (W. Bożejko), jaroslaw.pempera@pwr.edu.pl (J. Pempera), mariusz.uchronski@pwr.edu.pl (M. Uchroński), mieczyslaw.wodecki@pwr.edu.pl (M. Wodecki).

https://doi.org/10.1016/j.future.2024.02.016
Received 13 August 2023; Received in revised form 13 February 2024; Accepted 16 February 2024
Available online 19 February 2024

In the case of applying quantum annealing to scheduling problems, Ikeda et al. [5] proposes the application of Quantum Annealing to the Nurse Scheduling Problem using a D-Wave 2000Q quantum annealer (2048 qubits with a mean 6 connections per qubit) with the approximate results of computational experiments. Carugno et al. [6] propose quantum annealing with QUBO formulation of job shop scheduling on a D-Wave 2000Q. The authors state that quantum solvers generally provide a worse solution quality compared with the best classical ones. Bożejko et al. [7] propose to use D-Wave quantum environment for exact solving single machine scheduling problem with a total weighted tardiness cost function. Stogiannos et al. [8] compare several attempts to solve the traveling salesman problem (TSP) with using a D-Wave quantum computer. In turn, in the conference paper Bożejko et al. [9], the authors consider a similar problem of jobs scheduling on a single machine with the criterion of total weighted number of on-time jobs maximization. The authors proposed a truncated Branch and Bound scheme (with a limited number of tree levels). It allowed for determining the potential of the components of the method.

One of the simplest scheduling problems to formulate (though *NP*-hard) on one machine is the *Weighted Number of Tardy jobs minimization problem* (WNT for short) considered in this paper. It is defined as follows:

Problem 1 (*Weighted Number of Tardy Jobs Minimization Problem, WNT*)**.** There is a set of n jobs given $\mathcal{J} = \{1, 2, \dots, n\}$ which must be carried out on one machine without interruption. A machine can only perform one job at any given time. With every job $i \in \mathcal{J}$ there are the following properties linked:

p_i	–	*processing time*,
d_i	–	*due date*,
w_i	–	*cost of the penalty function*.

For a fixed order of execution of jobs on the machine, let C_i be the completion time for execution of job $i \in \mathcal{J}$. Then, *tardiness* $U_i = 1$, when $C_i > d_i$ or 0 otherwise. The value of $w_i U_i$ is *a weight of tardy job*. In the problem under consideration it is necessary to determine the order of execution of jobs on the machine that minimizes the weighted number of tardy jobs, i.e., $\sum_{i=1}^{n} w_i U_i$.

In the literature, such problems are denoted by $1||\sum w_i U_i$ and despite their simplicity of the formulation, they belong to the class of *NP*-hard problems (Karp [10], Lenstra and Rinnoy Kan [11]). It is also one of many single-machine scheduling problems which have been studied for many years. Its variants have constantly been considered, mainly due to the polynomial computational complexity. Later in the work, without loss of generality, we can assume that $d_i \geq 0$, $i = 1, 2, \dots, n$, since jobs with negative values of due dates can be moved to the end of the schedule (because they are always tardy and in the optimization process have the same cost regardless of the solution).

For the problem $1|p_i = 1|\sum w_i U_i$ (all jobs' execution times are the same) Monma in his work (Monma [12]) introduced algorithm complexity $O(n)$. Likewise, for the problem $1|w_i = c|\sum U_i$ with the same coefficients of the cost function, there is a Moore (Moore [13], Lawler [14]) algorithm with a complexity of $O(n \ln n)$.

If a partial order relation is specified on the job set then WNT problems are strongly *NP*-hard even when all the job execution times are unitary (Garey and Johnson [15]).

Only a few optimal algorithms have been published in the literature solving the WNT problem. They are based on the dynamic programming method (Lawler and Moore, 1977) – complexity $O(n \min\{\sum p_i, \max\{d_i\}\})$ and (Sahni [16]) – complexity $O(n \min\{\sum p_i, \sum w_i, \max\{d_i\}\})$ and on the Branch and Bound method (Villareal and Bulfin [17], Potts and Van Wassenhove [18], M'Hallah and Bulfin [19]). An extended version of the problem, with due dates d_i and deadlines $\widetilde{d}_i$, is in turn considered in (Hejl et al. [20]). MIP-based lower and upper bounds are considered in the work (Briand and Ourari [21]). The literature also deals with single-machine scheduling problems with uncertain execution times or desired completion dates (Rajba and Wodecki [22]).

The paper presents a hybrid optimal algorithm for solving the WNT problem, the construction of which is based on the (*Branch and Bound*, B&B) method. We propose a new approach based on determining the upper and lower bounds of the objective function on the D-Wave quantum computer. To determine the upper bound, we formulate an appropriate task in the form of binary quadratic programming with constraints. Quadratic programming is a natural way of formulating tasks for quantum annealer, as long as the objective function has at most quadratic form and the constraints are linear. Then it is easy to transform such a model, called Constraint Quadratic Model, CQM, to BQM (Binary Quadratic Model) using Lagrange's relaxation to build constraints into the objective function (this is done, for example, by the `dimod.cqm_to_bqm` procedure of D-Wave Ocean SDK, available under an open source license). BQM, in turn, can be run natively as quantum annealing on a D-Wave quantum computer — it can technically represent both the QUBO and Ising models.

In turn, when determining the lower bound, we use the directed formulated Lagrange relaxation method. The approximation of the maximum value of the Lagrange function is also determined on a quantum computer. As a criterion for selecting the next node in the solution tree, we use the upper bound of the objective function. Moreover, the results of the conducted computational experiments are presented later.

In the formulation and throughout the paper, we use the following notation:

n	–	number of all jobs;
$\mathcal{J}$	–	set of all jobs, $\lvert\mathcal{J}\rvert = n$;
p_i	–	time of job i execution;
w_i	–	job i tardiness cost coefficient;
d_i	–	due date of job i completion;
π	–	permutation of jobs;
Φ	–	set of all permutations of elements from $\mathcal{J}$;
S_i	–	moment of starting the job $i \in \mathcal{J}$;
C_i	–	moment of completing the job $i \in \mathcal{J}$;
U_i	–	binary tardiness of job i (0 or 1);
$F(\pi)$	–	sum of the costs of tardinesses (the criterion);
$\mathcal{H}$	–	solution tree;
h	–	level of the tree;
t	–	number of fixed jobs;
$\mathcal{Y}(\pi)$	–	set of free jobs;
UB_π	–	the upper bound of the objective function at node π;
LB_π	–	the lower bound of the objective function at node π;
L	–	dual Lagrange function;
$QALagrange$	–	value of the lower bound of the objective function of the WNT;
$QuantumAnnealing$	–	value of the upper bound of objective function of WNT;

2. Quantum annealing

Quantum annealing is a promising computational method that uses quantum phenomena to solve particularly difficult computational tasks. The rapid development of this type of computation has been caused by the presence of the D-Wave Systems quantum annealing device available on the market, as well as the works carried out by the Japanese NEC corporation. A quantum machine for quantum annealing handles some limited class of optimization problems. In order to solve the problem on the D-Wave quantum annealer, the problem under consideration should be formulated as the problem of Quadratic Unconstrained Binary Optimization (QUBO). Currently, there has been an intense search for the possibility of applying this technology in practice, by learning about its possibilities and limitations. The quantum annealing machine, through the continuous evolution of the quantum system,

searches for the minimum energy of the Ising Hamiltonian (Ajagekar et al. [23], Denkena et al. [24]).

The purchase of a D-Wave machine costs several million US dollars, but currently D-Wave provides users with computing time on multiple D-Wave computers distributed around the world within the *Leap* environment. The use of the new technology for optimization tasks requires an answer to the question whether the efficiency of computing on a quantum computer is significantly better than classical computers based on silicon systems.

In practice, the tasks formulated for a quantum machine implementing quantum annealing take the form of an Ising or QUBO model, where the translation of problems between these models is trivial. The Ising model is used in statistical mechanics and the criterion function has the following form:

$$E_{Ising}(s) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j, \tag{1}$$

where s_i, $i = 1, 2, \ldots, N$ express spins with the values $+1$ and -1, while the linear coefficients corresponding to qubit deviations are h_i, the quadratic coefficients corresponding to the coupling forces are $J_{i,j}$. On the other hand, in the QUBO model, the function subject to minimization takes the form

$$f(x) = \sum_{i=1}^{N} Q_{i,i} x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} Q_{i,j} x_i x_j, \tag{2}$$

where Q is an upper-triangular matrix of size $N \times N$ of real weights, whereas x is a vector of binary variables.

QUBO is an unconstrained model, which means that in practice all constraints of the problem must be included in the objective function. Some of the *Leap Hybrid* solvers can handle constraints natively – for them the translation of the constraints problem to the unconstrained one is done inside the solver. For such a model – specifically for *LeapHybridCQMSampler* (Constrained Quadratic Model, CQM) – the presented work below is dedicated. The problem formulation for the CQM model takes the form of minimization:

$$\sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{i,j} x_i x_j + c, \tag{3}$$

subject to constraints:

$$\sum_{i=1}^{N} a_i^{(m)} x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{i,j}^{(m)} x_i x_j + c^{(m)} \propto 0, \; m = 1, 2, \ldots, M \tag{4}$$

where x_i, $i = 1, 2, \ldots, N$ can be binary, integer or real variables, $a_i, b_{i,j}, c$, $i, j = 1, 2, \ldots, N$, are real values, whereas relation $\propto \in \{\geq, \leq, =\}$ and the number of constraints M is always an integer.

The tasks formulated to be solved by the D-Wave quantum computer must be in the form of the Constrained Quadratic Model (CQM) [25] and minimization of the objective function. In the case of the problem under consideration, it is a constrained integer linear programming task – then it is possible to use the *LeapHybridCQMSampler* solver for its solution that implements quantum annealing in hardware. The CQM solver runs with an Advantage [26–28] backend for the quantum portion of the quantum–classical hybrid solver. Advantage QPUs are based on the Pegasus graph topology [29] with size P16 containing at least 5,000 qubits with 15 couplers per qubit, totaling at least 35,000 couplers.

3. Solution method

Solutions to the WNT problem can be represented by permutations of the elements of the $\mathcal{J}$ job set. Let Φ be the set of all such permutations. Cost (value of the objective function) of the permutation $\pi \in \Phi$, $F(\pi) = \sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)}$, where the moment of completion of job $\pi(i) \in \mathcal{J}$, $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(i)}$, for $i = 1, 2, \ldots, n$. The considered problem of minimizing the sum of costs of tardy jobs comes down to finding the optimal permutation $\pi^* \in \Phi$ that is one for which $F(\pi^*) = \min\{F(\beta) : \beta \in \Phi\}$.

Branching strategy. The process of browsing the elements of a solution set Φ will be represented by directed *solution tree* $\mathcal{H}$. The root of the tree (the only node at level zero) is any $\pi_0 \in \Phi$ (starting) permutation in which all jobs are free. From the root one can generate n new nodes (permutations) on the first level. Each of them is created by fixing one job to the nth position. Likewise, with any first-level permutations by setting a free job in free $(n-1)$th position, there can be generated $n-1$ new permutations making up level 2 of the tree. Hence, a full tree has n levels. At the last nth level of the tree, there are $n!$ permutations (nodes) and in each of them all jobs are fixed.

If β permutation was generated from π by setting a free job on a free position, then the pair (π, β) creates an arc in the $\mathcal{H}$ tree. By $\mathcal{H}(\pi)$ we denote the subtree in $\mathcal{H}$, in which the root is the node π.

Any node π from hth level $(h = 0, 1, 2, \ldots, n)$ in the solution tree $\mathcal{H}$ is characterized by a set and a sequence of free jobs. Let us denote the *sequences of free jobs* as

$$\pi^{\mathcal{B}} = (\pi(1), \pi(2), \ldots, \pi(t)), \tag{5}$$

and *fixed jobs*

$$\pi^{\mathcal{E}} = (\pi(t+1), \pi(t+2), \ldots, \pi(n)), \tag{6}$$

where $t = n - h$ is the last free position in permutation π. Therefore, permutation $\pi = (\pi^{\mathcal{B}}, \pi^{\mathcal{E}})$ is a concatenation of subpermutations $\pi^{\mathcal{B}}$ and $\pi^{\mathcal{E}}$. A set

$$\mathcal{Y}(\pi) = \{\pi(1), \pi(2), \ldots, \pi(t)\} \tag{7}$$

contains free jobs in a permutation π.

Let h be a certain level of the $\mathcal{H}$ tree. Generating from π a new permutation (the node at the $(h+1)$th level of the tree) relies in determining, on position t, in β one free job from $\mathcal{Y}(\pi)$ set. The jobs are inserted into the t position by making the appropriate *insert*-type move (a detailed description of this move is provided in (Wodecki [30]). If β is generated by setting the free job $\pi(k)$ $(k \in \{1, 2, \ldots, t\}$ is a position number) on position t, then an arc(π, β) is added to the tree $\mathcal{H}$. In every successor of β (of permutation from $\mathcal{H}(\beta)$) the job $\pi(k)$ is fixed to position t. The solution to the WNT problem comes down to determining the node in the $\mathcal{H}$ tree (permutation) of the minimum value of the cost function. In each node π of the tree $\mathcal{H}$ free jobs from the set $\mathcal{Y}(\pi)$ are *candidates* to be fixed to the last free position. Therefore, $|\mathcal{Y}(\pi)|$ nodes can be generated directly from π.

Branching rule. The order in which candidates are selected to be determined has a significant impact on the eventual rejection of the subtree, in the algorithm based on the B&B scheme. As a rule, such candidates are selected in such a way that the algorithm determines the best solution as soon as possible. In practice, the most common choice is the node for which the lower estimate of the objective function value is the smallest.

The basic elements of the presented method of solving the WNT problem are algorithms for determining the lower and upper bounds of the value of the objective function. They have a direct impact on the number of generated nodes of the solution tree, and thus on the computation time. The lower bound of the objective function value at the node π of the solution tree $\mathcal{H}$ is

$$LB_{\pi} = \sum_{i=t+1}^{n} (w_{\pi(i)} U_{\pi(i)}) + LB_{\mathcal{Y}(\pi)} \tag{8}$$

where $LB_{\mathcal{Y}(\pi)}$ is the lower bound of the objective function for the free jobs from the set $\mathcal{Y}(\pi)$.

Similarly, the upper bound

$$UB_{\pi} = \sum_{i=t+1}^{n} (w_{\pi(i)} U_{\pi(i)}) + UB_{\mathcal{Y}(\pi)} \tag{9}$$

where $UB_{\mathcal{Y}(\pi)}$ is the upper bound of the objective function for the free jobs from the set $\mathcal{Y}(\pi)$. The use of an upper bound allows the B&B to

prune active nodes that have larger lower bound values, reducing the size of the searched region and shortening the computing time.

In order to determine the value of $LB_{\mathcal{Y}(\pi)}$ and $UB_{\mathcal{Y}(\pi)}$ we use the sampler of the quantum computer by the D-Wave company performing hardware quantum annealing. However, its application requires formulation of the problem of minimizing the sum of penalties for tardy jobs from the set $\mathcal{Y}(\pi)$ in the form of a square discrete optimization problem with constraints (CQM, *Constrained Quadratic Model*). We present this transformation below.

3.1. Task formulation for the D-wave machine

The tasks formulated to be solved by the D-Wave quantum computer must be in the form of the Constrained Quadratic Model (CQM) and minimization of the objective function. In the case of the problem under consideration, it is a constrained integer linear programming task — then it is possible to use the *LeapHybridCQMSampler* solver for its solution that implements quantum annealing in hardware.

To simplify the notation, let us assume that in the node $\mathcal{H}(\pi)$ free jobs from the set $\mathcal{Y}(\pi) = \{\pi(1), \pi(2), \dots, \pi(t)\}$ are numbered with consecutive numbers from 1 to t ($t \leq n$) such that $d_i \leq d_{i+1}$, $i = 1, 2, \dots, t-1$ (i.e., according to the *Earliest Due Date*, EDD rule). Then, the minimization of the objective function of the WNT problem considered in the work can be formulated equivalently:

$$\begin{aligned}
&\min \sum_{i=1}^{t} w_i U_i = \min \left(\sum_{i=1}^{t} w_i - \sum_{i=1}^{t} w_i + \sum_{i=1}^{t} w_i U_i \right) = \\
&= -\max \left[-\left(\sum_{i=1}^{t} w_i - \sum_{i=1}^{t} w_i + \sum_{i=1}^{t} w_i U_i \right) \right] = \\
&= \sum_{i=1}^{t} w_i - \max \left(\sum_{i=1}^{t} w_i - \sum_{i=1}^{t} w_i U_i \right) = \\
&= \sum_{i=1}^{t} w_i - \max \sum_{i=1}^{t} w_i (1 - U_i).
\end{aligned}$$

The first element $\sum_{i=1}^{t} w_i$ of the above expression is a constant, and the determination of the second comes down to solving the problem $1||\max \sum w_i(1-U_i)$ consisting in maximizing the weighted number of tasks from the set $\mathcal{Y}(\pi)$ executed on time (*Weighted Number of On-time jobs* problem, abbreviated toWNO). As in Lawler and Moore's work (Lawler and Moore [31]) – (see also M'Hallah and Bulfin [19]) – it can be formulated as follows:

$$\max \sum_{i=1}^{t} w_i x_i \tag{10}$$

with constraints:

$$\begin{aligned}
& p_1 x_1 && && && && \leq d_1, \\
& p_1 x_1 && + && p_2 x_2 && && \leq d_2, \\
& p_1 x_1 && + && p_2 x_2 && + && p_3 x_3 && \leq d_3, \\
& && && \vdots && && && \\
& p_1 x_1 && + && p_2 x_2 && + && \dots && + && p_t x_t && \leq d_t,
\end{aligned} \tag{11}$$

$$x_i \in \{0, 1\}, \ i = 1, 2, \dots, t. \tag{12}$$

The variable $x = (x_1, x_2, \dots, x_t)$ has the following interpretation: if $x_i = 1$, then job i is executed on time, i.e., it finishes before its due date d_i. Otherwise, $x_i = 0$ the job is late (tardy).

So, the value of the objective function (10) is the exact weighted number of jobs performed on-time. Regarding constraints, setting according to the EDD rule guarantees that if for job i, $x_i = 1$ (it is on time), i.e., it satisfies the ith constraint from (11), it also satisfies all subsequent constraints.

This is an NP-hard problem (see, e.g., M'Hallah and Bulfin [19]). On a quantum machine, we calculate its upper and lower bounds (see Fig. 1; the figure is illustrative – the problem is not continuous – the solutions mean individual solutions to the problem, here: permutations). They will be used to designate the constraints of $LB_{\mathcal{Y}(\pi)}$ and $UB_{\mathcal{Y}(\pi)}$ respectively, in expressions (8) and (9).

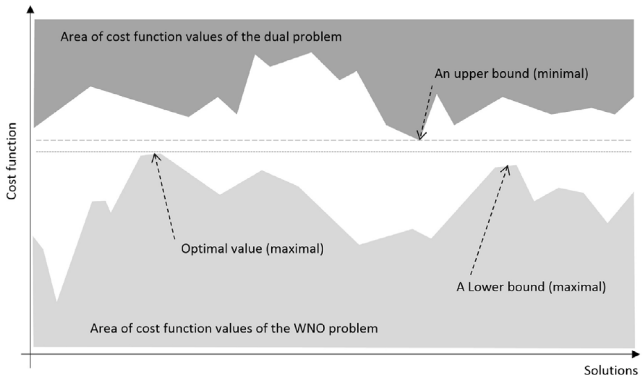

Fig. 1. Idea of the quantum lower and upper bound calculation.

3.2. Calculating the upper bound of the WNO problem on a D-wave machine

For a dual WNO problem, defined in (10)–(12), we use the Lagrange relaxation method. The Lagrange function with a vector of real multipliers $u = (u_1, u_2, \dots, u_t)$, $u_i \leq 0$, $i \in \mathcal{J}$, for a vector of binary variables $x = (x_1, x_2, \dots, x_t)$, takes the form:

$$\begin{aligned}
L(x,u) &= \sum_{i=1}^{t} w_i x_i + u_1 (\underbrace{p_1 x_1 - d_1}_{\leq 0 \text{ for feasible}}) + u_2 \underbrace{(p_1 x_1 + p_2 x_2 - d_2)}_{\leq 0 \text{ for feasible}} + \\
&\dots + u_t \underbrace{(p_1 x_1 + p_2 x_2 + \dots + p_t x_t - d_t)}_{\leq 0 \text{ for feasible solutions}} = \\
&= x_1(w_1 + u_1 p_1 + u_2 p_1 + \dots + u_t p_1) + x_2(w_2 + u_2 p_2 + u_3 p_2 + \dots + u_t p_2) + \\
&\dots + x_t(w_t + u_t p_t) - \sum_{i=1}^{t} u_i d_i = \sum_{i=1}^{t} x_i \left(w_i + p_i \sum_{j=i}^{t} u_j \right) - \sum_{i=1}^{t} u_i d_i.
\end{aligned} \tag{13}$$

Let

$$L_i(x_i, u) = x_i \left(w_i + p_i \sum_{j=i}^{t} u_j \right), \tag{14}$$

then

$$L(x,u) = \sum_{i=1}^{t} L_i(x_i, u) - \underbrace{\sum_{i=1}^{t} u_i d_i}_{\text{independent of } x}$$

whereby, maximization of $L(x,u)$ with respect to individual variables x_i, for fixed values of u_i, can be performed independently.

Let us note that for *any* $u_i \leq 0$, $i = 1, 2, \dots, t$ and the optimal solution of x^* to the WNO problem with the objective function (10), the following lemma applies:

Remark 1. For any value $u_i \leq 0$, $i = 1, 2, \dots, t$ of the Lagrange multipliers, the value $L(x,u)$ of the Lagrangian function is an upper bound on the optimal objective function value $\sum_{i=1}^{t} w_i x_i^*$ of the WNO problem.

Proof. Let

$$A = \{x = (x_1, x_2, \dots, x_t) : x_i \in \{0, 1\}, \ i = 1, 2, \dots, t\},$$

$$B = \{x = (x_1, x_2, \dots, x_t) : x \in A$$

$$\text{and } \sum_{j=1}^{i} x_j p_j \leq d_i, \ i = 1, 2, \dots, t\},$$

$$U = \{u = (u_1, u_2, \dots, u_t) : u_i \in \mathbb{R}, u_i \leq 0, \ i = 1, 2, \dots, t\}.$$

Then, for every sequence $u \in U$ there is

$$\sum_{i=1}^{t} w_i x_i^* \le \max_{x \in B} L(x,u) \le \max_{x \in A} L(x,u) = UB(u),$$

which results directly from the form (13) (to $\sum_{i=1}^{t} w_i x_i$ we add non-negative values, i.e., the products of non-positive u_i and non-positive $\sum_{j=1}^{i} x_j p_j - d_i$) and the fact that the set $B \subseteq A$ which makes the maximum after $x \in A$ the same or greater than after $x \in B$. Therefore

$$\sum_{i=1}^{t} w_i x_i^* \le \min_{u \in U} UB(u). \tag{15}$$

Using the Lemma 1 (inequality (15)) we obtain

$$\sum_{i=1}^{t} w_i x_i^* \le \min_{u \in U} \max_{x \in A} L(x,u) =$$

$$= \min_{u \in U} \left\{ \sum_{i=1}^{t} \max_{x_i \in \{0,1\}} L_i(x_i,u) - \sum_{i=1}^{t} u_i d_i \right\}. \tag{16}$$

It follows from the above inequality that

$$UB_{on-time} = \min_{u \in U} \left\{ \sum_{i=1}^{t} \max_{x_i \in \{0,1\}} L_i(x_i,u) - \sum_{i=1}^{t} u_i d_i \right\} \tag{17}$$

is the upper bound of the optimal value of the maximized objective function $\sum_{i=1}^{t} w_i x_i^*$ for the WNO problem.

In turn, to compute the upper bound (17) of the Lagrange function $UB_{on-time}$ on the D-Wave computer (the tree is built and searched on a regular, silicon computer but the bounds are calculated using a quantum computer), we calculate the values of the vector u using quantum annealing by solving the following CQM problem:

$$\min_{u \in U, x \in A} \left\{ \sum_{i=1}^{t} L_i(x_i,u) - \sum_{i=1}^{t} u_i d_i \right\}, \tag{18}$$

with constraints

$$L_i(x_i,u) \ge L_i(0,u), \tag{19}$$

and

$$L_i(x_i,u) \ge L_i(1,u), \tag{20}$$

for $u_i < 0$, $i = 1,2,\ldots,t$.

The x_i variables take two values of 0 or 1. Therefore, the maximization over the vector x in the expression (17) is replaced by (18) criterion and constraints (19) and (20), which are $2t$ in total. The resulting feasible solution with respect to x variable is optimal (maximal) because it satisfies constraints (18) and (19), thus satisfying (15).

Formally, x may not be a feasible solution (and therefore a maximum solution, due to the (19)–(20) constraints) and we should determine an admissible, let us call it x', solution on the CPU after completing the calculations in the D-Wave environment. This can be done independently for each x_i in time $O(n^2)$ according to the formula (14) by deciding whether to choose 0 or 1, obtaining maximum $L_i(x_i,u)$ for each $i = 1,2,\ldots,t$, $t \le n$. However, our experiments with the *LeapHybridSampler* solver always produced a feasible x.

The Lagrange function multipliers vector $u = (u_1, u_2, \ldots, u_t)$ formally consists of real numbers. It is uncomfortable to code them in the CQM model as a continuous-integer approach, that is why we decide to use an integers-value u vector and divide it by a power of 10 next (e.g., by 1000) to achieve a float number with fixed accuracy (e.g., 0.001).

Summing up, the calculation of $UB_{on-time}$ value according to (17) requires the determination of the optimal values of the variables x_i, $i = 1,2,\ldots,t$. For each x_i, $i = 1,2,\ldots,t$, optimization with respect to x_i can be performed on a CPU independently by checking two values of x_i, i.e., 0 or 1. Optimization with respect to u may be approximate (because for any $u_i \le 0$ the result is an upper bound, see Remark 1).

3.3. Objective function bounds in the B&B method for the WNT problem

In accordance with (8) the lower bound of the WNT problem is

$$LB_\pi = \sum_{i=t+1}^{n} w_{\pi(i)} U_{\pi(i)} + LB_{\mathcal{Y}(\pi)}$$

Let us assume that LB_π is designated on a D-Wave machine with the use of $QuantumAnnealingLB(\pi)$ function, therefore

$$LB_\pi = QuantumLagrangeLB(\pi) = \sum_{i=t+1}^{n} w_{\pi(i)} U_{\pi(i)} +$$

$$+ \left(\sum_{i=1}^{t} w_i - UB_{on-time} \right). \tag{21}$$

where $UB_{on-time}$ is defined in Eq. (17), and $(\sum_{i=1}^{t} w_i - UB_{on-time})$ is the lower bound of the objective function for the WNT problem for the set of free jobs $\mathcal{Y}(\pi)$.

Similarly, problem formulation (10)–(12), as linear programming, can directly apply the *LeapHybridCQMSampler* solver for hardware quantum annealing on the D-Wave machine. Let $LB_{on-time}$ be the value of this solution. Since on a quantum machine we do not have a guarantee of the optimal solution (due to the probabilistic nature of quantum computing), $LB_{on-time}$ is therefore the lower bound of the optimal value of the objective function for the WNO problem. Hence, according to (9) an upper bound

$$UB_\pi = \sum_{i=t+1}^{n} w_{\pi(i)} U_{\pi(i)} + UB_{\mathcal{Y}(\pi)}$$

is determined on the D-Wave by a $QuantumAnnealingUB(\pi)$ procedure,

$$UB_\pi = QuantumAnnealingUB(\pi) = \sum_{i=t+1}^{n} w_{\pi(i)} U_{\pi(i)} +$$

$$+ \left(\sum_{i=1}^{t} w_i - LB_{on-time} \right). \tag{22}$$

Both procedures for determining the lower and upper bounds of the WNT problem objective function at the node π of the solution tree $\mathcal{H}$ will be used in the B&B algorithm controlled by quantum annealing for solving the WNT problem.

4. Exact algorithm controlled by quantum annealing

In this section, we introduce the pseudocode of a hybrid exact algorithm solving the WNT problem. The key elements – lower and upper bounds – are calculated with using quantum annealing on the D-Wave computer. The identity permutation $\pi_0 = (1,2,3,\ldots,n)$ – a feasible one, since all the permutations are feasible – was assumed as the starting solution. The pseudocode is represented by the Algorithm 1.

Browsing the $\mathcal{H}$ solution tree uses the *best-first* strategy (due to upper bounds of nodes) with a priority queue implemented in a binary heap structure (*Heap* variable). In the nodes the following four are stored: (*permutation, number of free jobs, lower bound, and value of the objective function*). On line 14 of the algorithm, the candidate jobs are inserted into the last free position t. Then, on line 15, the lower bound is calculated. If its value is less than the best cost function value of a solution (line 16), then the best solution in the $\mathcal{H}(\pi)$ subtree is determined (in line 17) with using a quantum computer. It is worth noting that the ceiling ($\lceil LocalLB \rceil$) from the value of the lower bound is considered, because the objective function has integer values. In lines 18 and 19, the best solution found so far is modified if necessary. Having performed these calculations, a new node is generated (the root of the subtree), which is added to the priority queue *Heap* on line 20. The result of the algorithm's performance is the optimal permutation

Algorithm 1: QAdB&B

Input : $\pi_0 \leftarrow (1, 2, \ldots, n)$ – starting solution;
$root \leftarrow false$ – binary variable, arbitrarily $false$
Output: π^* – optimal solution;
1 $Heap \leftarrow \emptyset$: priority queue of partial solutions sorted in ascending order by their upper bounds $LocalUB$;
2 **if** $root = false$ **then**
3 $\quad \pi^* \leftarrow \pi_0$;
4 $\quad$ Put($Heap$, (π_0, n, 0, $F(\pi_0)$)); 0 is an initial lower bound
5 **else**
6 $\quad \pi_{LocalUB} \leftarrow \arg(QuantumAnnealingUB(\pi))$;
7 $\quad \pi^* \leftarrow \arg\min\{F(\pi_{LocalUB}), F(\pi_0)\}$;
8 $\quad$ Put($Heap$,(π_0, n, $QuantumLagrangeLB(\pi_0)$,$F(\pi^*)$));
9 **while** $Heap \neq \infty$ **do**
10 $\quad$ Get($Heap$, (π, t, LB_π, UB_π));
11 $\quad$ **if** $\lceil LB_\pi \rceil < F(\pi^*)$ **then**
12 $\quad\quad$ Designate the set K_π – jobs determined on t-th position;
13 $\quad\quad$ **for** $j \in K_\pi$ **do**
14 $\quad\quad\quad$ Swap($\pi, \pi^{-1}(j), t$); this procedure swaps jobs on positions $\pi^{-1}(j)$ and t in π permutation
15 $\quad\quad\quad$ $LocalLB \leftarrow QuantumLagrangeLB(\pi)$;
16 $\quad\quad\quad$ **if** $\lceil LocalLB \rceil < F(\pi^*)$ **then**
17 $\quad\quad\quad\quad$ $\pi_{LocalUB} \leftarrow \arg(QuantumAnnealingUB(\pi))$;
18 $\quad\quad\quad\quad$ **if** $\min\{F(\pi_{LocalUB}), F(\pi)\}) < F(\pi^*)$ **then**
19 $\quad\quad\quad\quad\quad$ $\pi^* \leftarrow \arg\min\{F(\pi_{LocalUB}), F(\pi)\}$;
20 $\quad\quad\quad\quad$ Put($Heap$, (π, $t-1$, $LocalLB$, $F(\pi_{LocalUB})$));
21 $\quad\quad\quad$ Swap($\pi, t, \pi^{-1}(j)$);

Table 1
Jobs' parameter of wt10 instance.

i	1	2	3	4	5	6	7	8	9	10
p_i	25	81	71	87	64	82	7	76	95	31
d_i	254	286	209	292	232	302	245	196	254	252
w_i	5	6	2	9	4	7	4	1	8	2

π^*. The $root$ switch determines whether to browse the root ($root = true$) or start the analysis from the 1st level (for $root = false$, arbitrarily). The parameter $root$ was used in our experiments to generate LB and UB, to not generate in the root. In the latter case, the analysis started from the first level (not zero) of the tree.

We illustrate the application of the QAdB&B quantum algorithm on a small example. We consider the problem of scheduling ten jobs ($n = 10$) on a single machine with the parameters presented in Table 1. Note that the data itself do not have to comply with the EDD rule, because the renumbering of the jobs takes place in each node when calculating the constraints for the dual problem.

As a starting solution of the QAdB&B algorithm there was adopted an identical (natural) permutation of jobs $\pi_0 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$. It is the only node at the level $h = 0$ of a solution tree $\mathcal{H}$. The set of free jobs $\pi^{\mathcal{B}} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, whereas the set of fixed jobs $\pi^{\mathcal{E}} = \emptyset$. There was $\pi^* = \pi^0$ adopted, $root = false$, lower bound value $LB_\pi = 0$ and the value of cost function $F(\pi^*) = 26$ was calculated.

By setting the free job on the last 10th position, there were 10 permutations generated from the π permutation — nodes of the $\mathcal{H}$ tree on the 1st level (the underlined job belongs to the set of fixed jobs, whereas the remaining jobs are free). For each of the 10 generated nodes, the value of the objective function and the values of the lower and upper bounds on the quantum machine are determined. The obtained results are presented in Table 2. The following columns contain: the node of the tree, the value of the lower bound (LB_π), and the value of the upper bound (UB_π). For comparison, the last two columns contain the

Table 2
Lower and upper bounds.

node	LB_π ($\lceil LB_\pi \rceil$)	UB_π	LB_{SM}	UB_{SM}
π_1	17.5532 (18)	18	17.7037 (18)	29
π_2	14.4602 (15)	15	14.6129 (15)	30
π_3	14.4069 (15)	15	13.8355 (14)	26
π_4	17.0756 (18)	–	17.2258 (18)	33
π_5	14.4091 (15)	15	13.8355 (14)	26
π_6	16.0670 (17)	–	16.2222 (17)	26
π_7	17.8889 (18)	–	18.0370 (19)	26
π_8	14.4056 (15)	15	13.8355 (14)	26
π_9	15.5568 (16)	–	15.7096 (16)	26
π_{10}	14.4104 (15)	15	13.8355 (14)	26

values of the lower and upper bounds (LB_{SM} and UB_{SM}, respectively) determined by the SMB&B (Silicon Machine B&B) algorithm executed on a classic computer with a silicon-based CPU.

In both algorithms (QAdB&B and SMB&B), the lowerbounds are calculated based on the Lagrange relaxation in a floating point arithmetic system. Since for any solution of the considered example, the objective function value is an integer, ultimately the value of the lower bound was assumed to be the smallest integer and not less than the value of the lower bound (function ceiling $\lceil \ \rceil$) determined by the $QuantumLagrangeLB$ function. Integer values are shown in parentheses (columns: LB_π and LB_{SM} in Table 2).

It should be noted that already in the second of the generated nodes π_2, the value of the upper bound $UB_{\pi_2} = 15$ determined by the quantum algorithm, is the optimal value. It has a permutation of $\pi_2^* = (7, 1, 9, 4, 6, 8, 3, 5, 10, 2)$ for which the value of the objective function $F(\pi_2^*) = 15$. Therefore, the following nodes: π_4, π_6, π_7 and π_9, for which the value of the lower bound (calculated in the QAdB&B algorithm) is not less than $UB_{\pi_2} = 15$ are not considered, and in particular, no upper bound for them is determined. The algorithm ends after generating all the first level nodes, because the lower bound for each of these nodes is not less than the optimal value $F(\pi_2^*) = 15$ determined in node π_2.

The QAdB&B algorithm results, listed in Table 2, were also compared with the results of the SMB&B algorithm. First, the values of the lower bounds determined by both algorithms were compared. In the case of the silicon algorithm, to compute the lower bound there was used a very good Powell algorithm (Powell, 1964) for the implemented continuous optimization. The obtained results are listed in Table 2 column 4. Only in 1 of 10 cases (node π_7) is the ceiling value (rounded to the nearest integer) from the lower bound set in the silicon algorithm slightly better (greater) than the value set by the quantum algorithm.

It is interesting to note that at each node of the solution tree, both the quantum and silicon algorithms solved the same subproblem. In the case of the lower bound, it was the optimization of the Lagrange function (by various methods: quantum annealing on the QPU and the Powell method on the CPU). For the upper bound, the QPU performed constant-time quantum annealing. In turn, the CPU in constant time was able to calculate only the value of the objective function of a single solution in the tree node. Hence the differences in the determined bounds values.

In the full run of the QAdB&B algorithm, 10 tree nodes were generated, and the optimal solution was obtained in the second of the generated nodes, π_2 (the algorithm generated the whole tree, calculations were not stopped after obtaining the optimum). However, the silicon algorithm generated as many as 1006 nodes, but the optimal solution was found only in the 207-th node.

The average runtime of a single iteration of the QAdB&B algorithm on the QPU is 15.26 ms and the average deviation of the runtime is only 0.034. Undoubtedly, it seems that the time of determining the lower and upper bounds does not depend on the data (i.e., the number of free jobs) and in practice can be considered as constant for this size of data. The total computation time on the quantum computer was 254 ms, with the optimal solution being obtained after less than 60 ms. In turn, the

calculation time of the algorithm on a silicon computer is 5960 ms, and the optimal solution was obtained after 5934 ms.

In summary, the QAdB&B algorithm running on a D-Wave environment generated about 100 times fewer solution tree nodes than the silicon algorithm, and the computation time was over 23 times shorter. Moreover, the quantum algorithm found the optimal solution much faster, both in terms of the time and the number of generated tree nodes.

5. Computational experiments

Two types of tests are presented in this section. The results of the QAdB&B (Quantum Annealing-driven B&B) algorithm, for instances of an average size ($n \leq 100$), were compared with the results of the SMB&B (Silicon Machine B&B) exact algorithm ran on a classic silicon computer with a CPU. However, for examples of a large size ($200 \leq n \leq 1000$), the results were compared with the results determined by the Gurobi software package. The proposed QAB&B algorithm was run in two versions: HybridQAB&B in the LeapHybridSampler environment using the CPU, GPU and QPU (and automatically defining tasks for the quantum processor performing quantum annealing) and QAB&B in the D-WaveSampler environment (using quantum annealing directly). A discussion about QPU usage by LeapHybridSampler can be found in the work of Stogiannos et al. [8].

Silicon-machine B&B (SMB&B) algorithm. The QAdB&B and SMB&B algorithms differ, in addition to the runtime environments (CPU+QPU versus CPU), in the method of calculating the lower and upper bounds. For the SMB&B, the upper bound value of $\pi_{LocalUB}$ is substituted as $\pi_{LocalUB} \leftarrow \pi$ (lines 6 and 17 of Algorithm 1). In turn, with regard to the lower bound in the algorithm SMB&B suboptimal values of the coefficients of vector u of the Lagrange function (13) are computed by the Powell method [32] of continuous optimization ($LocalLB$ in line 15 of Algorithm 1; also line 8). The algorithm for their determination ends when the obtained improvement in the Lagrange function value is not greater than the predetermined value of 10^{-6}. In each iteration, $O(n^2)$ values of the variables appearing in the Lagrange function are determined. In addition, the number of iterations of Powell's algorithm depends on the value of the data and the accuracy of the calculations. Algorithm SMB&B was programmed in C# and ran on a computer with a 3.4 GHz processor.

Results. The calculations were made on the well-known examples of test data for the single machine scheduling problem with weights, $1||\sum w_i T_i$, posted on the page OR-Library [33]. These are examples of various sizes and varying degrees of difficulty commonly used when testing algorithms for solving single machine problems with weights. The calculations were made on the selected 30 instances of medium size:

(a) 10 denoted by $wt40\_011 - wt40\_020$ of size $n = 40$,
(b) 10 denoted by $wt50\_011 - wt50\_020$ of size $n = 50$,
(c) 10 denoted by $wt100\_011 - wt100\_020$ of size $n = 100$,

and 30 instances of a larger size (Uchroński [34]):

(c) 10 denoted by $wt200\_021 - 025$ and $wt200\_071 - wt200\_075$ of size $n = 200$,
(d) 10 denoted by $wt500\_046 - 050$ and $wt500\_091 - wt500\_095$ of size $n = 500$,
(e) 10 denoted by $wt1000\_021 - 025$ and $wt1000\_046 - wt1000\_050$ of size $n = 1000$.

Additionally, a single instance wt_10 of the size $n = 10$ (apart from instances with $n = 40$) was presented in Table 5 for comparison. As part of the computational experiments carried out, there were comparisons made of the lower values and upper bounds determined by the quantum annealer with the values of the optimal, lower and upper bounds calculated on the silicon machine. The optimal values of the solutions were obtained using the GUROBI software package by formulating the WNT problem as linear programming.

The results of the experimental tests are presented in Table 3. The first column contains the name and size of the example. The columns contain the following values: lower bound LB_π (as well as $\lceil LB_\pi \rceil$ integer value), upper bound UB_π, and runtimes QPU_ACCESS_TIME and RUN_TIME of lower and upper bounds, respectively, calculated in Leap Hybrid quantum computing environment (see D-Wave timing documentation [35]). Next two columns show the value of the lower and upper bound set by the silicon computer, while the last column CPU_TIME presents the time of determining the lower bound on this computer. In the algorithm implemented on a silicon computer (SMB&B), the value of the objective function for the permutation generated according to the scheme of the B&B algorithm was adopted as the upper bound. 8th column contains a CPU runtime of SMB&B algorithm.

Since the values of the objective function considered in the work of the WNT problem are integers, ultimately the value of the lower bound was assumed to be the smallest integer not less than the calculated value (ceiling $\lceil\ \rceil$ function). These approximations are in parentheses in columns 'LB_π ($\lceil LB_\pi \rceil$)' for HybridQAdB&B and 'LB ($\lceil LB \rceil$)' for SMB&B. The QAdB&B algorithm was run with root consideration ($root = true$ – it can be skipped or not during the algorithm work, depending on the $root$ variable).

Having performed the calculations, it turned out that the upper bounds of the algorithm HybridQAdB&B (Hybrid because of running in the D-Wave Leap environment) (UB_π column in Table 3) determined in the root of the solution tree are optimal values (they are marked in bold). Using the Gurobi package, all the examples were solved and the same optimal solution values were obtained as in the column 3. These values are definitely smaller (even more than 4 times smaller, regarding the test example wt40_011) than those determined by the silicon CPU-based SMB&B algorithm (column 8). On the other hand, the lower bounds set by both algorithms (columns 'LB_π' and 'LB') differ little (due to the non-deterministic nature of the calculations), although for more examples the SMB&B algorithm set bounds of a lower value. The computation time of the silicon SMB&B algorithm (column CPU_TIME) grows exponentially with increasing data size, whereas the computation time of the HybridQAdB&B algorithm is practically constant (columns QPU_ACCESS_TIME and RUN_TIME) and is just over 15 ms on QPU and about 5 s as a RUN_TIME. Column δ presents a percentage gap between lower ($\lceil LB_\pi \rceil$) and upper bounds (UB_π) obtained by HybridQAdB&B algorithm.

Table 4 presents the results of calculations for larger examples (for $n = 200, 500, 1000$). The running time of the HybridQAdB&B algorithm was limited because the review of the entire tree was impossible in this case within the time limit of the D-Wave Leap computational environment at our disposal under an academic license. The obtained results of the HybridQAdB&B (*de facto* cut B&B) algorithm are therefore approximate solutions. They were compared with the exact results obtained using the Gurobi package. The meaning of the first 6 columns is the same as in Table 3. The last 3 columns, for the results of the Gurobi package, mean: the value of the obtained solution (OPT, optimal), the calculation time RUN_TIME, and percentage gap in relation to the value of the solution obtained with the HybridQAdB&B algorithm: $gap = \frac{UB_\pi - OPT}{OPT} \cdot 100\%$.

Fig. 2 graphically presents the summary results included in the Tables 3 and 4. The bar graph represents the average percentage relative error (gap) of the HybridQAdB&B algorithm, and the broken line represents the average computation time. The values of relative errors vary greatly. For the number of jobs $n = 40, 50, 100$ the error is 0 (the solutions are optimal). For $n = 200$ this error is 0.177. If we increase the number of jobs by 2.5 times (to $n = 500$), the error increases almost 18 times, to a value of 3.141. However, for the number of jobs $n = 1000$ the gap is 1.272, so in relation to the number of jobs $n = 200$ it increases

Table 3
Results of computational experiments (optimal solutions).

Instance	HybridQAdB&B					SMB&B		
	LB_π ($\lceil LB_\pi \rceil$)	UB_π	QPU_ACCESS_TIME [ms]	RUN_TIME [s]	δ [%]	LB ($\lceil LB \rceil$)	UB	CPU_TIME [ms]
wt40_011	32.07 (33)	**34**	15.29	5.08	2.94	33.49 (34)	137	366
wt40_012	35.71 (36)	**39**	15.34	5.00	7.69	36.83 (37)	114	328
wt40_013	46.90 (47)	**50**	15.35	5.00	6.00	47.54 (48)	166	331
wt40_014	34.17 (35)	**36**	15.28	5.00	2.78	34.41 (35)	130	324
wt40_015	48.72 (49)	**51**	15.33	5.00	3.92	47.49 (48)	149	324
wt40_016	99.93 (100)	**105**	15.32	5.41	4.76	99.24 (100)	167	324
wt40_017	100.68 (101)	**103**	15.34	5.21	1.94	100.09 (101)	168	324
wt40_018	101.69 (102)	**103**	15.34	5.00	0.97	102.18 (103)	174	329
wt40_019	96.67 (97)	**99**	15.33	5.09	2.02	97.12 (98)	170	326
wt40_020	87.08 (88)	**89**	15.33	5.43	1.12	86.90 (87)	201	326
average			15.32	5.12	3.41			330
wt50_011	55.46 (56)	**59**	15.33	5.31	5.08	56.97 (57)	160	593
wt50_012	43.01 (44)	**46**	15.20	5.00	4.35	44.97 (45)	177	593
wt50_013	62.73 (63)	**65**	15.33	5.02	3.08	60.48 (61)	181	583
wt50_014	70.20 (71)	**75**	15.34	5.35	5.33	72.13 (73)	174	619
wt50_015	54.29 (55)	**57**	15.27	5.46	3.51	56.63 (57)	132	599
wt50_016	107.21 (108)	**110**	15.33	5.00	1.82	107.21 (108)	249	592
wt50_017	88.03 (89)	**91**	15.33	5.15	2.20	87.54 (88)	221	598
wt50_018	107.36 (108)	**109**	15.33	5.33	0.92	106.90 (107)	247	584
wt50_019	110.25 (111)	**112**	15.35	5.29	0.89	111.13 (112)	228	589
wt50_020	89.93 (90)	**93**	15.34	5.05	3.23	90.39 (91)	185	758
average			15.31	5.20	3.04			610
wt100_011	119.14 (120)	**127**	15.29	5.32	5.51	121.81 (122)	186	4193
wt100_012	145.48 (146)	**154**	15.34	5.34	5.19	150.42 (151)	189	4189
wt100_013	115.46 (116)	**125**	15.35	5.36	7.20	122.86 (123)	171	4098
wt100_014	106.62 (107)	**116**	15.35	5.34	7.76	112.14 (113)	148	4165
wt100_015	124.57 (125)	**134**	15.30	5.00	6.72	128.54 (129)	188	4209
wt100_016	233.25 (234)	**237**	15.36	5.08	1.27	234.23 (235)	317	4169
wt100_017	191.64 (192)	**195**	15.36	5.49	1.54	189.72 (190)	260	4367
wt100_018	273.63 (274)	**278**	15.33	5.00	1.44	269.09 (270)	330	4119
wt100_019	245.48 (246)	**249**	15.28	5.46	1.20	246.65 (247)	348	4157
wt100_020	224.57 (225)	**227**	15.36	5.00	0.88	219.14 (220)	303	4298
average			15.33	5.24	3.87			4196

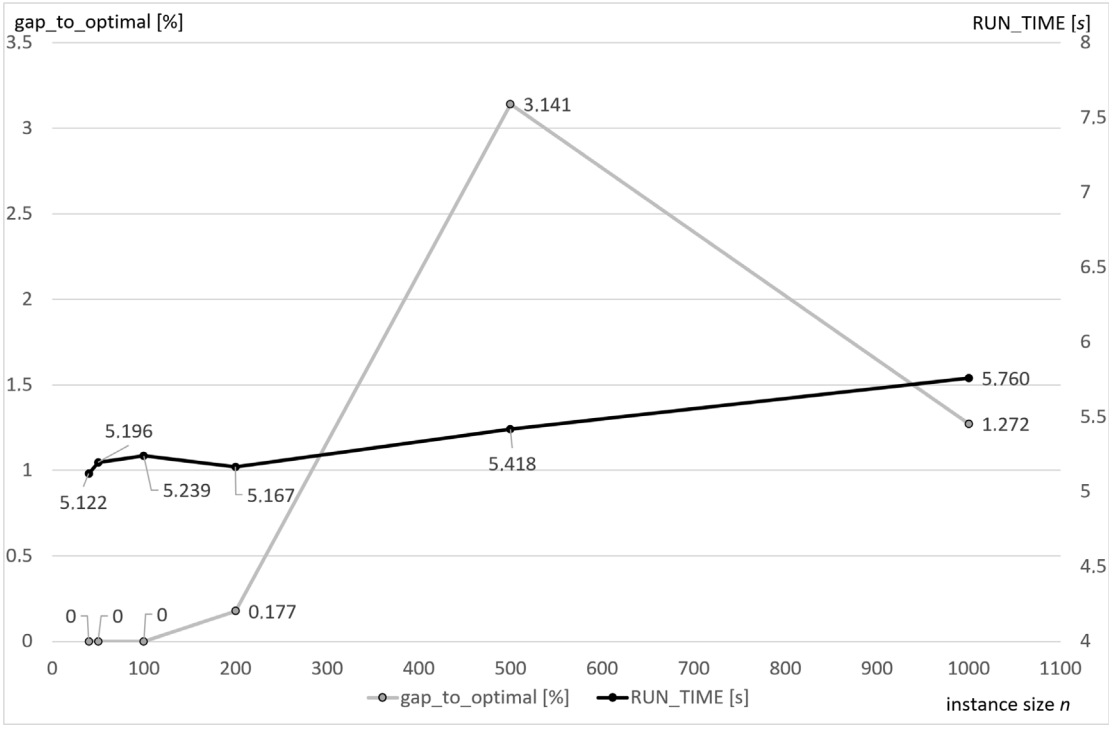

Fig. 2. HybridQAdB&B efficiency and runtime comparison for different instance sizes n.

only by about 7 times. Based on the calculations performed, it cannot be concluded that as the size of the examples (number of jobs) increases, the average relative error also increases. In general, the average total RUN_TIME computation times increase as the number of jobs increases

Table 4
Results of computational experiments (with limited computation time).

Instance	HybridQAdB&B with time limit					Gurobi		gap [%]
	LB_π ($\lceil LB_\pi \rceil$)	UB_π	QPU_ACCESS_TIME [ms]	RUN_TIME	δ [%]	OPT	RUN_TIME [ms]	
*wt*200_021	681.25	689	16.02	5.19	1.13	687	50	0.29
*wt*200_022	767.26	773	16.03	5.29	0.74	772	60	0.13
*wt*200_023	757.16	**764**	15.91	5.34	0.90	764	110	0.00
*wt*200_024	726.37	**734**	16.00	5.16	1.04	734	60	0.00
*wt*200_025	719.06	**724**	15.99	5.00	0.68	724	50	0.00
*wt*200_071	424.98	**435**	31.93	5.09	2.30	435	40	0.00
*wt*200_072	372.71	382	16.02	5.00	2.43	381	30	0.26
*wt*200_073	448.38	463	16.05	5.06	3.16	460	50	0.65
*wt*200_074	451.45	461	31.98	5.36	2.07	459	70	0.44
*wt*200_075	409.00	**418**	16.04	5.18	2.15	418	60	0.00
average			19.20	5.17	1.66		58	0.18
*wt*500_046	1183.16	1407	15.98	5.20	15.91	1377	320	2.18
*wt*500_047	907.64	1356	16.00	5.64	33.07	1336	190	1.50
*wt*500_048	1289.80	1355	16.05	5.57	4.81	1323	420	2.42
*wt*500_049	1106.48	1388	16.04	5.28	20.28	1360	280	2.06
*wt*500_050	1062.24	1380	15.92	5.65	23.03	1357	370	1.69
*wt*500_091	366.35	424	16.05	5.16	13.60	404	660	4.95
*wt*500_092	394.66	428	16.00	5.35	7.79	417	310	2.64
*wt*500_093	296.42	398	16.02	5.81	25.52	386	270	3.11
*wt*500_094	361.21	468	16.00	5.32	22.82	446	350	4.93
*wt*500_095	316.65	411	16.02	5.20	22.96	388	290	5.93
average			16.01	5.42	18.98		346	3.14
*wt*1000_021	2277.87	3620	16.04	5.61	37.08	3603	1360	0.47
*wt*1000_022	3293.14	3560	16.02	5.59	7.50	3542	2130	0.51
*wt*1000_023	2745.88	3644	16.03	5.66	24.65	3618	1540	0.72
*wt*1000_024	2369.86	3462	31.93	5.60	31.55	3446	1290	0.46
*wt*1000_025	3435.10	3627	16.01	5.79	5.29	3590	1230	1.03
*wt*1000_046	1835.11	2858	16.00	5.93	35.79	2802	1770	2.00
*wt*1000_047	1849.36	2731	16.03	5.91	32.28	2688	1130	1.60
*wt*1000_048	2305.27	2807	15.91	5.86	17.87	2747	1850	2.18
*wt*1000_049	1770.87	2759	16.02	5.77	35.81	2719	3420	1.47
*wt*1000_050	2336.74	2693	16.02	5.88	13.23	2633	1470	2.28
average			17.60	5.76	24.11		1719	1.27

Table 5
LeapHybridSampler and DWaveSampler comparison.

Instance	HybridQAdB&B – LeapHybridCQMSampler				QAdB&B – DWaveSampler		
	LB_π ($\lceil LB_\pi \rceil$)	UB_π	QPU_ACCESS_TIME [ms]	RUN_TIME [s]	LB_π ($\max\{0, LB_\pi\}$)	UB_π	QPU_ACCESS_TIME [ms]
wt10	14.46 (15)	**15**	31.98	5.07	−41.4886 (0)	32	15.91
wt40_011	32.07 (33)	**34**	15.29	5.08	−87.5467 (0)	161	16.04
wt40_012	35.71 (36)	**39**	15.34	5.00	−91.4260 (0)	122	16.05
wt40_013	46.90 (47)	**50**	15.35	5.00	−132.933 (0)	148	16.05
wt40_014	34.17 (35)	**36**	15.28	5.00	−85.579 (0)	105	15.93
wt40_015	48.72 (49)	**51**	15.33	5.00	−112.3733 (0)	148	15.94
wt40_016	99.93 (100)	**105**	15.32	5.41	−70.6645 (0)	211	15.96
wt40_017	100.68 (101)	**103**	15.34	5.21	−75.1634 (0)	178	15.93
wt40_018	101.69 (102)	**103**	15.34	5.00	−69.7733 (0)	182	15.95
wt40_019	96.67 (97)	**99**	15.33	5.09	−78.0840 (0)	178	15.95
wt40_020	87.08 (88)	**89**	15.33	5.43	−31.0918 (0)	216	15.94
average			15.32	5.12			15.97

(with the exception of $n = 200$). However, this increase is small, because for the smallest examples ($n = 40$) the calculation time is 5.122 s, and for the largest ($n = 1000$) the calculation time is 5.760 s The data size increased by 25 times, and the computation time increased by only about 0.6 s For data for which computational experiments were carried out, it is practically independent of the size of the example.

Table 5, for examples with the number of tasks $n = 10$ and $n = 40$, presents the values of the lower and upper bounds of the values of optimal solutions, QPU_ACCESS_TIME and RUN _TIME (for DWaveSampler, the QPU_ACCESS_TIME time is the total computation time on the QPU). The calculations were performed using two versions of the QAdB&B algorithm: in the LeapHybridSampler environment and DWaveSampler, the description of which is presented at the beginning of this section. For the algorithm version LeapHybridSampler maximal difference between upper and lower bounds $UB_\pi - \lceil LB_\pi \rceil = 5$ for example wt40_016. Generally, it is small (UB_π is the value of the optimal solution). In turn, the calculations of the second version of the algorithm, DWaveSampler, were performed directly using quantum annealing, i.e. only on the QPU. The values of the lower limits LB_π are determined in this case in two stages: (i) the upper bound UB_{WNO} for the WNO problem (maximizing $\sum w_i(1 - U_i)$ is calculated), (ii) lower bound $LB_\pi = \sum w_i U_i - UB_{WNO}$. Since the upper bounds UB_{WNO} for the WNO problem are weak here (much larger than the optimal values — just compare them with the corresponding values for the LeapHybridSampler version), hence the lower bounds for the WNT problem are negative. In practice, we assumed 0 (the value of the objective function is non-negative, see column 6 of Table 5, zeros in brackets). The computation times of both versions of the algorithms differ significantly. The proportion of QPU_ACCESS_TIME of DWaveSampler to RUN_TIME of LeapHybridCQMSampler is in our case (15.97 ms.)/(5.12 s) ≈ 0.003.

When analyzing Table 5, which compares the solvers: hybrid LeapHybridCQMSampler and native DWaveSampler, significant differences can be noticed. They result indirectly from a different representation of the considered optimization problem. For the HybridQAdB&B algorithm using the LeapHybridCQMSampler environment, this is the CQM model, i.e. Constrained Quadratic Model. It is solved by the hybrid solver using both classical methods (e.g., including the tabu search algorithm) and, in part, quantum methods. However, quantum methods require a QUBO representation which in practice can be used for relatively small subproblems of the problem under consideration. This is clearly visible in the relatively short QPU_ACCESS_TIME for this computation model. In turn, the use of the native DWaveSampler model implementing only quantum annealing requires the conversion of CQM to BQM (Binary Quadratic Model) at the model preparation level, which is associated with a significant increase in the number of variables (only binary ones) compared to the CQM model. The time of calculation QPU_ACCESS_TIME practically does not change – it is the time needed to physically implement the quantum annealing process – but the results differ in quality from those obtained in the hybrid model.

6. Conclusions

Performing computations on a quantum computer creates new possibilities for solving NP-hard discrete optimization problems. Although quantum annealing does not guarantee obtaining an optimal solution, it can be successfully used in the construction of an exact algorithm. This is because it significantly improves its efficiency through (nondeterministic) control of the process of searching the solution space. The paper presents the concept of a hybrid quantum exact algorithm based on the Branch and Bound method of solving a single-machine scheduling problem with the minimization of the weighted number of tardy jobs. The calculation results of the quantum annealing-based algorithm are used to determine the calculation trajectory.

The computational experiments performed showed that the relative difference between the upper and lower bound values calculated by the B&B algorithm with using a quantum machine is small and its mean value does not exceed four percent. On this basis, it is possible to approximate the value of the optimal solution with high accuracy. The measured execution times of the algorithm on the QPU compared with its version on the CPU indicate the possibility of quantum acceleration, i.e., the fact of solving the problem on a quantum machine faster than in its classic silicon equivalent. Computational experiments were performed on a D-Wave quantum machine. The presented algorithm can be adapted to solve other NP-hard permutation problems, such as the traveling salesman problem (TSP) or quadratic assignment problem (QAP).

The open perspectives for further work in the near future include extending the obtained results to other quantum machine programming environments, e.g., IBM Qiskit, both in the quadratic programming approach and in the quantum gate model. Another direction of research is related to the development of methods for embedding large examples on limited-size quantum processors. This requires the development of a new methodology for partitioning the solution space and the synchronization of parallel quantum computations.

CRediT authorship contribution statement

Wojciech Bożejko: Writing – review & editing, Methodology, Investigation, Conceptualization. **Jarosław Pempera:** Software, Methodology, Investigation, Data curation. **Mariusz Uchroński:** Visualization, Validation, Software, Methodology, Investigation, Funding acquisition. **Mieczysław Wodecki:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Centers: ACK CyfronetAGH) for providing computer facilities and support within computational grant no. PLG/2023/016790.

References

[1] W. Bozejko, J. Pempera, M. Uchroński, M. Wodecki, Distributed quantum annealing on D-wave for the single machine total weighted tardiness scheduling problem, in: D. Groen, C. de Mulatier, M. Paszynski, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2022, in: LNCS 13353, Springer, 2022, pp. 171–178.

[2] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, O. Regev, Adiabatic quantum computation is equivalent to standard quantum computation, SIAM Rev. 50 (4) (2008) 755–787.

[3] A. Montanaro, Quantum speedup of branch-and-bound algorithms, 2019, pp. 1–11, arXiv:1906.10375.

[4] E.A. Markevich, A.S. Trushechkin, Quantum branch-and-bound algorithm and its application to the travelling salesman problem, J. Math. Sci. 241 (2019) 168–184.

[5] K. Ikeda, Y. Nakamura, T.S. Humble, Application of quantum annealing to nurse scheduling problem, Sci. Rep. 9 (2019) 12837.

[6] C. Carugno, M. Ferrari Dacrema, P. Cremonesi, Evaluating the job shop scheduling problem on a D-wave quantum annealer, Sci. Rep. 12 (2022) 6539.

[7] W. Bozejko, R. Klempous J. Rozenblit, J. Pempera, M. Uchroński, M. Wodecki, Optimal solving of a scheduling problem using quantum annealing metaheuristics on the D-wave quantum solver, 2023, http://dx.doi.org/10.36227/techrxiv.22677721.v1, TechRxiv, Preprint.

[8] E. Stogiannos, C. Papalitsas, T. Andronikos, Experimental analysis of quantum annealers and hybrid solvers using benchmark optimization problems, Mathematics 10 (8) (2022) 1294, http://dx.doi.org/10.3390/math10081294.

[9] W. Bozejko, J. Pempera, M. Uchroński, M. Wodecki, An exact quantum annealing-driven branch and bound algorithm for maximizing the total weighted number of on-time jobs on a single machine, in: M. Pawelczyk, D. Bismor, S. Ogonowski, J. Kacprzyk (Eds.), Advanced, Contemporary Control. PCC 2023, in: Lecture Notes in Networks and Systems, vol. 709, Springer, 2023, pp. 79–89.

[10] R.M. Karp, Reducibility amomg combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computation, Plenum Press, New York, 1972, pp. 85–104.

[11] J.K. Lenstra, A.H.G. Rinnoy Kan, Complexity results for scheduling chains on a single machine, European J. Oper. Res. 4 (1980) 270–275.

[12] C.I. Monma, Linear-time algorithms for scheduling on parallel processor, Oper. Res. 30 (1982) 116–124.

[13] J.M. Moore, An n-job, one machine sequencing algorithm for minimising the number of late jobs, Manage. Sci. 15 (1968) 102–109.

[14] E.L. Lawler, A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, Ann. Discrete Math. 1 (1977) 331–342.

[15] M.R. Garey, D.S. Johnson, Scheduling tasks with nonuniform deadlines on two processor, J. ACM 23 (1976) 461–467.

[16] S.K. Sahni, Algorithms for scheduling independent jobs, J. Assoc. Comput. Mach. 23 (1976) 116–127.

[17] F.J. Villareal, R.L. Bulfin, Scheduling a single machine to minimize the weighted number of tardy jobs, IEE Trans. 15 (1983) 337–343.

[18] C.N. Potts, L.N. Van Wassenhove, A branch and bound algorithm for the total weighted tardiness problem, Oper. Res. 33 (1985) 177–181.

[19] R. M'Hallah, R.L. Bulfin, Minimizing the weighted number of tardy jobs on a single machine, European J. Oper. Res. 145 (2003) 45–56.

[20] L. Hejl, P. Š° ucha, A. Novák, Z. Hanzálek, Minimizing the weighted number of tardy jobs on a single machine: Strongly correlated instances, European J. Oper. Res. 298 (2) (2022) 413–424.

[21] C. Briand, S. Ourari, Minimizing the number of tardy jobs for the single machine scheduling problem: Mip-based lower and upper bounds, RAIRO-Oper. Res. (Recherche Opérationnelle) 47 (1) (2013) 33–46.

[22] P. Rajba, M. Wodecki, : Stability of scheduling with random processing times on one machine, Appl. Math. 39 (2012) 169–183.
[23] A. Ajagekar, T. Humble, F. You, Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems, Comput. Chem. Eng. 132 (2020) 106630.
[24] B. Denkena, F. Schinkel, J. Pirnay, S. Wilmsmeier, Quantum algorithms for process parallel flexible job shop scheduling, CIRP J. Manuf. Sci. Technol. 33 (2021) 100–114.
[25] C. McGeoch, P. Farre, Hybrid Solver for Constrained Quadratic Models, Technical Report, 2021, https://www.dwavesys.com/media/rldh2ghw/14-1055a-a_hybrid_solver_for_constrained_quadratic_models.pdf.
[26] C. McGeoch, P. Farre, The Advantage System: An Overview, Technical Report, 2020, https://www.dwavesys.com/media/s3qbjp3s/14-1049a-a_the_d-wave_advantage_system_an_overview.pdf.
[27] C. McGeoch, P. Farre, The Advantage System: Performance Update, Technical Report, 2021, https://www.dwavesys.com/media/kjtlcemb/14-1054a-a_advantage_system_performance_update.pdf.
[28] C. McGeoch, P. Farre, W. Bernoudy, Hybrid Solver Service Advantage: Technology Update, Technical Report, 2020, https://www.dwavesys.com/media/m2xbmlhs/14-1048a-a_d-wave_hybrid_solver_service_plus_advantage_technology_update.pdf.
[29] K. Boothby, P. Bunyk, J. Raymond, A. Ray, Next-Generation Topology of D-Wave Quantum Processors, Technical Report, 2019, https://www.dwavesys.com/media/jwwj5z3z/14-1026a-c_next-generation-topology-of-dw-quantum-processors.pdf.
[30] M. Wodecki, A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem, Int. J. Adv. Manuf. Technol. 37 (2008) 996–1004.
[31] E.L. Lawler, J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Manage. Sci. 16 (1) (1969) 77–84.
[32] M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Comput. J. 7 (2) (1964) 155–162.
[33] OR-Library: http://people.brunel.ac.uk/mastjjb/jeb/info.html.
[34] M. Uchroński, Test instances for a single-machine total weighted tardiness scheduling problem, instances data, 2022, https://zasobynauki.pl/zasoby/test-instances-for-a-single-machine-total-weighted-tardiness-scheduling-problem,51561/.
[35] D-Wave Timing Documentation https://docs.dwavesys.com/docs/latest/c_qpu_timing.html.

Wojciech Bożejko is a Full Professor at Wroclaw University of Technology. He obtained M.Sc. in University of Wroclaw, Institute of Computer Science in 1999, Ph.D. in Wroclaw University of Technology, Institute of Engineering Cybernetics in 2003, D.Sc. (habilitation in Wroclaw University of Technology, Faculty of Electronics, in 2011 and Full Professor title in 2020. From 2019 he is the Head of the Department of Control Systems in Wroclaw University of Science and Technology. He is an author of over 240 papers (over 1000 citations on Scopus, h-index: 16n published in peer-reviewed journals and conference proceedings from the field of parallel processing, scheduling and optimization. He is interested in quantum optimization, parallel algorithms, scheduling and discrete optimization. He is also a qualified musician, graduated (M.A. degreen in 1998 from the Academy of Music in Wroclaw in a specialization of piano.

Jaroslaw Pempera (1970–2023) was an Associated Professor at Wroclaw University of Science and Technology. He received the Ph.D. degree from Wroclaw University of Science and Technology in 2001. He was an Associated Professor in Department of Control Systems and Mechatronics, Faculty of Electronics. His main research interests included developing advanced optimization algorithms for scheduling in complex manufacturing systems. He cooperated with main industry companies in Poland designing AGV optimization and scheduling systems for manufacturing applications.

Mariusz Uchroński is a leader of programmers team at Wroclaw Center for Networking and Supercomputing(WCSSn and researcher at Wroclaw University of Science and Technology, Department of Control Systems and Mechatronics. He obtained his Ph.D. from Wroclaw University of Science and Technology, Institute of Computer Engineering, Control and Robotics in 2014 in the field of control and robotics. In 2022 he obtained D.Sc. (habilitation in the field of information and communication technology and taking position of Associate Professor at the Faculty of Information and Communication Technology. His areas of research interest include parallel algorithms, software design and development, HPC computing, scheduling, discrete optimization and quantum computing. His scientific achievements: co-author of several scientific papers published in peer-reviewed journals and conference proceedings in the field of parallel processing, software design and development, scheduling, optimization and quantum computing. He was involved in several R&D projects, such as PRACE 2IP/3IP/4IP/5IP/6IP, PLGrid, SPIN-LAB, AZON, RegSOC, AZON2, EuroHPC PL, Dariah.lab.

Mieczysław Wodecki is a Full Professor in Department of Telecommunications and Teleinformatics, Wroclaw University of Science and Technology. He is the author of two books, more than 250 (over 900 citations on Scopus, h-index: 15n articles, and many funded research projects. His current research interests involve the design and application of mathematical methods and constructions of classical and quantum algorithms to real problems in combinatorial optimization.