Optimal solving of scheduling problems on D-Wave quantum machines

Wojciech Bożejko

Faculty of Information and Communication Technology WIT
Wrocław University of Science and Technology
Poland
wojciech.bozejko@pwr.edu.pl

Wrocław University
of Science and Technology

Plan of the presentation

Introduction

- The concept of quantum computing and quantum computers emerged in the 1980s. Currently, there have been machines representing one of two approaches to quantum computing available.
- The first, represented by companies such as **Google, Honeywell, IBM and Intel**, are quantum computers with **quantum gate models** (e.g. Hadamard gate and Toffoli gate). Unlike many classical logic gates, quantum logic ones are reversible.

Programming in quantum gate model of computing is still a major challenge due to the small scale of solvable problems and the lack of a high-level approach adequate to high-level languages in programming of classical silicon-based computers.

Introduction

- The second approach, **quantum annealing**, by using effects known as quantum fluctuations and quantum tunneling, determines the possible best solution to the optimization problem. **D-Wave** Systems Company and **NEC** proposing an approach to computation that is admittedly limited to the use of quantum annealing, but which fits perfectly with the needs of the **operations research** discipline.

In this case, instead of expressing the algorithm for solving the problem under study in terms of quantum gates, the user presents it in terms of a **quadratic programming problem** (QUBO).

Introduction

Tasks formulated for a quantum machine implementing quantum annealing take the form of an Ising or QUBO model. The Ising model is used in statistical mechanics and the criterion function has the Hamiltonian 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 $J_{i,j}$.

Introduction

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

QUBO is an unconstrained model. Some of the *D-Wave* 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 below presented work is dedicated.

Introduction

- The main disadvantage of calculations on real quantum computers is its non-determinism. For optimization problems, of course, it is possible to get sensationally good results, but without a guarantee of the actual optimality of the result.
- A method which guarantees of optimality is proposed here.
- We use a D-Wave quantum machine working as a sampler implementing quantum annealing – an approach considered a hardware metaheuristic – to obtain upper and lower bounds of the objective function of the problem under consideration.
- The mechanism of a Branch and Bound scheme controlled by quantum annealing is applied, which allows us to obtain very quickly – because in constant time – the boundaries of the considered subproblems.

Case study – the single-machine problems

- The set of tasks is given $\mathcal{J} = \{1, 2, \ldots, n\}$.
- For the $i \in \mathcal{J}$ task, let us define:
  - p_i – processing time,
  - d_i – due date, and
  - w_i – weight of the cost function for the task's tardiness.
- Each task must be performed on the machine, the following restrictions must be met:
  (a) the machine can perform at most one task at any given time,
  (b) task execution cannot be interrupted,
  (c) the task execution may begin at time zero.

Model formulation

Any solution to the considered problem can be represented by the sequence $S_1, S_2, \ldots, S_n$ of tasks starting times, with the constraints:

$$S_i + p_i \leqslant S_j \vee S_j + p_j \leqslant S_i,\ i \neq j,\ i,j = 1,2,\ldots,n, \tag{3}$$

$$S_i \geqslant 0,\ i = 1,2,\ldots,n \tag{4}$$

Due to regularity of the goal function of the form of sum of tardinesses, solution $S_1, S_2, \ldots, S_n$ can be represented by the order of execution of tasks expressed by a permutation $\pi \in \Pi$ of elements of the set $\mathcal{J}$, where Π is the set of all such permutations.

Goal function

For any permutation of $\pi \in \Pi$, penalty for tasks tardinesses (solution cost) is

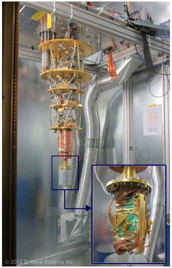
$$\mathcal{F}(\pi) = \sum_{i=1}^{n} w_{\pi(i)} T_{\pi(i)} \tag{5}$$

where $T_{\pi(i)} = \max\{0, S_{\pi(i)} + p_{\pi(i)} - d_{\pi(i)}\}$ is the tardiness (i.e. delay) of a task $\pi(i)$.

In the considered problem, the optimal order (permutation) $\pi^* \in \Pi$ in which tasks should be determined minimizing the total cost (i.e. sum of tardinesses weights).

Formulation for D-Wave quantum annealer

The problems formulated to be solved by the D-Wave machine can be in the form of quadratic programming with constraints (CQM, *Constrained Quadratic Model*), in particular integer linear programming – then it is possible to use the *LeapHybridCQM Sampler* solver for solving them performing hardware quantum annealing.



Formulation for D-Wave quantum annealer

Goal to minimize:

$$\sum_i w_i T_i \tag{6}$$

subject to constrains:

$$S_j - T_j + p_j - d_j \leqslant 0 \quad j = 1, \ldots, n, \tag{7}$$

$$-T_j \leqslant 0 \quad j = 1, \ldots, n, \tag{8}$$

$$S_k - S_j + (p_j - p_k)x_{jk} + 2(S_j - S_k)x_{jk} + p_k \leqslant 0 \quad j < k, \ j, k = 1, \ldots, n, \tag{9}$$

$$-S_j \leqslant -S_0 \quad j = 1, 2, \ldots, n. \tag{10}$$

We introduce integer variables S_i, T_i and binary variables $x_{i,j}$, which equals to 1 if job i precedes job j and 0 otherwise.

Formulation for D-Wave quantum annealer

- Implementation for D-Wave quantum annealer was prepared using `dimod` Python package from D-Wave Ocean Software.
- From the implementation perspective following steps was performed:
    - define CQM model,
    - define CQM variables,
    - add constraints,
    - define objective function,
    - call CQM solver.

Formulation for D-Wave quantum annealer

```python
def define_cqm_model(self):
    """Define CQM model."""

    self.cqm = ConstrainedQuadraticModel()

def define_variables(self):
    """Define CQM variables."""

    self.s = {
        i: Integer(f's{i}', lower_bound=0, upper_bound=sum(self.p))
        for i in range(1, self.n + 1)}

    self.t = {
        i: Integer(f't{i}', lower_bound=0, upper_bound=sum(self.p) + max(self.d))
        for i in range(1, self.n + 1)}

    # Add binary variable which equals to 1 if job i precedes job j
    self.x = {(i, j): Binary(f'x{i}_{j}')
              for i in range(1, self.n + 1)
              for j in range(1, self.n + 1)}
```

Formulation for D-Wave quantum annealer

```
def add_quadratic_overlap_constraint(self):
    for j in range(1, self.n + 1):
        for k in range(1, self.n + 1):
            if j < k:
                self.cqm.add_constraint(
                    self.s[j] - self.s[k] +
                    (self.p[k] - self.p[j]) * self.x[(j, k)]
                    + 2 * self.y[(j, k)] * (self.c[k] - self.s[j]) >=
                    self.p[k],
                    label=f'one_job_{j}_{k}')

def add_tardiness_constraint(self):
    for i in range(1, self.n + 1):
        self.cqm.add_constraint(
            self.t[i] - self.s[i] >= self.p[i] - self.d[i],
            label=f'tardiness_ctr{i}')
```

Formulation for D-Wave quantum annealer

```
def add_tardiness_constraint_zero(self):
    for i in range(1, self.n + 1):
        self.cqm.add_constraint(
            self.t[i] >= 0,
            label=f'tardiness_zero_ctr{i}')

def add_makespan_constraint(self):
    for i in range(1, self.n + 1):
        self.cqm.add_constraint(
            self.cmax - self.s[i] >= self.p[i],
            label=f'makespan_ctr{i}')

def define_objective_function(self):
    """Define objective function"""

    self.cqm.set_objective(
            sum([self.w[i] * self.t[i] for i in range(1, self.n + 1)])
            )
```

Formulation for D-Wave quantum annealer

```
def call_cqm_solver(self, time_limit=5):
    """Calls CQM solver.

    Args:
        time_limit: time limit in second
    """

    sampler = LeapHybridCQMSampler(label="WiTi")

    raw_sampleset = sampler.sample_cqm(self.cqm, time_limit=time_limit)
    feasible_sampleset = raw_sampleset.filter(lambda d: d.is_feasible)
    num_feasible = len(feasible_sampleset)
    if num_feasible > 0:
        best_samples = \
            feasible_sampleset.truncate(min(10, num_feasible))
    else:
        print("Warning: Did not find feasible solution")
        best_samples = raw_sampleset.truncate(10)

    self.best_sample = best_samples.first.sample

    self.solution = {
        i: self.best_sample[self.c[i].variables[0]]
        for i in range(1, self.n + 1)}
    items = list(self.solution.items())
    items.sort(key=lambda x: x[1])
    return [0] + [i[0] for i in items]
```

Formulation for D-Wave quantum annealer

```
def solve(n, p, w, d):

    # Create an empty JSS CQM model.
    model = TWTCQM(n, p, w, d)

    # Define CQM model.
    model.define_cqm_model()

    # Define CQM variables.
    model.define_variables()

    # Add constraint to enforce one job only on a machine.
    model.add_quadratic_overlap_constraint()

    model.add_tardiness_constraint()

    model.add_tardiness_constraint_zero()

    model.add_makespan_constraint()

    # Define objective function.
    model.define_objective_function()

    # Call cqm solver.
    return model.call_cqm_solver()
```

Formulation for D-Wave quantum annealer

```python
if __name__ == "__main__":

    # wt5_042
    n = 5
    p = [0, 37, 20, 4, 59, 95]
    w = [0, 6, 5, 1, 9, 7]
    d = [0, 68, 83, 15, 23, 76]

    solve(n, p, w, d)
```

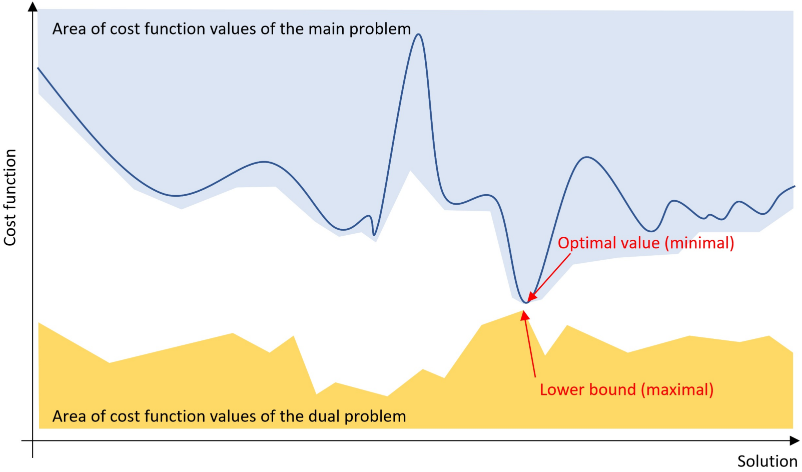
D-Wave results

	t1	t2	t3	t4	t5	x1	x2	x3	x4	...	y4_5	energy	num_oc.	...
0	28.0	33.0	105.0	36.0	139.0	59.0	96.0	116.0	0.0	...	1.0	1735.0	1	...
1	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
2	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
3	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
4	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
5	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
6	28.0	33.0	200.0	36.0	135.0	59.0	96.0	211.0	0.0	...	1.0	1802.0	1	...
7	52.0	0.0	0.0	60.0	139.0	83.0	4.0	0.0	24.0	...	1.0	1825.0	1	...
8	0.0	0.0	0.0	97.0	139.0	4.0	41.0	0.0	61.0	...	1.0	1846.0	1	...
9	0.0	37.0	0.0	77.0	139.0	4.0	100.0	0.0	41.0	...	1.0	1851.0	1	...

Exact solution method idea

- An exact hybrid algorithm for solving considered problem, the construction of which is based on the (*Branch and Bound* method, (B&B), is proposed here.
- For determining the upper bound on a D-Wave machine, a quadratic programming problem with constraints is formulated, which is a natural way of formulating computational tasks for this machine.
- In contrast, for determining the lower bound, the Lagrange relaxation method has been used. Approximation of the value of the extremum of the Lagrange function has been calculated on the D-Wave quantum computer.

Exact solution method idea



Rysunek: Idea of the quantum lower bound calculation.

Total Weighted Tardiness Problem (TWTP)

The considered TWTP problem can be written in the form of an optimization task:

$$\min_{S} \sum_{i=1}^{n} w_i T_i \tag{11}$$

s.t.

$$S_i + p_i - S_j \leqslant K(1 - y_{ij}),\ j = i+1, \ldots, n,\ \ i = 1, \ldots, n, \tag{12}$$

$$S_j + p_j - S_i \leqslant K y_{ij},\ j = i+1, \ldots, n,\ i = 1, \ldots, n, \tag{13}$$

$$y_{ij} \in \{0, 1\},\ j = i+1, \ldots, n,\ i = 1, \ldots, n, \tag{14}$$

$$S_i \geqslant 0,\ i = 1, \ldots, n, \tag{15}$$

where K is a sufficiently large number – for instance, $K = \sum_{i=1}^{n} p_i$. In turn, y_{ij} is a binary variable equal to 1 if i task precedes j and 0 otherwise.

Lagrange relaxation of the Lower Bound

Lagrange function with multipliers u_{ij} and v_{ij}, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, for the considered (main) goal function of our problem, assumes for the vector $S = (S_1, S_2, \ldots, S_n)$ and the matrix $y = [y_{ij}]_{n \times n}$ the following form:

$$L(S, y, u, v) = \sum_{i=1}^{n} w_i T_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} u_{ij}(S_i + p_i - S_j - K(1 - y_{ij})) +$$

$$+ \sum_{i=1}^{n} \sum_{j=i+1}^{n} v_{ij}(S_j + p_j - S_i - K y_{ij}).$$

Lagrange relaxation of the Lower Bound

By transforming this expression we obtain

$$L(S, y, u, v) = \sum_{i=1}^{n} L_i(S_i, u, v) + K \sum_{i=1}^{n} \sum_{j=i+1}^{n} Q_{ij}(y_{ij}, u, v) + V(u, v), \tag{16}$$

where

$$L_i(S_i, u, v) = w_i T_i + \alpha_i S_i,$$

$$\alpha_i = \sum_{j=i+1}^{n} (u_{ij} - v_{ij}) + \sum_{j=1}^{i-1} (v_{ji} - u_{ji}), \quad Q_{ij}(y_{ij}, u, v) = (u_{ij} - v_{ij}) y_{ij},$$

$$V(u, v) = \sum_{i=1}^{n} p_i \left(\sum_{j=1}^{i-1} v_{ji} + \sum_{j=i+1}^{n} u_{ij} \right) - K \sum_{i=1}^{n} \sum_{j=i+1}^{n} u_{ij}.$$

Lagrange relaxation of the Lower Bound

Note that if S^* is the optimal solution to the problem under consideration, then for *any non-negative* $u, v \geqslant 0$ there is

$$\sum_{j=1}^{n} w_j T_j \geqslant \sum_{j=1}^{n} w_j T_j + \sum_{i=1}^{n} \sum_{j=i+1}^{n} u_{ij}(S_i^* + p_i - S_j^* - K(1 - y_{ij})) +$$

$$+ \sum_{i=1}^{n} \sum_{j=i+1}^{n} v_{ij}(S_i^* + p_i - S_j^* - K y_{ij}) \geqslant \min_S \min_y L(S, y, u, v)$$

since S^* defines an feasible solution to a problem (11), for which the tasks are disjoint. Thus, $\min_S \min_y L(S, y, u, v)$ is a lower bound on the value of the objective function of the original problem.

Lagrange relaxation of the Lower Bound

Thus, while looking for a good lower bound, we need to determine the

$$LB = \max_{u,v} \min_{S,y} L(S,y,u,v) = \max_{u,v} \Bigg(\sum_{i=1}^{n} \min_{0 \leqslant S_i \leqslant \mathcal{T} - p_i} L_i(S_i,u,v) + $$

$$+K \sum_{i=1}^{n} \sum_{j=i+1}^{n} \min_{y} Q_{ij}(y_{ij},u,v) + V(u,v) \Bigg) \tag{17}$$

whereby maximization towards u and v can be approximate, while towards S and y must be exact.

This task boils down to finding the minimax point on the saddle surface.

Lagrange relaxation of the Lower Bound D-Wave formulation

$$LB = -\min_{u,v,S,y} \left[- \left(\sum_{i=1}^{n} L_i(S_i, u, v) + K \sum_{i=1}^{n} \sum_{j=i+1}^{n} Q_{ij}(y_{ij}, u, v) + V(u, v) \right) \right] \tag{18}$$

with constraints (s.t.):

$$L_i(S_i, u, v) \leqslant L_i(0, u, v), \;\; i = 1, 2, \ldots, n, \tag{19}$$

$$L_i(S_i, u, v) \leqslant L_i(1, u, v), \;\; i = 1, 2, \ldots, n, \tag{20}$$

$$\vdots$$

$$L_i(S_i, u, v) \leqslant L_i(T - p_i, u, v), \;\; i = 1, 2, \ldots, n, \tag{21}$$

Lagrange relaxation of the Lower Bound D-Wave formulation cont.

and

$$Q_{ij}(y_{ij}, u, v) \leqslant Q_{ij}(0, u, v), \;\; i, j = 1, 2, \ldots, n, \tag{22}$$

$$Q_{ij}(y_{ij}, u, v) \leqslant Q_{ij}(1, u, v), \;\; i, j = 1, 2, \ldots, n. \tag{23}$$

Lagrange relaxation of the Lower Bound D-Wave formulation cont.

From definition a tardiness $T_i = \max\{0, S_i + p_i - d_i\}$, but we must not use maximum as a constraint on D-Wave quantum annealer, so:

Algorithm 1: Adding S minimalization constraints to QUBO model

---

for $i = 1, 2, \ldots, n$ **do**
    for $t = 0, 1, 2, \ldots, T - p_i$ **do**
        if $(t + p_i - d_i > 0)$ **then**
            Add constraint $L_i(S_i, u, v) \leqslant w_i \cdot (t + p_i - d_i) + \alpha_j \cdot t$
        else
            Add constraint $L_i(S_i, u, v) \leqslant w_i \cdot 0 + \alpha_i \cdot t$

Hybrid QPU-CPU Quantum Annealing-driven B&B

Algorithm 2: QAB&B

Input : permutation π_0 – initial solution;
Output: permutation π^* – optimal solution;

Heap : priority queue of tree vertices sorted in ascending order by their upper bounds $LocalUB$;
Put($Heap$,(π_0, n, 0,$F(\pi_0)$)); ($F(\pi_0)$ *value as upper bound,* n *is a number of free tasks,* 0 *is a lower bound of the solution* π_0)
$\pi^* \leftarrow \pi_0$
while $Heap \neq \emptyset$ **do**
    Get($Heap$,(π, t, LB_π, UB_π));
    if $LB_\pi < F(\pi^*)$ **then**
        Determine a set of candidates K_π – tasks, which can be fixed on a t-th position;
        for $\beta \in K_\pi$ **do**
            Swap($\pi, \pi^{-1}(\beta), t$) (*swap tasks on positions* $\pi^{-1}(\beta)$ *and* t *in* π);
            $LocalLB \leftarrow QuantumLagrangeLB(\pi)$;
            if $LocalLB < F(\pi^*)$ **then**
                $\pi_{LocalUB} \leftarrow \arg(QuantumAnnealingUB(\pi))$;
                if $F(\pi_{LocalUB}) < F(\pi^*)$ **then**
                    $\pi^* \leftarrow \pi_{LocalUB}$;
                Put($Heap$,(π,$t-1$,$LocalLB$, $F(\pi_{LocalUB})$));
            Swap($\pi, t, \pi^{-1}(\beta)$) (*zamień z powrotem*);

Quantum computational experiments

Computer experiments have been conducted in D-Wave Leap environment on `hybrid_constrained_quadratic_model_version1p` solver executed on a North America quantum annealer.

Case Study. An instance of $n = 7$ has been generated for an experiment for checking usefulness of the proposed methodology.

i	1	2	3	4	5	6	7
p_i	3	82	26	5	3	81	81
d_i	0	24	133	47	120	87	119
w_i	1	10	2	1	9	10	9

Tabela: An instance `wt7_070` of $n = 7$ size.

Remark: for $n = 7$ the QUBO model generates $1.13 \cdot 10^{49}$ solutions, however 7!=5040 only.

iter.	π	$\pi_{LocalUB}$	$LocalUB^*$	$LocalLB^*$	h
1	[7, 2, 3, 4, 5, 6, 1]	[4, 5, 6, 2, 3, 7, 1]	3330	842.38	1
2	[1, 7, 3, 4, 5, 6, 2]	[1, 4, 6, 5, 3, 7, 2]	3313	2550.31	1
3	[1, 2, 7, 4, 5, 6, 3]	[1, 2, 5, 6, 4, 7, 3]	3080	866.95	1
4	[1, 2, 3, 7, 5, 6, 4]	[1, 2, 5, 7, 3, 6, 4]	3311	797.68	1
5	[1, 2, 3, 4, 7, 6, 5]	[1, 4, 6, 2, 7, 3, 5]	4429	2032.25	1
6	[1, 2, 3, 4, 5, 7, 6]	[4, 2, 5, 7, 3, 1, 6]	3366	2516.18	1
7	[1, 2, 3, 4, 5, 6, 7]	[1, 6, 5, 2, 3, 4, 7]	3188	2029.32	1
8	[6, 2, 7, 4, 5, 1, 3]	[5, 4, 6, 2, 7, 1, 3]	3238	1087.37	2
9	[1, 6, 7, 4, 5, 2, 3]	[4, 1, 6, 5, 7, 2, 3]	3120	2602.27	2
10	[1, 2, 6, 4, 5, 7, 3]	[1, 4, 2, 5, 6, 7, 3]	3053	2110.99	2
11	[1, 2, 7, 6, 5, 4, 3]	[1, 6, 5, 2, 7, 4, 3]	3136	1093.57	2
12	[1, 2, 7, 4, 6, 5, 3]	[4, 6, 1, 2, 7, 5, 3]	4267	2092.99	2
13	[1, 2, 7, 4, 5, 6, 3]	[5, 1, 2, 7, 4, 6, 3]	3199	2562.63	2
14	[5, 2, 6, 4, 1, 7, 3]	[2, 4, 5, 6, 1, 7, 3]	3154	2275.44	3
15	[1, 5, 6, 4, 2, 7, 3]	[1, 4, 6, 5, 2, 7, 3]	**3043**	3019.60	3

* results of quantum annealing on D-Wave machine in time 8ms

Summary for $1||\sum w_i T_i$

- The case considers the NP-hard single machine tasks scheduling problem with the criterion of minimizing the weighted sum of tardiness.
- An exact quantum annealing-driven branch and bound QAB&B algorithm has been proposed. Currently, the possibilities of quantum annealers (they are only produced by D-Wave company) allow us for optimal solving instances of the size $n \leqslant 10$ in time of minutes.
- The proposed methodology allows for optimal solving of similar problems (eg. TSP), waiting for the increase of computational possibilities of quantum computers.

Single machine weighted number of tardy jobs scheduling

Let $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(i)}$ be the moment of completion of the task $\pi(i)$ (in the optimal solution the schedule is shifted to the left). We define the objective function

$$\mathcal{F}(\pi) = \sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)} \tag{24}$$

Where

$$U_{\pi(i)} = \begin{cases} 0 & \text{if } C_{\pi(i)} \leqslant d_{\pi(i)}, \\ 1 & \text{if } C_{\pi(i)} > d_{\pi(i)}, \end{cases}$$

is the unit delay (binary tardiness) of task $\pi(i)$.

The problem under consideration is to determine the optimal permutation $\pi^* \in \Pi$ minimizing the cost of $\mathcal{F}(\pi^*)$. In Graham's notation, it is denoted by $1||\sum w_i U_i$ and is NP-hard.

Transformation to $1||max \sum w_i(1-U_i)$

A problem $1||min \sum w_i U_i$ can be formulated equivalently as the problem of maximizing the weighted number of tasks executed on time $1||max \sum w_i(1-U_i)$:

$$\max \sum_{i=1}^{t} w_i x_i \tag{25}$$

with constraints:

$$\begin{array}{lllllllll} p_1x_1 & & & & & & & \leqslant & d_1, \\ p_1x_1 & + & p_2x_2 & & & & & \leqslant & d_2, \\ p_1x_1 & + & p_2x_2 & + & p_3x_3 & & & \leqslant & d_3, \\ & & & & \vdots & & & & \\ p_1x_1 & + & p_2x_2 & + & \ldots & + & p_tx_t & \leqslant & d_t, \end{array} \tag{26}$$

$$x_i \in \{0,1\},\ i = 1,2,\ldots,t.$$

D-Wave quantum machine formulation

The Lagrange function with the real multipliers $u = (u_1, u_2, \ldots, u_t)$ takes the following form for binary $x = (x_1, x_2, \ldots, x_t)$:

$$L(x,u) = \sum_{i=1}^{t} w_i x_i + u_1(p_1 x_1 - d_1) + u_2(p_1 x_1 + p_2 x_2 - d_2) +$$

$$\ldots + u_t(p_1 x_1 + p_2 x_2 + \ldots + p_t x_t - d_t) =$$

$$= x_1(w_1 + u_1 p_1 + u_2 p_1 + \ldots + u_t p_1) + x_2(w_2 + u_2 p_2 + u_3 p_2 + \ldots + u_t p_2) +$$

$$\ldots + x_t(w_t + u_t p_t) - \sum_{i=1}^{t} u_i d_i = \sum_{i=1}^{t} x_i \left(w_i + p_i \sum_{j=i}^{t} u_j \right) - \sum_{i=1}^{t} u_i d_i.$$

D-Wave quantum machine formulation

Let

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

therefore

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

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

D-Wave quantum machine formulation

Note that for *any* $u_i \leqslant 0$, $i = 1, 2, \ldots, t$ and the optimal x^* being a solution to the objective function problem (25) the inequality holds

$$\sum_{i=1}^{t} w_i x_i^* \leqslant \min_u L(x^*, u) \leqslant \min_u \max_x L(x^*, u) =$$

$$= \min_u \left(\sum_{i=1}^{t} \max_x L_i(x_i, u) \right) - \sum_{i=1}^{t} u_i d_i \stackrel{\text{def}}{=} UB_{on-time}. \qquad (27)$$

D-Wave quantum machine formulation

To calculate the upper bound of the Lagrange function $UB_{on-time}$ on a D-Wave computer, we find the values of the u vector using quantum annealing by solving the following CQM problem:

$$UB_{on-time} \stackrel{\text{def}}{=} \min_{u,x} \left(\sum_{i=1}^{t} L_i(x_i, u) \right) - \sum_{i=1}^{t} u_i d_i, \tag{28}$$

with constraints

$$L_i(x_i, u) \geqslant L_i(0, u),$$

and

$$L_i(x_i, u) \geqslant L_i(1, u),$$

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

Computational experiments on a quantum machine

The calculations were performed in the D-Wave Leap environment using the `hybrid_constrained_quadratic_model_version1p` solver and run on a machine installed in North America.

The calculation time for the considered example with $n = 10$ on CPU+QPU was 50s, including the QPU time of 0.15s

Exact algorithm driven by quantum annealing
Case study

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

Tabela: Test instance `wt10_011` of problem size $n = 10$.

One of the optimal solutions:

- $\pi^* = (7, 1, 9, 4, 6, 8, 3, 5, 10, 2)$,
- $F(\pi^*) = 15$.

Exact algorithm driven by quantum annealing
Case study

poziom	t	$LocalLB$	$LocalUB$	$F(\pi)$	$F(\pi^*)$
1	1	17,55	18	29	18
1	2	14,62	15	30	15
1	3	14,41	26	26	15
1	4	17,08	15	33	15
1	5	14,41	15	26	15
1	6	16,07	15	26	15
1	7	17,89	15	26	15
1	8	14,41	15	26	15
1	9	15,56	15	26	15

Tabela: QAdB&B algorithm work

Computational experiments in D-Wave Leap environment

Tabela: Results of computational experiments

	QAdB&B				SMB&B		
instance	LB_π	UB_π	QPU [ms]	gap	LB	UB	CPU [ms]
wt40_011	32.07 (33)	**34**	15.29	2.94	33.49 (34)	137	366
wt40_012	35.71 (36)	**39**	15.34	7.69	36.83 (37)	114	328
wt40_013	46.90 (47)	**50**	15.35	6.00	47.54 (48)	166	331
wt40_014	34.17 (35)	**36**	15.28	2.78	34.41 (35)	130	324
wt40_015	48.72 (49)	**51**	15.33	3.92	47.49 (48)	149	324
wt40_016	99.93 (100)	**105**	15.32	4.76	99.24 (100)	167	324
wt40_017	100.68 (101)	**103**	15.34	1.94	100.09 (101)	168	324
wt40_018	101.69 (102)	**103**	15.34	0.97	102.18 (103)	174	329
wt40_019	96.67 (97)	**99**	15.33	2.02	97.12 (98)	170	326
wt40_020	87.08 (88)	**89**	15.33	1.12	86.90 (87)	201	326
average			15.32	3.41			330

Results of QAdB&B have been compared to classic B&B run on silicon-based CPU (SMB&B) with LB obtained by Powell continouous optimization of u in Lagrange function.

Computational experiments in D-Wave Leap environment

Tabela: Results of computational experiments

	QAdB&B				SMB&B		
instance	LB_π	UB_π	QPU [ms]	gap	LB	UB	CPU [ms]
wt50_011	55.46 (56)	**59**	15.33	5.08	56.97 (57)	160	593
wt50_012	43.01 (44)	**46**	15.20	4.35	44.97 (45)	177	593
wt50_013	62.73 (63)	**65**	15.33	3.08	60.48 (61)	181	583
wt50_014	70.20 (71)	**75**	15.34	5.33	72.13 (73)	174	619
wt50_015	54.29 (55)	**57**	15.27	3.51	56.63 (57)	132	599
wt50_016	107.21 (108)	**110**	15.33	1.82	107.21 (108)	249	592
wt50_017	88.03 (89)	**91**	15.33	2.20	87.54 (88)	221	598
wt50_018	107.36 (108)	**109**	15.33	0.92	106.90 (107)	247	584
wt50_019	110.25 (111)	**112**	15.35	0.89	111.13 (112)	228	589
wt50_020	89.93 (90)	**93**	15.34	3.23	90.39 (91)	185	758
average			15.31	3.04			610

Computational experiments in D-Wave Leap environment

Tabela: Results of computational experiments

	QAdB&B				SMB&B		
instance	LB_π	UB_π	QPU [ms]	gap	LB	UB	CPU [ms]
wt100_011	119.14 (120)	**127**	15.29	5.51	121.81 (122)	186	4193
wt100_012	145.48 (146)	**154**	15.34	5.19	150.42 (151)	189	4189
wt100_013	115.46 (116)	**125**	15.35	7.20	122.86 (123)	171	4098
wt100_014	106.62 (107)	**116**	15.35	7.76	112.14 (113)	148	4165
wt100_015	124.57 (125)	**134**	15.30	6.72	128.54 (129)	188	4209
wt100_016	233.25 (234)	**237**	15.36	1.27	234.23 (235)	317	4169
wt100_017	191.64 (192)	**195**	15.36	1.54	189.72 (190)	260	4367
wt100_018	273.63 (274)	**278**	15.33	1.44	269.09 (270)	330	4119
wt100_019	245.48 (246)	**249**	15.28	1.20	246.65 (247)	348	4157
wt100_020	224.57 (225)	**227**	15.36	0.88	219.14 (220)	303	4298
average			15.33	3.87			4196

Summary for $1||\sum w_i U_i$

- We considered the NP-hard problem of scheduling tasks on one machine with the criterion of minimizing the weighted number of tardy jobs.
- We are able to compute optimal (exact) solutions for $n \leqslant 100$.
- Currently, quantum annealing capabilities allow for optimal resolution of $n \leqslant 1000$ instances in a few seconds, but with no guarantee of optimality.
- The implementation of the algorithm was implemented in Python and tested in the D-Wave Leap environment as a hybrid method run alternately on the CPU and QPU.

Quantum Optimization Team

- Prof. Wojciech Bożejko 
- Prof. Mieczysław Wodecki 
- Prof. Czesław Smutnicki 
- Ph. D.Sc. Jarosław Pempera, Assoc. Prof. 
- Ph. D.Sc. Mariusz Uchroński, Assoc. Prof.