



# Foundations for Workflow Application Scheduling on D-Wave System



Dawid Tomasiewicz  
Maciej Pawlik  
Maciej Malawski  
Katarzyna Rycerz

## About me



# Goals of this work

Checking the usability of D-Wave 2000Q quantum annealer for solving problems related to the paradigm of workflow, which includes:

- choice of the problem type from workflow problems,
- QUBO formulation of such problem
- running on the D-Wave 2000Q
- discussion on the usability and scalability of the proposed solution.

- 1. Workflow problems**
- 2. D-Wave quantum annealer**
- 3. Jobshop problem on quantum annealer**
- 4. Running on D-Wave - QUBO formulation**
- 5. Running on D-Wave - Minor embedding**
- 6. Results**
- 7. Conclusions and possible improvements**



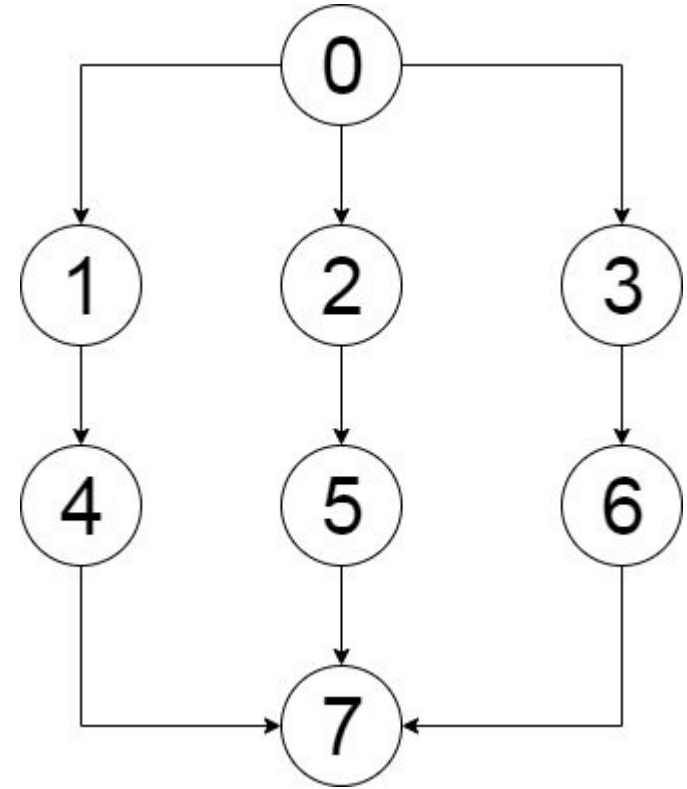
1. **Workflow problems**
2. D-Wave quantum annealer
3. Jobshop problem on quantum annealer
4. Running on D-Wave - QUBO formulation
5. Running on D-Wave - Minor embedding
6. Results
7. Conclusions and possible improvements

# Workflows

- A paradigm commonly used for describing complex scientific processes and applications.
- Usually represented as DAGs (Directed Acyclic Graphs)
- Vertices represent tasks
- Edges represent dependencies or data transfers between tasks

## Workflow scheduling

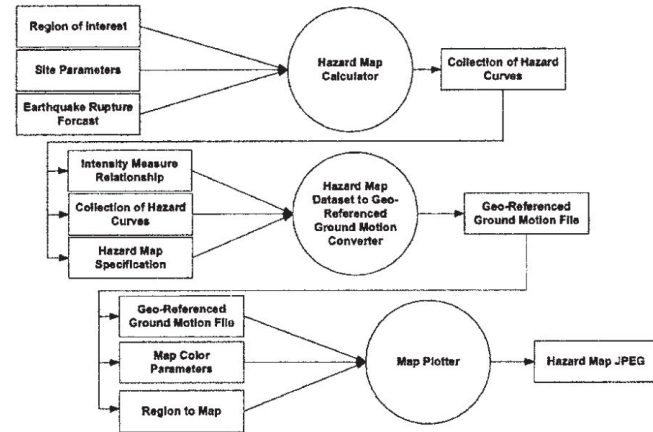
- Planning of execution with respect to given parameters such as deadline, budget and computing resources.
- An NP-hard problem - strong limitations on the size of practically solvable problems



An example DAG representing a workflow

# Workflows - example applications

- **Montage** - image mosaic software used to construct human perceptible images of sky features from multiple images captured by telescopes [1]
- Software calculating seismic hazard maps based on the data provided in GUI [2]



An example workflow developed in the work [2]

[1] Berriman, G., Good, J., Laity, A., et al.: Montage: A grid enabled image mosaic service for the national virtual observatory. In: Astronomical Data Analysis Software and Systems (ADASS) XIII. vol. 314, p. 593 (2004)

[2] Maechling, P., Chalupsky, H., Dougherty, M., et al.: Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment. ACM SIGMOD Record 34 (3), 24–30 (2005)

1. Workflow problems
- 2. D-Wave quantum annealer**
3. Jobshop problem on quantum annealer
4. Running on D-Wave - QUBO formulation
5. Running on D-Wave - Minor embedding
6. Results
7. Conclusions and possible improvements

# D-Wave quantum annealer

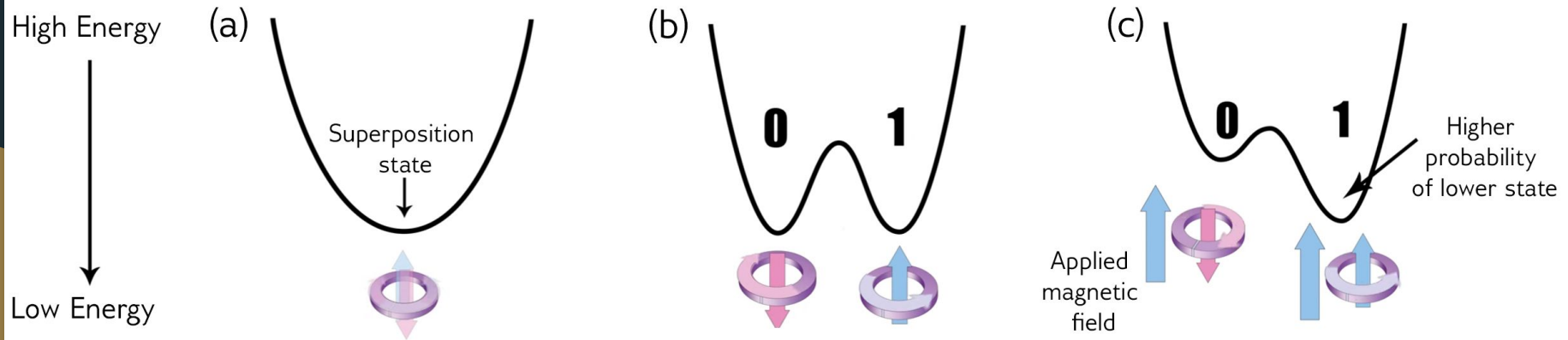
## Quantum annealing

- a metaheuristic for finding the global minimum of a given objective function
- searches a set of candidate solutions (candidate states)
- uses quantum fluctuations

Source: [https://en.wikipedia.org/wiki/Quantum\\_annealing](https://en.wikipedia.org/wiki/Quantum_annealing)

# D-Wave quantum annealer

Qubits in D-Wave 2000Q are superconducting loops. The qubit can have be in an undefined state (superposition), in which it's state is unknown (a), in one of two states with known energies (b), which can be marked as "0" and "1". The "free" qubit in superposition state has equal 50% probability of falling into each state. This probability can be changed by applying an external magnetic field.



**Bias** - a programmable quantity that controls the external magnetic field applied to each qubit

# D-Wave quantum annealer

Qubits can interact with each other:

- **Coupler** - a device that can link qubits together so they influence each other
- **Coupling strength** - a programmable value defining the correlation weights between qubits
- The bigger the coupling strength is, the more two qubits are likely to end up in the same state

Two programmable quantities that allow to control and program the quantum annealer:

- bias values
- coupling strengths

Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_2.html](https://docs.dwavesys.com/docs/latest/c_gs_2.html)

# D-Wave quantum annealer

From the physical point of view the operation done by the quantum annealer is finding eigenstate with the lowest eigenenergy of the Ising model Hamiltonian, which for the D-Wave Systems, takes the following form:

$$H = -\frac{A(s)}{2} \left( \sum_i \hat{\sigma}_x^{(i)} \right) + \frac{B(s)}{2} \left( \sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \right)$$

Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_2.html](https://docs.dwavesys.com/docs/latest/c_gs_2.html)



# D-Wave quantum annealer

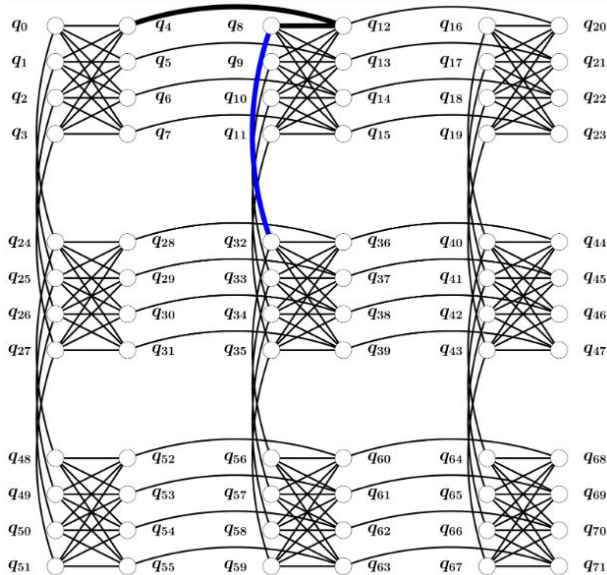
From the programmer's point of view quantum annealer searches for the minimum of the following function (QUBO).

$$f(x) = \sum_i Q_{i,i}x_i + \sum_{i<j} Q_{i,j}x_ix_j \quad x_i \in \{0, 1\} \quad \min_{x \in \{0,1\}^n} x^T Q x$$

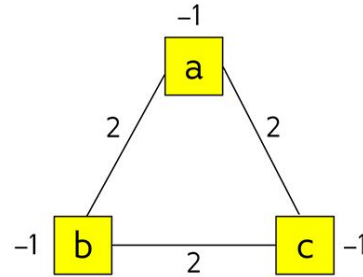
Programming the D-Wave quantum annealer requires providing one input value: **matrix Q**

Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_2.html](https://docs.dwavesys.com/docs/latest/c_gs_2.html)

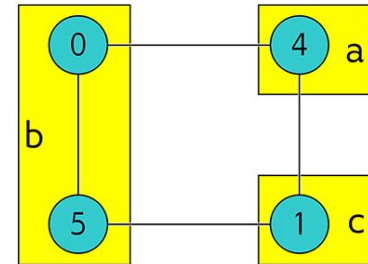
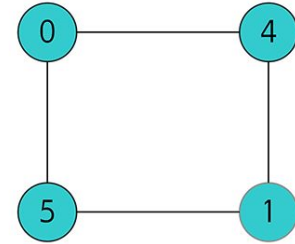
# Minor embedding



A part of Chimera graph, based on which the D-Wave 2000Q is built



An example of a very simple embedding of a triangle into a square



- Quantum computer architecture graph is not fully connected, but the problem solved must match it's architecture.
- The problem must be transformed to the form matching the architecture by
  - finding it's minor in this graph
  - (if the minor doesn't exist) representing one variable of a problem with many qubits using chains

Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_7.html](https://docs.dwavesys.com/docs/latest/c_gs_7.html)

# D-Wave quantum annealer

D-Wave 2000Q quantum annealer problem solving flow:

1. Formulating the problem as QUBO
2. Matching the QUBO with the annealer architecture
3. Obtaining the results from the machine
4. Unembedding the results into the initial QUBO
5. Reversing the QUBO into the actual problem

Note: matching the QUBO with the annealer's architecture (and unembedding) are theoretically optional, but practically it is almost impossible for QUBOs of problem to match the annealer's architecture without embedding.

1. Workflow problems
2. D-Wave quantum annealer
- 3. Jobshop problem on quantum annealer**
4. Running on D-Wave - QUBO formulation
5. Running on D-Wave - Minor embedding
6. Results
7. Conclusions and possible improvements

# Job Shop Scheduling Solver based on Quantum Annealing

Davide Venturelli<sup>1,2</sup>, Dominic J.J. Marchand<sup>3</sup>, Galo Rojo<sup>3</sup>

<sup>1</sup>*Quantum Artificial Intelligence Laboratory (QuAIL), NASA Ames*

<sup>2</sup>*U.S.R.A. Research Institute for Advanced Computer Science (RIACS)*

<sup>3</sup>*1QB Information Technologies (1QBit)*

Quantum annealing is emerging as a promising near-term quantum computing approach to solving combinatorial optimization problems. A solver for the job-shop scheduling problem that makes use of a quantum annealer is presented in detail. Inspired by methods used for constraint satisfaction problem (CSP) formulations, we first define the makespan-minimization problem as a series of decision instances before casting each instance into a time-indexed quadratic unconstrained binary optimization. Several pre-processing and graph-embedding strategies are employed to compile optimally parametrized families of problems for scheduling instances on the D-Wave Systems' Vesuvius quantum annealer (D-Wave Two). Problem simplifications and partitioning algorithms, including variable pruning, are discussed and the results from the processor are compared against classical global-optimum solvers.

# Job shop problem on quantum annealer

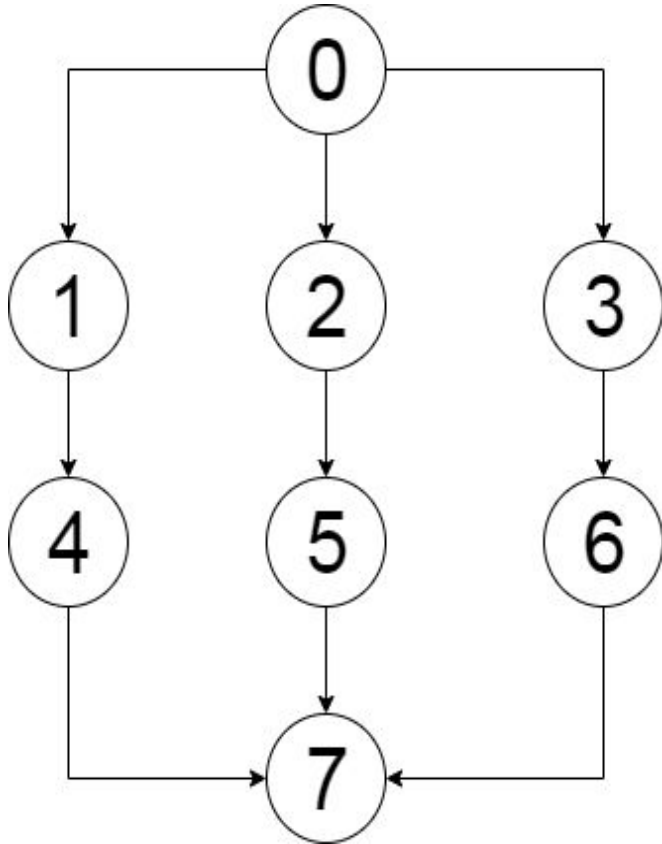
- Machines, operations, jobs
- Jobs include one or more operations
- Machines run operations, which have execution time specified

## **Constraints**

- Each task starts only once
- One machine can run one task at the single point of time
- Operations order among each job must be kept

1. Workflow problems
2. D-Wave quantum annealer
3. Jobshop problem on quantum annealer
- 4. Running on D-Wave - QUBO formulation**
5. Running on D-Wave - Minor embedding
6. Results
7. Conclusions and possible improvements

# Workflow scheduling problem (WSP) formulation example



P - Vector representing graph of paths

$$P = [[0, 1, 4, 7], [0, 2, 5, 7], [0, 3, 6, 7]]$$

T - matrix with execution times of each task (columns) on each machine (rows)

$$T = \begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 6 & 12 & 24 \\ 4 & 2 & 14 & 6 & 10 & 2 & 4 & 8 \\ 8 & 4 & 28 & 12 & 20 & 4 & 8 & 16 \end{bmatrix}$$

K - vector of costs of using machines per time unit

$$K = [10 \quad 18 \quad 6]$$

D - deadline for all tasks execution

$$D = 45$$



# QUBO formulation for WSP

The QUBO formulation of WSP must consider two factors:

1. total cost of executing workflow
2. constraints, which are:
  - Deadline
  - Machines assignments - every task has exactly one machine assigned

We assume infinite number of **instances** of machines and the strictly limited (as an input parameter) number of **types** of machines.

# QUBO formulation for Workflow Scheduling Problem (WSP)

Formulating the WSP as D-Wave runnable QUBO includes the three following steps:

- 1) WSP -> Binary Integer Linear Programming (BILP)
- 2) BILP -> QUBO
- 3) QUBO -> minor embedded QUBO

# Binary Integer Linear Programming

Minimize the objective function  
(represented by vector  $c$ ):

$$\min c^t X$$

The solutions must meet the  
constraints represented by matrix  $A$   
and vector  $b$ .

$$AX = b$$

$$\forall x \in X \quad x \in \{0, 1\}$$

# BILP for WSP - variables numbering

Binary variables are numbered with the following manner (table to the right):

- The matrices are row by row vectorized.
- (e.g.) If  $x_{17}=1$  then the task no. 1 is run on the machine no. 2,
- if  $x_{17}=0$  it is not.

		Task no.							
		0	1	2	3	4	5	6	7
Machine no	0	0	1	2	3	4	5	6	7
	1	8	9	10	11	12	13	14	15
	2	16	17	18	19	20	21	22	23

## BILP for WSP - objective function

$$K = [k_i]_m$$

$$T = [\tau_{i,j}]_{t \times m}$$

$$\gamma_{i,j} = \tau_{i,j} \cdot k_j$$

Machine number		Task number	0	1	2	3	4	5	6	7
0	10		12	6	42	18	30	6	12	24
1	18		4	2	14	6	10	2	4	8
2	6		8	4	28	12	20	4	8	16

Input data: costs vector and times matrix

0	1	2	3	4	5	6	7
120	60	420	180	300	60	120	240
72	36	252	108	180	36	72	144
48	24	168	72	120	24	48	96

Costs matrix created from input data

$$\min c^t X$$



$$\min \sum_{i=0}^{|X|-1} x_i c_i$$

The matrix of costs is then vectorized row by row resulting in the following vector

$$c = [120, 60, 420, 180, 300, 60, 120, 240, 72, 36, 252, 108, 180, 36, 72, 144, 48, 24, 168, 72, 120, 24, 48, 96]$$

The results of optimization take form  $X = [x_0, x_1, x_2, \dots, x_n]$  for example  $X = [1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]$   
(variables indexing described in the previous slide)

For the example results vector and costs matrix the costs function can be calculated in the following manner:

$$C(X) = 1 \cdot 120 + 0 \cdot 60 + 0 \cdot 420 + 1 \cdot 180 + \dots + 0 \cdot 48 + 1 \cdot 96$$

Such formulation of vector  $c$  from input vector of cost and times matrix is the exact objective function part from BILP formulation, therefore this part of the BILP formulation has been reached. The next part includes transformation of constraints.

# BILP for WSP - machines assignment constraint

$$AX = b \iff \forall j \in [0, n-1] \left( \sum_{i=0}^{|X|-1} x_i A_{ji} \right) = b_j$$

BILP constraints formulation  $AX=b$  can be equivalently written with the use of the sum operator

One task can be assigned to one machine only. As the variables representing the fact of running tasks on machines are binary, such constraint can be expressed for each task as sum of all variables representing the particular task being equal to 1, as in the examples below.

$$x_0 + x_8 + x_{16} = 1$$

$$x_1 + x_9 + x_{17} = 1$$

etc. for all tasks

		Task no.									
		0	1	2	3	4	5	6	7		
Machine no	0	0	1	2	3	4	5	6	7		
	1	8	9	10	11	12	13	14	15		
	2	16	17	18	19	20	21	22	23		

Reminder: binary variables' global indices

A																										b
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23			
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

A fragment of matrix A and vector b representing the machines assignment constraint

# BILP for - machines assignment constraint

- QUBO is an unconstrained model - all the constraints it groups must be weighed even if they are mathematically correct
- A new parameter **S** must be introduced - machines assignment constraint strength relative to the deadline constraint, with which the constraint takes the following

$$Sx_0 + Sx_8 + Sx_{16} = S$$

$$Sx_1 + Sx_9 + Sx_{17} = S$$

The part of matrix representing this constraint has the following values

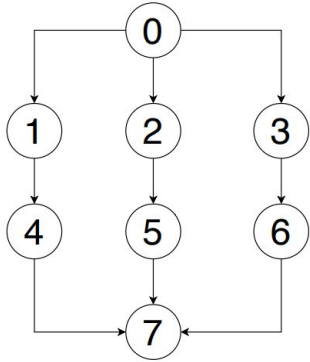
The part of matrix representing this constraint has the following values																									
A																									b
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23		
S	0	0	0	0	0	0	0	S	0	0	0	0	0	0	0	S	0	0	0	0	0	0	0		S
0	S	0	0	0	0	0	0	0	S	0	0	0	0	0	0	0	S	0	0	0	0	0	0		S

A part of matrix A and vector b for machines assignment constraint.

# BILP for WSP - deadline constraint

$$AX \stackrel{\circ}{=} b \iff \forall j \in [0, n-1] \quad (\sum_{i=0}^{|X|-1} x_i A_{ji}) = b_j$$

Note:  $\Psi_i$  is the actual execution time of task (after choice of machine),  $t_i$  is the execution time of task represented by variable "i" if the task is executed on the machine represented by this variable



$$\Psi_0 + \Psi_1 + \Psi_4 + \Psi_7 \leq D$$

sum of execution times for each path not greater then the deadline

$$\Psi_0 + \Psi_2 + \Psi_5 + \Psi_7 \leq D$$

$$\Psi_0 + \Psi_3 + \Psi_6 + \Psi_7 \leq D$$

For binary variables it means that sum of products  $x_i t_i$  must be less than or equal to deadline, which is presented in equation below for path [0,1,4,7] and in matrix for all 3 paths. Note that  $AX=b$  is an equation and below there is an inequality - it leads to slack variables (next slide)

$$x_0 t_0 + x_1 t_1 + x_4 t_4 + x_7 t_7 + x_8 t_9 + x_9 t_9 + x_{12} t_{12} + x_{15} t_{15} + x_{16} t_{16} + x_{17} t_{17} + x_{20} t_{20} + x_{23} t_{23} \stackrel{\circ}{\leq} 45$$

A																										b
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23			
t <sub>0</sub>	t <sub>1</sub>	0	0	t <sub>4</sub>	0	0	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>	0	0	t <sub>12</sub>	0	0	t <sub>15</sub>	t <sub>16</sub>	t <sub>17</sub>	0	0	t <sub>20</sub>	0	0	t <sub>23</sub>		45	
t <sub>0</sub>	0	t <sub>2</sub>	0	0	t <sub>5</sub>	0	t <sub>7</sub>	t <sub>8</sub>	0	t <sub>10</sub>	0	0	t <sub>13</sub>	0	t <sub>15</sub>	t <sub>16</sub>	0	t <sub>18</sub>	0	0	t <sub>21</sub>	0	t <sub>23</sub>		45	
t <sub>0</sub>	0	0	t <sub>3</sub>	0	0	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	0	0	t <sub>11</sub>	0	0	t <sub>14</sub>	t <sub>15</sub>	t <sub>16</sub>	0	0	t <sub>19</sub>	0	0	t <sub>22</sub>	t <sub>23</sub>		45	

A part of matrix A and vector b for the deadline constraint.



# BILP for WSP - slack variables

$$AX \stackrel{=}{=} b \iff \forall j \in [0, n-1] \left( \sum_{i=0}^{|X|-1} x_i A_{ji} \right) = b_j$$
$$x_0 t_0 + x_1 t_1 + x_4 t_4 + x_7 t_7 + x_8 t_9 + x_9 t_9 + x_{12} t_{12} + x_{15} t_{15} + x_{16} t_{16} + x_{17} t_{17} + x_{20} t_{20} + x_{23} t_{23} \leq 45$$

$AX=b$  from BILP is an equation. The deadline constraint is an inequality. Transformation from inequality to equation requires using additional variable - slack variable such that:

$$a \in \mathbb{N}, D \in \mathbb{N}, a \leq D \implies \exists s \in \mathbb{N} : a + s = D$$

One slack natural variable refers to one equality, therefore  $|P|$  such variables are necessary for the WSP formulation ( $P$  - list of paths,  $|P|=3$  for the example problem). For the Binary ILP slack variables must be binary, therefore the binary expansion is used to represent slack variables e.g.

$$s = 16s_4 + 8s_3 + 4s_2 + 2s_1 + s_0$$

- Slack variable's representation must be long enough to be able to represent the gap between the deadline and the minimum possible execution time for each path (separately).
- For the example problem each path needs 5-bit slack variable  $\rightarrow 3 \cdot 5 = 15$  additional binary variables added.
- The non-deadline parts of matrix  $A$  have the slack variables' values set to 0

This way the workflow scheduling problem has been formulated as binary integer linear programming.

# BILP transformation to QUBO

The goal of QUBO is to minimize  $x^T Q x$  where Q is an input matrix. Q can be calculated from BILP matrix A and vectors c and b in the following manner [1] (note: matrix C is a diagonal matrix with vector c values on the diagonal and zeros everywhere else)

$$y = x^T C x + \textcolor{red}{P} \cdot (Ax - b)^T (Ax - b) = x^T C x + x^T D x + c = x^T Q x + c$$

Transformation from BILP with matrices A,C and vector b to QUBO with matrix Q

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i < j} Q_{i,j} x_i x_j \quad x_i \in \{0, 1\}$$

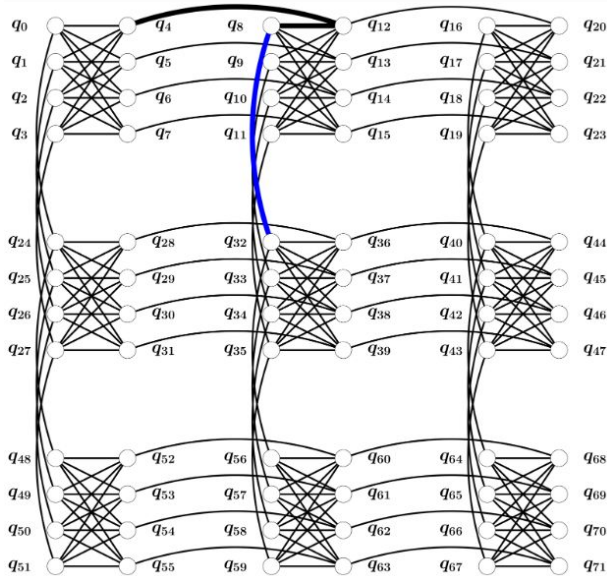
A function defined by QUBO matrix that is minimized in QUBO

A new parameter **P** needs to be introduced as QUBO has no space for constraints definition, it only allows one square matrix as an input. It defines all penalties strength relative to the costs function.

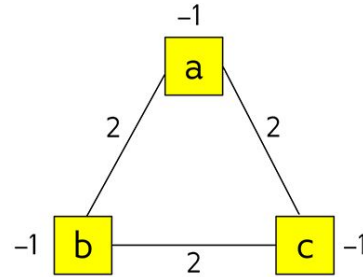
[1] Glover, F., Kochenberger, G., Du, Y.: A tutorial on formulating and using qubo models (2019)

1. Workflow problems
2. D-Wave quantum annealer
3. Jobshop problem on quantum annealer
4. Running on D-Wave - QUBO formulation
5. **Running on D-Wave - Minor embedding**
6. Results
7. Conclusions and possible improvements

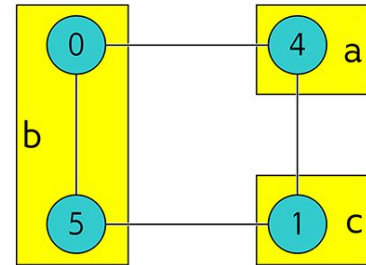
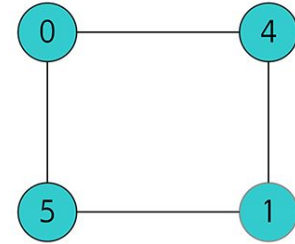
# Minor embedding



A part of Chimera graph, based on which the D-Wave 2000Q is built



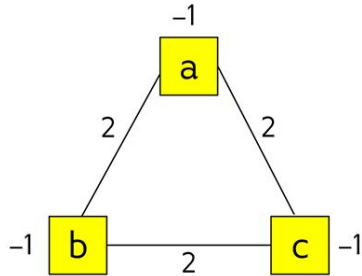
An example of a very simple embedding of a triangle into a square



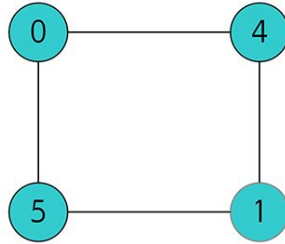
- Quantum computer architecture graph is not fully connected, but the problem solved must match its architecture.
- The problem must be transformed to the form matching the architecture by
  - finding its minor in this graph
  - (if the minor doesn't exist) representing one variable of a problem with many qubits using chains

Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_7.html](https://docs.dwavesys.com/docs/latest/c_gs_7.html)

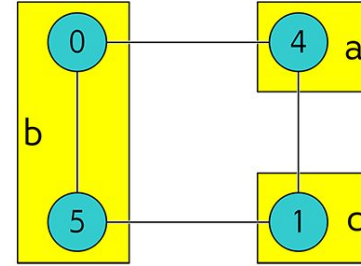
# Minor embedding



A very simple undirected graph, which could represent a problem - triangle



An extract from the Chimera graph (which is the D-Wave 2000Q architecture)



An embedding of the triangle into the presented part of the Chimera graph with the use of chain strength

A new parameter is introduced: **chain strength**, which is the value of coupling strength between physical qubits that are minor embedded as one logical qubit

# Minor embedding

A																										b
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23			
t <sub>0</sub>	t <sub>1</sub>	0	0	t <sub>4</sub>	0	0	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>	0	0	t <sub>12</sub>	0	0	t <sub>15</sub>	t <sub>16</sub>	t <sub>17</sub>	0	0	t <sub>20</sub>	0	0	t <sub>23</sub>		45	
t <sub>0</sub>	0	t <sub>2</sub>	0	0	t <sub>5</sub>	0	t <sub>7</sub>	t <sub>8</sub>	0	t <sub>10</sub>	0	0	t <sub>13</sub>	0	t <sub>15</sub>	t <sub>16</sub>	0	t <sub>18</sub>	0	0	t <sub>21</sub>	0	t <sub>23</sub>		45	
t <sub>0</sub>	0	0	t <sub>3</sub>	0	0	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	0	0	t <sub>11</sub>	0	0	t <sub>14</sub>	t <sub>15</sub>	t <sub>16</sub>	0	0	t <sub>19</sub>	0	0	t <sub>22</sub>	t <sub>23</sub>		45	

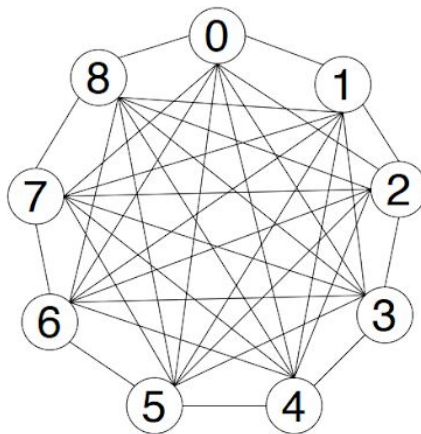
A part of matrix A and vector b for the deadline constraint.

There are lots of interactions between bits in this model e.g. bit representing  $t_0$  interacts with all other bits. For this reason the resulting QUBO matrix is dense (graph has high connectivity). Therefore we use the simple complete graph embedding (see: next slide).

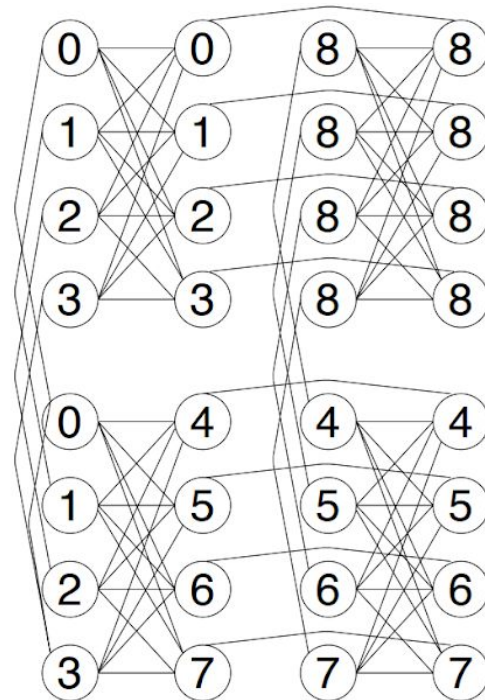
Source: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_7.html](https://docs.dwavesys.com/docs/latest/c_gs_7.html)

# Minor embedding

A complete graph can be embedded into Chimera graph as presented. Each vertex of complete graph  $K_9$  is represented by 3 vertices in Chimera graph (except no. 8). Every other graph with the size less than size of a complete graph (9 in this case) can be first embedded in the complete graph and then as complete graph into Chimera, because every graph is a minor of the complete graph with the same number of vertices.

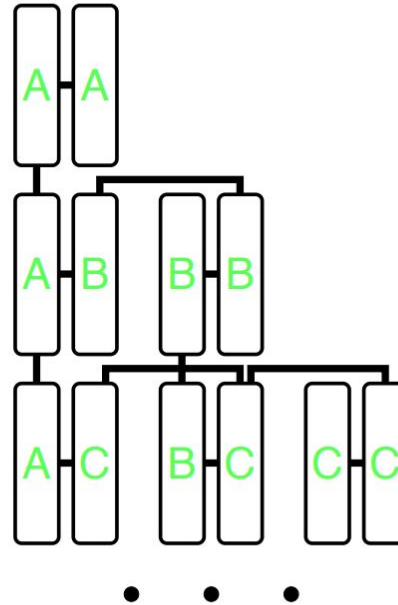
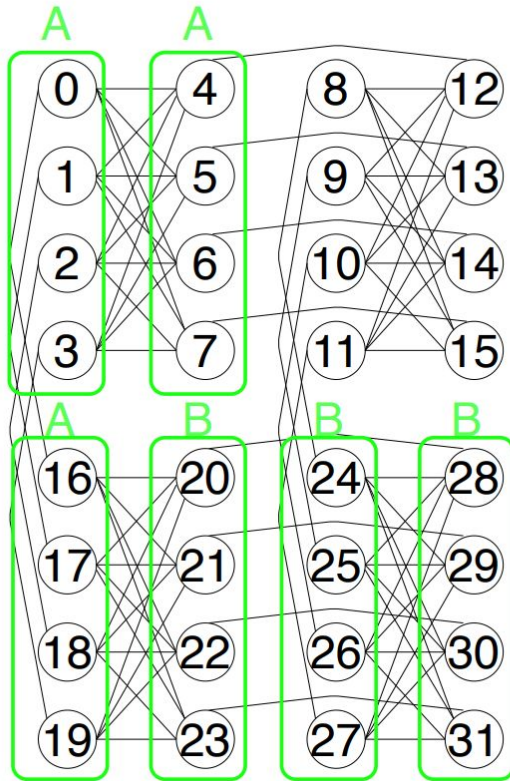


A complete  $K_9$  graph



An embedding of the  $K_9$  graph into 2x2 Chimera graph

# Minor embedding

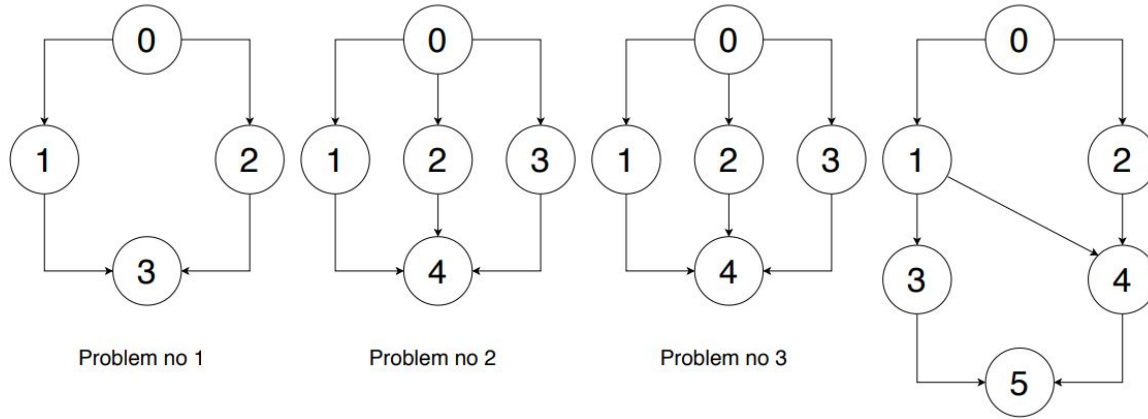


An expansion of the complete graph embedding technique into the potentially infinite Chimera lattice. Using this method it is possible to minor-embed the complete graph with 65 vertices on the 16x16 Chimera lattice.



1. Workflow problems
2. D-Wave quantum annealer
3. Jobshop problem on quantum annealer
4. Running on D-Wave - QUBO formulation
5. Running on D-Wave - Minor embedding
- 6. Results**
7. Conclusions and possible improvements

# Results - problem instances solved



The following instances of problems were solved in this work. Problems with numbers 2 and 3 have the same graph (and what follows: the same paths), but differ in number of machines and their costs.

No.	Binary variable count	$T$	$K$	paths
1	8	$\begin{bmatrix} 6 & 3 & 12 & 9 \\ 2 & 1 & 4 & 3 \end{bmatrix}$	[1,4]	[[0,1,3],[0,2,3]]
2	10	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \end{bmatrix}$	[1,4]	[[0,1,4],[0,2,4],[0,3,4]]
3	15	$\begin{bmatrix} 6 & 3 & 12 & 9 & 6 \\ 2 & 1 & 4 & 3 & 2 \\ 4 & 2 & 8 & 6 & 4 \end{bmatrix}$	[1,5,2]	[[0,1,4],[0,2,4],[0,3,4]]
4	18	$\begin{bmatrix} 12 & 6 & 42 & 18 & 30 & 24 \\ 4 & 2 & 14 & 6 & 10 & 8 \\ 8 & 4 & 28 & 12 & 20 & 16 \end{bmatrix}$	$K=[8,18,6]$	[[0,1,3,5],[0,1,4,5],[0,2,4,5]]

# Results - finding parameters' values

Three parameters need to be set up before creating the QUBO matrix:

- **P** - penalty strength relative to the costs function
- **S** - single execution constraint strength (relative to general penalty constraint parameter P)
- **Chain strength** - the value of coupling strength between physical qubits that are minor embedded as one logical qubit

For the purpose of this proof of concept

- values of parameters P and S were found with the use of Gurobi [1] sampler,
- chain strength was tested both with Gurobi and the D-Wave annealer

[1] <https://www.gurobi.com/>

## Results - parameters' values

No.	Binary variable number	Deadline	$P$	$S$	Chain strength	Percentage of correct solutions
1	8	19	8	10	1200	62,5%
2	10	19	14	25	6650	56,25%
3	15	17	11	10	2800	54,3%
4	18	70	6	40	18000	46,91%

In the previous parts three parameters were introduced:  $S$ ,  $P$  and chain strength. Each of these parameters must have been found. A Gurobi sampler was used to find parameters  $P$ ,  $S$ , for chain strength Gurobi with D-Wave 2000Q were used. Doing so is possible for small instances. For large instances the algorithms and heuristics for finding these values would be highly usable. The percentage of correct solutions value describes how likely the random solution (from uniform distribution over the whole solutions' space) is to be correct for. The correctness is defined as meeting all the constraints.

# Results

No.	Binary variables number	Correct solutions samples (from 2000 samples)	Unique correct solutions	Global optimum found	Time	Cost
1	8	98	10 (100%)	YES	19(d=19)	34(min=34)
2	10	27	14 (77.8%)	YES	16(d=19)	40(min=40)
3	15	4	4 (3.03%)	NO (6)	16(d=17)	45(min=40)
4	18	0	0 (0%)	NO (65862)	N/A	N/A

Each of the four problems was sampled 2000 times. The third column shows how many samples with correct results were returned. The next column describes how many unique samples with correct solutions were returned and their percentage from all the correct solutions (e.g. for the first problem all of the 10 correct solutions were found in 98 samples, in the third only 4 out of 132 possible correct solutions were found, one sample for each).

# Results - reference methods

Four methods were used to compare with the D-Wave 2000Q performance. Three of them always found the global optima. The method doing exactly the same as the D-Wave 2000Q (Gurobi solving the minor embedded QUBO) had a similar performance: for three problems found the global optimum, for the largest did not).

## Methods always finding the global optima

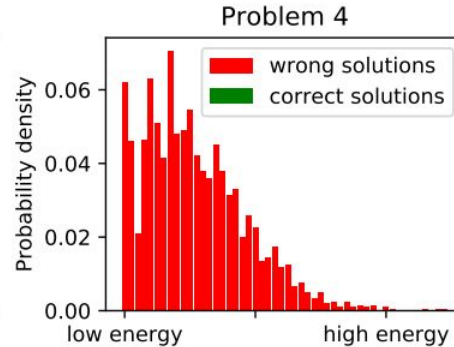
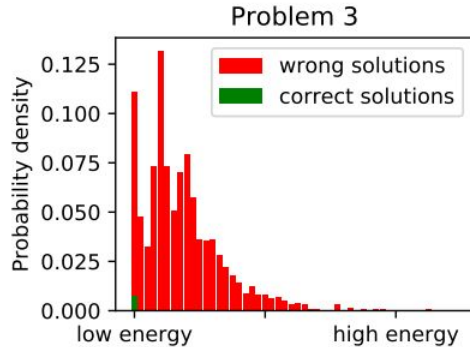
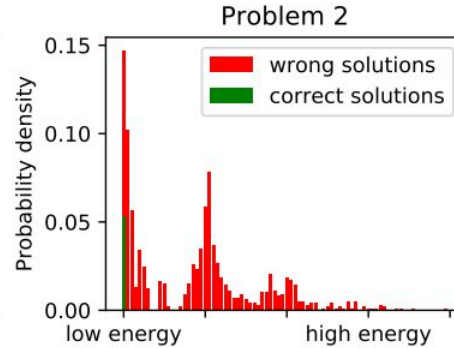
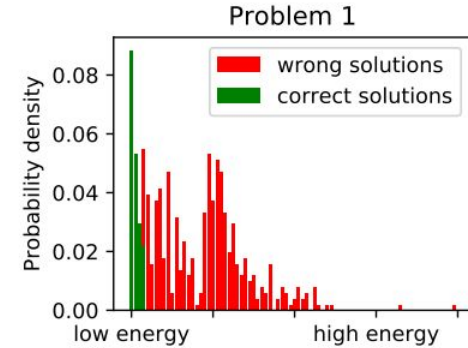
- Brute force for the WSP
- GNU Linear Programming Kit for BILP [1]
- Gurobi for QUBO

- Gurobi for minor embedded QUBO

No.	Binary variables number	Global optimum found
1	8	YES
2	10	YES
3	15	YES
4	18	NO, wrong (22831)

[1] <https://www.gnu.org/software/glpk/>

# Results



The histograms present the samples' distribution for each problem. The correct solutions are marked with green color. These results are coherent with the column from results table presented below.

Correct solutions samples (from 2000 samples)	
98	
27	
4	
0	

1. Workflow problems
2. D-Wave quantum annealer
3. Jobshop problem on quantum annealer
4. Running on D-Wave - QUBO formulation
5. Running on D-Wave - Minor embedding
6. Results
7. **Conclusions and possible improvements**



# Conclusions

It is possible to solve the basic instances of workflow scheduling problems

## **Obstacles:**

- Slack variables number - increasing logarithmically with the deadline value and linearly with the paths number
- P and S parameters - need for testing a lot - impossible for larger instances
- High connectivity in problem graph - long chains

## **Possible improvements:**

- Different problem translation (e.g. changing the encoding of variables [1])
- Using time windows
- Dividing the QUBO into smaller instances

# Foundations for Workflow Application Scheduling on D-Wave System

Dawid Tomasiewicz<sup>1</sup>, Maciej Pawlik<sup>1</sup>, Maciej Malawski<sup>1</sup>, and Katarzyna Rycerz<sup>1</sup>

Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland  
tomasiewicz.dawid@gmail.com, {mapawlik,malawski,kzajac}@agh.edu.pl

**Abstract.** Many scientific processes and applications can be represented in the standardized form of workflows. One of the key challenges related to managing and executing workflows is scheduling. As an NP-hard problem with exponential complexity it imposes limitations on the size of practically solvable problems. In this paper, we present a solution to the challenge of scheduling workflow applications with the help of the D-Wave quantum annealer. To the best of our knowledge, there is no other work directly addressing workflow scheduling using quantum computing. Our solution includes transformation into a Quadratic Unconstrained Binary Optimization (QUBO) problem and discussion of experimental results, as well as possible applications of the solution. For our experiments we choose four problem instances small enough to fit into the annealer's architecture. For two of our instances the quantum annealer finds the global optimum for scheduling. We thus show that it is possible to solve such problems with the help of the D-Wave machine and discuss the limitations of this approach.

Thank you