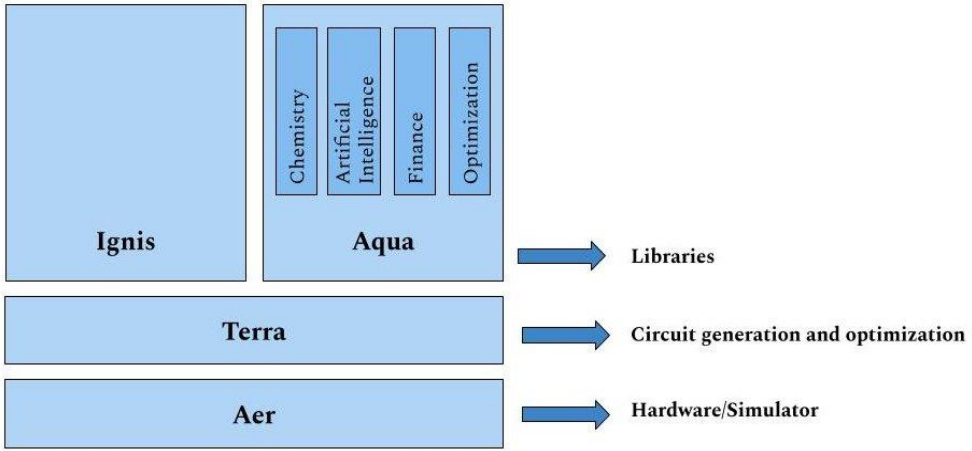
Solving Optimization problems using Qiskit Aqua

Katarzyna Rycerz, Małgorzata Stachoń, Dawid Tomasiewicz

Goals

- analize Qiskit Aqua in terms of solving optimisation problems
- apply this solution to a workflow scheduling problem
- comparison of simulator results with the real device



Qiskit framework for IBM-Q

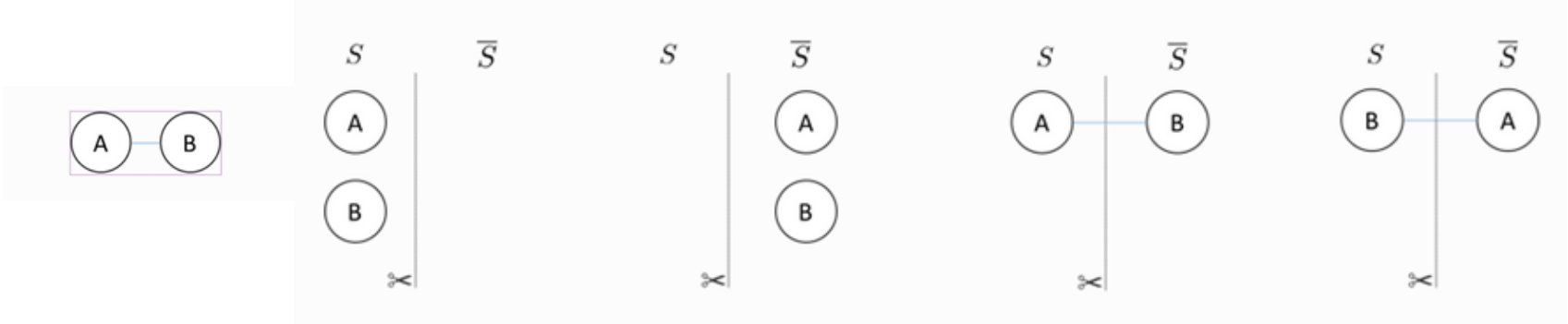
Aqua for optimization

1. Express you problem as a cost function:

$$\sum_{j=1}^{m} C_j(x),\ x_i \in \{0,1\}$$

2. Translate your function into a Hamiltonian H such that:
   a. minimal eigenvalue of H is the optimal value of the cost function
   b. corresponding eigenstate for the minimal value is a solution (corresponding to x)
3. Use existing algorithms (VQE, QAOA) for finding a solution
4. Helping tools: DocPlex library – helps to create Hamiltonian

Example: MaxCut problem



- Cost function

$$C(x_1, x_2) = x_1 + x_2 - 2x_1x_2 \qquad x_1, x_2 \in \{0, 1\}$$

- C(00)=0 C(11)=0
- C(01)=1 C(10)=1

Source: https://grove-docs.readthedocs.io/en/latest/qaoa.html

Example: MaxCut problem

- Cost function

$$C(x_1, x_2) = x_1 + x_2 - 2x_1x_2 \qquad x_1, x_2 \in \{0, 1\}$$

$$x \to \frac{1}{2}(1 - z)$$

$$C(z_1, z_2) = \frac{1}{2}(1 - z_1z_2) \qquad z_1, z_2 \in \{-1, 1\}$$

$$\sigma_z|0>=|0>, \lambda = 1 \qquad \sigma_z|1>=-|1>, \lambda = -1$$

- To build Hamiltonian

$$z \to \sigma_z$$

- Eigenvalues
  - 0 for |00> and |11>; C(00)=0 C(11)=0
  - 1 for |01> and |10> ; C(01)=1 C(10)=1

$$\begin{array}{c} \begin{matrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix} \end{array}$$

$$H_{i,j} = \frac{1}{2}(I - {\sigma_z}^i \otimes {\sigma_z}^j)$$

Source: https://grove-docs.readthedocs.io/en/latest/qaoa.html

Example: BILP problems

BILP Cost function and linear constraints for binary variables

$$f(x) = c^T x \qquad Ax = b \qquad x_i \in \{0, 1\}$$

C-diagonal, diag(C)=c

QUBO - quadratic and uncontraint cost function $f(x) = x^T Q x$

$$f(x) = x^T C x + P(Ax - b)^T (Ax - b)$$

Instead of solving Ax=b
we minimize inner product of Ax-b

$$x_i \rightarrow \frac{I - {\sigma_z}^i}{2} \qquad {\sigma_z}^i = I \otimes I \cdots \otimes \underbrace{\sigma_z}_{\text{i- th position}} \cdots \otimes I$$

Hamiltonian

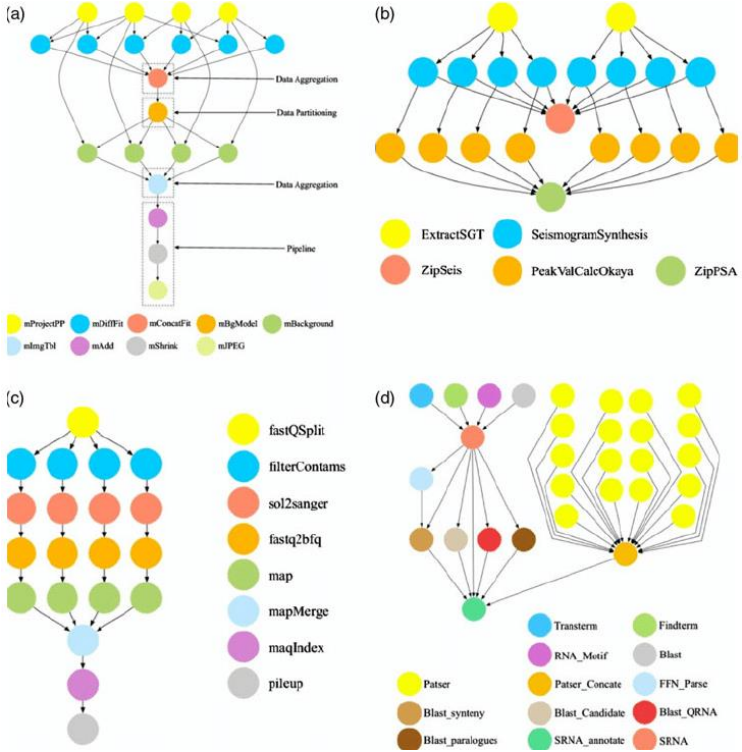
$$H = \sum_i c_i \frac{I - {\sigma_z}^i}{2} + P \sum_j \left(\sum_i a_{j,i} \left(I - \frac{{\sigma_z}^i}{2}\right) - b_i I\right)^2$$

Workflows

- A paradigm commonly used for describing complex scientific processes and applications.
- Usually represented as DAGs (Directed Acyclic Graphs)
- Example applications: Montage - construction human perceptible images from multiple telescope images [1], calculating seismic hazard maps [2]

Workflow scheduling (DAG scheduling, task 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



Examples of scientific workflows: (a) Montage workflow, (b) CyberShake workflow, (c) Epigenomics workflow and (d) Sipht workflow

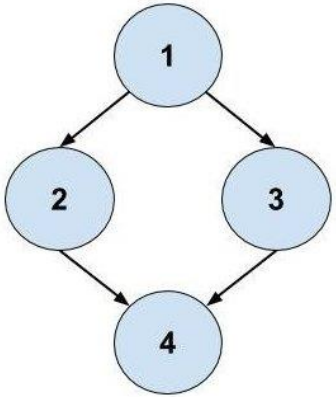
Source: Ghafarian, Toktam & Javadi, Bahman & Buyya, Rajkumar. (2014). Decentralised workflow scheduling in volunteer computing systems. International Journal of Parallel Emergent and Distributed Systems.

[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)

Workflow scheduling problem

- useful for Cloud serverless architectures
- given ***t*** tasks, ***m*** machine types
- infinite number of machine types instances
- tasks in a form of DAG (***p*** paths)
- cost matrix, time matrix, deadline
- **goal: to minimize cost**
- **constraint: to fulfill deadline d**



	task 1	task 2	task 3	task 4
machine 0	6	3	12	9
machine 1	8	4	16	12

Cost matrix

	task 1	task 2	task 3	task 4
machine 0	6	3	12	9
machine 1	2	1	4	3

Time Matrix

BILP for WSP

Objective function
t - # of tasks,
i-task number
j -machine number

$$\sum_{j,i} c_{t \cdot j+i} \cdot x_{t \cdot j+i}$$

		Task number				
		0	**1**	**2**	**3**	**4**
machine number	**0**	x_0	x_1	x_2	x_3	x_4
	1	x_5	x_6	x_7	x_8	x_9

Constraints

1.Machine usage: each row "remembers" binary variables where one of the tasks is assigned to different machines
i -binary variable index

$$u_{v,i} = \begin{cases} 1 \text{ if } (i \mod t) = v \\ 0 \text{ otherwise} \end{cases} \qquad \sum_{i=0}^{t \times m-1} u_{v,i} x_i = 1$$

2.Deadline: each row collects times for different DAG path

$$r_{w,i} = \begin{cases} t_i \text{ if path } w \text{ contains task } (i \mod t), \\ 0 \text{ otherwise} \end{cases}$$

Slack variables necessary!

$$Rx \leq D \rightarrow Rx + \sum_{s=0}^{S} 2^s x_{t \times m+s} = D$$

Tomasiewicz D., Pawlik M., Malawski M., Rycerz K. (2020) Foundations for Workflow Application Scheduling on D-Wave System. In: Krzhizhanovskaya V. et al. (eds) Computational Science – ICCS 2020.

Hamiltonian for WSP

- cost function $a = \sum_i^{t \times m-1} c_i \frac{I-{\sigma_z}^i}{2}$
- machine usage constraint $b = \sum_{v=0}^{t-1}(1 - \sum_{i=0}^{t \times m-1} u_{v,i} \frac{I-{\sigma_z}^i}{2})^2$
- deadline constraint

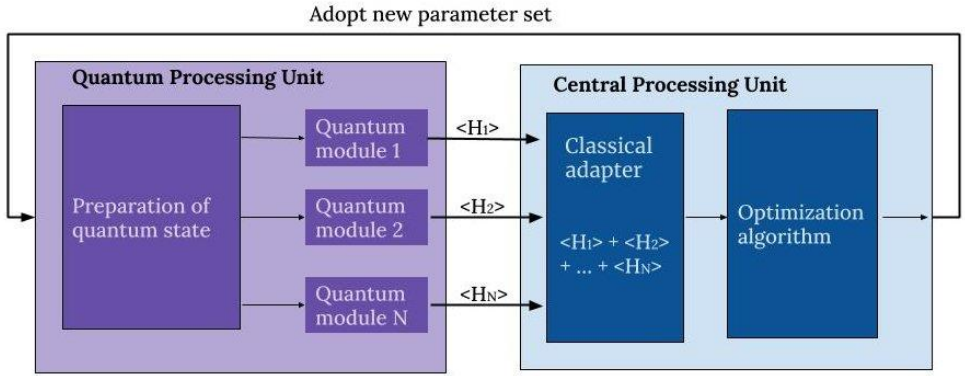
$$c = \sum_{w=0}^{p-1}(d - (\sum_{i=0}^{t \times m-1} r_{w,i} \frac{I-{\sigma_z}^i}{2} + \sum_{s=0}^{S} 2^s \frac{I-{\sigma_z}^s}{2}))^2$$

Putting it all together **requires to set up weigths A,B,C,** so:

$$H = A \cdot a + B \cdot b + C \cdot c$$

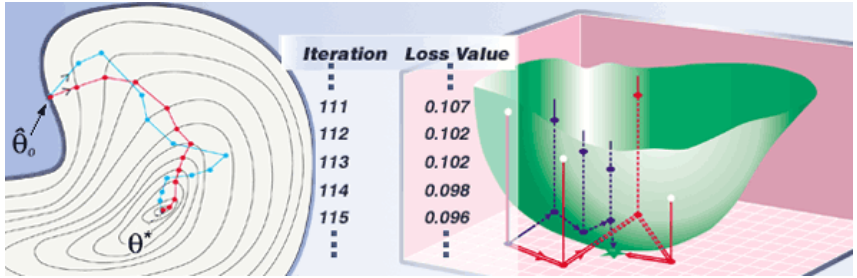
Variational Quantum Eigensolver

- a hybrid algorithm: a quantum subroutine runs inside of a classical optimization loop
- quantum part is responsible for :
  - preparing a quantum state (ansatz) basing on the parameters got from classical part
  - calculate expectation value of H for a given state (as a sum of expectation values of Pauli operators)
- the variational principle ensures that this expectation value is always greater than the smallest eigenvalue of H (lower bound)
- classical part uses non-linear optimizer to minimize the expectation value by varying ansatz parameters



VQE in Qiskit - classic optimizer choice

- various optimizers available(Adam, gradient descend variants, Nelder-Mead, …)
- possible to add a new one
- may depend on the specific problem
- for noise computations - Simultaneous Perturbation Stochastic Approximation (SPSA) is recommended (gradient estimator requires only two measurements of the objective function)

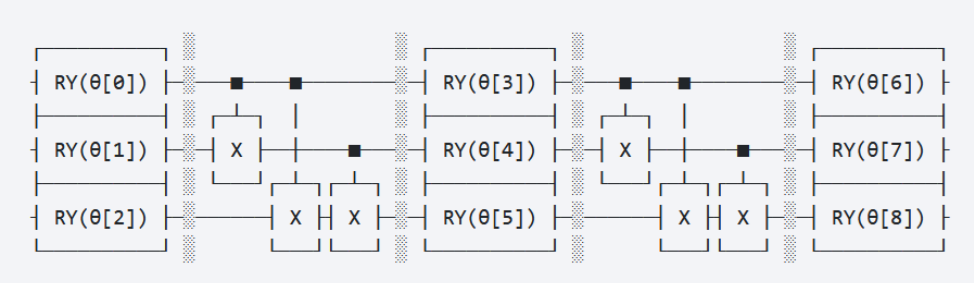


Spall, J. “AN OVERVIEW OF THE SIMULTANEOUS PERTURBATION METHOD FOR EFFICIENT OPTIMIZATION.” Johns Hopkins Apl Technical Digest 19 (1998): 482-492.

source: https://www.jhuapl.edu/SPSA/

VQE in Qiskit - variational form parameters

- variational form
  - parametrized quantum circuit called RealAmplitudes
  - quantum states with real amplitudes only (uses Y rotations and CNOTs)
  - need to set parameters:
    - entanglement (which qubits are entangled)
    - reps parameter (how often the sets of gates are repeated)



source: https://qiskit.org/documentation/stubs/qiskit.circuit.library.RealAmplitudes.html

VQE other important choices

- backend (simulator, real device)
- shots number for quantum algorithm
- seed parameter
- number of interactions for classical optimiser

Docplex tool

- generates Ising Hamiltonian for the specific optimization problem
- only binary variables can be used to represent the problem,
- only equality constraints,
- linear or quadratic optimized function

Two methods of using Docplex were tried:

1. Automatic generation of final Hamiltonian directly from BILP form (difficulties in setting arbitrary weights)
2. Docplex models for each part of QUBO + Qiskit quadratic programming converter

Test problem

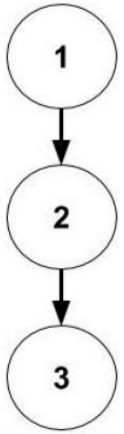
small problem (due to small number of qbits !)

	task 1	task 2	task 3
machine 0	6	3	12
machine 1	2	1	4

time matrix

	task 1	task 2	task 3
machine 0	6	3	12
machine 1	8	4	16

cost matrix



Configuration	Result vector	Total time	Total cost
$t_{1,1}+t_{2,1}+t_{3,1}$	0001111100	7	28
$t_{1,1}+t_{2,1}+t_{3,0}$	0011100100	15	24
$t_{1,1}+t_{2,0}+t_{3,1}$	0101011010	9	27
$t_{1,1}+t_{2,0}+t_{3,0}$	0111000010	17	23
$t_{1,0}+t_{2,1}+t_{3,1}$	1000111000	11	26
$t_{1,0}+t_{2,1}+t_{3,0}$	1010100000	19	22
$t_{1,0}+t_{2,0}+t_{3,1}$	1100010110	13	25
$t_{1,0}+t_{2,0}+t_{3,0}$	111000xxxxx	21	21

- 10 qbits (with slacks)
- 1024 possible configurations,
- 7 correct (grey)
- 1 optimal (blue),

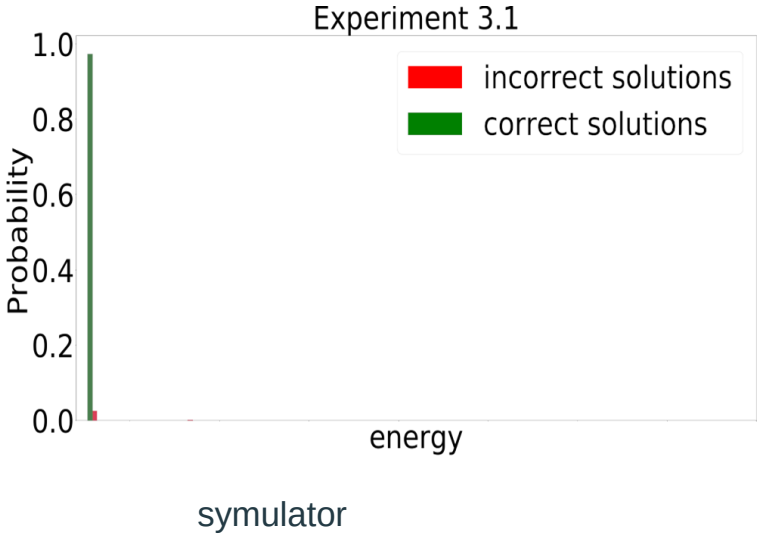
- *correct configs* represents the percentage of unique correct configurations that occurred in the experiment result.

Setting Parameters

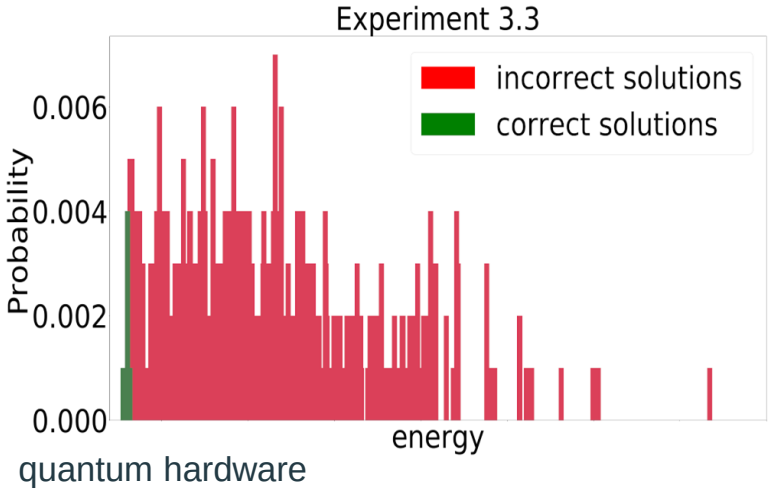
- automatic set of A, B, C weights by Docplex didn't returned satisfying results
- for A, B, C weigths the grid search method was used:
  - the lower bound was set to 1 and
  - the upper bound was set to quintuple of maximum from maximal task execution time and maximal task execution cost.
  - the energies for all eigenvalues were computed with the use of reference direct eigensolver (NumPyEigensolver)
  - the configurations were tested in qasm_simulator from Aer provider.
- the best results were achieved when setting parameters A, B, C to 1, 40, 1
- for the variational form
  - reps parameter set to 2
  - entanglement parameter set to full

Results

experiment number	backend	optimizer iterations	optimal solutions	correct solutions	incorrect solutions	correct configs
3.1	qasm_simulator	1000	97,3%	97,3%	2,7%	14,29%
3.2	qasm_simulator	2000	98,1%	98,1%	1,9%	14,29%
3.3	ibmq_16_melbourne	1000	0,1%	0,7%	99,3%	57,14%
3.4	ibmq_16_melbourne	2000	0%	0,8%	99,2%	57,14%



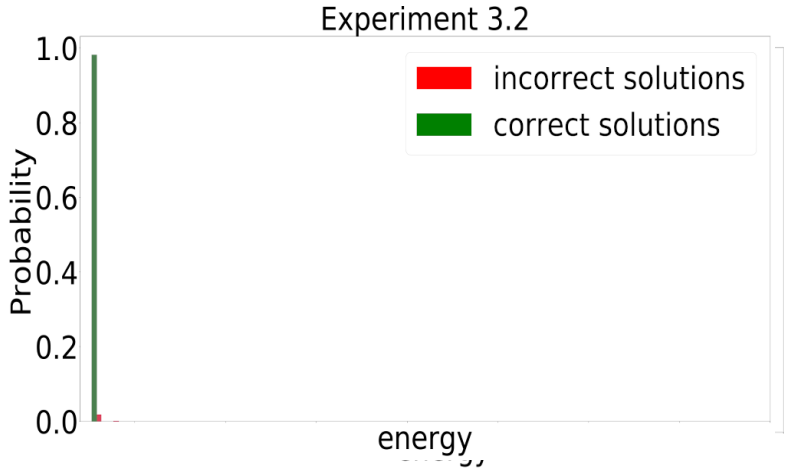
symulator



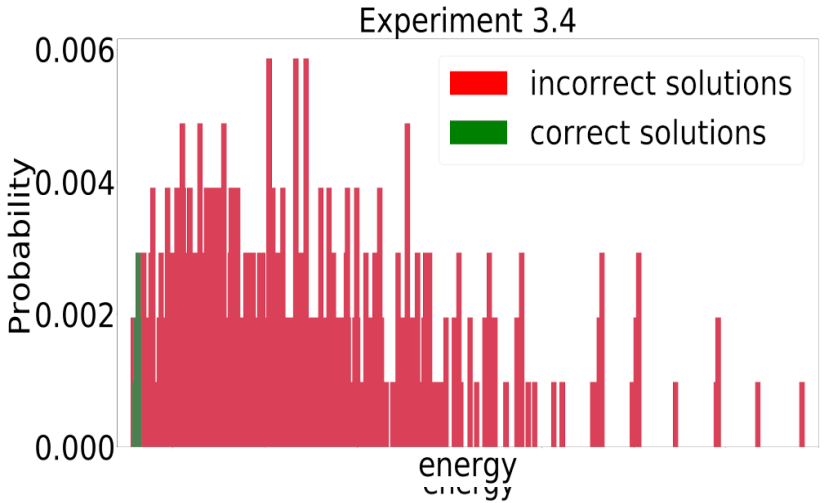
quantum hardware

Results

experiment number	backend	optimizer iterations	optimal solutions	correct solutions	incorrect solutions	correct configs
3.1	qasm_simulator	1000	97,3%	97,3%	2,7%	14,29%
3.2	qasm_simulator	2000	98,1%	98,1%	1,9%	14,29%
3.3	ibmq_16_melbourne	1000	0,1%	0,7%	99,3%	57,14%
3.4	ibmq_16_melbourne	2000	0%	0,8%	99,2%	57,14%



symulator



quantum hardware

Summary and conclusions

- We showed how to solve (small) workflow scheduling problem using Qiskit Aqua
- The simulator results confirm that the solution is correct
- Both Docplex and Quadratic Programming Support in Qiskit are helpful

Finding appropriate settings is difficult and often need to be supported by reference methods:

- setting weights for different problem conditions when building Hamiltonian
- choice of the classical optimizer part in hybrid algorithms
- setting different parameters for classical optimizer and quantum variational form
- appropriate setting depends strongly on the problem
- still too much noise in real hardware

Future (and current) Work

- Different problem translation (e.g. changing the encoding of variables [1,4])
- Using concepts from time windows [2]
- Dividing the QUBO into smaller instances [3]
- More experiments with variational form parameters (entanglement vs reps)
- Different optimisation algorithms (QAOA, Grover Adaptive Search)
- More help from strongly developing Qiskit (e.g. automatic translation from inequality to equality)
- Improving quantum device results: experiments with noise model, error mitigation, checking compiler optimization levels

[1] Nicholas Chancellor. Domain wall encoding of discrete variables for quantum annealing and qaoa. Quantum Science and Technology, 4(4):045004, 2019
[2]Kurowski, K., et al.: Hybrid quantum annealing heuristic method for solving Job Shop Scheduling Problem. ICCS (2020). https://doi.org/0.1007/978-3-030-50433-5_39
[3] D'Wave hybrid https://docs.ocean.dwavesys.com/projects/hybrid/en/latest/
[4] Adam Glos, Aleksandra Krawiec, Zoltán Zimborás Space-efficient binary optimization for variational computing https://arxiv.org/abs/2009.07309