EURO^2

Solving Optimisation Problems Using the NISQ Era Quantum Computers

Jiří Tomčala

13 June 2024

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Outline

- Introduction
- Emerging problems of NISQ computers
- Concept of Variational Quantum Algorithms (VQAs)
- Quantum Approximate Optimizing Algorithm (QAOA)
- Quadratic Unconstrained Binary Optimization (QUBO)
- QUBO hands-on examples
- High-order Unconstrained Binary Optimization (HUBO/HOBO)
- HUBO hands-on examples

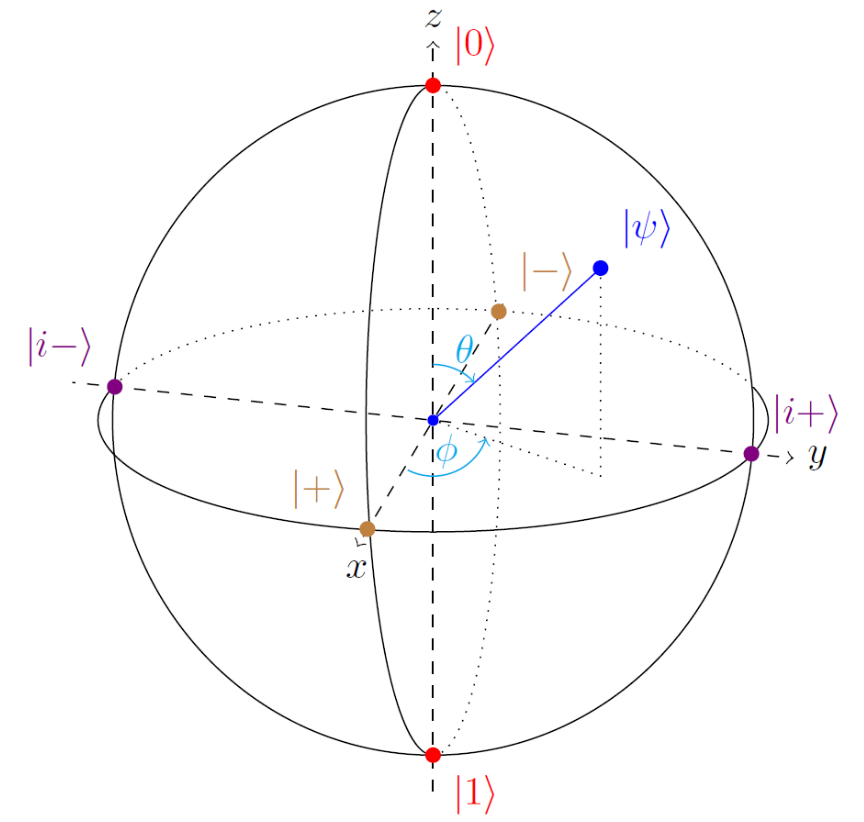
Introduction

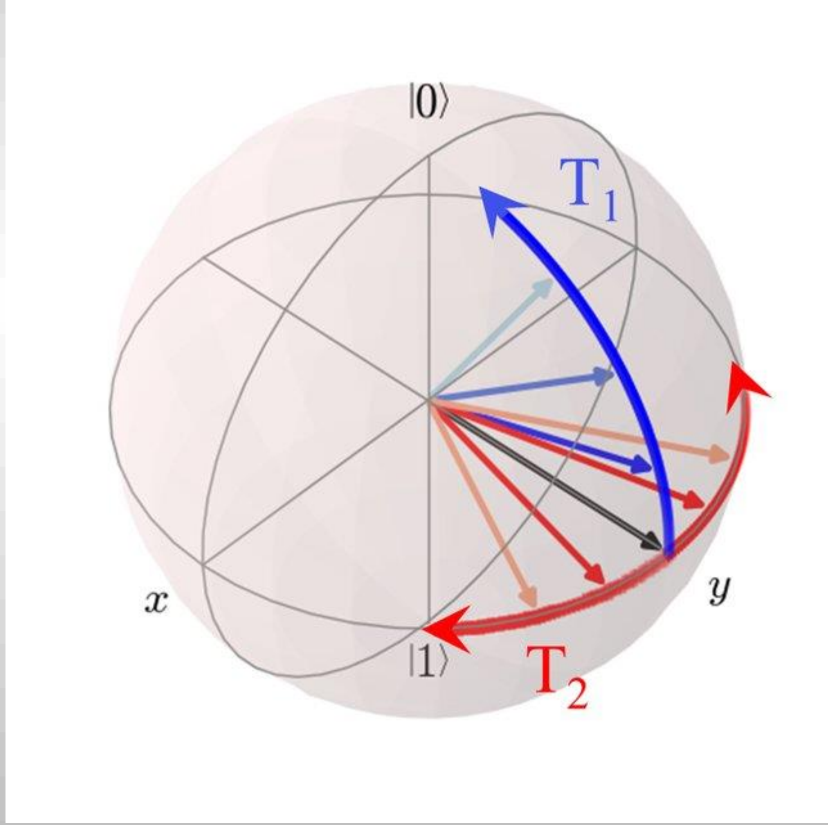
- The availability of fault-tolerant quantum computers is still many years or even decades away
- We are currently living in the NISQ era of quantum computing
- NISQ era = noisy intermediate-scale quantum era
- We have a limited number of qubits, limited qubit connectivity, and both coherent and incoherent errors that limit the depth of a quantum circuit
- These are the reasons why Variational Quantum Algorithms (VQAs) have emerged as the leading strategy to obtain quantum advantage on NISQ devices
- Accounting for all of the constraints imposed by NISQ computers with a single strategy requires an optimization-based or learning-based approach, which is precisely what VQAs do
- VQAs use parametrized quantum circuits to be run on the quantum computer, and then outsource the parameter optimization to a classical optimizer
- This approach has the added advantage of keeping the quantum circuit depth shallow and hence mitigating noise, in contrast to quantum algorithms developed for the fault-tolerant era

Emerging problems of NISQ computers

- **Sampling noise** – the number of samples is too small
- **Temperature fluctuations**
- **Mechanical vibrations**
- **Thermal noise** – quantum Brownian motion of qubit particles
- **Cosmic rays**
- **Interaction among qubits**
- **Energy relaxation** – causing spontaneous bit-flip error ($T_1 \approx 250\ \mu s$)
- **Dephasing** – causing spontaneous phase error ($T_2 \approx 150\ \mu s$)

Emerging problems of NISQ computers





Clément Godfrin. Quantum information processing using a molecular magnet single nuclear spin qudit. *Quantum Physics* [quant-ph]. Université Grenoble Alpes, 2017.

Concept of VQAs

Consider a <u>problem</u> to solve.

Define a cost (or loss) function C which encodes the solution to the problem.

Based on the cost function C, assemble the operator $\boldsymbol{H}$ (Hamiltonian) from a linear combination of Pauli spin operators.

Construct a suitable parameterized quantum circuit (ansatz) whose parameter set is denoted by $\boldsymbol{\theta}$.

Train ansatz in the hybrid quantum-classical loop to solve optimization task:

$$\boldsymbol{\theta}^* = \arg\min_{\theta} C(\boldsymbol{\theta})$$

where $\boldsymbol{\theta}^*$ is the set of parameters of the trained ansatz, from the measured output of which the desired <u>solution to the problem</u> can then be found.

Concept of VQAs

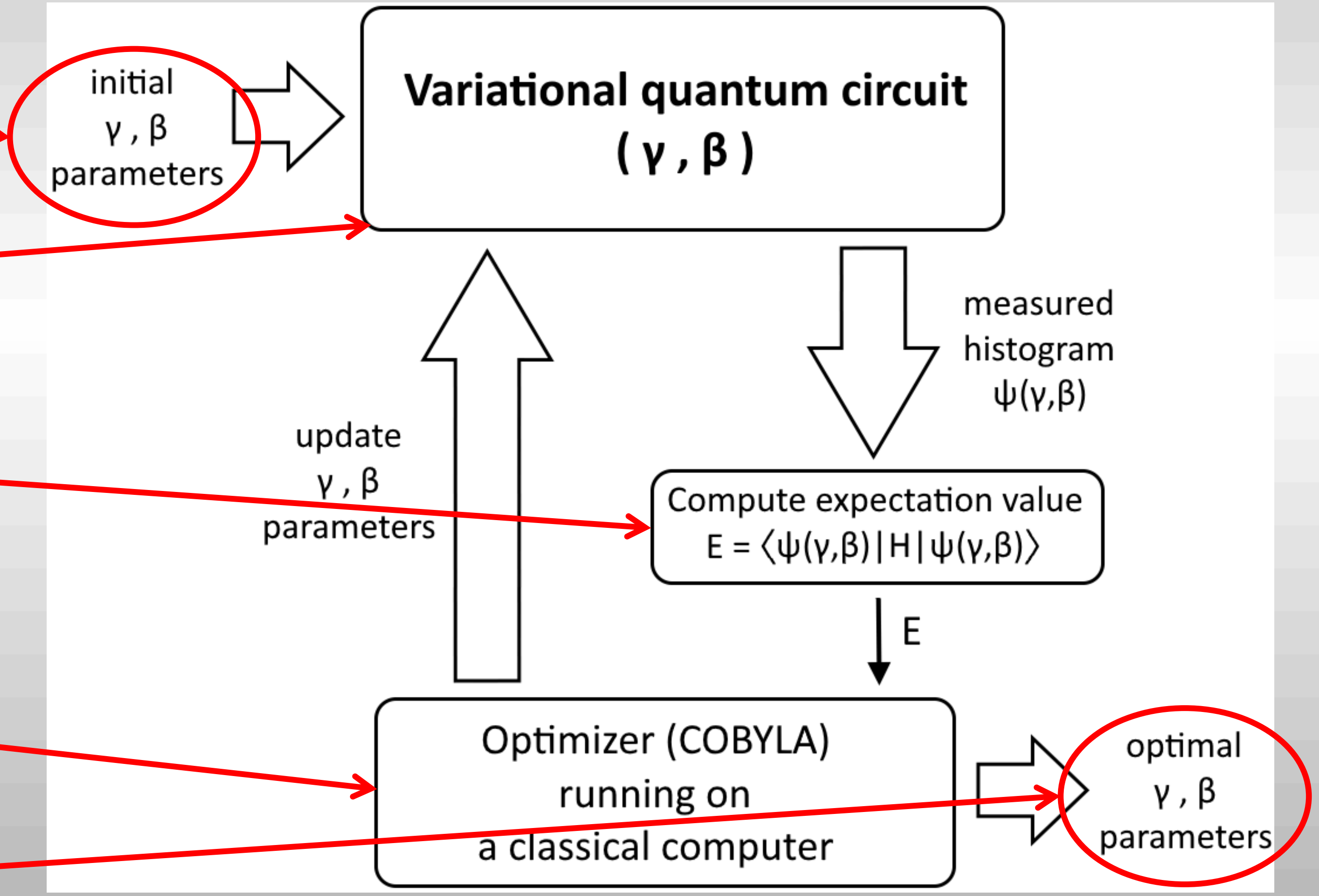
Initialize set $\boldsymbol{\theta} = \{\boldsymbol{\gamma}, \boldsymbol{\beta}\}$ to some starting values.

1. Measure output of ansatz.

2. Compute expectation value (value of the cost function).

3. Using classical optimizer update parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$.

Do steps **1** to **3** until E is minimal.



QAOA (Quantum Approximate Optimizing Algorithm)

Among the various VQAs, the QAOA is the most suitable algorithm for solving optimization problems.

The general concept of VQAs from the previous slide also applies to it.

Here too, a minimum expected value is sought.

BUT, the solution here is the state vector $\boldsymbol{\psi}$, which was measured for the optimal $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters that were found using the hybrid quantum-classical loop from the previous slide:

$$\boldsymbol{\psi}^* = \boldsymbol{\psi}(\boldsymbol{\theta}^*) = \boldsymbol{\psi}(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*).$$

QAOA (Quantum Approximate Optimizing Algorithm)

Detailed procedure:

- Build a real cost function C with binary variables $x_0, x_1, \cdots, x_{n-1}$:

$$C\colon \{0,1\}^n \to \mathbb{R}.$$

- Transform the cost function C into a spin operator $\boldsymbol{H}$ (Hamiltonian) composed of σ^z Pauli spin operators:

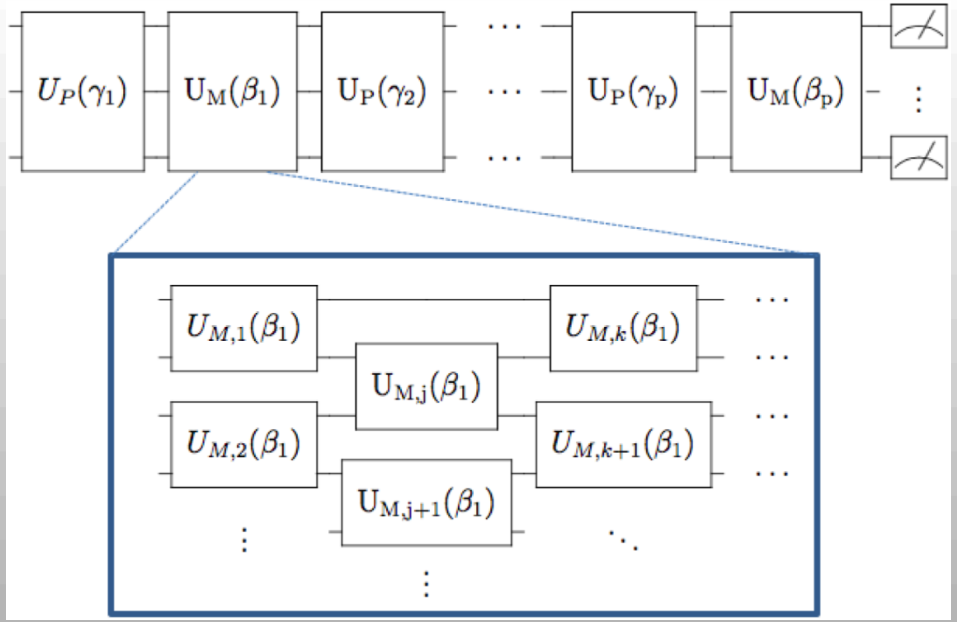
$$x_i = (1 - \sigma_i^z)/2\ , \text{ where } \ \sigma_i^z \in \{-1, +1\}.$$

- Make an ansatz. One of the universal types such as fully entangled, linearly entangled, or various combinations thereof can be used.
- However, the Quantum Alternating Operator Ansatz and so-called problem-specific ansatz is the most suitable for QAOA.

QAOA (Quantum Approximate Optimizing Algorithm)

***Quantum Alternating Operator Ansatz** (QAOA)*

$$U(\boldsymbol{\theta}) = U(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \prod_{l=1}^{p} e^{-i\,\beta_l \boldsymbol{H}_M}\; e^{-i\,\gamma_l \boldsymbol{H}_P}$$

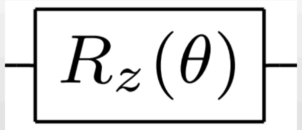


Hadfield, S.; Wang, Z.; O'Gorman, B.; Rieffel, E.G.; Venturelli, D.; Biswas, R. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* 2019, 12, 34. https://doi.org/10.3390/a12020034

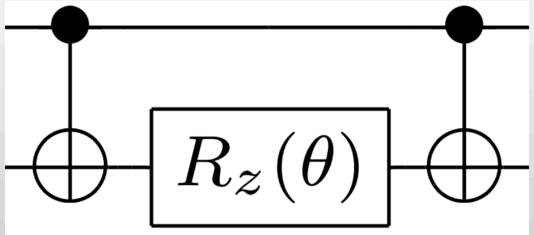
QAOA (Quantum Approximate Optimizing Algorithm)

<u>Problem-specific ansatz</u>

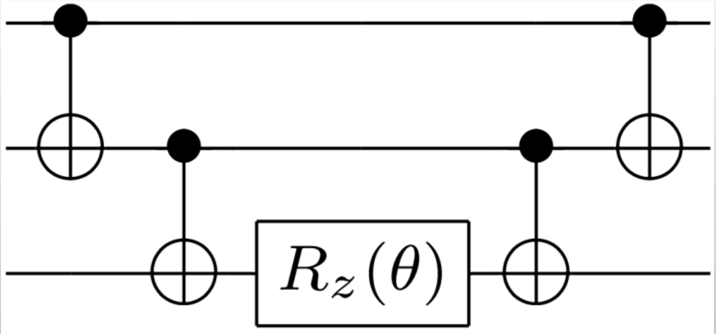
$$e^{-i\,\frac{\theta}{2}\sigma_0^z}$$



$$e^{-i\,\frac{\theta}{2}(\sigma_0^z \otimes \sigma_1^z)}$$



$$e^{-i\,\frac{\theta}{2}(\sigma_0^z \otimes \sigma_1^z \otimes \sigma_2^z)}$$



QUBO (Quadratic Unconstrained Binary Optimization)

The cost function contains at most quadratic terms.

It can therefore be expressed in the general form:

$$C(\boldsymbol{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}\, x_i\, x_j$$

Qiskit includes an implemented class for QUBO problem formulation.

Together with the use of a class for QAOA, it is then relatively easy to solve this kind of optimization problems there.

QUBO hands-on examples

There is prepared a simple example for finding the minimum of the function:

$$C(\boldsymbol{x}) = x_0x_1 - x_0x_2 + 2x_1x_2 + x_0 - 2x_1 + 3x_2$$

QAOA-QUBO1-basic_example.ipynb https://1url.cz/G1rBC

The task in the second example is to find two integers whose product is maximal under the condition that their sum is 9.

$$\max XY = \min[-XY] = \min[-(4x_2 + 2x_1 + x_0)(4y_2 + 2y_1 + y_0)]$$
$$9 = X + Y = 4x_2 + 2x_1 + x_0 + 4y_2 + 2y_1 + y_0$$

$$C(\boldsymbol{x}, \boldsymbol{y}) = k(4x_2 + 2x_1 + x_0 + 4y_2 + 2y_1 + y_0 - 9)^2 - (4x_2 + 2x_1 + x_0)(4y_2 + 2y_1 + y_0)$$

QAOA-QUBO2-Max_product.ipynb https://1url.cz/01JSO

HUBO (High-order Unconstrained Binary Optimization)

EURO[2]

The cost function contains terms of order higher than quadratic.

It can be expressed in the general form:

$$C(\boldsymbol{x}) = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{i_1, i_2, \dots, i_m} \; x_{i_1} x_{i_2} \dots x_{i_m}$$

There is no special class for HUBO-type problems in Qiskit, so there are two ways to solve them there:

- transform the cost function from the HUBO to the QUBO formulation
- or program the whole thing by yourself

(which, for educational reasons, was my way ;-))

HUBO hands-on examples

There is prepared an example for factorization of the number 119.

Factorization problem is here converted into an optimization problem:

$$N = 119 = (1110111)_2 = P \times R = \ (1\ p_3\ p_2\ p_1\ 1)_2\ \times\ (1\ r_1\ 1)_2$$

$$\begin{array}{ccccccc}
 & & 1 & p_3 & p_2 & p_1 & 1 \\
 & & & & 1 & r_1 & 1 \\
\hline
 & & 1 & p_3 & p_2 & p_1 & 1 \\
 & r_1 & p_3 r_1 & p_2 r_1 & p_1 r_1 & r_1 & \\
1 & p_3 & p_2 & p_1 & 1 & & \\
\hline
 & c_{45} & c_{34} & c_{23} & c_{12} & & \\
\hline
1 & 1 & 1 & 0 & 1 & 1 & 1
\end{array}$$

HUBO hands-on examples

$$
\begin{aligned}
p_1 + r_1 &= 1 && \Rightarrow && p_1 + r_1 - 1 = 0 \\
p_1 + p_3 &= 2c_{34} && \Rightarrow && p_1 + p_3 - 2c_{34} = 0 \\
p_3 r_1 + c_{34} &= 2c_{45} && \Rightarrow && p_3 r_1 + c_{34} - 2c_{45} = 0 \\
p_3 + r_1 + c_{45} &= 1 && \Rightarrow && p_3 + r_1 + c_{45} - 1 = 0
\end{aligned}
$$

Constructing the cost function:

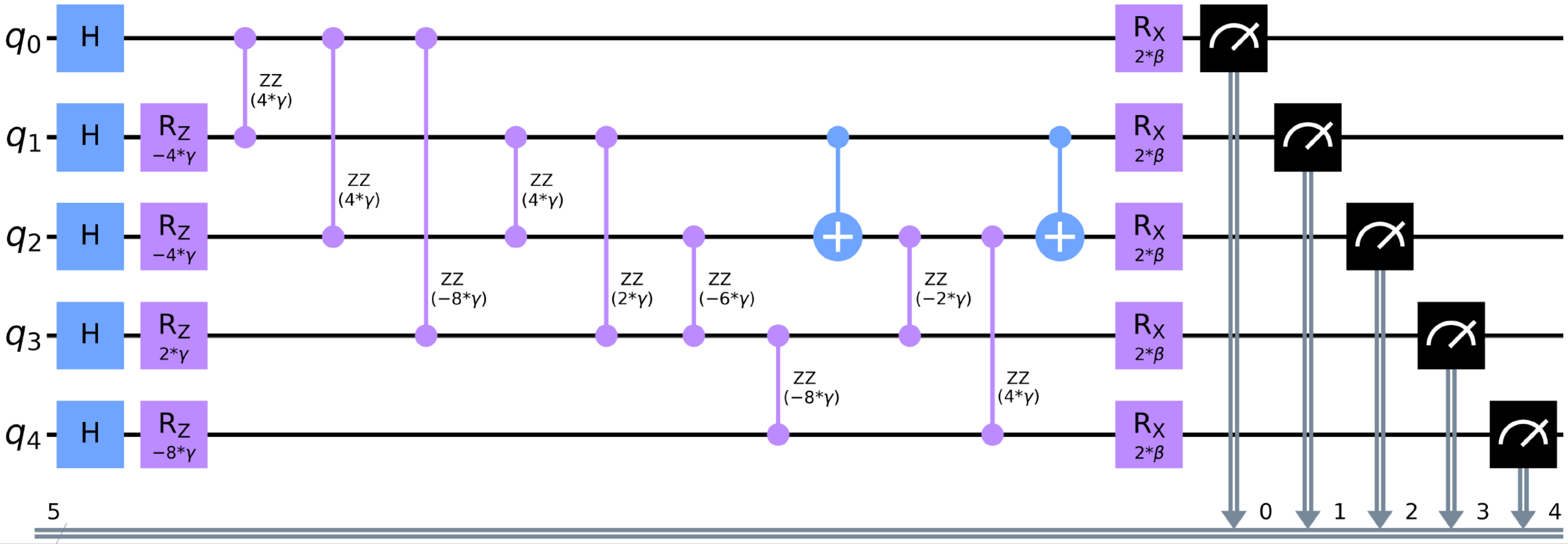
$$
C = (p_1 + r_1 - 1)^2 + (p_1 + p_3 - 2c_{34})^2 + (p_3 r_1 + c_{34} - 2c_{45})^2 + (p_3 + r_1 + c_{45} - 1)^2 =
$$

$$
\begin{aligned}
&= 2 - 2r_1 + 5c_{34} + 3c_{45} + 2p_1 r_1 + 2p_1 p_3 - 4p_1 c_{34} + 3p_3 r_1 + 2r_1 c_{45} + 2p_3 c_{45} - \\
&\quad -4p_3 c_{34} - 4c_{34} c_{45} + 2p_3 r_1 c_{34} - 4p_3 r_1 c_{45}
\end{aligned}
$$

HUBO hands-on examples

The final shape of the cost Hamiltonian:

$$H_c = 18 - 2\sigma_z^1 - 2\sigma_z^2 + \sigma_z^3 - 4\sigma_z^4 + 2\sigma_z^0\sigma_z^1 + 2\sigma_z^0\sigma_z^2 - 4\sigma_z^0\sigma_z^3 + 2\sigma_z^1\sigma_z^2 + \sigma_z^1\sigma_z^3 -$$
$$-3\sigma_z^2\sigma_z^3 - 4\sigma_z^3\sigma_z^4 - \sigma_z^1\sigma_z^2\sigma_z^3 + 2\sigma_z^1\sigma_z^2\sigma_z^4 \qquad \left(R_z(2\gamma) = e^{-i\gamma\sigma_z}\right)$$



HUBO hands-on examples

The factorization of the number 119 is programmed here:

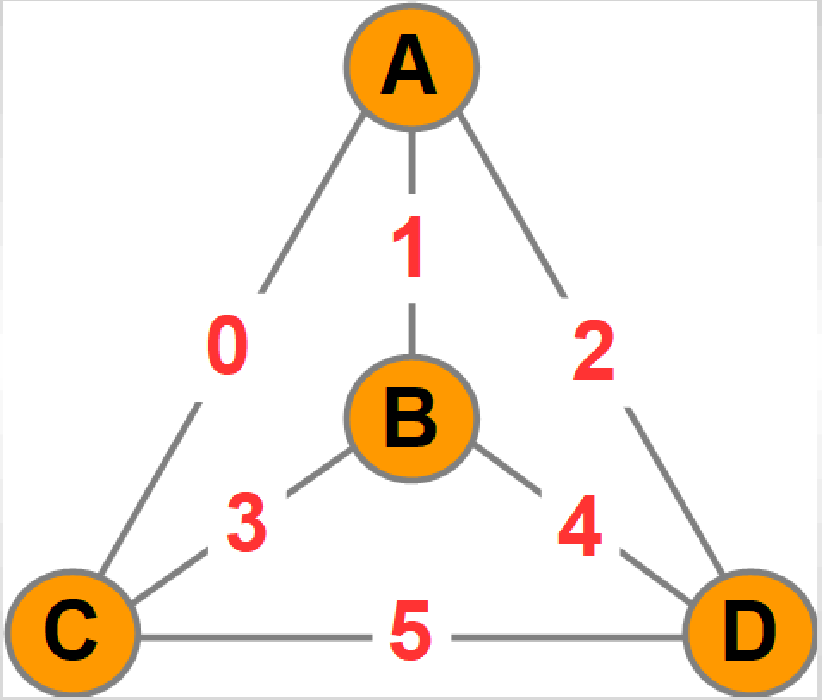
QAOA-HUBO1-Factorization_of_119.ipynb ……… https://1url.cz/V1rpt

HUBO hands-on examples

Last example is optimization of simple network connectivity.

There is a cost to maintain the connection of each pair of nodes in the network.

The optimization task is to find a way to connect all nodes at the lowest total cost.



This problem is solved here:

QAOA-HUBO2-Network_optimization.ipynb https://1url.cz/W1rpe

Thank you for your attention!

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia