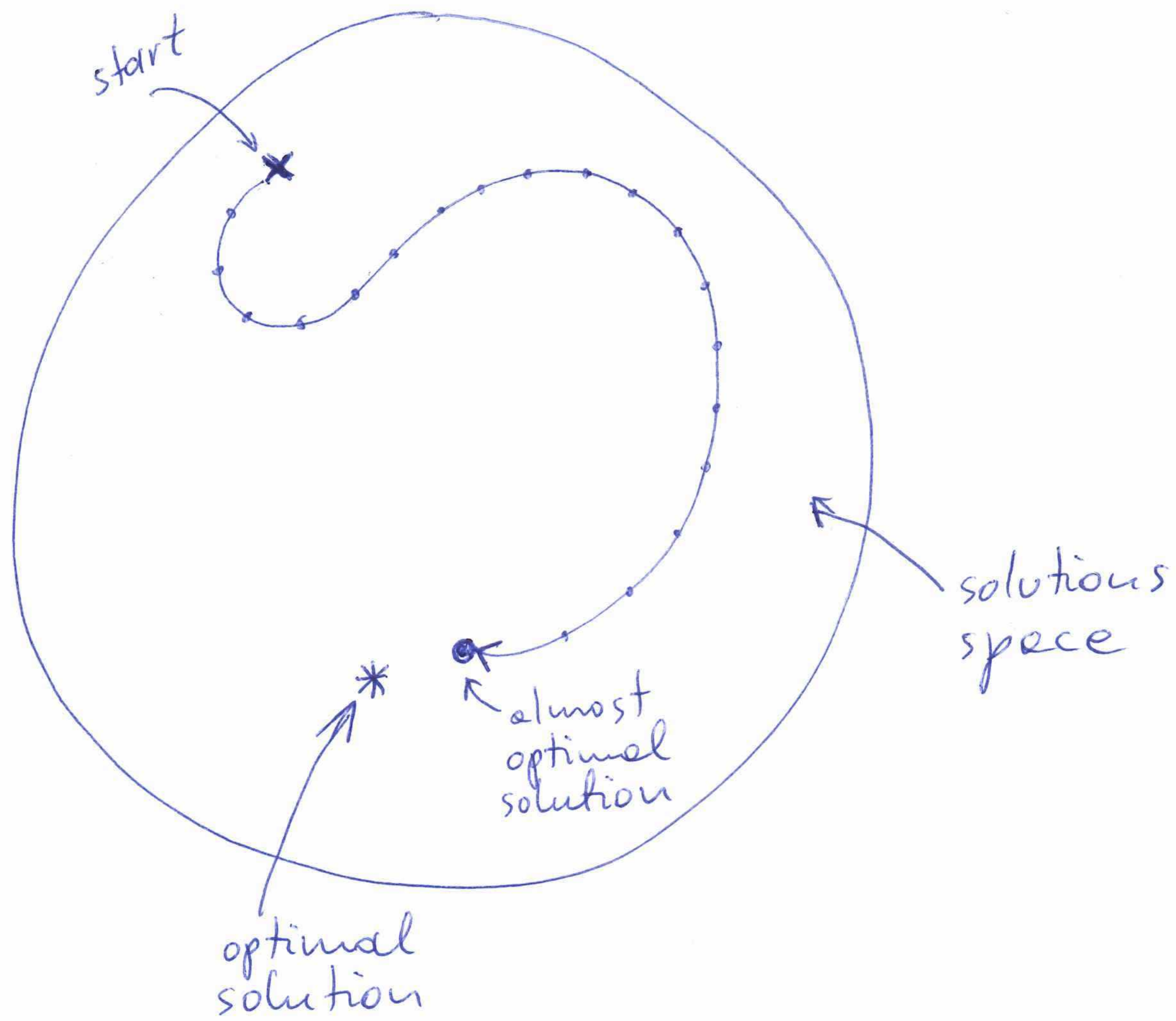
Variational algorithms

- these are hybrid (quantum-classical) algorithms
- we parametrize quantum circuit and then look for values of parameters which best represent our solution
- for that we need some criteria function (objective function or cost function) which needs to be optimized (minimized usually)

  Examples: cost, distance, weight, processing time, energy, etc.
- Having the circuit parametrized, we can forget how the gates work, what are the errors, etc. the circuit works as black box with parameters, which can be adjusted to output the solution.
- QPU is used for doing classically difficult calculations

↓


start
solutions
space
almost
optimal
solution
optimal
solution

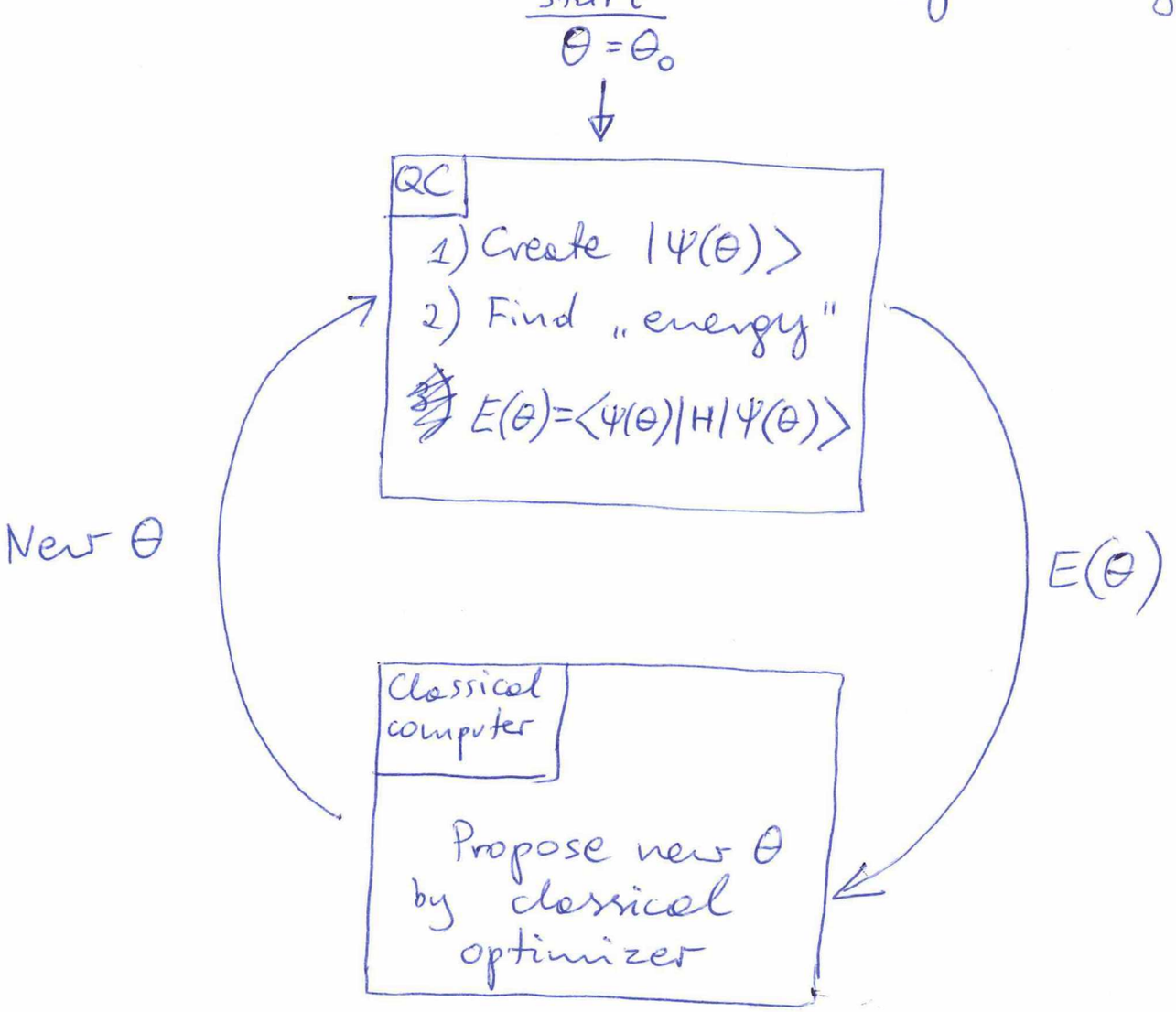
↓

Incept of variational algorithm:

$|\psi(\theta)\rangle$ - parametrized good state

θ - set of variational parameters

We can „minimize the energy" changing θ.



So, we need:

1) Circuit (ansatz) - ansatz ≡ educated guess
2) Cost function
3) Training algorithm (minimizing procedure)

↓

Two most famous variational algorithms are VQE and QAOA

VQE ≡ Variational Quantum Eigensolver

QAOA ≡ Quantum Approximate Optimization Algorithm

Other algorithms are:

QNN - Quantum Neural Networks
QSVM - Quantum Support Vector Machines
VQC - Variational Quantum Classifier

VQE algorithm

- Given Hermitian matrix H we want to find minimal eigenvalue E_{min} associated with eigenvector $|\psi_{min}\rangle$.
- VQE allows us to find $E_{approx}, |\psi_{approx}\rangle$, which fulfill:

$$E_{min} \leq E_{approx} \equiv \langle \psi_{approx} | H | \psi_{approx} \rangle$$

- We look for parametrized circuit $U(\theta)$ which generates state $|\psi_{approx}\rangle$ from some initial state $|\psi_{init}\rangle$

$$\boxed{|\psi_{approx}\rangle = U(\theta)\,|\psi_{init}\rangle}$$

- We iteratively look for θ which will give us minimal E_{approx}
- We need to balance between state general form and reasonable number of parameters
- the not-trivial problem is how to maximize volume of Hilbert space which can be searched

Possible optimizers:

- gradient descent
- SPSA (Simulated Perturbation Stochastic Approximation)
  ↓
  perturbating simultaneously all parameters in contrast to gradient descent
- SLSQP (Sequential Least Squares Programming)
- COBYLA (Constrained Optimization by Linear Approximation)

QAOA algorithm

Quantum Approximate Optimization Algorithm

- QAOA is a special form of VQA which has very specific, fixed ansatz:

$$|\psi(\theta)\rangle = |\psi(\beta,\gamma)\rangle = e^{-i\beta_p B} e^{-i\gamma_p C} \cdots e^{-i\beta_1 B} e^{-i\gamma_1 C} H^{\otimes n} |0\rangle$$

where:

C – problem Hamiltonian (cost Hamiltonian)

B – so called „mixer" Hamiltonian

$\beta_1, \ldots, \beta_p, \gamma_1, \ldots, \gamma_p$ – parameters

p – parameter

- we have $2p$ parameters to optimize
- β, γ are called „angles"
- we need to choose β and γ in such way, that trial state will be as close to eigenstate of the cost Hamiltonian corresponding to highest/lowest eigenvalue
- So, we maximize/minimize

$$f(\beta,\gamma) = \langle \psi(\beta,\gamma) | C | \psi(\beta,\gamma) \rangle$$

↓

Intuition why such ansatz form was chosen:

1) Relation to adiabatic quantum computation

Adiabatic quantum computing is a way of finding lowest energy state by adiabatic (slow) evolution

- we prepare system in initial state
- we adiabatically interpolate it to eigenstate of problem Hamiltonian
- interpolation is given by:

$$H(t) = (1 - s(t))\, H_0 + s(t)\, H_p$$

$s(t)$ - smooth function

$s(t=0) = 0$

$s(t=T) = 1$

$$\Downarrow$$

$$H_0 \longrightarrow H_p$$

in QAOA we have p „time steps" of the analog to adiabatic evolution

2) Time-dependent Schrödinger equation

$$i\hbar \frac{\partial \psi}{\partial t} = H\psi$$

if H does not depend on time

then

$$\psi(t=T) = e^{-i\frac{H}{\hbar}T}\, \psi(t=0)$$

$$\Downarrow$$

$$U(t) = e^{-i\frac{H}{\hbar}t}$$ → time evolution operator („propagator")

this is exactly the form we see in QAOA:

$$e^{-i\frac{H}{\hbar}t} = \alpha \quad \longrightarrow \quad e^{-i\alpha H}$$

Why we need mixer Hamiltonians?

- we look for eigenstate of cost Hamiltonian (C)
- if we would stuck with one of other eigenvalues of C, we would stick with it forever as eigenstate is not changed by time evolution
- Mixer B can not commute with cost C !!!

QAOA example : MAXCUT

- we have graph
- we need to divide nodes into 2 sets in such a way, that there is maximal number of edges going from one set to another

s_i – class of node i

$$s_i = \begin{cases} -1 & \text{if node } i \text{ is in one set} \\ +1 & \text{if node } i \text{ is in second set} \end{cases}$$

$$C = \frac{1}{2} \sum_{i,j \in E} (1 - s_i s_j) = \frac{1}{2} \sum_{i,j \in E} (-s_i s_j) + \text{const.}$$

- We need to maximize this operator
- We construct Hamiltonian by mapping binary variables s_i into Z matrices

$$s_i \to Z_i$$

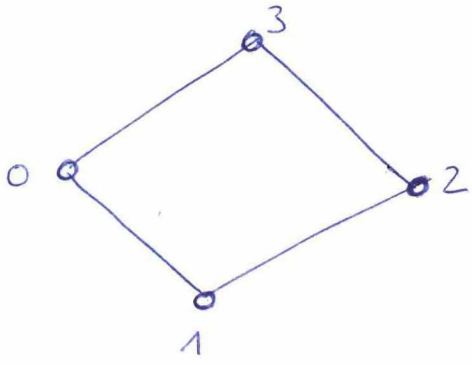
and nodes of graph to qubits:

$$C = \sum_{i,j \in E} (+ Z_i Z_j)$$

↑ switching from maximizing to minimizing

↓

For example such graph:



will give the following cost Hamiltonian:

$$C = \frac{1}{2}\left(Z_0Z_1 + Z_1Z_2 + Z_2Z_3 + Z_3Z_0\right) =$$

$$= \frac{1}{2}\Big(Z\otimes Z\otimes \mathbb{1}\otimes \mathbb{1} + \mathbb{1}\otimes Z\otimes Z\otimes \mathbb{1} + \mathbb{1}\otimes \mathbb{1}\otimes Z\otimes Z + Z\otimes \mathbb{1}\otimes \mathbb{1}\otimes Z\Big)$$

Weighted MAXCUT

- graph edges have additionally weights
- Then we look for maximum sum of weights of edges connecting nodes belonging to different sets

$$H = \frac{1}{2} \sum_{i,j \in E} w_{ij} (1 - S_i S_j)$$

In general we have following mappings:

$$S_i \in \{-1, 1\} \longrightarrow Z_i$$

$$b_i \in \{0, 1\} \longrightarrow \frac{1}{2}(1 - Z_i)$$