Variational Quantum Eigensolver ? && Quantum Approximate Optimization Algorithm?

VQE - helps us to find upper bound of the lowest eigenvalue of a given Hamiltonian

Hamiltonian is the matrix describes possible energies?

Lowest eigenvalue corresponds to ground state energy

System will go to ground state naturally given time.

VQE - helps us estimate the energy of the ground state of a given mechanical system
Estimate :: upper bound of the lowest eigenvalue = ground state
Mechanical system :: described by Hamiltonian = physical system geometry

Lower/upper bound is defined for sets, not single values?
Upper bound of one-element-set that has given value = upper bound of a value?

Knowing ground state is very useful, but hard to obtain using classical computer!
Many calculations in chemistry depend on the finding ground state.
If the ground state has an error, it will then introduce a bigger error in subsequent calculations.

Variational principle?
Start with the equation $H|\psi\lambda\rangle=\lambda|\psi\lambda\rangle$
We don't know what is $|\psi\lambda\rangle$ or λ
So we will try to estimate it !

Get energy $\langle\psi|H|\psi\rangle=E(\psi)$ or $\langle\psi\lambda|H|\psi\lambda\rangle=E\lambda$
Now I would think sample randomly and get lowest?

Let's see …

Ansatz :: intuition :: ansatzes are set of gates … RX RY?
We want ansatzes such that ::

- Cover all states .. as many as possible
- Not many ansatzes
- Less parameters

VQE consists of ::

- Ansatz - prepare the state in order to apply variational principle?
- Hamiltonian * :: we can use only Pauli (X Y Z) and combination of them
- Measurement

Computational basis :: 0 and 1 … Quantum computers can only output 2 values

Some text I am completely lost in?
RY(-pi/2) for X
RX(pi/2) for Y
I for Z
We do these in parallel for every Hamiltonian sum term
Now $|\psi\rangle$ is created cause ansatz?
H = H1 + H2 + H3 + …
Now $\langle\psi|H|\psi\rangle=E(\psi)$ and $\langle\psi|H1|\psi\rangle=E1$
But how is it that we can get $\langle\psi|H1|\psi\rangle$? Why is this an obvious question to ask?
If we measure ψ we can get Hi as an approximation? We might if we choose the right basis? Do that multiple times and average? Measuring will give us always just 0 and 1, but averaging will give us expectation value of measurements?

H = 2*Z + X + I ? Example … {{3,1},{1,-1}} matrix.

1) Choose an ansatz? RY(θ) ? cos(θ) |0> + sin(θ) |1> ?
2) H1 = 2 * Z, H2 = X, H3 = I
3) For Z gates we do nothing
4) For X we need o do RY(-pi/2) gate our circuit?
5) H3 we do nothing too. These are rescaling factors
6) Okay now just do the circuit and measure ?

VQE is Hybrid ?

Start with a regular optimization problem of finding good parameters for the ansatz.
Choose any SGD (stochastic gradient descent), BFGS or Nelder-Mead?
QPU (quantum processing unit) .
Okay so the whole set up is to randomly choose intial parameters and then apply quantum circuit to find energies for chosen parameters?

- Given set of parameters, return a set of measurements?
- Given set of measurements, calculate energy value (average)
- Perform optimization
- Calculate what that new set of parameters should be
- Check if we have reached a minimum?
- Check if we are doing too many iteration? Repeat otherwise.

Okay so all we need to push this through a simple circuit and measure. Good for NISQ.

- We can only use N operation per qubit (circuit depth is limited)
- There are a lot of errors, so we can’t be sure
- Only so many qubits
- Simple ansatzes give reasonable results, but more is better (when we have more qubits)
- Chosen ansatzes can be simple enough to avoid bad errors, tailored.
- Some industries actually don’t need that many qubits

Michał Stęchły

PART2: QAOA

Quantum Approximate Optimization Algorithm

Video : Eddie Farhi 2015

Combinatorial optimization

n bits and m clauses
Cost function is the sum of sub cost function for each clause, sub-cost functions can be {0,1} satisfies and not.
We would like to find maximum of the cost function first and then we want ot get approximation ratio : ratio of cost function and max subcost. We want to maximize this.

You might be happy enough with not the best approximation ratio and we can just find a good one.

Let's look into quantum algorithm?
Cost function is a unitary. Initial state superpositions are also needed. Those unitaries we can construct, all of them?
So we act on the initial state with all those unitaries? And we end up in the end state.
So all we need is to construct a circuit, at worst it is m*p+p depth, but could be reduced.

Mp is a next max cost function. Mp >= M(p-1) so limit p->inf Mp = Cmax and Cmax is the maximum cost function.
Slowly going towards maximum function? Adiabatic path? For a very long time if we could run we would end up in the max cost state.

Now obviously we cannot run forever our algo. So we are going to keep p low.
Fix p and make an initial state.
Compute and measure. Do it enough times to get an estimate. Update? Change angles? Gradient descent? Any routine that searches for maximum. These functions don't have exponential peaks.

Another way to do it. We can classically pre-process to find best angles to measure. For a fixed p we can do that. MaxCut on a 3-regular graph. Every vertex has 3 edges. Now we can create cost function. Vertex is has a spin 1/-1. We want to separate them to maximum. MaxCut is a well studied problem. So we figure out angles.

Some impossible 3 possible subgraphs and commuting/conjugations and formulas of j and k?

Each subgraph depends on the gamma and beta. Classical computer 4,5,6 qubit system can be looked into. We can evaluate those. Add them up classically and determine best angles. We run quantum computer to produce the state with those angles and measure.

We get 0.6924 optimal cut at p = 1.
Now quantum did all instances of 3-regular max cut. So just running once will give all answers. Not better than classical for this problem. Random guessing was first idea, but then it was improved classically to 0.85 … 0.9 for 3-regular.

Now if we go up in p, increase p. For p=2 we can get quantumly 0.7. Classically we can preprocess independently of N. So it does not matter, but getting maxCut classically is hard.

QAOA approximation ratio is (2p+1) / (2p+2) independent of N. Note for p = 1, we get ¾. Every string output will do it's best ¾ , this algorithm always produces best upper bound outcomes, it does not produce bad outcomes at all.

Find a string that maximizes the number of satisfied equations. NP hard to find optimal solution, but we can find a good one using quantum. (½)

Bound occurrence: every variable is no more than D equations. (½ + const/D)

Quantum Algo : (½ + const/(D^(¾))

Classical Algo : (½ + const/(D^(½)) Sneaky approach, string has to exist so we force condition.

Now QAOA : (½ + 1/(cD^(½)lnD))

Typical : (fixing factograph) : (½ + 1/(2sqrt(3e)D^(½)))

Can we get rid of the lnD ?

If there is exists algorithm that does (½ + const/D^(½)) with large constant we get P=NP

Run on the small Quantum Computer, industry were getting interested in it 2015 lul
QAOA : simple gates and low circuit depth, study fault tolerance.

Okay back to the mustythoughts.

Combinatorial optimization problem (COP). Combinatorial problems depend on O(n!), compared to the O(a^n) for the exponential problems.

Continuous optimization requires to deal with infinite number of points (real numbers)
Okay there bunch of steps, but it's easy

Steps:

- Create a cost hamiltonian Hc and operator U(Hc, gamma) = exp(-igammaHc)
- Construct Hb = sum(j=0,n){sigma(j,x)}
- And the operator U(Hb, beta)
- Now QAOA just does state |gamma, beta> = U(Hb,betaP)U(Hc,gammaP)....U(Hb,beta1)U(Hc, gamma1)|s> where P is number of steps and |s> is the initial state, which is usually |0...0>
- Beta and gamma are angles
- So that's it, just start with random parameters, measure, update!

wHY shOUsLD IT eVEn wORK at All?

Adiabatic Quantum Computation (AQC)

Quantum system with a simple Hamiltonian Hs and known ground state.
And now we want to find ground state of the Hc?
H(alpha) = (1-alpha)Hs + alphaHc. For alpha = 0 we have Hs, for alpha = 1 we have Hc …

The theorem :: if we slowly go from alpha 0 to alpha 1 that is going from Hs to Hc very slowly and having Hs being in the ground state, the whole process H(alpha) will be in the ground state. Therefore, when alpha = 1, we are still in the ground state, but now H(alpha=1) = Hc. Awesome.

Ising model

Big assumption in the previous paragraph is that we can do Hc ground state, but what if we cannot do that? AQC then would be very useless.
But big help comes form Ising model.

Chain of particles with different neighboring states
And Hamiltonian would be:
H(sigma) = -sum(i,j){-Jijsigmaisigmaj)
Sigmai is the spin of the i-th particle and sigma is a string. Jij is the strength of the interaction of between particles. Nicely it works in lattices too. If we change the sign to positive next to Jij hen we would get same spin particles.

This model can be mapped on quantum computer? So we do sigmai to sigma(i,z)
Actually we can encode a lot of COPs using Ising model. It is nice that we only need to use sigma(z) (Pauli Z gates basically).

Now. Problems with AQC?

- AQC needs to be isolated from the outside, which is really hard in practice.
- It might require a very long time to run
- Not a gate-based computer friendly.

Basically, the bigger your problem is, the harder it is to isolate and longer it is to run :(

Maybe quantum annealing? (QA)

The idea is QA is the same as AQC, start with some easy Hamiltonian and slowly evolve your state towards the ground state of an interesting Hamiltonian. So what's the difference? QA is an imperfect implementation of AQC? It trades some AQC's power for easier implementation ?

- QA can be flexible
- Operates at finite temperature.

Time evolution: Shrodinger equation has a time solution for a state basically saying we can act on the ground state with exp(iHt) and we will arrive at some phi_t. It is okay, but also remember that Hamiltonian H isn't actually time independent for AQC (H(a) = (1-a)Hs-aHc
With a = t, H has time dependency of its own.

Trotterization :
Basically approximation of the ground state of the Hamiltonian.
Time evolution operator U(t) is actually U(t,t0). The bigger number of those U(t) we use the better the approximation (cause there would be shorter t-t0)

Hmm, trotterization is just an expansion of the exp(A+B) = lim(n->inf){exp(A/n)exp(B/n)^n
We can get approximate A+B by applying alternatively A and B for time intervals of 1/n.

Now we get back to QAOA ::

QAOA creates the state:
|gamma,beta> = U(Hb,betap)U(Hc, gammap) … U(Hb, beta1)U(Hc,gamma1)|s>
Where U(Hb,beta) = exp(-ibetaHb)
Where U(Hc,gamma) = exp(-igammaHc)

Now notice that UbUc look a lot like trotterization of exp(Hb+Hc), but is not pure trotterization, it resembles piecewise approximation thought. What kind of approximation is it?

Another point: QAOA mimics adiabatic evolution from the ground state Hb to the ground state Hc (like Ising models?). QAOA could be viewed as the approximation to the AQC.

What is the actual relation between QAOA and AQC?
Why should QAOA work?
Why do we actually need Hb?

Relation to AQC? :

Differences: AQC goal is to get the ground state of the Hamiltonian. QAOA goal is to get approximate ground state? We could choose angles for QAOA that the evolution would mimic

AQC, but we don't do that. In QAOA we can choose angles arbitrary using classical optimization strategy? AQC will starts from Hb and move towards Hc, but in QAOA we alternate between Hb and Hc?

Why should it work? :

QAOA could choose an infinite number of steps and get angles which exactly mimic adiabatic paths. Universal approximation theorem? UTA says if you have a one-layer neural network it can approximate any function, solve any problem, given that it is big enough. For one layer neural network has to be so big that it's impractical. So we just stack more layers. QAOA is similar in that regard is that, we could be just increasing number is steps and it will give us answer, but in practice it is not practical. Instead we try to solve with fewer number of steps, update and try again.

We don't know if it will work? No one knows if quantum computer will ever outperform classical on a combinatorial search problem.

Why is Hb important?

Hc corresponds to the problem we want to solve and hence we know why it looks the way it looks. (remember Hc is a cost Hamiltonian?)
Why do we choose sum(i, N){sigmai^x)

Well, maybe it doesn't need to be sum(i,N){sigmai^x}, it does not, but this particular choice does not commute with Hc and it is easy to implement (just a layer of Rx gates).

So, just by applying Hc we might end up in some local minima and we would never get out of it? Thus we use Hb to push it out of it sometimes?

Some circuit example.

Relation to VQE? :

VQE good for chemistry and QAOA for combinatorial optimization. QAOA is a special case of VQE?
QAOA has ansatz that is limited in form.
QAOA is using restricted types of Hamiltonian (Ising Hamiltonians)
QAOA tries to find the solution to a problem posed, not to find some ground state of a Hamiltonian for a chemistry problem.\

Okay

****

Ruslan Shaydulin :: QAOA ::

Complexity :

- P - solvable in polynomial time
- NP - can verify the solution in polynomial time
- NP-complete - there is no polynomial algorithm (Maxcut)
- PSPACE - solvable in polynomial time (and unlimited space?)
- BQP (bounded-error quantum polynomial time) - solvable on quantum computer in polynomial time with an error probability of at most ⅓ for all instances (somewhat similar for classical P class, but for quantum)

We don't know yet the relationship between NP and BQP
P < NP, BQP < PSPACE
Maxcut generally is a NP-complete (worst case)
Maxcut on cycle graph is a P problem

Algorithms with proven performance:

- Matrix multiplication
- Dijkstra's shortest path
- Shor's factoring
- Grover's search

Heuristic methods:

- Gradient descent (non-convex problems)
- Simulated annealing
- Genetic algorithm
- Quantum annealing
- QAOA
- More?

Many of the most powerful classical algorithms are heuristic. NISQ hardware provides an opportunity to try quantum heuristic algorithms

Paradigmatic (typical) example? Maxcut
General rules of construction of Hamiltonian

Maxcut problem!:

- Goal is to split vertices into two groups with maximum edges between them.
- We can color into -1, 1
- Then we can do this: ans = max(sum(1-si*sj)/2) to get the solution to maxcut
- If si != sj then there is 1 contribution, and if si == sj there is not contribution

How to map into a quantum computer?:

- So we map max(C) where C = sum(1-si*sj)/2 into Hamiltonian C.
- Solution will be some set s = {-1,1}^n, but in quantum we get highest energy eigenstate |s> (largest eigenvalue eigenvector)!?

How does Hamiltonial loos like?:

- Basically a diagonal matrix. H acting on a |s> whilld just do H|s> = f(s)|s>
- Problem is that it is too large to construct explicitly? Also very silly of an idea. But in quantum you can have all that constructed more compactly.

How to construct Hamiltonian then?:

- Objective we have this: max(sum(1-si*sj)/2)
- So we need to do is C = sum(I-Zi*Zj)/2
- I am so confused here, why this is the case but we are saying we are mapping binary numbers si onto the eignevalue of Z? Makes no sense but ok
- Is this true? C|x> = C(x)|x>?

Most important tool, Z?:

- Pauli Z operator Z = [1,0; 0,-1]
- Z|0> = +1 |0>
- Z|1> = -1 |1>
- What does Zi mean? So we have n+1 qubits {0,n}? And ‘i’ is an element of {0,n}? Zi means a Z gate acting on the ‘i’th qubit.
- Zi|x0,x1,...,xn> = I*...Zi…*I|x0,x1,...,xn>
- Zi*Zj|x0,...,xn> = I * … I * Zi * … I * Zj * I … * I|x0,...,xn> = (-1)^xi * (-1)^xj|x0,...,xn>

Verify Maxcut Hamiltonian?:

- Do substitution from si -> xi then. Where xi = (1-si)/2;
- Si = +1 -> xi = 0 ---------- (-1)^xi = +1 = si
- Si = -1 -> xi = 1 ---------- (-1)^xi = -1 = si
- So we work with xi instead
- max(sum(1-(-1)^xi*(-1)^xj)/2) where xi {0,1} bitstring

- Basically the gates sum(I-Zi*Zj)/2|x0,...,xn> does give us the max(sum(1-(-1)^xi*(-1)^xj)/2)|x0,...,xn>

What is the size of Hamiltonian C?

- C = sum(I-Zi*Zj)/2
- Each term in the sum is 2^n*2^n matrix if written explicitly, but we don’t write it like that

- We only do gates in order of application Zi*Zj so it does the same effect as the whole matrix Hamiltonian just doing it in sequence instead and no need to explicitly get the matrix. But couldn't we do the same classically?

Constructing a Hamiltonain for a general problem?:

- C|x> = f(x)|x> where x {0,1}^n bitstring
- How can we do this for an arbitrary function?
- How do we map boolean and real functions to diagonal Hamiltonian?
- Let's work with a simple example first. F = max2
- We know we can always change variable from si to xi maybe?
- But we can also represent functions as multilinear polynomial
- For max2(x1,x2) = ½+½ x1 + ½ x2 - ½ x1x2
- Which is amazing? Well any function can be mapped!
- The reason is f(x) = sum(f(a)*I(a,x)) for a = {1, -1}
- Where I(a,x) is an indicator polynomial
- I(a,x) = (1+a1x1)/2*(1+a2x2)/2…(1+an*xn)/2
- If a == x it is 1? Otherwise I(a,x) = 0
- Remember a = {1,-1} so there we go
- Because of this expansion we can always recover multilinear polynomial of any function
- We can do some Fourier expansion, hard, but there is always a unique expansion?

Okay

- So Classically we have this f(x) = sum(f(a)*I(a,x))
- But Quantumly we have C = sum(f(x)|x><x|) where |x><x| = Pr(x)
- There is similarity of I(a,x) ~ Pr(x)
- Now Fourier expansion of Pr(x) , we just need to do prod(Zi|x>)
- Awesome? General recipe is P-hard to construct Fourier expansion
- However, there is a table of simple boolean building blocks we can do that Fourier expansion transformation (Hadfield, Stuart, 2018)

QAOA:

- QAOA prepares a parametrized trial (ansatz) state of the form: exp(B)exp(C).... H^n|0...0>
- Classical optimizer than varies gamma and beta parameters
- With infinite p QAOA approximates adiabatic quantum evolution.
- For small p we think there is a potential. Evaluating energy is not that hard. Sampling isn't easy though still. There are papers indicate there is a good chance.

How do we implement this circuit?:

- We need X,Y,Z,H,S,Rz=exp(tetta*Z)

- If we have exp(ZZ) = exp(Z*Z) where * is a tensor and Z*Z apply on different qubits
- In the end though exp(Za*Zb) just adds phase exp(-1^(a+b))
- Aka, parity?
- And because ZZ commute we can put them in any order and exp(ZZ) = exp(Z)*exp(Z)
- Thus they can be simulated sequentially
- Therefore, we can do the whole exp(C) thing = exp(sum(I-ZZ))

Implementing QAOA:

- exp(B) = exp(sum(X))
- Trick to use is HZH = X so we can still see Hexp(Z)H = exp(X)

Geometric interpretation?:

- Classical optimizer varies the parameters beta and gamma to push towards the target state. Somewhat similar to Grover's algorithm, but a random walk?
- How do we really change beta and gamma? Well we better use a very good classical optimizer!?

Adiabatic Quantum Computation:

- The idea is inspired by adiabatic approximation theorems.
- The basic idea is that starting from ground state we can change Hamiltonian to another ground state, the target.
- We need to do is slowly enough!
- Interpolation is time-dependent H(t) = (1-s(t))Hd + s(t)Hp where is Hp is the target?
- s(t) must be smooth and slow to be valid.

How to apply AQC to maxcut?:

- Hd = sum(X)
- Hp = C = sum(I-ZZ)
- S = |+> also known as ground state?

How to simulate this?:

- Okay so we need shrodinger
- |phi(t)> = U(t)|phi(0)>
- Shrodinger: i(dU(t)/dt) = H(t)U(t)
- The solution here is exp(sum(H(t)dt))...
- For H, time independent the solution is: exp(H)

Problem is that some operators don't commute, use Trotter?:

- lim{n->inf}(exp(A)exp(B)/n)^n = exp(A+B)
- We can discretize U(t) to be exp(sum(H(t))
- With small enough dt we get something nice
- prod(exp(1-s)exp(Hd)exp(s)exp(Hp))
- So the whole point is to show that AQC is equivalent at the limit and adjusting the parameters beta and gamma form the QAOA to be (1-s)dt and sdt respectively
- Beta = (1-s)dt and gamma = sdt
- The connection is similar, however, there is quite a big difference.
- QAOA is more closely related to Grover at small p? Also QAOA doesn't have to choose beta and gamma to be the AQC parameters

Evaluation QAOA energy classically?:

- If we could classically sample efficiently quantum C matrix then there wouldn't be a point for quantum algorithm. Sometimes it is possible.
- For p = 1 the sampling is the sum for every j,k
- There is however a bound and a way without looking through the whole empty graph, but just meaningful edges and solution is more constant.
- Oh the idea is that we don't need to simulate a lot of qubits for a shallow depth QAOA
- Therefore we can do it classically! Awesome! Up to 6 qubits for p = 1.
- however , for p > 1 there are more terms that touch j,k edges which we need to simulate and that will increase qubits that we need to simulate classically! So for more depth we might need QAOA and do we quantumly not classically!

Summary!:

- QAOA does not solve NP-complete problems
- QAOA is a promising heuristic for NISQ era
- For any boolean function we can construct a unique n-qubit Hamiltonian representing it!
- QAOA is deeply connected to the AQC
- QAOA energy can be often evaluated classically for small p?