Quantum kernel methods for graph-structured data

Yanting Teng, Stefano Troffa, Hrushikesh Patil

Qhack 2023
Power-ups: AWS, NVIDIA

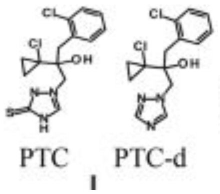
Ref: dataset add link, toxicity

Motivation: graph classification

What we thrive to implement was the task of binary classification on graphs. The datasets of this type is constituted by a set of nodes and edges, which may or may not each have labels or other attributes.

We picked the PTC-FM dataset, containing graph labels based on carcinogenicity on rats. Being able to compute a binary classification in such compounds is a relevant task either for drug discovery purposes and for health safety concerns. A procedure that works on this dataset should be easily extendible to similar problems

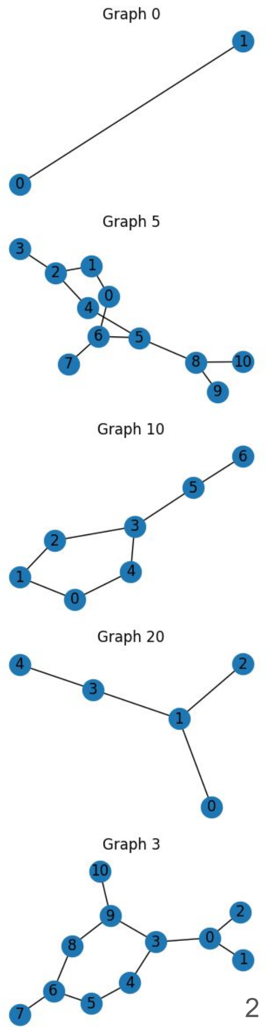
Important for carcinogenic compounds identification!



toxic??

Binary classification

Mathematically, given two graphs, G_A, G_B assert if they are "similar". Similarity is defined through a kernel function: $K(G_A, G_B)$



Part I. QEK

What is quantum evolving kernel (QEK)

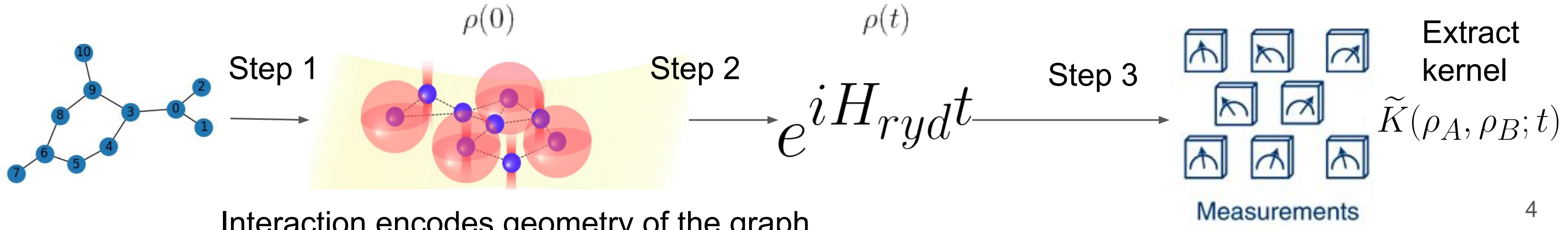
Ref:
braket docs,
QEK ref,
kernel methods

QEK is a kernel computed from two quantum states by:

1. Embed graph onto **neural atoms** locations in the simulator (init 0s)
2. Time evolve the initial state with H_{ryd}
3. Take snapshots and compute kernel $K(G_A, G_B) = \widetilde{K}(\rho_A, \rho_B; t)$

$$H = \frac{1}{2}\sum_{\ell}[\Omega\sigma_\ell^z + \underbrace{\delta\sigma_\ell^x}_{\text{transverse field}}] + \frac{1}{2}\sum_{\ell\neq\ell'}\frac{V_{\ell,\ell'}}{4}\underbrace{(1-\sigma_\ell^x)(1-\sigma_{\ell'}^x)}_{\text{Rydberg blockade}}$$

magic is in the interaction!
geometrically local on graphs



Interaction encodes geometry of the graph

Ref: braket docs, 2211.16337, Pennylane hamiltonian from graph

Implementation of QEK

We implement this algorithm using

1. Braket analog simulator/Aquila: scalable
2. PennyLane digital trotterization of H$_{\text{ryd}}$ to as a variational circuit

Advantage: current device at QeEra can scales to ~100 nodes!

Analog aquila → less controls so less gates!

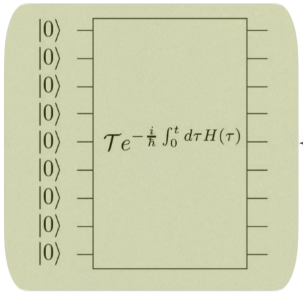
PennyLane-Braket → faster digital computation

Kernel:

1. Pick observable $O = \sum_i \sigma_i^z$
2. Compute distribution $\mathbb{P}_O(\rho_A(t))$

$$\widetilde{K}_{A,B}(t) = e^{-JSD(\mathbb{P}_A|\mathbb{P}_B)}$$

Aquila

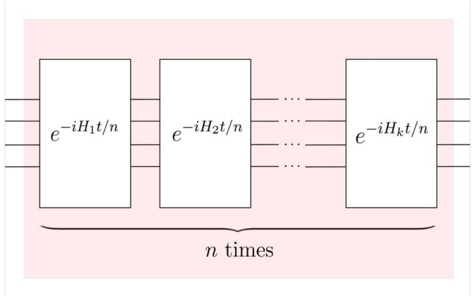


Analog quantum simulator

pennylane-braket plugin!

$$\longleftrightarrow$$

PennyLane



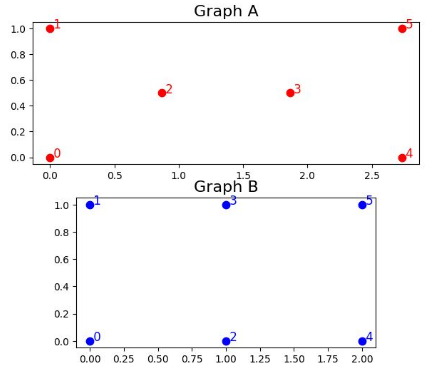
Digital simulation given a H$_{\text{ryd}}$

Ref: notebook link

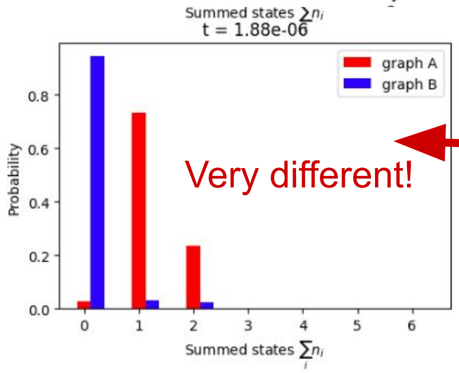
Demonstration (QEK with Braket)

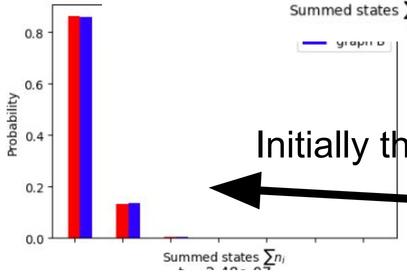
Given two distinct graphs G_A and G_B → **learn difference!**

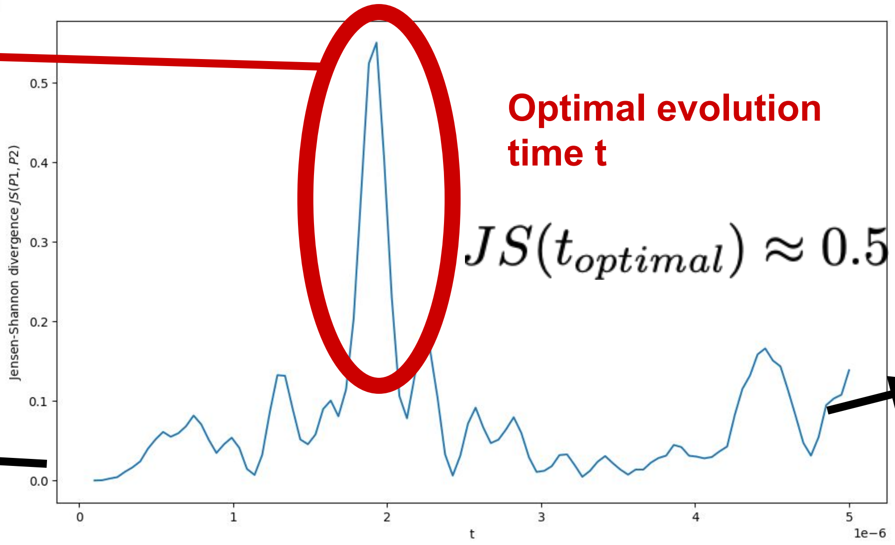
observable $\hat{O} = \sum_i (1 + \sigma_i^z)/2$

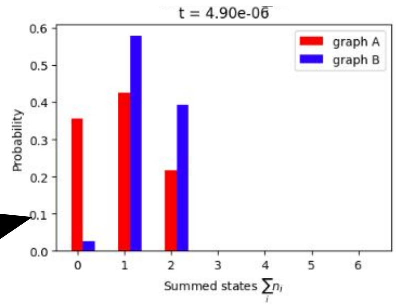


$\mathbb{P}_O$





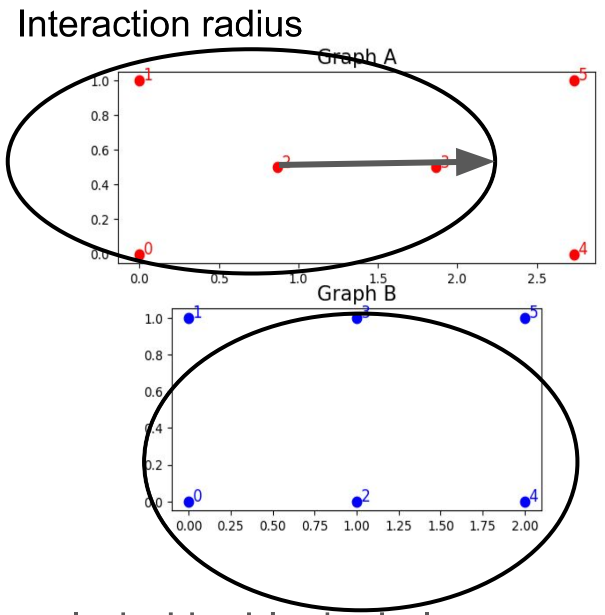


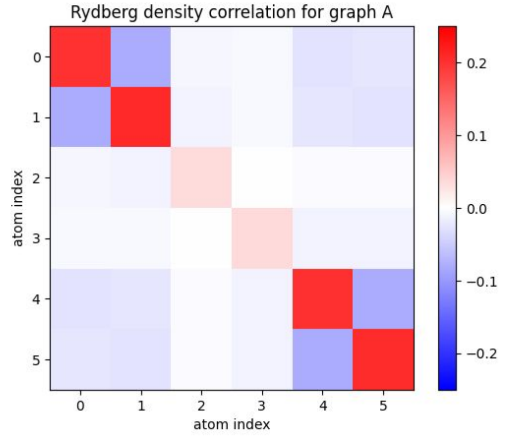


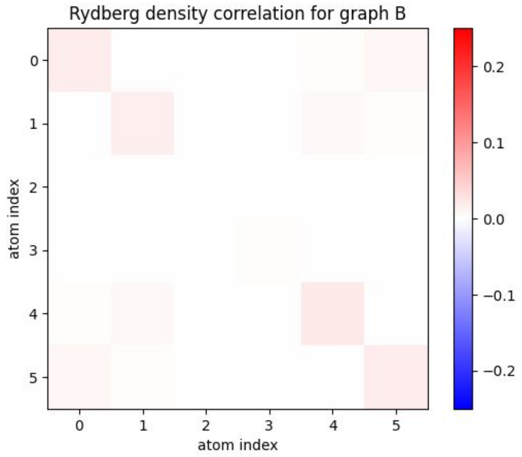
Ref: notebook link

Intuitions from graph geometry

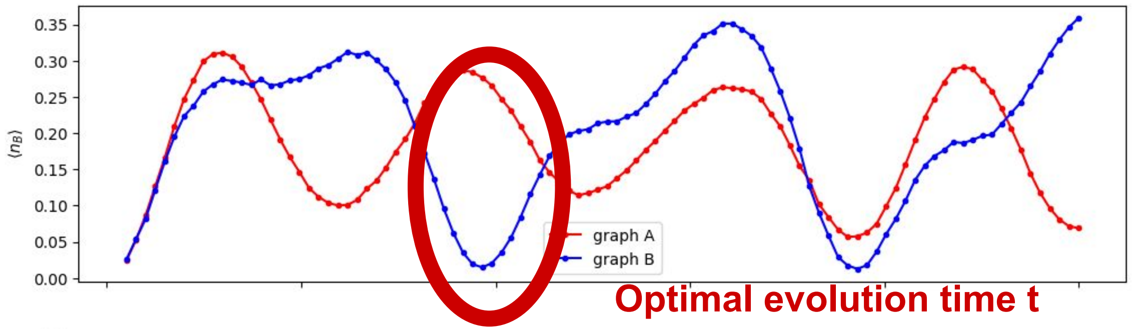
Interactions in the middle are distinct!







This is average qubits excluded by black circle



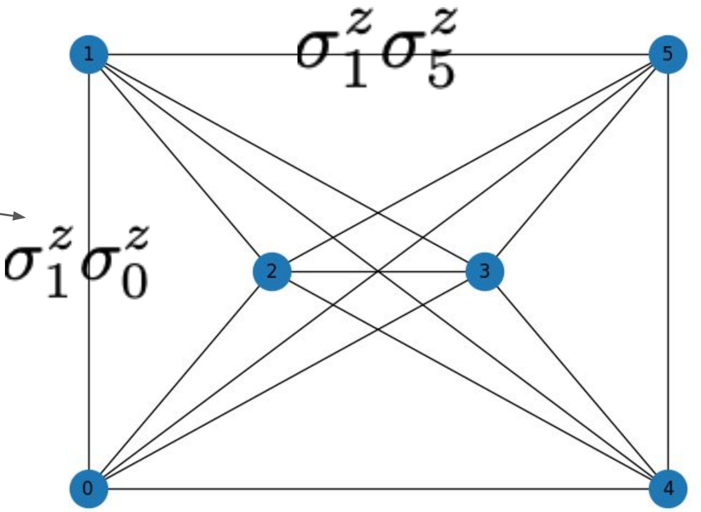
Optimal evolution time t

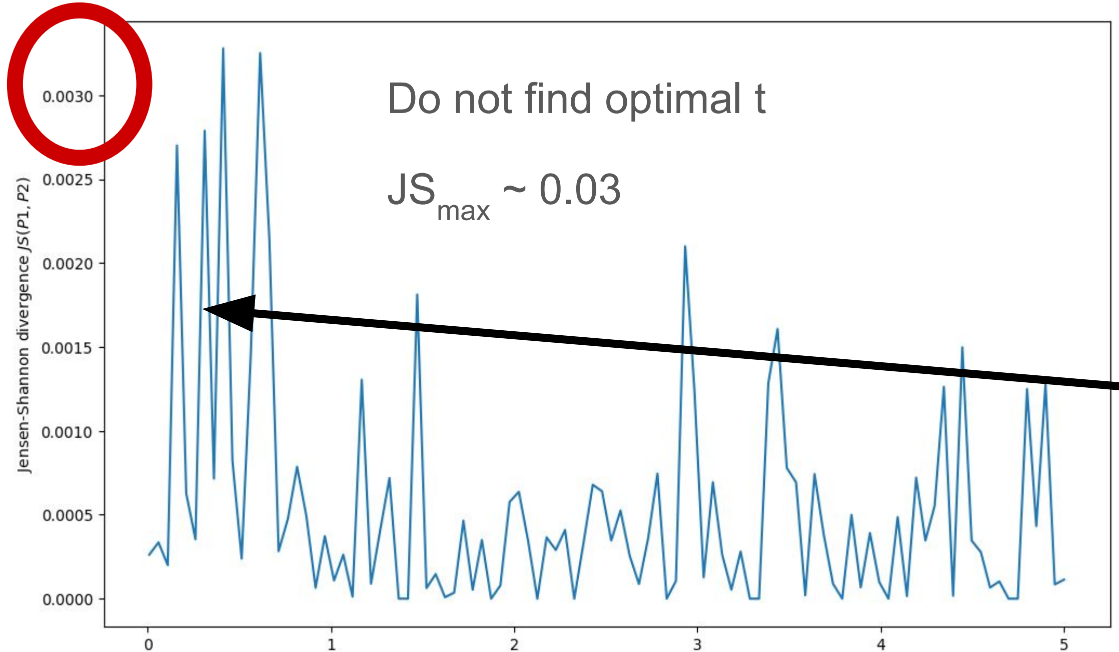
Ref: notebook link

Comparison (QEK with pennylane)

Trotterization of hamilotonian with interaction *connected by edges*

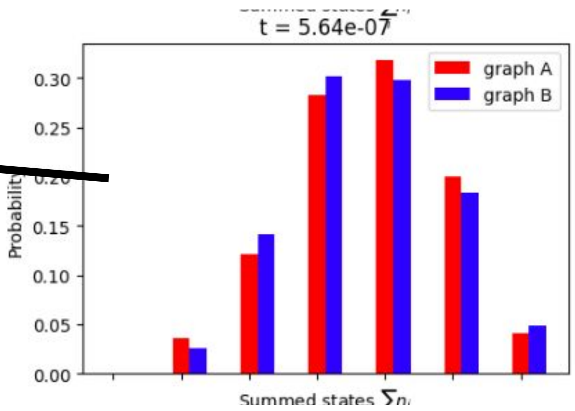
Suggest long range interactions are important! (decaying with distance 1/r^6)





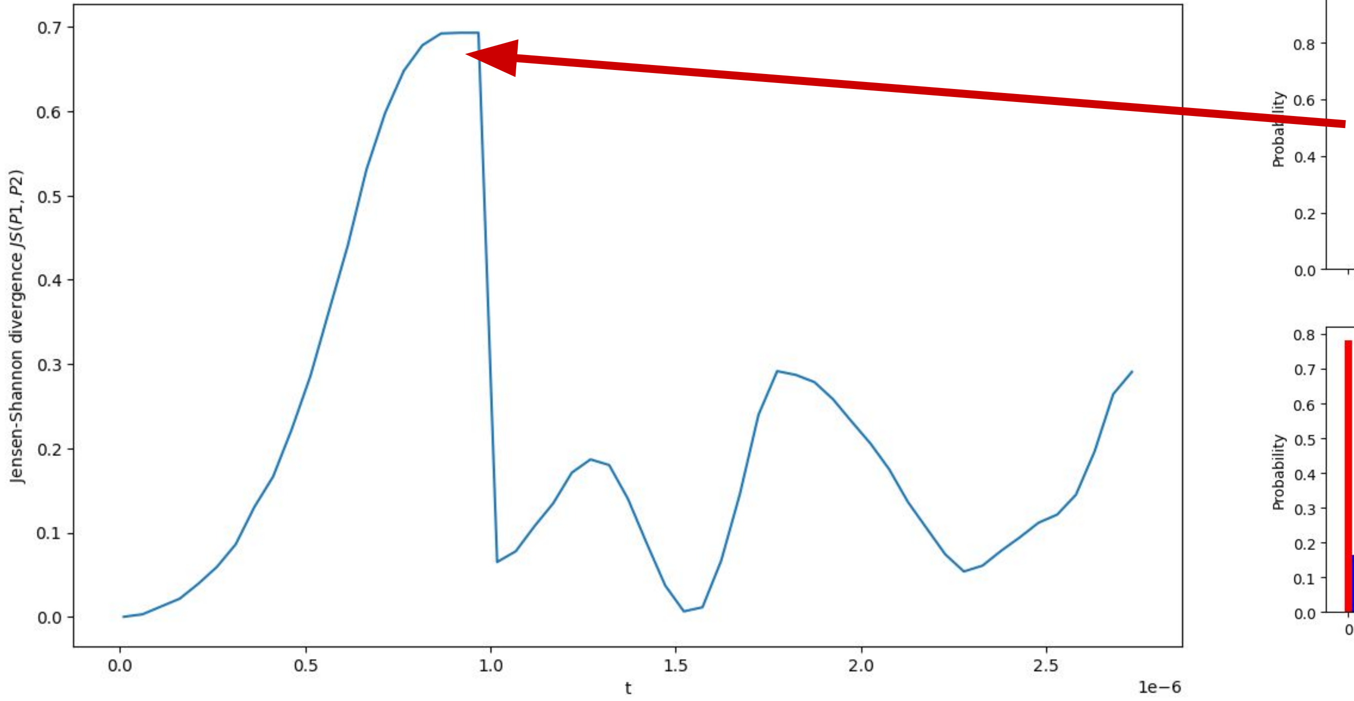
Do not find optimal t

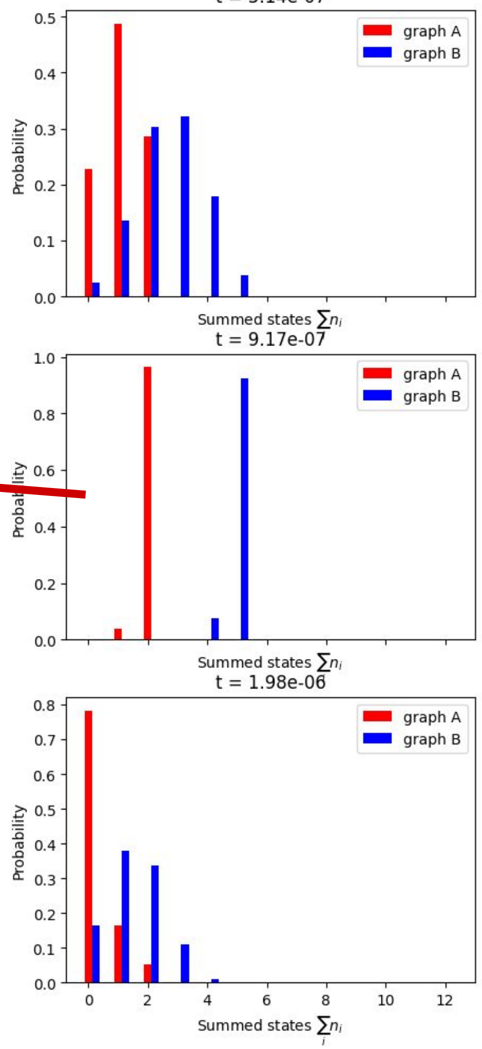
JS_{max} ~ 0.03



Preliminary results on PTC dataset

We find optimal time for two sample graphs from PTC dataset





Part II. DQGNN

Quantum Graph Neural Networks

QGNNs were first introduced in 2019 by Guillaume Verdon and Trevor McCourt. The idea behind these Quantum Neural Network ansatze is to provide a natural representation of Graph Data. A vanilla procedure consists in assigning an Hilbert Space to each node and use the graph structure to establish coupling between nodes. In the simplest scenario, we map:

- Each node v, to a qubit
- Each edge (i,j) to an entangling gate between nodes i,j

Credits to [4]

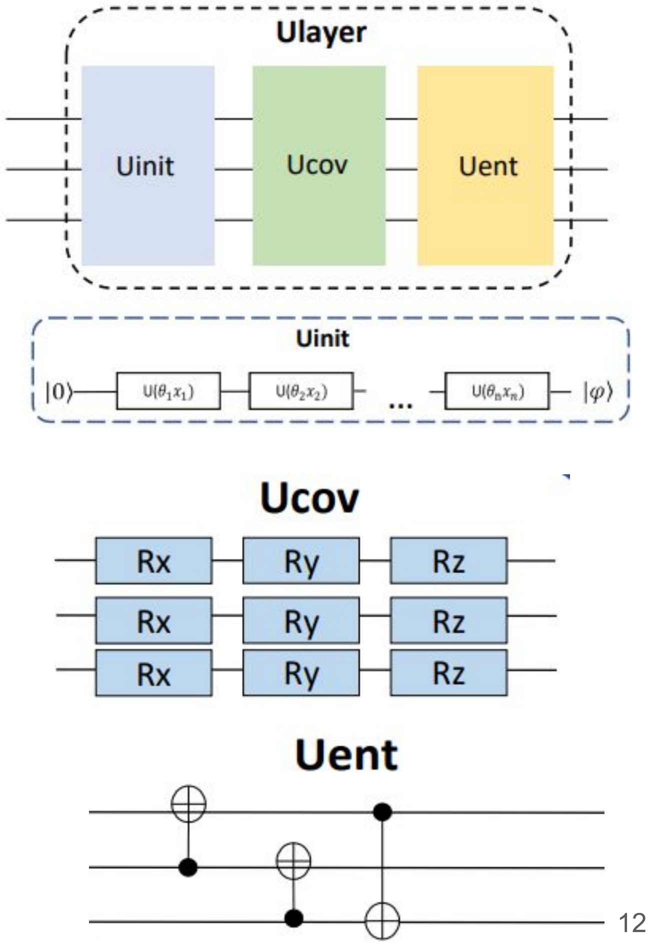
Decompositional QGNN:

The idea behind this ansatze is to reduce the dimensionality of the Hilbert Space via decomposing the graphs in subgraphs, i.e. nearest neighbours graph for each node. Then we process the information by applying each subgraph representation as a subsequent layer to our circuit.

The structure of the algorithm can be seen on the right

The initialization is used to encode information about each node. Then there is a layer of rotations, followed by entanglement through CNOTS.

We modified the layer Ucov to account for entanglement following the adjacency matrix of the edges.



Results from DQGNN

Our objective was to confront QPU implementations on rydberg atoms, with the DQGNN strategy on simulators and real device.

Due to the complexity of the algorithm, and time constraints we did not manage to gather results on the classification capability, and expressivity of the network.

However it is believed that the notebook presents a clear core structure that allows for modifications, thus it can be use for didactic reasons and provide a clear way to work with graphs and decompose them and insert the results into a quantum circuit.

Despite the simulation not be completed and still the necessity to fine tune some parameters it is believed that the procedure as a whole may be of help for who wants to study large graphs through decomposition strategies. Implementations are needed to deal with very large graphs due to the coherence time of devices bottleneck. Specifically it is needed to encode the information after

Conclusion and future works

- Our goal is graph classifications (local geometries, global connectivity, etc.…), which is important in drug discoveries
- We implemented QEK algorithm using

  1) Braket/Aquila and 2) Bracket-PennyLane

We find the native interactions in neural atom simulators are important in differentiating graphs that are geometrically different!

- We also implemented (D) QGNN architecture in Pennylane, as a fully personalizable algorithm.

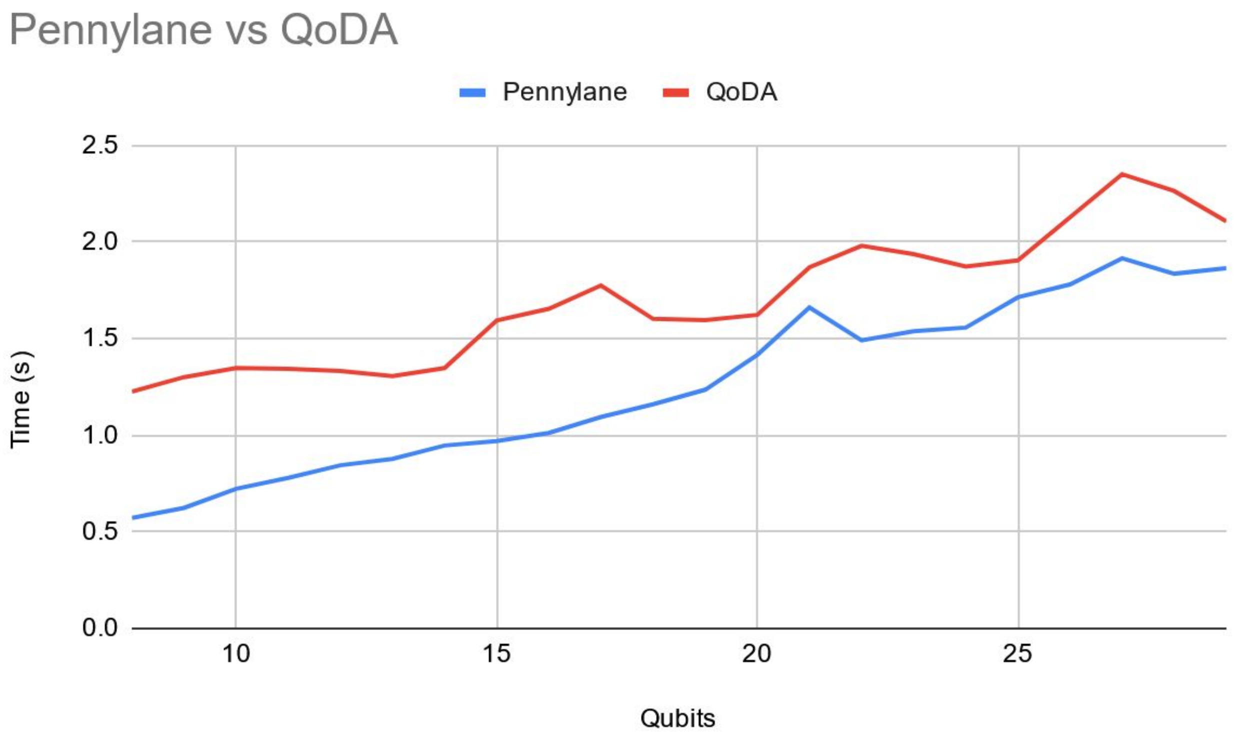
Work in progress

- We are currently training and analyzing the full PTC-FM dataset. We plan to update the results soon.
- We are also working on D-QGNN, and we plan to compare this decomposed quantum circuit with that of the trotterized QEK.

Implementation with QoDA

- One of the ideas we had proposed was to use CAFQA[2] to accelerate our optimization procedure.
- CAFQA substitutes the gates in an ansatz with clifford gates by changing the initialization parameters to 0, pi/2 etc.
- To accelerate the classical simulation we proposed use of QoDA and GPU’s.
- While we were not able to finish CAFQA procedure, as we couldn’t figure out how to calculate cost using QoDA we were able to write code in QoDA.
- However we were not able to test it due to time constraints. However we find that QoDA is able to efficiently allocate qubit sizes beyond 29 whereas pennylane could not.

Pennylane vs QoDA



References

[1]Lingyu Hu, Xiaofang Wang, Zhiwei Bao, Qihao Xu, Mingrong Qian, Yuanxiang Jin, The fungicide prothioconazole and its metabolite prothioconazole-desthio disturbed the liver-gut axis in mice,https://doi.org/10.1016/j.chemosphere.2022.136141

[2] Gokul Subramanian Ravi, Pranav Gokhale, Yi Ding, William Kirby, Kaitlin Smith, Jonathan M. Baker, Peter J. Love, Henry Hoffmann, Kenneth R. Brown, and Frederic T. Chong. 2022. CAFQA: A Classical Simulation Bootstrap for Variational Quantum Algorithms. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS 2023). Association for Computing Machinery, New York, NY, USA, 15–29. https://doi.org/10.1145/3567955.3567958

[3] Guillaume Verdon Trevor McCourt Quantum Graph Neural Networks 1909.12264.pdf (arxiv.org)

[4] Xing Ai, Zhihong Zhang, Luzhe Sun, Junchi Yan, Edwin Hancock, Decompositional Quantum Graph Neural Network 2201.05158.pdf (arxiv.org)