Enhancing QAOA using Graph Convolutional Networks for Ising Models

By Michael Halpern and Rebecca Tsekanovskiy

Mapping Ising to Protein Folding

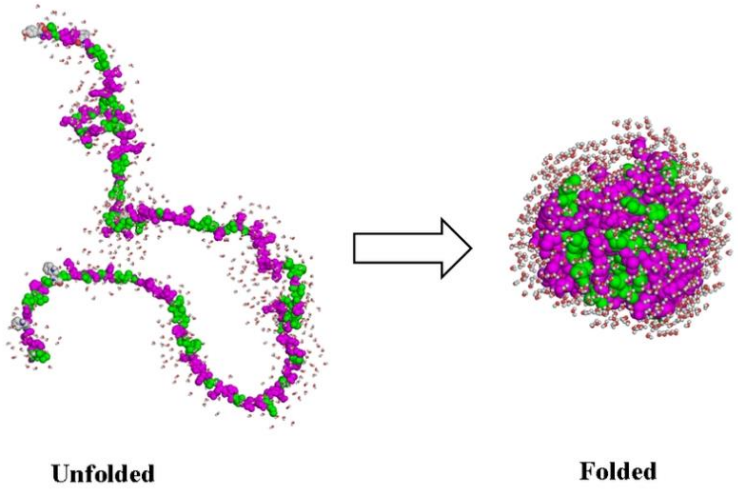
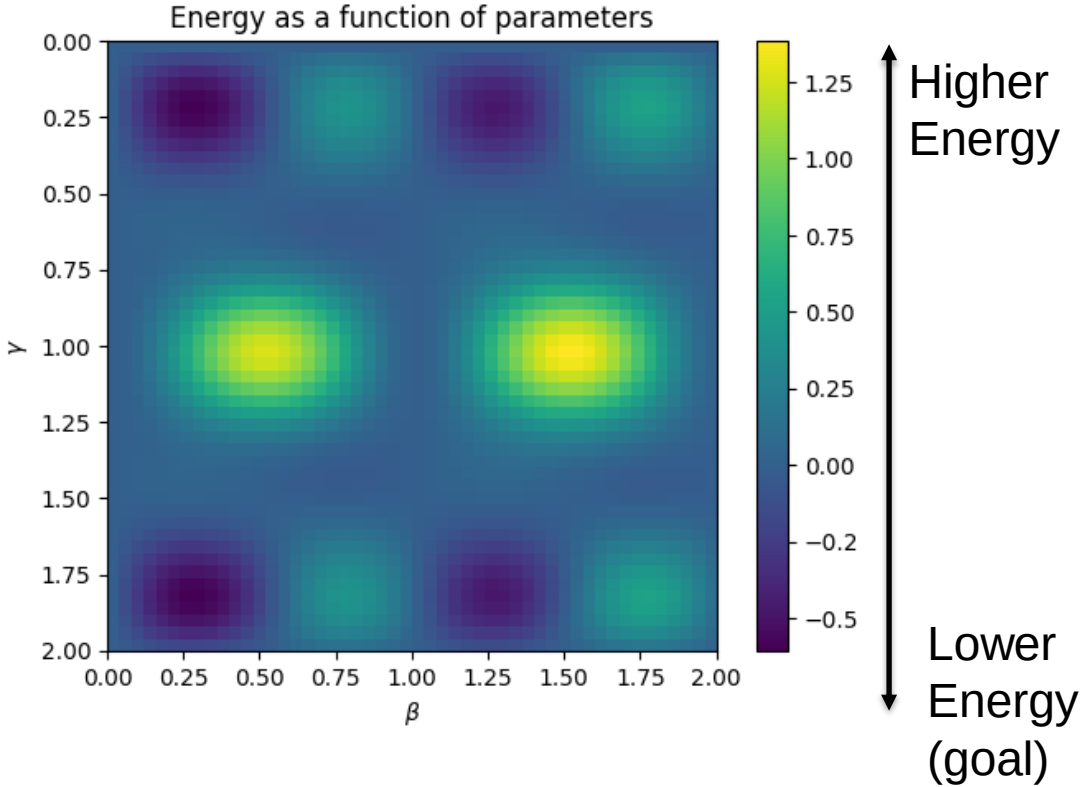


Figure B

- Protein Folding: Folded or Unfolded
- Low Energy states means protein has found its most stable state

ProFAX: A hardware acceleration of a protein folding algorithm - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Protein-folding-from-primary-to-tertiary-structure-12_fig2_310360618 [accessed 20 Jul 2024]
Hoang, T. X., Sushko, N., Li, M. S., & Cieplak, M. (2000). Spin analogues of proteins: Scaling of 'folding' properties. *Institute of Physics, Polish Academy of Sciences*, Al Lotnikow 32/46, 02-668 Warsaw, Poland.

Motivation

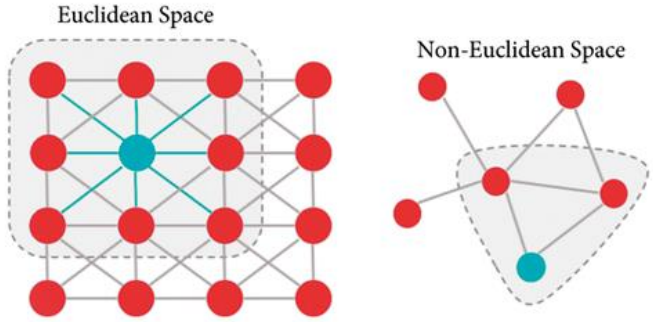


- Traditional Methods: High computational costs and inefficiency in QAOA parameter selection

Google Quantum AI. "QAOA for Ising model on Cirq." Accessed July 20, 2024. https://quantumai.google/cirq/experiments/qaoa/qaoa_ising

Motivation for using a GCN

- GCN model can capture complex relationships within graph structures, enabling the prediction of optimal γ and β values
- GCNs scale well in large graphs and best understand non-Euclidean spaces
- Graphs live in a non-Euclidean space



Uzair Aslam Bhatti, Hao Tang, Guilu Wu, Shah Marjan, and Aamir Hussain. 2023. “Deep Learn-205ing with Graph Convolutional Networks: An Overview and Latest Applications in Computational206Intelligence” 2023 (February): 1–28. https://doi.org/10.1155/2023/8342104.207

- Data creation used for model training
  - Random selection -> Nodes: 2-24, Fill Rate: 0.1-0.9
  - Qiskit simulator for optimal γ and β values

Graph ID	Adjacency List	Gamma Value	Beta Value
1	[[1,3….],[…],[….],..]	-0.711902	-2.3869
..	…	..	..
500	..	..	…

Methodology – Code

Parse in the Data Set

- Create adjacency matrix
- Initialize class to keep track of Gamma and Beta values

GCN Class Initialize

- GCN Convolutional
- GCN Batch normalization
- GCN Linear

Initialize Custom Loss

Train the model

- Have the model predict gamma and beta values off a given graph

Use the RPI-IBM quantum system one.

- Calculate the expected value for both classical and GCN models
- Track and compare time

Custom Loss vs Traditional Metrics

Aspect	Our Custom Loss	Traditional metrics (MAE,MSE,..)
Focus	Focuses on the expected cost differences (predicted vs. actual) based on the QAOA performance	Focuses on minimizing numerical differences (predicted vs. actual) values.
Impact on training	Directly tied to the effectiveness of parameters in the QAOA energy minimization.	Minimizes overall error but may not fully capture the impact on QAOA's specific objectives.
Calculation	Calculates the absolute difference between the predicted and expected costs using a specific cost function	MSE: Calculates the squared difference. MAE: Calculates the absolute difference.

- Sample 5 random indices from the batch
- For each index compute the expected cost for both predicted and known γ and β values
- Calculate the absolute difference between predicted and expected

- Successfully created and implemented a GCN model in PyTorch, and trained it for 20 epochs for about 251 minutes on a dataset size of 500 and a batch size of 32 using RPI’s AiMos NPL cluster

Testing the GCN Model

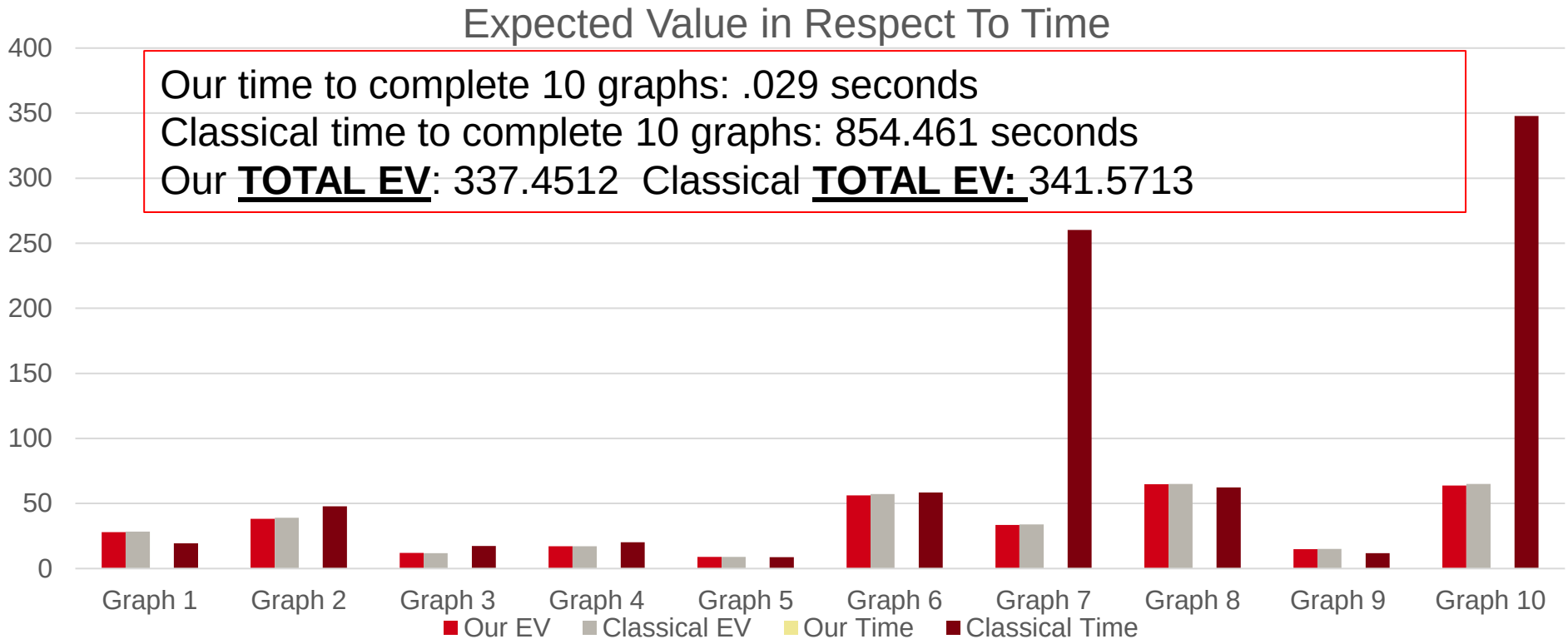
```python
def QAOA(beta: float, gamma: float, fakeBackend: bool, depth: int,
    adjacencyList: list[list[int]],) -> dict:
    if fakeBackend:
        backend = GenericBackendV2(len(adjacencyList))
    else:
        backend = rensslearBackend
    qc = QAOA_circuit(beta, gamma, depth, adjacencyList)
    job = backend.run(qc, shots=1024)
    result = job.result()
    return result.get_counts()
```

Selecting the proper backend, calling the QAOA circuit, and getting back bit string counts

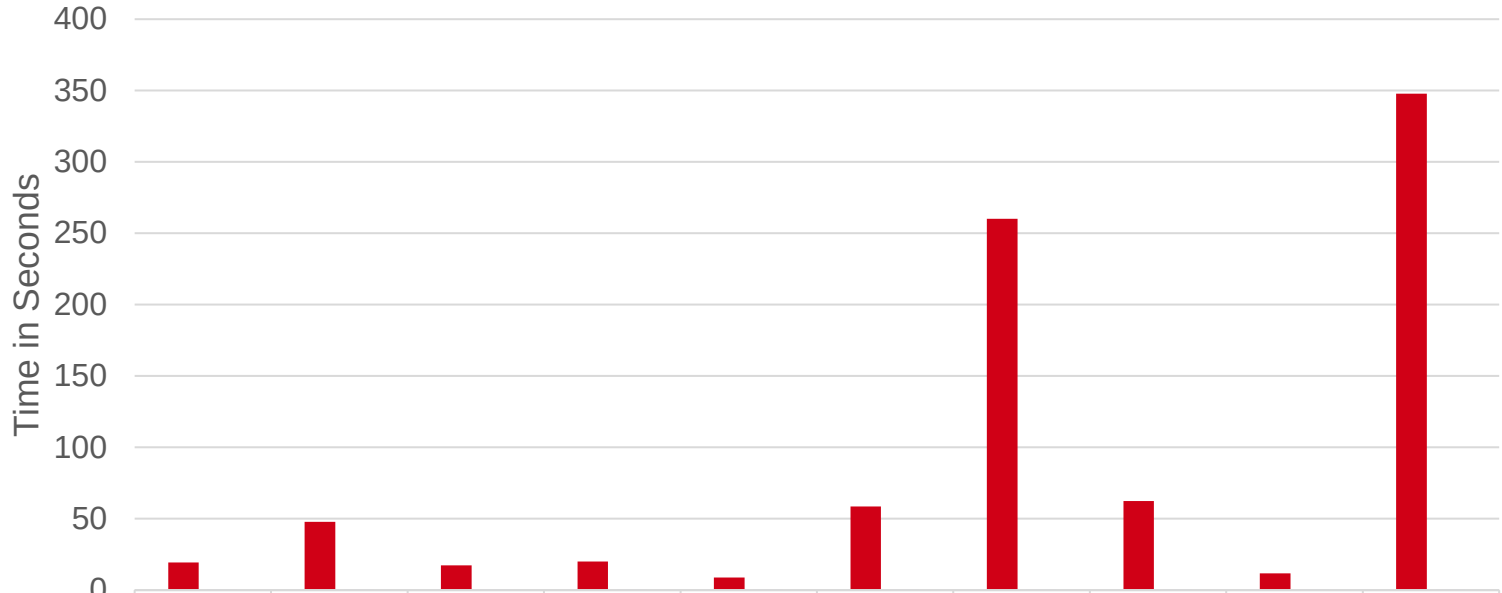
The γ and β values predicted by the model are then used by the QAOA circuit function to construct, optimize, and run the quantum circuit.

```python
def QAOA_circuit(beta: float, gamma: float, depth: int,
    adjacencyList: list[list[int]]):
    num_qubits = len(adjacencyList)
    qc = QuantumCircuit(num_qubits, num_qubits)
    qc.h(range(num_qubits))
    qc.barrier()
    for _ in range(depth):
        for edgeIndex, neighbors in enumerate(adjacencyList):
            for edge in neighbors:
                if edge > edgeIndex:
                    qc.cz(edgeIndex, edge)
                    qc.rz(2 * gamma, edgeIndex)
                    qc.cz(edge, edgeIndex)
        qc.barrier()
        for qubit in range(num_qubits):
            qc.rx(2 * beta, qubit)
        qc.barrier()
    qc.measure(range(num_qubits), range(num_qubits))
    return qc
```

Expected Value Analysis: Classical Method vs GCN Method



Expected Value Analysis: Classical Method vs GCN Method



	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5	Graph 6	Graph 7	Graph 8	Graph 9	Graph 10
Classical Time	19.4474	47.9293	17.3558	20.1487	8.82437	58.561	260.222	62.3465	11.7692	347.856
GCN Time	0.00384	0.00297	0.00292	0.00273	0.00272	0.00286	0.00273	0.00273	0.00273	0.00276

Novelty

Graph Learning for Parameter Prediction of Quantum Approximate Optimization Algorithm

Zhiding Liang* Gang Liu* Zheyuan Liu* Jinglei Cheng¶ Tianyi Hao† Kecheng Liu* Hang Ren‡
Zhixin Song§ Ji Liu‖ Fanny Ye* Yiyu Shi*
* University of Notre Dame, ¶ Purdue University, † University of Wisconsin - Madison, ‡ University of Californi
Berkeley, § Georgia Institute of Technology, ‖ Argonne National Laboratory

Graph neural network initialisation of quantum approximate optimisation

Nishant Jain, Brian Coyle, Elham Kashefi, Niraj Kumar

Graph neural network initialisation of quantum approximate optimisation

Nishant Jain, Brian Coyle, Elham Kashefi, Niraj Kumar

17 Nov 2021 - arXiv: Quantum Physics

Other approaches either use ensemble methods or alternate ML algorithms

Accelerating Quantum Approximate Optimization Algorithm using Machine Learning

Publisher: IEEE Cite This PDF

Mahabubul Alam ; Abdullah Ash-Saki ; Swaroop Ghosh All Authors

Conclusion

- Based on the results from 10 graphs, the GCN was 14.24 minutes faster, averaging a speedup of 1.42 minutes per graph
- The GCN model's expected value is approximately 1.2% lower than that of the classical method.
- GCNs offer a substantial speed advantage in finding gamma and beta values once the model is trained, though the overall expected value remains slightly lower.

Future Improvements

- Expanding the dataset past 500 graphs will lead to better model generalization.
- Allowing the model more time to learn might improve its accuracy by extending the amount of training epochs.
- Spend more time fine tuning the model:
  - Experiment with smaller or larger batch sizes may improve the model's ability to learn from the data.
  - Experiment with different classical machine learning optimizers like SGD to see if they can reduce total training time and increase accuracy performance.
  - Implement learning rate schedulers to dynamically update learning rates, improving model stability.

Rensselaer200