Classical Parameter-Setting Strategies for the Quantum Approximate Optimization Algorithm (QAOA)

August 8, 2024

Group Members:
Rie Fujii (Ochanomizu University)
Shinichiro Kakuta (Waseda University)
Nadav Kohen (Indiana University)
Spencer Lee (Michigan State University)
Kanon Sakurai (Ochanomizu University)

Academic Mentor:
Phillip Kerger (University of California, Berkeley)

Industry Mentors:
Aruto Hosaka, Isamu Kudo, Tsuyoshi Yoshida
(Mitsubishi Electric Corporation)

Overview

1. Introduce combinatorial optimization.
2. Introduce the QAOA.
3. Describe the homogeneous proxy.
4. Describe our newly proposed QAOA distribution proxy.
5. Present computational results.
6. Future Work.

What can quantum computing and QAOA be useful for?

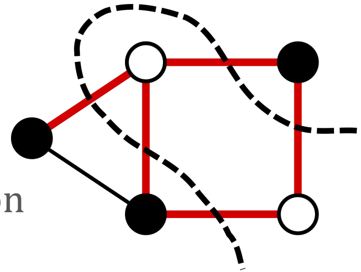
Motivation:Why Combinatorial Optimization Problems?

We want to solve **<u>Combinatorial Optimization Problems</u>**!

→ to find the best combination from **discrete** and **finite** options

Max Cut

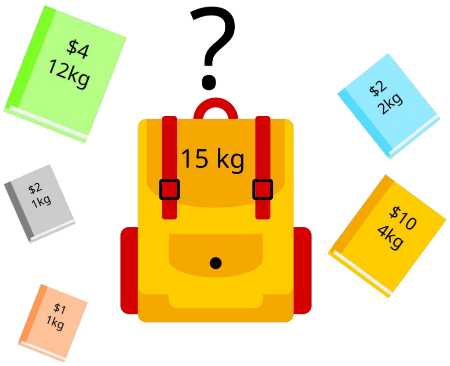
→ to solve the best grouping under complex conditions.



⭐ It has important application in computer chip design.

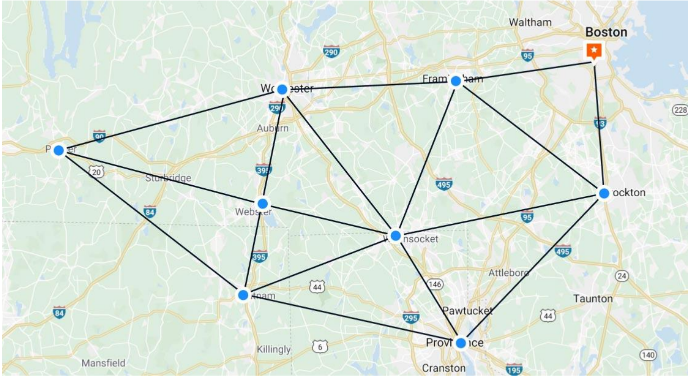
Knapsack Problem

→ to find the best way of packing a bag, or truck or airplane, that has limited space.



Traveling Salesman Problem

→ to find the fastest way to visit all of them.



Motivation: Why Quantum Computing?

Quantum computers:

- Impressive in theory (much faster than classical computer)
- Limited in practice (high noise, high error ...)

The Quantum Approximate Optimization Algorithm (QAOA)

doesn't need many quantum bits (qubits)

↓

suitable for near-term devices

⭐ QAOA solves combinatorial optimization problems!

GOAL: Propose more efficient (and better quality) solutions

to **combinatorial optimization problems** using QAOA

Motivation: Why QAOA?

⭐QAOA solves combinatorial optimization problems!

Recent Results:

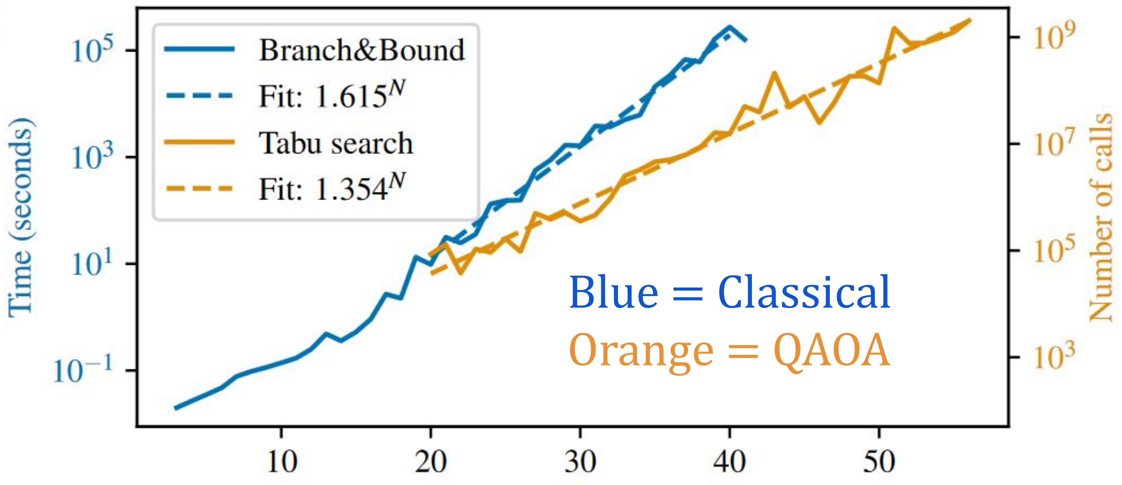
QAOA is advantageous in terms of scaling for certain problems.

↓

QAOA might be **faster**

than classical optimization for certain problems!

Time to Solve vs LABS Problem Size



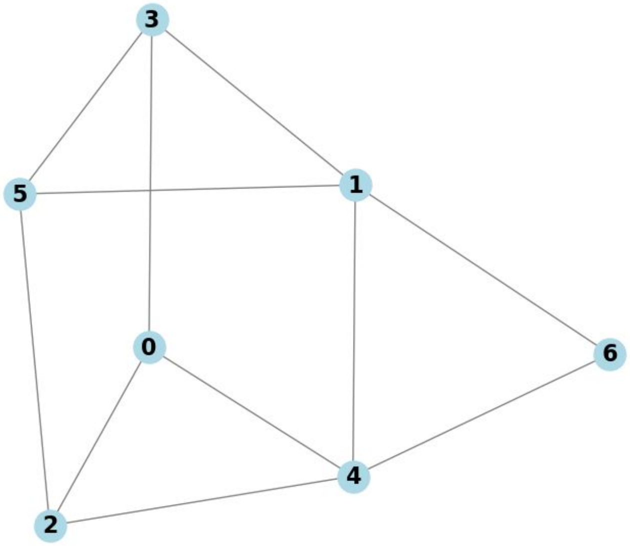
Problem Size

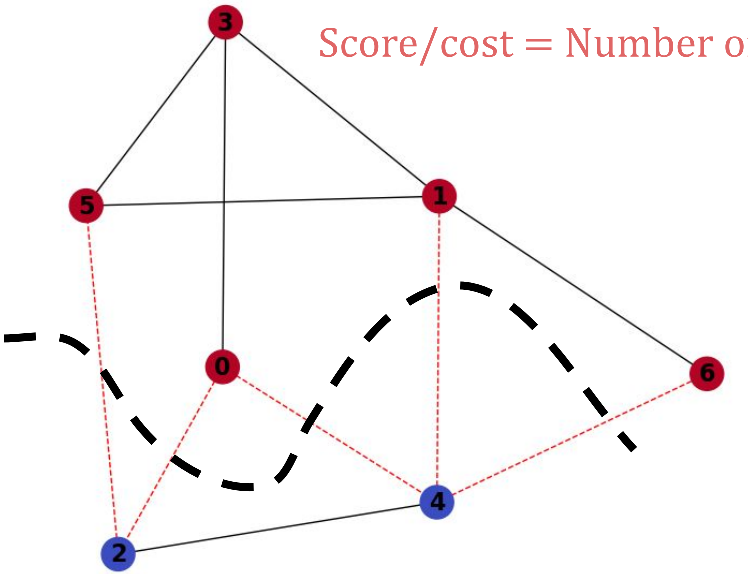
→ QAOA require less computation time.

Motivation: What is MAX-CUT?

With QAOA, we solve the Max Cut Problem:

Divide a graph into two groups to maximize the number of edges connecting the two groups.





Score/cost = Number of edges “cut”

Motivation: Why MAX-CUT?

Problem statement is easy to understand.

NP-hard ⇒ Doing well on MaxCut is impressive.

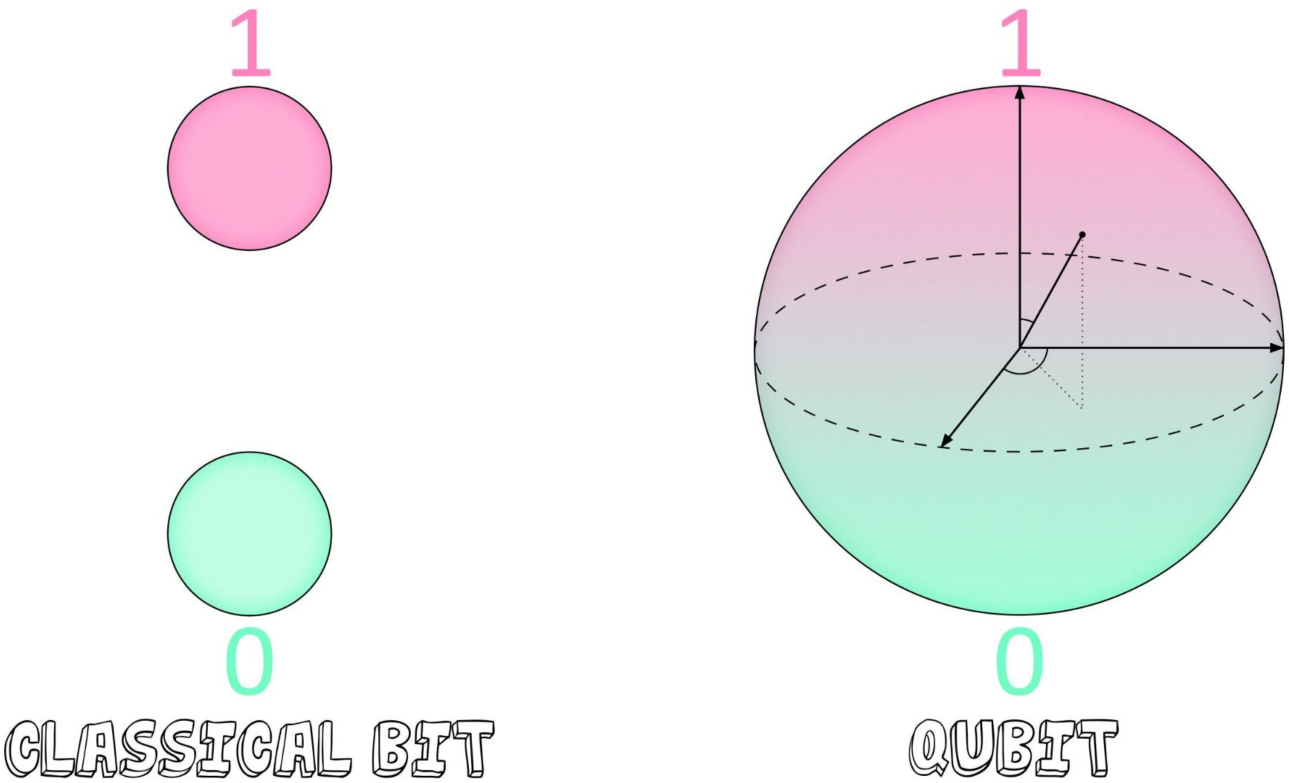
Applying QAOA to MaxCut is simple.

(Effective method has already been established.)

The method is promising for different types of graphs (ex: network)

Background

Qubits



Quantum Computing

With n qubits, we can represent a superposition of 2^n bitstrings of 1's and 0's:

$$|x\rangle = \sum_{i=1}^{2^n} q(y_i)|y_i\rangle, \quad \in \mathbb{C}^{2^n}$$

For $n = 3$:

$$y_1 = 000, \quad y_2 = 001, \quad y_3 = 010, \quad y_4 = 011,$$
$$y_5 = 100, \quad y_6 = 101, \quad y_7 = 110, \quad y_8 = 111$$

$$\text{Probability of measuring } |y_1\rangle = |000\rangle \text{ is } q(y_1)$$

What is the QAOA?

With n qubits, we can represent a superposition of 2^n bitstrings of 1's and 0's:

$$|x\rangle = \sum_{i=1}^{2^n} q(y_i)|y_i\rangle, \quad \in \mathbb{C}^{2^n}$$

In QAOA, each bitstring represents a possible solution.

Can measure **expectation** : average "strength" of the solutions:

$$\langle x|H_p|x\rangle = \vec{x}^\dagger H_p \vec{x}$$

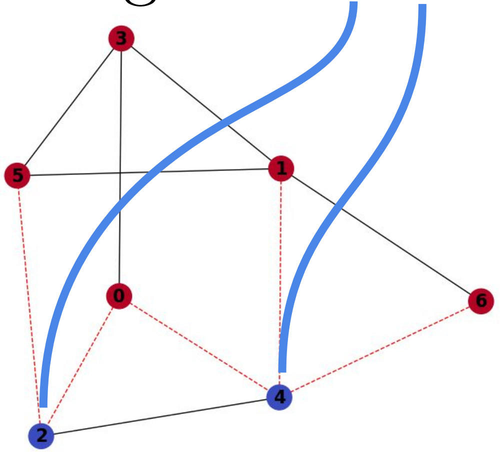
High expectation ⇒ high chance of measuring a "good" solution.

How to Encode MaxCut?

For N vertices, we use N qubits.

Each bit in a bitstring determines which partition to place a vertex in.

$$\text{Bitstring} = 0010100$$



Measure costs using Ising model: $H_p = \sum_{(i,j)\in E} -w_{ij}(1 - \sigma_{x,i}\sigma_{x,j})$

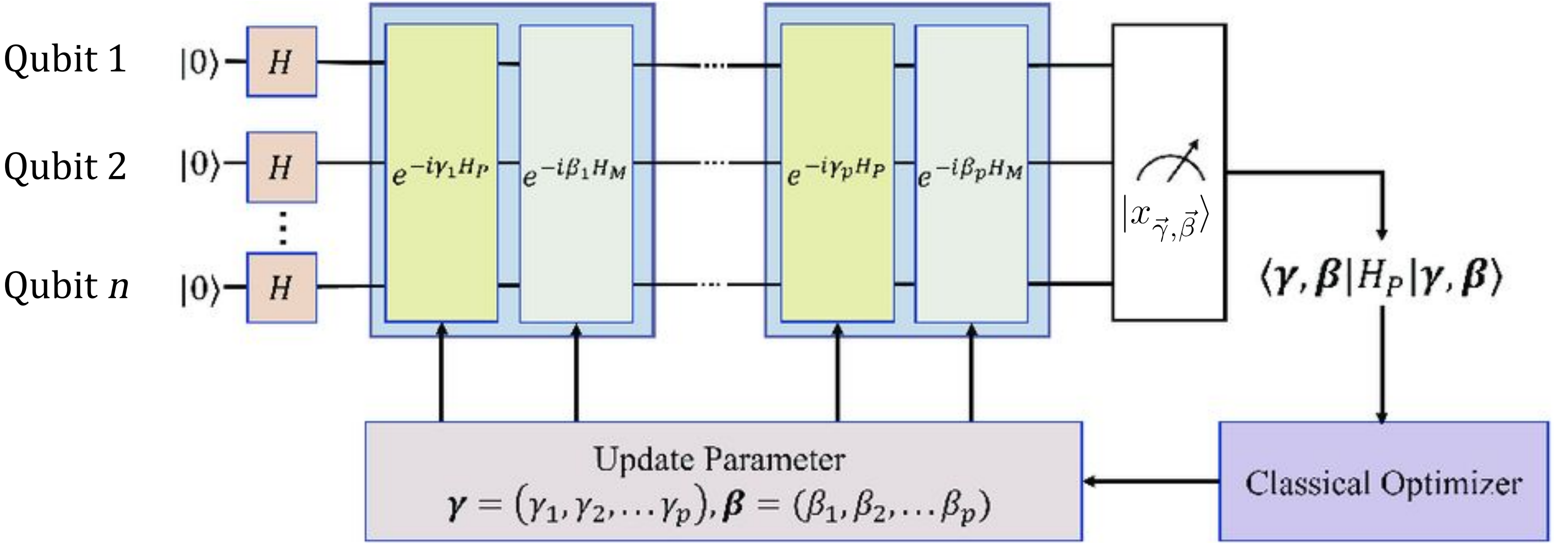
QAOA Circuit

$|x_{\vec{\gamma},\vec{\beta}}\rangle$ "explores" $\mathbb{C}^{2^n}$ as we vary parameters $\vec{\gamma}, \vec{\beta} \in \mathbb{R}^p$.

p determines the number of QAOA "layers."

Physically motivated: more layers ⇒ **can get closer to optimal cost.**

But optimizing the parameters is harder!

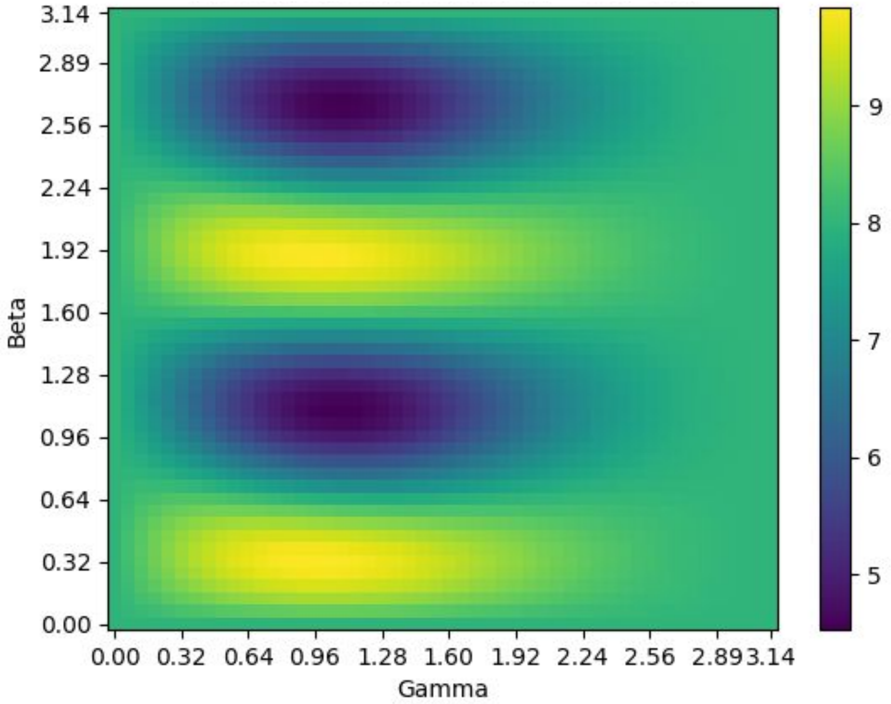


Optimizing QAOA Circuit

Simple *p=1* case.

Vary $\vec{\gamma}, \vec{\beta} \in \mathbb{R}^p$ until we find the maximum.

Expectation (weighted cost average) vs Parameters



Methodology

Parameter Setting

We have a lot of parameters.

Optimizing them is **expensive** !

Simulating QAOA is **expensive** !

Real quantum computing is **hard!**

Figure
The number of parameters we tune.
(Moog synthesizer)

Homogeneous Proxy

QAOA produces nearly "homogeneous" states, where (basis) states with the same cost have similar probability amplitudes.

If the amplitudes are *exactly the same*, we can reduce the size of our model.

Full QAOA

Homogeneous Model

How the Homogenous Proxy Works

Before proxy: $$|x_\ell\rangle = \sum_{i=1}^{2^n} q_\ell(y_i)|y_i\rangle$$

$$|x_\ell\rangle \in \mathbb{C}^{\#\text{ Possible Bitstrings } = 2^n}$$

After proxy: $$|z_\ell\rangle = \sum_{i=1}^{n^2} Q_\ell(c_i)|c_i\rangle$$

$$|z_\ell\rangle \in \mathbb{C}^{\#\text{ Possible Costs } = \#\text{ Edges}}$$

SMALLER!

How the Homogenous Proxy Works

To calculate the proxy state changes through each layer of QAOA:

$$|z_\ell\rangle = \sum_c Q_\ell(c)|c\rangle$$

$$Q_\ell(c) = \sum_{d,\hat{c}} \mathrm{coeff} \times Q_{\ell-1}(\hat{c}) \boxed{N(c; d, \hat{c})}$$

Bitstrings with cost c and distance d from bitstrings with cost $\hat{c}$

How the Homogenous Proxy Works

$$\boxed{N(c; d, \hat{c})}$$

Bitstrings with cost *c* and distance *d* from bitstrings with cost

Generally, the distribution *N* doesn't really exist. It approximates the real distribution:

$$n(\boxed{x}; d, \hat{c})$$

Depends on bitstring! たかい!

Our Research Objective

$$\boxed{N(c; d, \hat{c})}$$

\# Bitstrings with cost c and distance d
from bitstrings with cost

Our research is about finding new distributions ***N*** **to approximate** ***n*****.**

Binomial and Multinomial Approximations

Calculation of

$$N(c; d, \hat{c})$$

requires binomial and multinomial probabilities, many times for the proxy calculation

→ We want **approximations** to speed it up

Binomial and Multinomial Approximations

The safest method for both of them is to use the

NORMAL DISTRIBUTION .

Other possibilities to approximate:

- Binomial distribution
- Multinomial distribution

Binomial Approximations

The normal distribution

SAFETY

The Poisson distribution

LIMIT

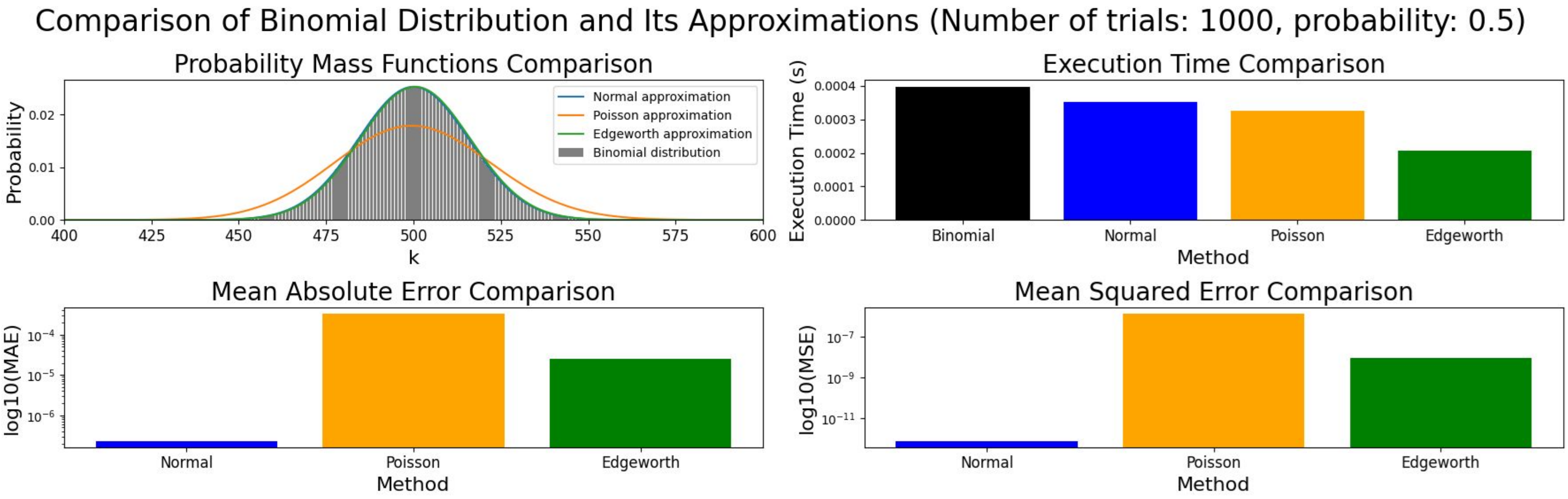
The Edgeworth expansion method

CORRECTION

Multinomial Approximations

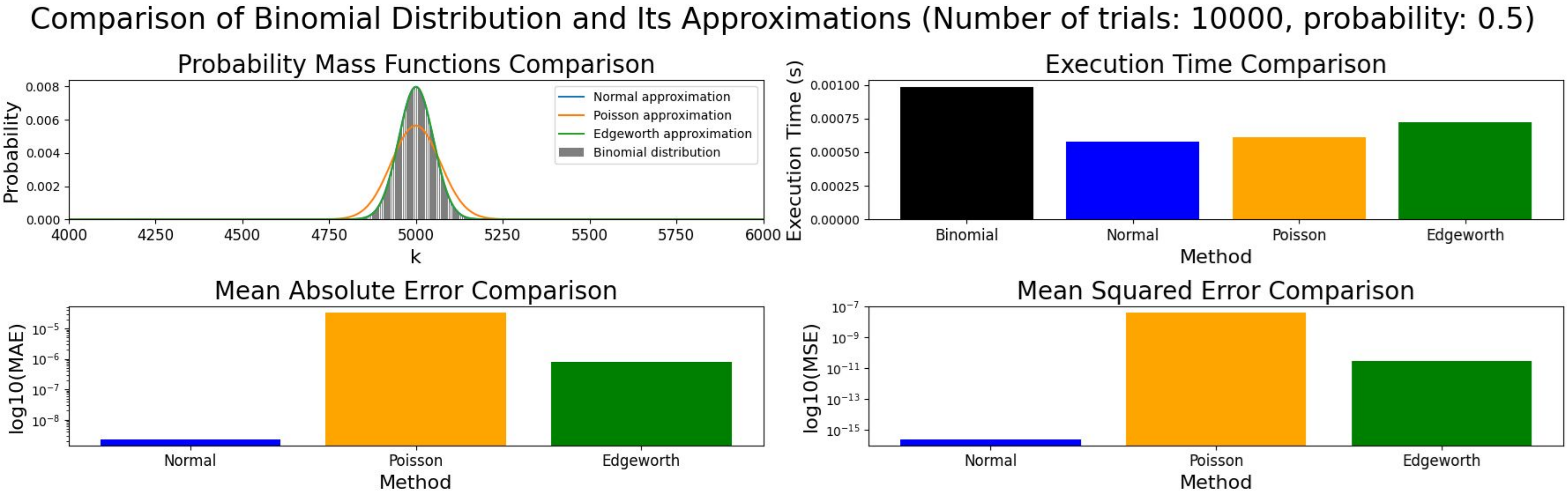
The normal distribution	SAFETY AS WELL
The Poisson distribution	SORT OF LIMIT
The Edgeworth method	ALSO CORRECTION
The Laplace approximation	ESTIMATION FOR SHAPE
The Markov Chain Monte Carlo (MCMC)	SAMPLING METHOD

Binomial Approximations



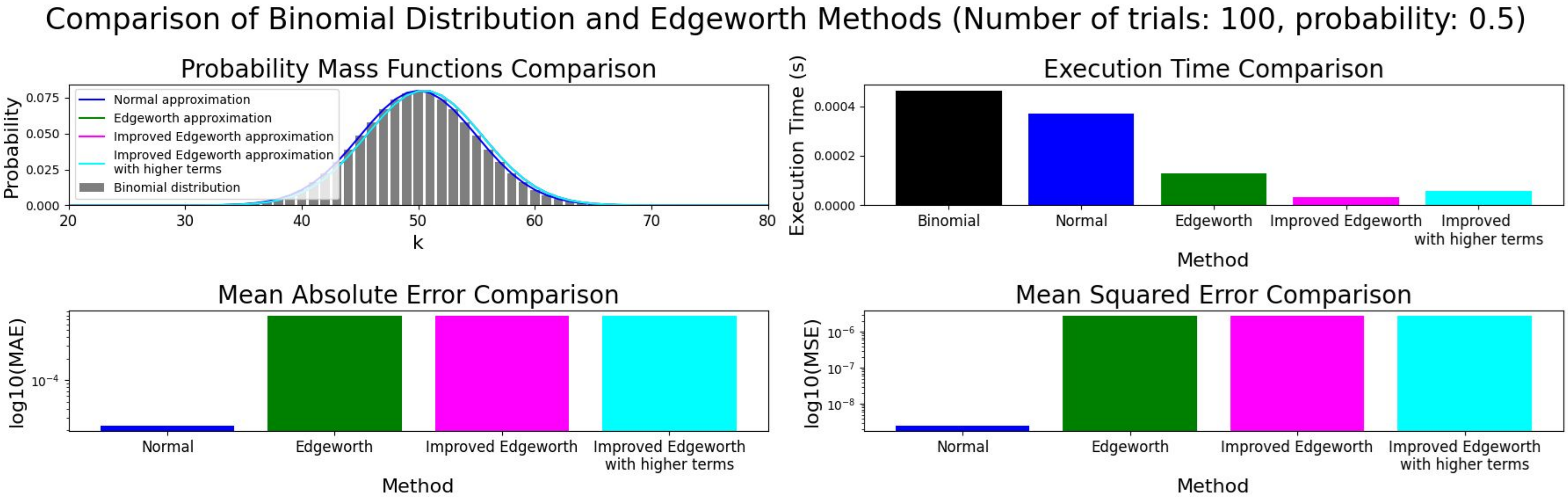
RUNTIME OF THE EDGEWORTH SEEMS TO BE SO FAST……

Binomial Approximations



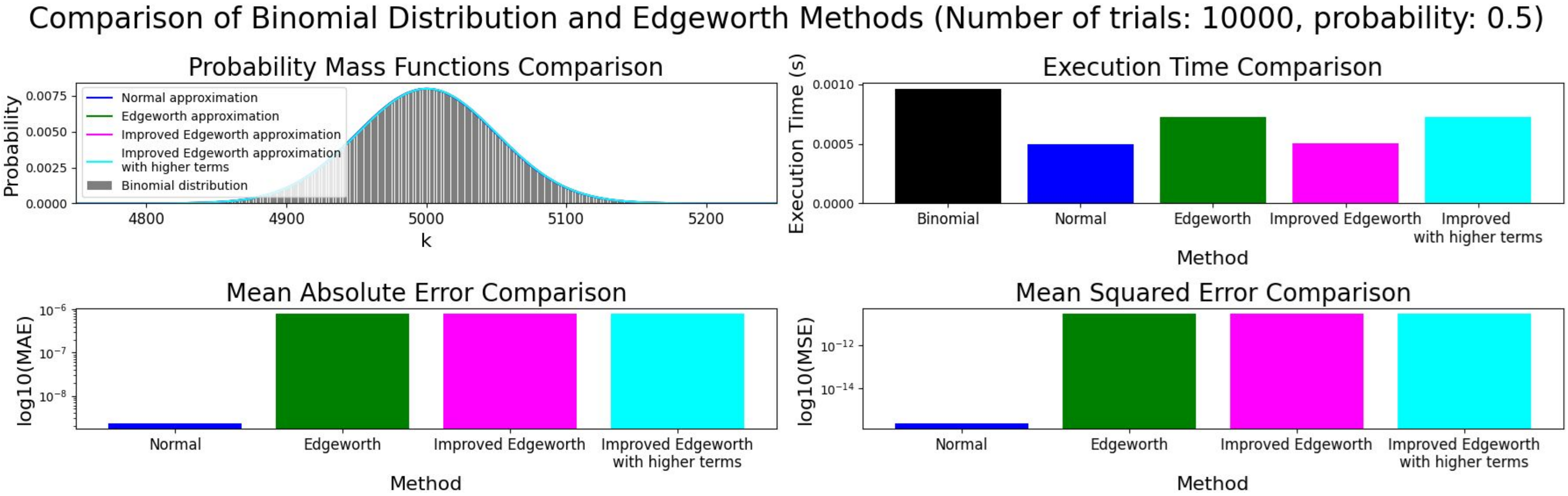
BUT IT EXCEEDS THE NORMAL FOR SUFFICIENTLY LARGE SIZE.

Binomial Approximations



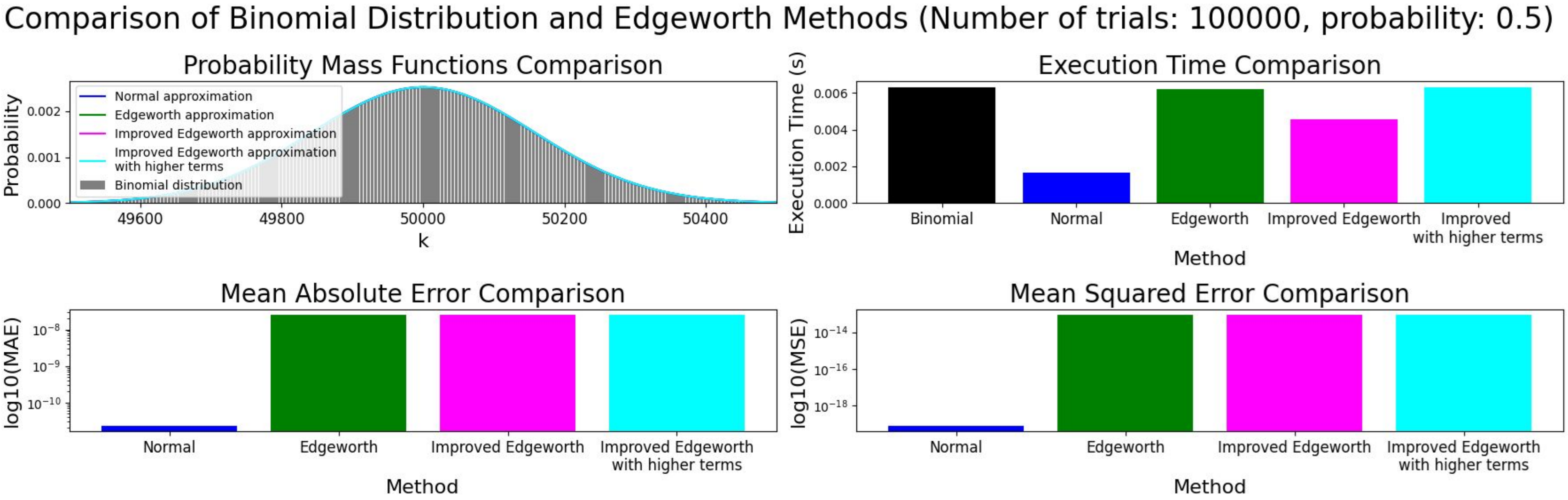
FASTER EDGEWORTH METHODS DO EXIST.

Binomial Approximations



THE METHOD WITH HIGHER TERMS IS SLOWER FOR LARGE n……

Binomial Approximations



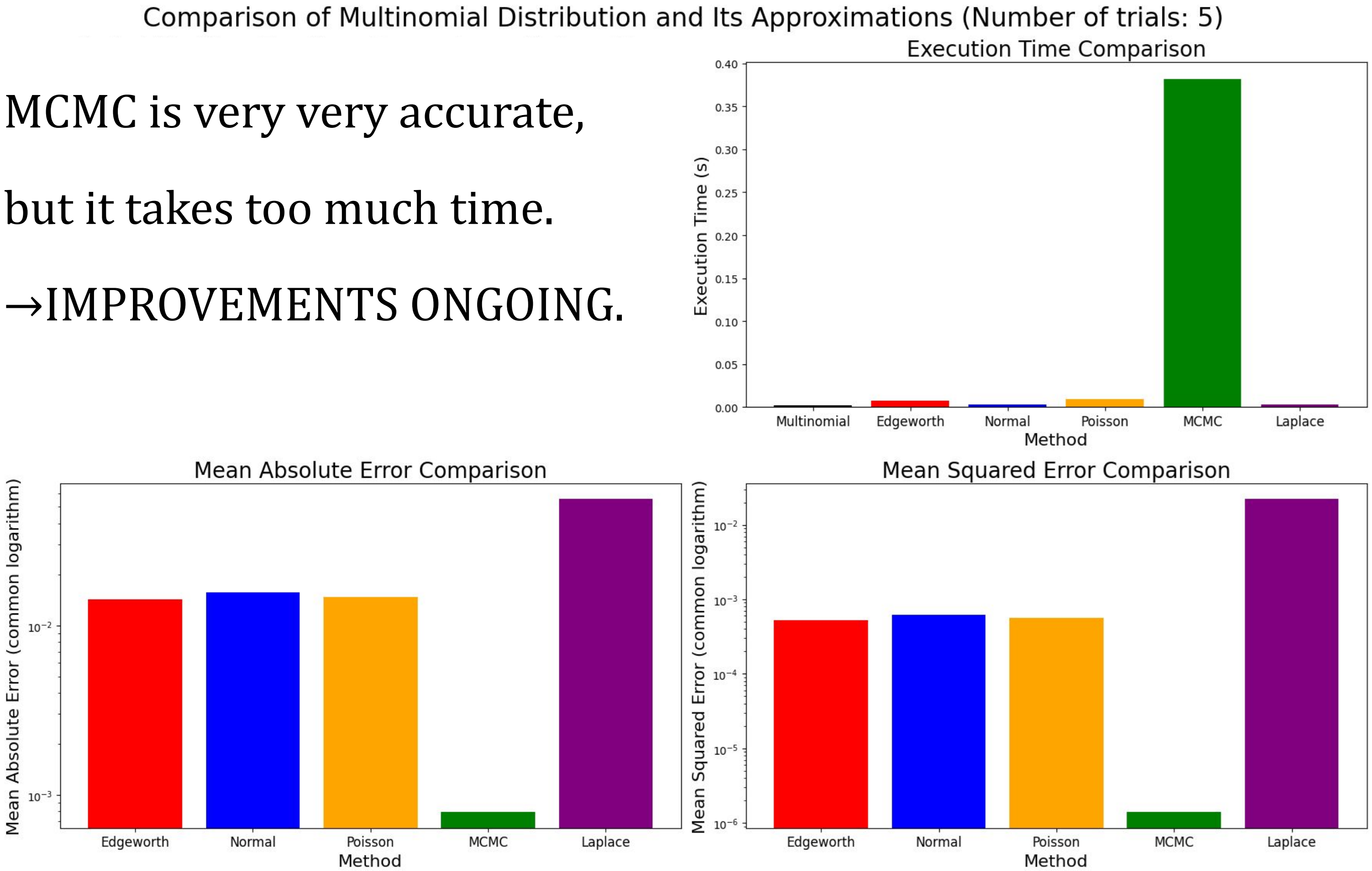
EVEN IMPROVED EDGEWORTH METHODS UNDERPERFORM FOR SOME RANGE.

Multinomial Approximations

MCMC is very very accurate,

but it takes too much time.

→IMPROVEMENTS ONGOING.



Modifying the Homogeneous Proxy

Paper Proxy Distribution

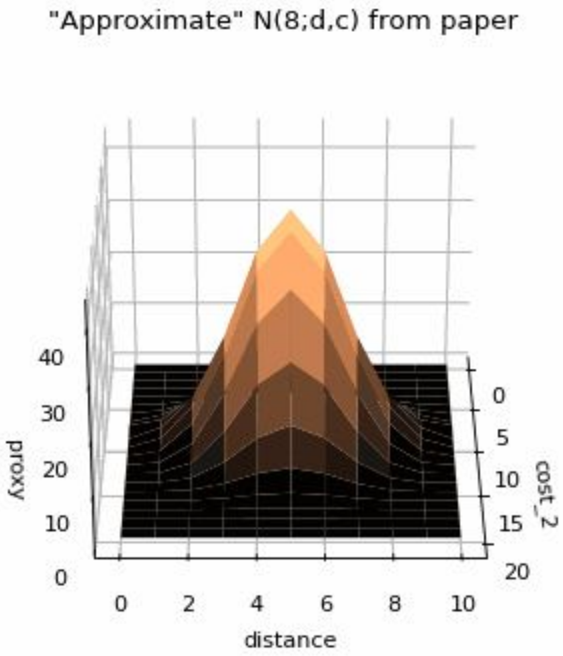
Theoretically derives average distribution for Random CSPs.

Uses that derived form with MaxCut probabilities.

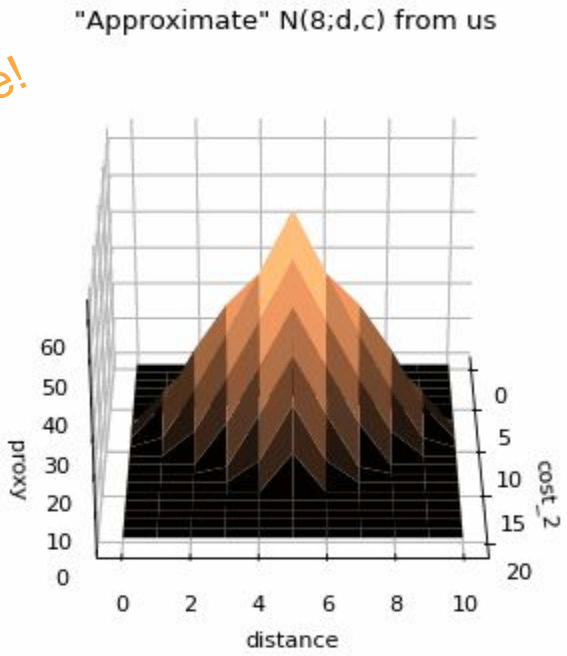
Claims to achieve an **acceptable** approximation.

Relatively **inefficient** to compute.

Our Proxy Distribution



simple!



fast!

wow!

Our Proxy

Full QAOA

Homogeneous Model

Paper Proxy

Our Proxy

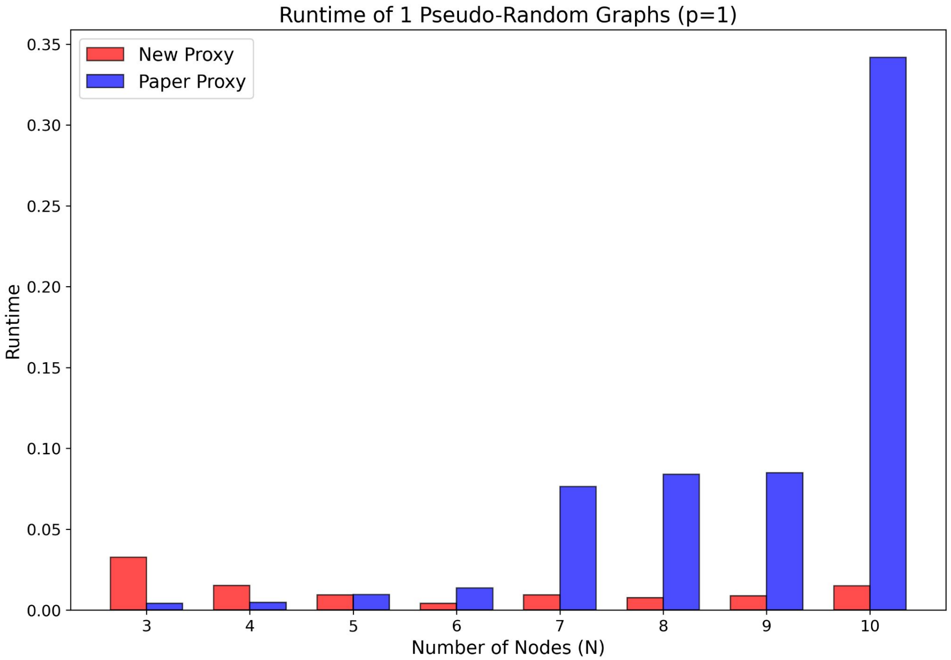
Our Proxy Distribution:

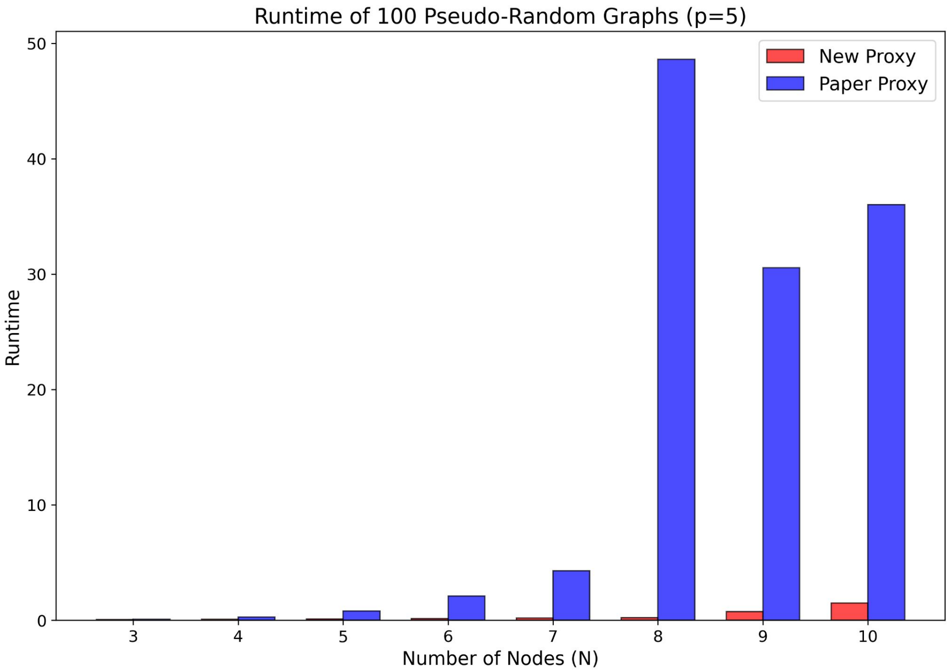
- Small
- Simple
- *Very fast!*

Computational Results

Runtime

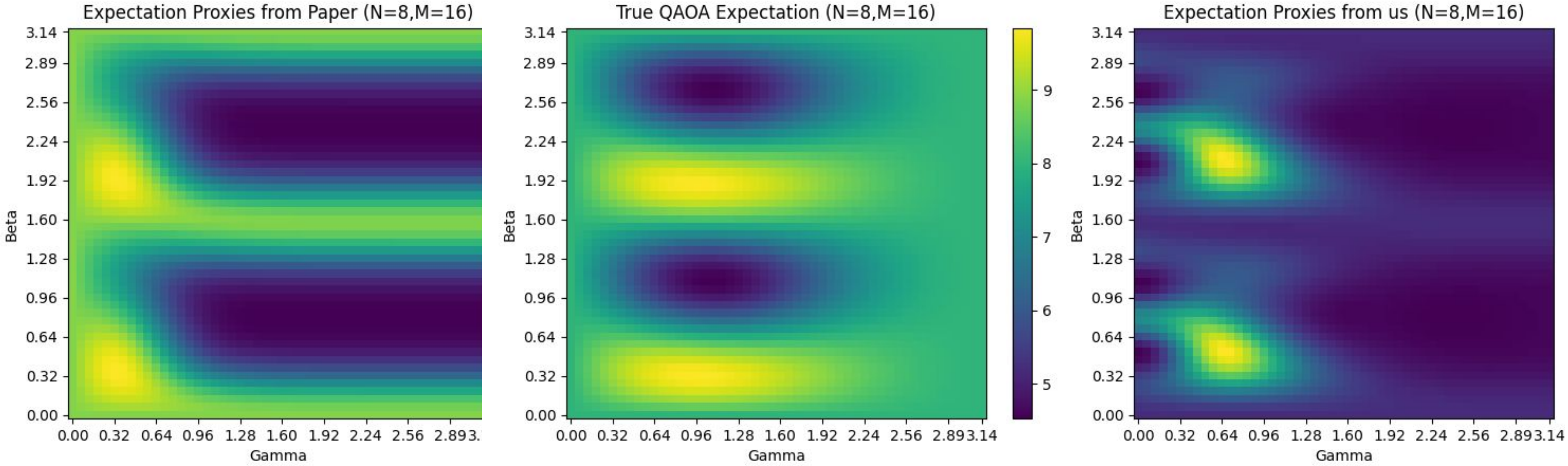
Our new proxy requires **less runtime** .





Expectation

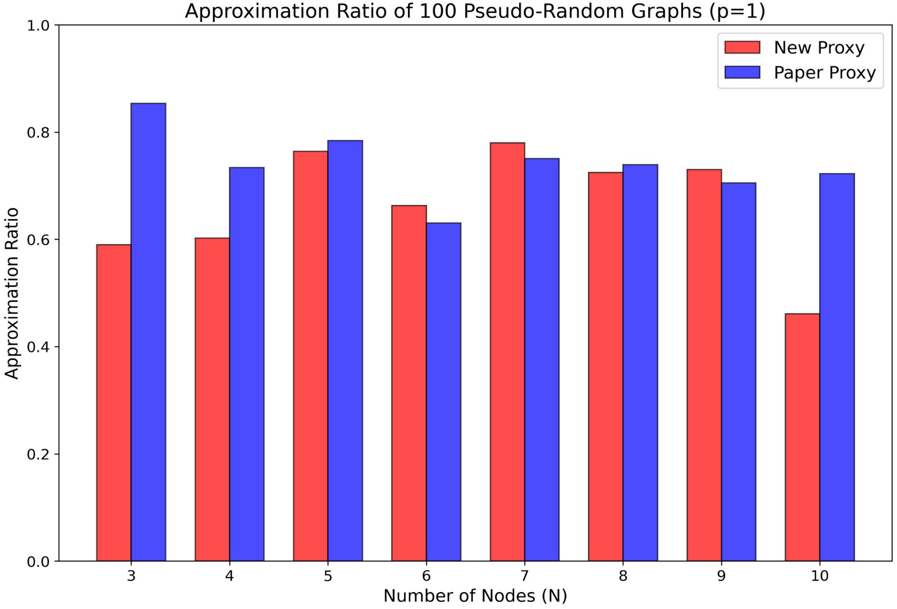
Our proxy demonstrates **a similar ability** to accurately capture the locations of the maxima (for p=1).

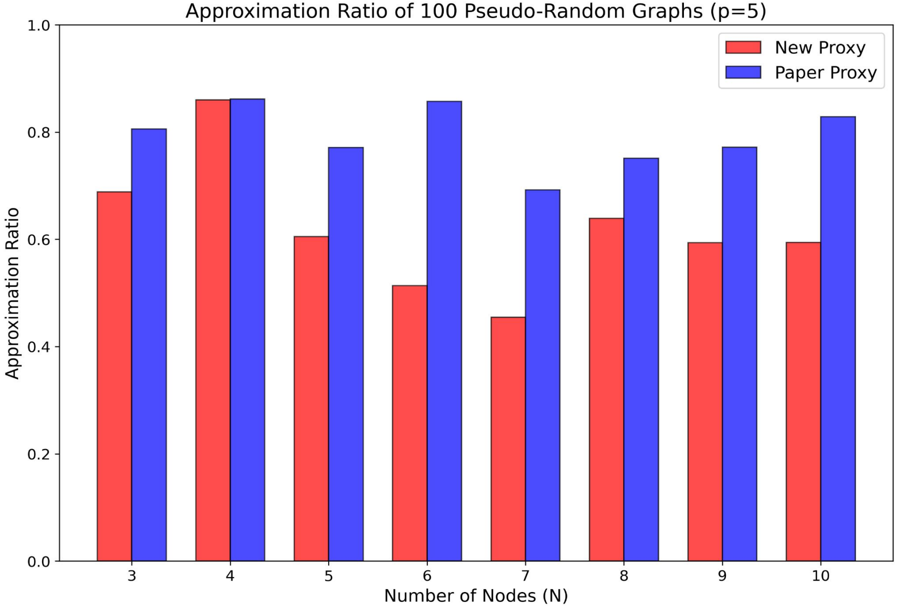


Approximation Ratio

$$\text{approximation ratio} = \frac{\text{expectation}}{\text{maximum cost}}$$

Our new proxy is **simpler** than and **comparable** to the paper proxy (for p=1).

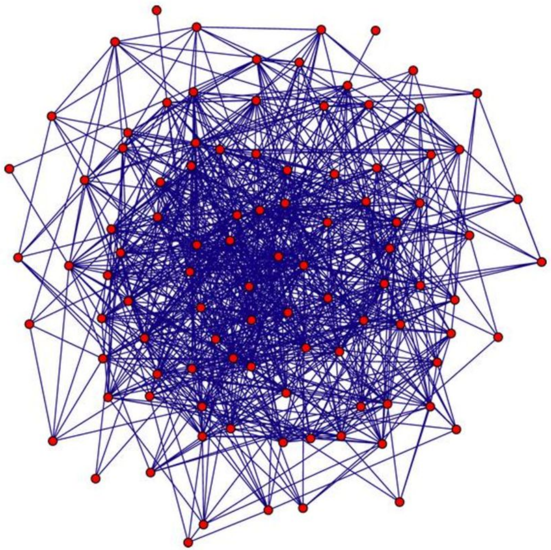




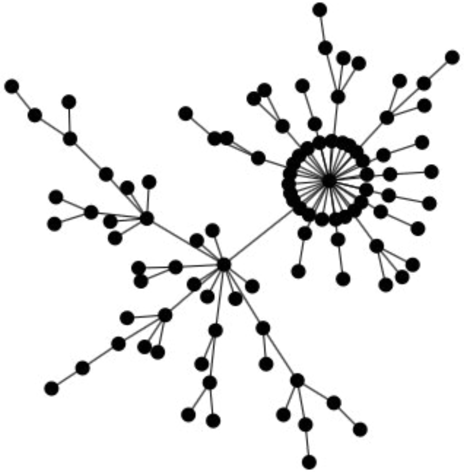
Application to Other Graphs

The current proxy algorithm can **only** be used with a specific class of graph.

Erdős–Rényi model

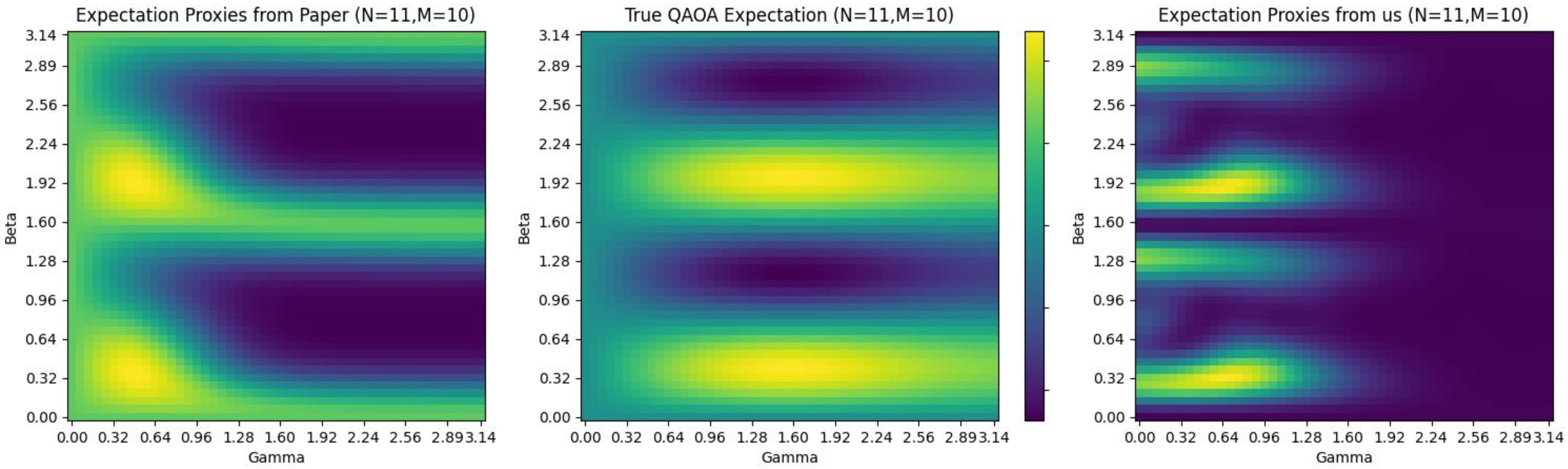


Barabási–Albert model



Application to Other Graphs

Currently, our proxy and paper proxy **fail to accurately capture the location of the maxima** in other graphs.

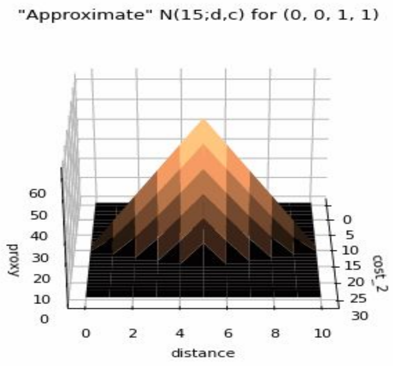


Parameterized Proxy Improvements (ongoing…)

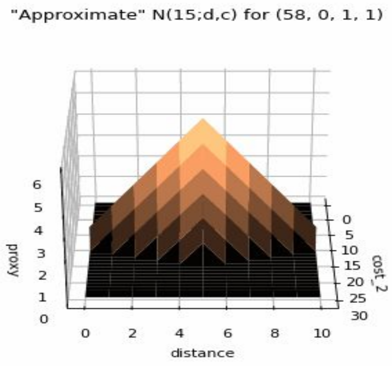
Parameterization of Our Proxy - for Flexibility!

(height squash, peak shift, left flatten, right flatten)

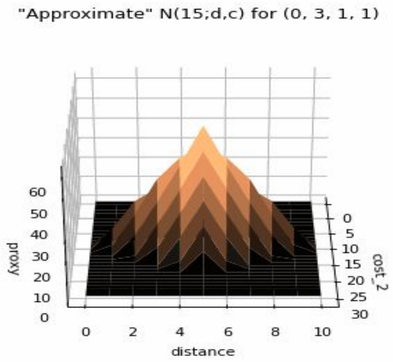
Original



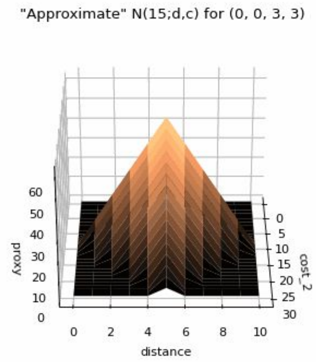
Squashed



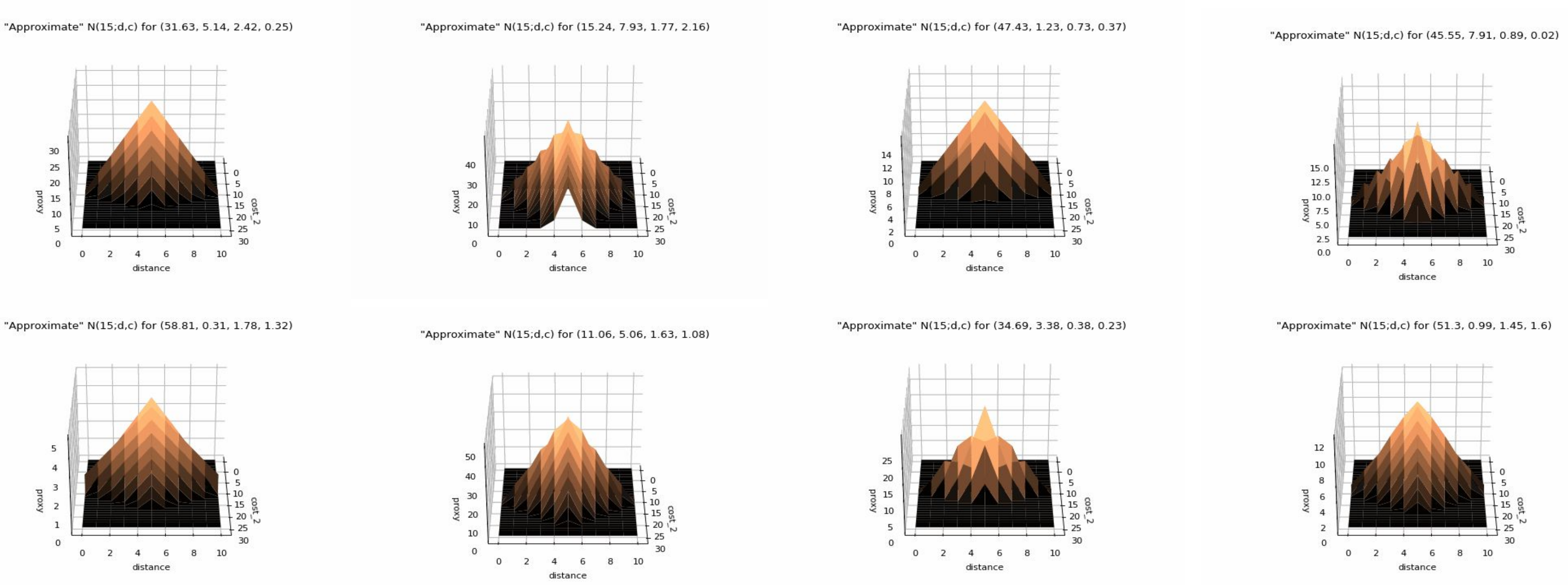
Off-Center



Wider



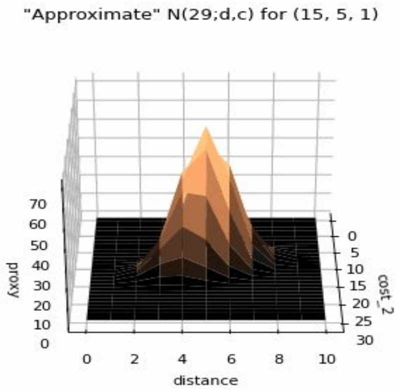
Parameterization of Our Proxy

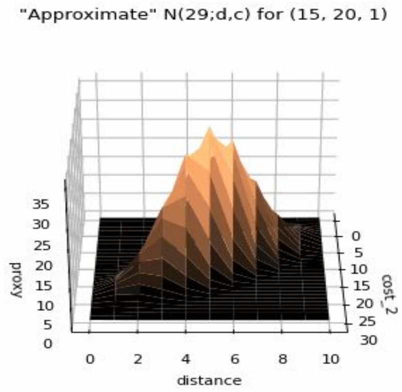


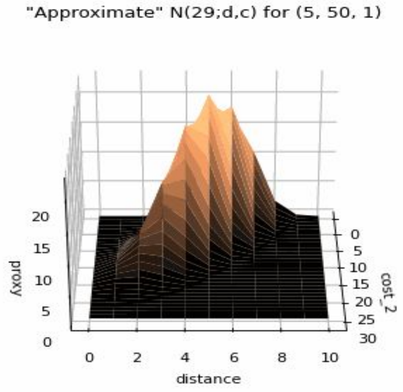
Different parameters at each layer?

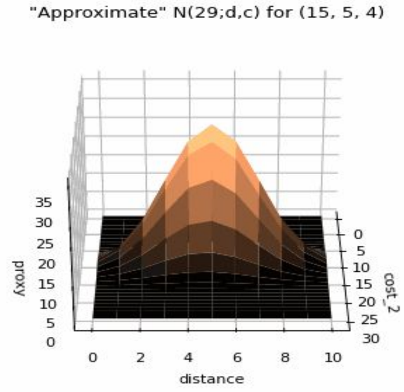
$$Q_\ell(c) = \sum_{d,\hat{c}} \text{coeff} \times Q_{\ell-1}(\hat{c}) \boxed{N(c; d, \hat{c})}$$

Parameterized Normal Proxy









Parameterized Proxies

How should we optimize our distributions?

- Fit to multinomial approximation
- Fit to real distribution data
  - For each problem instance
- Fit to give best QAOA performance

Future Work

- **Improve** the proxy and **extend** its applicability to generic problems.
    - Parameterized approach
    - “Learning” good distributions to use for a problem class

- A **“mathematical” explanation** of why our proxy is better for p=1.

- Write a paper!

Takeaways

Key Takeaways

- **QAOA parameter sensitivity** and **optimization challenges**

- Improvements in classical methods for **precomputing** good parameters
  - by polishing a parameter-setting approach based on the homogeneous QAOA proxy

- Some results in existing work are potentially **misleading**
  - This issue may be present in multiple existing works!
  - That puts into question the usefulness of some those approaches

Key Takeaways

- The models **need to be revised** to achieve stronger performance in general
    - Some approaches which could lead to better or more general classical parameter precomputing methods were not successful yet.

- The existing literature is **narrow** in the problems that QAOA is applied to.
    - Some existing methods may not generalize well.

- Development of **an advanced open-source toolkit** for an existing parameter setting heuristic and many QAOA-related utilitie
    - Python-compatible code, Julia backend, GPU-compatible code, many utilities for QAOA-related tasks
    - It makes QAOA research and application more accessible and efficient

Thanks For Listening!

Special Thanks
Mitsubishi Electric, AIMR, IPAM, MathCCS, TFC
Industry Mentors: Aruto Hosaka, Isamu Kudo, Tsuyoshi Yoshida

AIMR

ipam

MathCCS
Mathematical Science Center
for Co-creative Society,
Tohoku University

Tohoku Forum
for Creativity

Q&A

Classical Computer VS Quantum Computer

Classical Computer		Quantum Computer
Bit 0 or 1	**minimal unit**	Qubit (=quantum bit)
Always either 0 or 1	**state**	Superposition : $\|\psi\rangle \in \mathbb{C}^2$ $\|\psi\rangle = \alpha_0\|0\rangle + \alpha_1\|1\rangle$ $(\|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad \|1\rangle = \begin{pmatrix}0\\1\end{pmatrix})$
Time is proportional to the computational complexity. Error correction is possible.	**feature**	Entanglement : qubits affect other qubits *n qubits : $\|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x\|x\rangle \in \mathbb{C}^{2^n}$ Sometimes exponentially faster ←good !
Widely applicable (data processing, software execution etc…)	**application**	Specifically applicable (optimization problems, cryptanalysis etc…)

Graph 2: Barabási–Albert Model

- Characteristics
  - **Growth** : Network expands over time with the addition of new nodes
  - **Preferential Attachment** : New nodes are more likely to connect to nodes with many connections

- Applications
  - **Internet** : Web page link structures
  - **Social Networks** : Connections between people
  - **Biological Networks** : Protein interactions

Graph 3: Watts–Strogatz model

- Characteristics
  - **High Clustering** : Nodes tend to form tightly-knit groups
  - **Short Path Lengths** : Average distance between nodes is short

- Applications
  - **Social Networks** : Social interactions and information spread
  - **Neuroscience** : Brain connectivity and information processing
  - **Epidemiology** : Spread of diseases and viruses

Erdős–Rényi Model

n/N will be different for different classes of graphs. We focus on Erdős–Rényi:

Consider all graphs with N vertices and M edges. Pick one randomly.

All $N = 3$, $M = 2$ graphs