G-RIPS Mitsubishi-B Project: Final Report

Project Members: Rie Fujii, Shin-ichiro Kakuta, Phillip Kerger[1], Nadav Kohen, Spencer Lee[2], and Kanon Sakurai
Industry Mentors: Aruto Hosaka, Isamu Kudo, and Tsuyoshi Yoshida

[1]Academic Mentor, [2]Project Manager
August 8, 2024

Abstract

The quantum approximate optimization algorithm (QAOA) is a promising algorithm for solving various optimization problems on near-term NISQ hardware. The algorithm alternates applying mixer and cost Hamiltonians and can be interpreted as a trotterized version of adiabatic quantum computation, with the goal of gradually evolving a set of qubits to determine the ground state of the cost Hamiltonian. QAOA is typically parameterized by a circuit-depth parameter p together with parameters $\gamma_1, ..., \gamma_p$ and $\beta_1, ..., \beta_p$ that control the applications of the cost and mixer Hamiltonians, respectively, and strongly influence the performance of the algorithm. The problem of finding a good set of parameters for QAOA remains a challenging subject of ongoing research. Recently, classical-computationally tractable methods for finding good parameters were introduced by making use of the observation that states in QAOA tend to be homogeneous (Sud, Hadfield, Rieffel, Tubman, & Hogg, 2024). Assuming homogeneity, one can construct a simplified QAOA proxy that allows to maintain one probability amplitude per problem *cost*, instead of one probability amplitude per possible solution. In the present work, we aim to extend and improve upon these existing methods. In particular, we aim to construct a faster proxy by further simplifying the model via the distributions involved in the proxy via approximations, and we also propose a scheme for parameterizing these distributions to devise a classical parameter-setting framework that is problem agnostic. For the latter, we propose to *learn* an appropriate approximate distribution to use for the homogeneous proxy for the problem or problem class in question. No claims can currently be made that our proposed method yields any *improvement*, as much more fine-tuning and experimentation needs to be done. However, the approach and principles behind it make us hopeful that the method can be made practical.

Contents

1 Introduction

The quantum approximate optimization algorithm, or QAOA, (Farhi, Goldstone, & Gutmann, 2014) is one of a handful of algorithms that is well suited to be implemented on noisy intermediate-scale quantum (NISQ) hardware, and is thus an exciting area of research within quantum computing due to possible near-term impacts. The algorithm itself uses a parameterized quantum circuit to attempt to (approximately) solve an optimization problem. The algorithm can be viewed as a hybrid quantum-classical algorithm; one typically executes the QAOA circuit with a specific set of parameters, passes the result to be processed by a classical machine that determined how to update the parameters to (hopefully) improve the result, and then executes the QAOA circuit with those updated parameters and repeats the process. Finding a set of parameters that reliably produces good solutions to the target optimization problem is extremely challenging and the focus of much ongoing research (Sureshbabu et al., 2023; Fernández-Pendás, Combarro, Vallecorsa, Ranilla, & Rúa, 2021; Streif & Leib, 2019; Hadfield, Hogg, & Rieffel, 2022; Sud et al., 2024). The strategy of running QAOA with a classical outer optimization loop for the parameters may lead to prohibitively many runs of QAOA needed, even with sophisticated optimization techniques because the landscape over the parameters of the solution quality is in general nonconvex with many local minima, and extremely difficult to analyze. Recent work has addressed this issue by investigating *classical* methods to find good parameters for QAOA (Sud et al., 2024). In this work, the authors made use of the experimental observations that states in QAOA circuits tend to be close to *homogenous*, meaning that basis states with the same costs share the same amplitude, which allows to build a simplified, classically tractable proxy for QAOA following the same parameterization as full QAOA. However, this existing model was built specifically for the MaxCut problem and only tested on limited data. In the present report, we aim to improve and extend this classically tractable parameter-setting heuristic. Our results are mixed: on the one hand, we observe some encouraging preliminary data in very simple settings, but on the other hand the methods we have proposed struggle to perform well in even slightly more complicated settings. However, we hope to overcome these shortcomings in the future through some proposed potential improvements to our methodology.

This report is organized as follows. Section 2 provides necessary background on quantum computing and QAOA. Section 3 describes the homogeneous QAOA proxy model which serves as the basis of an existing classical parameter-setting heuristic that we aimed to improve and generalize. Section 5 presents

and summarizes computational experiments performed. Section 4 discusses the performance of methods when the type of random graph used is changed, which highlights challenges in designing a generalizable parameter-setting method for QAOA. Finally, Section 7 describes possible future work through outlining approaches that we see as promising to potentially improve our methods.

1.1 Key Takeaways

We provide here the key takeaways from our research to summarize what we have found throughout the project.

- QAOA performance is extremely sensitive to parameter setting, and parameter landscapes are very difficult to optimize over (see Figure 2). QAOA with poorly chosen parameters seems to be no better than randomly sampling solutions.
- Classical methods for precomputing good parameters are promising, and some interesting ideas exist for this goal. We aimed to improve on a particular existing strategy by refining a parameter-setting approach based on the so-called *homogeneous model* proxy for QAOA (see Section 3).
- Some results in existing work are potentially misleading. Some methods achieve an approximation ratio of 80% for max cut on ER-random graphs, which seems good – however we realized that a similar approximation ratio can be achieved by simply taking a random balanced partition of the sets. This issue may be present in multiple existing works, which puts into question the usefulness of some those approaches (see Section 5.3).
- There are multiple avenues which *could* lead to better or more general (problem-agnostic) classical parameter precomputing methods. However, some modifications we have tried were not successful outside of very particular problem settings, and the models need to be revised to achieve stronger performance in general (see Sections 5, 4, 7).
- The existing literature is very narrow in the problems QAOA is applied to – not only does literature focus strongly on max cut, but even within max cut mostly ER-random graphs are used. This is important because some methods may not generalize well, either to different problems or to different random graphs (Section 4).
- An open-source toolkit for an existing parameter setting heuristic and many QAOA-related utilities was built for this project. This includes Python-compatible code accelerated by a Julia backend, GPU-compatible code, and many utilities for a broad range of QAOA-related tasks (see Section 3.3). Mitsubishi now has a more advanced QAOA simulation package than JP Morgan.

2 Background

2.1 What is a Quantum Computer?

A quantum computer is a computational machine that processes information using the principles of quantum mechanics. While traditional computers (classical computers) process information using bits, quantum computers use quantum bits (qubits). Qubits have a property called superposition, which allows them to be in both the 0 and 1 states simultaneously. In addition to this characteristic of superposition, qubits can also have a special correlation called entanglement. This enables quantum computers to provide a new computational framework that is different from traditional computers.

2.2 Why are Quantum Computers Powerful?

The computational capabilities quantum computers surpass those of traditional computers for certain tasks. We provide brief motivations for the power of quantum computers below.

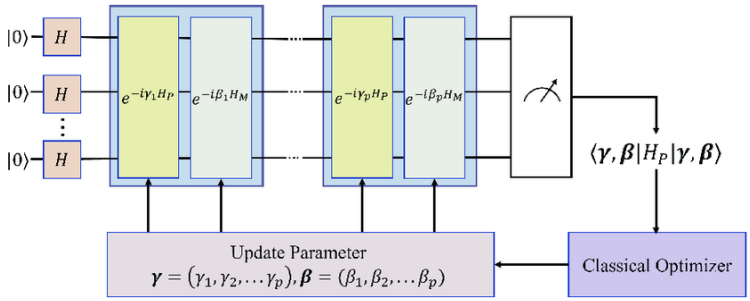

Figure 1: Circuit diagram of the Quantum Approximate Optimization Algorithm (QAOA). Image credit: Huang, Zhaolong et al.(Huang et al., 2022)

2.2.1 Superposition

Qubits can exist in superpositions of both 0 and 1 states simultaneously. Roughly speaking, this means that a quantum computer can perform operations on multiple states. Only one of the states can be measured at one time, so that quantum computers cannot simply perform many operations in parallel on multiple states and measure the result for all states simultaneously. However, clever techniques such as amplitude amplification can be used to take advantage of superposition and achieve speedups over classical computers in certain applications. Additionally, superposition enables entanglement, an important feature of quantum computing.

2.2.2 Entanglement

Entanglement refers to a state in which multiple subsystems of qubits are interrelated in such a way that their relationship cannot be represented by a classical product state. The presence of entanglement allows quantum systems to possess nonclassical correlations, resulting in a system with a much higher dimension than the sum of the dimensions of the subsystems. Consequently, the dimension of the state space representing n qubits is 2^n. Such a quantum state in general has 2^n amplitudes each corresponding to a basis state. This allows a relatively small number of qubits to represent an extremely large amount of information.

2.2.3 Quantum Algorithms

Due to these characteristics, quantum computers have the potential to demonstrate superior performance over traditional computers for specific computational problems. Their capabilities are particularly anticipated in fields such as simulations of quantum many-body systems and cryptographic analysis. For example, Shor's algorithm can efficiently perform prime factorization of large numbers exponentially faster than any known classical algorithm can, though significant advances in hardware and error correction will be needed to make this advantage practically realizable.

2.3 What is a Quantum Circuit?

A quantum circuit is a fundamental concept that represents a sequence of quantum gates operating on qubits. It consists of a series of quantum operations, analogous to classical logic gates in traditional computing, but operating under the rules of quantum mechanics. A quantum circuit typically begins with a set of qubits initialized in a known state (often the ground state), and through the application of quantum gates, it manipulates these qubits to perform computations. In particular, a quantum gate can be applied to inputs in superposition and will return corresponding outputs in superposition.

Figure 1 provides an example of a quantum circuit. In the figure, each horizontal line represents a qubit, and the boxes denote quantum gates applied to these qubits. The arrows indicate the flow of time or

computation, moving from left to right (except for the arrows stemming from the classical optimization and parameter update boxes, which are used to represent a repeating optimization loop).

2.4 NISQ

NISQ stands for "Noisy Intermediate-Scale Quantum" devices. These are current-generation quantum computers that are characterized by:

Limited Qubit Coherence: Quantum states are short-lived due to exposure to environmental noise.

Imperfect Gate Operations: Quantum gates are prone to errors due to noise and imperfect calibration.

Small Qubit Numbers: Current quantum computers have tens to hundreds of qubits, limiting the complexity of computations that can be performed.

Although well-known algorithms such as Shor's algorithm cannot be run on NISQ devices, NISQ devices are still valuable for exploring and experimenting with variational quantum algorithms like QAOA, with the expectation that future advancements will address current constraints.

2.5 Variational Quantum Algorithms

Variational quantum algorithms(Stechly, 2024) leverage the flexibility of quantum circuits by introducing parameters into the gates. These parameters are adjustable values that can be tuned to optimize the output of the quantum computation according to a specific objective or cost function. The key idea is to define a parameterized quantum circuit whose output depends on these parameters, and then optimize these parameters to minimize or maximize the cost function, which represents the desired computational goal. After good parameters are found, one measures the qubits to obtain a good solution to the original problem. The choice of how to parameterize the circuit is called the *ansatz*.

Optimization is a critical aspect of variational quantum algorithms. It involves iterative adjustment of the circuit parameters to find values that yield the best possible solution to the problem at hand. There are various optimization techniques used in quantum computing, such as gradient descent, evolutionary algorithms, and quantum-inspired optimization methods. These techniques aim to efficiently navigate the parameter space of the quantum circuit to achieve optimal performance.

Variational quantum algorithms find applications across diverse fields, including machine learning, quantum chemistry, and optimization problems. They harness the unique properties of quantum mechanics, such as superposition and entanglement, to potentially outperform classical algorithms in specific tasks. This potential for quantum advantage makes variational quantum algorithms a promising area of research and development in the rapidly evolving field of quantum computing.

2.6 Introduction to QAOA

QAOA, or Quantum Approximate Optimization Algorithm(Farhi et al., 2014; Blekos et al., 2024), is a variational quantum algorithm designed to solve combinatorial optimization problems. The key idea behind QAOA is to use a parameterized quantum circuit to prepare a quantum state that represents a superposition of candidate solutions to the optimization problem. By optimizing the parameters of this circuit, QAOA aims to find the optimal (or near-optimal) solution to the given problem.

In QAOA, we aim to find an n-bit binary string $z = z_1 z_2 \dots z_n$ (where each z_i is either a 0 or 1) that minimizes the cost function

$$C(z) = \sum_{\alpha} C_\alpha(z).$$

Here, each $C_\alpha(z)$ is a clause that depends on the bit string z.

To solve this optimization problem, we use an n-qubit system. QAOA utilizes $2p$ parameters: $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$. A quantum circuit is then run to transform the initial state

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} |z\rangle ,$$

where $|+\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$ is the equal superposition state, to the final state

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = U_X(\beta_p) U_C(\gamma_p) \ldots U_X(\beta_1) U_C(\gamma_1) |s\rangle .$$

The unitary operators U_C and U_X are defined by

$$U_C(\gamma_i) = e^{-i\gamma_i C} = \prod_{\alpha} e^{-i\gamma_i C_\alpha}, \tag{2.1}$$

$$U_X(\beta_i) = e^{-i\beta_i \sum_{j=1}^{n} X_j} = \prod_{j=1}^{n} e^{-i\beta_i X_j}. \tag{2.2}$$

The final state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ is a superposition of the bit strings which are candidate solutions of the problem. Measuring the system results in the system collapsing to just one of the candidate solutions. As the cost function approaches its minimum, the probability of reading a bit string for which the cost function is nearly minimized when measuring the state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ is increased. In addition to determining the number of parameters to optimize, p also determines the depth of the quantum circuit, which is the number of "columns" of quantum gates in the circuit, and consequently we sometimes refer to p as the *depth* of QAOA. The depth of a circuit is closely related to the real runtime of a the circuit when quantum hardware (as opposed to classical simulation of a quantum circuit). The problem of finding good parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ is extremely difficult. Figure 2 shows how complicated the landscape of QAOA expectation over different parameter choices can become as p increases.

In applications where QAOA has been successful, each $C_\alpha(z)$ typically depends only on a few of the bits in the bit string z. For example, in the Ising model the cost function is the sum of terms of the form $C_\alpha(z) = z_i \cdot z_j$, which each depend on only two bits. The success of QAOA in these applications might be explained by the results of (Cerezo, Sone, Volkoff, Cincio, & Coles, 2021), which rigorously proves that for a certain class of ansatzes defining the cost function C in terms of global observables results in exponentially vanishing gradients (barren plateaus), while defining C in terms of local variables leads to gradients which vanish at worst polynomially for shallow quantum circuits. Consequently, problems with a local cost function are expected to be more easily optimized than ones with a global cost function.

2.7 Example

We will illustrate QAOA using the Max-Cut problem as an example.

Problem Description: Given an undirected graph, find a way to partition its vertices into two subsets such that the number of edges between the subsets (cut edges) is maximized.

Cost Function: The cost function for Max-Cut can be represented as:

$$C(x) = \sum_{(i,j)\in E} -w_{ij}(1 - x_i x_j) \tag{2.3}$$

where x_i and x_j are binary variables indicating the partition of vertices i and j, respectively, and w_{ij} is the weight of edge (i, j).

QAOA Approach: QAOA constructs a parameterized quantum circuit that prepares a quantum state representing a superposition of all possible partitions of the graph's vertices. By optimizing the parameters of this circuit based on measurements of the cost function, QAOA aims to find a partition that maximizes the number of cut edges.

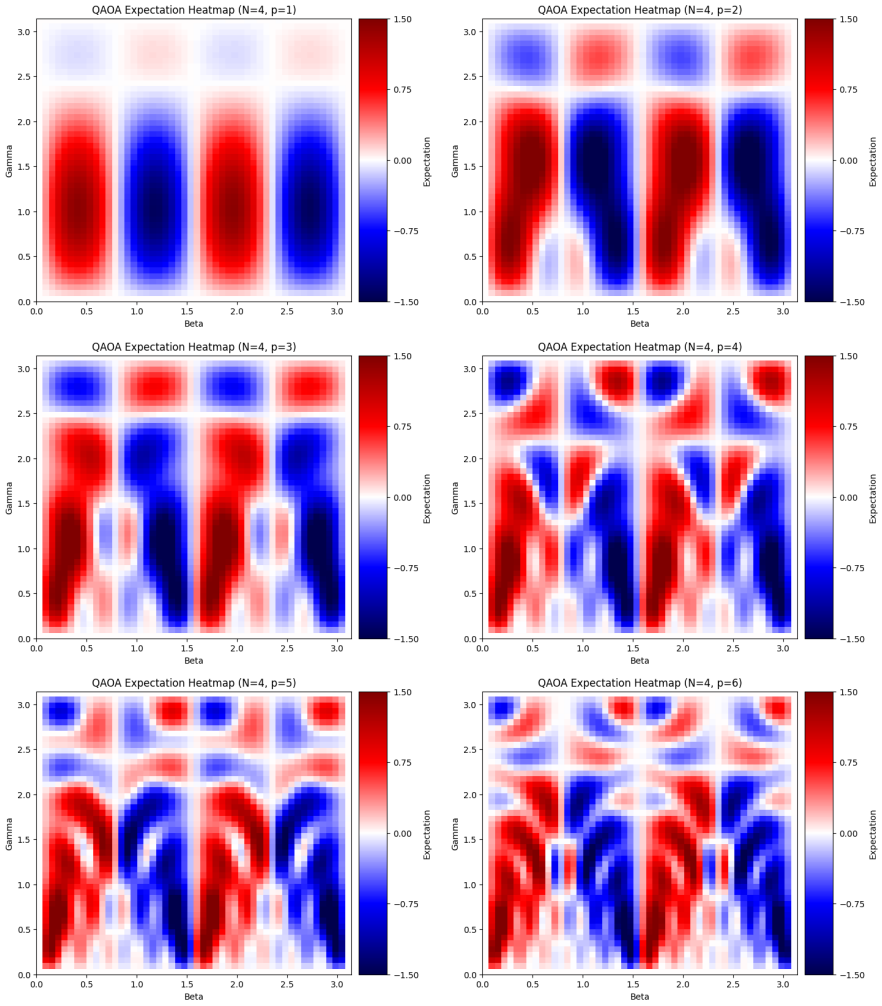

Figure 2: Heatmaps of the expectation landscape of QAOA (over a two-dimensional subspace of the parameter space) with $p = 1, \ldots, 6$ for a single instance of $\mathcal{G}(4, 1/2)$.

Measurement and Optimization: After preparing the quantum state, QAOA measures the expectation value of the objective function $C(x)$. Classical optimization techniques are then employed to adjust the parameters of the quantum circuit to improve the partitioning, iteratively moving towards a solution that approximates the optimal Max-Cut. This example demonstrates how QAOA can be applied to combinatorial optimization problems, leveraging quantum principles to potentially outperform classical algorithms, especially as quantum computing technologies advance.

3 Parameter Setting

3.1 Homogeneous Proxy Motivation

The homogeneous proxy is based on the empirical observation that computational basis states corresponding to bitstrings with the same cost tend to have similar probability amplitudes in the QAOA algorithm. If we assume that the probability amplitudes are *exactly* the same, then the dimension of our system can be drastically reduced, so that the system grows *polynomially* with the problem size instead of exponentially,

which makes the system much less computationally expensive to simulate classically. Here we summarize the arguments by (Sud et al., 2024) and (Hadfield et al., 2022) which motivate the construction of the proxy.

Using the identity $X_j^2 = I$, we can see that

$$\begin{aligned} e^{-i\beta X_j} &= I - i\beta X_j - \frac{\beta^2}{2}X_j^2 + i\frac{\beta^3}{3!}X_j^3 + \frac{\beta^4}{4!}X_j^4 - i\frac{\beta^5}{5!}X_j^5 + \dots \\ &= \left(I - \frac{\beta^2}{2}X_j^2 + \frac{\beta^4}{4!}X_j^4 + \dots\right) + \left(-i\beta X_j + i\frac{\beta^3}{3!}X_j^3 - i\frac{\beta^5}{5!}X_j^5 + \dots\right) \\ &= I\left(1 - \frac{\beta^2}{2} + \frac{\beta^4}{4!} + \dots\right) - iX_j\left(\beta + \frac{\beta^3}{3!} - \frac{\beta^5}{5!} + \dots\right) \\ &= I\cos(\beta) - iX_j\sin(\beta). \end{aligned}$$

Using this identity, along with the fact that the identity and mixing operators all commute ($[I, X_j] = 0$ and $[X_j, X_k] = 0$), we have in (2.2) that

$$U_X(\beta_i) = e^{-i\beta_i \sum_{j=1}^n X_j} = \prod_{j=1}^n e^{-i\beta_i X_j} = \prod_{j=1}^n \left(I\cos(\beta_i) - i\sin(\beta_i)X_j\right).$$

Consider two computational basis states $|x\rangle$ and $|y\rangle$. Then

$$\langle x| I\cos(\beta_i) - iX_j\sin(\beta_i)|y\rangle = \begin{cases} \cos(\beta_i) & |x\rangle = |y\rangle \\ -i\sin(\beta_i) & |x\rangle = X_j|y\rangle \\ 0 & \text{otherwise} \end{cases},$$

which can be generalized to

$$\langle x| \prod_{j=1}^n \left(\mathrm{I}\cos(\beta_i) - i\sin(\beta_i)X_j\right)|y\rangle = (\cos(\beta_i))^{n-d(x,y)}(-i\sin(\beta_i))^{d(x,y)}, \tag{3.1}$$

where $d(x, y)$ is the hamming distance between bitstrings x and y. Using equation (3.1), we can write the probability amplitude $q_\ell(x; \boldsymbol{\gamma_i}, \boldsymbol{\beta})$ of the bitstring x after performing ℓ layers of QAOA (i.e. ℓ applications of U_C and U_X) as

$$q_\ell(x) = \sum_{k=1}^{2^n} q_{\ell-1}(y_k)(\cos(\beta_i))^{n-d(x,y_k)}(-i\sin(\beta_i))^{d(x,y_k)}e^{-i\gamma_i C_{y_k}},$$

where the sum is over all computational basis states, and C_y is the cost associated with bitstring y. The terms above can be grouped as

$$q_\ell(x) = \sum_{d,c}(\cos(\beta_i))^{n-d}(-i\sin(\beta_i))^d e^{-i\gamma_i c} \sum_{\substack{y \text{ s.t.} \\ d(x,y)=d, \\ C_y=c}} q_{\ell-1}(y),$$

where the first sum is over all possible hamming distances and bitstring costs.

If we assume that the state after $\ell - 1$ QAOA layers is homogeneous, then the above equation can be written as

$$q_\ell(x) = \sum_{d,c}(\cos(\beta))^{n-d}(-i\sin(\beta))^d e^{-i\gamma_i c} Q_{\ell-1}(c) n(x; d, c), \tag{3.2}$$

where $Q_{\ell-1}(c)$ is the proxy for probability amplitude (strictly speaking, a real probability amplitude) for bitstrings with cost c after ℓ layers of QAOA, and $n(x; d, c)$ is the number of bitstrings with cost c that are Hamming distance d from the bitstring x.

The probability amplitude in equation (3.2) still depends on the bitstring x. Therefore, computing the probability amplitudes of the state after ℓ QAOA layers requires a separate computation for each bitstring, so that the computational cost grows exponentially as the problem size (e.g. the number of vertices in the graph) increases. However, if we further assume that $n(x; d, c)$ may be replaced by a $N(c'; d, c)$, which is a proxy to the number of bitstrings with cost c which are Hamming distance d from the bitstrings with cost c' then the computational cost will grow with the number of Hamming distances and distinct costs. The number of Hamming distances grows linearly with the problem size, and the number of distinct costs tends to grow polynomially with the problem size, depending on the problem. In general, an exact distribution $N(c'; d, c)$ does not exist, since bitstring with cost c may be a different Hamming distance from two other bitstrings which both have cost c'. However, (Sud et al., 2024) give numerical evidence for the effectiveness of using this assumption as a proxy. Under this assumption, equation (3.2) becomes

$$q_\ell(x) \simeq Q_\ell(c') = \sum_{d,c} \cos(\beta_i)^{n-d}(-i\sin(\beta_i))^d e^{-i\gamma_i c} Q_{\ell-1}(c) N(c'; d, c). \tag{3.3}$$

Now the number of probability amplitudes to be computed only depends on the number of distinct costs in the combinatorial optimization problem, which typically grows polynomially and not exponentially. In the following section

3.2 Our Homogeneous Distribution Proxy

There are two main assumptions underlying the homogeneous model of (Sud et al., 2024):

1. If $c(x) = c(x')$, then $q(x) = q(x')$. That is, if two bitstrings have the same objective cost then they will have the same QAOA amplitudes. Thus $q(x)$ is replaced with $Q(c)$ in the model.
   - This assumption is given the name *Homogeneity*.
   - We were able to obtain some numerical evidence that this assumption is close to true in MaxCut.
2. If $c(x) = c(x')$, then $n(x; d, c) = n(x'; d, c)$. That is, if two bitstrings have the same objective cost then the number of bitstrings Hamming distance d away with cost c will be equal for all d and c. Thus $n(x; d, c)$ is replaced with $N(c'; d, c)$ in the model.
   - This assumption is made in (Sud et al., 2024) primarily out of necessity, since the $n(x; d, c)$ factors in the equation for proxy QAOA evolution before this assumption is applied (i.e. equation (3.2)) are the only factors remaining that depend on individual bitstrings, x, instead of just their costs $c' = c(x)$. Without replacing $n(x; d, c)$ by some $N(c'; d, c)$ depending only on c', it is impossible to reduce the size of "simulating" QAOA down to being polynomial in the size of the problem (and in circuit depth p).
   - Unlike the homogeneity assumption above, the numerical evidence for approximating $n(x; d, c)$ with the $N(c'; d, c)$ proposed in (Sud et al., 2024) is far weaker, especially for large values of c', which we suggest are more important for optimization.

Upon these two assumptions, (Sud et al., 2024) rigorously derive distributions $N(c'; d, c)$ which approximate the average over $n(x; d, c)$ where $c(x) = c'$. In particular, for many instances of constraint satisfaction problems, they propose a multinomial distribution derived from random-SAT.

In section V of (Sud et al., 2024), it is argued that averaging over $n(x; d, c)$ gives a satisfactory proxy for individual such distributions, and that the proposed distributions $N(c'; d, c)$ give decent approximations of these averages.

However, it is notable that for large values of c', both of these claims have significantly weaker numerical evidence (and to some extent, have numerical evidence to the contrary), see for example figure 3. The authors of the original analysis claim that this is acceptable since values of c' far from $\frac{\text{number of constraints}}{2}$ (where the approximation is best) have lesser weight in the proxy for expectation due to having lesser probability,

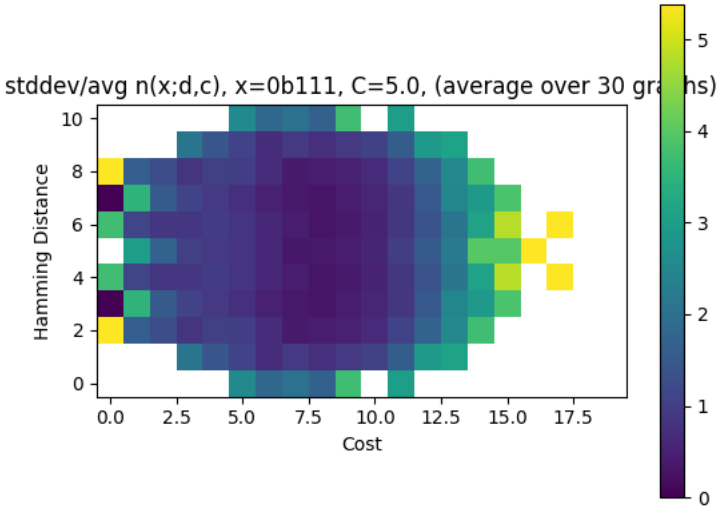

Figure 3: Deviation of $n(x; d, c)$ from the average over x.

but for high values of c' this is partially counter-acted by the weight contributed to these factors by the cost. Furthermore our preliminary exploration suggests that the (proxy) amplitudes on higher costs are the primary contributors to improving the proxy's efficacy. Lastly, computing multinomial distribution values is relatively expensive causing the proxy of (Sud et al., 2024) to be relatively slow.

For all of these reasons, we propose replacing the original $N(c'; d, c)$ with a simpler and more efficient alternative. Furthermore we propose parameterizing this alternative so that we can train our algorithm to "learn" a good shape for $N(c'; d, c)$ for a given problem class. We will refer to the QAOA proxy proposed in (Sud et al., 2024) as the "paper proxy," and our proposed proxy as either "our proxy" or the "new proxy."

Additionally, initial experiments suggest that values for γ and β which maximize the contributions of the highest-cost amplitudes are relatively close to good values for γ and β for actually running QAOA, while the same is not true for the lower-cost amplitudes. Thus, we have found that only including high-cost amplitudes in the final proxy for expectation has a de-noising effect on the optimization landscape. This also lends further credibility to the hypothesis that the homogenous proxy for parameter setting benefits from selecting $N(c'; d, c)$ to be a good approximation of distributions $n(x; d, c)$ for high-cost x, as opposed to averaging over all x. See figure 4. For more significant evidence, we would need to test beyond $p = 1$ and test on graphs besides Erdős-Rényi.

Lastly, having more fine-tuned control over $N(c'; d, c)$ led us to investigate the possibility of encoding known problem symmetries into our proxy distributions. But initial experimentation suggests that the effect of including these symmetries is the introduction of more local maxima in the γ-β optimization landscape (better reflecting the actual QAOA optimization landscape), but no improvement in the actual utility of these local maxima. See figure 5.

As a preliminary result, figure 6 displays heatmaps of the expectation value of the state returned by QAOA with $p = 1$ as β and γ change. Here, $N = 8$ represents the number of vertices (i.e., the number of qubits) and $M = 16$ denotes the number of edges (i.e., the number of constraints/clauses). Although the structure of the global expectation landscapes produced by the paper proxy and our proxy are noticeably different from that of true QAOA, it appears that for this example our proxy is able to capture the location of the global maxima with roughly the same accuracy as the paper proxy.

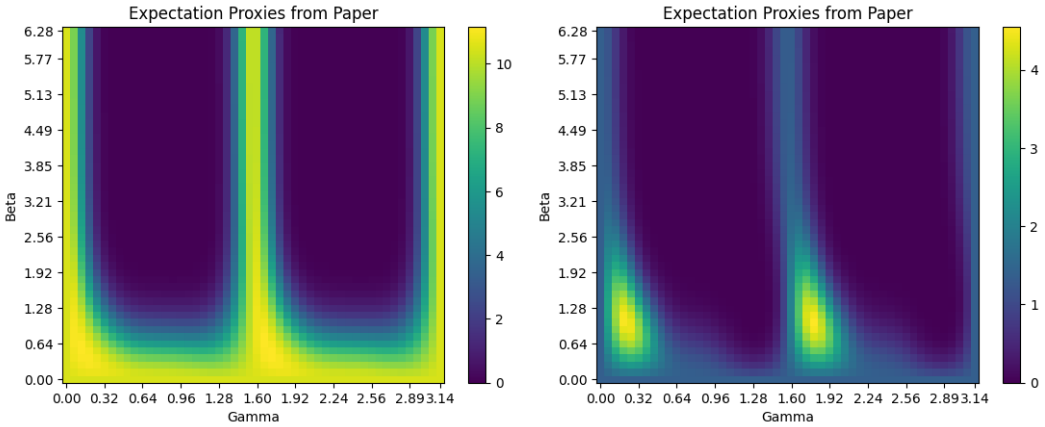

Figure 4: (Left) The results of the original proxy. (Right) The results of only using high-cost amplitudes from the original proxy.

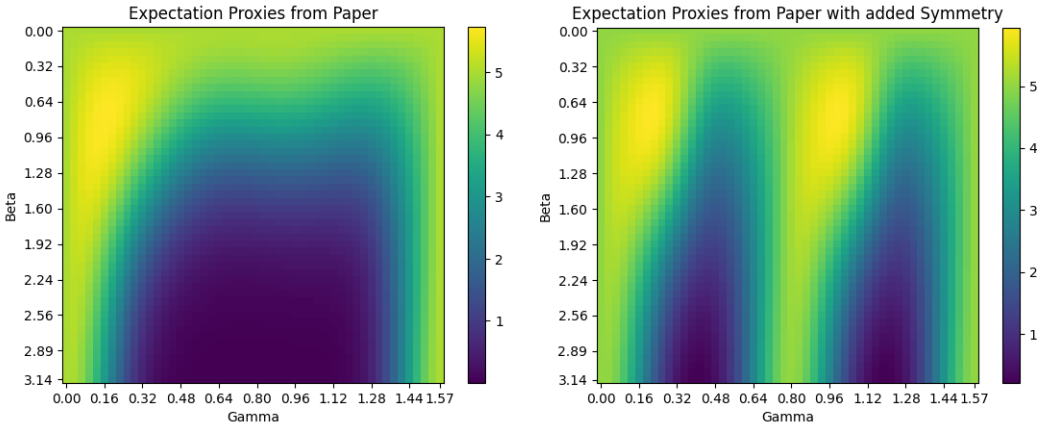

Figure 5: (Left) The results of the original proxy. (Right) The results of reflecting d to lie in $[0, \frac{max\_d}{2}]$.

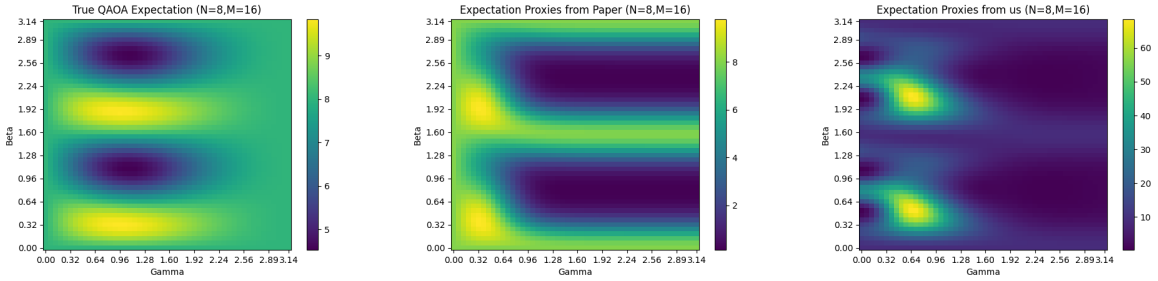

Figure 6: Expectation Map from Real QAOA, Paper Proxy and Our Proxy.

3.3 Julia Programming and GPU Usage

A major obstacle in our comparison of the homogeneous proxies with classical simulations of real QAOA was that our original implementation of the proxy from the paper (Sud et al., 2024) was slow. In general, we expect that sampling from our distribution proxy should be faster than sampling from the paper proxy because our proxy is based on simple RELU-type functions.

In addition, our original implementation of QAOA using the paper proxy was written in Python using the Numpy (Harris et al., 2020) and SciPy (Virtanen et al., 2020) libraries. Computing the probability amplitudes as in equation (3.3) requires nested for-loops to iterate over the possible costs and Hamming distance value. Nested for-loops can be very slow in an interpreted language like Python. The Numba JIT compiler for Python (Lam, Pitrou, & Seibert, 2015) can be used to compile Python code for significant performance improvements (provided the code is simple enough). Indeed, we originally used Numba to JIT compile our QAOA implementation using our new proxy, with great results (100-200x speedup). However, this approach did not work for the paper proxy because the SciPy implementations of the binomial and multinomial calculations were unable to be JIT-compiled by Numba.

To address this, we rewrote our proxy-QAOA implementations in Julia (Bezanson, Edelman, Karpinski, & Shah, 2017), a JIT-compiled language widely used in computational science, and called the compiled functions in Python using the PythonCall and JuliaCall packages (Rowley, 2022). For evaluation of binomial and multinomial distributions, we used the Distributions.jl package for Julia (Lin et al., 2019).

Figure 7 compares the runtime of running QAOA with the paper proxy and our proxy, using the Python and Julia implementations. When reimplementing the paper proxy in Julia, we observed a speedup on the order of 100x. Even after both proxies were implemented in Julia, our proxy executed faster than the paper proxy, with a speedup on the order of 10x.

We also note that the QOKit package supports GPU acceleration. Using an Nvidia A100 Tensor GPU provided by Hiroshi Suito, we were able to perform classical simulations of large QAOA circuits which would have been impractical using CPU-based simulation. A comparison of the runtime of CPU and GPU simulation of 100 iterations of QAOA on a problem of increasing size is shown in figure 8. Using the GPU, we are able to simulate QAOA circuits using up to 30 qubits over a timeframe of minutes to hours, whereas performing those same simulations using a CPU could have taken days to weeks.

3.4 Binomial and Multinomial Approximations

In this subsection, we consider approximations of the binomial and multinomial distributions used in the homogeneous proxy in (Sud et al., 2024), and think of the implications and the meanings of replacing parts of the algorithm with other distributions or methods. This consideration has two purposes. First, it has the aim of developing alternative methods to improve the paper's proxy, by potentially informing our choice of distributions for our approach of parameterizing $N(c', c, d)$. Secondly, we can more fairly compare computation time needed in comparison to the paper proxy by considering computationally inexpensive approximations of the distributions they use instead of computing them exactly. This section is like a study

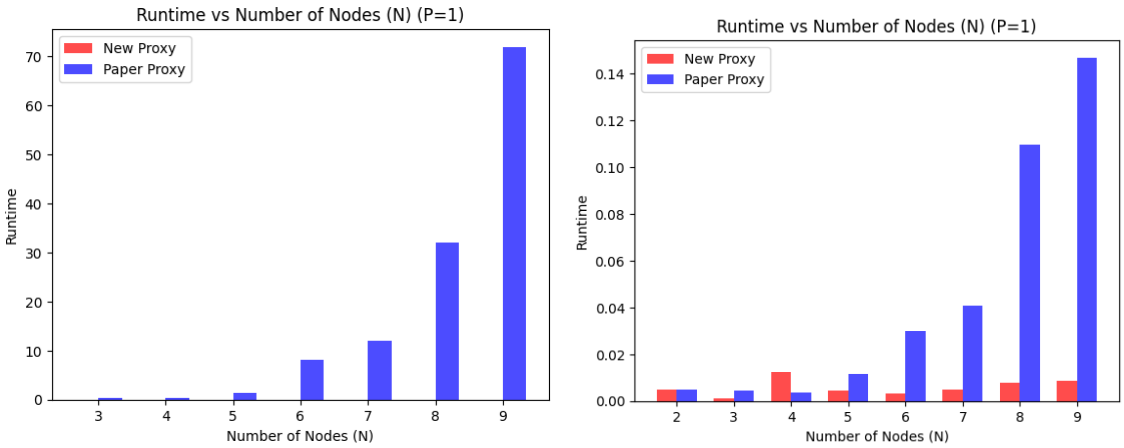

Figure 7: Runtime vs number of nodes (problem size). Left: Python implementations (with Numba for our proxy). Right: Julia implementations.

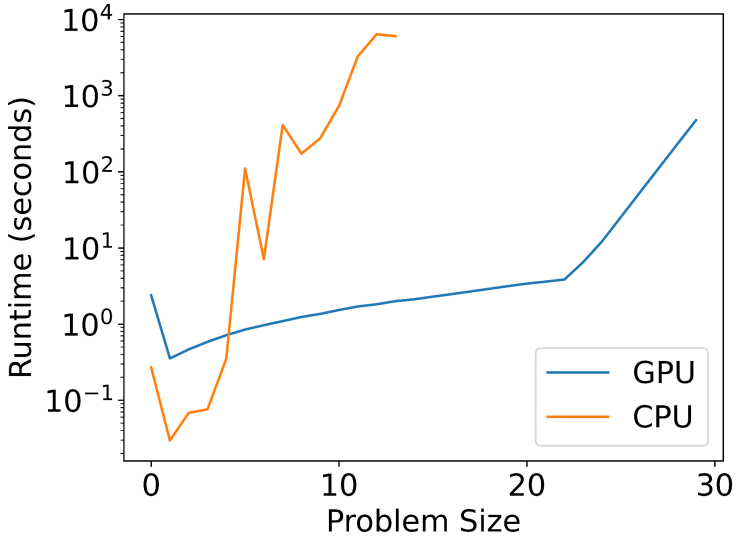

Figure 8: Problem size vs runtime for performing 100 iterations of QAOA, using CPU and GPU simulation.

of these two sides of the same coin. In general, the normal distribution must be the safest alternative approximation, but it is certainly flawed, as we will see later. It is therefore worthwhile to implement or improve other methods, even if they are within a limited scope.

For the binomial distribution, we considered approximation methods using the normal distribution, the Poisson distribution, and the Edgeworth expansion. It is well known that when the number of trials is n and the probability is p, the approximation of the binomial distribution by the normal distribution is expressed as $N(\mu, \sigma^2)$ with mean $\mu = np$ and standard deviation $\sigma = \sqrt{np(1-p)}$. Next, we discuss the approximation of the binomial distribution using the Edgeworth expansion. First, let $z = (x - \mu)/\sigma$, and let the probability density function (PDF) $\varphi(z)$ and cumulative distribution function (CDF) $\Phi(z)$ of the normal distribution be

$$\varphi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right), \quad \Phi(z) = \int_{-\infty}^{z} \phi(t)dt.$$

The Edgeworth expansion has skewness γ_1 and kurtosis γ_2 coefficients caluculated as

$$\gamma_1 = \frac{1-2p}{\sqrt{np(1-p)}}, \quad \gamma_2 = \frac{1-6p(1-p)}{np(1-p)},$$

and the correction term $\varepsilon(z)$ in the Edgeworth expansion is given by

$$\varepsilon(z) = \frac{\gamma_1}{6}(z^2-1)\varphi(z) + \frac{\gamma_2}{24}(z^3-3z)\varphi(z) - \frac{\gamma_1^2}{36}(z^5-10z^3+15z)\varphi(z).$$

The CDF approximation using the expansion is $\Phi_{\mathrm{E}}(z) = \Phi(z) + \varepsilon(z)$. Thus, we can take the following difference to obtain the probability mass function (PMF) approximation from the $\Phi_{\mathrm{E}}(z)$:

$$\mathrm{PMF}_{\mathrm{E}}(k) = \Phi_{\mathrm{E}}(k) - \Phi_{\mathrm{E}}(k-1).$$

In Figures 9 and 10, we compare these approximations with the binomial distribution in terms of PMF values, runtime, and accuracy in both absolute mean error and mean squared error, where the probability is fixed at 0.5 to match the proxy of (Sud et al., 2024). The Edgeworth expansion method is by far more accurate than other approximate distributions except the normal. On the other hand, care must be taken with the runtime. If the number of trials n becomes too large, the execution time increases, and at $n = 100000$, there is almost no difference from the binomial distribution. Here n also means the number of constraints for the problem we are dealing with. Research is needed to overcome this difficulty.

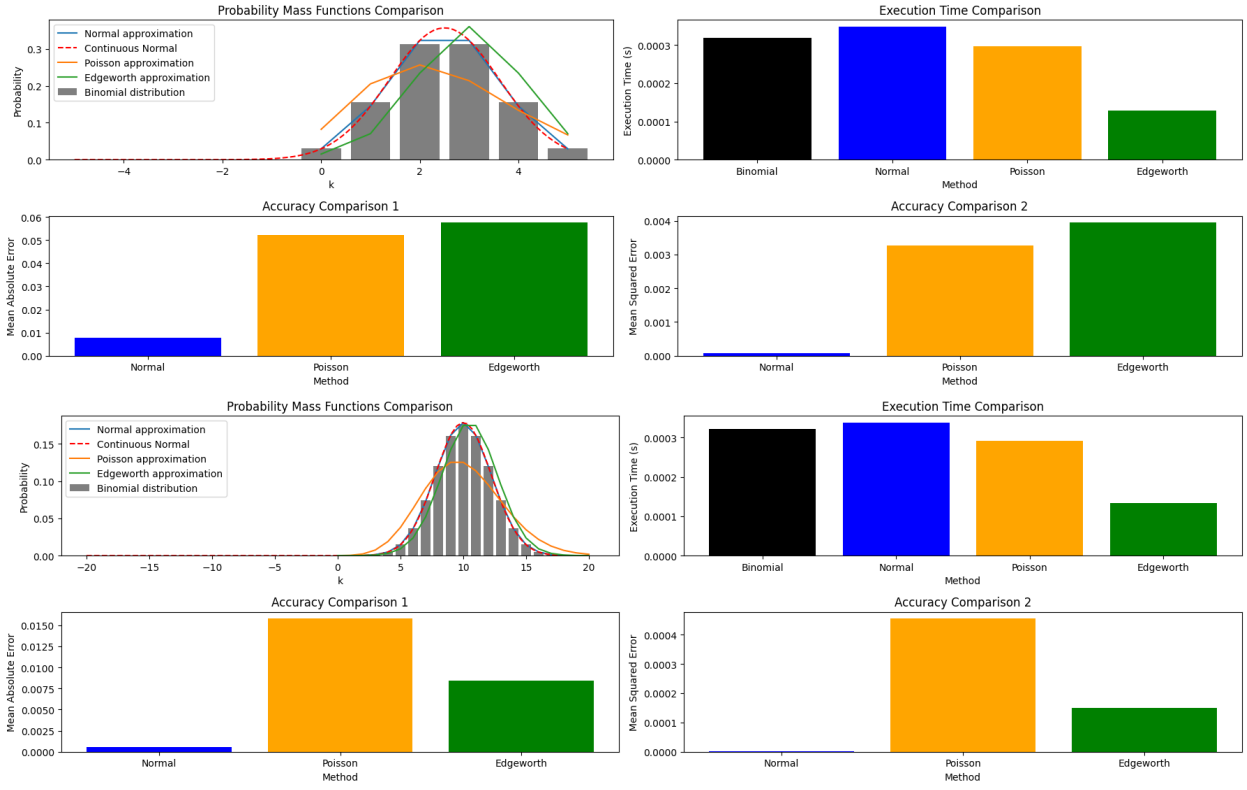

Figure 9: Comparison of binomial distribution and its approximations with the number of smaller trials $n = 5, 20$ (from top to bottom) and the probability at 0.5.

Next, we attempted to make the Edgeworth algorithm faster as an approximation method with potential for speed-up. Specifically, where the naive method used the SciPy library to compute the PDF and CDF

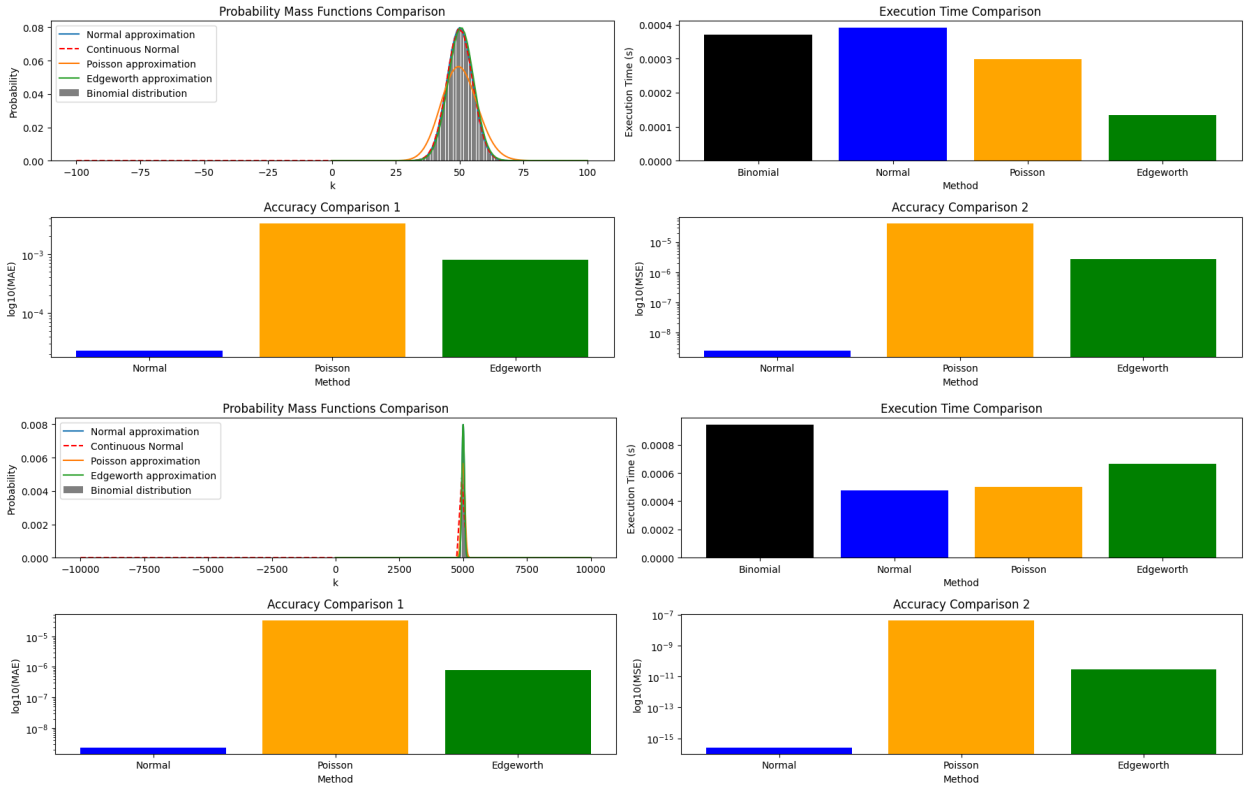

Figure 10: Comparison of binomial distribution and its approximations with the number of greater trials $n = 100, 10000$ (from top to bottom) and the probability at 0.5.

of the normal distribution, instead, the PDF was calculated directly $\varphi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-z^2/2\right)$ and the error function was used for the CDF:

$$\Phi(z) = \int_{-\infty}^{z} \varphi(t)dt = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)\right],$$

where

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt.$$

At the same time, we also tried to improve accuracy by adding higher-order terms such as

$$\Phi(z) + \frac{\gamma_1}{6}(z^2-1)\varphi(z) + \frac{\gamma_2}{24}(z^3-3z)\varphi(z) - \frac{\gamma_1^2}{36}(z^5-10z^3+15z)\varphi(z) + \frac{\gamma_1^3}{120}(z^6-15z^4+30z^2-20)\varphi(z)$$

in the accelerated Edgeworth method.

The results of a comparison between the normal distribution and several Edgeworth methods for the binomial distribution are shown in Figure 11 and Figure 12. For better or worse, the accuracy remained largely the same, but we were able to speed up the Edgeworth method significantly. In particular, for problems that are not very large, there is little difference in execution time, even with higher order terms. Here we need to mention the case where n becomes large: if n is sufficiently large (e.g. $n = 10000$), the Edgeworth method is still not so dominant over the normal distribution. In conclusion for this discussion, when using the Edgeworth method as an approximation method for the binomial distribution to increase speed, one must be careful about the size of the variable corresponding to the number of trials.

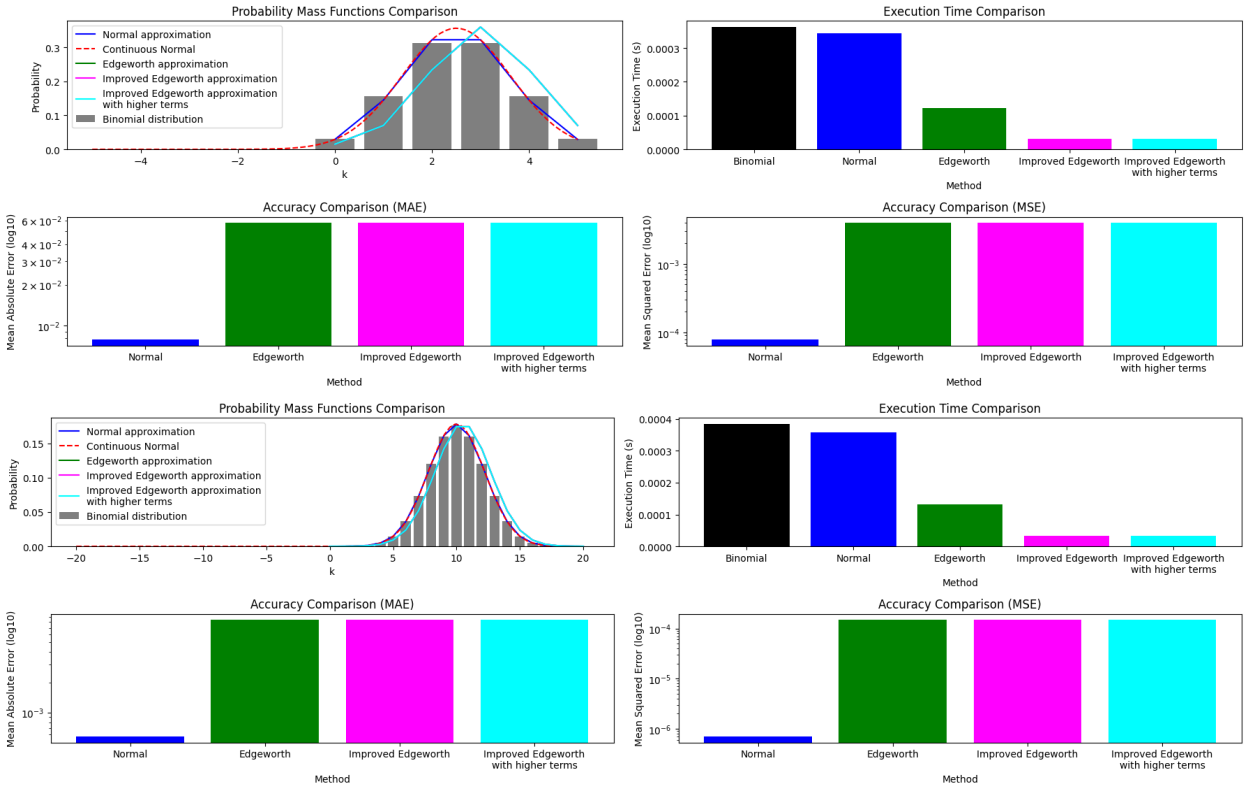

Figure 11: Comparison of binomial distribution and various Edgeworth methods with the number of smaller trials $n = 5, 20$ (from top to bottom) and the probability at 0.5.

For the multinomial distribution, we approximate it using the normal distribution, the Poisson distribution, the Laplace approximation, the Edgeworth expansion method, and the Markov chain Monte Carlo (MCMC) method. The first approximation can be expressed by the multivariate normal distribution with mean $\mu_i = np_i$ and variance $\sigma^2 = np_i(1-p_i)$; the PMF calculation is $\prod_{i=1}^{k} \varphi((x_i - \mu_i)/\sigma_i)$. The Poisson approximation which assumes the counts x_i are able to be approximated by independent Poisson distributions for (small) probabilities p_i and large n has Poisson parameters $\lambda_i = np_i$, and the PMF is calculated as $\prod_{i=1}^{k} e^{-\lambda_i} \lambda_i^{x_i}/(x_i!)$. The method using the Laplace approximation requires mean $\mu_i = np_i$ and variance $\sigma_i = np_i(1-p_i)$ and its PMF is caluculated by the product of the multinomial coefficient $n!/(x_1! \cdots x_k!)$ and the normal approximation term which is the same as the PMF calculation for the first method. In addition to the two kind of values, (standardized) counts $z_i = (x_i - \mu_i)/\sigma_i$, skewness $\gamma_{1,i} = (1-2p_i)/\sqrt{np_i(1-p_i)}$, kurtosis $\gamma_{2,i} = (1 - 6p_i(1-p_i))/(np_i(1-p_i))$ are necessary for the Edgeworth method, and there are three kinds of correction terms of it:

$$\frac{\gamma_{1,i}}{6}(z_i^3 - 3z_i)\varphi(z_i), \quad \frac{\gamma_{2,i}}{24}(z_i^4 - 6z_i^2 + 3)\varphi(z_i), \quad \text{and} \quad \frac{\gamma_{1,i}^2}{72}(z_i^6 - 15z_i^4 + 45z_i^2 - 15)\varphi(z_i).$$

The approximation, the Edgeworth PMF, is finally represented as the product of the following three elements: the multinomial coefficient, the normal approximation term, and the products of all

$$\Phi(z_i) + \frac{\gamma_{1,i}}{6}(z_i^3 - 3z_i)\varphi(z_i) + \frac{\gamma_{2,i}}{24}(z_i^4 - 6z_i^2 + 3)\varphi(z_i) + \frac{\gamma_{1,i}^2}{72}(z_i^6 - 15z_i^4 + 45z_i^2 - 15)\varphi(z_i).$$

The MCMC method simulates to approximate the multinomial PMF by generating samples from the multinomial distribution and calculates the frequency of the observed counts. More specifically, it generates a

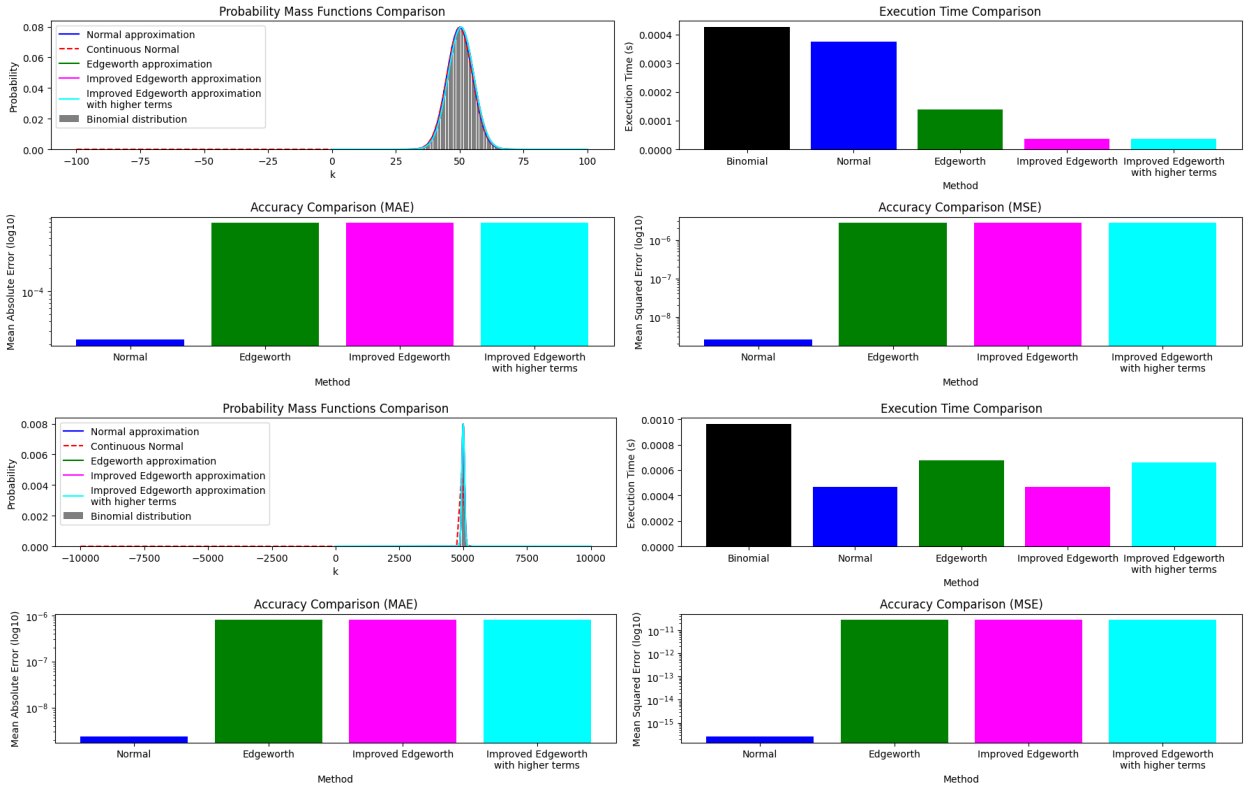

Figure 12: Comparison of binomial distribution and various Edgeworth methods with the number of greater trials $n = 100, 10000$ (from top to bottom) and the probability at 0.5.

large number of samples (e.g., 10,000) from the multinomial distribution with given the number of trials (that is, constraints for our research) and the probability and computes the proportion of the samples that are consistent with the observed counts for the PMF of this.

Figures 13 and 14 compare the multinomial distribution and their approximation methods from the same perspectives as for the binomial distribution, where the size of categories is fixed at 4 in the context of (Sud et al., 2024) as well. The approximation of the multinomial distribution can be seen directly in the comparison of the probability mass functions in the upper left-hand corner of each figure, and in the lower part of each figure, where the actual errors are shown in terms of absolute mean error and mean squared error. There is also a bar chart in the top right hand corner comparing runtimes. If one focuses only on the accuracy of the approximation, the MCMC method appears to be the most effective. However, this alone cannot simply be judged as an effective approximation. In fact, as can be seen from the figures, the method has serious shortcomings in terms of runtime. Conversely, if MCMC methods could be successfully speeded up by some improvement measures, they could replace the multinomial distribution in the proxy of (Sud et al., 2024). At present, the MCMC method is rather naively coded and there is room for improvement in the algorithm; efficiency gains could be achieved by using the Metropolis-Hastings algorithm, Gibbs sampling and Hamiltonian Monte Carlo, which are currently under consideration. Other possible improvements include parallel processing such as multi-threading, multi-processing or vectorisation, and the use of GPUs to perform massively parallel computations.

Comparing the above discussion on the approximation of the two distributions with the results of their experiments, the approximation method using the normal distribution can be seen as a compromise, but even it is also not perfect as the figures show. Whether to fully trust the normal distribution or to focus on improving the efficiency of other approximation methods is both an issue for the future.

4 Changing the Random Graph Distributions

In studies of the MaxCut problem using the QAOA, research has predominantly focused on specific families of graphs, namely d-regular graphs and Erdős–Rényi (uniform random) graphs, or ER random graphs (Basso, Farhi, Marwaha, Villalonga, & Zhou, 2022; Wurtz & Love, 2021; Rančić, 2023; Farhi et al., 2014). In our research, we plan to extend the scope of experimentation to include Barabási–Albert Scale-Free Graphs and Watts–Strogatz Small-World Graphs.

Barabási–Albert Scale-Free Graphs are characterized by a power-law degree distribution, where a few nodes (hubs) have a very high degree while the majority of nodes have a relatively low degree. This type of graph models many real-world networks, such as social networks, citation networks, and the World Wide Web, where some nodes act as highly connected hubs (Barabási & Albert, 1999). In such graphs where there are a few nodes with very high degree, a max cut solution will typically have those nodes in one side of the partition, and the many nodes with few connections in the other side. In contrast, ER-random graphs typically have max cut solutions yielding partitions of similar size. Hence, the structure of the instances and typical solutions under the Barabási–Albert random graph model are very different from when one uses ER-random graphs, so it is an informative class of max cut problems to investigate.

Watts–Strogatz Small-World Graphs, on the other hand, exhibit high clustering and short average path lengths, capturing the small-world phenomenon observed in many real-world networks. These graphs are created by randomly rewiring edges of a regular lattice, resulting in a structure that maintains high local clustering while allowing for short global connections. This model is often used to represent social networks, neural networks, and other systems where local clusters and long-range connections coexist (Watts, 1998). This provides another interesting class of max-cut instances with its own structure for investigating algorithm performance.

Currently, neither our proxy nor the proxy of (Sud et al., 2024) are capable of accurately capturing the location of the global maxima on heatmaps for the MaxCut problem for graphs other than Erdős–Rényi graphs. Further research is required to ensure that high expectation values can be obtained at the same locations as real QAOA for other graph models. Figure 15 illustrates this issue using a random graph generated by the Barabási–Albert preferential attachment model.

The problem of parameter-setting for QAOA remains a significant challenge for effective implementations of the algorithm. We investigated an existing classical method to precompute parameters as studied in (Sud et al., 2024), and proposed two modifications. First, to use simple approximations of a distribution that the method depends on, and secondly to parameterize this distribution with the goal of being able to adapt the method to new problem classes in an automatic way. We presented some computational experiments, where for the $p = 1$ case our method seemed to give some improvement over the existing method of (Sud et al., 2024). However, neither method produces particularly good end results in that case, because QAOA with depth $p = 1$ on Erdős-Rényi random graphs simply does not perform well compared to even generating random solutions to Max Cut. Further, our method has struggled to work well with larger depth $p > 1$, or with different types of random graphs.

5 Results

5.1 Computational Scaling Experiments

Our first objective in the project was to run simulations of QAOA to examine how the algorithm scales with problem size and circuit depth, which was done by using the Python package QOKit(Lykov, Shaydulin, Sun, Alexeev, & Pistoia, 2023). These experiments were not novel research, but they were an important step for understanding the computatioanl challenges of QAOA. As our test problem, we use the Max-Cut cost function (2.3) Erdős-Rényi graphs $\mathcal{G}(N, P)$. N denotes the number of vertices, and each pair of vertices has a probability P of being connected by an edge.

Figure 16 compares the circuit depth against the expectation value of the cost function after optimizing $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, as well as the number of QAOA calls needed to optimize $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. The expectation is the main

measure of the quality of the solution obtained by QAOA, as it represents the average cost of the bitstrings that QAOA outputs after the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ have been tuned. The number of QAOA calls is the main measure of the expected runtime of the QAOA algorithm on an actual quantum computer, since we can roughly assume that a single QAOA call takes a constant amount of time to execute (in practice this depends on the hardware implementation of the ansatz).

Figure 17 shows heatmaps of the expectation landscape of QAOA for single instances of $\mathcal{G}(N, 1/2)$ for $N = 1, \ldots, 6$. As N increases, the expectation landscape appears to converge, even though each graph instance is random. This may be related to the observation by Streif and Lieb (Streif & Leib, 2019) that the distributions of optimal QAOA parameters for 4-regular graphs become narrower as the system size becomes larger.

Figure 2 shows heatmaps of the expectation landscape of QAOA with $p = 1, \ldots, 6$ for a single instance of $\mathcal{G}(4, 1/2)$. We observe that the landscape becomes more complicated as p increases and potentially contains multiple local minima, even though the problem graph is very small with only 4 vertices. For the higher-dimensional parameter spaces ($p > 1$) we cannot visually evaluate the parameter landscape fully, but the cross-section heatmaps in Figure suffice to demonstrate that they are difficult to optimize over.

5.2 Paper Proxy and New Proxy Comparisons

In this section, by using the new proxy under conditions where the paper proxy performs poorly, we aim to demonstrate that the QAOA scaling technique is an effective method for the max-cut problem on Erdős-Rényi graphs, which have garnered attention in network science. Moreover, Erdős-Rényi graphs are very simple to implement and analyze, and consequently they are commonly used when evaluating the effectiveness of algorithms for solving MaxCut.

5.2.1 Runtime

One of the most significant advantages of using our new proxy is its reduced runtime. As described in Section 3.3, the QAOA-proxy evaluations are implemented in Julia, with Python serving as "glue code" for tasks such as managing the QAOA-proxy optimization loop. Figure 18 compares the runtime of optimizing the QAOA-proxy using both the paper proxy and our new proxy, with both implementations in Julia.

For this experiment, we initialized $\boldsymbol{\gamma} = \boldsymbol{\beta} = [0.1, \ldots, 0.1]^T$ and measured the time required to complete one run of each proxy, i.e., a single simulation of the QAOA circuit. The results were averaged over 100 runs to ensure statistical significance. The graph used in this comparison is an Erdős-Rényi random graph with a probability $p = 0.5$ for edge formation and varying the number of nodes (N) from 3 to 10.

We observe that our method consistently outperforms the paper proxy in terms of execution time, particularly as the number of nodes increases. This reduction in runtime underscores the efficiency of our new proxy for combinatorial optimization problems using QAOA. Specifically, our new proxy is more efficient than the paper proxy across different scenarios, as demonstrated by the shorter runtimes, especially when the parameter P (the number of layers in the unitary gate) is small (e.g., $P = 1$) and when it is larger (e.g., $P = 5$).

5.2.2 Approximation Ratio

The approximation ratio is one measure of the quality of the solution obtained by the QAOA algorithm. In this case, it is the expectation value of the state produced by QAOA divided by the cost associated with the optimal solution of the combinatorial optimization problem, which is the maximum value along the diagonal of the cost Hamiltonian (in the computational basis):

$$\text{approxmation ratio} = \frac{\text{expectation}}{\text{maximum cost}}. \tag{5.1}$$

The value of the approximation ratio indicates how well QAOA optimized over all potential solutions, as measured by the objective function. Figure 19 compares the approximation ratios obtained when using $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ by optimizing proxy-QAOA using the paper proxy and our proxy. We now describe the conditions of this experiment. The initial values $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are all set to $[0.1, \ldots, 0.1]^T$. For each graph, the parameters are optimized using the paper proxy and the new proxy, giving us two optimized values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Using the values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ obtained using the paper proxy and the new proxy, we calculate the expectation of the state vector after running real QAOA with those values, and use the result to compute the approximation ratio for the paper proxy and new proxy values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. We average this result over T random graphs. The results of this experiment are shown in a bar graph in figure 19.

For the minimal circuit depth $p = 1$, the approximation ratios obtained using the paper proxy and our proxy are comparable, and in most cases our proxy achieves a higher approximation ratio. Because our proxy executes much faster than the paper proxy, we conclude that for the minimal circuit depth $p = 1$, on this particular class of graphs, and for the graph sizes tested in this experiment, our proxy is superior at parameter setting for real QAOA. Granted, the maximum graph size we consider is 10 vertices, which is not very large. Moreover, for circuit depth $p = 5$ the approximation ratios obtained by the new proxy are generally inferior to those obtained by the paper proxy. Because the maximum expectation achievable by QAOA is expected to increase with p, this demonstrates that more work is needed on our proxy in order for it to be useful.

5.2.3 Success Probability

The success probability is a measure of how well the obtained solution agrees with the exact optimal solution, i.e. how close the probability amplitude of the optimal state is to one. A high overlap between the final state of QAOA with the optimal state means high success probability, which is the probability of a run of QAOA on quantum hardware producing the optimal bitstring. A solution produced by QAOA may have a high approximation ratio but a low success probability if, for example, the QAOA solution has a high probability amplitude associated with bitstrings which have costs *almost* as high as the optimal solution. A comparison of the success probability obtained after optimizing parameters using the QAOA-proxies is provided in figure 20.

The procedure for this experiment is the same as the procedure described in section 5.2.2 for computing the approximation ratios, only instead of using the parameters output by the proxies to compute the approximation ratio, we use them to compute the success probability (overlap between final state and the computational basis state with the highest cost).

The results show that for small graphs $5 \leq N$ and minimal circuit depth $p = 1$, the new proxy gives comparable or superior success probabilities as the paper proxy. The success probability obtained by the paper proxy is small, but significantly larger. For a larger circuit depth $p = 5$, and small graph size $N \leq 5$, the new proxy obtains success probabilities comparable to (but still less than) those from the paper proxy. And for $N \geq 5$, our proxy begins to give significantly worse success probabilities. However, we remark that a low success probability does not necessarily mean the parameter setting was a failure. For example, some parameters may cause the final QAOA state to be the computational basis state with the second-highest cost. In that case, the success probability would be zero, even though the the solution obtained is the second best possible solution, which for practical purposes would be considered a great success.

5.3 Discussion

The results of section 5.2 show that for minimal depth $p = 1$ circuits, our proxy can outperform or at least nearly match the paper proxy in terms of quality of the parameters obtained. Moreover, our proxy is much faster than the paper proxy, and therefore in the extremely limited case of $p = 1$ for Erdős-Rényi graphs $\mathcal{G}(N, 1/2)$ we have obtained a significant improvement.

However, late into the project we made a surprising discovery which potentially indicates a flaw in our methodology and the methodology of (Sud et al., 2024). Any two vertices in an instance of $\mathcal{G}(N, 1/2)$ have a probability of $1/2$ of being connected by an edge. Therefore, if choose a solution bitstring which consists

of half 1's and half 0's, but is otherwise random, we will on average cut half of the edges in the graph. Furthermore, we have found in our experience that the maximum cut for most instances of $\mathcal{G}(N, 1/2)$ is only slightly more than half the number of edges. Consequently, the simple classical strategy of choosing a random bitstring with an equal number of 1's and 0's typically achieves approximation ratios in the range of 75%-85%. The approximation ratios achieved by both our proxy and the paper proxy rarely exceed 85%, which makes the results of both proxies far less impressive than we initially thought. For more structured graphs, such as those discussed in section 4, this should not be an issue, although significant adjustments and improvements must be made to our proxy before it can be effectively applied to new classes of graphs.

While developing our new proxy, we also made discoveries which could be used to improve the paper proxy.

By encoding known problem symmetries into our proxy distributions, we found that Erdős-Rényi graphs with $p = 1$, the proxies could capture more local minima than the original paper proxy could, as demonstrated in figure 5. Before this change, the paper proxy predicted half as many local minima as there were in the real QAOA landscape. However, the quality of each local minima was not significantly affected by the added symmetry, so the utility of this change is limited.

Finally, we found that for Erdős-Rényi graphs with $p = 1$, ignoring low-cost amplitudes in the proxy has a denoising effect on the optimization landscape, as demonstrated in figure 4. If this result generalizes to $p > 1$ and graphs other than Erdős-Rényi, it could result in improvements to the optimization in the parameter setting process.

6 Conclusions

The following conclusions can be drawn from our work. Our proxy can, for a specific class of graphs (Erdős-Rényi) and for a circuit depth of $p = 1$ (which is a *severe* limitation), achieve results similar to or better than the paper proxy in a much shorter runtime. In this context, we have found that enforcing known symmetries of the problem into our model results in an optimization landscape which captures more local minima. We also observed a denoising effect on the optimization landscape when ignoring low-cost amplitudes in our proxy. Although it is difficult to identify the consequences of the latter two observations, they could potentially make optimizing the expectation landscape easier, leading to faster parameter-setting. For graphs other than Erdős-Rényi, our method does not currently perform well, and the same is true for larger circuit depths, but optimization and machine learning techniques could possibly be used to adjust our model (which is designed to be flexible and easily modifiable) to achieve greater performance on other classes of graphs.

7 Future Work

We have studied the problem of parameter setting in QAOA and investigated approaches for classically precomputing QAOA parameters. We have proposed modifications to existing methods, for which numerical experiments have yielded mixed results. This makes us believe that our ideas are promising but require additional modifications and improvement strategies to be effective and comprise a meaningful contribution to cutting-edge QAOA research. In this section, we present these directions that we hope to investigate in the future.

There are three primary avenues of unresolved investigation:

1. An empirical investigation of difference-making components of the homogeneous proxy.
2. Attempting to replace the mathematical model of a distribution $N(c'; d, c)$ in the homogeneous proxy with fitting to or learning from sample data.
3. Attempting to replace the mathematical model of a distribution $N(c'; d, c)$ in the homogeneous proxy with members of a parameterized family of distributions which can be optimized/learned from small problem instances.

The homogeneous proxy proposed in (Sud et al., 2024) contains some justification of the efficacy of their proposed distributions; namely, they explicitly or implicitly make three claims:

(a) That their distributions are good approximations of the average of real distributions.

(b) That averages of real distributions don't deviate too much from real distributions

(c) That using good approximations of real distributions within the homogeneous model yields good results.

Claim (c) can be experimentally verified directly by replacing models for $N(c'; d, c)$ in proxy execution with actual distribution data, but this has not yet been done. A positive result in experimentally verifying claim (c) would suggest that point 2 above is promising because we may be able to better fit/learn real distributions than the mathematical models, and further we may be able to accomplish this in a problem-agnostic way.

Claim (b) is justified numerically in (Sud et al., 2024) for values of c' close to $\frac{|E|}{2}$ but results are far less conclusive for larger values of c'. Our preliminary investigations of the impact of low, middle, and high cost amplitude proxies on the final expectation proxy seems to suggest that high-cost amplitudes are central to the efficacy of the homogeneous proxy while middle and low cost amplitudes have very little positive effect on the fitness of the resulting parameter setting landscape (for $p = 1$). Further experimentation is required to make any solid conclusions on this topic. But if these observations hold in any generality, it would suggest that point 3 above is promising because we may be able to more optimally weight our amplitude contributions to the final expectation proxy, as well as better tune $N(c'; d, c)$ to work optimally within the context of the homogeneous proxy where true distribution values may not be the optimal solution.

If the homogeneous proxy reacts positively to accurate approximations of real distribution data (especially as p increases), then rather than using a mathematical model approximating $N(c'; d, c)$ for a class of problems, we instead propose a methodology of sampling from $N(c'; d, c)$ for the individual problem instance one is trying to solve and then fitting a distribution (e.g. Gaussian) to this data to play the role of $N(c'; d, c)$ in the homogeneous proxy. This procedure has the benefit of being widely applicable beyond MaxCut as it is problem-agnostic and also avoids the possibility that claim (b) may not be true for other kinds of problems (and may even be somewhat false for MaxCut). Thus, it is possible that this approach may outperform the originally proposed proxy while still being classically computable in polynomial time.

In the case that fitting real distribution data does not yield optimal parameter-setting results in the homogeneous proxy model, replacing $N(c'; d, c)$ with a family of distributions optimized for parameter-setting performance (and not for fidelity towards real distributions) could yield better performance and accuracy compared to the originally proposed proxy. Initial experimentation suggests that this approach is likely effective when $p = 1$, but may require more distribution parameters as p increases. However, initial experimentation also suggests that this approach yields a relatively difficult optimization problem and that naive approaches to optimizing $N(c'; d, c)$ yield poor results, even for $p = 1$. Nonetheless, it remains possible that some of these obstacles can be overcome (e.g. by using a fit to real distribution data to set initial parameters) and that this approach could simultaneously yield better approximation ratios while also being problem agnostic.

7.1 Target Schedule for Academic Paper Publication

We would like to write a paper if our proxy results with the modifications we have proposed to investigate next (parameterization to make it applicable to a wider class of problems) are successful and constitue a meaningful scientific contribution, which we hope to be the case. The journal *Mathamatics* (MDPI) is having a special issue in *Quantum Computing Algorithms and Quantum Computing Simulators* (see here https://www.mdpi.com/journal/mathematics/special_issues/Quantum_Comput). Our work would fit extremely well in the scope they have defined and the turnaround time of the review process is very fast. The deadline

to submit is December 31, 2024, so this can be our target for a full, thorough paper. We will submit to ArXiv at least two weeks before this date so that we can possibly obtain community feedback before making the submission. Another option would be the open journal *Quantum*, which is very well-regarded by the community, but has extremely slow processing times. If possible, we will try to submit at an earlier date than the stated deadlines.

Acknowledgements

We would like to than the organizations and individuals who made this research possible. Thank you to Mitsubishi Electric for sponsoring this project, and to University of California Los Angeles, the Institute for Pure and Applied Mathematics (IPAM), Tohoku University, the Advanced Institute for Materials Research (AIMR), the Mathematical Science Center for Co-creative Society and the Tohoku Forum for Creativity for organizing the Graduate-level Research in Industrial Projects for Students (G-RIPS) Sendai 2024 program. Also special thanks to Hiroshi Suito for directing the G-RIPS program, and providing us access to an NVIDIA A100 Tensor Core GPU, which was essential for our ability to simulate QAOA for large numbers of qubits. Thanks also to all Tohoku University staff who helped ensure the smooth operation of the GRIPS program throughout.

Figure 13: Comparison of binomial distribution and its approximations with the number of smaller trials $n = 2, 3$ (from top to bottom) and 4 categories.

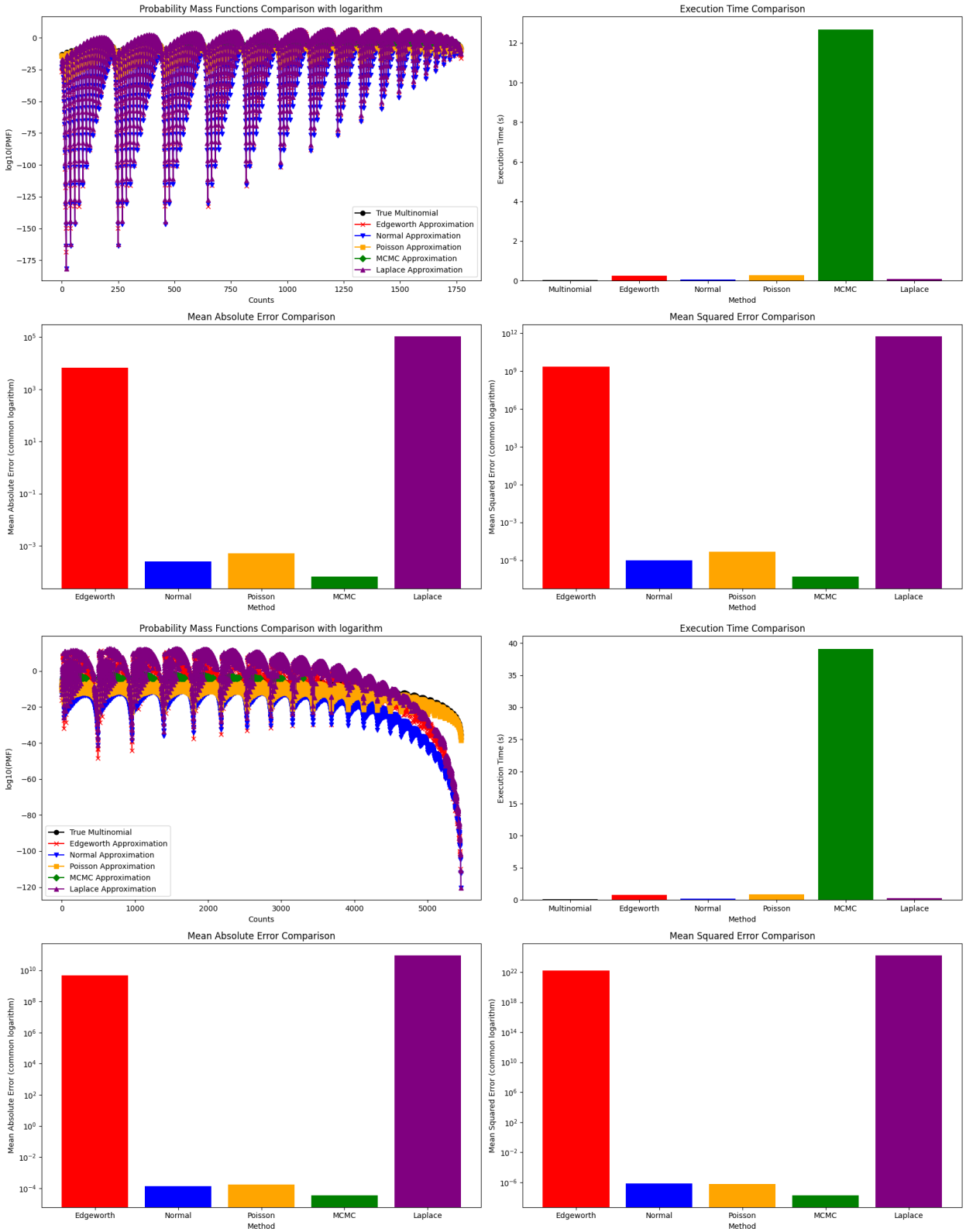

Figure 14: Comparison of binomial distribution and its approximations with the number of greater trials $n = 20, 30$ (from top to bottom) and 4 categories.

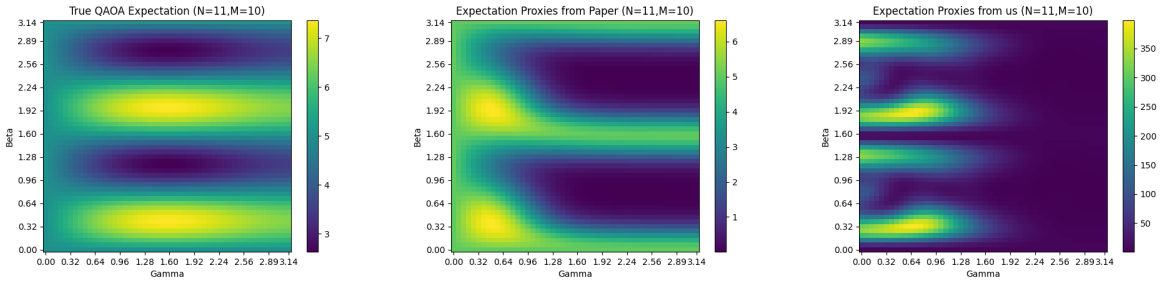

Figure 15: Expectation Map from Real QAOA, Paper Proxy and Our Proxy.

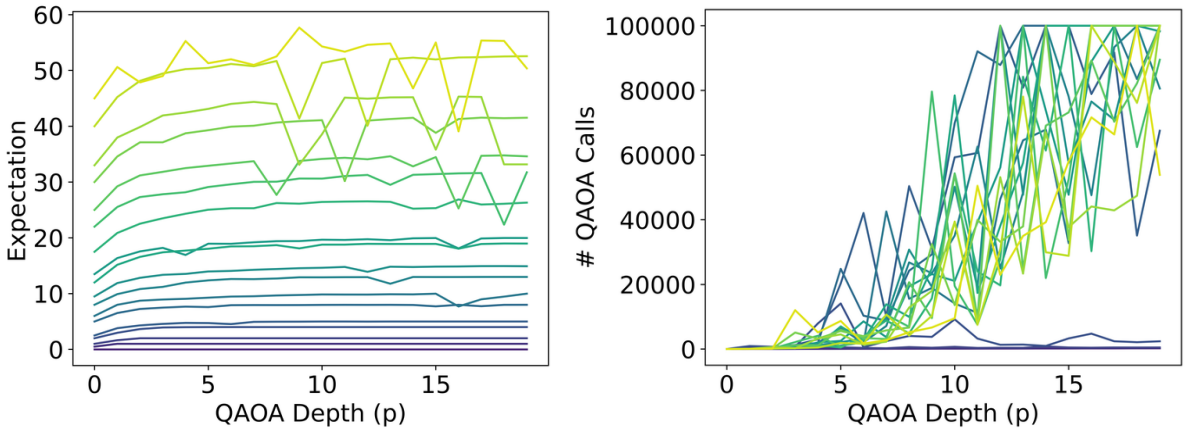

Figure 16: Comparison of circuit depth vs expectation after optimizing $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, as well as the number of QAOA calls needed to optimize $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, for single random instances of $\mathcal{G}(N, 1/2)$, $N = 1, \ldots, 18$. The lines vary from blue to yellow as N increases.

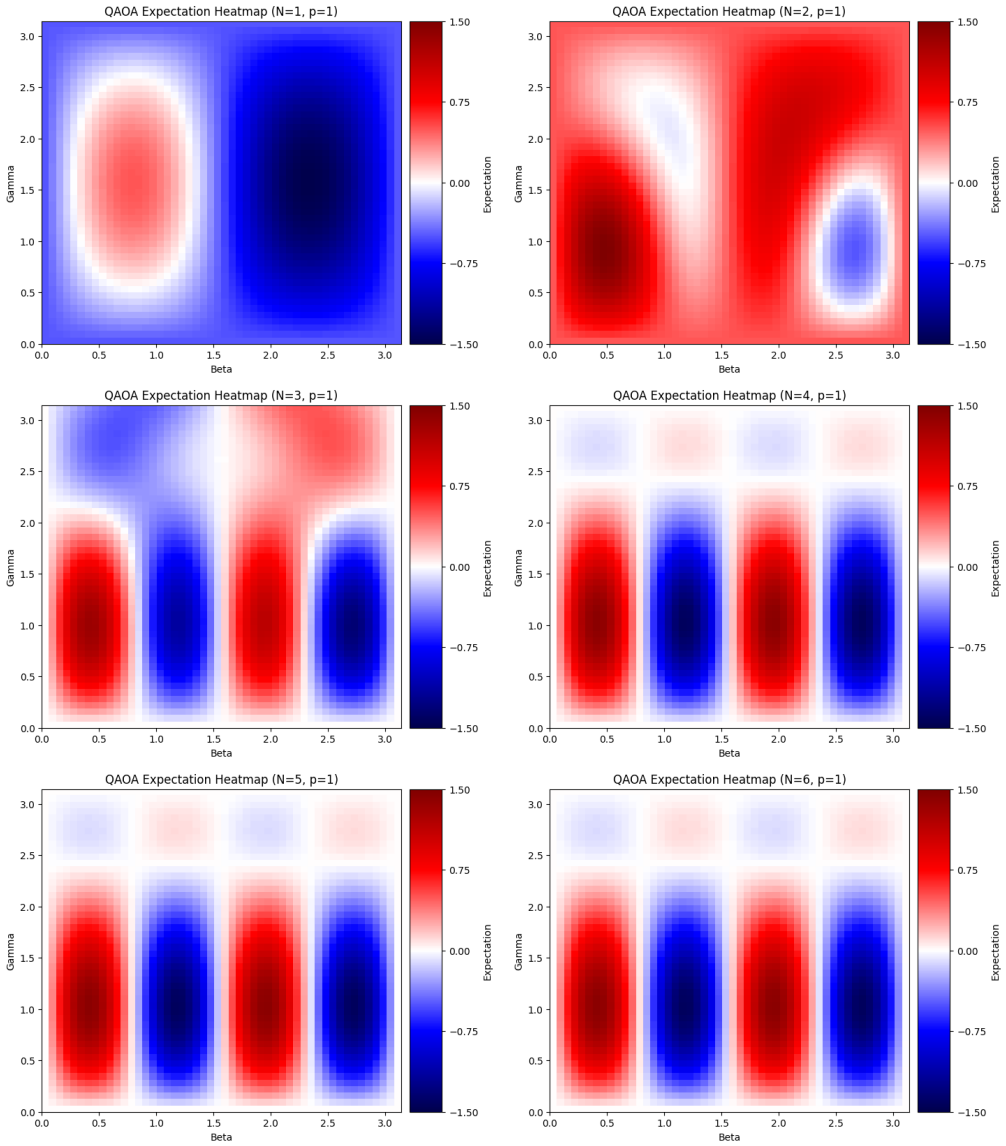

Figure 17: Heatmaps of the expectation landscape of QAOA for single instances of $\mathcal{G}(N, 1/2)$ for $N = 1, \ldots, 6$.

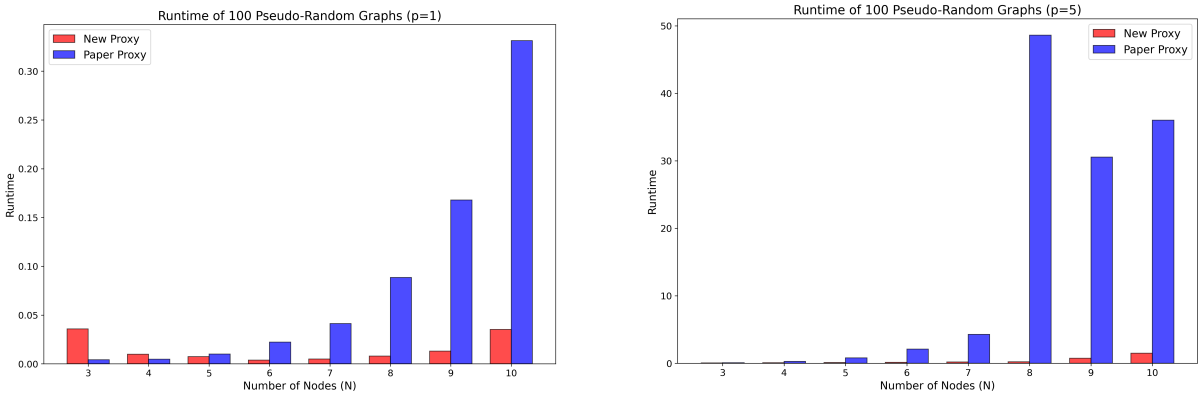

Figure 18: Runtimes for optimizing QAOA-proxy as problem size increases. Left: $p = 1$. Right: $p = 5$.

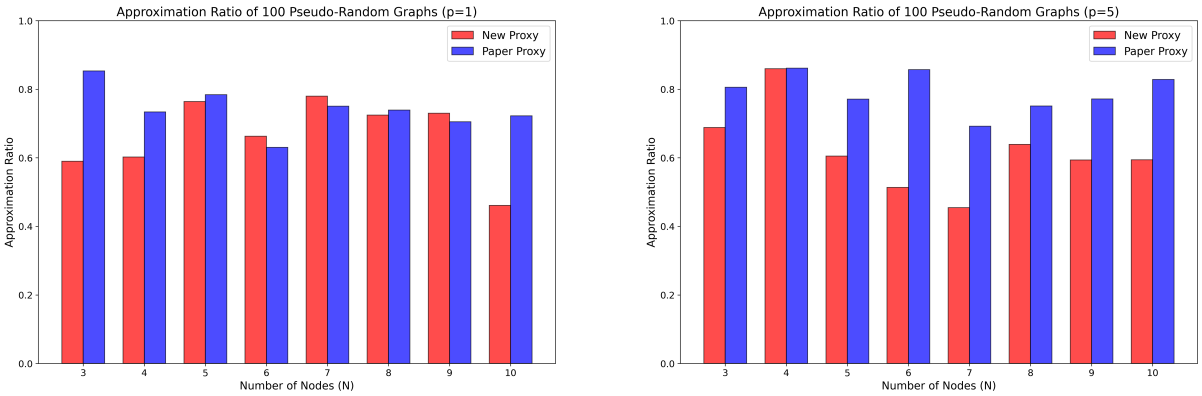

Figure 19: Approximation ratio vs Number of Nodes. Left: $p = 1$, $T = 100$. Right: $p = 5$, $T = 100$.

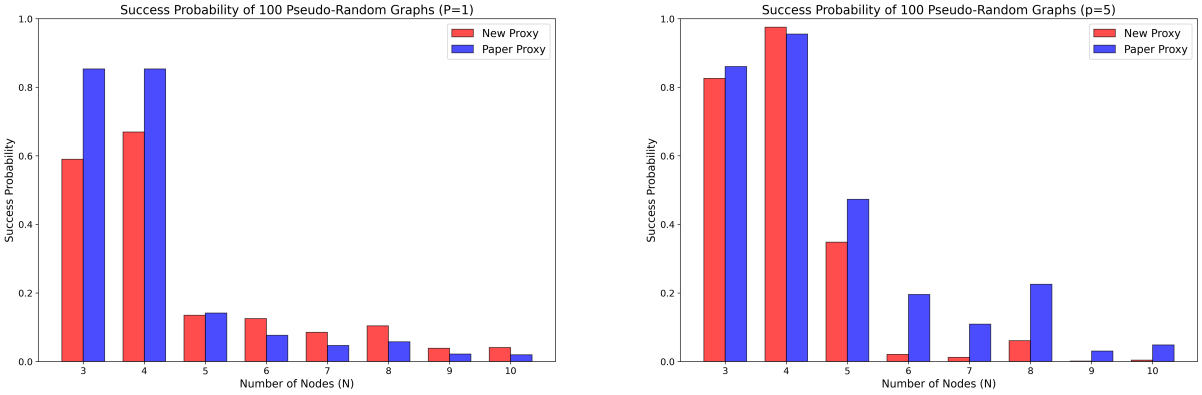

Figure 20: Success Probability vs Number of Nodes. Left: $p = 1$, $T = 100$. Right: $p = 5$, $T = 100$.

Appendix

Our code is available on GitHub at https://github.com/nkohen/QOKit, in the `grips` branch. The repository is forked from the original QOKit repository https://github.com/jpmorganchase/QOKit.

Our contributions are all located within the `grips/` directory, and an overview of the most important files is provided in `README.md`, as well as in this appendix.

Important / Source Files

- `QAOA_Simulator.py`: Contains utilities for simulating QAOA on arbitrary Ising model Hamiltonians.
- `plot_utils.py`: Contains utilities which delegate to matplotlib but which all have the same interface.
- `QAOA_paper_proxy.py`: Implements the original parameter-setting heuristic from the paper (Sud et al., 2024).
- `QAOA_proxy.py`: Implements our parameter-setting heuristic, which is much faster.
- `setup_juliacall.py`: Python script for installing the software necessary for the Julia backend for the QAOA proxies.
- `paper_proxy.jl`: Julia implementation of the original parameter-setting heuristic from the paper (Sud et al., 2024).
- `QAOA_proxy.jl`: Julia implementation of our parameter-setting heuristic.

Example Files

- `QAOA_plots.ipynb`: Contains example usage of `QAOA_Simulator`.
- `QAOA_proxy_plots.ipynb`: Contains example usage of `QAOA_paper_proxy` and `QAOA_proxy`.
- `real_distributions.ipynb`: Contains example code for sampling real MaxCut distribution data.

References

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*(5439), 509-512. Retrieved from https://www.science.org/doi/abs/10.1126/science.286.5439.509 doi: 10.1126/science.286.5439.509

Basso, J., Farhi, E., Marwaha, K., Villalonga, B., & Zhou, L. (2022). The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model. In F. Le Gall & T. Morimae (Eds.), *17th conference on the theory of quantum computation, communication and cryptography (tqc 2022)* (Vol. 232, pp. 7:1–7:21). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Retrieved from https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2022.7 doi: 10.4230/LIPIcs.TQC.2022.7

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, *59*(1), 65–98. Retrieved from https://doi.org/10.1137/141000671

Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., & Summer, A. (2024, June). A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, *1068*, 1–66. Retrieved from http://dx.doi.org/10.1016/j.physrep.2024.03.002 doi: 10.1016/j.physrep.2024.03.002

Cerezo, M., Sone, A., Volkoff, T., Cincio, L., & Coles, P. J. (2021, March). Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, *12*(1). Retrieved from http://dx.doi.org/10.1038/s41467-021-21728-w doi: 10.1038/s41467-021-21728-w

Farhi, E., Goldstone, J., & Gutmann, S. (2014). *A quantum approximate optimization algorithm.* Retrieved from https://arxiv.org/abs/1411.4028

Fernández-Pendás, M., Combarro, E. F., Vallecorsa, S., Ranilla, J., & Rúa, I. F. (2021). A study of the performance of classical minimizers in the quantum approximate optimization algorithm. *J. Comput. Appl. Math.*, *404*, 113388. Retrieved from https://api.semanticscholar.org/CorpusID:234149074

Hadfield, S., Hogg, T., & Rieffel, E. G. (2022, December). Analytical framework for quantum alternating operator ansätze. *Quantum Science and Technology*, *8*(1), 015017. Retrieved from http://dx.doi.org/10.1088/2058-9565/aca3ce doi: 10.1088/2058-9565/aca3ce

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from https://doi.org/10.1038/s41586-020-2649-2 doi: 10.1038/s41586-020-2649-2

Huang, Z., Li, Q., Zhao, J., & Song, M. (2022, 11). Variational quantum algorithm applied to collision avoidance of unmanned aerial vehicles. *Entropy*, *24*, 1685. doi: 10.3390/e24111685

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: a llvm-based python jit compiler. In *Proceedings of the second workshop on the llvm compiler infrastructure in hpc.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2833157.2833162 doi: 10.1145/2833157.2833162

Lin, D., White, J. M., Byrne, S., Bates, D., Noack, A., Pearson, J., ... other contributors (2019, July). *JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions.* Retrieved from https://doi.org/10.5281/zenodo.2647458 doi: 10.5281/zenodo.2647458

Lykov, D., Shaydulin, R., Sun, Y., Alexeev, Y., & Pistoia, M. (2023, November). Fast simulation of high-depth qaoa circuits. In *Proceedings of the sc '23 workshops of the international conference on high performance computing, network, storage, and analysis.* ACM. Retrieved from http://dx.doi.org/10.1145/3624062.3624216 doi: 10.1145/3624062.3624216

Rančić, M. J. (2023, Feb). Noisy intermediate-scale quantum computing algorithm for solving an n-vertex maxcut problem with $\log(n)$ qubits. *Phys. Rev. Res.*, *5*, L012021. Retrieved from https://link.aps.org/doi/10.1103/PhysRevResearch.5.L012021 doi: 10.1103/PhysRevResearch.5.L012021

Rowley, C. (2022). *Pythoncall.jl: Python and julia in harmony.* Retrieved from https://github.com/JuliaPy/PythonCall.jl

Stechly, M. (2024). *Introduction to variational quantum algorithms.* Retrieved from https://arxiv.org/abs/2402.15879

Streif, M., & Leib, M. (2019). *Training the quantum approximate optimization algorithm without access to a quantum processing unit.* Retrieved from https://arxiv.org/abs/1908.08862

Sud, J., Hadfield, S., Rieffel, E., Tubman, N., & Hogg, T. (2024, May). Parameter-setting heuristic for the quantum alternating operator ansatz. *Phys. Rev. Res.*, *6*, 023171. Retrieved from https://link.aps.org/doi/10.1103/PhysRevResearch.6.023171 doi: 10.1103/PhysRevResearch.6.023171

Sureshbabu, S. H., Herman, D., Shaydulin, R., Basso, J., Chakrabarti, S., Sun, Y., & Pistoia, M. (2023). Parameter setting in quantum approximate optimization of weighted problems. *Quantum*, *8*, 1231. Retrieved from https://api.semanticscholar.org/CorpusID:258865516

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Watts, S., D.and Strogatz. (1998, June). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440-442. Retrieved from https://doi.org/10.1038/30918 doi: 10.1038/30918

Wurtz, J., & Love, P. (2021, Apr). Maxcut quantum approximate optimization algorithm performance guarantees for p ¿ 1. *Phys. Rev. A*, *103*, 042612. Retrieved from https://link.aps.org/doi/10.1103/PhysRevA.103.042612 doi: 10.1103/PhysRevA.103.042612