QAOA ON THE TRAVELLING SALESMAN PROBLEM

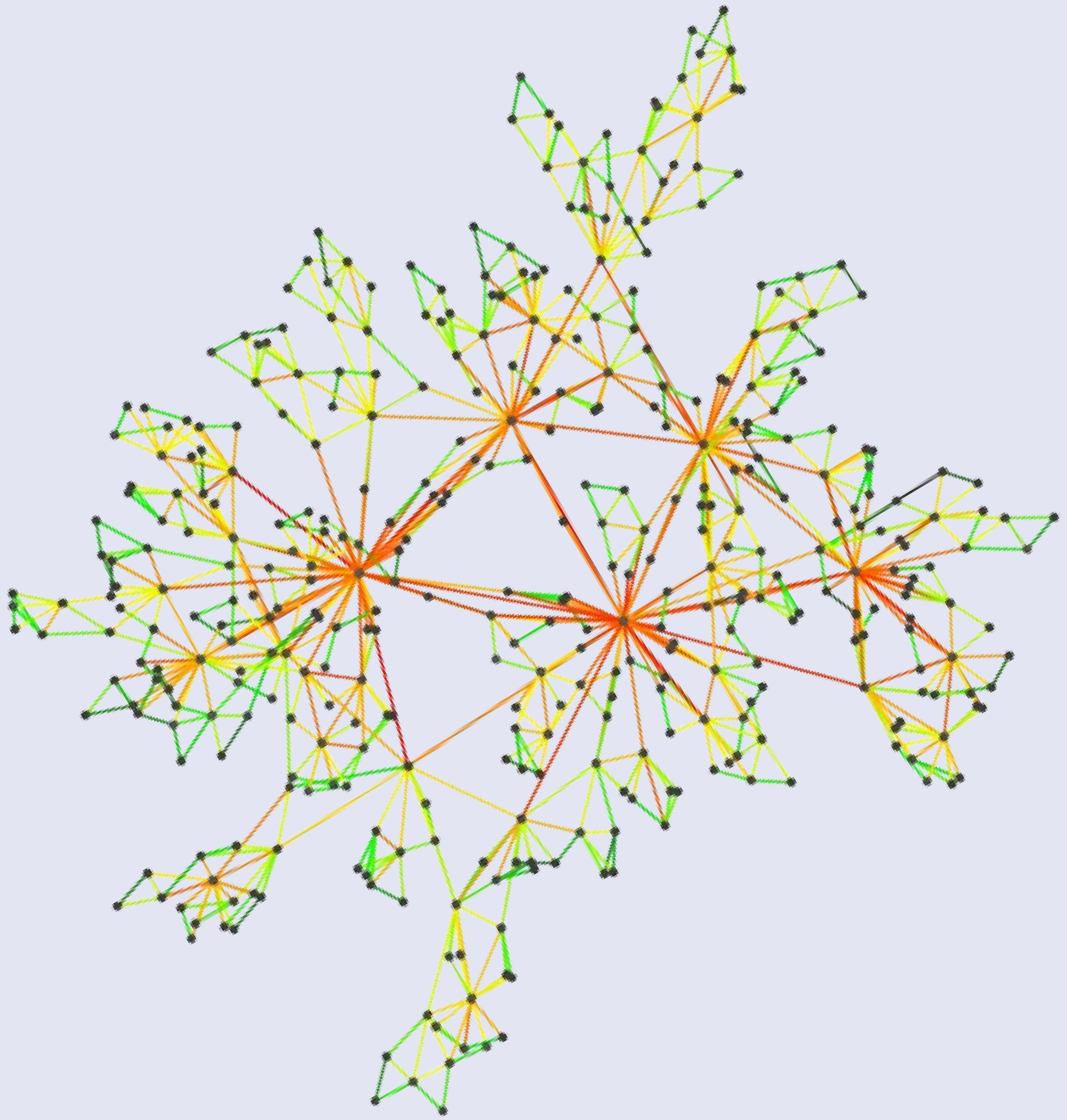
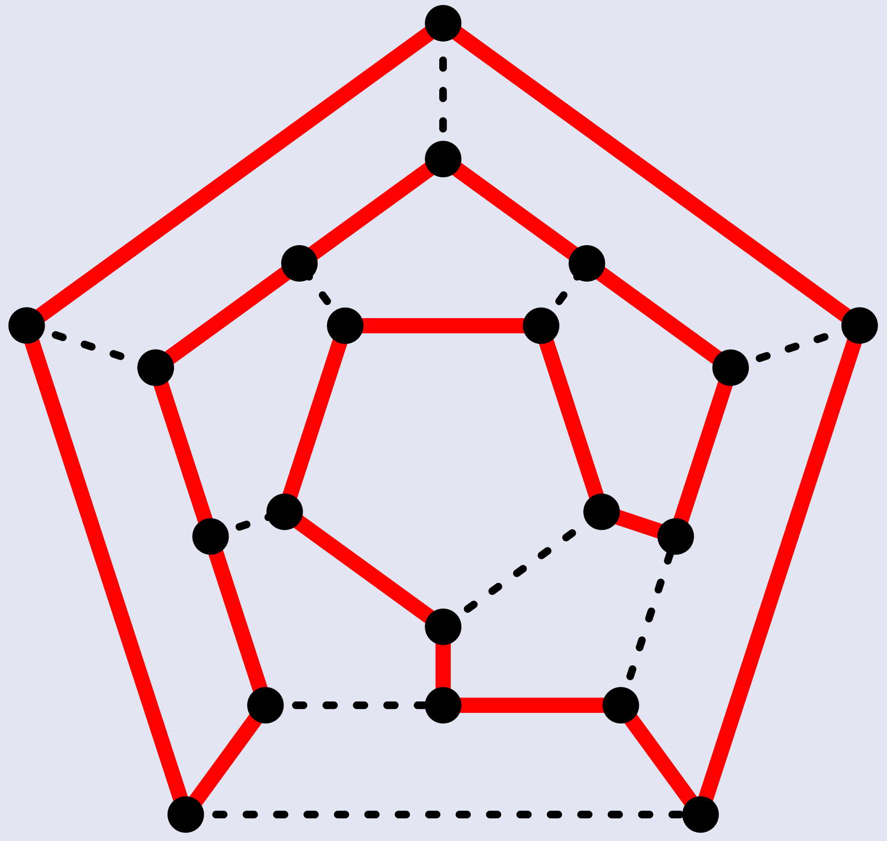


TABLE OF CONTENTS:



WHAT IS THE TRAVELLING SALESMAN PROBLEM?

Given a complete weighted graph (where the vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road), find a Hamiltonian cycle with the least weight.

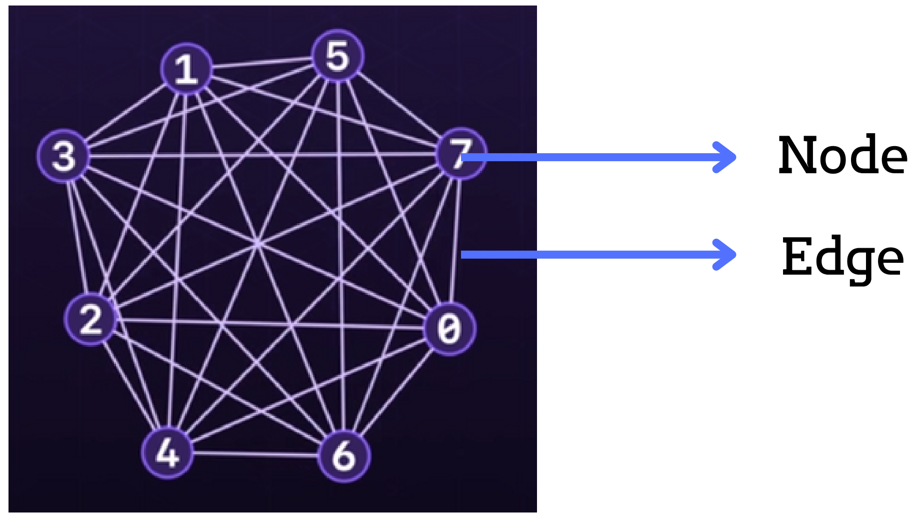
Minus the jargon, "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"

It is an NP-hard problem in combinatorial optimization and can take up exponential time to solve because there is no known algorithm that can solve it in polynomial time.

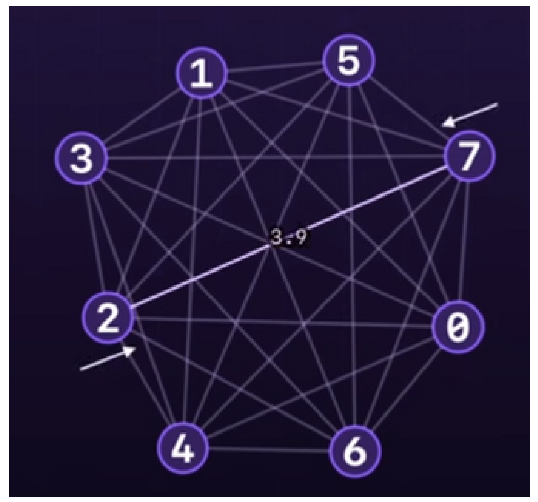
The TSP and its variants have many applications in fields ranging from microchip manufacturing to DNA sequencing.

PROBLEM DEFINITION

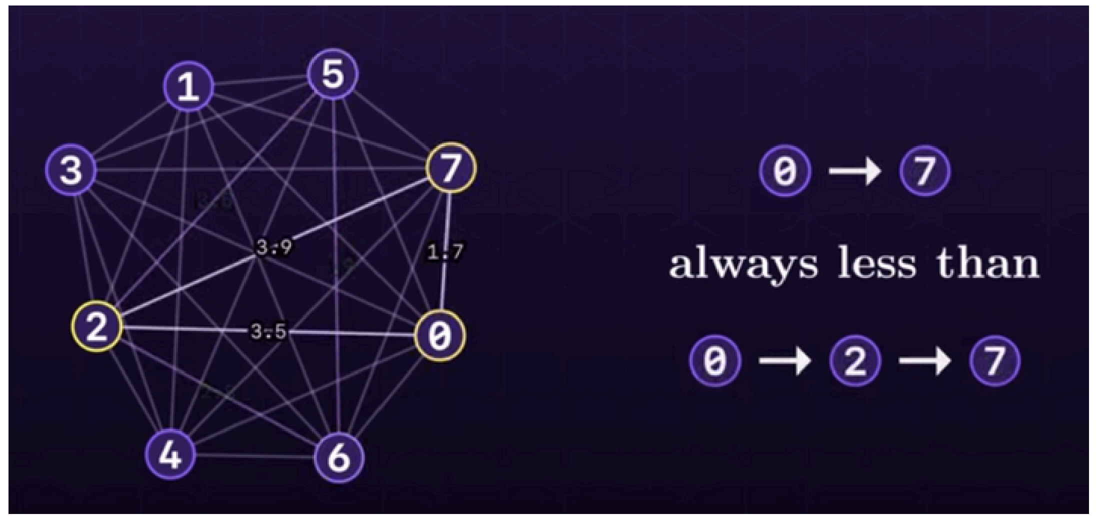
- The Graph is assumed to be complete – There exists a direct edge between each pair of nodes/vertices



- Symmetric TSP – The distance (cost) from city A to city B is the same for city B to city A

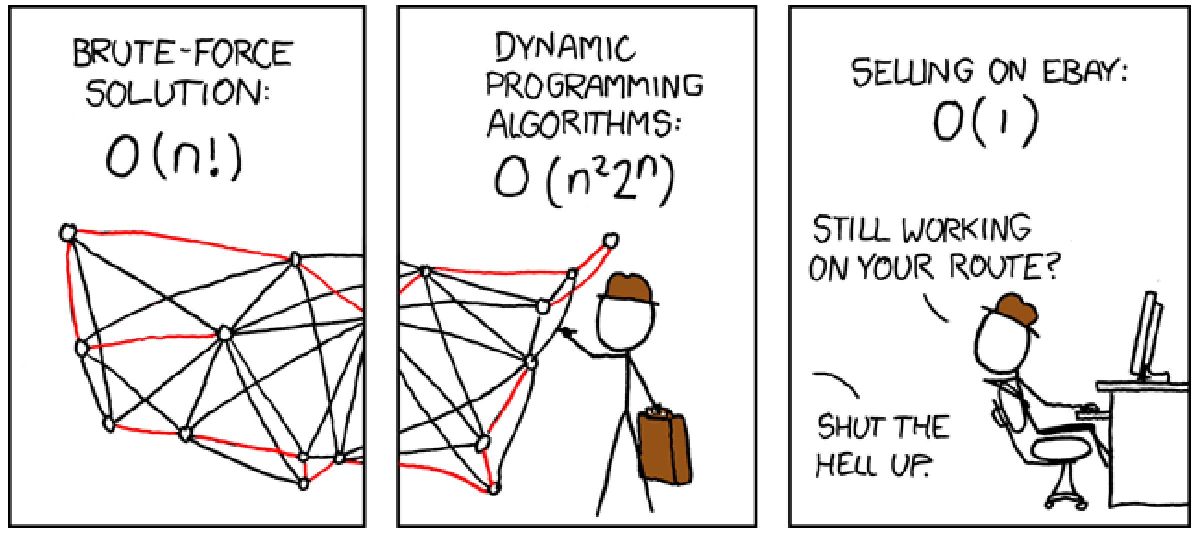


- Triangle Inequality – The direct path between 2 cities is always the shortest path between them



CLASSICAL APPROACHES AND WHY THEY DON'T WORK

An example of an exact algorithm would be to try out all permutations (brute force) and search for the cheapest. The running time for this approach lies within a polynomial factor of $O(n!)$, the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. Since the total number of possible Hamiltonian cycles is n! for a complete graph.
One of the earliest applications of dynamic programming i.e.,the Held–Karp algorithm, solves the problem in time $O(n^2 2^n)$.

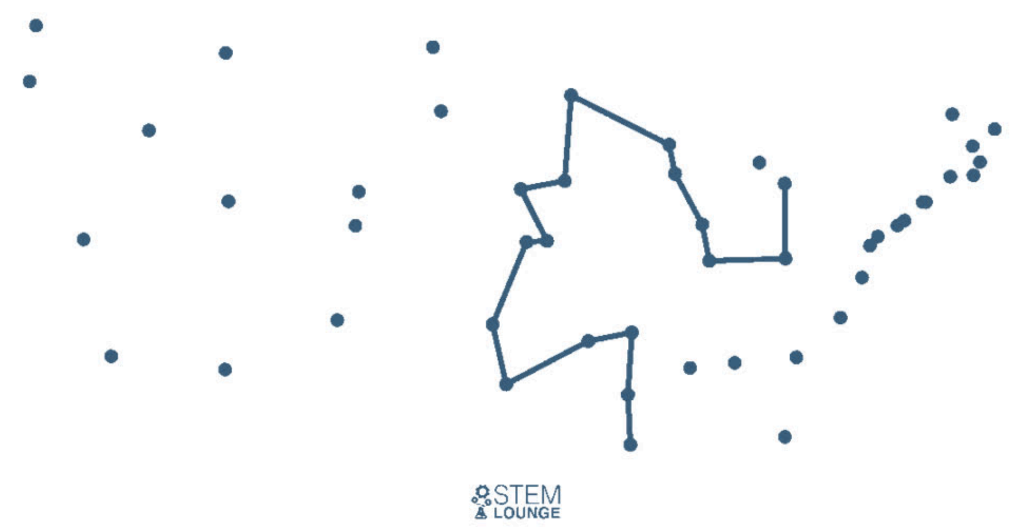


The nearest neighbour (NN) algorithm (a greedy algorithm) lets the salesman choose the nearest unvisited city as his next move. This algorithm quickly yields an effectively short route. For N cities randomly distributed on a plane, the algorithm on average yields a path 25% longer than the 1-tree Lower Bound (which is derived from Minimum Spanning Trees).

IMPLEMENTATION OF THE CLASSICAL GREEDY NEAREST NEIGHBOURS ALGORITHM

Greedy algorithms are a class of algorithms that make <u>locally optimal choices at each step</u> with the hope of finding a global optimum solution. Decisions are made based on the information available at the current moment without considering the consequences of these decisions in the future. The key idea is to select the best possible choice at each step, leading to a solution that may not always be the most optimal but is often good enough for many problems.

The Nearest Neighbour Algorithm is a simple and intuitive approximation for the TSP. It starts at an arbitrary city and repeatedly **selects the nearest unvisited city** until all cities have been visited.

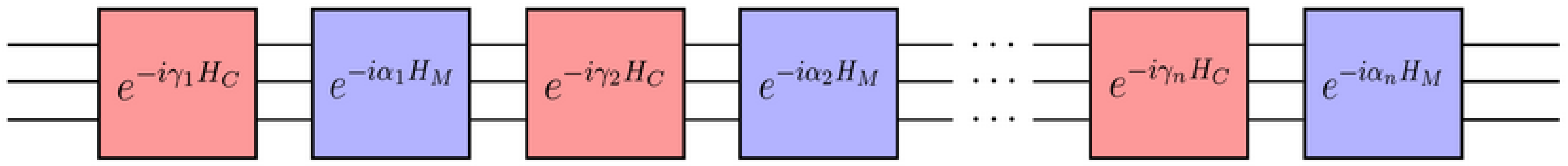


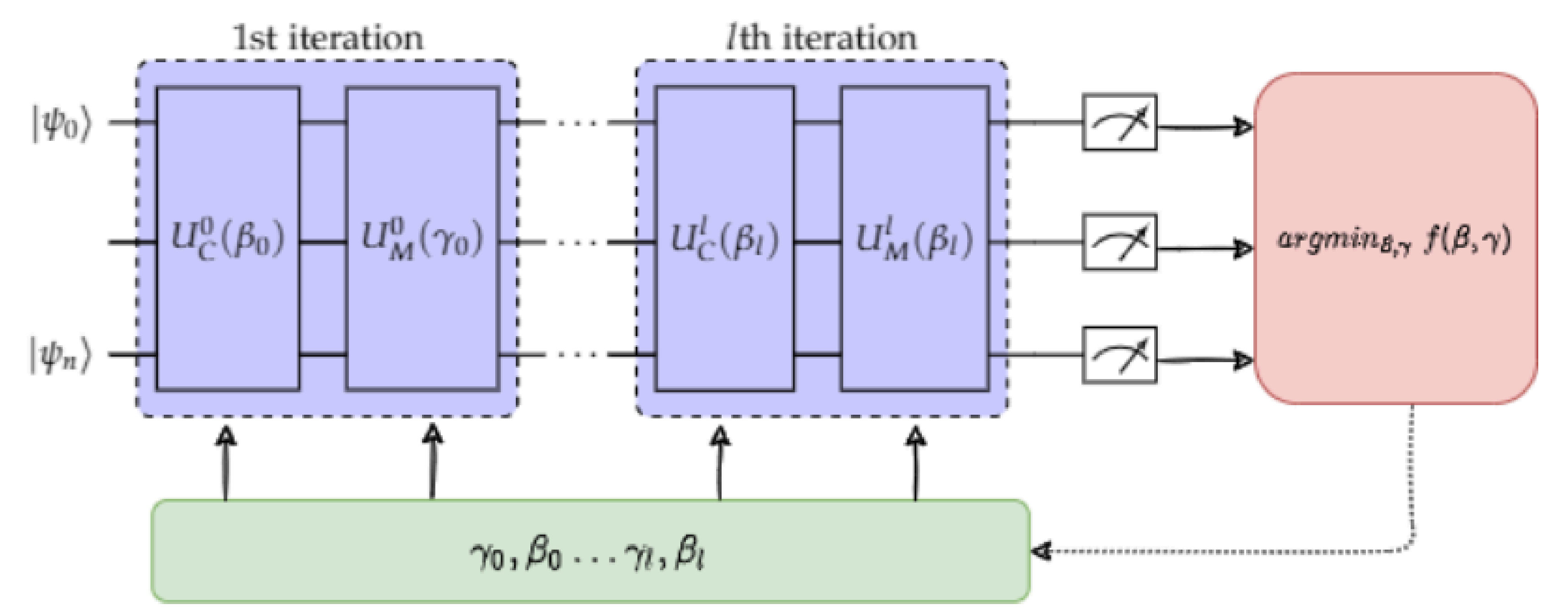
The time complexity of the nearest neighbor algorithm is O(n^2).
While the Nearest Neighbour Algorithm is not guaranteed to provide the optimal solution to the TSP, it often produces good results, especially for small or medium-sized instances of the problem.

BRIEF OVERVIEW OF QAOA

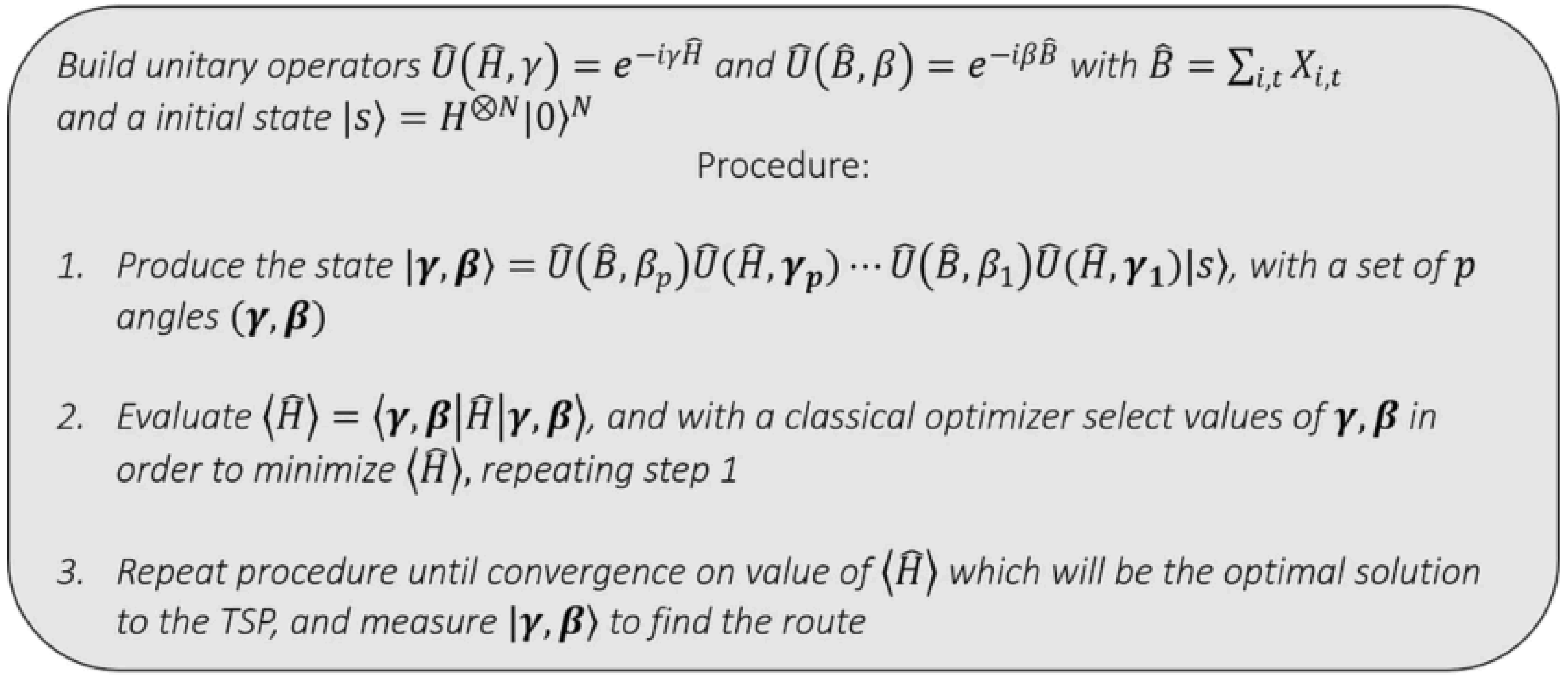
QAOA is a hybrid quantum-classical method designed for tackling combinatorial optimization problems and has shown promising results in addressing NP-hard problems such as TSP. The algorithm operates by encoding the given problem into a quantum state using unitary transformations.

The fundamental components of QAOA consist of a parameterized quantum circuit and classical optimization techniques. The parameterized quantum circuit serves as a function representing the problem and generates an output value based on specific parameters. The aim is to minimize this value, which corresponds to finding the optimal or near-optimal solution to the problem. On the other hand, classical optimization techniques are used to iteratively adjust these parameters to approach the minimum value. This iterative process of optimization leverages the capabilities of both quantum and classical computing, hence categorizing QAOA as a variational quantum algorithm.





QAOA Ansatz with *l* iterations, equipped with a classical optimization



Build unitary operators $\hat{U}(\hat{H},\gamma) = e^{-i\gamma\hat{H}}$ *and* $\hat{U}(\hat{B},\beta) = e^{-i\beta\hat{B}}$ *with* $\hat{B} = \sum_{i,t} X_{i,t}$ *and a initial state* $|s\rangle = H^{\otimes N}|0\rangle^N$

Procedure:

1. *Produce the state* $|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle = \hat{U}(\hat{B},\beta_p)\hat{U}(\hat{H},\boldsymbol{\gamma}_p)\cdots\hat{U}(\hat{B},\beta_1)\hat{U}(\hat{H},\boldsymbol{\gamma}_1)|s\rangle$, *with a set of* p *angles* $(\boldsymbol{\gamma},\boldsymbol{\beta})$
2. *Evaluate* $\langle\hat{H}\rangle = \langle\boldsymbol{\gamma},\boldsymbol{\beta}|\hat{H}|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$, *and with a classical optimizer select values of* $\boldsymbol{\gamma},\boldsymbol{\beta}$ *in order to minimize* $\langle\hat{H}\rangle$, *repeating step 1*
3. *Repeat procedure until convergence on value of* $\langle\hat{H}\rangle$ *which will be the optimal solution to the TSP, and measure* $|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$ *to find the route*

QAOA Pipeline

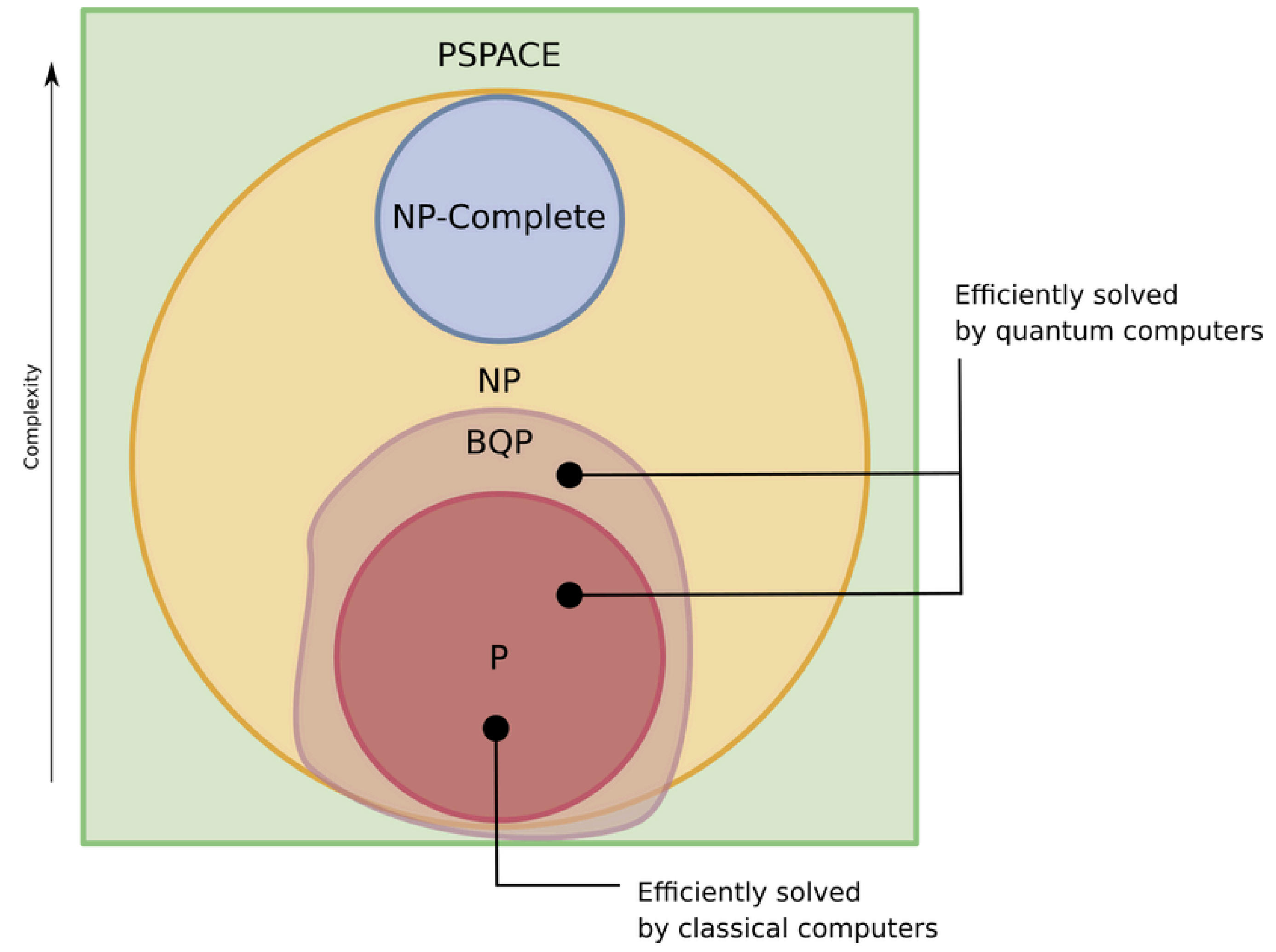
QAOA consists of the following steps:

1. Define a cost Hamiltonian HC such that its ground state encodes the solution to the optimization problem.
2. Define a mixer Hamiltonian HM
3. Construct the circuits e−iγHC and e−iαHM. These are called the cost and mixer layers, respectively.
4. Choose a parameter n≥1 and build the circuit U(γ, α) = e^(−iαnHM) * e^(−iγnHC) ... e^(−iα1HM) e^(−iγ1HC), consisting of repeated application of the cost and mixer layers.
5. Prepare an initial state, apply U(γ,α) and use classical techniques to optimize the parameters.
6. After the circuit has been optimized, measurements of the output state reveal approximate solutions to the optimization problem.

In summary, the starting point of QAOA is the specification of cost and mixer Hamiltonians. We then use time evolution and layering to create a variational circuit and optimize its parameters. The algorithm concludes by sampling from the circuit to get an approximate solution to the optimization problem.

WHY IS QAOA A BETTER ALTERNATIVE?

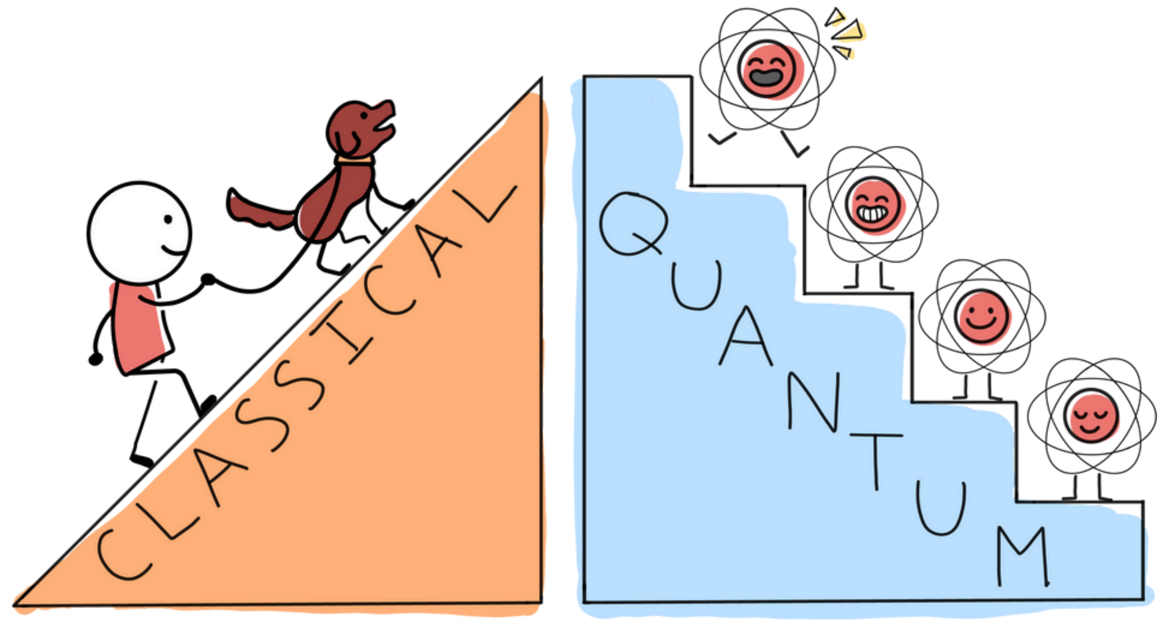
- Superposition: Quantum computers can explore multiple solutions simultaneously due to the principle of superposition. A quantum state can represent a combination of many possible solutions, allowing the algorithm to evaluate many paths at once.
- Entanglement: More efficient exploration and exploitation of the solution space compared to classical algorithms.
- Amplitude Amplification: Techniques such as Grover's search algorithm can amplify the probability of the correct solution being measured, potentially providing a quadratic speedup over classical search algorithms.
- Flexibility: QAOA can be adapted to a wide range of optimization problems by appropriately defining the cost and mixer Hamiltonians. This flexibility allows for tailored approaches that can be more effective than generic classical algorithms.


PSPACE
NP-Complete
NP
BQP
P
Complexity
Efficiently solved by quantum computers
Efficiently solved by classical computers

- Exponential Solution Space: For NP-hard problems like TSP, the solution space grows exponentially with the number of cities. Quantum algorithms can explore this space more efficiently due to their inherent parallelism and entanglement properties.
- Hard Problems: QAOA is particularly suited for hard combinatorial optimization problems where classical algorithms struggle to find good solutions within reasonable timeframes. As quantum hardware improves, the potential advantage of QAOA over classical algorithms is expected to become more pronounced.

WHY IS QAOA A BETTER ALTERNATIVE?

- **Mixer Hamiltonian**: The mixer Hamiltonian in QAOA helps explore the solution space by creating superpositions of different states. This is analogous to the exploration phase in classical optimization algorithms.
- **Cost Hamiltonian**: The cost Hamiltonian helps exploit the best solutions by gradually increasing the probability of the optimal state. This balances the exploration and exploitation phases more effectively than many classical algorithms.



- **Quantum-Classical Hybrid**: QAOA combines quantum circuits with classical optimization routines. The quantum part generates potential solutions, while the classical part optimizes the parameters of the quantum circuit. This hybrid approach leverages the strengths of both quantum and classical computation, potentially leading to better performance than purely classical algorithms.
- **NISQ Suitability**: QAOA is designed to work on Noisy Intermediate-Scale Quantum (NISQ) devices, which are the current generation of quantum computers. Its variational nature helps it mitigate some of the noise and errors inherent in these devices.

APPLYING QAOA TO TSP

$$x_{i,t} = \begin{cases} 1, & \text{if solution is at vertex } i \text{ at time } t \\ 0, & \text{otherwise} \end{cases}$$

Equation 1: Boolean mapping to define path

$$\begin{array}{c|c} & \xrightarrow{\quad i \quad} \\ \hline \downarrow t & \boldsymbol{x} = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \longrightarrow \end{array}$$

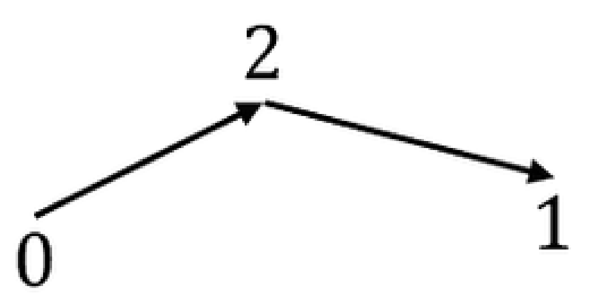
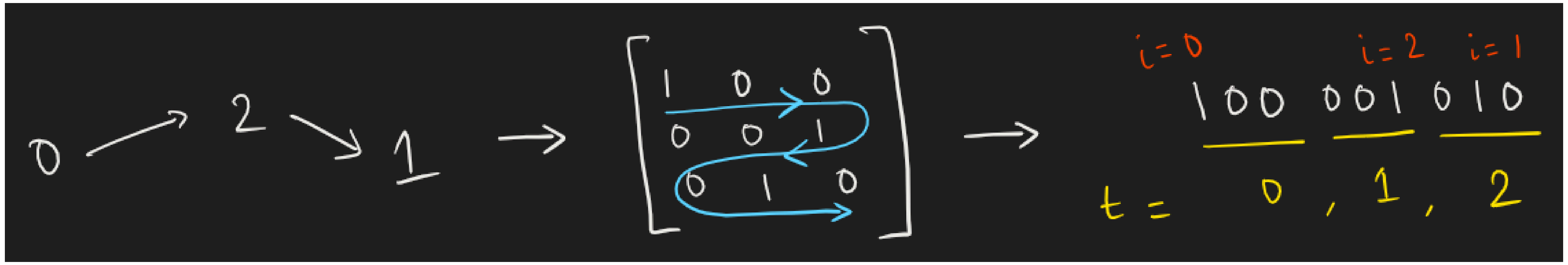


Figure 2: Matrix **x** defining a path

Let $x_{\alpha,j}$ be a set of n^2 binary variables where α denotes a city and j denotes a time step in some path. Thus $x_{2,0}=1$ means the path starts in city 2 in the zeroth time step. Likewise $x_{4,n-1}=1$ means the path end in city 4 on the last time step. There are n time steps as we will visit each city exactly once. We form a bitstring that represents a path as follows:

$$x_{0,0}x_{1,0},\ldots,x_{n1,0}x_{0,1},\ldots,x_{n-1,n-1}$$

HOW TO READ A BITSTRING ⟶



APPLYING QAOA TO TSP

TWO CONSTRAINTS

$$\sum_i x_{i,p} = 1 \ \forall p \longrightarrow$$

You can only visit one city at a particular instant (Can't be in two cities at simultaneously)

$$\sum_p x_{i,p} = 1 \ \forall i. \longrightarrow$$

Visit a city only once throughout the cycle (for all times)

$$D(\boldsymbol{x}) = \sum_{i,j} w_{i,j} \sum_t x_{i,t} x_{j,t+1}$$

Equation 2: Total distance Traveled

$$C(\boldsymbol{x}) = D(\boldsymbol{x}) + A\left(\sum_i (1 - \sum_t x_{i,t})^2 + \sum_t (1 - \sum_i x_{i,t})^2\right)$$

Equation 3: Cost function with constraints

C(x) is the sum of the original distance function and a penalty function. The Cost function depends on a parameter A, that we must select in order to ensure that the constraints are obeyed. Choose A to be much larger than any of the edge weights in the graph.
If the two mentioned constraints are obeyed, C(x) reduces to D(x)

APPLYING QAOA TO TSP

$$x_{i,t} \rightarrow \hat{x}_{i,t} = \frac{1 - \hat{Z}_{i,t}}{2} \quad \text{with } \hat{Z}_{i,t} = \boldsymbol{I} \otimes \boldsymbol{I} \otimes \cdots \otimes \boldsymbol{\sigma}^{z}_{i,t} \otimes \cdots \otimes \boldsymbol{I}$$

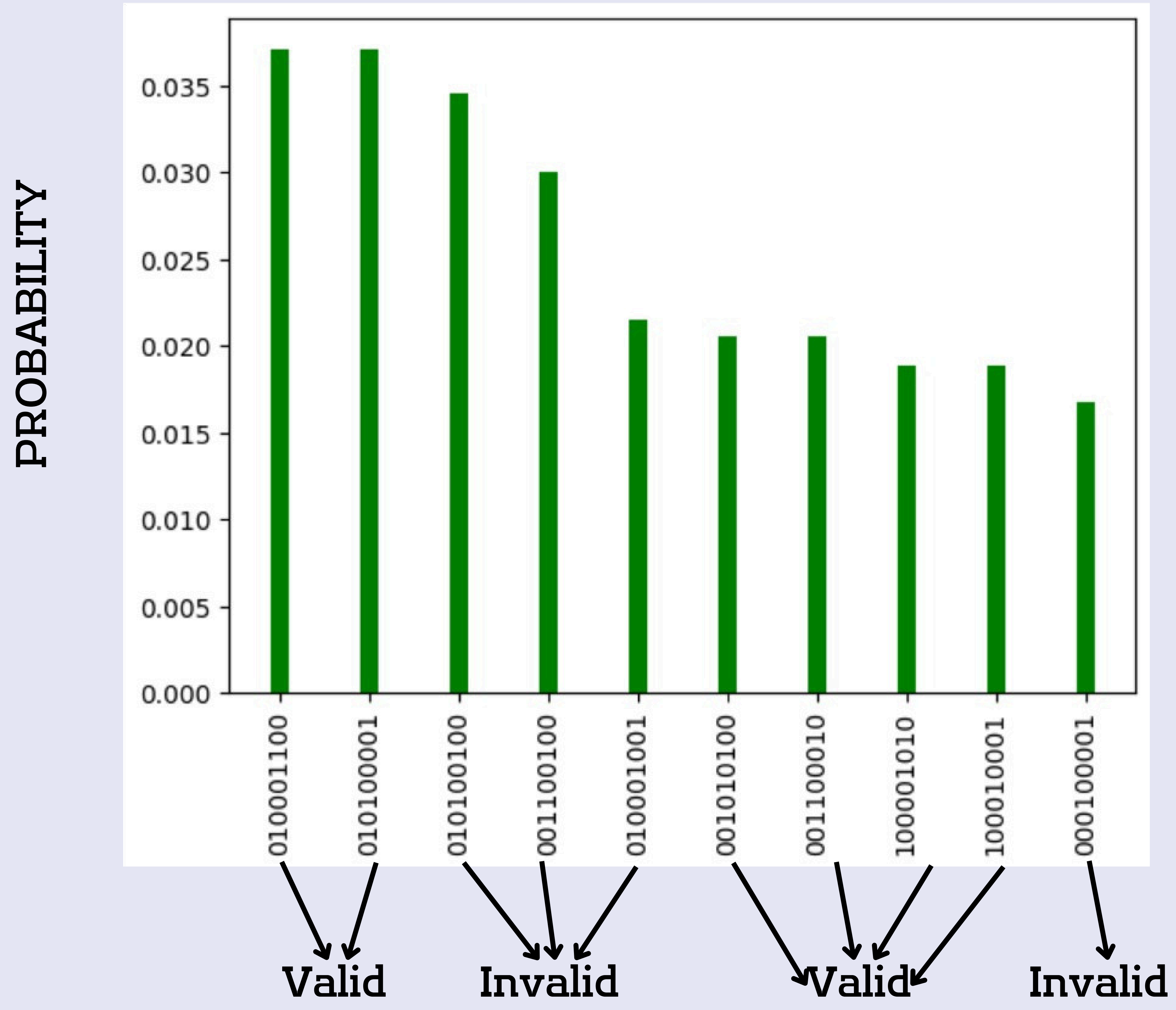
Equation 4: Mapping problem to a quantum computer

Now, we can map the problem of minimizing C(x), to a quantum analogue we promote the boolean variable x(i,t) to a quantum operator. With this notation then if qubit i,t is in state |1⟩ then the solution is at vertex i at time t. Then, using this operator we can transform the cost function C(x) into a Hamiltonian for the system of n^2 qubits.

Note that the ground state of this Hamiltonian is the optimal solution to the TSP problem!

After generating the Cost Hamiltonian, we also built a Driver Hamiltonian (for the Mixer Layer). This Driver Hamiltonian is parameterized in addition to the Cost Hamiltonian, and is included to ensure that the space over the states is changing, allowing for the superposition to move and ultimately settle into a ground state. This Driver Hamiltonian is defined as $I(q_0) - \sum_{i=0}^{n^2} X(q_i)$

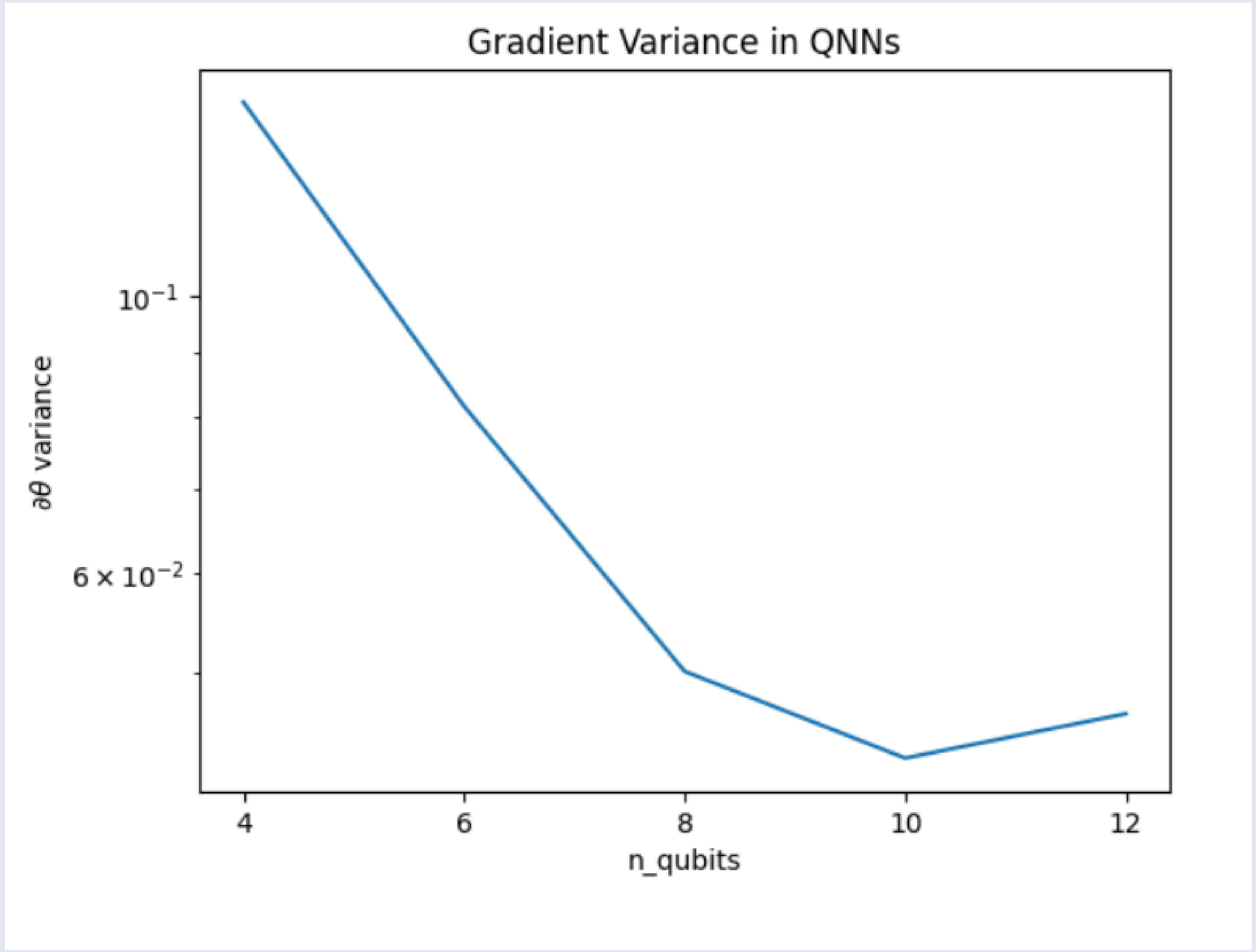
IMPLEMENTATION OF QAOA ON TSP



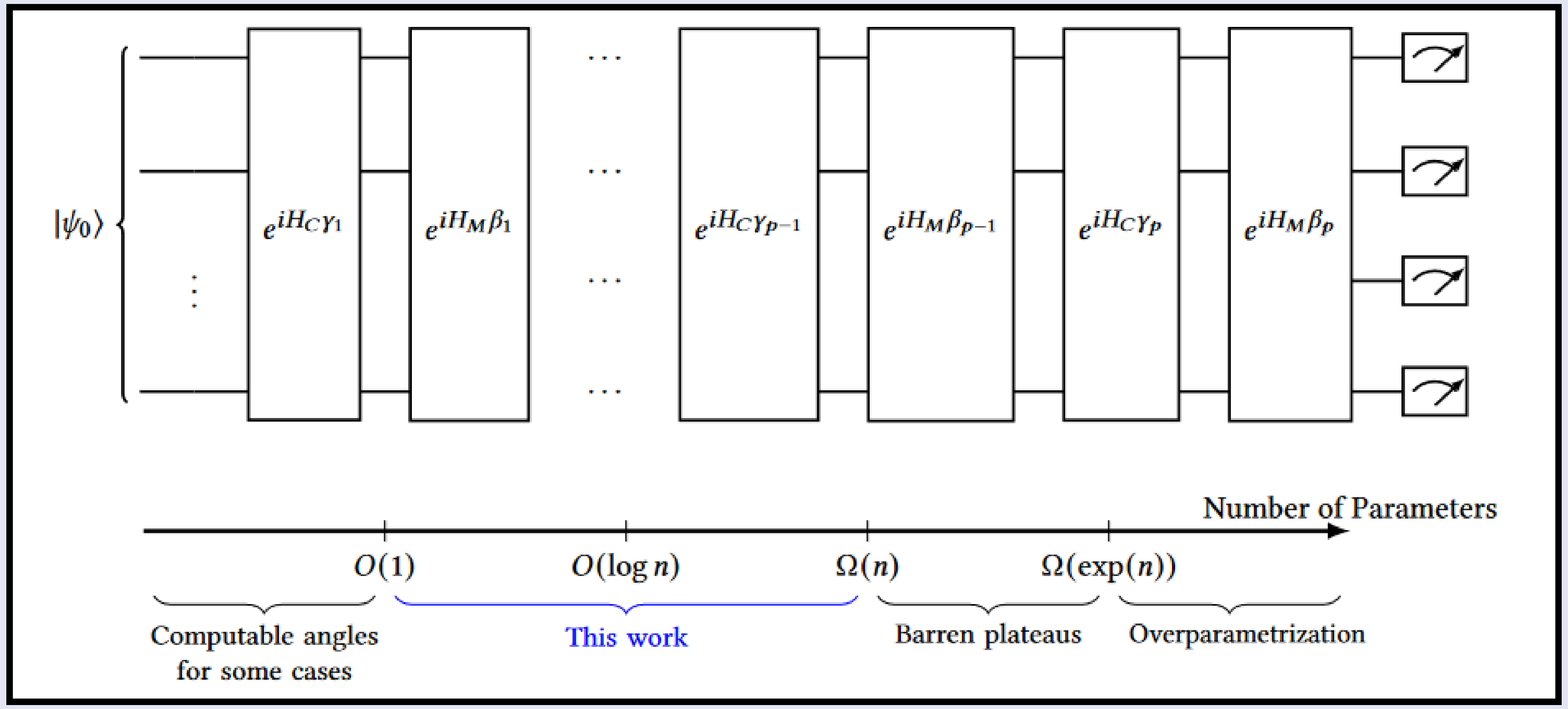
There are 2^(n^2) possible bitstrings. Here n=3, so 512 possible solutions exist.
Out of these, only 6 satisfy the constraints.
In the graph, we can see that 6 of the top 10 solutions (highest probability) are valid, indicating that QAOA has run successfully.
The presence of invalid solutions is due to us being unable to find the optimal weights to impose on the penalty function that would disregard them. There is also the possibility of noise in the circuit, which may have caused some error in our results.
We are currently in the NISQ (Noisy Intermediate-Scale Quantum) Devices era therefore it is expected that quantum operations may not be entirely perfect.

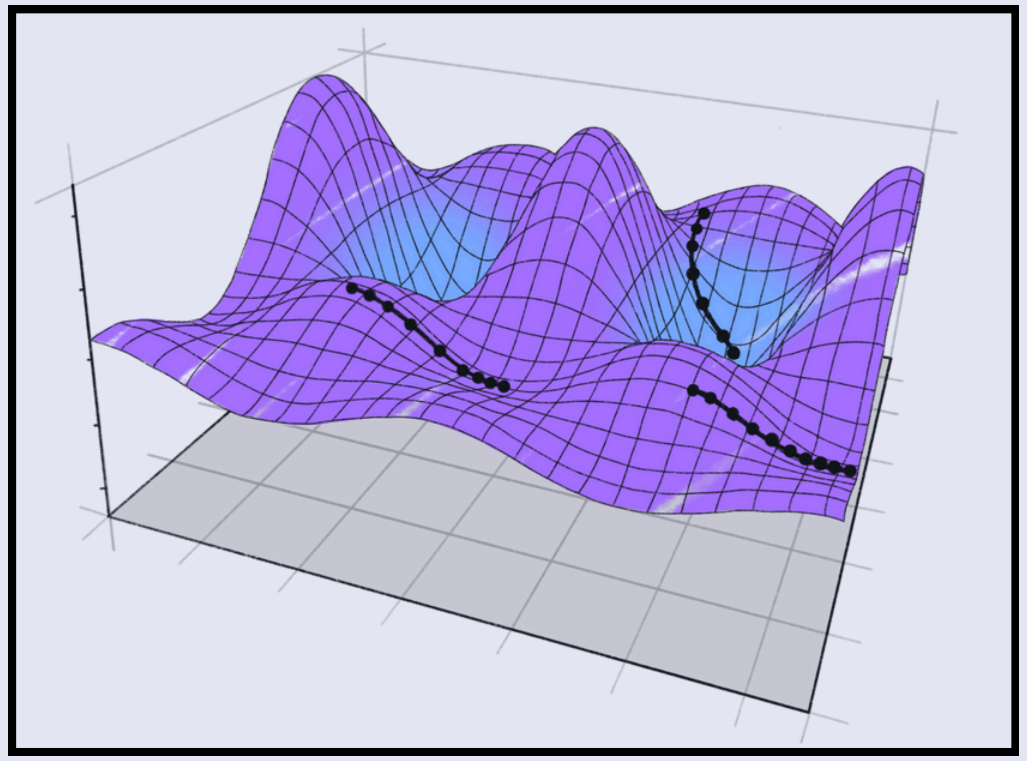
BARREN PLATEAUS

Barren plateaus are large regions of the cost function's parameter space where the variance of the gradients of parametrized quantum circuits become exponentially small with respect to the number of qubits; or, put another way, the cost function landscape becomes flat. This means that a variational circuit initialized in one of these areas will be untrainable using any gradient-based algorithm, making optimization difficult and potentially impossible.



BARREN PLATEAUS





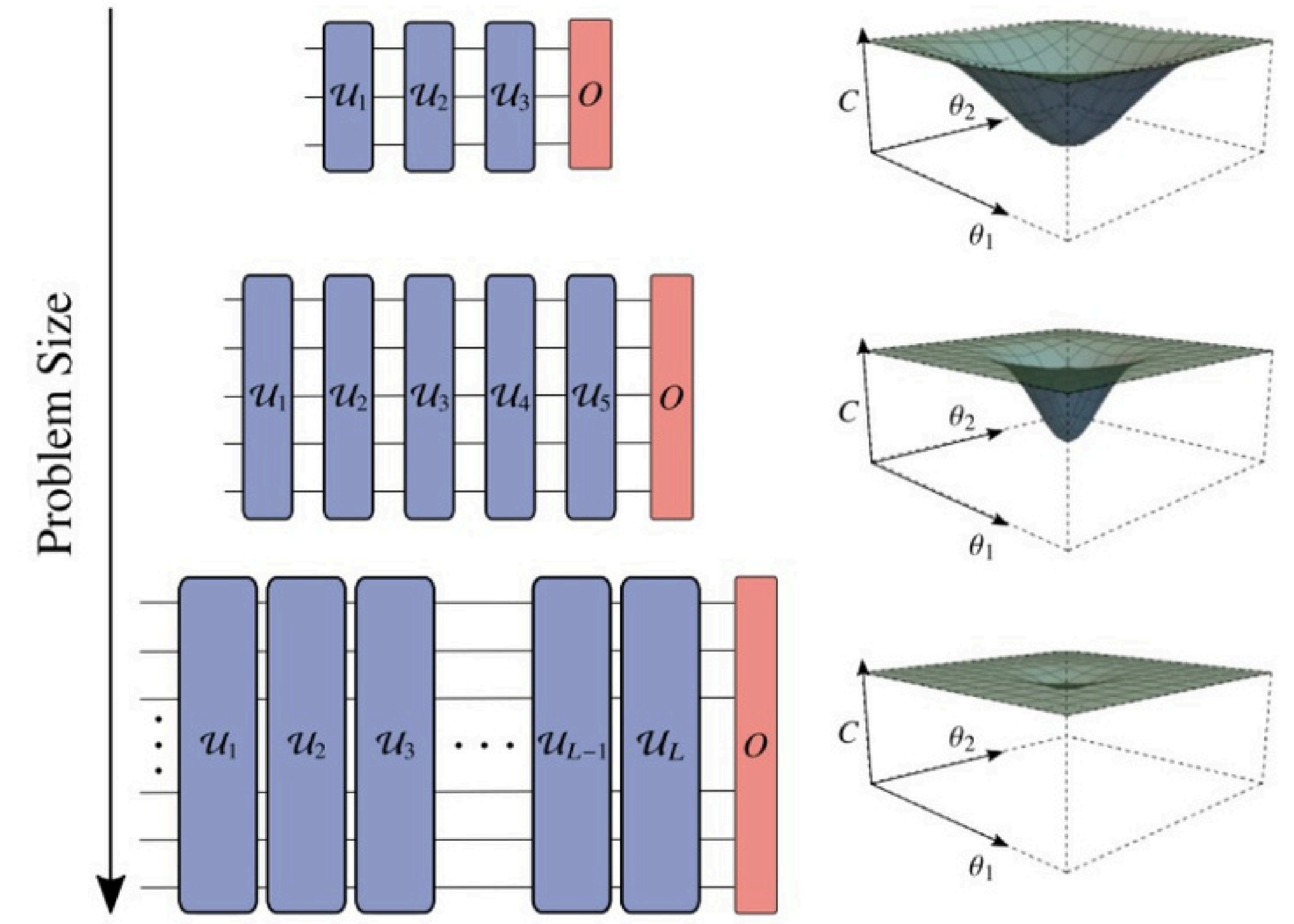


Fig. 1 Schematic diagram of the Noise-Induced Barren Plateau (NIBP) phenomenon. For various applications such as chemistry and optimization,

As the number of layers increases, the quantum circuit becomes more complex. This complexity can lead to interference effects that cause the gradients to average out to very small values over the parameter space. The optimization landscape becomes higher-dimensional and often flatter, making it harder to find gradients that lead to significant improvements.

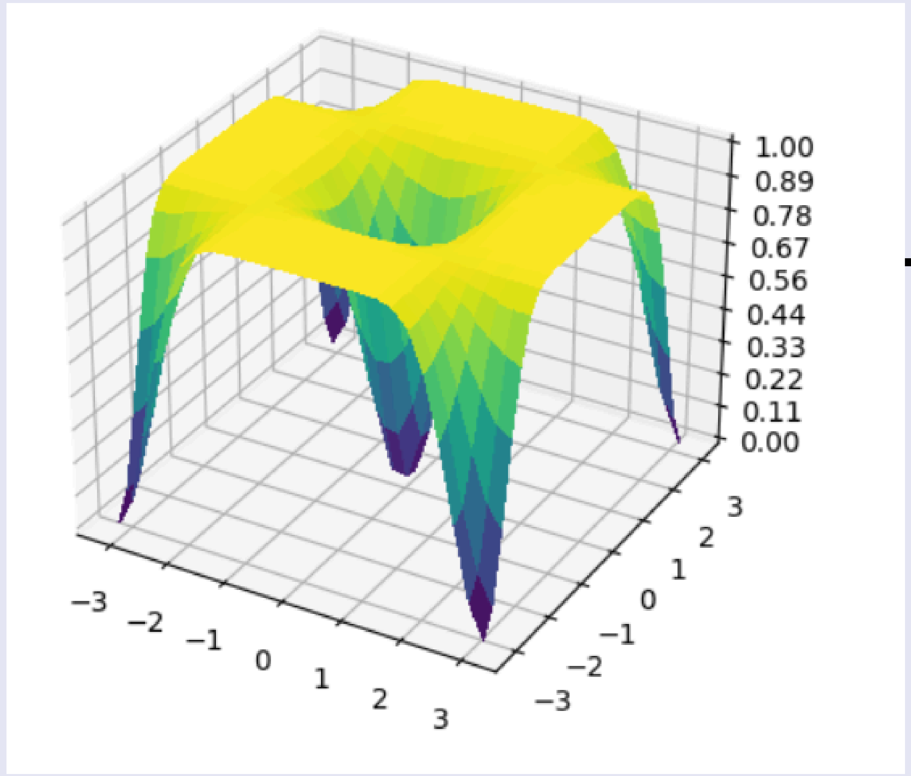
HOW TO MITIGATE BARREN PLATEAUS

Barren plateau phenomenon can, under some circumstances, be avoided by using local cost functions that only have information from part of the circuit. These local cost functions can be more robust against noise.

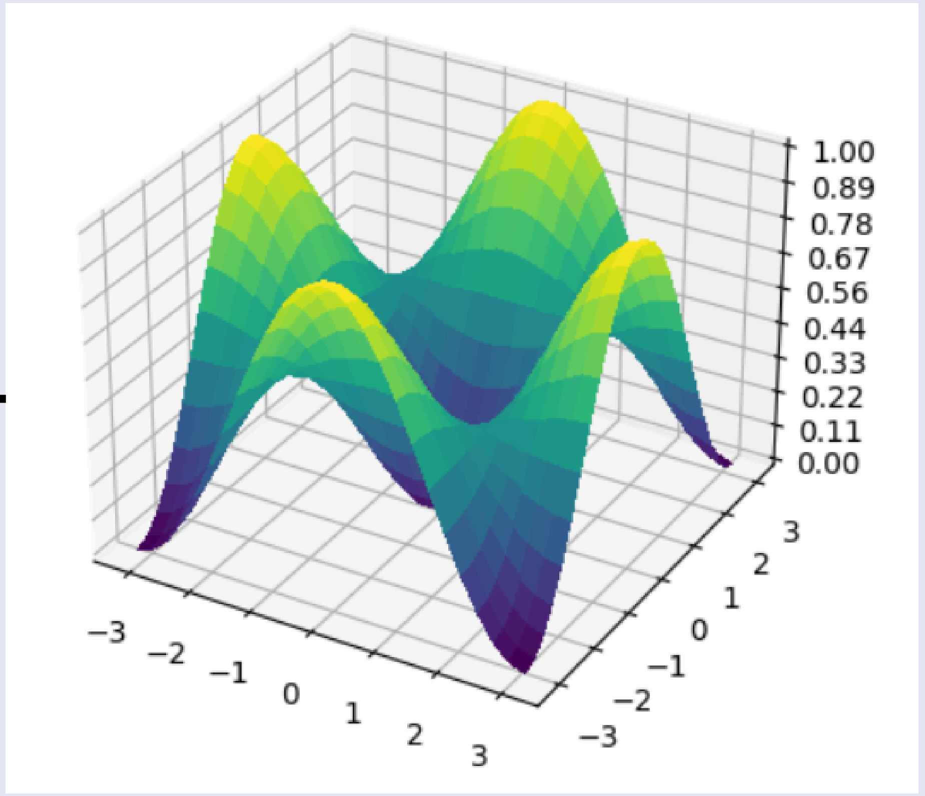
A local cost function only considers information from a few qubits, and attempts to analyse the behaviour of the entire circuit from this limited scope. They are bounded by the global ones.

In the global case, anywhere between 70-80% of starting positions are untrainable, a significant number. It is likely that, as the complexity of our ansatz—and the number of qubits—increases, this factor will increase.

We can compare that to our local cost function, where every single area is trained, and most are even trained in less time.



$\longrightarrow$ Global Cost Function

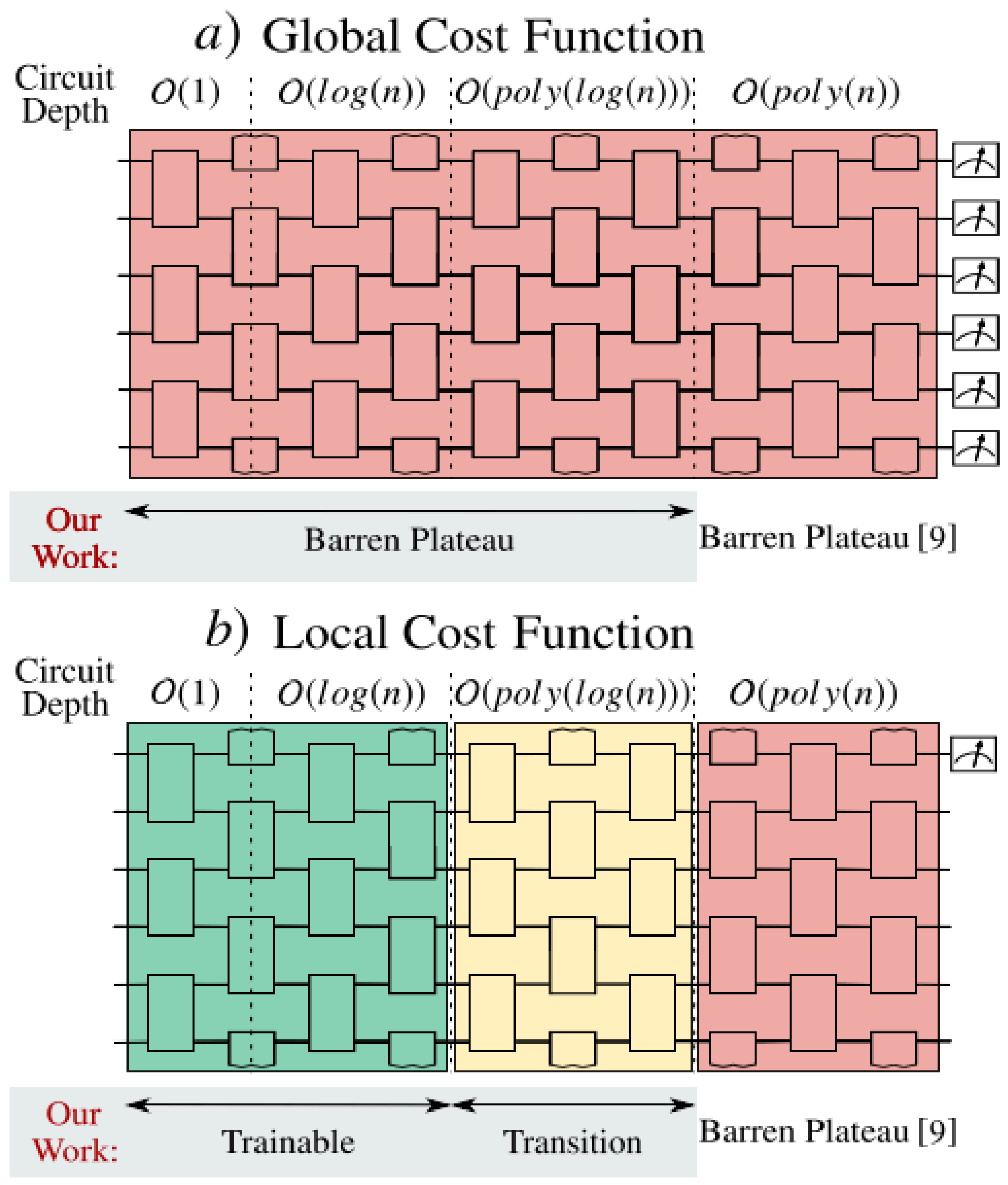


Local Cost Function $\longleftarrow$

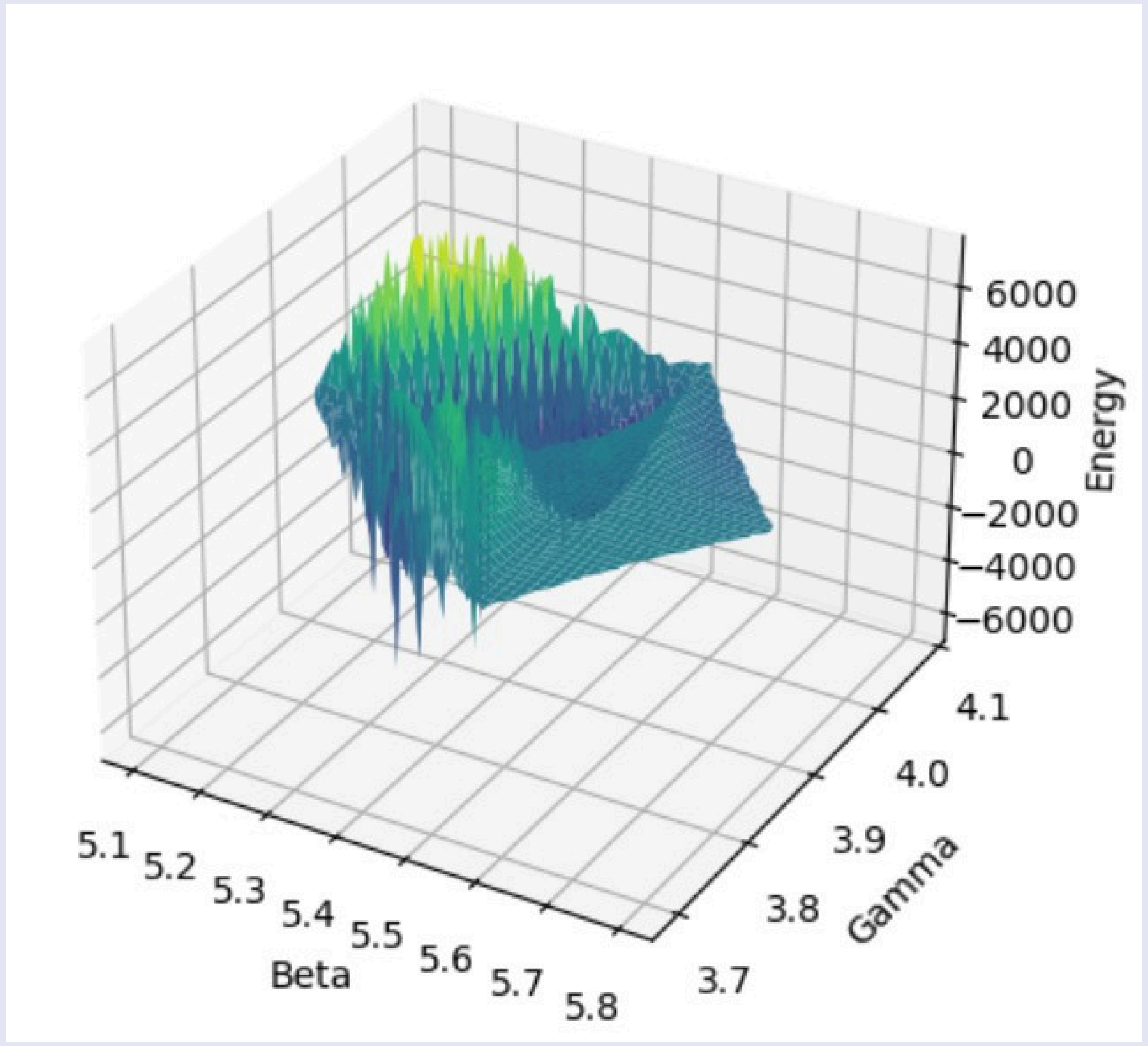
HOW TO MITIGATE BARREN PLATEAUS

- Experimenting with gradient-free optimizers: Unlike gradient-based optimization algorithms, optimizers such as COBYLA do not rely on gradient information to optimize the parameters and are therefore less likely to be affected by the barren plateau.

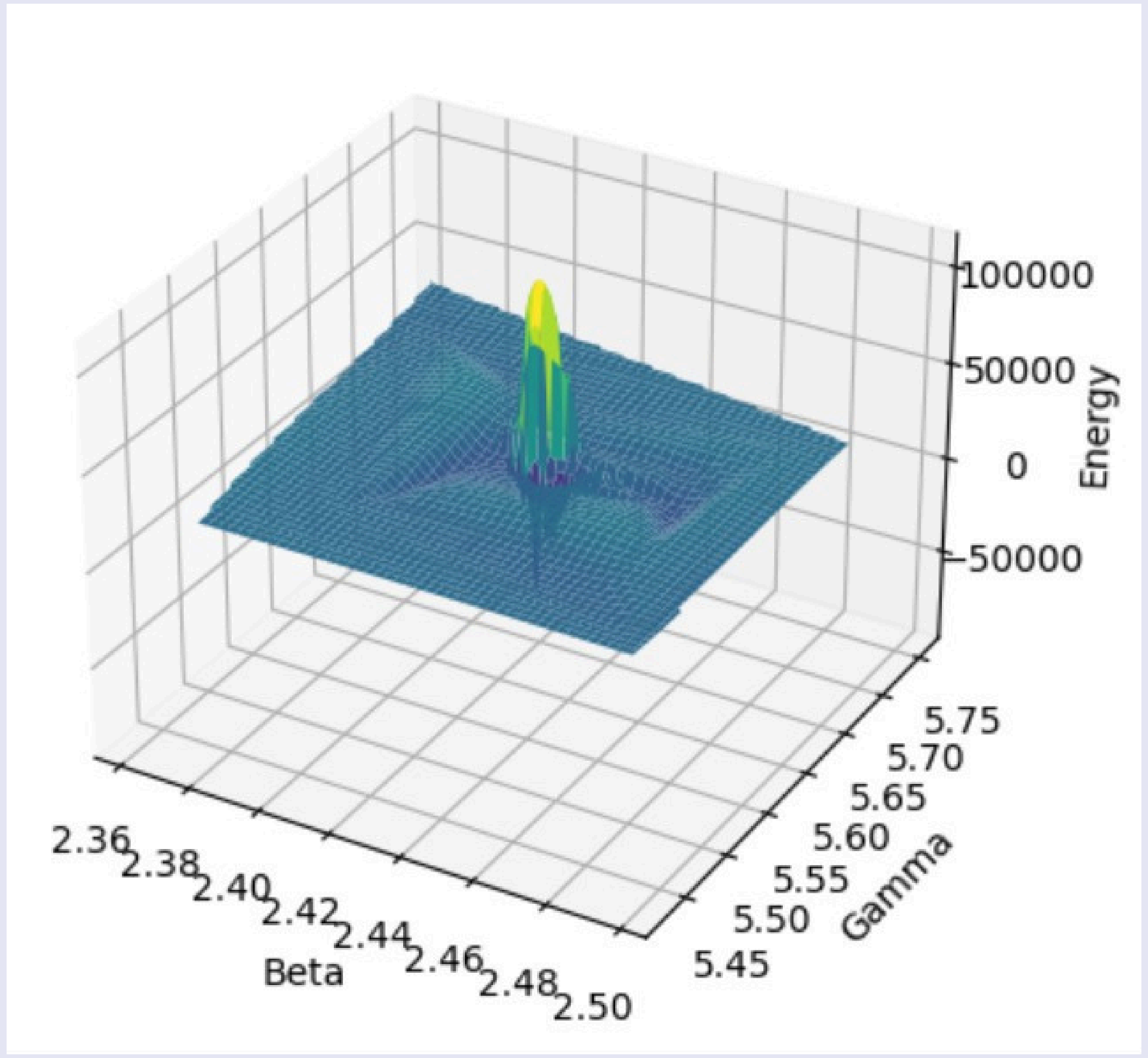
- Bootstrapping can help the optimization loop avoid getting stuck in a parameter space where the gradient is small. (What is Bootstrapping? - It is a resampling technique that involves repeatedly drawing samples from our source data with replacement, often to estimate a population parameter)



BARREN PLATEAUS – OUR IMPLEMENTATION



For 3 layers



For 5 layers

CONCLUSION

Local cost functions provide more detailed feedback during optimization, preventing the flat landscapes that hinder gradient-based methods. This approach significantly improves QAOA's performance, making it more effective for solving complex problems like the Traveling Salesman Problem.

QAOA's hybrid quantum-classical approach shows potential in handling TSP and other combinatorial optimization problems by encoding it into a cost Hamiltonian and using variational methods for optimization.

In conclusion, while QAOA is still in development, its promise in solving complex problems like TSP suggests a transformative impact on various fields. As quantum technology and hardware matures, QAOA will only get more robust.

REFERENCES

- https://cs269q.stanford.edu/projects2019/DudasHenry_Y.pdf
- https://pennylane.ai/qml/demos/tutorial_local_cost_functions/
- https://learning.quantum.ibm.com/course/variational-algorithm-design/optimization-loops#gradient-free
- https://medium.com/mit-6-s089-intro-to-quantum-computing/quantum-approximate-optimization-algorithms-on-the-traveling-salesman-problem-703b8aee6624
- https://www.youtube.com/watch?v=GiDsjIBOVoA
- https://mediatum.ub.tum.de/doc/1661426/1661426.pdf
- https://qiskit-community.github.io/qiskit-optimization/tutorials/06_examples_max_cut_and_tsp.html
- https://cs269q.stanford.edu/projects2019/radzihovsky_murphy_swofford_Y.pdf
- https://arxiv.org/abs/2001.00550
- https://arxiv.org/abs/2101.08448
- https://chatgpt.com/
- https://www.tensorflow.org/quantum/tutorials/barren_plateaus