Quantum Optimization at UNC Charlotte

This project is a hands-on way to understand how **quantum computers can tackle real-world optimization problems** using the **Quantum Approximate Optimization Algorithm (QAOA)**. We're focusing on a famous challenge called the **Traveling Salesman Problem (TSP)** — finding the shortest route that visits a set of locations once and returns to the start.

To make it personal, we'll use **UNC Charlotte buildings** instead of random cities. You can enter any buildings you want (Woodward Hall, EPIC, Atkins, etc.), and the notebook will calculate the best route between them — both **classically** and **quantumly**.

The Big Idea

Energy Landscapes

Quantum computers don't think in paths or distances like humans do. They think in **energy landscapes** — where low energy corresponds to better (shorter) routes.

QAOA lets us encode our problem into a **Hamiltonian**, which is basically a function describing how "expensive" each possible route is.

Distance → Energy

The distance between your buildings becomes energy in the quantum system

Parallel Exploration

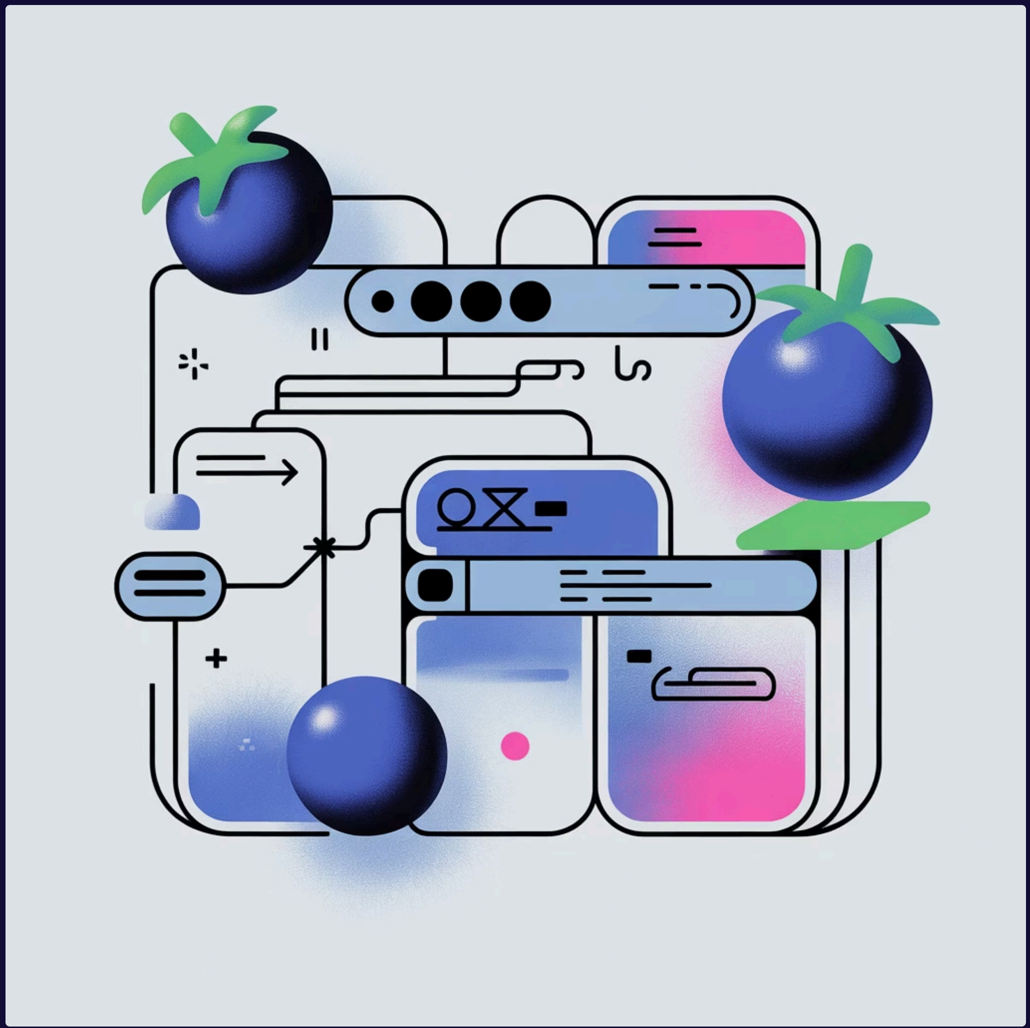
The quantum circuit explores this landscape in parallel

Classical Optimization

A classical optimizer nudges it toward lower energy states

Two Worlds: Classical vs Quantum

Classical (OR-Tools)



We use Google's OR-Tools as a baseline solver. It checks routes and gives the guaranteed shortest path. That's our **ground truth** — fast, exact, and easy to interpret.

- Deterministic approach
- Guaranteed optimal solution
- Fast for small problems

Quantum (QAOA)

The quantum version builds the *same* problem as an energy model, then uses a **parameterized circuit** to minimize that energy. Instead of brute-forcing routes, it tries to "slide downhill" in the energy landscape.

- Probabilistic approach
- Approximate solution
- Parallel exploration

How It Works Step-by-Step

01

Input Your Buildings

Choose your personal UNC Charlotte landmarks

02

Fill Distance Matrix

Enter numbers from Google Maps or walking times

03

Visualize the Network

See every connection and cost on a circle graph

04

Classical Solver

OR-Tools finds the optimal route

05

Quantum Solver

QAOA represents data as energy function and learns best path

06

Compare & Visualize

See both routes side by side

07

Inspect Hamiltonian

See how distance numbers became quantum physics math

The Sliders: Controlling Quantum Intelligence

If your notebook includes sliders (like reps, maxiter, shots, or penalty), they control how "smart" or "deep" the quantum circuit goes. You can **drag these sliders** to see how quantum optimization trades speed for accuracy in real time.

Reps (Layers)

How many alternating "cost" and "mixing" layers the circuit uses. More layers = more learning power, but slower runs.

Maxiter

How many times the classical optimizer tweaks the parameters. More iterations = better results, longer runtime.

Shots

How many times the circuit is measured per iteration. Higher shots = smoother statistics, but slower.

Penalty Weight

How hard we punish invalid routes (like visiting a city twice). High penalty = stricter, more accurate tours.

Why It Matters

This is not just a "lab" — it's a micro-version of real quantum research. The same math and code structure are used in projects at **IBM Quantum**, **NASA**, **Google**, and **CERN**, just on larger systems.

Problem Translation

How to translate a real-world problem into a quantum-compatible form (QUBO/Hamiltonian)

Hybrid Optimization

How quantum-classical optimization loops actually function in practice

Result Interpretation

How to interpret energy, bitstrings, and measured states as real-world results

> "Even though we're just walking between campus buildings, this is exactly how researchers test and benchmark quantum algorithms today."

What You'll See

- **Clear Classical Route**

  The perfect answer from OR-Tools

- **Quantum Approximation**

  A nearly identical route from QAOA

- **Visual Graphs**

  Order and cost of each path displayed

- **Energy Minimization**

  Watch the circuit learn in action

- **Comparison Table**

  Proof that both approaches agree

When both routes have the same cost, it means your quantum solver correctly matched classical reality — proof that your mapping worked.

What to Tell the Room

> **"We're not just finding paths. We're learning how to make real-world problems understandable to quantum systems."**
>
> Every distance number you type in becomes an energy term that a circuit can literally feel.

That's what makes this project powerful — it connects abstract quantum theory directly to something you can see, change, and understand. It bridges the gap between theoretical quantum mechanics and practical problem-solving.

Background: What You're Actually Doing

This project takes a **classical optimization problem** (the Traveling Salesman Problem, or TSP) and shows how **quantum computers** can represent and attempt to solve it using **QAOA (Quantum Approximate Optimization Algorithm)**.

The core goal is *not speed*. The point is to **learn how to express a real-world problem as a Hamiltonian** (an energy function a quantum circuit can minimize). Once you can map a real cost problem to energy, you're doing genuine quantum optimization work — the same math used in current IBM and Google research.

This fundamental skill of problem translation is what separates theoretical understanding from practical quantum computing applications.

What's the TSP (Traveling Salesman Problem)?

Imagine you want to visit several UNC Charlotte buildings — like Woodward Hall, Atkins Library, EPIC, and the Student Union — **exactly once** and return to the starting point.

The Question

What's the shortest possible loop that does that?

The Challenge

Brute force checks all possible orders — but that's exponential

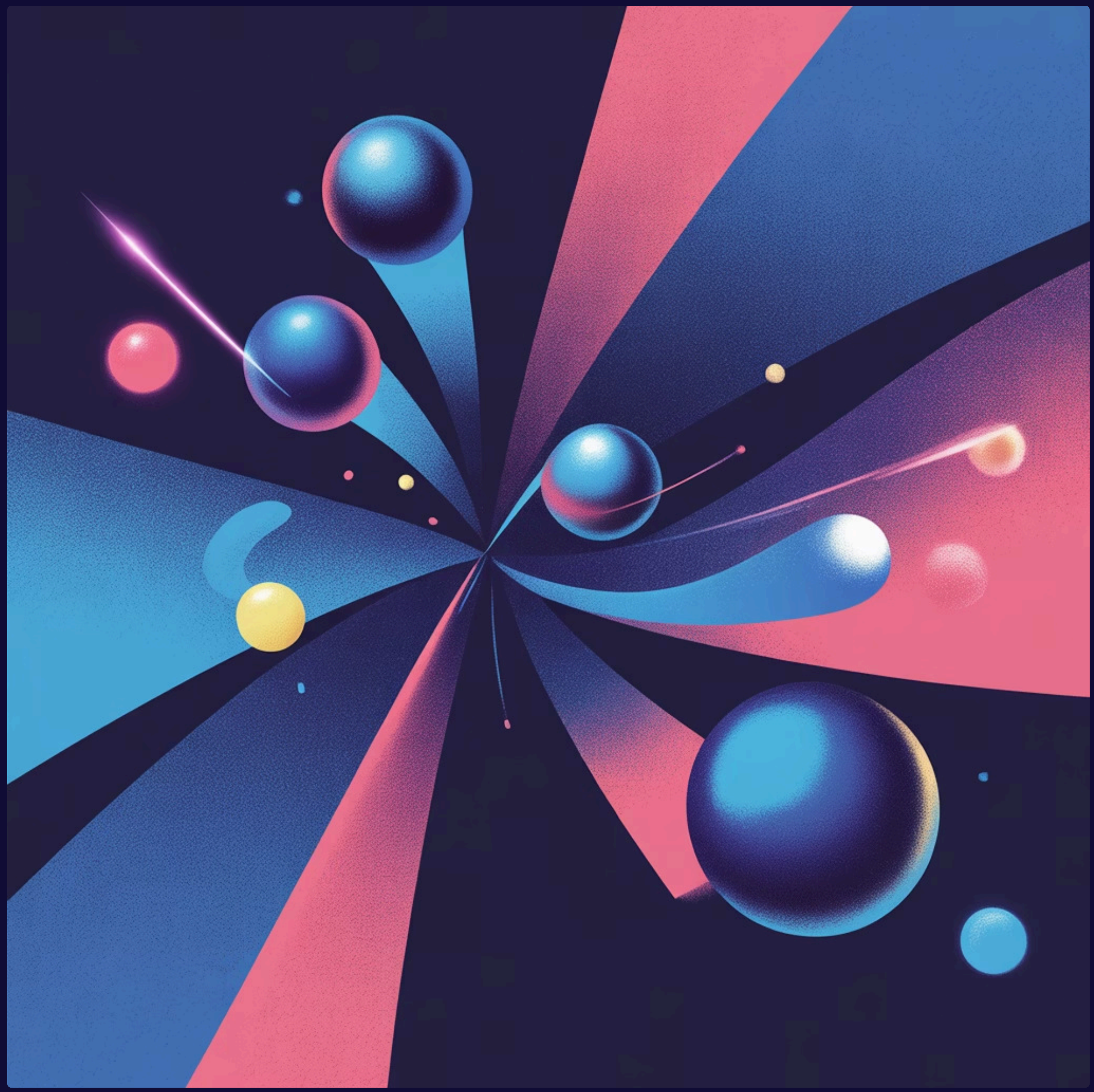
The Scale

With 4 buildings, it's easy. With 20, there are billions of possible tours

Why Quantum Fits Here

Parallel Exploration

Quantum computers are naturally good at exploring **many possible combinations at once**. The QAOA algorithm takes your problem (the TSP) and turns it into an **energy landscape** — low energy = good routes, high energy = bad routes.



Energy Surface

Problem becomes a landscape to explore

Parameterized Circuit

Quantum circuit "feels" the energy

Learning Process

Circuit learns where shortest path hides

The classical optimizer adjusts angles in the quantum circuit to reduce that energy — kind of like the circuit learning where the shortest path hides.

What the Distance Matrix Means

The **distance matrix (D)** is literally your map of travel times between every pair of buildings. It's the foundation of everything that follows.

	Woodward	Atkins	EPIC	Union
Woodward	0	8	6	7
Atkins	8	0	5	4
EPIC	6	5	0	9
Union	7	4	9	0

Row i → column j means cost from i to j

Diagonal = 0 because you can't "travel" from a place to itself

D[i][j] = D[j][i] means travel is symmetric

In the notebook, you can edit this manually — that's what makes it *personal and fun*. Everyone in class can pick their own UNCC spots.

Classical Solver: OR-Tools

Before quantum, you run a classical algorithm (Google OR-Tools). This establishes our baseline for comparison.

01

Convert to Integers

OR-Tools likes whole numbers, so we convert your matrix

02

Build Routing Model

Create the mathematical representation of the problem

03

Find Best Path

Algorithm searches for optimal solution

04

Calculate Total Cost

Sum up the distances for the complete route

3

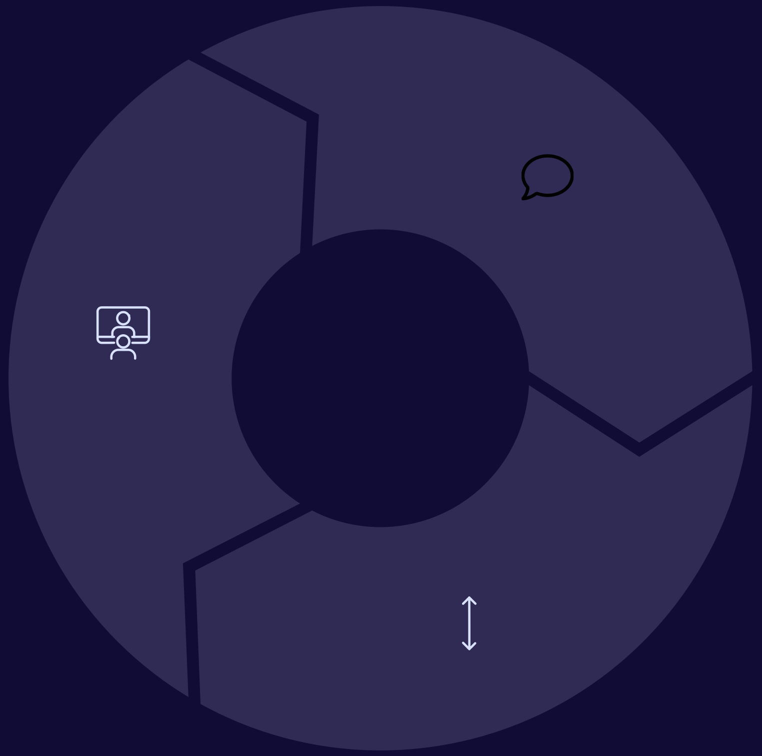
Key Outputs

Route by index, route by name, and total cost

This gives you a **ground truth** answer to compare against the quantum one.

Quantum Part: How QAOA Works

QAOA = Quantum Approximate Optimization Algorithm. It's a *hybrid* approach that combines the best of both worlds.



Classical Computer

Builds the problem as a Hamiltonian (energy function)

Quantum Circuit

Runs with adjustable parameters (angles)

Classical Optimizer

Tweaks parameters to make measured energy smaller

Repeat until the energy (cost) stops improving. Each variable (x_{it}) in your code represents whether building i is in position t of the tour. The circuit encodes all possible tours as quantum states, and its energy landscape mirrors your distance matrix.

What Each Slider Means

If your notebook includes sliders, here's what they control and how they affect your results:

Slider	What It Controls	What Happens When You Increase It
reps (layers)	How many alternating layers of "cost" and "mixing" unitaries QAOA applies	More expressive circuit → can find better solutions, but slower
maxiter	How many times the classical optimizer tweaks the angles	More tuning → possibly better energy, but longer runtime
shots	How many times you measure the circuit per iteration	More shots → better statistical confidence, slower runtime
penalty weight	How strongly we punish invalid routes	Higher = stricter constraint satisfaction

When you drag those sliders, you're balancing **accuracy vs speed**. Small values → fast but approximate. Large values → more precise but slow.

The Steps in Order

- **Import Qiskit & Libraries**

  Set up dependencies

- **Define Buildings + Matrix**

  Personalize it

- **Visualize Network**

  Graph with edge labels

- **Run OR-Tools**

  Get perfect classical route

- **Build QUBO**

  Encode distances as equations

- **Convert to Ising/Hamiltonian**

  Quantum-friendly format

- **Run QAOA**

  Circuit minimizes energy

- **Decode Bitstring**

  Translate to city order

- **Compare & Visualize**

  See how close quantum gets

- **Inspect Single D[i][j]**

  See where number becomes energy

Reading the Outputs

Objective Value

The minimized Hamiltonian energy (includes penalties)

Tour by Index

Numerical order in your list

Tour by Name

Human-readable route

Recomputed Cost

Real travel distance/time using your matrix D

QAOA objective ≈ Classical cost → quantum found the optimum!

Why This Isn't "Just a Dumb Lab"

You're not just solving a toy math puzzle. You're learning **how to translate the real world into quantum language.**

QUBO Mapping

How to map real data (distances) into a binary quadratic model

QAOA Behavior

How QAOA behaves and what it's optimizing

Hybrid Loops

How quantum-classical hybrid loops actually work (feedback optimization)

Realistic Expectations

Why quantum doesn't always beat classical, but why it's powerful when scaled

At big labs (IBM Quantum, Google Quantum AI, CERN), researchers use this same mapping strategy for logistics, molecule folding, power-grid optimization, and materials discovery. Your small campus version is the same concept, just tiny enough to run in class.

How to Visualize Everything

When presenting this project, you'll have multiple visualization tools at your disposal:

Graph of Buildings

Shows every connection and distance between locations

Circle Tour Plot

Thick line follows the visiting order around a circle

Comparison Table

Side-by-side routes and costs for easy comparison

Energy Plot

Shows cost vs iteration — watch QAOA learning over time

Heatmap

Shows which distances dominate your cost

When you drag the sliders, re-run the QAOA cell, and watch the energy drop — that's the circuit getting "smarter."

Big Picture Summary

Classical vs Quantum	What's Happening	What You Learn
OR-Tools	Searches deterministically for the best route	How classical solvers define and optimize cost functions
QAOA	Builds an energy landscape and minimizes it using a circuit	How optimization maps onto quantum hardware
Both Together	Give you the same cost (for small n)	Proof your mapping and reasoning are correct

The Real Achievement

When both approaches converge on the same answer, you've successfully bridged two computational paradigms. This is the foundation of practical quantum computing.

You've learned to speak the language of quantum optimization — a skill that will only become more valuable as quantum computers scale up and tackle increasingly complex real-world problems.