SKKU 2025 Challenge #3: Traveling Salesman Problem

Introduction

The TSP (Traveling Salesman Problem) is a classic combinatorial optimization problem (see, for example https://en.wikipedia.org/wiki/Travelling_salesman_problem). We refer to it as a combinatorial problem because the space of solutions is not continuous and instead has a discrete structure. The problem, simply stated, is to visit each of a set of cities for which the pairwise distances are specified. We visit the cities one at a time in a specified order, accumulating the pairwise distance between each pair of cities as we make progress. The tour, having visited all cities, returns to its starting point. The total distance required by the tour is to be minimized.

This problem belongs to a class of optimization problems called NP-hard problems. For this reason, it is interesting to study the performance of quantum computing algorithms applied to this problem. When applying a circuit model quantum processor to a combinatorial optimization problem, a standard tool is the Quantum Approximate Optimization Algorithm (QAOA), first described by Farhi et al. In this challenge, we will analyze two different encodings for a 4-city TSP within the context of a QAOA circuit.

Both encodings of this problem require six qubits and thus might be expected to exhibit similar performance, however there is a crucial difference between the two encodings. In the first encoding, which we call leg-variable encoding, there are constraints which must be satisfied for the pattern of activated qubits to correspond to a valid solution. In the second encoding scheme, which we call switch-network encoding, there are no constraints, but the objective function is more complicated than the one arising from leg-variable encoding. The highest order terms in the objective function for the leg-variable encoding are quadratic, whereas for the switch-network encoding there are sixth order terms.

The goal of this challenge is to compare these two encoding schemes in the context of a 4-city TSP as handled by QAOA. We will create the QAOA circuit for both encodings for various values of p, the parameter that controls the number of layers in the QAOA circuit ansatz. Given a limit on circuit depth, we will find the maximum probability that can be attained in the ground state for both encodings.

Part 1 : Leg-variable encoding of the TSP

Leg-variable encoding associates a Boolean variable to each pair of cities in the TSP. Denote the four cities a, b, c, d and the six variables $x_{ab}, x_{ac}, x_{ad}, x_{bc}, x_{bd}, x_{cd}$. Variable x_{ij} has the value 1 if a tour passes from city i to city j directly via the leg connecting the two cities. If a tour does not traverse this leg, then x_{ij} has the value 0.

A TSP over N cities generally has $(N-1)!/2$ distinct tours. Any permutation of the N cities corresponds to a tour and there are $N!$ permutations of the N cities, however the permutations do not all correspond to distinct tours. The starting point of a tour can be any of the N cities of the permutation, and the direction of a tour can either proceed forward or backward along the permutation. This means that $2N$ permutations correspond to the same tour and thus the $N!$ distinct permutations give rise to $\frac{N!}{2N} = (N-1)!/2$ distinct tours.

For four cities there are three distinct tours, as follows:

LEG	VAR	TOUR 1	TOUR 2	TOUR 3
AB	x_{ab}	1	1	0
AC	x_{ac}	1	0	1
AD	x_{ad}	0	1	1
BC	x_{bc}	0	1	1
BD	x_{bd}	1	0	1
CD	x_{cd}	1	1	0

Each tour traverses four legs and so each of the columns corresponding to a tour has four 1 entries for the legs contained in the tour and two 0 entries for the legs not contained in the tour. Denoting the distance between cities i and j by d_{ij} the total tour length is then:

$$D = \sum_{x_{ij}} d_{ij} x_{ij}$$

This encoding scheme utilizes six variables and so the number of assignments of $\{0,1\}$ values to the six variables is $2^6 = 64$. Of these 64 configurations, most do not correspond

to valid tours. An example of a configuration that minimizes the objective but does not correspond to a valid tour arises when all x_{ij} are set to 0. In this case the tour has length 0 but clearly is not valid, since the tour never goes anywhere. To have a properly posed problem, additional terms must be added to the objective that are minimized when the configuration corresponds to a valid tour. We call these additional terms penalties.

To ensure that the configuration corresponds to a valid tour, we must enter and exit each city one time. Each city has three legs connecting it to other cities, so the sum of the corresponding three leg variables must equal to 2. Here are the equations capturing these constraints:

$$x_{ab} + x_{ac} + x_{ad} = 2$$

$$x_{ab} + x_{bc} + x_{bd} = 2$$

$$x_{ac} + x_{bc} + x_{cd} = 2$$

$$x_{ad} + x_{bd} + x_{cd} = 2$$

Each of these constraint equations can be converted into a penalty, which vanishes when the constraint is true and otherwise is positive. Here are the corresponding penalties summed together to give a single term:

$$P = (x_{ab} + x_{ac} + x_{ad} - 2)^2 + (x_{ab} + x_{bc} + x_{bd} - 2)^2 + (x_{ac} + x_{bc} + x_{cd} - 2)^2 + (x_{ad} + x_{bd} + x_{cd} - 2)^2$$

The objective for the leg-encoding TSP is a sum of the distance portion D and the penalty portion P. An overall scaling factor λ multiplies the penalty term, so that the penalty is sufficiently strong to cause the system to obey the penalties but is not so strong that the distance portion of the objective is overwhelmed. The final objective for leg-encoding is:

$$Obj = D + \lambda P$$

Exercise for Part 1:

1) Suppose the distances between cities are:

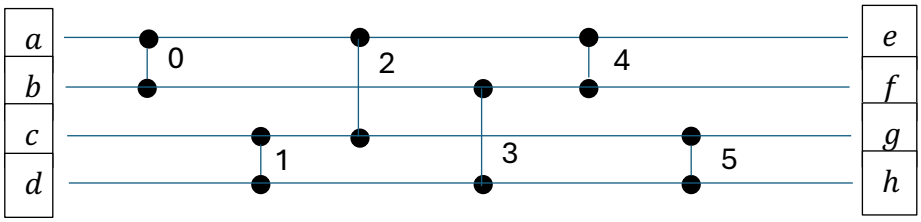
$$(d_{ab}, d_{ac}, d_{ad}, d_{bc}, d_{bd}, d_{cd}) = (1,1,1,2,3,1)$$

What is the smallest integer value of λ such that the Obj values for all valid tours are strictly less than the Obj values for all invalid tours?

Part 2 : Switch-network encoding of the TSP

Switch networks originated in the telecom industry to connect communication endpoints. There are many kinds of switch networks possessing different properties. One interesting example of a switch network (see https://en.wikipedia.org/wiki/Sorting_network) has the property that it can permute numeric inputs so that they arrive at the output of the switch network in sorted order. For our exercise we do not need to sort the inputs, but we do require that the network can permute the inputs into any desired output order. This is how we will parametrize the possible tours for the TSP.

The switch network that we will use has six switches arranged as follows:



Imagine data flowing from left to right through the network on the horizontal lines. When data encounters one of the six switches, it will either move straight through, or it will exchange places with the other data entering the switch. For example, when a and b encounter switch 0, they will either exit on the same lines, or they will interchange lines. Each switch has two settings and since there are six switches in the network, there are $2^6 = 64$ possible settings for the switches. We state without proof that all possible permutations of the input ports to the output ports can be realized through some setting of the switches.

We use this network to encode the 4-city TSP as follows. Inject the four cities into the input ports on the left according to the alphabetic labels on the switch network. Denote the variables controlling the switches by $x_0, x_1, x_2, x_3, x_4, x_5$. If variable x_i has the value 0 then its two inputs move straight through, but if it has the value 1 the two inputs are interchanged as they flow to the output. The cities will be permuted as they travel through the network so that their order at the output is $\sigma(a), \sigma(b), \sigma(c), \sigma(d)$ where σ is a permutation of four elements. This sequence is the order in which the cities will be visited by the tour, so the distance of the tour is given by:

$$D = d_{\sigma(a),\sigma(b)} + d_{\sigma(b),\sigma(c)} + d_{\sigma(c),\sigma(d)} + d_{\sigma(d),\sigma(a)}$$

Now we need to express D as a function of $x_0, x_1, x_2, x_3, x_4, x_5$. To do this we introduce indicator functions. These are functions of the six switch variables whose value tells us whether a given input is routed to a given output.

We start with an example. The indicator function τ_{ae} tells us whether the top input into the network is routed to the top output. If so, then τ_{ae} has the value 1 and otherwise it has the value 0. There are exactly two possible ways to route the top input to the top output. The first possibility requires switches x_0, x_2, x_4 to all be turned off or, equivalently, equal to 0. The second possibility requires switches x_0 and x_4 to be turned on and switch x_3 to be turned off. We can express this as follows:

$$\tau_{ae} = (1 - x_0)(1 - x_2)(1 - x_4) + x_0(1 - x_3)x_4$$

Note that even though τ_{ae} is conceptually dependent on all six switch variables, it is independent of x_1 and x_5.

Since there are four input ports and four output ports, there is a total of sixteen indicator functions. Each one assumes values 0 or 1 depending on whether the setting of the switch variables routes its input port to its output port or not. We can use these sixteen indicator functions to express D as a function of the six switch variables. Suppose i and j are two input cities. D will contain the term d_{ij} provided cities i and j arrive at adjacent output ports.

One way that cities i and $j$arrive at adjacent output ports is if i is routed to output port e and j is routed to output port f. These two conditions are expressed by indicator functions τ_{ie} and τ_{jf} respectively. The product $\tau_{ie}\tau_{jf}$ then is equal to 1 when i is routed to e and j is routed to f. To count the distance between cities i and j we would then include the term $d_{ij}\tau_{ie}\tau_{jf}$ in D. This is only one of eight terms in D that multiply the distance d_{ij}. The complete list of terms containing the distance d_{ij} is as follows:

$$d_{ij}(\tau_{ie}\tau_{jf} + \tau_{if}\tau_{je} + \tau_{if}\tau_{jg} + \tau_{ig}\tau_{jf} + \tau_{ig}\tau_{jh} + \tau_{ih}\tau_{jg} + \tau_{ih}\tau_{je} + \tau_{ie}\tau_{jh})$$

The distance d_{ij} is one of six distances provided for the 4-city TSP. Each of the six distances makes a similar contribution to D.

Note that the indicator function τ_{ae} is a cubic function of the switch variables. This is true for all sixteen indicators functions. This can be seen by noting that any route from an input port to an output port must pass through three switches. When we form products of two indicator functions, as we do when we build D, the terms will contain products of up to six switch variables. The essential trade-off in using the switch-network encoding is that we have avoided constraints in the objective. As a result the objective has higher order terms.

Exercises for Part 2:

2.a) Write out the objective D as a function of the six switch variables $x_0, x_1, x_2, x_3, x_4, x_5$. (Hint: rather than keeping track of all the variables manually, use a symbolic processing

package like SymPy). Remember that Boolean variables are idempotent, meaning that they equal their own square:

$$x^2 = x$$

Apply this equation for each switch variable to make sure that none of the switch variables appears squared or to any power higher than one.

2.b) Evaluate the objective D for all 64 possible settings of the switch variables. How many times does each of the three possible tours appear?

Part 3 : Comparison of QAOA performance

QAOA circuits begin with a Hadamard gate applied to each qubit to initialize the entire qubit register into a uniform superposition of all computational basis states. This is followed by p circuit layers, each of which consists of the exponentiated Hamiltonian followed by a mixer operator. Both the exponentiated Hamiltonian and the mixer operator are controlled by parameters. The i^{th} exponentiated Hamiltonian is controlled by parameter γ_i and the i^{th} mixer is controlled by parameter β_i. Following these circuit layers, all qubits are measured to yield the states of the corresponding Boolean variables. QAOA can be viewed as standardized approach to build a variational wavefunction where the structure of the wavefunction is governed by the circuit. See Farhi et al in arXiv:1411.4028 for more information.

Follow these four steps to build a QAOA circuit:

1. Build the QUBO (or HUBO): A QUBO is a Quadratic Unconstrained Binary Objective and a HUBO replaces Quadratic with Higher order. In either case, the objective is a function of variables x_i taking their values from the set $\{0,1\}$. The objective is constructed so that when it is minimized, the variables represent the solution to some problem. For the TSP, minimizing the objective means that the variables take on values which represent the shortest tour.
2. Replace the x_i variables with spins s_i according to the equation: $x_i = \frac{1-s_i}{2}$ so that the s_i take their values from the set $\{+1, -1\}$. We call the s_i spins following physics convention, since they are commonly used to represent a classical spin which can point either up or down. Rewrite the QUBO (or HUBO) in terms of the spin variables to obtain an Ising model. This is the name for a model of a system of magnetic spins which can interact with each other and an external magnetic field.
3. Replace each spin s_i with a corresponding Pauli $\sigma_z^{(i)}$, which is a 2×2 matrix representing an operator which generates rotations around the z-axis. When spins s_i and s_j appear in a product in the objective, they become an implicit tensor

product of the corresponding Pauli matrices. To form the operator which acts on the wavefunction, each spin which is missing from a term in the objective is replaced by an identity matrix. (Note that physics notation generally suppresses all tensor products in favor of indices). To build the corresponding matrix using a programming library like NumPy, each spin index i will be associated with either $\sigma_z^{(i)}$ or an identity matrix, and the entire collection of matrices is combined using tensor products. The result is that the Ising model becomes a quantum Hamiltonian H. Since all the operators appearing in the construction are diagonal, H is also purely diagonal. Any value that can arise when the QUBO (or HUBO) is evaluated will appear on the diagonal of the matrix representing H.

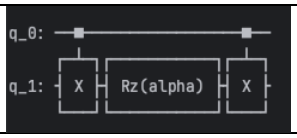
4. Now that we have specified H we can write the ansatz that guides the construction of the quantum circuit. The QAOA circuit begins in a uniform superposition of all computational basis states, which can be achieved by applying a Hadamard gate to each of the Q qubits. Following this, the circuit consists of p repeating layers. Each layer starts with an exponentiated Hamiltonian which corresponds to time evolution through a time γ_i for layer i. After applying the exponentiated Hamiltonian, apply a mixer operator. This corresponds to time evolution through a time β_i for each qubit. Repeat this layer consisting of an exponentiated Hamiltonian followed by a mixer operator p times. The final state produced is a function of all the individual γ_i and β_i parameters which we denote $|\vec{\gamma}, \vec{\beta}\rangle$:

$$|\vec{\gamma}, \vec{\beta}\rangle = \left(\prod_{j=1}^{Q} e^{-i\beta_p \sigma_x^{(j)}}\right) e^{-i\gamma_p H} \dots \left(\prod_{j=1}^{Q} e^{-i\beta_1 \sigma_x^{(j)}}\right) e^{-i\gamma_1 H} |+\rangle$$

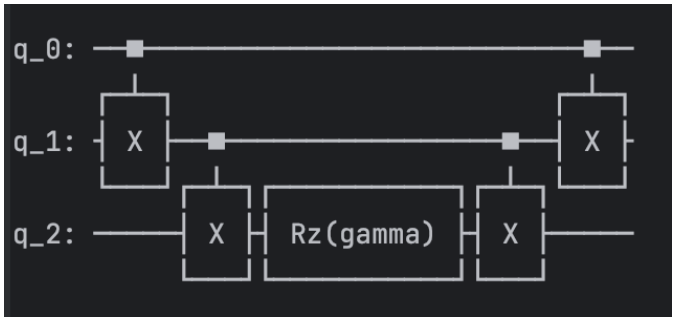
To obtain the range over which we should vary the β_i parameters, note that the eigenspectrum of $\sigma_x^{(j)}$ is $\{+1, -1\}$. The gap between the two states is 2 and so the range for β_i is $[0, \pi]$ because if β_i is outside this range, its action on the two states can be replicated by some other value in $[0, \pi]$. A similar argument can be applied to the γ_i parameter, but since the eigenvalues of H are integers, the range becomes $[0,2\pi]$.

Different ideas for initializing and tuning the γ_i and β_i parameters have been explored. If we mimic adiabatic evolution and apply Trotterization, both parameters follow a linear trajectory with i. The β_i parameter starts at its maximal value and decreases linearly with i while the γ_i parameter starts at its minimum value and increases linearly with i.

To complete the conversion of an exponentiated Hamiltonian and mixer to gates in a quantum circuit, refer to the following table:

Operator	Gate	Simplified gadget
$e^{-i\gamma a\sigma_z^{(i)}}$		
$e^{-i\gamma b\sigma_z^{(i)}\sigma_z^{(j)}}$		
$e^{-i\beta\sigma_x^{(i)}}$		

This table shows how to exponentiate Hamiltonian terms linear and quadratic in $\sigma_z^{(i)}$. This is sufficient for the leg-variable encoding of the TSP, but the switch network contains terms up to order six in the $\sigma_z^{(i)}$. Converting these higher order terms to gadgets requires a simple extension of the method used to handle the quadratic terms in the table. We illustrate the extension via an example with a single cubic term. To exponentiate $\sigma_z^{(i)}\sigma_z^{(j)}\sigma_z^{(k)}$ extend the network of CNOT gates as follows:



This pattern extends for higher order terms. For an n^{th} order term, the gadget begins and ends with a ladder of $n-1$ CNOT gates and these two ladders sandwich a single RZ gate. For a sixth order term there will be two ladders comprising five CNOT gates each and a single RZ gate sandwiched in the middle. The ladder technique requires a total of $2n-2$ CNOT gates for an n^{th} order term. See Campbell et al in arXiv:2402.11099v1 for an optimization technique which uses parity networks to reduce the count of CNOT gates required by the ladder technique.

Exercises for Part 3:

3.a) Using the pairwise distances between cities, build a QAOA circuit for $p=1$. How many CNOT gates does it contain?

3.b) Using the switch network encoding described in Part 2, build a QAOA circuit for $p=1$. How many CNOT gates does it contain?

3.c) Build a QAOA circuit for arbitrary p for the leg-variable encoding and initialize both β_i and γ_i to be linear functions of i. Choose p so that the total number of CNOT gates in the circuit does not exceed 3000. What is the maximum ground state probability you can obtain by varying the linear trajectories for β_i and γ_i?

3.d) Build a QAOA circuit for arbitrary p for the switch network encoding and initialize both β_i and γ_i to be linear functions of i. Choose p so that the total number of CNOT gates in the circuit does not exceed 3000, taking advantage of gate cancellation where possible. What is the maximum ground state probability you can obtain? Note that the ground state can be achieved in more than one computational basis state, so the ground state probability is a sum over probabilities obtained from appropriate computational basis states.