Lesson 6.2: QAOA (Quantum Approximate Optimization Algorithm)

Mathematical Intuition · Module 6 — Capstone

Node 6.2

Overview

Complete decoding of QAOA—a hybrid quantum-classical algorithm for combinatorial optimization.

The Algorithm

QAOA State Preparation

$|\psi(\beta,\gamma)\rangle$ = U(β_p,γ_p)...U(β_1,γ_1)|+⟩^⊗n

where each layer is:

U(β,γ) = e^(-iβĤ_mixer)e^(-iγĤ_cost)

Cost Function

C(β,γ) = $\langle\psi(\beta,\gamma)|$Ĥ_cost|$\psi(\beta,\gamma)\rangle$

Symbol-by-Symbol Breakdown

Initial State: |+⟩^⊗n

$|+\rangle$ (plus state) - **Definition**: $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ - **Meaning**: Equal superposition of 0 and 1 - **Preparation**: Apply Hadamard gate to $|0\rangle$

⊗**n** (tensor product, n times) - **Meaning**: n qubits, each in $|+\rangle$ state - **Result**: $|+\rangle^{\otimes n} = |+\rangle_1 \otimes |+\rangle_2 \otimes \ldots \otimes |+\rangle_n$ - **Superposition**: Equal amplitude on all 2^n basis states - **Purpose**: Start with maximum uncertainty (explore all solutions)

Cost Hamiltonian: Ĥ_cost

Definition: Encodes optimization problem

Ĥ_cost = Σ_edges J_ij σ_z^i σ_z^j + Σ_nodes h_i σ_z^i

For MaxCut problem:

Ĥ_cost = -½ Σ_(i,j)∈E (1 - σ_z^i σ_z^j)

Breaking it down:

σ_z^i (Pauli Z on qubit i) - **Eigenvalues**: +1 (for $|0\rangle$), -1 (for $|1\rangle$) - **Meaning**: Measures qubit state - **Role**: Encodes binary variable (0 or 1)

σ_z^i σ_z^j (product of Z operators) - **Eigenvalues**: +1 (same state), -1 (different states) - **Meaning**: Measures agreement between qubits i and j - **MaxCut**: Reward different states (cut edge)

J__ij (coupling strength) - **Values**: Problem-dependent weights - **Meaning**: Importance of edge (i,j) - **Sign**: Negative for MaxCut (reward cuts)

Cost Unitary: `e^(-iγĤ_cost)`

e^(-iγĤ__cost) (exponential of cost Hamiltonian) - γ: Angle parameter (to be optimized) - **Operation**: Unitary transformation - **Effect**: Encodes problem structure into quantum state - **Intuition**: "Rotate toward low-cost states"

For diagonal Ĥ__cost:

`e^(-iγĤ_cost)|z⟩ = e^(-iγC(z))|z⟩`

- **C(z)**: Classical cost of bitstring z
- **Effect**: Phase proportional to cost
- **Low cost → small phase; high cost → large phase**

Mixer Hamiltonian: `Ĥ_mixer`

Standard mixer:

`Ĥ_mixer = Σ_i σ_x^i`

σ_x^i (Pauli X on qubit i) - **Operation**: Bit flip ($|0\rangle \leftrightarrow |1\rangle$) - **Eigenvalues**: ± 1 - **Eigenstates**: $|+\rangle$, $|-\rangle$ - **Purpose**: Explores solution space

Why X? - X doesn't commute with Z (cost operator) - Creates superpositions - Enables quantum tunneling between solutions - Classical analog: random walk

Mixer Unitary: `e^(-iβĤ_mixer)`

e^(-iβĤ__mixer) (exponential of mixer) - β: Angle parameter (to be optimized) - **Operation**: Unitary transformation - **Effect**: Mixes computational basis states - **Intuition**: "Explore nearby solutions"

For Ĥ_mixer = Σ_i σ_x^i:

`e^(-iβσ_x) = cos(β)I - i sin(β)σ_x`

- $\beta = 0$: No mixing (identity)
- $\beta = \pi/2$: Hadamard-like (maximum mixing)
- $\beta = \pi$: Full bit flip

QAOA Layer: `U(β,γ) = e^(-iβĤ_mixer)e^(-iγĤ_cost)`

Order matters! 1. **First**: Apply cost unitary e^(-iγĤ_cost) - Encodes problem structure - Adds phase based on cost 2. **Second**: Apply mixer unitary e^(-iβĤ_mixer) - Explores solution space - Creates superpositions

Intuition: "Bias toward good solutions, then explore"

Multiple Layers: `U(β_p,γ_p)...U(β_1,γ_1)`

p layers (circuit depth) - **p = 1**: Single layer (limited expressibility) - **p** $\to \infty$: Can reach any state (universal) - **Trade-off**: More layers → better solutions but more noise

Parameters: 2p total (β_1, γ_1, β_2, γ_2, ..., β_p, γ_p) - Each layer has independent parameters - Optimized classically

Final State: $|\psi(\beta,\gamma)\rangle$

Full expression:

$|\psi(\beta,\gamma)\rangle$ = e^(-iβ_p$\hat{H}$_mixer)e^(-iγ_p$\hat{H}$_cost)...e^(-iβ_1$\hat{H}$_mixer)e^(-iγ_1$\hat{H}$_cost)|+$\rangle$^$\otimes$n

Properties: - Superposition of all 2^n bitstrings - Amplitudes depend on β, γ - Good solutions have higher amplitude (ideally)

Cost Function: C(β,γ) = $\langle\psi(\beta,\gamma)|\hat{H}$_cost$|\psi(\beta,\gamma)\rangle$

$\langle\psi(\beta,\gamma)|$ (bra) - Dual of $|\psi(\beta,\gamma)\rangle$ - Used for inner product

$\hat{\mathbf{H}}$**_cost** (cost Hamiltonian) - Same as before - Observable to measure

$\langle\psi(\beta,\gamma)|\hat{\mathbf{H}}$**_cost**$|\psi(\beta,\gamma)\rangle$ (expectation value) - **Meaning**: Average cost of state $|\psi(\beta,\gamma)\rangle$ - **Measurement**: Sample bitstrings, compute average cost - **Goal**: Minimize C(β,γ) over β, γ

The Complete QAOA Workflow

Step 1: Problem Encoding

- Define cost function C(z) for bitstring z
- Construct $\hat{H}$_cost such that eigenvalues are costs
- Example (MaxCut): $\hat{H}$_cost = -½Σ_(i,j)(1 - σ_z^i σ_z^j)

Step 2: Initialize Parameters

- Choose p (number of layers)
- Initialize $\beta = (\beta_1, \ldots, \beta\_p)$, $\gamma = (\gamma_1, \ldots, \gamma\_p)$
- Common: random or heuristic initialization

Step 3: Quantum Circuit

- Prepare $|+\rangle^{\otimes n}$
- For each layer k = 1 to p:
  - Apply e^(-iγ_k$\hat{H}$_cost)
  - Apply e^(-iβ_k$\hat{H}$_mixer)
- Measure in computational basis

Step 4: Cost Evaluation

- Repeat circuit N_shots times
- Collect bitstrings $\{z_1, z_2, \ldots, z\_N\}$
- Estimate: $C(\beta,\gamma) \approx (1/N)\Sigma_i\ C(z_i)$

Step 5: Classical Optimization

- Compute gradient: $\partial C/\partial\beta$_k, $\partial C/\partial\gamma$_k (parameter-shift rule)
- Update: $\beta \leftarrow \beta - \alpha \cdot \partial C/\partial\beta$, $\gamma \leftarrow \gamma - \alpha \cdot \partial C/\partial\gamma$
- Repeat until convergence

Step 6: Solution Extraction

- Run final circuit with optimal β, γ
- Sample bitstrings
- Return best bitstring (lowest cost)

Example: MaxCut on Triangle Graph

Problem

- **Graph**: 3 vertices, 3 edges (triangle)
- **Goal**: Maximize number of cut edges
- **Optimal**: 2 edges cut (any partition with 1 vs 2 vertices)

Cost Hamiltonian

```
Ĥ_cost = -½[(1 - σ_z^0 σ_z^1) + (1 - σ_z^1 σ_z^2) + (1 - σ_z^2 σ_z^0)]
       = -3/2 + ½(σ_z^0 σ_z^1 + σ_z^1 σ_z^2 + σ_z^2 σ_z^0)
```

Mixer Hamiltonian

```
Ĥ_mixer = σ_x^0 + σ_x^1 + σ_x^2
```

QAOA Circuit (p=1)

1. **Initialize**: H⊗H⊗H|000⟩ = |+++⟩
2. **Cost layer**: e^(-iγĤ_cost)
   - Diagonal in Z basis
   - Adds phases based on cost
3. **Mixer layer**: e^(-iβĤ_mixer)
   - Product of single-qubit rotations
   - e^(-iβσ_x) on each qubit

Optimization

- **Parameters**: β, γ (2 parameters for p=1)
- **Landscape**: C(β,γ) has local minima
- **Optimal**: Found via gradient descent or grid search
- **Result**: $|\psi(\beta,\gamma)\rangle$ has high amplitude on good solutions

Physical Intuition

Adiabatic Connection

- **QAOA ≈ discretized adiabatic evolution**
- **p** $\to \infty$: Approaches adiabatic algorithm
- **Finite p**: Approximate, but faster

Quantum Annealing Analogy

- **Cost layer**: Encodes problem (like problem Hamiltonian)
- **Mixer layer**: Provides quantum fluctuations (like transverse field)
- **Alternation**: Discrete time steps (vs continuous annealing)

Classical Analog

- **Cost layer**: Evaluate objective function
- **Mixer layer**: Random walk / local search
- **Quantum advantage**: Tunneling through barriers

Performance and Limitations

Approximation Ratio

- **Definition**: C_QAOA / C_optimal
- **Depends on**: p (layers), problem instance, optimization quality
- **Typical**: Better than random, worse than best classical (for small p)

Noise Sensitivity

- **Gate errors**: Accumulate over layers
- **Decoherence**: Limits effective p
- **Mitigation**: Error mitigation, noise-aware optimization

Scalability

- **Circuit depth**: O(p) (shallow for fixed p)
- **Parameters**: 2p (grows slowly)
- **Measurements**: O(number of terms in $\hat{H}$_cost)

Practice Problems

1. **MaxCut cost**: For bitstring z = 010 on triangle graph, compute C(z).
2. **Expectation value**: Show that $\langle\psi|\sigma$_zˆi σ_zˆj$|\psi\rangle$ = 2P(same) - 1.
3. **Gradient**: Derive $\partial C/\partial\gamma$ using parameter-shift rule.
4. **Optimal p=1**: For 2-qubit MaxCut, find optimal β, γ analytically.
5. **Mixer design**: Why is $\hat{H}$_mixer = $\Sigma\sigma$_xˆi a good choice? What if we used $\Sigma\sigma$_zˆi?

Key Takeaways

- QAOA alternates cost and mixer unitaries
- Cost layer encodes problem structure
- Mixer layer explores solution space
- Parameters β, γ optimized classically
- More layers (p) $\rightarrow$ better approximation
- Hybrid quantum-classical algorithm
- Applicable to combinatorial optimization problems

Next Steps

Next, we'll decode VQE—the Variational Quantum Eigensolver for finding ground states of quantum systems.

Mathematical Intuition · Module 6 — Capstone · Node 6.2
Concepts: parameterized-circuits, cost-function, superposition, tensor-product, graph-theory, unitary-evolution

hbar.university — own.your.cognition