Network Topology — QUBO + QAOA

Bhagyesh Sharma

September 2025

1 Network Topology Optimization with QUBO and QAOA

1.1 Motivation and Problem Setting

Large-scale cloud networks must route heterogeneous traffic demands across multi-vendor infrastructures. Operators face three conflicting objectives:

1. **Capacity feasibility:** ensuring that no link exceeds its bandwidth limit.
2. **SLA compliance:** meeting end-to-end latency requirements.
3. **Energy efficiency:** minimizing the power drawn by network elements, which directly contributes to carbon footprint.

These requirements make routing optimization **NP-hard**(Non Deterministic polynomial time): decisions for one flow affect others through shared capacity. Classical solvers (ILP/ Integer linear programming, heuristics) are effective at moderate scales, but they become computationally expensive for large, dynamic networks. This motivates exploring **quantum optimization** techniques such as QAOA.

—

1.2 Classical Formulation (ILP $\rightarrow$ QUBO)

We represent the network as a graph $G = (V, E)$, with link attributes: where V: number of nodes, E: logical links

- Capacity: cap_{uv} (Mbps)
- Latency: lat_{uv} (ms)
- Energy cost per unit traffic: ϵ_{uv} (W/Mbps)

Traffic demands $D = \{d_1, d_2, \ldots, d_N\}$ each specify a source, destination, bandwidth, and latency bound. Candidate routes are pre-computed using the *k-shortest paths* algorithm ($k = 3$, latency-weighted).

We introduce binary decision variables:

$$x_{d,p} = \begin{cases} 1 & \text{if demand } d \text{ is routed on path } p, \\ 0 & \text{otherwise.} \end{cases}$$

$$z_d = \begin{cases} 1 & \text{if demand } d \text{ is dropped,} \\ 0 & \text{otherwise.} \end{cases}$$

Objective Function:

$$\min \sum_{d \in D} \sum_{p \in P_d} \left(bw_d \cdot \sum_{(i,j) \in p} \epsilon_{ij} \right) x_{d,p} + \sum_{d \in D} \text{DROP\_PENALTY} \cdot z_d \qquad (1)$$

This minimizes energy while discouraging dropped demands.

Constraints:

$\sum_{d \in D} \sum_{p \in P_d : (u,v) \in p} bw_d \cdot x_{d,p} \leq \text{cap}_{uv}, \quad \forall (u,v) \in E (\text{capacity constraint})$

$\text{x}_{d,p}, z_d \in \{0,1\}, \quad \forall d,p (\text{binary constraint})$

This ILP can be solved classically with CBC. However, we can also encode it as a QUBO by folding constraints into the objective with penalty terms, enabling quantum solvers.

—

1.3 QAOA Formulation

QAOA is a variational quantum algorithm for combinatorial optimization.

1. **Encoding the problem:** The QUBO is mapped into a Hamiltonian H. Each binary decision variable corresponds to a qubit, with Pauli-Z operators encoding 0/1 values.

2. **Variational Ansatz:** Qubits are initialized in a uniform superposition:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

A sequence of alternating operators is applied for bit flip which corresponds to explore nearby solution in solution space

$$U(H, \gamma) = e^{-i\gamma H} \qquad \text{(problem unitary)} \qquad (2)$$

$$U(B, \beta) = e^{-i\beta \sum_i X_i} \qquad \text{(mixer unitary)} \qquad (3)$$

3. **Classical feedback loop:** A classical optimizer updates parameters (γ, β) to minimize the expectation value:

$$\langle \psi(\gamma, \beta) | H | \psi(\gamma, \beta) \rangle$$

4. **Measurement:** Repeated sampling yields a probability distribution over bitstrings. The most probable low-energy bitstrings correspond to near-optimal routing decisions.

—

1.4 Results and Interpretation

- **Classical ILP:** provides exact solutions on small-to-medium instances.
- **QAOA:** reproduces feasible solutions on small instances (4–12 qubits), with probabilities concentrated on low-energy bitstrings.
- Each bitstring maps back to a routing decision (e.g., `1010` means demand $1 \rightarrow$ path A, demand 2 dropped, demand 3 $\rightarrow$ path B, etc.).

Scaling to $\sim$40 demands (>40 qubits) is infeasible for current simulators, since statevector simulation requires storing 2^{40} amplitudes ($\approx 10^{12}$). Therefore, we validated QAOA on reduced problem sizes.

—

1.5 Challenges and Limitations

1. **Scalability:** Current simulators fail beyond $\sim$25 qubits; real hardware is noisy and limited.
2. **Penalty sensitivity:** Penalty weights in QUBO strongly affect feasibility and solution quality.
3. **Interpretability:** Mapping between bitstrings and routing variables requires careful decoding.

—

1.6 Outlook and Next Steps

- Hybrid methods: run QAOA on reduced subproblems while handling the rest classically.
- Quantum annealers: D-Wave can already embed thousands of QUBO variables, albeit with different physics.
- Hardware progress: scalable, fault-tolerant quantum devices will enable realistic routing optimization.
- Sustainability: even small improvements in energy efficiency directly reduce the carbon footprint of cloud operators.

Summary: We formulated energy-aware routing as a QUBO, solved it classically (ILP) and tested QAOA for small instances. While current hardware cannot yet handle realistic demand sizes, the workflow demonstrates how quantum optimization can complement classical solvers. Future progress lies in hybrid strategies, annealing hardware, and leveraging quantum advantage for sustainable networking.

2 Heuristic Extensions: ACO and GA

While the classical ILP and quantum QAOA approaches provide a strong foundation, heuristic algorithms such as Ant Colony Optimization (ACO) and Genetic Algorithms (GA) can be incorporated for load balancing and scalable routing. Below we present their mathematical formulations.

2.1 Ant Colony Optimization (ACO)

ACO is inspired by the collective behavior of ants that discover shortest paths using pheromone trails.

Path Selection Probability. Each ant k constructs a path probabilistically. The probability of choosing edge (i, j) at time t is:

$$P_{ij}^k = \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij})^\beta}{\sum_{(i,l)\in \text{allowed}} (\tau_{il}(t))^\alpha \cdot (\eta_{il})^\beta}, \tag{4}$$

where:

- $\tau_{ij}(t)$ = pheromone intensity on edge (i, j) at time t,
- $\eta_{ij} = 1/\text{latency}_{ij}$ = heuristic information (inverse of latency),
- α, β = parameters controlling the influence of pheromone vs. heuristic.

Pheromone Update Rule. After all ants build their paths, pheromones are updated as:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \tag{5}$$

with

$$\Delta\tau_{ij}^k = \begin{cases} Q/C^k, & \text{if ant } k \text{ used edge } (i,j), \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where ρ is the evaporation rate, Q is a constant, and C^k is the cost (energy/latency) of the path chosen by ant k.

Intuition. Ants collectively explore multiple routing options. Good solutions (low energy and balanced utilization) reinforce pheromone trails, biasing future ants toward efficient and balanced routes.

2.2 Genetic Algorithm (GA)

GA mimics biological evolution by iteratively evolving a population of candidate solutions.

Chromosome Encoding. Each chromosome represents a routing assignment: genes correspond to demands, and the value encodes the selected path (or a drop option).

Fitness Function. We minimize a composite cost:

$$f(x) = \text{Energy}(x) + \lambda_1 \cdot \text{CapViol}(x) + \lambda_2 \cdot \text{Var}(u(x)), \tag{7}$$

where $\text{CapViol}(x)$ penalizes capacity violations and $\text{Var}(u(x))$ is the variance of link utilization (a load-balancing metric).

Selection. Chromosomes are selected with probability proportional to their fitness (roulette wheel):

$$P_i = \frac{1/f(x_i)}{\sum_j 1/f(x_j)}. \tag{8}$$

Crossover. Two parents x^a, x^b exchange genes to produce a child:

$$x^{child} = \text{crossover}(x^a, x^b). \tag{9}$$

Mutation. A gene is flipped with mutation probability p_{mut}:

$$x^{mut}(j) = \begin{cases} 1 - x(j), & \text{with probability } p_{mut}, \\ x(j), & \text{otherwise.} \end{cases} \tag{10}$$

Intuition. Over successive generations, fitter chromosomes (low energy, no capacity violation, balanced load) dominate the population, converging to efficient routing assignments.

2.3 Discussion

- **ACO** emphasizes exploration guided by distributed pheromone trails, naturally balancing load across links.
- **GA** emphasizes exploitation of good solutions via crossover and mutation, allowing the search space to be covered efficiently.

- Both can be integrated into the same pipeline as our ILP and QAOA formulations, providing baselines and hybrid strategies for large-scale instances.

3 Extending the QUBO: Load-Balancing & Capacity Penalties

We now extend the QUBO formulation used for energy-aware routing to include (i) a load-balancing objective (variance of link utilizations) and (ii) explicit penalties for link capacity violations. The goal is to keep the final cost function quadratic in the binary decision variables so it can be solved (or approximated) by QAOA after conversion to an Ising Hamiltonian.

3.1 Notation (repeated for clarity)

- $G = (V, E)$: network graph with links $e \in E$.
- D : set of demands. For each demand $d \in D$ let b_d be its bandwidth and sla_d its latency bound.
- P_d : candidate path set for demand d. Index the binary decision variables as
$$x_{d,p} = \begin{cases} 1 & \text{if demand } d \text{ is routed on path } p \in P_d, \\ 0 & \text{otherwise.} \end{cases}$$
(We may also include a drop variable z_d per demand; same treatment applies.)
- For each link e we have capacity cap_e, latency lat_e, and energy-per-Mbps ϵ_e.
- Use a flattened index i for each variable $x_{d,p}$ when convenient (so $x_i \in \{0, 1\}$).
- Define the incidence indicator
$$a_{e,i} = \begin{cases} 1 & \text{if variable } i \text{ (path } p) \text{ traverses edge } e, \\ 0 & \text{otherwise.} \end{cases}$$
and let b_i be the bandwidth associated to variable i (i.e., $b_i = b_d$).

3.2 Link utilization and variance (mathematical derivation)

For each link e the (absolute) load is

$$U_e \;=\; \sum_i b_i \, a_{e,i} \, x_i.$$

The utilization fraction is

$$u_e \;=\; \frac{U_e}{\mathrm{cap}_e}.$$

We use a load-balancing objective that minimizes the variance of utilizations across links:

$$\mathrm{Var}(u) \;=\; \frac{1}{|E|}\sum_{e\in E}(u_e-\bar{u})^2, \qquad \bar{u} \;=\; \frac{1}{|E|}\sum_{e\in E} u_e.$$

Expand $\mathrm{Var}(u)$ to obtain an expression quadratic in x:

$$\begin{aligned}\mathrm{Var}(u) &= \frac{1}{|E|}\sum_e u_e^2 - \bar{u}^2 \\ &= \frac{1}{|E|}\sum_e\left(\frac{U_e^2}{\mathrm{cap}_e^2}\right) - \frac{1}{|E|^2}\left(\sum_e \frac{U_e}{\mathrm{cap}_e}\right)^2. \end{aligned} \tag{11}$$

Insert $U_e = \sum_i b_i a_{e,i} x_i$. Each quadratic term becomes sums over variable pairs i, j:

$$U_e^2 = \sum_i\sum_j b_i b_j a_{e,i} a_{e,j}\, x_i x_j.$$

Therefore the variance contributes pairwise quadratic coefficients to the QUBO. Define the shorthand

$$S_{e,ij} \;=\; \frac{b_i b_j\, a_{e,i} a_{e,j}}{\mathrm{cap}_e^2}, \qquad T_i \;=\; b_i \sum_e \frac{a_{e,i}}{\mathrm{cap}_e}.$$

Then from (11) we get the coefficient contribution for $x_i x_j$:

$$Q_{ij}^{(\mathrm{var})} \;=\; \frac{1}{|E|}\sum_e S_{e,ij} \;-\; \frac{1}{|E|^2}\, T_i T_j. \tag{12}$$

(For diagonal terms $i = j$ the same formula applies, producing the corresponding linear coefficient in the QUBO.)

Intuition: the first term rewards configurations where high bandwidth choices do not overlap on small-capacity links (it penalizes pairs that concentrate load on the same small link). The second term subtracts a global square of the mean utilization; together they measure spread (variance) and are quadratic in the x_i.

3.3 Capacity violation penalty

Link capacity constraints in the ILP are

$$U_e \le \mathrm{cap}_e, \qquad \forall e \in E.$$

In a QUBO we typically impose such constraints as soft penalties. Two practical choices:

(A) Simple quadratic penalty (approximate, easy to implement). Add for each edge e the penalty term

$$\lambda_{\text{cap}} \, (U_e - \text{cap}_e)^2.$$

Expanding gives quadratic contributions

$$(U_e - \text{cap}_e)^2 = U_e^2 - 2\text{cap}_e U_e + \text{cap}_e^2,$$

which is a quadratic polynomial in the x variables (since U_e is linear in x). This penalizes both overflow and underflow relative to cap_e, so choose λ_{cap} carefully or combine this with the variance objective (below) so under-utilization is not excessively punished.

(B) One-sided (precise) overflow penalty using binary slack variables. To penalize only overflow $\max(0, U_e - \text{cap}_e)$ we introduce binary expansion variables $\{y_{e,t}\}$ that encode the nonnegative overflow in binary:

$$\text{overflow}_e \approx \sum_{t=0}^{T-1} 2^t \delta \, y_{e,t}, \quad y_{e,t} \in \{0, 1\},$$

where δ sets the granularity and T is chosen to cover the expected overflow range. Enforce

$$U_e - \text{cap}_e - \sum_t 2^t \delta \, y_{e,t} \leq 0$$

via a penalty squared constraint in the QUBO:

$$\lambda_{\text{slack}} \left(U_e - \text{cap}_e - \sum_t 2^t \delta \, y_{e,t} \right)^2 .$$

Then penalize the binary slack magnitude by adding $\lambda_{\text{overflow}} \sum_t 2^t \delta \, y_{e,t}$ (or square it). This approach is more exact but increases the qubit count by T per edge.

3.4 Full QUBO objective

Collecting terms, the total QUBO cost (minimize) becomes

$$\begin{aligned}\mathcal{L}(x,z) \;=\; & \underbrace{\sum_i c_i\, x_i}_{\text{energy cost: linear}} \quad \underbrace{+\,\Gamma_{\text{drop}} \sum_d z_d}_{\text{drop penalties}} \\ & \underbrace{+\,\lambda_A \sum_d \left(\sum_{p \in P_d} x_{d,p} + z_d - 1 \right)^2}_{\text{one-hot (path or drop) constraint}} \\ & \underbrace{+\,\lambda_{\text{cap}} \sum_e (U_e - \text{cap}_e)^2}_{\text{(soft) capacity penalties}} \\ & + \underbrace{\lambda_{\text{var}}\, \text{Var}(u)}_{\text{load-balance (variance) term}} \\ & + \underbrace{\lambda_{\text{sla}} \sum_i \mathbb{I}(\text{lat}_i > \text{sla}_{d(i)})\, x_i}_{\text{strong SLA penalties for infeasible paths}} \quad .\end{aligned} \tag{13}$$

Here:

- c_i is the energy cost of routing demand d on path p (linear term),
- λ_A is the one-hot penalty (must be large enough to enforce feasibility),
- λ_{cap} and λ_{var} weigh capacity and variance respectively,
- Γ_{drop} is a large penalty to disincentivize dropping (unless necessary).

Every term in (13) expands into constant, linear, and quadratic contributions in the binary variables x_i and optional slack variables $y_{e,t}$. The variance term expansion is given explicitly by (12).

3.5 From QUBO to Ising for QAOA

Once we have the QUBO matrix Q (upper triangular) and offset constant C, we convert to an Ising Hamiltonian by the standard change of variables

$$x_i = \frac{1 - z_i}{2}, \qquad z_i \in \{+1, -1\}.$$

This substitution maps quadratic terms $x_i x_j$ into linear and quadratic Pauli-Z operator terms. After conversion we obtain the cost Hamiltonian

$$H_C = \sum_i \alpha_i Z_i + \sum_{i<j} \beta_{ij} Z_i Z_j + \text{const.}$$

which is then used in QAOA as the problem Hamiltonian.

3.6 Practical guidelines for penalties and scaling

- **Penalty magnitudes:** choose λ_A (one-hot) so that violating a feasibility constraint is always much worse than any small improvement in the objective function. A practical heuristic: pick $\lambda_A \geq 10\times$ (maximum possible linear energy cost).
- **Balance capacity vs variance terms:** λ_{cap} should be large enough to avoid capacity breaches in final solutions; λ_{var} controls the tradeoff between energy minimization and load balancing. Tune by small experiments on representative small instances.
- **Normalization:** because U_e and cap_e may have widely different scales across edges, it helps to pre-normalize bandwidths and capacities (e.g., divide all capacities by a common factor) so penalty coefficients are numerically stable.
- **Size management:** the number of binary variables is roughly $|D| \times k$ (demands $\times$ candidate paths). If this exceeds the available qubits, reduce k (candidate paths) or solve by decomposition (solve subproblems per region / hierarchical routing).

3.7 Integrating ACO / GA heuristics into the QUBO (optional)

Pheromone bias (ACO $\rightarrow$ QUBO). Ant colony pheromones $\tau_{d,p}$ (accumulated from prior classical ACO runs) can be injected as a linear bias favoring certain path variables:

$$c_i \leftarrow c_i - \kappa\, \tau_i,$$

so higher pheromone reduces the linear cost of selecting that path. This provides a hybrid classical–quantum workflow: run a few ACO iterations to get pheromone priors, then encode those priors into the QUBO linear coefficients before QAOA.

GA seeding / hybridization. Genetic Algorithm output can provide high-quality candidate bitstrings. Those candidates can be:

- used to initialize classical optimizers in the QAOA outer loop (initial parameter guesses); or
- used to validate and compare QAOA samples (empirical baseline).

3.8 Summary and remarks

- The variance term (12) is quadratic in binary variables and therefore *directly* addable to the QUBO; it explicitly promotes load balancing by penalizing high overlap of heavy demands on small links.

- Capacity violations can be handled approximately with $(U_e - \text{cap}_e)^2$ penalties, or exactly (one-sided) using binary slack variables (at the cost of more qubits).
- ACO/GA can be integrated as priors or seeders for QAOA-enabling hybrid heuristics that combine classical exploration and quantum refinement.
- In practice you should (i) implement and test the exact coefficient computation on small instances, (ii) tune λ weights, and (iii) reduce problem size (prune candidate paths) before attempting QAOA on hardware/simulators.