QAOA for Integer Programs: Lessons from an OR practitioner in the quantum realm

Lucas Parada†

Postdoctoral researcher in operations research
†Université Laval - CIRRELT
lupar105@ulaval.ca

Montreal, September 8, 2025

QAOA

- Hybrid quantum–classical algorithm: QAOA (Quantum Approximate Optimization Algorithm) uses a quantum circuit with tunable parameters, optimized by a classical routine, to approximate solutions to combinatorial optimization problems.
- It alternates between applying a problem Hamiltonian (encoding the cost function, typically a Quadratic Unconstrained Binary Optimization Problem or QUBO) and a mixer Hamiltonian (exploring the solution space), with depth (number of alternations) controlling accuracy.
- Applications and promise: Designed for NP-hard optimization tasks (e.g., Max-Cut, scheduling, routing), QAOA is one of the leading candidates for demonstrating quantum advantage on near-term devices.

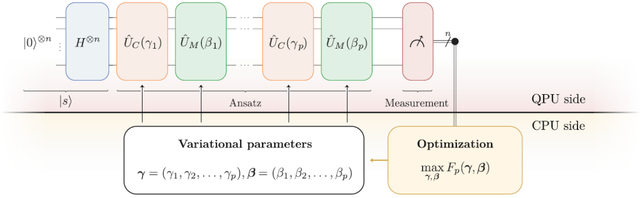


Figure: Schematics of QAOA (taken from Blekos et al., 2024)

Integer Program for an Assignment Problem

- Problem from Azad et al. (2022): "Solving Vehicle Routing Problem Using QAOA" (not a VRP! check 'Subtours' slides at the end).
- $G = (V, A)$ with $V = N \cup \{0\}$, $A = \{(i, j) : i \neq j\}$.
- Traveling on (i, j) costs c_{ij}.
- k identical vehicles, capacity $= 1$.
- $x_{ij} \in \{0, 1\}$ decision vars, $O(n^2)$ total.

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

Subject to the following constraints:

$$\sum_{j\in V\setminus\{i\}} x_{ij} = 1, \qquad \forall i \in N, \tag{2}$$

$$\sum_{j\in V\setminus\{i\}} x_{ji} = 1, \qquad \forall i \in N, \tag{3}$$

$$\sum_{j\in N} x_{0j} = k, \tag{4}$$

$$\sum_{j\in N} x_{j0} = k, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \qquad (i, j) \in A. \tag{6}$$

Integer vs. QUBO Formulation

Integer Program

- $G = (V, A)$ directed graph, $V = N \cup \{0\}$.
- Arc costs c_{ij}, decision vars $x_{ij} \in \{0, 1\}$.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\sum_{j\in V\setminus\{i\}} x_{ij} = 1, \qquad \forall i \in N \tag{2}$$

$$\sum_{j\in V\setminus\{i\}} x_{ji} = 1, \qquad \forall i \in N \tag{3}$$

$$\sum_{j\in N} x_{0j} = k \tag{4}$$

$$\sum_{j\in N} x_{j0} = k \tag{5}$$

$$x_{ij} \in \{0, 1\}, \qquad (i, j) \in A \tag{6}$$

QUBO Formulation

- Quadratic unconstrained binary optimization.
- Constraints absorbed as penalty terms.

$$H(x) = \sum_{(i,j)\in A} c_{ij}x_{ij} + M\Bigg[\sum_{i\in N}\Big(\sum_{j\in V\setminus\{i\}} x_{ij} - 1\Big)^2 + \sum_{i\in N}\Big(\sum_{j\in V\setminus\{i\}} x_{ji} - 1\Big)^2 + \Big(\sum_{j\in N} x_{0j} - k\Big)^2 + \Big(\sum_{j\in N} x_{j0} - k\Big)^2\Bigg] \tag{1}$$

- $M \gg \max c_{ij}$ ensures feasibility.

QUBO vs. Ising Formulation

QUBO Formulation

$$H(x) = \sum_{(i,j)\in A} c_{ij}x_{ij} + M\Bigg[\sum_{i\in N}\Big(\sum_{j\in V\setminus\{i\}} x_{ij} - 1\Big)^2 + \sum_{i\in N}\Big(\sum_{j\in V\setminus\{i\}} x_{ji} - 1\Big)^2 + \Big(\sum_{j\in N} x_{0j} - k\Big)^2 + \Big(\sum_{j\in N} x_{j0} - k\Big)^2\Bigg]$$

Ising Formulation

- Replace $x_{ij} = \frac{1}{2}(1 + z_{ij})$ with spins $z_{ij} \in \{-1, +1\}$.
- $O(n^2)$ **qbits.**
- Expand to standard Ising Hamiltonian:

$$H(z) = \sum_{(i,j)} h_{ij}\, z_{ij} + \sum_{(i,j)<(k,l)} J_{(ij)(kl)}\, z_{ij} z_{kl} + \text{const.} \tag{2}$$

- h_{ij}: local fields (linear terms from costs and penalties).
- $J_{(ij)(kl)}$: couplings (quadratic terms from penalties).
- Constant offset does not affect optimization.

Practical Insights

- There are substantial linear algebra operations involved from the base model, to the QUBO formulation, to the Ising Hamiltonian → mistakes/bugs which are tough to find/debug in the QAOA.
- Use the commercial solvers' built-in methods to print and solve the QUBO **before** going into PennyLane.

```
//In Cplex
IloCplex cplex(model); // model is an instantce of the
    IloModel class
cplex.exportModel("qubo.lp");
//In Gurobi
GRBModel model = GRBModel(env); // env is an instance of the
     GRBEnv class
model.write("qubo.lp");
```

Practical Insights

```
$ git clone https://github.com/lucasparada20/QAOA-for-the-VRP.git && cd QAOA-for
    -the-VRP
# Edit the Makefile to set the absolute path to you Cplex installed library in
    CPLEX_DIR = /some/path/to/Cplex
$ make
$ ./vrp_qubo # Solves the QUBO and generates the qubo.lp file
```

Minimize

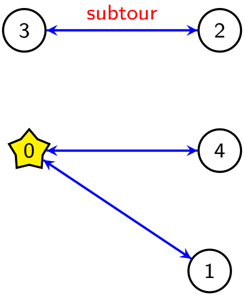
$$
\begin{aligned}
&- 1912.688x_{2,1} - 1952.738x_{3,1} - 1960.523x_{4,1} - 1912.688x_{1,2} \\
&- 1990.289x_{3,2} - 1957.113x_{4,2} - 1952.738x_{1,3} - 1990.289x_{2,3} \\
&- 1959.020x_{4,3} - 1960.523x_{1,4} - 1957.113x_{2,4} - 1959.020x_{3,4} \\
&- 3993.206x_{0,1} - 3938.347x_{0,2} - 3975.443x_{0,3} - 3952.233x_{0,4} \\
&- 3993.206x_{1,0} - 3938.347x_{2,0} - 3975.443x_{3,0} - 3952.233x_{4,0} \\
&+ \frac{1}{2}\left[4000x_{2,1}x_{3,1} + 4000x_{2,1}x_{4,1} + \cdots\right] \\
&+ 16000
\end{aligned}
$$

Bounds:

$$0 \leq x_{2,1} \leq 1, \quad 0 \leq x_{3,1} \leq 1, \quad \ldots \quad 0 \leq x_{4,0} \leq 1$$

Practical Insights

```
lucasparada20@DESKTOP-80I
work$ git clone https://github.com/lucasparada20/QAOA-for-the-VRP.git &&
 cd QAOA-for-the-VRP
Cloning into 'QAOA-for-the-VRP'...
remote: Enumerating objects: 99, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 99 (delta 29), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (99/99), 38.11 KiB | 975.00 KiB/s, done.
Resolving deltas: 100% (29/29), done.
QAOA-for-the-VRP$ nano Makefile
QAOA-for-the-VRP$ make
g++ -O3 -I/home/lucasparada20/CPLEX_Studio2211/cplex/include -I/home/luc
asparada20/CPLEX_Studio2211/concert/include main.cpp -L/home/lucasparada
20/CPLEX_Studio2211/cplex/lib/x86-64_linux/static_pic -L/home/lucasparad
a20/CPLEX_Studio2211/concert/lib/x86-64_linux/static_pic -lilocplex -lcp
lex -lconcert -lm -lpthread -o vrp_qubo
QAOA-for-the-VRP$ ./vrp_qubo
Default row names c1, c2 ... being created.
Status: Optimal
QUBO objective: 128.54
Arcs with x=1:
  0 -> 1
  0 -> 4
  1 -> 0
  2 -> 3
  3 -> 2
  4 -> 0
Route 1: 0 -> 1 -> 0
Route 2: 0 -> 4 -> 0
QAOA-for-the-VRP$
```



The solution $x = \{(0,1),(0,4),(2,3)\}$ with $k = 2,\ N = \{1,2,3,4\}$ is optimal with cost 128.54. Same solution from Azad et al. (2022).

QAOA Circuit

- **Initialization:** Each qubit is prepared in the uniform superposition state $|+\rangle$ with Hadamard gates (**lines 11-12**), so all bitstrings are equally likely at the start.
- **Alternating operators:** Each layer applies (i) a cost Hamiltonian evolution $e^{-i\gamma H}$, encoding the objective function (line 3), and (ii) RX mixer rotations $e^{-i\beta X}$ (**lines 4-5**), which explore the solution space.
- **Layer repetition:** The circuit stacks p such layers with parameters $\{\gamma_1, \ldots, \gamma_p\}$ and $\{\beta_1, \ldots, \beta_p\}$ (**lines 13-14**); tuning these values balances problem encoding with exploration.

```
1  def qaoa_layer(self, gamma, beta, H):
2      """One QAOA layer."""
3      qml.templates.ApproxTimeEvolution(H,
           gamma, 1)
4      for i in range(len(H.wires)):
5          qml.RX(2 * beta, wires=i)
6
7  def qaoa_circuit(self, params, H, n_wires):
8      """Full QAOA circuit with p layers."""
9      p = len(params) // 2
10     gammas, betas = params[:p], params[p:]
11     for i in range(n_wires):
12         qml.Hadamard(wires=i)  # Initial |+>
               state
13     for gamma, beta in zip(gammas, betas):
14         self.qaoa_layer(gamma, beta, H)
```

Intuition

The cost unitary “evaluates” solutions while the mixer reshuffles amplitudes, gradually steering the quantum state toward better solutions.

QAOA Optimization Loop - Part I

- **Quantum nodes (QNodes):** Define circuits to sample bitstrings from the QAOA circuit (**lines 4-7**) and evaluate the expectation value of the cost Hamiltonian (**lines 9-15**), providing the energy used for optimization.
- **Cost function:** The `cost_fn_verbose` (**lines 17-23**)computes the energy, counts iterations, and prints cost and parameters, serving as the objective for the optimizer.

```
1  dev = qml.device("default.qubit",
2                   wires=n_wires, shots=
                           n_shots)
3
4  @qml.qnode(dev)
5  def qnode(params):
6      self.qaoa_circuit(params, H, n_wires)
7      return qml.sample()
8
9  @qml.qnode(dev)
10 def raw_energy_qnode(params):
11     self.qaoa_circuit(params, H, n_wires)
12     return qml.expval(H)
13
14 def energy_qnode(params):
15     return raw_energy_qnode(params)
16
17 # Verbose cost fn
18 iteration_counter = {'count': 0}
19 def cost_fn_verbose(params):
20     energy = energy_qnode(params)
21     iteration_counter['count'] += 1
22     print(f"[Iter {iteration_counter['count
           ']:03d}] Cost: {energy:.6f},
           Params: {np.round(params, 4)}")
23     return energy
```

QAOA Optimization Loop - Part II

- **Multiple restarts:** The optimizer is run several times from different random initial parameters to avoid local minima.
- **Best parameters:** After all restarts, the parameter set with the lowest energy value is stored as the optimal solution.
- **Summary and sampling:** The optimal parameters, corresponding minimum cost, and constant term are printed, and the circuit is finally sampled to generate bitstrings.

```
best_val, best_params = float("inf"),None
for seed in range(4):  # restarts
    init_params = np.random.uniform(0, np.pi, size=(2 * p,))
    print(f"\n--- Restart {seed+1} ---")
    res = minimize(cost_fn_verbose, init_params, method="COBYLA", options={"maxiter": 5000})
    if res.fun < best_val:
        best_val, best_params = res.fun, res.x

print("\n========= Optimization Summary =========")
print("Optimal QAOA parameters:", best_params)
print("Minimum cost:", best_val)
print("Constant:", const_term)

samples = qnode(best_params)
```

Results

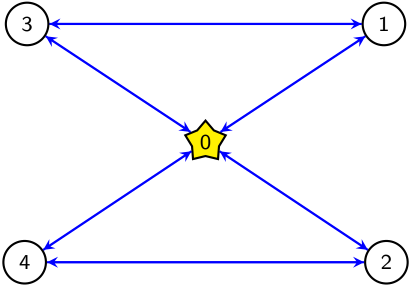


Figure: Simulation solution with $p = 6$ and 1 restart.
QUBO cost = 42,407.7 due depot constraint violations.
Should be incorrect for the problem!
Issue stems from an incorrect formulation of depot connectivity constraints (conflicting constraints in the paper) (Toth and Vigo, 2014)

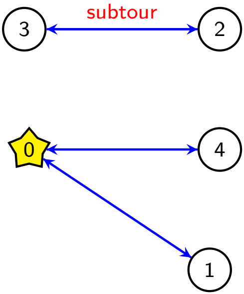


Figure: The optimal solution $x = \{(0,1),(0,4),(2,3)\}$ for the 2nd instance in the paper with $k = 2$, $N = \{1,2,3,4\}$.
Qubo cost = 4,128.5

Conclusion

- QAOA used in for optimization problems that have a "natural" QUBO formulation (i.e. Max-Cut)
- Lots of opportunities for other NP-Complete problems without natural QUBO formulations: Integer and continuous variables (not just binary). VRP is one example.
- Extensive experimentation required for parameter tuning (nb layers and penalties).
- Error correction? Did not get to launch in Yukon (no QPU used), but first need to correct modeling errors.
- Scaling? First, we need to correct the math behind the QUBO models. Then, the CPU side can work using SLURM (?) in HPC and multithreading.
- Motivation? **Keep a pulse on the technology.** OR practitioners do not want to wake up 10 years later to find that everything is computed on QPUs!

MANY THANKS!
And thanks to the unwavering support of the team at Calcul Québec (Juliette, Kim, Lydia, and everybody else)!

MANY THANKS!
And thanks to the unwavering support of the team at Calcul Québec (Juliette, Kim, Lydia, and everybody else)!

References I

Azad, U., Behera, B. K., Ahmed, E. A., Panigrahi, P. K., and Farouk, A. (2022). Solving vehicle routing problem using quantum approximate optimization algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):7564–7573.

Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., and Summer, A. (2024). A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

VRP Model for Unoriented Graphs

- Since Q in QUBO $\rightarrow$ Ising Hamiltonian is required to be Hermitian, such that the eigenvalues are real, it's nice to work on the symmetric VRP.
- From Toth and Vigo (2014) the symmetric model follows:

$$
\begin{aligned}
& \min \ c^\top x \\
\text{s.t.} \quad & x(\delta(i)) = 2, && \forall i \in N, \ \forall j \in N, \\
& x(\delta(0)) = 2|K|, && \\
& x(\delta(S)) \geq 2r(S), && \forall S \subseteq N, \ S \neq \emptyset, \\
& x_e \in \{0, 1, 2\}, && \forall e \in \delta(0), \\
& x_e \in \{0, 1\}, && \forall e \in E \setminus \delta(0).
\end{aligned}
$$

where:

- for a set $S \subseteq N$, $\delta(S) = \{e = (i, j) \in E : i \in S, j \notin S\}$ is the cutset induced by S,
- $r(S)$ denotes the minimum number of routes that must cross the cut defined by S.
- K is the fleet size.

Subtours

- The model in (1)–(5) is not a VRP or TSP due to the absence of subtour elimination constraints.

- We can use the Miller–Tucker–Zemlin (MTZ) formulation to forbid subtours (SEC) by introducing $O(n^2)$ variables $u_i \in i \in N$ (unoriented case needed for the paper):

$$u_i - u_j + Qx_{ij} \leq Q - q_j \quad \forall i,j \in N,\ i \neq j \tag{6a}$$

$$u_j - u_i + Qx_{ij} \leq Q - q_i \quad \forall i,j \in N,\ i \neq j \tag{6b}$$

$$q_i \leq u_i \leq Q \quad \forall i \in N \tag{7}$$

For edge $\{2,3\}$ with $Q = |N| = 4, q_2 = q_3 = 1, x_{23} = 1$:

$$u_2 \leq u_3 - 1, \quad u_3 \leq u_2 - 1,$$

a contradiction. Hence the subtour $(2,3)$ is forbidden.

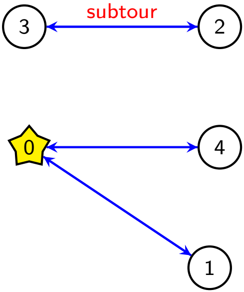


Figure: The optimal solution $x = \{(0,1),(0,4),(2,3)\}$ for the 2nd instance in the paper with $k = 2$, $N = \{1,2,3,4\}$.

Feasible Solution with MTZ Check

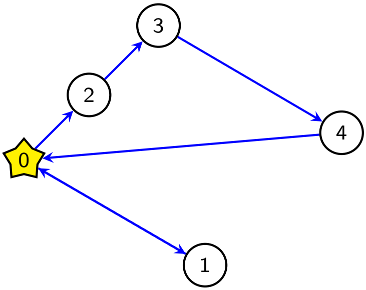


Figure: Solution with $k = 2$: $\{(0,1),(0,2),(0,4),(2,3),(3,4)\}$ in orientation $0 \to 1 \to 0$ and $0 \to 2 \to 3 \to 4 \to 0$.

MTZ constraints with $Q = |N| = 4$, $q_i = 1$:
Assignments:

$$u_1 = 1,\ u_2 = 1,\ u_3 = 2,\ u_4 = 3$$

Capacity bounds:

$$1 \leq u_i \leq 4 \quad \forall i \in N \ \checkmark$$

Active edges:

$$(2,3) : u_2 - u_3 + 4 \leq 3 \ \Rightarrow \ 1 - 2 + 4 = 3 \leq 3 \checkmark$$
$$(3,4) : u_3 - u_4 + 4 \leq 3 \ \Rightarrow \ 2 - 3 + 4 = 3 \leq 3 \checkmark$$

Inactive edge $(2,4)$:

$$u_2 - u_4 = -2 \leq 3, \quad u_4 - u_2 = 2 \leq 3 \checkmark$$

All MTZ constraints are satisfied.