VRP via Quantum Optimization

Anthony Palmieri
ap.anthonypalmieri@gmail.com

University of Padua

November 27, 2023

Vehicle Routing Problem

- The Vehicle Routing Problem (VRP) is a classic combinatorial optimization problem that involves finding the optimal set of routes for a fleet of vehicles to serve a set of customers.
- The goal is to minimize the total distance traveled by the vehicles while ensuring that each customer is served by exactly one vehicle and that the capacity of each vehicle is not exceeded.
- The simplest formulation is known as CVRP (Capacitated Vehicle Routing Problem) though many other formulations are possible, with increasing levels of complexity.

A good resource about VRP is [1].

(Meta)Heuristics

Due to its complexity, exact methods for solving the VRP may not be feasible for large problem instances. Heuristics and Metaheuristics methods provide a practical alternative for finding good solutions to the VRP.
A comprehensive review of the most notable and powerful (meta)heuristics can be found in [2]. The analyzed algorithms include:

- Simulated Annealing
- Tabu Search
- Variable Neighborhood Search
- Large Neighborhood Search
- Iterated Local Search
- GRASP
- Genetic Algorithms
- Memetic Algorithms
- Ant Colony Optimization
- Particle Swarm Optimization

An overview of such methods applied to VRP can be found in [3].

QUBO and Ising formulations

QUBO (Quadratic Unconstrained Binary Optimization) is a mathematical formulation used to solve combinatorial optimization problems. The general form of a QUBO problem is

$$\min_{x \in \{0,1\}^n} \quad x^t Q x + c$$

where Q can be assumed to be a symmetric matrix.
With a change of variable $x = \frac{1-s}{2}$ we get the *Ising Energy*

$$\mathcal{E}(s_1, \cdots, s_n) = -\sum_{ij} J_{ij} s_i s_j - \sum_{1}^{n} h_i s_i, \quad s_i \in \{-1, +1\} \, \forall i.$$

Substituting each s_i with the related Pauli-Z operator

$$\sigma_i^z = I^{\otimes i-1} \otimes Z \otimes I^{\otimes n-i}$$

we get the "problem" *Ising Hamiltonian* H. More details in [4, 5].

Ising Hamiltonian

Question: if we wish to minimize the Ising Energy function E, then why do we consider the Ising Hamiltonian H (which is a square matrix of order 2^n)? We have

$$
\begin{aligned}
H|z\rangle &= (-1)\left(\sum_{i\in V(G)} h_i\sigma_i^z + \sum_{ij\in E(G)} J_{ij}\sigma_i^z\sigma_j^z\right)|z_1\rangle\otimes\cdots\otimes|z_n\rangle \\
&= -\sum_{i\in V(G)} h_i|z_1\rangle\otimes\cdots\otimes(-1)^{z_i}|z_i\rangle\otimes\cdots\otimes|z_n\rangle + \\
&- \sum_{ij\in E(G)} J_{ij}|z_1\rangle\otimes\cdots\otimes(-1)^{z_i}|z_i\rangle\otimes\cdots\otimes(-1)^{z_j}|z_j\rangle\otimes\cdots\otimes|z_n\rangle \\
&= -\left(\sum_{i\in V(G)} h_i(-1)^{z_i} + \sum_{ij\in E(G)} J_{ij}(-1)^{z_i}(-1)^{z_j}\right)|z_1\rangle\otimes\cdots\otimes|z_n\rangle
\end{aligned}
$$

Ising Hamiltonian

Therefore, we have

$$H|z\rangle = -\left(\sum_{i\in V(G)} h_i(-1)^{z_i} + \sum_{ij\in E(G)} J_{ij}(-1)^{z_i}(-1)^{z_j}\right)|z\rangle$$

where $z_i \in \{0,1\}$ for $i = 1, \cdots, n$.
Defining $s := (s_1, s_2, \ldots, s_n)$ with $s_i := (-1)^{z_i} \in \{-1, +1\}$ we find

$$H|z\rangle = \mathcal{E}(s)|z\rangle.$$

In the following, to ease the notation we use $s = s(z) = (-1)^z$.
Thus, there is a correspondence between the energy function $\mathcal{E}$ and the eigen-structure of H. In particular

- $\min \mathcal{E}$ = smallest eigenvalue of H;
- $\arg\min \mathcal{E}$ = groundstate of H.

Notice also that H is diagonal in the computational basis.

Ising Hamiltonian: an example

Consider the problem

$$\begin{aligned} &\min\ f(x) = x_1 x_2 \\ &\text{s.t.}\ \ x_1, x_2 \in \{0, 1\}, \end{aligned}$$

we have

- $\min f = 0$;
- $\arg\min \mathcal{E} = (0,0), (0,1), (1,0)$.

With the change of variables $x_i = \frac{1-s_i}{2}$, $s_i \in \{-1, +1\}$ we get

$$\min \mathcal{E} = \frac{1}{4}(1 - s_1 - s_2 + s_1 s_2) \sim \min \mathcal{E} = -s_1 - s_2 + s_1 s_2.$$

Clearly:

- $\min \mathcal{E} = -1$;
- $\arg\min \mathcal{E} = (1,1), (1,-1), (-1,1)$.

Ising Hamiltonian: an example

Substituting s_i with $\sigma_i^z = I^{\otimes i-1} \otimes Z \otimes I^{\otimes 2-i}$ we get

$$H = -(Z \otimes I) - (I \otimes Z) + (Z \otimes Z) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

and we have

- $\lambda_{min}(H) = -1$;
- $groundstates(H) = \{e_1 = |00\rangle, e_2 = |01\rangle, e_3 = |10\rangle\}$,

which are exactly the solutions we were looking for. Recall that

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

How to QUBO

Consider the Integer Program

$$\begin{aligned} \min\ & f(x) \\ \text{s.t.}\ & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in \mathbf{Z}^n, \end{aligned}$$

in order to convert it to QUBO we have to

1. transform inequalities into equalities by introducing integer slack variables;
2. binarize the variables (using for instance binary encoding, unary encoding or bounded encoding);
3. quadratize the objective and the constraints;
4. remove the constraints and add them as penalties to the objective.

More on Quadratization

Theorem

Any pseudo-Boolean function defined as $f : \{0,1\}^n \to \mathbf{R}$ *can be written uniquely as a sum of multi-linear functions.*

$$f(x) = a_0 + \sum_i a_i x_i + \sum_{ij} a_{ij} x_i x_j + \sum_{ijk} a_{ijk} x_i x_j x_k + \dots$$

We then define new variables $x_{ij} = x_i x_j$ for all terms of order greater than 2 in the above function, continuously until we are left with just quadratic terms. By using this procedure any such objective function can be converted to a quadratic polynomial. To account for $x_{ij} = x_i x_j$ we need however to introduce new constraints. We define

$$H(x) = 3x_{ij} + x_i x_j - 2x_{ji} x_i - 2x_{ij} x_j$$

and we have that $x_{ij} = x_i x_j$ iff $H(x) = 0$.

More on Unconstraining

Consider the combinatorial problem

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \in \{0,1\}^n, \end{aligned}$$

After unconstraining, the total energy function of the QUBO can be written as

$$H(x) = c^T x + \rho \sum_i (\sum_j A_{ij} x_j - b_i)^2$$

the first term being the problem objective while the second accounting for the unconstraining. How to choose ρ? Guarantee that the minimal change in infeasibility is larger than the maximal case in objective.

$$\rho \geq \frac{\Delta H_B^{max}}{\Delta H_A^{min}} = \frac{\sum_i \max\{c_i, 0\}}{\min_{\sigma_i \in \{0,1\}, j} (\max[1, \frac{1}{2} \sum_i (-1)^{\sigma_i} A_{ij}])}$$

Intro to Quantum Computing

We begin our journey into the Quantum Computing realm via the lectures notes proposed by Ronald de Wolf [6]. The chapters we mostly focused on are:

- Quantum Computing
- The Circuit Model and the Deutsch-Jozsa Algorithm
- Simon's Algorithm
- The Fourier Transform
- Grover's Search Algorithm
- Quantum Walk Algorithms
- Hamiltonian Simulation
- The HHL Algorithm
- Quantum Query Lower Bounds
- Quantum Machine Learning

Variational Quantum Algorithms

We finally move on to Quantum Optimization, starting with a general overview on Variational Quantum Algorithms (VQAs) [7].

- Variational Quantum Algorithms (VQAs) have emerged as the leading strategy to obtain quantum advantage on NISQ devices.
- VQAs leverage the toolbox of classical optimization: they use parameterized quantum circuits (ansatzes) to be run on the quantum computer, and then outsource the parameter optimization to a classical optimizer.
- The two most notable VAQs are: Variational Quantum Eigensolver (VQE)[8] and Quantum Approximate Optimization Algorithm (QAOA)[9].

VQAs step by step

- Define a quantum circuit $U(\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ that can prepare an initial state $|\psi(\boldsymbol{\theta})\rangle$.
- Choose a cost function $C(\boldsymbol{\theta})$ that measures the performance of the quantum circuit.
  In our case: $C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H_p(z_1, \cdots, z_N)|\psi(\boldsymbol{\theta})\rangle$.
- Choose a classical optimizer that will minimize the cost function by adjusting the parameters $\boldsymbol{\theta}$.
- Initialize the parameters $\boldsymbol{\theta}$ with some initial guess, and set a convergence criterion for the optimizer.
- Run the VQA by repeatedly applying the quantum circuit $U(\boldsymbol{\theta})$ to prepare the state $|\psi(\boldsymbol{\theta})\rangle$, measuring the state in some basis, and using the measurement outcomes to compute the cost function $C(\boldsymbol{\theta})$. Then, update the parameters $\boldsymbol{\theta}$ using the classical optimizer. Repeat this process until the convergence criterion is met.

Classically Simulating VQAs

- The loss $C(\theta)$ at each iteration is an actual mean value due to $|\psi(\theta)\rangle$ being a superposition of states

$$|\psi(\theta)\rangle = \sum_{x\in\{0,1\}^n} \alpha_x(\theta)|x\rangle, \quad \sum_{x\in\{0,1\}^n} |\alpha_x(\theta)|^2 = 1$$

- Since H_p is diagonal in the (orthonormal) computational basis

$$C(\theta) = \sum_{x\in\{0,1\}^n} |\alpha_x(\theta)|^2 \langle x|H_p|x\rangle = \sum_{x\in\{0,1\}^n} |\alpha_x(\theta)|^2 \mathcal{E}((-1)^x)$$

- Thus, classical simulation of VAQs requires estimating such mean value, typically through Monte Carlo samplings, which can make the simulation inefficient.
- The number of samples needed to estimate the mean value accurately can be generally high.
- An approach is to keep sampling until the unbiased sample variance falls below a given threshold ϵ.

VQE

- VQE is a VQA used for solving optimization problems and calculating the ground state energy of quantum systems.
- VQE is particularly well-suited for simulating molecular systems, accurately predicting electronic structures and properties.
- Many practical applications in chemistry and materials science.
- Usually, VQE employs a parameterized ansatz known as the Unitary Coupled Cluster (UCC) ansatz.
- Several variations exist.

QAOA

- The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm designed to solve combinatorial optimization problems.
- QAOA works by encoding the optimization problem as a Hamiltonian and using a sequence of unitary operations to prepare a quantum state that can be measured to obtain the optimal solution.
- In QAOA we alternate the use of two different Hamiltonians, the “mixing” Hamiltonian and the “problem” Hamiltonian.
- The first one is used to prepare a superposition of all possible solutions to the optimization problem; the second one is used to rotate the state towards the optimal solution.

QAOA ansatz

The QAOA circuit ansatz typically consists of alternating layers of two types of unitary operations, parameterized by the angles γ and β respectively. Here is an example of a QAOA ansatz with p layers:

$$|\gamma, \beta\rangle = U_{\beta_p} U_{\gamma_p} \cdots U_{\beta_1} U_{\gamma_1} |+\rangle^{\otimes n}$$

where

$$U_{\beta_j} = e^{-i\beta_j \sum_1^n \sigma_j^x}, \qquad U_{\gamma_j} = e^{-i\gamma_j H_p}, \qquad |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

The vectors γ and β are p-dimensional vectors of parameters that need to be optimized to obtain the best approximation of the optimal solution.
Interestingly, [10] proved(??) that the choice of such ansatz is optimal when the angles γ, β vary in a fixed range.

QAOA: towards a geometric interpretation

To ease the notation, in the following we use

$$\sigma_i^x = X_i = I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i},$$

where X denotes the Pauli-X matrix.
Let's consider the mixing Hamiltonian in QAOA, we have:

$$\begin{aligned} e^{-i\beta H_M} &= e^{-i\beta \sum_i X_i} = e^{-i\beta \sum_i I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i}} \\ &\overset{(1)}{=} \prod_i e^{-i\beta (I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i})} \\ &\overset{(2)}{=} \prod_i I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i} \end{aligned}$$

In the following slides we try to explain the reasons behind $(1), (2)$.
We first need to introduce matrix exponentiation.

Matrix Exponentiation

Given a square matrix A, the matrix exponentiation of A is computed using the Taylor series expansion of the exponential function:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = I + A + \frac{1}{2}A^2 + \cdots$$

notice that if A is diagonalizable: $A = UDU^{-1}$ then we have:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \sum_{k=0}^{\infty} \frac{(UDU^{-1})^k}{k!} = \sum_{k=0}^{\infty} \frac{UD^kU^{-1}}{k!} = U\left(\sum_{k=0}^{\infty} \frac{D^k}{k!}\right)U^{-1}.$$

Be aware: matrix exponentiation doesn't follow the classical rules of exponentiation!

Matrix Exponentiation

Given two matrices A, B the followings are equivalent:

- $e^{A+B} = e^A e^B$;
- A, B commute;
- A, B are simultaneously diagonalizable;
- there exists U change of basis such that

$$A = UD_A U^{-1}, \quad B = UD_B U^{-1}$$

  with D_A, D_B diagonal matrices.

We can now finally understand the reason behind (1).

(1) explained

The Pauli-X matrix is unitarily diagonalible, that is $\exists U$ unitary and D diagonal such that $X = UDU^{-1}$.
Since for each identity matrix we have $I = UIU^{-1}$ we can easily change the basis on each of the vector spaces in the tensor product so to have

$$I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i} \to I^{\otimes i-1} \otimes D \otimes I^{\otimes n-i} \quad \forall i.$$

Since all the matrices in the sum are simultaneously diagonalizable, it holds:

$$e^{-i\beta \sum_i X_i} = e^{-i\beta \sum_i I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i}} = \prod_i e^{-i\beta (I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i})}$$

which justifies (1).

(2) explained

$$
\begin{aligned}
e^{\alpha I \otimes A \otimes I} &= \sum_k \frac{(\alpha I \otimes A \otimes I)^k}{k!} \\
&= I \otimes I_A \otimes I + \alpha(I \otimes A \otimes I) + \frac{1}{2}\alpha^2 (I \otimes A \otimes I)^2 + \cdots \\
&= I \otimes I_A \otimes I + \alpha(I \otimes A \otimes I) + \frac{1}{2}\alpha^2 (I \otimes A \otimes I)(I \otimes A \otimes I) + \cdots \\
&\overset{(2.1)}{=} I \otimes I_A \otimes I + \alpha(I \otimes A \otimes I) + \frac{1}{2}\alpha^2 (II \otimes AA \otimes II) + \cdots \\
&= I \otimes I_A \otimes I + \alpha(I \otimes A \otimes I) + \frac{1}{2}\alpha^2 (I \otimes A^2 \otimes I) + \cdots \\
&\overset{(2.2)}{=} I \otimes I_A \otimes I + (I \otimes \alpha A \otimes I) + (I \otimes \frac{1}{2}\alpha^2 A^2 \otimes I) + \cdots \\
&= I \otimes \sum_k \frac{(\alpha A)^k}{k!} \otimes I \\
&= I \otimes e^{\alpha A} \otimes I
\end{aligned}
$$

(2) explained

In the previous derivation we used the following two properties of the tensor product:

2.1 $(A \otimes B)(C \otimes D) = AC \otimes BD$;

2.2 $\alpha(A \otimes B) = \alpha A \otimes B = A \otimes \alpha B$ for every $\alpha \in \mathbf{C}$.

Generalizing the previous calculations we finally prove

$$e^{-i\beta(I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i})} = I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i} \quad \forall i.$$

To conclude, we proved that:

$$e^{-i\beta \sum_i X_i} = e^{-i\beta \sum_i I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i}} = \prod_i e^{-i\beta(I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i})}$$

$$= \prod_i I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i}.$$

QAOA mixing operator

At this point we have:

$$
\begin{aligned}
& I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i} |\psi_1 \cdots \psi_n\rangle \\
&= (I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i})(|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle) \\
&= I|\psi_1\rangle \otimes \cdots \otimes I|\psi_{i-1}\rangle \otimes e^{-i\beta X}|\psi_i\rangle \otimes I|\psi_{i+1}\rangle \otimes \cdots \otimes I|\psi_n\rangle \\
&= |\psi_1\rangle \otimes \cdots \otimes |\psi_{i-1}\rangle \otimes e^{-i\beta X}|\psi_i\rangle \otimes |\psi_{i+1}\rangle \otimes \cdots \otimes |\psi_n\rangle.
\end{aligned}
$$

We find that the transformation leaves all the qubits unchanged but for the i-th one, which is transformed via $|\psi_i\rangle \to e^{-i\beta X}|\psi_i\rangle$. Now the question is, what is such transformation actually doing?

QAOA mixing operator

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{-1}$$
$$= U \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot U^{-1}$$

where in this case $U^{-1} = U^T = U$. At this point

$$e^{-i\beta X} = U \cdot \begin{bmatrix} e^{-i\beta} & 0 \\ 0 & e^{i\beta} \end{bmatrix} \cdot U^{-1}.$$

We then have:

$$e^{-i\beta X}|0\rangle = e^{-i\beta X} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \cdots = \cos\beta|0\rangle - i\sin\beta|1\rangle$$

$$e^{-i\beta X}|1\rangle = e^{-i\beta X} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \cdots = -i\sin\beta|0\rangle + \cos\beta|1\rangle$$

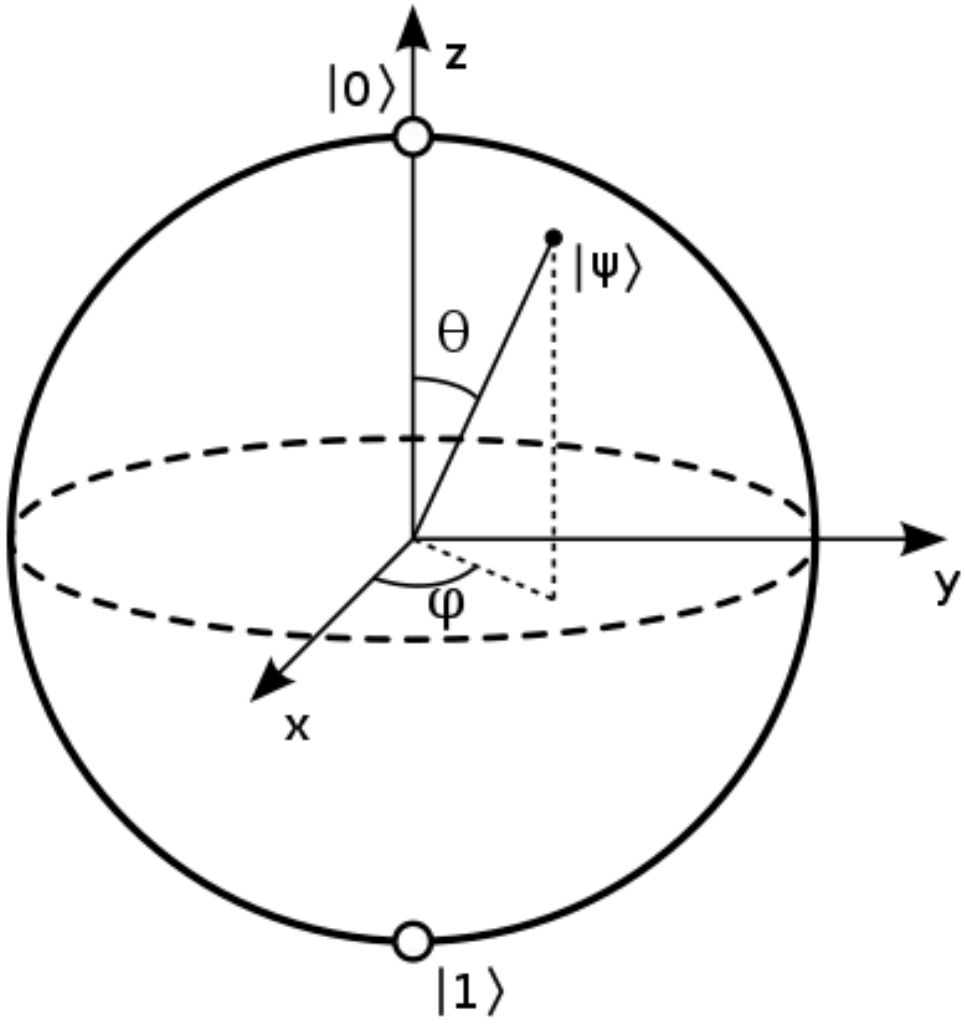
Bloch Sphere

To undertsand the geometric interpretation of QAOA we first need to define the Bloch sphere. Recall that a qubit is a superposition of the two basis states $|0\rangle$ and $|1\rangle$, namely:

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1 \\ &= \cos\theta|0\rangle + \sin\theta|1\rangle \\ &= \cos\theta|0\rangle + e^{i\phi}\sin\theta|1\rangle, \end{aligned}$$

The parameters ϕ and θ uniquely specify a point on the unit sphere of euclidean space $\mathbf{R}^3$, which is usually called the Bloch sphere.

Bloch Sphere



Bloch sphere

QAOA geometric interpretation

At this point the geometric interpretation follows easily:

- When applied on the state $|\psi\rangle$, the driver Hamiltonian applies a qubit-wise rotation to the state, rotating each qubit inside its own Bloch Sphere.
- This follows from the factorization

$$e^{-i\beta\sum_i X_i} = \prod_i I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i}.$$

- The i-th term in the product leaves all the qubits in $|\psi\rangle$ unchanged but for the i-th one, on which a rotation is applied.
- Applying all the terms in the product performs a collective rotation of the entire state.

QAOA geometric interpretation

- Such rotations allow the algorithm to explore the configuration space, looking for optimal solutions.
- The goal is to find the right angles so to rotate the initial state towards the optimal solution of the optimization problem.
- Similarly to H_M, also H_p can be thought of as a "rotation operator", where the rotation is guided by the underlying problem knowledge enclosed in the problem Hamiltonian.
- Since H_p is diagonal in the computational basis, we have:

$$e^{-i\gamma H_p} \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle = \sum_{x\in\{0,1\}^n} e^{-i\gamma\lambda_x} \alpha_x |x\rangle$$

where λ_x is the eigenvalue of H_p corresponding to the basis state $|x\rangle$, namely $H_p|x\rangle = \lambda_x|x\rangle$.

QAOA mixing operator

Let's now also take a look at some QAOA variants. Specifically, what if we consider a different mixing Hamiltonian?

- The choice of the transverse field mixing Hamiltonian in QAOA has been a popular and convenient approach. However, it is not the only option available.
- In certain scenarios, it can be advantageous to design custom mixers tailored to the problem at hand, especially for constrained optimization problems.
- Custom mixers are designed with two objectives in mind: preserving feasibility and allowing the exploration of the entire feasible region.
- By carefully crafting a mixer, it becomes possible to encode problem-specific knowledge or exploit the underlying structure of the optimization problem. This can lead to improved performances and potentially better solutions.

QAOA mixing operator

- Having a fixed mixing operator (driver) means that all problem dependence must be captured in the cost Hamiltonian H_p.
- The general strategy is to incorporate the hard constraints as penalty terms in the cost function and then convert the cost function to a cost Hamiltonian.
- This means that the algorithm must search a much larger space than if the evolution were confined to feasible configurations only, making the search less efficient.
- This issue suggests to replace the fixed driver by an alternative driver that confines the evolution to the feasible subspace.

QAOA mixing operator

Intuitively, mixing operators that restrict the search to the feasible subspace should result in better-performing algorithms.
In [11] Venturelli et al. lay down two fundamental design criteria a good mixing operator should satisfy:

- Preserve the feasible subspace;
- Provide transitions between all pairs of states corresponding to feasible points.

The paper provides several mixers for some combinatorial optimization problems, like graph coloring and job shop scheduling, for instance. The main idea behind the crafting of such mixers is to identify a swap rule that allows to easily transition between feasible solutions.

QAOA with custom mixer

Problem. Given a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, find the largest subset $V_0 \subset V$ of mutually non-adjacent vertices.

- The configuration space is the set of n-bit strings representing subsets $V_0 \subset V$ of vertices, where $i \in V_0$ if and only if $xi = 1$.
- The feasible region F is represented by the subset of all n-bit strings corresponding to independent sets of G.
- Given an independent set V_0, we can add a vertex $w \notin V_0$ to V_0 while maintaining feasibility only if none of its neighboring vertices $nbhd(w)$ are already in V_0.
- On the other hand, we can always remove any vertex $w \in V_0$ without affecting feasibility.
- Hence, a bit-flip operation at a vertex, controlled by its neighbors (adjacent vertices), suffices both to remove and add vertices while maintaining the independence property.

These considerations suggest the custom mixer described below.

QAOA with custom mixer

For each vertex, define the partial mixer as a multiply-controlled X operator

$$H_v = 2^{-D_v} X_v \otimes \bigotimes_{w \in nbhd(v)} (I + Z)_w,$$

where D_v is the degree of v in G.
Notice that for each of the vertices non explicitly appearing in H_v we have an underlying and not written tensor product with an Identity matrix.
The total mixer is then given by $H = \sum_{v \in V(G)} H_v$ with corresponding mixing unitary

$$U = e^{-i\beta H} = e^{-i\beta \sum_v H_v}.$$

The paper also discusses some partitioning strategies to efficiently implement such mixer. More details in the reference [11].

QAOA Parameter Optimization

- Optimizing QAOA parameters is challenging due to the non-convex nature of the objective function and the presence of low-quality local optima.
- Indeed, parameters optimization is extremely susceptible to starting point, i.e. the initial values of β, γ.
- In some special cases optimal angles can be found analytically [12], but this is generally not the case.
- Regardless, prior knowledge about the problem can guide the choice of angle intervals.
- If we expect a certain qubit to be in a specific state (e.g., $|0\rangle$ or $|1\rangle$) in the optimal solution, we can restrict the corresponding angle to a specific interval.
- Restricting the angles helps speed up the optimization, however the choice of angle restriction should strike a balance between exploration and exploitation.

QAOA Parameter Optimization

- It's been empirically observed that QAOA optimal parameters tend to cluster in the same region for problems belonging to the same class.
- This clustering behavior is observed across instances of different sizes and complexities.
- The phenomenon has been observed for problems like Maxcut [13], Max2Sat [14], and Modularity Maximization [15] on graphs.
- The clustering behavior of optimal parameters suggests a machine learning approach.
- **Idea:** train the model on smaller problem instances and leverage the learned parameters as a starting point for optimization on larger and more complex instances.

QAOA Parameter Optimization

- To train the model, we need a training set of T instances with known ground state $|g_t\rangle$.
- The objective is to find the best set of QAOA parameters that reproduce the ground state for these training instances.
- The trained parameters can be directly used to build the final circuit for the problem.
- Alternatively, they can serve as a warm start for further optimization on test instances.
- One possibility [14] is to consider the *Mean Squared Overlap* loss

$$L(\beta, \gamma) = \frac{1}{T} \sum_{t=1}^{T} |\langle \psi(\beta, \gamma) | g_t \rangle|^2$$

- Classically optimize this loss to obtain optimal parameters.

QAOA Parameter Optimization

- Obviously many other approaches can be used to obtain warm start angles.
- [16] for example considers the loss

$$L(\beta, \gamma) = \frac{1}{T} \sum_{t=1}^{T} |\langle \psi(\beta, \gamma) | H_t | \psi(\beta, \gamma) \rangle$$

  with H_t Hamiltonian associated to the t-th training instance.
- In their work, the authors simulate a gate-based model using *QuantumFlow*, a quantum circuit simulator implemented with TensorFlow.
- A recent study [17] instead employs Deep Reinforcement Learning [18] and Kernel Density Estimation techniques [19].
- By leveraging learned knowledge, they achieved efficient solutions with significantly fewer quantum circuit evaluations.

QAOA Gate Depth

- QAOA performance is influenced by the number of layers p in the algorithm.
- Being a discretization of Quantum Adiabatic Optimization (QAO) through Trotterization, for $p \to \infty$ QAOA finds the optimal solution.
- However, with current quantum hardware architectures, $p > 2$ is often impractical due to resource limitations.
- In fact, increasing the number of layers leads to more calls to the quantum hardware, which can significantly impact both resources and execution time.
- Empirically, it takes $2^{O(p)}$ random initial guesses to converge to a good solution.
- How to extend QAOA ansatz to efficiently handle larger values of p?

QAOA Gate Depth

- [13] also noted that as the depth p increases, optimal parameters again tend to cluster in the same region.
- By leveraging the clustering behavior, we can generate an initial set of $(p+1)$-depth parameters from p-depth optimal parameters.
- Such warm start can lead to better solutions compared to random initialization.
- The authors propose two heuristics, namely the FOURIER and INTERP heuristics, for building $(p+1)$-depth parameters using the previously obtained optimal parameters.
- Their strategy reduces the number of initial guesses down to $O(poly(p))$.

QAOA Gate Depth

- The FOURIER heuristic combines the previous p-depth optimal parameters using a Fourier sum-like approach.
- The INTERP heuristic produces $p+1$ initial parameters by linearly interpolating the previously found optimal parameters.
- Moreover, for problems like Maxcut, Max2Sat and Modularity Maximization, it has been observed that as p increases, the β parameter tends to decrease while the γ parameter tends to increase.
- This schedule aligns with the annealing process, starting from H_0 and slowly evolving into H_p.
- Again, this is not so surprising since QAOA is a time-discretization of Quantum Adiabatic Optimization.

QAOA Classical Optimizer

All the previous strategies help the optimization process achieve better solutions, but what about the underlying classical optimizer? Some frequently used algorithms include

- Nelder-Mead [20];
- Gradient Methods like L-BFGS (gradient estimated via finite differences);
- Bayesian Optimization [21];
- COBYLA;
- BOBYQA [22];
- Multi Level Single Linkage [23];
- APOSMM [24].

A comparison among some of these methods can be found in [25].

QAOA Execution Time

Finally, we list the main factors contributing to overall execution time:

- Quantum circuit execution time.
- Time to send the problem to the cloud architecture.
- Scheduling time on the quantum processor.
- Classical optimization execution time.
- Any pre-processing or post-processing steps.

Quantum Adiabatic Algorithm

- The Quantum Adiabatic Algorithm (QAA) solves optimization problems by gradually changing the system's Hamiltonian.
- It transitions from an initial Hamiltonian H_0 to a final Hamiltonian H_p encoding the problem we wish to solve.
- The algorithm starts in the ground state of H_0 and slowly varies the Hamiltonian over time.
- The adiabatic theorem guarantees that if the Hamiltonian changes slowly enough, the system stays in its ground state throughout the process.
- At the end of the evolution we get the ground state of H_p, optimal solution to the optimization problem.

For more details, refer to the reference [26].

Adiabatic Quantum Optimization

The complete (or system) Hamiltonian at a time t is given by

$$H(t) = \left(1 - s(\frac{t}{T})\right) H_0 + s(\frac{t}{T}) H_p$$

for $t \in [0, T]$ where s increases monotonically from $s(0) = 0$ to $s(1) = 1$ and T is the total running time of the algorithm.
If T is large enough, which is determined by the minimum spectral gap (the difference between the two lowest energy levels) of the system Hamiltonian, the adiabatic theorem guarantees the state at time t will be the groundstate of $H(t)$, leading to the solution, the ground state of $H(T) = H_p$.

Adiabatic Quantum Optimization

- The adiabatic approximation states that for a system initially prepared in an eigenstate, such as the ground state $|\psi(0)\rangle$ of a time-dependent Hamiltonian $H(t)$, the time evolution governed by the Schrödinger equation will approximately maintain the system in the corresponding instantaneous ground state (or other eigenstate) of $H(t)$ throughout the process.
- This approximation holds when the Hamiltonian $H(t)$ varies "sufficiently slowly".
- The total annealing time can be upper-bounded by the square of the inverse of the minimal eigen-gap of $H(t)$.
- As expected, determing such eigen-gap is usually as hard as solving the optimization problem itself. In general guesses need to be made.

Solving the Schrödinger equation

- Recall the Schrödinger equation

$$i\hbar \frac{\partial |\psi(t)\rangle}{\partial t} = H(t)|\psi(t)\rangle$$

- The solution to the Schrödinger equation yields a unitary time-operator $U(t, t_0)$

$$U(t, t_0) = \tau e^{\frac{-i}{\hbar} \int_{t_0}^{t} H(t')dt'}$$

  such that $|\psi(t)\rangle = U(t, t_0)|\psi(t_0)\rangle$.
- The time-operator is often computed using numerical approximation techniques, such as the Trotter-Suzuki decomposition.
- Main idea is to break down the time evolution operator into a product of simpler operators.

Solving the Schrödinger equation

- Divide the interval $[t_0,\ t]$ into smaller disjoint intervals of size δ, with δ small enough such that $H(t')$ is almost constant on $[t_0 + k\delta,\ t_0 + (k+1)\delta]$.
- This allows us to take $H(t')$ out of the integral and obtain

$$U(t, t_0) = \tau e^{\frac{-i}{\hbar}\sum_k H(t_0+k\delta)\delta}.$$

- Using Trotter-Lie formula we obtain the approximation

$$U(t, t_0) \sim \tau \prod_k e^{\frac{-i}{\hbar}H(t_0+k\delta)\delta}.$$

- Recalling that $H(t) = \left(1 - s(\frac{t}{T})\right) H_0 + s(\frac{t}{T})H_p$ and using again the Trotter approximation we get

$$U(t, t_0) \sim \tau \prod_k e^{-i\beta_k H_0} e^{-i\gamma_k H_p},$$

which also tells us that QAOA is just a Trotterization of Adiabatic Quantum Computation.

Quantum annealing in the transverse Ising mode

In [27] the authors study the evolution of the interpolating Hamiltonian

$$H(t) = H_p + \Gamma(t)H_0 = -\sum_{ij} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z - \Gamma(t)\sum_i \sigma_i^x,$$

and infer interesting analogies between the transverse field term $\Gamma(t)$ and the temperature $T(t)$ in classical Simulated Annealing. In particular, changing $\Gamma(t)$ in QA and $T(t)$ in SA from infinity to zero with the same functional forms

$$\Gamma(t) = T(t) = c/t, c/\sqrt{t}, c/\ln(t+1), \quad t : 0 \to \infty$$

yielded promising results, allowing QA to converge also faster then classical SA on the considered problem instances.

AQO with custom mixers

- Although the transverse field Hamiltonian is a convenient choice, other mixers have been defined for AQO.
- The problem is that, unlike QAOA, where we can start the algorithm from an arbitrary initial state, in AQO we require the ground state of the mixer to initialize the algorithm.
- Obtaining the ground state of a custom mixer can be challenging and computationally demanding.
- The transverse field Hamiltonian is often preferred in AQO because it has a known ground state, which is a uniform superposition of all computational basis states.
- Such ground state can also be prepared efficiently with simple *Hadamard gates*.

VQAs vs QAA

Variational Quantum Algorithm

- PRO: Uses an arbitrary parameterized quantum circuit to minimize the problem Hamiltonian
- PRO: Searches for the gradient in parameter space using classical optimization.
- CON: May get lost in parameter space.
- CON: Uses a large number of measurements.

Adiabatic Quantum Evolution

- PRO: Always finds its way to the solution.
- PRO: Can be optimized to pick a more efficient adiabatic path.
- CON: May be exponentially slow depending on the eigengap between the lowest and the second lowest eigenvalues

Adiabatically Assisted VQE

Idea: combine the virtues of adiabatic evolution with those of the VQE strategy. AAVQE works by repeatedly applying VQE to a series of Hamiltonians $\hat{H}(s) = (1-s)H_0 + sH_p$ that keep evolving from H_0 to H_p using discrete time steps Δs.
The algorithm can be summarized as follows:

1. Prepare the ground state of a simple Hamiltonian $\hat{H}(s_0)$ with a quantum circuit using VQE. Its final characterization is given by the parameters $\theta_{s_0}^{(f)}$.
2. Add a step Δs, that is $s_{i+1} = s_i + \Delta s$.
3. Run a VQE on $\hat{H}(s_{i+1})$ using as initial parameters $\theta_{s_{i+1}}^{(0)} = \theta_{s_i}^{(f)}$, the final parameters from the previous step.
4. If $s = 1$ stop, else go to 2).

More details can be found in [28].

Quantum Assisted Eigensolver

- QAE [29] is a novel hybrid classical-quantum algorithm for estimating the ground state of Hamiltonians.
- Differently from VQAs, QAE does not use a classical-quantum feedback loop during optimization.
- Quantum hardware is used just once to compute the matrices for the classical optimization problem.
- For a given Hamiltonian $H = \sum_i \beta_i U_i$, $U_i \in SU(2^n)$ the QAE algorithm involves three steps:
  1. Selection of an Ansatz,
  2. Calculation of the overlap matrices (D and E) on a quantum computer, and
  3. Execution of the corresponding QCQP on a classical computer.

QAE

Consider a set of m quantum states $\{|\phi_j\rangle\}_{j=1}^m$ such that $\langle\phi_j|\phi_k\rangle \neq 1$ for $j \neq k$ and $|\phi_j\rangle = V_j|0^{\otimes n}\rangle$, for some efficiently implementable unitary $V_j \in SU(2^n)$.
The Ansatz is chosen to be a linear combination of such quantum states

$$|\phi(\alpha)\rangle = \sum_{j=1}^{m} \alpha_j |\phi_j\rangle$$

We can then compute the expected energy of the Hamiltonian

$$\langle H(\alpha)\rangle = \sum_{j,k,i} \beta_i \alpha_j \langle\phi_j|U_i|\phi_k\rangle \alpha_k = \alpha^T D \alpha$$

where we define $D = \{D_{jk}\}_{j,k=1,\ldots,m}$ as

$$D_{jk} = \sum_i \beta_i \langle\phi_j|U_i|\phi_k\rangle \quad \forall j,k = 1,\ldots,m.$$

QAE

Imposing that $|\phi(\alpha)\rangle$ is a valid quantum state we get the constraint

$$\sum_{j,k=1}^{m} \alpha_j \langle \phi_j | \phi_k \rangle \alpha_k \stackrel{!}{=} 1$$

which can be written as

$$\alpha^T E \alpha = 1$$

where we define $E = \{E_{jk}\}_{j,k=1,\ldots,m}$ as

$$E_{jk} = \langle \phi_j | \phi_k \rangle \quad \forall j, k = 1, \ldots, m.$$

QAE

- We get the final Quadratically Constrained Quadratic Problem

$$\begin{aligned} \min\ & \alpha^T D \alpha \\ \text{s.t.}\ & \alpha^T E \alpha = 1 \\ & \alpha \in \mathbf{C}^m, \end{aligned}$$

  which can then be solved classically.
- The quantum computer is used just once to compute D, E, after that we can proceed with classical computers.
- QAE is currently rather unexplored, the author himself suggests possible future directions to refine it.
- The main disadvantage is the choice of initial states, which has not been fully discussed and is of critical importance for good results.

Integer Programs via Algebraic Geometry

We now consider a different approach to Integer Programming by means of Algebraic Geometry theory, for more details refer to [30]. Let $f : \mathbf{R}^n \to \mathbf{R}$ be a real-valued function. We want to solve the general non-linear integer optimization problem:

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t.}\ \ Ax = b \quad A \in \mathbf{Z}^{m\times n}, b \in \mathbf{Z}^m \\ &\qquad l \leq x \leq u \qquad\qquad x, l, u \in \mathbf{Z}^n \end{aligned}$$

One approach to solving such problem is to use an augmentation procedure: start from an initial feasible solution (which itself can be hard to find) and take an improvement step (augmentation) until one reaches the optimal solution. Augmentation procedures such as these need *test sets* or *optimality certificates*.

Test set

Definition

A set $S \in \mathbf{Z}^n$ is called a test set or optimality certificate if for every nonoptimal but feasible solution x_0 there exists $t \in S$ and $\lambda \in \mathbf{Z}_+$ such that $x_0 + \lambda t$ is feasible and $f(x_0 + \lambda t) \leq f(x_0)$. The vector t (or λt) is called the improving or augmenting direction.

- If the optimality certificate is given, any initial feasible solution x_0 can be augmented to the optimal solution in polynomial time!
- If S is finite, one can enumerate over all $t \in S$ and check if it is augmenting (improving);
- If S is not practically finite, or if all elements $t \in S$ are not available in advance, it is still practically enough to find a subset of S that is feasible and augmenting.

How to find Test sets?

The question now is, how do we find test sets for a given Integer Program? Algebraic Geometry offers us two ways:

- Reduced Gröbner basis of the *Toric Ideal* of A:

$$< x^u - x^v | u - v \in ker(A) \cap \mathbf{Z}^n_+ >;$$

- Graver basis of the integer kernel of A:

$$\mathcal{L}(A) = \{x \in \mathbf{Z}^n | Ax = 0, x \neq 0\} = (ker(A) \setminus \{0\}) \cap \mathbf{Z}^n.$$

Before moving on notice that:

- test set computation does not involve the objective function at all! Indeed, augmentation procedures treat the objective as an oracle. This allows us to extend such methods also to complex non-linear integer programs;
- In augmentation procedures it does not matter from which feasible solution one begins, nor does the sequence of improving steps taken: the final stop is an optimal solution;

Gröbner basis

A term order on $\mathbf{K}[x_1, ..., x_n]$ is a total order $\prec$ on the set of all monomials $x^a = x_1^{a_1} \cdots x_n^{a_n}$ which has the following two properties:

1. It is multiplicative; i.e., $x^a \prec x^b$ implies $x^{a+c} \prec x^{b+c}$ for all $a, b, c \in \mathbf{N}^n$.
2. The constant monomial is the smallest; i.e., $1 \prec x^a$ for all $a \in \mathbf{N}^n \setminus \{0\}$.

If we fix a term order $\prec$, then every polynomial f has a unique initial term $in_\prec(f)$, which is the $\prec$-largest monomial occuring with nonzero coefficient in the expansion of f.
Suppose now that I is an ideal in $\mathbf{K}[x_1, ..., x_n]$. Then its *initial ideal* $in_\prec(I)$ is the ideal generated by the initial terms of all the polynomials in I: $in_\prec(I) = \langle in_\prec(f) : f \in I \rangle$.
A finite subset G of I is a Gröbner basis with respect to the term order $\prec$ if the initial terms of the elements in G suffice to generate the initial ideal: $in_\prec(I) = \langle in_\prec(g) : g \in G \rangle$. More details in [31].

Graver Basis

Definition

Let $x, y \in \mathbf{R}^n$. We say that x is conformal to y, written $x \sqsubseteq y$, when $x_i y_i \geq 0$ (x and y lie on the same orthant) and $|x_i| \leq |y_i|$ for $i = 1, \ldots, n$.

Definition

The Graver basis of an integer matrix A is defined to be the finite set of $\sqsubseteq$-minimal elements (indecomposable elements) in the lattice $\mathcal{L}(A) = \{x \in \mathbf{Z}^n | Ax = 0, x \neq 0\}$. We denote by $G(A) \subset \mathbf{Z}^n$ the Graver basis of A.

Graver Augmented Multi-seed Algorithm

In the following we present GAMA, an augmentation algorithm which uses the test set provided by the Graver basis to optimize a given Integer Program. *Multiseed* refers to the use of multiple feasible solutions as starting points, each one of them augmented in parallel using the Graver basis. At the end, we select as best solution the one achieving lowest objective function value. More details in [32].

Algorithm 1 Graver Augmented Multi-seeded Algorithm (GAMA)

inputs: Matrix A, vector b, cost function $f(x)$, bounds $[l, u]$, as in Equation (1)
output: Global solution(s) x^*.
initialize: terminated solutions set, $termSols = \{\emptyset\}$.
Using any Graver extraction Algorithm input: A, extract $\mathcal{G}(A)$ (see section 3)
Find multiple feasible solutions, satisfying $l \leqslant x \leqslant u$ (subsec. *Feasible Solutions*)
for any feasible solution $x = x_0$ **do**
  while $g \in \mathcal{G}(A)$ **do**
    if $l \leqslant (x + g) \leqslant u$ and $f(x + g) < f(x)$ **then**
      $x = x + g$
    end if
  end while
  $termSols \leftarrow (x, f(x))$
end for
return $x^* = \{x \,|\, f(x) = \min_f(termSols)\}$

Computing the Basis

- As one could (or maybe not) expect, the two basis are related to each other and in some special cases they also coincide.
- There are various algorithms to compute the Gröbner and Graver basis, like for example the Buchberger algorithm and the Pottier algorithm respectively.
- Although many speed up exists, unfortunately computing such basis is usually computationally intractable on classical computers. That's where Quantum Computing kicks in!

Idea: use quantum hardware to compute the basis (Graver) and then classically iterate over its elements to augment a feasible solution towards the optimum.

GAMA via Quantum Annealing

Idea: reformulate the problem of finding the integer kernel of A as a QUBO problem. Then convert such QUBO to Ising and solve it via QAC. In more details:

- Consider the problem $\min[x^T A^T A x]$ s.t. $l \leq x \leq u$, with $l, x, u \in \mathbf{Z}^n, A \in \mathbf{Z}^{m \times n}$;
- binarize each x_i. For example $x_i = e_i^T X_i$ where $X_i^T = [X_{i1}, \ldots, X_{ik_i}], e_i^T = [2^0, 2^1, \ldots 2^{k_i}]$, $X_{ij} \in \{0, 1\}\ \forall i, j$;
- redefine $x = L + EX$ where L is an offset introduced to account for the bounds on x and $E = diag(e_1^T, \ldots, e_n^T)$;
- we have the QUBO problem: $\min[(L + EX)^T A^T A (L + EX)]$ s.t. $X_i \in \{0, 1\}$ for each i;
- convert to Ising using $X_i = \frac{1 - Z_i}{2}$, $Z_i \in \{-1, 1\}$ for each i;
- solve the Ising formulation via quantum annealing;
- filter the set of final solutions so to make it $\sqsubseteq$-minimal. The resulting set is the (partial) Graver basis we were looking for.

GAMA via QA

The algorithm reported above is a simplified version of the original algorithm proposed in [33]. In the original paper the authors perform a total of 4 classical post-processing routines:

1. a post-processing strategy to transform quantum annealer's near-optimal solutions into optimal solutions;
2. a filtering on the set of optimal solutions so to make it $\sqsubseteq$-minimal and extract Graver elements;
3. a further post-processing to turn the unused elements from the Kernel into additional Graver elements;
4. an adaptive binarization that shifts the middle point and the length of the encoding based on suboptimal solutions obtained in the previous iteration.

The first 3 routines aim at extracting as many Graver elements as possible while using the smallest number of calls to the annealer; the adaptive binarization instead aims at optimizing the number of qubits in the Ising formulation.

GAMA via Quantum Annealing

Employing the same strategy we can also find several feasible solutions which will act as starting points for the augmentation. To this end just notice that finding x such that

$$Ax = b \quad A \in \mathbf{Z}^{m \times n}, b \in \mathbf{Z}^n$$
$$l \leq x \leq u \quad l, x, u \in \mathbf{Z}^n$$

is equivalent to minimize $||Ax - b||^2 = (Ax - b)^T (Ax - b)$ which, after binary encoding, leads us to

$$\begin{aligned} \min\ & X^T Q_b X \\ & Q_b = E^T A^T A E + 2diag(L^T A^T A - b^T A) E \\ & X_i \in \{0, 1\} \quad \forall i \end{aligned}$$

Solving this QUBO using an Ising solver will give us many feasible solutions. Once such points are found, we can augment each point in parallel using our (partial) Graver Basis. These augmentations will lead us to the optimum point with high likelihood.

Minor Embedding and Chain-breaking

- Both minor embedding and chain-breaking strategies have a significant impact on the performance of the annealer, as highlighted in [33].
- The number of Graver elements obtained from a single call to the annealer is greatly influenced by chain-breaks.
- The authors utilized the built-in library provided by D-Wave for computing the minor embedding.
- Two different chain-breaking strategies were tested: energy minimization and majority voting.
- Energy minimization, while computationally more demanding, generally yielded a larger number of Graver elements in each call, although exceptions may occur.

Before moving on, let's focus briefly on the minor embedding problem.

Minor Embedding

- Given the current quantum architectures, we are often faced with the challenge of mapping a logical qubit (qubit used in the problem formulation) onto a physical qubit (qubit available in the hardware).
- The process of mapping a logical qubit to a collection of physical qubits is known as minor embedding.
- The goal of minor embedding is to find an efficient mapping that preserves the logical qubit's operations and interactions while utilizing the available physical qubits.
- It enables the execution of quantum algorithms on hardware with a limited number of physical qubits.
- Finding an optimal embedding can be computationally demanding and generally hard. We are dealing with a graph homomorphism problem!

Minor Embedding

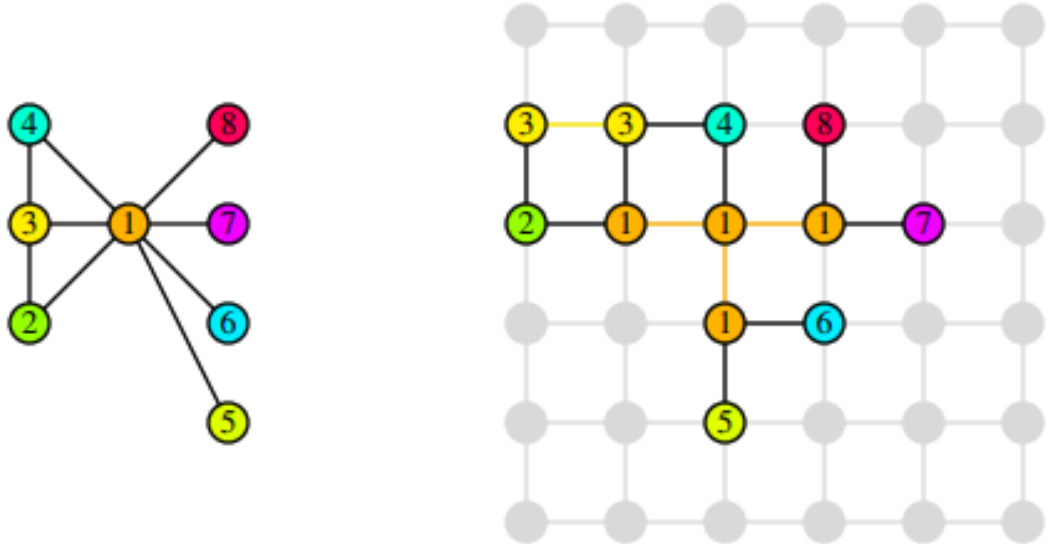
Definition

Let U be a fixed hardware graph. Given G, the minor-embedding of G is defined by the map $\phi : G \to U$ such that:

- each vertex v_i in $V(G)$ is mapped to a connected subtree T_i of U;
- there exists a map $\tau : V(G) \times V(G) \to V(U)$ such that for each $ij \in E(G)$, there are corresponding $i_{\tau(i,j)} \in V(T_i)$ and $j_{\tau(j,i)} \in V(T_j)$ with $i_{\tau(i,j)} j_{\tau(j,i)} \in E(U)$.

Given G, if ϕ exists, we say that G is embeddable in U.
In graph theory, G is called a minor of U. We denote the minor-embedding $\phi(G)$ of G by G_{emb}.

Minor Embedding: an example



G G_{emb}

G_{emb}(right) is a minor-embedding of G(left) in the square lattice U. Each vertex (called a logical qubit) of G is mapped to a (connected) subtree of (same color/label) vertices (called physical qubits) of U. G is called a (graph) minor of U.

Embedded Hamiltonian

- The embedded Hamiltonian represents the problem Hamiltonian formulated on the physical qubits after minor embedding.
- To ensure the correct execution of the algorithm and obtain the optimal solution, we aim to make the embedded Hamiltonian equivalent to the original Hamiltonian.
- The key idea is to adjust the qubit biases (h) and coupler constants (J_{ij}) in the embedded Hamiltonian to match the energy landscape of the original Hamiltonian.
- This alignment is crucial because it allows the optimal solution of the embedded Hamiltonian to correspond to the optimal solution of the original problem.

Embedded Hamiltonian

- Finding the optimal qubit biases and coupler constants involves an optimization process, which can be hard to solve given the noise and imperfections of the physical qubits and their interactions.
- Various strategies can be employed for optimization. The aim is to find the optimal parameter values that make the embedded Hamiltonian equivalent to the original one.
- An upper-bound on the embedding parameters for a given Ising Hamiltonian is provided in [34].

Quantum Vehicle Routing Problem

- Having laid the basis of Quantum Integer Programming, we can finally begin the study of quantum algorithms for VRP.
- The first work we mention is [35], a Systematic Literature Review of Quantum Computing for Routing Problems.
- It spans 53 different papers, from 2004 to 2021, in the intersection of Quantum Computing and routing problems.
- TSP engages most of the researchers (32 out of 53 papers), while the VRP accounts for 13 out of 53 papers. The rest of the papers deal with other routing problems.
- Useful starting point for further research.

In the following we discuss the most relevant among the papers we found.

QVRP [36]

- VRPTW via Route-based formulation.
- Set of possible routes built in a greedy fashion.
- Authors propose a new encoding to reduce number of qubits from n_c to $n_q = 1 + \log_2(n_c)$, with n_c being the number of variables.
- Instead of having one qubit per variable, they use the encoding

$$|\psi(\theta)\rangle = \sum_{k=1}^{n_c} \beta_k(\theta)[a_k(\theta)|0\rangle_a + b_k(\theta)|1\rangle_a] \otimes |\phi_k\rangle_r$$

where the *register* qubits $|\phi_k\rangle_r$ identify the variable x_k and the *ancilla* qubit $[a_k(\theta)|0\rangle_a + b_k(\theta)|1\rangle_a]$ stores its value.

QVRP [36]

- They propose VQE approach with Hardware Efficient Ansatz

$$U(\theta) = \prod_{l=1}^{L} \left(\bigotimes_{j=0}^{n_q} R_Y(\theta_{lj})_j \cdot \prod_{j=0}^{n_q-1} CNOT(j, j+1) \right) \cdot \bigotimes_{j=0}^{n_q} H_j,$$

  where $CNOT(j, j+1) = |0\rangle\langle 0|_j \otimes I_{j+1} + |1\rangle\langle 1|_j \otimes X_{j+1}$. The subscripts indicate the qubits the transformations are acting upon.

- Initial state is the all zero qubits state $|\psi_0\rangle = \bigotimes_{j=0}^{n_q} |0\rangle$.
- Hadamard gates introduce superposition; CNOT gates entangle subsequent qubits; Y-axis rotations introduce variational parameters for optimization.
- They find that reduced encoding can achieve same results of full encoding on small problem instances.
- However, for larger problems (currently intractable with full encoding) their strategy fails to find optimal solutions.

QVRP [37]

- CVRP with multiple vehicles via node-based formulation.
- $x_{v,t}^{k} \in \{0, 1\}$ marks whether vehicle k visits customer v at time step t.
- maximum time step set to $T = n$ for the worst case of one vehicle visiting all customers.
- $K \cdot V \cdot T$ total variables... too many!
- Heuristic two-phase approach splitting the CVRP into a Clustering Phase (CP) and a subsequent TSP.
- Clustering Phase aims to group customers such that the demands within a group do not exceed the truck's capacity and the groups stay close together (modeled as Multi-Knapsack problem).
- Then, TSP is solved for each cluster.
- This method drastically reduces the number of qubits allowing for implementation on NISQ devices.

QVRP [37]

- Authors focus solely on the TSP problem; Multi-Knapsack not considered. Instances of sizes 4, 5 and 6.
- They test both QAOA and VQE, with several variants.
- QAOA; Recursive QAOA; Warm-start QAOA; Constraints-Preserving Custom Mixer QAOA, with TSP custom mixer as given in [11].
- VQE with Hardware Efficient Ansatz

$$U(\theta) = \prod_j CNOT_{j,j+1}(\gamma_{j+1}) \cdot \bigotimes_j R_Z(\beta_j)_j \cdot \bigotimes_j R_X(\alpha_j)_j,$$

where

$$CNOT_{j,j+1}(\gamma_{j+1}) = |0\rangle\langle 0|_j \otimes I_{j+1} + |1\rangle\langle 1|_j \otimes R_X(\gamma_{j+1})_{j+1}.$$

Subscripts indicate the qubits the transformations are acting upon.

QVRP [37]

- X and Z-axis rotations introduce variational parameters for optimization; CNOT gates entangle subsequent qubits.
- Extensive study of hyperparameters is carried out, including the choice of the classical optimizer and the penalty factors in the QUBO formulation.
- VQE tends to perform way better than QAOA (possibly due to problem formulation), even across different optimizers.
- Powell and NFT best optimizer (for the considered instances).

QVRP [38]

- In this work, the authors use simulated VQE to solve VRP instances of 3 and 4 cities.
- $x_{ij} \in \{0, 1\}$ with $(i, j) \in E$ marks whether arc (i, j) is included in a route.
- Main focus is VQE robustness to several examples of noisy quantum channels.
- Findings reveal that performance of VQE depends heavily on what noise model is used. In general, noise is detrimental, but not equally so among different noise sources.
- In most cases, the effects of noise are minimal at the first layer of the circuits.
- While additional layers improve upon the results in the noiseless case, the opposite is valid with the introduction of noise.
- Empirical findings suggest COBYLA optimizer performs better for VQE circuits compared to the other available optimizers in Qiskit.

QVRP [39]

- Here the authors use QAOA to tackle Heterogeneous VRP instances of sizes $(N, V) = (3, 1), (3, 2), (4, 1)$.
- Different kinds of vehicles, with different capacities.
- The problem is decomposed into a routing problem (TSP) and a capacity problem (Knapsack problem).
- $y_{i\alpha}^{v} \in \{0, 1\}$ marks whether vehicle v visits node i in position α of its route; $z_{k}^{v} \in \{0, 1\}$ indicates whether vehicle v has capacity k.
- For the Knapsack problem, a log formulation is adopted to reduce the number of variables z_{k}^{v}.
- Final Hamiltonian is the sum of routing Hamiltonian and Knapsack Hamiltonian.
- Overall number of qubits $\#q = N_0^2 \cdot V + \sum_{v=0}^{V} \lfloor \log_2 Q_v \rfloor + 1$.

QVRP [39]

- Authors investigate performance of the algorithm with respect to both the classical optimizer and the depth p of the quantum circuit.
- For the choice of optimizer, they found that the basin-hopping with BFGS local search and differential-evolution work well.
- However, this performance came at the expense of longer optimization times.
- Still, final results are not competitive with classical approaches.
- In general, routing with additional capacity constraints is a difficult problem for VQAs.
- Biggest challenge seems to be the capacity constraints. Indeed, the energy landscape of KSP has multiple scattered local minima.

QVRP [40]

- In this work, QAOA to tackle VRP instances of sizes $(n, k) = (4, 2), (5, 2), (5, 3)$, no capacity constraints.
- $x_{ij} \in \{0, 1\}$ marks whether arc (i, j) is visited by some routes.
- Several depths p are tested, mainly with COBYLA optimizer.
- For instances $(4, 2), (5, 3)$ the optimal solution is found with high probability.
- For instance $(5, 2)$ the optimal solution is not found.

QVRP [41]

- This work introduces a QUBO formulation for *time-scheduled multiple capacitated VRP* (or TS-mCVRP).
- $x^{(i)}_{\tau a,|\, c_1,\ldots,c_M} \in \{0, 1\}$ marks whether vehicle i visits node a at time τ, with capacities $c_1, \ldots, c_M$.
- Time discretization makes τ a finite integer variable.
- Multi-capacities can also be interpreted as possible states the vehicle is in.
- QUBO formulation is tested on D-WAVE 2000Q (2048 qubits).
- Minor embedding via built-in library, with chains broken according to minimum energy strategy.
- Instance size of $(6, 3, 2h, 15m)$, namely 6 customers with 3 vehicles in 2 hours. The unit of time-scheduling is chosen to be 15 minutes.

QVRP [42]

- CVRP via the 2-Phase-Heuristic approach: clustering and routing phases.
- Clustering phase can be considered as the Knapsack Problem with additional distance minimization between the customers to be grouped.
- Routing phase can be reduced to the familiar Traveling Salesman Problem.
- Thus, it is possible to split the CVRP down to two individual QUBO problems and execute them sequentially.
- Authors propose a novel approach: clustering via classical algorithm and routing via Quantum Annealing.
- In particular, the clustering phase can be subdivided: (1) cluster generation and (2) cluster improvement.
- **Note:** clustering is not solved by KSP!

QVRP [42] - Cluster Generation

- Choose the core stop of a cluster, i.e. the first customer in a cluster, based on maximum demand or largest distance to the depot.
- Compute the geometric center of the cluster.
- From the set of unclustered customers, select the customer with the smallest distance to the cluster center and add it to the cluster.
- Recompute cluster center and repeat previous steps until the demand of a customer to be added would exceed the vehicle's capacity.
- If this is the case, a new core stop is selected and the still unclustered customers are assigned to the new cluster.
- This procedure stops when each customer has been assigned to a cluster.

QVRP [42] - Cluster Improvement

- Assign a customer v_i belonging to cluster C_k to another cluster C_j, if that would reduce the distance to the cluster center.
- Beware: assigning a customer to a new cluster must not violate the capacity limitation.
- If the reassignment is possible and valid, clusters' centers are recalculated and the improvement process begins again.
- The improvement step terminates if it is not possible to assign a customer to another cluster or when a certain stop criterion is reached (e.g., number of iterations).

QVRP [42]

- Authors use *QBSolv* tool to fit large QUBO problems to the physical hardware.
- QBSolv splits the QUBO into smaller components which are then solved independently of each other.
- This process is executed iteratively as long as there is an improvement.
- The QUBO matrix is split into different components using a classical tabu search heuristic in each iteration.
- Besides embedding and splitting the QUBO into subQUBOs, *QBSolv* also takes care of the unembedding and the merging of the subproblems' solutions.

QVRP [42]

- Several CVRP benchmark datasets are tested.
- Best Known Solutions are not achieved, even though some good approximations are found.
- Still, the approach is not competitive with some other classical heuristics.

References I

[1] Paolo Toth and Daniele Vigo. *Vehicle Routing. Problems, Methods, and Applications*. 2th. 2014.

[2] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. 3th. International Series in Operations Research & Management Science. Springer, 2019.

[3] Nanda Kumar Suresh and Panneerselvam Ramasamy. "A Survey on the Vehicle Routing Problem and Its Variants". In: *Intelligent Information Management* (2012). DOI: 10.4236/iim.2012.43010.

[4] Fred Glover, Gary Kochenberger, and Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 2019.

[5] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005. URL: https://doi.org/10.3389%5C%2Ffphy.2014.00005.

References II

[6] Ronald de Wolf. *Quantum Computing: Lecture Notes*. 2023. arXiv: 1907.09415 [quant-ph].

[7] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644.

[8] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (July 2014).

[9] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[10] Zhi-Cheng Yang et al. "Optimizing Variational Quantum Algorithms Using Pontryagin's Minimum Principle". In: *Physical Review X* 7.2 (May 2017). DOI: 10.1103/physrevx.7.021027. URL: https://doi.org/10.1103%5C%2Fphysrevx.7.021027.

References III

[11] Stuart Hadfield et al. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (Feb. 2019), p. 34. DOI: 10.3390/a12020034. URL: https://doi.org/10.3390%5C%2Fa12020034.

[12] Ojas D. Parekh, Ciaran Ryan-Anderson, and Sevag Gharibian. "Quantum Optimization and Approximation Algorithms.". In: (Jan. 2019). DOI: 10.2172/1492737. URL: https://www.osti.gov/biblio/1492737.

[13] Leo Zhou et al. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices". In: *Physical Review X* 10.2 (June 2020). DOI: 10.1103/physrevx.10.021067. URL: https://doi.org/10.1103%5C%2Fphysrevx.10.021067.

References IV

[14] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. "Training a quantum optimizer". In: *Physical Review A* 94.2 (Aug. 2016). DOI: 10.1103/physreva.94.022309. URL: https://doi.org/10.1103%5C%2Fphysreva.94.022309.

[15] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. "Multistart Methods for Quantum Approximate optimization". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2019. DOI: 10.1109/hpec.2019.8916288. URL: https://doi.org/10.1109%5C%2Fhpec.2019.8916288.

[16] Gavin E. Crooks. *Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem*. 2018. arXiv: 1811.08419 [quant-ph].

References V

[17] Sami Khairy et al. "Learning to Optimize Variational Quantum Circuits to Solve Combinatorial Problems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03 (Apr. 2020), pp. 2367–2375. DOI: 10.1609/aaai.v34i03.5616. URL: https://doi.org/10.1609%5C%2Faaai.v34i03.5616.

[18] Andrew Ng. *CS229 Lecture Notes*. 2022.

[19] Yen-Chi Chen. *A Tutorial on Kernel Density Estimation and Recent Advances*. 2017. arXiv: 1704.03924 [stat.ME].

[20] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313.

[21] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML].

References VI

[22] M. Powell. "The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives". In: *Technical Report, Department of Applied Mathematics and Theoretical Physics* (Jan. 2009).

[23] A. H. G. Rinnooy Kan and G. T. Timmer. *The Multi Level Single Linkage Method For Unconstrained And Constrained Global Optimization*. Econometric Institute Archives 272327. Erasmus University Rotterdam, 1985. URL: https://ideas.repec.org/p/ags/eureia/272327.html.

[24] Jeffrey Larson and Stefan Wild. "Asynchronously parallel optimization solver for finding multiple minima". In: *Mathematical Programming Computation* 10 (Feb. 2018), pp. 1–30. DOI: 10.1007/s12532-017-0131-4.

References VII

[25] Aidan Pellow-Jarman et al. "A comparison of various classical optimizers for a variational quantum linear solver". In: *Quantum Information Processing* 20.6 (June 2021). DOI: 10.1007/s11128-021-03140-x. URL: https://doi.org/10.1007%5C%2Fs11128-021-03140-x.

[26] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. arXiv: quant-ph/0001106 [quant-ph].

[27] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". In: *Phys. Rev. E* 58 (5 Nov. 1998), pp. 5355–5363. DOI: 10.1103/PhysRevE.58.5355. URL: https://link.aps.org/doi/10.1103/PhysRevE.58.5355.

[28] A. Garcia-Saez and J. I. Latorre. *Addressing hard classical problems with Adiabatically Assisted Variational Quantum Eigensolvers*. 2018. arXiv: 1806.02287 [quant-ph].

References VIII

[29] Kishor Bharti. *Quantum Assisted Eigensolver*. 2020. arXiv: 2009.11001 [quant-ph].

[30] David E. Bernal, Sridhar Tayur, and Davide Venturelli. *Quantum Integer Programming (QuIP) 47-779: Lecture Notes*. 2021. arXiv: 2012.11382 [quant-ph].

[31] Bernd Sturmfels. "What is a Gröbner basis?" In: 52.10 (2005).

[32] Hedayat Alghassi, Raouf Dridi, and Sridhar Tayur. *GAMA: A Novel Algorithm for Non-Convex Integer Programs*. 2019. arXiv: 1907.10930 [math.OC].

[33] Hedayat Alghassi, Raouf Dridi, and Sridhar Tayur. *Graver Bases via Quantum Annealing with Application to Non-Linear Integer Programs*. 2019. arXiv: 1902.04215 [quant-ph].

References IX

[34] Vicky Choi. *Minor-Embedding in Adiabatic Quantum Computation: I. The Parameter Setting Problem*. 2008. arXiv: 0804.4884 [quant-ph].

[35] Eneko Osaba, Esther Villar-Rodriguez, and Izaskun Oregi. "A Systematic Literature Review of Quantum Computing for Routing Problems". In: *IEEE Access* 10 (2022), pp. 55805–55817. DOI: 10.1109/ACCESS.2022.3177790.

[36] Ioannis D. Leonidas et al. *Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the NISQ era*. 2023. arXiv: 2306.08507 [quant-ph].

[37] Lilly Palackal et al. *Quantum-Assisted Solution Paths for the Capacitated Vehicle Routing Problem*. 2023. arXiv: 2304.09629 [quant-ph].

References X

[38] Nishikanta Mohanty, Bikash K. Behera, and Christopher Ferrie. *Analysis of The Vehicle Routing Problem Solved via Hybrid Quantum Algorithms in Presence of Noisy Channels*. 2023. arXiv: 2205.07630 [quant-ph].

[39] David Fitzek et al. *Applying quantum approximate optimization to the heterogeneous vehicle routing problem*. 2021. arXiv: 2110.06799 [quant-ph].

[40] Utkarsh Azad et al. "Solving Vehicle Routing Problem Using Quantum Approximate Optimization Algorithm". In: *IEEE Transactions on Intelligent Transportation Systems* (2022), pp. 1–10. DOI: 10.1109/tits.2022.3172241. URL: https://doi.org/10.1109%2Ftits.2022.3172241.

[41] Hirotaka Irie et al. *Quantum Annealing of Vehicle Routing Problem with Time, State and Capacity*. 2019. arXiv: 1903.06322 [quant-ph].

References XI

[42] Sebastian Feld et al. "A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer". In: *Frontiers in ICT* 6 (June 2019). DOI: 10.3389/fict.2019.00013. URL: https://doi.org/10.3389%2Ffict.2019.00013.

[43] Andrew M. Childs. *Lecture Notes on Quantum Algorithms*. 2022.

[44] Robert M. Parrish et al. "Quantum Computation of Electronic Transitions Using a Variational Quantum Eigensolver". In: *Physical Review Letters* 122.23 (June 2019). DOI: 10.1103/physrevlett.122.230401. URL: https://doi.org/10.1103%5C%2Fphysrevlett.122.230401.

References XII

[45] Nikolaj Moll et al. "Quantum optimization using variational algorithms on near-term quantum devices". In: *Quantum Science and Technology* 3.3 (June 2018), p. 030503. DOI: 10.1088/2058-9565/aab822. URL: https://doi.org/10.1088%5C%2F2058-9565%5C%2Faab822.

[46] Walter Vinci and Daniel A. Lidar. "Non-stoquastic Hamiltonians in quantum annealing via geometric phases". In: *npj Quantum Information* 3.1 (Sept. 2017). DOI: 10.1038/s41534-017-0037-z. URL: https://doi.org/10.1038%5C%2Fs41534-017-0037-z.

[47] Tameem Albash and Daniel A. Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (Jan. 2018). DOI: 10.1103/revmodphys.90.015002. URL: https://doi.org/10.1103%5C%2Frevmodphys.90.015002.

References XIII

[48] Jonathan Romero et al. *Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz*. 2018. arXiv: 1701.02691 [quant-ph].

[49] Ruslan Shaydulin et al. "Network Community Detection on Small Quantum Computers". In: *Advanced Quantum Technologies* 2.9 (June 2019), p. 1900029. DOI: 10.1002/qute.201900029. URL: https://doi.org/10.1002%5C%2Fqute.201900029.