Optimising the Optimiser

Sayantan Pramanik
TCS Research, TATA Consultancy Services

1 (Combinatorial) Optimisation

Optimisation problems are ubiquitous. They arise whenever a decision needs to be made. When the problems are small enough, we can solve them mentally without sparing much thought. However, such decision-problems can get mind-numbingly large and complex. Such problems are being solved, at multiple levels, every time an AI model is trained or an e-commerce platform delivers our parcels.

The anatomy of optimisation problems consists mainly of three parts: (a) the **decision variables** which need to be tuned, (b) the rules or **constraints** they must abide by, and (c) the **objective function** that must be minimised or maximised. When the decision variables can only assume a finite set of discrete values, the resulting problems are **combinatorial** in nature. Combinatorial optimisation problems are, counterintuitively and in general, more difficult to solve than their continuous counterparts [1]. A particularly interesting thing about many combinatorial optimisation problems is that they can be transformed from one to another [1]. As a consequence, if an algorithm can be designed to solve one class of combinatorial optimisation problems, all the problems that can be *reduced* to instances of the said class can be solved using the same algorithm.

1.1 The MaxCut Problem

In this exercise, we will attempt to solve the *MaxCut* problem. We are given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices in the graph, and $\mathcal{E}$ is the set of edges between the said vertices. Each edge also has a weight associated with it, with the weight being zero if the corresponding edge is absent. The objective is to partition the vertices into two disjoint sets such that the sum of edges crossing between groups is maximised. The objective function for MaxCut is given by:

$$\max_{x_i \in \{\pm 1\}, i \in \mathcal{V}} \sum_{i,j} w_{ij}(1 - x_i x_j), \tag{1}$$

where w_{ij} is the weight of the edge between the vertices i and j in $\mathcal{V}$, and the decision variables x_i can either be $+1$ or -1, representing the two groups.

2 Variational Quantum Algorithms

Combinatorial optimisation is a prime candidate for quantum advantage. The expectation is that the problem can be mapped onto a physical system in such a way that its ground state represents the solution to the problem. Then, the system may be allowed to evolve, and the resulting state measured. Accordingly, for the MaxCut problem, the physical Hamiltonian may be derived by replacing the decision variables in Eq. (1) with Pauli Z matrices:

$$\mathbf{H} = \sum_{i,j} w_{ij}(Z_i Z_j). \tag{2}$$

[1] Discussing this at length is beyond the scope of this exercise, but I would encourage you to read more and discover the reason behind this bizarre claim.

It is then easy to verify that the problem in Eq. (1) is equivalent to:

$$\min_{|\phi\rangle} \langle\phi|\mathbf{H}|\phi\rangle \,. \tag{3}$$

Subsequently, measuring $|\phi\rangle$ in the computational basis gives us the MaxCut solution.

However, searching over all possible $|\phi\rangle$ is akin to bruteforcing the problem. Variational Quantum Algorithms (VQAs) [2] proposes an alternative. The quantum state is parametrised such that $|\phi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\iota\rangle$, where $|\iota\rangle$ is some initial state (conventionally chosen to be $|0\rangle^{\otimes n}$ or $|+\rangle^{\otimes n}$), and $U(\boldsymbol{\theta})$ is a unitary operator called the *ansatz*. The optimisation problem has now morphed into:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \Big(f(\boldsymbol{\theta}) := \langle\phi(\boldsymbol{\theta})|\mathbf{H}|\phi(\boldsymbol{\theta})\rangle \Big). \tag{4}$$

Here, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of optimisable parameters, and U is a hyperparameter. The Quantum Approximate Optimisation Algorithm (QAOA) [3] is a special kind of VQA which prescribes a way to construct an appropriate U based on the problem being solved. For the purpose of this exercise, the choice of U is left to you. The parameters are typically optimised iteratively using a classical optimisation algorithm Gradient Descent (GD)[2]:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla f(\boldsymbol{\theta}^{(t)}), \tag{5}$$

where $\alpha > 0$ is a positive *step-size* or *learning rate*. It is important to note that the gradient vector may be computed (or rather *estimated*) using the Parameter-Shift Rules [4].

3 Quantum Imaginary Time Evolution

Let us keep quantum aside for a moment. How would we solve the problem of finding the maximum eigenvalue of a diagonalisable matrix $\mathbf{M}$? The answer lies in a method called *Power Iteration* [5]. The algorithm starts with a random vector $\mathbf{x}^{(0)}$, and iterates as follows:

$$\mathbf{x}^{(t+1)} = \frac{\mathbf{M}\mathbf{x}^{(t)}}{||\mathbf{M}\mathbf{x}^{(t)}||}. \tag{6}$$

The number of iterations required depends on the *spectral gap* of $\mathbf{M}$. Now instead, if we want to find the minimum eigenvalue/eigenvector of $\mathbf{M}$, we simply replace $\mathbf{M}$ with $\mathbf{M}^{-1}$ in the algorithm (in which case, the matrix better be invertible).

However, using the Power Iteration method is almost impossible for a Hamiltonian $\mathbf{H}$ for one simple reason: the size of the matrix for interesting problems. For an n-qubit system, $\mathbf{H}$ has dimensions of $2^n \times 2^n$! Storing such large matrices and vectors is a challenge, let alone multiplying them. For quantum systems, as usual, we fall back on the Schrodinger equation. The dynamics of Power Iteration is replaced by time-evolution of a quantum system whose Hamiltonian is $\mathbf{H}$:

$$|\psi(t)\rangle = e^{-i\mathbf{H}t}\,|\psi(0)\rangle \,. \tag{7}$$

But again, there is a problem with this approach. The time-evolution operator $e^{-i\mathbf{H}t}$ is unitary. Unitary operators conserve some quantity, in this case, the energy of the system. This is detrimental for us because the system needs to dissipate energy to settle into its ground state. The solution for this is extraordinary, to say the least: treat time itself as imaginary [6]! Let the imaginary time τ be:

$$\tau = it. \tag{8}$$

[2]This is the same algorithm that is used to train AI models, and more popularly, LLMs!

The resulting Quantum Imaginary Time Evolution (QITE) algorithm is as follows [7]:

$$|\psi(\tau)\rangle = \frac{e^{-\mathbf{H}\tau}\,|\psi(0)\rangle}{\sqrt{\langle\psi(0)|e^{-2\mathbf{H}\tau}|\psi(0)\rangle}}. \tag{9}$$

An important caveat for this algorithm to work is that the initial state must have a non-zero overlap with the ground-state. Implementing this non-unitary dynamics on a quantum computer (which, as we know, works with unitary operations) is difficult. A possible solution is to create a variational state and transform it in a way that *tracks the trajectory of* $|\psi(\tau)\rangle$. This is what Variational Quantum Imaginary Time Evolution (VarQITE) enables us to do.

3.1 Variational Quantum Imaginary Time Evolution

First, we need to understand how the state $|\psi(\tau)\rangle$ evolves with τ. On one hand, consider the Schrodinger equation with imaginary time:

$$\frac{\partial\,|\tilde{\psi}\rangle}{\partial\tau} = -\mathbf{H}\,|\tilde{\psi}\rangle\,. \tag{10}$$

This raises another issue. Schrodinger equation with imaginary time does not preserve normalisation of the state $|\tilde{\psi}\rangle$[3]. The issue of normalisation is solved by considering Wick rotated Schrodinger equation [6]:

$$\frac{\partial\,|\psi(\tau)\rangle}{\partial\tau} = -(\mathbf{H} - E_\tau\mathbf{I})\,|\psi(\tau)\rangle\,. \tag{11}$$

Here, the state $|\psi(\tau)\rangle$ continues to be normalised, and $E_\tau = \langle\psi(\tau)|\mathbf{H}|\psi(\tau)\rangle$.

On the other hand consider the parametrised quantum state $|\phi(\boldsymbol{\theta})\rangle$. For $j \in [d]$, define:

$$\dot{\boldsymbol{\theta}}_j := \frac{\partial\boldsymbol{\theta}_j}{\partial\tau},\ \text{and}\ |\partial_j\phi\rangle := \frac{\partial\,|\phi(\boldsymbol{\theta})\rangle}{\partial\boldsymbol{\theta}_j}. \tag{12}$$

Consequently,

$$\frac{\partial\,|\phi\rangle}{\partial\tau} = \sum_{j=1}^{d}\dot{\boldsymbol{\theta}}_j\,|\partial_j\phi\rangle\,. \tag{13}$$

Now, for the variational state to track the trajectory of QITE, we want equations (11) and (13) to be as close as possible. To do this, we solve the following *mean-squared* sub-optimisation problem:

$$\min_{\dot{\boldsymbol{\theta}}\in\mathbb{R}^d}\left|\left|\frac{\partial\,|\psi(\tau)\rangle}{\partial\tau} - \frac{\partial\,|\phi\rangle}{\partial\tau}\right|\right|^2 \tag{14}$$

As daunting as Eq. (14) may seem, it is easy to solve analytically. The solution is given by [6]:

$$\dot{\boldsymbol{\theta}} = -\frac{1}{2}\mathbf{A}(\boldsymbol{\theta})^{-1}\nabla f(\boldsymbol{\theta}). \tag{15}$$

Here, $\mathbf{A}(\boldsymbol{\theta})_{ij} = \Re\,\langle\partial_i\phi|\partial_j\phi\rangle$. Of course, the above equation assumes that $\mathbf{A}(\boldsymbol{\theta})$ is invertible. If not, for all practical purposes, the pseudoinverse may be used. With this, we have finally arrived at an update-rule for the parameters in our ansatz:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \beta\mathbf{A}(\boldsymbol{\theta}^{(t)})^{-1}\nabla f(\boldsymbol{\theta}^{(t)}), \tag{16}$$

where $\beta = {}^{\Delta\tau}/_{2}$ is analogous to a step-size.

[3] I promise that we are not simply moving from one problem to another, but making steady and directed progress towards an overall solution. Please have some faith in me!

4 An Alternative to Gradient Descent

That is all the Algebra we need, I swear. No more Maths beyond this point. Instead, let us take a step back and reflect on what we have achieved so far. Compare equations (5) and (16), and notice how similar they look. Eq. (16) may be obtained by replacing $\nabla f(\boldsymbol{\theta}^{(t)})$ with $\mathbf{A}(\boldsymbol{\theta}^{(t)})^{-1}\nabla f(\boldsymbol{\theta}^{(t)})$ in Eq. (5)! This similarity is not accidental. Gradient descent, in a way, updates all the parameters equivalently, i.e., it performs updates in Euclidean space. However, the objective function may be more sensitive to a certain set of parameters than others. To capture this heterogeneous sensitivity, space needs to be distorted from *spherical* to *elliptical*. That is exactly what the matrix $\mathbf{A}(\boldsymbol{\theta}^{(t)})^{-1}$ does. By multiplying the gradient with this *preconditioning* matrix, we perform geometrically-meaningful updates to the parameters in the ansatz with the hope that this will lead to accelerated convergence. Correspondingly, the algorithm in (16) is referred to as **Preconditioned Gradient Descent (PGD)**.

5 Challenge

Your task is to solve a weighted MaxCut problem using two optimisation methods for variational quantum algorithms: gradient descent (GD) and variational quantum imaginary-time evolution (VarQITE). You may refer to the code in [7], and implement VarQITE using Qiskit's inbuilt functions, but implementing it from scratch will earn additional credit. You may use your favourite ansatz for this challenge.

5.1 Data

Shared along with this PDF is a ".npy" file which contains a list of tuples. The data in the file can be used to construct a graph. Each tuple in the list is of the form (i, j, w_{ij}), where i and j are nodes in the graph, and w_{ij} is the weight of the edge between them.

5.2 Objectives

1. Use a simulator to solve the MaxCut problem on the shared graph with GD and VarQITE (PGD). You may ignore the effect of all kinds of noise (including shot-noise). Try running the experiments with different values of step-sizes.
2. Report the number of circuits evaluated per iteration with GD and PGD.
3. Plot the value of $\langle\phi(\boldsymbol{\theta}^{(\tau)})|\mathbf{H}|\phi(\boldsymbol{\theta}^{(\tau)})\rangle$ against the number of circuit-evaluations for both PGD and GD.
4. For both GD and PGD, report the objective value corresponding to the best-found solution + $\langle\phi(\boldsymbol{\theta}^{(T)})|\mathbf{H}|\phi(\boldsymbol{\theta}^{(T)})\rangle$, where $\boldsymbol{\theta}^{(T)}$ is the optimised set of parameters.
5. Try solving the problem with a finite number of shots. How do the results change as the number of shots is varied?
6. Examine the structure of the shared graph carefully. Can you identify properties that make the MaxCut instance easier or harder to solve?

Best of luck, and have fun!

References

[1] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014.

[2] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[4] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), mar 2019.

[5] Power iteration - Wikipedia. `https://en.wikipedia.org/wiki/Power_iteration`. [Accessed 26-10-2026].

[6] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon C. Benjamin, and Xiao Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5(1), 2019.

[7] Andreu Anglés-Castillo, Luca Ion, Tanmoy Pandit, Rafael Gomez-Lurbe, Rodrigo Martínez, and Miguel Angel Garcia-March. Understanding quantum imaginary time evolution and its variational form, 2025.