Team Pineapple: iQuHACK 2023

Tarushii Goel, Alex Hu, Rishab Parthasarathy, Rowechen Zhong, Isaac Zhu

January 29, 2023

Contents

Question 1 Stochastic Gradient Descent

We propose an improvement to the derivation of the angles (γ, β) using stochastic gradient descent instead of a random grid search. Because each instance of quantum circuits is quite complex to run, we implement a stochastic gradient descent algorithm using a technique known as the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm.

Now, we describe precise algorithmic details. We begin with randomly generated vectors $\gamma, \beta \in [0, 1]^n$. To calculate gradients for γ, β over each step, we define a stepsize $\alpha = 0.0001$ and corresponding random perturbations $\Delta\gamma, \Delta\beta \in [\alpha * 0.1, \alpha]^n$.

Using the SPSA algorithm, we can now establish that

$$(\nabla\gamma)_i = \frac{F(\gamma + \Delta\gamma) - F(\gamma - \Delta\gamma)}{2 * (\Delta\gamma)_i} \tag{1}$$

with F denoting the energy of the system following evolution and a corresponding equation for β.

Hence, we utilize only four energy computations total to implement gradient descent, a small constant factor increase of runtime that is scalable to an arbitrarily large number

of dimensions. Specifically, training the model on the same number of iterations, with a learning rate of $\eta = 0.0001$, the model achieves comparable results of energy of $E = 4.9345$ to the randomized maxcut on a small-dimensional graph, while also presenting potential for further optimization on higher-dimensional graphs where randomized grid search may become infeasible.

Open-Ended Investigation: Improved Max-Cut using Subgraphs

We improve the max-cut techniques from https://arxiv.org/pdf/1411.4028.pdf, where we begin in the 2^n-dimensional Hilbert space in a uniform superposition $|s\rangle$ of the computational basis states, and attempt to optimize the Hamiltonian $C = \sum_{\langle jk\rangle} \frac{1}{2}(1 - Z_j Z_k) \equiv \sum_{\langle jk\rangle} C_{\langle jk\rangle}$. We also define the operator $B = \sum X_i$. We then take $2p$ angular parameters $\gamma_1, \cdots, \gamma_p, \beta_1, \cdots, \beta_p$ which parameterize rotational operators $U(C, \gamma) = e^{-i\gamma C}, U(B, \beta) = e^{-i\beta B}$. Using these, we construct the state

$$|\gamma, \beta\rangle \equiv U(B, \beta_p)U(C, \gamma_p) \cdots U(B, \beta_1)U(C, \gamma_1) |s\rangle .$$

Our mission is to find the maximum eigenvalue of C, so we will attempt to maximize the value $F_p(\gamma, \beta) = \langle \gamma, \beta| C |\gamma, \beta\rangle$. To compute this quantity, we observe that the operator associated with edge $\langle jk\rangle$, i.e.

$$U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) C_{\langle jk\rangle} U(B, \beta_p) \cdots U(C, \gamma_1),$$

only affects qubits whose distance to $\langle jk\rangle$ on the graph is at most p. So to compute the overall energy $F_p(\gamma, \beta)$, we can

- loop over all edges $\langle jk\rangle$ in the graph
- only need to look at the subgraph g generated by all vertices at most distance p from $\langle jk\rangle$
- compute the energy contribution

$$f_g(\gamma, \beta) \equiv \langle s, g| U^\dagger(C_g, \gamma_1) \cdots U^\dagger(B_g, \beta_p) C_{\langle jk\rangle} U(B_g, \beta_p) \cdots U(C_g, \gamma_1) |s, g\rangle$$

- report the total energy $F_p(\gamma, \beta) = \sum_g w_g f_g(\gamma, \beta)$, where w_g is a combinatorial factor counting the number of times we encounter a sugraph of shape g as we loop over all edges

This energy computation is costly if we iterate a large number of times as we search across the (γ, β) space, since we have to run the quantum circuit every time we want to compute a value of $f_g(\gamma, \beta)$. Our method to improve this is as follows: Instead of computing $f_g(\gamma, \beta)$ every time as we search across (γ, β) space, we precompute a model $\tilde{f}$ of $f_g(\gamma, \beta)$ for each g that we might be interested in (i.e. subgraphs g generated from our value of p). In particular, we will precompute the values of f on a lattice in (γ, β) space, for each g (by running the quantum circuit). We then extend from the lattice to the entire (γ, β) space by taking the Fourier transform of our data and cutting off the high frequency modes (and then taking the inverse Fourier transform to get a function $\tilde{f}$ on all of (γ, β) space). (The high frequency modes are cut off since they represent the noise in the measurement). Since we expect the true f to be smooth on (γ, β) space, we expect this Fourier-lattice-extension $\tilde{f}$ to approximate the true f well, especially for small enough lattice spacing. The upshot of all of this is that once we have precomputed the $\tilde{f}$ for all the g we might be interested in, we never have to perform the costly quantum

computations ever again: when you give me a graph G, all I effectively have to do is enumerate the subgraph combinatorial factors w_g, and then access the precomputed values $\tilde{f}_g(\gamma, \beta)$, as I search across (γ, β) space. In other words, we simply optimize the quickly computable quantity $\tilde{F}_p(\gamma, \beta) = \sum_g w_g \tilde{f}_g(\gamma, \beta)$. We can do this efficiently using the same methods described previously, i.e. stochastic gradient descent in (γ, β) space through SPSA.

In terms of explicit results, when we run the described optimizations with $p = 1$ (there are only three relevant subgraphs to precompute), we are able to quickly handle graphs G with up to around 30 vertices, and return the correct maxcut with high probability within 50 shots. We also precomputed a large majority of the $p = 2$ subgraphs, which was able to significantly increase the probability, albeit with a slightly slower runtime (since (γ, β) space is larger).

Question 2 Expanding Past Max-Cut

§2.1 Maximum Dominating Set

§2.1.1 Problem Encoding

We developed a mapping of the maximum dominating set problem to a Hamiltonian cost matrix H_p that can be used to solve the problem with QAOA.

Suppose you have a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. Let $N(v) = \{i \mid (i, v) \in E\}\{v\}$. Each vertex v is associated with a variable $x_v \in \{0, 1\}$. If $x_v = 1$, vertex v is in the dominating set. The cost matrix is

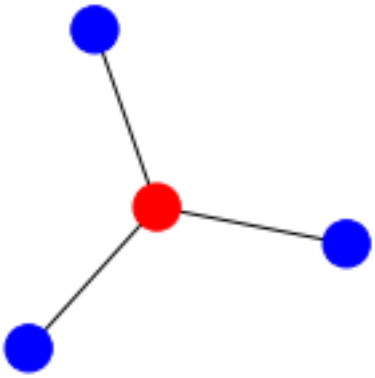
$$C = -\sum_v \prod_{i \in N(v)} (1 - x_i) - \gamma * \sum_v x_v$$

where γ is a variable that scales how much we should prioritize satisfying the constraints of the problem (the solution is actually a dominating set) as opposed to maximizing the objective (generating the smallest set, i.e. the smallest $\sum x_i$). Each term in the first summation encodes a constraint for one vertex. If $x_i = 0 \quad \forall i \in N(v)$, then the term enforces a cost of -1, otherwise, the cost is 0. By plugging in $x_i = \frac{1}{2}(1 - Z_i)$, where Z_i is the Pauli Z operator acting on qubit i, you can get the Hamiltonian H_p.

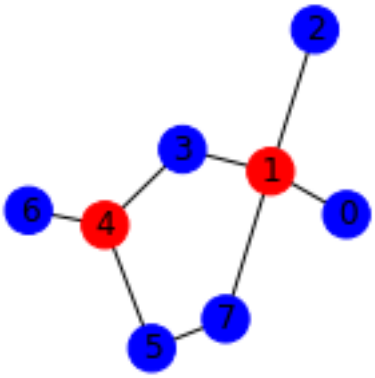
$$H_p = -\sum_v \prod_{i \in N(v)} \frac{1}{2}(1 + Z_i) - \gamma * \sum_v \frac{1}{2}(1 - Z_v)$$

Thus, the highest energy state of the hamiltonian is associated with satisfying all the constraints.

§2.1.2 Results



For this 3-node graph, the probability of the node in the maximum dominating set (the node in the center) being in the state $|1\rangle$ is higher than the probability of the other nodes in the graph. The probability of node it being activated (across all 5000 shots) is 0.8, but the probability of the other nodes being activated is 0.42.



For this 8-node graph, the probability of the nodes in the maximum dominating set (nodes 1 and 4) being in the state $|1\rangle$ are higher than the probability of the other nodes in the graph. For example, the probability of node 1 being activated (across all 5000 shots) is 0.78, but the probability of node 0 being activated is 0.43.

§2.2 Part 2: Four-Coloring

§2.2.1 Problem Encoding

We developed a mapping of the four coloring problem to a Hamiltonian cost matrix H_p that can be used to solve the problem with QAOA.

Suppose you have a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We will have two qubits associated with each vertex: each possible measurement result of these qubits is associated with a coloring of that vertex (2 bits = 4 colors). Z_i is the Pauli Z operator acting on the first qubit associated with vertex i and Z_i' is the Pauli Z operator acting on the second qubit associated with vertex i.

$$H_p = -\sum_{(i,j)\in E} \frac{1}{4}(1 + Z_i Z_j)(1 + Z_i' Z_j')$$

§2.2.2 Results

We ran this algorithm on the 3-clique. The highest probability state was associated with a valid solution to the 3-coloring problem.

One of the limitations is that we needed 2 qubits for every node and the simulator only supported a maximum of 10 qubits.

Question 4 Symbolic and cached circuit recompilation

The original QAOA implementation involves repeated recompilations of the problem and mixer hamiltonian circuits for various values of γ and β, requiring 35.9 s to run on qBraid. To mitigate the computational load of such recompilations, we propose and compare three potential solutions to the problem.

§4.1 Binary Lifting

Our first approach we call binary-lifting. Since γ and β are bounded between 0 and 1, we pre-compute operators $C_p(\gamma)$ and $C_p(\beta)$ for $\gamma, \beta = \frac{1}{2^i}$ where $0 \leq i \leq 10$. These operators are then compiled prior to constructing the various quantum circuits.

Because the Hamiltonians are composed of completely commuting terms, we have the relation that the exponential operators satisfy:

$$\begin{aligned} C_p(\gamma_1 + \gamma_2) &= C_p(\gamma_1)C_p(\gamma_2) \\ C_p(\beta_1 + \beta_2) &= C_p(\beta_1)C_p(\beta_2) \end{aligned} \tag{2}$$

Hence, for any β and γ, when we arrive at the value during the optimization stage of runtime, we simply decompose β and γ into their binary representations and compose the corresponding quantum circuits.

In total, we require 1.3 seconds of precompute and 13.9 s of runtime.

§4.2 Uniform pre-computing

For this model, we simply precompute $C_p(\gamma)$ and $C_p(\beta)$ for $\gamma, \beta = \frac{i}{1024}$, allowing us to perform runtime in 4.5 seconds, despite precomputation being extremely computationally intensive.

Overall, for a more complex space, this might be more optimal, as pretraining becomes less of an impact compared to the additional runtime of binary lifting.

§4.3 Symbolic recompilation

We use the fact that our problem and mixer circuits are parameterized by γ and β to efficiently recompile the circuit across multiple values of γ and β.

We achieve a speedup from the original time of 35.9 seconds to 4.9 seconds for the entire QAOA maxcut optimization problem.