Quantum Approximate Optimization for Hard Problems in Linear Algebra

Ajinkya Borle*
CSEE Department
University of Maryland, Baltimore County
Baltimore, USA

Vincent E. Elfving†
Qu & Co BV
PO Box 75872, 1070 AW
Amsterdam, the Netherlands

Samuel J. Lomonaco‡
CSEE Department
University of Maryland, Baltimore County
Baltimore, USA

(Dated: November 26, 2020)

The Quantum Approximate Optimization Algorithm (QAOA) by Farhi et al. is a quantum computational framework for solving quantum or classical optimization tasks. Here, we explore using QAOA for Binary Linear Least Squares (BLLS); a problem that can serve as a building block of several other hard problems in linear algebra. Most of the previous efforts in quantum computing for solving these problems were done using the quantum annealing paradigm. For the scope of this work, our experiments were done on the QISKIT simulator and an IBM Q 5 qubit machine. We highlight the possibilities of using QAOA and QAOA-like variational algorithms for solving such problems, where trial solutions can be obtained directly as samples, rather than being amplitude-encoded in the quantum wavefunction. Our simulations show that Simulated Annealing can outperform QAOA for BLLS at a circuit depth of $p \leq 3$ for the probability of sampling the ground state. Finally, we point out some of the challenges involved in current-day experimental implementations of this technique on a cloud-based quantum computer.

I. INTRODUCTION

The application of quantum computing to hard optimization problems is a candidate where quantum computing may eventually outperform classical computation [1–6]. At the time of writing this paper, Noisy Intermediate Scale Quantum (NISQ) computers [5] are being developed by several firms and research groups [7–13]. The two main approaches to quantum optimization are (i) the Quantum Annealing (QA) physical heuristic [6] and (ii) Quantum Approximate Optimization Algorithm (QAOA) [4] on the gate-model quantum computer [1].

In this paper, we explore and propose the use of QAOA for hard problems in linear algebra. In particular, we focus on the problem of binary linear least squares (BLLS) [14–16]. This problem and its variant (binary compressive sensing [17]) have applications in signal [16, 17] and image processing [15, 18]. Another interesting context of BLLS is that it can be a building block for other hard problems in linear algebra[19]. This can be seen in previous works that applied quantum annealing to this domain of problems [20–23]. We hope that our work provides insights to fellow researchers to further explore the use of NISQ era methods [4, 24] for problems in linear algebra and numerical computation. In Section 2, we cover the necessary background and related work for our paper. Section 3 is about formulating the BLLS problem for the QAOA ansatz. The experiments, results and discussion are detailed in Section 4. We finally conclude our paper in Section 5. We also have Appendices to complement and support the information in the main paper.

II. BACKGROUND AND RELATED WORK

A. Background

1. The binary linear least squares problem

Given a matrix $A \in \mathbb{R}^{m \times n}$, an unknown column vector of variables $x \in \{0,1\}^n$ and a column vector $b \in \mathbb{R}^m$ (Where $m > n$). The linear BLLS problem is to find the x that would minimize $\|Ax - b\|$ the most. In other words, it can be described as:

$$\arg\min_x \|Ax - b\| \tag{1}$$

The motivation behind choosing the BLLS problem as a target problem is twofold: firstly, its an NP-Hard problem [16] that makes it a relevant target for potential speedup. Secondly, it can act as a building block for

* aborle1@umbc.edu
† vincent.elfving@quandco.com
‡ lomonaco@umbc.edu

other hard problems in linear algebra, such as the Non-negative Binary Matrix Factorization [21]. One reason why one may view BLLS a building block for other problems is because multiple binary variables can be clubbed together for a fixed point approximation of a real variable [20, 22, 25–28]. Amongst these, there are some problems that are NP-hard for which an approximate solution would be acceptable [22, 28]. In these cases, QAOA may be able to provide an improvement finding approximate solutions compared to classical solvers and increase the probability of sampling the best solution.

2. Non-negative Binary Matrix Factorization (NBMF)

NBMF is a specialized version of the Non-negative Matrix Factorization (NMF) problem. Given a matrix $V \in \mathbb{R}_{\geq 0}^{m \times n}$, the problem is to factorize it into matrices $W \in \mathbb{R}_{\geq 0}^{m \times r}$ and $H \in \{0,1\}^{r \times n}$ (H would have non-negative real entries in NMF).

NMF and its variants are used in multiple disciplines such as computer vision[29], astronomy[30] and data mining[31], just to name a few. BLLS can be used in order to solve the NBMF variant by using the alternating least squares method [32].

Algorithm 1 Alternating least squares for NMF

1: **procedure** MAIN(V) ▷ V is the matrix to be factorized
2: Randomly initialize the matrix $H \in \{0,1\}^{r \times n}$
3: **while** not converged **do**
4: **for** row i from 1 to m **do**
5: $(W_i)^T \leftarrow \arg\min_{W_i^T} \|V_i^T - H^T (W_i)^T\|_2$ such that $W \in \mathbb{R}_{\geq 0}^{m \times r}$
6: **end for**
7: **for** column j from 1 to n **do**
8: $H_j \leftarrow \arg\min_{H_j} \|V_j - W H_j\|_2$ such that $H \in \{0,1\}^{r \times n}$
9: **end for**
10: **end while**
11: **return** W,H
12: **end procedure**

In Algorithm 1, line 5 is solved classically since efficient algorithms exist for it[33], it's line 8 that is solved using BLLS (where QAOA would be applied).

In the past, quantum annealing was used as a subroutine within this algorithm to solve NBMF and other NMF related problems[21, 22]. Based on our work with this paper, QAOA can be an alternative to quantum annealing for NBMF, which can be explored in the future.

3. Quadratic Unconstrained Binary Optimization (QUBO)

The QUBO Objective function is as follows,

$$F(q) = \sum_a v_a q_a + \sum_{a<b} w_{ab} q_a q_b \tag{2}$$

where $q_a \in \{0,1\}$, v_a and w_{ab} are real coefficients for the linear and quadratic parts of the function respectively. The QUBO objective function is NP-hard in nature [34]. One salient feature of this objective function is that many application domain problems map naturally to QUBO [20–22, 35, 36]. In the process of applying BLLS to gate model quantum devices, we use the QUBO formulation as an intermediate stage of expressing the problem.

4. The Quantum Approximate Optimization Algorithm (QAOA)

Algorithm 2 Quantum Approximate Optimization Algorithm (minimize)

1: **procedure** MAIN($\hat{B}, \hat{C}, p$) ▷ The main routine of the algorithm
2: $\beta \leftarrow \{\emptyset\}, \gamma \leftarrow \{\emptyset\}, expt\_val \leftarrow \emptyset, best\_res \leftarrow \emptyset$
3: Pick at random $\beta \in [0,\pi]^p, \gamma \in [0,2\pi]^p$
4: **while** (β, γ) can be further optimized, or a limit is reached **do**
5: Initialize $res\_set \leftarrow \{\emptyset\}$
6: **for** `a fixed number of shots` **do**
7: $res\_set \leftarrow res\_set \cup$ QAOA($\hat{B}, \hat{C}, \beta, \gamma, p$)
8: **end for**
9: From res_set, calculate the expectation value and store in $expt\_val$
10: Based on the $expt\_val$, pick new $2p$ angles (β, γ) by classical optimization
11: **end while**
12: From the final res_set, set the result with lowest energy, $best\_res \leftarrow min(res\_set)$
13: **return** $best\_res$
14: **end procedure**
15: **procedure** QAOA($\hat{B}, \hat{C}, \beta, \gamma, t$)
16: Initialize n qubits, $|\psi\rangle \leftarrow |0\rangle^{\otimes n}$
17: Apply Hadamard transform, $|\psi\rangle = 1/\sqrt{2^n}(|0\rangle + |1\rangle)^{\otimes n}$
18: $j \leftarrow 1$
19: **while** $j \leq t$ **do**
20: $|\psi\rangle \leftarrow e^{-i\gamma_j \hat{C}} |\psi\rangle$
21: $|\psi\rangle \leftarrow e^{-i\beta_j \hat{B}} |\psi\rangle$
22: $j \leftarrow j + 1$
23: **end while**
24: Measure $|\psi\rangle$ in standard basis and store in a classical register o
25: **return** o
26: **end procedure**

In 2014 Farhi et al. proposed an algorithm that uses both quantum and classical computation for solving optimization problems [4]. The potential advantage of using this algorithm is that it can be implemented by using low depth quantum circuits [37], making it suitable for NISQ devices. We here briefly summarize the QAOA formalism applied to binary optimization problems. For the required preliminaries of quantum computing, the authors recommend the textbook by Nielsen and Chuang [1].

One popular method of encoding an optimization problem to be solved using QAOA, is to first formulate the

problem as an Ising Objective function.

$$F(\sigma) = \sum_a h_a \sigma_a + \sum_{a<b} J_{ab} \sigma_a \sigma_b \tag{3}$$

$$\text{where } \sigma_a = 2q_a - 1 \tag{4}$$

Where $\sigma_a \in \{-1, 1\}$, h and J are coefficients associated with individual and coupled binary variables respectively. The Ising model is a popular statistical mechanics model, associated primarily with ferromagnetism [38]. Because it has been shown to be NP-Complete in nature [39], the objective function associated with it can be used to represent hard problems [40]. It is important to note here that we still don't know if quantum computing can help solve NP-Complete problems efficiently [41]. Our hope for quantum algorithms, at the very least, is to be able to compete with classical heuristics when it comes to certain classes of hard problems.

The problem then would be to maximize or minimize Eqn(3), depending on how it is set up. The quantum Ising Hamiltonian, which naturally maps the Ising objective Eqn(3) to qubits, can be expressed as:

$$\hat{C} = \sum_a h_a \hat{\sigma}_a^{(z)} + \sum_{a<b} J_{ab} \hat{\sigma}_a^{(z)} \hat{\sigma}_b^{(z)} \tag{5}$$

$$\text{where } \hat{\sigma}_a^{(z)} = (\otimes_{i=1}^{a-1} \hat{I}) \otimes (\hat{\sigma}^{(z)}) \otimes (\otimes_{i=a+1}^{n} \hat{I}) \tag{6}$$

$$\text{and } \hat{\sigma}^{(z)} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

Here, indices a, b, i label the qubits, n is the total number of qubits, $\hat{\sigma}^{(z)}$ is the Pauli Z operator and I is the identity operator. The other type of Hamiltonian in the QAOA process is a summation of individual Pauli X operators for each qubit involved in the process, which intuitively represents a transverse field in the Ising model:

$$\hat{B} = \sum_a \hat{\sigma}_a^{(x)} \tag{8}$$

$$\text{where } \hat{\sigma}_a^{(x)} = (\otimes_{i=1}^{a-1} \hat{I}) \otimes (\hat{\sigma}^{(x)}) \otimes (\otimes_{i=a+1}^{n} \hat{I}) \tag{9}$$

$$\text{and } \hat{\sigma}^{(x)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{10}$$

In QAOA, the qubits are first put in a uniform superposition over the computational basis states by applying a Hadamard gate, which maps $|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}$, on every qubit. Then, the Hamiltonian pair $\hat{C}$ and $\hat{B}$ is applied p number of times using a set of angles γ and β, where, for $1 \leq l \leq p$, each[42] $\gamma_l \in [0, 2\pi]$ and $\beta_l \in [0, \pi]$ [4]. The expectation value of the Hamiltonian $\hat{C}$ with respect to the resultant state $|\psi(\gamma, \beta)\rangle$ is calculated as

$$C(\gamma, \beta) = \langle \psi(\gamma, \beta) | \hat{C} | \psi(\gamma, \beta) \rangle \tag{11}$$

A classical black-box optimizer then uses the expectation value as its input and suggests new γ and β sets (of length p each). The hope is that as the number of qubits (more specifically, variables) n involved in the optimization increases, if for circuit depth $p \ll n$ we are able to efficiently sample the best solution, we would have an advantage in using QAOA over classical methods. Although algorithm 2 is a summary of the QAOA method, we recommend readers the original paper [4] for further details.

5. *Implicit filtering optimization*

As mentioned before, QAOA requires us to give it the sets of angles γ and β in order to change the state of the quantum system. The most common way to do this is to use classical black-box optimization techniques that do not need the derivative information of the problem[9, 43, 44]. Since the expectation value $C(\gamma, \beta)$ of the objective function cost (or energy) would be approximate in nature, we need an optimization technique that can handle noisy data. The technique of our choice for this work is the Implicit Filtering algorithm [45].

In essence, Implicit Filtering or ImFil is a derivative-free, bounded black-box optimization technique that accommodates noise when it tries to suggest the best parameters to minimize the objective function. Various other techniques for noisy optimization exist, such as Bayesian Optimization[46], COMPASS[47], SPSA[48], etc. However, we found Implicit Filtering the best for our current efforts. For further details, we recommend the book by C.T Kelly on the topic [45].

B. Related work

One of the first applications of quantum computing for solving problems in the field of linear algebra is the HHL algorithm for solving a system of linear equations [49]. This was followed by works for solving linear least squares [50], preconditioned system of linear equations [51], recommendation systems [52] and many others [53–56]. Although the classical counterparts of the above mentioned algorithms run in polynomial time, the quantum algorithms mentioned above run in the polylog time complexity.

However, there are some caveats with such kind of algorithms [57]. Among the many caveats, we'd like to emphasize on the two that affect the practicality of their utility in the near future. Firstly, they require fault tolerant quantum computers whereas, at the time of writing this paper, we have just entered the NISQ era [5]. Secondly, for the algorithms focused on linear system of equations [49, 51, 53] and least squares [50, 54], the output data is encoded as a normalized vector of a quantum state $|x\rangle$ (which means that the probability amplitudes of the basis states encode the data). This means that we need an efficient method to prepare the input data as a quantum state [57–60]; and the output will be a quantum state as well, which means it wouldn't be available for us in the classical world directly by performing measurement in the standard computational basis. This can

be mitigated by either measuring the final state in a basis of our choice if our goal is to know some statistical information about x [49, 61] or learning certain values in x (though that will eliminate the exponential speedup [57]).

With respect to quantum annealing, O'Malley and Vesselinov's paper in 2016 [20] was one of the first that proposed to solve linear least squares. Other works in this domain were for solving specific NMF problems [21, 22, 62], polynomial system of equations [25], underdetermined binary linear systems [23] and polynomial least squares [27]. It's hard to speculate about speedups analytically with (i) D-wave's noisy implementation of quantum annealing [63] and (ii) the problem of exponential gap-closing between the problem Hamiltonian's ground state and its excited states [64]. In the work by O'Malley and Vesselinov [21], they used a time to target benchmark in which classical solvers (Tabu search [65] and gurobi [66] in their comparison) have to match or find better solutions than the ones returned by quantum annealing (not necessarily the optimal solution) in the same amount of time. The D-wave quantum annealer was able to beat those classical solvers for the benchmark, but the authors also mention that a combination of the two classical techniques would probably perform better than the D-wave by compensating for each other's shortcomings. The other important result in subsequent papers [22, 62] was to show that combining reverse and forward annealing improved results over just using forward annealing for most cases. Golden and O'Malley saw an improvement of 12% over forward annealing [62], but that came at the price of having at least 7 reverse annealing runs per QUBO (which was reported to have the QPU runtime of 29 forward anneals). It is important to note that certain quantum inspired algorithms may perform just as well or better than quantum annealers for such highly dense problems in terms of variable interactions [67]. The above mentioned quantum annealing techniques use the Ising objective function for problem formulation. This means that measuring the post annealing quantum state in computational basis gives us a bitstring which directly encodes the vector x (which we hope to be the best solution to Eqn(1)) , unlike a lot of gate-model algorithms like the ones mentioned above [49–51, 53, 54] that encode the solution in the amplitudes of $|x\rangle$.

NISQ-compatible algorithms for efficiently solving linear algebra problems are highly desirable as of the time of writing this paper. The work by Chen et. al [68] proposes a hybrid algorithm that uses quantum random walks for solving a particular type of linear system, producing a classical result in $O(n \log n)$. However, the closest related works to ours are the recent papers that employ variational algorithms [69–71]. The major difference however, is that, in those papers : i) The output is encoded as the vector of probability amplitudes of the quantum state $|x\rangle$ and ii) The problems explored thus far are convex in nature and solved in polynomial time classically.

We in this paper implement QAOA on similar problems which were implemented on D-wave's quantum annealer previously, and therefore briefly mention a comparison here. The standard QAOA circuit strategy can be seen as similar to a bang-bang quantum annealing schedule, where cost and driver Hamiltonians are alternated. The Quantum Alternating Operator Ansatz extends this with more general operators [72] than available to Ising Hamiltonian annealers. Furthermore, QAOA is a gate-based quantum computational algorithm, a type of framework which promises universal programmability in terms of mapping an arbitrary problem graph to a qubit layout, even if the latter is not all-to-all connected. Conversely, in quantum annealing architectures mapping the logical problem qubits to a graph of physical hardware qubits can be a significant challenge in the general case [73]. Our work is certainly not the first in applying QAOA to various relevant computational problems, and we refer the reader to a small list of examples [74–76]; in this work, we make an attempt to highlight some of the salient features and challenges of QAOA in the context of problems applicable to linear algebra and numerical analysis.

III. QAOA FOR BLLS

Before we go deeper, we here set the context of how QAOA will be used in this work. Rather than treating QAOA as an approximation algorithm with theoretical guarantees for the quality of solution obtained, it is used as a heuristic supported by empirical results.

A. Problem formulation

O'Malley and Vesselinov first gave a QUBO formulation for the BLLS problem [20]. The details of how that is done is in Appendix A. Referring back to Eqn(1), if $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $x \in \{0,1\}^n$, we can refine Eqn(2) to be

$$F(x) = \sum_j v_j x_j + \sum_{j<k} w_{jk} x_j x_k \tag{12}$$

$$\text{where } v_j = \sum_i A_{ij}(A_{ij} - 2b_i) \tag{13}$$

$$\text{and } w_{jk} = 2\sum_i A_{ij} A_{ik} \tag{14}$$

Which means that the number of qubits depends only upon the size of the column vector x. All the rows in Matrix A and vector b are preprocessed classically in order to produce the coefficients of the QUBO problem.

By the equivalence stated in Eqn(4), we can then convert the problem into an Ising objective function (plus

an offset value, irrelevant for optimization)

$$F(\sigma) = \sum_j h_j \sigma_j + \sum_{j<k} J_{jk}\sigma_j\sigma_k + \text{offset} \tag{15}$$

$$\text{where } \sigma_j = 2x_j - 1 \tag{16}$$

B. Mapping to quantum gates

Using the h,J coefficients from Eqn(15) along with the mapping to a quantum Ising Hamiltonian given in Eqn(6) we get:

$$\hat{C} = \sum_j h_j \hat{\sigma}_j^{(z)} + \sum_{j<k} J_{jk}\hat{\sigma}_j^{(z)}\hat{\sigma}_k^{(z)} \tag{17}$$

Because the individual components of Eqn(17) commute [4], we can express the Hamiltonian simulation of $\hat{C}$ with an angle γ_l as follows

$$e^{-i\gamma_l \hat{C}} = \prod_j e^{(-ih_j\gamma_l)\hat{\sigma}_j^{(z)}} \prod_{j<k} e^{(-iJ_{jk}\gamma_l)\hat{\sigma}_j^{(z)}\hat{\sigma}_k^{(z)}} \tag{18}$$

Similarly, the exponential of hamiltonian B can be broken down as

$$e^{-i\beta_l \hat{B}} = \prod_j e^{(-i\beta_l)\hat{\sigma}_j^{(x)}} \tag{19}$$

In order to realize Eqn(18) and Eqn(19), we use the following gates

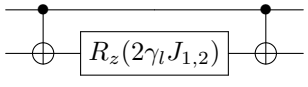
$$R_x(\omega) = e^{-i\frac{\omega}{2}\hat{\sigma}^{(x)}} = \begin{pmatrix} \cos\omega/2 & -i\sin\omega/2 \\ -i\sin\omega/2 & \cos\omega/2 \end{pmatrix} \tag{20}$$

$$R_z(\omega) = e^{-i\frac{\omega}{2}\hat{\sigma}^{(z)}} = \begin{pmatrix} e^{-i\omega/2} & 0 \\ 0 & e^{i\omega/2} \end{pmatrix} \tag{21}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{22}$$

While Eqn(20) is the only gate needed to realize Eqn(19), Eqn (21) alone can merely help with the single qubit components of Eqn(18). For the components that require two qubit interaction, the following gate combination (expressed diagrammatically) is used as a template

$$e^{(-iJ_{1,2}\gamma_l)\hat{\sigma}_1^{(z)}\hat{\sigma}_2^{(z)}} = \tag{23}$$



While Eqn(23) shows the ZZ interactions for adjacent qubits, this strategy can be generalized to any pair of qubits in the system. Appendix B provides an example of a QAOA circuit for BLLS.

1. For IBM Q specific gates

Our experiments were done on an IBM Q device (`ibmq_london`) available to us through the IBM Q Network. This machine has the following basis gates: $\{U_1, U_2, U_3, \text{CNOT}, I\}$. The first three gates in the set can be described as

$$U_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \tag{24}$$

$$U_2(\lambda,\phi) = \begin{pmatrix} 1/\sqrt{2} & -e^{i\lambda}/\sqrt{2} \\ e^{i\phi}/\sqrt{2} & e^{i(\lambda+\phi)}/\sqrt{2} \end{pmatrix} \tag{25}$$

$$U_3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\theta/2 & -e^{i\lambda}\sin\theta/2 \\ e^{i\phi}\sin\theta/2 & e^{i(\lambda+\phi)}\cos\theta/2 \end{pmatrix} \tag{26}$$

We can implement Eqn(20) and Eqn(21) [77] as

$$R_x(\omega) = U_3(\omega, -\frac{\pi}{2}, \frac{\pi}{2}) \tag{27}$$

$$R_z(\omega) = U_3(\pi,0,\pi)\, U_1(-\frac{\omega}{2})\, U_3(\pi,0,\pi)\, U_1(\frac{\omega}{2}) \tag{28}$$

Another practical consideration to be taken is the qubit connectivity of a real quantum computer. As the number of qubits increase, it is safe to assume that full connectivity between physical qubits is not feasible to engineer. This means that for distant-qubits to interact with each other, we would need logical qubit replacement using SWAP operations. Appendix C elaborates on this with a demonstration with IBM Q gates.

C. Experiment methods

The dataset used in our experiments was randomly generated (seeded for reproducibility) consisting of $A \in \mathbb{R}^{40\times n}$, $b \in \mathbb{R}^{40}$ and $x \in \{0,1\}^n$ where $n \in \{3,4,5,9,10\}$ is the size of the problem. Due to the exponential nature of simulating quantum computation on classical hardware and the limited resources in our hands, we decided to limit our problem sizes to 10 qubits. Instead of allocating computational resources to even larger qubit numbers, we focus on: i) increasing the number of problem instances we average over, ii) comparing exact wavefunction versus bitstring-sampling experiments, iiii) comparing QAOA performance (waveform) with Simulated Annealing and random sampling. All values for A in the dataset are generated by uniformly sampling floating point approximations of real values in the interval $[-1.0, 1.0)$, and then rounding the values to 3 decimal places. For each value of n, we generate 100 test cases with 40 cases in which $Ax^* = b$, where the best solution x^* is sampled randomly and $b \leftarrow Ax^*$. The other 60 cases have $Ax^* \neq b$, where b is generated similarly to A and the best solution x^* is found by going through all 2^n possible values for x. This is done to cover both scenarios of the least squares problem. The matrix A is a sparse matrix

having density of 0.2, this was done because sparse matrices have a lot of applications in numerical computation and machine learning [78–80].

We use the QISKIT [81] SDK to write our own implementation of the QAOA algorithm. As mentioned before, ImFil [45] is our black-box optimizer of choice. The only parameter of ImFil we choose to control here is the budget, which governs the maximum iteration limit. The rest of the ImFil parameters for our experiments use their default values. Similarly, unless explicitly stated, all qiskit parameters values taken are default as well. All classical simulations were conducted on standard `x86-64` based laptops. Following is a list of the experiments we conducted.

1. *Experiments with no noise*

Our first set of experiments on the dataset were done on a simulator with the statevector backend, giving us the exact waveform. This means that we are able to compute the exact expectation value $C(\gamma, \beta)$ for the set of angles γ and β. These experiments help assess the performance of QAOA in a perfectly noiseless environment for a large dataset.

The above set of experiments were done for p =1, 2 and 3 with random starting points: 20 for p =1, 40 for 2 and 60 for 3 (seeded for reproducibility). Our preliminary study suggested a budget of 200 iterations for $p = 1, 2$ and 400 iterations for $p = 3$ respectively. This ensured that at least 70% of our tests converged within the budget while being computationally feasible. At the end of the process, for each problem, the best result from all the starting points is chosen and recorded.

2. *Experiments to compare no noise and shot-noise performance*

For our next set of experiments, we use measurement based results on the simulator. Each circuit is run a number of times, specified by the 'shots' parameter. This means that the expectation value we get for a given γ and β is approximate in nature. Thus, while quantum circuit simulation itself is noiseless and deterministic in producing the same wavefunction before taking each shot, a finite number of shots is sampled from the resulting wavefunction output probability distribution, introducing a stochastic component. In a real quantum device, one is always limited to this finite tomography, as one has no direct access to the qubit register's quantum wavefunction.

Since in a real quantum device, we do not have access to the qubit register's waveform, simulations with shot-noise are important to conduct. Each experiment was done 10 times per shot value. The shot values chosen for these experiments are in the set $\{2^i | n-2 \leq i \leq n+2, i \in \mathbb{Z}\}$. We chose this range in order to observe the performance in the limit of perfectly reproducing the wavefunction.

Also, the problem instances chosen for this set of experiments are a random subset of the original dataset. For each problem size $n \in \{3, 4, 5, 9, 10\}$, we randomly choose 5 problems of the 100 problems (while maintaining the 2 : 3 ratio of the problems by their type). This is done because doing the shot-noise experiments on the original dataset would be computationally infeasible for the limited computational resources at our disposal, since each shot-noise experiment is at least 50 times slower than its statevector counterpart.

The parameters of these experiments have also been modified accordingly. They were done for p =1, 2 and 3, for a budget of 200 iterations with random starting points: 5 for p =1, 10 for 2 and 15 for 3 (seeded for reproducibility), with the best result being chosen and recorded. For fair comparison, this subset of problem instances was also run with the statevector backend for the same parameters.

3. *Experiments on an IBM Q device*

Based on the results of the first two sets of experiments, we design our experiments for the 5 qubit IBM Q device '`ibmq_london`'. In a real device like this one, the qubits face decoherence issues, coherent gate errors, control errors, incoherent gate errors, leakage, cross-talk, readout noise and more. The first set of IBM Q experiments was to run QAOA for problems with n =5, for parameters p =1, budget of 200 iterations and a shot value of 1024. The reason for choosing these parameters for QAOA is to take into account the gate depth limitation and noisy computation, thus choosing the minimal number of qubits while still covering a non-trivial problem graph structure, which can still be easily verified with classical methods at this size. The next set of experiments was to take the γ and β from the results of the statevector experiments done in Section III C 2 (where n =5) and to try and recreate the distribution and expectation values using the quantum computer.

D. Results

Two metrics we use here to quantify the performance of QAOA in the simulations are (i) the probability of sampling the best possible solution (or the ground state of Hamiltonian $\hat{C}$) and (ii) the relative error ϵ_r of the expectation value $C(\gamma, \beta)$ (from Eqn(11)) with respect to the ground state energy E_{gs}. We define ϵ_r as

$$\epsilon_r = \left| \frac{C(\gamma, \beta) - E_{gs}}{E_{gs}} \right| \tag{29}$$

We note that ϵ_r is zero when the expectation value is exactly equal to the groundstate energy, and can grow

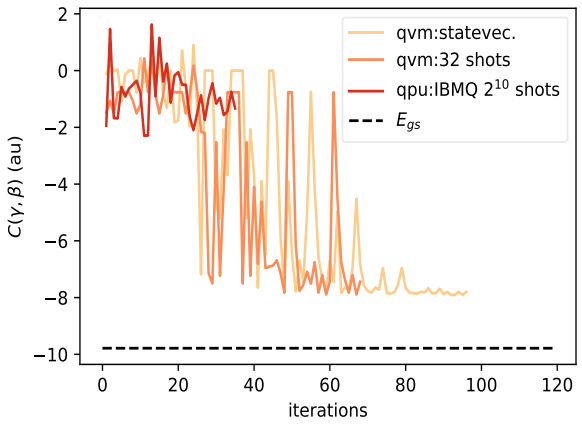


FIG. 1. One instance of convergence behaviour for the QAOA optimization routines run on an exact statevector simulator, shot-noise simulator with $2^5 = 32$ shots, and the 5-qubit IBM Q (`ibmq_london`) quantum processor with $2^{10} = 1024$ shots, each with 5 variables and a circuit depth of p=1. All $C(\gamma, \beta)$ values dealing with shot-noise (qvm and qpu) are approximate in nature. For comparison, we include the exact ground state energy. The data in the figure is a result of the experiments described in Sections III C 2 and III C 3.

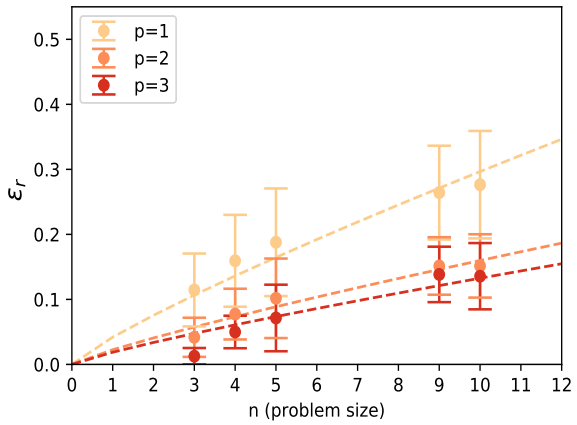


FIG. 2. Comparison of converged final relative error ϵ_r (median), for $p \in \{1, 2, 3\}$, as a function of problem size $n \in \{3, 4, 5, 9, 10\}$. Simulations performed with access to the exact wave form (statevector backend) done on 100 problem instances per n with an optimization budget of 200 iterations for $p \in \{1, 2\}$ and 400 iterations for $p = 3$. The dashed curves in the plot are fitted to the experimental data. The information presented in this figure is the output of experiments described in Section III C 1.

beyond 1 depending on the spectral width as compared to the groundstate cost.

1. *Optimization trajectory for QAOA*

In Figure 1, we see an example of how QAOA with ImFil performs on a BLLS problem. As the iterations progress, the fluctuations in the energy expectation value also reduces. This happens either till the black box optimizer converges to a solution (depending on default internal parameters in our case) or the iterations have reached the maximum threshold (governed by the budget). Here the experiments done with the statevector backend, which has access to the exact energy expectation value, sets the baseline for the other modes of experiments. While our simulations containing shot-noise due to measurement do relatively well against statevector results, the experiments on a real quantum device are mixed. At the time of writing, the IBM Q device we tested on did not approximate the theoretically-optimal QAOA result distribution very well, but it still finds the best solution every time. We have discussed this further in Section III D 6 and have included the results for reference.

2. *QAOA results with no-noise*

Before we study the results of QAOA for BLLS with shot-noise, it is important to evaluate the theoretical performance of the same without any noise at all. Figure 2 shows the relative error growth with respect to the problem size n for the experiments described in Section III C 1. We use Median as measure of central tendency and Median Absolute Deviation (MAD) for our error bars. Simulations larger than $p = 3$ take a lot more time for the complete dataset and were computationally infeasible for this project.

Using the information from the medians calculated, we attempt to fit curves for different values of p. Based on the curve fit, we find that the growth in ϵ_r to be polynomial in nature (as described by $\epsilon_r = a \times n^b$, where a and b are coefficients, see Appendix E for details). This polynomial growth may be partly attributed to some spin configurations with energies close to the ground state. You can see that going from $p = 1$ to $p = 2$ decreases the relative error moderately. The difference in performance between $p = 2$ and $p = 3$ is more modest, particularly for the larger problem sizes n. There may be room for further improvement if we allow a larger simulation time budget, for example by tightening the classical optimizer's convergence parameters and increasing the number of initial starting points for the optimizer. Further rigorous experimentation would be required to draw definite conclusions about the scaling based on such numerics.

3. *No noise vs Shot-noise optimization*

Figure 3 shows us how QAOA with ImFil performs for the parameters described in Section III C 2 for statevector and measurement based results for 2^{n+2} shots. We have 5 different problem instances per $n \in \{3, 4, 5, 9, 10\}$. The

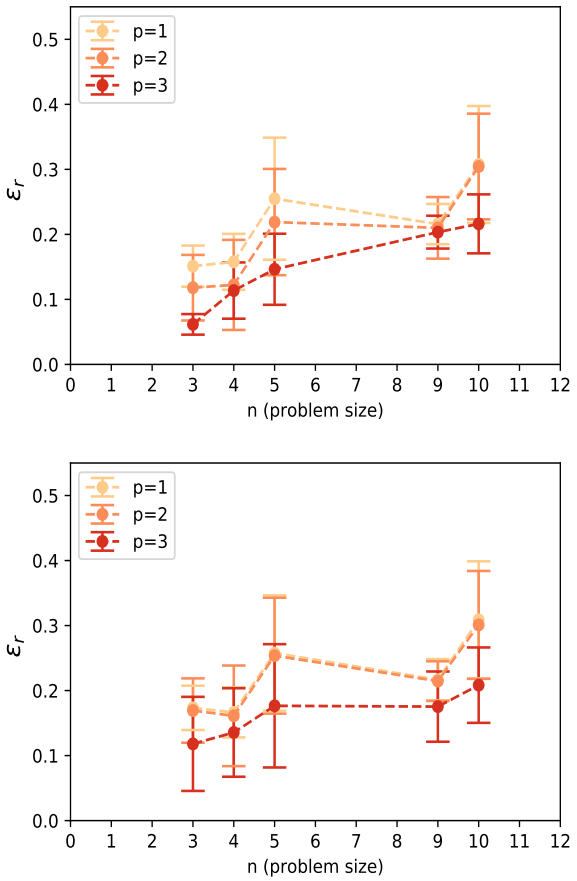


FIG. 3. Comparison of converged final relative error (median), for $p \in \{1, 2, 3\}$, as a function of problem size $n \in \{3, 4, 5, 9, 10\}$, done on 5 problem instances per n with a budget of 200 iterations. (TOP) shows the results from the optimization having access to the exact waveform (statevector simulator), while (BOTTOM) shows the results using a shot-noise simulator with 2^{n+2} shots. For drawing the bottom plot, we use the best angles found using shot-based optimization and used the statevector backend for one more run at those angles in order to compute the exact expectation value and corresponding relative error. These results are from the experiments described in Section III C 2.

reason for choosing 2^{n+2} shots for this comparison was to try and see if the optimizer could replicate the statevector results given plentiful shots. In subsequent figures, we'll be showing the performance of the optimizer with fewer number of shots.

We can see the similarities between the top and bottom plots in Figure 3. The main difference however, seems to be the result and error bar overlap between the results of p =1 and 2. While the two lines are close to each other in the statevector results uptil problem size of 9, the measurement-based results for the two parameters are extremely close to each other (when considering median and MAD). This could be attributed to the noise due to approximate results, or due to the small number of experiments we average over, as detailed in Section III C 2. Another effect of the smaller dataset here is that the relative error's growth doesn't seem fully monotonic to the problem size, unlike in Section III D 2. However, it still shows a general upward trajectory. Simulations for $p = 4$ and upwards become computationally infeasible due to the time required, even for the smaller dataset we worked with in Figure 3. This can be attributed to the exponential growth in runtime as a function of the circuit depth p [4].

4. *Probability of sampling the ground state*

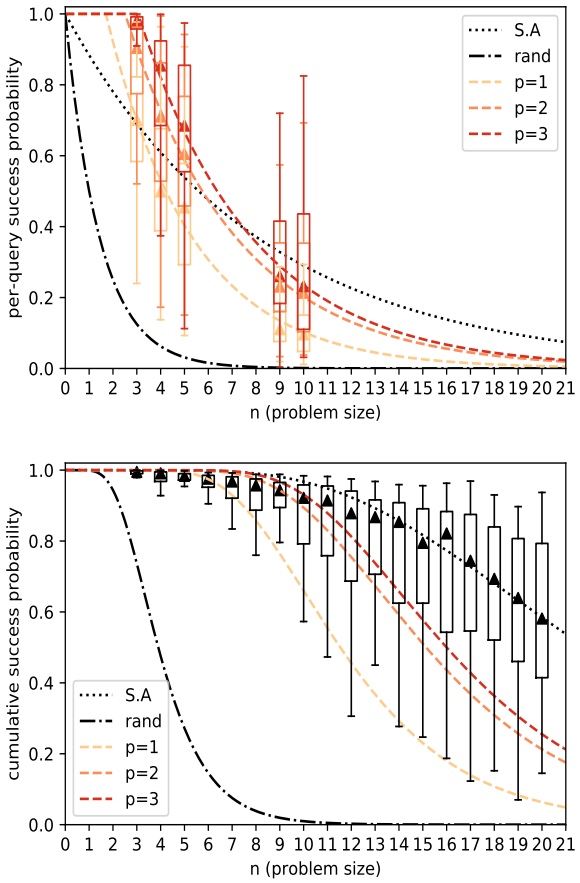


FIG. 4. Comparison of success probabilities of sampling the ground state, for QAOA at circuit depths $p = 1, 2, 3$, Simulated Annealing and random sampling. (TOP) shows the success probability per-query and (BOTTOM) shows the cumulative success probability after 10 queries. The box plots indicate the nature of the experimental results (with medians as triangles) while curves in the figure are fitted to them. The QAOA results are from the statevector (no noise) experiments as described in Section III C 1 and the approach for comparing it with Simulated Annealing and random sampling is described in Section III D 4.

We compare the probability of sampling the ground

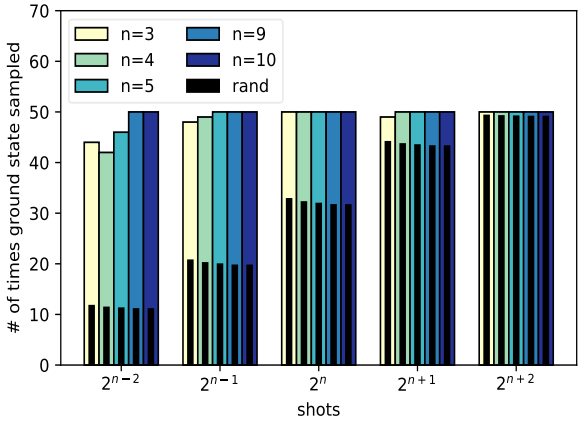


FIG. 5. In this bar graph, we collect the number of experiment instances in which we observe the exact ground state bitstring, at least once (on the Y axis) for $n \in \{3, 4, 5, 9, 10\}$. Each n has 5 problem instances, which is repeated 10 times for a given shot value. We compare the QAOA shot-noise simulator experiments (color-coded with the number of binary variables, n), with the results one would expect randomly sampling from a uniform distribution *shots* times (black bars, labeled with rand, x-axis positioning corresponding to its colour-labeled partner). The QAOA data in this figure comes from sampling the circuit with optimized angle sets γ^* and β^*, after acquiring them by optimizing for a circuit depth of $p = 3$ and a budget of 200 iterations, as described in Section III C 2.

state of the BLLS problem (henceforth also referred to as the success probability) for QAOA with two classical methods: random sampling and Simulated Annealing. For the Simulated Annealing experiments, we chose 10 steps, an exponential decay schedule, and ran it on problem sizes n from 3 to 20 (with 100 problems per n). For this problem size, Simulated Annealing is very cheap computationally on a classical computer, as compared to simulating the quantum circuit with a classical computer. This allows us to consider statistics of a larger number of random problems per n. See Appendix E for further details.

The success probabilities for QAOA are calculated from the outputs of the statevector experiments described in Section III C 1. More specifically, these would be the success probabilities from sampling the optimized waveform $|\psi(\gamma, \beta)\rangle$.

While the success probability of uniform random sampling is easy to calculate per query as $1/2^n$, things are not as straightforward with Simulated Annealing. The main reason being that Simulated Annealing requires k number of steps to arrive at a result. Assigning $k = 1$ would be equivalent to random sampling, but by this logic, assigning $k > 1$ would yield us a cumulative success probability for random sampling.

Thus, our proposed comparison method assumes an effective probability $\mathcal{P}_{\text{eff}}$ which is the success probability per query for Simulated Annealing. This $\mathcal{P}_{\text{eff}}$ is not directly observed, but calculated by extrapolation from Simulated Annealing run on $k > 1$ steps. After k steps, the cumulative success probability would be:

$$\text{cumulative success prob.} = 1 - (1 - \mathcal{P}_{\text{eff}})^k \tag{30}$$

Doing a curve fit modeled on Eqn(30) for the results of Simulated Annealing with $k = 10$ steps yields us the value of $\mathcal{P}_{\text{eff}}$ which would be the effective success probability, for Simulated Annealing, per query. Conversely, curve fits were done on the QAOA per query success probabilities and were extrapolated to cumulative probabilities for 10 queries. The resulting information is plotted in Figure 4 along with the box plots from the experiments. We refer the interested reader to Appendix E for the details.

As Figure 4 suggests, the success probability for all methods decrease exponentially as the size of the problem increases. The comparison of QAOA with uniform random sampling for BLLS corroborates with previous work done on applying QAOA to a different problem[9]. Even with p =1, QAOA performs much better than uniform random sampling. Simulated Annealing on BLLS however, yields a higher success probability than QAOA at circuit depth $p = 3$ when $n > 8$. This empirical result is supported by a recent theoretical work on QAOA for MAXCUT [82] that indicate that there exists classes of graphs for which quantum advantage is not possible for $p < 6$. This would support Simulated Annealing beating QAOA at $p = 3$ for BLLS, which has a highly dense graph structure [20, 21].

It is one thing to calculate probability, it is another to sample the best solution (or ground state) from a quantum state after QAOA. Figure 5 displays the number of experimental instances where we sample the ground state, at least once, for a particular set of parameters, across various problem sizes and instances (for shot-noise experiments in Section III C 2. We contrast this with the analytical results of getting the ground state by uniform random sampling. Here, we see that for optimization done with upto 2^n shots, QAOA has a clear advantage over random sampling. This can be explained by the mechanism of QAOA, which selectively amplifies those bitstring sampling probabilities which have the lowest energy, while suppressing those with higher energy. In this way, the success probabilities may be greatly enhanced over the naive random sampling from the uniform probability distribution.

5. *Effect of shot number on optimization*

For QAOA to become practical, the shot number chosen for the computation has to be far less than the number of eigenstates for our cost Hamiltonian (2^n for our case). For this work, we chose not to randomly guess a shot number value but rather get an understanding of the optimization performance for a set of shot numbers in $\{2^i | n - 2 \leq i \leq n + 2, i \in \mathbb{Z}\}$. We hope this helps

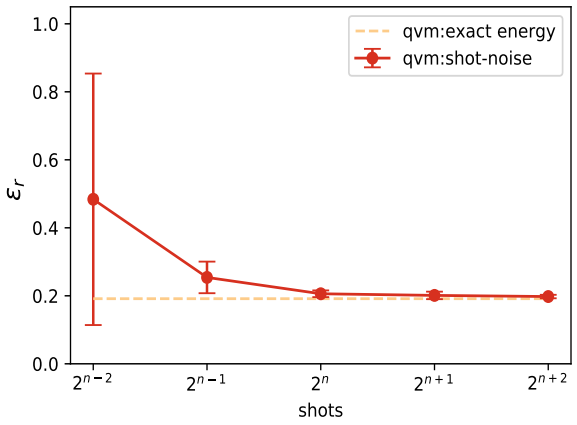


FIG. 6. Relative error of the optimized QAOA output distribution (y-axis), as a function of number of shots (x-axis), for a problem instance of the $n = 5$ case. Circles indicate median, and we plot the error-bars (using MAD) when approximating the expectation value with a finite number of shots. Data represented in the figure is an output of an experiment done in Section III C 2.

all future research work in finding better estimates for the least amount of shots required for QAOA, especially for these type of applications. In Figure 6 we see the optimization result for a problem instance where $n = 5$. The shot optimization is compared with the statevector optimization. Here it is important to point that we optimized using the stochastic blackbox (using a given shot number) and then calculate the exact expectation value using the wavefunction (i.e, with the statevector backend) in order to assess the true value of relative error. For the most part, our experiments show that as the problem size increases, we see the optimizer do well even with 2^{n-2} shots. This seems to indicate that the number of shots required to get a good optimization may not be exponential in terms of the problem size, or at least with a smaller exponent than applying random sampling from a uniform distribution. Further research is needed.

6. *IBM Q device performance*

We briefly mentioned our real device results in Section III D 1. The good news is that IBM Q was always able to find the best solution for our optimization experiments. But that came at the price of taking 1024 shots for each QAOA iteration, which is relatively expensive for a problem size of 5 qubits. When we lowered the shot number, the optimal bitstring was not always sampled and the convergence deteriorated further. The immediate cause of why the optimization process on the device was not close to the simulation results, is the inability of the device to approximate the distribution of the measured bitstrings (for a given circuit). Although this result is not surprising for QAOA on the current generation of cloud accessible quantum computing devices [83–86], it is important to emphasize on it particularly for readers from other domains who may be interested in QAOA for their own applications. For reference, we provide an example of this in Appendix D for the readers.

It is crucial to consider the entire context here. Firstly, the problem graph of our use case is fully connected. Due to the sparse connectivity of the on-chip qubits in the device, logical qubits have to be swapped around a number of times for them to be able to entangle with each other (for the ZZ interactions of our problem). This makes the average gate depth for the final (transpiled) circuits that run on the `ibmq_london` machine to be about 35. Since two qubit gate fidelity is still low (at the time of writing), the error propagates across the circuit. Secondly, due to the large circuit depth on the real device, we need to take decoherence into account. Thirdly, readout-errors were not considered here and they have significant impact on the noise in the qubit measurement results.

It should also be emphasized that in this work, we primarily focused on how to model the BLLS problem using QAOA. Thus, the experiments on the real devices were done "as is", in order to demonstrate the near-term implementability, without any error mitigation [87]. This could be looked at for future work.

E. Discussion

We can see the various possibilities and potential advantages QAOA may provide in solving BLLS and similar problems. However, there are challenges that need to be addressed. These are both theoretical and practical in nature.

One theoretical challenge is the proper pre-processing of the problem Hamiltonian by scaling and shifting the coefficients of the objective function, such that we optimally make use of the parameter space $\beta_l \in [0, \pi], \gamma_l \in [0, 2\pi]$ (most of the problems in the dataset did not suffer from this issue, as we found the default scaling to work well already). However, scaling the problem way beyond necessity also creates issues as the energy landscape is periodic in nature [4]. Thus, one possible way is to use scaling as a heuristic within the QAOA process, and treat it as a hyperparameter to optimize over.

Another challenge, which is both theoretical and practical in nature, is the full connectivity in our problem and in most hard optimization problems in general [88].Computationally non-trivial problems typically require a high degree of graph connectivity (for instance, planar graphs are easy to solve classically [89]). Simultaneously, a high connectivity poses a challenge in quantum chip implementation because not all gatesets implement non-nearest neighbour interactions natively. Those need then be implemented effectively by means of a swap network approach [86]. For future work, one option is to modify the problem formulation by not considering the ZZ interactions of a pair of qubits, if its coefficient's mag-

nitude falls below a user defined threshold. This can potentially make it easier for QAOA to run, but its effectiveness in finding the ground state would come under scrutiny. Nonetheless, it can make for interesting future work. Also, error mitigation techniques and readout error correction will also help in improving the results [86]. It would take a combination of the above mentioned approaches to improve performance on a real device.

Challenges aside, one of the next steps would be to explore if QAOA can be valuable in applications that require BLLS as a subroutine, such as NBMF [21]. Another step can be to try the BLLS problem on other types of quantum computers [9–13] to see how different hardware implementations fare. Finally, from a practical standpoint, it may also be beneficial to apply a modified QAOA-like ansatz based on just nearest-neighbor connected CNOT gates, along with a larger number of parameters [90, 91], or apply a SWAP-network QAOA approach [86]. This may potentially lead to faster convergence and better qpu performance for architectures with limited connectivity.

IV. CONCLUSION

In this work, we described the implementation of a binary optimization problem, relevant to hard problems in linear algebra, on a gate-based quantum computer via a QAOA approach suitable for NISQ devices. In our simulations, we show how Simulated Annealing can outperform QAOA for $p \leq 3$ for problem size $n > 8$ in terms of sampling the ground state for BLLS type problems. We show that the ImFil optimizer backend performs well under shot noise for this problem type. From our experiments on the IBM Q cloud-based quantum processor, we conclude that that it is still challenging to implement linear-depth, high-connectivity circuits on the latest hardware available as of the time of this work. We expect a future experimental implementation would benefit greatly from gate-error mitigation techniques and post-processing readout errors. It would furthermore be very interesting to see what other hard problems in linear algebra may be implemented using the QAOA ansatz and what their expected performance would be.

ACKNOWLEDGMENT

The authors would like to thank the people who made this collaboration possible. From UMBC: Dean Drake, Wendy Martin and Cameron McAdams from the Office of the Vice President for Research (OVPR), the Office of Technology Development (OTD) and the Office of Sponsored Programs (OSP) respectively. From Qu & Co: we'd like to thank Benno Broer, CEO and co-founder of Qu & Co. This work was performed in part using IBM Quantum systems as part of the IBM Q Network.

Appendix A: Detailed QUBO Formulation for Binary Linear Least Squares

In this section of the Appendix, we describe the method by which O'Malley and Vesselinov [20] formulated the binary linear least squares (BLLS) problem. This QUBO formulation will be converted into its equivalent Ising objective function and used in QAOA. Let us begin by writing out $Ax - b$ which would help us in minimizing x and thereby solve Eqn(1)

$$Ax - b = \begin{pmatrix} A_{11} & A_{12} & ... & A_{1n} \\ A_{21} & A_{22} & ... & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & ... & A_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{A1}$$

$$Ax - b = \begin{pmatrix} A_{11}x_1 + A_{12}x_2 + ... + A_{1n}x_n - b_1 \\ A_{21}x_1 + A_{22}x_2 + ... + A_{2n}x_n - b_2 \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + ... + A_{mn}x_n - b_m \end{pmatrix} \tag{A2}$$

Taking the 2 norm square of the resultant vector of Eqn(A2), we get

$$\|Ax - b\|_2^2 = \sum_{i=1}^{m} (|A_{i1}x_1 + A_{i2}x_2 + ... + A_{in}x_n - b_i|)^2 \tag{A3}$$

Because we are dealing with real numbers for A and b, $(|.|)^2 = (.)^2$

$$\|Ax - b\|_2^2 = \sum_{i=1}^{m} (A_{i1}x_1 + A_{i2}x_2 + ... + A_{in}x_n - b_i)^2 \tag{A4}$$

And thus, the coefficients in Eqn(2) are found by expanding Eqn(A4) to be

$$v_j = \sum_i A_{ij}(A_{ij} - 2b_i) \tag{A5}$$

$$w_{jk} = 2\sum_i A_{ij}A_{ik} \tag{A6}$$

You will notice that there is a constant value from Eqn(A4) that we leave out of Eqn(A5) and Eqn(A6). Because this value is not a coefficient for any of the variables, we can't optimize over it and it's left as is, which is $\|b\|_2^2$. Also, the ground state energy (QUBO) for when $\|Ax^* - b\|_2 = 0$ where x^* is the best solution, is $-\|b\|_2^2$.

Appendix B: Example QAOA Circuit for BLLS

Let us consider a simple problem, without loss of generality, to demonstrate how quantum circuits for BLLS

can be designed. Consider the following problem, Find $x \in \{0,1\}^3$ such that $\|Ax - b\|$ is minimized, where

$$A = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 1 & -1 \\ 1 & 2 & 3 \end{pmatrix} \tag{B1}$$

$$\text{and } b = \begin{pmatrix} 3 \\ 0 \\ 3 \end{pmatrix} \tag{B2}$$

This particular problem is more appropriately categorized as a linear system of equations ($A \in \mathbb{R}^{n^2}$) and has a solution $x^* = (1,1,0)^T$, such that $Ax^* = b$ or $\|Ax^* - b\|_2 = 0$. However, our problem formulation does not change.

In order to solve this problem using QAOA, we require 3 qubits. Using the formulation process detailed in the Appendix A and Eqn(12), the QUBO formulation we get is

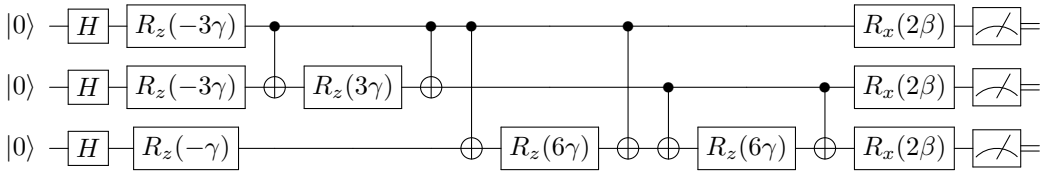
$$F(x) = -12x_1 - 12x_2 - 13x_3 + 6x_1x_2 + 12x_1x_3 + 12x_2x_3 \tag{B3}$$

The constant value that didn't make it to the QUBO here is $\|b\|_2^2 = 18$. Converting the QUBO into Ising using Eqn(16), we get

$$F(\sigma) = -1.5\sigma_1 - 1.5\sigma_2 - 0.5\sigma_3 + 1.5\sigma_1\sigma_2 + 3\sigma_1\sigma_3 + 3\sigma_2\sigma_3 \tag{B4}$$

The offset when going from QUBO to Ising is -11. Therefore, the Ising ground state for this problem is $-18 - (-11) = -7$. Now let us assume that we are designing a circuit for QAOA where $p = 1$. So for a given pair of angles (β, γ) a circuit would look like the one shown down below in Eqn(B5).

The results of this circuit will be used to calculate the expectation value $C(\gamma, \beta)$. Based on the expectation value, a new pair of β and γ will be calculated using a classical black box optimization algorithm (like ImFil). These new angles will be fed into another such circuit till the optimization loop converges.



(B5)

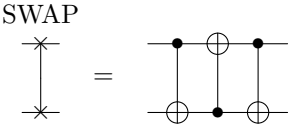
The results from Eqn (B5) would be classical bitstrings when measured in the standard basis. In order to calculate the energy or cost of a particular bitstring with respect to the Ising Cost function Eqn(3), we would first need to substitute a 1 for each 0 and -1 for each 1 in the bitstring. In short, this is because $\hat{\sigma}^{(z)}$ describes a quantum state to have an energy of +1 for $|0\rangle$ and -1 for $|1\rangle$ (in arbitrary units). For our example, if we measure a bitstring ξ to be $\{\xi_1 = 0, \xi_2 = 0, \xi_3 = 1\}$ the equivalent ising set would be $\{\sigma_1 = 1, \sigma_2 = 1, \sigma_3 = -1\}$. Using Eqn(B4), we get back an energy of -7. This way, we can calculate $C(\gamma, \beta)$ for diagonal $\hat{C}$ Hamiltonians by averaging the energies of all measured bitstrings.

Appendix C: Implementing Two-Qubit Interactions on a QPU

As mentioned in Section III B 1, most practical quantum computers would not have all to all qubit connectivity. But if the problem that we need to solve on a quantum device requires dense connectivity (such as the BLLS), we need SWAP gates for allowing distant qubits to interact with one another. Let us first describe the SWAP gate in its matrix form

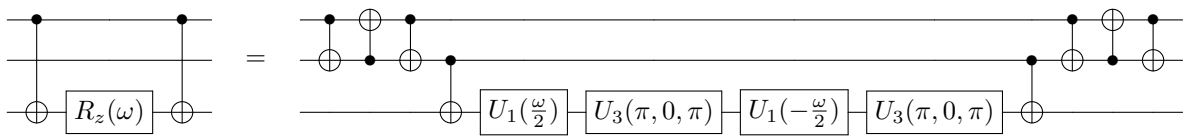
$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{C1}$$

Diagrammatically, we can decompose it with CNOT gates as



(C2)

To illustrate how this takes place, consider a hypothetical device where every qubit is only connected to the adjacent qubit in a line. Thus in order to realize the gates described on the LHS of Eqn(C3), one way is to SWAP between the top and the middle qubit so that it could interact with the bottom qubit.



(C3)

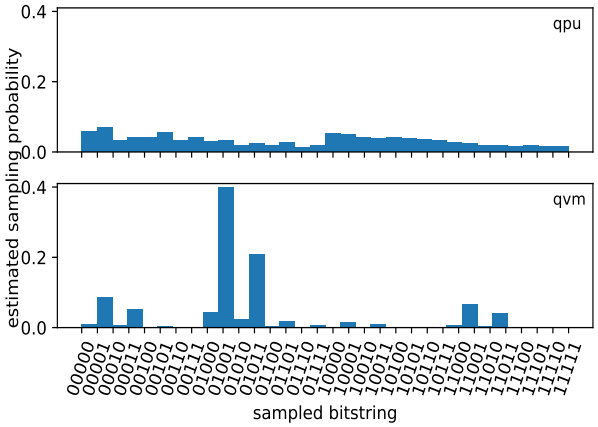


FIG. 7. A typical instance of QAOA-circuit optimized probability distributions of the measured qubit bitstring output. We compare the result for IBM Q device (TOP) versus the qiskit shot-based simulator (BOTTOM), for the same problem instance for a total of 10240 shots. This figure represents data from an experiment done in Section III C 3.

After our desired two-qubit interaction takes place, the top and middle qubits are swapped again, returning the logical qubits to their original place. Of course, there are other methods (involving SWAP) that the compiler may take to realize the original unitary operations to be performed. In Eqn(C3), we also show the decomposition of the R_z gate as defined in Eqn(28).

Appendix D: Probability Distribution of Running a QAOA Circuit for BLLS on a Real QPU

At the time of writing, our experiments on the quantum processor (`ibmq_london`) are unable to produce results similar to what a simulator produces. Figure 7 shows an example of the distribution we get from running a QAOA circuit with optimized (and fixed) β and γ angles. The simulation suggests a few bitstrings with a high probability of being measured (with the ground state having the highest), whereas the distribution from the quantum processor is more evenly spread out, with the ground state not having a significantly high probability of being measured.

TABLE I. Curve fit details for various QAOA circuit depths from Figure 2 where the curve modeled is of the type: $a \times n^b$ for $b = 0.85$. All numerical values are rounded to a precision of 4 decimal places.

Circuit Depth	Coeff. a	Fit Error[aa]
$p = 1$	0.0419	0.0852
$p = 2$	0.0226	0.0911
$p = 3$	0.0187	0.1911

[aa] rel. error of the output produced by the curve wrt observed data

Appendix E: Curve fit and Simulated Annealing details

1. Curve fit

In this part, we describe the details about the curve fits that appear in Figure 2 and 4. We found that for Figure 2, the polynomial curve of type $a \times n^b$ for $b = 0.85$ fits the data moderately well, Table I shows further information. We can see from Table I, that the fit error (a relative error value) doubles when we go from $p = 2$ to $p = 3$. However, when we explored other values for the exponent, we discovered that $b \approx 1.26$ reduces the fit error of circuit depth $p = 3$ the most to about 0.13, but increases the error fit of $p = 1$ and 2 to about 0.22 and 0.17 respectively. On the other end, a value of $b \approx 0.65$ reduces the fit error for $p = 1$ the most to about 0.04 but raises $p = 2$ and $p = 3$ to around 0.12 and 0.26 respectively. Thus the $b = 0.85$ value is a compromise that works the best for all circuit depths we consider in this paper. Further research on more problem sizes n may help to fit better growth curves and get a more definitive value for the exponent b.

The per query and cumulative success probability curves for QAOA and Simulated Annealing as seen in Figure 4 fits the curve type $1 - (1 - a/2^{bn})^k$ well (which turns out to be just $a/2^{bn}$ for $k = 1$). For Simulated Annealing, we fix $a = 1$ such that $\mathcal{P}_{\text{eff}} \to 1$ as $n \to 0$, this is because $a \approx 0.63$ when its value is calculated in the curve fitting process, which prevents $\mathcal{P}_{\text{eff}}$ being 1 at $n = 0$. The increase in error when $a = 1$ is negligible for the Simulated Annealing curve. However, it is not so negligible in the case of QAOA, and therefore we don't fix the value of a to 1. This results in the per-query curves to go above 1 and cumulative curves to be less than 1 (after reaching 1) for $n < 3$. We consider this to be a

TABLE II. Curve fit details for various QAOA circuit depths from Figure 4 where the curve modeled is of the type: $1-(1-a/2^{bn})^k$ at $k=1$ for QAOA and $k=10$ for Simulated Annealing (and extrapolated to $k=10$ and $k=1$ respectively for plotting purposes). All numerical values are rounded to a precision of 4 decimal places.

Optimization Method	Coeff. a	Coeff. b	Fit Error[aa]
QAOA, $p=1$	1.5968	0.3967	0.0655
QAOA, $p=2$	1.7127	0.3090	0.0247
QAOA, $p=3$	1.8926	0.3013	0.0380
Sim. Anneal[bb]	0.6359	0.1397	0.0217
Sim. Anneal[cc]	1	0.1786	0.0312

[aa] rel. error of the output produced by the curve wrt observed data
[bb] First attempt
[cc] Chosen for plot

modeling artifact and fix the success probability to 1 for $n<3$ for both cases.

2. Simulated Annealing Experiments

The Simulated Annealing results referenced in Section III D 4, Figure 4 and Table II were conducted for BLLS problems of sizes 3 to 20. An initial and final temperature of $T_0 = 100$ and $T_f = 0.01$ (in arbitrary units) were chosen respectively. We define and use a exponential decay schedule with temperature at iteration i as

$$T_i = T_0\left(\frac{T_f}{T_0}\right)^{i/k} \tag{E1}$$

Each problem was run 1000 times with a set of randomization seeds for initialization of the Monte Carlo process. Each Simulated Annealing run was done with $k=10$ steps. Our (cumulative) success probability is the count of all the runs where we end up with the ground state divided by 1000.

[1] M. A. Nielsen and I. Chuang, Quantum computation and quantum information (2002).
[2] E. G. Rieffel and W. H. Polak, *Quantum computing: A gentle introduction* (MIT Press, 2011).
[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv preprint quant-ph/0001106 (2000).
[4] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).
[5] J. Preskill, Quantum computing in the nisq era and beyond, Quantum **2**, 79 (2018).
[6] T. Kadowaki and H. Nishimori, Quantum annealing in the transverse ising model, Physical Review E **58**, 5355 (1998).
[7] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk, Practical annealing-based quantum computing, Computer **52**, 38 (2019).
[8] A. Cross, The ibm q experience and qiskit open-source quantum computing software, Bulletin of the American Physical Society **63** (2018).
[9] J. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, *et al.*, Unsupervised machine learning on a hybrid quantum computer, arXiv preprint arXiv:1712.05771 (2017).
[10] K. Wright, K. Beck, S. Debnath, J. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. Pisenti, M. Chmielewski, C. Collins, *et al.*, Benchmarking an 11-qubit quantum computer, Nature Communications **10**, 1 (2019).
[11] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).
[12] J. Gaebler, B. Bjork, D. Stack, M. Swallows, M. Fabrikant, A. Reed, B. Spaun, J. Pino, J. Dreiling, and C. Figgatt, Progress toward scalable quantum computing at honeywell quantum solutions, Bulletin of the American Physical Society **64** (2019).
[13] T. Last, N. Samkharadze, P. Eendebak, R. Versluis, X. Xue, A. Sammak, D. Brousse, K. Loh, H. Polinder, G. Scappucci, *et al.*, Quantum inspire: Qutech's platform for co-development and collaboration in quantum computing, in *Novel Patterning Technologies for Semiconductors, MEMS/NEMS and MOEMS 2020*, Vol. 11324 (International Society for Optics and Photonics, 2020) p. 113240J.
[14] S. Chrétien and F. Corset, Least squares reconstruction of binary images using eigenvalue optimization, in *Compstat* (Springer, 2002) pp. 419–424.
[15] S. Chrétien and F. Corset, Using the eigenvalue relaxation for binary least-squares estimation problems, Signal processing **89**, 2079 (2009).
[16] E. Tsakonas, J. Jaldén, and B. Ottersten, Robust binary least squares: Relaxations and algorithms, in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2011) pp. 3780–3783.
[17] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors, IEEE Transactions on Information Theory **59**, 2082 (2013).
[18] A. Bourquard and M. Unser, Binary compressed imaging, IEEE Transactions on Image Processing **22**, 1042 (2012).
[19] Explained in Section II A 1.
[20] D. O'Malley and V. V. Vesselinov, Toq. jl: A high-level programming language for d-wave machines based on julia, in *2016 IEEE High Performance Extreme Computing Conference (HPEC)* (IEEE, 2016) pp. 1–7.
[21] D. O'Malley, V. V. Vesselinov, B. S. Alexandrov, and L. B. Alexandrov, Nonnegative/binary matrix factorization with a d-wave quantum annealer, PloS one **13**, e0206653 (2018).
[22] D. Ottaviani and A. Amendola, Low rank non-negative matrix factorization with d-wave 2000q, arXiv preprint

arXiv:1808.08721 (2018).
[23] R. Ayanzadeh, S. Mousavi, M. Halem, and T. Finin, Quantum annealing based binary compressive sensing with matrix uncertainty, arXiv preprint arXiv:1901.00088 (2019).
[24] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nature communications **5**, 4213 (2014).
[25] C. C. Chang, A. Gambhir, T. S. Humble, and S. Sota, Quantum annealing for systems of polynomial equations, Scientific reports **9**, 1 (2019).
[26] A. Borle and S. J. Lomonaco, Analyzing the quantum annealing approach for solving linear least squares problems, in *International Workshop on Algorithms and Computation* (Springer, 2019) pp. 289–301.
[27] T. H. Chang, T. C. Lux, and S. S. Tipirneni, Least-squares solutions to polynomial systems of equations with quantum annealing, Quantum Information Processing **18**, 374 (2019).
[28] N. Gillis, Introduction to nonnegative matrix factorization, arXiv preprint arXiv:1703.00663 (2017).
[29] A. Shashua and T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, in *Proceedings of the 22nd international conference on Machine learning* (ACM, 2005) pp. 792–799.
[30] O. Berne, C. Joblin, Y. Deville, J. Smith, M. Rapacioli, J. Bernard, J. Thomas, W. Reach, and A. Abergel, Analysis of the emission of very small dust particles from spitzer spectro-imagery data using blind signal separation methods, Astronomy & Astrophysics **469**, 575 (2007).
[31] F. Å. Nielsen, D. Balslev, and L. K. Hansen, Mining the posterior cingulate: segregation between memory and pain components, Neuroimage **27**, 520 (2005).
[32] C.-J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural computation **19**, 2756 (2007).
[33] D. Chen and R. J. Plemmons, Nonnegativity constraints in numerical analysis, in *The birth of numerical analysis* (World Scientific, 2010) pp. 109–139.
[34] F. Glover, G. Kochenberger, and Y. Du, A tutorial on formulating and using qubo models, arXiv preprint arXiv:1811.11538 (2018).
[35] B. O'Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy, Bayesian network structure learning using quantum annealing, The European Physical Journal Special Topics **224**, 163 (2015).
[36] F. Neukart, G. Compostella, C. Seidel, D. Von Dollen, S. Yarkoni, and B. Parney, Traffic flow optimization using a quantum annealer, Frontiers in ICT **4**, 29 (2017).
[37] E. Farhi and A. W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, arXiv preprint arXiv:1602.07674 (2016).
[38] G. Gallavotti, *Statistical mechanics: A short treatise* (Springer Science & Business Media, 2013).
[39] B. A. Cipra, The ising model is np-complete, SIAM News **33**, 1 (2000).
[40] C. C. McGeoch and C. Wang, Experimental evaluation of an adiabiatic quantum system for combinatorial optimization, in *Proceedings of the ACM International Conference on Computing Frontiers* (ACM, 2013) p. 23.
[41] S. Aaronson, Guest column: Np-complete problems and physical reality, ACM Sigact News **36**, 30 (2005).
[42] This is true as long as all possible variable combinations have obj. function costs that are ≥ 1 in magnitude.
[43] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury, Simulation optimization: a review of algorithms and applications, Annals of Operations Research **240**, 351 (2016).
[44] L. M. Rios and N. V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, Journal of Global Optimization **56**, 1247 (2013).
[45] C. T. Kelley, *Implicit filtering*, Vol. 23 (SIAM, 2011).
[46] J. Močkus, On bayesian methods for seeking the extremum, in *Optimization techniques IFIP technical conference* (Springer, 1975) pp. 400–404.
[47] A. Mayer, T. Mora, O. Rivoire, and A. M. Walczak, Diversity of immune strategies explained by adaptation to pathogen statistics, Proceedings of the National Academy of Sciences **113**, 8630 (2016).
[48] J. C. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, IEEE Transactions on aerospace and electronic systems **34**, 817 (1998).
[49] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters **103**, 150502 (2009).
[50] N. Wiebe, D. Braun, and S. Lloyd, Quantum algorithm for data fitting, Physical review letters **109**, 050505 (2012).
[51] B. D. Clader, B. C. Jacobs, and C. R. Sprouse, Preconditioned quantum linear system algorithm, Physical review letters **110**, 250504 (2013).
[52] I. Kerenidis and A. Prakash, Quantum recommendation systems, arXiv preprint arXiv:1603.08675 (2016).
[53] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM Journal on Computing **46**, 1920 (2017).
[54] H. Wang and H. Xiang, Quantum algorithm for total least squares data fitting, Physics Letters A **383**, 2235 (2019).
[55] I. Kerenidis and A. Prakash, Quantum gradient descent for linear systems and least squares, Physical Review A **101**, 022316 (2020).
[56] Y. Tong, D. An, N. Wiebe, and L. Lin, Fast inversion, preconditioned quantum linear system solvers, and fast evaluation of matrix functions, arXiv preprint arXiv:2008.13295 (2020).
[57] S. Aaronson, Read the fine print, Nature Physics **11**, 291 (2015).
[58] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum random access memory, Physical review letters **100**, 160501 (2008).
[59] V. Giovannetti, S. Lloyd, and L. Maccone, Architectures for a quantum random access memory, Physical Review A **78**, 052310 (2008).
[60] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Quantum machine learning: a classical perspective, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**, 20170551 (2018).
[61] A. Montanaro and S. Pallister, Quantum algorithms and the finite element method, Physical Review A **93**, 032324

(2016).
[62] J. Golden and D. O'Malley, Reverse annealing for nonnegative/binary matrix factorization, arXiv preprint arXiv:2007.05565 (2020).
[63] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, How" quantum" is the d-wave machine?, arXiv preprint arXiv:1401.7087 (2014).
[64] T. ALBASH and I. HEN, Future of physical quantum annealers: Impediments and hopes, SCIENCE AND CULTURE (2019).
[65] F. Glover, Tabu search: A tutorial, Interfaces **20**, 74 (1990).
[66] L. Gurobi Optimization, Gurobi optimizer reference manual (2020).
[67] S. Mandra, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches, Physical Review A **94**, 022337 (2016).
[68] C.-C. Chen, S.-Y. Shiau, M.-F. Wu, and Y.-R. Wu, Hybrid classical-quantum linear solver using noisy intermediate-scale quantum machines, Scientific reports **9**, 1 (2019).
[69] X. Xu, J. Sun, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, Variational algorithms for linear algebra, arXiv preprint arXiv:1909.03898 (2019).
[70] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. Coles, Variational quantum linear solver: A hybrid algorithm for linear systems, Bulletin of the American Physical Society **65** (2020).
[71] X. Wang, Z. Song, and Y. Wang, Variational quantum singular value decomposition, arXiv preprint arXiv:2006.02336 (2020).
[72] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms **12**, 34 (2019).
[73] V. Choi, Minor-embedding in adiabatic quantum computation: I. the parameter setting problem, Quantum Information Processing **7**, 193 (2008).
[74] Z.-C. Yang, A. Rahmani, A. Shabani, H. Neven, and C. Chamon, Optimizing variational quantum algorithms using pontryagin's minimum principle, Physical Review X **7**, 021027 (2017).
[75] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Bulletin of the American Physical Society **64** (2019).
[76] L. T. Brady, C. L. Baldwin, A. Bapat, Y. Kharkov, and A. V. Gorshkov, Optimal protocols in quantum annealing and qaoa problems, arXiv preprint arXiv:2003.08952 (2020).
[77] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, *et al.*, Quantum algorithm implementations for beginners, arXiv preprint arXiv:1804.03719 (2018).
[78] D. Rose, *Sparse Matrices and their Applications: Proceedings of a Symposium on Sparse Matrices and Their Applications, held September 9 10, 1971, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, the National Science Foundation, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department.* (Springer Science & Business Media, 2012).
[79] D. Chung and S. Keles, Sparse partial least squares classification for high dimensional data, Statistical applications in genetics and molecular biology **9** (2010).
[80] S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, Presto: distributed machine learning and graph processing with sparse matrices, in *Proceedings of the 8th ACM European Conference on Computer Systems* (2013) pp. 197–210.
[81] H. Abraham, AduOffei, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, G. Alexandrowics, E. Arbel, A. Asfaw, C. Azaustre, AzizNgoueya, P. Barkoutsos, G. Barron, L. Bello, Y. Ben-Haim, and D. B. et al., Qiskit: An open-source framework for quantum computing (2019).
[82] J. Wurtz and P. J. Love, Bounds on maxcut qaoa performance for p¿ 1, arXiv preprint arXiv:2010.11209 (2020).
[83] M. Alam, A. Ash-Saki, and S. Ghosh, Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits, arXiv preprint arXiv:1907.09631 (2019).
[84] B. K. Behera, P. K. Panigrahi, *et al.*, Solving vehicle routing problem using quantum approximate optimization algorithm, arXiv preprint arXiv:2002.01351 (2020).
[85] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, Benchmarking the quantum approximate optimization algorithm, Quantum Information Processing **19**, 197 (2020).
[86] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, *et al.*, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, arXiv preprint arXiv:2004.04197 (2020).
[87] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Error mitigation extends the computational reach of a noisy quantum processor, Nature **567**, 491 (2019).
[88] V. Akshay, H. Philathong, M. Morales, and J. Biamonte, Reachability deficits in quantum approximate optimization, Physical Review Letters **124**, 090504 (2020).
[89] F. Hadlock, Finding a maximum cut of a planar graph in polynomial time, SIAM Journal on Computing **4**, 221 (1975).
[90] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).
[91] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, Improving variational quantum optimization using cvar, Quantum **4**, 256 (2020).