Two-round QAOA applied to the Minimum Vertex Cover Problem

Alan Gabriel Martins Silva, Marcus Henrique Soares Mendes, José Augusto Miranda Nacif, Leonardo Antônio Mendes Souza e Romeu Rossi Junior

***Abstract*— Nondeterministic polynomial time (NP) problems remain a major challenge for classical computing. Quantum computing promises to overcome this barrier using quantum properties. Among the most studied approaches is the Quantum Approximate Optimization Algorithm (QAOA), a quantum-classical hybrid. This paper shows how the Minimum Vertex Cover Problem (MVCP) can be tackled via QAOA on a quantum simulator, proposing a two-round strategy: first to select a good starting point, then to optimize the solution. Tested with different classical algorithms and simulation settings, our method is evaluated with different circuit depths. Results show high probabilities of optimal solutions, even at low depths.**

***Keywords*— Quantum Approximate Optimization Algorithm, Minimum Vertex Cover, Quantum Computer Simulator.**

I. Introduction

In computer theory, problems of the class NP(nondeterministic polynomial time) are problems with no known polynomial time methods to be solved, but that can be verified in such time complexity. Those types of problems are a barrier in modern day computation, even for supercomputers [1]. Several modern day problems with real applications, such as the Traveling Salesman Problem [2] and the Job Shop Scheduling Problem [3] are examples of NP problems [4][5]. The Minimum Vertex Cover Problem (MVCP) also lies in this class of problems, being NP-complete [4], and many heuristic methods to find near-optimal solutions for it have been widely used, such as annealing, genetic, DNA, particle swarm, and many others algorithms [6]. With advances in quantum physics and computer theory, some new methods of computation beyond the classical computer have been proposed using the principles of quantum physics [7]. From that moment - the advent of Quantum Computing - some algorithms were formulated aiming to improve already existing classical analogs, for instance, the famous Grover's [8] search algorithm and Shor's [9] algorithm for number factoring. Another promising quantum algorithm is the Quantum Approximate Optimization Algorithm(QAOA) [10], a quantum-classical hybrid algorithm to solve combinatorial optimization problems - such as the MVCP - that combines a quantum circuit and a classical optimizer to find an approximate solution to the given problem. In this paper, a formulation of the MVCP problem to be solved by the QAOA is proposed, implemented by following the usual methods to obtain the Quadratic Unconstrained Binary Optimization(QUBO) related to the problem [11] and the creation of an Ising model and a quantum circuit associated to the QUBO formulation [12]. Afterwards, the QAOA routine is applied to the problem using two rounds of the QAOA, one to select a preferred starting point among some sampled ones and a second, starting from the select starting position, to execute the final parameter angles optimization. The quantum circuit was run in different quantum computer simulators and several classical optimizers to complete the quantum-classical loop. For each optimizer, the circuit was constructed with different depths. The results from each run are then analyzed, by comparing relevant data such as the approximation ratio of the results and probability of finding the optimal solution and the number of iterations to convergence for each classical optimizer.

Alan Gabriel Martins Silva, Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Florestal-MG, e-mail: alan.g.silva@ufv.br; Marcus Henrique Soares Mendes, Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Florestal-MG, e-mail: marcus.mendes@ufv.br; José Augusto Miranda Nacif, Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Florestal-MG, e-mail: jnacif@ufv.br; Leonardo Antônio Mendes Souza, Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Florestal-MG, e-mail: leonardoamsouza@ufv.br; Romeu Rossi Junior, Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Florestal-MG, e-mail: romeu.rossi@ufv.br

II. The Minimum Vertex Cover Problem and its Implementation in QAOA

For the context of this paper, the MVCP problem is defined as the following: for any given unoriented graph, without loops and no multiple edges $G(V, E)$, where V and E are its vertices and edges(a set of unordered pairs $\{v, u \in V\}$), respectively, a vertex cover for it is a set of vertices $V' \in V$ where $\forall\ e \in E, v_e \vee u_e \in V'$, that is, for every edge, at least one of its endpoints are in V'. Thus, the Minimum Vertex Cover is the vertex cover V' with the minimum number of vertices, V^*. This problem can be described as:

$$\begin{aligned} \min \quad & \sum_{i=0}^{|V|} x_i, \\ \text{subject to} \quad & x_i + x_j \geq 1, \forall \{x_i, x_j\} \in E \end{aligned} \tag{1}$$

Where x_n represents the vertices of the graph, with $x_n = 1$ if the vertex $v_n \in V'$ and $x_n = 0$ otherwise.
In order to solve this problem in a quantum computer using QAOA, first, it is required to rewrite and abstract is formulation so it can be understood and solved by a quantum processor, in the case of this paper, a gate based quantum computer. To do so, the MVCP is written as a QUBO problem (equation 2), as it easily translates to a Ising Hamiltonian (equation 3), which is the Hamiltonian (energy) function

of a mathematical model describing quantum interaction of particles organized in a lattice.

$$f(\vec{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=i}^{n} Q_{ij}x_i x_j \tag{2}$$

$$H = -\sum_{j,k} J_{jk} z_j z_k - \sum_{j} h_j z_j \tag{3}$$

The Hamiltonian of the problem is then used to build a quantum circuit in the quantum computer, which will most likely prepare a state close or equal to the system ground state, which will give the solution to the original problem. To write the MVCP as a QUBO problem, the hard constraints are transformed into soft constraints (penalizations) and added to the objective function [11], as the following:

$$\min \quad A\sum_{i,j\in E}(1-x_i)(1-x_j) + B\sum_{i=0}^{|V|} x_i \tag{4}$$

Here, A and B are weights for the penalization and the objective, respectively, and $A > B$ [11]. We can then write it in the Ising model Hamiltonian from 3 as the problem Hamiltonian H_C by changing its bit variables $\{0, 1\}$ to spin variables $\{1, -1\}$, having:

$$\begin{aligned}
H_C &= A\sum_{i,j\in E}\left(1-\frac{1-z_i}{2}\right)\left(1-\frac{1-z_j}{2}\right) + B\sum_{i=1}^{|V|}\frac{1-z_i}{2} \\
&= A\sum_{i,j\in E}\left(\frac{1+z_i}{2}\right)\left(\frac{1+z_j}{2}\right) + B\sum_{i=1}^{|V|}\frac{1-z_i}{2} \\
&= \frac{A}{4}\sum_{i,j\in E}\left(1+z_i+z_j+z_iz_j\right) + \frac{B}{2}\sum_{i=1}^{|V|}(1-z_i) \\
&= \frac{A}{4}\sum_{i,j\in E} z_iz_j + \frac{A}{4}\sum_{i,j\in E}(z_i+z_j) + \frac{A\,|E|}{4} \\
&\quad - \frac{B}{2}\sum_{i=1}^{|V|} z_i + \frac{B\,|V|}{2} \\
&= \frac{A}{4}\sum_{i,j\in E} z_iz_j \; + \; \frac{A}{4}\sum_{i=1}^{|V|} d_i z_i - \frac{B}{2}\sum_{i=1}^{|V|} z_i \\
&= \frac{A}{4}\sum_{i,j\in E} z_iz_j + \frac{1}{4}\sum_{i=1}^{|V|}\left(Ad_i - 2B\right) z_i
\end{aligned} \tag{5}$$

Where d_i is the degree of the vertex v_i. Having the problem written as such, it is possible to assemble the quantum circuit from it, as the one in Figure 2, and then solve the Minimum Vertex Cover Problem using QAOA. What the QAOA will do is prepare a initial quantum state using the problem and the mixing (an easy to solve) Hamiltonians H_C and H_B of $2p$ rotation angles variables γ and β, where p is the depth of the quantum circuit, that is, how many times the circuit pattern is repeated. Then, the energy $F(\gamma, \beta)$ of this state is calculated, and the values of γ and β are updated, which will be fed back in to the circuit, in order to minimize the energy of the system. This part is done using a classical optimizer. When the optimal values of γ and β are found, the state prepared by the quantum circuit is more likely to be (or to be close enough) to the ground state of the system, retrieving the optimal solution to the original problem.

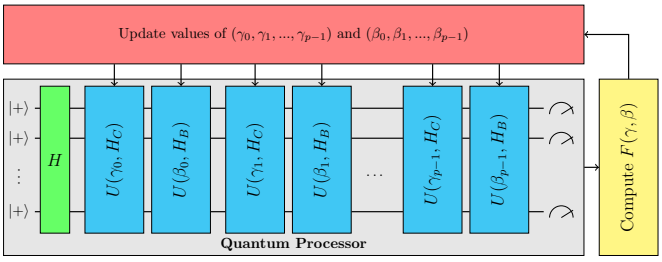


Fig. 1: QAOA schematic.

III. Material and Methods

Based on [13], in this article, a comparative study will be conducted on the classical optimizers used in QAOA and how they affect the results of the algorithm, while varying the number of layers of the quantum circuit. In order to do so, ten different optimization algorithms were selected(see table I). The classical and quantum algorithms were implemented in Python 3.11.7, while the quantum computer was simulated through the AerSimulator, from IBM's Qiskit library [14]. Qiskit was also used for all quantum computer related implementations, such as circuits, estimation of energy, and sampling for the optimized circuit. For the gradient-based algorithms, the generalized parameter-shift rule [15] was applied to estimate the gradient of the objective function.

In this paper, the following approach was used for each one of the algorithms: for different values of p(depth of the circuit), varying from 1 to 10, ten starting points for the $2p$ angle parameters $\gamma \in [0, 2\pi]$ and $\beta \in [0, \pi]$ were selected by Latin Hypercube Sampling [16]. Afterwards, for every starting point, a routine of the QAOA was run to minimize the objective function, and the starting points associated with the lowest cost function value found by the optimizer were selected. Then, from the selected starting point, the QAOA was run again ten times, and the optimized parameters associated with the lowest cost function value found by the optimizer were selected to run the circuit one final time to obtain the results. This optimized circuit was run 100 times, to avoid misleading results due to the nondeterministic characteristics of the routine. Significant data on the runs were recorded, such as the number of iterations of the classical optimizer before stopping and their respective value of the cost function, the optimized parameters found, the approximation ratio (AR) as in 6, the final bitstring result, and the probability of retrieving the optimal solution by QAOA. Two methods were utilized to simulate the quantum computer for each algorithm: the state vector and shot-based simulation, both using the Qiskit Aer Simulator. For shot-based simulations, 10000 shots were used.

$$\mathrm{AR} = \frac{f(\vec{x})}{f(\vec{x}^*)} \tag{6}$$

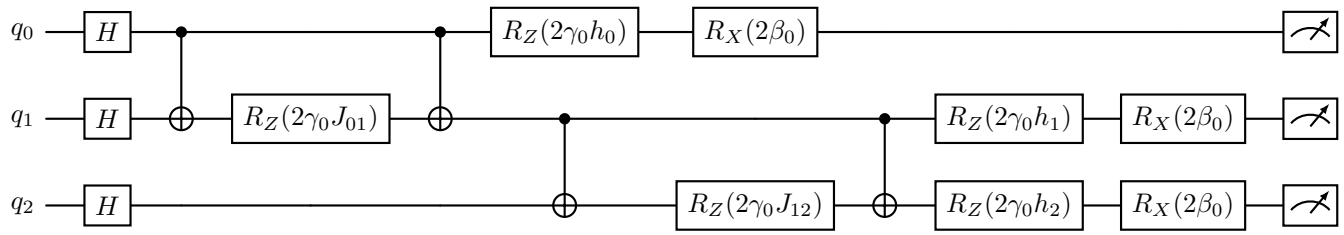


Fig. 2: Example of an ansatz circuit with $p = 1$ to solve the graph P_3.

TABLE I: Optimization algorithms applied

Gradient-based methods	
Algorithm	**Implementation**
Adaptive Moment Estimation (ADAM)	PyTorch [17]
ADAM variant with convergence (AMSGrad)	PyTorch [18]
Broyden–Fletcher–Goldfarb–Shanno (BFGS)	SciPy [19]
Limited-memory BFGS with bounds (L-BFGS-B)	SciPy [20]
Conjugate Gradient (CG)	SciPy [21]
Sequential Least Squares Programming (SLSQP)	SciPy [22]
Gradient-free methods	
Algorithm	**Implementation**
Constrained Optimization by Linear Approximations (COBYLA)	SciPy [23]
Nelder-Mead simplex search (Nelder-Mead)	SciPy [24]
Powell's direction-set method (Powell)	SciPy [25]
Simultaneous Perturbation Stochastic Approximation (SPSA)	Nevergrad [26]

In summary, two rounds of the optimization algorithm are performed: The first one, to find the best starting point from the sampled ones; and the second one, starting from the optimized starting point to find the final optimized parameters. Finally, an optimized circuit is run to retrieve the final results.

In [13], the tests for each algorithm were performed varying p from only 1 to 5, and only after finding the optimizer with the best performance(SPSA), circuits with more layers were accounted. In this paper, for a broader approach, the values of p from 1 to 10 were applied to every algorithm, in addition to the different routine with two runs. The graph selected to run the tests was one of the six graphs used by [13]: a simple graph with five vertices and six edges, as in Figure 3.

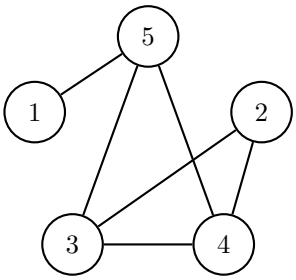


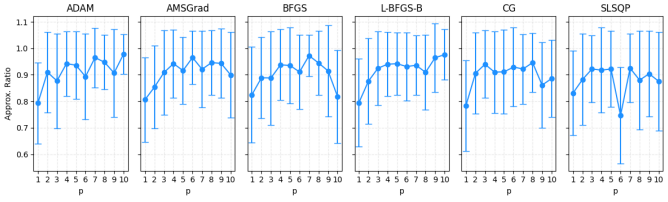
Fig. 3: Graph selected to conduct the tests.

IV. Results

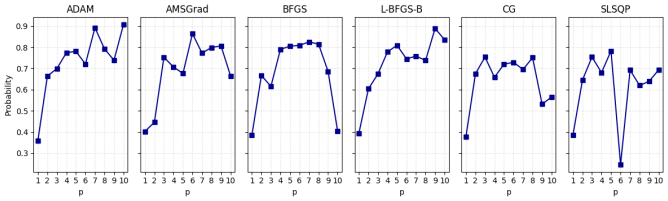
After executing the tests for the state vector simulation and the shot-based simulation as seen in Figures 4, 5 and 6, 7, respectively, it is observed that both the approximation ratio and the probability of finding the optimal solution tend to increase with higher values of p. This is true for both methods and weather the optimizer is gradient-based or not, with slight variations for each one. Most of them produced a very high approximation ratio ($0.80 \sim 0.95$), likely due to the selection of a better starting position in the first run of the routine.

For the state vector simulation, shown in Figure 4, using the gradient-based algorithms, at lower values of p $(1 - 5)$, the approximation ratio values (Figure 4a) and probability (Figure 4b) values generally increase, with few exceptions, and L-GBFS-B had a better performance for these depths. As for higher values of p $(1 - 10)$, there is not a consistent improvement as before, the results tend to vary more. For AMSGrad, BFGS, CG and SLSQP, the results get worse with the last values of p, while using ADAM and L-GBFS-B, the best results are achieved with higher circuit depth. The worsening for more circuit layers for BFGS, CG and SLSQP go as down as the results of the very first values of p, and it is interesting to notice that, for SLSQP with a circuit depth of 6, it yielded its worst result, by far. As this is a nondeterministic algorithm, even with multiple starting points in the first run, it was not able to find a good starting point for this configuration. In general, the best result achieved was using ADAM with ten layers, and for the other algorithms, their best performances had p varying from 6 to 8, except for L-BFGS-B, with the best results also at ten layers.

Still in the state vector simulation results, but this time using gradient-free algorithms, seen in Figure 5, the tendency to im-



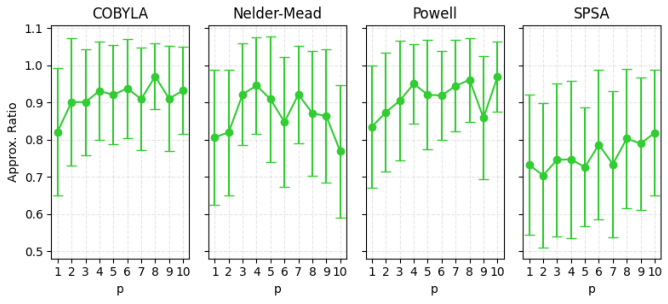
(a) Approximation Ratio vs p for gradient-based algorithms.



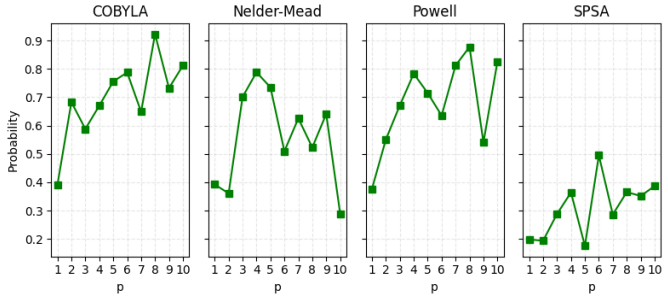
(b) Probability to find the optimal result vs p for gradient-based algorithms.

Fig. 4: Comparison of gradient-based algorithms using state vector simulation.

prove the approximation ratio (Figure 5a) and the probability (Figure 5a) as p increases for lower values appeared again. The COBYLA and Powell algorithms had better performances in general, with the COBYLA results turning into a plateau for $p > 4$, and Powell had better performances with higher p. As for Nelder-Mead, its values started to decrease for more than three circuit layers. SPSA had the worst overall results, even with results getting better for higher p.



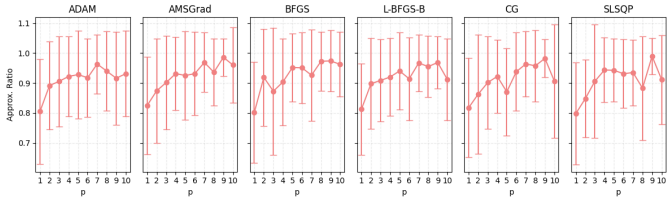
(a) Approximation Ratio vs p for gradient-free algorithms.



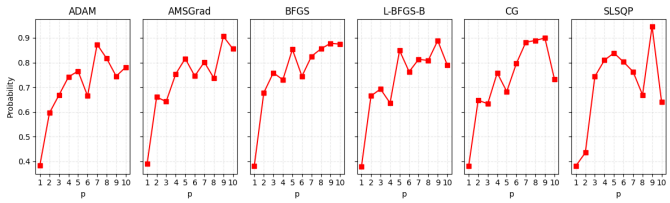
(b) Probability to find the optimal result vs p for gradient-free algorithms.

Fig. 5: Comparison of gradient-free algorithms using state vector simulation.

In the shot-based simulation, the results for the gradient-based algorithms presented in Figure 6 had a behavior similar to that of them in the state vector simulation in both AR (Figure 6a) and the probability of obtaining the optimal result (Figure 6b) at least for the lower values of p. For higher numbers of layers, there was instability again, but they did not tend to worsen as much with higher depths, all having the best performance at $p = 9$ (except for ADAM) and usually producing better results than those for smaller p. SLSQP with nine layers had the best performance, and BFGS had inconsistencies even with fewer layers, while the other methods had a somewhat more consistent increase as p grew.



(a) Approximation Ratio vs p for gradient-based algorithms.



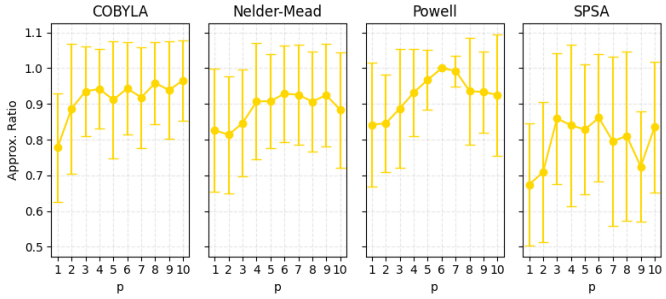
(b) Probability to find the optimal result vs p for gradient-based algorithms.

Fig. 6: Comparison of gradient-based algorithms using shot-based simulation.

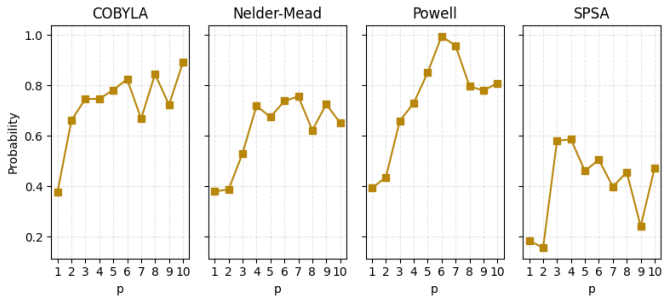
Regarding the gradient-free algorithms results, shown in Figure 7, most of them exhibited similar performance to when combined with the state vector simulation, except for the surprisingly good performance for the Powell algorithm with 6 layers, retrieving the optimal solution in 100% of the results. Of course, this is very likely to be an outlier, although the Powell algorithm started to decrease its performance for $p > 6$. This time, Nelder-Mead kept similar results for more than 4 layers instead of decreasing. The SPSA algorithm still had the worst overall results, although it performed better than it did with state vector simulations.

V. Conclusion

To summarize, when comparing the results for this study with those from [13], it is clear that pre-selecting the starting points in a first round run of the QAOA significantly increases the outcomes for both AR and probability to yield optimal results, reinforcing how landscape and starting point dependent this method is. However, some similarities and discrepancies can be observed. In both cases, there is a pattern of increasing results as the number of layers increases from 1 to 5. The main difference is the performance of the SPSA optimizer, which had the best results in their work, whereas it had the worst in this paper. This is most likely due to the fine-tuning of the algorithm parameters, to which the final outcomes can be



(a) Approximation Ratio vs p for gradient-based algorithms.



(b) Probability to find the optimal result vs p for gradient-based algorithms.

Fig. 7: Comparison of gradient-based algorithms using shot-based simulation.

really sensitive. From the presented results, it is also seen that not always the configuration with the best probabilities returns the best approximation ratio results, and vice versa. Moreover, it is observed that even with low probabilities of finding the optimal result, AR can achieve values around 0.8, which means that even though the algorithm did not find the optimal result, it retrieved a combination close enough. Although it is a good metric, this can be misleading, as the tests were run for a small graph, and several results could retrieve good approximation ratios. For every simulator-classical optimizer configuration, the results were satisfactory, but Nelder-Mead, Powell, and specially SPSA had more inconsistent results.

As for the time complexity and computational resources efficiency of this routine, the scheme presented here has a disadvantage as it uses two rounds of QAOA optimization, although it is still better than exponential time complexity. This could be improved by optimizing the first round, as its only purpose is to find better starting positions. As shown in [6], when comparing the results of the QAOA approach with a classical one for solving the MVCP, it has a clear advantage, as its complexity is polynomial $O(poly(p)+poly(m))$, where m is the number of iterations of the classical optimizer, and does not depend on the entry size n, rather other classic approaches, where its complexity depends on n. Moreover, since the time complexity depends on the number of layers, the improvement in the probability of finding the optimal result for a smaller p compared to the results of [13] is especially significant.

Regarding future research, it is suggested to improve the first round so that it does not require many computational resources as the final optimization run. Reproducing the experiments in real quantum processors is also a form of validating this proposed approach, along with using graphs with more vertices and edges to conduct the experiments.

REFERENCES

[1] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, Boston, MA, 2012.

[2] N. L. Biggs, E. K. Lloyd and R. J. Wilson, *Graph Theory*. Clarendon Press, Oxford, UK, 1999.

[3] R. L. Graham, "Bounds of Certain Multiprocessing Anomalies," *The Bell System Technical Journal*, vol. 45, pp. 1563–1581, 1966.

[4] R. M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (eds). Springer, Boston, MA, 1972.

[5] P. Sharma and A. Jain, "A review on job shop scheduling with setup times," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 230, pp. 1–17, 2015.

[6] Y. J. Zhang *et al.*, "Applying the quantum approximate optimization algorithm to the minimum vertex cover problem," *Applied Soft Computing*, vol. 118, p. 108554, 2022.

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2011.

[8] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, 1996, pp. 212–219.

[9] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proc. 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, 1994, pp. 124–134. p. 025022, 2024.

[10] E. Farhi and J. Goldstone, "A Quantum Approximate Optimization Algorithm," 2014.

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014.

[12] E. F. Combarro and S. González-Castillo, *A Practical Guide to Quantum Machine Learning and Quantum Optimization*. Packt Publishing, Birmingham, UK, 2023.

[13] A. Pellow-Jarman *et al.*, "The effect of classical optimizers and Ansatz depth on QAOA performance in noisy devices," *Scientific Reports*, vol. 14, 2024.

[14] IBM, "IBM Quantum Qiskit." Available at: `https://www.ibm.com/quantum/qiskit`. Accessed: June 23, 2025.

[15] D. Wierichs, J. Izaac, C. Wang and C.Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, 2022.

[16] M. D. McKay, R. J. Beckman and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, pp. 239–245, 1979.

[17] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," 2017.

[18] S. J. Reddi, S. Kale and S. Kumar, "On the Convergence of ADAM and Beyond," 2018.

[19] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, pp. 317–322, 1970.

[20] R. H. Byrd, P. Lu, J. Nocedal and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing*, vol. 16, pp. 1190–1208, 1995.

[21] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, 1952.

[22] P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, vol. 4, pp. 1–51, 1996.

[23] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," *Advances in Optimization and Numerical Analysis*, vol. 275, pp. 51–67, 1994.

[24] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308–313, 1965.

[25] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, pp. 155–162, 1964.

[26] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, vol. 37, pp. 332–341, 1992.