An exponentially more efficient optimization algorithm for noisy quantum computers

Marko J. Rančić

TotalEnergies, Nano-INNOV – Bât. 861, 8, Boulevard Thomas Gobert – 91120 Palaiseau – France

*To whom correspondence should be addressed; E-mail: marko.rancic@totalenergies.com

Quantum computers are devices which allow the solution of problems unsolvable to their classical counterparts. As an error-corrected quantum computer is still a decade away the quantum computing community has dedicated much attention to developing algorithms for currently available Noisy Intermediate-Scale Quantum computers (NISQ). Thus far, within NISQ, optimization problems are one of the most commonly studied and are exclusively tackled with the Quantum Approximate Optimization Algorithm (QAOA). This algorithm predominantly computes graph partitions with a maximal separation of edges (MaxCut), but can also be modified to calculate other properties of graphs. Here, I present a novel quantum optimization algorithm which uses exponentially less qubits as compared to the QAOA while requiring a significantly reduced number of quantum operations to solve the MaxCut problem. Such an improved performance allowed me to partition graphs with 32 nodes on publicly available 5 qubit gate-based quantum computers without any preprocessing such as division of the graph into smaller subgraphs. This results represent a 40% increase in graph size as compared to state-of-art experiments.
on gate-based quantum computers such as Google Sycamore. The obtained lower bound is 54.9% on the solution for actual hardware benchmarks and 77.6% on ideal simulators of quantum computers. Furthermore, large-scale optimization problems represented by graphs of a 128 nodes are tackled with simulators of quantum computers, again without any pre-division into smaller subproblems and a lower solution bound of 67.9% is achieved.

**Introduction**

A universal quantum computer has been the holy grail of quantum technology (1). Such a device would allow more efficient searching through databases (2), prime number factorization (3), and more efficient solutions of systems of linear equations (4), just to name a few. However, universal quantum computers require millions of qubits with quantum error correction implemented and are still a decade away from being materialized according to estimations from leading quantum hardware vendors. On the other hand, devices with up to 65 noisy qubits are readily available. This steered the scientific community towards exploring potential computation advantages which such devices could bring. In the rapidly expanding field of Noisy Intermediate-Scale Quantum Computing (NISQ) (5) two algorithms stand out in prospect: the Variational Quantum Eigensolver (VQE) (6) and the before mentioned Quantum Approximate Optimization Algorithm (QAOA) (7). The VQE is mainly applied to problems in chemistry and material science while the QAOA is a predominantly a graph MaxCut algorithm. Given a graph $G = (V, E)$ comprising of $|V|$ vertices and $|E|$ nodes the QAOA algorithm requires $n = |V|$ qubits and $p (|E| + |V|)$ quantum operations to calculate the MaxCut (8, 9). Here, $p$ is the phenomenological depth parameter.

In this manuscript a novel variational MaxCut algorithm requiring $n = \lceil \log_2 |V| \rceil$ qubits is introduced, where $\lceil \rceil$ stands for the ceiling function. For example if $x = 2.1$, $\lceil x \rceil = 3$, if
In similarity with all other NISQ algorithms the algorithm presented here iteratively improves a trial solution in a hybrid quantum-classical optimization loop. The trial solution is implemented with at most $2^n - 2n + 5$ single qubit gates and at most $2^n - 2$ two qubit CNOT gates (in total up to $2^{n+1} - 2n + 3$ gates). Exploiting the fact that large graphs can be treated with the algorithm at a low resource overhead I demonstrate the calculation of a \textit{MaxCut} of a graph of 32 nodes on a publicly available device of only 5 qubits. This is a $40\%$ increase in graph size to state-of-art experiments with QAOA on gate-based quantum computers such as the Google Sycamore \textsuperscript{(10)}\textsuperscript{1}. The algorithm presented here opens perspective for immediate quantum speedup with contemporary algorithms, given that the quantum hardware community is still some years away from producing processors with hundreds of qubits required for quantum speedup with QAOA \textsuperscript{(12)}. Furthermore, a graph of 128 nodes is partitioned on contemporary simulators of quantum computers. With this methodology, simulators of quantum computers become a powerful tool for graph partitioning, being able to tackle graphs of hundreds of nodes without dividing the problem into smaller subgraphs, such as the work done in \textsuperscript{(13)}.

\textbf{Methods}

\textbf{Variable reduction}

Now a variable reduction technique compatible with the \textit{MaxCut} problem is going to be presented. Due to the fact that the binary optimization problem of finding a \textit{MaxCut} is NP-hard, a first approach to approximately solve the problem would be to linearly relax the problem. Meaning that instead of assuming that binary optimization variables in the \textit{MaxCut} problem are integers 0 or 1, one assumes that they are continual variables $[0, 1]$. In the field of semi-definite programming \textsuperscript{(14)} a different, more efficient approach is taken, binary variables are

$\textsuperscript{1}$The reader should be made aware of the state-of-art QAOA experiments with 40 trapped ion qubits \textsuperscript{(11)} However, here the focus was on finding the ground state of the linear Ising model rather than optimizing a partition of a realistic graph.
substituted with vectors. Such classical method of approximately solving the MaxCut problem is state-of-the-art and has maximally possible performance guarantee for of 0.87856 as proved by Goemans-Williamson (14).

Here an alternative approach compatible with quantum computing is presented. Let me now introduce a continuous, differentiable function of the following form

$$R_f(x, q, m) = \exp \left( - \exp \left( 2^{m-q} \sin(2^q x + x_0(q, m)) \right) \right), \quad (1)$$

where $x_0(q, m) = \arcsin \left( \log_2(-\log_2(0.5)) / 2^{m-q} \right)$ and the integer $m \geq |V|$ and $0 \leq q \leq |V| - 2$.

Assume a graph where $|V| \gg 1$, consequently $m$ is a large number. $R_f(x, 0, m)$ is such a function which is mostly 0 in the region of $0 \leq x \leq \pi$ and rapidly changes to 1 in the region $\pi < x \leq 2\pi$ (see red line in Figure 1). I can therefore substitute any binary variable $\theta_1$ taking values 0 or 1 with $\theta_1 \rightarrow R_f(x, 0, m)$. The second function $R_f(x, 1, m) = 0$ for $0 \leq x \leq \pi/2$ and $\pi \leq x \leq 3\pi/2$ and $R_f(x, 1, m) = 1$ for $\pi/2 < x < \pi$ and $3\pi/2 < x < 2\pi$. Therefore, I substitute the second binary variable $\theta_2 \rightarrow R_f(x, 1, m)$ (see green dashed line in Figure 1). By substituting all $\theta_1, \ldots, \theta_{|V|-1}$ variables with $R_f(x, 0, m), \ldots, R_f(x, |V|-2, m)$ I have mapped the $|V| - 1$ dimensional binary optimization problem to one-dimensional multi-modal continual

![Figure 1: Eq. (1) for $q = 0$ red, $q = 1$ dashed green and $q = 2$ dotted black for $m = 4$.](image-url)
variable optimization problem.

The algorithm

A Laplacian of a graph $L(G)$, where $G = (V, E)$ is a $|V| \times |V|$ matrix with $|V|$ positive terms on the diagonal and $2|E|$ off-diagonal terms. The $i$th diagonal term of the graph Laplacian corresponds to the number of connections the node $i$ has with remaining nodes in the graph and the $ij$th off-diagonal term of the matrix is the negative weight between the $i$th and the $j$th node.

Similarly to a real-valued Hamiltonian in quantum mechanics, the graph Laplacian is symmetric, furthermore it has a spectrum (eigenvalue range) between 0 and its largest eigenvalue. Now, I introduce a partition vector $V$ of length $|V|$ with $i$th term equaling $+1$ if $i$th vertex of $G$ belongs to the first sub-graph in the graph partition and $i$th term $-1$ if the $i$th node belongs to the second sub-graph in the graph partition. Then, the number of cuts in the graph bi-partition is $N_{\text{cuts}} = V^T L V / 4$ (15), and this formula is a central piece of the algorithm presented here. By finding the vector $V$ which maximizes $N_{\text{cuts}}$ a MaxCut of the graph is found. Vector $V$ has $2^{|V|}$ possible values and there is no known algorithm which can exactly find $V$ which maximizes $N_{\text{cuts}}$ with a computational complexity which is a polynomial of $|V|$.

Now I present the structure of the algorithm for a pre-selection of only one optimization variable $x$. The possible number of variables spans from 1 to $|V| - 1$.

1. Trivial unconnected vertices are added to the graph so it has a dimension which is a power of two.

2. The graph Laplacian $L(G)$ is represented as a sum of tensor products of unitary matrices, and denoted as $L(G)$ in such form.

3. If a graph has $|V|$ nodes a Haddamard gate is applied to $\lceil \log_2 |V| \rceil$ qubits. This operation is denoted with $H_G$.

4. A variational ansatz in a form of diagonal gate is applied
$U = \text{diag}(e^{i\pi R_f(x,0,m)}, \ldots, e^{i\pi R_f(x,|V|-2,m)}, 1, \ldots, 1)$. 

5. The number of cuts is calculated as

$$N_{\text{cuts}} = 2^{n-2}\langle 0 | H_G U \mathcal{L}(G) U H_G | 0 \rangle. \quad (2)$$

6. Variational parameter $x$ is adjusted with a classical optimizer and steps 3-5 are repeated until a maximum is reached.

The algorithm presented here maps the MaxCut problem of a graph $G = (V, E)$ comprising of $|V|$ vertices and $|E|$ edges to a problem of $|V|$ energy levels coupled with $|E|$ coupling terms described by a Hamiltonian $\mathcal{L}(G)$. The weight between the nodes $w_{ij}$ becomes a coupling strength between energy levels $i$ and $j$. Energy levels $i$ and $j$ are residing at an energy equal to the connectivity of the node $i$ and $j$ respectively.

In Figure 2 I represent a simple example of a graph with four nodes. For instance the node 1 is connected with two other nodes (Figure 2(a)). Therefore, the energy level 1 lies at an energy $E = 2$ in Figure 2(b). Node 1 is connected with nodes 2 and 3 so the level 1 is coupled with levels 2 and 3 in the energy scheme. And such logic applies for all nodes of any graph. The algorithm searches for a unitary transformation of the Hamiltonian which maximizes the number of cuts.

**Circuit depth, computational complexity and quantum speedup**

Given a graph with $|V|$ nodes $n = \lceil \log_2 (|V|) \rceil$ qubits are required to implement the algorithm. The multi-control multi-target qubit gate on $n$ qubits required to realize the diagonal gate $U$ in Eq. (2) can be straightforwardly realized with Grey codes (16) or in the context of follow up work (17) at a cost of $(23/48) \times 4^n - (3/2) \times 2^n + 4/3$ CNOT gates. However, exploiting the fact that $U$ is a diagonal gate and following on the works of (17) and (18) it can be realized with $2^n - 2$ CNOT gates. This means that the ansatz is implemented with less than $2|V|$ two qubit CNOT gates which is in stark contrast with the QAOA ansatz which requires $p|E|$ two
Figure 2: (a) A simple graph where $w_{ij}$ are the off diagonal elements of a graph Laplacian. (b) The mapping of the graph to a set of coupled energy levels.

qubit gates, where $p$ is the depth parameter. Given that $|E| \gg |V|$ the algorithm presented here is much more efficient in the number of two qubit gates as compared to even the lowest depth $p = 1$ QAOA.

The algorithm presented here is a heuristic, meaning that its depth is case dependent. However, the quantum implementation of the heuristic can be compared with its classical counterpart for every step of the evaluation. A classical computing variant of Eq. (2) is a vector-matrix-vector multiplication. For a $|V| \times |V|$ matrix the computational complexity of such an evaluation is $O(|V|^2)$. On a quantum computer the computational complexity of evaluating Eq. (2) is $O(|V|^3)$ (one power of $|V|$ coming from the ansatz and up to $|V|^2$ summands yielding $\mathcal{L}(G)$). However, simulating a $d-$sparse Hamiltonian ($d-$ regular graph) is done in maximally $O(d^2(d + \log^* n))$ queries (19,20), and efficiently simulating a Hamiltonian is equivalent to calculating an expectation value of a Hamiltonian (21). So for $d-$regular graphs every step in the heuristic algorithm executes in $O(|V|d^2(d + \log^* |V|))$ which is smaller than the classical $O(|V|^2)$.

Throughout this manuscript I assumed that either the graph Laplacian is given in terms of
sums of tensor products of Pauli matrices or that an efficient decomposition into a sum of tensor product of unitary matrices exists. The latter is factually true for $d$—sparse matrices (graphs), see for instance Appendix A of Ref. (22) and references therein for an in-depth discussion.

In Table 1 I summarize the main differences between the algorithm presented here and the QAOA. The diagonal gates required to preform the algorithm presented here require an all-in-all connectivity between qubits for optimal performance. On the other hand side, QAOA preforms best when the qubits are connected in the same way as the nodes of the graph (10).

| Complexity (1 evaluation step) | QAOA                  | New alg.                  | New alg. $d$—regular graphs |
|-------------------------------|-----------------------|---------------------------|-------------------------------|
| Qubit number                  | $O(p(|E| + |V|))$       | $O(|V|^3)$                | $O(|V|d^2(d + \log^* \lceil \log |V| \rceil))$ |
| Optimal connectivity          | $|V|$ graph inspired   | $\lceil \log_2 |V| \rceil$ all-in-all | $\lceil \log_2 |V| \rceil$ all-in-all |

Table 1: A table summarizing the difference between the approach presented here and QAOA.

**Results**

In Figure 3 I compare the output of a simulator with the output of publicly available IBMQ Santiago. Although pure dephasing, shot noise and relaxation may distort the optimization landscape maxima are clearly noticed although equal local maxima become unequal. The $N_{\text{cuts}}$ is estimated for 100 equidistant values of $x$.

I further present results obtained by benchmarking randomly generated 3-regular graphs of 32 nodes on actual quantum computers and simulators and randomly generated 3-regular graphs of 128 nodes on a simulator of quantum computers (Qiskit). Intensive testing showed that the algorithm performs best when the number of variables is kept at 8 or 16 for graphs of the size 32–128 nodes. Intensive numerical testing also showed that a genetic optimizer is best suited for finding the maximum of the function - not too surprising as genetic optimizer is indeed best used for multi-modal cost functions. On top of the genetic algorithm, a number of classical op-
timizers were tried (COBYLA, Neldear-Mead, Basin-Hopping, Particle Swarm, EGO). Further details about the setting of the genetic optimizer can be found in the Supplementary material.

For the case of 3-regular graphs of 32 nodes variable reduction is preformed (as described in the previous section) so that the optimization landscape has 8 variables. The MaxCut is calculated with 5 qubits. The Goemans-Williamson algorithm (GW) has a performance guarantee of 0.87856. I define the approximation ratio of the algorithm presented here with respect to the exact solution as

$$
0.87856 \frac{\text{MaxCut}}{\text{MaxCut}_{GW}} \leq r \leq \frac{\text{MaxCut}}{\text{MaxCut}_{GW}}
$$

(3)

where $\text{MaxCut}$ is the value obtained with the algorithm presented here and $\text{MaxCut}_{GW}$ is the value obtained with Goemans-Williamson. For the first graph Figure 4 (a-c) the algorithm preformed on a simulator of quantum computers yields $0.776 \leq r \leq 0.884$ and the algorithm executed of IBMQ Quito $0.552 \leq r \leq 0.628$. For the second graph Figure 4 (d-f) the algorithm preformed on a simulator of quantum computers yields $0.857 \leq r \leq 0.975$ and the algorithm executed of IBMQ Santiago $0.549 \leq r \leq 0.625$. For both realizations there is a clear difference

![Figure 3: The number of cuts of a 4-node graph simulated obtained with 2 qubits.](image-url)
between actual hardware benchmarks and ideal simulation. I assume that the main reason for this is the distortion of the optimization landscape due to pure dephasing and relaxation. I expect that shot noise contributed less as the trials were executed for the maximally allowed 8192 shots. A $d$-regular graph with $|V|$ nodes has $d \times |V|/2$ edges (23). An average random bi-partition of such a graph is $d \times |V|/4$ (24), or in the case of 3-regular graphs with 32 nodes the average random bi-partition is 24. So both the quantum hardware results and the simulator results stay above the average random bi-partition value.

For the case of 3-regular graphs of 128 nodes variable reduction is performed (as described in the previous section) so that the optimization landscape has 16 variables. The MaxCut is calculated with 7 ideally simulated qubits. Given that devices larger than 5 qubits are unavailable to the author for these 3 graphs I stayed in the domain of simulators of quantum computers. For the first graph Figure 5 (a-b) $0.679 \leq r \leq 0.773$, second graph Figure 5 (c-d) (a) GW MaxCut = 43 (b) Qiskit MaxCut = 38 (c) IBMQ Quito MaxCut = 27

(d) GW MaxCut = 40 (e) Qiskit MaxCut = 39 (f) IBMQ Santiago MaxCut = 25

Figure 4: Randomly generated 3-regular graphs of 32 nodes. Nodes belonging to different partitions are marked in green and orange respectively and the MaxCut value is written on top of the graph. Graphs are randomly initialized by Python’s networkx package where seed 20 is used for graphs (a-c) and seed 30 for graphs (e-g). GW stands for Goemans-Williamson.
Figure 5: Randomly generated 3-regular graphs of 128 nodes. Nodes belonging to different partitions are marked in green and orange respectively and the \textit{MaxCut} value is written on top of the graph. Graphs are randomly initialized by Python’s networkx package where seed 7 being used for graphs (a-b), seed 8 for graphs (c-d) and seed 9 for graphs (e-f). GW stands for Goemans-Williamson.
$0.743 \leq r \leq 0.846$, third graph Figure 5(c-d) $0.709 \leq r \leq 0.807$. Values do not converge nicely as for smaller graphs likely because the genetic algorithm gets trapped in a local minimum with increasing system size. An average random bi-partition of a 3-regular graph of 128 nodes is $3 \times 128/4 = 96$.

**Conclusions**

In conclusion I have presented a novel algorithm for noisy intermediate-scale quantum computers requiring logarithmically less qubits and significantly less quantum gates as compared to contemporary state-of-art algorithm - QAOA. I went through to calculate the MaxCut of a randomly generated 3-regular graph of 32 nodes, a 40% increase compared to experiments of state-of-art gate-based quantum computers (Google Sycamore). I did so with publicly available IBM hardware, and obtained a lower bound of 54.9% on the solution for actual hardware benchmarks and 77.6% on ideal simulators of quantum computers. Furthermore, I calculated the MaxCut of a 3-regular graph of 128 nodes with quantum simulator obtaining a lower bound of 67.9% on the solution and with no pre-processing of the graph what-so-ever.

**Acknowledgements**

I acknowledge discussions with Adrien Suau, Vedran Dunjko, Thomas Bäck and Zaid Ally-bokus, with the latter especially in the domain of use cases to which the algorithm may be applied. I am thankful to the EU for funding within the H2020 project ⟨NE|AS|QC⟩.

**Competing interests**

The work presented here is a part of a broader provisional patent claim ”Method for optimizing a functioning relative to a set of elements and associated computer program product” submission
number EP21306155.9 submitted on 26.8.2021. with Marko J. Rančić and Zaid Allybokus being listed as inventors. The author declares no further competing interests.

**References**

1. R. P. Feynman, *International Journal of Theoretical Physics* **21**, 467 (1982).

2. L. K. Grover, *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), pp. 212–219.

3. P. W. Shor, *Proceedings 35th annual symposium on foundations of computer science* (Ieee, 1994), pp. 124–134.

4. A. W. Harrow, A. Hassidim, S. Lloyd, *Phys. Rev. Lett.* **103**, 150502 (2009).

5. J. Preskill, *Quantum* **2**, 79 (2018).

6. A. Peruzzo, *et al.*, *Nature communications* **5**, 4213 (2014).

7. E. Farhi, J. Goldstone, S. Gutmann, *arXiv preprint arXiv:1411.4028* (2014).

8. L. Zhou, S.-T. Wang, S. Choi, H. Pichler, M. D. Lukin, *Phys. Rev. X* **10**, 021067 (2020).

9. M. Cerezo, *et al.*, *Nature Reviews Physics* pp. 1–20 (2021).

10. M. P. Harrigan, *et al.*, *Nature Physics* **17**, 332 (2021).

11. G. Pagano, *et al.*, *Proceedings of the National Academy of Sciences* **117**, 25396 (2020).

12. G. G. Guerreschi, A. Y. Matsuura, *Scientific reports* **9**, 1 (2019).

13. J. Li, M. Alam, S. Ghosh, *arXiv preprint arXiv:2102.13288* (2021).

14. M. X. Goemans, D. P. Williamson, *Journal of the ACM (JACM)* **42**, 1115 (1995).
15. A. Pothen, H. D. Simon, K.-P. Liou, *SIAM journal on matrix analysis and applications* **11**, 430 (1990).

16. M. Möttönen, J. J. Vartiainen, V. Bergholm, M. M. Salomaa, *Phys. Rev. Lett.* **93**, 130502 (2004).

17. V. V. Shende, S. S. Bullock, I. L. Markov, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **25**, 1000 (2006).

18. E. Malvetti, R. Iten, R. Colbeck, *Quantum* **5**, 412 (2021).

19. D. W. Berry, G. Ahokas, R. Cleve, B. C. Sanders, *Communications in Mathematical Physics* **270**, 359 (2007).

20. A. M. Childs, R. Kothari, *Conference on Quantum Computation, Communication, and Cryptography* (Springer, 2010), pp. 94–103.

21. E. Knill, G. Ortiz, R. D. Somma, *Phys. Rev. A* **75**, 012328 (2007).

22. C. Bravo-Prieto, *et al.*, *arXiv preprint arXiv:1909.05820* (2019).

23. J. M. Aldous, R. J. Wilson, *Graphs and applications: an introductory approach* (Springer Science & Business Media, 2003).

24. R. Motwani, P. Raghavan, *Randomized algorithms* (Cambridge university press, 1995).

25. R. M. Solgi, *https://github.com/rmsolgi/geneticalgorithm* (2020).

26. S. Katoch, S. S. Chauhan, V. Kumar, *Multimedia Tools and Applications* **80**, 8091 (2021).
Supplementary Material: An exponentially more efficient optimization algorithm for noisy quantum computers

The classical optimizer

After intensive numerical testing a genetic optimizer preformed the best in converging to the maximal cost function. An open GitHub code was incorporated Ref. (25). A genetic algorithm is an optimizer inspired in evolution (26). To obtain results displayed in Figure 4 the following optimizer setting was used.

```
algorithm_param = {'max_num_iteration': 40,
                   'population_size': 5,
                   'mutation_probability': 0.25,
                   'elit_ratio': 0.01,
                   'crossover_probability': 0.5,
                   'parents_portion': 0.3,
                   'crossover_type': 'uniform',
                   'max_iteration_without_improv': None}
```

For results displayed in Figure 5 a maximum of 200 iterations were set with a population size of 14. In Figure S1 I show how the genetic optimizer improves the cost function as a function of iteration. The value is negative as the optimizer is tuned to search for the minimum of the negative graph Laplacian under a unitary transformation.
Figure S1: Evolution of cost functions as a function of the number of iteration for Figure 4.