Comparison of the AQO and the QAOA for the vertex coloring problem

Author: Eric Vidal[†‡*]. Advisors: Marta P Estarellas[‡], Jordi Riu[‡] & Lluís Garrido[†].
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain,[†] *and*
Qilimanjaro Quantum Tech, 08007 Barcelona, Spain.[‡]

We benchmark the time resources needed to execute the adiabatic quantum optimization (AQO), and to run the Quantum Approximate Optimization Algorithm (QAOA), for the vertex coloring problem. This is done via a numerical simulation of 20 Erdős-Rény random graphs for different cases ranging from 8 to 21 qubits. With this comparison, we explore two of the most important algorithms of the analog and gate-based quantum computing paradigms, respectively. We apply the canonical implementation for both algorithms, so their initial Hamiltonian is the same, the one with the typical sum of Pauli-X matrices. In this line, we consider linear scheduling time for the AQO. For final adiabatic time $T = 100$ ns, the AQO achieves an overlap with the degenerate solutions over 0.9 in all cases. Meanwhile, the QAOA using the Powell classical optimizer, 5 layers and thousands of iterations has an overlap around 0.5 for the 8 qubits case and below 0.25 for the other cases. So, our results for the given task and idealized conditions indicate that the AQO significantly outperforms the QAOA in terms of time and success probability.

I. INTRODUCTION

The recent improvements in quantum computation hardware has led to the noisy intermediate-scale quantum (NISQ) era. In this context, an increasing number of variational quantum algorithms (VQAs) have arisen [1] with the target of solving combinatorial optimization problems (COPs) and evaluating the potential to achieve a quantum advantage. The main idea of VQAs is to run a parameterized quantum circuit in a model gate-based quantum computer and optimize the parameters with the aid of classical optimizers.

A promising prospect is the QAOA, introduced by Farhi et al. in 2014 [2], which is inspired by a Trotterized adiabatic transformation of the adiabatic evolution applied to the gate-based model. In contrast, the Quantum Adiabatic Algorithm (QAA) (Farhi et al. [3]) is a continuous-time algorithm based on the adiabatic theorem of quantum mechanics [4], which requires analog quantum computers for its implementation.

There is a strong link between these two algorithms. In fact, an important achievement for QAOA is the proof of its convergence [5], which is based on the convergence of QAA. Therefore, it is interesting to compare these two schemes' performance in actual quantum computers [6], since both approaches have different strengths.

It is worth mentioning that quantum annealing (QA), as well as VQAs, was conceived as a heuristic approach to solve COPs, especially the ones that can be expressed as binary optimization problems with Boolean variables $x_i \in \{0,\ 1\}$. In this work, for implementing the QAA we will deal with adiabatic quantum computing (AQC) [7]. Note that AQC is a restricted case of QA, as it only considers adiabatic evolution, while QA is a broader concept enabling non-ideal and diabatic processes.

Nowadays due to the technical complexity that supposes to develop an *annealer*, this kind of quantum computer is restricted to solving only QUBO problems with 2-local interactions, with $H_{QUBO}(\vec{x}) = \sum_i h_i x_i + \sum_{ij} Q_{ij} x_i x_j$ as the cost function.

Those might seem like strong restrictions, however, lots of problems of interest not only for industry but for academia as well, fall into this category. Most of them are contained in Karp's 21 NP-complete problems, with a close connection to Ising spin models and QUBO formulation, as many other NP-hard problems [8].

In this work, we propose a comparison between AQO and QAOA applied to the vertex coloring problem which is an NP-complete problem contained in the class of graph coloring problems (GCP). The GCP class has different types, methods, and applications within real-life problems, [9], such as scheduling problems including assigning frequencies for radio stations or for a mobile phone network, job shop scheduling, resource allocation, or flight-gate assignments. The latter use case has been explored in [10] using QAOA and in [11] using AQO.

The work is organized as follows, in Section II the embedding of a COP problem into a quantum computer is described. Section III provides a brief explanation of the inner workings of AQO. Similarly, Section IV does the same for the QAOA. In Section V we present the formulation for the vertex coloring problem. Finally, in Section VI the results of the simulations of AQO and QAOA are presented and analyzed using time metrics, leading to the conclusions and future work in Section VII.

II. COP ENCODING ON QUANTUM COMPUTER

When dealing with generic COPs of size N usually one have to either minimize (or maximize) a cost function, also known as objective or loss function, $f : Z(N) \to \mathbb{R}$,

[*] Electronic adress: evidalma52@alumnes.ub.edu

where $Z(N)$ is the set of bit strings with length N,

$$\min_{z\in S} f(z), \quad S \subseteq Z(N), \tag{1}$$

where S is the set of feasible bit strings. Then, S_{min} is defined as the subset of all solutions minimizing f. If $S = Z(N)$, it is said that the problem is *unconstrained.*

To work with a quantum computer we have to express each bit string, z, in the computational basis, $|z\rangle$ of the N-qubit space, $\mathcal{H} = \mathbb{C}^{2^N}$. This means that each bit, $z_i = 0,\ 1$ of the bit string z is replaced by a spin-1/2 qubit labeled by $|z_i\rangle$, in a way that $|z\rangle$ is the Kronecker product of all the qubit states. These are eigenstates of the z component of the i-th spin, $|0\rangle$, $|1\rangle$, so $\frac{1}{2}(1-\sigma_z^{(i)})\,|z_i\rangle = z_i\,|z_i\rangle$, where $\sigma_z^{(i)}$ is the corresponding Pauli matrix. Lastly, the Hamiltonian that embeds the cost function can be constructed as, $C = \sum_{z\in Z(N)} f(z)\,|z\rangle\langle z|$. Therefore, the optimal solution of the problem in S_{min} is encoded in the ground state of the problem Hamiltonian, which corresponds to the state (or Hilbert subspace in case of degenerate solutions) that has the minimum energy.

III. ADIABATIC QUANTUM OPTIMIZATION

The main idea of AQO is to reach the ground state of the problem Hamiltonian, H_p, which codifies the problem constraints, by adiabatically evolving the system during a time, T, from the *easy*-to-prepare ground state of a Hamiltonian, H_0, known as driving or easy Hamiltonian. Then the quantum system evolves according to the Schrödinger equation,

$$i\frac{d}{dt}\,|\psi(t)\rangle = H(t)\,|\psi(t)\rangle\,, \tag{2}$$

where the time-dependent Hamiltonian must be,

$$H(t) = (1-s(t))\,H_0 + s(t)H_p, \tag{3}$$

defining the scheduling time, $0 \leq s(t) \leq 1$, in a way that T controls the rate at which the interpolation is done. This scheme is based on the adiabatic theorem of quantum mechanics, so since the system is initialized in the *easy*-to-prepare ground state, it will remain in the ground state for infinitely slow evolution. Thus for a T large enough, the system will remain close to the ground state and we will be able to find the solution when $H(T) = H_p$.

The order of T is approximately determined by the minimum gap energy, g_{min}, as $T \sim g_{min}^{-2}$. The g_{min} is the minimum necessary energy to excite the ground state to the first excited state, which is not necessarily their energy difference when it comes to degenerate states, as it is further discussed in-depth in Section VI A.

It is also worth noting that the $s(t)$ is usually a polynomial function of (t/T) in a way that the closer it gets to the minimum gap, the more slowly it varies, so the evolution can be performed faster in other points and save time. Unfortunately, estimating the value of the gap is an NP-hard problem by itself, so finding the optimal function $s(t)$ is a very complex task.

IV. QAOA

QAOA is a VQA that iteratively applies a parameterized quantum circuit of p layers applied to a fixed initial quantum state, $|s\rangle$. Each layer contains a pair of parameter-dependent unitary operators, the *mixer*, $U_B(\beta) = e^{-i\beta B}$, which depends on an initial Hamiltonian, B, and a parameter $\beta \in [0, \pi]$; and, the *phase separator*, $U_C(\gamma) = e^{-i\gamma C}$, which depends on the problem Hamiltonian, C, and a parameter $\gamma \in [0, 2\pi]$.

Thus, a parameterized trial state is constructed,

$$\left|\vec{\beta},\vec{\gamma}\right\rangle = V(\vec{\beta},\vec{\gamma})\,|s\rangle = \left(\prod_{q=1}^{p} U_B(\beta_q)U_C(\gamma_q)\right)|s\rangle\,, \tag{4}$$

where p is the number of layers.

The trial state parameters β and γ are optimized by minimizing the value of $F_p(\vec{\beta},\vec{\gamma}) = \left\langle\vec{\beta},\vec{\gamma}\right|C\left|\vec{\beta},\vec{\gamma}\right\rangle$, i.e. the expected value of the cost Hamiltonian, using a classical optimizer. Once this process converges to the final state, $\left|\vec{\beta_{out}},\vec{\gamma_{out}}\right\rangle$, one can obtain a solution to the problem by measuring the state in the computational basis. If properly optimized, the success probability improves with p, eventually reaching the solution [2, 5].

V. VERTEX COLORING PROBLEM

Solving the vertex coloring problem consists in checking if a considered undirected graph, $G = (V, E)$, can have its vertices colored such that no edge is connecting two vertices with the same color. It is expressed mathematically in graph theory as the proper mapping $f : V(G) \to \mathbb{N}$ such that,

$$\forall_{v_i,v_j\in V(G),i\neq j}\exists(e_i,\ e_j) \implies f(i) \neq f(j), \tag{5}$$

where v_i is a vertex contained in the vertices set V, and $(e_i,\ e_j)$ is an edge connecting v_i and v_j.

Note that for any case there are several degenerate solutions due to all color permutations, and there is a minimum number of colors needed to meet the vertex coloring problem constraints, called the chromatic number, $\chi(G)$. Determining $\chi(G)$ is an NP-hard problem because it means adding a constraint to the cost function that has to be minimized, instead of verified. An easy-to-solve example is a fully connected graph, as $\chi(G)$ is equal to the number of vertices.

A. QUBO formulation and implementation

Consider the undirected graph $G = (V, E)$ and several different colors c. The number of vertices and edges is given by $w = |V|$ and $m = |E|$, respectively. Next, define the binary variable, $x_{v,i}$, which is 1 if vertex v is colored with the color i, and 0 otherwise. So, we find the total number of binary variables (qubits), $N = w \times c$, and the cost function,

$$H = A \sum_{v=1}^{w} \left(1 - \sum_{i=1}^{c} x_{v,i}\right)^2 + A \sum_{(uv)\in E} \sum_{i=1}^{c} x_{u,i} x_{v,i}, \quad (6)$$

whose ground state (with energy 0) encodes the degenerate solutions of the problem. The first term ensures that each vertex has only one color assigned, and the second term penalizes two adjacent vertices $x_{v,i}$ and $x_{u,i}$ ($v \neq u$) with the same color.

The cost function (6) is translated to a quantum Hamiltonian by mapping each binary variable, $x_{v,i}$, to the operator $\hat{\mathcal{Z}}_{v,i} = \frac{\mathcal{I} - \sigma^z_{v,i}}{2}$, with $\sigma^z_{v,i}$ being the Pauli-Z operator acting on the v,ith qubit in a Hilbert space of N qubits. This transformation has eigenvectors $|0\rangle$ and $|1\rangle$ with eigenvalues 0 and 1, respectively, as desired.

Then for both AQO and QAOA, the initial Hamiltonian is $B = H_0 = \sum_{i=1}^{N} \frac{\mathcal{I}_i - \sigma^x_i}{2}$, in order to have as a ground and initial state with energy 0 the superposition of all computational basis states, $|s\rangle = \frac{1}{\sqrt{2^N}} \sum_{z\in Z(N)} |z\rangle$, so all the bit strings are examined, because it is an unconstrained problem.

VI. NUMERICAL RESULTS

In this work, we simulate both algorithms using a classical computer and extrapolate known data from state-of-the-art quantum computers to add to our benchmark metrics. The evolution operator is computed each step of time, $U(t) = e^{-iH(t)\delta t}$, using the open-source programming framework for quantum computers **Qibo**. The classical optimizer used for the QAOA study is the gradient-free Powell's method, and the QAOA parameters β and γ are random with uniform probability from 0 to their respective highest possible value, π and 2π.

A. Minimum gap study

As we already know every vertex coloring problem has $D = c!$ degenerate solutions, which means that finding the minimum gap in AQO is not straightforward.

In Figure 1 can be seen the energy spectrum of the 2 possible ground states (GSs) and the first excited state (FES) of the problem Hamiltonian, H_p, and the minimum energy to get excited from GS 0 to the FES, Δ, for the problem connecting 2 vertices with 2 different colors. As we can see, it would be misleading to consider that Δ is equal to the instantaneous energy difference between the initial state GS 0 and the FES, as it is done in a non-degenerate problem. Although one indeed starts with the GS 0, which is also the ground state of H_0, it can get excited to the GS 1 and then to the FES with less energy than the difference between the GS 0 and the FES energies, which must be accounted to compute Δ.

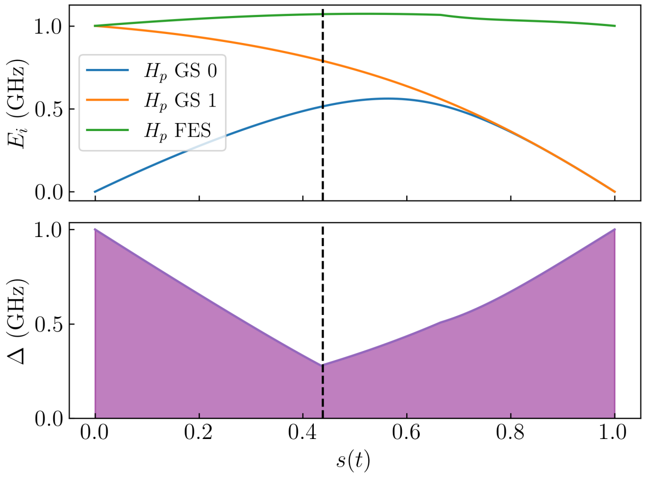


FIG. 1: Instantaneous energies of the H_p ground states (GSs) and first excited state (FES), E_i, (top), and instantaneous gap energy, Δ, (bottom), evolution during the scheduling time, $s(t)$. The studied case has $w = 2$ connected vertices colorable with $c = 2$ colors. The colored area is the allowed Δ for adiabatic evolution and the dashed vertical line marks where the minimum gap happens.

B. Simulation and comparison between the AQO and the QAOA

To analyze the quality of the outcome of our simulations, firstly, we have to define the probability of success, which, for a degenerate system, is the sum of the final state overlap with each possible solution,

$$\sum_{s=1}^{D} |\langle\psi_s|\psi_f\rangle|^2, \quad (7)$$

where ψ_s is a targeted solution that in our case is a state of the computational basis, and ψ_f is the final state of the optimization process either for AQO or QAOA.

With this in mind, we performed 20 Erdős-Rény random graphs [12] simulations for both algorithms and several cases of v vertices and c colors, using an edge probability of $p = 0.5$.

We consider superconducting flux qubits as the hardware implementation from which we draw the data for our benchmark metrics, as they are a widespread kind of qubits used for both AQO and QAOA. According to [13], the energy between the ground state and first excited state is around $\omega/2\pi = 1$ GHz which leads to a time scale in the order of ns for AQO. This might be a good approximation for an idealized case of no more than 3 or 4 qubits in a real *annealer* implementation, but for a larger number of qubits, it actually should be taken into account the graph topology and embedding, plus all the interactions that come along. Although in this work we are considering an idealized case, just to have an idea, according to DWave API, it leads to actual annealing times between 1 and 2000 μs.

On the one hand, we have to determine the annealing time for AQO which is mainly ruled by the minimum gap. Unfortunately, this is not studied because it is not possible due to computational limitations for the cases considered. Instead, we compare the solution quality of the algorithm while increasing the annealing time.

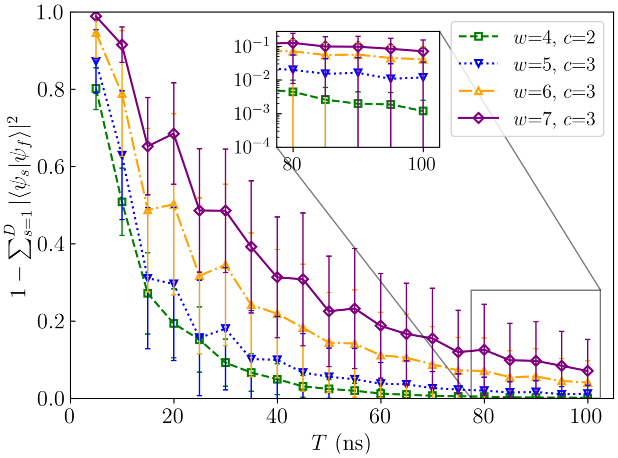


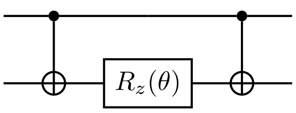
FIG. 2: Failure probability for AQO process depending on the adiabatic time, T, for several cases of w and c.

In Figure 2 we show how the failure probability for AQO and different cases diminishes with the increase of T tending to 0 for all the cases considered. Although it considerably performs worse with the number of qubits increase, as expected, and as we can see in the inset it varies by orders of magnitude.

On the other hand, the time for each QAOA layer is related to the circuit depth and the time each native gate in the quantum computer needs to be applied. Each layer is applied p times (equivalent to one circuit execution) and the outcome is evaluated n_{fev} times in order to run the classical optimization (we neglect the classical optimization time).

Note that both the mixer and phase separator need to be compiled into a set of native gates of the quantum computer. How optimal is this compilation depends on whether these gates can be applied in parallel and this, along with the specific application times of each native gate, determines the runtime of each layer.

Each term of (6) can be compiled as,

$e^{\sigma^z\otimes\sigma^z} =$ 

following the Staircase Algorithm, [14]. The CNOT gates require a much larger implementing time compared with the rotation, as terms e^{σ^z} are assigned to gates T, which their application time can be neglected.

For the mixer Hamiltonian,

$$e^{-i\sigma^x} = He^{-i\sigma^z}H = HTH,$$

the gate application time of T can still be neglected, but the two Hadamards are considerable in comparison with the CNOT.

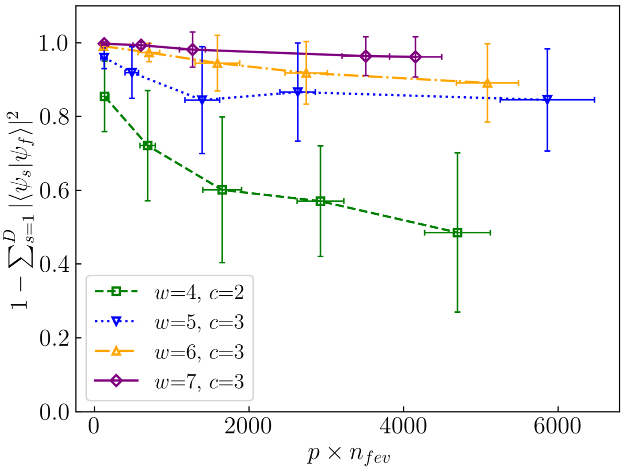


FIG. 3: Failure probability for the QAOA depending on the number of layers (which ranges from 1 to 5) times the number of function evaluations, $p\times n_{fev}$, for several cases of w and c. Note that to obtain the total execution time one needs to define the compilation procedure used, which determines T_{layer}.

In the first term of (6), the only one that will be hardware costly is the quadratic one, as the other ones just represent phases and rotations that are negligible in terms of estimating the runtime. Setting $A=1$ without loss of generality, it can be seen that,

$$\left(\sum_{i=1}^{c}\sigma^z_{v,i}\right)^2 = \sum_{i,j=1}^{c}\sigma^z_{v,i}\sigma^z_{v,j}. \tag{8}$$

The QAOA runtime grows as $T_{QAOA} = T_{layer}(p\times n_{fev})$, and for the worst-case scenario without parallelizing any gates in the aforementioned compilation,

$$\begin{aligned} T_{QAOA} \approx \big[&(w\times c^2 + m\times c)\times(2\,t_{CNOT}) \\ &+(w\times c)\times(2\,t_H)\big]\times(p\times n_{fev}) \end{aligned} \tag{9}$$

where t_{CNOT} is the application time for the CNOT gate, which is around 20 ns, and $t_H = 6$ ns for the Hadamard gate, according to [15].

In Figure 3 is shown how the failure probability slowly decreases for QAOA depending on the number of times a layer is applied, $p\times n_{fev}$, for different cases. As the number of qubits increases means worse performance again, but in comparison with the AQO, for the simplest case of 8 qubits, QAOA already struggles to reach a 0.5 overlap for 5 layers. Even assuming the best-case scenario, it is clear that the total runtime for QAOA, T_{QAOA}, is much larger than the time to run the AQO process, T, as for all cases, the final convergence value of $p\times n_{fev}$ is already larger than T, without taking into account the proportionality with the layer time, T_{layer}, which is in the order of at least 100 ns. Just to have an idea of how this grows (without optimizing the gates compilation), for the 8 qubits case, $w = 4$, $c = 2$, and considering approximately $m = [(w-1) \times w]/4$ as the number of possible vertices pairs (edges) divided by 2 due to the 0.5 of an edge appears, according to (9) the runtime for QAOA grows as $T_{QAOA} \approx 976 \times (p \times n_{fev})$ ns.

VII. CONCLUSIONS

From the previous results, it is clear that the AQO significantly outperforms the QAOA in terms of time and performance in the idealized conditions considered, even if a suitable compilation procedure is designed. Despite both the classical optimizer and the compilation procedure can be more competitive, the AQO schedule can also be improved. Using the simplest implementation of both algorithms, the AQO achieves an overlap with the ground state of over 0.9 at $T = 100$ ns, for all cases ranging from 8 to 21 qubits. The QAOA with five layers can roughly surpass an overlap of 0.5 for 8 qubits with the performance decreasing below 0.25 as the number of qubits grows. However, its performance mainly improves with the number of layers, p, as expected.

Another important feature is that we are not considering temperature for the adiabatic evolution and the connectivity of the quantum chip. If these phenomena were taken into account, the AQO on current hardware would require an embedding process, which is an NP-hard problem in itself, forcing us to use many more ancillary qubits and therefore increasing the likelihood of errors. Along these lines, a more realistic comparison would take into account the existence of errors, both random and systematic, in the application of gates or analog Hamiltonians.

For all these reasons, a future direction for this research would be to implement both algorithms in actual quantum computers instead of simulating them and benchmark the current state of the art of current hardware. By implementing the algorithms in real quantum hardware we also expect to be able to perform a scaling analysis of the time to get a solution for both approaches as more qubits could be used, as we are limited in the number of qubits we can simulate with classical computers. Nevertheless, the preliminary simulations of this study are sound and suggest that for optimization problems the analog model of computation might have clear advantages over the digital algorithms, in different quantum computing platforms.

Acknowledgments

I am profoundly grateful for the great work and advice from my supervisors both from Qilimanjaro, Marta P Estarellas and Jordi Riu, as well as, my former university professor at the UB, Lluís Garrido. Apart from that I would like to thank my family, friends, and Qilimanjaro colleagues for the discussions and knowledge, with a special mention for the support provided by Josep Bosch and David López.

[1] Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., & Coles, P. J. (2020). Variational Quantum Algorithms. *arXiv.* https://doi.org/10.1038/s42254-021-00348-9

[2] Farhi, E., Goldstone, J., & Gutmann, S. (2014). A Quantum Approximate Optimization Algorithm. *arXiv.* https://arxiv.org/abs/1411.4028v1

[3] Farhi, E., Goldstone, J., Gutmann, S., & Sipser, M. (2000). Quantum Computation by Adiabatic Evolution. *arXiv.* https://arxiv.org/abs/quant-ph/0001106v1

[4] Kato, T. (1950). On the Adiabatic Theorem of Quantum Mechanics. *Journal of the Physical Society of Japan, 5*(6), 435–439. https://doi.org/10.1143/JPSJ.5.435

[5] Binkowski, L., Koßmann, G., Ziegler, T., & Schwonnek, R. (2023). Elementary Proof of QAOA Convergence. *arXiv.* https://arxiv.org/abs/2302.04968

[6] Pelofske, E., Bärtschi, A., & Eidenbenz, S. (2023). Quantum Annealing vs. QAOA: 127 Qubit Higher-Order Ising Problems on NISQ Computers. *arXiv.* https://doi.org/10.1007/978-3-031-32041-5_13

[7] Albash, T., & Lidar, D. A. (2016). Adiabatic Quantum Computing. *arXiv.* https://doi.org/10.1103/RevModPhys.90.015002

[8] Lucas, A. (2014). Ising formulations of many NP problems. *Frontiers in Physics, 2.* https://doi.org/10.3389/fphy.2014.00005

[9] Formanowicz, P., & Tanaś, K. (2012). A survey of graph coloring - its types, methods and applications. *Foundations of Computing and Decision Sciences, 37*(3), 223-238. https://doi.org/10.2478/v10209-011-0012-y

[10] Stollenwerk, T., Hadfield, S., & Wang, Z. (2020). Toward Quantum Gate-Model Heuristics for Real-World Planning Problems. *IEEE Transactions on Quantum Engineering, 1*, 1-16. https://doi.org/10.1109/TQE.2020.3030609

[11] Stollenwerk, T., Lobe, E., & Jung, M. (2018). Flight Gate Assignment with a Quantum Annealer. *arXiv.* https://arxiv.org/abs/1811.09465

[12] Erdős, P., & Rényi, A. (1959). On Random Graphs. I. *Publicationes Mathematicae, 6*(3-4), 290-297. https://doi.org/10.5486/PMD.1959.6.3-4.12

[13] Ivakhnenko, O. V., Shevchenko, S. N., & Nori, F. (2022). Nonadiabatic Landau-Zener-Stuckelberg-Majorana transitions, dynamics, and interference. *arXiv.* https://doi.org/10.1016/j.physrep.2022.10.002

[14] Mansky, M. B., Puigvert, V. R., & Castillo, S. L. (2023). Decomposition Algorithm of an Arbitrary Pauli Exponential through a Quantum Circuit. *arXiv.* https://arxiv.org/abs/2305.04807

[15] Suchara, M., Faruque, A., Lai, C., Paz, G., Chong, F. T., & Kubiatowicz, J. (2013). QuRE: The Quantum Resource Estimator toolbox. *arXiv.* https://doi.org/10.1109/ICCD.2013.6657074

APPENDIX: Erdős-Rény random graphs distribution

The error bar in Figures 2 and 3 measure the distribution of the Erdős-Rény random graphs, because there are no systematical errors because we are considering ideal conditions for both cases. In this appendix study, we perform a simulation and analysis of a particular case for AQO and a larger number of graphs, specifically 100 random graphs, instead of 20. These simulations are very time expensive, so this is the reason why just one case is explored, and this is not applied throughout the whole work.

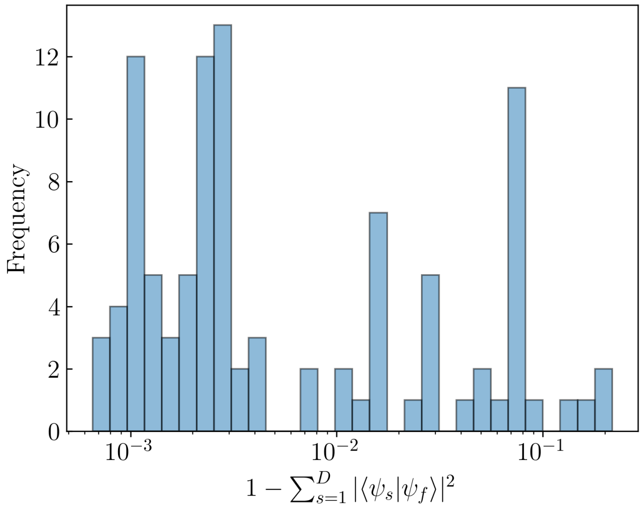


FIG. 4: Frequency for 100 hundred Erdős-Rény random graphs failure probability of logarithmic width boxes in the x-axis. The case considered is a AQO process of, $T = 100$ ns and $w = 6$ and $c = 3$.

The frequencies of the failure probability are plotted in a histogram in Figure 4. This refers to a $T = 100$ ns case of AQO with $w = 6$ and $c = 3$, so it constitutes a 18 qubits problem. We can appreciate that most colorable graphs are located in the same zone around a failure probability of 0.1%, which constitute the low and medium complexity graphs with less edges. Then, there are a few more graphs of higher failure probability between 1% and 10% that are expected to contain more edges, thus it is likely to be more complex. This distribution is due to the probability of 0.5 to create an edge makes it more likely to find graphs around $(w-1) \times w/2$ edges, but they have to be colorable, so this is lean to fewer edges as the more edges and complexity, the less colorable graphs.