PAPER • OPEN ACCESS

# Parallel Quantum Computation Approach for Quantum Deep Learning and Classical-Quantum Models

To cite this article: E.D. Payares and J.C. Martinez-Santos 2021 *J. Phys.: Conf. Ser.* **2090** 012171

# Parallel Quantum Computation Approach for Quantum Deep Learning and Classical-Quantum Models

**E.D. Payares[1], J.C. Martinez-Santos[1]**

[1]Faculty of Engineering, Universidad Tecnológica de Bolívar, Cartagena de Indias, Colombia

E-mail: `epayares@utb.edu.co`

**Abstract.** The paradigm of Quantum computing and artificial intelligence has been growing steadily in recent years and given the potential of this technology by recognizing the computer as a physical system that can take advantage of quantum mechanics for solving problems faster, more efficiently, and accurately. We suggest experimentation of this potential through an architecture of different quantum models computed in parallel. In this work, we present encouraging results of how it is possible to use Quantum Processing Units analogically to Graphics Processing Units to accelerate algorithms and improve the performance of machine learning models through three experiments. The first experiment was a reproduction of a parity function, allowing us to see how the convergence of a given Quantum model is influenced significantly by computing it in parallel. For the second and third experiments, we implemented an image classification problem by training quantum neural networks and using pre-trained models to compare their performances with the same experiments carried out with parallel quantum computations. We obtained very similar results in the accuracies, which were close to 100% and significantly improved the execution time, approximately 15 times faster in the best-case scenario. We also propose an alternative as a proof of concept to address emotion recognition problems using optimization algorithms and how execution times can be positively affected by parallel quantum computation. To do this, we use tools such as the cross-platform software library PennyLane and Amazon Web Services to access high-end simulators with Amazon Braket and IBM quantum experience.

## 1. Introduction

Parallel computing nowadays is used to improve the performance of many systems, models, and algorithms significantly, thanks to its capacity to complete large and complex tasks in a reasonable time [1–4]. In recent years, there has been an increment of outstanding achievements on how Quantum computing (QC) can solve very complex problems that are impossible for a Classical Computer (CC) to solve in a suitable time [5]. In addition, there are ways to exploit their capabilities to enhance classic applications and perform specific tasks so that it is impossible for a CC to keep up with the pace, which is called quantum advantage and is one of the leading research interests in the field today. Thus, there is plenty of hope in how QC can improve existing technologies, one of the highlights of which is the case of machine learning [6].

Quantum machine learning is a field of study that investigates the interaction of concepts from quantum computation and machine learning [7, 8], with great achievements in the last few years finding a place in implementations such as classification [9–11] and applications in many fields, from image and emotion recognition to cybersecurity systems [12–16].

The limits of what machines can learn have always been defined by the computer hardware we run our algorithms on. For example, parallel Graphics processing unit (GPU) clusters enable modern-day deep learning with neural networks.

Quantum machine learning extends the hardware pool for machine learning by an entirely new type of computing device: The quantum computer. This comparison makes sense since both are alternatives to speed up computation for expensive computational resource tasks. However, the scalability of the problems that we can address is quite different. To the best of our knowledge, this is the first time that a study has been carried out to unveil the potential of parallel quantum computation for Noisy Intermediate-scale Quantum (NISQ) [17,18] hardware in near-term applications such as quantum deep learning.

## 2. Methods

### *2.1. Framework and Materials*

For the development of this work and presentation of results, we mainly use the PennyLane framework, which is a cross-platform Python library for differentiable programming of quantum computers, allowing us to build and optimize hybrid computations [19]. We can perform these executions on real quantum hardware or a classical simulator. Both options are available today on the platforms of IBM, Amazon, among others.

### *2.2. Parallel Quantum Computation Architecture*

Quantum computation is known for its potential of improving performance in the execution of tasks thanks to a physical phenomenon called quantum parallelism, which consists of the capability of quantum systems to do many evaluations simultaneously thanks to the property of superposition [20]. However, for this implementation, we extend this concept as parallel quantum computation. The difference is that regardless of superposition. There is parallel execution of evaluations of circuits taking place in two or more Quantum Processing Units (QPUs) or in a simulator that has enabled distributed parallel executions of circuit batches, such as the mainly used in this work, the SV1 by Amazon Braket. Theoretically, there is also an analogy between parallel computing and parallel quantum computing proposed in [21], where is defined the Quantum Parallel Complexity **QNC** in Equation (1), which is the quantum version of parallel complexity **NC** and states that to design a shallow parallel circuit for a given quantum operator, we want to be able to use additional quantum bits (qubits) or *"ancillae"* as workspace in the computation, equivalent to other processors in a parallel quantum computer. Also, as is presented in [22] it is possible to emulate parallel quantum computation to process signals in parallel mainly through encoding them into a superposition state of two signals, $\psi$ and $\varphi$ (2).

$$\mathbf{QNC} = \cup_k \mathbf{QNC}^k \;\; where \;\; \mathbf{QNC}^k \parallel \mathcal{O}(n \log{}^k n) \tag{1}$$

$$\langle \varphi | \psi \rangle = \frac{1}{T} \int_0^T \varphi(t)^* \psi(t)\, dt \tag{2}$$

We used an architecture that systematically proved the advantage of parallel quantum computation in different quantum processors in various models for different data types for our implementation.

The circuit in Figure 1 shows a generic quantum classifier mainly used for the realization of the experiments of this work. It consists of a minimum of two layers and single-qubit rotations, and CNOT gates as entanglers (3) and a measurement layer (4) mutable depending on the experiment in each one of them, based on the model presented in [23].

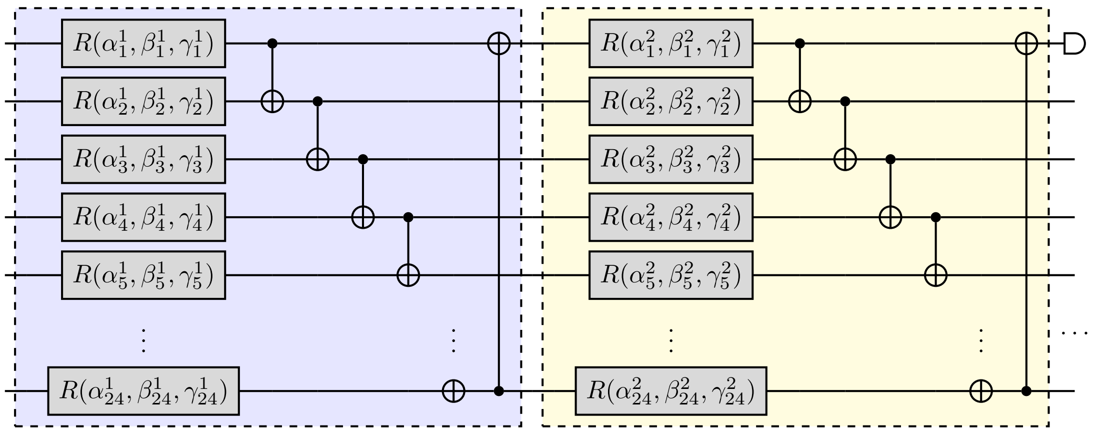
$R(\alpha_1^1, \beta_1^1, \gamma_1^1)$
$R(\alpha_2^1, \beta_2^1, \gamma_2^1)$
$R(\alpha_3^1, \beta_3^1, \gamma_3^1)$
$R(\alpha_4^1, \beta_4^1, \gamma_4^1)$
$R(\alpha_5^1, \beta_5^1, \gamma_5^1)$
$R(\alpha_{24}^1, \beta_{24}^1, \gamma_{24}^1)$
$R(\alpha_1^2, \beta_1^2, \gamma_1^2)$
$R(\alpha_2^2, \beta_2^2, \gamma_2^2)$
$R(\alpha_3^2, \beta_3^2, \gamma_3^2)$
$R(\alpha_4^2, \beta_4^2, \gamma_4^2)$
$R(\alpha_5^2, \beta_5^2, \gamma_5^2)$
$R(\alpha_{24}^2, \beta_{24}^2, \gamma_{24}^2)$

**Figure 1.** General representation of the circuit ansatz used on each experiment. There are four qubits and two layers for the minimum configuration, 24 qubits for the maximum, and at least one measurement in the Z basis of the Bloch sphere in one qubit. Note that this is a general representation, which means that it can vary depending on the experiment.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

$$\mathcal{M}(|y\rangle) = \begin{bmatrix} \langle y| Z \bigotimes \mathbb{I} \bigotimes \mathbb{I} \bigotimes \mathbb{I} |y\rangle \\ \langle y| \mathbb{I} \bigotimes Z \bigotimes \mathbb{I} \bigotimes \mathbb{I} |y\rangle \\ \langle y| \mathbb{I} \bigotimes \mathbb{I} \bigotimes Z \bigotimes \mathbb{I} |y\rangle \\ \langle y| \mathbb{I} \bigotimes \mathbb{I} \bigotimes \mathbb{I} \bigotimes Z |y\rangle \end{bmatrix} \tag{4}$$

As stated above, the parallel execution of the circuit shown in Figure 1 occurs in two ways. First, a given circuit runs on many devices asynchronously, where each device could be a simulator or an actual QPU. Thus allowing the collection of shots on the circuit to be more efficient, or in batches in one device, where the same circuit is executed multiple times in parallel. The SV1 simulator of Amazon Braket allows us to execute up to 20 circuits in parallel or simultaneously. We can harness this capability during circuit training, which requires lots of circuit variations to execute. As we know, calculating the gradient involves multiple device executions. For each trainable parameter, we must run our circuit on the device typically more than once. Practical applications involve many trainable parameters. The result of this is a vast number of device executions for each optimization step.

This approach allows us to compare each execution with its respective result with three experiments in which we see the ability of quantum parallel computation to improve quantum deep learning algorithms. These experiments were carried out with the same parameters for every device to prevent bias in the simulations. Furthermore, we used 1000 shots per cost function evaluation on every experiment. In the following sections, the presentation of results labeled as "unparalleled", means sequential execution of the circuits for each training process.

It is important to emphasize that the devices used in the experiments were all simulators. This means that we have not considered the influence of noise on the models because we wanted to keep the integrity of each experiment and each device as close as possible. So as not to fall into the error of reaching conclusions based on a single experience, but in general. The parameters for this case study are the same for each experiment, but be aware that noise is an external factor that influences the behavior and performance of current quantum devices.

## 3. Experiments and Results

### *3.1. Experiment 1: Parity function*

In this experiment, we demonstrate that our variational quantum classifier can reproduce the parity function (5) more efficiently through parallel execution on two SV1 simulators by amazon Braket in the training process.

$$f : x \in \{0,1\}^{\otimes n} \to y = \begin{cases} 1 \text{ if uneven number of ones in } x \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

This optimization example demonstrates how to encode binary inputs into the initial state of the variational circuit, which is simply a computational basis state. We can see in Figure 2 and Figure 3 that for the parallel execution, we begin to see an increase in the convergence of the model from about the 6th iteration approximately. For the cost function in Figure 2 we evidenced a significant improvement for the parallel execution, reaching convergence in 10 iterations approximately with a considerable advantage over the 25 iterations of the unparalleled implementation of the same model. For the accuracy in Figure 3, although the model behaves similarly, we evidence a difference of 5 iterations between the parallel and unparalleled execution by the time each of them reaches maximum convergence.

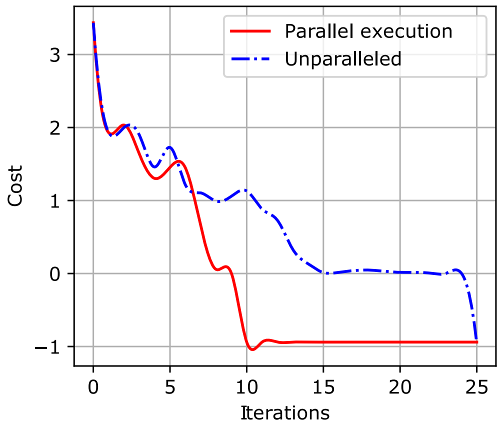

**Figure 2.** Cost function during the training process of experiment 1.

As we can see in Table 1, the parallel execution achieved convergence in 1.5 times fewer iterations in terms of accuracy. This result is relevant because it gives us an eye-opener on how parallel execution can improve the efficiency and performance of quantum models. Furthermore, if we scale this experiment and apply it to a problem of higher complexity, that value could become very significant.

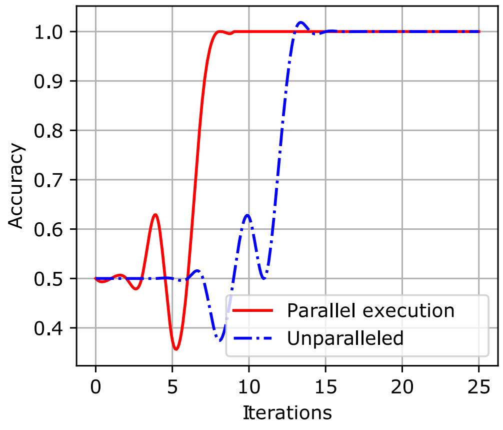

**Figure 3.** Accuracy during the training process of experiment 1.

**Table 1.** Experiment 1.

| **QPU** | **Execution** | **Convergence Iterations** |
|---|---|---|
| `ibmq_simulator` | Unparalleled | 14 |
| `SV1` | Parallel | 9 |

## *3.2. Experiment 2: Hymenoptera Dataset Classification*

In this experiment, we used the scheme presented in [13], where we numerically trained and tested the model, which is a quantum model inspired by the official PyTorch tutorial on classical transfer learning using PennyLane with the PyTorch interface [19]. In this case, the parallel execution occurs on two devices.

In Figure 4 we can see a preview of the data, which consist of high resolution images with two classes, *ants* and *bees*. The data was encoded into quantum states via local embedding. It means that all qubits are first initialized in a balanced superposition of up and down states. Then, they are rotated according to the input parameters.

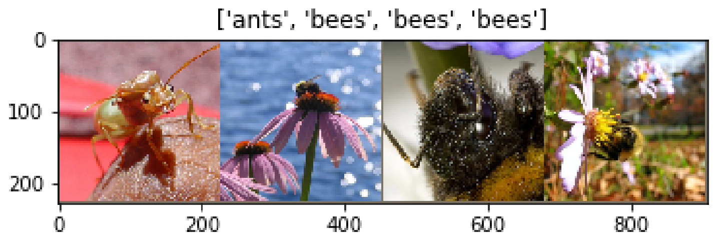

**Figure 4.** Random batch of four images sampled from the train dataset.

Figure 5 shows the evolution through the training process of the loss landscape for each model. First, we trained the variational parameters of the model for 30 epochs over the training dataset with a batch size of four and an initial learning rate of $n = 0.0004$, which was successively reduced by a factor of 0.1 every ten epochs [13]. Then, after each epoch, the model was validated concerning the test dataset, obtaining a maximum accuracy of 0.9477.

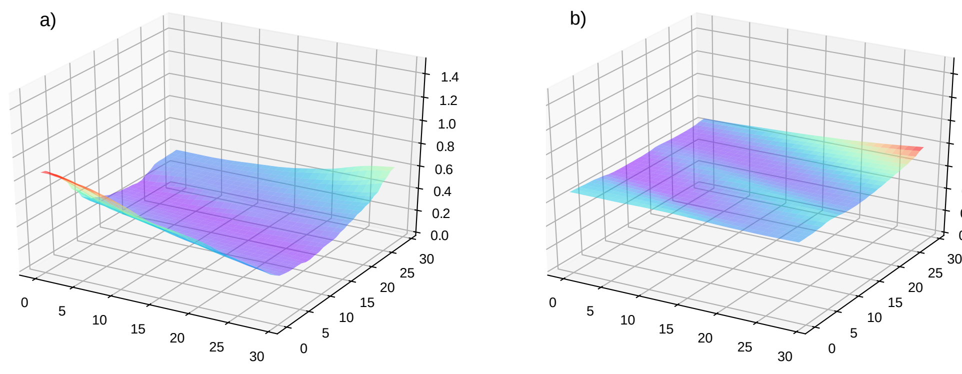

**Figure 5.** **a)** Surface plot of loss landscape during training of the dressed quantum neural network with unparalleled execution. **b)** Surface plot of loss landscape during training of the dressed quantum neural network with parallel execution.

**Table 2.** Experiment 2.

| **QPU** | **Execution** | **Accuracy** | **Loss** | **Run time** |
|---|---|---|---|---|
| `SV1` | Unparalleled | 0.9477 | 0.367 | 131m 51s |
| `SV1` | Parallel | 0.9412 | 0.6432 | 17m 47s |

Also, we can see in Figure 5 that the loss landscape doesn't vary too much from the unparalleled execution to the parallel execution. The reason is that the model's hyperparameters were optimized successfully in both cases, with a bit of loss peak at the end of the training process with parallel execution. However, in Table 2 we see that where the significant difference is in the execution time, where we evidenced a remarkable speedup of 7.52 times faster run time.

### *3.3. Experiment 3: CIFAR Transfer Learning Classification*

In this experiment, we make use of a pre-trained model and implementation presented in [13], where we evaluate its performance with the same architecture as Section 3.2. Still, with a different dataset, the standard CIFAR-10 restricted to two classes, *cats* and *dogs* as we can see in Figure 6. The data was encoded into quantum states in the same way as the previous section via local embedding. All qubits are first initialized in a balanced superposition of up and down states. Then they are rotated according to the input parameters, determined by the matrix of the image previously processed for use in the model.

As we can see in Table 3, the results are nearly the same as expected except for the execution time, in which we evidenced a speedup of approximately 15.17 times faster execution. Also, in Figure 7 we see the loss landscape of the model, which was the same for both implementations since it is a pre-trained model. However, we expect that similarly to the experiment in Section 3.2 there will be variations in the loss of the model since the parallel computation allows better performance as hyperparameter optimization is done more efficiently.

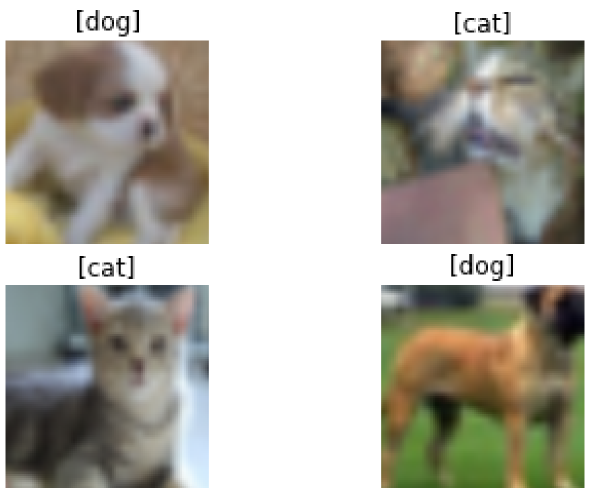

**Figure 6.** Random batch of four blurry images sampled from the test dataset classified by the classical-quantum model. Predictions are reported in square brackets above each image.

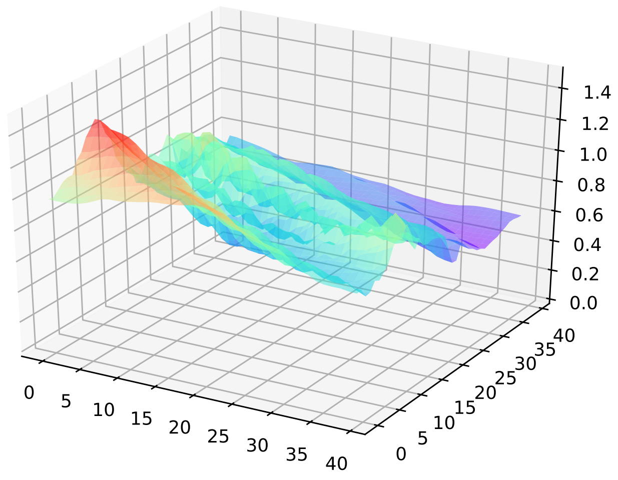

**Figure 7.** Loss landscape of the model during the evaluation process.

**Table 3.** Experiment 3.

| QPU | Execution | Accuracy | Loss | Run time |
|---|---|---|---|---|
| `SV1` | Unparalleled | 0.8250 | 0.4059 | 19m 42s |
| `SV1` | Parallel | 0.8250 | 0.4059 | 1m 28s |

## 4. Proof of Concept: Graph Optimization

In this proof of concept, we are going to evaluate the feasibility of using the implementation presented in [14], with a parallel architecture to exploit to the full the capabilities of quantum computing in graph optimization as a near-term application.

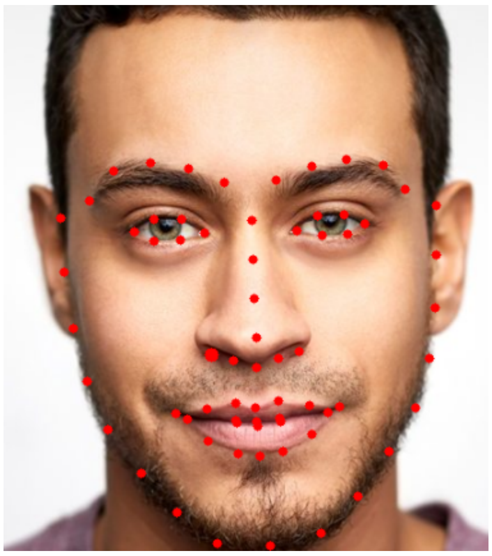

**Figure 8.** Landmarks recognition of human face as points. The image of the human face is *taken from:* [24]

The implementation presented in [14] consists of a quantum classifier for facial expression recognition using graphs. Each graph was encoded in a quantum state, making use of the elements of the adjacency vector into the amplitudes of the quantum state, where $|G\rangle$ is the quantum state constructed in Equation (6), and $G$ is the graph.

$$|G\rangle = \frac{1}{\gamma}\sum_{k=1}^{d} g_k\,|k\rangle \tag{6}$$

In Figure 8 we can see the example of a human face whose landmarks have been recognized and drawn as points, which will act as nodes in the graph associated with each class. As this is not an experimental reproduction, we scale the problem for this proof of concept. We take 20 nodes that represent the qubits of our architecture described in Figure 1, and the landmark points of the main feature, and make the graph. In this case, as we don't classify the expression of the face, our graph is arbitrary to fulfill the purpose of this proof of concept whose basis is to demonstrate the efficiency of parallel evaluation of circuits and how we can exploit this for quantum deep learning models. We address an optimization problem called *the maximum cut* , which consists of finding a partition of nodes into two sets, such that the number of edges in the graph with endpoints in different groups is maximized. That is achieved using the Quantum Approximate Optimization Algorithm (QAOA), a widely-studied method for solving combinatorial optimization problems on NISQ devices. QAOA begins by associating the optimization problem with a cost Hamiltonian $H_C$ (7) and choosing a mixer Hamiltonian $H_M$. It proceeds by repetitively applying multiple layers of the unitaries $\exp(-i\gamma_i H_C)$ and $\exp(-i\alpha_i H_M)$ with controllable parameters $\gamma_i$ and $\alpha_i$.

$$H_C = \frac{1}{2}\sum_{(i,j)\in E(G)} \left(Z_i Z_j - \mathbb{I}\right) \tag{7}$$

As we approach the problem, the algorithm just required us to pass a graph to it. This graph, shown in Figure 9 represents the landmark points of the feature of a face, in which we perform QAOA to solve the maximum cut, which is an NP-hard problem in terms of computational complexity. This problem consists in finding the partition of the graph's nodes into two groups so that the number of edges crossed or "cut" by the partition is maximized. We choose this problem because it is very computationally demanding.

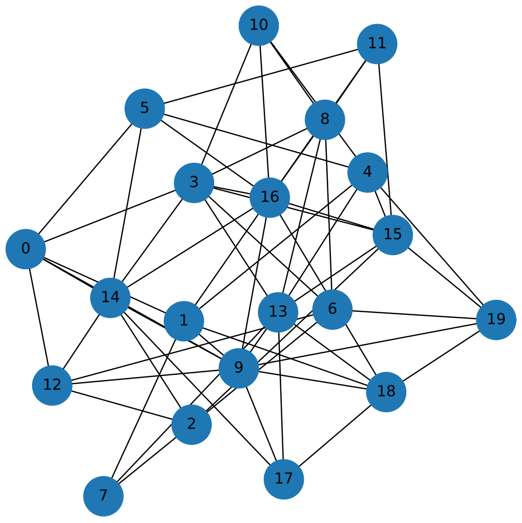

**Figure 9.** Arbitrary graph of 20 nodes representing the landmark points.

Table 4 shows the summary of results of this proof of concept, where we can see how important it would be to fully address graph optimization problems in quantum machine learning applications by running quantum circuits in parallel. We have evidenced a speedup of about 12.8 times faster execution for the QAOA in question.

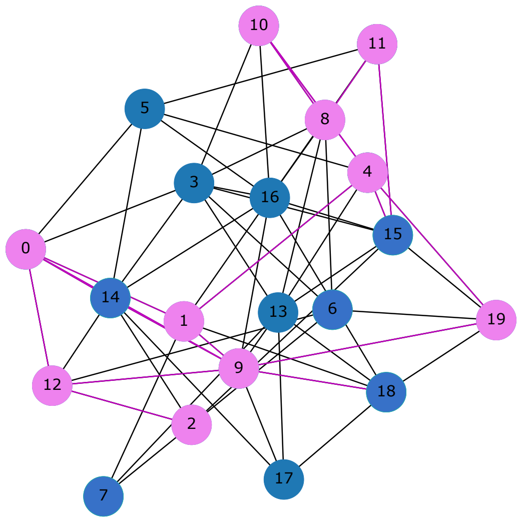

**Figure 10.** Highlighted edges and nodes results of maximum cut.

In Figure 10 we can see the nodes resulting from the QAOA optimization of maximum cut, where the purple violet represents the set $S$, and the blue ones mean the set $T$. Their shared edges are highlighted in magenta.

**Table 4.** Summary of results of the proof of concept.

| QPU | Execution | Problem | Run time |
|---|---|---|---|
| `braket.local` | Unparalleled | Maximum cut | 219.6m |
| `SV1` | Parallel | Maximum cut | 17.16m |

## 5. Conclusions and Future Work

This work presented an implementation of parallel quantum computation for near-term applications. We proved the efficiency of this methodology by conducting three experiments, and we proposed a proof of concept. However, although the results are encouraging, quantum parallelism is still very limited to the hardware available today to demonstrate the true potential of quantum computers and how they process information. Still, as shown in this paper, there are viable alternatives to improve the way quantum algorithms work in the rapidly advancing NISQ era.

For future work, we aim to fully address the proof of concept introduced in this paper by creating a model that would perform graph optimization for quantum machine learning tasks through a parallel quantum architecture. We also aim to introduce noise models to the simulations to see how the results are affected by this parameter in order to have a closer look at the behavior of quantum models in real hardware.

**Acknowledgment**

We would like to thank the anonymous reviewers for their comments and feedback on the ideas in this paper. This work was supported by Universidad Tecnológica de Bolívar research department.

## References

[1] Scott L R, Clark T and Bagheri B 2021 *Scientific parallel computing* (Princeton University Press)
[2] Alerstam E, Svensson T and Andersson-Engels S 2008 *Journal of biomedical optics* **13** 060504
[3] Cybenko G 2017 Parallel computing for machine learning in social network analysis *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (IEEE) pp 1464–1471
[4] Cavuoti S, Garofalo M, Brescia M, Longo G, Ventre G *et al.* 2013 Genetic algorithm modeling with gpu parallel computing technology *Neural Nets and Surroundings* (Springer) pp 29–39
[5] Arute F, Arya K, Babbush R, Bacon D, Bardin J C, Barends R, Biswas R, Boixo S, Brandao F G S L, Buell D A, Burkett B, Chen Y, Chen Z, Chiaro B, Collins R, Courtney W, Dunsworth A, Farhi E, Foxen B, Fowler A, Gidney C, Giustina M, Graff R, Guerin K, Habegger S, Harrigan M P, Hartmann M J, Ho A, Hoffmann M, Huang T, Humble T S, Isakov S V, Jeffrey E, Jiang Z, Kafri D, Kechedzhi K, Kelly J, Klimov P V, Knysh S, Korotkov A, Kostritsa F, Landhuis D, Lindmark M, Lucero E, Lyakh D, Mandrà S, McClean J R, McEwen M, Megrant A, Mi X, Michielsen K, Mohseni M, Mutus J, Naaman O, Neeley M, Neill C, Niu M Y, Ostby E, Petukhov A, Platt J C, Quintana C, Rieffel E G, Roushan P, Rubin N C, Sank D, Satzinger K J, Smelyanskiy V, Sung K J, Trevithick M D, Vainsencher A, Villalonga B, White T, Yao Z J, Yeh P, Zalcman A, Neven H and Martinis J M 2019 *Nature* **574** 505–510 ISSN 1476-4687
[6] Ristè D, da Silva M P, Ryan C A, Cross A W, Córcoles A D, Smolin J A, Gambetta J M, Chow J M and Johnson B R 2017 *npj Quantum Information* **3** 1–5 ISSN 2056-6387 URL `https://www.nature.com/articles/s41534-017-0017-3`
[7] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195–202
[8] Schuld M, Sinayskiy I and Petruccione F 2015 *Contemporary Physics* **56** 172–185
[9] Kerenidis I and Luongo A 2020 *Physical Review A* **101** 062327 ISSN 2469-9926, 2469-9934 arXiv: 1805.08837 URL `http://arxiv.org/abs/1805.08837`

[10] Rebentrost P, Mohseni M and Lloyd S 2014 *Physical Review Letters* **113** 130503 ISSN 0031-9007, 1079-7114 URL `https://link.aps.org/doi/10.1103/PhysRevLett.113.130503`

[11] Havlíček V, Córcoles A D, Temme K, Harrow A W, Kandala A, Chow J M and Gambetta J M 2019 *Nature* **567** 209–212 ISSN 0028-0836, 1476-4687 URL `http://www.nature.com/articles/s41586-019-0980-2`

[12] Li Y, Zhou R G, Xu R, Luo J and Hu W 2020 *Quantum Science and Technology* **5** 044003 URL `https://doi.org/10.1088/2058-9565/ab9f93`

[13] Mari A, Bromley T R, Izaac J, Schuld M and Killoran N 2020 *Quantum* **4** 340 ISSN 2521-327X URL `http://dx.doi.org/10.22331/q-2020-10-09-340`

[14] Mengoni R, Incudini M and Di Pierro A 2021 *Quantum Machine Intelligence* **3** 8 ISSN 2524-4906, 2524-4914 URL `http://link.springer.com/10.1007/s42484-020-00035-5`

[15] Payares E and Martinez-Santos J C 2021 Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview *Quantum Computing, Communication, and Simulation* ed Hemmer P R and Migdall A L (Online Only, United States: SPIE) p 47 ISBN 9781510642331 9781510642348 URL `https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11699/2593297/Quantum-machine-learning-for-intrusion-detection-of-distributed-denial-of/10.1117/12.2593297.full`

[16] Killoran N, Bromley T R, Arrazola J M, Schuld M, Quesada N and Lloyd S 2019 *Physical Review Research* **1** 033063 ISSN 2643-1564 URL `https://link.aps.org/doi/10.1103/PhysRevResearch.1.033063`

[17] Bharti K, Cervera-Lierta A, Kyaw T H, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann J S, Menke T, Mok W K, Sim S, Kwek L C and Aspuru-Guzik A 2021 Noisy intermediate-scale quantum (nisq) algorithms (*Preprint* `2101.08448`)

[18] Preskill J 2018 *Quantum* **2** 79 ISSN 2521-327X URL `http://dx.doi.org/10.22331/q-2018-08-06-79`

[19] Bergholm V, Izaac J, Schuld M, Gogolin C, Alam M S, Ahmed S, Arrazola J M, Blank C, Delgado A, Jahangiri S, McKiernan K, Meyer J J, Niu Z, Száva A and Killoran N 2020 *arXiv:1811.04968 [physics, physics:quant-ph]* ArXiv: 1811.04968 URL `http://arxiv.org/abs/1811.04968`

[20] Nielsen M A and Chuang I L 2011 *Quantum Computation and Quantum Information: 10th Anniversary Edition* hardcover ed (Cambridge University Press) ISBN 978-1107002173

[21] Moore C and Nilsson M 2001 *SIAM Journal on Computing* **31** 799–815 URL `https://doi.org/10.1137/s0097539799355053`

[22] La Cour B R, Andrew Lanham S and Ostrove C I 2018 *2018 IEEE International Conference on Rebooting Computing (ICRC)* URL `http://dx.doi.org/10.1109/ICRC.2018.8638597`

[23] Schuld M, Bocharov A, Svore K M and Wiebe N 2020 *Physical Review A* **101** ISSN 2469-9934 URL `http://dx.doi.org/10.1103/PhysRevA.101.032308`

[24] *Genetics helps determine the shape of a person's face.* URL `https://www.medicalnewstoday.com/articles/asymmetrical-face`