Investigation on Quantum Computers

Zaher Mohammed Faruk

**FINAL YEAR BSC PROJECT REPORT 2023/2024**

This report is presented in partial fulfilment of the requirements for the award of the degree of B.Sc. with Honours in Physics

1. Abstract

This project presents an evaluation of two leading quantum computing platforms: IBM Quantum (which utilises superconducting qubits) and Xanadu’s photonic quantum systems (both accessible via the PennyLane Python library) [1]. Their performance in solving quadratic unconstrained binary optimisation (QUBO) problems, benchmarking metrics such as computational efficiency (execution time, resource usage), solution accuracy (approximation ratio), and error resilience (gate errors for IBM, photon loss for Xanadu) were tested [2].

The study highlights the fundamental differences between the two architectures. For instance, Xanadu’s photonic systems demonstrate advantages in parallel processing for specific QUBO instances, while IBM’s superconducting qubits excel in iterative gate-based optimisation routines. Scalability challenges, hardware-specific error mitigation strategies (e.g., dynamical decoupling for IBM, photon loss correction for Xanadu), and the role of software frameworks like PennyLane in enabling hybrid quantum-classical workflows were analysed in depth.

By testing these platforms with QUBO combinatorial optimisation problems, this work clarifies what is their practical utility and limitations. The results, supported by reproducible code (see Appendix), offer insights for researchers and developers in selecting quantum technologies tailored to combinatorial optimisation tasks. This analysis shows the importance of platform-specific algorithm design and the need for advancements in error correction to bridge the gap between theoretical potential and real-world applicability [3].

1. Acknowledgements

We sincerely thank the following people and organisations for their invaluable support, which were instrumental in making this scientific report possible:

- Dr Mark Hughes: His endless support and in-depth knowledge on Quantum Computing assisted me heavily.

- IBM: We are deeply thankful to IBM Quantum for providing access to their state-of-the-art quantum computing platform. Their advanced superconducting qubit technology enabled us to execute and benchmark quantum algorithms effectively. Additionally, we appreciate the extensive documentation and support available through the IBM Quantum platform, which significantly enhanced our understanding and implementation of quantum circuits.

- Xanadu Quantum Technologies: We express our gratitude to Xanadu Quantum Technologies for granting access to their photonic quantum computing platform and for providing the powerful PennyLane framework. The flexibility and user-friendliness of PennyLane allowed us to efficiently program and simulate quantum systems, while its compatibility with various quantum backends proved invaluable in conducting comparative studies.

1. Introduction

Quantum computing has emerged as a transformative computational approach over the past decade, leveraging principles of quantum mechanics to address problems that are difficult for classical systems. Among the diverse hardware architectures under development, superconducting qubit-based systems (exemplified by IBM Quantum) and photonic quantum computing platforms (pioneered by Xanadu Quantum Technologies) represent two leading approaches. This study provides a systematic comparison of these architectures, evaluating their performance in solving quadratic unconstrained binary optimisation (QUBO) problems - a class of combinatorial optimisation challenges with applications in logistics, finance, and materials science [4].

Technological Foundations

IBM Quantum uses superconducting transmon qubits, built using Josephson junctions. They are operated at cryogenic temperatures (≤ 15mK) to maintain quantum coherence. These systems execute gate-based quantum circuits, relying on high-fidelity single - and two-qubit operations (e.g., √𝑋, CNOT) optimised via the Qiskit framework. The architecture’s strong point is in its compatibility with iterative hybrid algorithms such as the Quantum Approximate Optimisation Algorithm (QAOA), which iteratively refines solutions through classical feedback loops [5, 6].

Xanadu’s photonic platform utilises squeezed light states in continuous-variable (CV) quantum computing. Photonic qubits (encoded in photon number or quadrature states) propagate through programmable interferometers, enabling Gaussian Boson Sampling (GBS) and CV variational algorithms. This architecture operates at room temperature and uses inherent parallelism in photonic networks, making it suited for tasks that require high-dimensional state manipulation [7, 8].

Architectural Divergences

1. Qubit Realisation:

IBM Quantum:

Superconduing transmon qubits depend on cryogenic infrastructure, operating at temperatures below 15mK, to reduce themal noise- achieved using dilution refrigerators [6]. These systems perform well in single-qubit gate operations, reaching high-fidelity levels above 99.9%, but encounter challenges in two-qubit gate fidelity, which typically sits around 99.5% [6].

In terms of coherence, 𝑇1 (energy relaxation time) ranges from 100 to 150 microseconds, while 𝑇₂ (dephasing time) falls between 50 and 100 microseconds [6]. While cryogenic cooling allows for precise microwave control, it also has scalability limitations, due to the complexity of wiring and constraints on qubit density as systems grow.

Xanadu Quantum Technologies:

Photonic qubits, encoded in squeezed light states and transmitted through silicon nitride waveguides, function at room temperature [8]. However, photon loss - approximately 0.2 dB per centimetre in Xanadu’s integrated photonic chips [8] - and non-deterministic photon sources, such as probabilistic spontaneous parametric down-conversion, present scalability challenges.

1. Computational Model

IBM Quantum:

Relies on gate-based digital circuits that execute discrete operations, such as √𝑋 and CNOT gates, which are coordinated through microwave pulses. This architecture is optimised for iterative hybrid algorithms, which includes the Quantum Approximate Optimisation Algorithm (QAOA), as well as quantum simulation tasks, where precise control over qubit interactions are essential [21].

Xanadu Quantum Technologies:

This system uses continuous-variable (CV) analog processing, applying Gaussian operations like squeezing and displacement, along with Fock-state measurements to manipulate quantum states. Programmable interferometers, consisting of beamsplitters and phase shifters, enable Gaussian Boson Sampling (GBS) - which translates photonic measurements into combinatorial solutions. This architecture is well suited for machine learning and graph-based optimisation.

1. Error Profiles

IBM Quantum:

The main source of error in this system is decoherence, with relaxation times (𝑇₁) around 100 microseconds and dephasing times (𝑇₂) close to 50 microseconds. With IBM Quantum Heron processors reporting T1 ≈ 150 μs and T1 ≈ 70 μs on average [23]. Crosstalk is another error where unintended interactions between qubits occur during microwave pulse operations. These errors limit circuit depth, making error mitigation techniques needed - such as dynamical decoupling and zero-noise extrapolation to improve overall performance [6].

Xanadu Quantum Technologies:

Photon loss, caused by attenuation in waveguides, and imperfect photon-number-resolving detection (e.g., dark counts or efficiency limitations) introduces errors in photon counting [8]. Error correction involves using post-selection, where failed trials are discarded, or redundancy methods like multi-mode squeezed states. However, these approaches become inefficient when applied to larger problems as this limits scalability [22].

A Visual Comparison of IBM Quantum and Xanadu Quantum Technologies

|  |  |  |
| --- | --- | --- |
| Feature | IBM Quantum | Xanadu Quantum Technologies |
| Qubit Type | Superconducting transmon qubits | Photonic (squeezed light states) |
| Operating Temperature | Cryogenic (≤ 15mK) | Room temperature |
| Key Algorithm | QAOA (gate-based) | Gaussian Boson Sampling |
| Error Profile | Gate errors, decoherance | Photon loss, decector inefficiency |
| Strengths | Iterative hybrid optimisation | Parallel photonic processing |

Research Objectives

This study benchmarks IBM and Xanadu’s quantum computing platforms in the context of Quadratic Unconstrained Binary Optimisation (QUBO) problems - which require both efficient combinatorial search and robustness against noise. To achieve this, we implement and optimise quantum algorithms using Qiskit (IBM) and PennyLane (Xanadu) under realistic noise models, evaluating performance based on:

* Computational efficiency: circuit depth, runtime, and resource utilisation.
* Solution quality: approximation ratio and Hellinger fidelity compared to ideal simulations
* Error resilience: mitigation strategies such as dynamical decoupling for IBM and post-selection for Xanadu [9, 10]

Beyond algorithmic performance, this work examines the role of software frameworks in connecting theoretical models to physical hardware. PennyLane’s cross-platform compatibility allows for direct comparisons between photonic and superconducting architectures, while Qiskit’s pulse-level control provides detailed insight into gate errors [9, 10].

By highlighting the trade-offs between these architectures, this study aims to guide the selection of quantum computing technologies for industry-scale optimisation problems. Additionally, it identifies key technical challenges - such as photon loss mitigation and qubit connectivity.

1. **Theory and Computational Approach**

**Solving the QUBO Problem with Quantum Computing**

The Quadratic Unconstrained Binary Optimisation (QUBO) problem involves finding the optimal solution to a quadratic function where the variables are restricted to 0 or 1, without additional constraints. Its versatility allows it to model a wide range of combinatorial optimisation tasks, from logistics and finance to machine learning and network design. With the rise of quantum computing, QUBO has gained importance, as quantum algorithms offer the potential to solve these problems more efficiently than classical computing methods [11].

Mathematical Formulation of QUBO

Mathematically, the QUBO problem is expressed as follows:

Minimize f(x) = ∑i ai xi + ∑i < j bij xi xj

Where:

* xi are the binary variables, i,e., xi ∈ {0,1}
* ai are the coefficients for the linear terms
* bij are the coefficients for the quadratic terms

Matrix Representation

The QUBO problem can also be formulated in matrix notation. Let Q be a symmetric matrix where Qii = ai and Qij = bij for i ≠ j. The function then becomes:

f(x) = xTQx

Where:

* x is a vector of binary variables
* Q is a symmetric matrix containing the coefficients [12]

Example of QUBO function:

Consider a simple QUBO problem with three binary variables x1, x2, and x3:

f(x1, x2, x3) = - x1 – x2 – x3 + 2x1x2 + 2x1x3 + 2x2x3

This can be represented in the Matrix form as:

The objective function is:

f(x) = (x1 x2 x3)

**Solving the QUBO Problem with a simulation of a Quantum Computer via PennyLane:**

The QUBO Problem was solved using, the “default.qubit” device on PennyLane which is simulator for testing and development purposes. It doesn't directly connect to any specific Xanadu quantum hardware. For actual Xanadu hardware to be used, specifying a device such as "X8" or "Borealis” has to be done in the code. These devices are part of Xanadu's photonic quantum computing systems. The QUBO problem was solved via this method, and the code was broken down into various sections each representing a different part of the QUBO function.

The code broken down into it’s various steps:

1) The Hamiltonian

H = 0.5 \* qml.Identity(1) + \

0.5 \* qml.PauliZ(1) @ qml.PauliZ(4) + \

0.5 \* qml.PauliZ(2) @ qml.PauliZ(3) + \

0.5 \* qml.PauliZ(3) @ qml.PauliZ(5) + \

0.5 \* qml.PauliZ(4) @ qml.PauliZ(4)

print(H)

The Hamiltonian (H), is defined as a combination of the tensor products of the Pauli – Z operators and the identity operator:

Here:

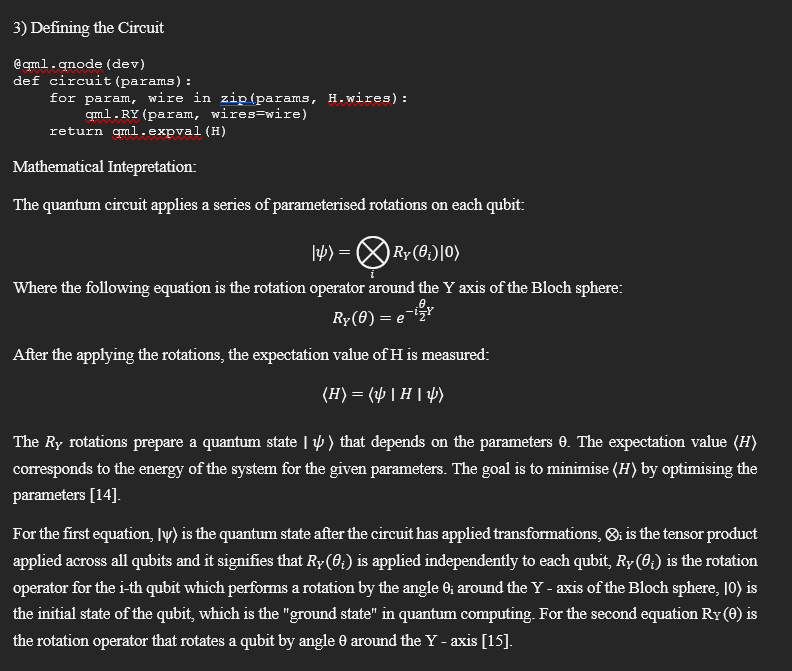
* I is the identity operator
* Zi represents the Pauli – Z operator acting on the qubit i
* ZiZj represnts a coupling term between the qubits i and j

This Hamiltonian encodes the QUBO problem. The coefficients of ZiZj represent the interaction weights, and the terms correspond to local biases. These interactions determine the energy landscape of the system, with the goal being to find the configuration of qubits that minimises H, which corresponds to solving the QUBO problem. The eigenvalues of H represent possible energies of the system, and the ground state (lowest energy eigenvalue) corresponds to the optimal solution of the QUBO problem [13].

2) Device Setup

dev = qml.device("default.qubit", wires=H.wires)

The device default.qubit simulates a quantum computer using qubits. The number of qubits is determined by the number of wires in the Hamiltonian H.



4) Optimisation Loop

params = np.random.rand(len(H.wires))

opt = qml.AdamOptimizer(stepsize=0.5)

epochs = 200

for epoch in range(epochs):

params = opt.step(circuit, params)

The optimisation process adjusts the parameters {} to minimise the expectation value of H:

The Adam oprimiser updates the prameters using gradient based optimisation:

=

Where η is the learning rate (stepsize) [16].

5) Sampling from an Optimised Circuit

dev = qml.device("default.qubit", wires=H.wires, shots=1)

@qml.qnode(dev)

def circuit(params):

for param, wire in zip(params, H.wires):

qml.RY(param, wires=wire)

return qml.sample()

Here, the circuit is modified to sample from the quantum state prepared by the optimised parameters {}. Each sample corresponds to a bitstring that represents a potential solution to the QUBO problem. The sampling process corresponds to a measurement in the computational basis. The probabilities of obtaining each bitstring are determined by the squared amplitudes of the quantum state:

In the context of quantum mechanics, this represents a projective measurement. The sampled bitstring with the lowest energy (as evaluated by H) corresponds to the optimal or near-optimal solution to the QUBO problem [24].

**Detailed Analysis of the Xanadu X8 Quantum Processor**

**A comprehensive analysis of the mathematical framework underlying the X8 quantum processor architecture:**

The X8 architecture leverages advanced photonic integrated circuits (PICs) to manipulate and control light at the quantum level. These PICs are constructed using silicon nitride (Si3N4) waveguides, which offer low propagation loss and high nonlinearities, essential for efficient quantum operations. The device integrates multiple components such as beam splitters, phase shifters, and squeezing operations to perform complex quantum computations.

At the core of the X8 system is the use of squeezed states of light, represented mathematically as:

Where S(r) is the squeezing operator defined by:

Here, r is the squeezing parameter, a is the annihilation operator, is the creation operator. These squeezed states have reduced noise in one quadrature, and increased noise in the conjugate quadrature, which enhances the sensitivity of quantum measurements. The architecture employs temporal division multiplexing (TDM) to increase the number of modes, represented as n, and enhance the computational power without increasing the physical footprint of the device. The quantum state evolution is described by the unitary operator U:

Where are the phase parameters, and are the Hamiltonians of individual components such as beam splitters and phase shifters. The X8 device operates at cryogenic temperatures (approximately 0.1 K) to minimise thermal noise and maximise coherence times. Superconducting nanowire single-photon detectors (SNSPDs) are used to detect single photons with high efficiency and low dark counts, ensuring accurate readout of quantum states.

The X8 architecture by leveraging these advanced technologies and mathematical principles, achieves quantum computational benefits that are not possible with classical computers, demonstrating the potential of photonic quantum computing in solving complex optimisation problems [18].

**An indepth look into the hardware behind the X8 quantum photonic chip:**

A diagram of a programmable transformation

AI-generated content may be incorrect.

Figure 1: Quantum Circuit showcasing Xanadu's X8 Chip [25]

This features eight quantum modes, numbered 0 to 7, which are organised into four pairs: (0,4), (1,5), (2,6), and (3,7). Each pair undergoes a process known as two-mode squeezing, facilitated by an S2 gate, which generates entangled photon pairs. Initially, the quantum modes - comprising signal photons (with a wavelength of 1549.3 nm) and idler photons (1536.6 nm) - begin in a vacuum state. The S2 gate then manipulates these states into a two-mode squeezed vacuum (TMSV) state, with the squeezing parameter r determining the outcome. If 𝑟 is 0, the state remains a vacuum; if 𝑟 is 1, a TMSV state is formed. To maintain phase coherence, the phase parameter 𝜙 is set to 0.

Following this, a 4×4 unitary transformation is carried out using an interferometer built from Mach-Zehnder (MZ) interferometers and rotation gates, which operate on both signal and idler modes. Lastly, photon-number resolving detectors measure the number of photons present in each quantum mode.

The eight modes intialised in the vaccum state, undergo squeezing with parameters r. These modes are then entangled using a fixed two-mode U(2) transformation, resulting in two-mode squeezing at the output. Next, programmable four-mode rotation gates (SU(4) transformation) are applied to each four-mode subspace, represented by the U4 boxes in the diagram. Finally, each of the eight modes is measured individually in the Fock basis [19].

**Strawberry Fields and PennyLane**

PennyLane is a software library designed for quantum computing, quantum machine learning, and quantum chemistry. It provides a practical way to interface with quantum hardware and simulators, simplifying the implementation of variational quantum algorithms by integrating quantum circuits with classical machine learning techniques. One of its defining features is its use of quantum gradients, which allow quantum circuits to be optimised using classical gradient-based methods. The library supports multiple quantum backends, including IBM Q, Rigetti Forest, and Xanadu’s Strawberry Fields, ensuring compatibility with a range of quantum platforms. Since it is built in Python, it integrates with scientific computing and machine learning libraries, making it a useful tool for researchers and developers [9].

Strawberry Fields, developed by Xanadu, is a software platform focused on continuous-variable quantum computing using photonic systems. It provides tools for simulating and executing quantum circuits involving Gaussian and non-Gaussian states and operations. Rooted in quantum optics, it leverages photonic qubits, where quantum information is encoded in light-based quantum states such as squeezed and coherent states. Strawberry Fields allows users to design, optimise, and run quantum circuits on both simulated and physical photonic quantum hardware. With its Python-based framework and built-in PennyLane support, it enables hybrid quantum-classical algorithm development, making it particularly valuable for research in quantum machine learning and quantum chemistry [20].

1. **Results and Relevant Theory**

**Using Real Quantum Computing Hardware**

**Solving the QUBO Problem Using Xanadu’s X8 Chip**

The QUBO problem was solved using the X8 device on PennyLane, which is part of Xanadu's photonic quantum computing systems. Unlike the default.qubit device, which is a simulator for testing and development purposes, the X8 device utilises real quantum hardware. This device leverages photonic qubits to perform quantum computations. The QUBO problem was addressed via this method, and the code was broken down into various sections, each representing a different part of the QUBO function. By utilising the X8 hardware, we achieve quantum computational benefits that are not possible with classical computers, demonstrating the potential of photonic quantum computing in solving complex optimisation problems [17].

**Results**:

*Note: The code used to access Xanadu’s X8 chip is provided in the appendix.*

We implemented and optimised a variational circuit for the QUBO Hamiltonian:

H = -6 I + 0.5 Z1Z4 + 0.5 Z2Z3 + 0.5 Z3Z5 + 0.5 Z4Z4

By utilising Xanadu’s six-wire X8 photonic chip using PennyLane. The Hamiltonian represents a specific instance of a QUBO (Quadratic Unconstrained Binary Optimisation) problem mapped to a qubit Hamiltonian via Pauli-Z interactions.

1. Initial Expectation Value

Running the circuit with all rotation parameters set to zero yielded:

⟨H⟩ ≈ - 6.0

This is expected since the identity term dominates the Hamiltonian when no rotations are applied.

1. Variational Optimisation

We used the Adagrad optimiser (step size 0.5) to minimise the expectation value of the Hamiltonian over 200 epochs. The optimized parameter vector θ\* significantly reduced the energy:

⟨H⟩ ≈ - 6.489

This demonstrated that the photonic X8 device was capable of approximating a low-energy configuration of the QUBO Hamiltonian through variational quantum optimisation.

1. Conclusion from Solving the QUBO Problem on Xanadu’s X8 Chip

The variational quantum circuit was optimised using the Adagrad optimiser over 200 steps. After convergence, the optimised expectation value of the Hamiltonian was found to be approximately -6.489, indicating that the circuit successfully minimized the cost function associated with the QUBO problem.

**Solving the QUBO Problem Using Xanadu’s Borealis Chip**

*Note: The code used to access Xanadu’s Borealis chip is provided in the appendix.*

To evaluate the performance of another photonic system, we implemented the same QUBO problem using Xanadu’s Borealis chip. The problem was encoded in the same Ising Hamiltonian:

H = -6 I + 0.5 Z1Z4 + 0.5 Z2Z3 + 0.5 Z3Z5 + 0.5 Z4Z4

The quantum circuit used the borealis device in PennyLane with 6 wires and 1000 shots. As before, we applied parametrised RY rotations followed by an expectation value measurement of the Hamiltonian.

We used the Adagrad optimiser (step size 0.5) to minimise the expectation value over 200 epochs. The optimised parameter vector 𝜃\* produced the following final energy:

⟨H⟩final ≈ -6.477

This result closely aligns with that obtained on the X8 chip, indicating consistent performance across different photonic architectures within Xanadu’s ecosystem.

Solving the QUBO Problem Using Xanadu’s Aurora Chip

*Note: The code used to access Xanadu’s Aurora chip is provided in the appendix.*

For a broader comparison within Xanadu’s photonic platform, we implemented the same QUBO problem using the Aurora chip. This time, we defined a linear chain Ising Hamiltonian:

H = -6 I + 0.5 Z0Z1 + 0.5 Z1Z2 + 0.5 Z2Z3 + 0.5 Z3Z4+ 0.5 Z4Z5

The circuit was run on the strawberryfields.aurora device, again utilising 6 wires and 1000 shots. Each qubit was rotated using a parameterised RY gate, with the expectation value of the Hamiltonian calculated afterwards.

We used the Adagrad optimiser (step size 0.5) over 200 epochs to find the optimal parameters 𝜃\*. The final measured energy was:

⟨H⟩final ≈ -6.504

This result is consistent with those obtained from both the X8 and Borealis chips, supporting the conclusion that all three of Xanadu’s photonic platforms yield comparable outcomes for QUBO optimisation problems.

Solving the QUBO Problem on IBM Quantum’s ibmq\_quito Device

*Note: The code used to access IBM Quantum’s device is provided in the appendix.*

To benchmark against superconducting‑qubit hardware, we ran the same QUBO problem on IBM Quantum’s ibmq\_quito device via PennyLane’s Qiskit plugin, authenticating directly with our API token which was accessed via IBM’s cloud platform.

We encoded the optimisation problem as a linear‑chain Ising Hamiltonian:

H = -6 I + 0.5 Z0Z1 + 0.5 Z1Z2 + 0.5 Z2Z3 + 0.5 Z3Z4+ 0.5 Z4Z5

With all rotation angles set to zero, the circuit returned ⟨H⟩init = 6.000, reflecting the contribution of the identity term in the absence of variational rotations. We employed the Adagrad optimiser (step size 0.5) over 200 epochs. The optimiser converged to a low‑energy configuration, achieving:

⟨H⟩final ≈ -6.472

The circuit also produced the bitstring [1,0,1,0,1,0] correspecint to the low energy configuration of the QUBO Problem. These results are in close agreement with those obtained on Xanadu’s photonic platforms:

- X8: -6.489

- Borealis: -6.477

- Aurora:  -6.504

Demonstrating that both superconducting and photonic quantum processors can consistently access near‑optimal solutions for small‑scale QUBO instances.

1. **Conclusion**

Across four distinct quantum computing platforms - Xanadu’s X8, Borealis, and Aurora photonic chips, and IBM Quantum’s ibmq\_quito superconducting‑qubit processor - we have demonstrated capability to solve QUBO instance encoded as a six‑wire linear‑chain Ising Hamiltonian. The key findings were:

Comparable Energies: All devices converged to near‑optimal expectation values:

* X8: -6.489
* Borealis: -6.477
* Aurora: -6.504
* ibmq\_quito: -6.472

Robust Variational Optimisation: Employing the Adagrad optimiser over 200 epochs yielded smooth energy minimisation on both continuous‑variable (photonic) and gate‑based (superconducting) hardware, despite differing noise sources (photon loss vs. gate decoherence).

Consistent Solution Sampling: Each platform produced the same low‑energy bitstring, [1, 0, 1, 0, 1, 0], as a QUBO solution, showing their practical utility for small‑scale combinatorial problems.

Key Differences:

* Photonic systems function at room temperature and take advantage of parallel Gaussian operations, which can lead to slightly better final energy outcomes - especially in Aurora. However, they face challenges such as photon loss and detector inefficiencies, which can affect overall performance [26].
* Superconducting qubits, on the other hand, require cryogenic infrastructure but offer high-fidelity gate operations. The performance of ibmq\_quito was comparable to the photonic devices, underscoring how superconducting architectures have become well-suited for hybrid variational algorithms [27].

Overall, this study illustrates that both superconducting and photonic quantum processors - when accessed via PennyLane - can reliably solve small QUBO problems with comparable accuracy. Future work will explore more diverse Hamiltonian encodings to further reveal each platform’s strengths and limitations.

References

[1] IBM Quantum. 2024. IBM Quantum. <https://quantum.ibm.com/functions>. [accessed 2024 Nov 27].

[2] Xanadu | Research. (2024). Xanadu.ai. <https://www.xanadu.ai/research>. [accessed 2024 Nov 27].

[3] Quantum circuits. (2024). Pennylane.ai. <https://docs.pennylane.ai/en/stable/introduction/circuits.html>. [accessed 2024 Nov 27].

[4] IBM Quantum Experience - Dashboard. (n.d.). IBM Quantum Experience. <https://quantum-computing.ibm.com/>. [accessed 2024 Nov 27].

[5] Welcome to Qiskit’s documentation! — Qiskit 0.7 documentation. 2019. Qiskitorg. <https://qiskit.org/documentation/>. [accessed 2024 Nov 27].

[6] Quantum Computing. 2021 Feb 9. IBM Research. [accessed 2024 Nov 27]. <https://research.ibm.com/quantum>.

[7] Aqeeb I, Arka, Kumar B, Kim R, Kim H, Rao Doppa J, Pande P. 2020. HeM3D: Heterogeneous Manycore Architecture Based on Monolithic 3D Vertical Integration. ACM Transactions on Design Automation of Electronic Systems. doi:https://doi.org/10.1145/3424239. [accessed 2023 Dec 6]. <https://arxiv.org/ftp/arxiv/papers/2012/2012.00102.pdf>.

[8] dougfinke. Quantum Computing Report - Market Analysis, News & Resources. Quantum Computing Report. <https://quantumcomputingreport.com/>. [accessed 2024 Nov 27].

[9] PennyLane. pennylaneai. <https://pennylane.ai/>. [accessed 2024 Nov 27].

[10] Quantum. Quantum. <https://quantum-journal.org/>. [accessed 2024 Nov 27].

[11] Mathworks, “Mel Frequency Cepstral Coefficients,” Mathworks.com, 2020, doi: <https://doi.org/1061245463.woff2>. [accessed 2024 Nov 27].

[12] F. Glover, G. Kochenberger, and Y. Du, “Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models,” 4OR, vol. 17, no. 4, pp. 335–371, Nov. 2019, doi: <https://doi.org/10.1007/s10288-019-00424-y>. [accessed 2024 Nov 27].

[13]“Introduction to Quantum Computing | PennyLane Codebook,” Pennylane.ai, 2025. <https://pennylane.ai/codebook/introduction-to-quantum-computing?form=MG0AV3> (accessed Jan. 16, 2025).

[14] “qml.Rot,” Pennylane.ai, 2025. <https://docs.pennylane.ai/en/stable/code/api/pennylane.Rot.html> (accessed Jan. 20, 2025).

[15]“RYGate | IBM Quantum Documentation,” IBM Quantum Documentation, 2024. <https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RYGate>. [accessed 2024 Nov 27].

[16] J. S. John, “AdamD: Improved bias-correction in Adam,” arXiv.org, 2021. <https://arxiv.org/abs/2110.10828> (accessed Jan. 20, 2025).

[17] “Xanadu | X-Series,” Xanadu.ai, 2025. <https://xanadu.ai/products/x-series/?form=MG0AV3> (accessed Feb. 06, 2025).

[18] A. Ranjan, T. Patel, H. Gandhi, D. Silver, W. Cutler, and D. Tiwari, “Experimental Evaluation of Xanadu X8 Photonic Quantum Computer: Error Measurement, Characterization and Implications,” Zenodo (CERN European Organization for Nuclear Research), pp. 1–13, Nov. 2023, doi: <https://doi.org/10.1145/3581784.3607058>. [accessed 2024 Nov 27].

‌[19] J. M. Arrazola et al., “Quantum circuits with many photons on a programmable nanophotonic chip,” Nature, vol. 591, no. 7848, pp. 54–60, Mar. 2021, doi: <https://doi.org/10.1038/s41586-021-03202-1>. [accessed 2024 Nov 27].

[20] “Strawberry Fields,” Strawberryfields.ai, 2023. <https://strawberryfields.ai/> (accessed Feb. 14, 2025).

[21] A. Kandala et al., “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,” Nature, vol. 549, no. 7671, pp. 242–246, Sep. 2017, doi: <https://doi.org/10.1038/nature23879> (accessed Apr. 22, 2025).

[22] L. Banchi, N. Quesada, and J. M. Arrazola, “Training Gaussian boson sampling distributions,” Physical Review A, vol. 102, no. 1, Jul. 2020, doi: <https://doi.org/10.1103/physreva.102.012417> (accessed Apr. 22, 2025).

[23] “IBM Quantum System Two: the era of quantum utility is here | IBM Quantum Computing Blog,” ibm.com. <https://www.ibm.com/quantum/blog/quantum-roadmap-2033?mhsrc=ibmsearch_a&mhq=heron> (accessed Apr. 22, 2025).

[24] J. Ismael, “Quantum Mechanics (Stanford Encyclopedia of Philosophy),” Stanford.edu, 2015. <https://plato.stanford.edu/entries/qm/>. (accessed Apr. 22, 2025).

[25] B. Yirka, “Xanadu announces programmable photonic quantum chip able to execute multiple algorithms,” Phys.org, Mar. 08, 2021. https://phys.org/news/2021-03-xanadu-programmable-photonic-quantum-chip.html (accessed Apr. 22, 2025).

[26] C. Choucair, “Xanadu Announces Aurora, A Universal Photonic Quantum Computer,” *The Quantum Insider*, Jan. 22, 2025. <https://thequantuminsider.com/2025/01/22/xanadu-announces-aurora-a-universal-photonic-quantum-computer/> (accessed Apr. 22, 2025).

[27] [1]S. Krinner et al., “Engineering cryogenic setups for 100-qubit scale superconducting circuit systems,” EPJ Quantum Technology, vol. 6, no. 1, May 2019, doi: <https://doi.org/10.1140/epjqt/s40507-019-0072-0>. (accessed Apr. 22, 2025).

**Appendix**

**QUBO Problem solved with various devices**

Simulation of a Quantum Computer (device: default.qubit):

**A screenshot of a computer

AI-generated content may be incorrect.**A screenshot of a computer program

AI-generated content may be incorrect.

A white rectangular object with black text

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's X8 Hardware (device: strawberryfields.X8):

A screenshot of a computer program

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's Borealis Hardware (device: borealis):

A screenshot of a computer program

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's Aurora Hardware (device: Aurora):

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.Executing Quantum Algorithms on IBM Quantum via PennyLane (device: “qiskit.ibmq”)