OpenQUBO: A Unified Platform for QUBO Analysis and Hybrid Quantum–Classical Optimization

Micah Arthur Benjamin Shaw
SemperKite Quantum Labs
Monterey Park, CA, USA
Email: micahshawdev@gmail.com

***Abstract*—OpenQUBO is a unified software platform designed for the formulation, validation, and deployment of Quadratic Unconstrained Binary Optimization (QUBO) problems across quantum and classical computing backends. The system integrates gate-based, annealing-based, and classical solvers within a single workflow, enabling automated embedding, backend-aware analysis, and reproducible benchmarking pipelines.**

OpenQUBO consolidates structural diagnostics, execution strategy selection, and code generation into a single environment, supporting scalable hybrid optimization experiments on both real hardware and simulators. The platform emphasizes transparency, interoperability, and practical usability for researchers and developers.

***Index Terms*—QUBO, Quantum Optimization, Quantum Annealing, Gate-Based Computing, Hybrid Algorithms**

I. Introduction

OpenQUBO is the first software application developed under SemperKite Quantum Labs. It provides a unified environment for quantum researchers and laboratories to centralize QUBO problem development and enable seamless export to real quantum hardware.

Quadratic Unconstrained Binary Optimization (QUBO) problems commonly arise in combinatorial optimization and are typically represented by symmetric or upper-triangular matrices that encode weighted interactions among binary decision variables [1], [2]. The objective is to identify variable assignments that optimize the associated quadratic form.

Mapping QUBO problems to Ising Hamiltonians enables their direct implementation on quantum hardware and remains a foundational component of many quantum optimization frameworks [3], [4].

II. Background and Related Work

QUBO solving paradigms are commonly divided into gate-based quantum computing and quantum annealing [5], [6]. Each paradigm imposes constraints on connectivity, circuit depth, embedding complexity, and execution cost.

Due to hardware limitations, careful representation and execution planning are required. Suboptimal layouts can lead to infeasible embeddings and inefficient hardware utilization [7], [9]. Automated, backend-aware optimization is therefore essential for practical deployment.

Existing software frameworks such as Qiskit Optimization and PyQUBO provide foundational tooling for QUBO modeling and execution [11], [12]. OpenQUBO extends these efforts through unified workflow management and backend-aware diagnostics.

III. System Overview

OpenQUBO addresses the challenges of executing QUBO problems across heterogeneous computational backends. Given a QUBO matrix, the system determines suitable mappings for gate-based or annealing-based execution and provides layout recommendations alongside a QUBO quality score.

Users may inspect graph representations of problem structure, couplings, and overlays. OpenQUBO additionally generates execution-ready code for IBM Quantum platforms and Gurobi-based classical solvers. Built-in examples, including Triangle and Max-Cut QUBOs, support benchmarking and experimentation.

A. Architecture

The OpenQUBO workflow consists of:

1) QUBO ingestion and preprocessing
2) Input validation and formulation verification
3) Structural analysis and metric extraction
4) Execution strategy assessment (gate, anneal, or hybrid)
5) Backend-specific compilation, embedding, and export

The annealing path emphasizes topology-aware embedding and chain analysis, while the gate-based path focuses on circuit depth, routing efficiency, and qubit utilization.

For annealing-based execution, OpenQUBO utilizes the *minorminer* embedding method to convert the QUBO interaction graph into a hardware-compatible topology [8], [3]. Embedding feasibility, chain lengths, and coupling strengths are analyzed prior to submission.

B. Software Availability

OpenQUBO is distributed through SemperKite Quantum Labs and is available to researchers under an academic and research license.

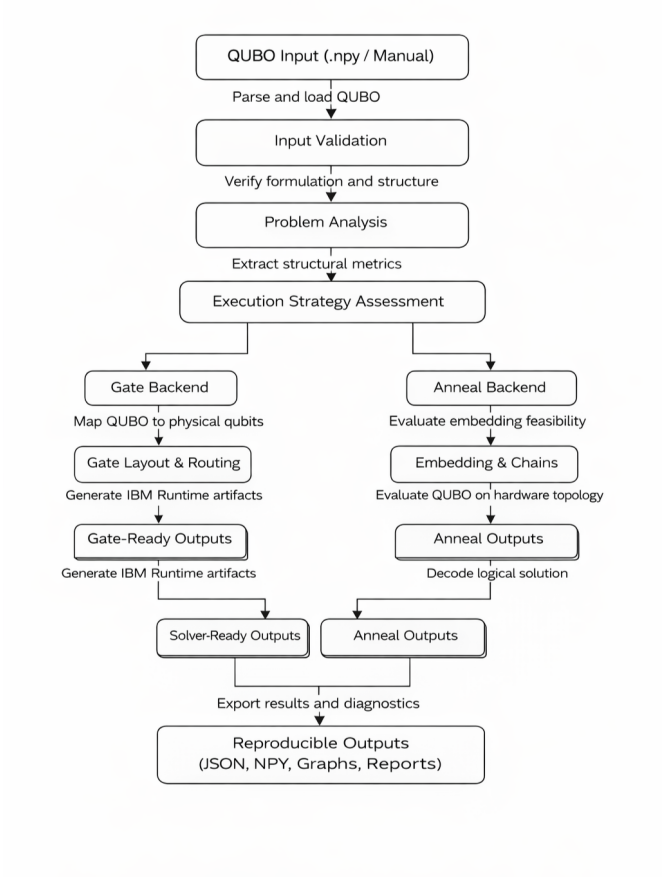

Fig. 1: OpenQUBO system workflow and execution strategy selection pipeline.

IV. User Interface

A. Home Screen

Fig. 2: OpenQUBO Home Screen Interface.

B. Summary Tab

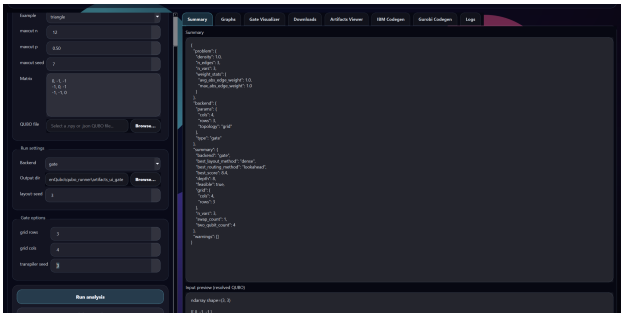

Fig. 3: Summary Tab Output.

C. Graphs Tab

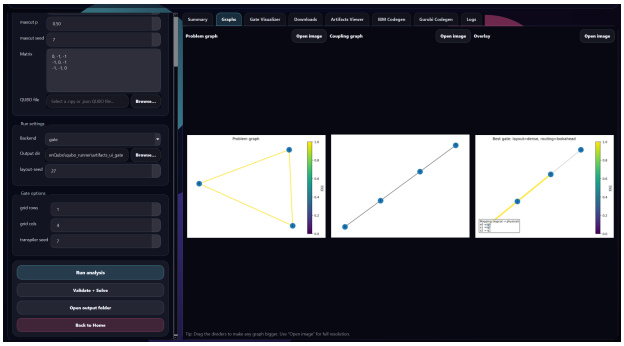

Fig. 4: Graphs tab showing coupling and overlay structures.

D. Gate Backend Visualizer

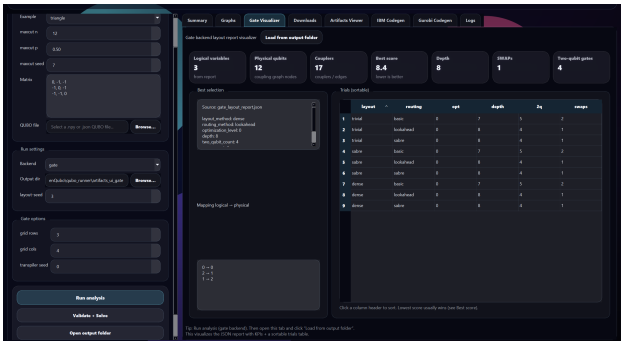

Fig. 5: Gate backend visualizer showing routing and scoring diagnostics.

E. Annealer Backend Graphs

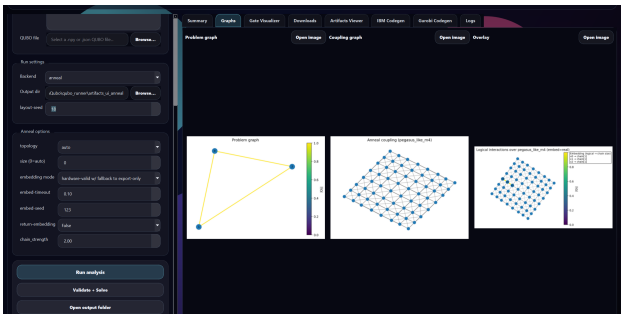

Fig. 6: Annealer backend graphs showing problem structure and embedding.

V. Validation and Solving

The validation workflow verifies matrix dimensionality, symmetry, index consistency, and coefficient finiteness. The solve stage applies scaling and executes backend-specific optimization [10].

Validation reports are exported in structured JSON format.

VI. Code Generation

OpenQUBO produces execution-ready scripts for IBM Quantum QAOA pipelines and Gurobi classical solvers [5], [6].

VII. Limitations

Current backends are heuristic and probabilistic. Dense QUBO embeddings remain challenging for annealing systems [7], [9].

VIII. Future Work

Future work will emphasize hybrid integration, scalability, and expanded hardware coverage [13], [14].

IX. Conclusion

This work presents OpenQUBO as a unified platform for QUBO formulation, validation, and deployment across quantum and classical systems. The platform enables reproducible and scalable hybrid optimization workflows.

References

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

[2] F. Glover, G. Kochenberger, and Y. Du, "A Tutorial on Formulating and Using QUBO Models," arXiv:1811.11538, 2018.

[3] V. Choi, "Minor-Embedding in Adiabatic Quantum Computation," Quantum Inf. Proc., 2008.

[4] T. Albash and D. A. Lidar, "Adiabatic Quantum Computation and Quantum Annealing," Rev. Mod. Phys., 2018.

[5] E. Farhi et al., "A Quantum Approximate Optimization Algorithm," arXiv:1411.4028, 2014.

[6] L. Zhou et al., "QAOA Performance and Mechanism," arXiv:1812.01041, 2018.

[7] A. D. King et al., "Scaling advantage over path-integral Monte Carlo," Nature Physics, 2015.

[8] A. D. King et al., "Pegasus Connectivity," arXiv:1901.07636, 2019.

[9] K. Boothby et al., "Next-Generation Topology of D-Wave Processors," arXiv:2003.00133, 2020.

[10] A. Mazumder and S. Tayur, "Five Starter Problems for QUBO," arXiv:2401.08989, 2024.

[11] A. Javadi-Abhari et al., "Quantum Computing with Qiskit," arXiv:2405.08810, 2024.

[12] M. Zaman et al., "PyQUBO," arXiv:2103.01708, 2021.

[13] J. Preskill, "Quantum Computing in the NISQ Era," Quantum, 2018.

[14] F. Neukart et al., "Traffic Flow Optimization Using a Quantum Annealer," arXiv:1708.01625, 2017.