Qiskit

Group: Error 404
Github Repo: https://github.com/aneeshsharma/Matsat
Group Members:

1. J Hemanth Kumar - B181004CS
2. Adil Chery - B180372CS
3. Arjun Syam - B180031CS
4. Namburi Soujanya - B180491CS
5. Anish Sharma - B181065CS
6. Dev Sony - B180297CS

---

1. Features & Facilities

A multitude of applications are possible using qiskit. In order to better understand and summarise them, we've broken them up into the following categories:

1. Research Applications
2. Algorithms
3. Toolbox for Experiments
4. Quantum Circuits
5. Additional Extensions
6. Testing Circuits on Real Quantum Systems
7. Simulate Quantum Hardware locally

1.1 Research Applications

Qiskit allows for departments involved with Research and Development to come up with use cases that can benefit highly with quantum advantage. Some applications are as follows:

- **Optimization:** This package provides easy to use optimization algorithms that can run on classical simulators and real quantum hardware. Due to its modular design, it is easily extensible and facilitates rapid development and testing of new algorithms as

well.

- **Finance:** The package contains components to load uncertainty models, random data for finance experiments and coupled with the Optimization package, can help to produce models for optimization problems.

- **Machine Learning:** The package contains numerous sample datasets. With the provided algorithms such as QSVM, VQC and QGAN, the datasets can be used for experimentational purposes.

- **Chemistry:** This package contains ground state energy computations, excited states and dipole moments of molecules, both open and closed-shell. It has chemistry drivers which helps to produce data regarding molecular configurations.

1.2 Algorithms

In order to facilitate these research applications, Qiskit provides a generic framework of cross-domain quantum algorithms. Some of the provided algorithms include:

- **Grover's algorithm:** It is a quantum search algorithm that provides quadratic speedup for searching through unstructured collections of records in search of particular targets. Quantum search algorithms derive their importance from widespread use of different search algorithms used in classical algorithms. These were then adapted to obtain faster quantum algorithms such as Grover's.

- **Variational Quantum Eigensolver (VQE):** This algorithm helps to provide techniques combining quantum and classical computations which facilitate in finding the minimum eigenvalue of the given system's Hamiltonian.

- **Quantum Approximate Optimization Algorithm (QAOA):** It is an extension of VQE, using its own fine-tuned variational form to solve problems such as maximum-cut.

- **Quantum SVM algorithm (QSVM):** Support Vector Machines are used to classify data into categories using classical algorithms. This algorithm provides techniques to solve classical algorithms which aren't very efficient using their quantum versions.

1.3 Toolbox for Experiments

Qiskit provides frameworks that contain circuits and analysis methods to identify the source of noise impacting our devices. This helps with studying the interaction rate and control-errors in gates. Some of the frameworks include:

- **Characterization:** The framework provides analysis parameters and circuits to users.
- **Verification:** The framework provides experiments to verify gates and small circuit performance through tomography, quantum volume and randomized benchmarking.
- **Calibration:** It allows users to optimize pulse parameters to minimize errors.

1.4 Quantum Circuits

Qiskit helps to provide tools for composing quantum programs to simulate circuits and pulses, optimizing them for constraints of a particular physical quantum processor and to manage batch execution of experiments on remote-access backends.

- **Compiler:** It helps with simulating quantum circuits on real hardware and to optimize such circuits using pass managers.
- **Circuit Library:** It allows higher levels of abstraction of well studied circuits and gates for experimentation purposes.
- **Pulse:** The tool allows lower level programming of quantum circuits interacting with real quantum hardware.

1.5 Additional Extensions

It allows us to use extensions developed by the open source community consisting of startups, researchers and independent developers. A few examples are:

- **Qiskit_rng:** Generates quantum certified random numbers.

- **Kaleidoscope:** Provide interactive visualizations for quantum states, distributions and device properties.

- **Q-CTRL:** It allows the development and deployment of established error-robust quantum control protocols.

1.6 Testing Circuits on Real Quantum Systems

Using the tools provided, we can formulate our own quantum artifacts. These can then be run on actual Quantum Systems. Your job would be put in a queue, and then returns results that can be processed to display them appropriately.

1.7 Simulate Quantum Hardware locally

In order to run simulations, Qiskit provides high performance simulator frameworks. Optimized C++ simulator backends are provided for executing compiled circuits and tools for constructing highly configurable noise models for performing realistic noisy simulations of the errors that occur during execution on real devices. This gets rid of the waiting queues to work on real quantum systems.

---

2. Drawbacks

There is a difference between Quantum Computations and a Quantum Computer. The first is a mathematical formalism whereby computations are carried out through vectors in the Hilbert space as opposed to its classical counterparts of binary integers. Like classical computations, quantum computations are analogous to the device it is run on as it is just a theoretical mathematical framework. A Quantum Computer is still an unsolved part of quantum computing. We know to formalise the concept using algorithms to define a controlled quantum environment, that is based on our current understanding, in order to build the **'Quantum Computers'** of today.

Due to this, a vast amount of drawbacks at the moment are unknown. Although there are a few that we witnessed over the course of developing the algorithm for the project:

- **Wait Times**: Even now, the amount of Quantum Hardware available is just a handful around the world. Due to this, jobs are not instantaneous, rather queued, and a small change could mean waiting around the queue again. As of today, most algorithms are first tested with the local simulator before being sent as a job in order to counter this hindrance.

- **Lacks commercial/realistic use:** Due to the wait times and limited number of resources, instantaneous results are not attainable at the moment. Hence, commercial usage for replacing everyday applications is still under development until we can manage to have more realistic quantum hardware available (preferably ones that aren't in a controlled environment). The quantum speedup that we hope for achieving is still not possible.

- **Noise:** From comparing the results between the local simulator and real quantum hardware, we can easily notice the amount of noise; It is quite a lot. This leads to error in calculations and lack of accuracy about Quantum World we hope to understand.

- **Limited to Learning/Research:** As of now, Qiskit accomplishes everything a normal computer can do locally. Its main application is thus limited to understanding the Quantum world and its physics rather than actually obtaining the speedup to achieve groundbreaking results.

- **It is a tool, not a translator:** Qiskit merely provides us with the tools to understand and explore Quantum Mechanics, not an automated process to convert theory based quantum circuits into implementable ones. In theory, any unitary operator is a valid quantum gate, but in practice, we only have a finite set of so-called physical gates, which are actual manipulations of real qubits. We are still used to the classical world, which limits our progress towards understanding the workings at a quantum level.

---

3. Improvements and Enhancement

Quantum computers are exceedingly difficult to engineer, build and program. As a result, they are cripppled by errors in the form of noise, faults and loss of quantum coherence, which is crucial to their operation and yet falls apart before any nontrivial program has a chance to run to completion.

This loss of coherence (called decoherence), caused by vibrations, temperature fluctuations, electromagnetic waves and other interactions with the outside environment, ultimately destroys the exotic quantum properties of the computer. Given the current pervasiveness of decoherence and other errors, contemporary quantum computers are unlikely to return correct answers for programs of even modest execution time.

To run algorithms efficiently we reduce the number of gates in an attempt to finish execution before decoherence and other sources of errors have a chance to unacceptably reduce the likelihood of success. We use machine learning to translate, or compile, a quantum circuit into an optimally short equivalent that is specific to a particular quantum computer. Until recently, we have employed machine-learning methods on classical computers to search for shortened versions of quantum programs. Now, in a recent breakthrough, we have devised an approach that uses currently available quantum computers to compile their own quantum algorithms. That will avoid the massive computational overhead required to simulate quantum dynamics on classical computers.

Another way to increase the efficiency of the algorithm is to remove the states having low probability or impractical solutions out of the dataset.

---

4. Design a new artifact (algorithm/circuit) and run it on your system.

Tic Tac Toe Implementation:
https://github.com/aneeshsharma/Matsat/blob/main/quantum_tictactoe.ipynb