Applying Evolutionary Algorithms to Automatically Design Quantum-Gate Circuits

B. Theron

April, 2024

Introduction

- Classical computers are approaching their theoretical limits.
- Quantum computers appear to be the solution.
- Quantum systems are highly specialised and remain expensive.

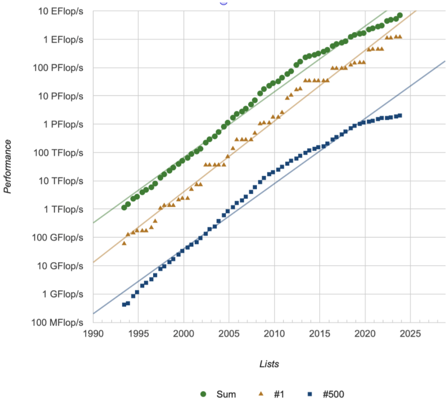


Figure: Top500 List Between 1990-2024 [4].

Background - Quantum Computers

- Quantum computers use *qubits*.
- Quantum systems are modelled in bra-ket notation.
- Quantum circuits are a representation of quantum algorithms.
- Circuit optimisation involves reducing the number/type of gates.

$$|\uparrow\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |\downarrow\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Figure: The Standard Basis in Bra-Ket Notation.

Background - Evolutionary Algorithms

- Evolutionary algorithms (EAs) are a set of stochastic optimisation algorithms.
- EAs have shown prior success.
- Genetic algorithms are a subset of EAs which proved the most successful [1] [2].

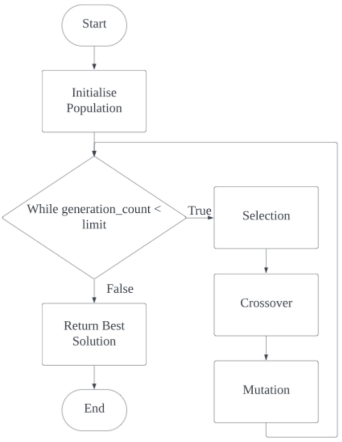


Figure: A Standard Flowchart For an Evolutionary Algorithm.

- The application of the QFT on n qubits is represented by:

$$j \mapsto \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} k \qquad (1)$$

- Transforms a quantum state from the computational basis to the Fourier basis.

- QFT is exponentially faster than the classical variant.

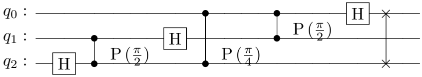


Figure: The 3-Qubit QFT Circuit Evolved in This Project.

Background - Grover's Search Algorithm

- Searches for a quantum state in an unordered database.
- Queries an oracle function and uses Grover's Diffusion Operator.
- Quadratically faster than the classical variant, running in $O(\sqrt{N})$.

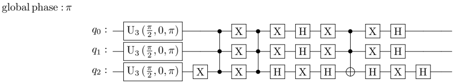


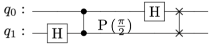
Figure: The 3-Qubit Grover's Search Circuit Evolved in This Project.

Technical Background - Resources Used

- Jupyter Notebook is used for the development environment.
- Qiskit is used for the quantum functionality.
- Matplotlib is used to display results.
- Deap is used for the EA functionality.
- The built in Deap tournament selection operator is used.
- The unittest Python module was used for unit testing.

Technical Background - Quantum Circuit Representation

- A variable length list stores each quantum circuit.
- Mirrors Qiskit's built in representation.
- A dictionary is used to store the gate set.



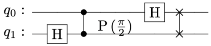
circuit = [[1, [1]], [3, [0, 1], [10, [1], [1, [0]], [2, [1, 0]]

gate_set = {1:HGate(), 2:SwapGate(), 3:CPhaseGate(math.pi/2), 10:"WIRE"}

Figure: The Gate Set and Representation Used to Store the 2-Qubit QFT Circuit.

Technical Background - EA Operators & Fitness Method

- Single gene mutation is used at a rate of 0.8.
- Double point crossover is used at a rate of 0.6.
- A fitness function which finds absolute difference is used.
- Elitist selection is used for 5% of the population.
- Tournament selection is used for the remaining 95%.



circuit = [[1, [1]], [3, [0, 1], [10, [1], [1, [0]], [2, [1, 0]]

gate_set = {1:HGate(), 2:SwapGate(), 3:CPhaseGate(math.pi/2), 10:"WIRE"}

Figure: The Gate Set and Representation Used to Store the 2-Qubit QFT Circuit.

Technical Background - Unit Testing

- Unit testing carried out for each proprietary functions.
- Test structure adheres unittest's standard structure.
- No need integration or interface testing carried out.

Technical Background & Results - Experiments I

- Mutation rate and crossover rate were examined between 0 and 1, in increments of 0.1.
- Elitism rate was examined between 0 and 0.45 in increments of 0.05.
- Population and generation size was examined between 100 and 600 in increments of 100.
- Tournament size was examined between 2 and 10 in increments of 1.

Experiment	Parameter Value	Average Fitness	Average Size	Average Time Taken (s)
Mutation Rate	0.8	36.76	37.8	63.51
Crossover Rate	0.6	28.62	36.9	67.27
Elitism Rate	0.45	38.14	38.2	61.28
Population Size	600	25.51	36.2	328.46
Generation Size	600	29.95	37.3	410.77
Tournament Size	4	39.17	36.8	68.08

Figure: The Results of the Parameter Experiments

Technical Background & Results - Experiments II

- A multi-objective EA which optimises fitness and circuit size was implemented.
- These results were worse than anticipated.
- Indicates a faulty implementation.

Problem	Success Rate	Best Fitness	Interquartile Range	Size Closeness	Average Time Taken(s)
GDO-2	0%	(4, 3)	(0, 3)	11.5	58.56
GDO-3	0%	(17.19, 14)	(0.49, 10)	16.6	146.6
GDO-4	0%	(46.75, 108)	(10.625, 31)	33.7	2727.58
QFT-2	60%	(0, 4)	(2.83, 1)	0.3	145.70
QFT-3	0%	(4.33, 6)	(3.39, 1)	1.1	302.34
QFT-4	0%	(36.61, 9)	(8.45, 1)	2.5	516.94

Figure: The Results of the Multi-Objective EA

Technical Background & Results - Experiments III

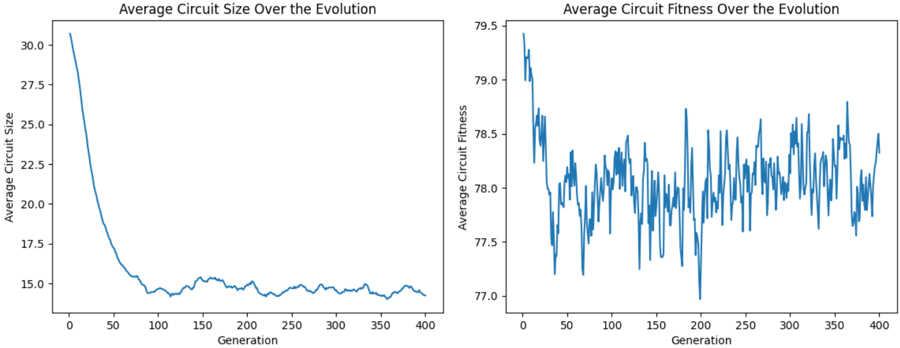


Figure: Graphical Results For a Run of the Multi-Objective EA on the 4-Qubit QFT Circuit

Results - Main EA

- Results of the main EA surpass the benchmark results found in [3].
- The EA scales up exponentially.
- The EA meets its functional and non-functional requirements.
- The EA surpasses its initial aims.

Problem	Success Rate	Best Fitness	Interquartile Range	Size Closeness	Average Time Taken(s)
GDO-2	100%	0	0	0.7	130.86
GDO-3	0%	11.31	2.13	19.7	341.6
GDO-4	0%	15.75	28.69	126.6	6235.323
QFT-2	100%	0	0	2.2	270.49
QFT-3	100%	0	0	5.6	524.96
QFT-4	30%	0	22.63	16.5	1019.31

Figure: The Results of the Main EA

Further Work

- Use a more sophisticated fitness function, i.e. Mean Square Fidelity [3].
- Introduce external noise to reduce gate redundancy.
- Extend the EA to *n*-qubit problems.
- Investigate using sub-systems to approximate larger circuits.
- Further extend the experiments used in the project.

Code Demo

Code Demo.

Conclusion

- This project has successfully created an EA which can automatically design quantum circuits.
- There is still lots of further work to be done.
- Quantum computing holds the promise of profoundly reshaping our world.

Project Links

- The presentation video can be found at:
- The source code files can be found at:
- The GitHub hosting the project files can be found at: `https://github.com/BenjaminTheron/ECM3401_Dissertation`

References

[1] J. F. Miller, P. Thomson, T. Fogarty, *et al.*, "Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study," *Genetic algorithms and evolution strategies in engineering and computer science*, pp. 105–131, 1997.

[2] T. Yabuki and H. Iba, "Genetic algorithms for quantum circuit design –evolving a simpler teleportation circuit–,", 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:16687378.

[3] T. Atkinson, A. Karsa, J. Drake, and J. Swan, "Quantum program synthesis: Swarm algorithms and benchmarks.," in *Genetic Programming*, S. L., H. T., L. N., R. H., and G.-S. P., Eds., vol. 11451, 2019, pp. 19–34, ISBN: 978-3-030-16670-0. DOI: 10.1007/978-3-030-16670-0_2.

[4] Top500, "Projected performance development,", 2024. [Online]. Available: https://top500.org/statistics/perfdevel/.