Genetic Algorithms for Quantum Circuits

By Arturo, Badr, Cesar,
Lukas and Tim

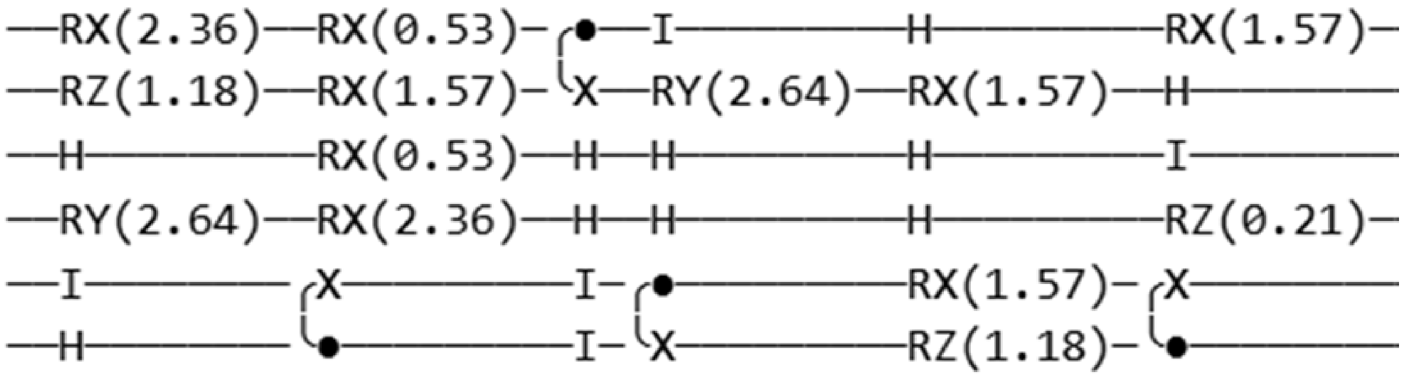
Outline

- Introduction of the problem
  - Physical problem
  - Quantum circuits and VQE
  - Genetic algorithms
- Code implementation
  - Tensor and circuit generations
  - Tournament and selection
  - Crossover
  - Mutations
  - Hyperparameter Optimisation
- Results
  - Energies
  - Phase transition with Z magnetisation
- Conclusions

Introduction
Physics Problem At Hand

- Transverse-Field Ising model
- Maps computational problems

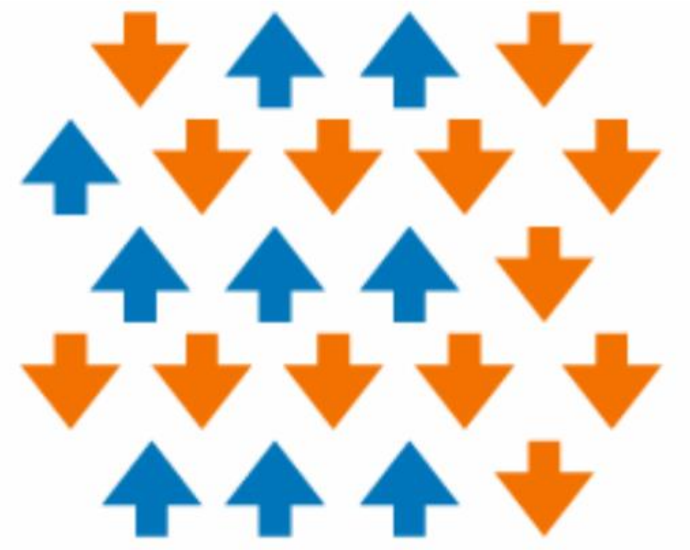
$$H = -J \sum_{(i,j)\in E} Z_i Z_j + h \sum_i X_i$$



Introduction
Physics Problem At Hand

https://sketchplanations.com/the-travelling-salesman-problem

Solution found by minimizing loss function



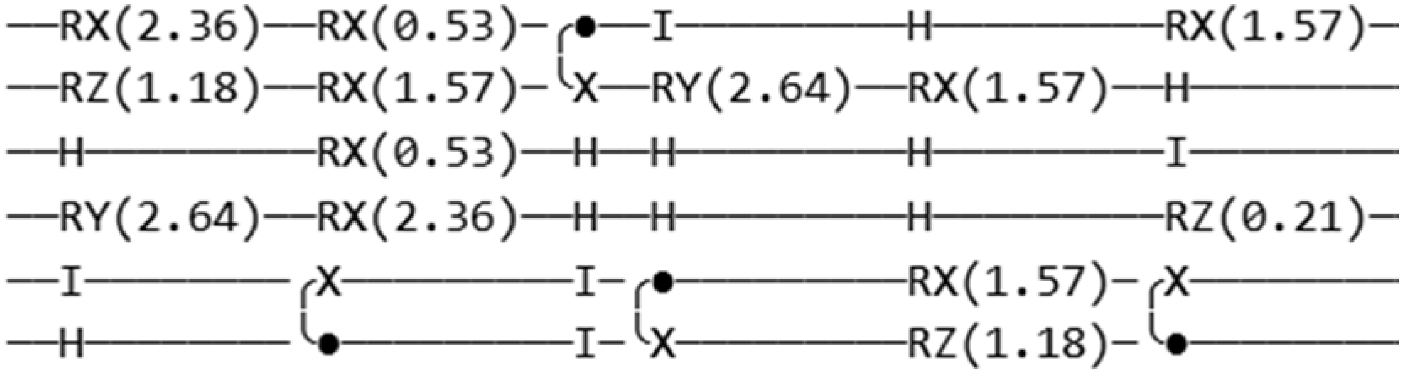
E. Greplova: AP3751 Lecture 1

Solution found by calculating lowest eigenvalue

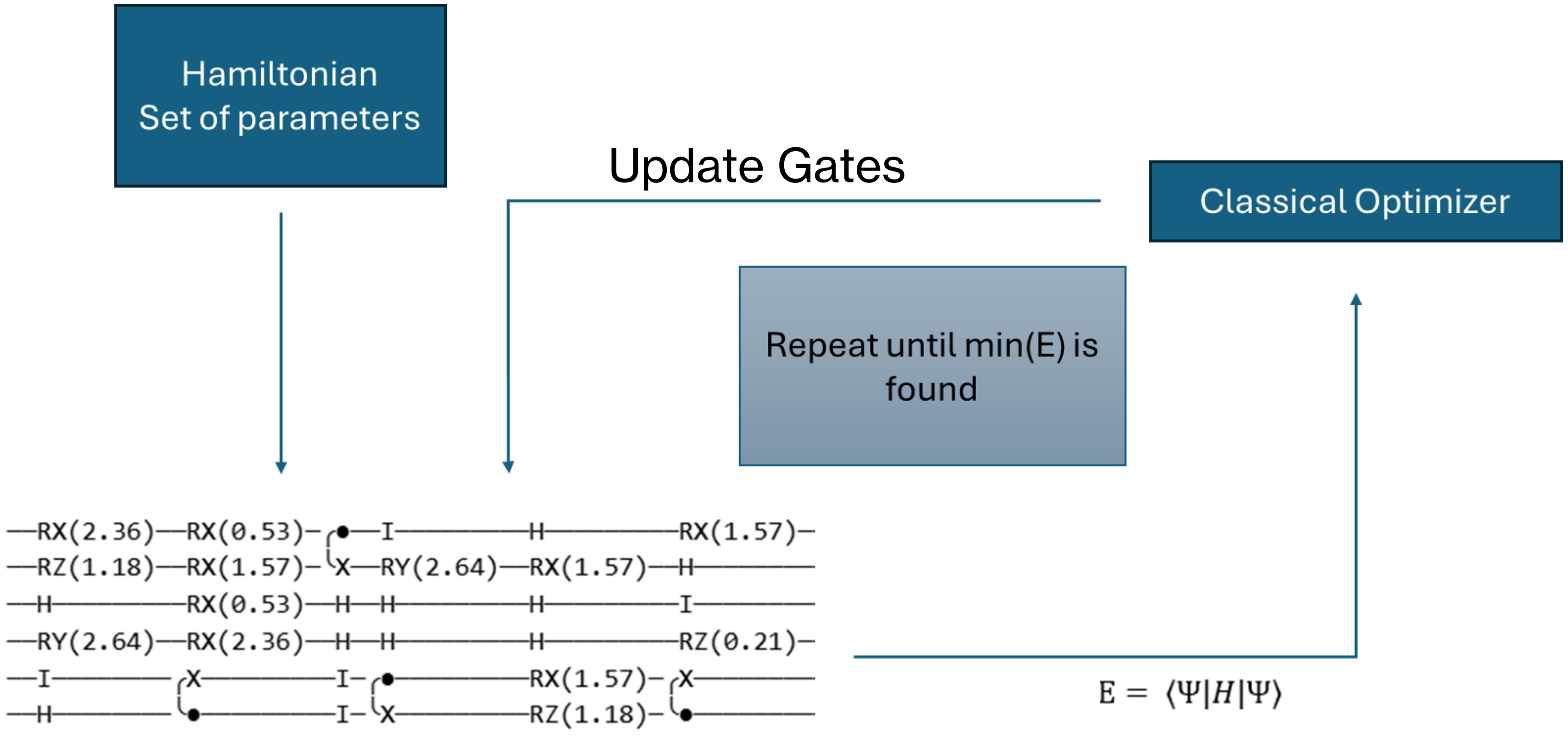
Introduction
Physics Problem At Hand

- Transverse-Field Ising model
- Maps computational problems
- Solution: Finding lowest eigenvalue
- Exponential scaling with system size

$$H = -J \sum_{(i,j)\in E} Z_i Z_j + h \sum_i X_i$$



Introduction
Existing Solutions: VQE



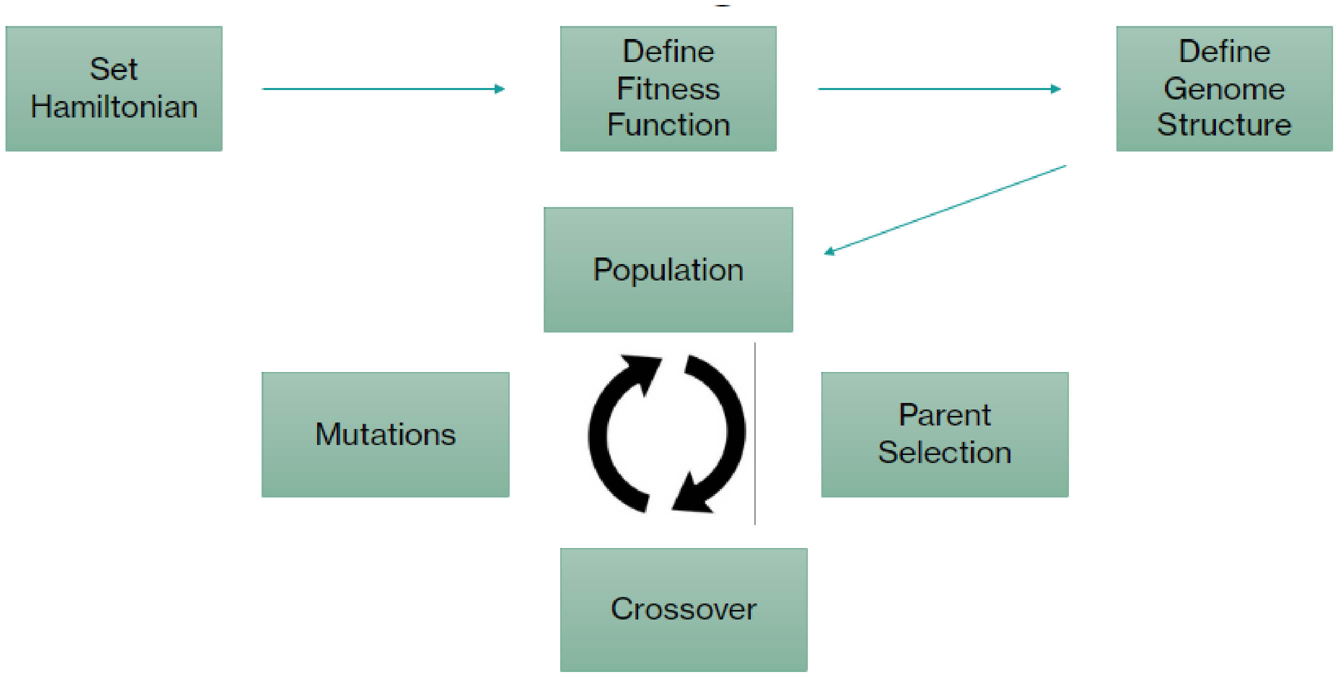
Need for automated circuit optimization

Introduction
Genetic Algorithms

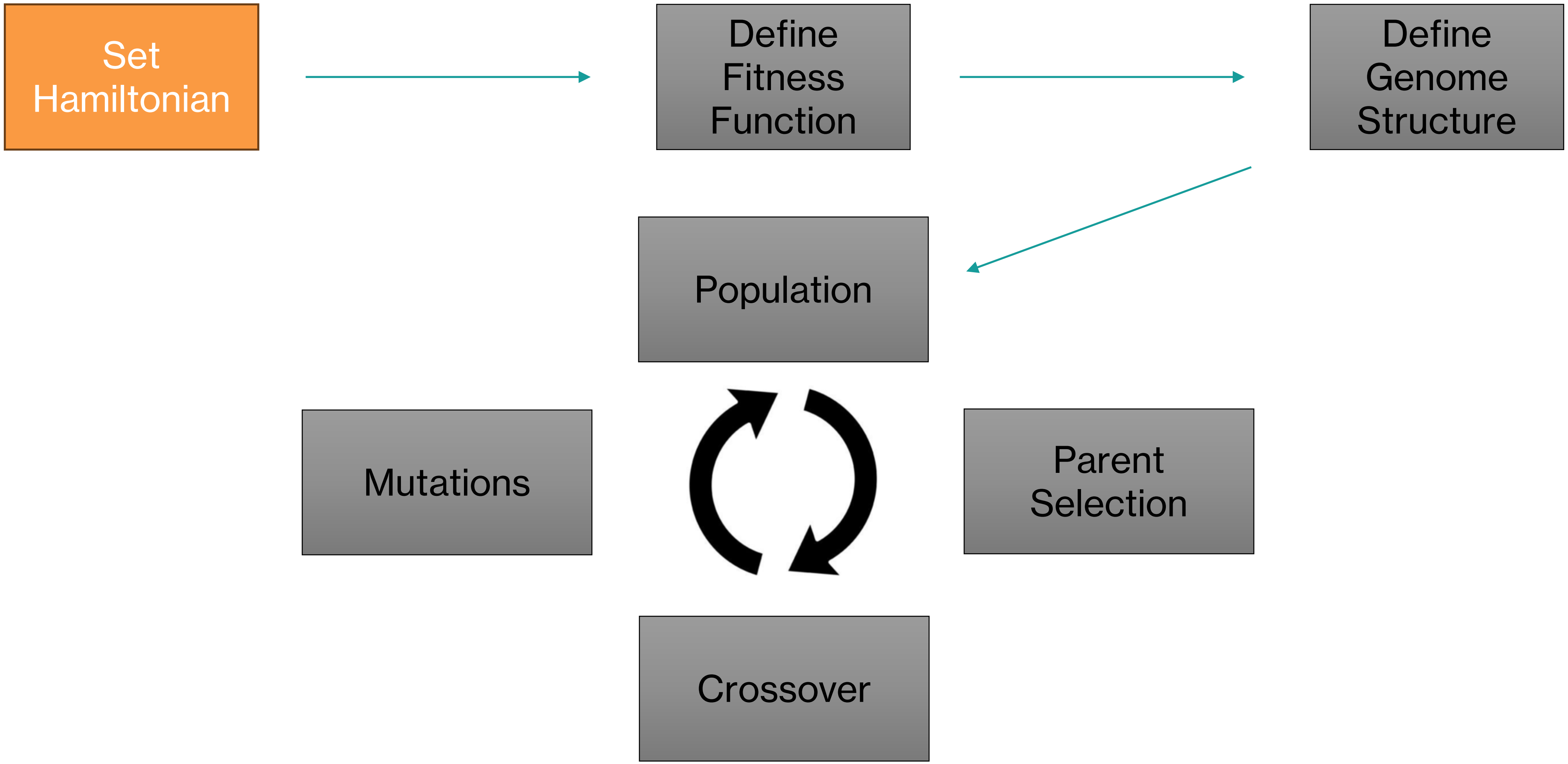
- Evolution theory
- Survival of the fittest
- Crossover/Mutations

General Advantages

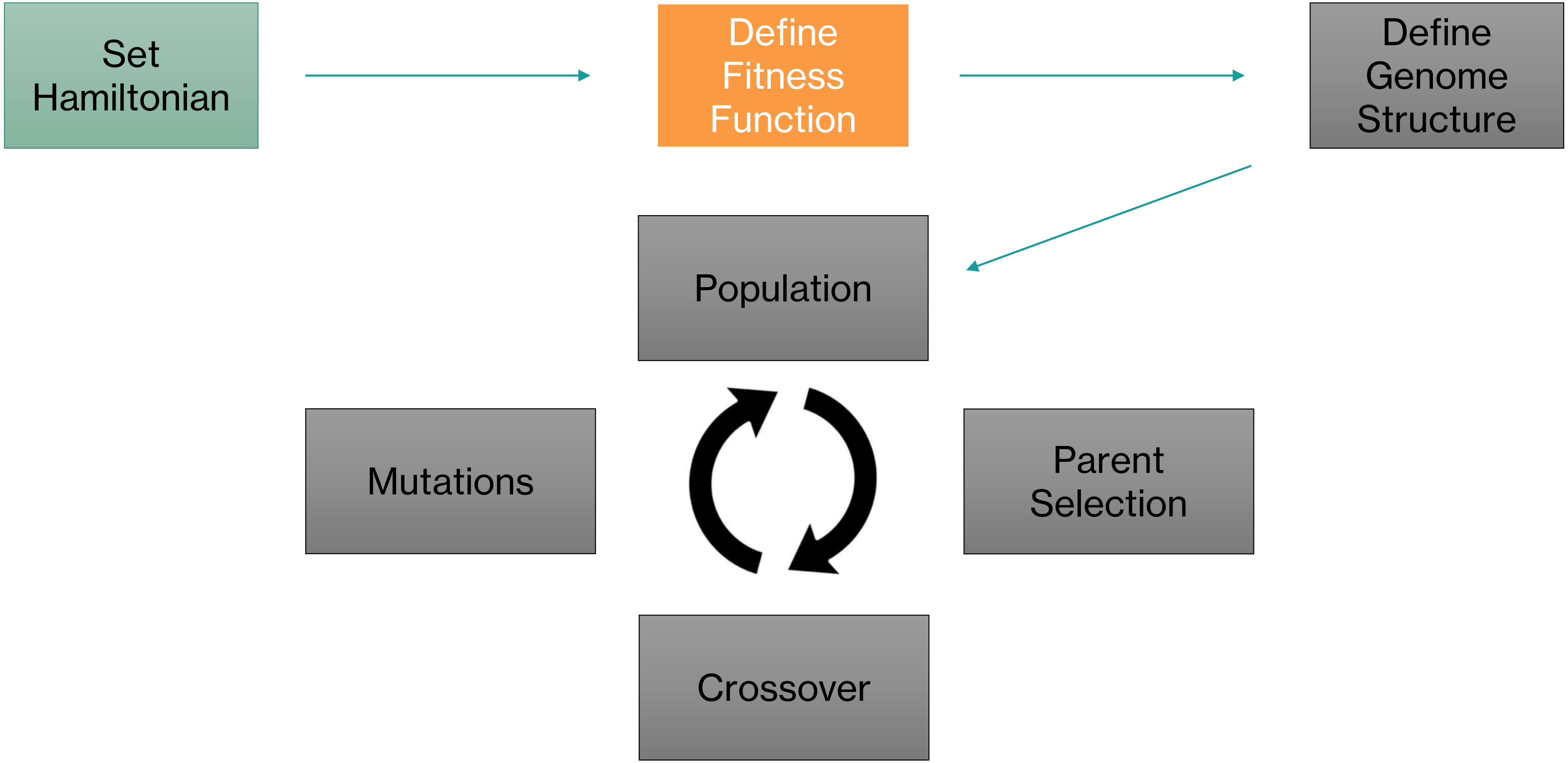
- **No gradient information**
- **Naturally handles discrete problems**



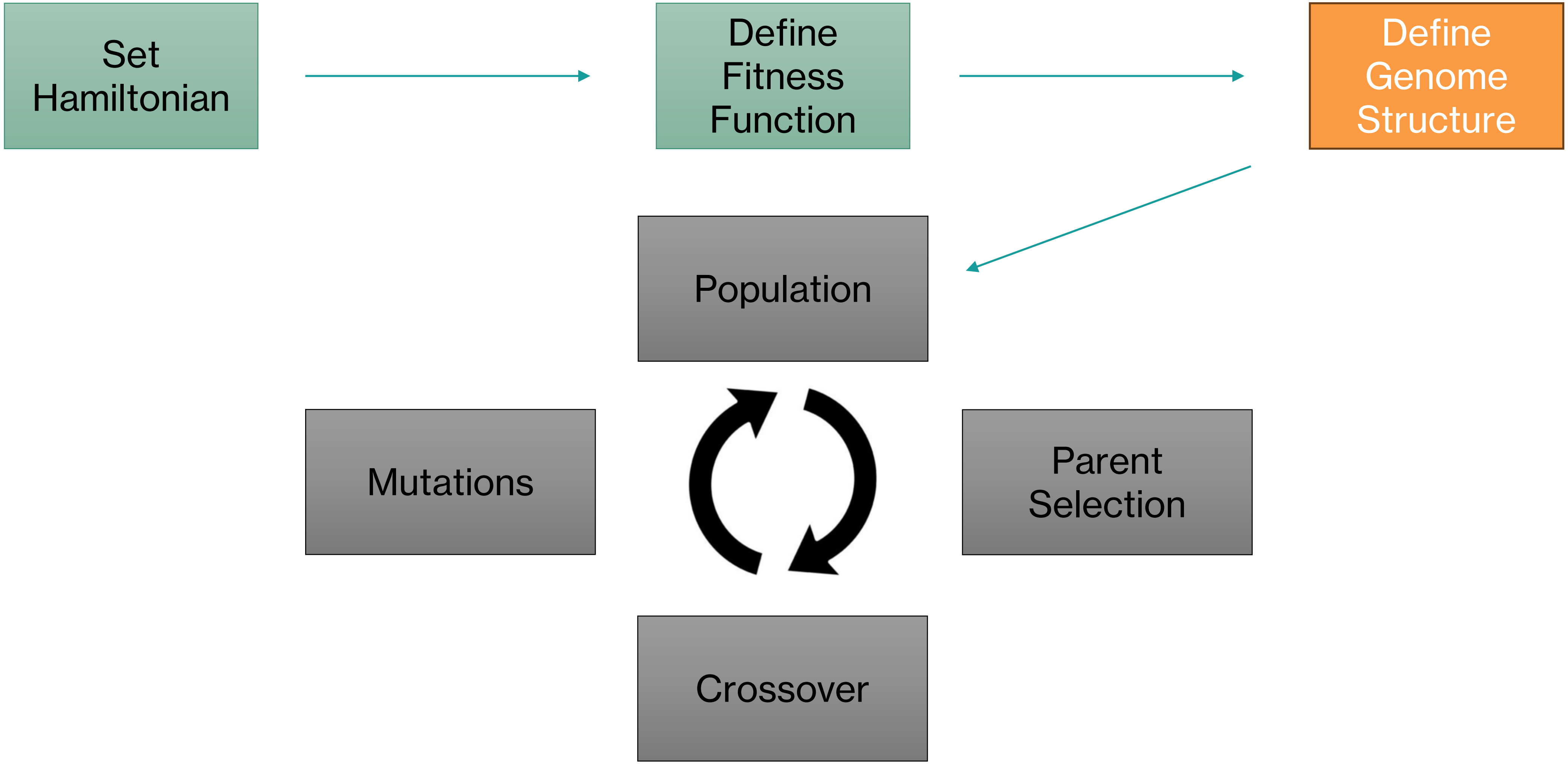
Introduction
Genetic Algorithm



Introduction
Genetic Algorithm



Code Implementation
Structure

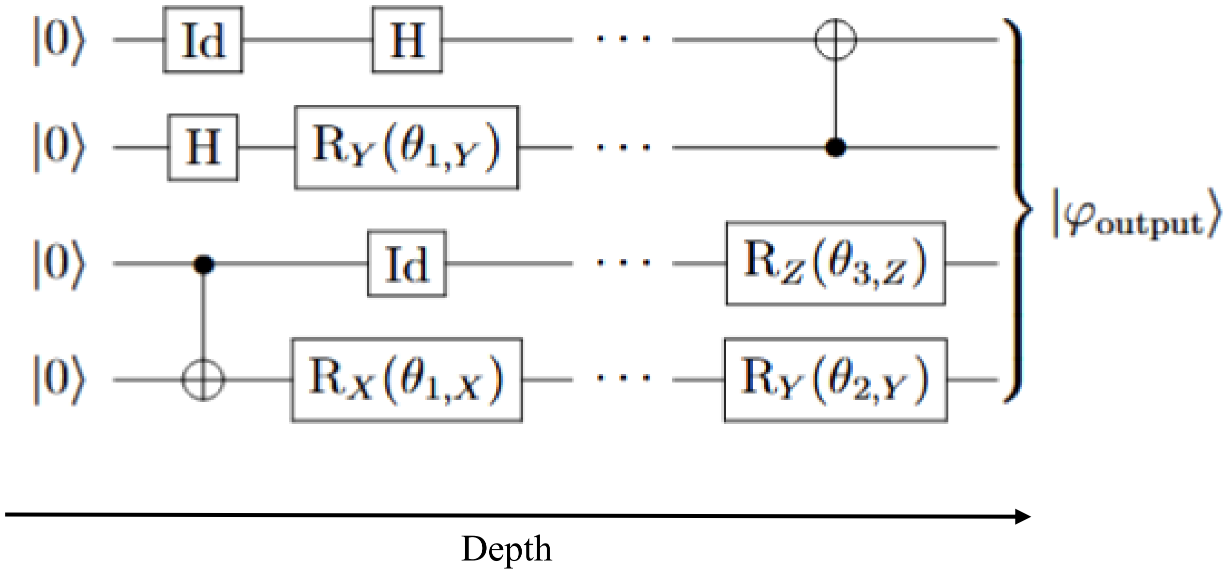


Code Implementation
Structure

Genetic Algorithm file	Main file
❖ First class defines the circuits and methods (Hamiltonian, mutation, crossover, etc) ❖ Second class performs the evolution by selecting the parents and generating the offsprings	❖ Calls **GA file** to execute the evolutionary algorithm ❖ Plots the energies and magnetisation

Code Implementation
Tensor and circuit generation

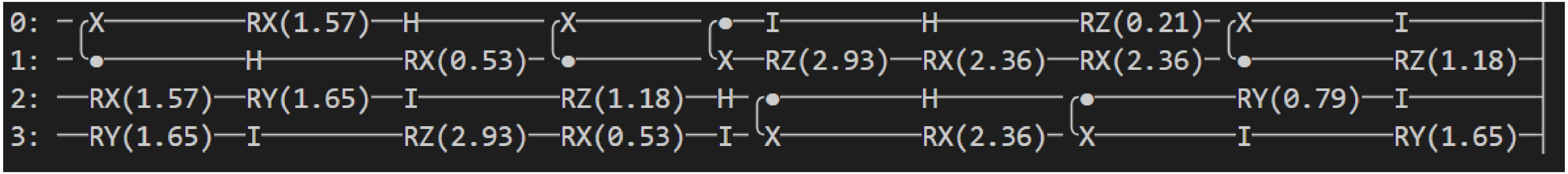
- Fixed number of qubits
- Implemented gates:
  - Identity (Id)
  - Hadamard (Hd)
  - CNOT
  - X, Y, Z rotations:
    - Three predetermined angles



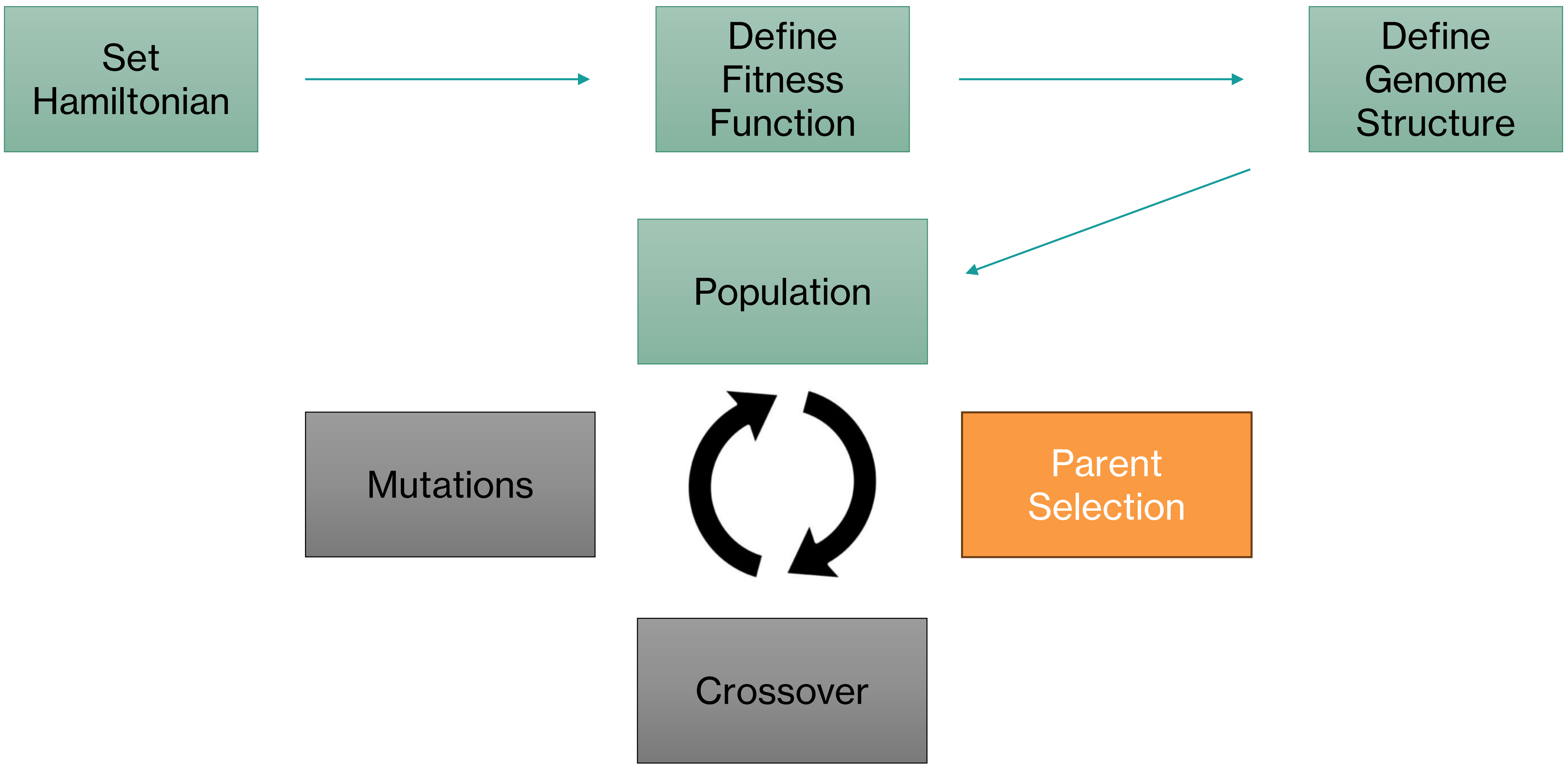
Code Implementation
Tensor and circuit generation

- Circuits saved as (*num_qubits* × *depth*) tensor
- Converted to PennyLane quantum circuits for evaluating their energies (fitness)
- Initialisation: generate *num_circuits* random circuits

$$\text{Number of qubits}\left\{\underbrace{\begin{bmatrix} \mathrm{C_0NOT_1} & \mathrm{R_X}(\theta_{2,X}) & \dots \\ \mathrm{C_0NOT_1} & & \ddots \\ \vdots & & \end{bmatrix}}_{\text{Depth}}\right.$$

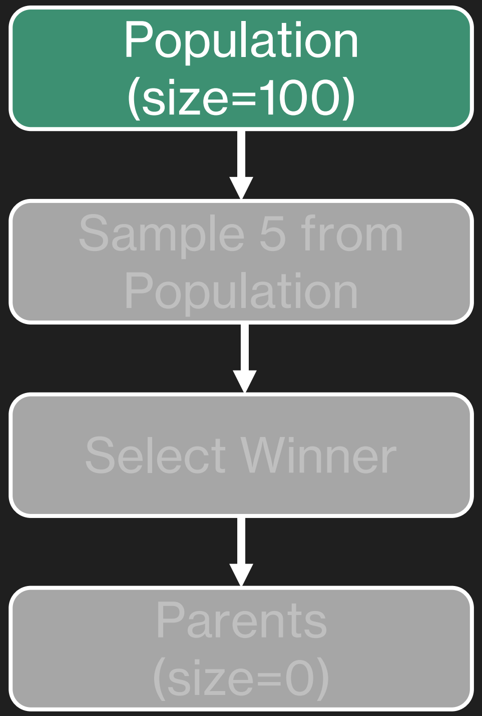


Code Implementation



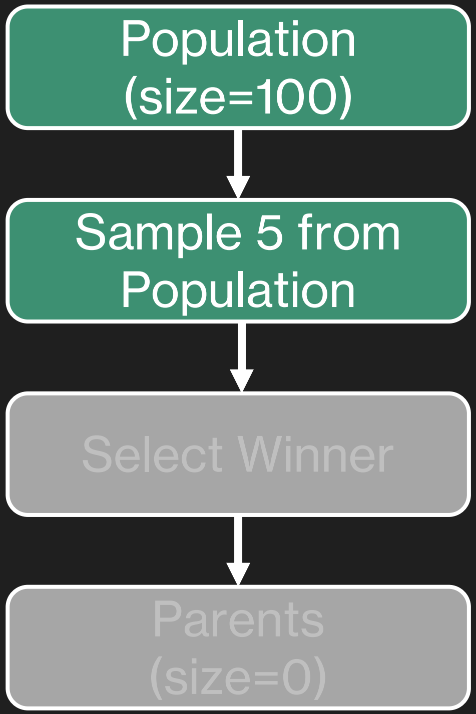
Code Implementation
Parent Selection

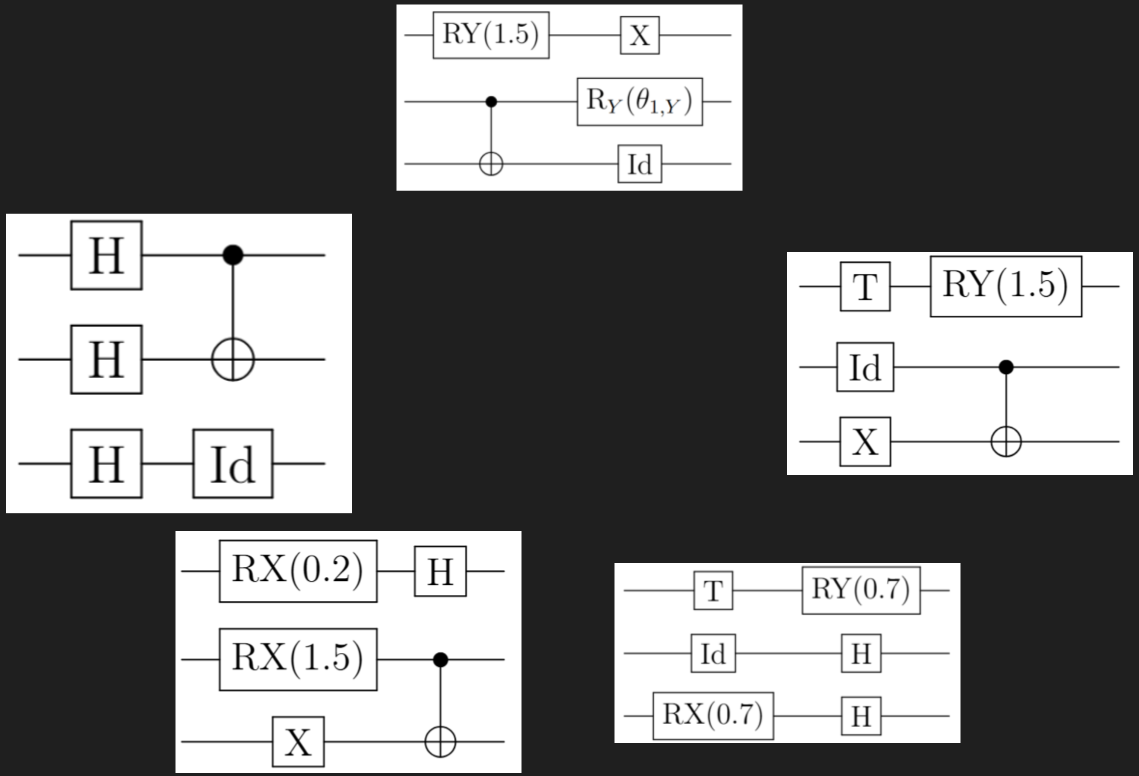
```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```



Code Implementation
Parent Selection

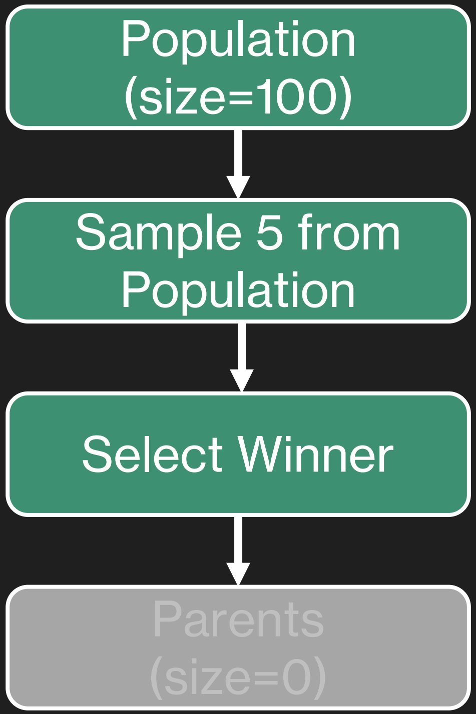
```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```

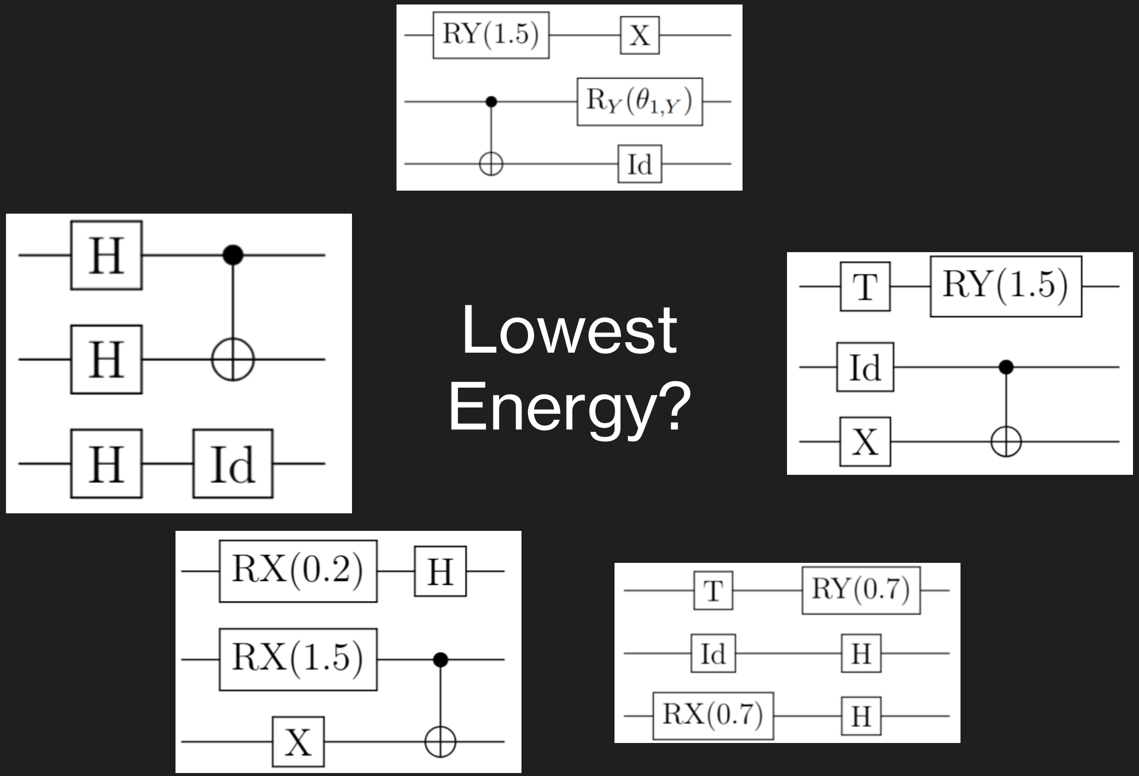




Code Implementation
Parent Selection

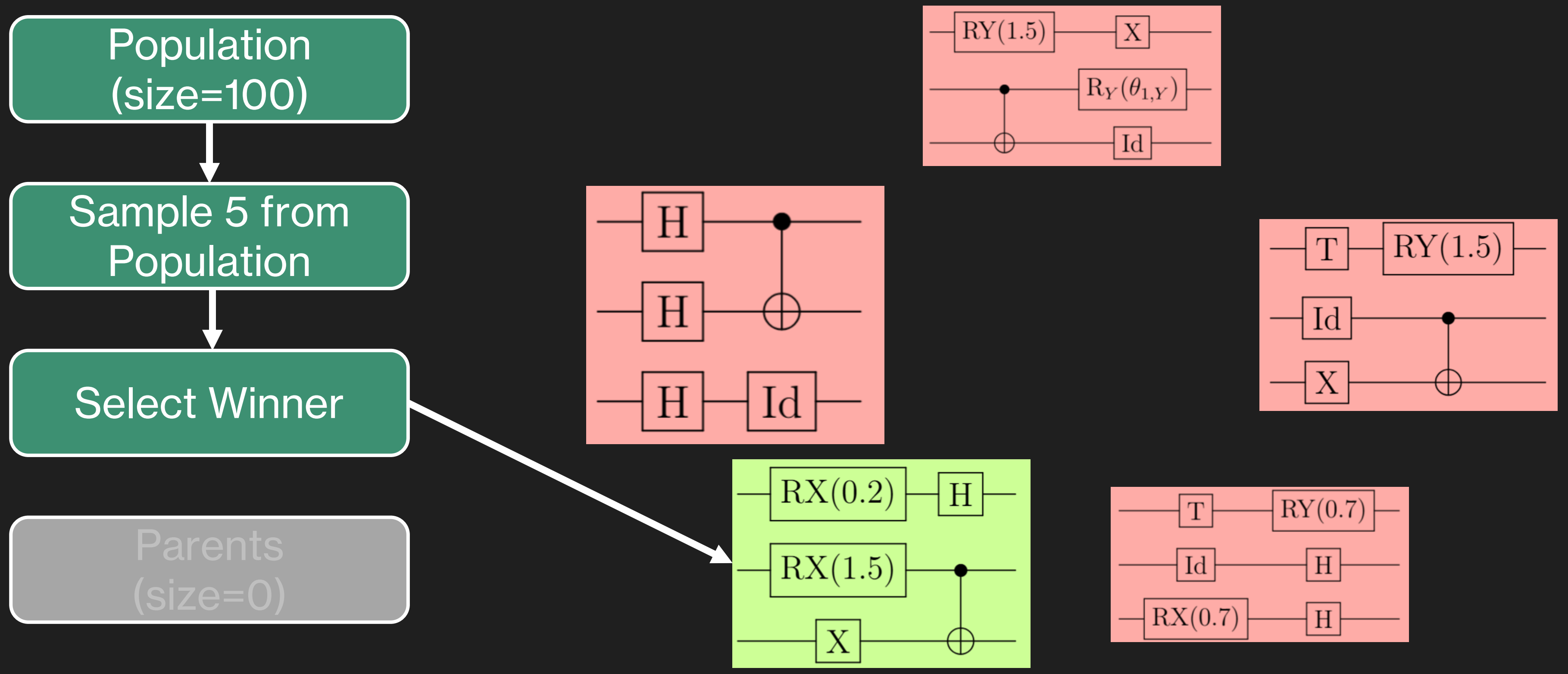
```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```





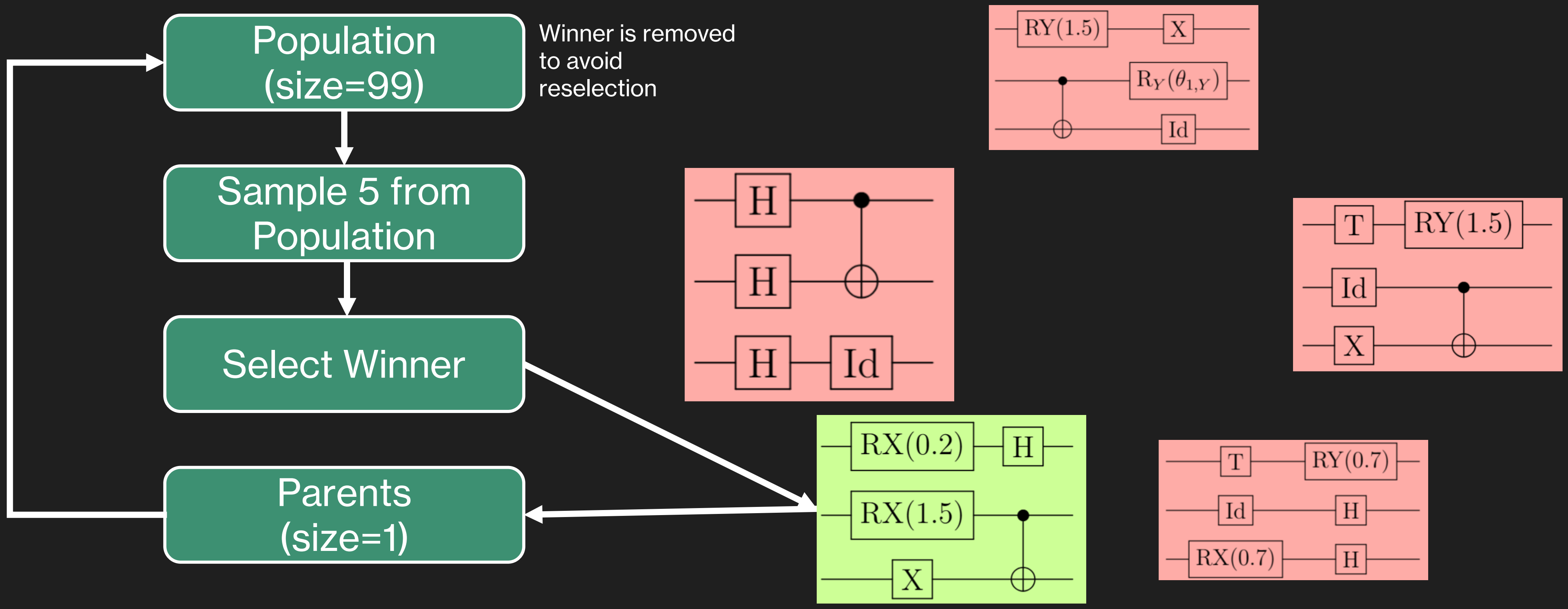
Code Implementation
Parent Selection

```python
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```



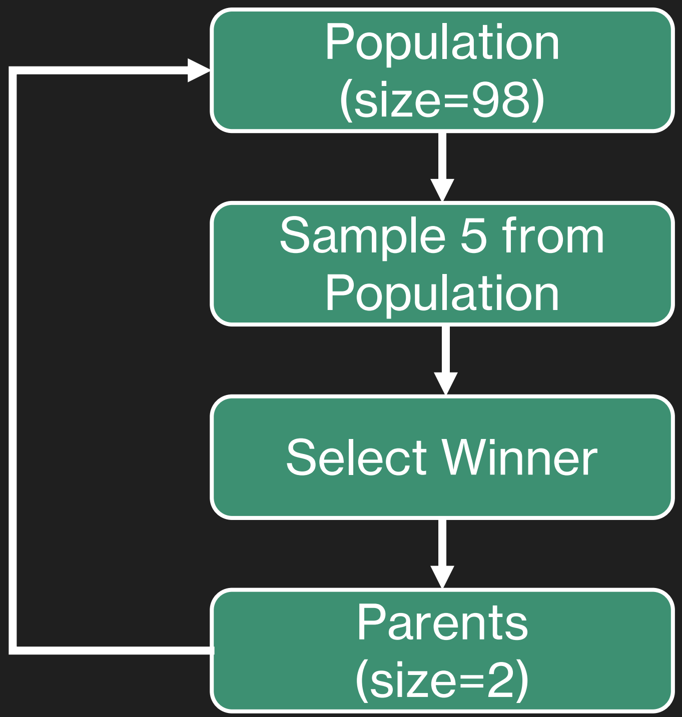
Code Implementation
Parent Selection

```python
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```



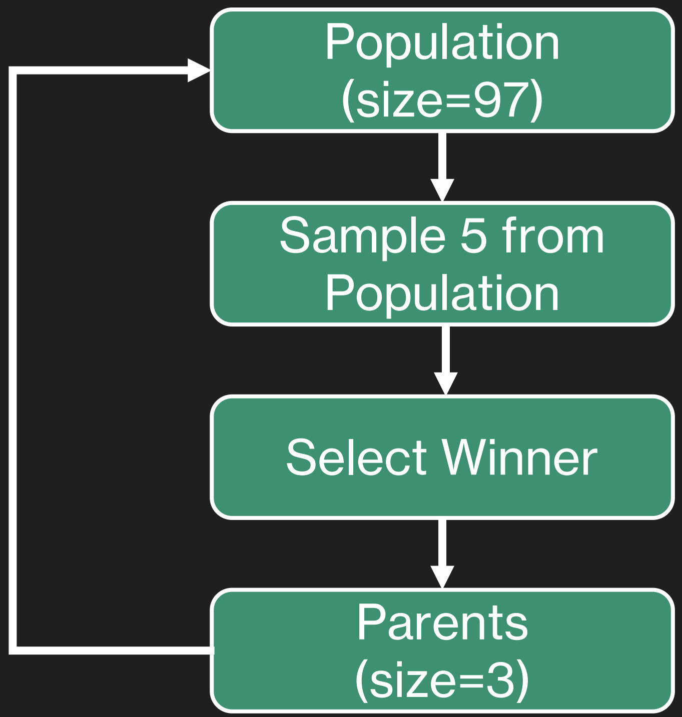
Code Implementation
Parent Selection

```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```



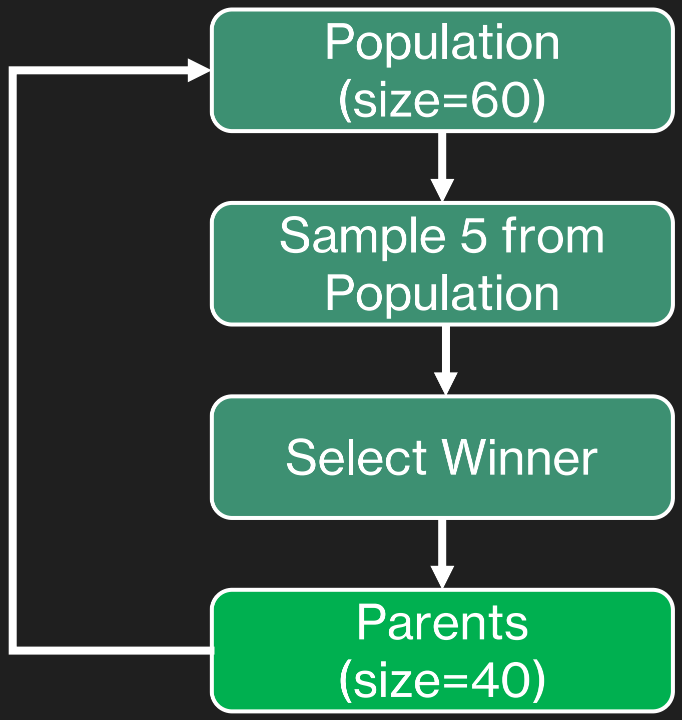
Code Implementation
Parent Selection

```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```

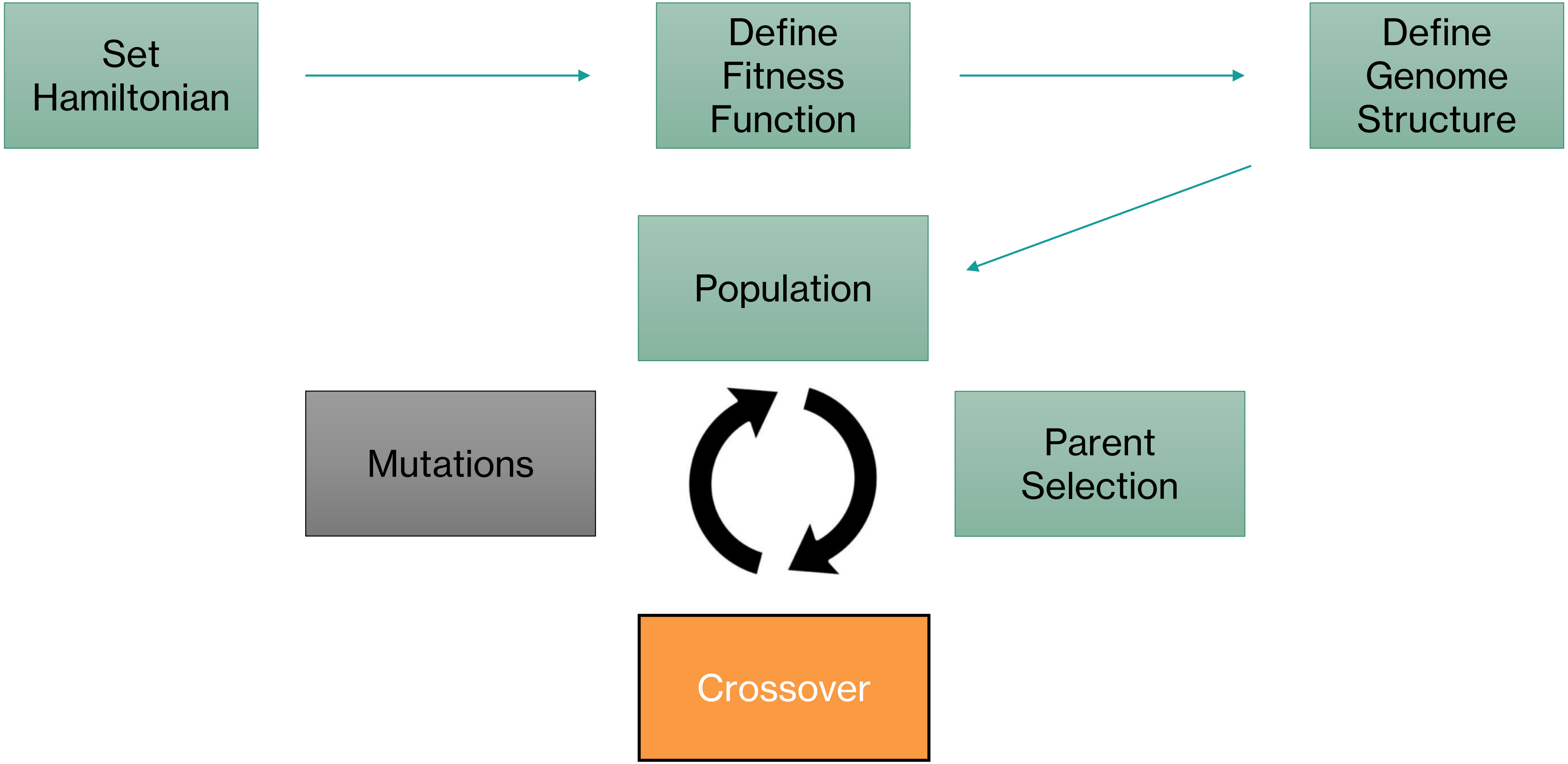


Code Implementation
Parent Selection

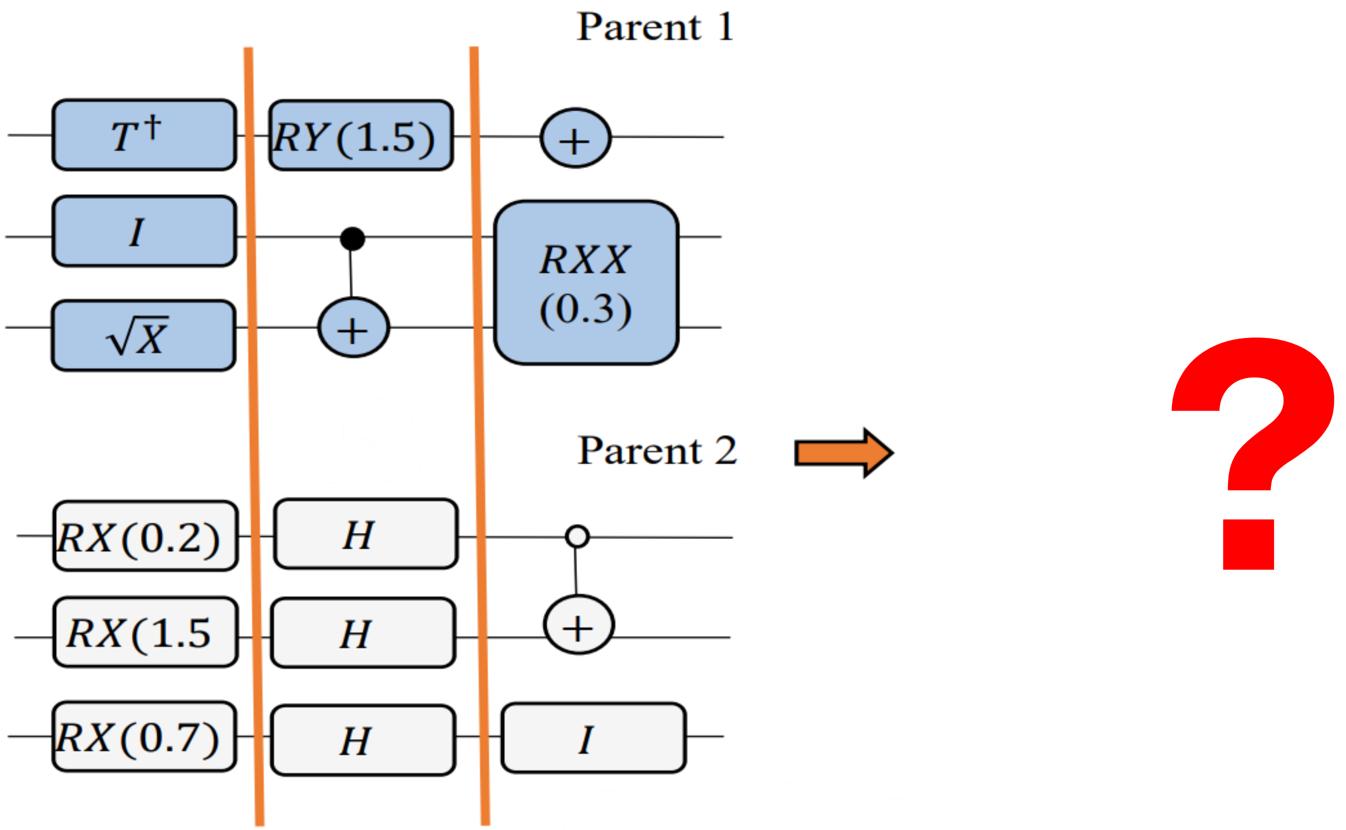
```
def selection(self, tournament_size=5, num_selected=0.4*num_circuits):
```



Code Implementation



Code Implementation
Crossover – Two Strategies

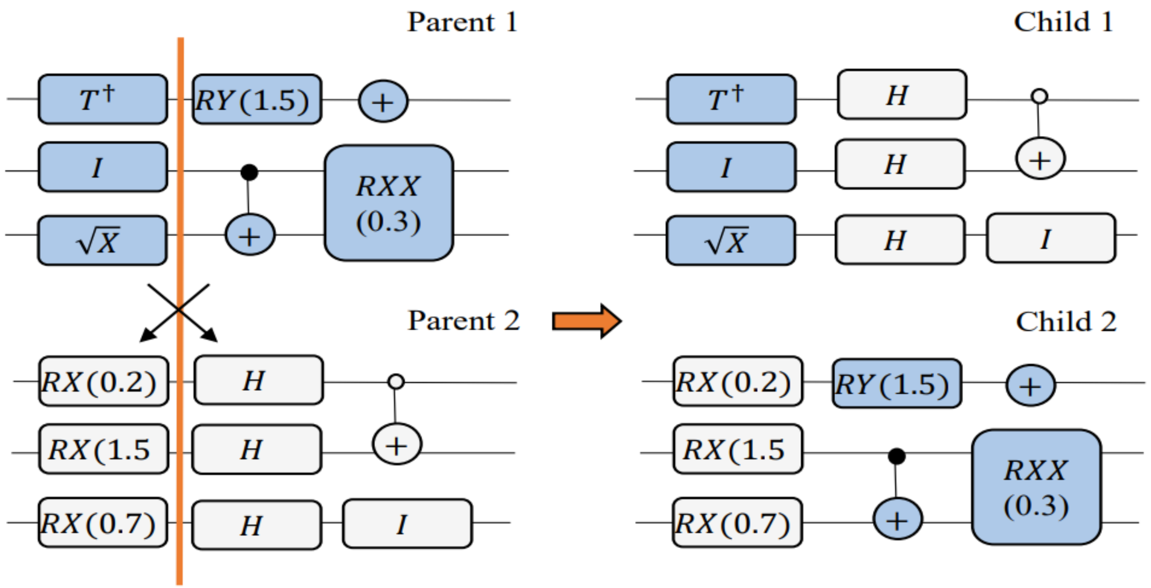


Sünkel, L., Martyniuk, D., Mattern, D., Jung, J., & Paschke, A. (2023). GA4QCO: genetic algorithm for quantum circuit optimization.

Code Implementation
Strategy 1 – Blind Crossover

Step 1: Select Random Crossover Layer

Step 2: Split and Recombine



Single-point crossover

Sünkel, L., Martyniuk, D., Mattern, D., Jung, J., & Paschke, A. (2023). GA4QCO: genetic algorithm for quantum circuit optimization.

Code Implementation

Strategy 2: Entanglement-aware Crossover

arXiv > cs > arXiv:2504.17923

Computer Science > Emerging Technologies

[Submitted on 24 Apr 2025]

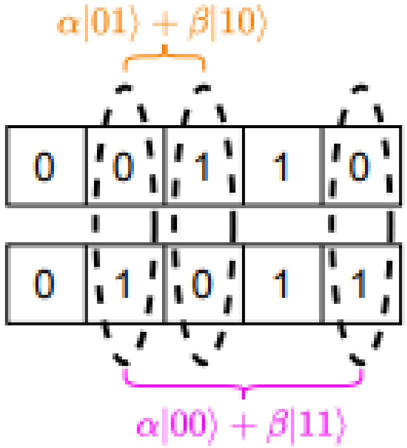
EAQGA: A Quantum-Enhanced Genetic Algorithm with Novel Entanglement-Aware Crossovers

Mohammad Kashfi Haghighi, Matthieu Fortin-Deschênes, Christophe Pere, Mickaël Camus

EAQGA: A Quantum-Enhanced Genetic Algorithm with Novel Entanglement-Aware Crossovers, Haghighi et al., 2025

Code Implementation
Strategy 2: Entanglement-aware Crossover

Consider a five-qubit problem with the following two parent solutions:



Ensure highly correlated qubits remain entangled across generations

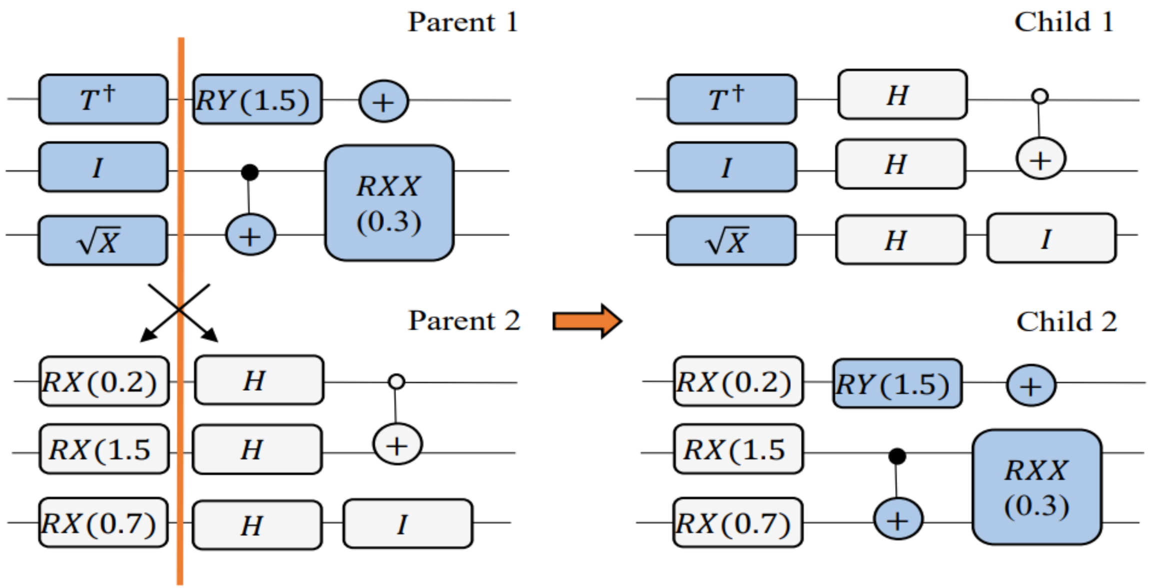
EAQGA: A Quantum-Enhanced Genetic Algorithm with Novel Entanglement-Aware Crossovers, Haghighi et al., 2025

Code Implementation
Strategy 2 – Entanglement-aware Crossover

Step 1: Find Compatible Blocks – i.e. blocks with equal entanglement generation

Step 2: Select Compatible Crossover Layer

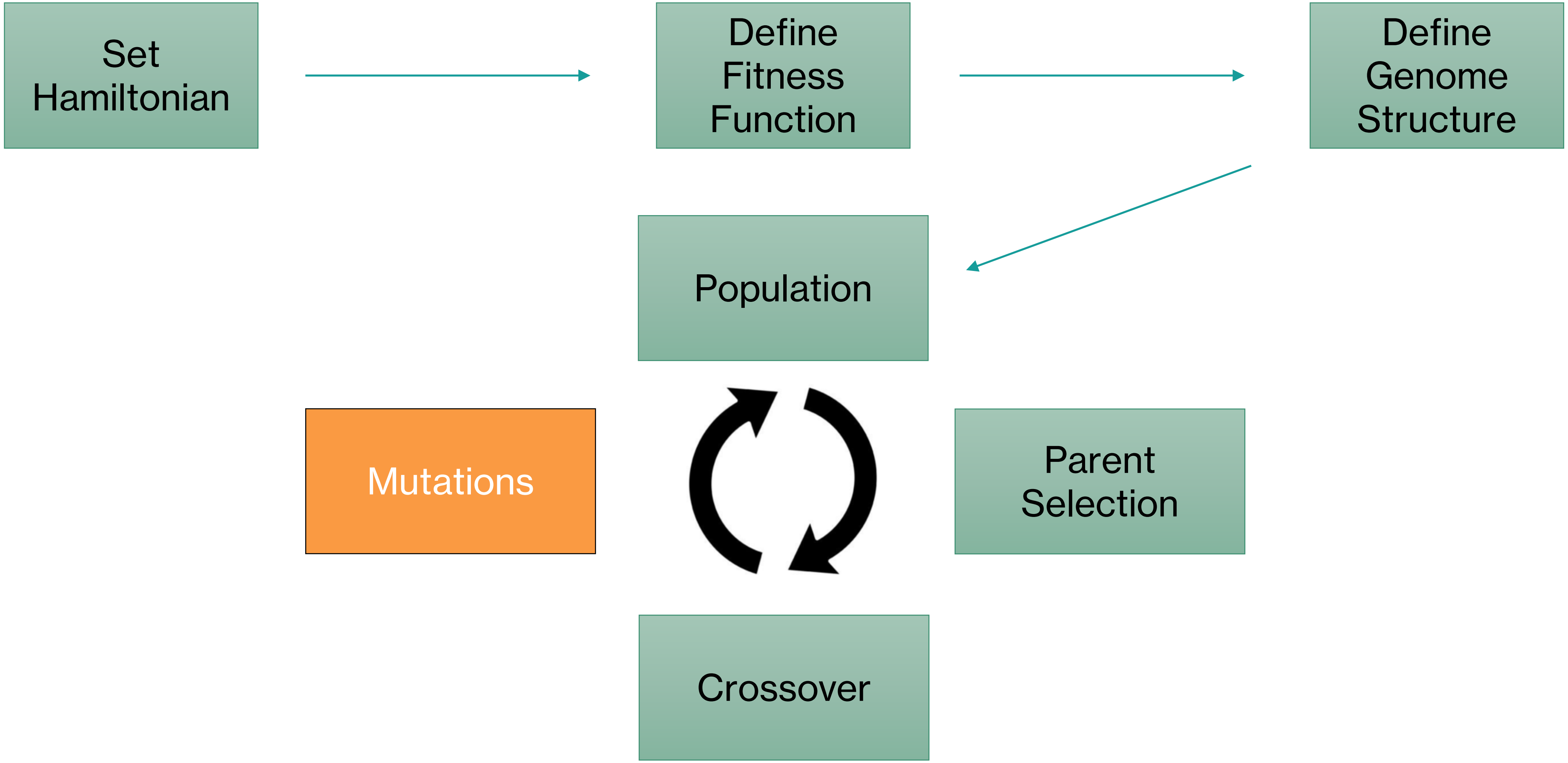
Step 3: Split and Recombine



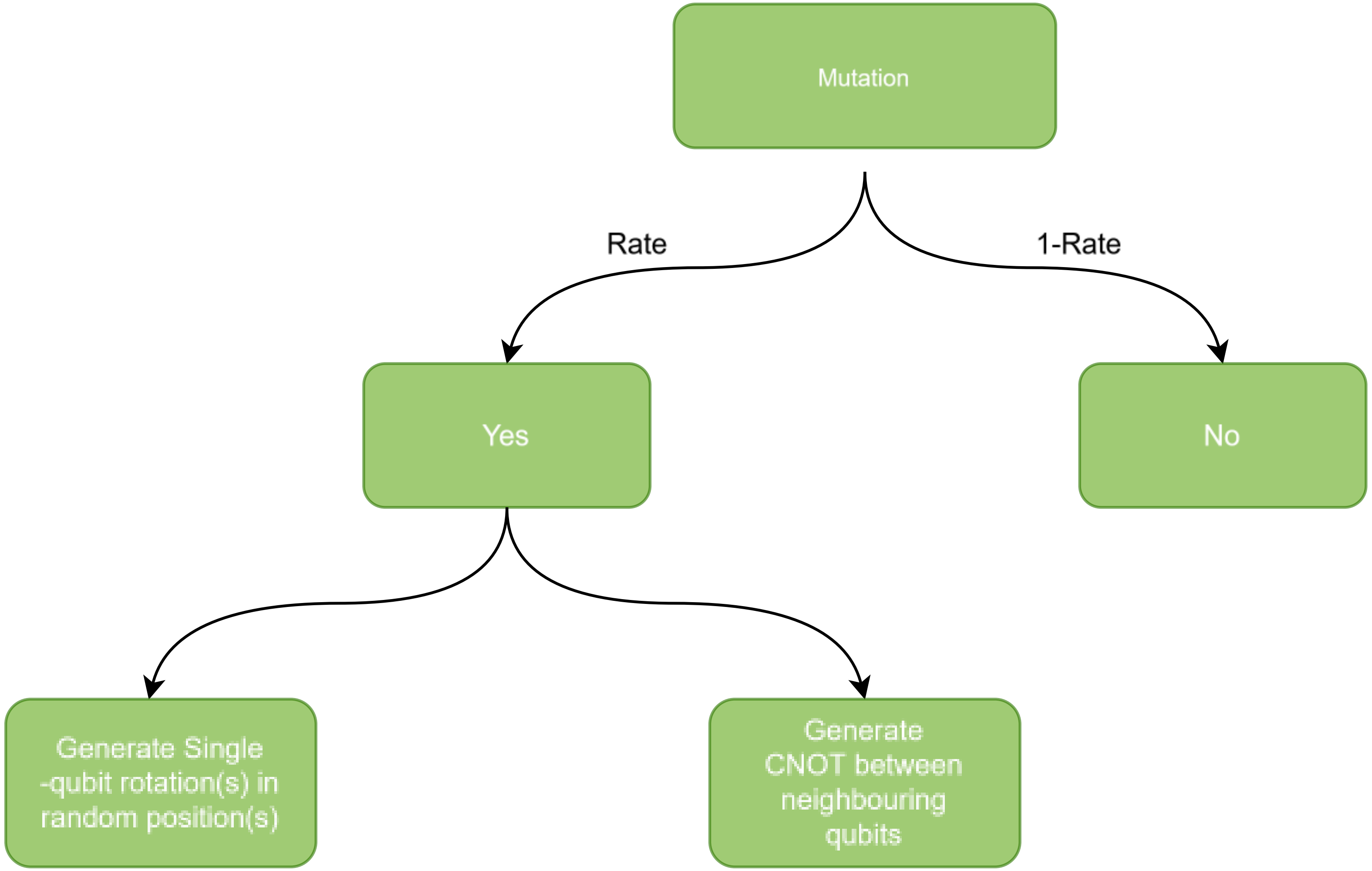
Single-point crossover

Sünkel, L., Martyniuk, D., Mattern, D., Jung, J., & Paschke, A. (2023). GA4QCO: genetic algorithm for quantum circuit optimization.

Code Implementation

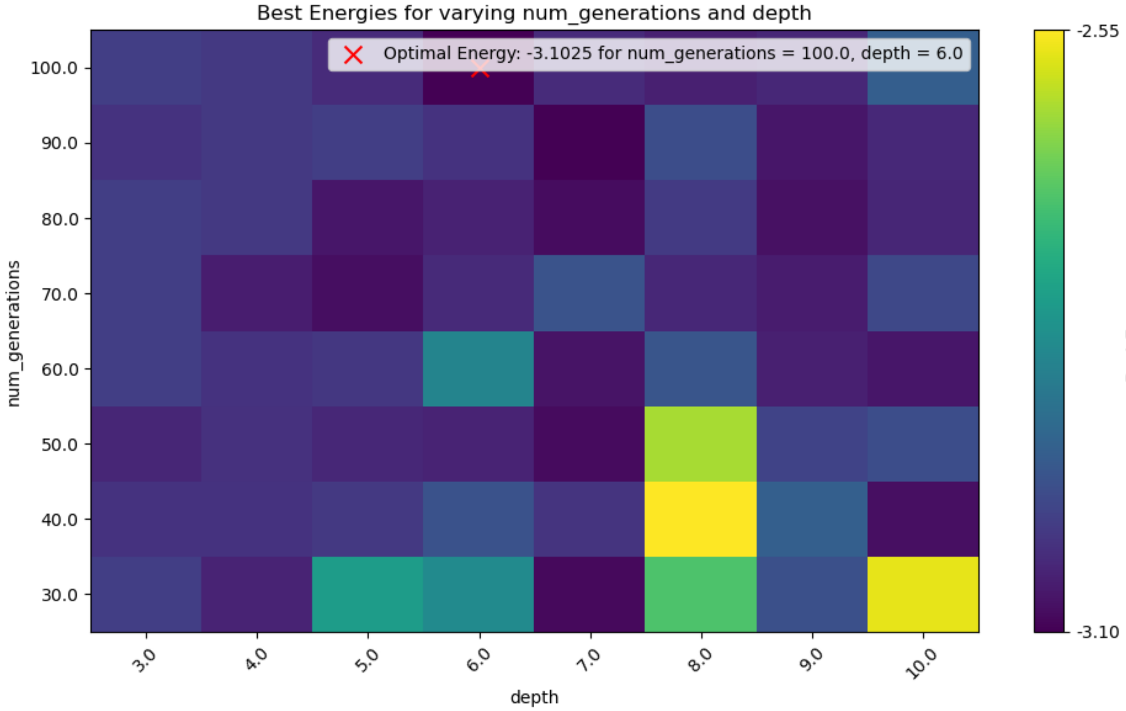


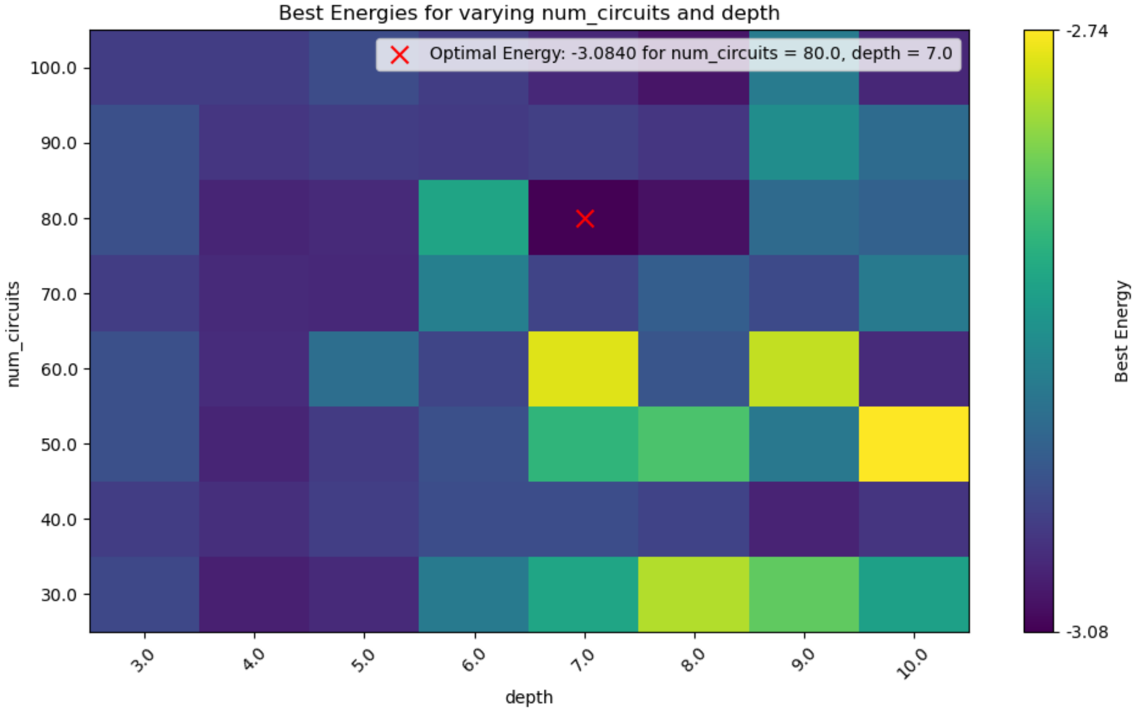
Code Implementation
Mutations



Code Implementation
Hyperparameter optimisation

- h=0.32





Results

Can Evolutionary Algorithms be used to find ground state energies?

Can we replicate the phase transition of the transverse-field Ising Hamiltonian?

Results

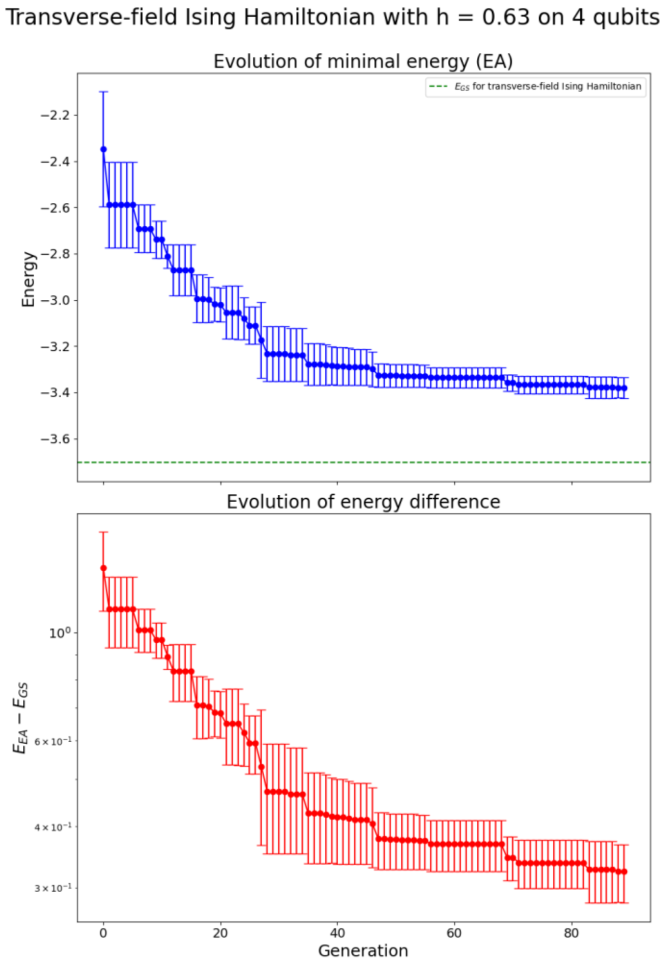
Can Evolutionary Algorithms be used to find ground state energies?

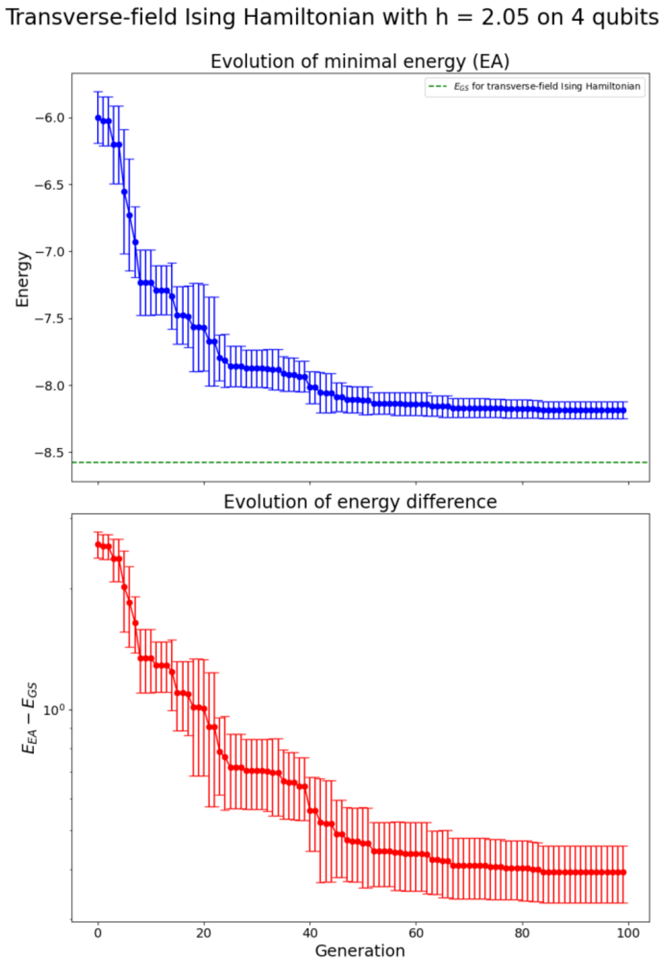
Recall:

$$H = -J \sum_{(i,j)\in E} Z_i Z_j + h \sum_i X_i$$

We measure (and optimise) energy $\mathrm{E} = \langle \Psi | H | \Psi \rangle$ by changing gates rather than angles

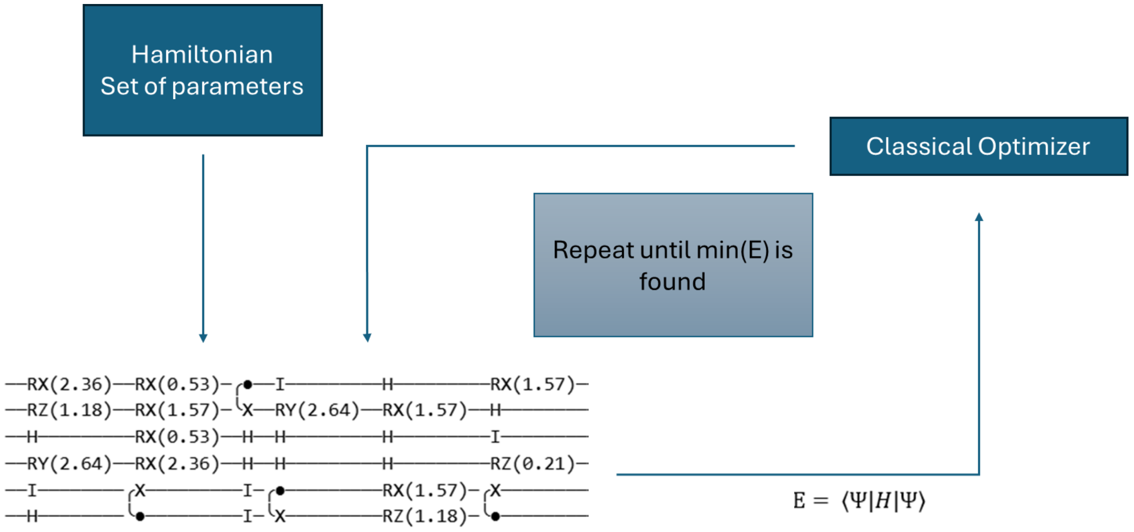
Results

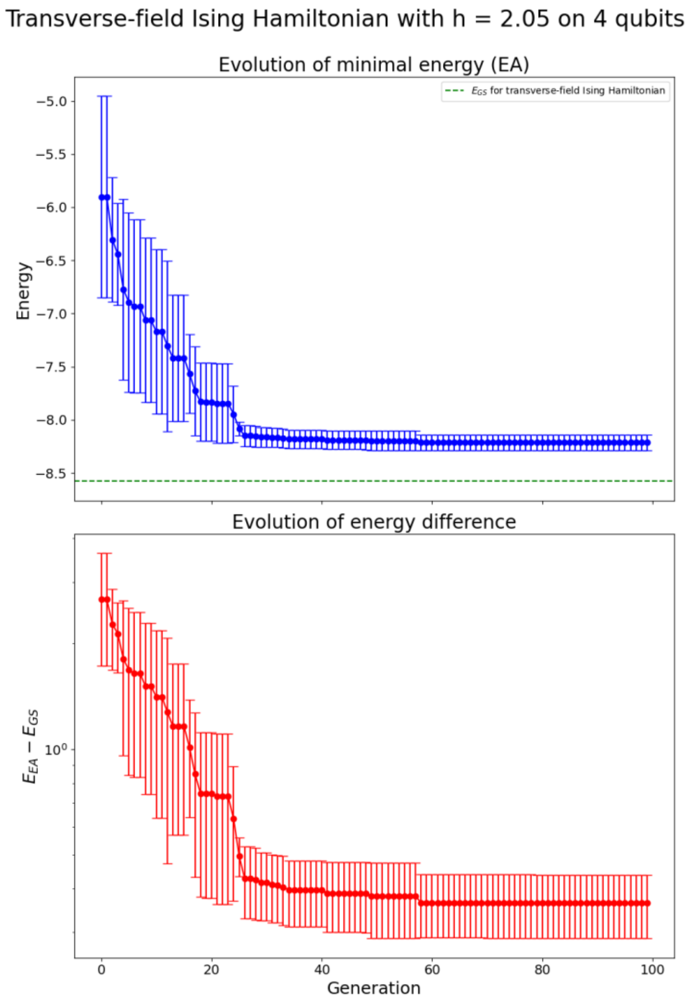




Energy optimisation over GA generations for four qubits, averaged over five runs.
Left: result for $h = 0.63$, Right: result for $h = 2.05$

Results





Energy optimisation over GA generations (+VQE) for four qubits, averaged over five runs.

Results

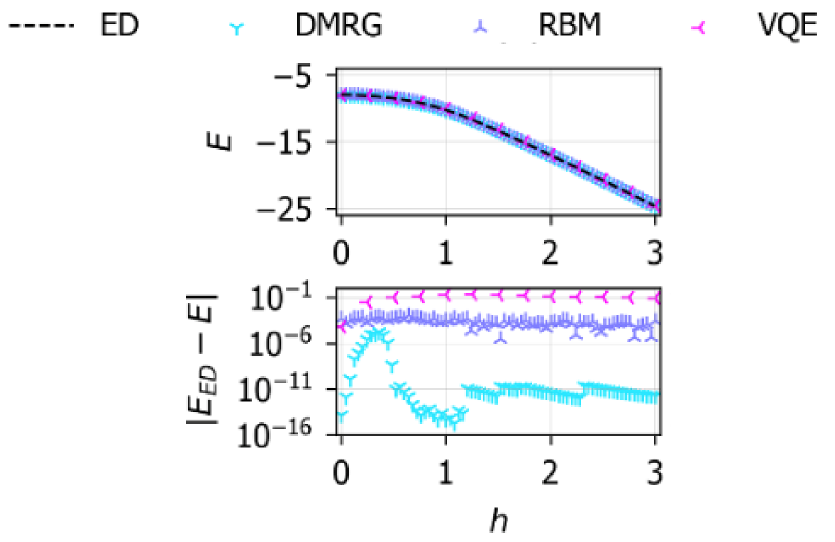
arXiv > quant-ph > arXiv:2409.13008

Quantum Physics

[Submitted on 19 Sep 2024 (v1), last revised 9 Apr 2025 (this version, v2)]

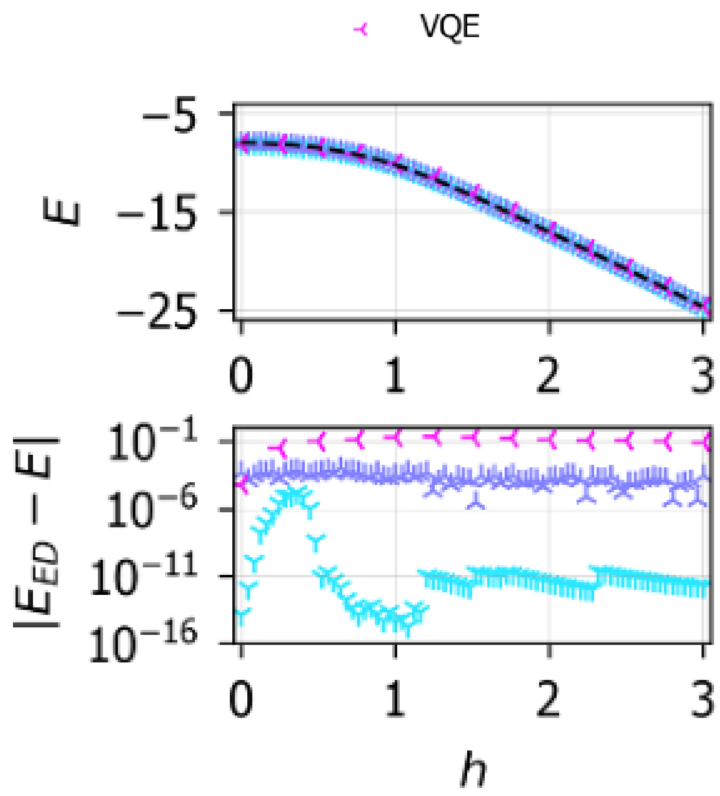
Quantum resources of quantum and classical variational methods

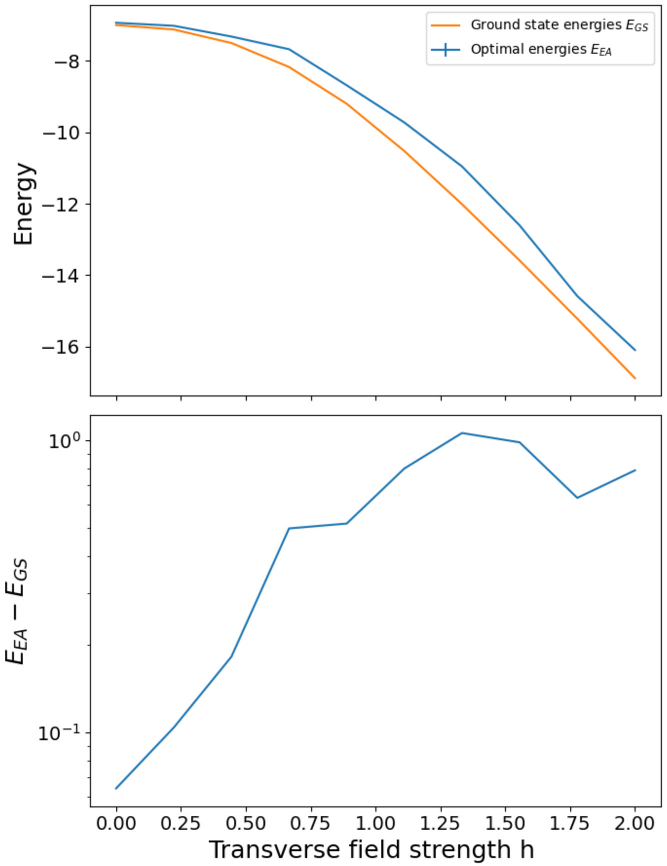
Thomas Spriggs, Arash Ahmadi, Bokai Chen, Eliska Greplova



Quantum resources of quantum and classical variational methods, Spriggs et al., 2024

Results





Energy and energy difference to GS vs. transverse-field strength h.
Left: VQE result on eight qubits, Right: EA (+VQE) result on eight qubits.

Results

Can Evolutionary Algorithms be used to find ground state energies?

Yes, either as a stand-alone approach or as ansatz for VQE optimisation. ☑

Can we replicate the phase transition of the transverse-field Ising Hamiltonian?

Results

Can we replicate the phase transition of the transverse-field Ising Hamiltonian?

Recall:

$$H = -J \sum_{(i,j)\in E} Z_i Z_j + h \sum_i X_i$$

Quantum phase transition in (absolute) Z - magnetisation $Z_m = \frac{1}{N} \left| \sum_i Z_i \right|$ at

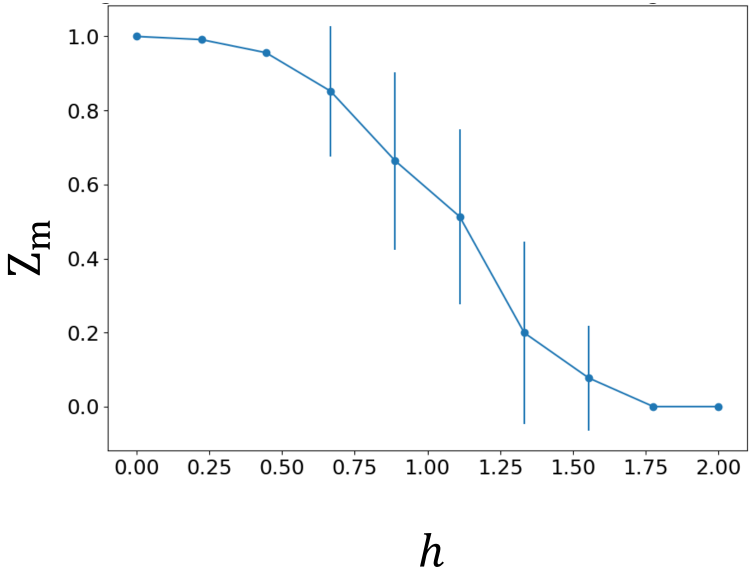
$$\left|\frac{h}{J}\right| = 1 \quad \Longrightarrow \quad |h| = 1$$

Ordered phase (ferromagnetic regime) for $|h| < 1$, disordered phase for $|h| > 1$

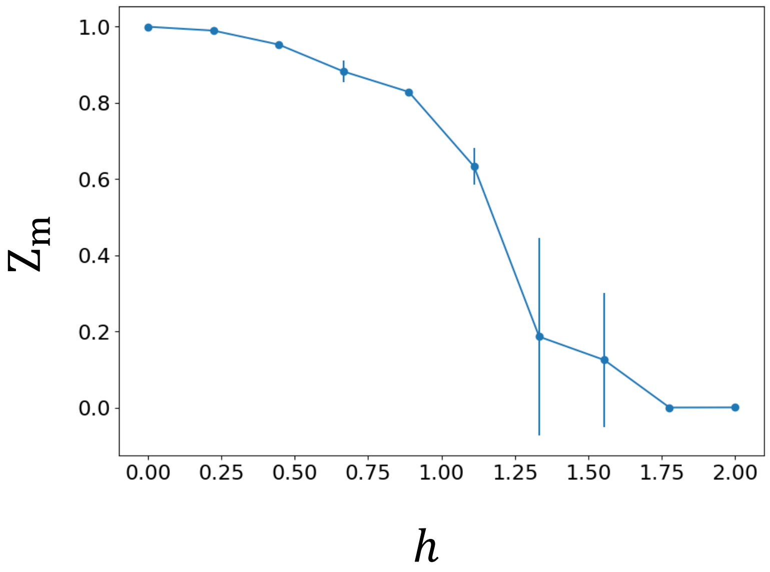
Results – Phase Transition

Quantum phase transition in $\mathrm{Z_m}$ at $|h| = 1$:

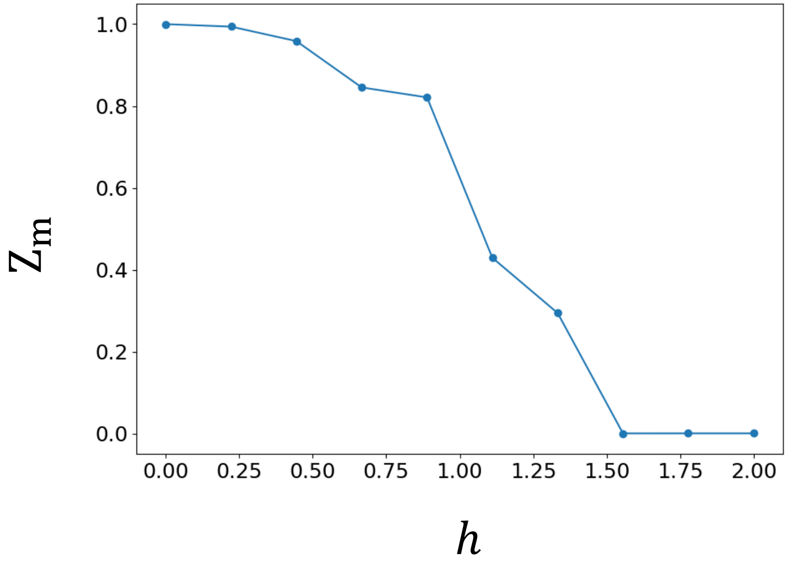
$\mathrm{Z_m}$ vs. h, five qubits



$\mathrm{Z_m}$ vs. h, six qubits



$\mathrm{Z_m}$ vs. h, eight qubits



Absolute Z-magnetisation vs. transverse-field strength h for various system sizes.
All results incl. VQE optimisation, and are averaged over 25, three and one run, respectively.

Results

Can Evolutionary Algorithms be used to find ground state energies?

Yes, either as a stand-alone approach or as ansatz for VQE optimisation. ☑

Can we replicate the phase transition of the transverse-field Ising Hamiltonian?

The absolute Z-magnetisation shows a (smeared) phase transition. ☑

Conclusion

- Evolution-inspired principles of selection, crossover and mutation are applicable finding ground state energies and identifying phase transitions
- Genetic Algorithms rely on Exploration (mutation) vs. Exploitation (selection) trade-off to optimize gates rather than angles
- GAs can act as an ansatz for VQE optimisation

Limitations & Future Work

- Inherently probabilistic approach, no convergence guarantees
- Noise-free setting (but handling of sampling uncertainty)
- (Over-)Reliance on fixed set of angles; could be alleviated by weight-agnostic* models
- Abundance of hyperparameters to investigate (also with respect to scaling)
- Scalable approach due to gradient-free design (?)

* cf. Weight Agnostic Neural Networks, Gaier and Ha, 2019

References

[1] Adam Gaier and David Ha. *Weight Agnostic Neural Networks*. 2019. arXiv: 1906.04358 [cs.LG]. URL: https://arxiv.org/abs/1906.04358.

[2] Mohammad Kashfi Haghighi et al. *EAQGA: A Quantum-Enhanced Genetic Algorithm with Novel Entanglement-Aware Crossovers*. 2025. arXiv: 2504.17923 [cs.ET]. URL: https://arxiv.org/abs/2504.17923.

[3] Moshe Sipper. *How to Build a Genetic Algorithm from Scratch in Python With Just 33 Lines of Code: Genetic Algorithm for Quantum Circuit Optimization*. 2023. URL: https://levelup.gitconnected.com/tiny-genetic-algorithm-33-line-version-and-3-line-version-38a851141512.

[4] Thomas Spriggs et al. *Quantum resources of quantum and classical variational methods*. 2025. arXiv: 2409.13008 [quant-ph]. URL: https://arxiv.org/abs/2409.13008.

[5] Leo Sünkel et al. *GA4QCO: Genetic Algorithm for Quantum Circuit Optimization*. 2023. arXiv: 2302.01303 [quant-ph]. URL: https://arxiv.org/abs/2302.01303.

Genetic Algorithms for Quantum Circuits

By Arturo, Badr, Cesar, Lukas and Tim

Appendix: Code Implementation

Gate encoding

```python
# Define a dtype for each gate (this is a class attribute common for all instances of the class)
gate_dtype = np.dtype([
    ("name", "U10"),  # e.g. 'RX', 'CNOT'
    ("qubit_id", "i4"),  # single qubit this gate is applied on
    ("affected_qubits", "O"),  # list of affected qubits (for multi-qubit gates)
    ("parameters", "O"),  # list or float (e.g., angle for rotation gates)
    ("control_qubits", "O"),  # list of control qubits
    ("target_qubits", "O")  # list of target qubits
])

# We define the finite gate set to consist of identity, discrete rotations by certain angles, CNOT and Hadamard
gate_set_names=["Id", "RX", "RY", "RZ", "CNOT", "H"]

angles_x=np.array([0.53, 1.57, 2.36]) # Angles (in radians) for the X rotations
angles_y=np.array([0.79, 1.65, 2.64]) # Angles (in radians) for the Y rotations
angles_z=np.array([0.21, 1.18, 2.93]) # Angles (in radians) for the Z rotations
```

Appendix: Code Implementation
Mutations

- With a certain probability (*rate*) a mutation is performed in each quantum circuit
- A portion of them are made to mutate to a single-qubit gate:
  - If a CNOT is detected for the mutation, then the two qubits involved will be mutated to two (different) single-qubit gates
- The other portion of these mutations will be to generate CNOTs
  - Take a random qubit and a random neighbour to generate a CNOT between them

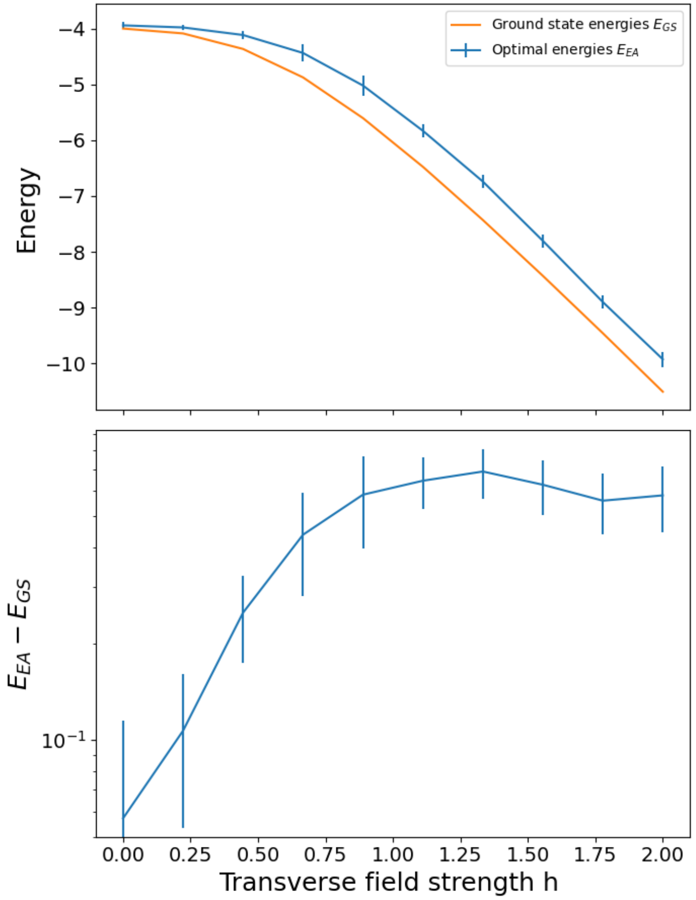
Appendix: Hyperparameters

```
#######################################
############ PARAMETERS ###############
#######################################


num_qubits = 8
mutation_rate = 0.5
hamiltonian_label = 'transverse_field_ising'
transverse_field_strengths = np.linspace(0,3,20)

# variable parameters (for each transverse field strength h)
depths = [4,7,7,5,9,9,9,8,8,9,8,7,7,6,6,10,9,9,9,9]
num_circuits = [80,70,40,60,90,80,90,100,80,90,60,40,80,100,90,90,90,90,100,100]
num_of_generations = [100,100,100,100,90,70,80,90,90,90,90,60,90,100,90,100,90,100,100,100]
```

Results



Energy and energy difference to GS vs. transverse-field strength h.
EA + VQE result on five qubits over 25 iterations.