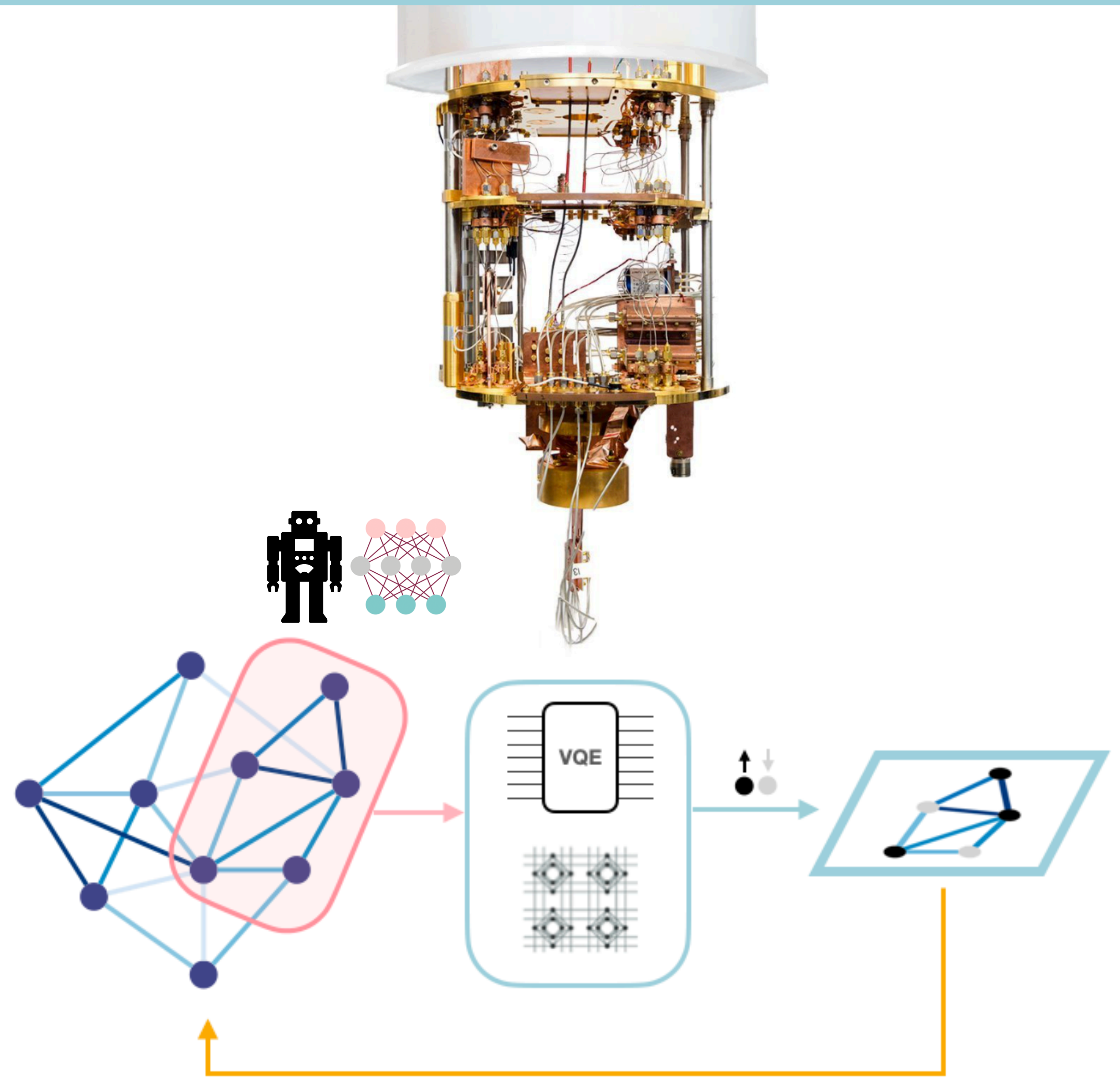
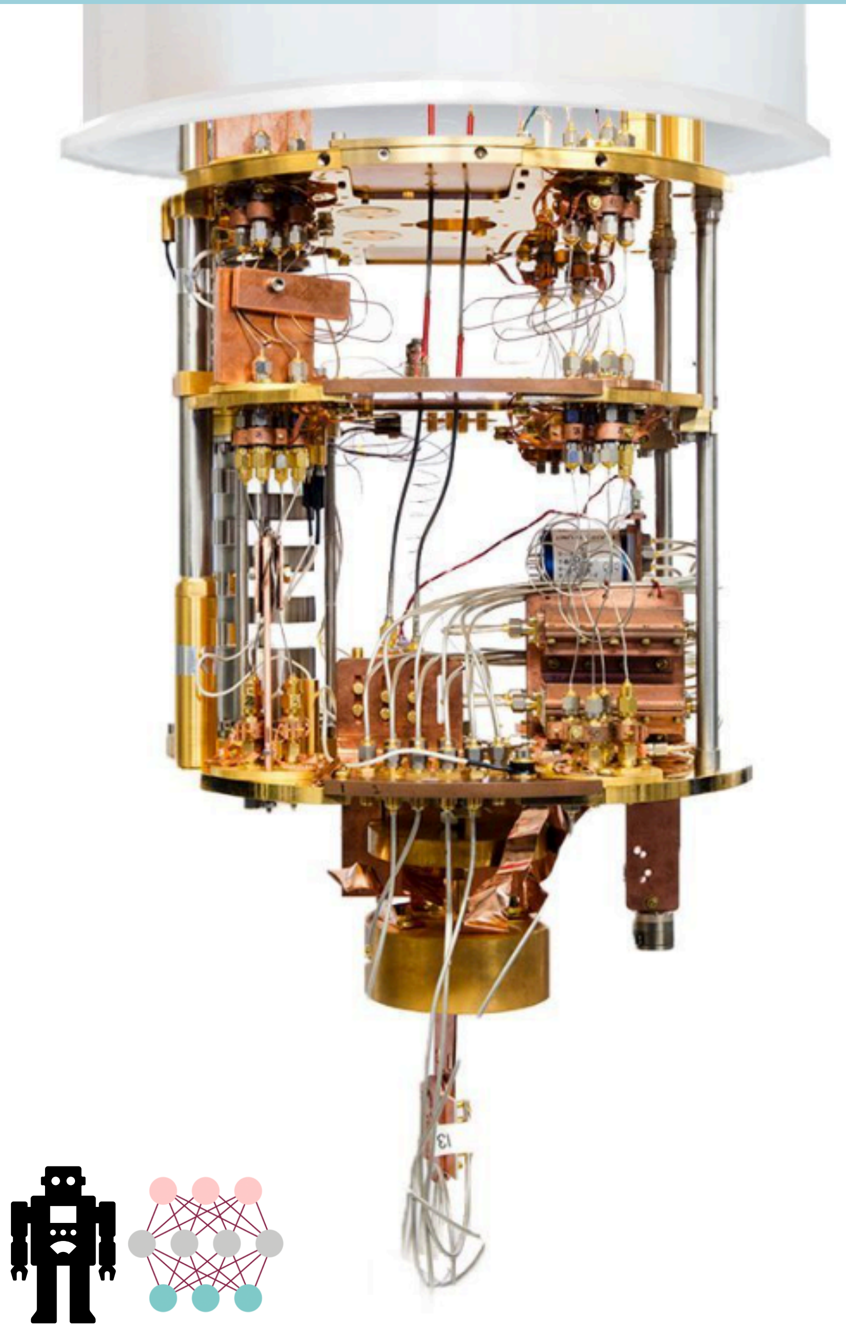


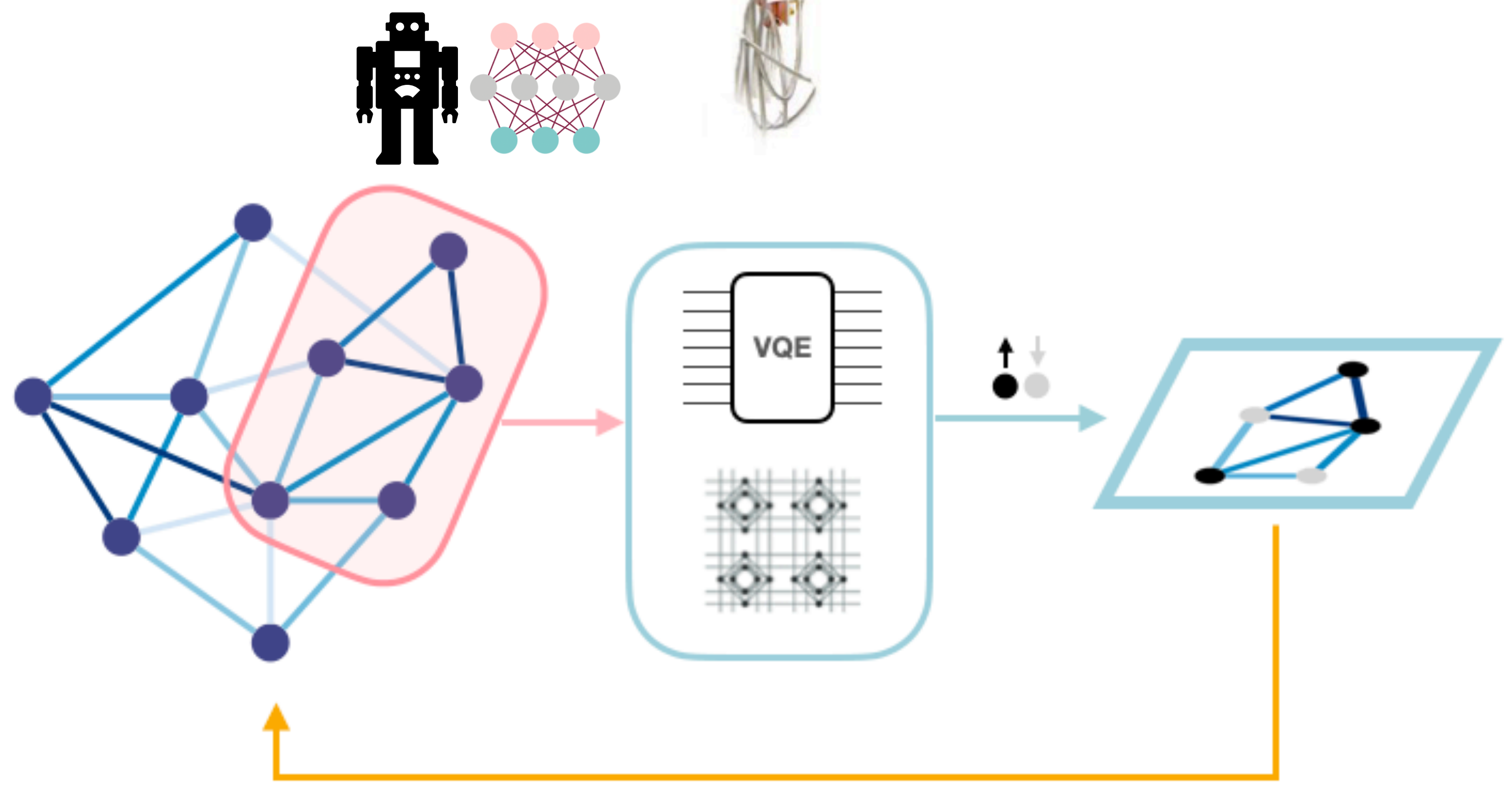
Reinforcement Learning Quantum Local Search

Team *TaipeiQC*

Outline

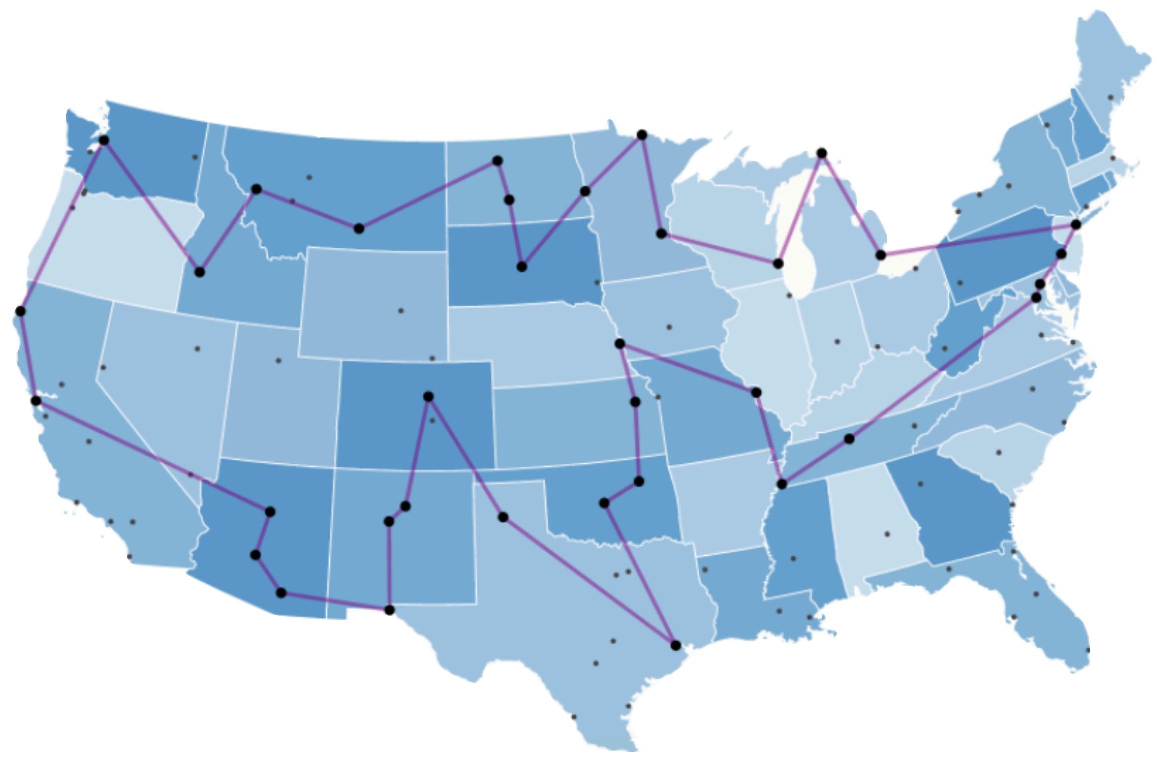
- *Combinatorial Optimization Problems*
- *Quantum Local Search*
- *Reinforcement Learning Quantum Local Search*
- *Result*
  - *Training*
  - *Testing*
  - *Functionality*
- *Summary*
- *Future Work*



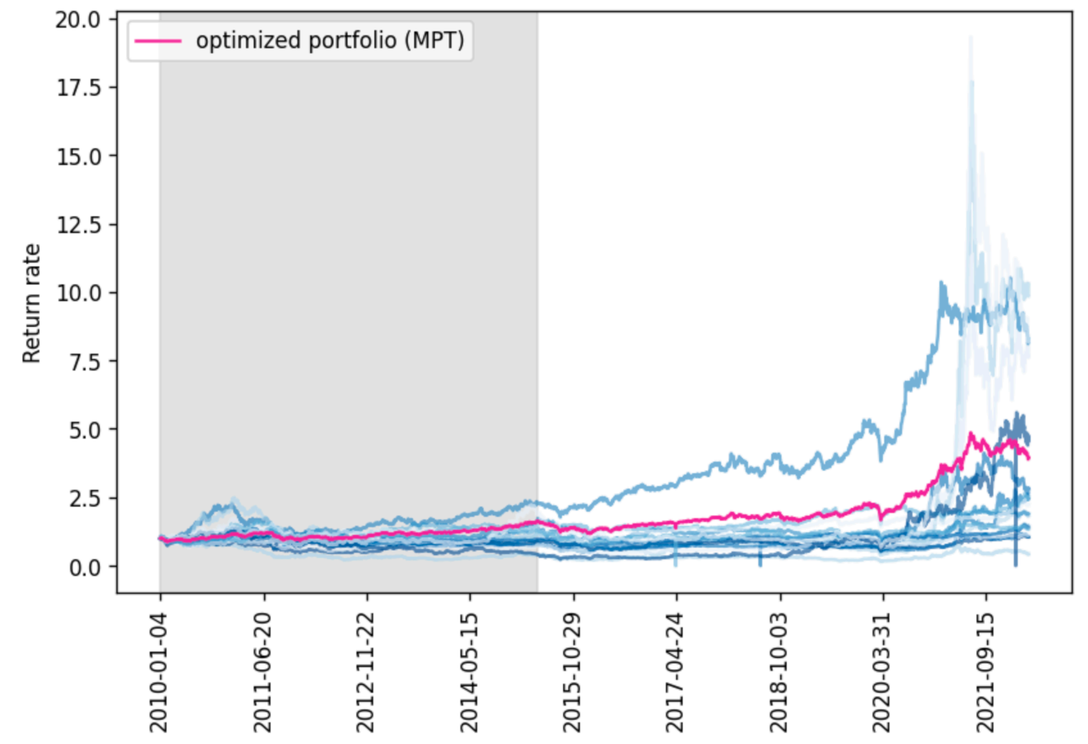


Combinatorial Optimization Problems

Combinatorial optimization problem



Traveling Salesman Problem. Source [1]



Portfolio optimization problem. Source [2]

$\rightarrow$

Quadratic unconstrained binary optimization (QUBO)

$$y(\vec{x}) = \sum_{i=1}^{N_p} w_i x_i + \sum_{i>j}^{N_p} W_{ij} x_i x_j$$

$$x_i \in \{0,1\}$$

$\rightarrow$

$$H = \sum_{i=1}^{N_p} h_i s_i + \sum_{i>j}^{N_p} J_{ij} s_i s_j$$

Exact diagonalization
Matrix product state
NN quantum state
Simulated Annealing
Variational Quantum Eigensolver (VQE)
Quantum Annealing (QA)
⋮

recent years

Cloud quantum computing platform

IBM **Quantum**

Amazon Braket

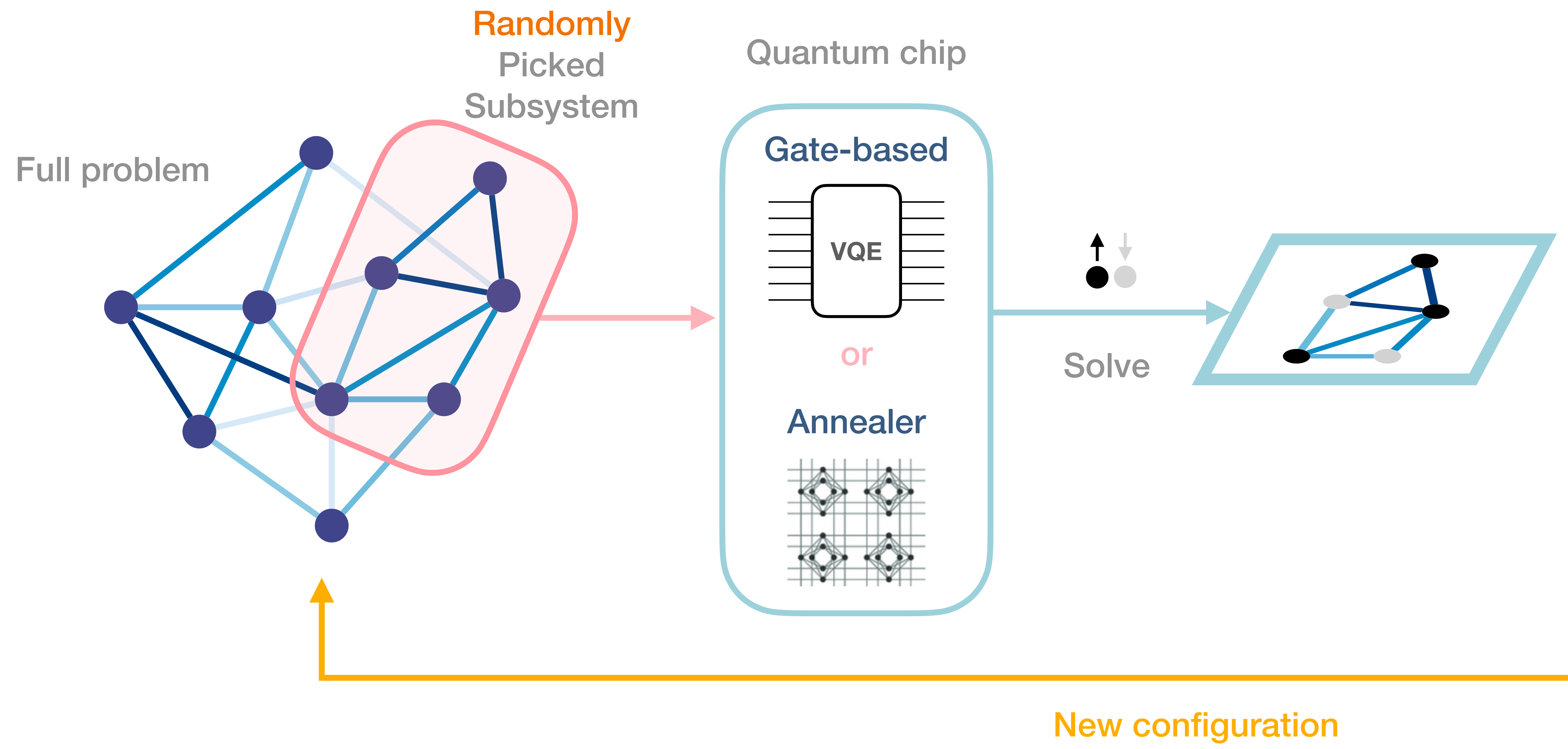
[1]: https://github.com/ildoonet/simulated-annealing-for-tsp
[2]: Harry Markowitz, The Journal of Finance 7.1 pp. 77-91 (1952)

Quantum Local Search

Given a random initial configuration of the combinatorial optimization problem, iteratively update the local configurations by quantum algorithms until the cost function converges.

In the sense that quantum hardware with few qubits are capable for solving large-size combinatorial optimization problem.

Iterate over N_{iter} iterations :



$$\Delta E = E_{\mathbf{new}} - E_{\mathbf{old}}$$

- If $\Delta E < 0$:
  - **Accept** the update, since it gives the lower cost function value.
- Else if $\Delta E \geq 0$:
  - **Accept** the update with the probability $\exp(-\gamma \Delta E)$, where γ is a hyperparameter.
  - **Reject** otherwise.

Some details (e.g. the accept mechanism) are different from its original paper : Adv. Quantum Technol., 2: 1900029.

Reinforcement Learning Quantum Local Search

Reinforcement Learning the subsystem to chose, given the problem graph and current configuration of combinatorial optimization problem.

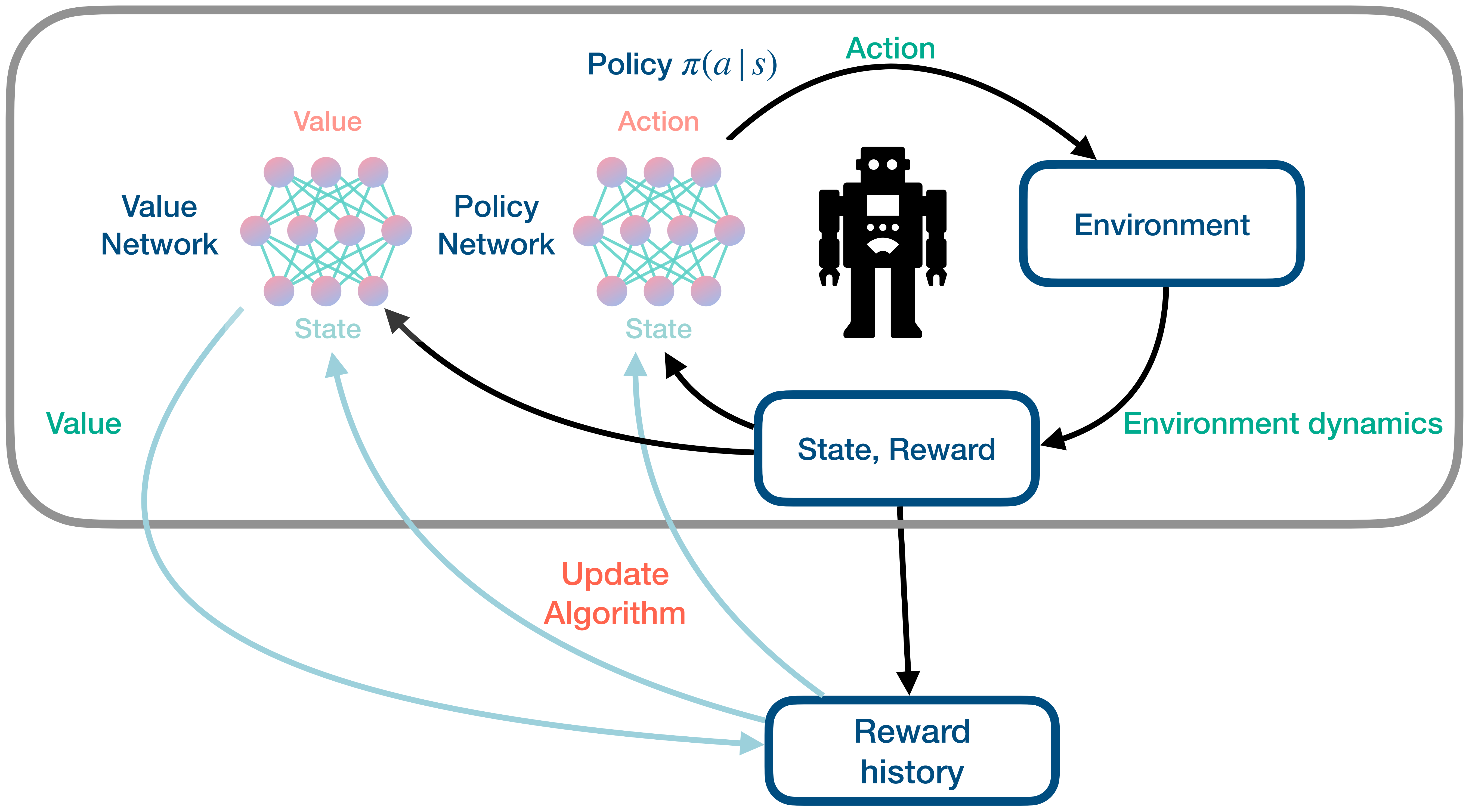
Learn the subsystem choosing strategies with lower cost function value

Key ingredient of Reinforcement Learning :

Observation (state) :
Problem graph, historical configuration, historical action

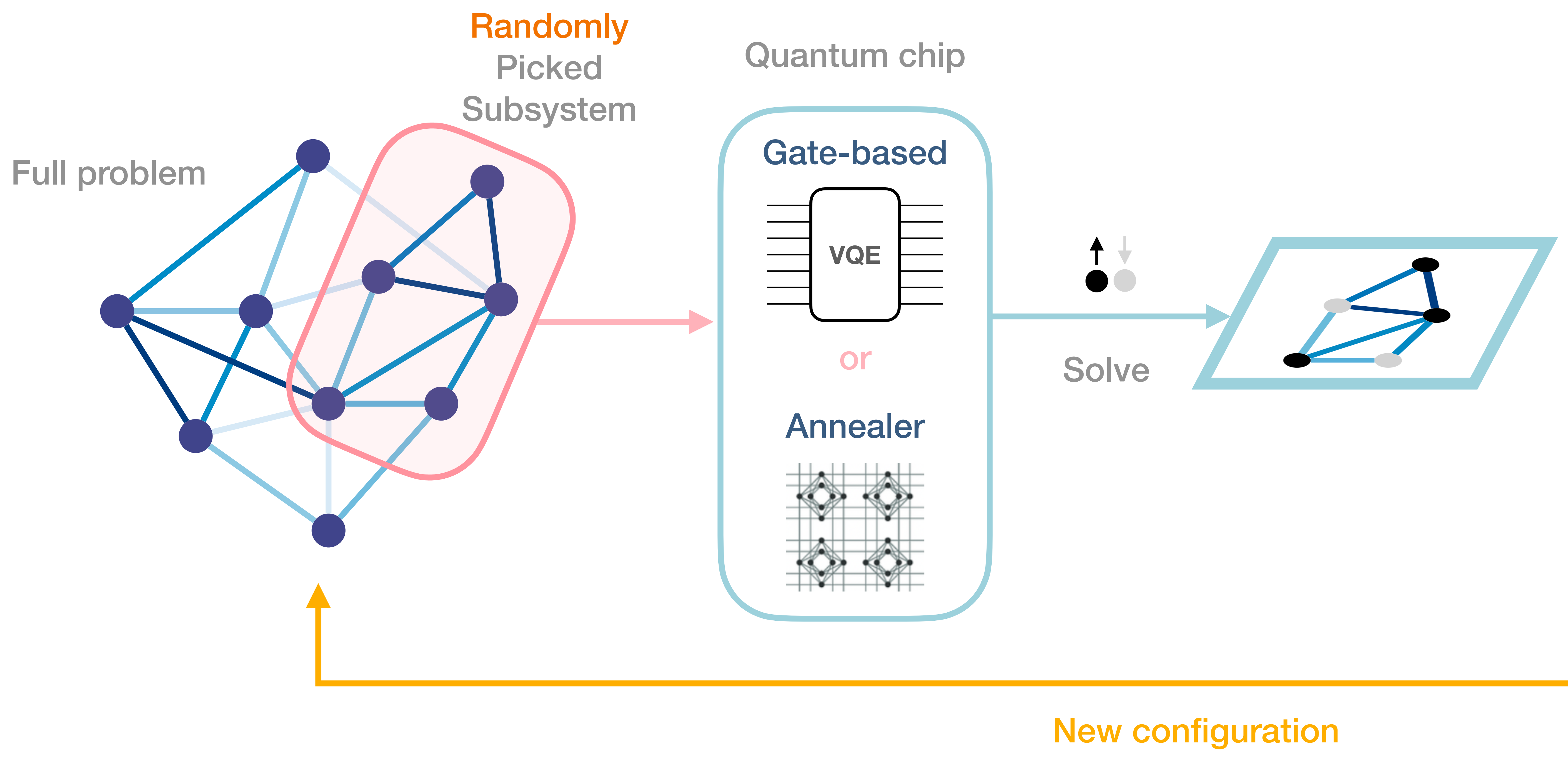
Action :
Outputted subsystem indices from policy network

Reward :
Negative cost function value or approximation ratio of the full Hamiltonian



Reinforcement Learning Quantum Local Search

Iterate over N_{iter} iterations :



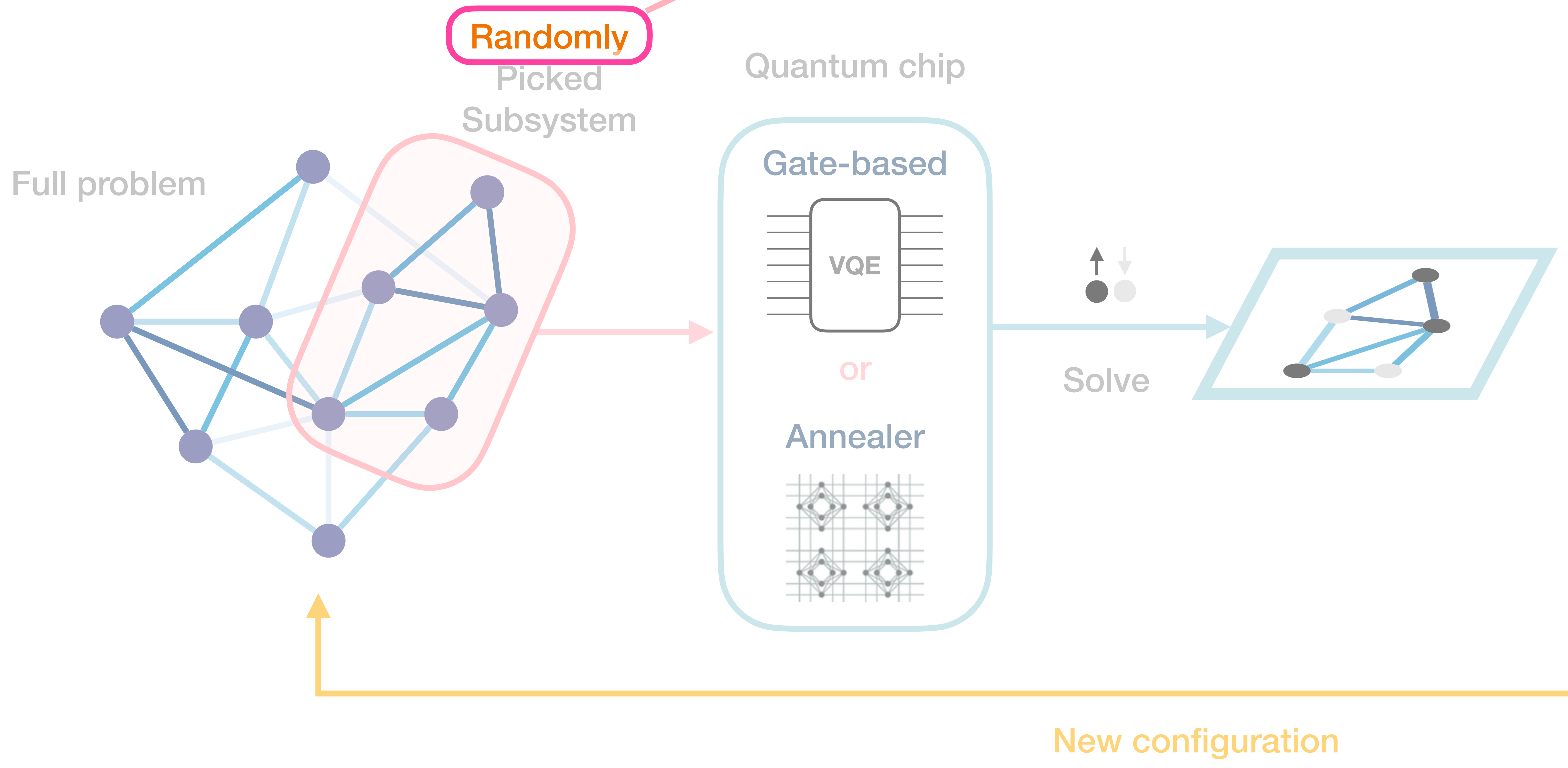
$$\Delta E = E_{\mathbf{new}} - E_{\mathbf{old}}$$

- If $\Delta E < 0$:
  - **Accept** the update, since it gives the lower cost function value.
- Else if $\Delta E \geq 0$:
  - **Accept** the update with the probability $\exp(-\gamma \Delta E)$, where γ is a hyperparameter.
  - **Reject** otherwise.

Reinforcement Learning Quantum Local Search

Iterate over N_{iter} iterations :

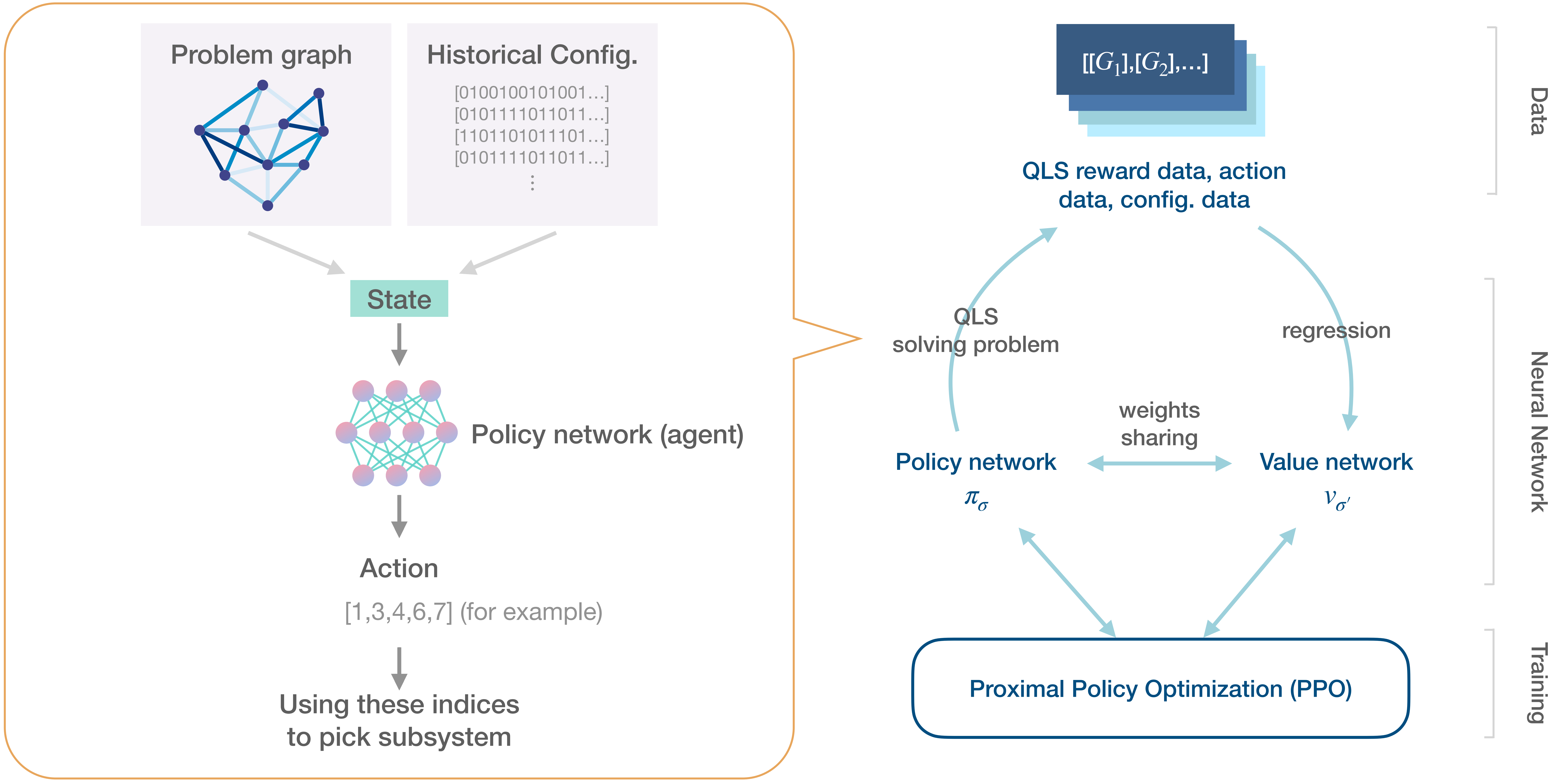
This random process can be improved by strategy found by reinforcement learning agent !



$\Delta E = E_{\text{new}} - E_{\text{old}}$

- If $\Delta E < 0$:
  - Accept the update, since it gives the lower cost function value.
- Else if $\Delta E \geq 0$:
  - Accept the update with the probability $\exp(-\gamma \Delta E)$, where γ is a hyperparameter.
  - Reject otherwise.

Reinforcement Learning Quantum Local Search



Besides PPO, we can also use other RL algorithms such as IMPALA, DQN, …

Result - Training

The fully-connected random Ising problems (corresponding to some combinatorial optimization problems in QUBO form) :

$$H = \sum_{i=1}^{N_p} h_i s_i + \sum_{i>j}^{N_p} J_{ij} s_i s_j, \ h_i \in (-1,1), \ J_{ij} \in (-1,1)$$

3 different RL agent models are trained for the following schemes :

- 32-variables combinatorial (Ising) problem with 5-variables solver.
- 64-variables combinatorial (Ising) problem with 5-variables solver.
- 128-variables combinatorial (Ising) problem with 5-variables solver.

N_g- variables solver means that the size of the subsystem is N_g, which is possible to use the size-N_g quantum solver. For example, VQE or QAOA with N_g qubits, Quantum Annealing with N_g qubits.

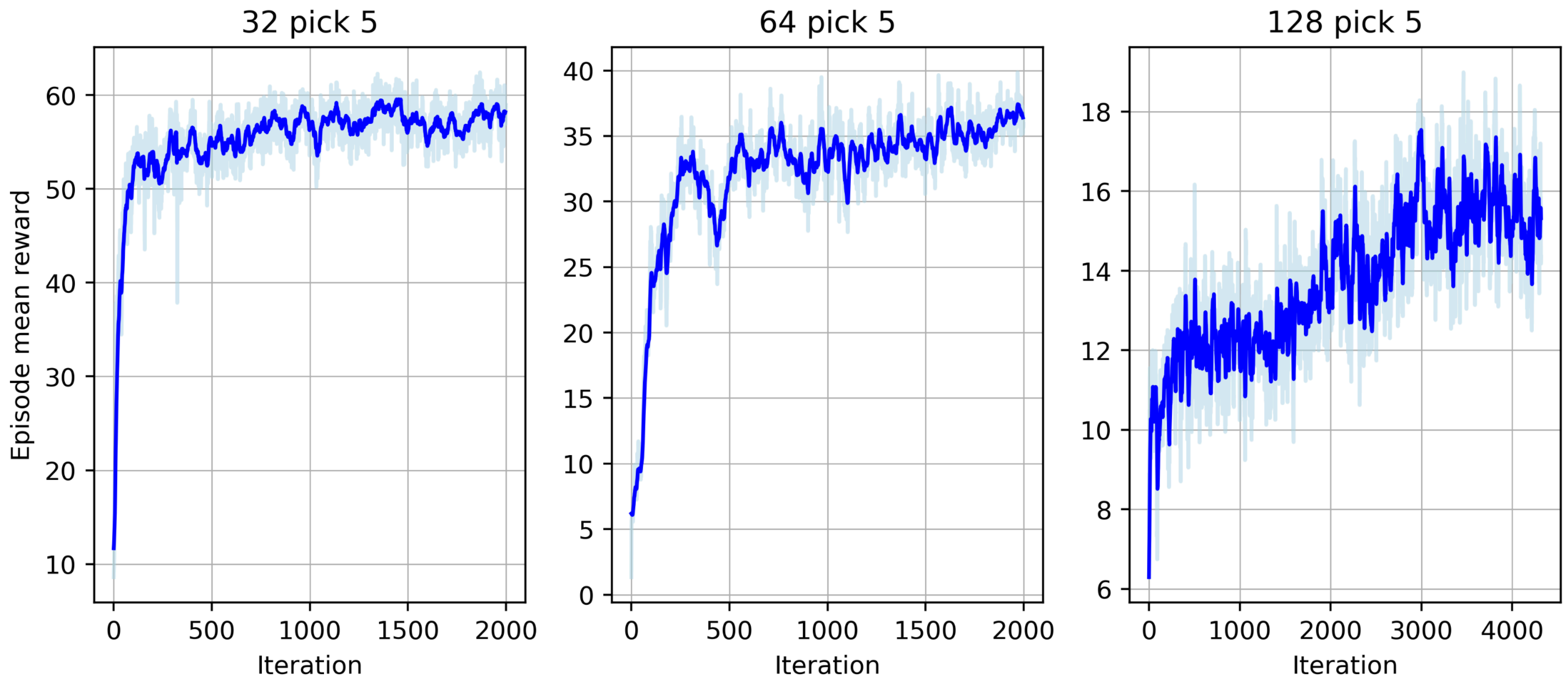
Training Algorithm : Importance Weighted Actor-Learner Architectures (IMPALA)

- For each training iteration, roughly 100 episodes are evaluated.
- For each episode, 200 steps of Local Search iterations are evaluated to generate reward data.
- 1000 Ising Hamiltonians (combinatorial optimization problem) are generated as training data, such that for each episode, the agent will start from one of the Hamiltonians. This setting allows the RL agent to have a more general sense of solving different Ising problems (combinatorial optimization problems).

Result - Training

Figure : Episode mean reward vs. training iteration

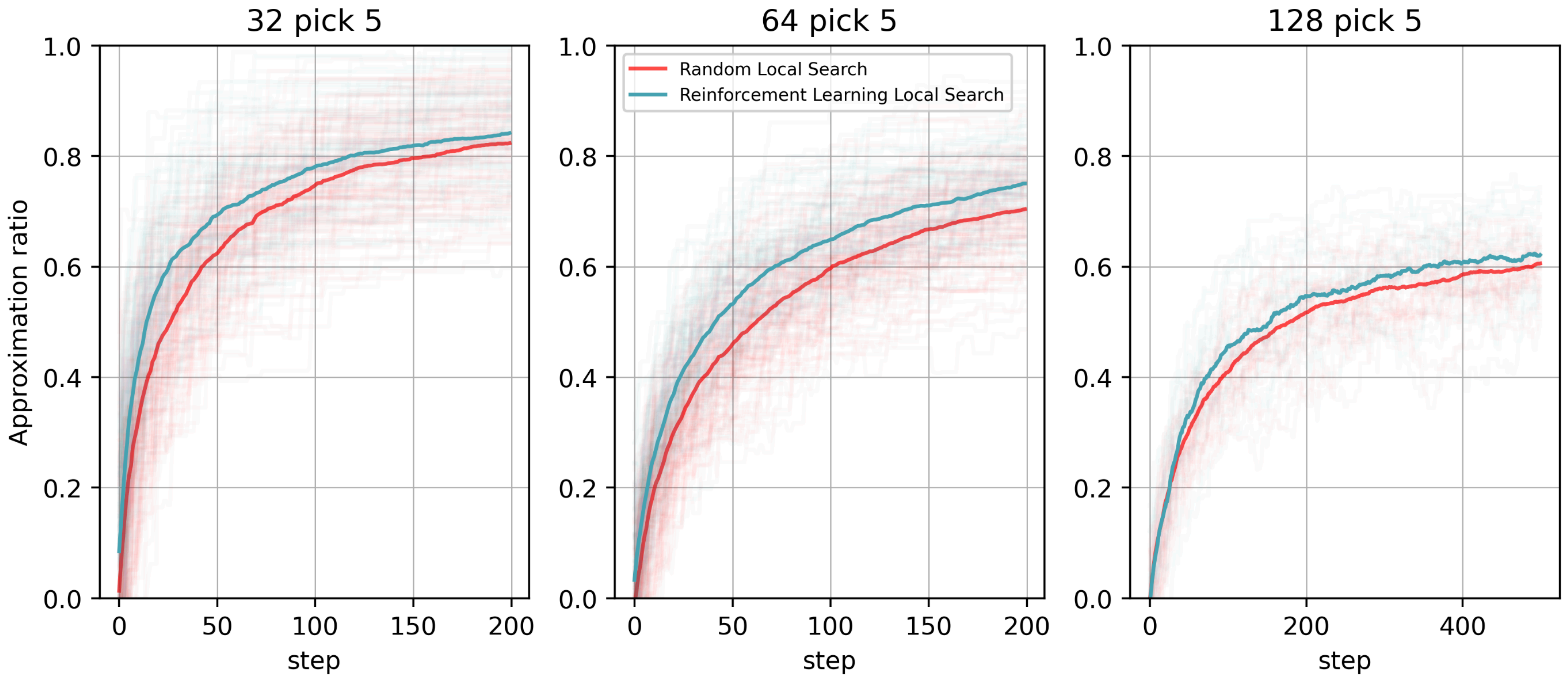
- 32-variables combinatorial (Ising) problem with 5-variables solver.
- 64-variables combinatorial (Ising) problem with 5-variables solver.
- 128-variables combinatorial (Ising) problem with 5-variables solver.



Result - Testing

Figure : Approximation ratio vs. Search step

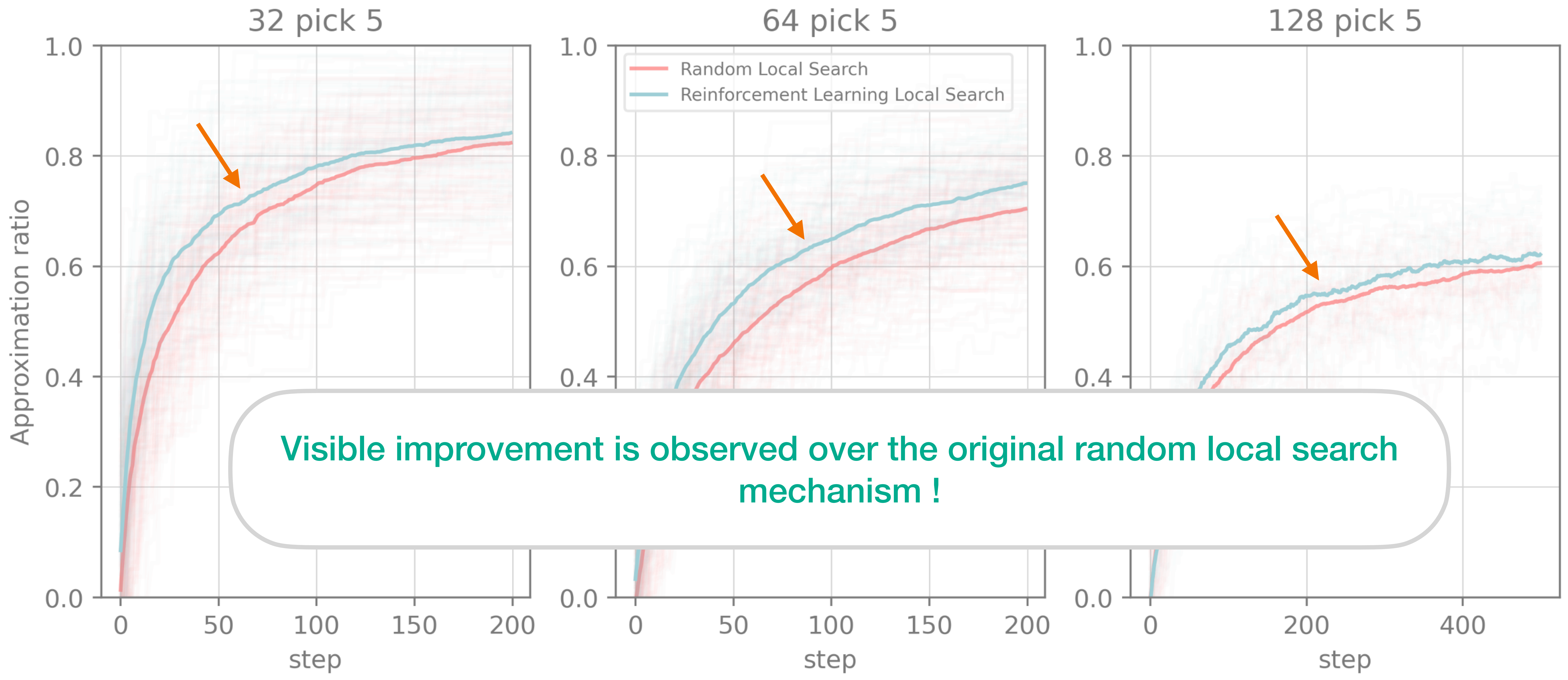
100 fully-connected random Ising problems are generated as the testing data for “32 pick 5” and “64 pick 5” cases, 30 fully-connected random Ising problems are generated as the testing data for “128 pick 5” case. For each problem, both random local search and reinforcement learning local search are applied to search for high approximation ratio solution configuration of the problem.



Result - Testing

Figure : Approximation ratio vs. Search step

100 fully-connected random Ising problems are generated as the testing data for “32 pick 5” and “64 pick 5” cases, 30 fully-connected random Ising problems are generated as the testing data for “128 pick 5” case. For each problem, both random local search and reinforcement learning local search are applied to search for high approximation ratio solution configuration of the problem.

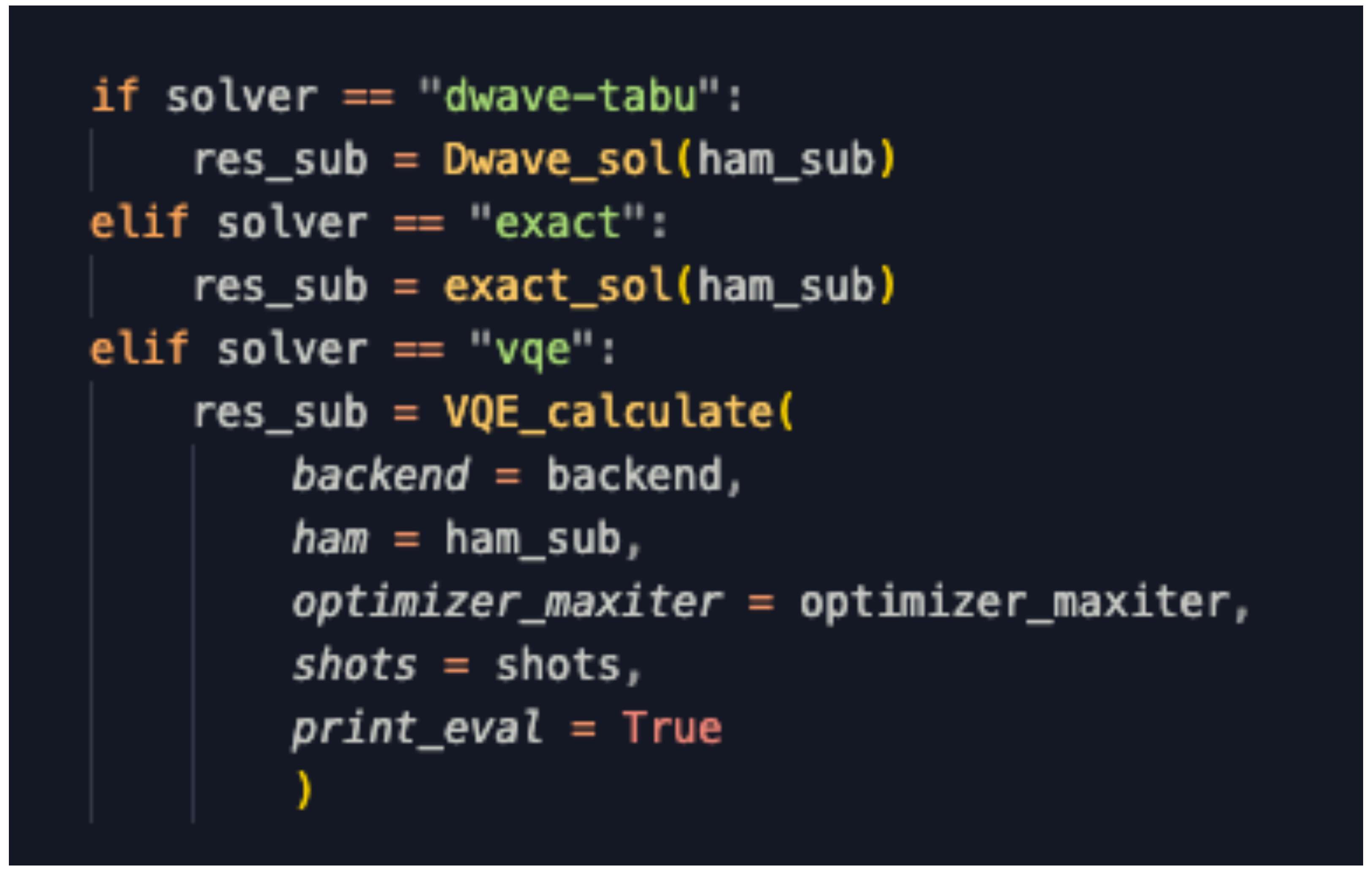


Visible improvement is observed over the original random local search mechanism !

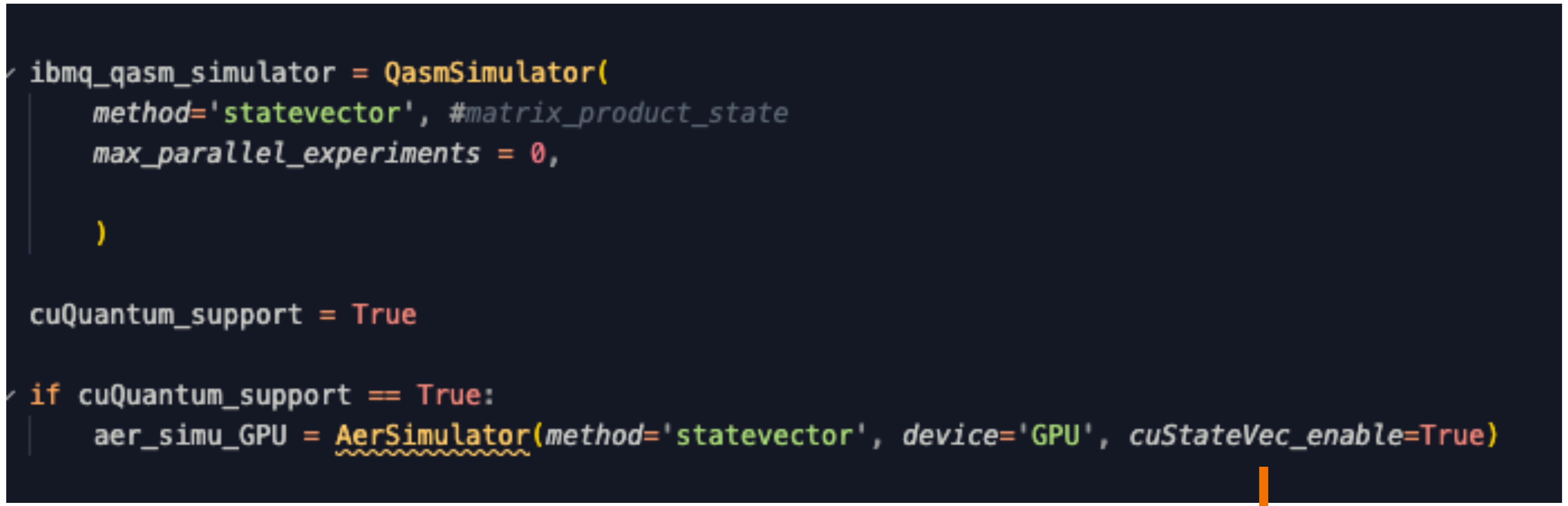
Result - Functionality

Supported solver for subsystems:

Gate-based VQE simulator (cuQuantum, qiskit), DWave tabu solver, exact solver



```python
if solver == "dwave-tabu":
    res_sub = Dwave_sol(ham_sub)
elif solver == "exact":
    res_sub = exact_sol(ham_sub)
elif solver == "vqe":
    res_sub = VQE_calculate(
        backend = backend,
        ham = ham_sub,
        optimizer_maxiter = optimizer_maxiter,
        shots = shots,
        print_eval = True
        )
```



```python
ibmq_qasm_simulator = QasmSimulator(
    method='statevector', #matrix_product_state
    max_parallel_experiments = 0,

    )

cuQuantum_support = True

if cuQuantum_support == True:
    aer_simu_GPU = AerSimulator(method='statevector', device='GPU', cuStateVec_enable=True)
```

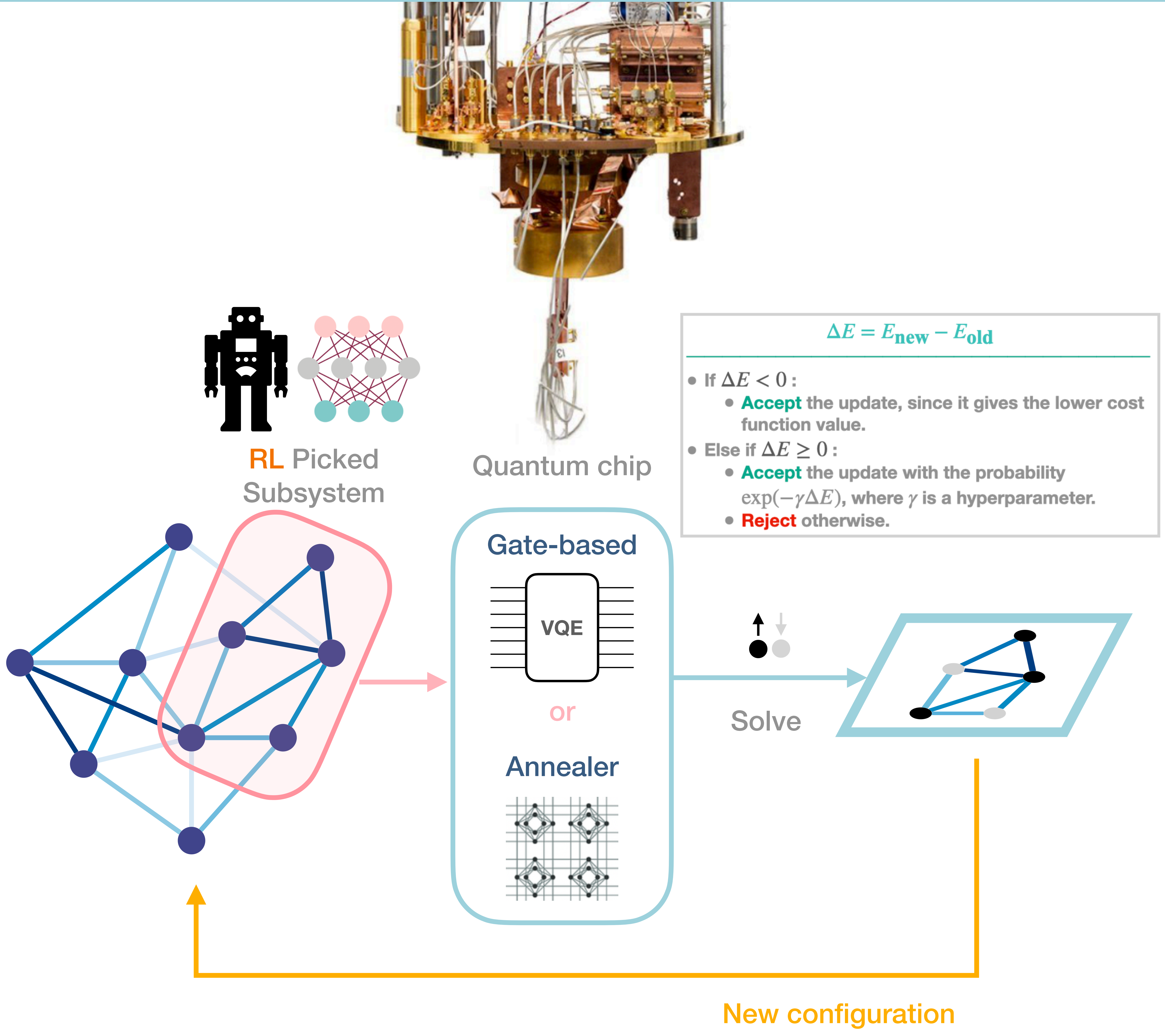
To use this, qiskit-aer must be built from source, with cuQuantum support.

Summary - *Reinforcement Learning Quantum Local Search*

Tackling the problem with size much larger than our current hardware

- **General framework across different classical/ quantum computing hardware.**
  The quantum solvers are utilized as the subsystem solver, and the choice is flexible, including both CPU and GPU simulation of quantum circuit.

- **General consideration of Ising problems**
  Instead of applicability of only one Ising problem that same as training data,
  after training the RL agent, the agent is able to tackle the unseen fully-connected random Ising problems with improvement over the original random local search mechanism.



$\Delta E = E_{\text{new}} - E_{\text{old}}$

- If $\Delta E < 0$:
  - **Accept** the update, since it gives the lower cost function value.
- Else if $\Delta E \geq 0$:
  - **Accept** the update with the probability $\exp(-\gamma \Delta E)$, where γ is a hyperparameter.
  - **Reject** otherwise.

Future Work

How can the RL-QLS idea to be improved ?

- **Increase the subsystem size to be solved**

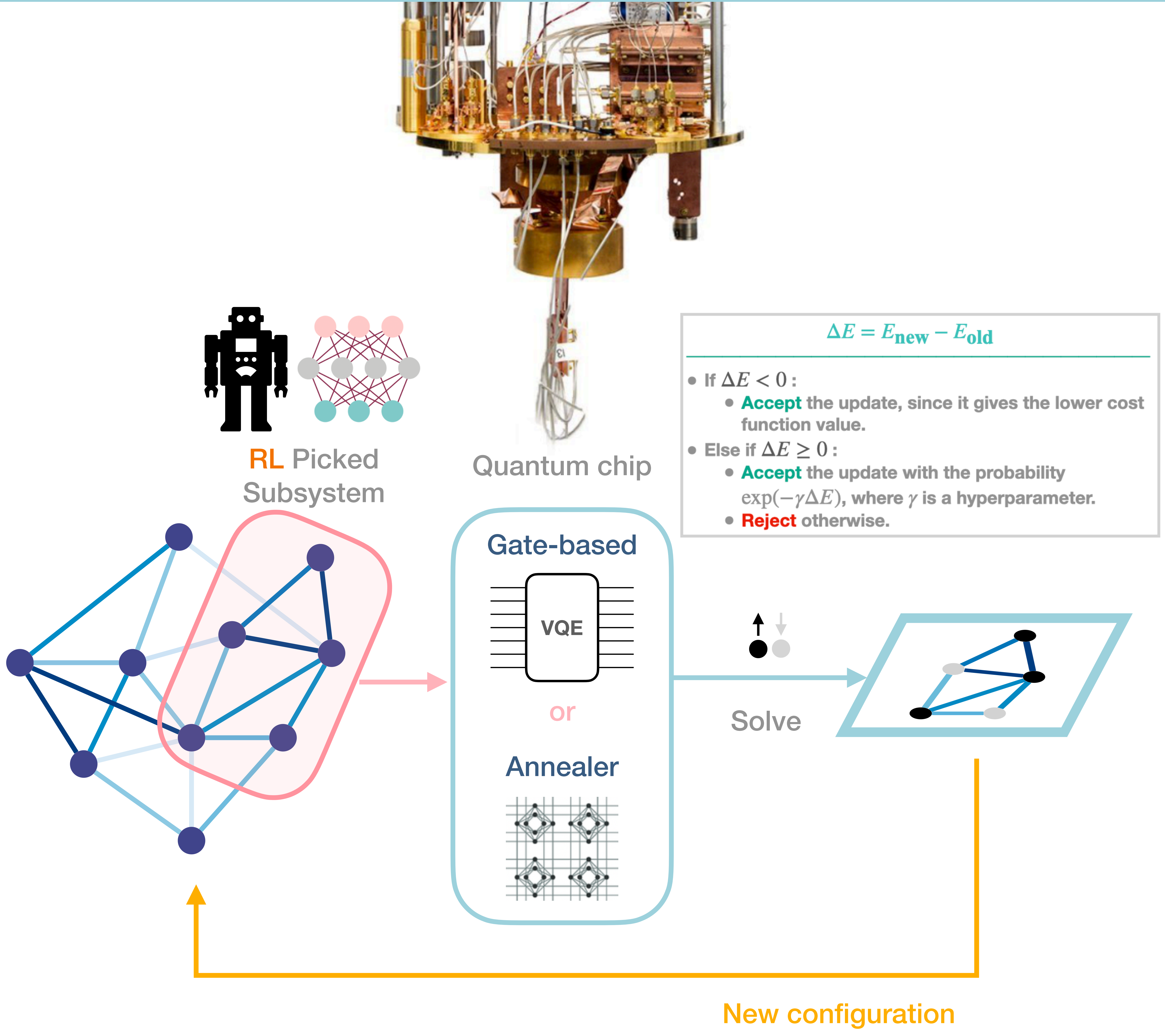
  For now, the subsystem size is 5, which could be scaled up in the future, with possible improvement of RL-QLS process, since the RL agent handles larger sub-problem at a time.

- **Applicability of specific problem**

  Can we train on fully-connected random graph and apply the RL agent to the regular-N graph combinatorial optimization problems ? or addition training is required ?

- **Transferability of different models**

  For now, each problem size and sub-problem size requires different RL model, can these models share some common "knowledge" and reduce the required training time ?



$\Delta E = E_{\text{new}} - E_{\text{old}}$

- If $\Delta E < 0$:
  - **Accept** the update, since it gives the lower cost function value.
- Else if $\Delta E \geq 0$:
  - **Accept** the update with the probability $\exp(-\gamma \Delta E)$, where γ is a hyperparameter.
  - **Reject** otherwise.