

Opti-maze

Playing with QAOA

Nikita Astrakhantsev
Vasiliy Bokov
Alberto Colombo
Reinis Irmejs
Tom Westerhout

Goals

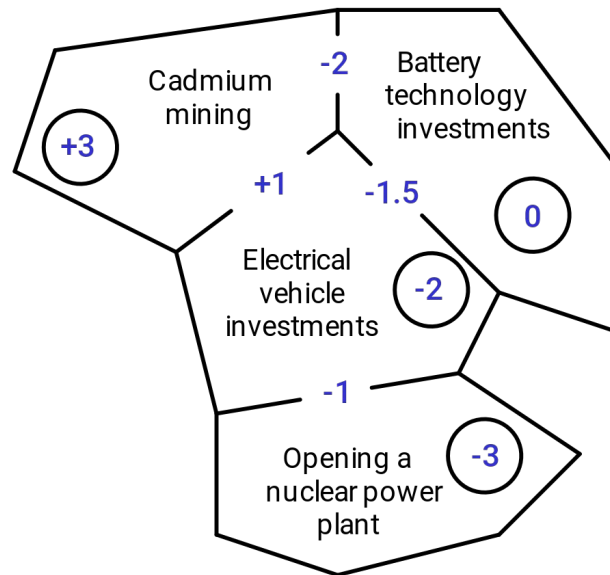
- Popularize quantum computing by teaching about its capabilities and limitations;
 - focus on combinatorial optimization and QAOA;
 - create a game that uses it;
- Apply quantum computing to real-world problems;
 - focus on climate change
 - efficient resource allocation
- Develop a framework that could aid with complicated optimization problems in the future

Premise of the game:

- In many popular games like Monopoly and Sid Mayer's Civilization in which players have to find optimal solution to resource allocation.
- In Opti-maze the goal of the player is to **perform optimization better** than the *Quantum Artificial Player* who is using the **QAOA**

Example optimization problem

- Each investment has a cost (either positive or negative);
- Some combinations of investments may lead to synergy and lower the overall energy;
- The problem is mapped to a classical Ising-like model (no higher-order terms to reduce the circuit depth);
- The problem is NP-hard in general;
- Any two regions can interact – maps well to IonQ's topology;

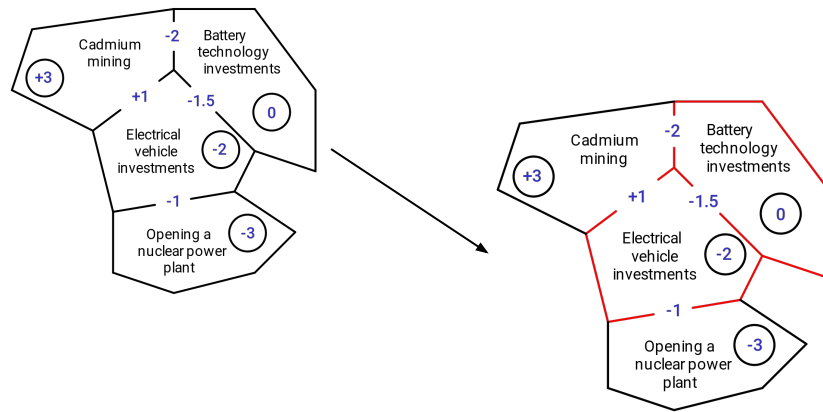


$$H = \sum_{i,j} J_{ij} S_i S_j + \sum_i h_i S_i$$

Approach: Opti-maze, a game

General idea: try to minimize the carbon emissions by selecting areas in which to invest

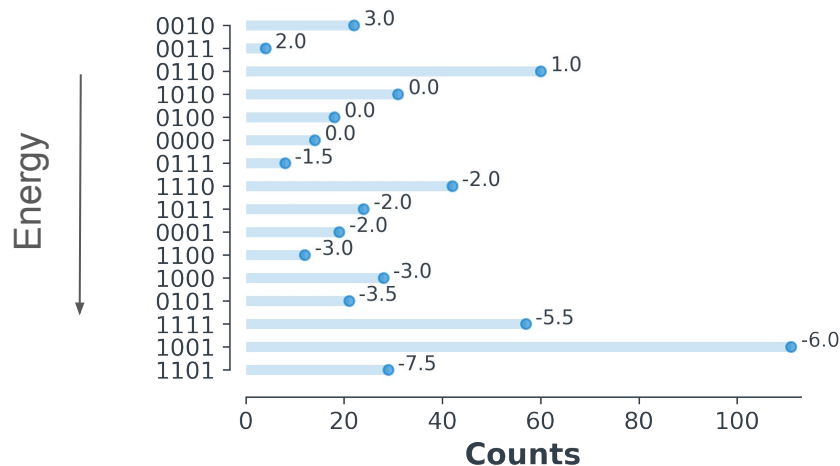
- Two modes:
 - Standard: compete against QAOA
 - Cooperative: use QAOA as a tool



Standard mode: can you beat IonQ?

- Player tries to obtain a lower energy than QAOA;
- Number of shots in QAOA defines the difficulty level;

Basis state



```
1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
```

Game mode: 1) standard, 2) cooperative:

1

1

1

1

1

1

1

1

1

```
1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
5 # select standard regime
```

Game mode: 1) standard, 2) cooperative:


```
1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
5 # difficulty level = number of samples used to train and later sample from the QAUA circuit
```

Game mode: 1) standard, 2) cooperative: 1

Difficulty level (between 1 and 500):

1

1

1

1

1

1

1

1

```

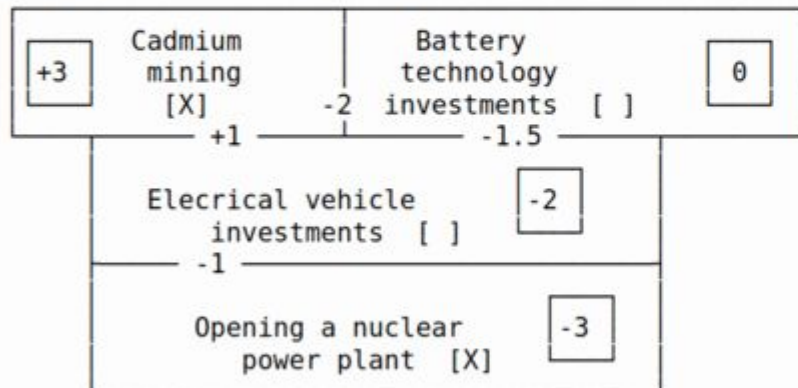
1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
5 # the game immediately shows QA0A energy of -7.5 found by the QA0A
6 # a player starts with a random string and can invert selected bits one by one

```

Game mode: 1) standard, 2) cooperative: 1

Difficulty level (between 1 and 500): 250

Running QA0A with 250 shots ...



Current energy: 0.0 (current state: [1 0 0 1])

QA0A energy: -7.5

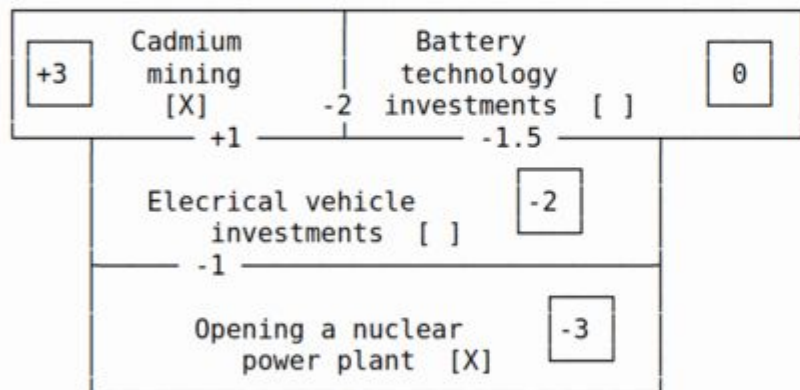
Specify the index of a component to select/deselect:

```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # let us flip bit 0

```

Game mode: 1) standard, 2) cooperative: 1
 Difficulty level (between 1 and 500): 250
 Running QAOA with 250 shots ...



Current energy: 0.0 (current state: [1 0 0 1])
 QAOA energy: -7.5

Specify the index of a component to select/deselect:

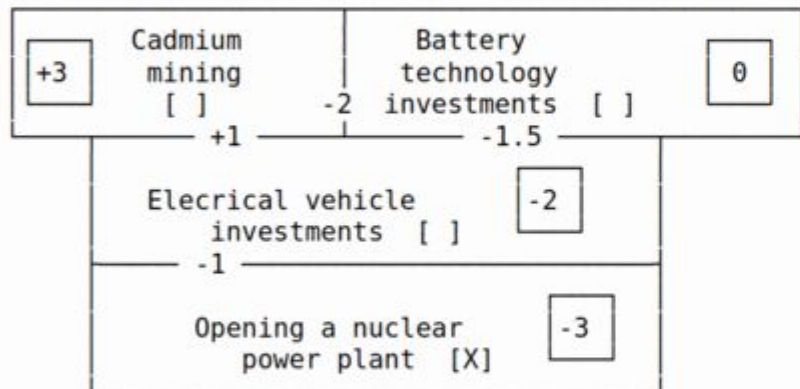
```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # energy decreased from 0 to -3

```

Current energy: -3.0 (current state: [0 0 0 1])
 QAOA energy: -7.5

Specify the index of a component to select/deselect: 0



Current energy: -3.0 (current state: [0 0 0 1])

QAOA energy: -7.5

Specify the index of a component to select/deselect:

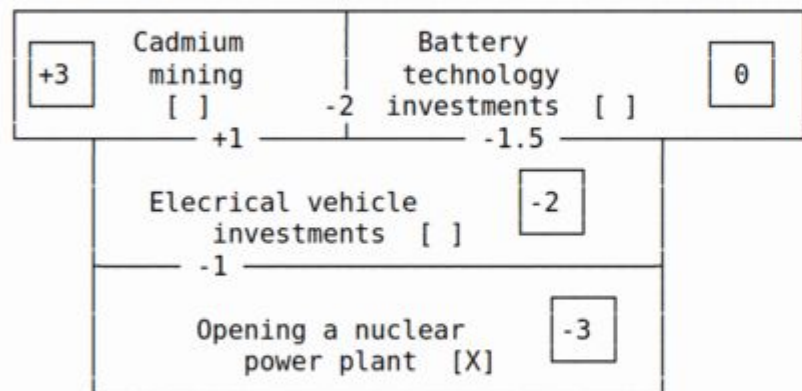
```

1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
5 # let us flip spin 2

```

QA0A energy: -7.5

Specify the index of a component to select/deselect: 0



Current energy: -3.0 (current state: [0 0 0 1])

QA0A energy: -7.5

Specify the index of a component to select/deselect: 2

```

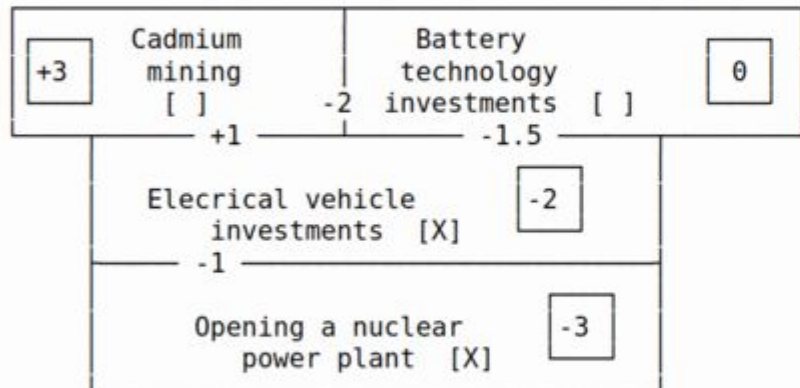
1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # energy decreased from -3 to -6

```

Current energy: -7.5 (current state: [0 0 1 1])

QAOA energy: -7.5

Specify the index of a component to select/deselect: 2



Current energy: -6.0 (current state: [0 0 1 1])

QAOA energy: -7.5

Specify the index of a component to select/deselect:

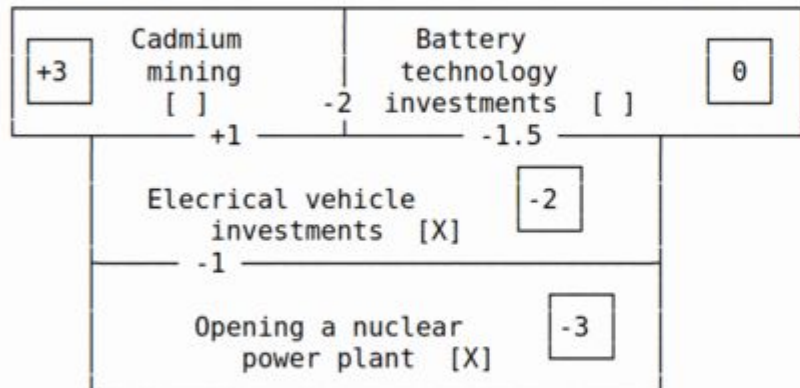
```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # let us flip 0

```

QAOA energy: -7.5

Specify the index of a component to select/deselect: 2



Current energy: -6.0 (current state: [0 0 1 1])

QAOA energy: -7.5

Specify the index of a component to select/deselect:

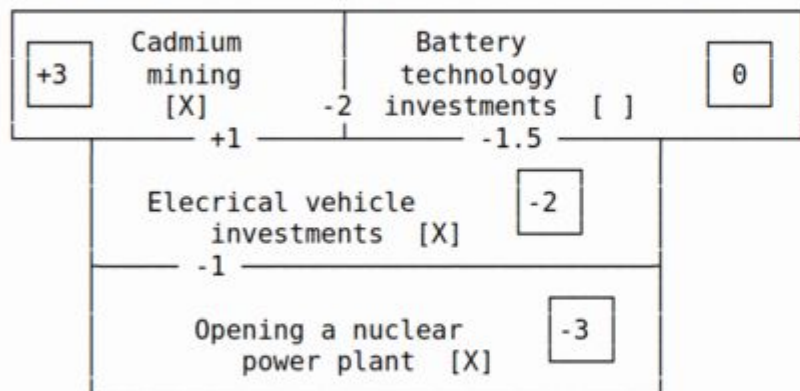
```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # energy grew from -3 to -2

```

Current energy: -2.0 (current state: [1 0 1 1])
 QAOA energy: -7.5

Specify the index of a component to select/deselect: 0



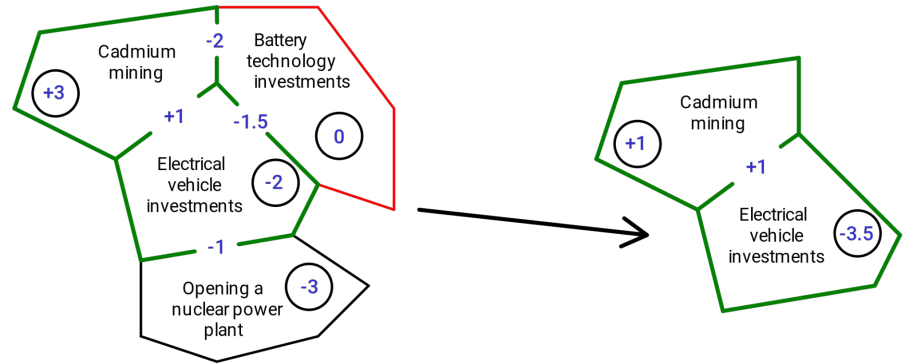
Current energy: -2.0 (current state: [1 0 1 1])

QAOA energy: -7.5

Specify the index of a component to select/deselect:

Cooperation mode: QAOA to the rescue!

- The user can still select/unselect regions of investment;
- No QAOA solution to compete against;
- QAOA can be used to solve subproblems;
- Applicable to cases when the whole graph does not fit into a quantum computer;



$$H = \sum_{i,j} J_{ij} S_i S_j + \sum_i h_i S_i$$

$$H' = \sum_{i,j \in \mathcal{G}} J_{ij} S_i S_j + \sum_{i \in \mathcal{G}} \left(h_i + \sum_{j \notin \mathcal{G}} J_{ij} S_j \right) \cdot S_i$$

```
1 import main_loop
2 import QA0A
3
4 main_loop.main(qaoa_solve=QA0A.qaoa_solve)
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # select "cooperative" regime in which QAOA helps player find the best solution
6 # namely, user specifies subgraph as set of indices and QAOA applies its best solution on this subgraph
```

Game mode: 1) standard, 2) cooperative:

1

1

1

1

1

1

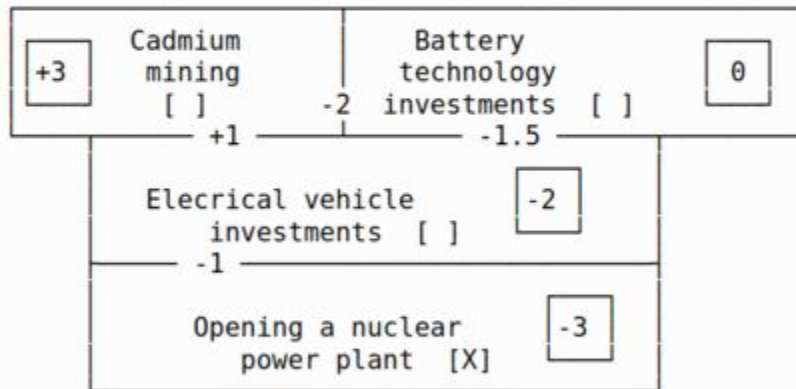
1

```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # we start with current energy of -3 and current state
6 # we need to provide a subcluster that QAOA will solve for us

```

Game mode: 1) standard, 2) cooperative: 2



Current energy: -3.0 (current state: [0 0 0 1])

Specify the index of a component to select/deselect: solve 0,2

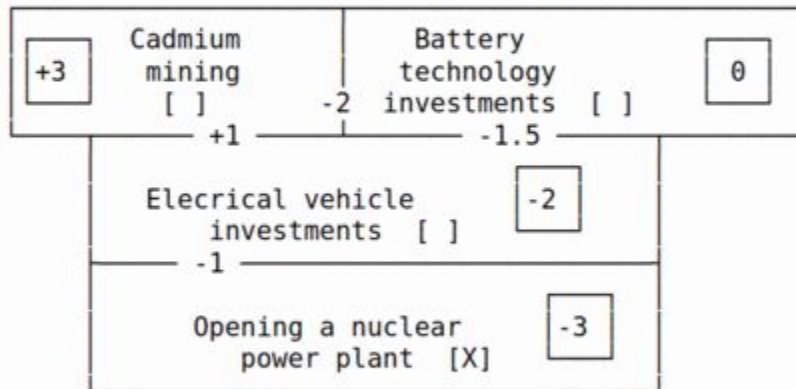
1

```

1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # we try to solve the subgraph of spins 0, 2

```

Game mode: 1) standard, 2) cooperative: 2



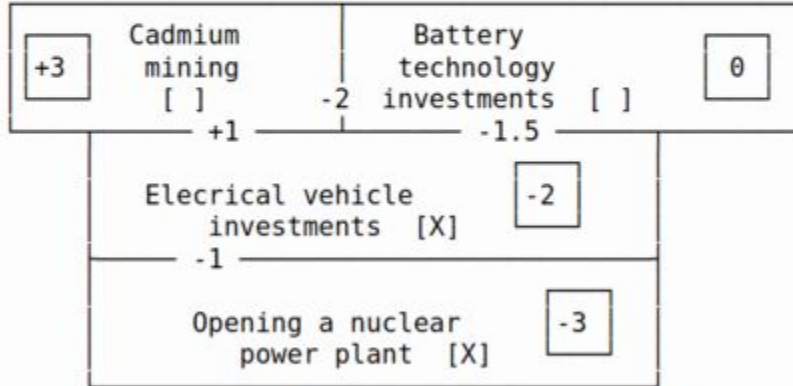
Current energy: -3.0 (current state: [0 0 0 1])

Specify the index of a component to select/deselect:

1

```
1 import main_loop
2 import QAOA
3
4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
5 # state changes and energy decreased to -6
```

(Current energy: -3.0 (current state: [0 0 0 1])
Specify the index of a component to select/deselect: solve 0,2
Running QAOA with 100 shots ...



(Current energy: -6.0 (current state: [0 0 1 1])

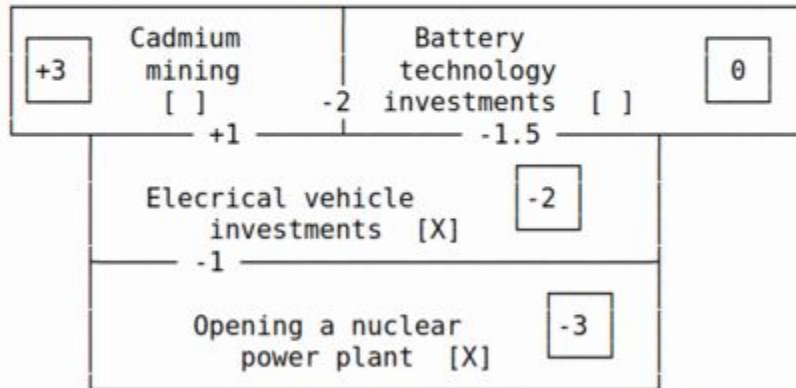
Specify the index of a component to select/deselect:

```

: 1 import main_loop
  2 import QAOA
  3
  4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
  5 # state changes and energy decreased to -6. we try to select graph 1, 3

```

Current energy: -3.0 (current state: [0 0 0 1])
 Specify the index of a component to select/deselect: solve 0,2
 Running QAOA with 100 shots ...



Current energy: -6.0 (current state: [0 0 1 1])

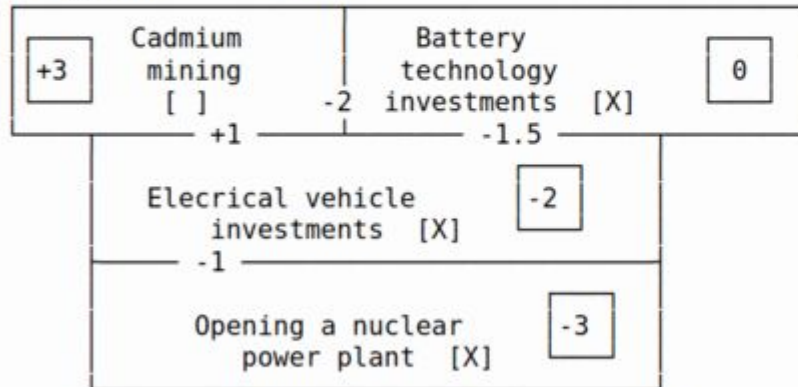
Specify the index of a component to select/deselect:

```

: 1 import main_loop
  2 import QAOA
  3
  4 main_loop.main(qaoa_solve=QAOA.qaoa_solve)
  5 # energy decreased to -7.5. The ground state energy is found!

```

Current energy: -6.0 (current state: [0 0 1 1])
 Specify the index of a component to select/deselect: solve 1, 3
 Running QAOA with 100 shots ...



Current energy: -7.5 (current state: [0 1 1 1])

Specify the index of a component to select/deselect:

Outlook

- Hamiltonian specified by a domain expert;
- Leaderboard to compete against each other in cooperative mode;
- ... technicalities like a proper GUI;