# Reinforcement Learning for Solving Russian Puzzle

Cheng-Hsin Yu(于承鑫), Huai-En Tseng(曾懷恩), Po-Yen Yu(游柏晏), and Hong-Yu Chu(曲宏宇)
Department of Physics, National Chung Cheng University, Chiayi, Taiwan

## Motivation

In recent years, the artificial intelligence has a huge progress. Computers with AI take advantage in repeated training and help us predicting the results. By memorizing successful and failed cases, artifical intelligence can make the most direct and fast judgment.

## Purpose

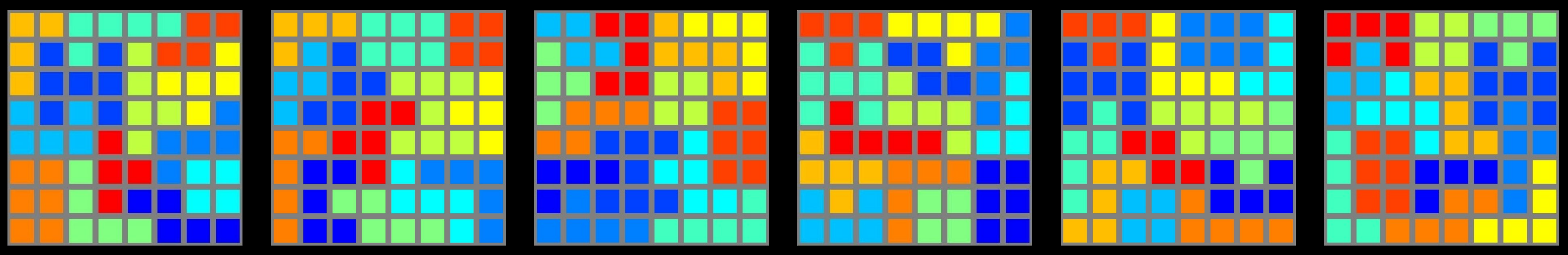
To efficiently find the solution to the Russian puzzle, we introduce machine learning to assist us. Starting from the Random method of mechanical thinking, and then introducing the Score matrix to give it the ability to select puzzles intelligently. Finally through the Q-Table method, the computer have memory can record successes and failures, thus allowing it to avoid repeated failures.

## Device

Computer :
- CPU : Intel i5-4460
- GPU : NVIDIA GeForce GTX 1050

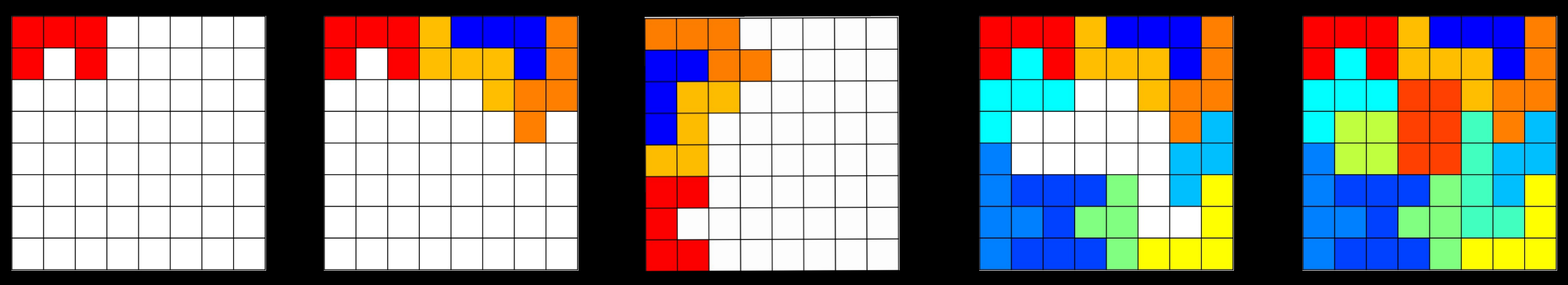
System :
- MATLAB 2019b

## Russian puzzle

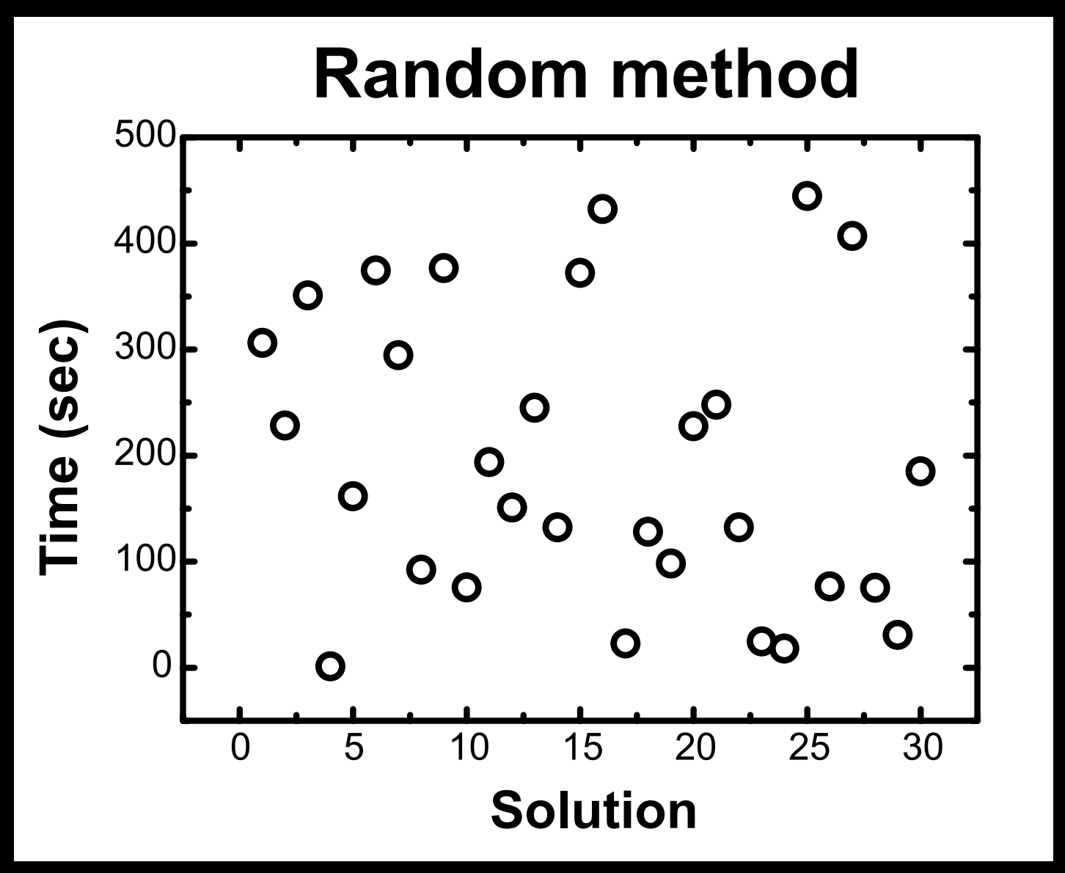


## Random

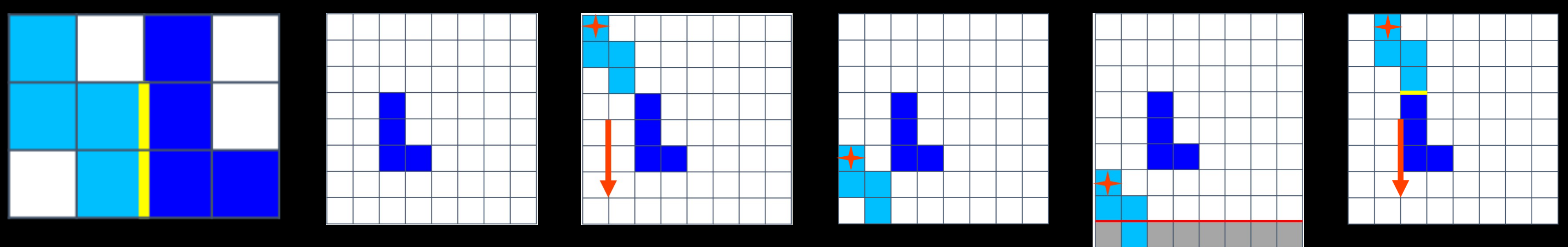
By random choose the puzzle, the rule we need to follow is avoid creating an unfillable hole and overlapping. If the selected puzzle violates the rule, we will back to previous step and randomly reselect choose another puzzle to try again.

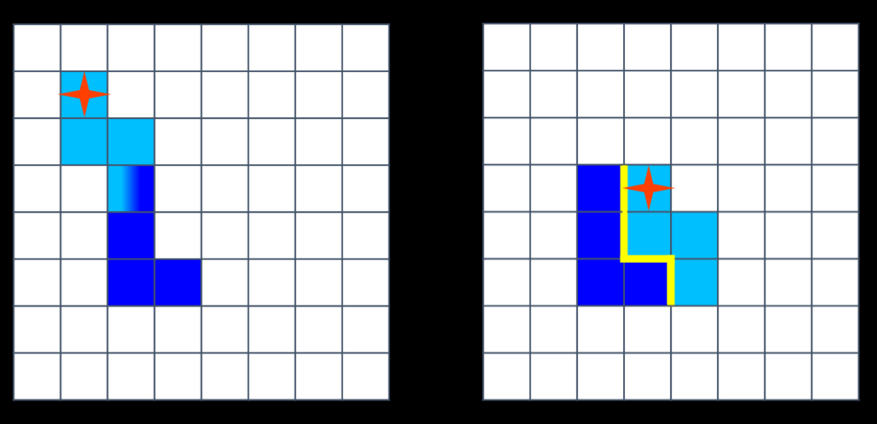

A1 A2 A3 A4 A5 A6

A1. We randomly choose one puzzle and specifies to place it from the upper-left corner.
A2. Put the puzzles in order from the upper-left to right until the top layer is filled.
A3. Rotate the plane counterclockwise and repeat A2.
A4. Repeat A3 until the outermost layers are filled.
A5. If the remaining puzzles can't fill in middle area, we'll clear the plane and back to A1.
A6. In random way, we found that it takes between 60 to 200 seconds to solve a solution and does not converge.



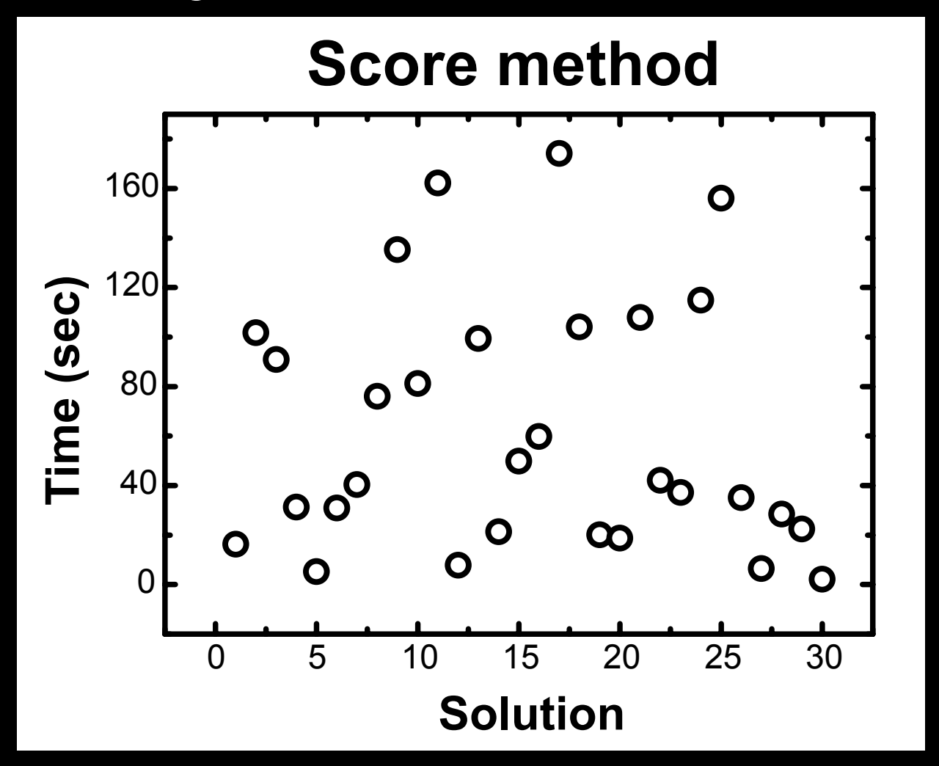
## Score


B1 B2 B3 B4 B5 B6


B7 B8

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | -100 | -100 |
| 1 | -100 | 2 | 0 | 0 | 0 | 0 | -100 |
| 2 | -100 | -100 | 1 | 0 | 0 | 0 | -100 |
| 2 | -100 | -100 | 4 | 0 | 0 | 0 | -100 |
| 1 | -100 | -100 | -100 | 1 | 0 | 0 | -100 |
| -100 | 2 | -100 | -100 | 1 | 0 | 0 | -100 |
| -100 | -100 | -100 | -100 | -100 | -100 | -100 | -100 |
| -100 | -100 | -100 | -100 | -100 | -100 | -100 | -100 |

B9

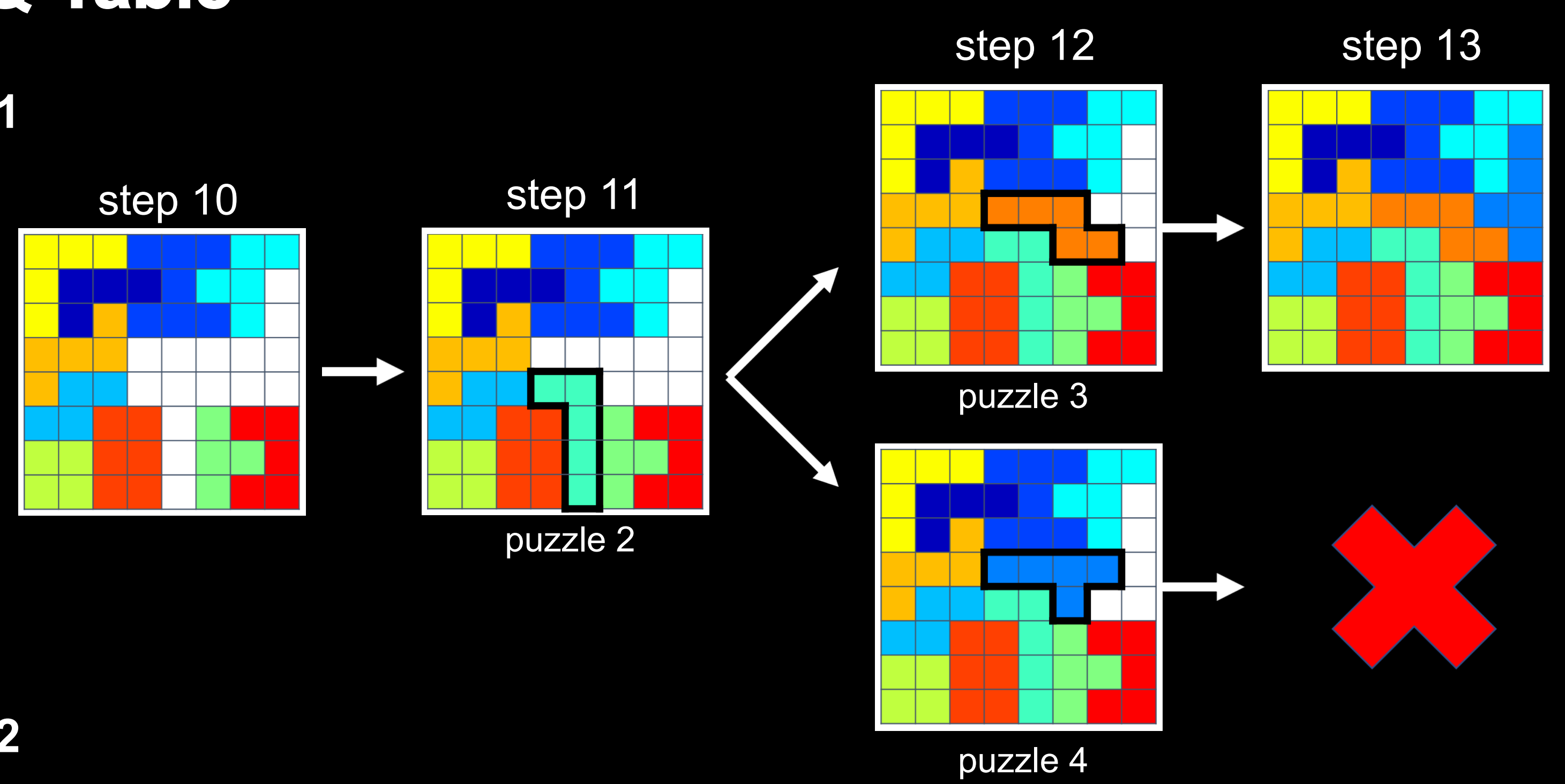


B10

We choose the upper-left corner of the puzzle as the reference point.
B1. Scores are based on the touching edges of the puzzle, such as the yellow part in the B1 and its score is 2 point.
B2. The navy blue squares represents the puzzles that has been put in.
B3. We put the test puzzle from the upper-left corner to the bottom-left corner and calculate the score at each position.
B4,B5. The test puzzle form a hole or out of the bounds, we'll change the score to -100.
B6. After calculating the score of the first column, we'll put the test puzzle on top of the second column and repeat B3.
B7. The puzzle overlaps on the other puzzle, so we change the score to -100.
B8. The test puzzle gets the highest score in this position.
B9. We can get the score matrix of this puzzle after trying all the positions, then we repeat B2-B7 for the remain puzzle. Finially, we choose the next puzzle with highest score in the score matrix, and put at that position which we think is best solution.
B10. By using Score method, we found that it takes between 60 to 200 seconds to solve a solution. However, it is still randomly distributed.

## Q-Table

C1



layer 11

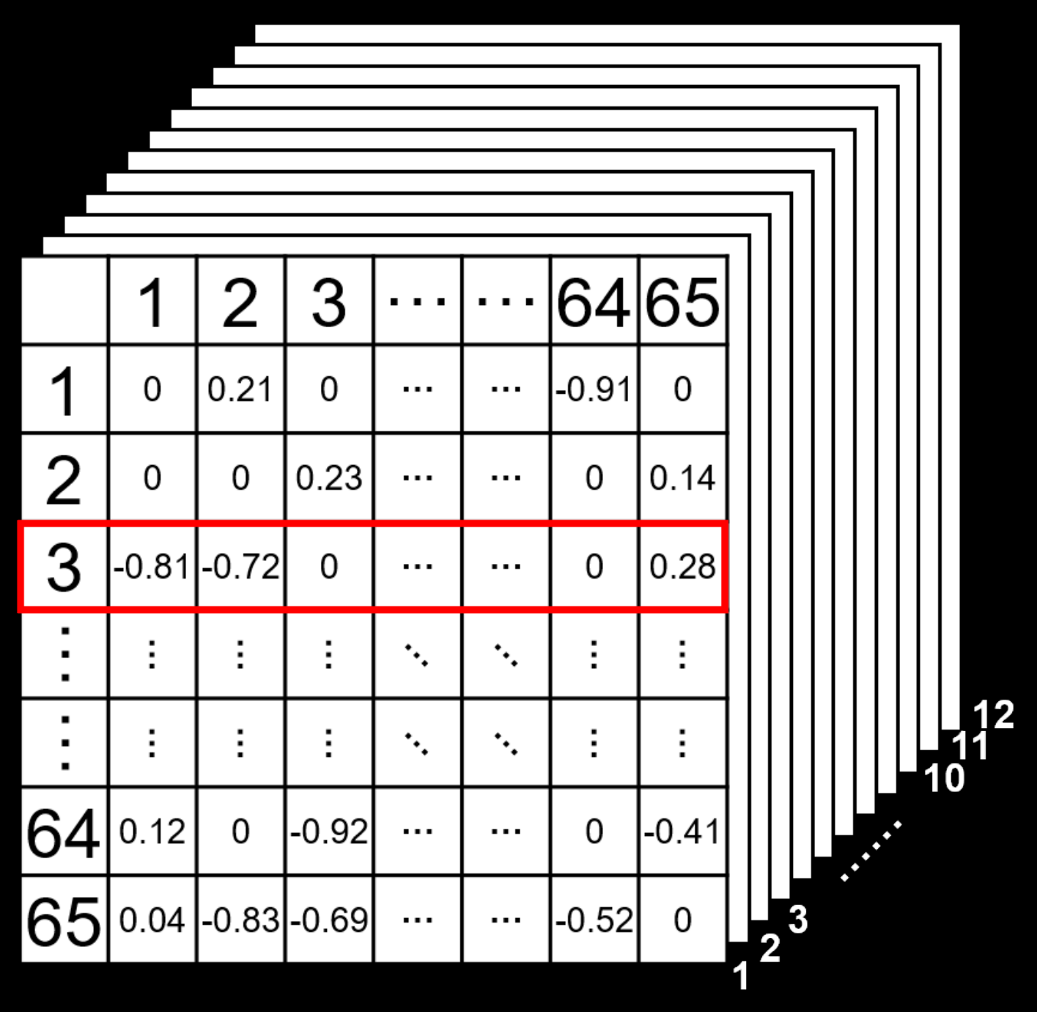
| | 1 | 2 | 3 | 4 | ⋯ | 13 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| 2 | 0 | 0 | +1 | 0 | ⋯ | 0 |
| 3 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| 4 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| 13 | 0 | 0 | 0 | 0 | ⋯ | 0 |

| | 1 | 2 | 3 | 4 | ⋯ | 13 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| 2 | 0 | 0 | 0 | -1 | ⋯ | 0 |
| 3 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| 4 | 0 | 0 | 0 | 0 | ⋯ | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| 13 | 0 | 0 | 0 | 0 | ⋯ | 0 |

C2

$$\boxed{Q(S,A) = Q(S,A) + \alpha[R + \gamma max_a Q(S',a) - Q(S,A)]}$$

C3



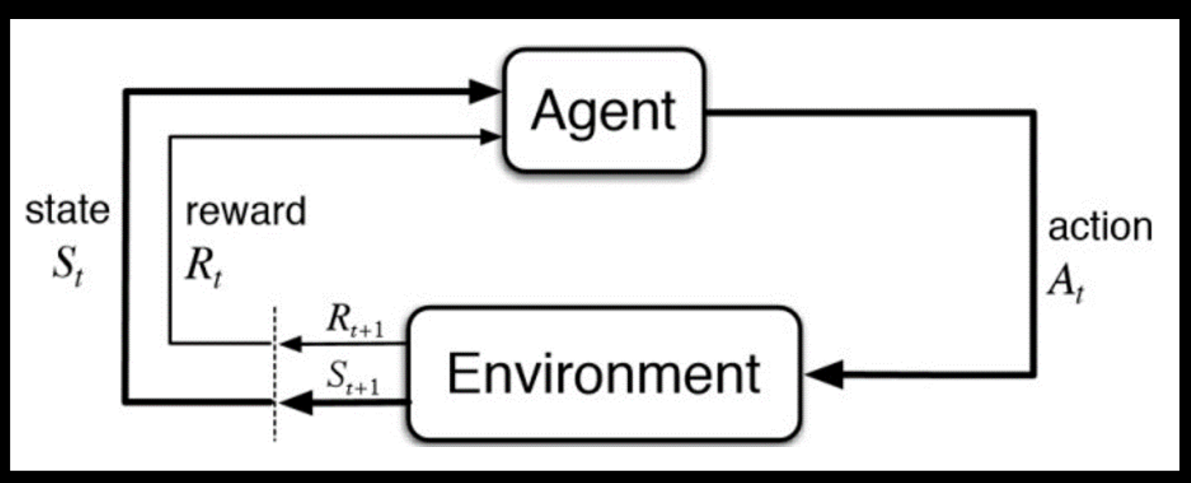
| | 1 | 2 | 3 | ⋯ | ⋯ | 64 | 65 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.21 | 0 | ⋯ | ⋯ | -0.91 | 0 |
| 2 | 0 | 0 | 0.23 | ⋯ | ⋯ | 0 | 0.14 |
| 3 | -0.81 | -0.72 | 0 | ⋯ | ⋯ | 0 | 0.28 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋱ | ⋮ | ⋮ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋱ | ⋮ | ⋮ |
| 64 | 0.12 | 0 | -0.92 | ⋯ | ⋯ | 0 | -0.41 |
| 65 | 0.04 | -0.83 | -0.69 | ⋯ | ⋯ | -0.52 | 0 |

C1. In step 11, we put puzzle 2 into the matrix. The puzzle 3 and puzzle 4 are remain. If we choose puzzle 3 as the next piece, it will be fully filled in the end. Then we will +1 to the Q-value in layer 1 to layer 12 of the corresponding positions. Otherwise, if we choose the puzzle 4, it will not be able to be filled, so Q-value will be deducted.

C2. This is the Q-learning algorithm. α is the learning rate and ε is the discount rate.

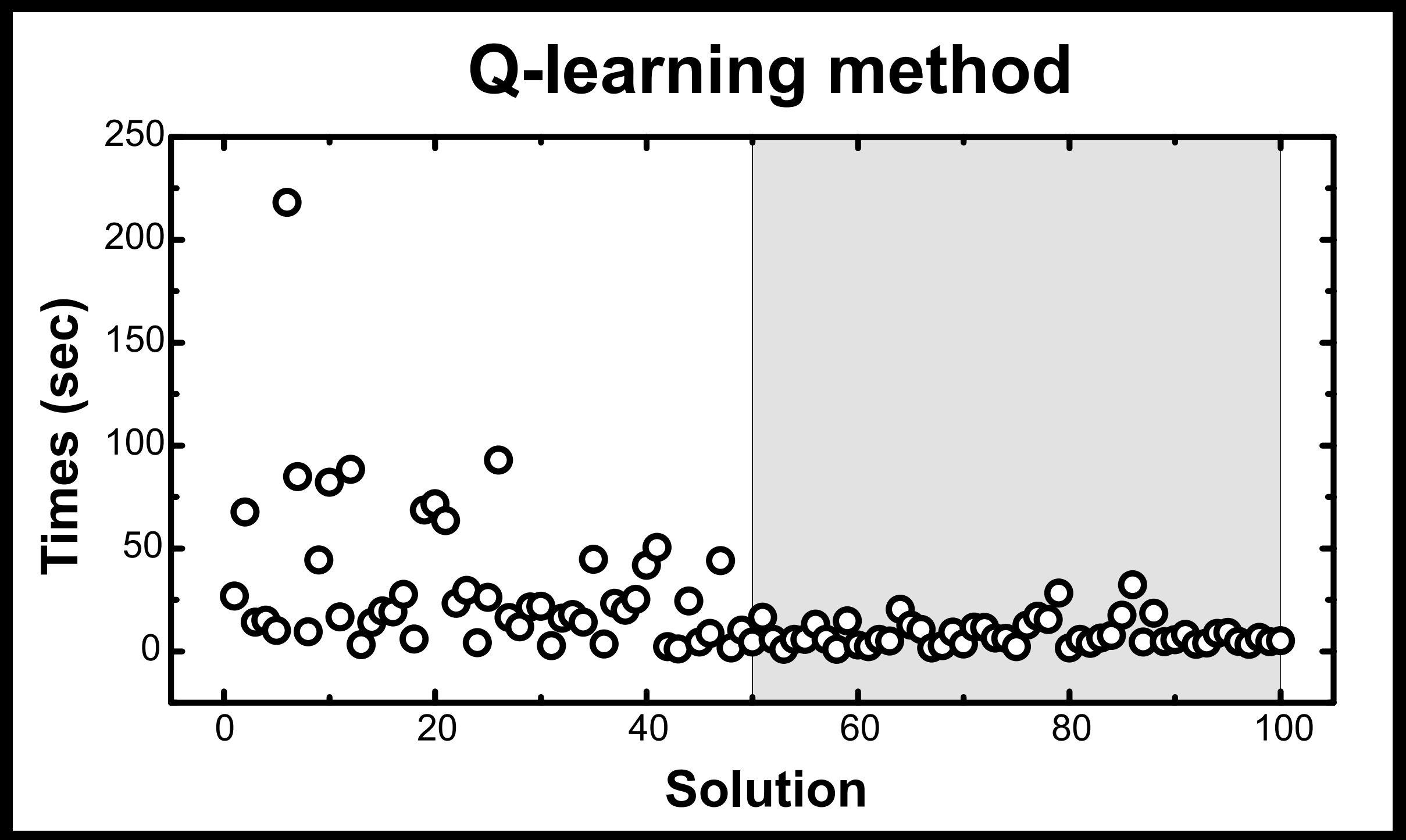
C3. Since each puzzle is different when rotated and mirrored, we take these cases apart so that the total is 65 instead of 13. By continuously using C2 to revise Q-value, the value in the Q-table will closer to the real situation.

C4



C4 is the concept of the Reinforcement Learning. In our situation, Q-table is the agent. It accepts the state from the environment, then judges the best choice. Finally, the environment gives feedback to Q-table to make corrections.

C5



C5. With Q-learning method, we can see the time gradually decreases and approaches a constant value. After 50 success cases training the Q-table, we can get the solution in 5 to 20 seconds.

## Analysis

| | random | score | Q-learning (solution 50-100) |
|---|---|---|---|
| average time (sec) | 196.9 | 62.6 | 8.4 |

## Conclution

At the begin, we use Random method to solve the puzzle, the program is so complex and non-efficient. In order to reduce our program and make smart choice, we introduce the concept of Score. It helped us greatly simplify the program and save a lot of time. At the end, we use Q-learning to allow computers learning from past experiences, and reduce time by choosing puzzle pieces with a higher chance of success.

## Reference

https://www.mathworks.com/help/reinforcement-learning/ug/train-q-learning-agent-to-solve-basic-grid-world.html
https://www.mathworks.com/help/reinforcement-learning/ref/rlqagent.html