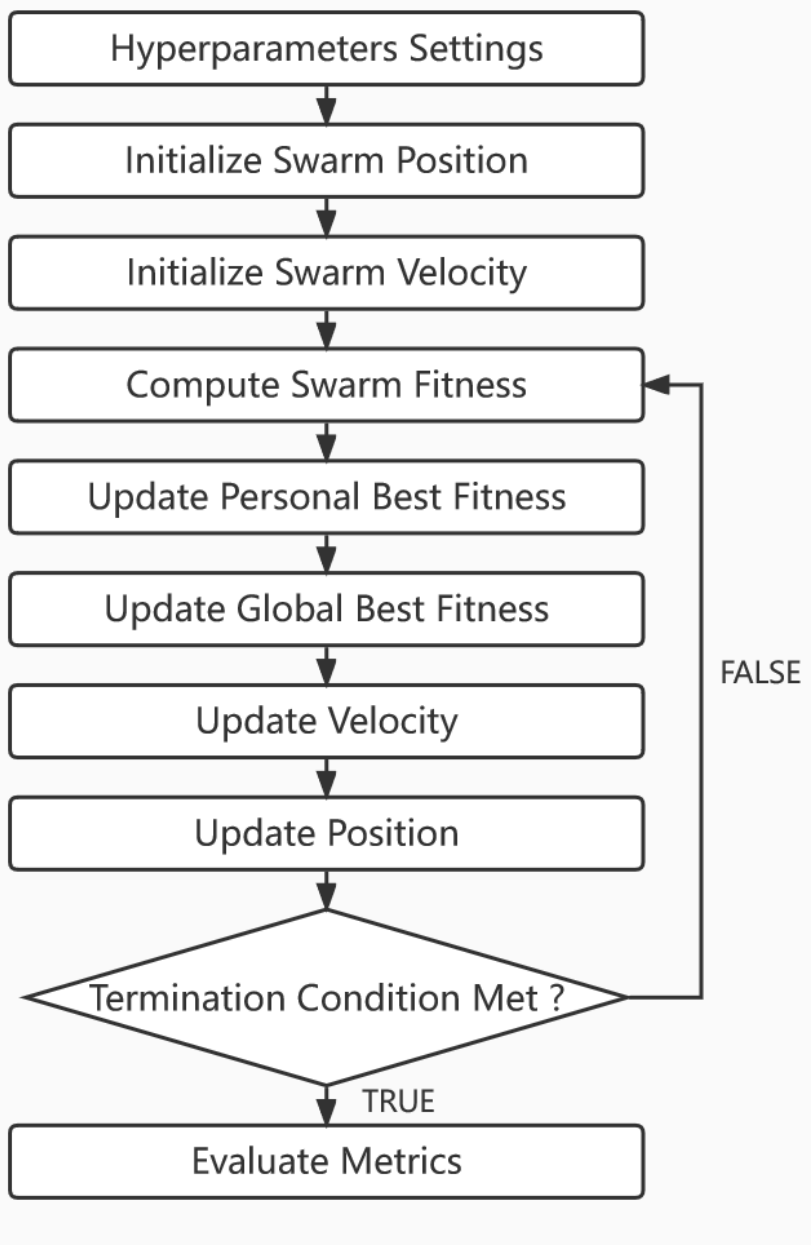
Task: Particles Swarm Algorithm to N-Queen and N-Knight Problem
Authors: Deyi Zeng, Zhanpeng Li

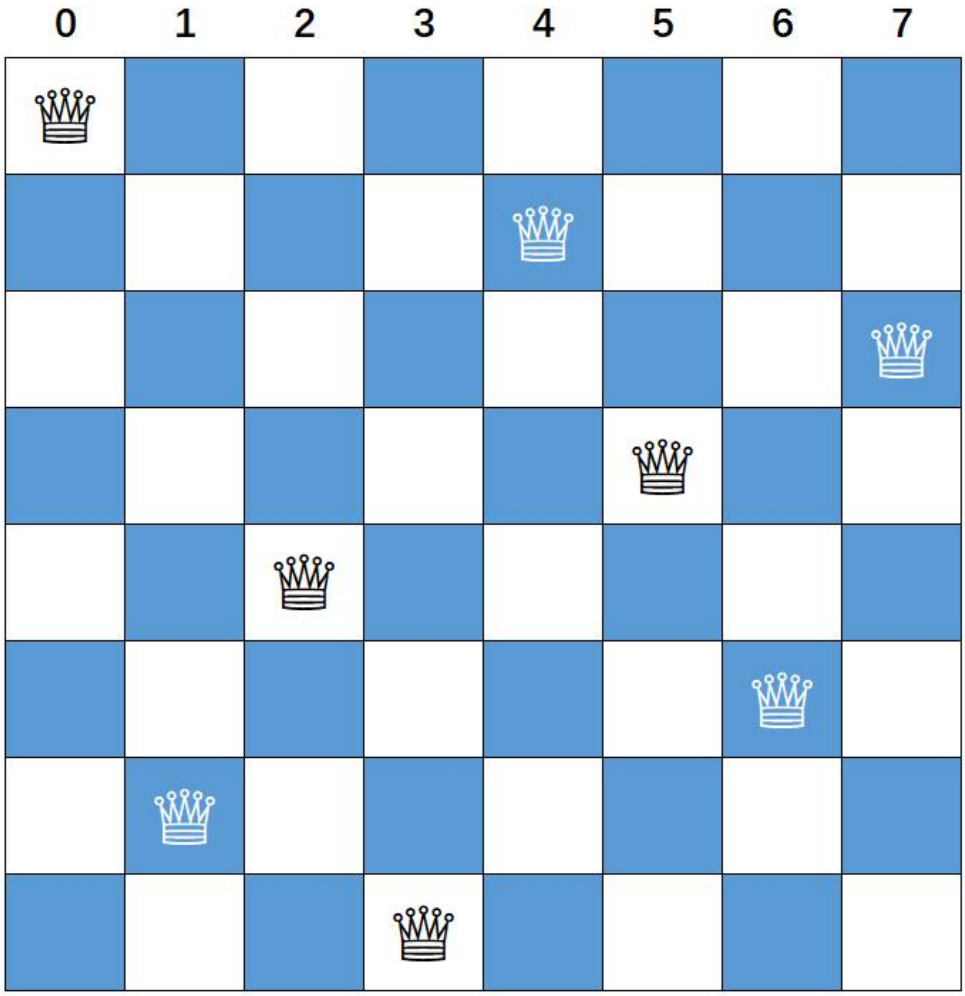
1 N-Queen Problem

The N-Queen model and algorithm have been established by Y. -R. Wang et al.[1].



1.1 Swarm Encoding

Since the queen can attack vertically and horizontally, we can notice that: There can be no more than one queen placed in the same row or column. Given an n x n chessboard and n queens, we may encode the positions of queens as a list from 0 to n-1. For example, in the 8 queens question: [0,4,7,5,2,6,1,3]



1.2 Swarm Position Initialization

To accelerate the convergence, we do not start from scratch. When encoding the swarm/positions of queens,we start from a random number at position 0, then adding random number to the list such that no collisions happen, until all selections fall. Look at the following example:

Figure 1: Refined Initialization: Step 1 to 4

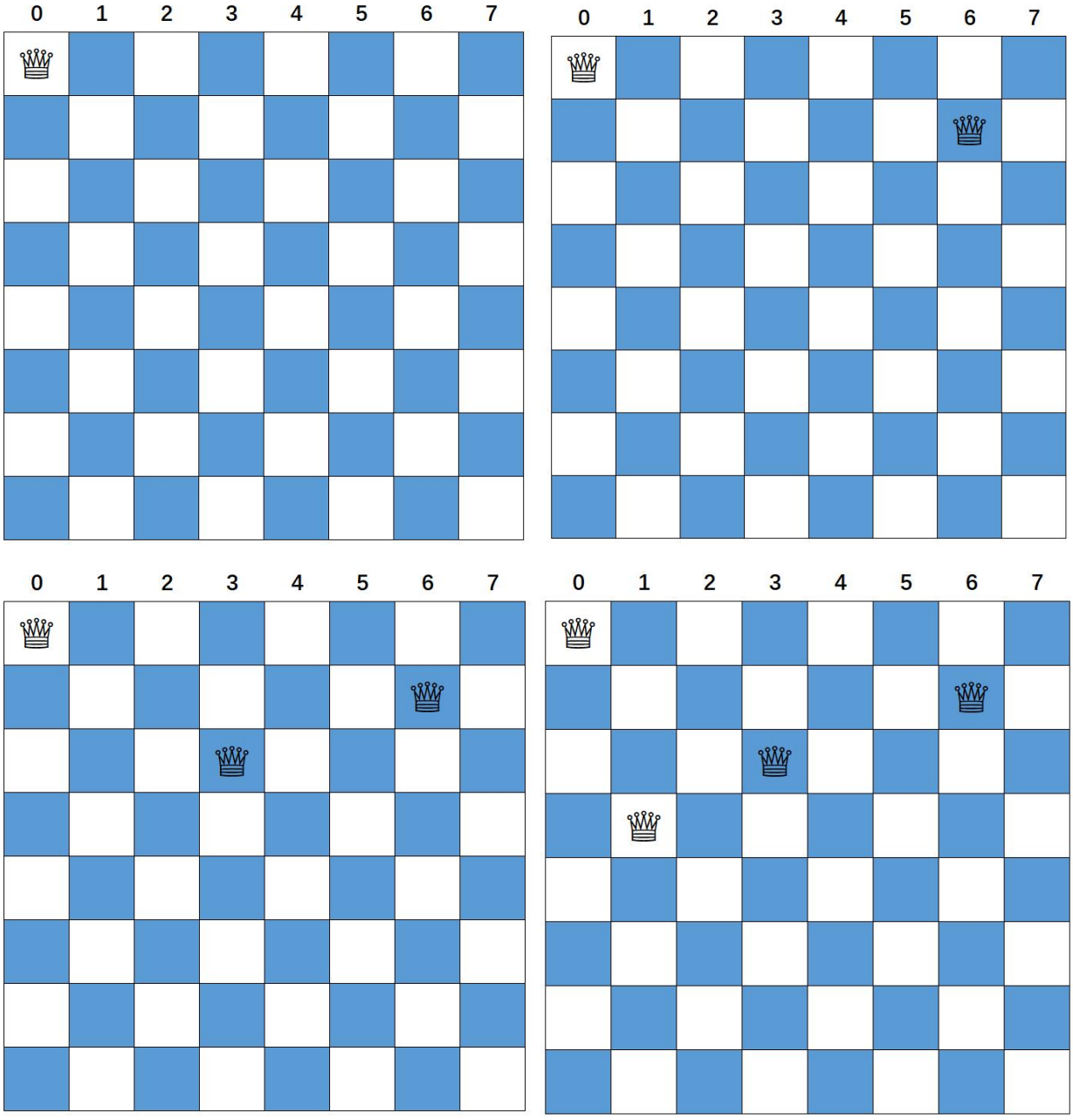
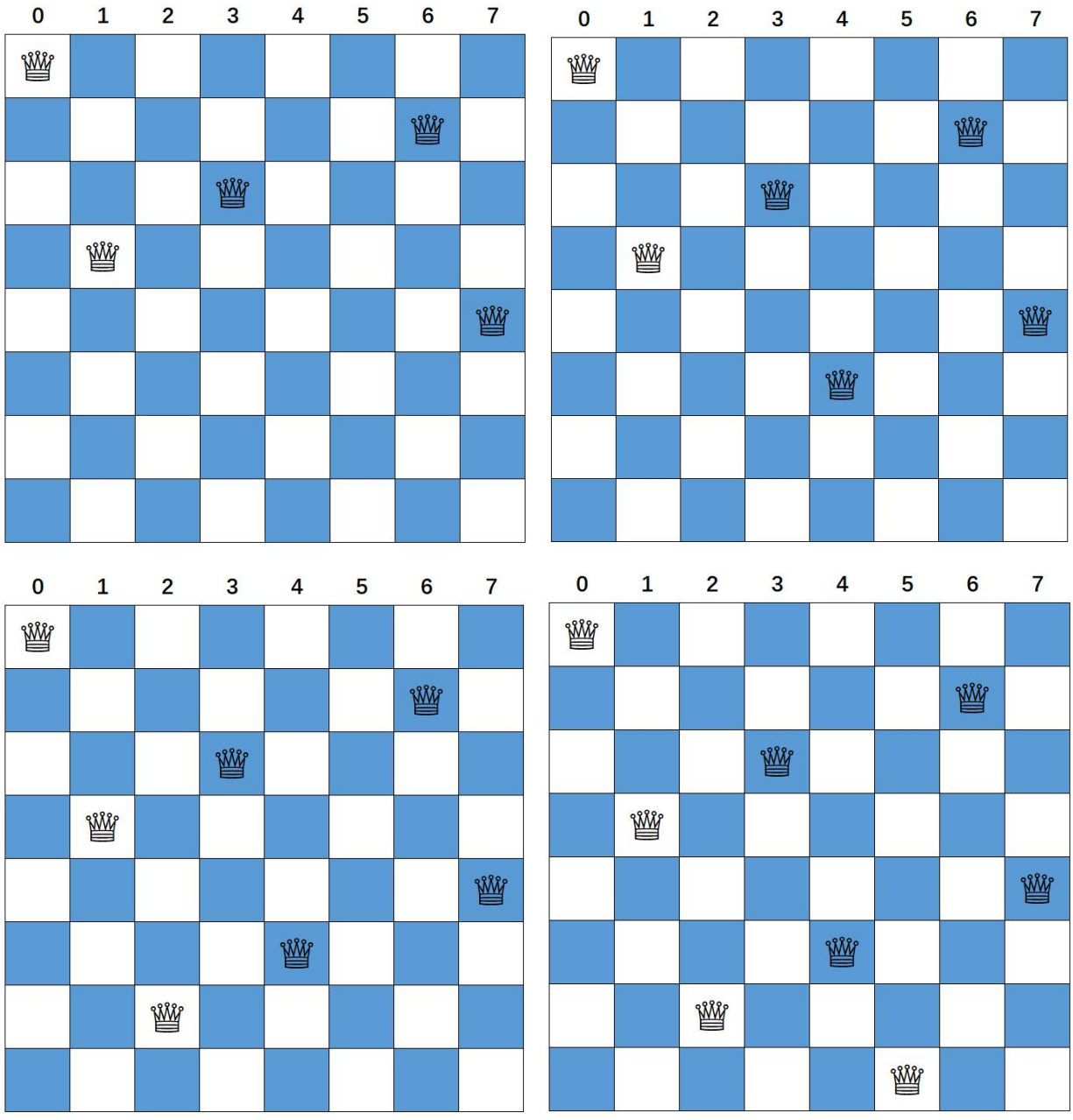


Figure 2: Refined Initialization: Step 5 to 8



We randomly select a number for component 0, and we get 0 (Code:[0]). Then we may randomly select a number from 2 to 7, as position 1 in row 2 incurs collision and position 0 has been chosen. Suppose we get 6, for component 1 (Code:[0,6]). Then we can randomly select from 1,3,4, suppose we get 3 for component 2 (Code:[0,6,3]). Then we can select from 1,5,7, suppose we get 1 for component 3 (Code:[0,6,3,1]). Then we can select only from 7, as 0,1,3,6 have been occupied by previous components, and 2,4,5 can be attacked by previous components . Then component 4 is 7 (Code:[0,6,3,1,7]). For the same reason, component 5 is selected to be 4 and component 6 is selected to be 2 (Code:[0,6,3,1,7,4,2]). Now, if we keep adding queen to the list, there are collisions. So we just randomly append them after the current list, regardless of collisions. (Code:[0,6,3,1,7,4,2,5])

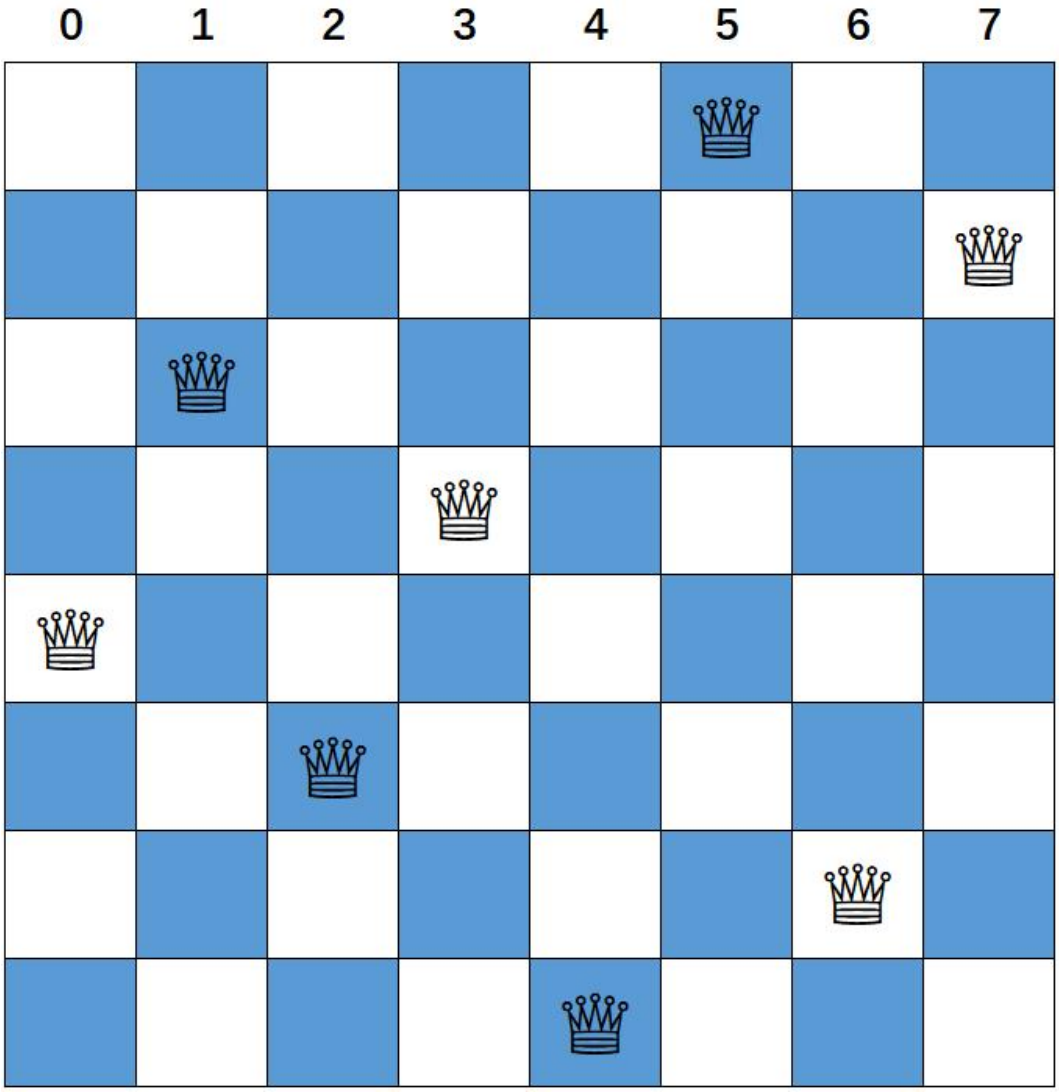
Display of one of the results:

Figure 3: Swarm

```
[[5, 7, 1, 3, 0, 2, 6, 4],
 [6, 4, 1, 5, 0, 3, 7, 2],
 [5, 1, 4, 6, 0, 2, 7, 3],
 [4, 1, 3, 0, 5, 7, 2, 6],
 [4, 6, 1, 2, 5, 3, 0, 7],
 [3, 6, 0, 1, 4, 7, 5, 2],
 [3, 6, 0, 2, 4, 1, 7, 5],
 [1, 6, 0, 3, 7, 4, 2, 5],
 [4, 1, 6, 2, 5, 7, 0, 3],
 [2, 7, 4, 6, 1, 3, 5, 0],
 [5, 1, 6, 3, 0, 7, 4, 2],
 [6, 0, 2, 5, 1, 3, 7, 4],
 [5, 3, 0, 4, 1, 7, 2, 6],
 [6, 0, 2, 4, 1, 7, 5, 3],
 [7, 3, 1, 6, 4, 2, 0, 5],
 [4, 7, 0, 6, 2, 5, 1, 3],
 [1, 5, 6, 0, 3, 7, 4, 2],
 [4, 6, 0, 3, 1, 7, 5, 2],
 [6, 3, 0, 4, 7, 2, 5, 1],
 [6, 3, 0, 4, 1, 7, 5, 2]]
```

The row number stands for the number of particles, while the column counterpart represents the list. Therefore, there 20 particles (P = 20) and this is an 8-Queen problem (N = 8). (Indeed, we can also solve other number of queens problem.) For example, the row 0 means that, swarm 0 [5,7,1,3,0,2,6,4] display a chessboard like this:

Figure 4: Code: [5,7,1,3,0,2,6,4]



1.3 Velocity Initialization

The size of Velocity is radically the same as the array of swarm. But the rule is different. We just assigned a shuffled list([0,1,2,3,4,5,6,7]) to each row/swarm. The following is an example:

Figure 5: Velocity

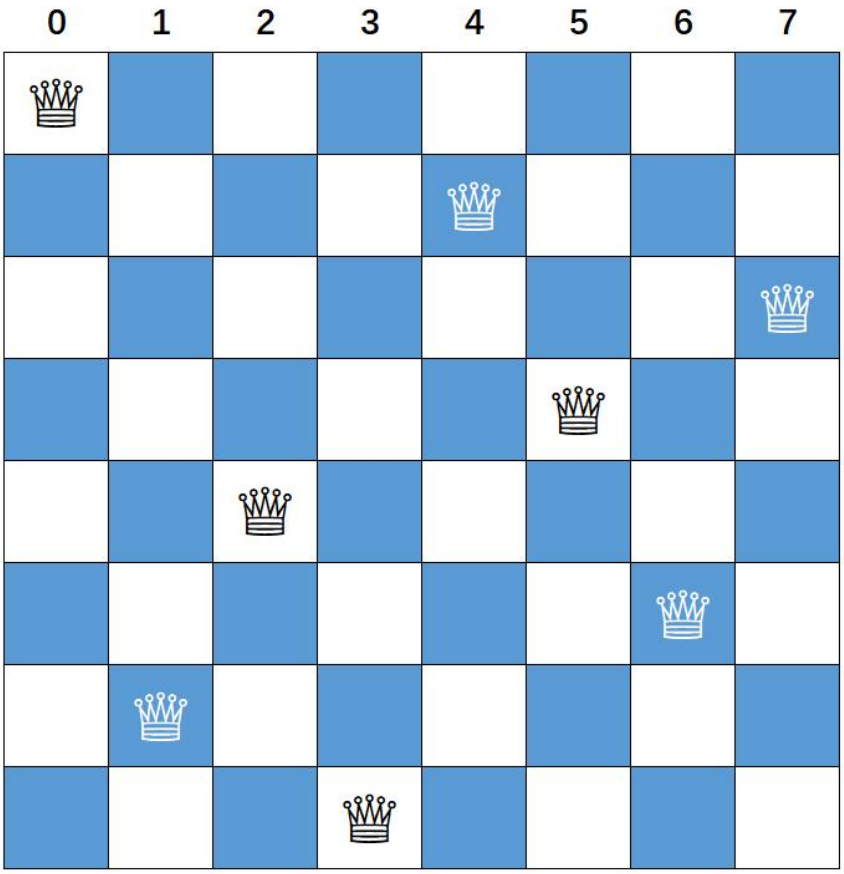
```
[[4, 3, 1, 7, 2, 6, 0, 5],
 [3, 7, 5, 6, 4, 2, 1, 0],
 [1, 2, 6, 5, 4, 7, 0, 3],
 [3, 7, 1, 6, 0, 2, 5, 4],
 [0, 4, 6, 5, 2, 1, 3, 7],
 [0, 5, 6, 7, 2, 4, 1, 3],
 [6, 1, 0, 4, 3, 5, 2, 7],
 [0, 1, 5, 2, 7, 6, 4, 3],
 [6, 4, 5, 2, 3, 0, 1, 7],
 [4, 0, 2, 1, 7, 5, 6, 3],
 [5, 0, 1, 7, 2, 6, 3, 4],
 [0, 1, 5, 4, 6, 2, 3, 7],
 [7, 4, 3, 2, 5, 1, 6, 0],
 [3, 0, 1, 5, 4, 2, 7, 6],
 [4, 0, 1, 7, 6, 5, 3, 2],
 [6, 4, 7, 0, 5, 1, 3, 2],
 [3, 5, 0, 2, 4, 1, 6, 7],
 [2, 3, 6, 5, 0, 7, 1, 4],
 [1, 3, 2, 7, 6, 0, 5, 4],
 [4, 1, 0, 7, 3, 5, 6, 2]]
```

1.4 Fitness

The fitness is treated as a loss function, and therefore we should define it carefully. In N-Queen Problem, the fitness is defined as follows: The aggregate of collisions, provided that a queen can only collide with no more than one other queen in the same direction. Maybe you are somewhat perplexed by definition here, but we can fathom it through some examples.

1.4.1 Example 1

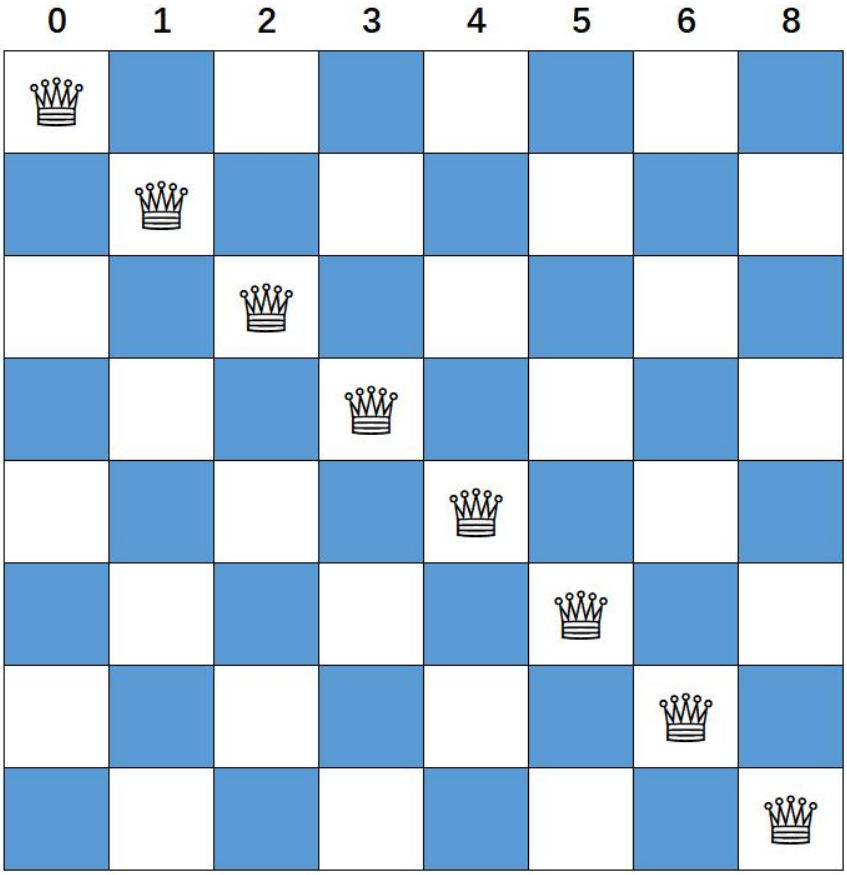
Figure 6: Fitness = 0



It is obvious that there are no collisions. Since the fitness equals to the aggregate collision number, fitness = 0.

1.4.2 Example 2

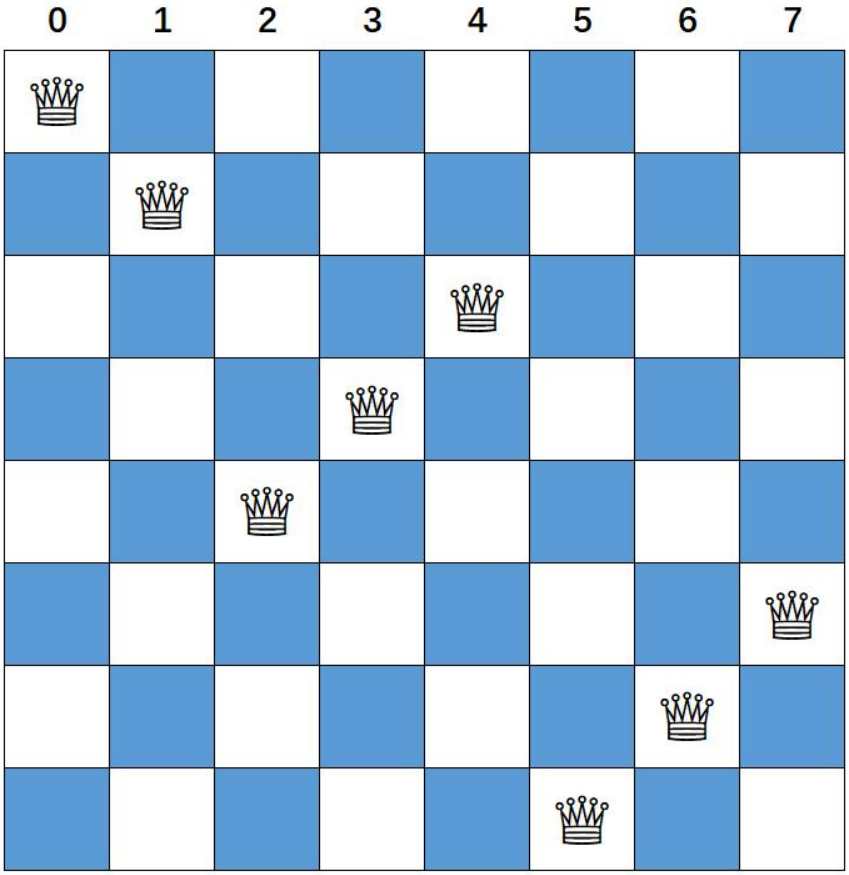
Figure 7: Fitness = 7



In this case, queen on row 0 collides with queen on row 1, so the collision number of queen on row 0 is 1; the queen on row 1 collides with both queen on row 0 and queen on row 2. But we count the collisions pairwisely, so only one collision will be counted. So the collisions of queen on row 1 is also 1. Then, for each queen, the collision number is counted to be one. The fitness is defined to be the aggregate collison number. We also need to notice that a queen can only collides with no more than one other queens in the same direction. That is, when queen on row 2 collides with queen on row 3, then queen on row 2 cannot collide with queen on row4, 5,6, and 7.

1.4.3 Example 3

Figure 8: Fitness = 8



In this case, queen on row 0 collides with queen on row 1, so the collision number of queen on row 0 is 1; the queen on row 1 collides with queen on row 3; the queen on row 2 collides with queen on row 3 and row 5; the queen on row 3 collides with queen on row 4; the queen on row 4 collides with queen on row 7; the queen on row 5 collides with queen on row 6; the queen on row 6 collides with queen on row 7; queen on row 7 collides with nothing else. Therefore, the aggregate collision number is 8.

1.5 Velocity Update

The updated velocity is the weighted average of three parts: Inertia, Personal Best and Global Best. The weight are w, c_1 and c_2 respectively. $rand_1$ and $rand_2$ are random number between 0 and 1. Swap in equation (5) is a function, we will give the definition in the next subsection.

$$V_{new} = |w * V_{old} + C_1 * Rand_1 * (P_{best} - Pos_{old}) + C2 * Rand_2 * (G_{best} - Pos_{old})| \quad (1)$$

$$w = 1 - c_1 - c_2 \quad (2)$$

$$0 \leq c_1 \leq 1 \quad (3)$$

$$0 \leq c_2 \leq 1 \quad (4)$$

$$0 \leq rand_1 \leq 1 \quad (5)$$

$$0 \leq rand_2 \leq 1 \quad (6)$$

1.5.1 Example

$w = 0.2, C_1 = 0.4, C_2 = 0.4, rand_1 = 0.5, rand_2 = 0.3$

Figure 9: Compute V_{new} with given information

Gbest	0	6	3	1	7	4	2	5
Pbest	0	1	2	3	4	5	6	7
Posold	7	6	5	4	3	2	1	0
Vold	5	7	1	3	0	2	6	4
Vnew	1.24	0.4	0.64	0.04	0.68	1.24	2.32	2.8

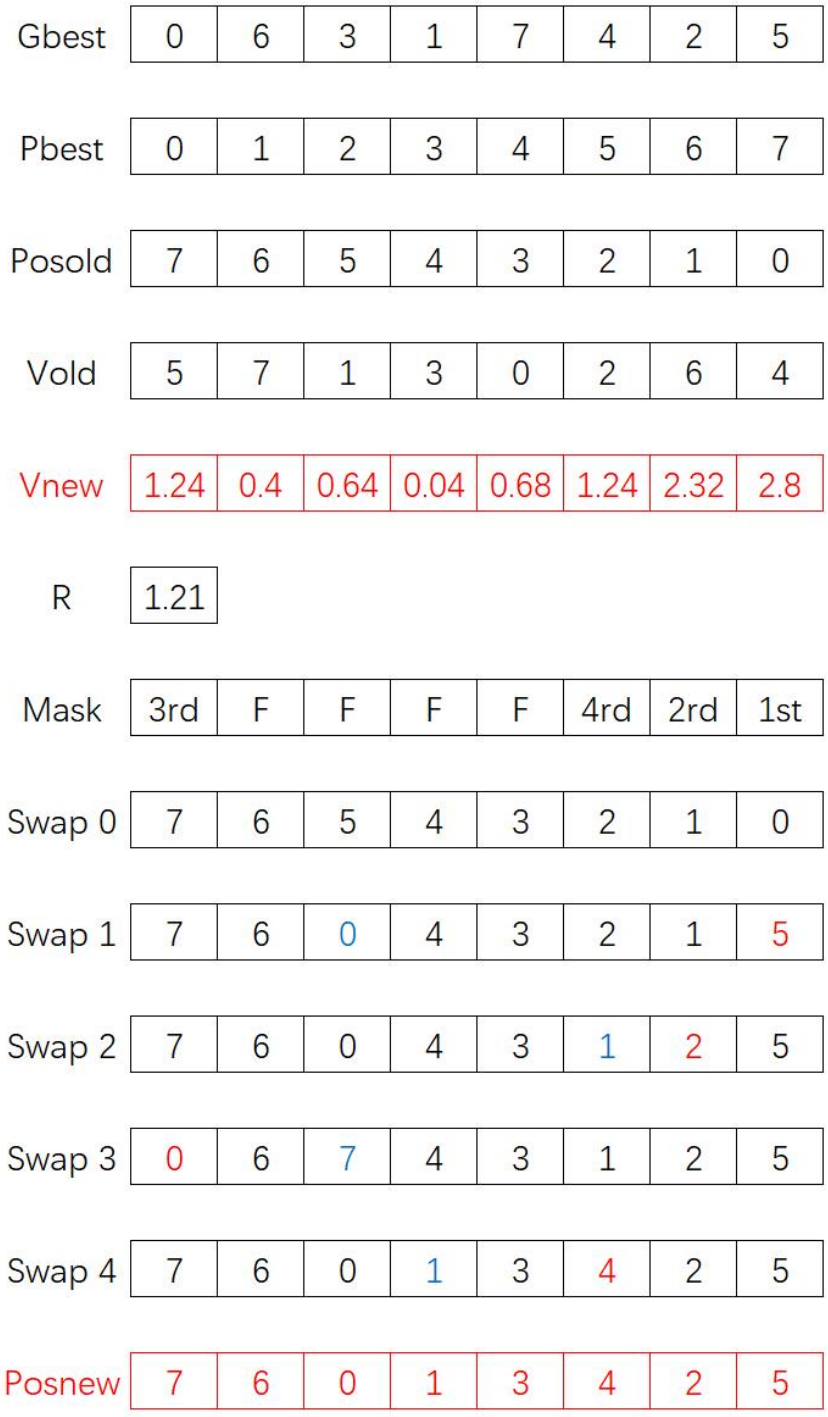
1.6 Swarm Position Update

Conventionally, we may add the velocity to the old position to form the new one. But this incurs problem as the position code cannot contain repeated elements, and the components should be integers as well. We need to use another method called 'swap' to mimic the process.

$$Pos_{new} = swap(Pos_{old}, V_{new}) \tag{7}$$

Randomly select a number R from the interval: $[min(V\_new), max(V\_new)]$. Only consider the component with velocity larger than R. Then swap the considered component with other component in the list, such that the swapped list has the same number as the old one, at the component under consideration. Repeat the process until all considered components have been exhausted. Consider the example continued.

Figure 10: Compute Pos_{new} with given information

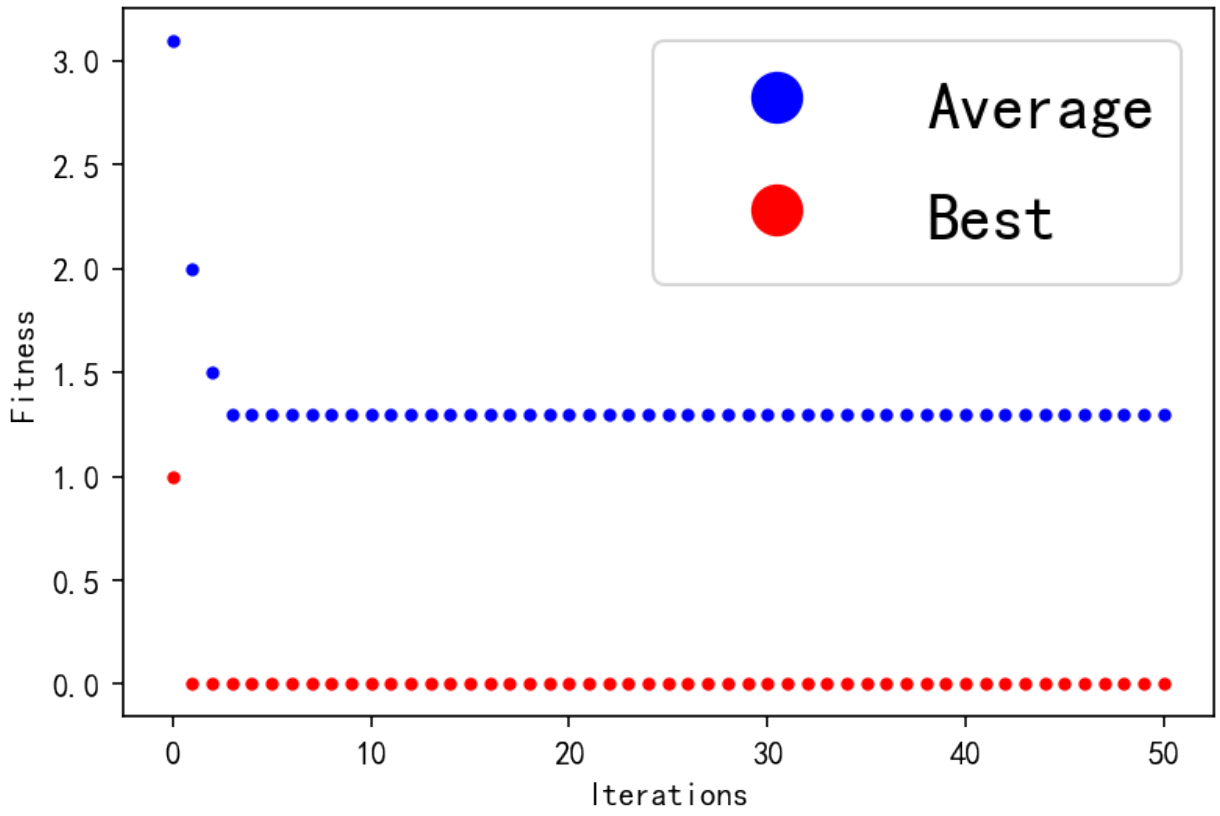


1.7 Metrics Evaluation

We consider two metrics in this problem: 'Average' and 'Best', which are the average fitness of swarms and the best fitness among swarms at a certain epoch, respectively.

1.7.1 Metrics of 8-Queen Problem

Settings: # of Iterations = 50, # of Particles = 10, $W = 0.4$, $C_1 = 0.3$, $C_2 = 0.3$

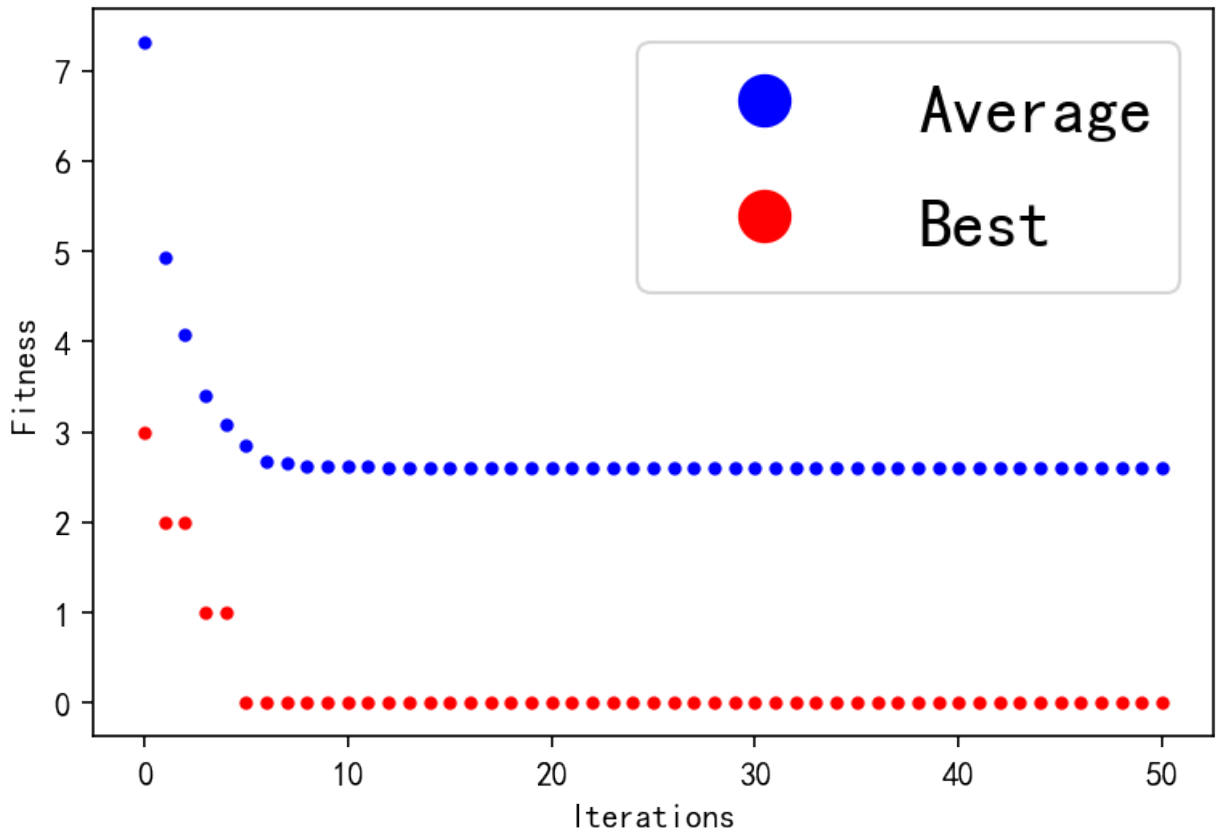


Analysis: It converges too fast to recognize the process of convergence, we may implement this in a higher dimensional chessboard.

One of solutions: [4, 1, 7, 0, 3, 6, 2, 5]

1.7.2 Metrics of 16-Queen Problem

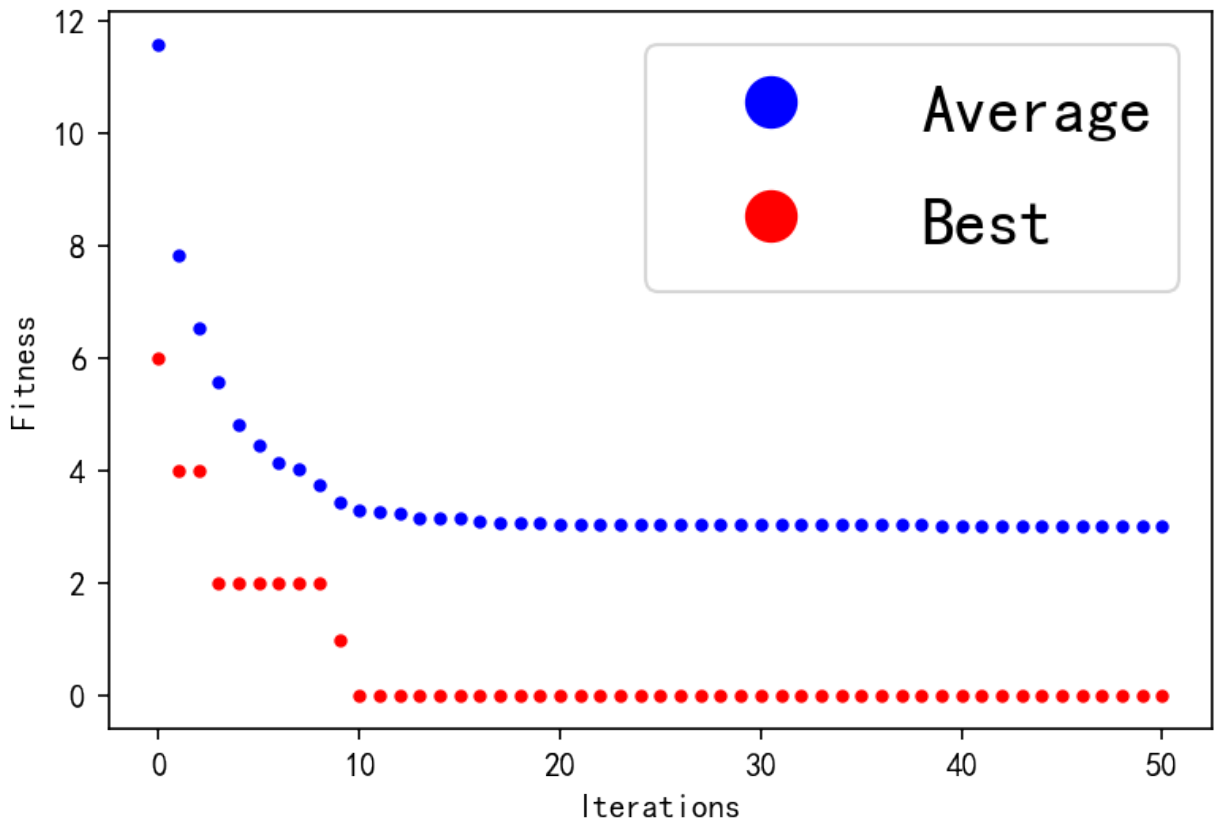
Settings: # of Iterations = 50, # of Particles = 50, $W = 0.6$, $C_1 = 0.2$, $C_2 = 0.2$



One of solutions: [7, 9, 13, 2, 0, 8, 12, 5, 3, 10, 15, 11, 14, 4, 6, 1]

1.7.3 Metrics of 24-Queen Problem

Settings: # of Iterations = 50, # of Particles = 50, $W = 0.85$, $C_1 = 0.1$, $C_2 = 0.05$



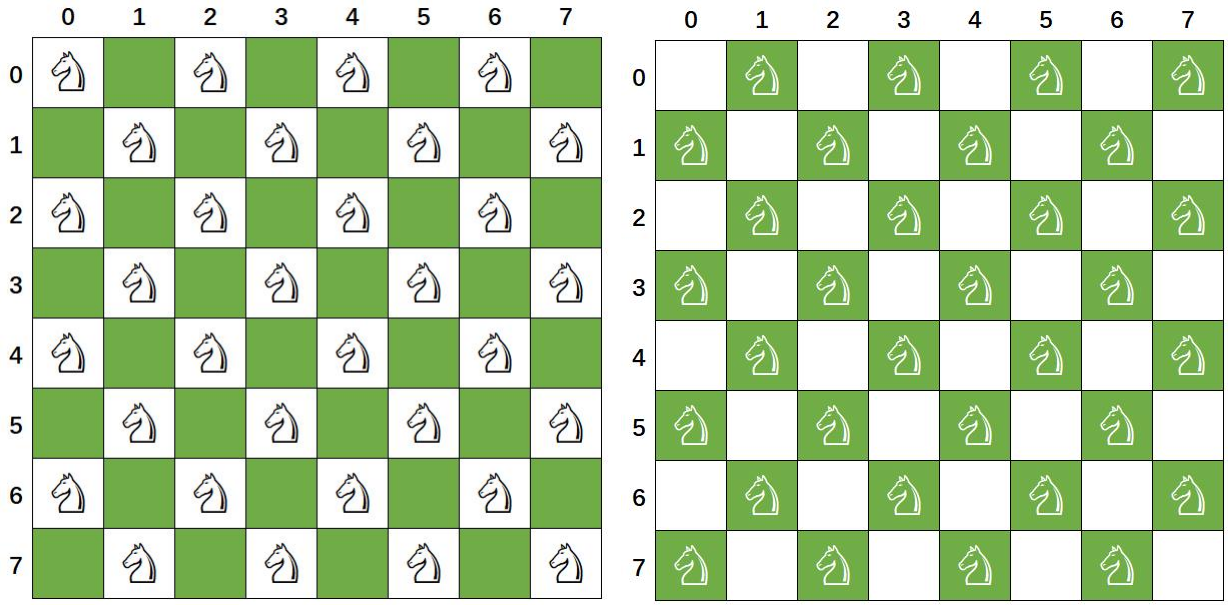
Analysis: Notice that the C_1, C_2 and w are carefully picked up. Not following these hyperparameters results that the best value does not converge to 0(for example, it may converge to 1). You may have noticed that, when the N increases, the required c_1 and c_2 decreases and w increases. This implies that the guidance from Personal Best Value and Global Best Value are more important when N is smaller, while the Inertia is more important when N is large.

One of Solutions = [12, 8, 5, 2, 0, 19, 21, 9, 13, 17, 23, 4, 10, 1, 20, 15, 3, 6, 22, 16, 11, 7, 14, 18]

2 N-Knight Problem

2.1 Proof of Theory

It is manifest that at least we can place 32 knights on the chessboard, in the following two methods:



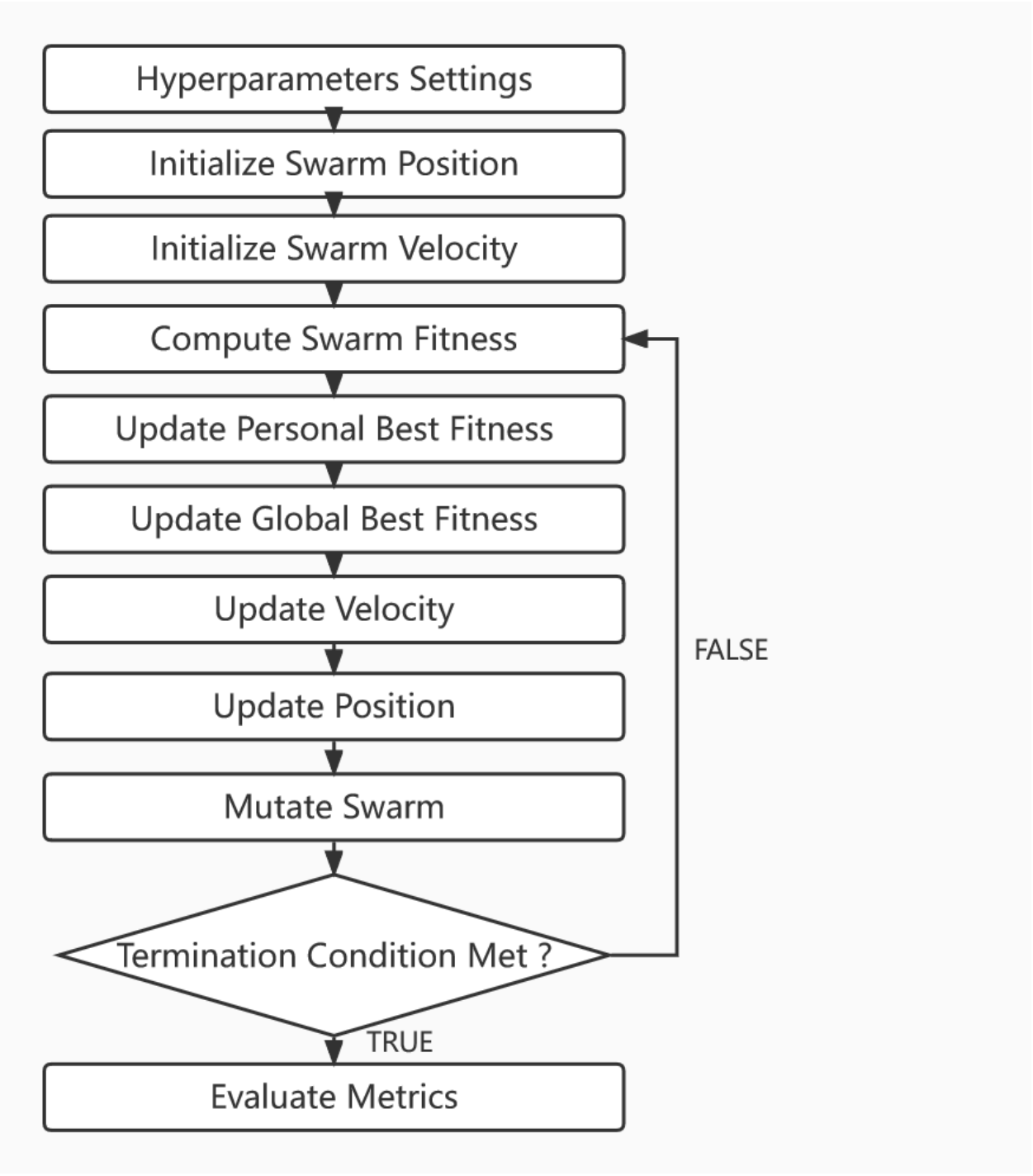
We claim that the largest compatible knights on 8x8 chessboard are exactly 16. This can be proved rigorously: We prove by induction. We can prove by enumerating that when N = 4(on 4x4 chessboard), there are no more than 8 compatible knights. The programming is simple, considering all 11440 cases when there are exactly 9 knights on the chessboard. From the programming(codes has been attached), we know that, none of 11440 cases can be compatible, let alone the cases with more than 9 knights.

When N = 8(on 8x8 chessboard), we can see the chessboard as a combination of four 4x4 chessboard, so there must be no more than 8 * 4 = 32 knights.

We have given two possible solutions for exactly 32 knights, coupled with the proof that there can be no more than 32 knights on 8x8 chessboard, we conclude that the the largest compatible knights are 32 on a 8x8 chessboard.

Generally and similarly, we can also prove that on a When N = 4k(4kx4k Chessboard), where k is a positive integer, the largest compatible knights are $k^2/2$

Follow this idea, we can start directly from randomly assigning 32 knights on the board, making them wandering and searching on the chessboard such that there are compatible.

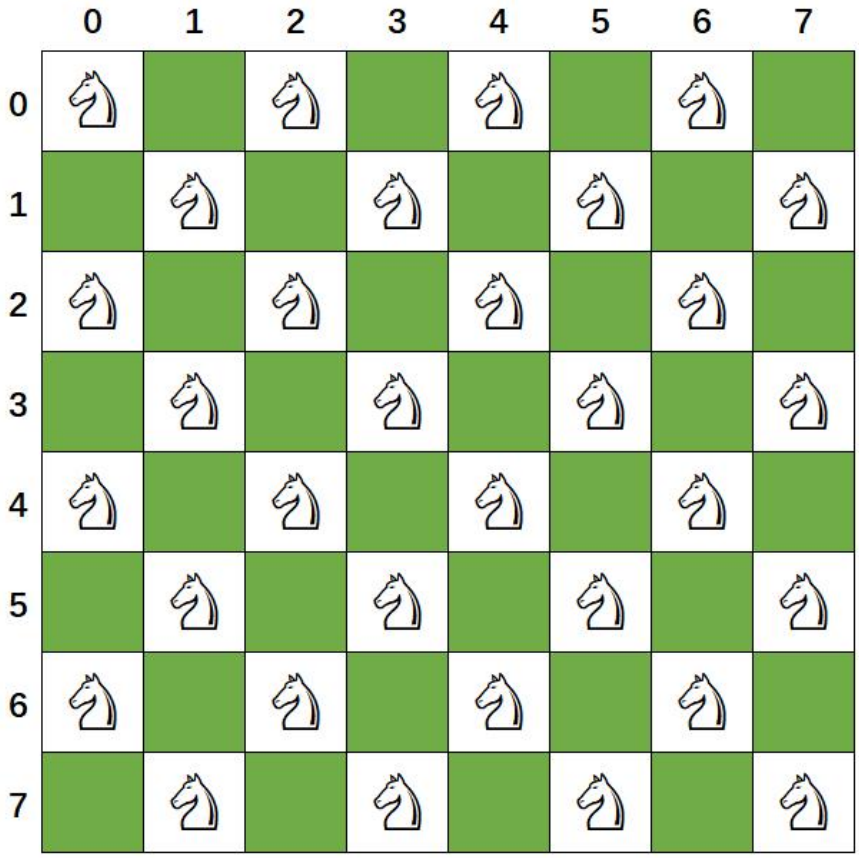

Hyperparameters Settings
Initialize Swarm Position
Initialize Swarm Velocity
Compute Swarm Fitness
Update Personal Best Fitness
Update Global Best Fitness
Update Velocity
Update Position
Mutate Swarm
Termination Condition Met ?
FALSE
TRUE
Evaluate Metrics

2.2 Swarm Encoding

We prepare two coding systems for N-Knight Problem: Index code and Duple Code. This is the Code of Chessboard.

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63

We should represent each swarm with a 32-length list, employing Index Code. For example, the following sample is represented as: [0,2,4,6,9,11,13,15,16,18,20,22,25,27,29,31, 32,34,36,38,41,43,45,47,48,50,52,54,57,59,61,63]



2.3 Swarm Position Initialization

Suppose that you have P particles, then you will get a Px32 matrix, and each column is a shuffle of number from 0 to 63.

2.4 Swarm Velocity Initialization

Suppose that you have P particles, then you will get a Px32 matrix, and each column is a shuffle of number from 0 to 31.

2.5 Fitness

The fitness is defined to be the number of pairwisely-trampling(attacking) pairs of knights.

2.6 Velocity Update

The formula of computing velocity is somewhat different from the N-Queen Question. In this question, we employ Manhattan distance (under duple code) to depict the distance between two particles, which is denoted as d.

$$V_{new} = |w * V_{old} + C_1 * Rand_1 * d(P_{best}, Pos_{old}) + C2 * Rand_2 * d(G_{best}, Pos_{old})| \quad (8)$$

$$w = 1 - c_1 - c_2 \quad (9)$$

$$d(x, y) = (|x \ \mathbf{div}\ 8 - y \ \mathbf{div}\ 8| + |x \ \mathbf{mod}\ 8 - y \ \mathbf{mod}\ 8|)^2 \quad (10)$$

$$0 \leq c_1 \leq 1 \quad (11)$$

$$0 \leq c_2 \leq 1 \quad (12)$$

$$0 \leq rand_1 \leq 1 \quad (13)$$

$$0 \leq rand_2 \leq 1 \quad (14)$$

2.7 Position Update

We can retain the thinking in N-Queen problem, with some modifications. Still, randomly select a R, between min(velocity) and max(velocity). Then only the value larger than R will be considered in the next step(the same as that in N-Queen Question). Then we start from the largest velocity: The component swap with other component in the same list such that it is the same as the Gbest.(This is the same as in N-Queen Question). We need to notice that, since there are only 32 components, but 64 possible cases in Gbest. So if this component does not exist in the list, just replace the original one with it, rather than swapping. We can demonstrate it with an example:

2.8 Position Mutation

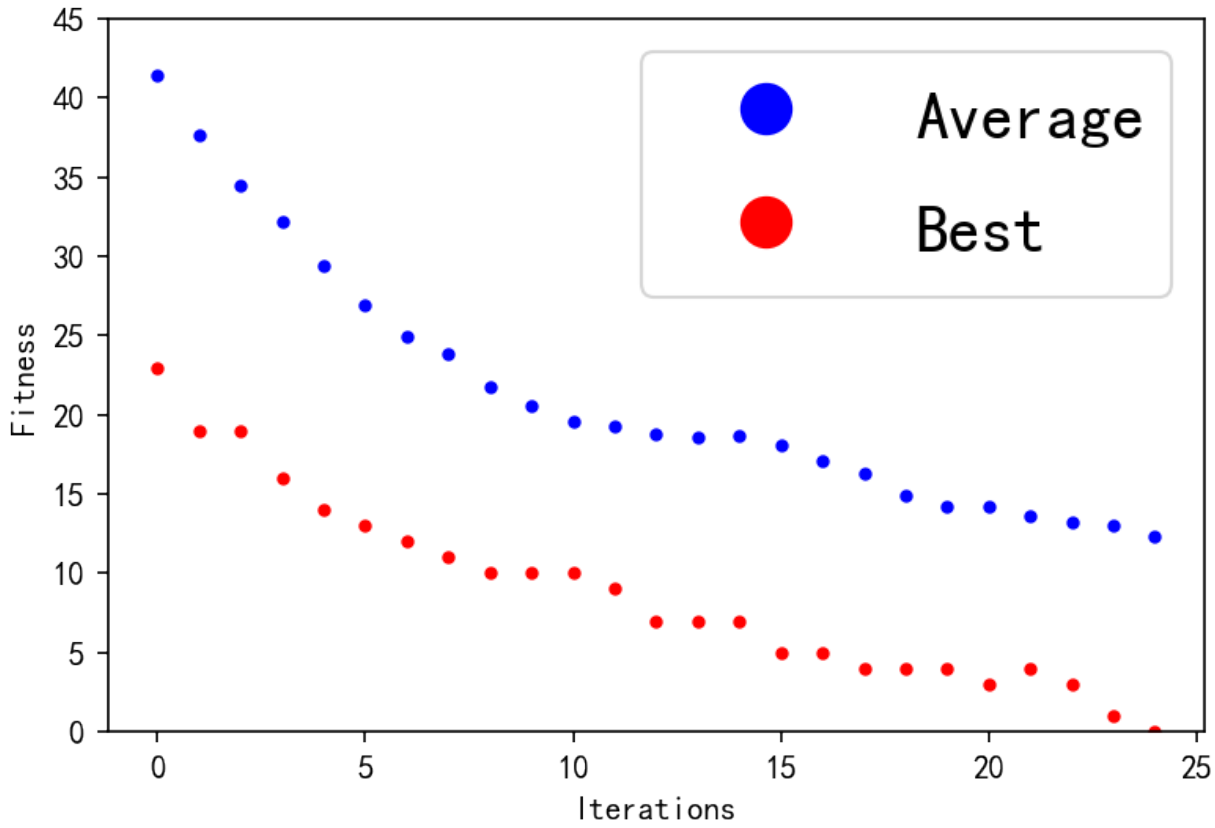
This step is one of the most vital ones. Without mutation, the problem will be trapped into a local minimum and cannot run out of it. With the mutation we can have other possibilities. We just replace one component of the code with unused one.

Gbest	61	33	20	1	55	45	9	5
Posold	0	1	2	3	4	5	6	7
Vnew	1.2	0.4	0.6	0	0.7	1.2	2.3	2.8
R	1.2							
Mask	3rd	F	F	F	F	4rd	2rd	1st
Swap 0	0	1	2	3	4	5	6	7
Swap 1	0	1	2	3	4	7	6	5
Swap 2	0	1	2	3	4	7	9	5
Swap 3	61	1	2	3	4	7	9	5
Swap 4	61	1	2	3	4	45	9	5
Posnew	61	1	2	3	4	45	9	5

Posnew	61	1	2	3	4	45	9	5
Mutated	61	1	2	3	4	59	9	5

2.9 Metrics Evaluation

of Particles = 3000; # of Iterations = 25; w = 0.8; $c_1 = 0.1$; $c_2 = 0.1$;



Analysis: N-knights problem requires a large amount of particles, compared to the N-Queens problem, to achieve a satisfying result. It converges swiftly, which can be attributed to the Mutation. Without mutation, the best value will be stable on fitness = 3 or 4(or some other small value) but never 0. In this figure, we know that the swarm can be further improved, but since a compatible solution is found, termination condition is met.

2.10 Display of Results

One of Solutions:[42, 62, 44, 40, 39, 46, 53, 7, 10, 3, 17, 30, 58, 12, 51, 21, 1, 33, 37, 5, 55, 8, 28, 56, 24, 49, 14, 23, 35, 60, 19, 26]

3 Reference

[1] Y. -R. Wang, H. -L. Lin and L. Yang, "Swarm Refinement PSO for Solving N-queens Problem," 2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications, 2012, pp. 29-33, doi: 10.1109/IBICA.2012.43.