Shirin Mohebbi

Evolutionary Computing

Assignment #2

Case1:

Random initialization

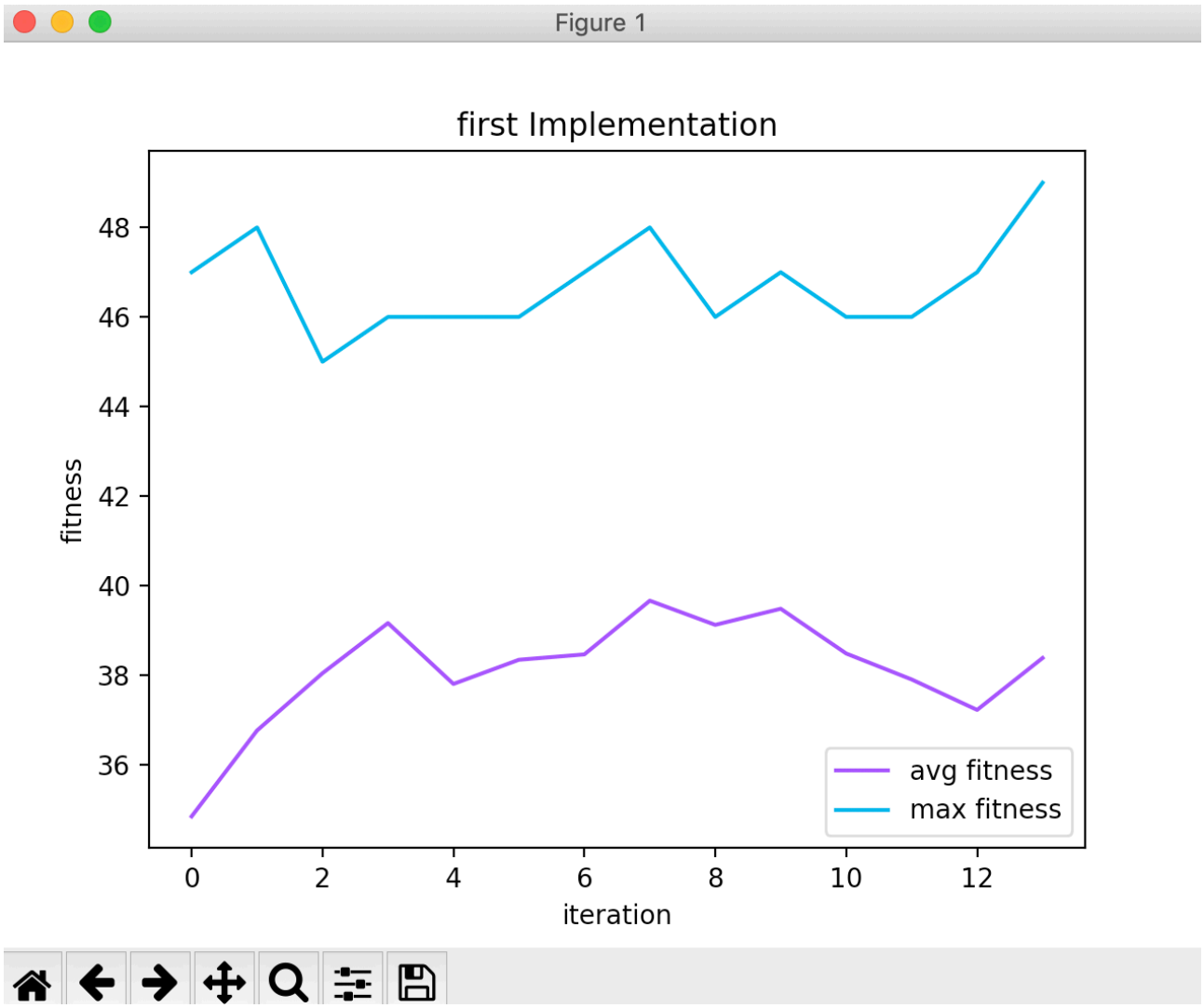
Population size: 50

Tournament selection

10 % elitism

mutation swap

Result:



2

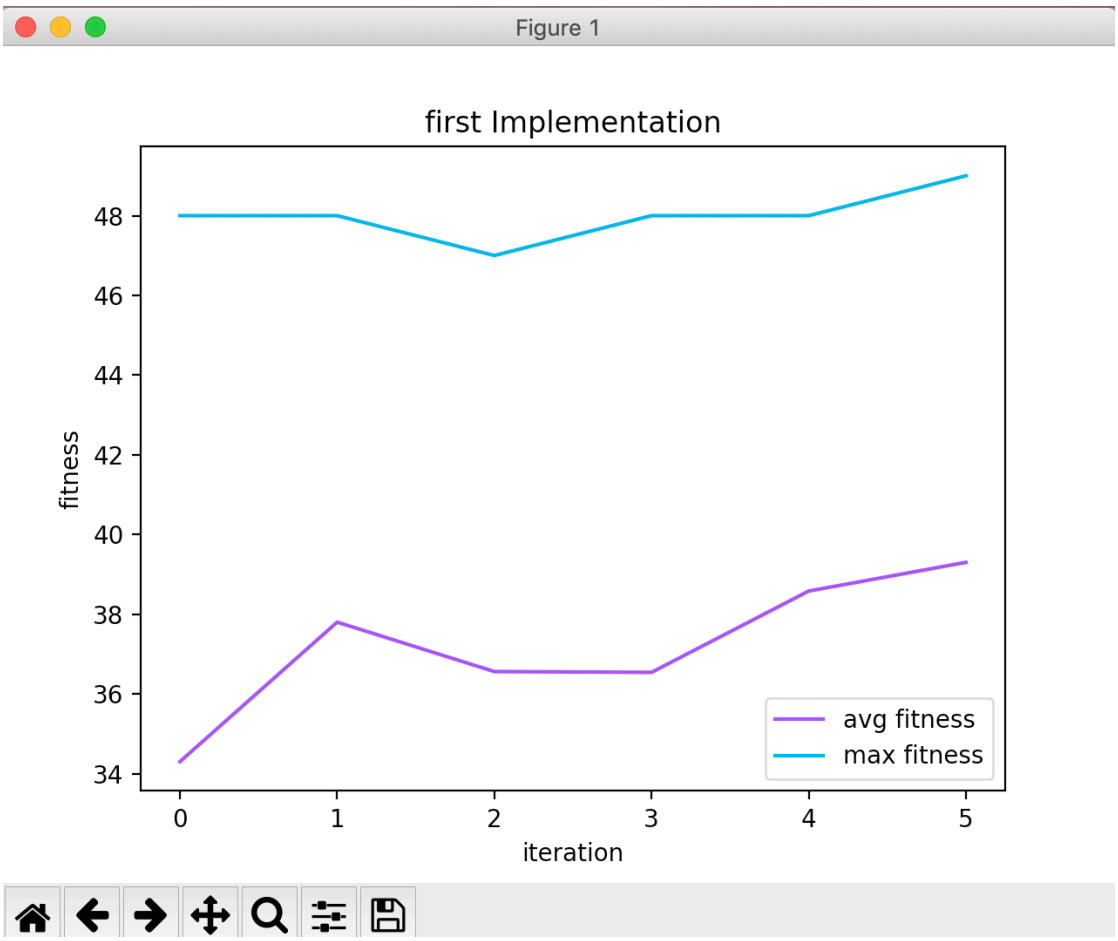


Figure 1
first Implementation
fitness
iteration
avg fitness
max fitness
48
46
44
42
40
38
36
34
0
1
2
3
4
5

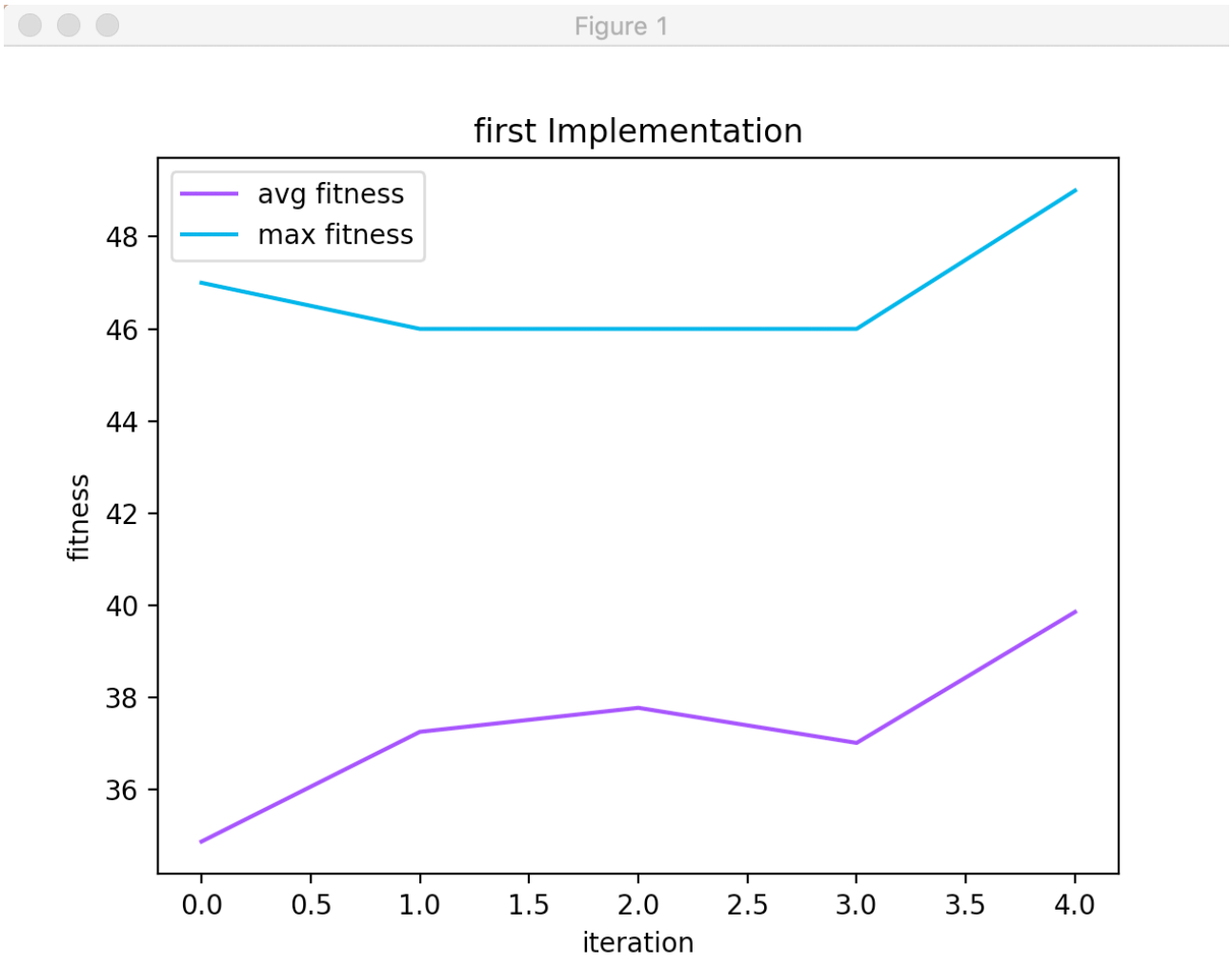


Figure 1
first Implementation
fitness
iteration
avg fitness
max fitness
48
46
44
42
40
38
36
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

1.Best strategy is: [0, 8, 8, 4]

How does the best strategy work?

Colonel knows that it's not possible to win all 4 battles, so he decides to lose one battle with minimum number of solders(0), so he can use more solders for 3 other battles and increase his wining chance, average solders on each battle is 5, colonel has two 8, so he wins this two battle more likely. and 4 has still good amount of chance for wining.

2.Best strategy is: [8, 7, 3, 2]

How does the best strategy work?

This strategy is pretty much like first strategy, but he doesn't lose a battle for sure like the above, instead he uses 2 or 3 solders, so he still has small chance of winning. and he has 7 and 8, which has strong chance of wining.

3.Best strategy is: [6, 6, 4, 4]

How does the best strategy work?

This strategy works completely different from two other, two 6 which are one more than average(5) and more likely to win and two 4 which are one less than average but still has good chance of winning

Case2:

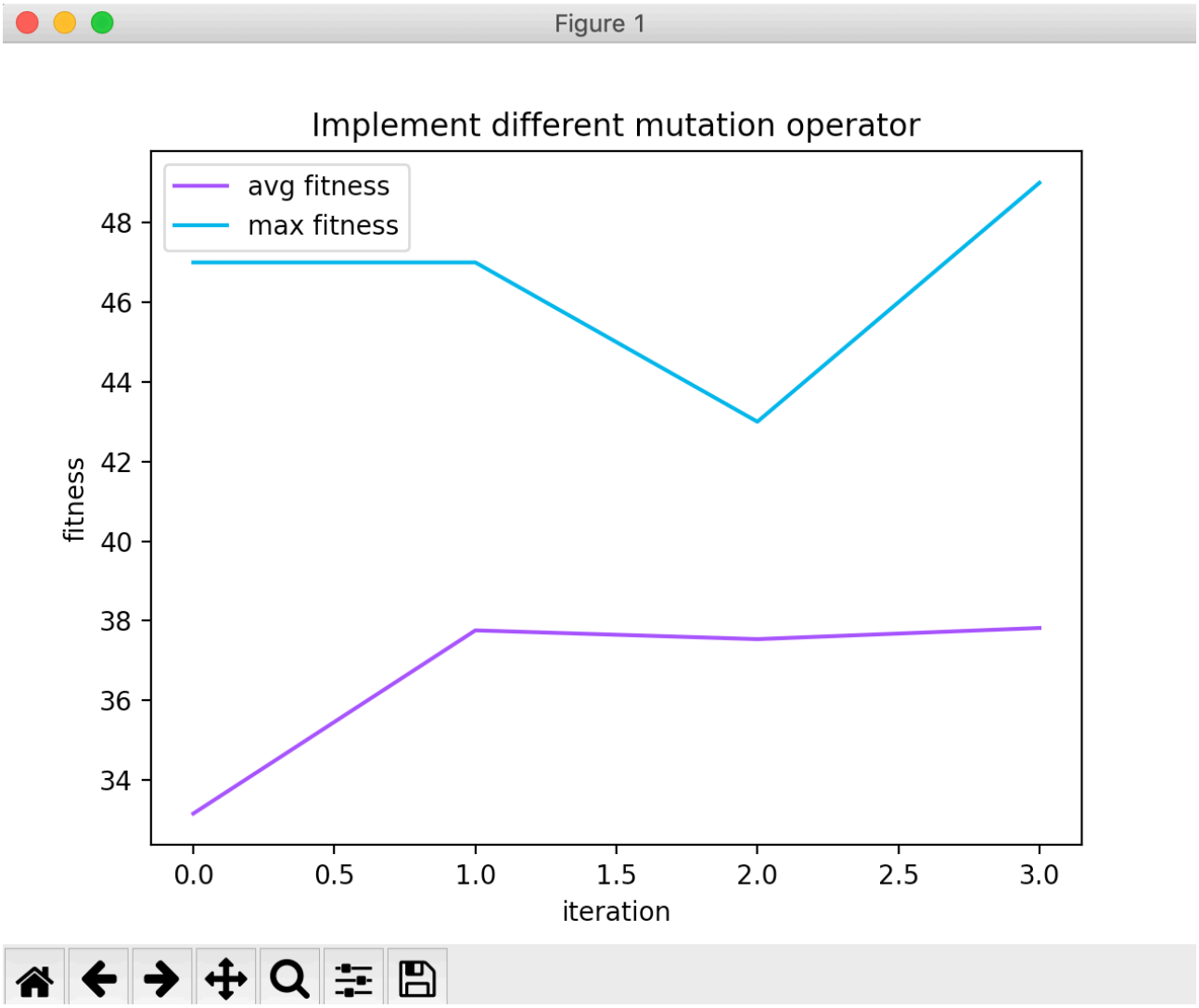
Random initialization

Population size: 50

Tournament selection

10 % elitism

mutation transfer one solder



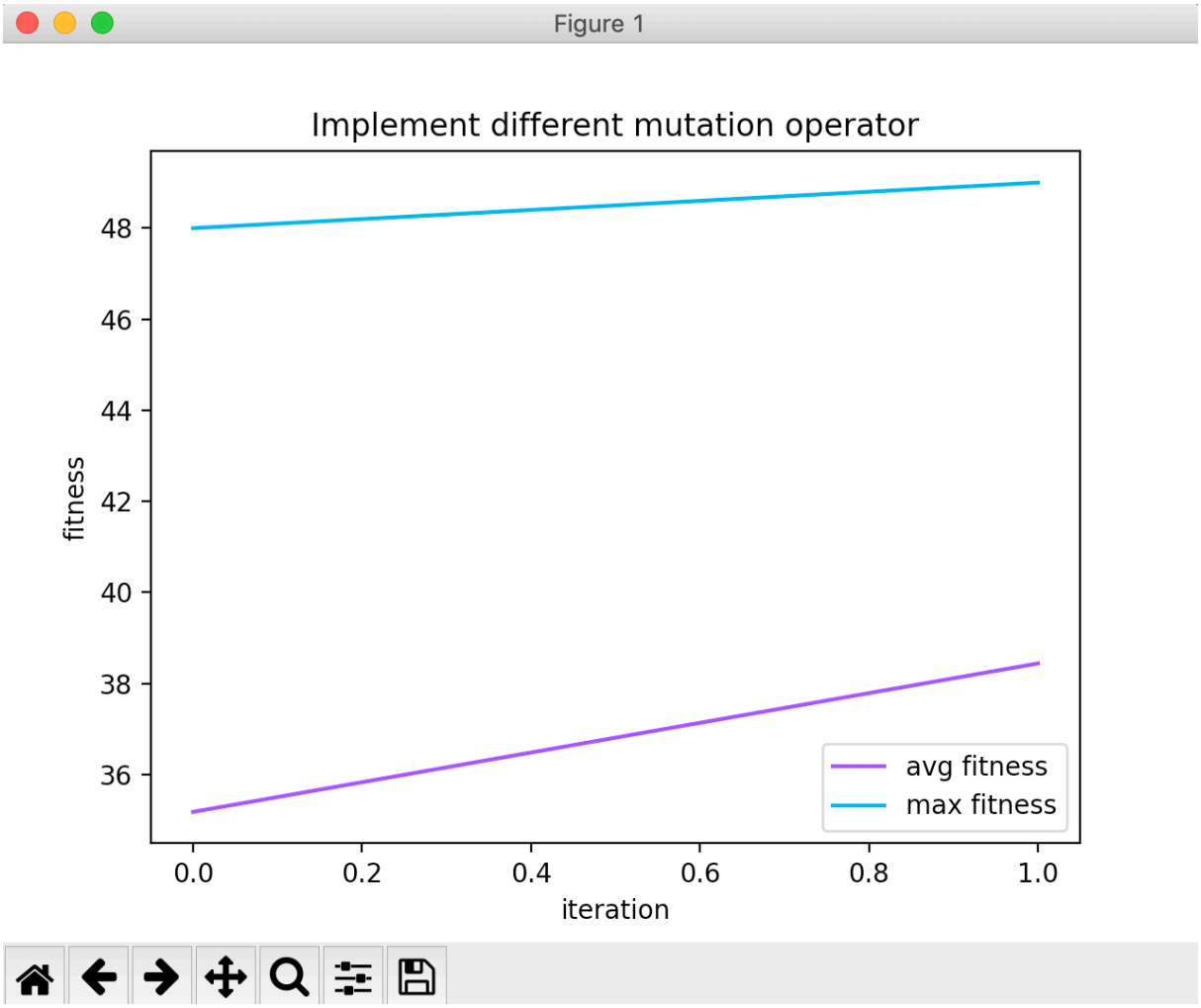


Figure 1
Implement different mutation operator
fitness
iteration
avg fitness
max fitness
48
46
44
42
40
38
36
0.0
0.2
0.4
0.6
0.8
1.0

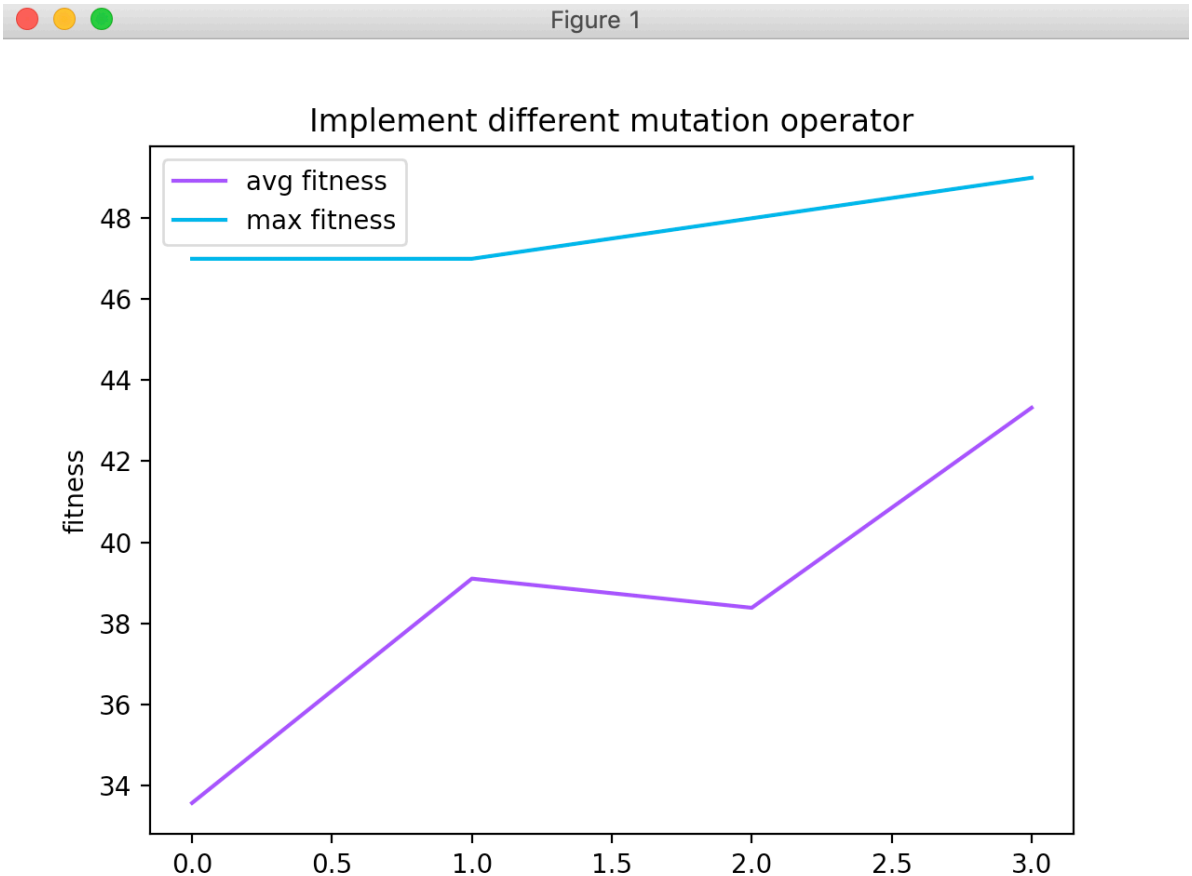


Figure 1
Implement different mutation operator
fitness
avg fitness
max fitness
48
46
44
42
40
38
36
34
0.0
0.5
1.0
1.5
2.0
2.5
3.0

1.best strategy is: [2, 4, 7, 7]

How does the best strategy work?

it has two 7, so two win with high probability and 4 has still good chance of wining, and 2 has small chance of wining too.

2.best strategy is: [3, 7, 8, 2]

How does the best strategy work?

it has 7 and 8, so two win with high probability and 3 and 2 has still chance of wining.

3.best strategy is: [9, 8, 1, 2]

How does the best strategy work?

it has two 7 and 8, so two win with high probability. 1 and 2 has small chance of wining too.

Case3:

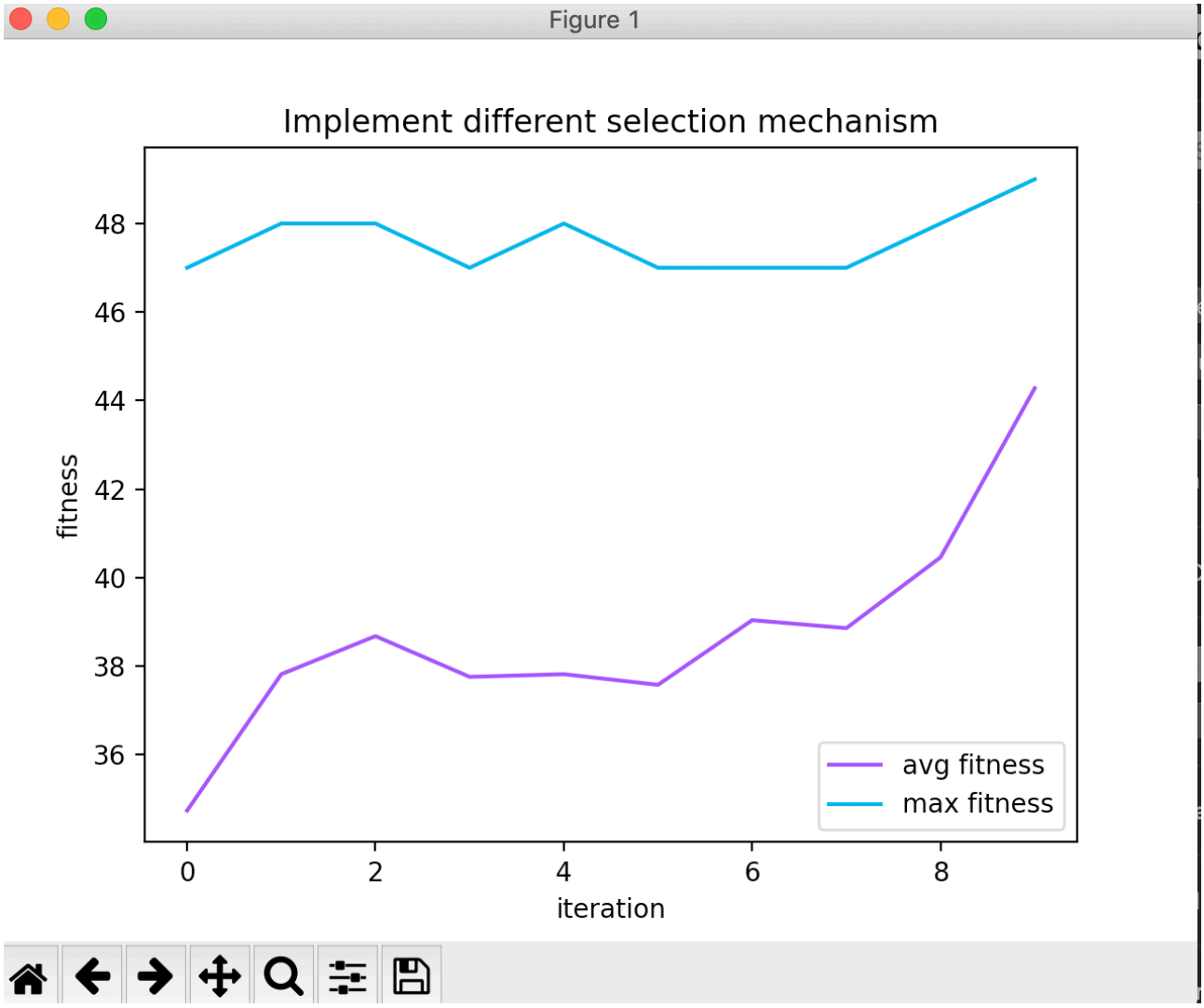
Random initialization

Population size: 50

Stochastic universal sampling selection

10 % elitism

mutation one point



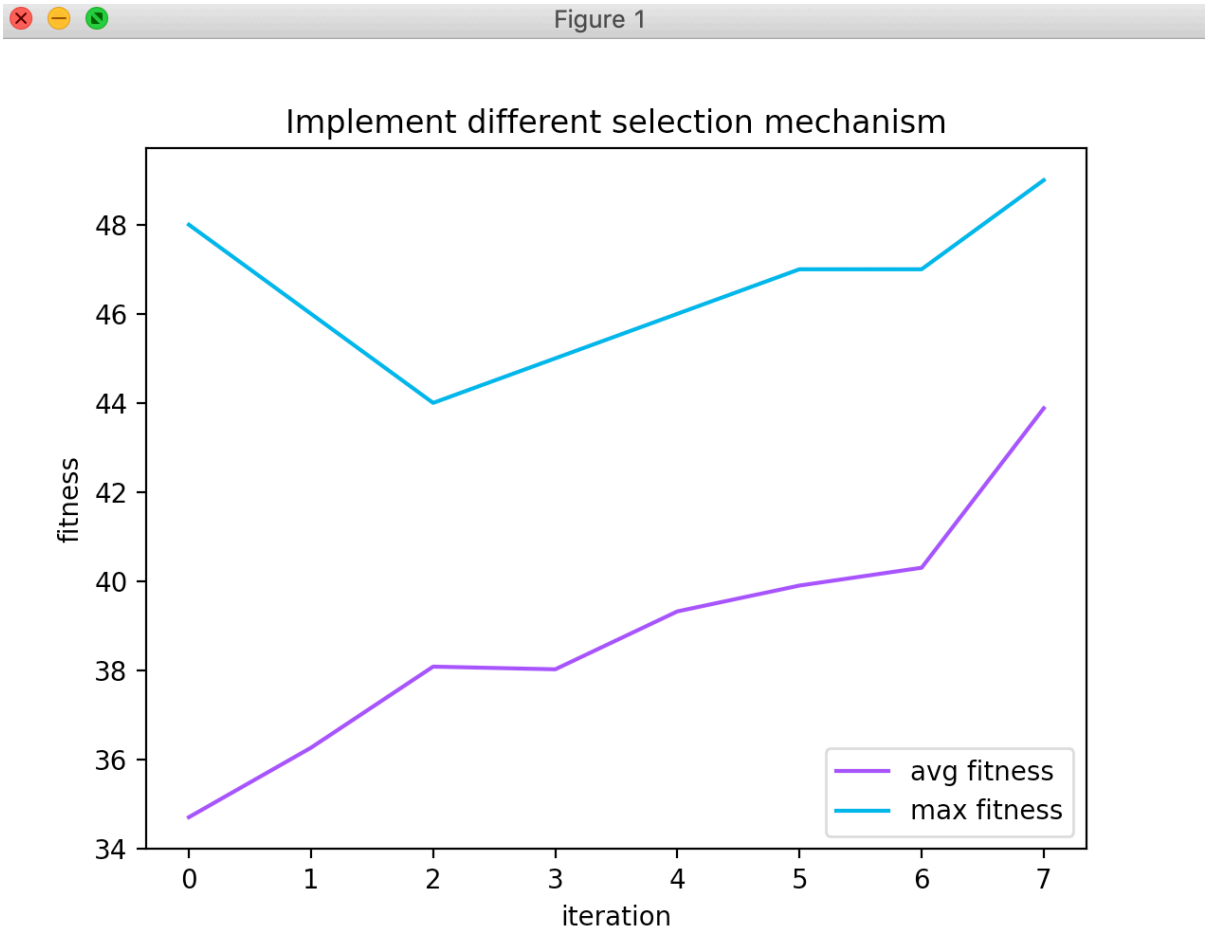


Figure 1
Implement different selection mechanism
fitness
iteration
avg fitness
max fitness
34
36
38
40
42
44
46
48
0
1
2
3
4
5
6
7

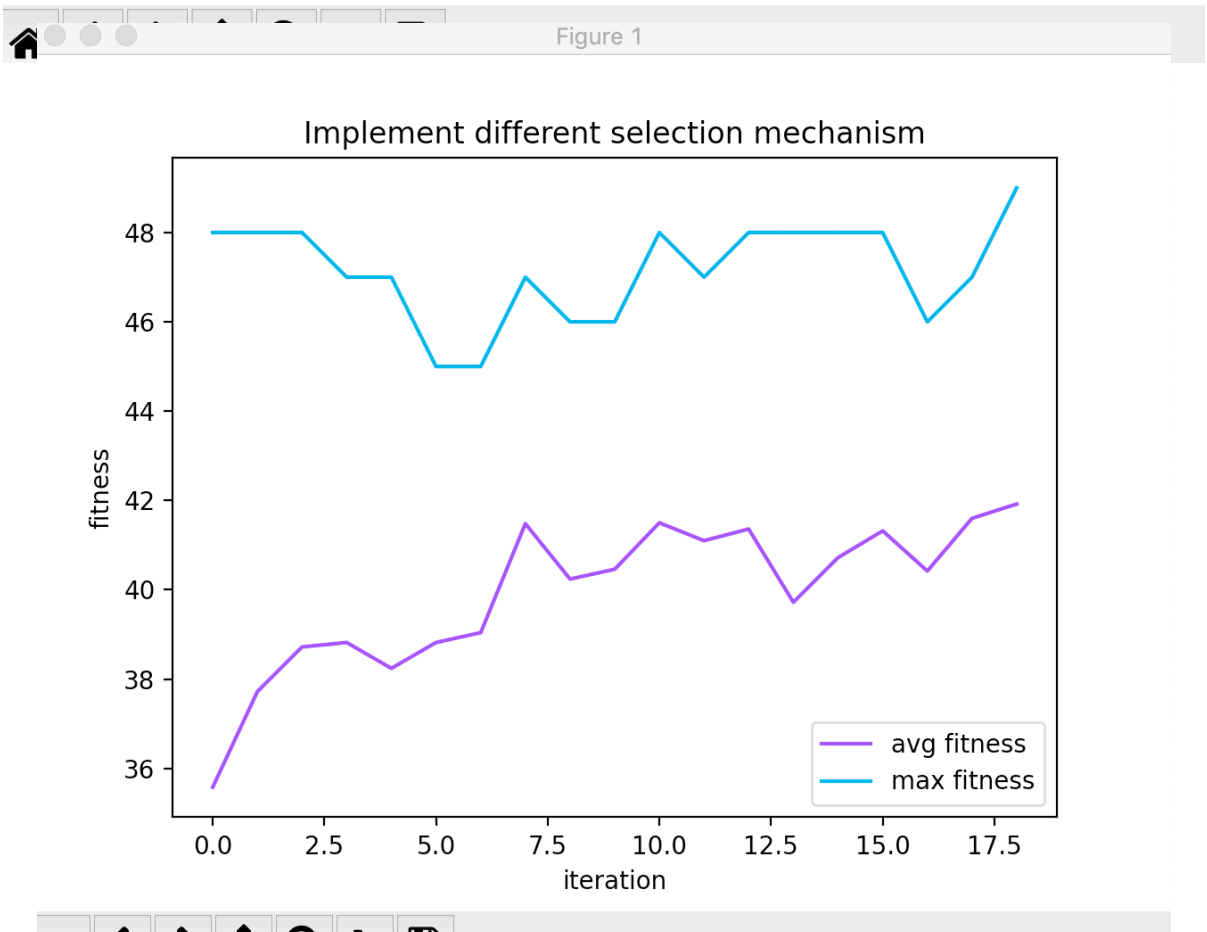


Figure 1
Implement different selection mechanism
fitness
iteration
avg fitness
max fitness
36
38
40
42
44
46
48
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

1.best strategy is [2, 9, 1, 8]

How does the best strategy work?

it has two 9 and 8, so two win with high probability. 1 and 2 has small chance of wining too.

2.best strategy is [1, 9, 5, 5]

How does the best strategy work?

it has one 9, so one win with high probability. two 5 which has good chance of wining. and has very small chance of wining too.

3.best strategy is [0, 7, 8, 5]

How does the best strategy work?

Colonel knows that it's not possible to win all 4 battles, so he decides to lose one battle with minimum number of solders(0), so he can use more solders for 3 other battles and increase his wining chance, average solders on each battle is 5, colonel has two 8, 7, so he wins this two battle with high probability and 5 has good chance of wining.

Case4:

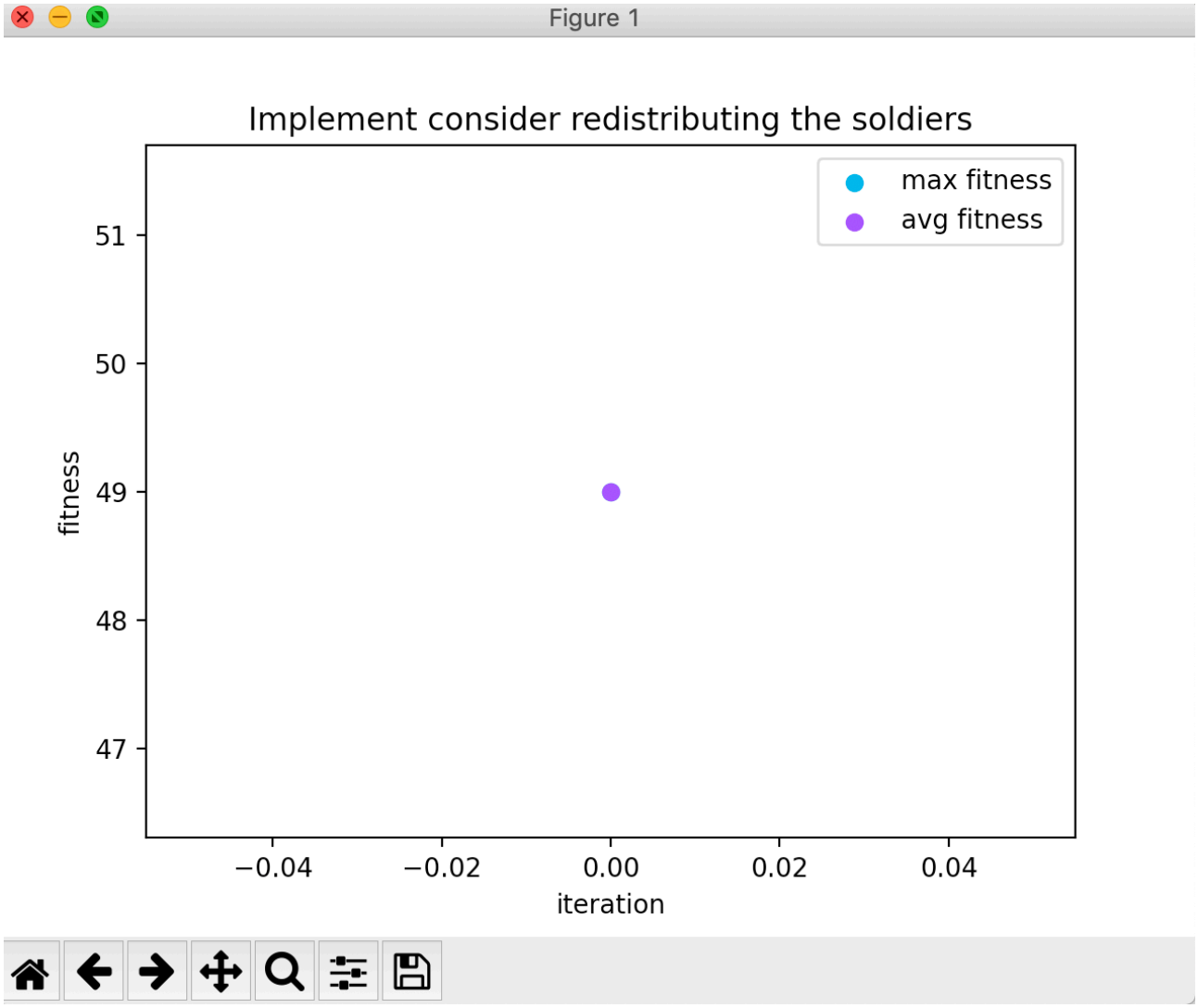
Random initialization

Population size: 50

Stochastic universal sampling selection

10 % elitism

different fitness function considering redistributing the soldiers



in this case each strategy win or tie because it uses its solders wisely

How does the best strategy work?

for example:

c1 [9, 8, 0, 3]

c2 [6, 9, 3, 2]

battle 1: c1 uses 7 solders versus 6 solders from c2. 2 solders remained.

c1[0] = 7

extra solders = 2

score1 = 2

battle 2: 2 solders remained but [2/3] = 0. 3 battles remained and we should redistributing solders in all succeeding battles evenly. he'll use them in 3rd and 4th battle. c1[1] doesn't change and tie this battle

c[1] = 8

extra solders = 2

score1 = 3

battle3: 2 solders remained and should distribute evenly between 3rd and 4th battle [2/2] = 1. so c1[2] get 1 solders but can't win.

c[2] = 1

extra solders = 1

score1 = 3

battle4: 1 solders remained so c1[3] = 4

extra solders = 0

score1 = 5

c1 : [9, 8, 0, 3] => [7, 8, 1, 4]

Eventually c1 wins against c2 and get 1 point.

example2:

c1[19, 0, 1, 0] (chance of losing is high, because of 0,1,0)

c2[13, 0, 7, 0]

c1 => [14, 1, 3, 2] (because of using extra solders, it became flexible base on c2, and win or tie with c2)

same thing happened when c1 has war with 48 other colonels.

so every colonel who use this method (redistributing extra solders) can win or tie war against all 48 other colonels. and get 49 fitness.