CT421 Assignment 2

Leo Chui (20343266), Aoife Mulligan (20307646)

GitHub Repo

N.B There are 70 images in the appendix beginning at page 24, hence the 93 page count.

Development Approach

Brainstorming

Our development process began by discussing the assignment. We first discussed the assignment to see each other's interpretation regarding the assignment and brainstorm to come up with how to approach part 1 and ideas for part 2, and collected all of our ideas into a text file to look back at during development.

After a discussion regarding multi-agent systems and graph colouring, and how the two can come together, we decided to take an OOP approach, by creating a `Node` class that represents the agents.

Structure of Code

Our solutions for the assignment are split based on the problems. Both Python modules contain mostly the same code:

- a definition for a `Node` class, which includes various OOP-related functions, colour-changing functions, `communicate()`, and object functions like `__eq__()` and `__hash__()`
- `generate_graph()`, a graph generator function that generates either a trivial, 4-node, 4-edge graph if no parameters are passed in, or a Small-World graph with n nodes and m edges
- `minimum_colours()`, a function to determine the theoretical minimum chromatic colour of a graph
- `node_fitness()` and `fitness_function`, first of which calculates conflicts on the individual level, and the latter is a fitness function that calculates the number of conflicts in the entire graph
- `init_nodes()`, a function that gives graph a random colouring from the global list of colours, gives all nodes a list of colours that is `min_colour_num` long, and updates all nodes with their neighbours
- `algorithm()`, the main function for both problems. It calculates the fitness of the graph at any given iteration, checks whether it has converged or not, plots the graph, and gets all nodes to communicate with their neighbours.

The basic idea is to use `Networkx`, the Python package for everything networks science and graphs, to create graphs and populate them with instances of the `Node` class, which will be our agents.

Node

All nodes maintain two lists, `neighbours`, which are nodes they are connected to via an edge in the graph, and `known_colours`, a list of colours that is sufficiently sized to reach a minimum graph colouring.

In order for `Networkx` to be able to create graphs using our custom `Node` class, it implements some functions that serves no direct purpose for the assignment (__eq__, __hash__). The class then has various getters and setters, for variables such as its colour and neighbours.

In the class, the main function of interest in the class is `communicate()`. General idea of the function is that the node will check for colour conflicts with all of its neighbours.

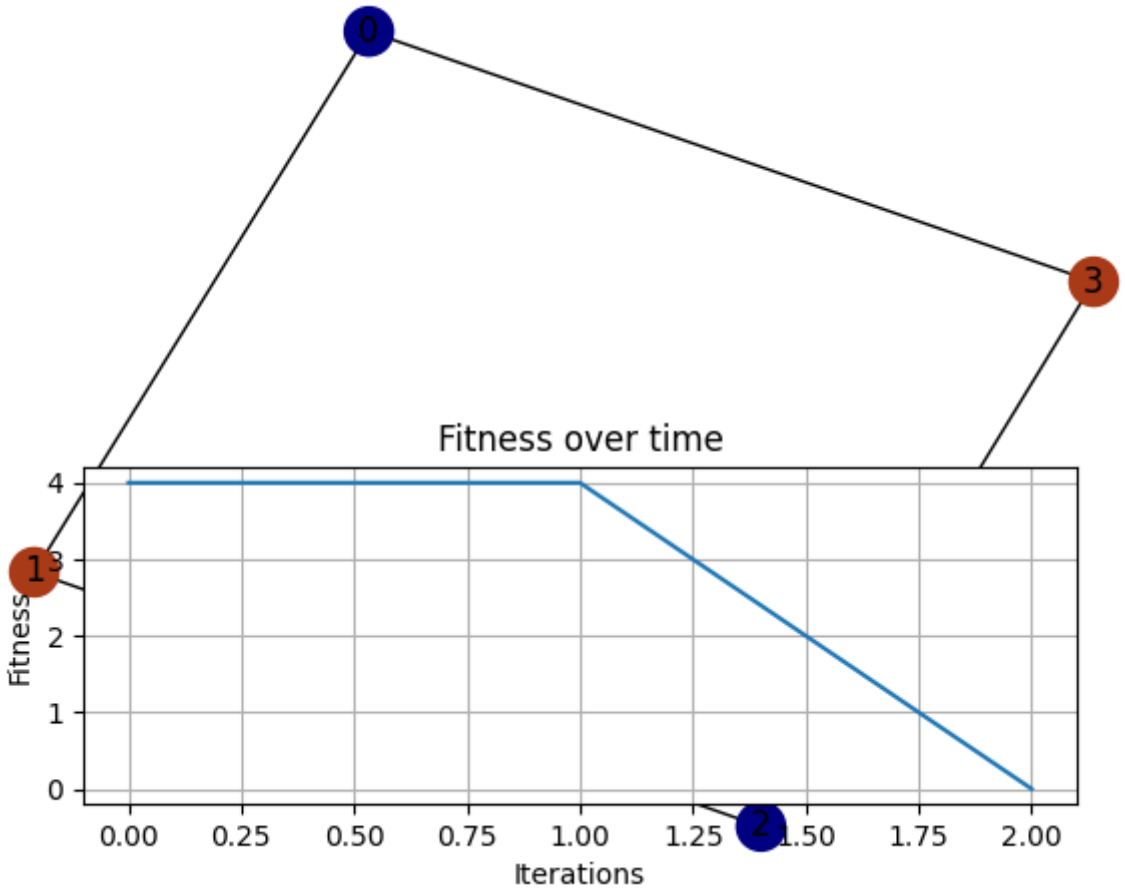
- In the case of no-conflicts, nothing occurs.
- In the case of a conflict, both nodes will check for their respective unavailable and available colours
  - `unavailable-colours`, list of colours of the neighbours
  - `available_colours`, list of colours in the `known_colours` list that are not in `unavailable-colours`

Based on whether `available_colours` of either node 1 or 2 is empty, the node with the non-empty `available_colours` will change colour to one in that list, or both will change to a random colour in `known_colours`.

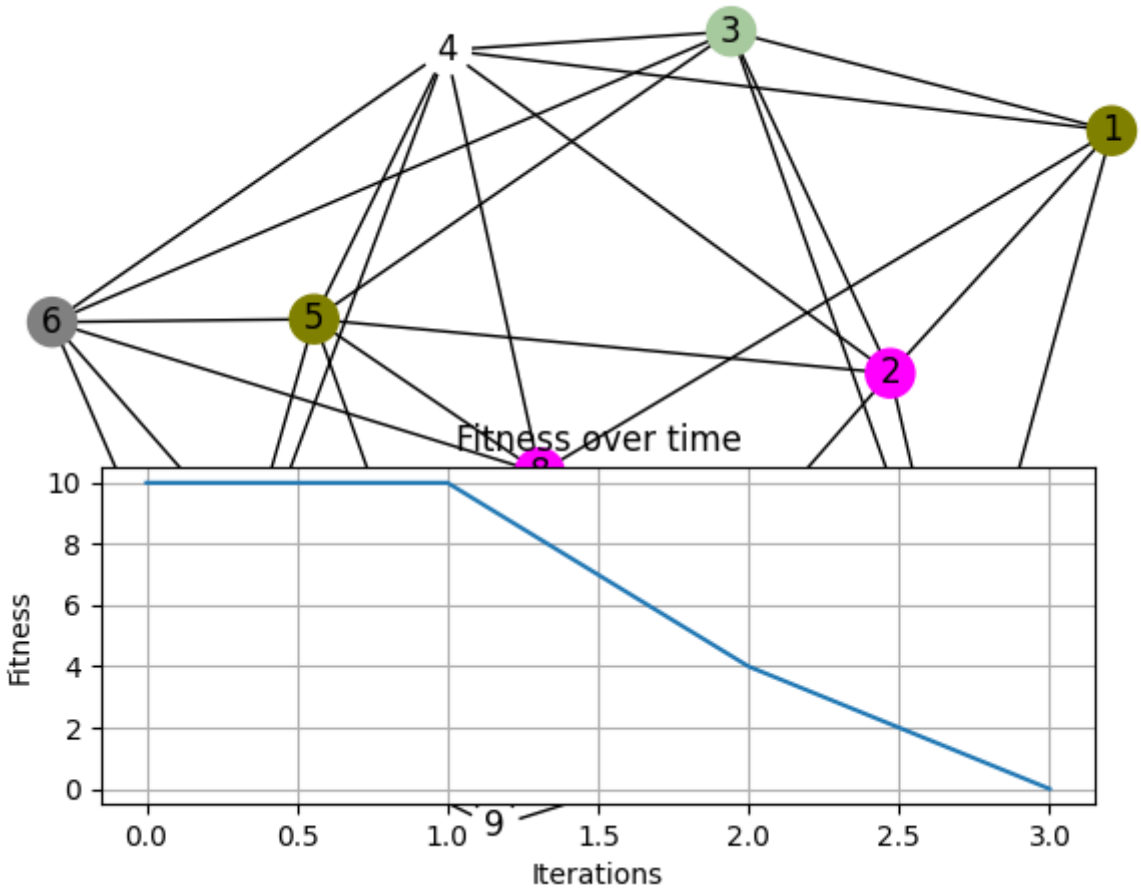
This is slightly different in problem 2, which will be discussed in further details in its own section.

Graphs

For both problems, we begin testing our the validity of our approach by using a small graph on 4 nodes, $1, 2, 3, \text{ and } 4$, and the edges list are as follows: $(1, 2), (2, 3), (3, 4), \text{ and } (4, 1)$. The minimum colouring for such a graph is 2.



The topology we chose was Small-World.

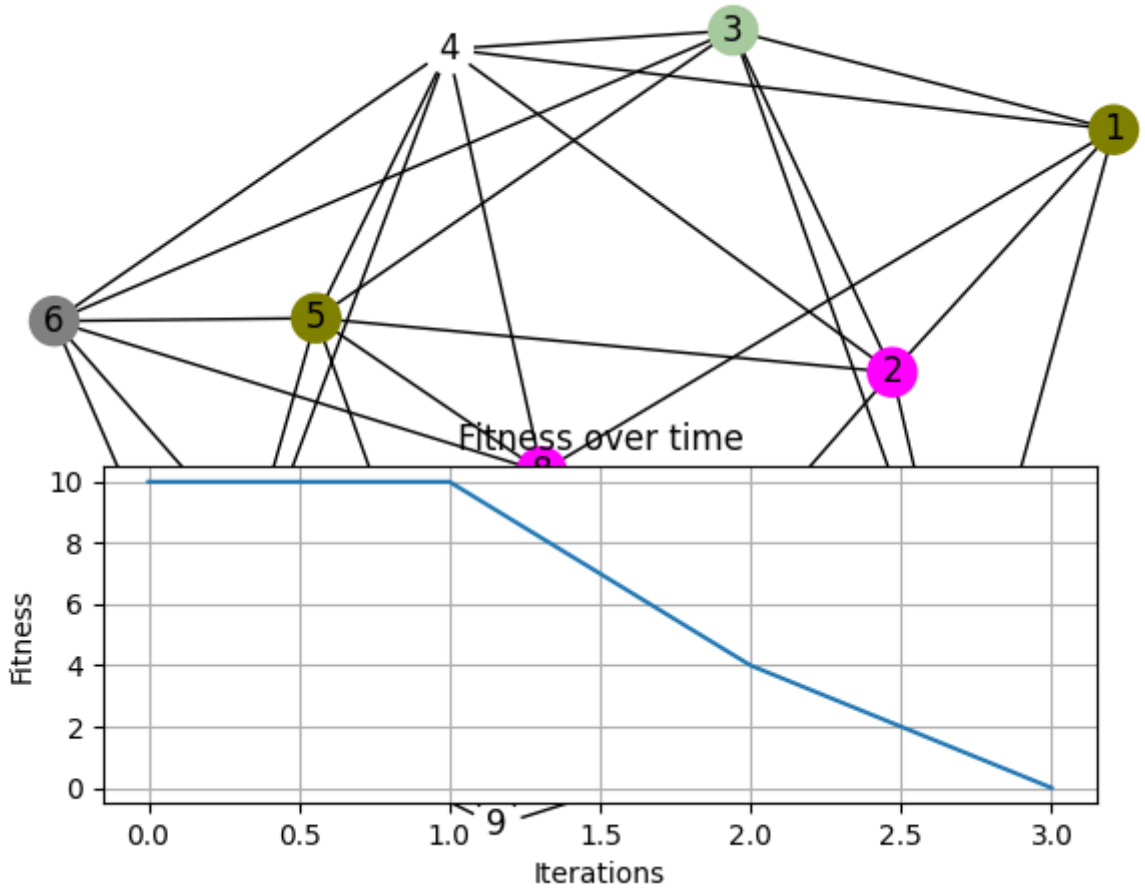

4
3
1
6
5
2
Fitness over time
10
8
6
4
2
0
Fitness
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations

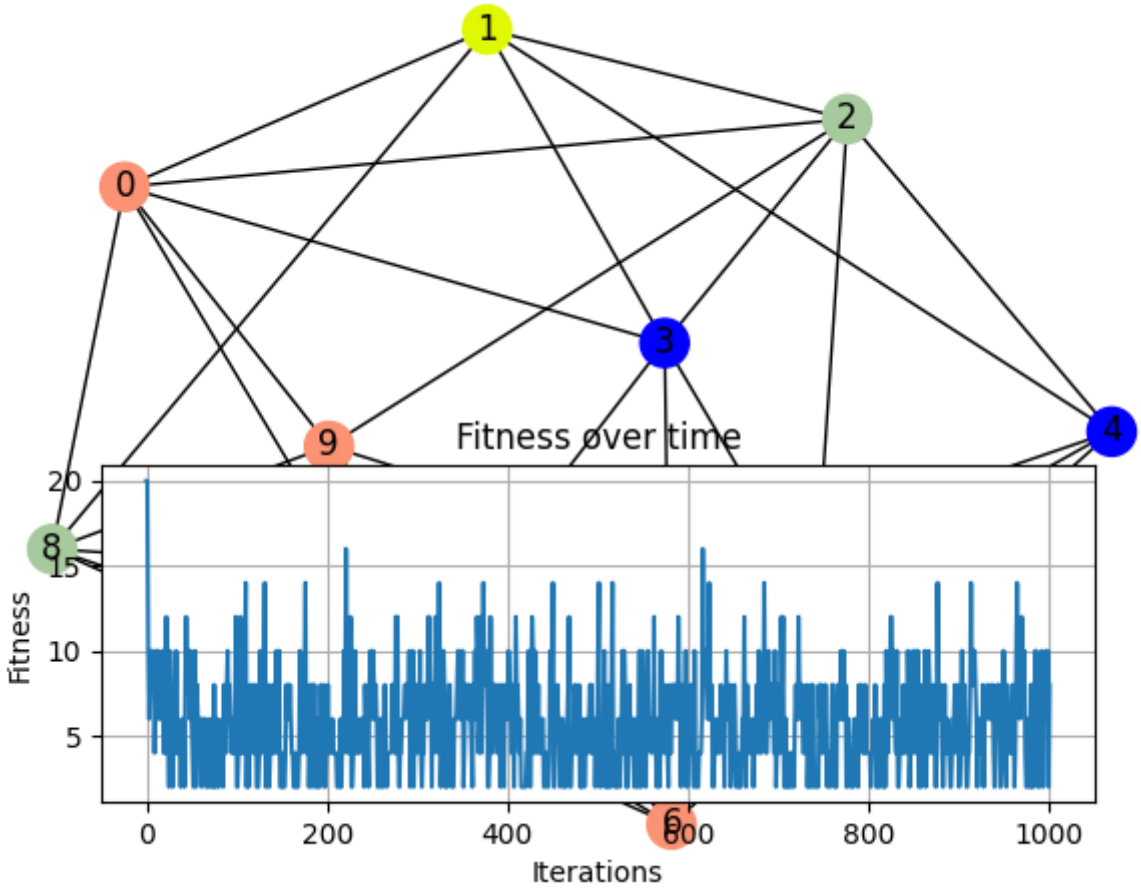
Part 1

27/30 converged runs suggests that the algorithm performs well. For the runs that did not converge, our theory is that the initialisation of the graph may sometimes mislead and give the graph a local optima of colouring, as during development, we had great difficulty reaching convergence even for a trivial 4-node graph when we randomly coloured the graph with only the minimum amount of colours needed.

10 Node 6 Degree

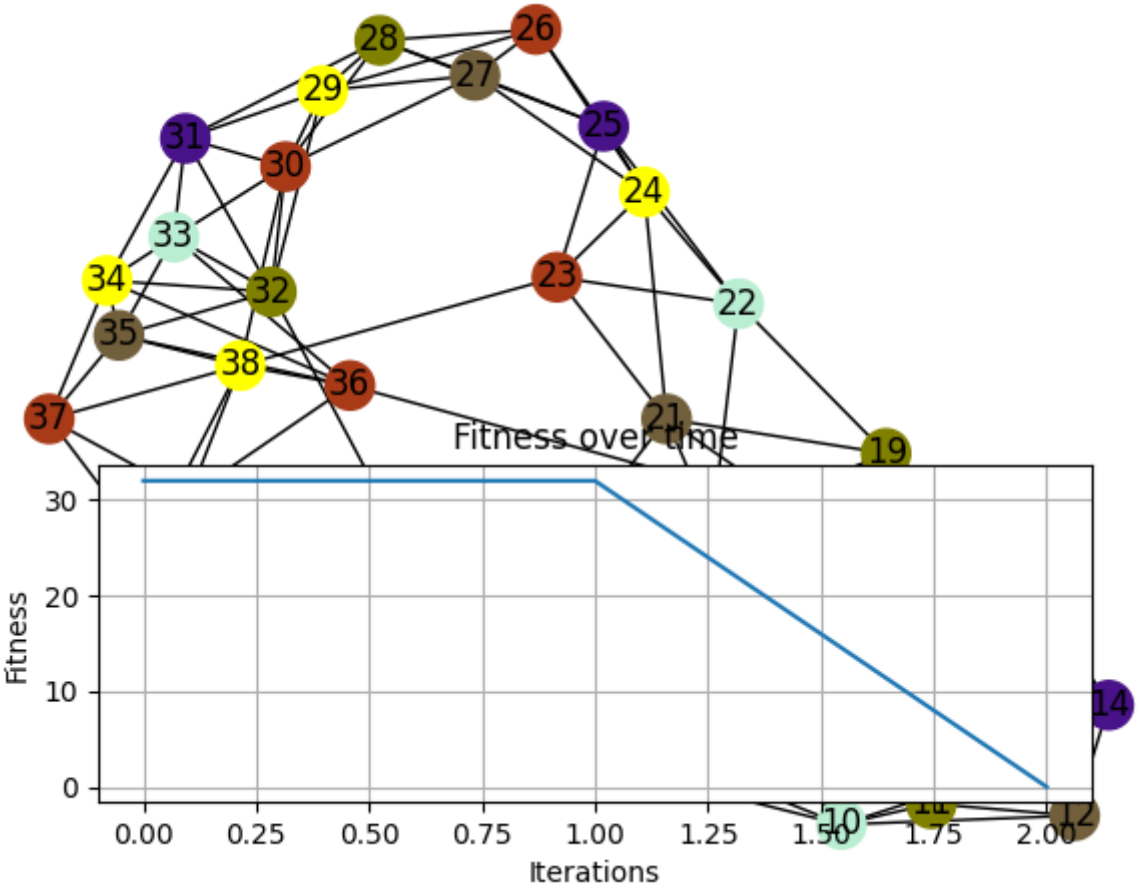
For a small-world graph with 10 nodes and 6 degrees, it reaches a minimum colouring 8 times out of 10. Below shows two runs for 1000 iterations, 1 success, 1 fail. More images can be found at the end of the report in the Appendix.

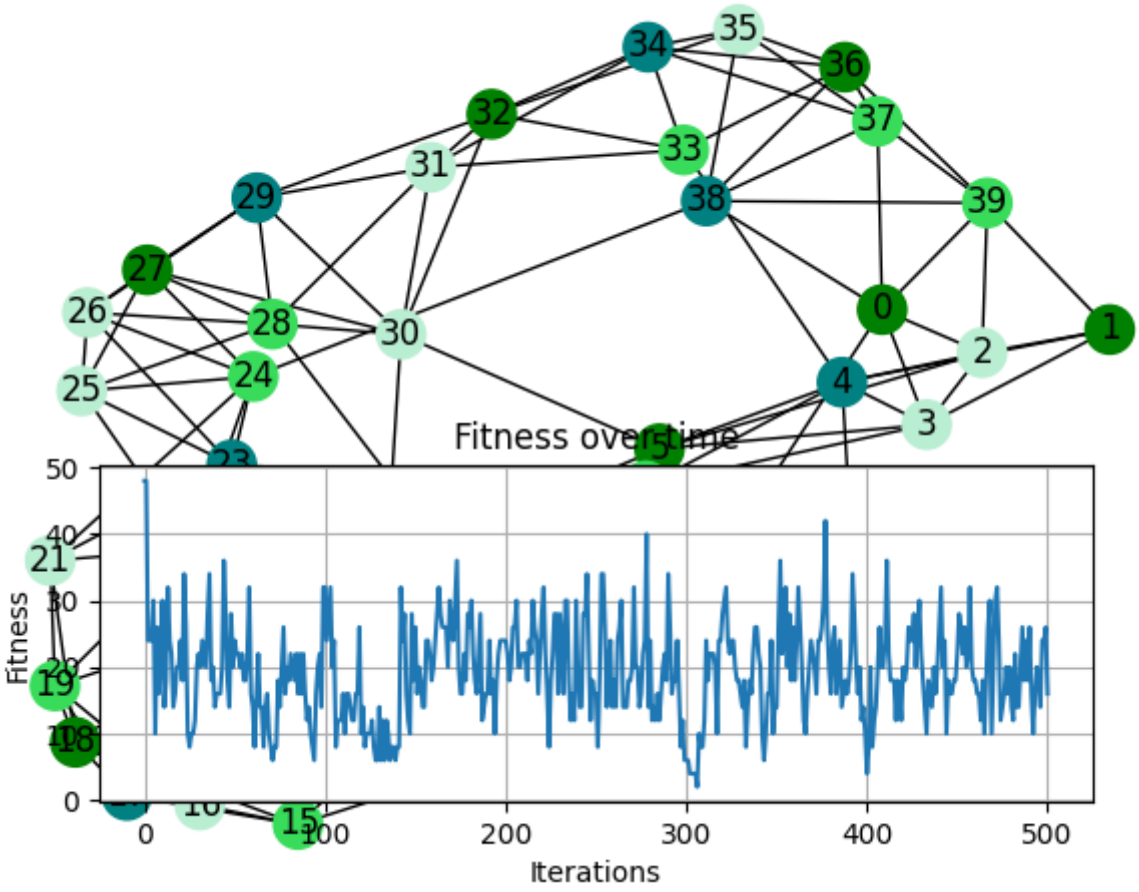




40 Node 6 Degree

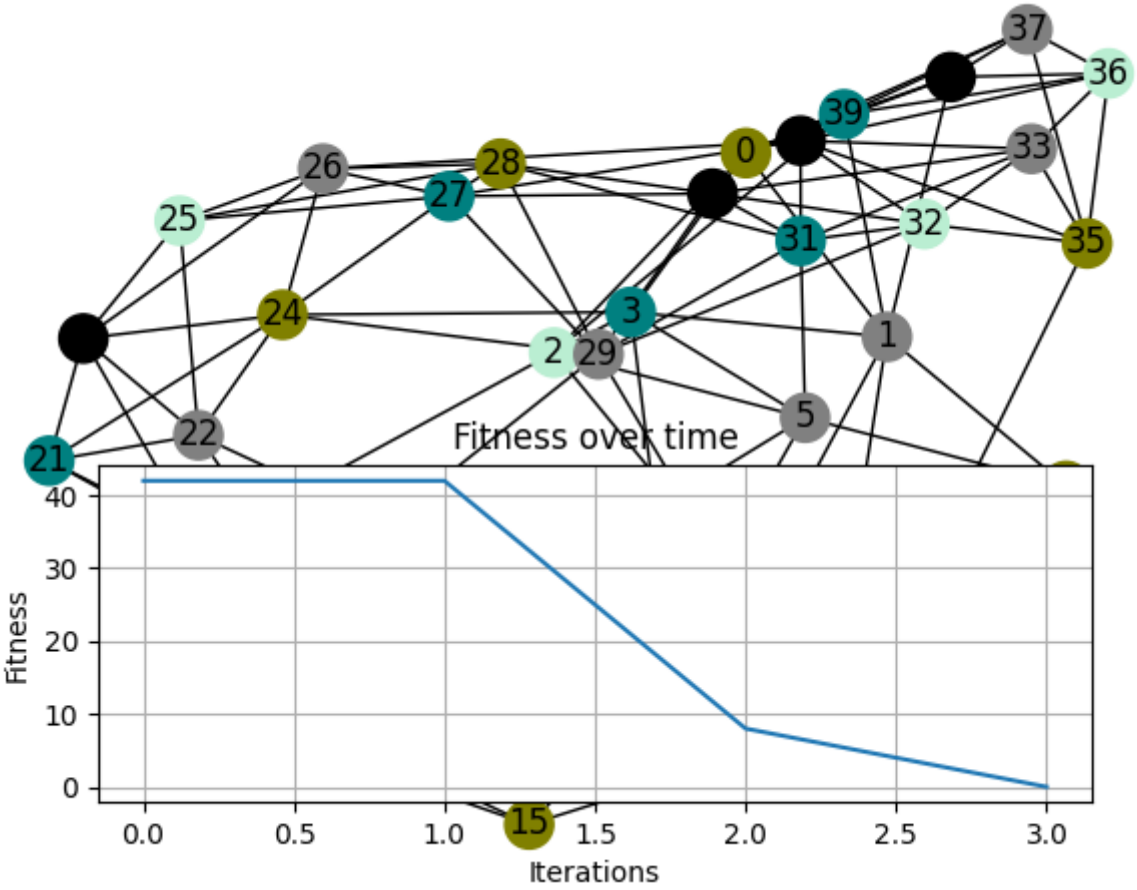
Part 1 solution performed slightly better over 10 runs, reaching convergence 9 times.

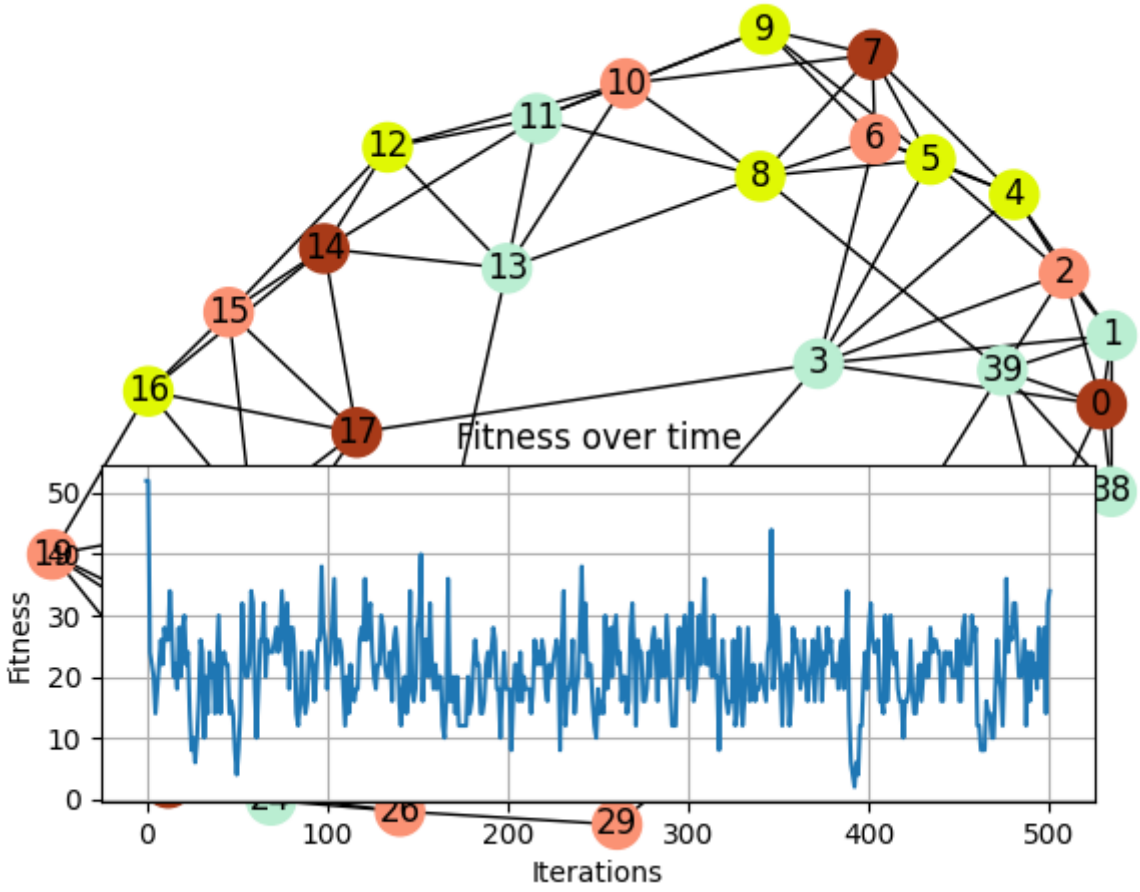

Fitness over time
Fitness
Iterations
0
10
20
30
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



40 Node 7 Degree

The solution again reached convergence 8 runs out of 10.


37
36
39
0
33
26
28
27
25
31
32
35
24
3
1
2
29
5
22
21
15
Fitness over time
Fitness
40
30
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations


Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0
100
200
300
400
500
9
7
10
11
12
6
5
8
4
14
13
2
15
1
3
39
16
0
17
38
26
29

Part 2

Changes to Communicate

We decided to explore what effects may be present if nodes were given "personalities". The inspiration came when the network was interpreted as a network of people, and that the colouring was some sort of "negotiation" process.

During our discussion for part 1, questions such as "should both nodes change colour at the same time?" or "which node would have precedence if they are not to change at the same time?" were raised, and an attempt to address those questions was to give nodes a personality.

A node can either be "Aggressive/Assertive" or "Passive". This is represented as an integer, either 0 representing *Passive*, or 1 representing *Aggressive*.

The `personality` affects the way nodes communicate, as thwe two types of personality has different "priority" when changing colours, and the new set of rules when communicating is listed below.

Assuming conflict, both check available colours and personalities.

If personalities clash (both aggressive or both passive):

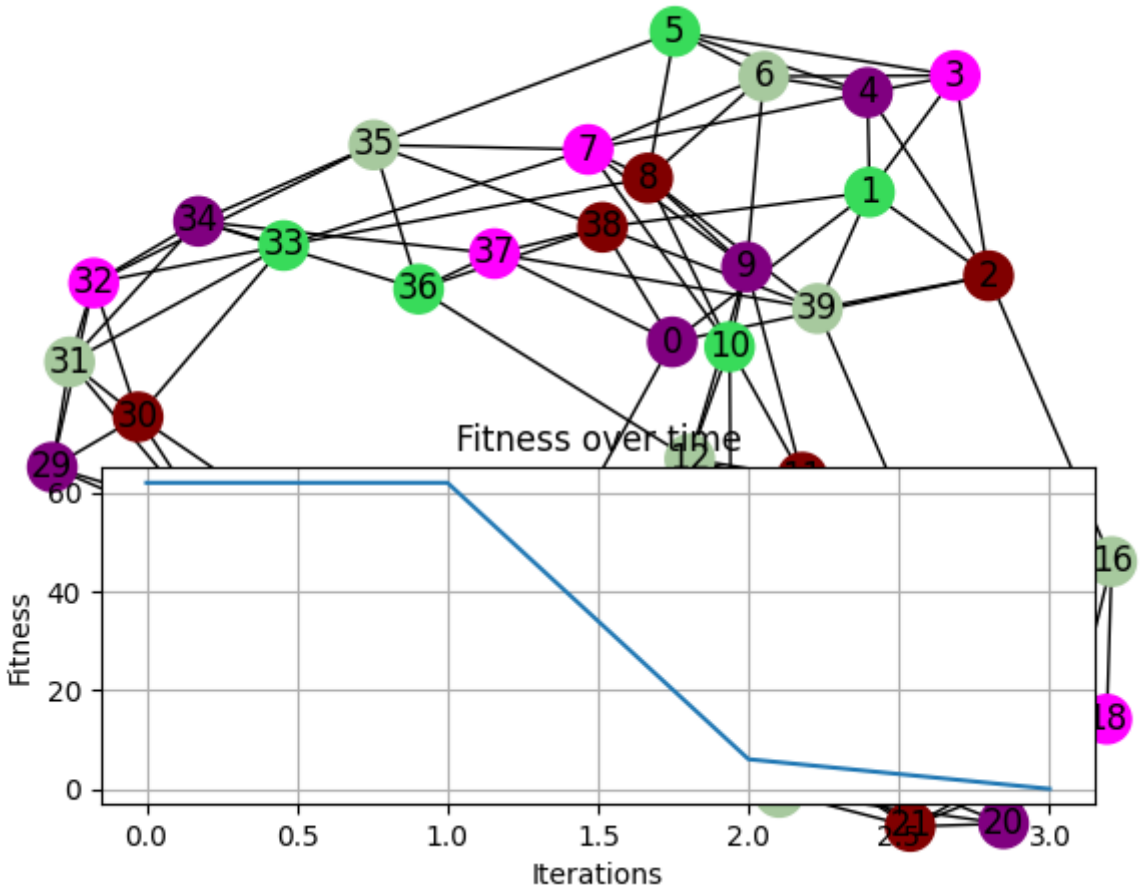
- if either node has available colours: the node with available colours will change
- else both change to random

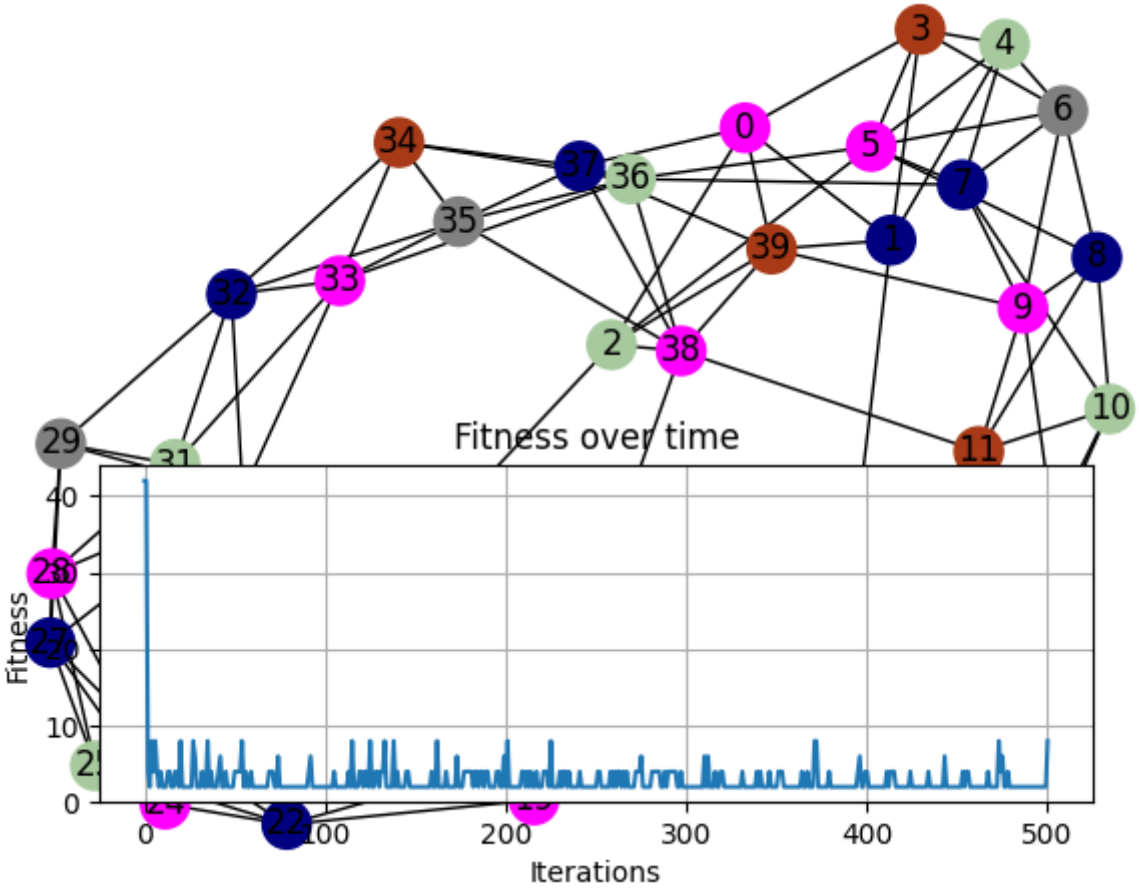
Else, the aggressive node will tell the passive node to either change to an available colour, or a random colour, depending on whether or not their lists of available colours are empty.

To test the approach, we decided to stick with the 40 node, 6 degree, small-world graph. The numbers were chosen arbitrarily. Below will compare the different percentages of aggressive nodes. Please note that the percentage of aggressive nodes are approximated, as the random function is used.

10% Aggressive

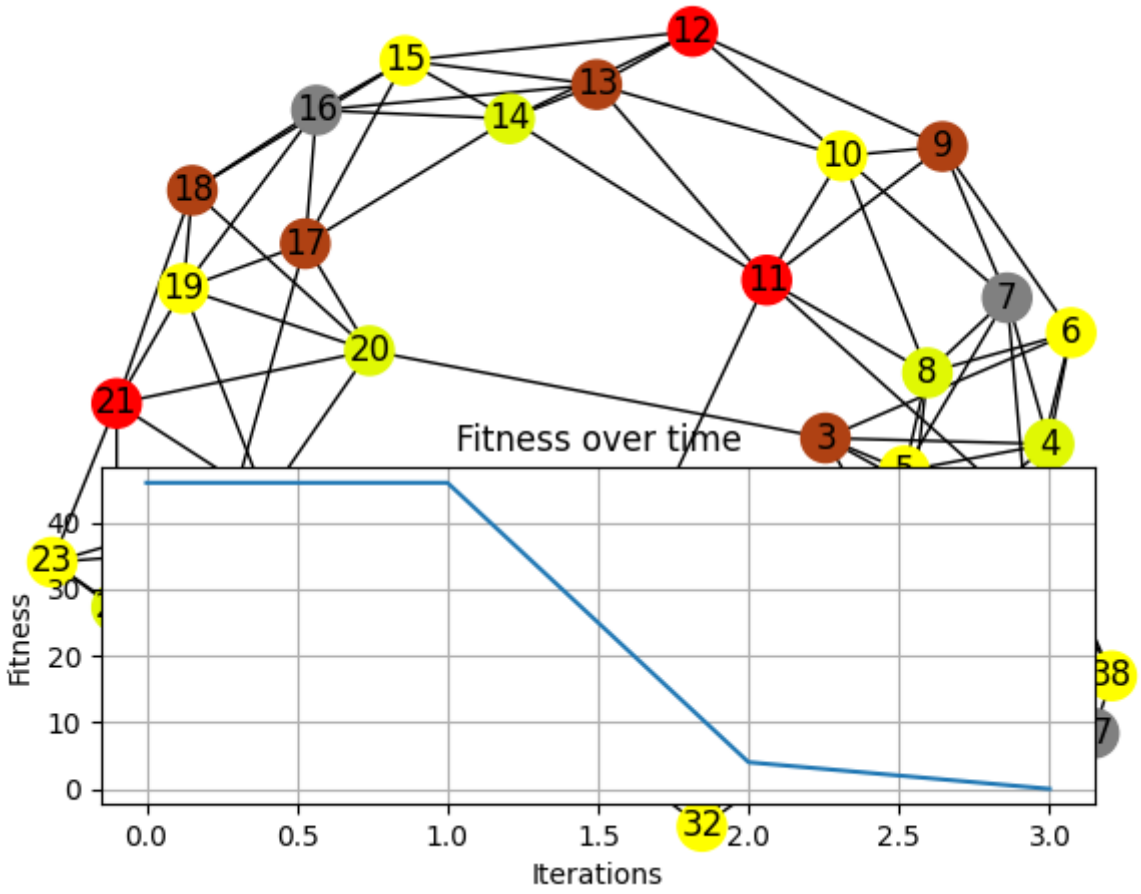
Below features two runs that are most representative of all runs. Runs either converge under 10 iterations, or never converge, only reaching a fitness that approaches convergence.

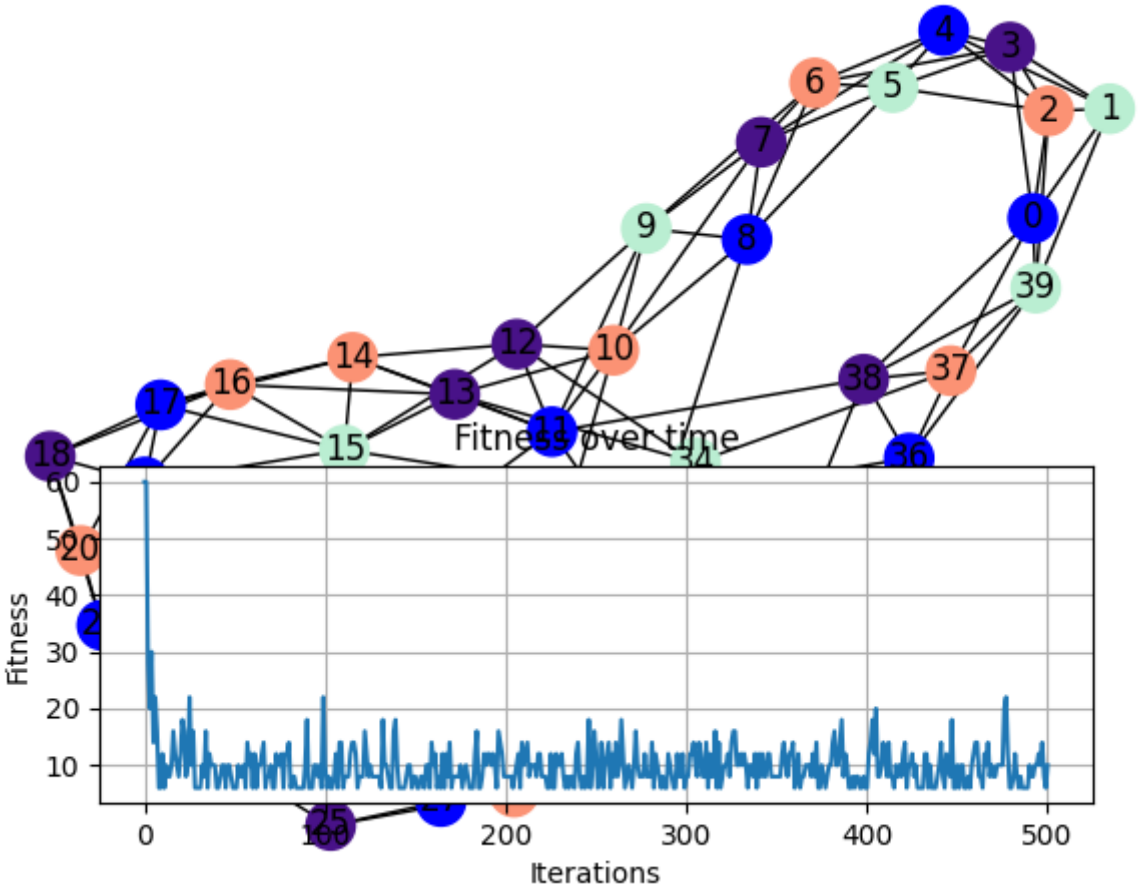

Fitness over time
Fitness
Iterations
0
20
40
60
0.0
0.5
1.0
1.5
2.0
3.0
5
6
3
4
35
7
8
1
34
33
38
37
9
2
32
36
39
0
10
31
30
29
16
18
20

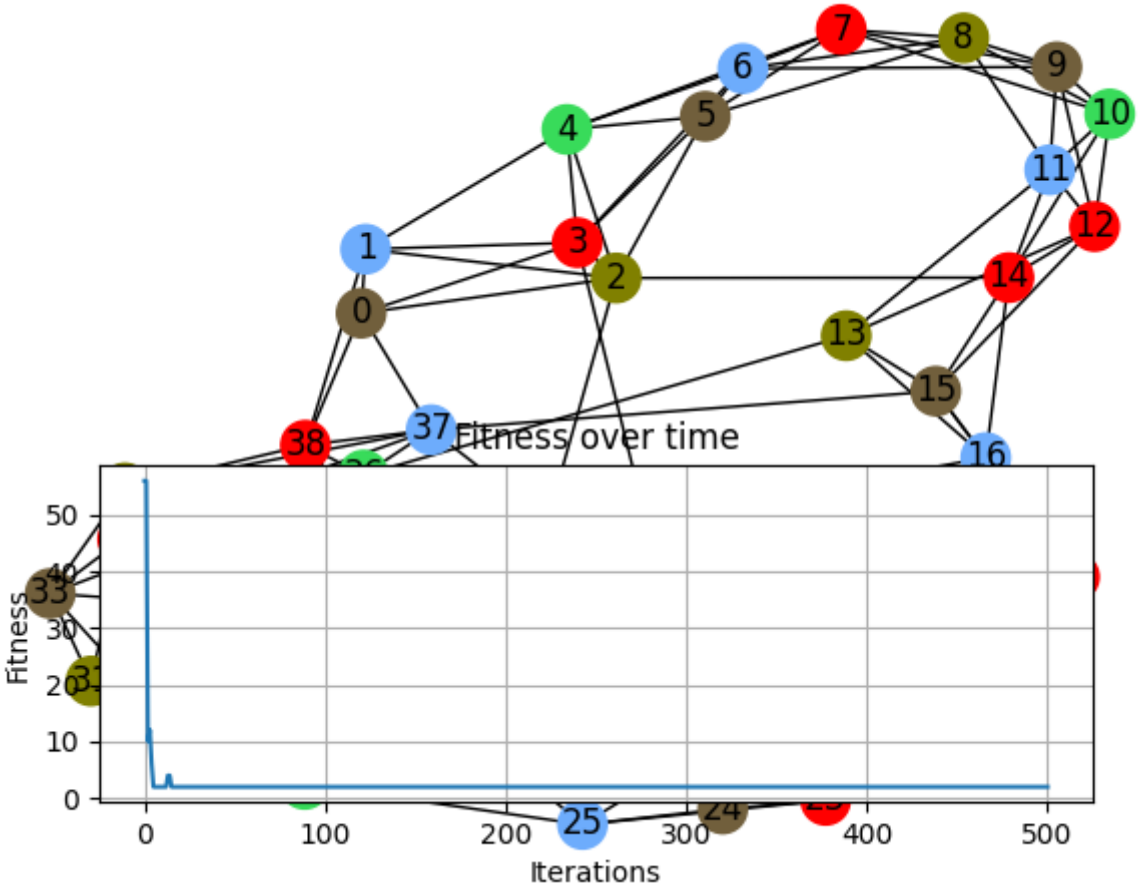


30% Aggressive

A similar trend to 10% aggressive, with a slight increase in instability on the non-convergence runs.

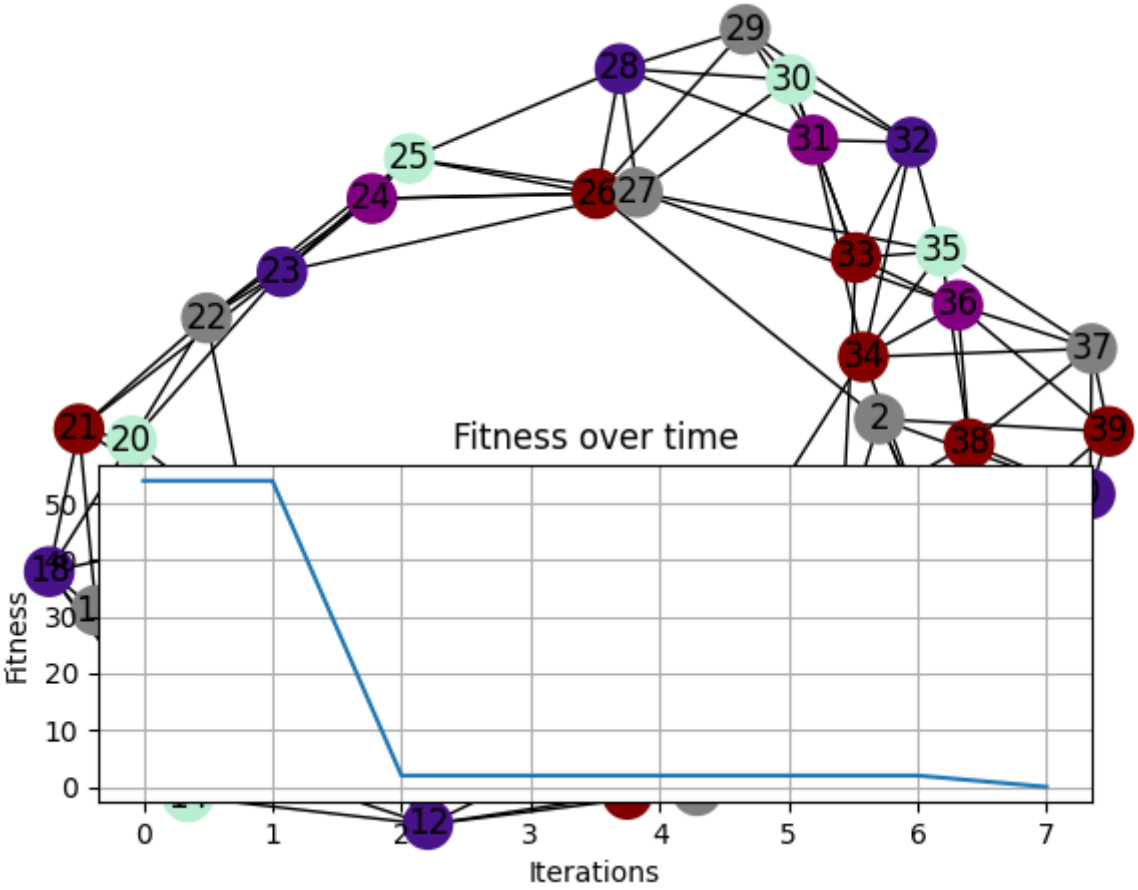

Fitness over time
Fitness
Iterations
0
10
20
30
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0

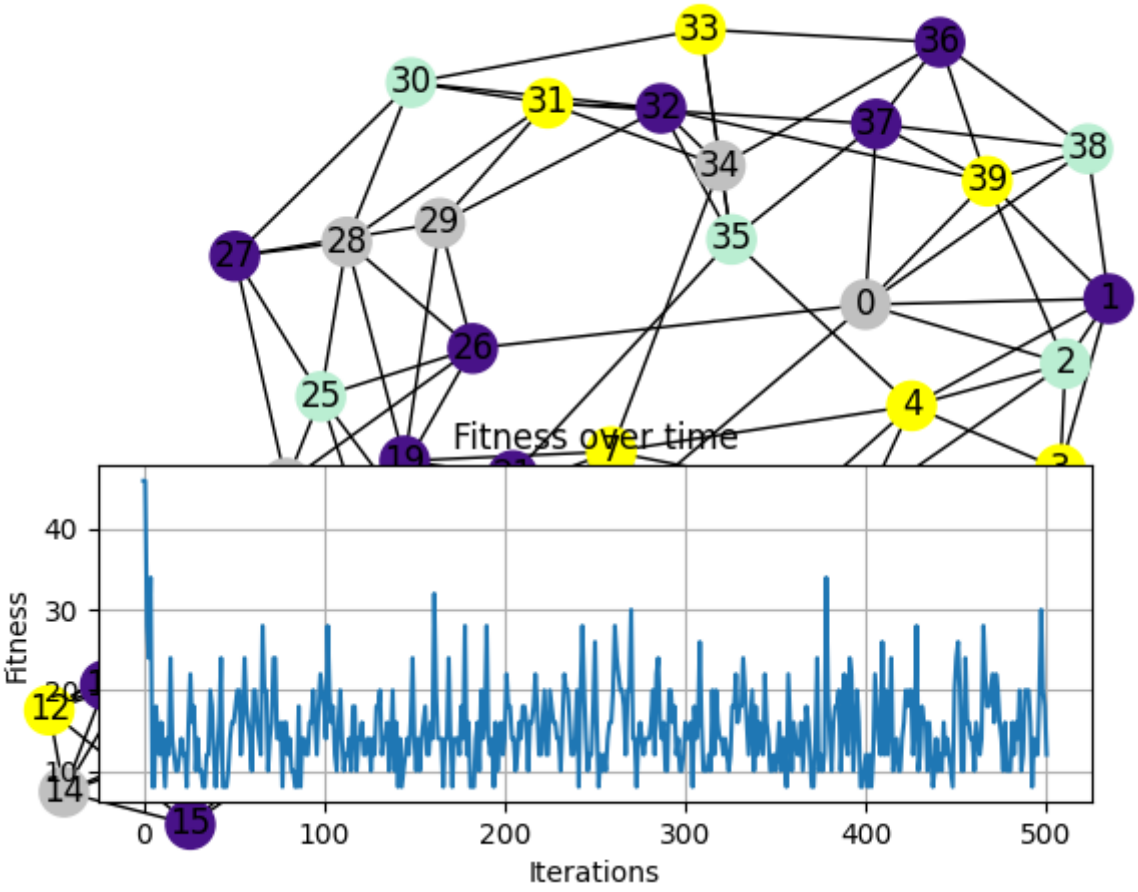

Fitness over time
Fitness
Iterations
0
200
300
400
500
10
20
30
40
50
60



50% Aggressive

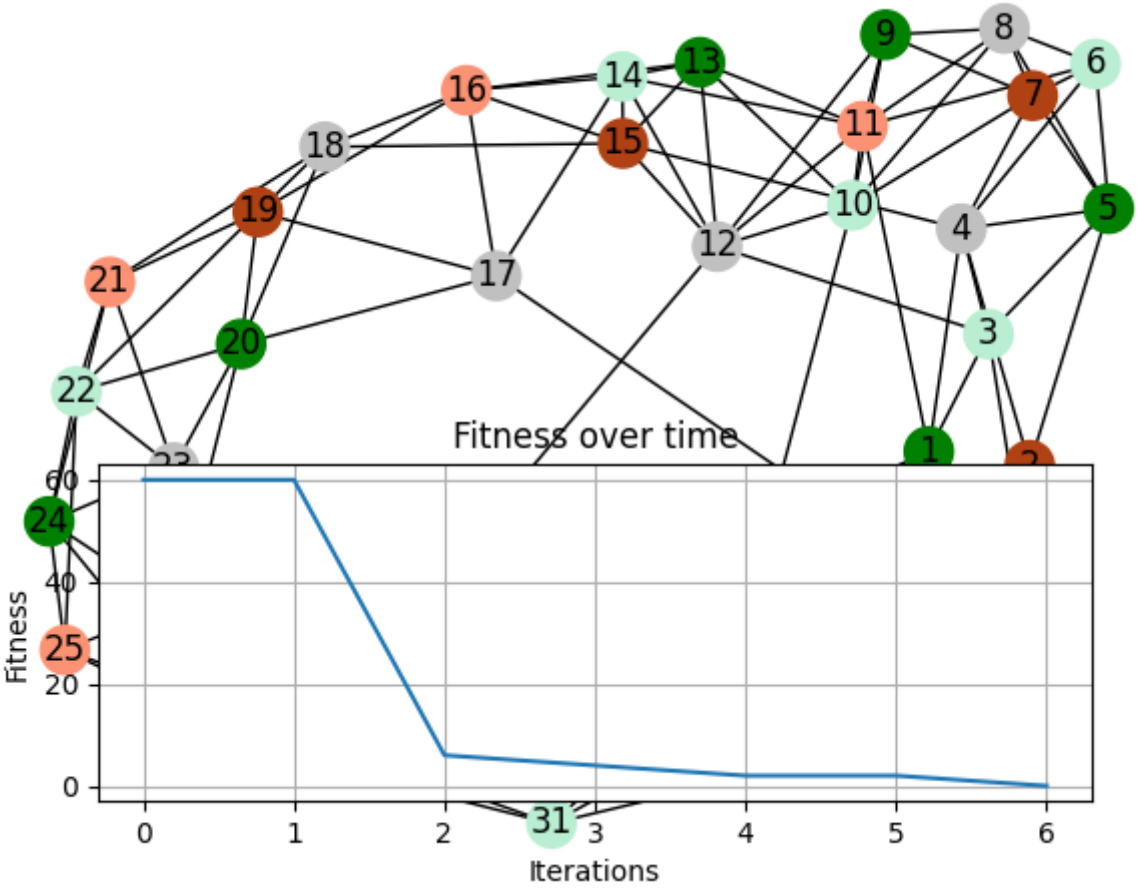
Similar to 30%, but more unstable.

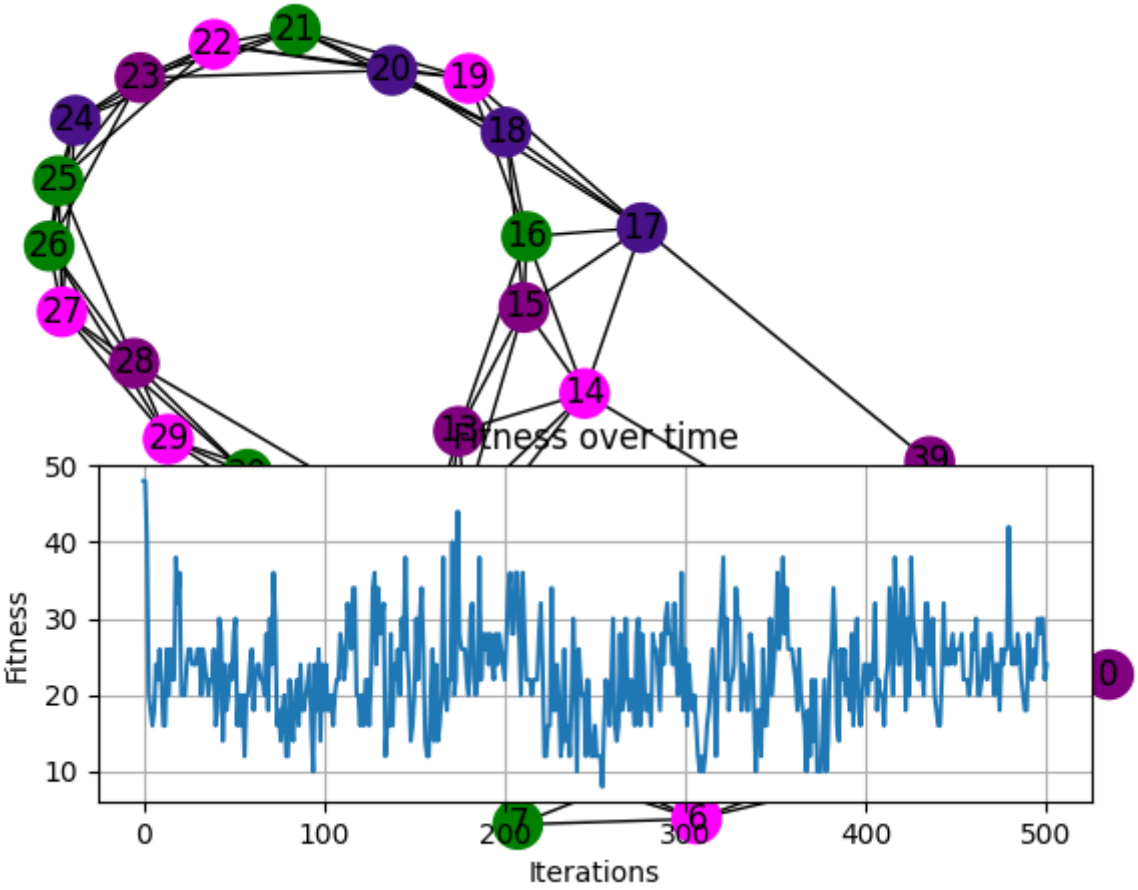

Fitness over time
Fitness
Iterations
0
10
20
30
50
0
1
2
3
4
5
6
7
29
28
30
31
32
25
24
26
27
33
35
23
36
22
37
34
21
20
2
38
39
18
12



70% Aggressive

Similar trend in fitness to the other splits, except even more unstable.


Fitness over time
Fitness
Iterations
0
20
40
60
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
24
25
31


Fitness over time
Fitness
Iterations
50
40
30
20
10
0
100
200
300
400
500

Discussion

Overcomplicated?

In hindsight, it is possible that we overcomplicated the assignment. It may have sufficed to leave part 1's communication mechanism as "both node change randomly on conflict", or something simple, along that line of thinking. It was what we previously had implemented, but later changed as we felt it was not enough.

Results

There are no significant, obvious change to the performance introduced by the concept of personality. This might in part be due to the fact that we possibly overcomplicated part 1, which is discussed in previous section.

On the non-convergence runs, part 2 does seem to perform better, on average reaching a lower conflict count. It also seems more stable than part 1, which does not seem to retain good colourings. This however may be an issue when the graph reaches a colouring that is an local optima.

Conclusion

Our solution to agent-based graph colouring relies on the communication mechanism built into our custom agents, which are the agents for our graphs.

In part 1, we described the approach in closer detail, and briefly listed some of the results, as well as explaining why we think the results came to be.

In part 2, we introduced the concept of "personalities" to nodes, where a node can either be aggressive or passive. We detailed the changes in communication rules based on personalities, and demonstrated the fitness progressions of some of the runs.

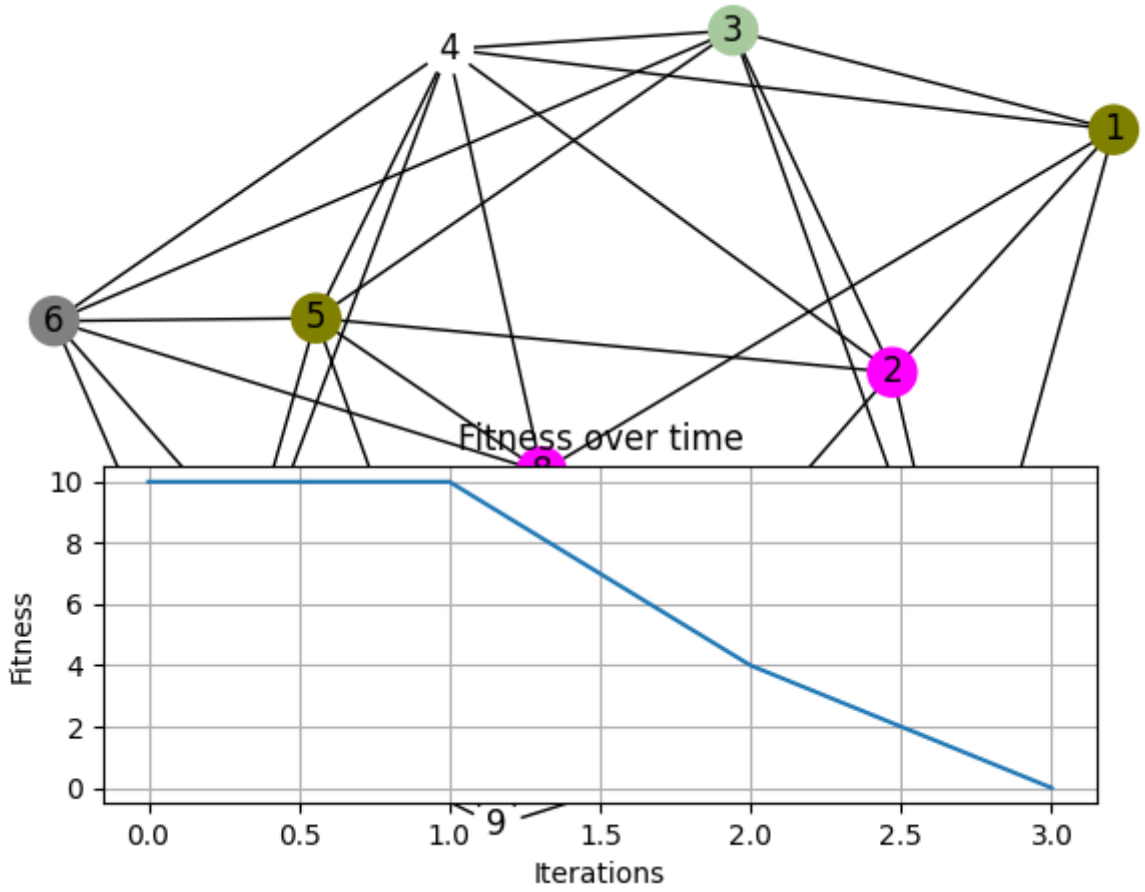
Lastly, we briefly discussed our thoughts on our approach.

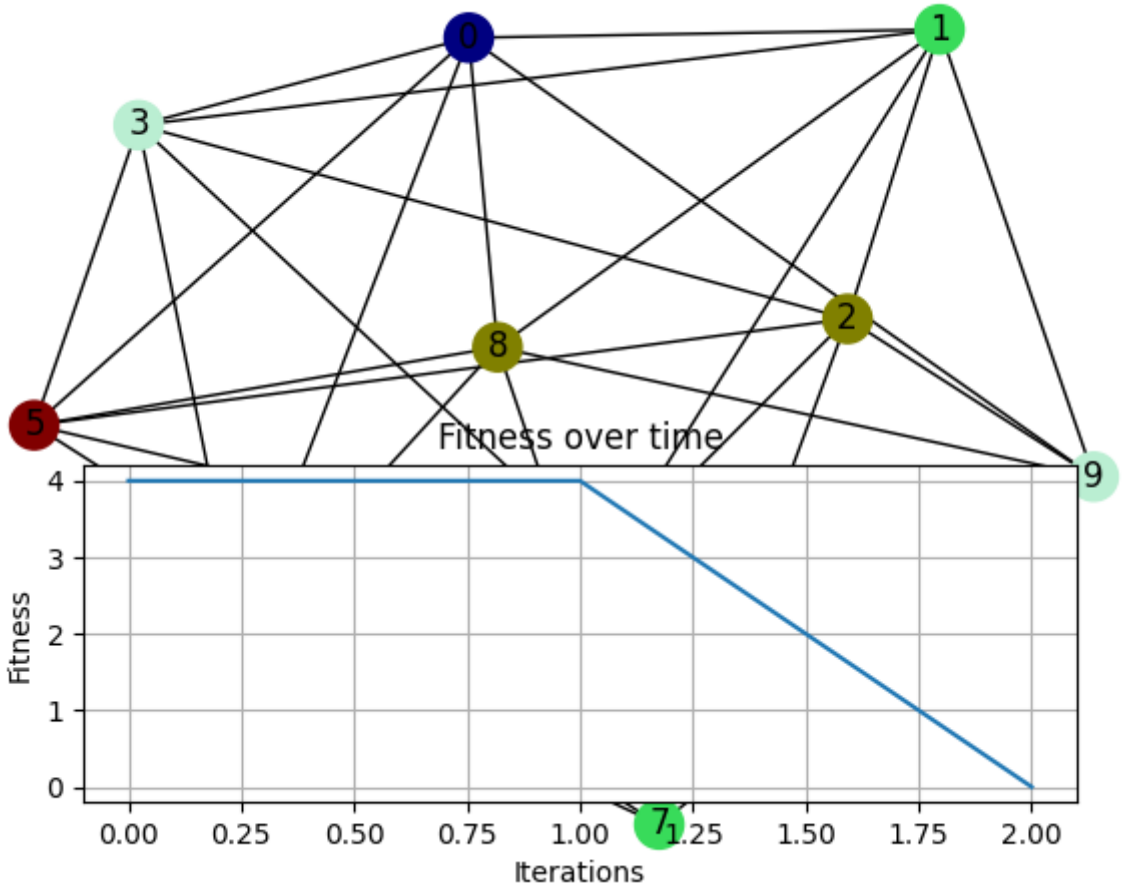
Fitness graphs of *all* runs can be found in the appendix.

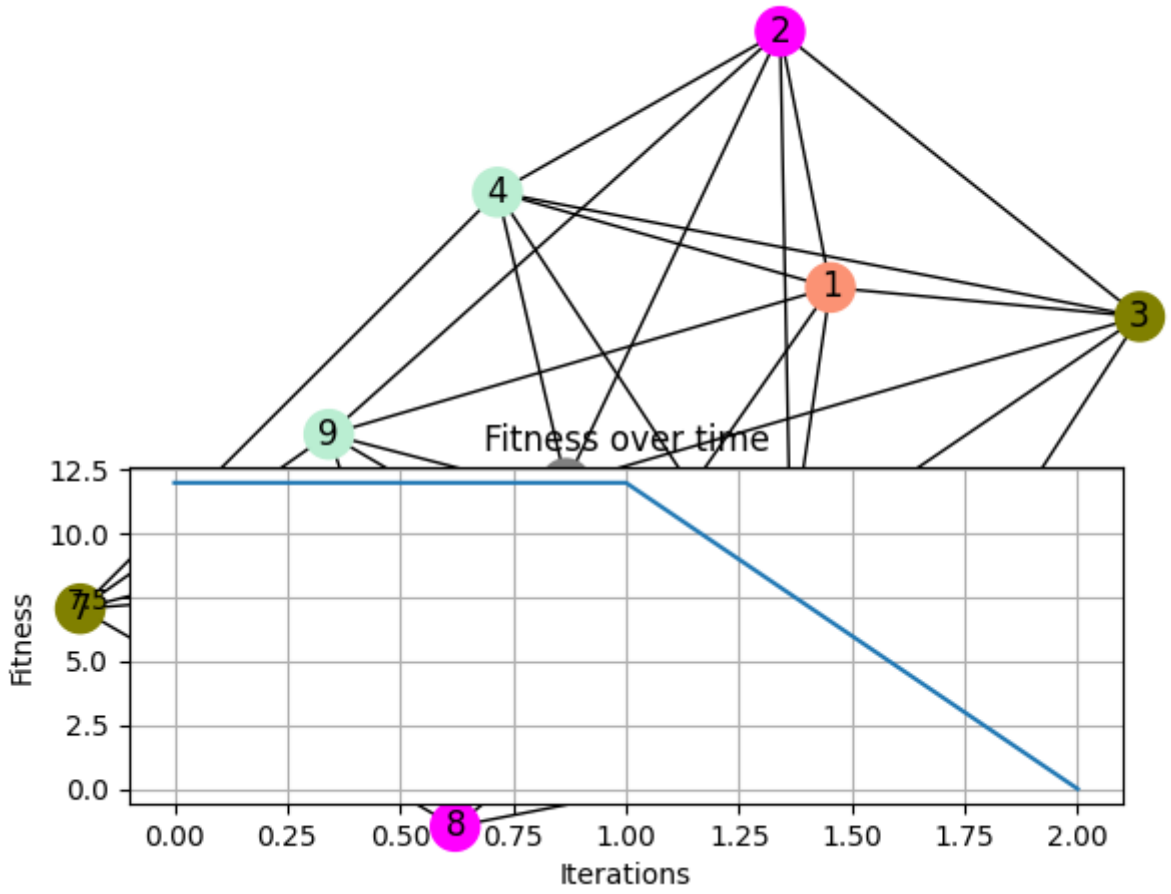
Appendix

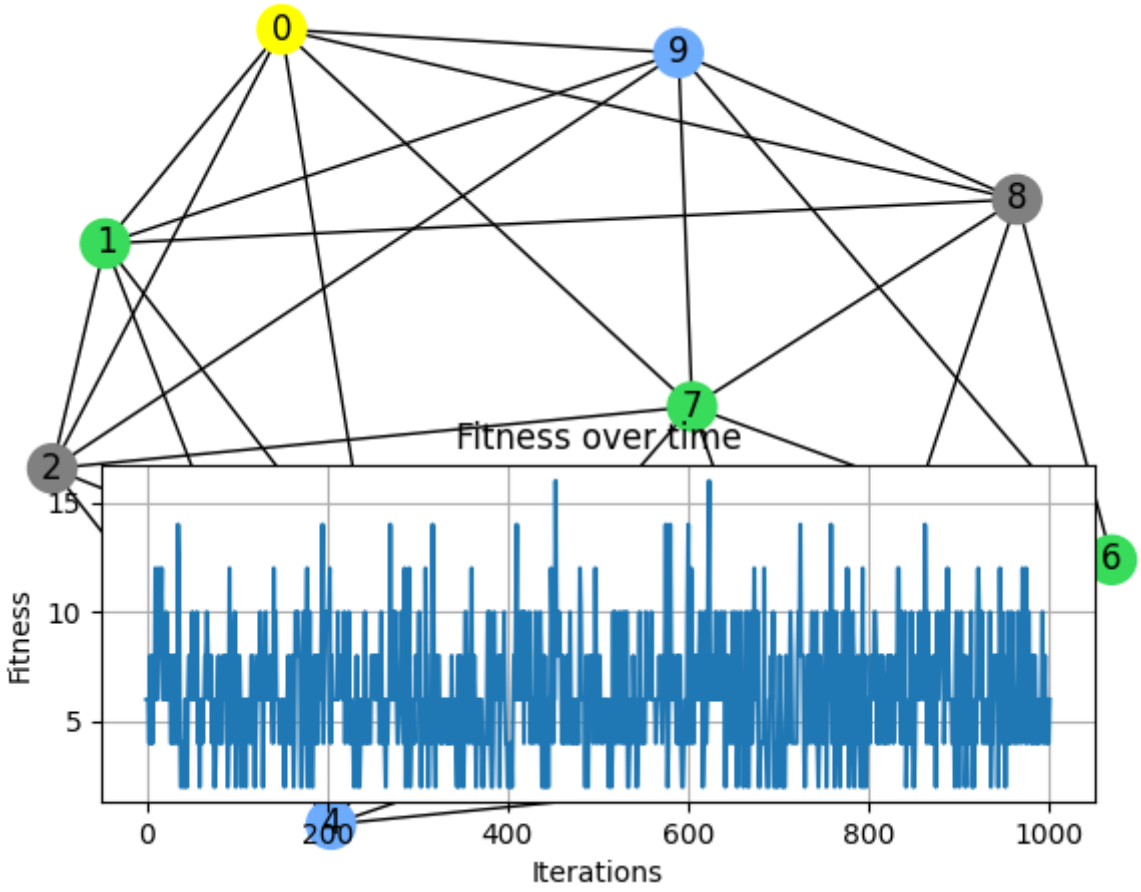
Problem 1

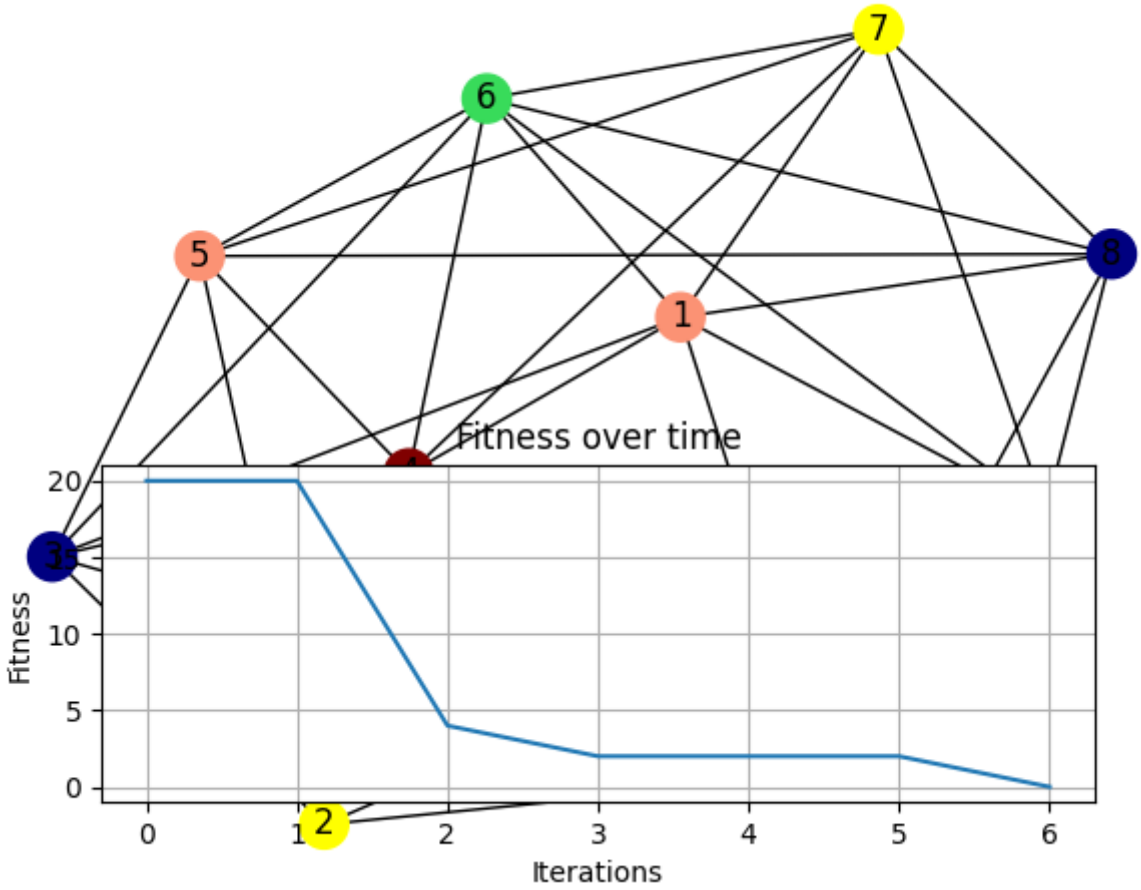
10 Node 6 Degree

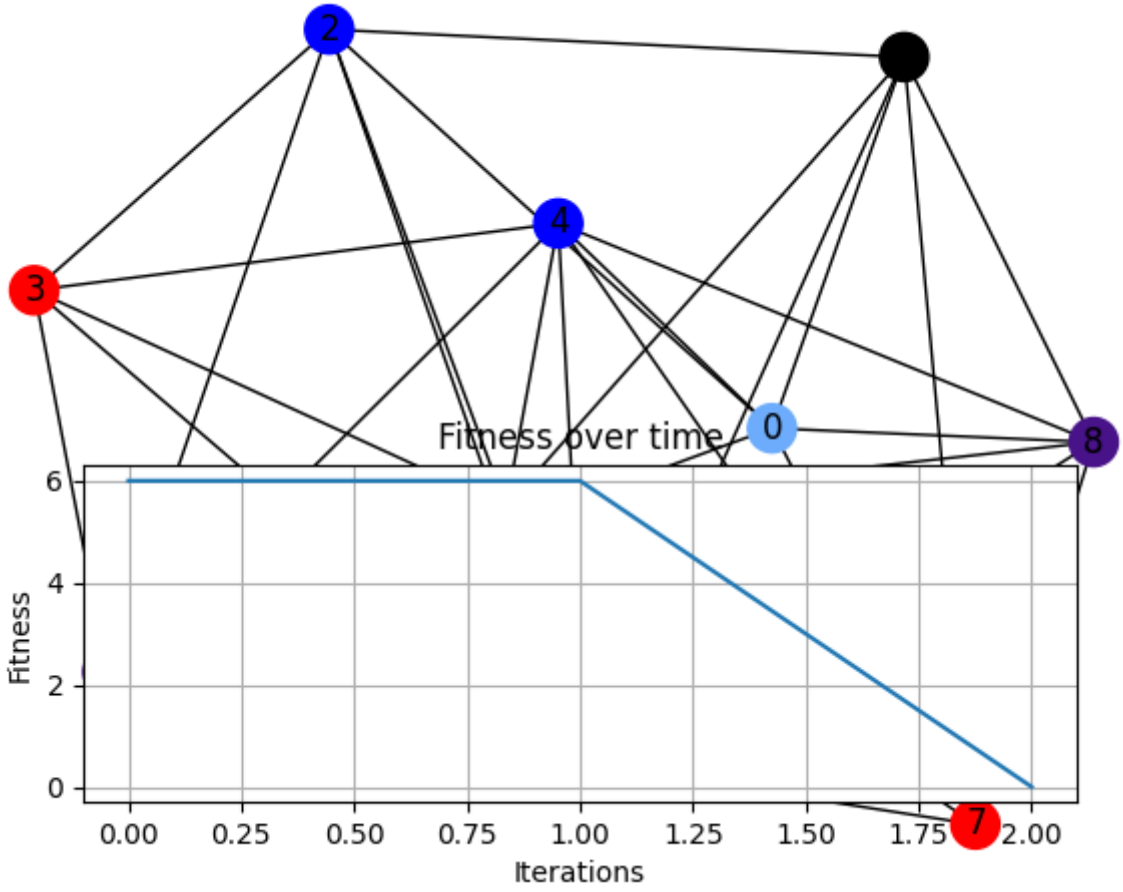


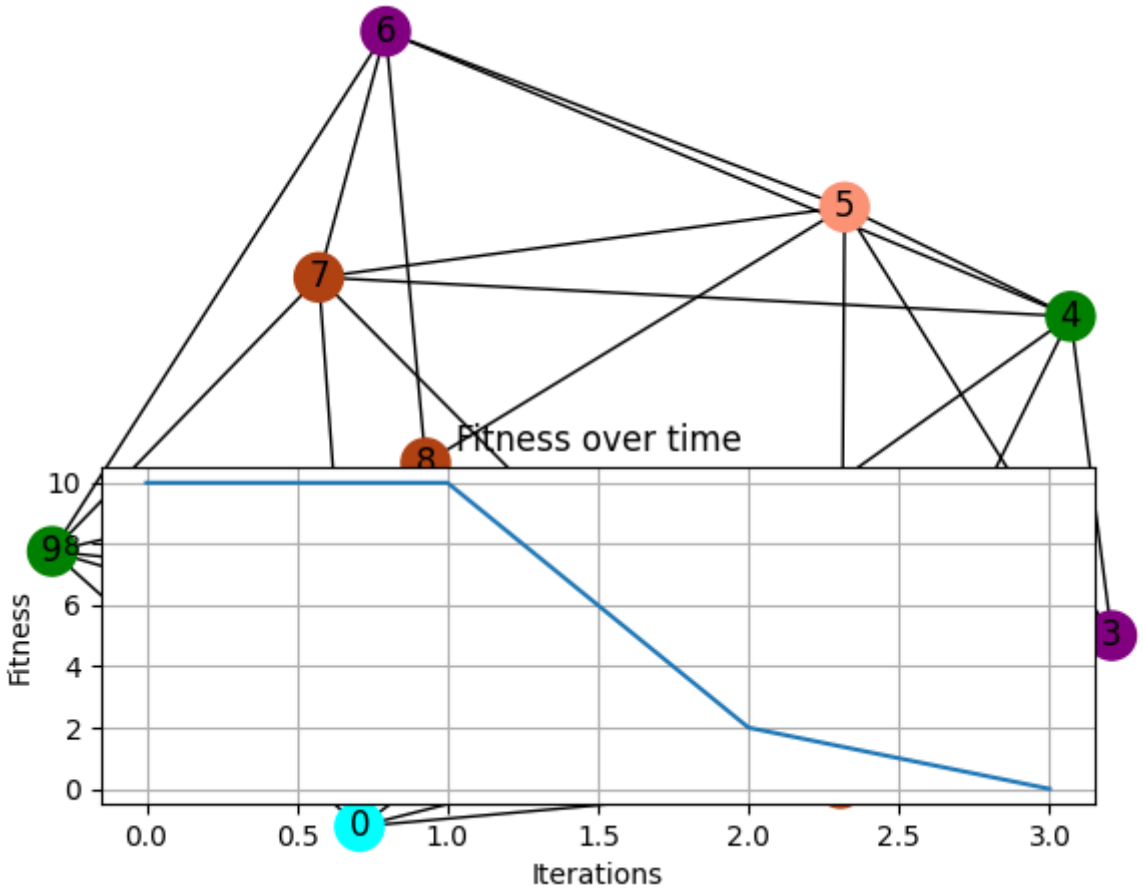

0
1
3
2
8
5
9
7
Fitness over time
Fitness
Iterations
0
1
2
3
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

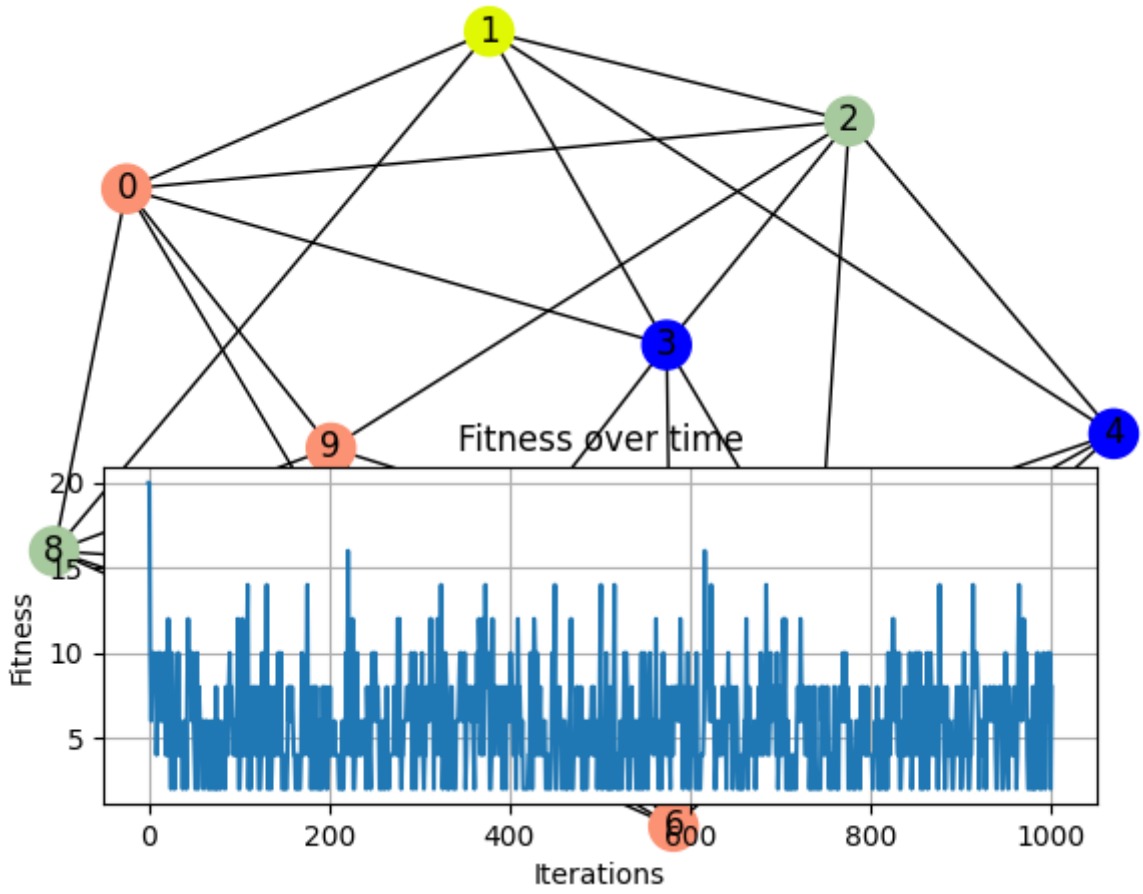

2
4
1
3
9
7
8
Fitness over time
12.5
10.0
7.5
5.0
2.5
0.0
Fitness
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Iterations

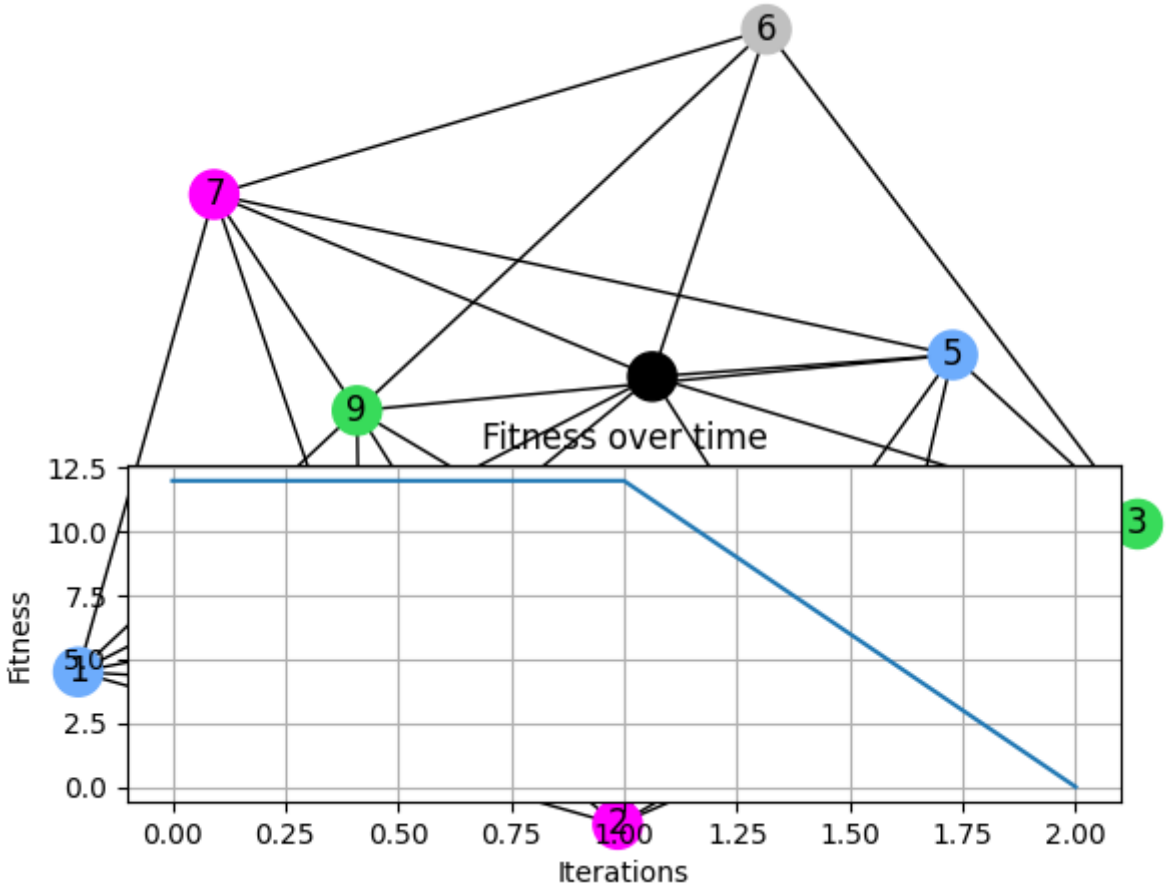

0
9
8
1
7
2
6
4
Fitness over time
15
10
5
Fitness
0
200
400
600
800
1000
Iterations

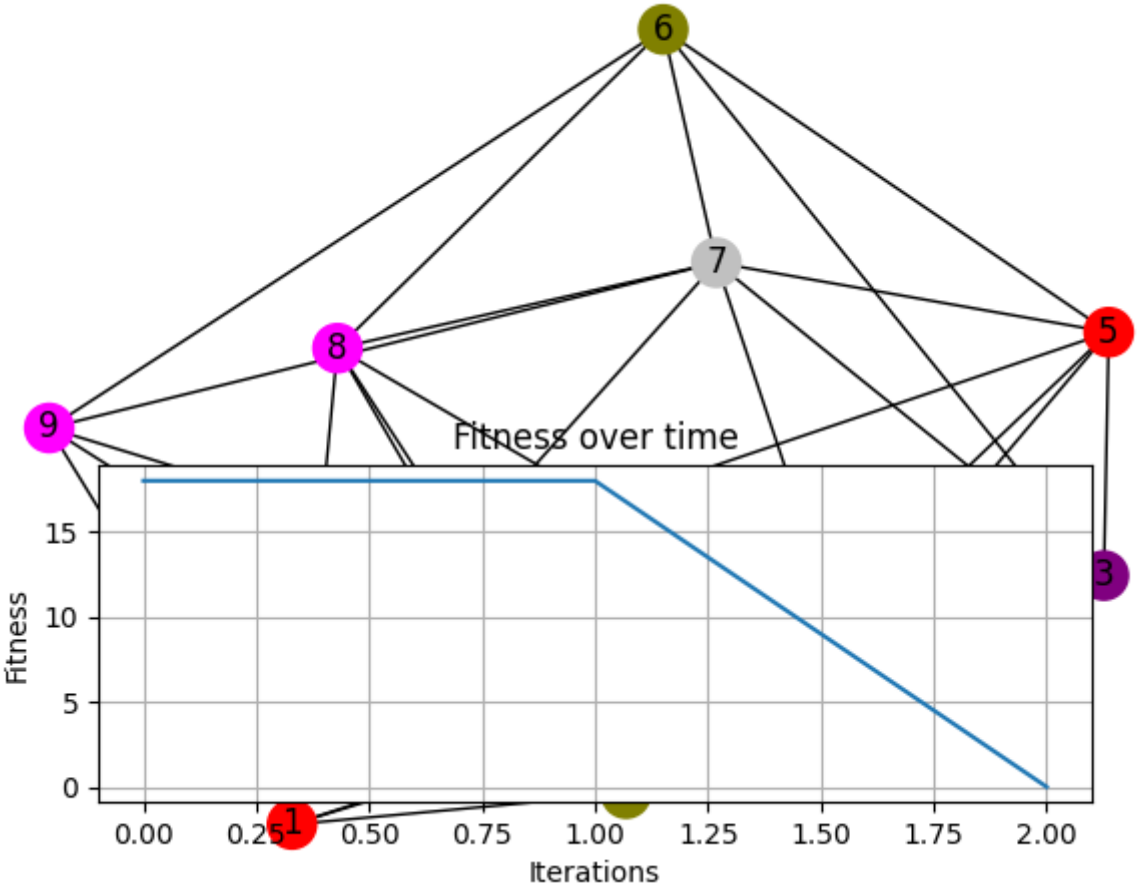

7
6
5
8
1
3
2
Fitness over time
20
10
5
0
Fitness
0
1
2
3
4
5
6
Iterations


2
4
3
0
8
7
Fitness over time
Fitness
Iterations
0
2
4
6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

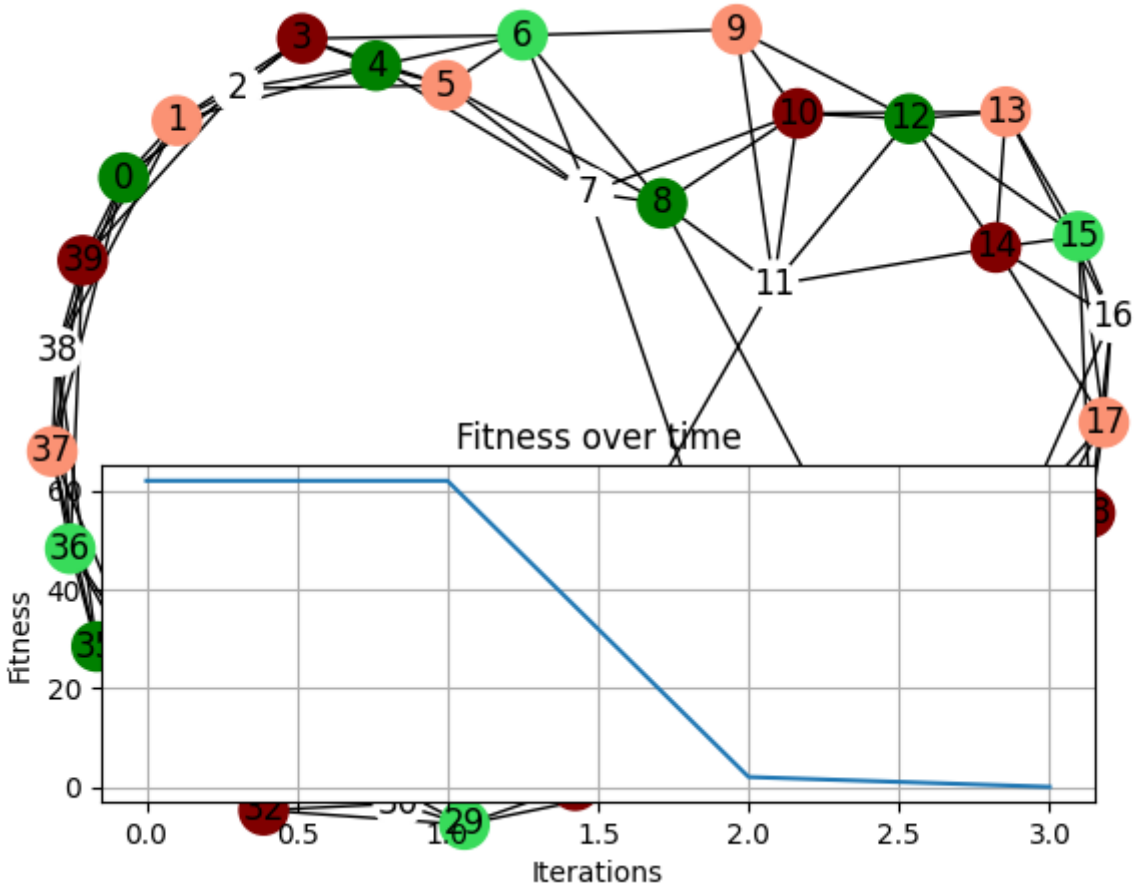

6
5
7
4
8
9
3
0
Fitness over time
Fitness
Iterations
10
8
6
4
2
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

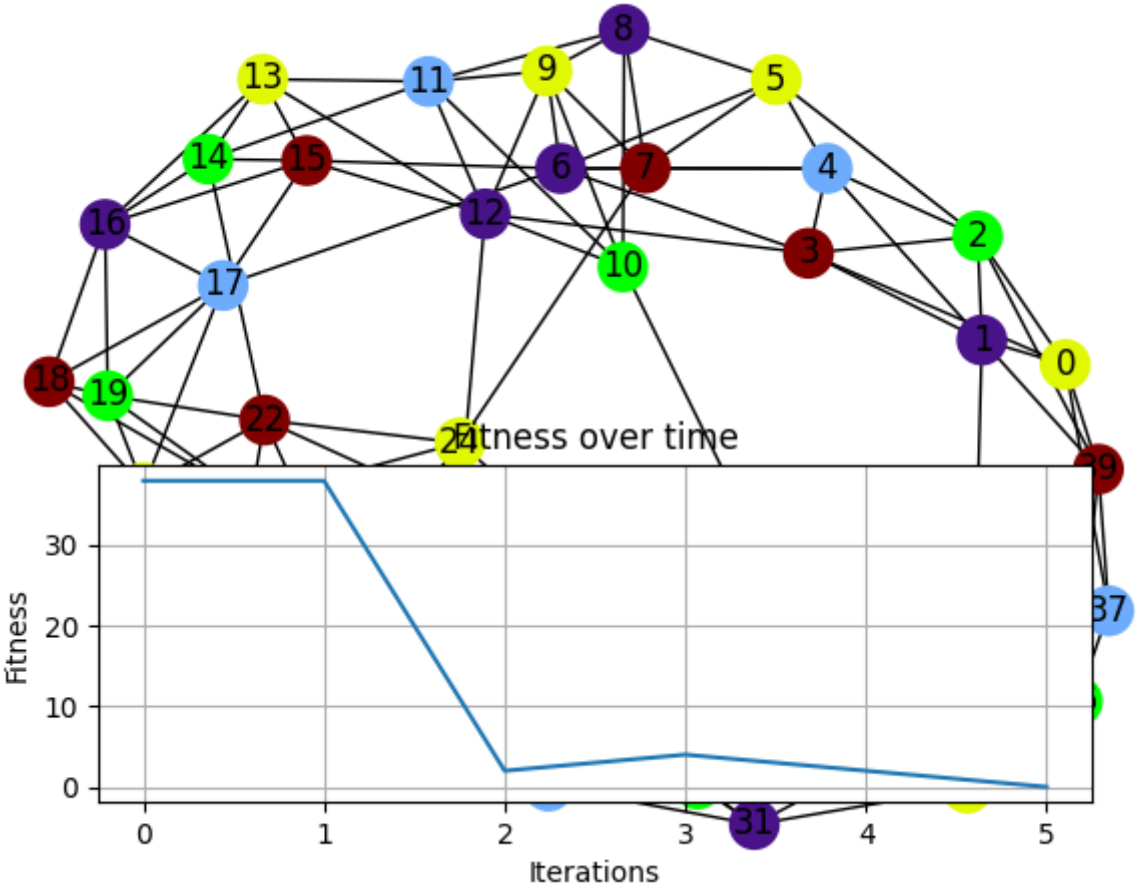

1
2
0
3
4
9
8
6
Fitness over time
Fitness
20
15
10
5
0
200
400
600
800
1000
Iterations

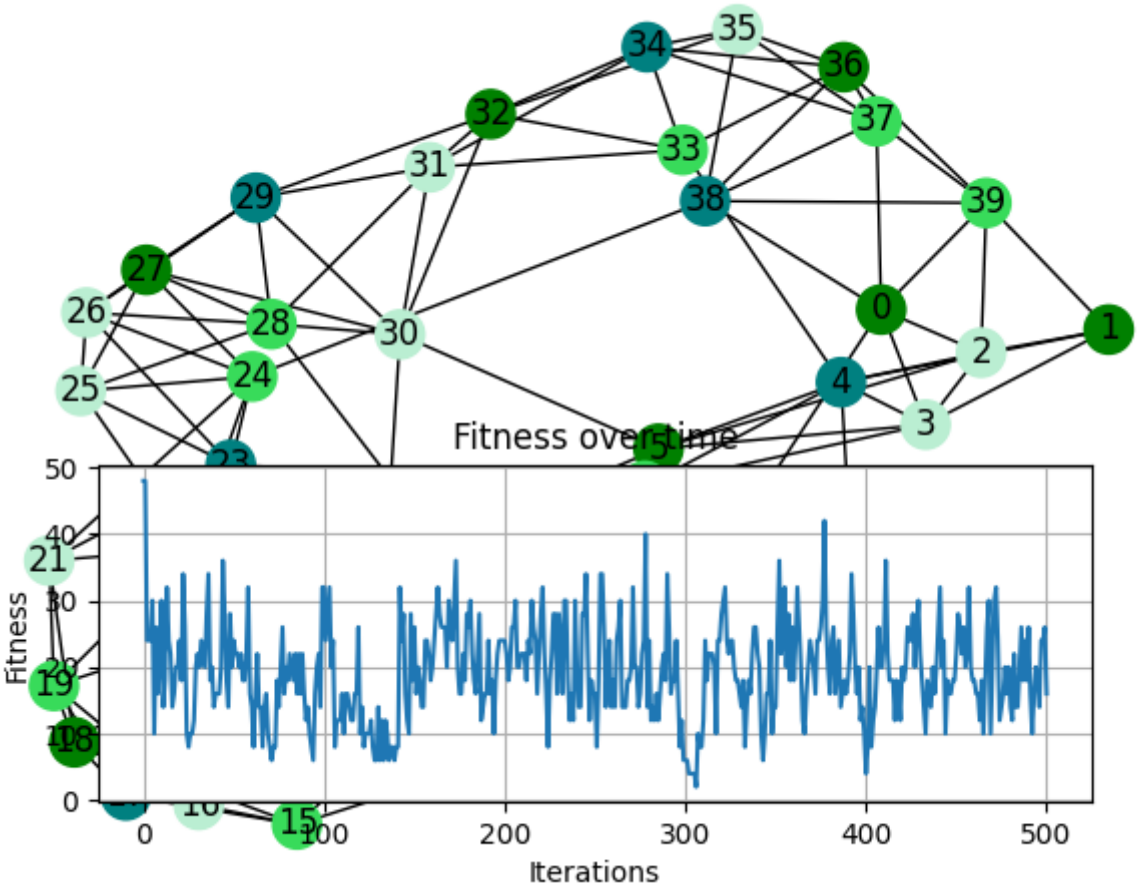

6
7
5
9
Fitness over time
12.5
10.0
7.5
5.0
2.5
0.0
Fitness
3
1
2
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Iterations

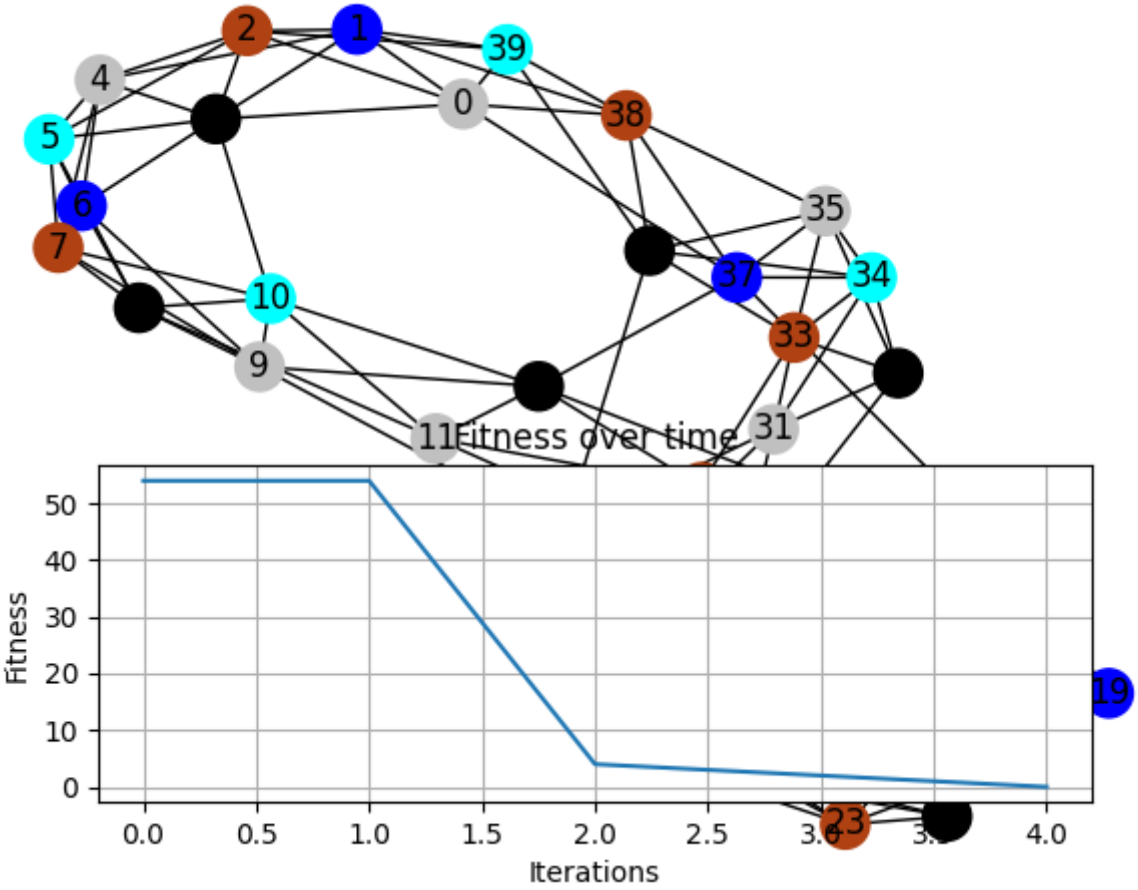


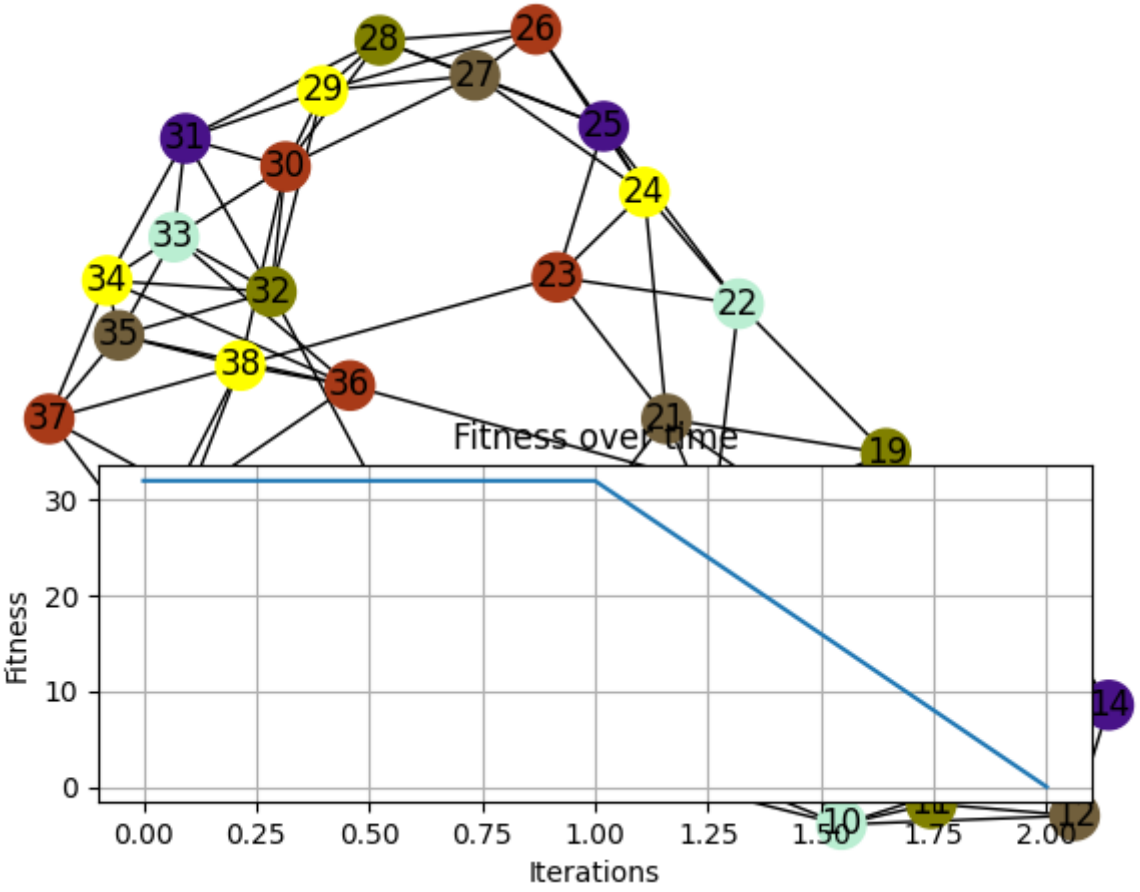
40 Node 6 Degree

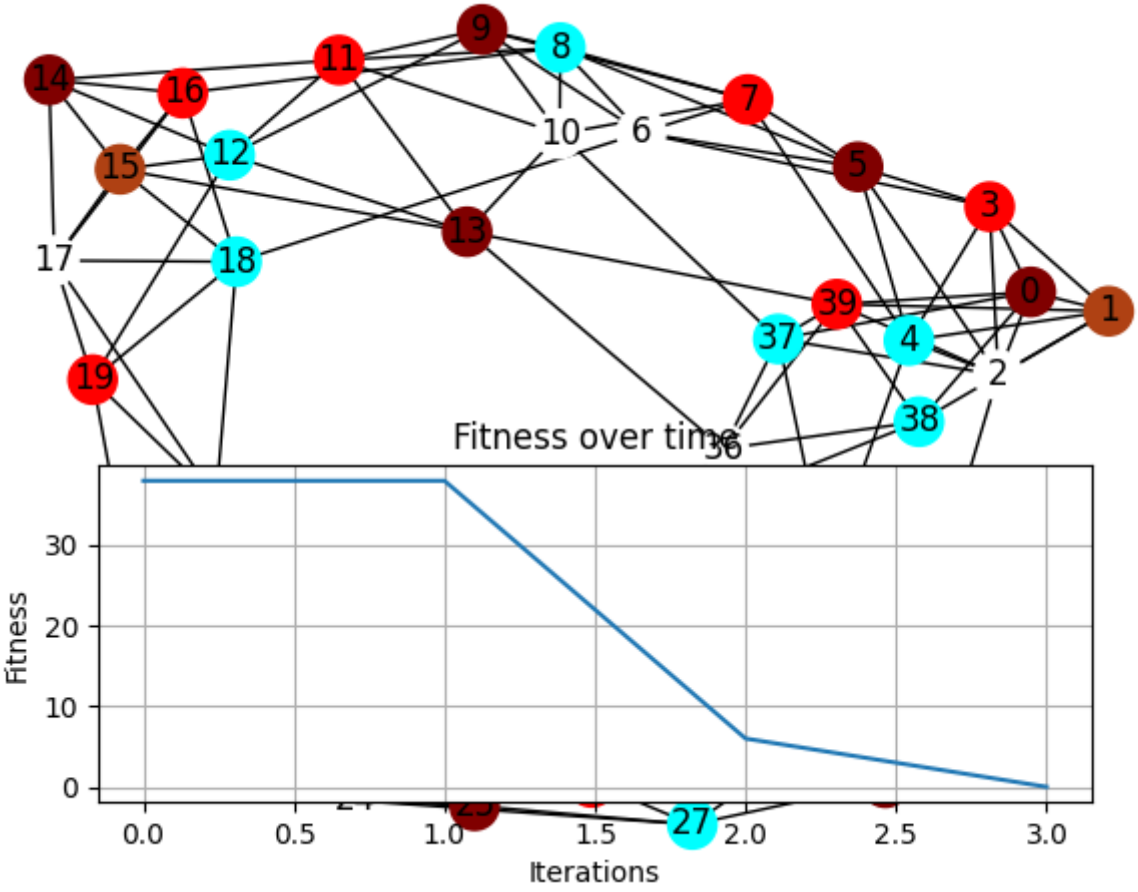

Fitness over time
Fitness
Iterations
0
20
40
0.0
0.5
1.5
2.0
2.5
3.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
20
29
32
36
37
38
39

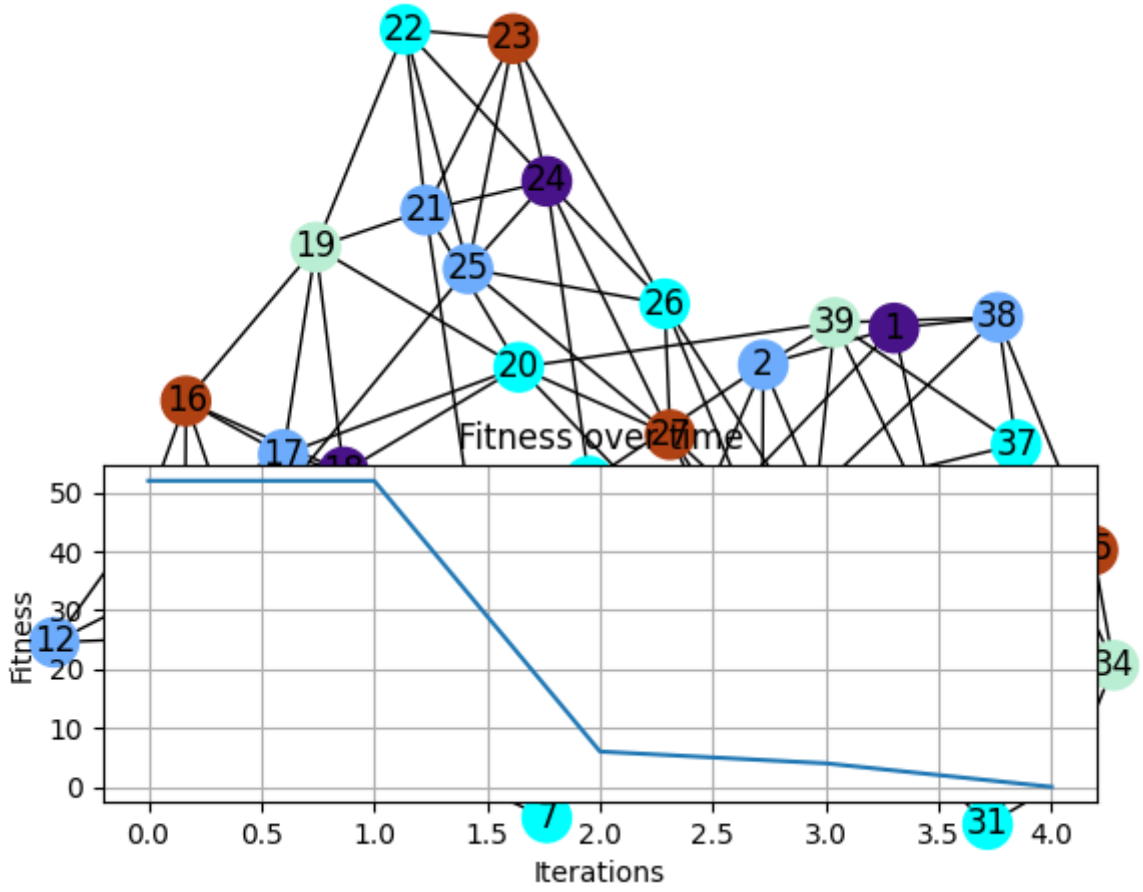

Fitness over time
Fitness
Iterations
0
10
20
30
0
1
2
3
4
5

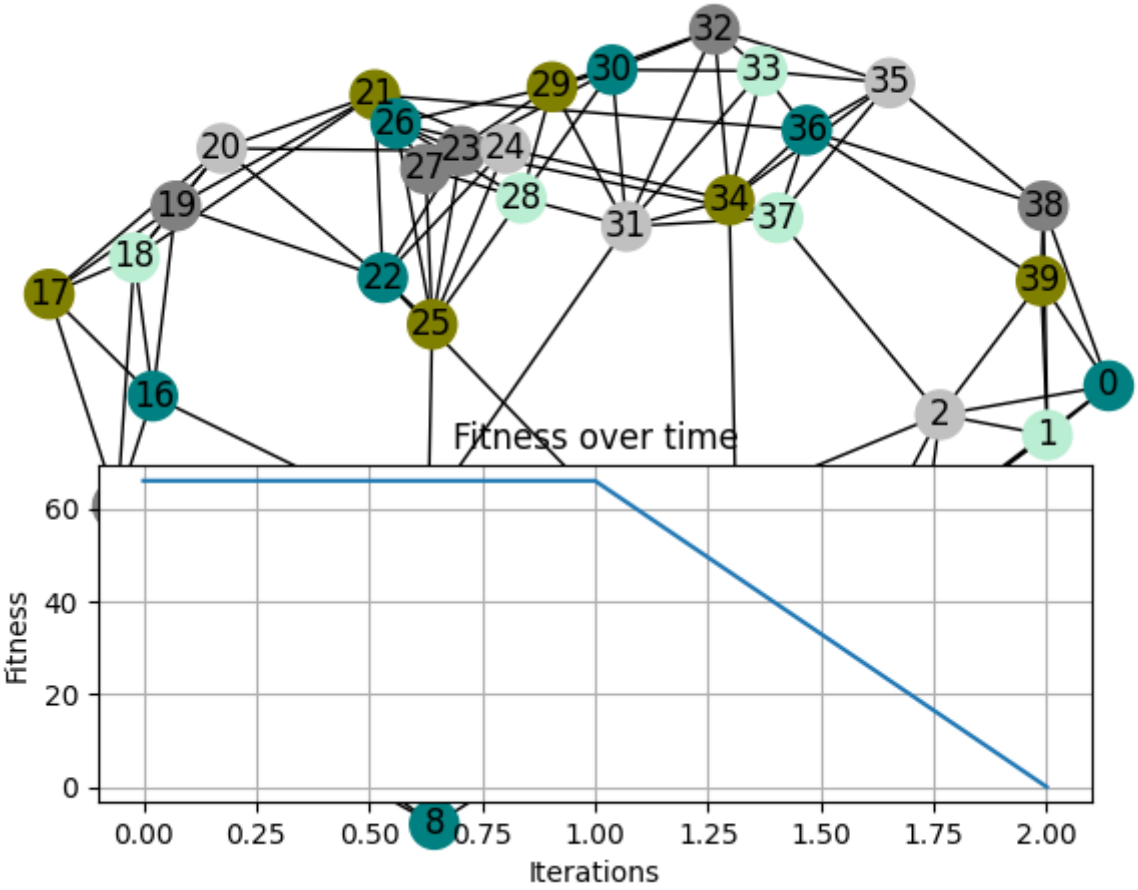

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
50
40
0

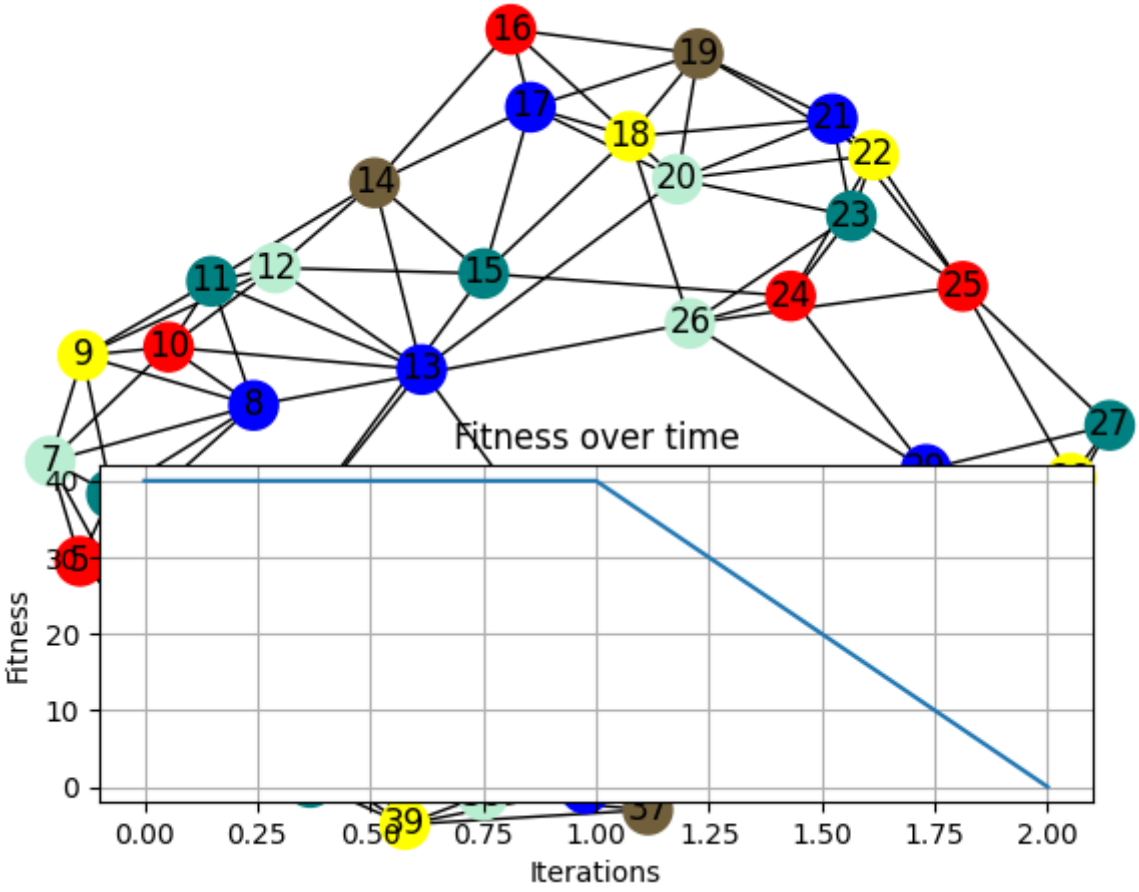

2
1
39
4
0
38
5
6
35
7
37
34
10
33
9
11
31
19
23
Fitness over time
Fitness
Iterations
50
40
30
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5
4.0

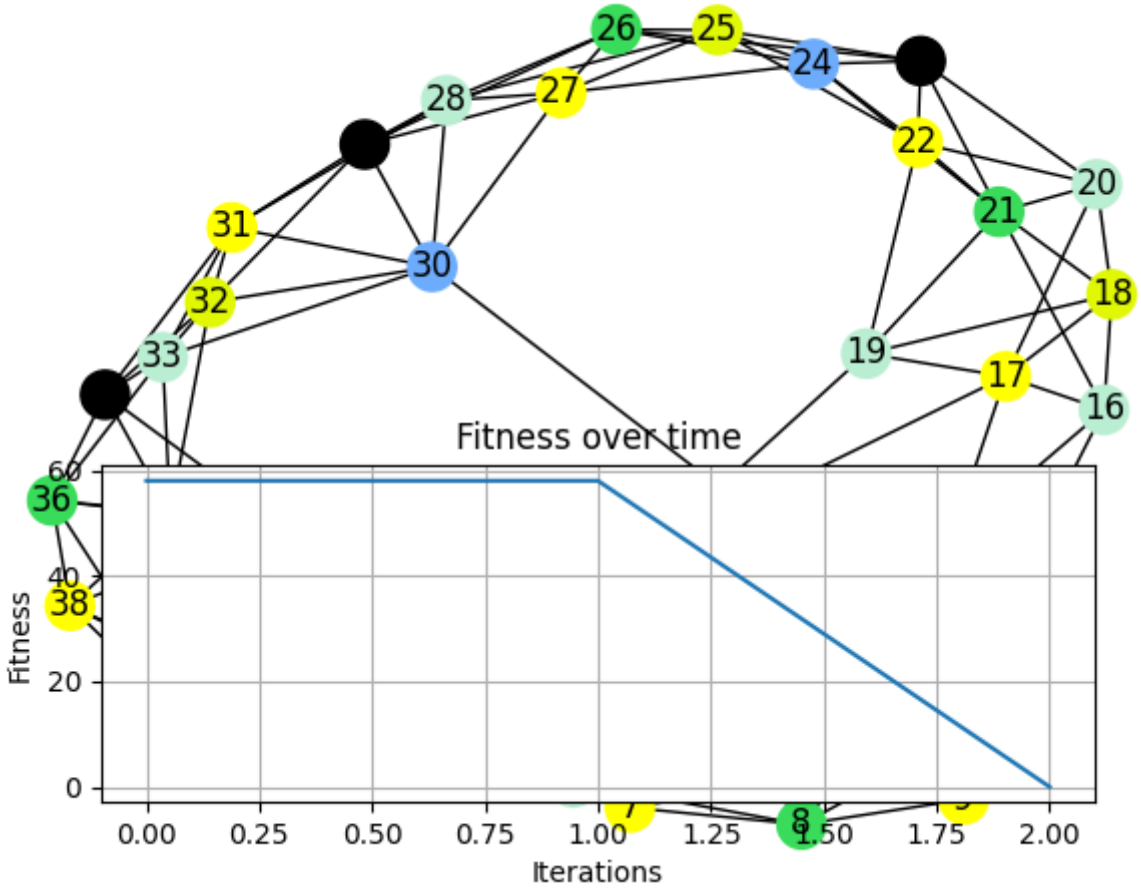

Fitness over time
Fitness
Iterations
0
10
20
30
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Fitness over time
Fitness
Iterations
0
10
20
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0

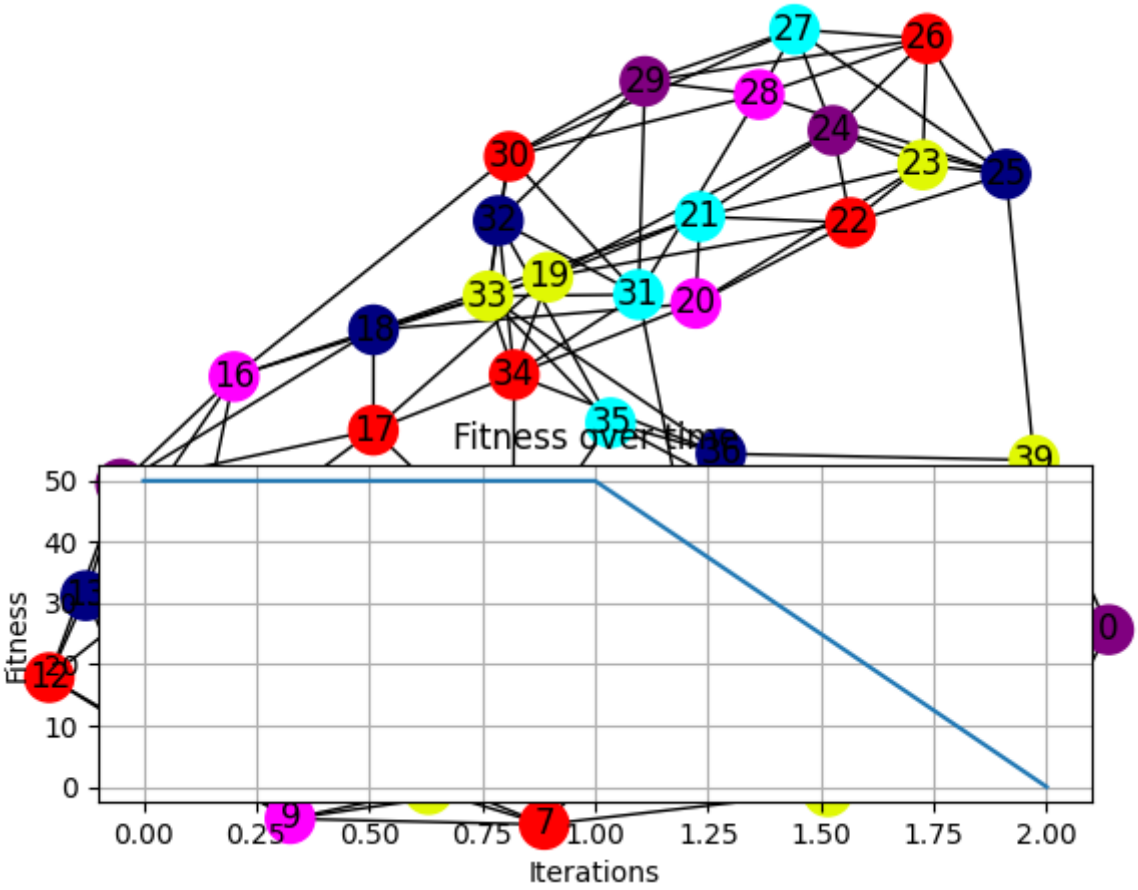

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
22
23
24
21
19
25
26
39
1
38
20
2
16
17
37
12
34
7
31

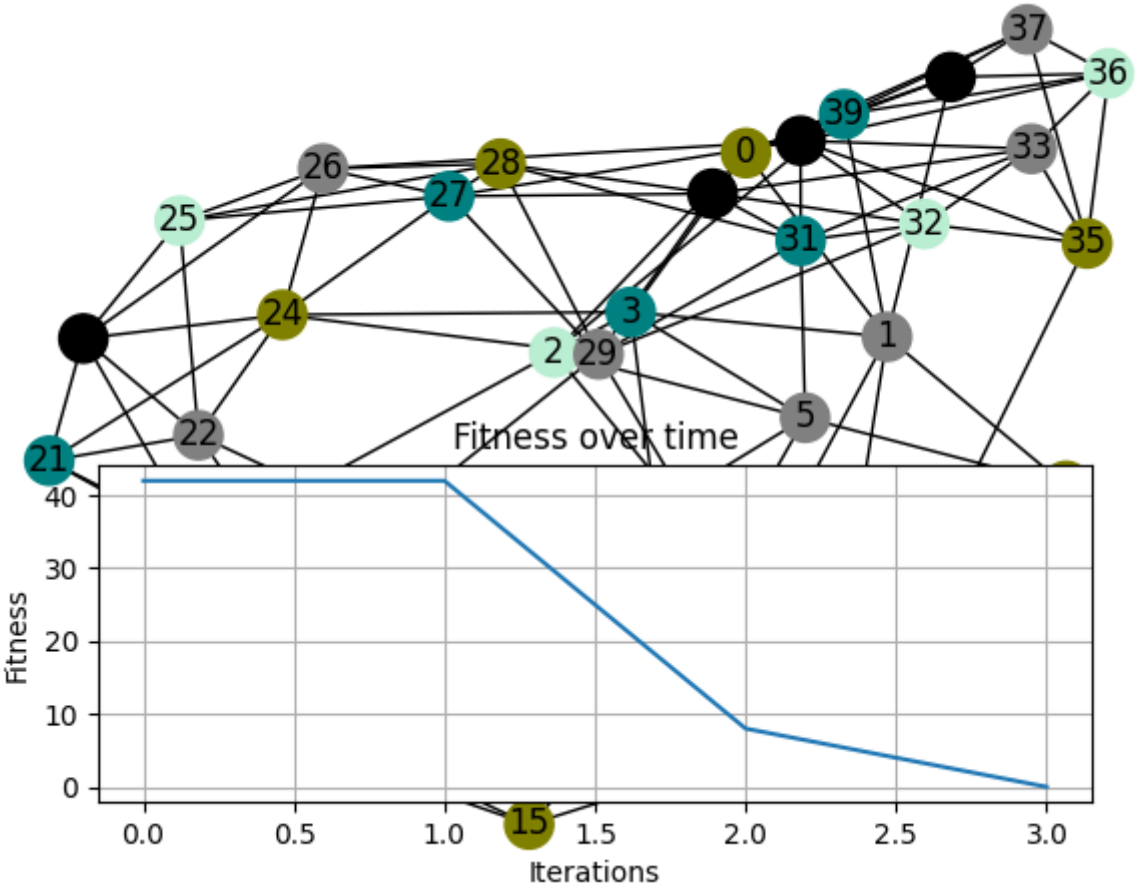

Fitness over time
Fitness
Iterations
0
20
40
60
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

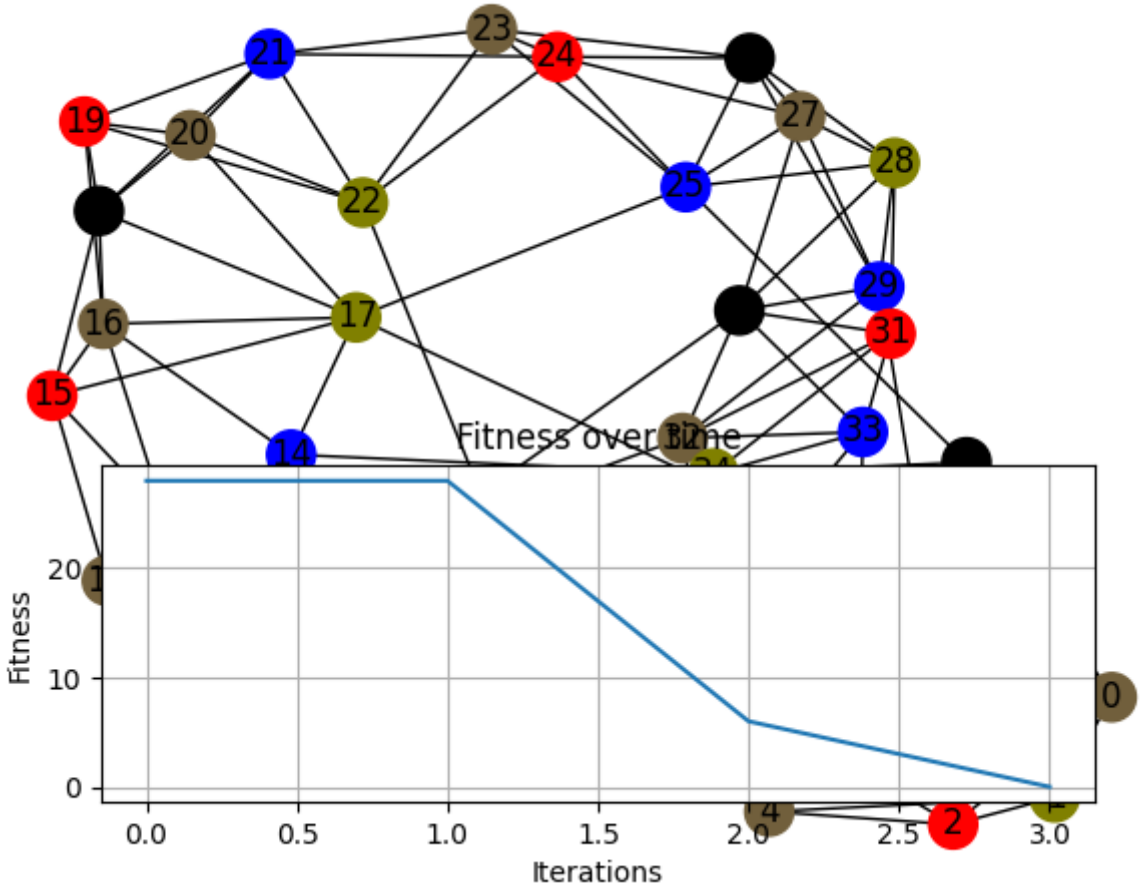

Fitness over time
Fitness
Iterations
0
10
20
40
0.00
0.25
0.75
1.00
1.25
1.50
1.75
2.00

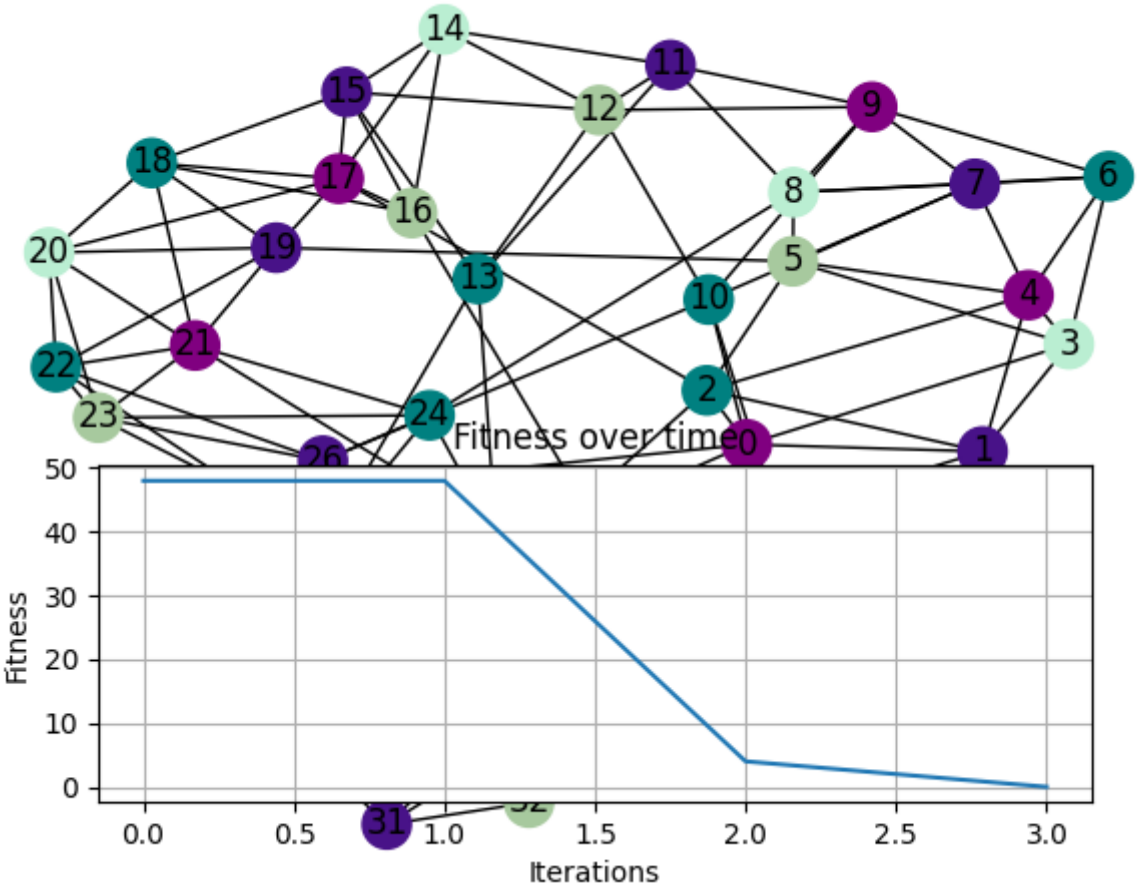


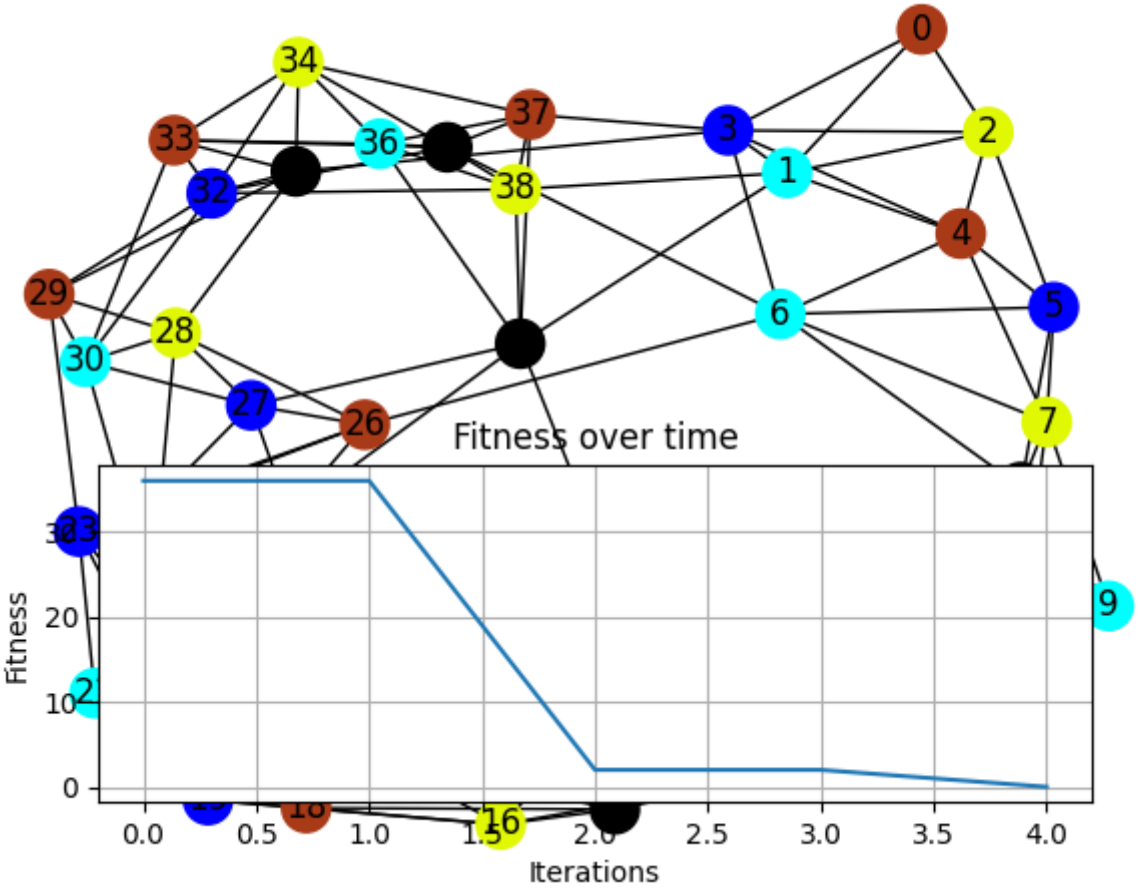
40 Node 7 Degree

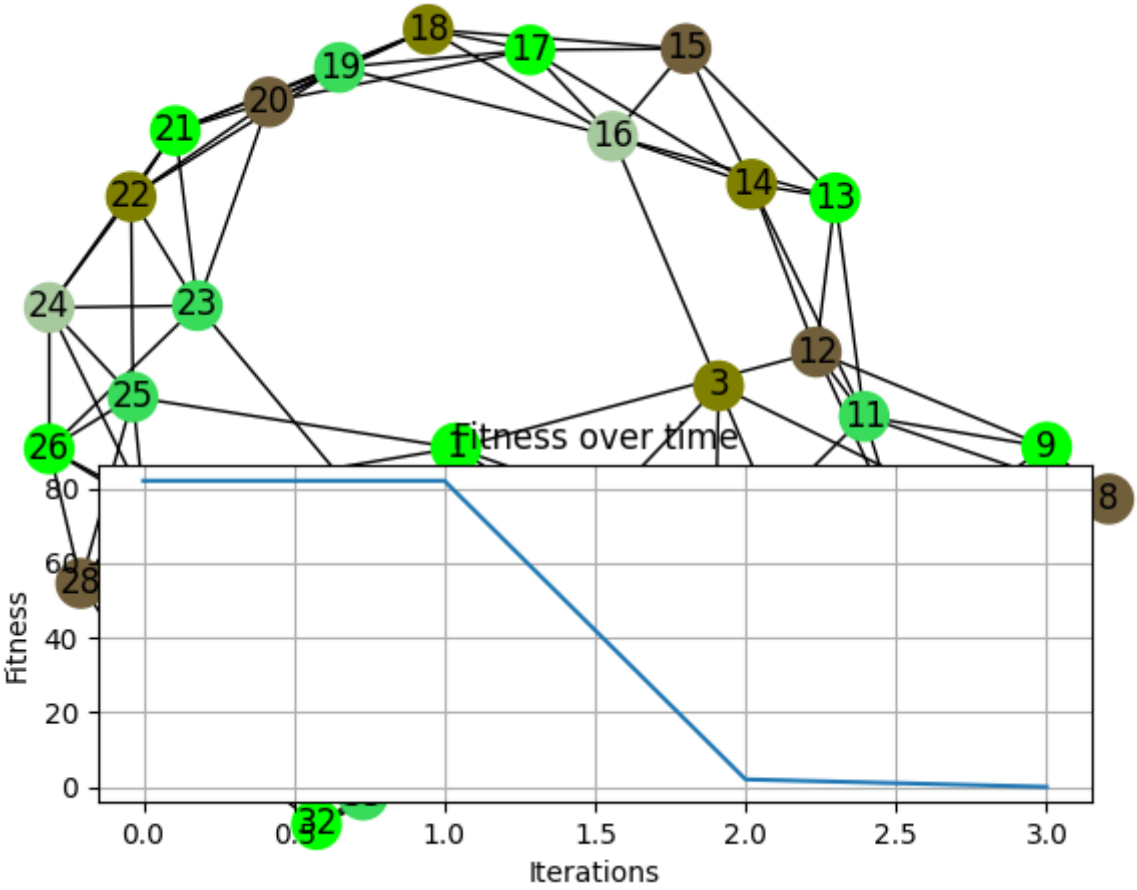

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

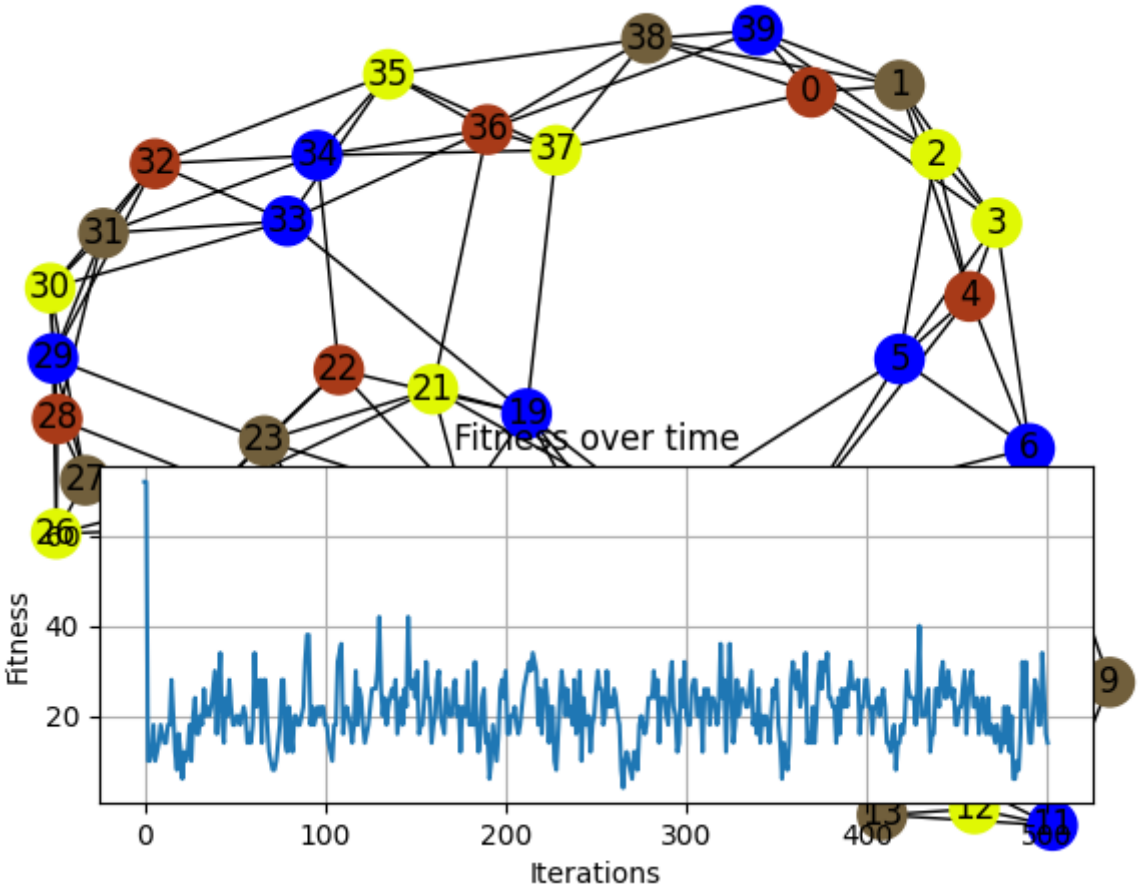

37
36
39
0
33
26
28
27
25
31
32
35
24
3
1
2
29
5
22
21
15
Fitness over time
Fitness
40
30
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations

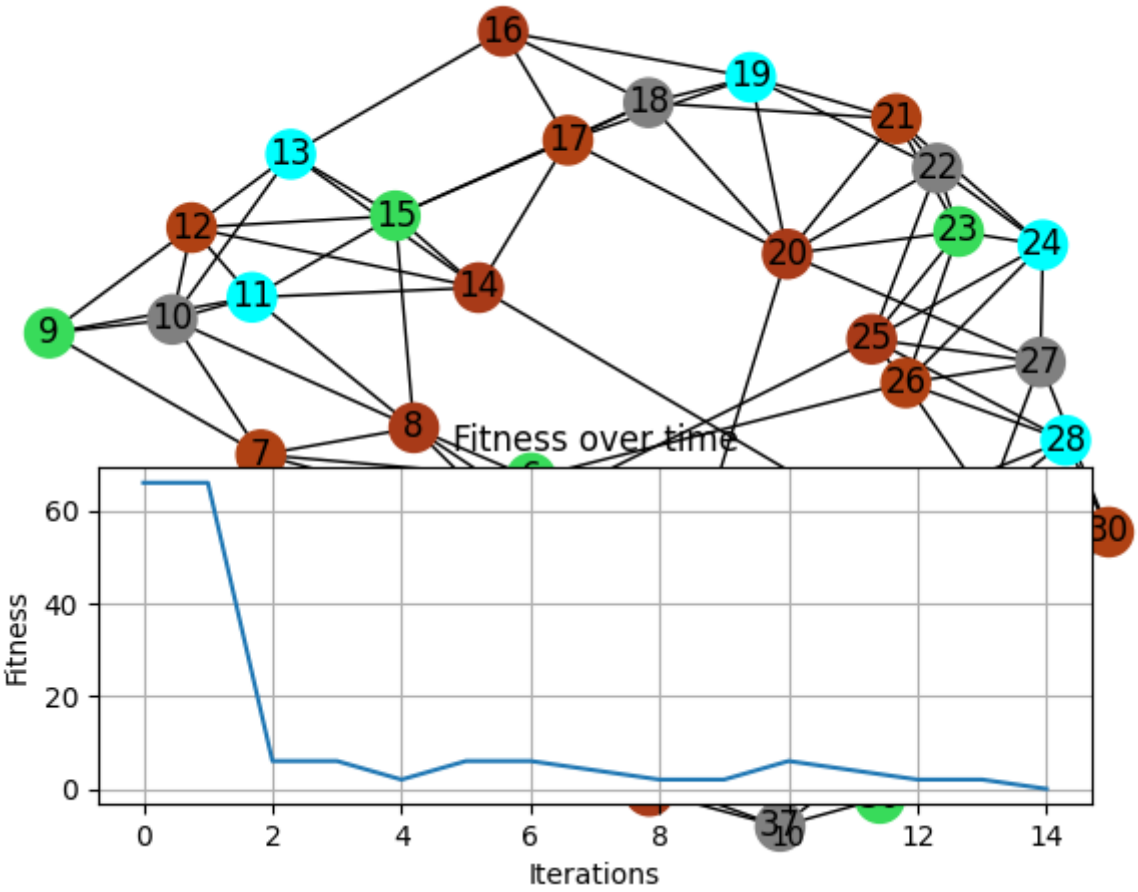

Fitness over time
Fitness
Iterations
0
10
20
0.0
0.5
1.0
1.5
2.0
2.5
3.0

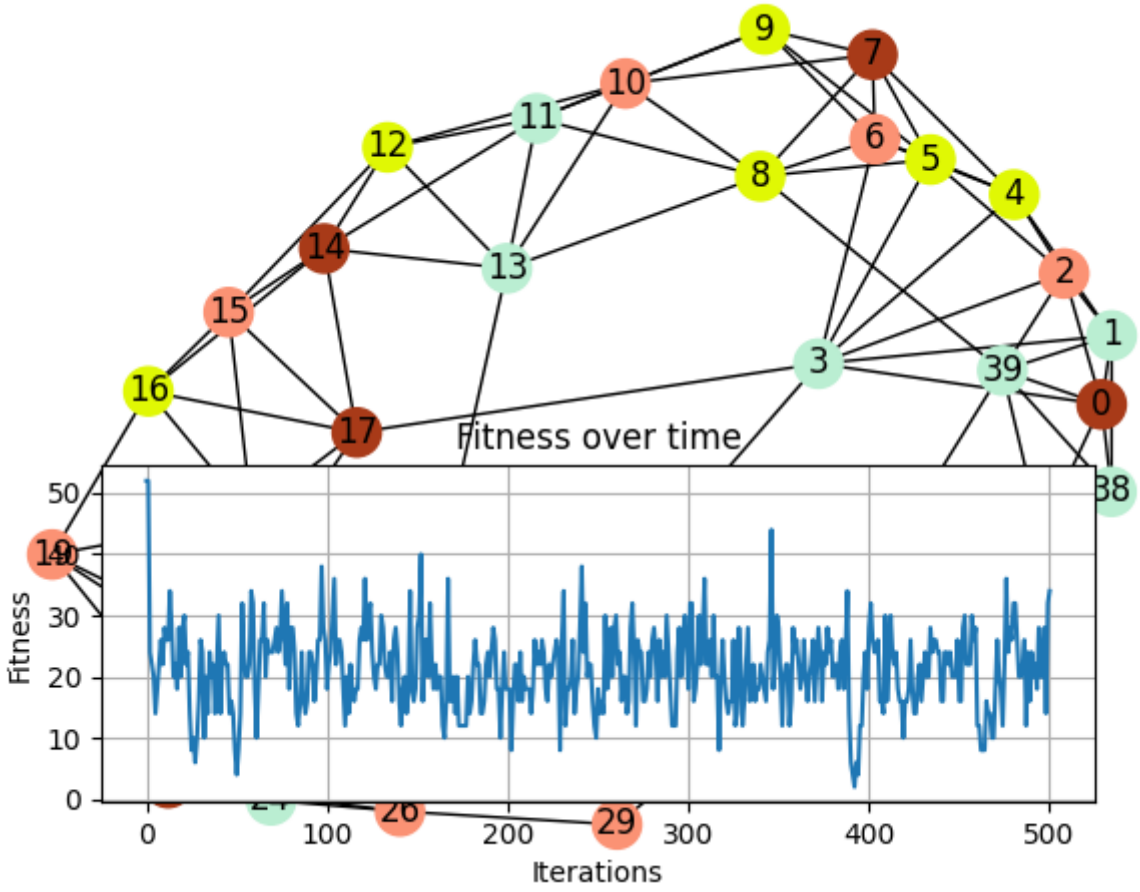

Fitness over time
Fitness
Iterations
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
10
20
30
40
50

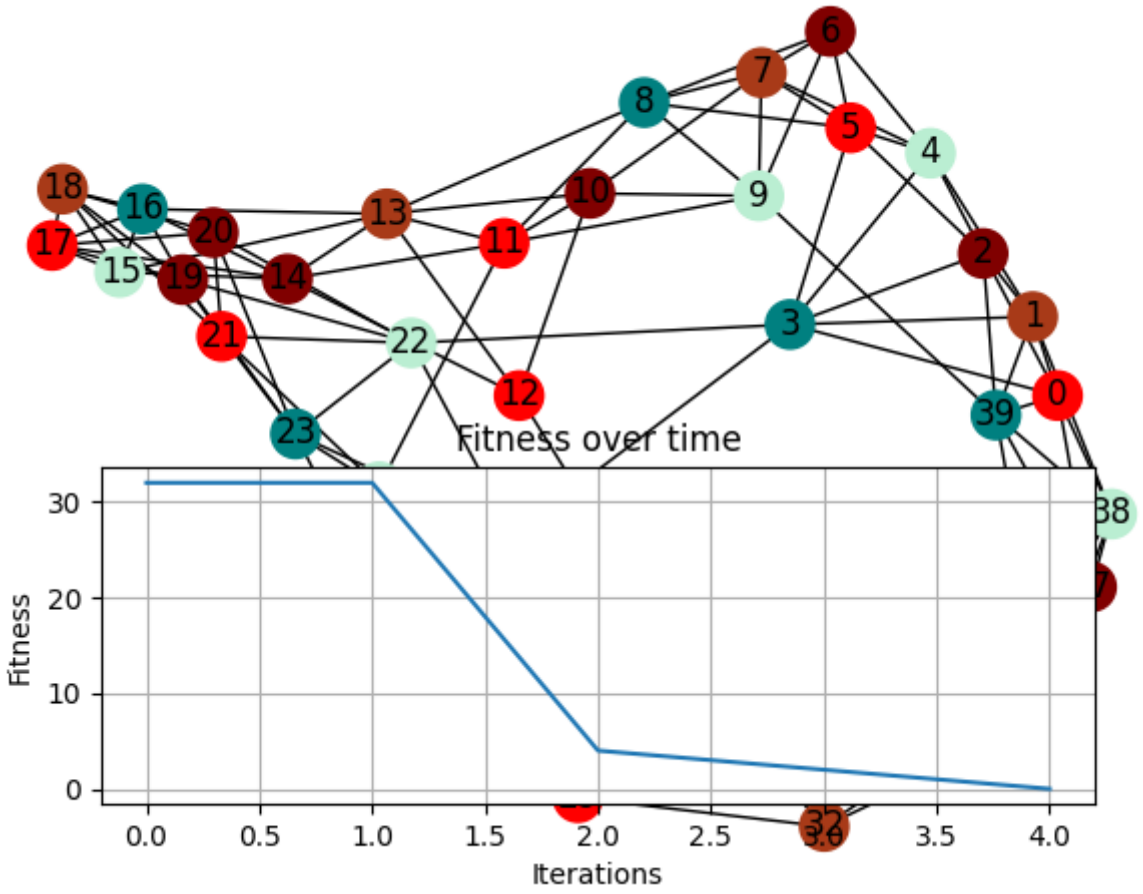

Fitness over time
Fitness
Iterations
0
20
0.0
0.5
1.0
2.0
2.5
3.0
3.5
4.0
0
1
2
3
4
5
6
7
9
16
26
27
28
29
30
32
33
34
36
37
38


Fitness over time
Fitness
Iterations
0
20
40
60
80
0.0
0.5
1.0
1.5
2.0
2.5
3.0
18
17
15
19
20
21
16
22
14
13
24
23
12
3
25
11
26
9
8
28
32


Fitness over time
Fitness
Iterations
0
100
200
300
400
20
40

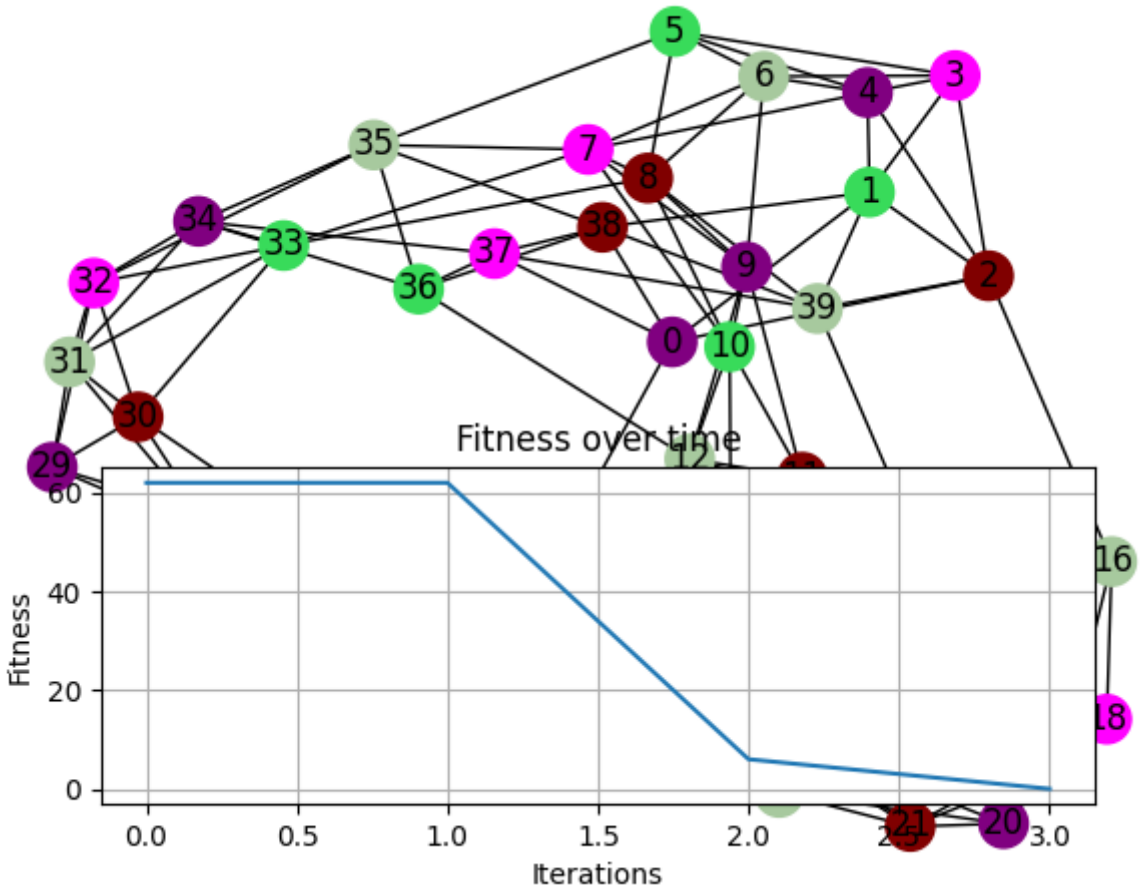

Fitness over time
Fitness
Iterations
0
20
40
60
0
2
4
6
8
10
12
14

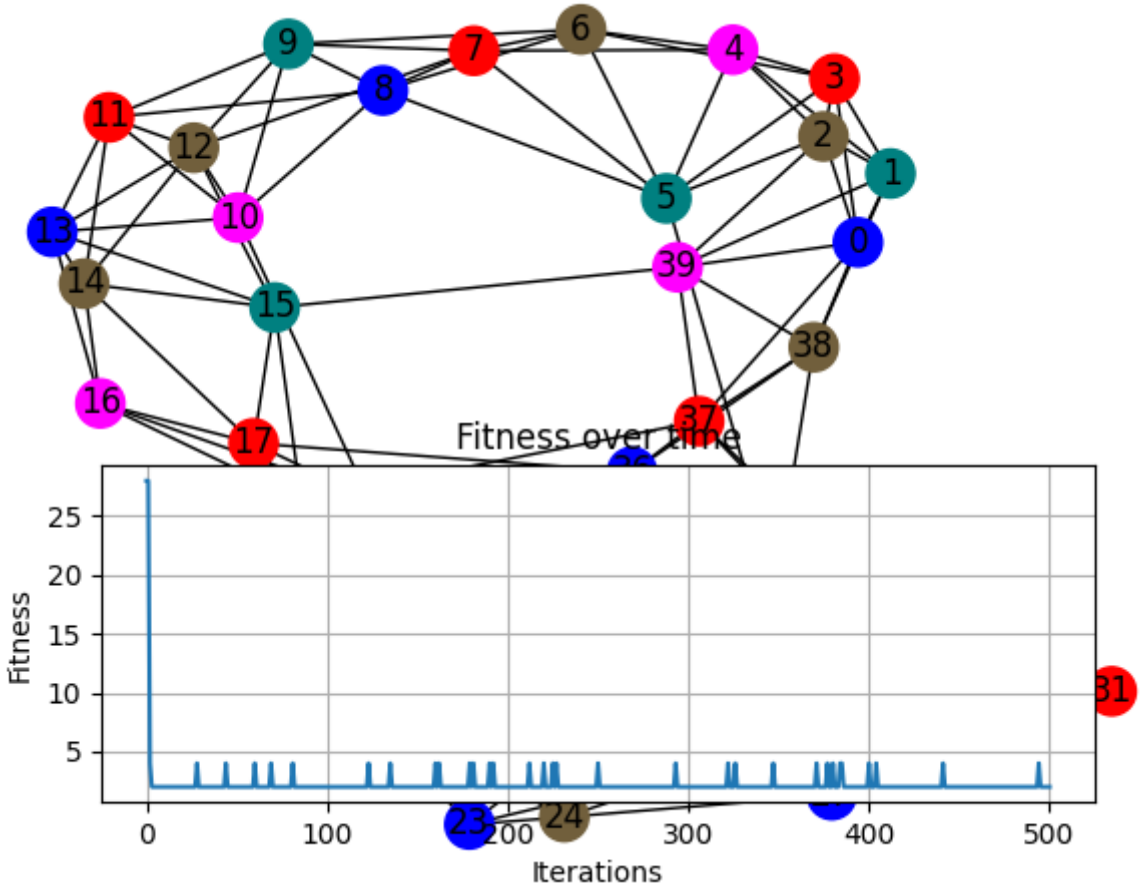

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0
100
200
300
400
500
9
7
10
11
12
6
5
8
4
14
13
2
15
1
3
39
16
0
17
38
26
29

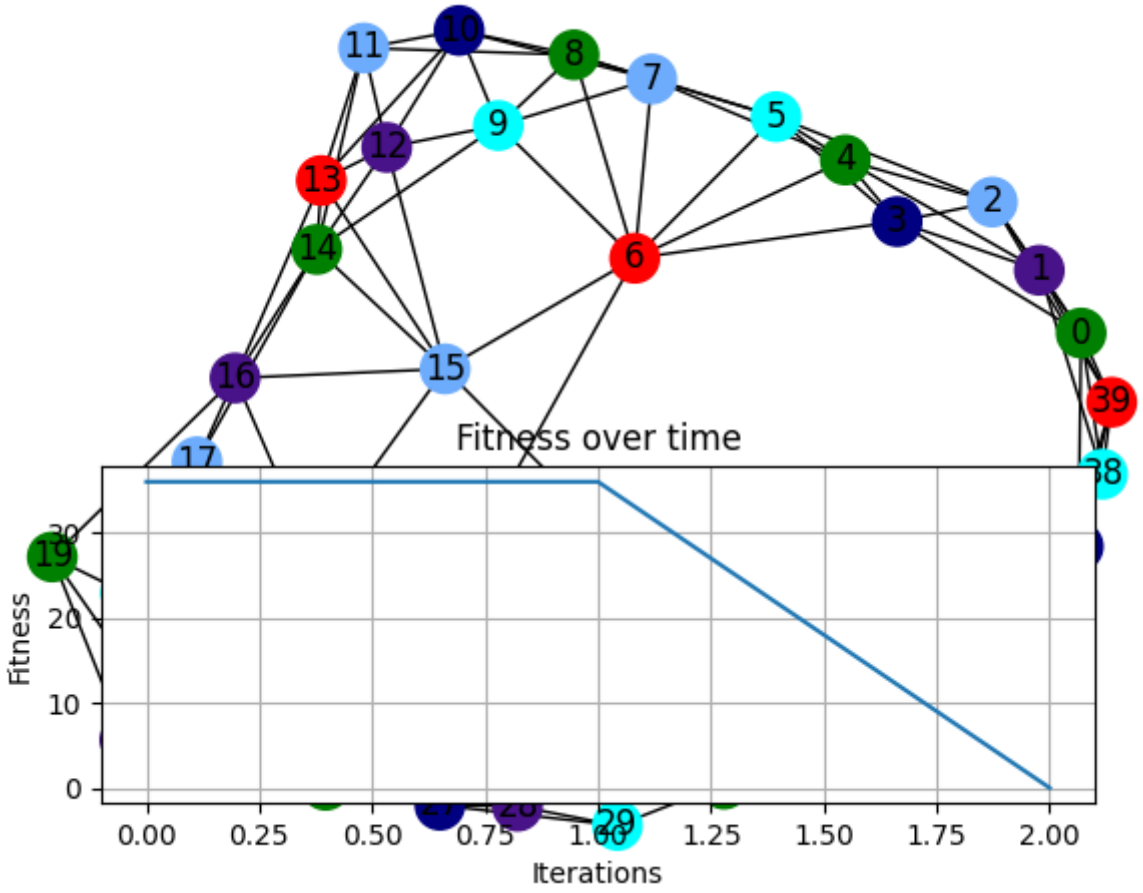


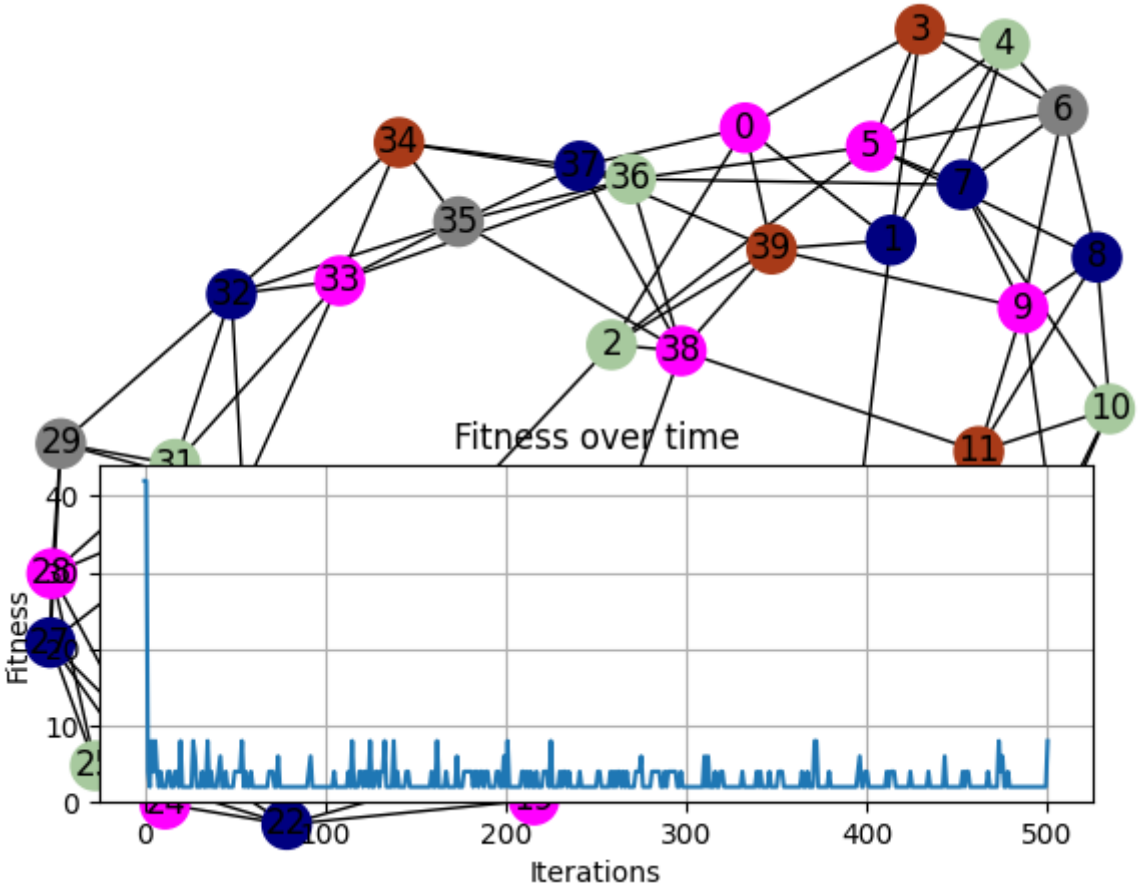
Problem 2

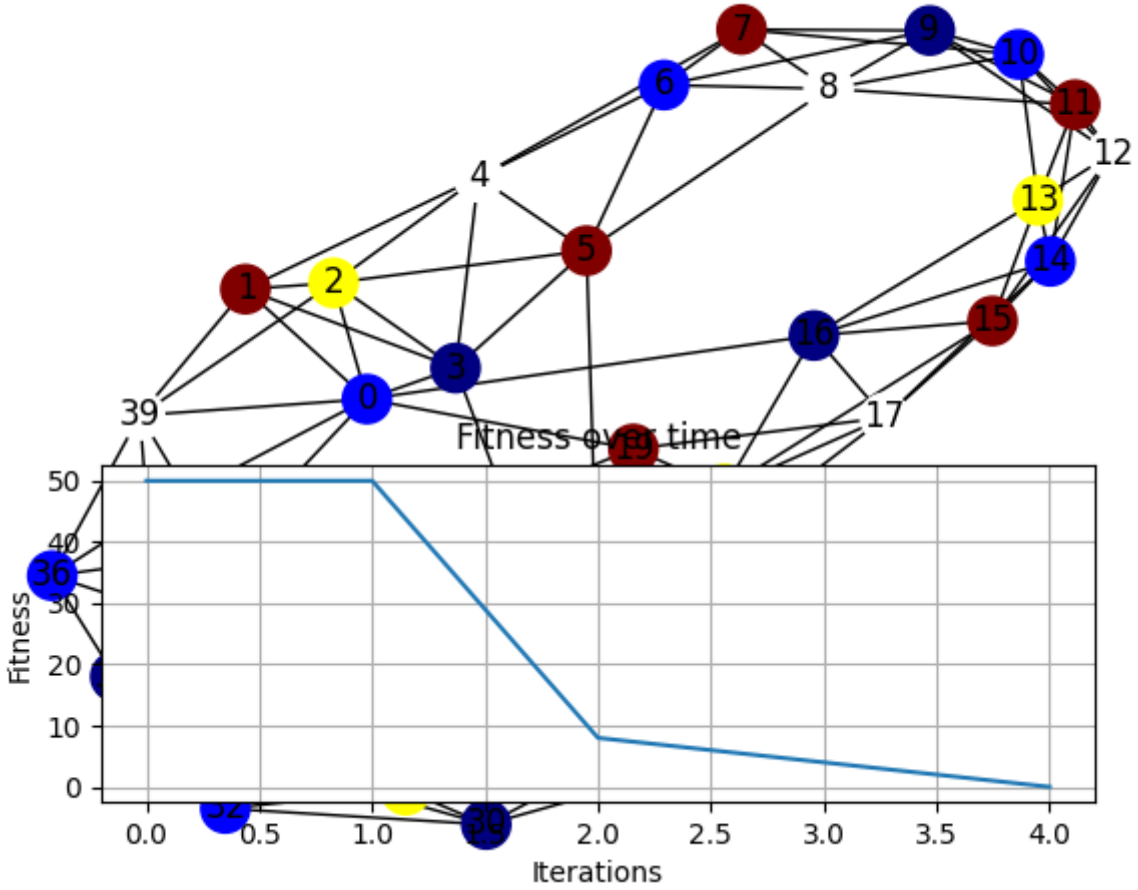
10% Aggressive

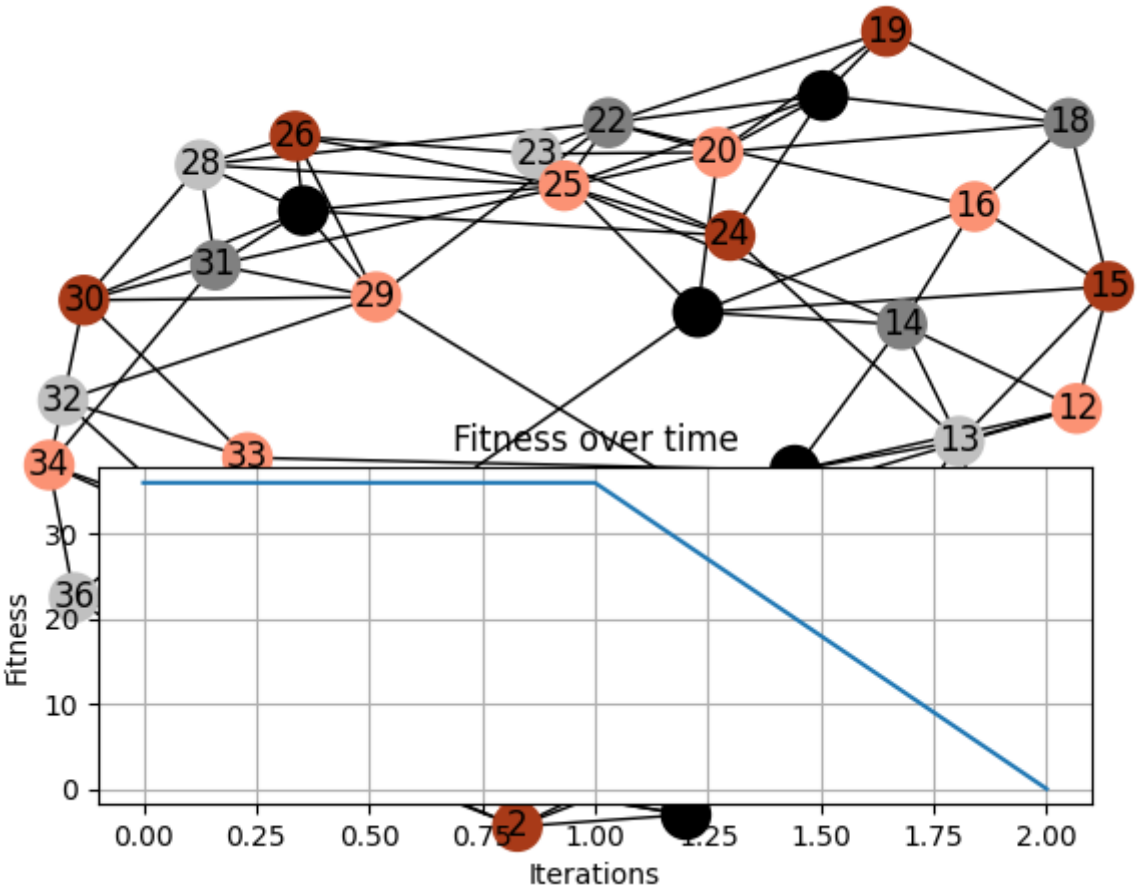

Fitness over time
Fitness
Iterations
0
20
40
60
0.0
0.5
1.0
1.5
2.0
3.0
5
6
3
4
35
7
8
1
34
33
38
37
9
2
32
36
39
0
10
31
30
29
16
18
20

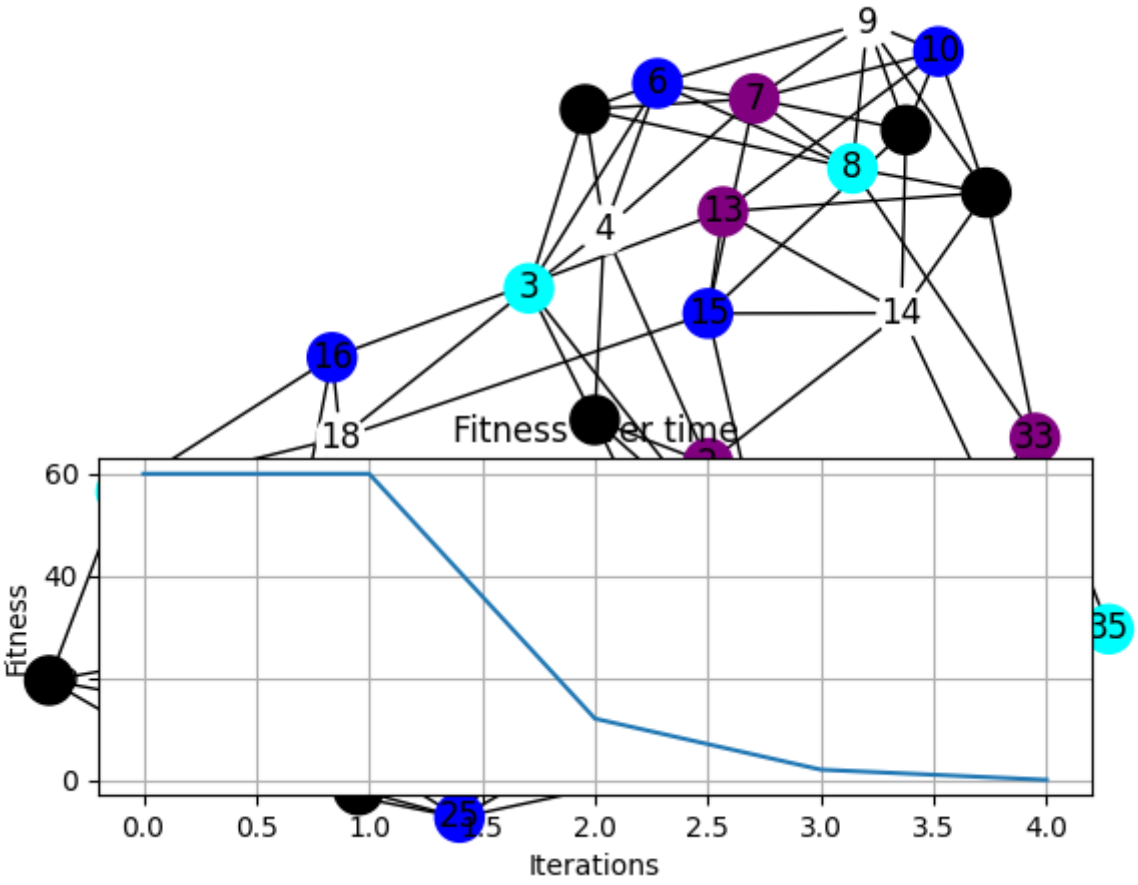

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
5
10
15
20
25

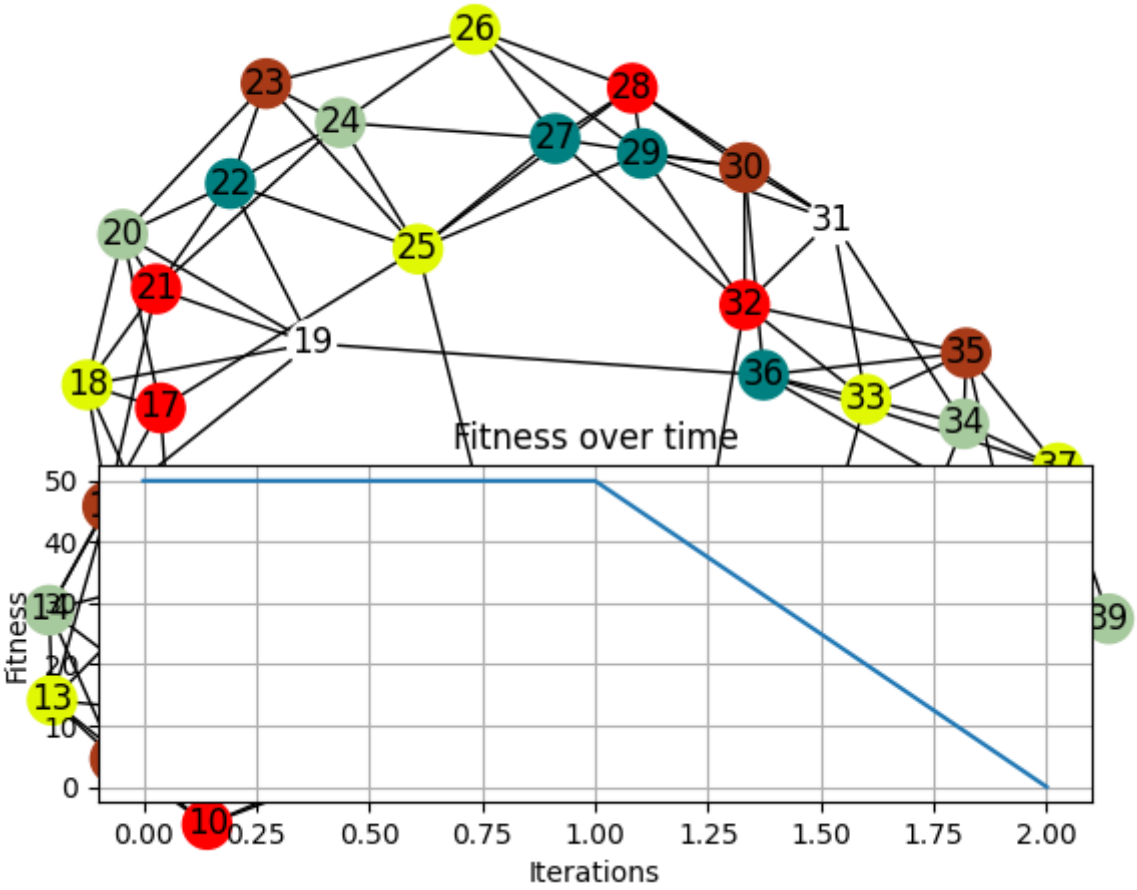

Fitness over time
Fitness
Iterations
0
10
20
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

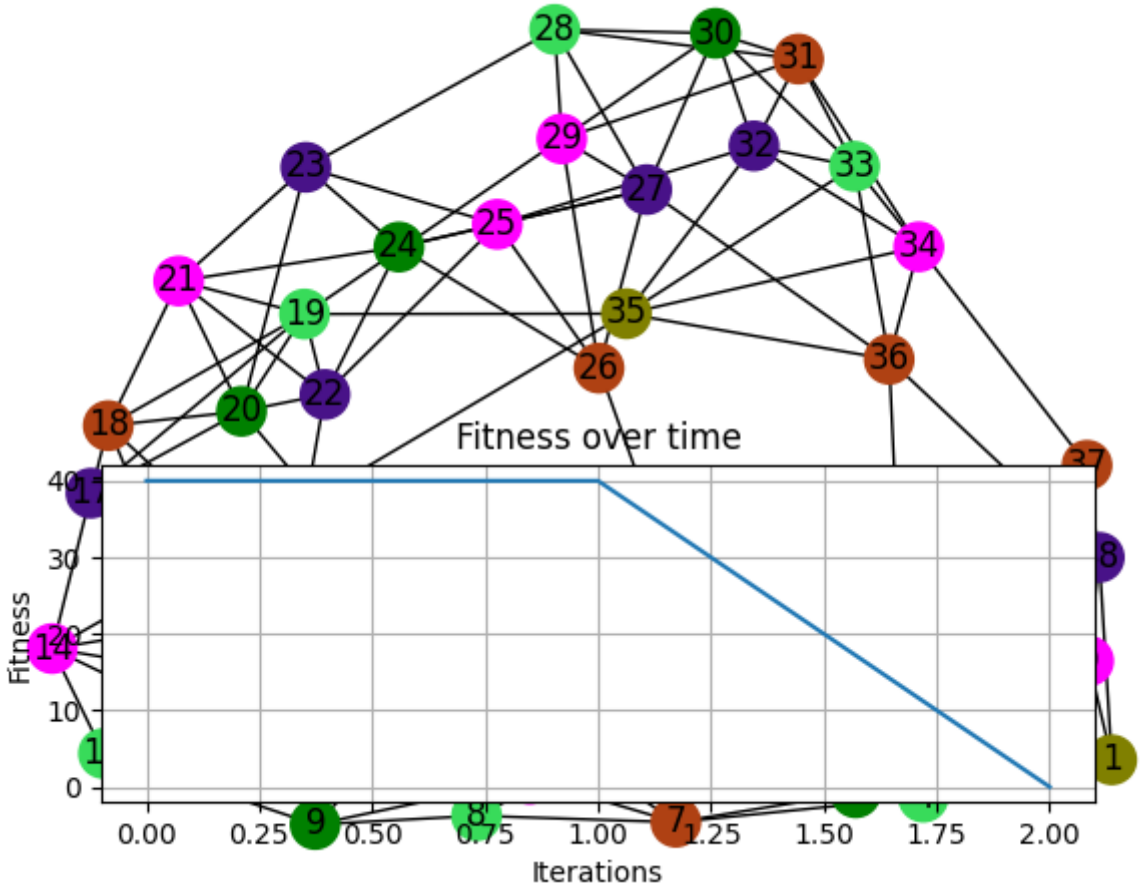

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
0
10
40
3
4
6
34
0
5
37
36
7
35
39
1
8
32
33
9
2
38
10
29
11

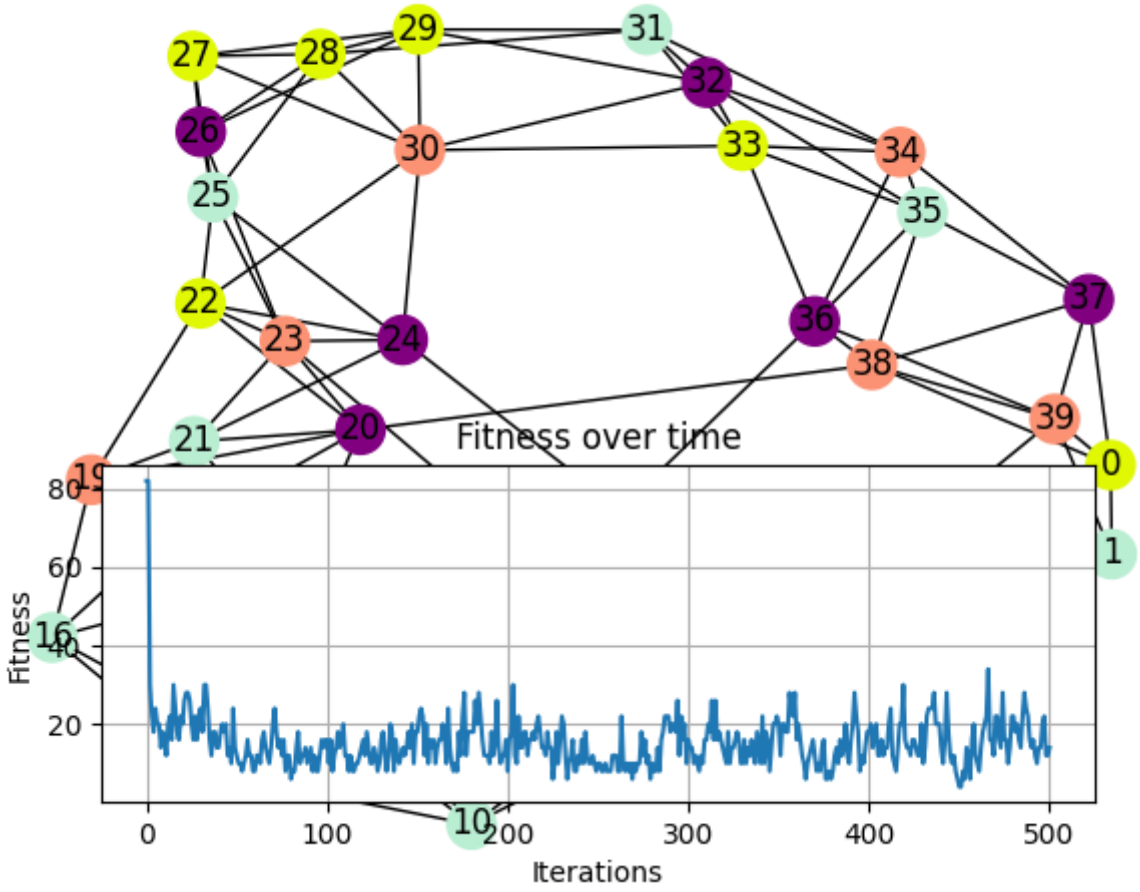

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0.0
0.5
1.0
2.0
2.5
3.0
3.5
4.0


19
18
22
26
23
20
28
25
16
24
31
15
30
29
14
32
12
13
34
33
36
2
Fitness over time
Fitness
30
20
10
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Iterations

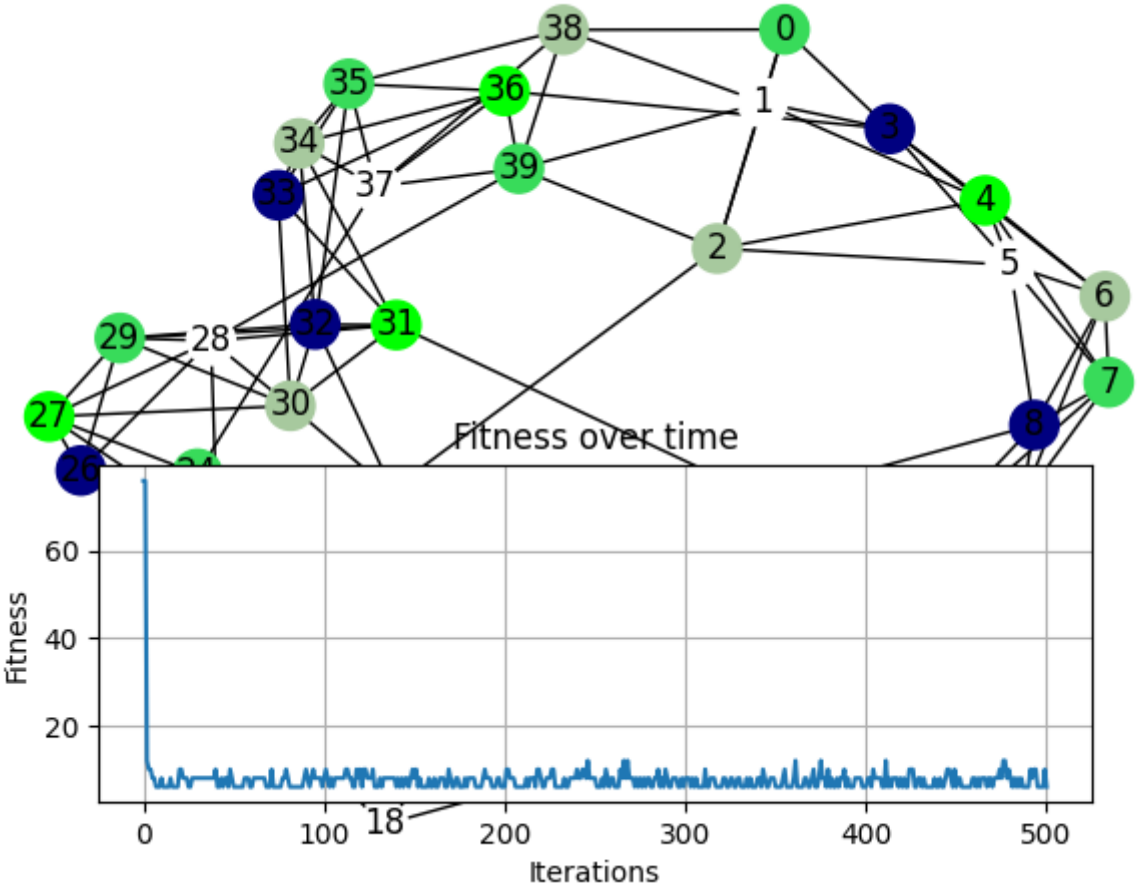

9
10
6
7
8
4
13
3
15
14
16
18
33
35
25
Fitness
60
40
0
0.0
0.5
1.0
2.0
2.5
3.0
3.5
4.0
Iterations

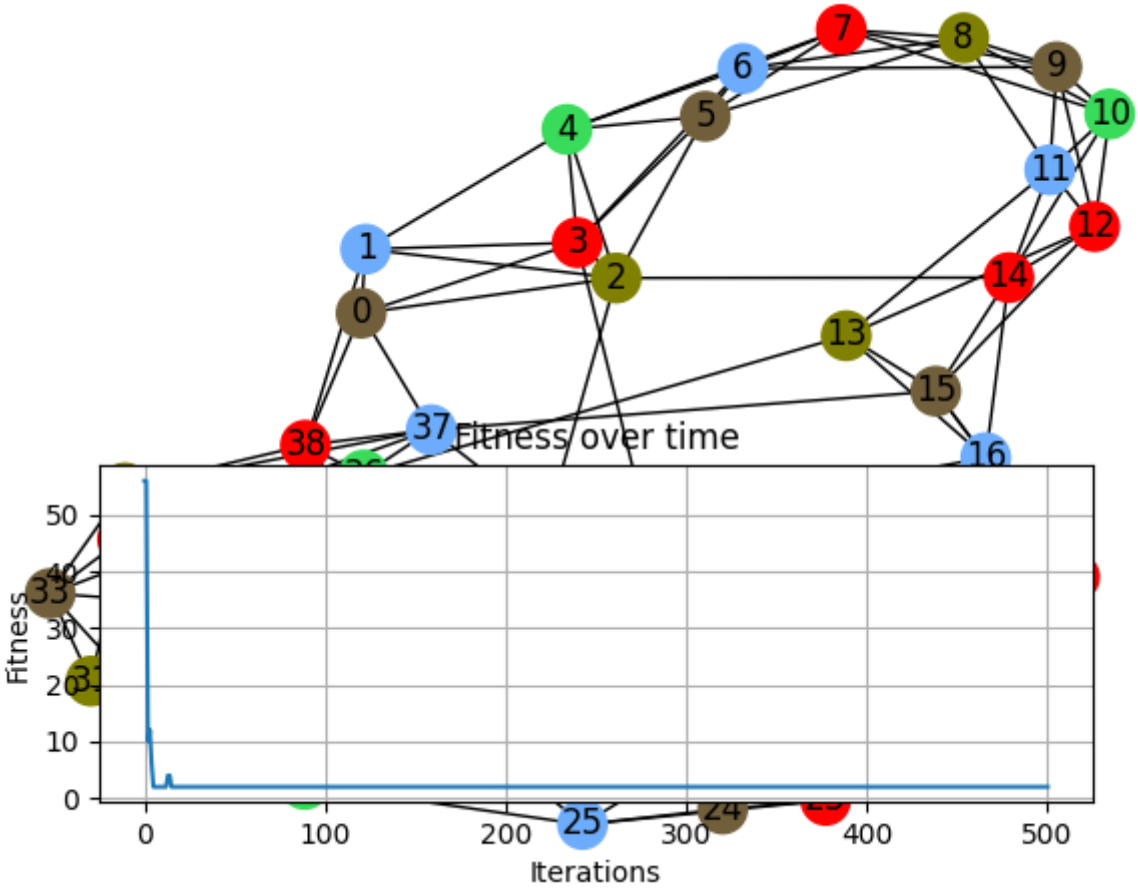

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
10
13
14
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
39

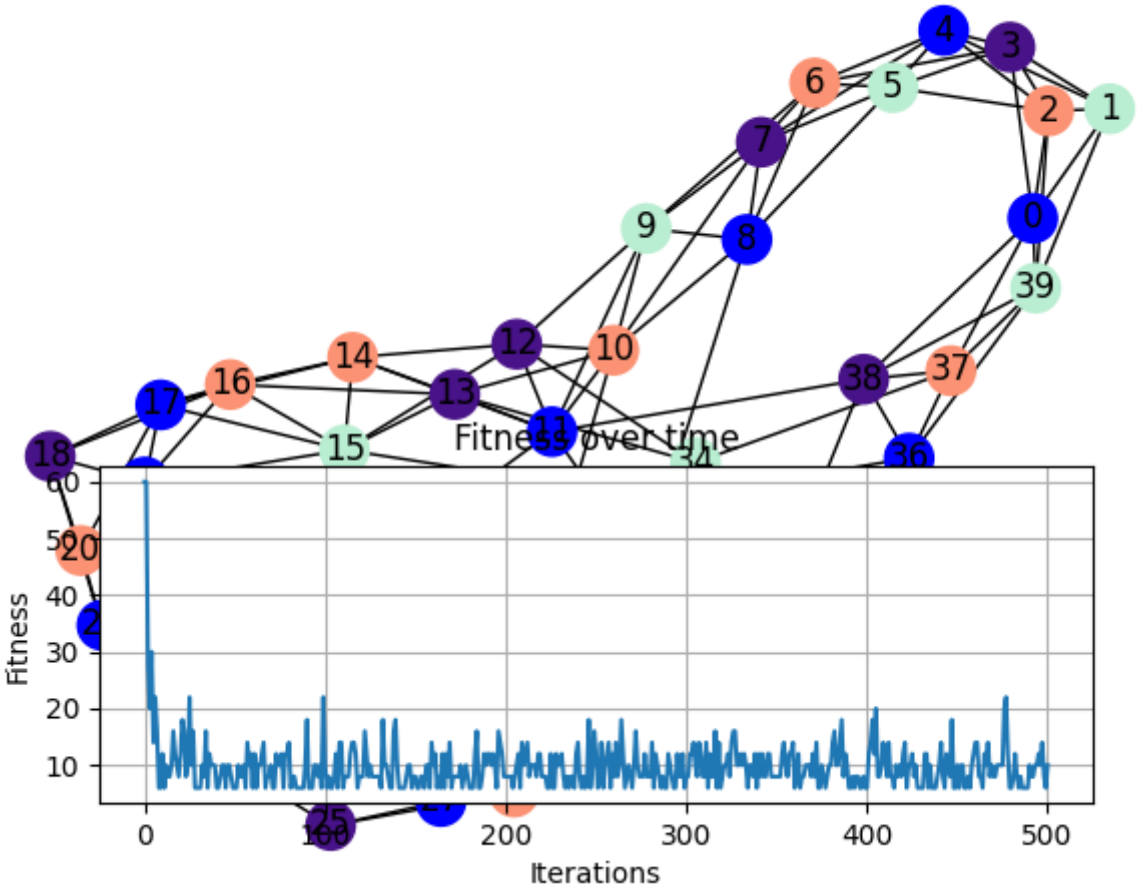

Fitness over time
Fitness
Iterations
0
10
30
40
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

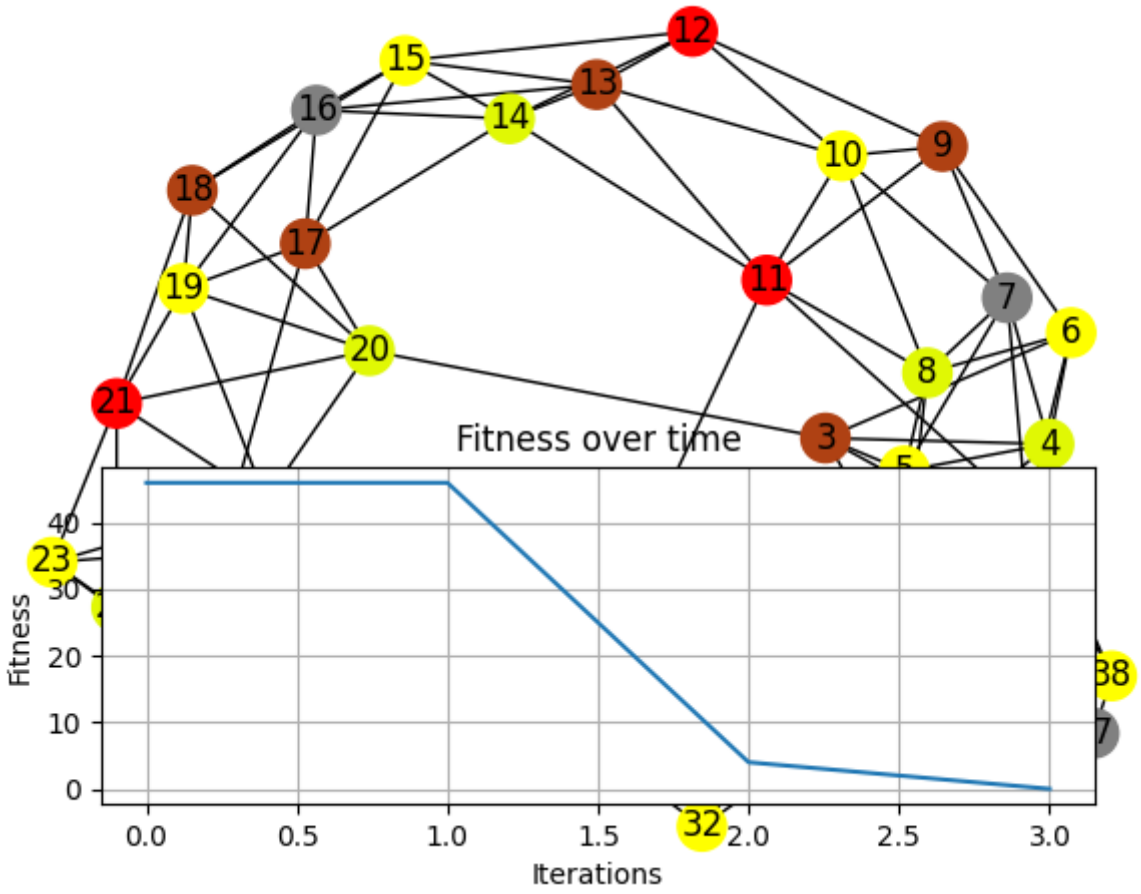


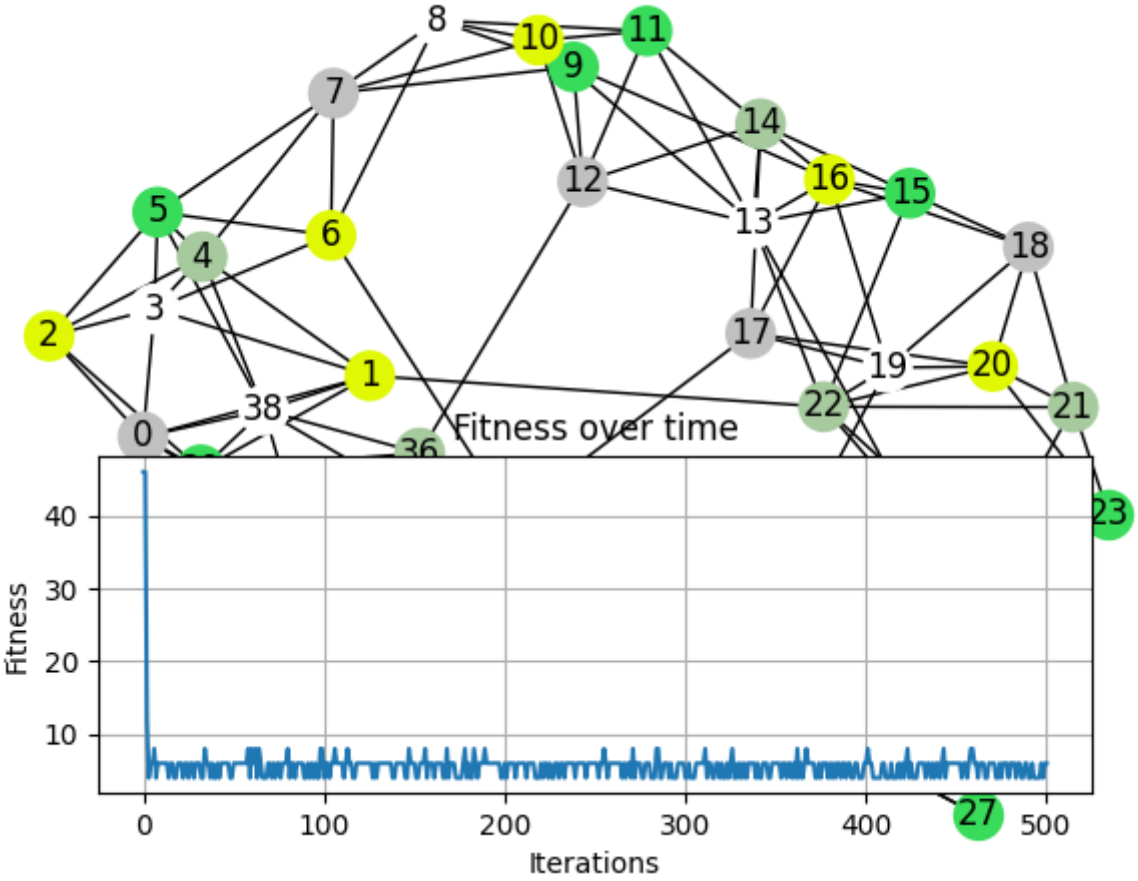
30% Aggressive

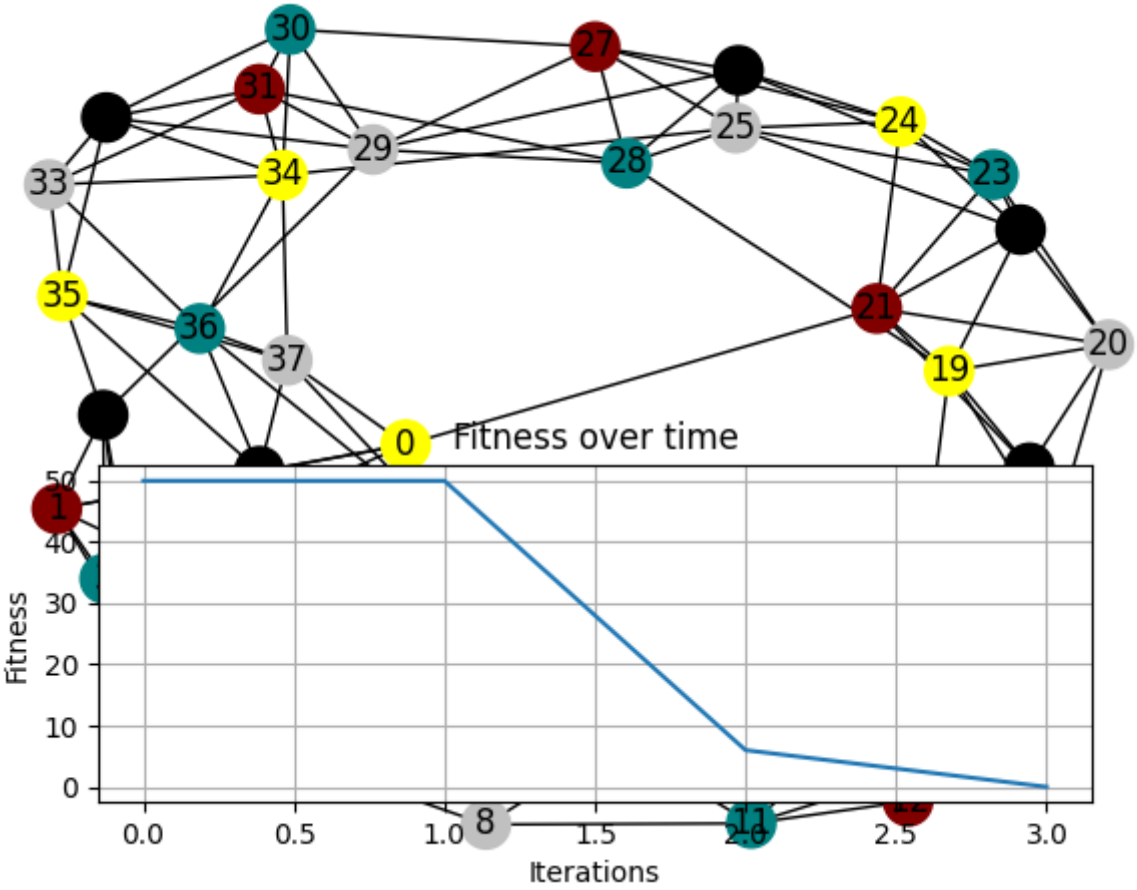

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
20
40
60
0
1
2
3
4
5
6
7
8
18
26
27
28
29
30
31
32
33
34
35
36
37
38
39

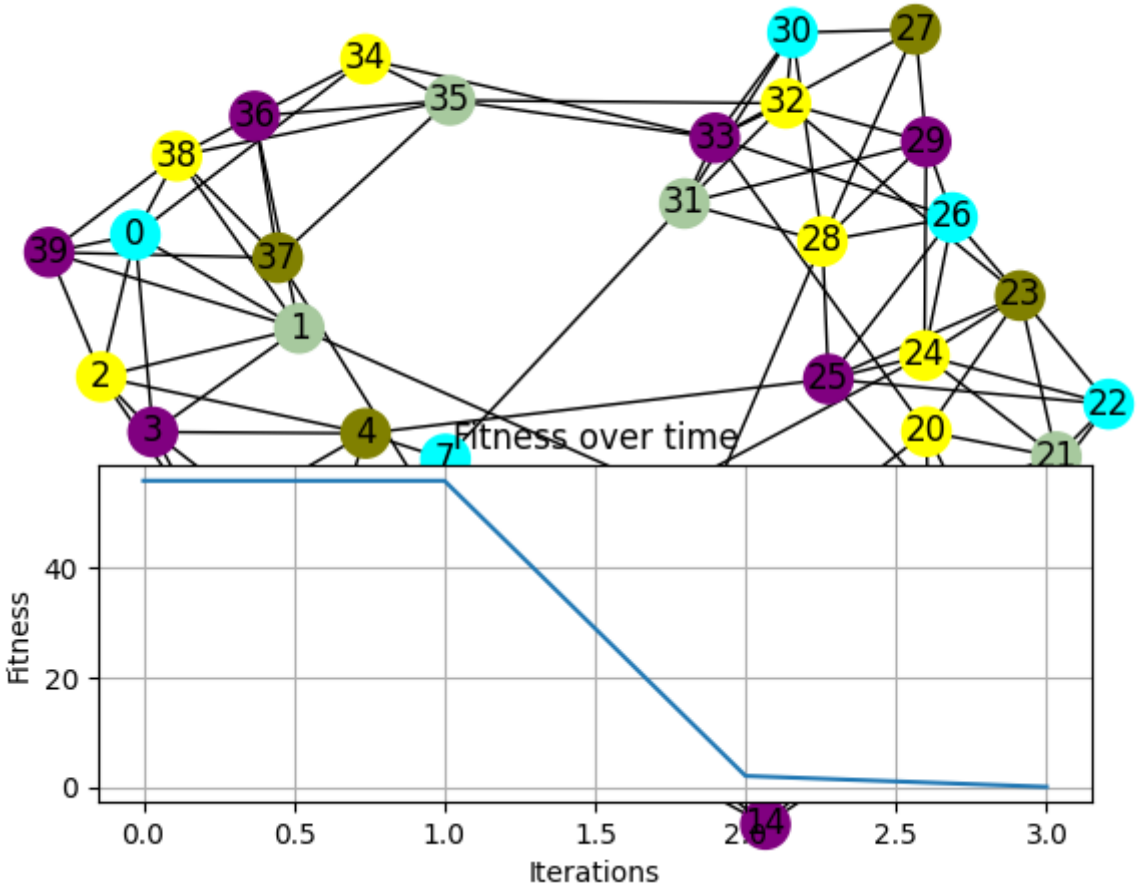

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0
100
200
300
400
500

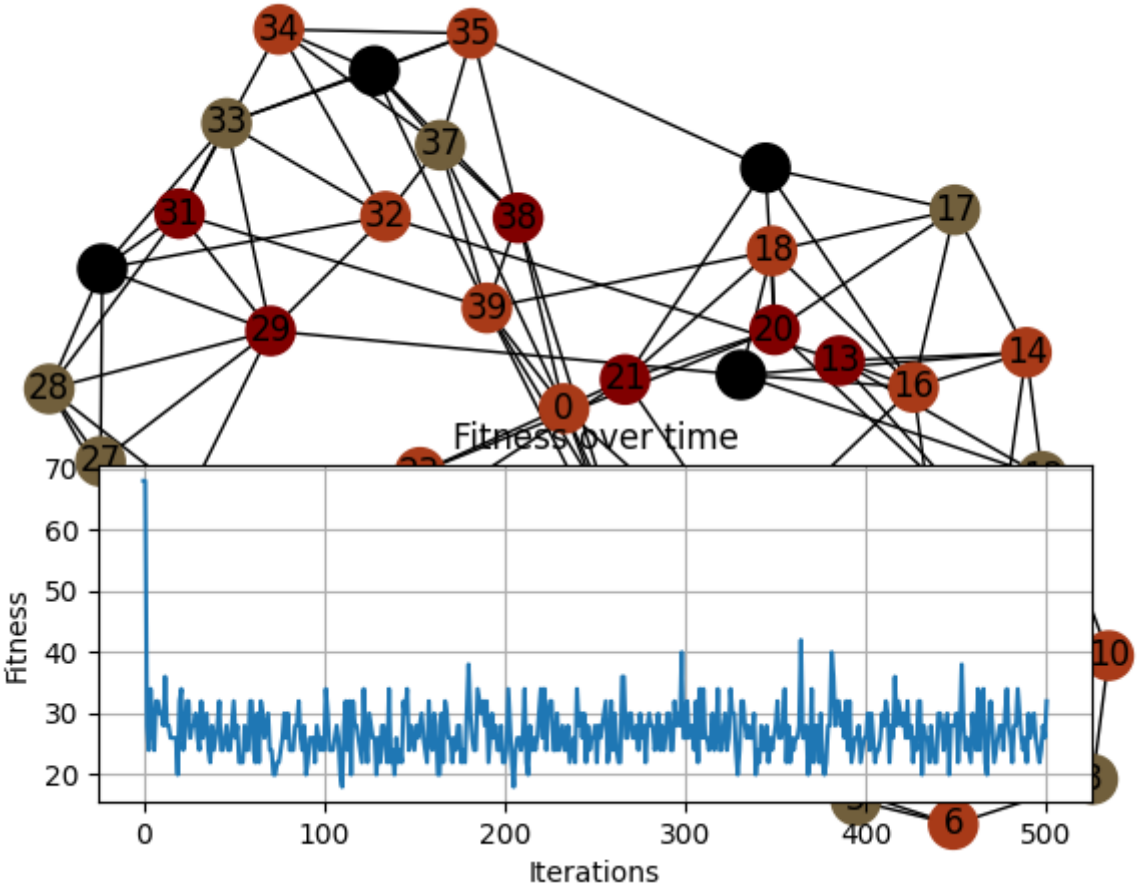

Fitness over time
Fitness
Iterations
0
200
300
400
500
10
20
30
40
50
60

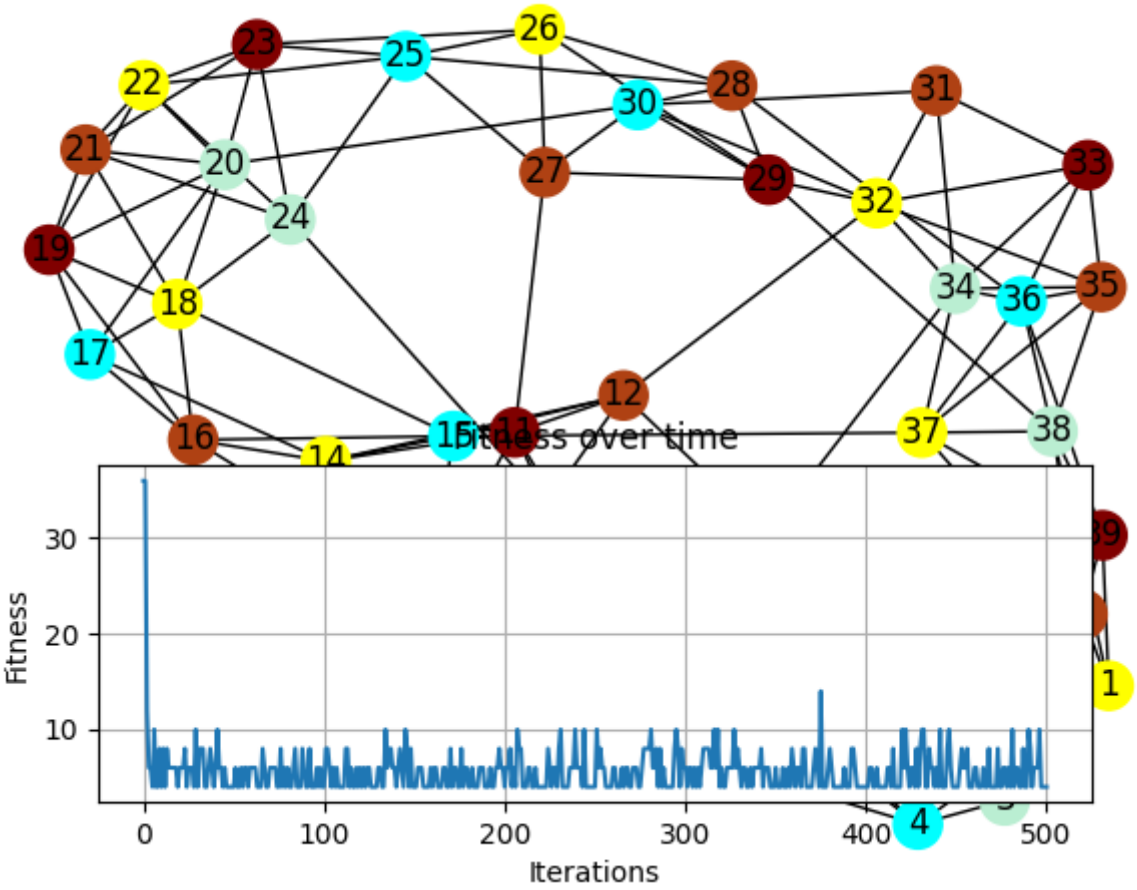

Fitness over time
Fitness
Iterations
0
10
20
30
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0

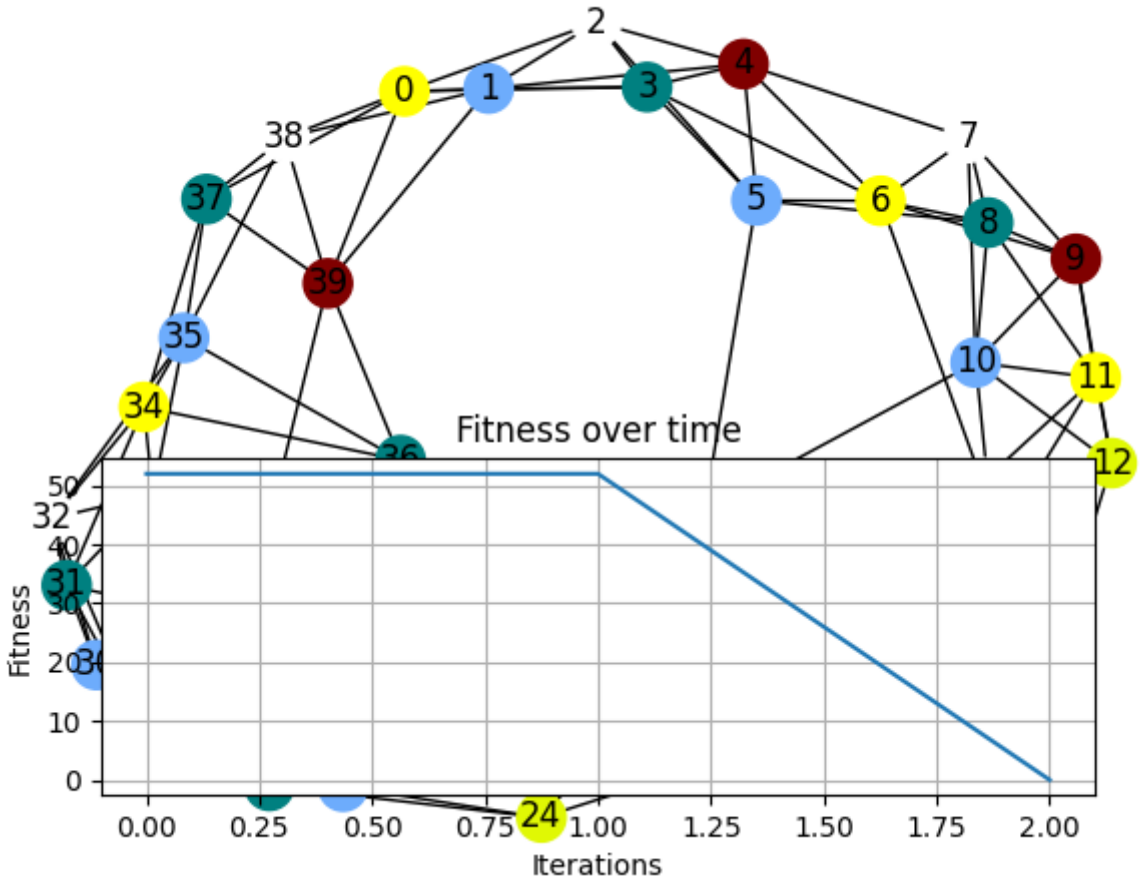

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
10
20
30
40


Fitness over time
Fitness
Iterations
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
10
20
30
40
50
30
27
31
25
24
29
28
23
33
34
35
36
37
21
20
19
0
1
8
11

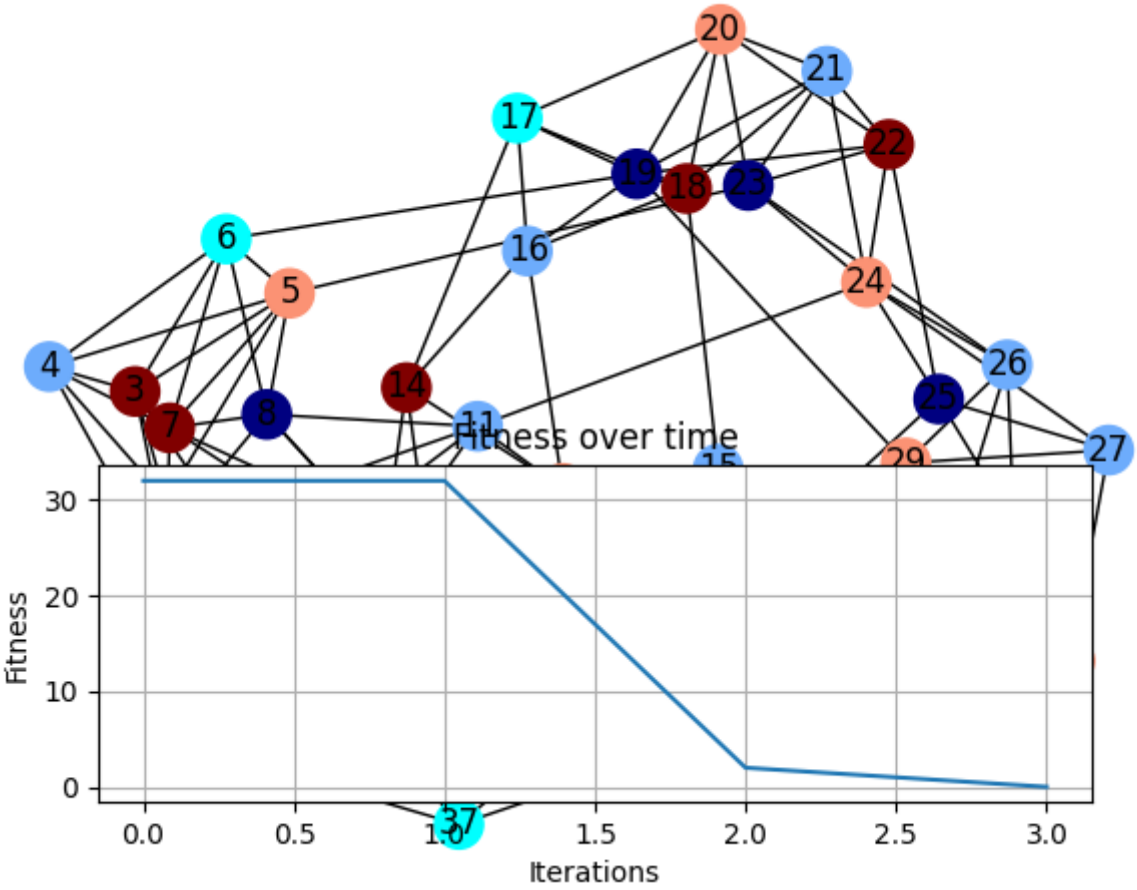

Fitness over time
Fitness
Iterations
0
20
40
0.0
0.5
1.0
1.5
2.5
3.0

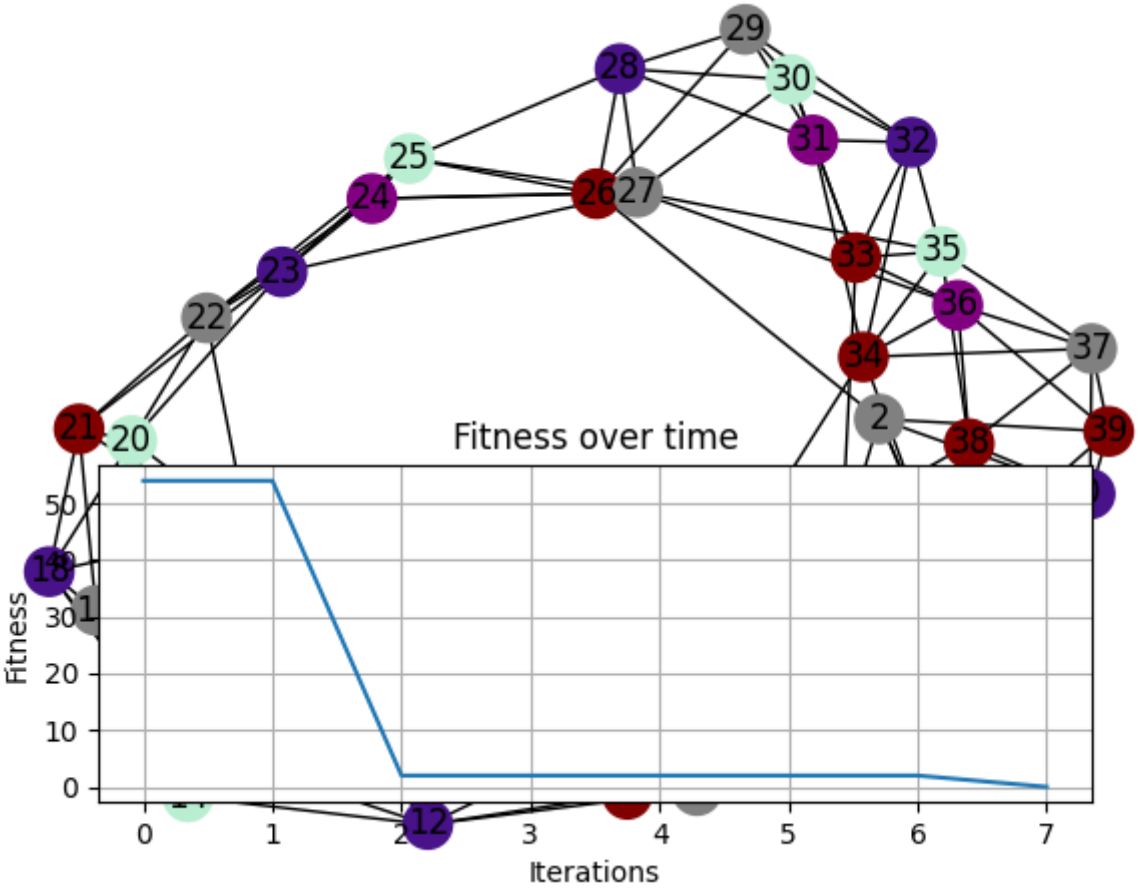

Fitness over time
Fitness
Iterations
70
60
50
40
30
20
0
100
200
300
400
500
34
35
33
37
31
32
38
17
18
39
29
20
13
14
28
21
16
0
27
10
6

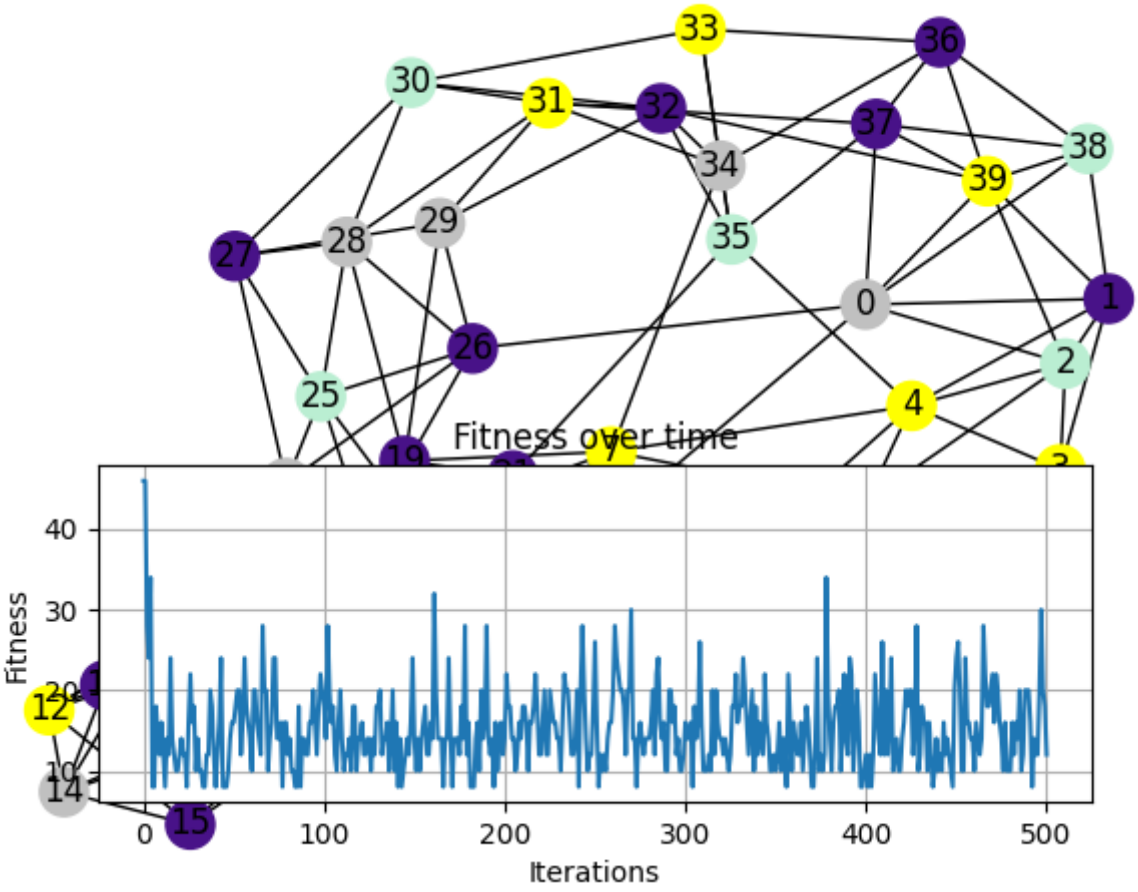

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
10
20
30

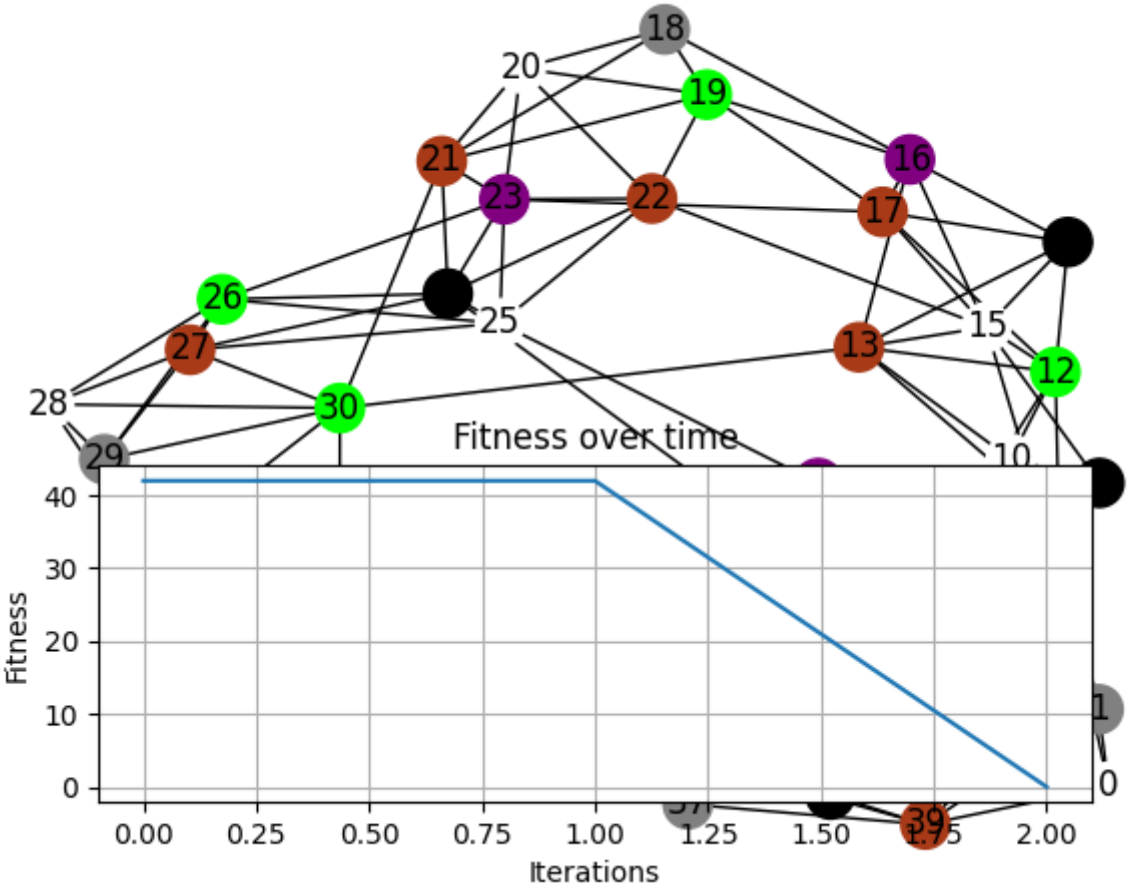


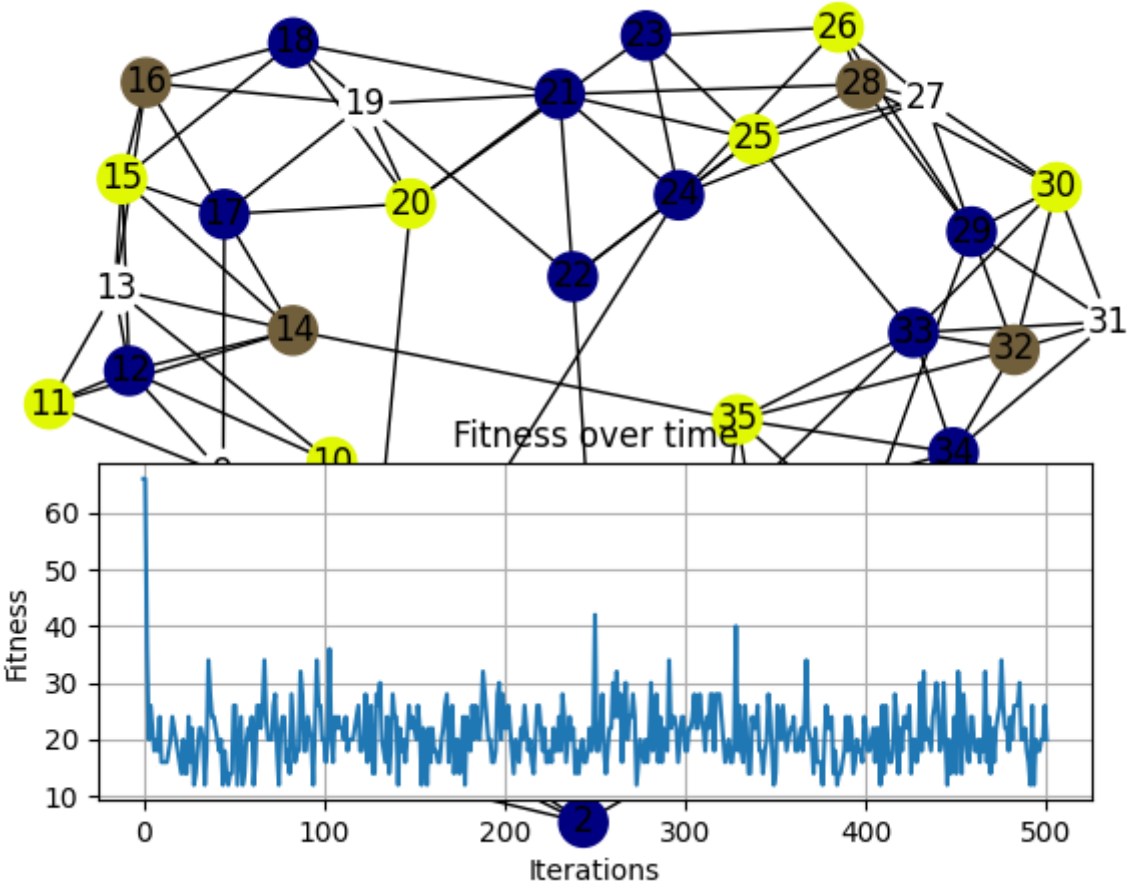
50% Aggressive

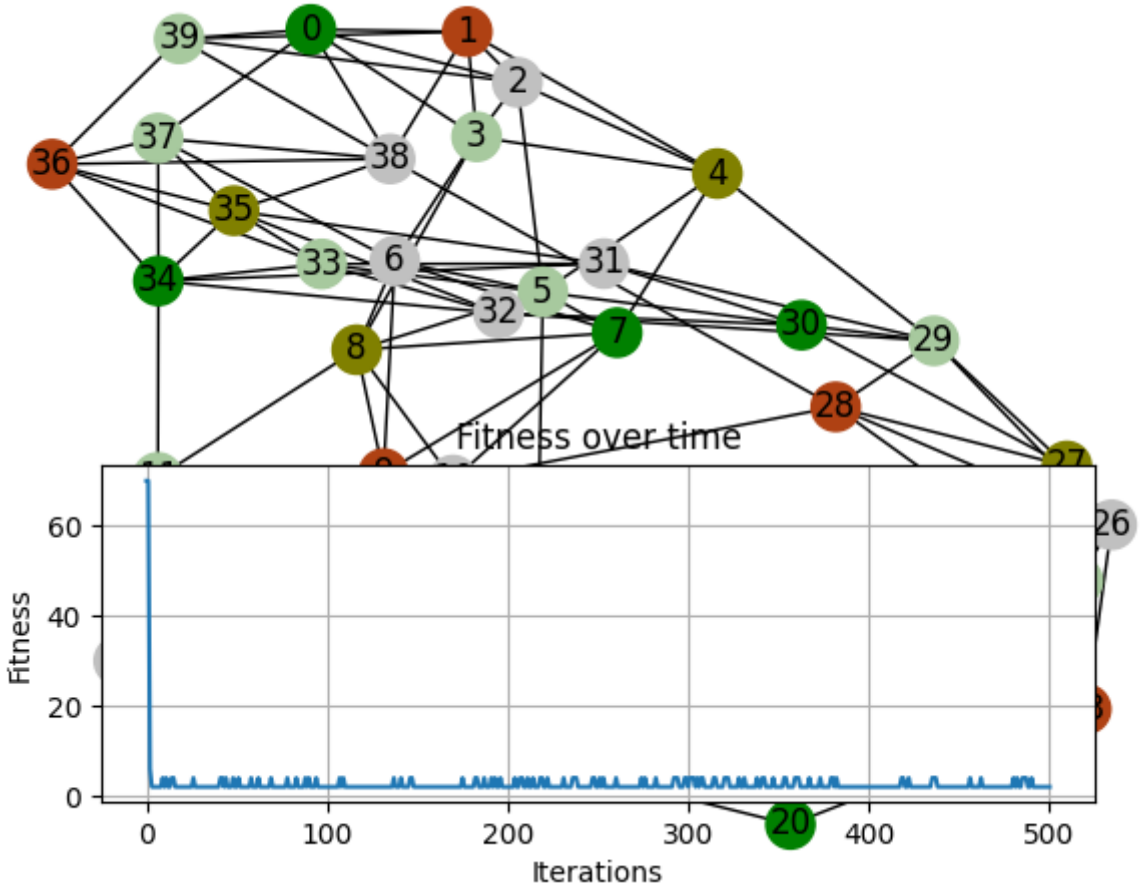

Fitness over time
Fitness
Iterations
0
10
20
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0

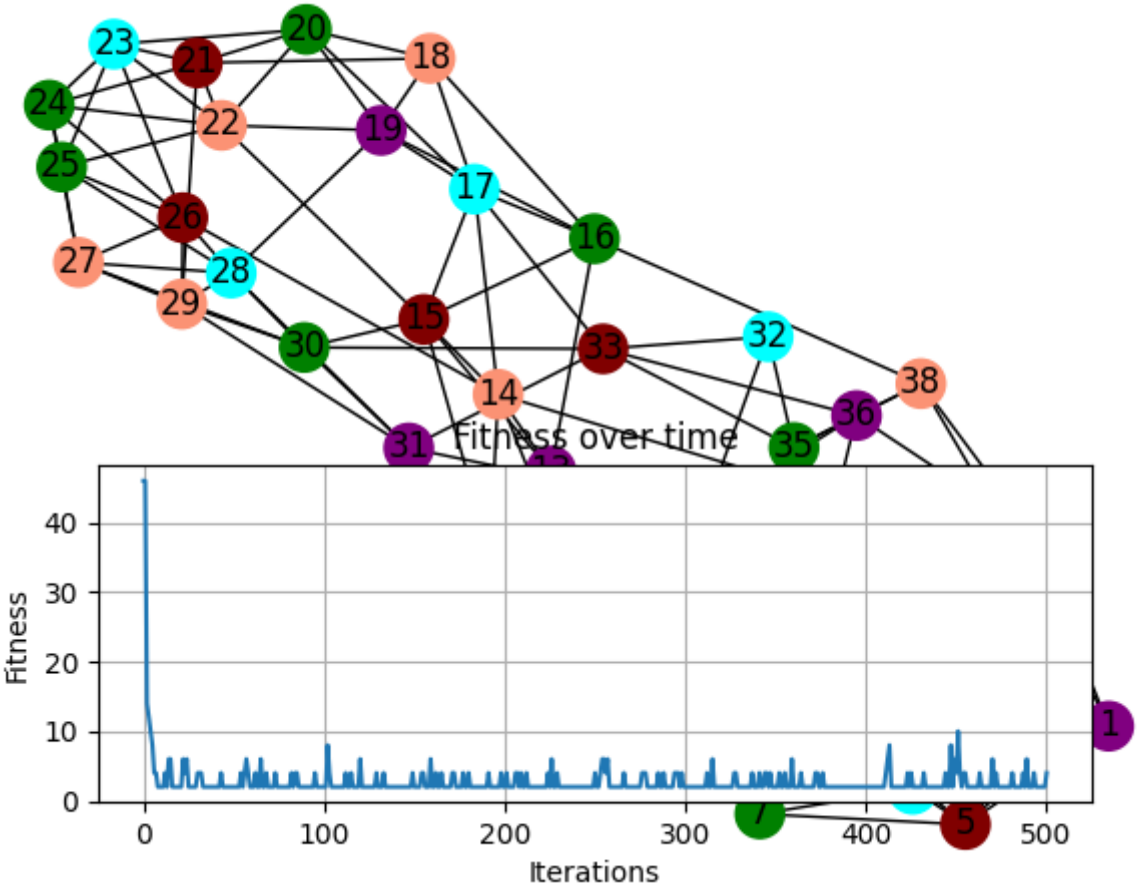

Fitness over time
Fitness
Iterations
0
10
20
30
50
0
1
2
3
4
5
6
7
29
28
30
31
32
25
24
26
27
33
35
23
36
22
37
34
21
20
2
38
39
18
12

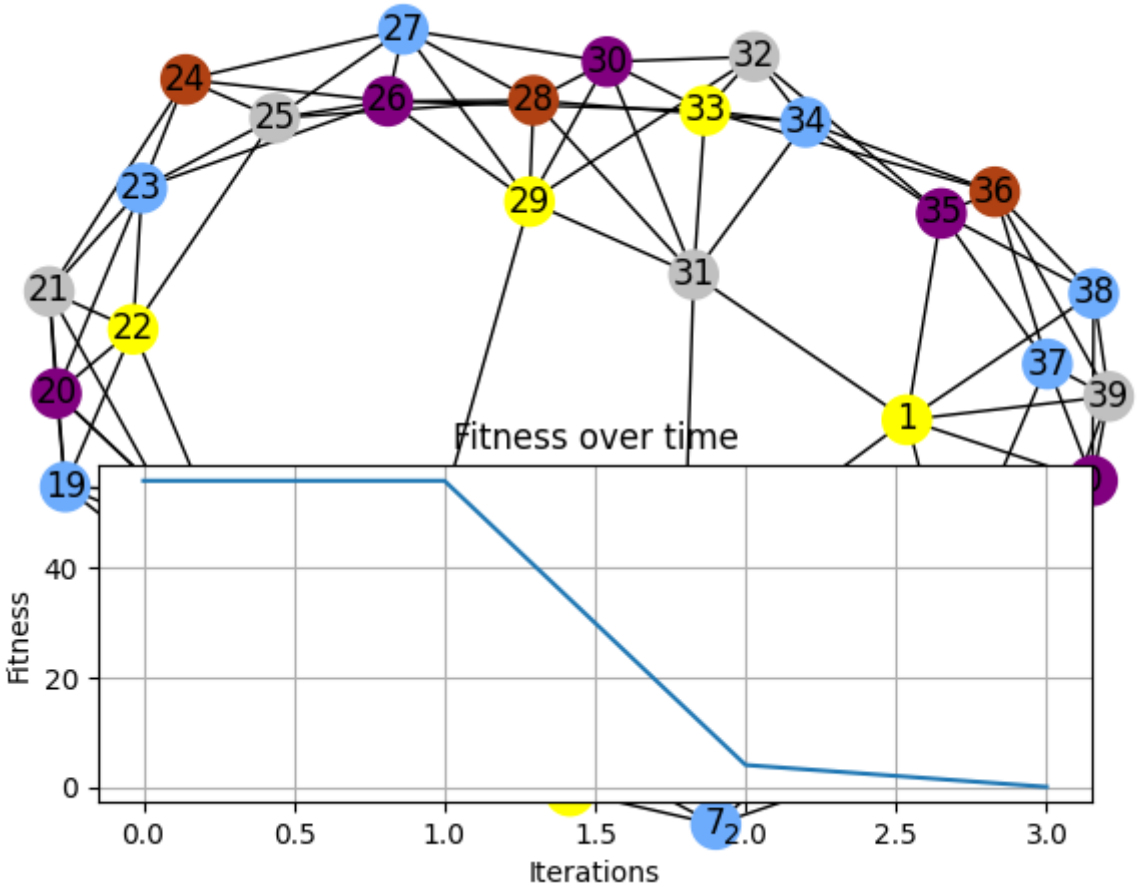

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
10
20
30
40
0
1
2
4
12
14
15
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

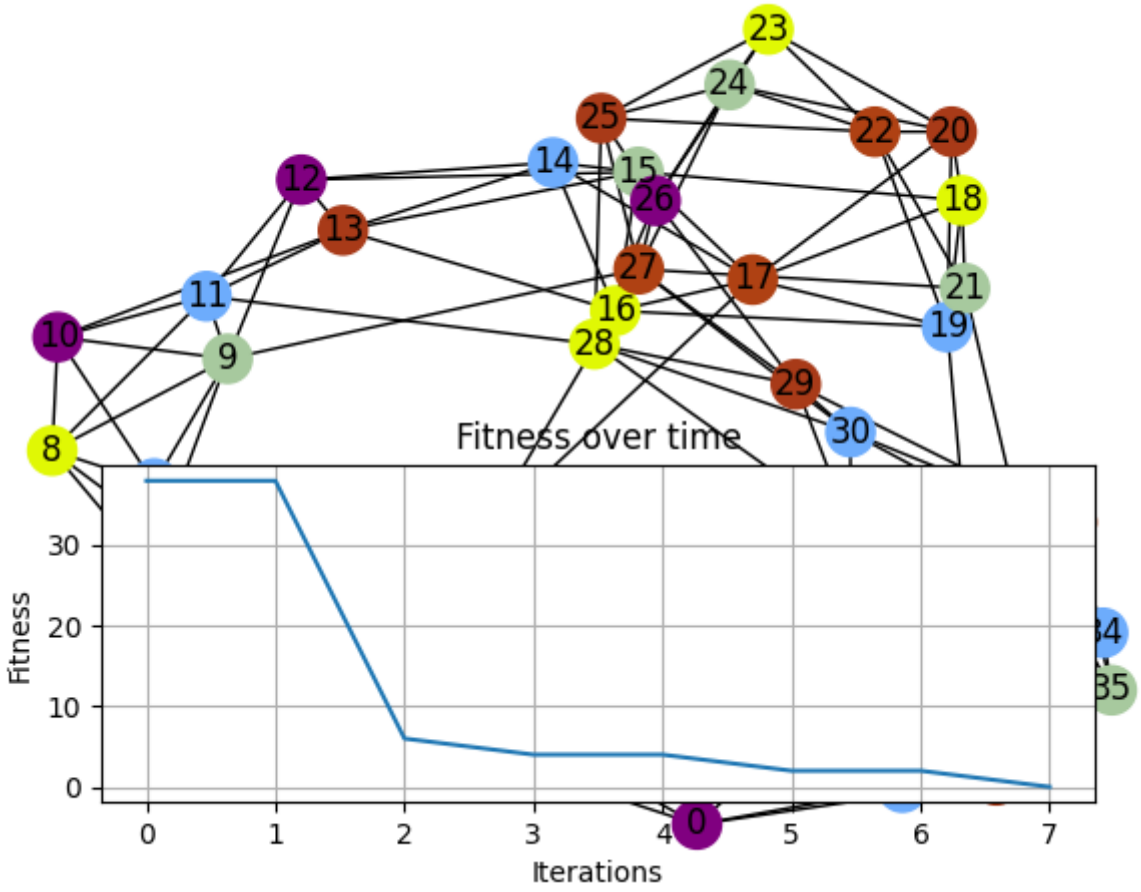

Fitness over time
Fitness
Iterations
0
10
20
30
40
0.00
0.25
0.50
0.75
1.00
1.25
1.50
2.00

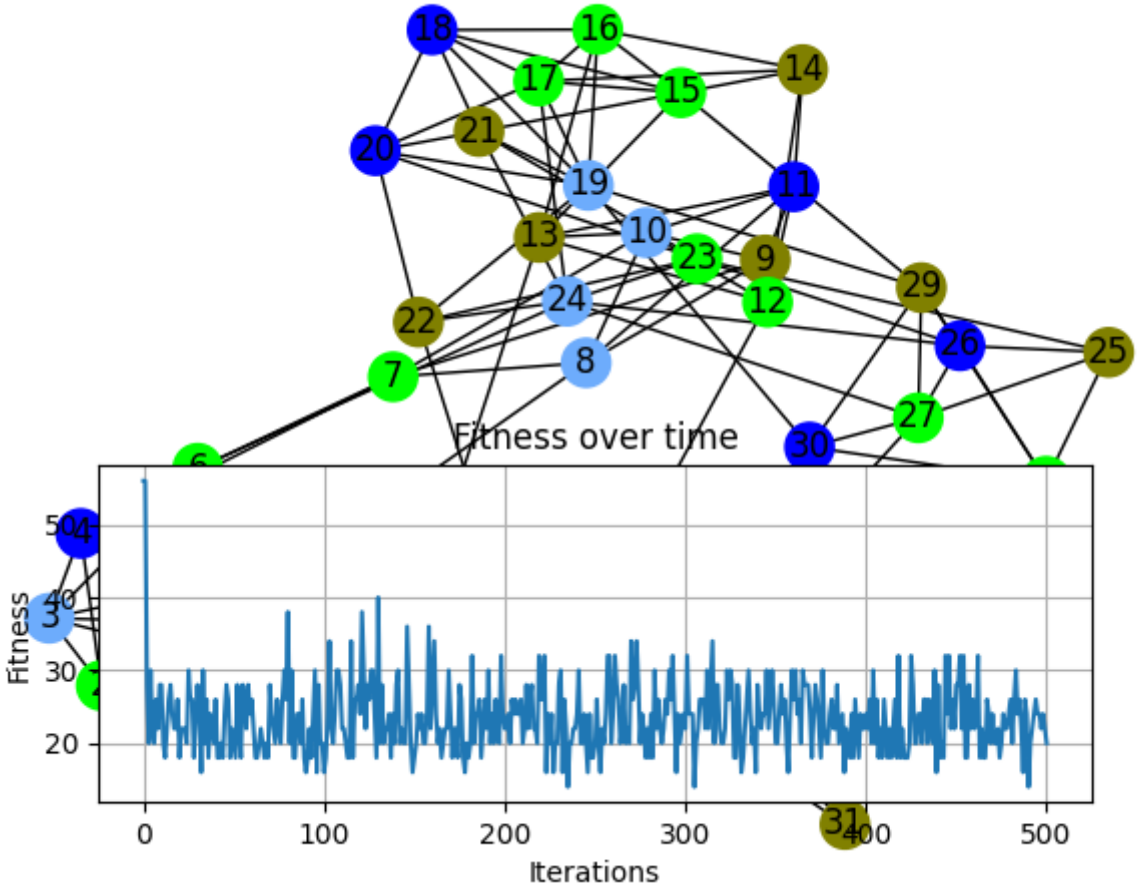

Fitness over time
Fitness
Iterations
10
20
30
40
50
60
0
100
200
300
400
500


Fitness over time
Fitness
Iterations
0
20
40
60
0
100
200
300
400
500

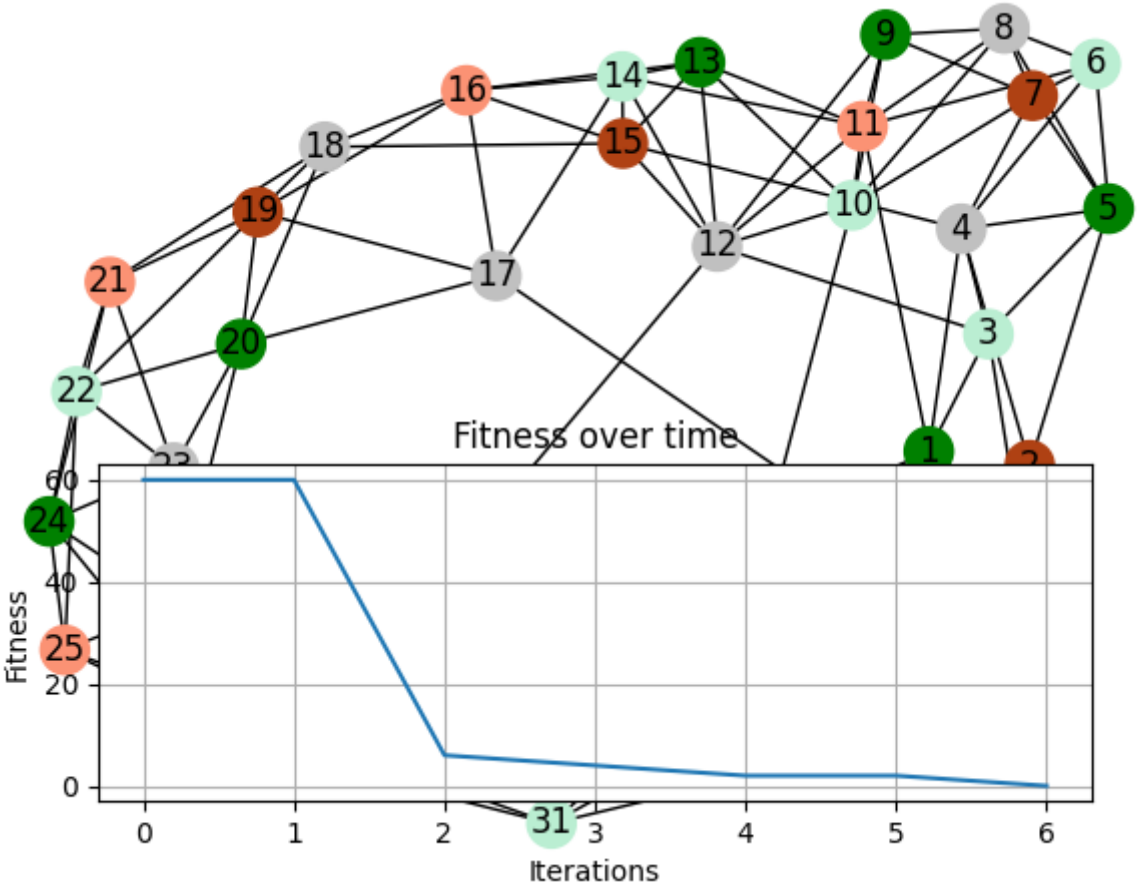

Fitness over time
Fitness
Iterations
0
10
20
30
40
0
100
200
300
400
500

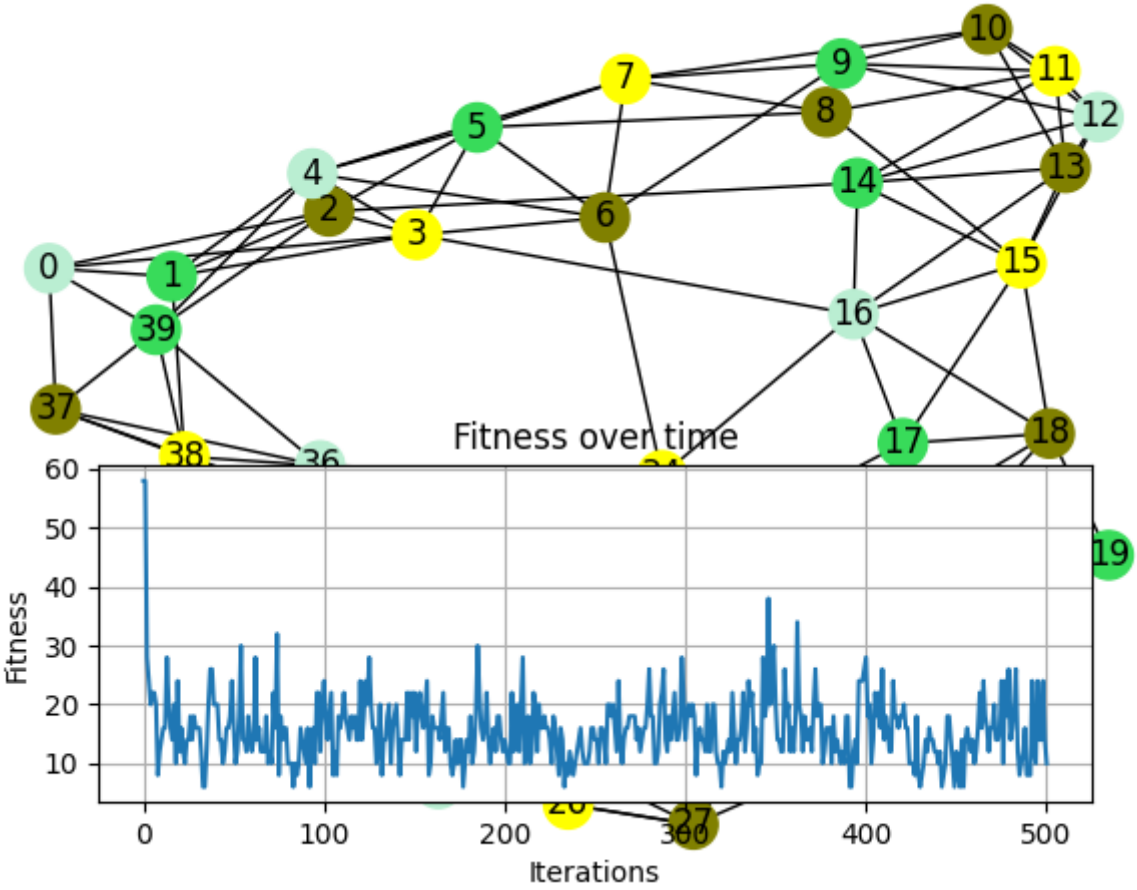

Fitness over time
Fitness
Iterations
0
20
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1
7
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

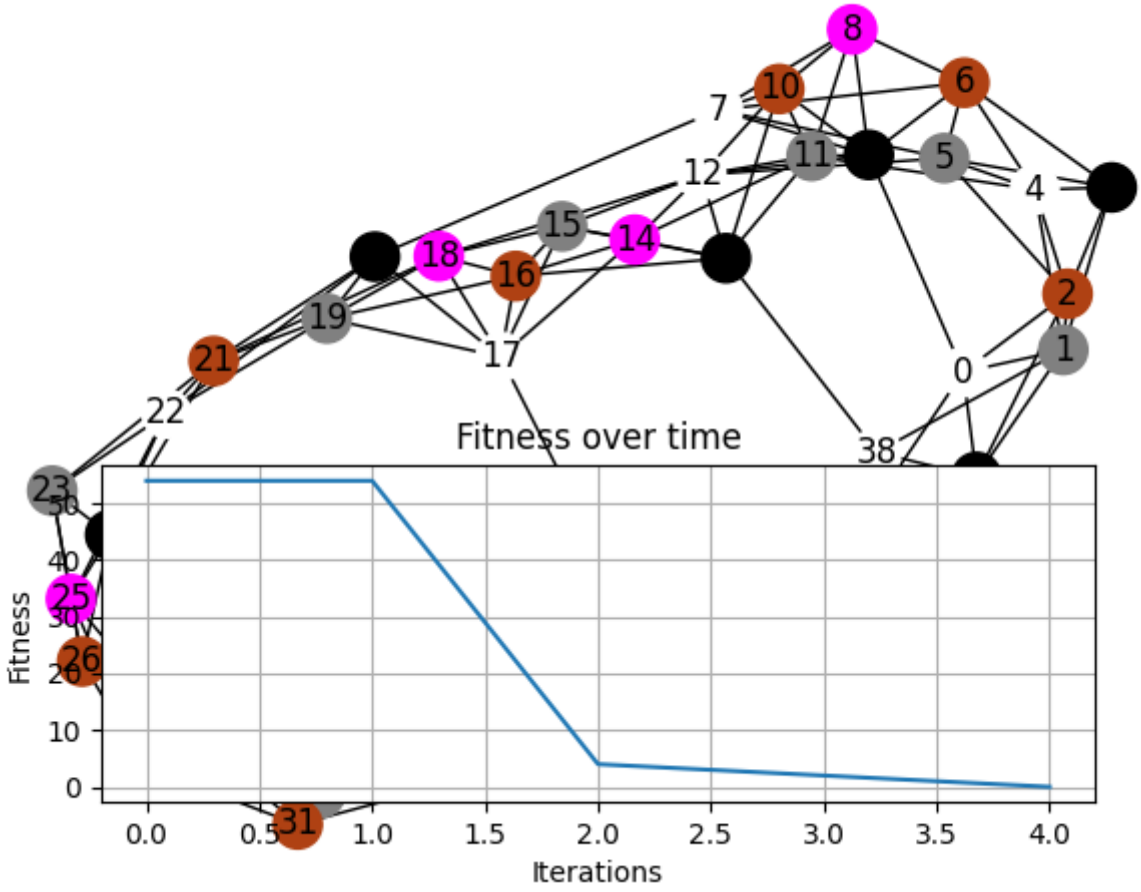

Fitness over time
Fitness
Iterations
0
10
20
30
0
1
2
3
4
5
6
7

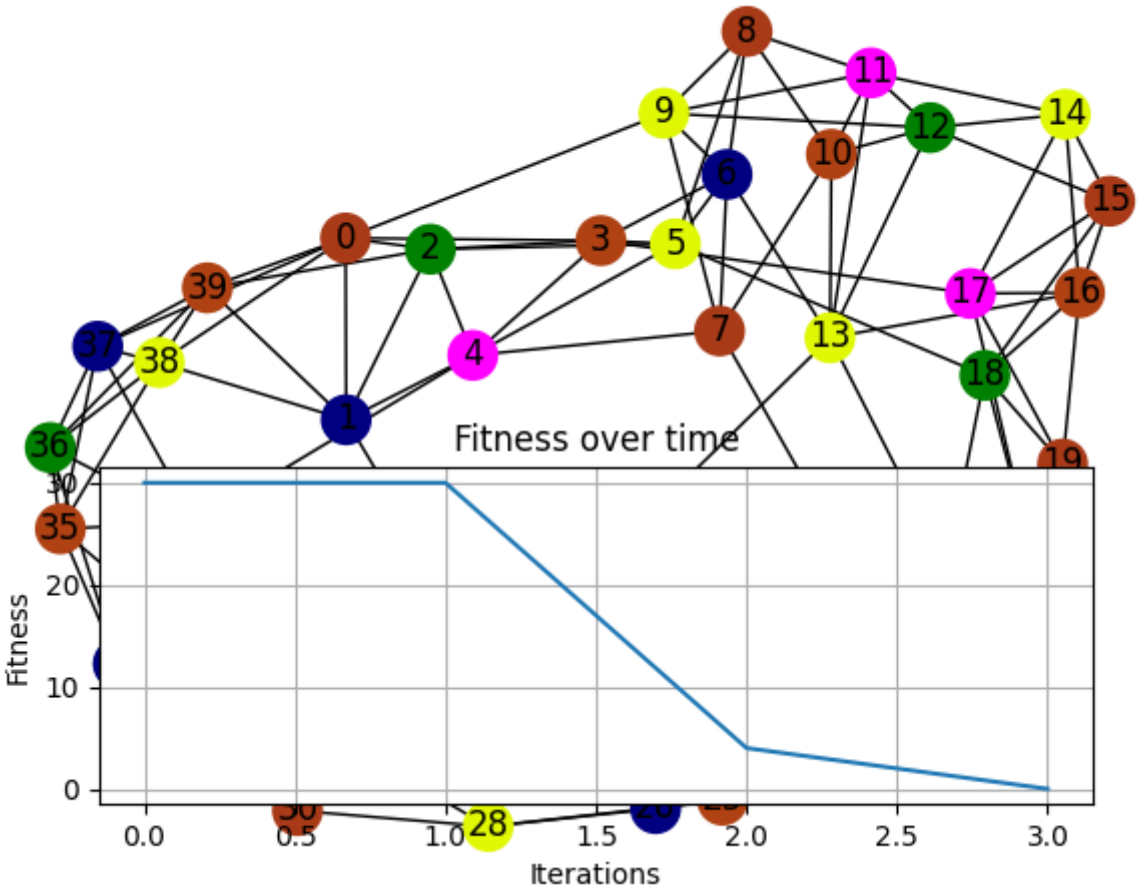


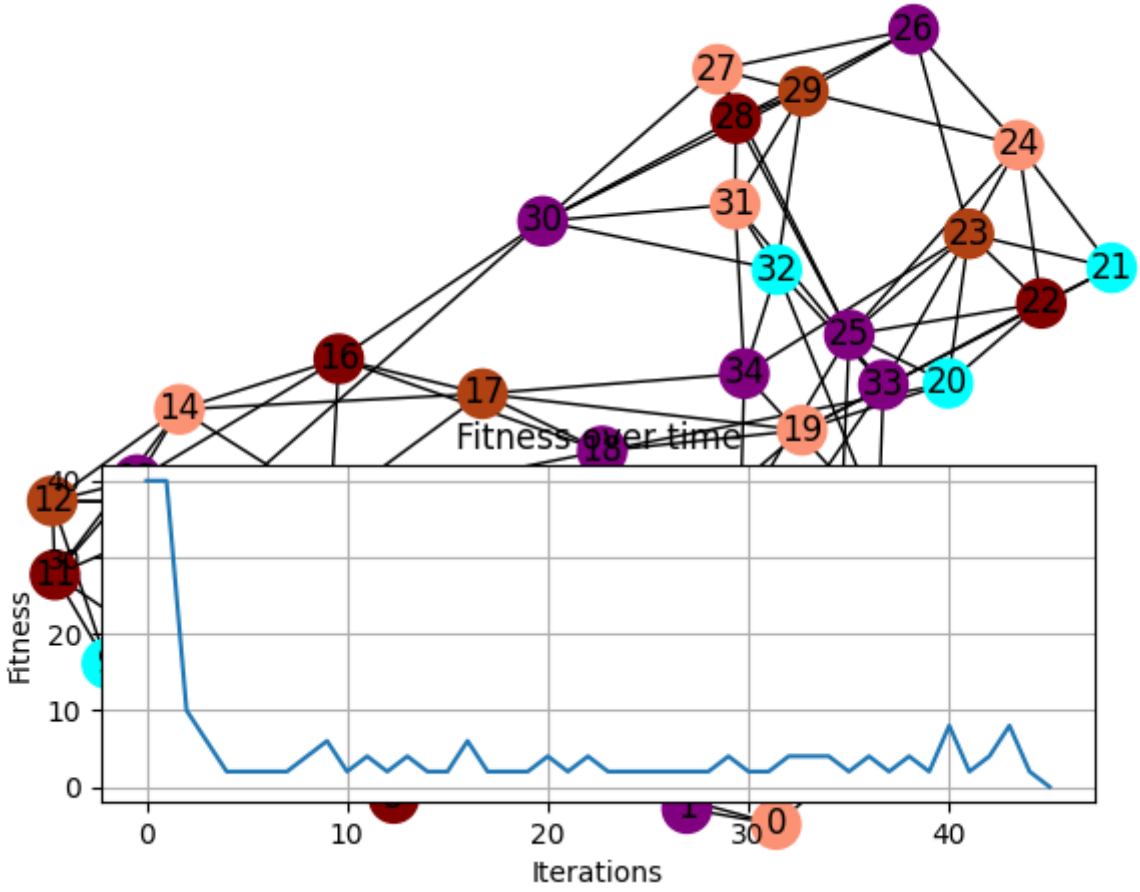
70% Aggressive

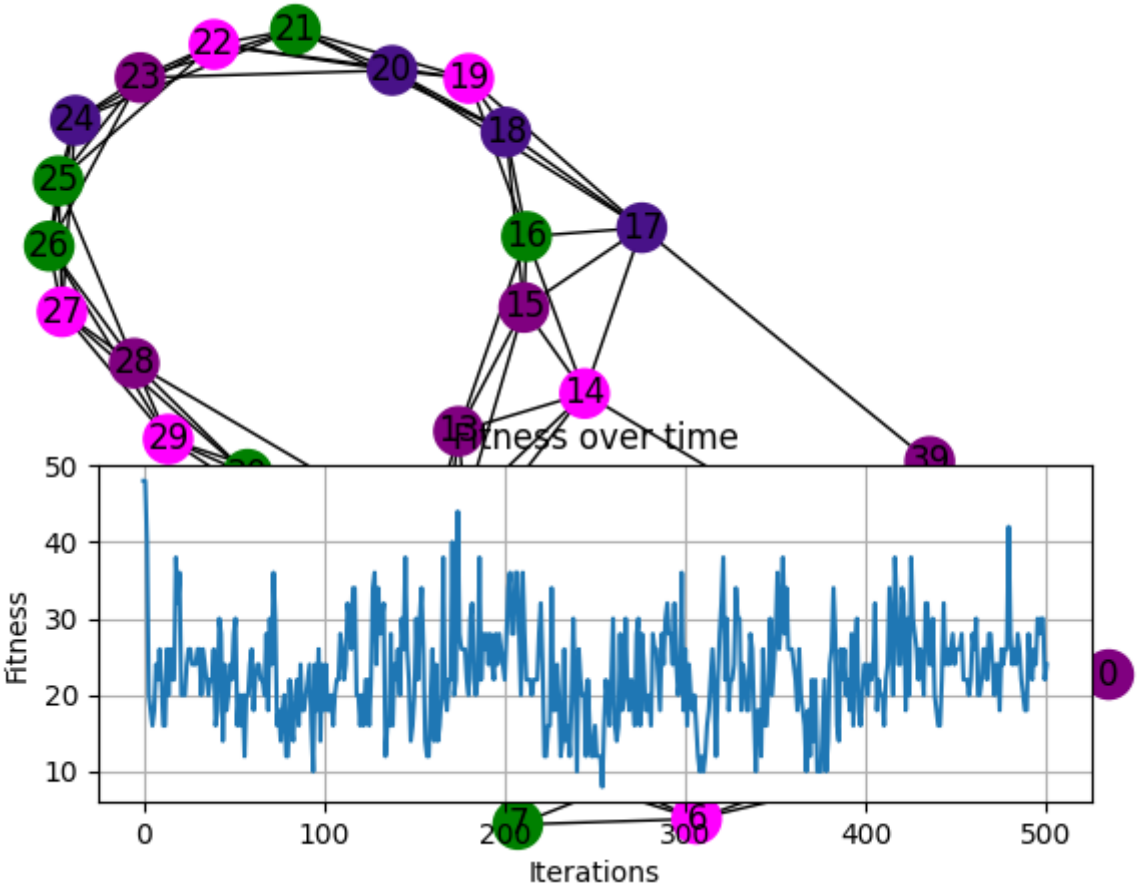

Fitness over time
Fitness
Iterations
0
20
40
60
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
24
25
31

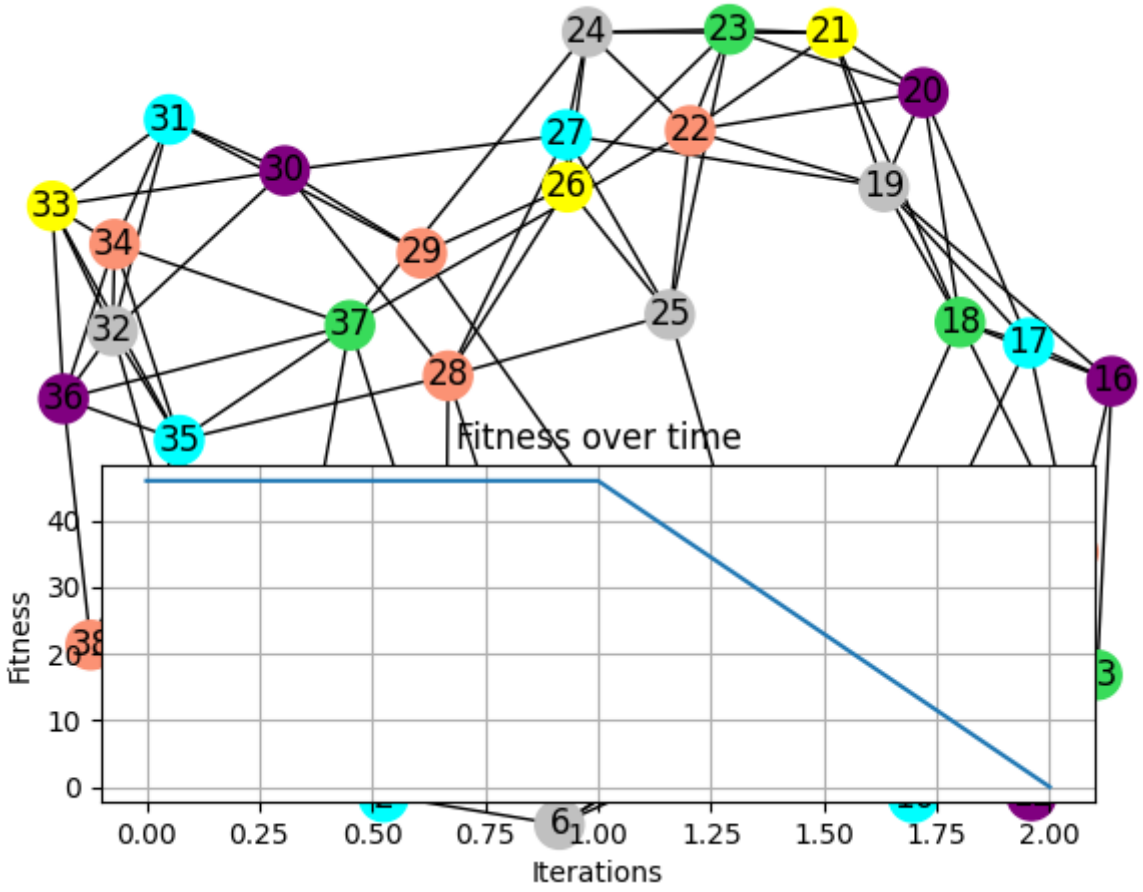

Fitness over time
Fitness
Iterations
0
100
200
300
400
500
10
20
30
40
50
60
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
27
37
38
39

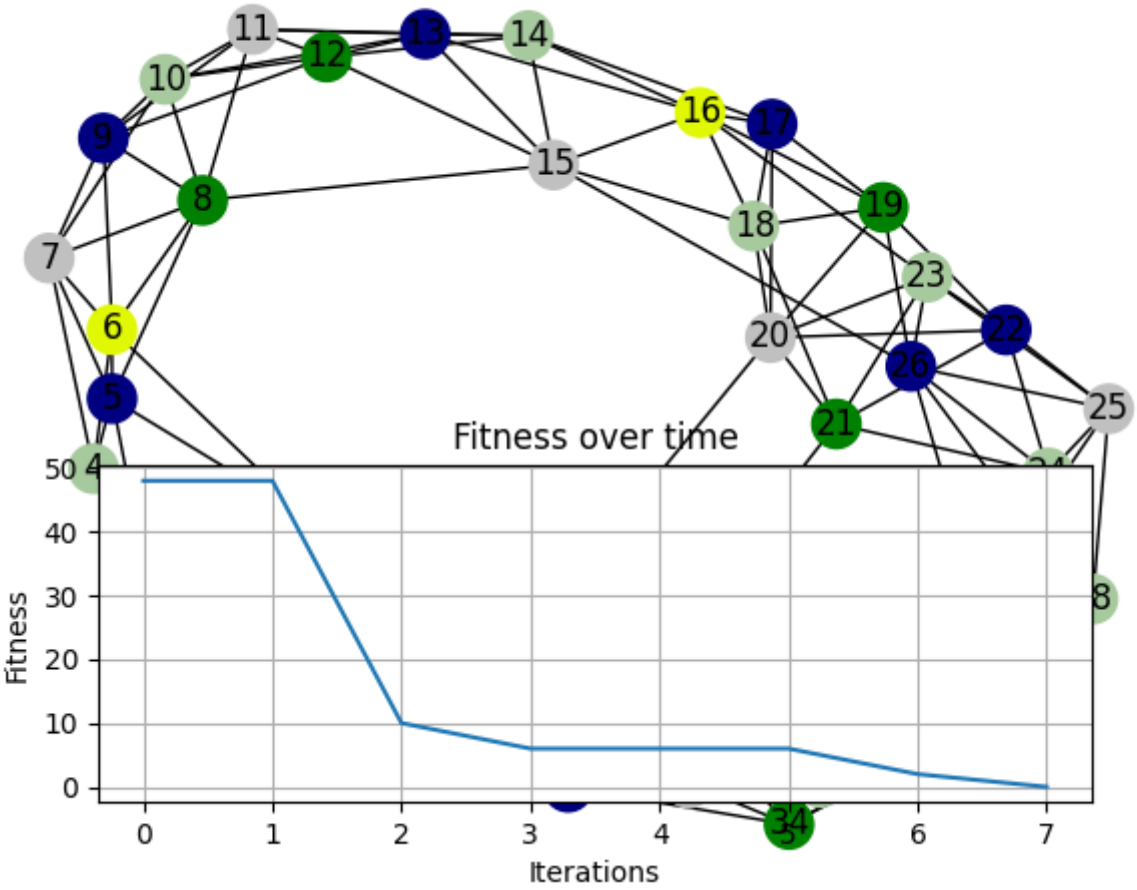

Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

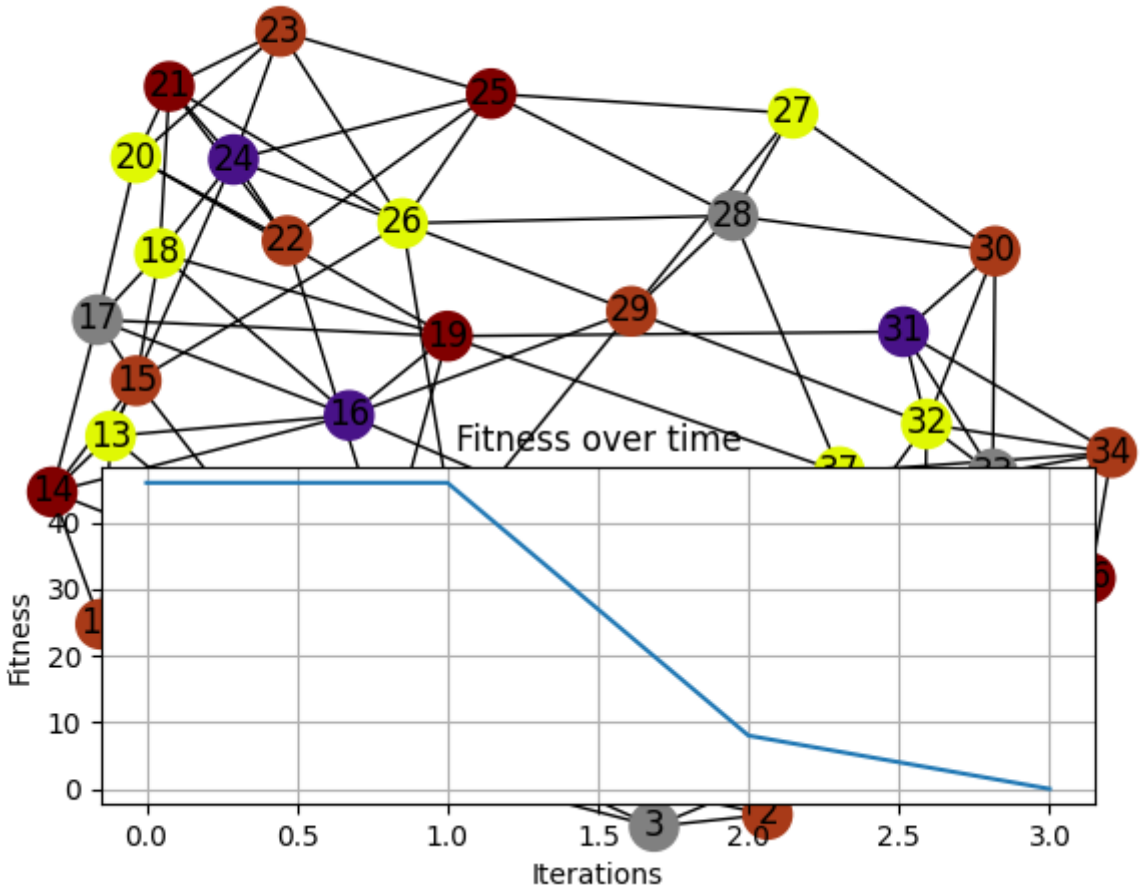

Fitness over time
Fitness
Iterations
0
10
20
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
28
35
36
37
38
39

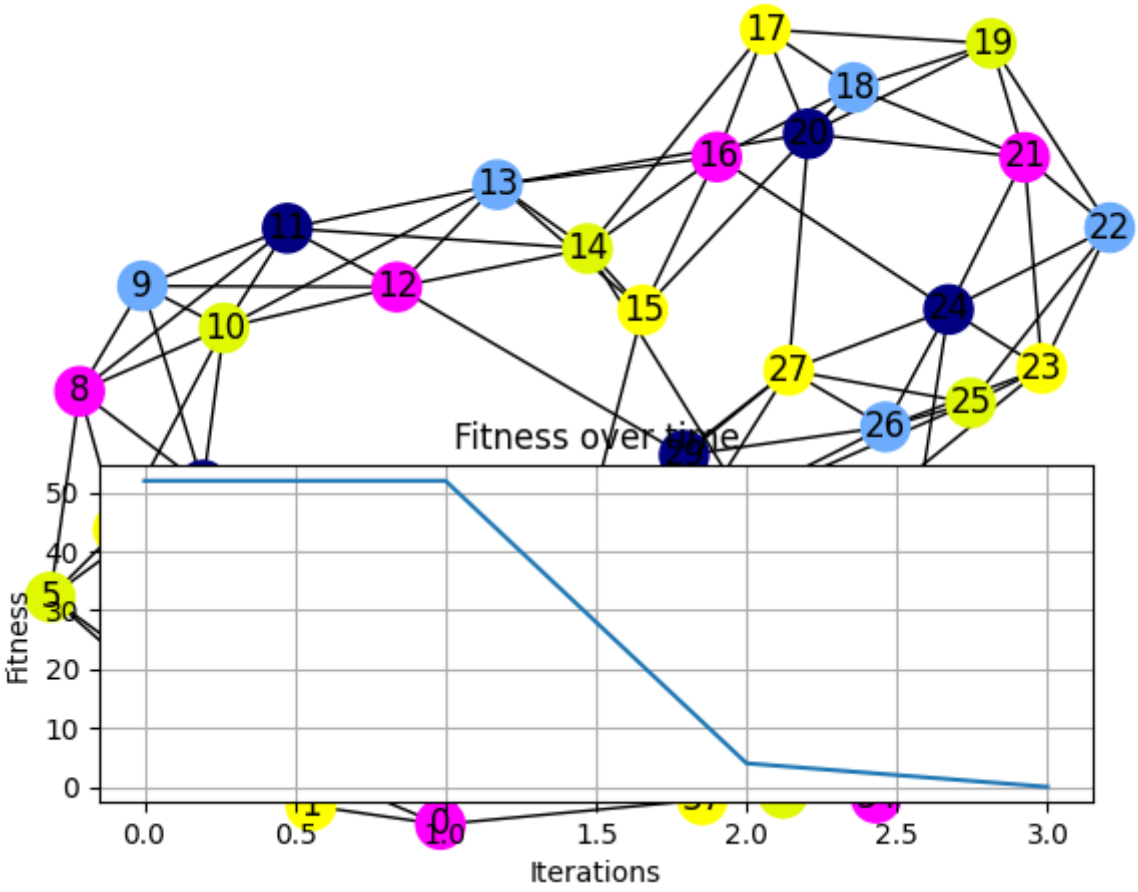

Fitness over time
Fitness
Iterations
0
10
20
30
40


Fitness over time
Fitness
Iterations
50
40
30
20
10
0
100
200
300
400
500


Fitness over time
Fitness
Iterations
0
10
20
30
40
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Fitness over time
Fitness
Iterations
0
10
20
30
40
50
0
1
2
3
4
6
7


Fitness over time
Fitness
Iterations
0
10
20
30
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0



this report is probably overcomplicated too