

# Evaluating the maze navigation problem in random and fixed starting positions using Objective based fitness and Novelty Search approach

Arslan Siddique  
a.siddique@innopolis.university

## Introduction

The objective of this project is to evaluate the maze navigation problem using objective based fitness and Novelty Search method with random starting points and 9 approximately evenly spaced points. The main inspiration for this project comes from the work of [1] who has evaluated the performance of Objective based fitness and Novelty Search method for a medium and hard maze navigation environment. Iaroslav [2] implemented the work in GO programming language and provided the open source code. He also provided the open source code on this work in Python language in [3]. This code has been used and modified for my work. Only hard maze environment has been focussed in this project.

## Experimental setup

### The task:

The maze navigation problem is a classic computer science problem that is closely related to creating autonomous navigation agents that can find a path through ambiguous environments. The maze environment is an illustrative domain for the class of problems that have a deceptive fitness landscape. This means that the goal-oriented fitness function can have steep gradients of fitness scores in dead ends in the maze that are close to the final goal point. Such areas of the maze become the local optima for objective-based search algorithms that may converge in these areas. When the search algorithm converges in such deceptive local optima, it cannot find an adequate maze-solver agent. In the following example, you can see a two-dimensional maze with local optima dead ends, which are shaded in pink:

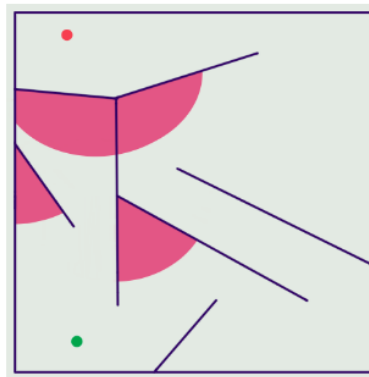


Fig. 1. Hard maze navigation environment in which local optima have been shown by pink color

### Sensors and actuators of agent:

The agent navigating through the maze is a robot equipped with a set of sensors, allowing it to detect nearby obstacles and get the direction to the maze exit. The motion of the robot is controlled by two actuators, which affect the linear and angular movement of the robot body. The actuators of the robot are controlled by an ANN, which receives input from the sensors and produces the two control signals for the actuators.

The following diagram shows the schematic drawing of the maze agent used in the maze-solving simulation:

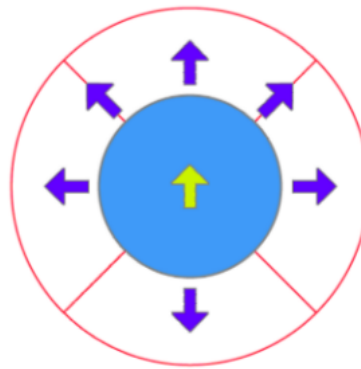


Fig 2. Maze agent diagram showing heading towards up indicated by yellow arrow

The specific radar sensor becomes activated when the line from the goal point to the center of the robot falls within its **field of view (FOV)**. The detection range of the radar sensor is limited by the area of the maze that falls into its FOV. Thus, at any given time, one of the four radar sensors is activated, indicating the maze exit direction.

The radar sensors have the following FOV zones relative to the robot's heading:

Sensors	Field of view, FOV (degrees)
Front	315 ~ 405
Left	45 ~ 135
Back	135 ~ 225
Right	225 ~ 315

The rangefinder sensor is a trace ray drawn from the center of the robot in a specific direction. It becomes activated when intersecting with any obstacle and returns a distance to the detected obstacle. The detection range of this sensor is defined by a particular configuration parameter.

The rangefinder sensors of the robot monitor the following directions relative to the agent

heading:

Sensor	Direction (degrees)
Right	-90
Front-right	-45
Front	0
Front-left	45
Left	90
Back	180

The movement of the robot is controlled by two actuators that apply forces that turn and/or propel the agent frame, that is, change its linear and/or angular velocity.

#### Architecture of neural network policy:

The neuroevolution algorithm starts with a very basic initial ANN configuration that only has input nodes for sensors and output nodes for actuators, which gradually becomes more complex until a successful maze solver is found.

#### Training algorithm and fitness function:

Two approaches will be analyzed in this project. The first approach is the Goal Oriented objective based approach. This objective function is based on the estimation of the fitness score of the maze solver by measuring the distance between its final position and the maze exit after executing the 400 simulation steps. Thus, the objective function is goal-oriented and solely depends on the ultimate goal of the experiment: reaching the maze exit area.

Fitness is defined by the difference between initial Euclidean distance between start and exit position ( $D_{initial}$ ) and Euclidean distance between current and exit position ( $D$ ). It is normalized to bring fitness in range [0, 1]

Hence, at the start fitness is zero  $F = \frac{D_{initial} - D}{D_{initial}}$  and it will be 1 when  $D$  is zero i.e agent is at its goal. The algorithm will try to maximize this fitness.

The other approach is the Novelty Search optimization (NS) approach which rewards based on the novelty of behaviour shown by the agent rather than the distance to final point. This idea is inspired by natural evolution. When looking for a successful solution, it is not always obvious the exact steps that should be taken. Natural evolution continuously produces novel forms, with different phenotypes trying to exploit the surrounding environment and adapt to the changes.

The behavioral space of the maze solver agent is defined by its trajectory through the maze while running the maze-solving simulation. An effective novelty score implementation needs to compute the sparseness at any point in such a behavioral space. Thus, any area with a denser cluster of visited points of behavior space is less novel, giving fewer rewards to the solver agent. the most straightforward measure of sparseness at a point is the average distance from it to the  $k$ -nearest neighbors. The sparse areas have higher distance values, and the denser areas have lower distance values, correspondingly. The following formula gives the sparseness at point  $x$  of the behavioral space:

$$p(x) = \frac{1}{k} \sum_{i=0}^k \text{dist}(x, u_i)$$

where,  $u_i$  is the  $i^{\text{th}}$  nearest neighbor of  $x$ .

### Parameters of experiment:

Experiments are performed on hard maze environment by initializing the experiment for 9 fixed starting positions. Maze environment is 200 X 200 and starting locations chosen are :

```
start_locations = [(40, 40), (40, 110), (40, 180), (110, 50),
(110, 110), (110, 180), (180, 40), (180, 110), (180, 180)]
```

Fig. 1 illustrates the locations of these starting positions. The red circle indicates exit point or goal position while green circles indicate the starting points. Note that origin is top left point. This is to keep consistency with the experimental setup of [1, 2, 3].

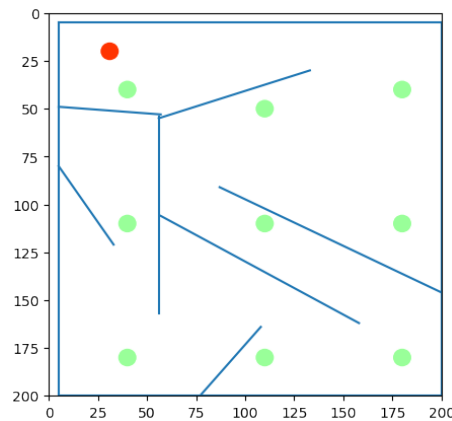


Fig. 1. Red circle indicates goal point and green circles indicate starting positions

Following the experimental setting of the author, I have set 250 individuals in 1 population for all experiments. However since author's best performance in Novelty search is Fig. 2 with 400 time steps for each individual, I expect that increasing number of steps will allow the individual

to reach goal in case of Novelty search algorithm. Hence, I have set 700 time steps for each individual.

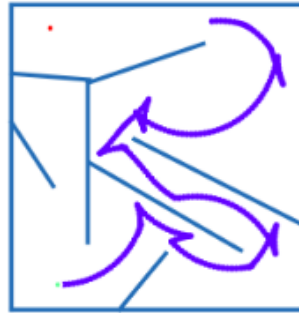


Fig. 2. Best Novelty search result from [3] for 400 time steps for each individual

## Results:

### Experiments with fixed starting positions

#### Start Location (40, 40)

Since these points are very close and there is no local optima between these two points, the robot should reach goal very easily. Experimental results verify the expectation. The paths for best solver agent for both techniques are shown in Table 1:

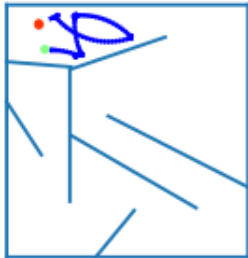

Goal Oriented Objective based approach	Novelty Search Optimization
 <p>Path taken by best solving agent</p>	 <p>Path taken by Best solving Agent</p>

Table 1. Comparison of path taken by best solving agent for both techniques

Now let us analyze the performance across generations. Fig. 4. gives us fitness scores of best individual in each generation.

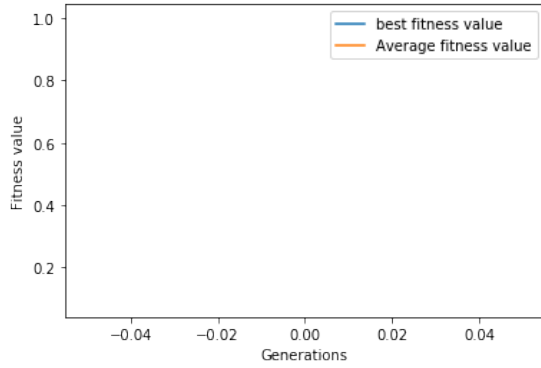


Fig. 4. Fitness values across generations for objective based approach

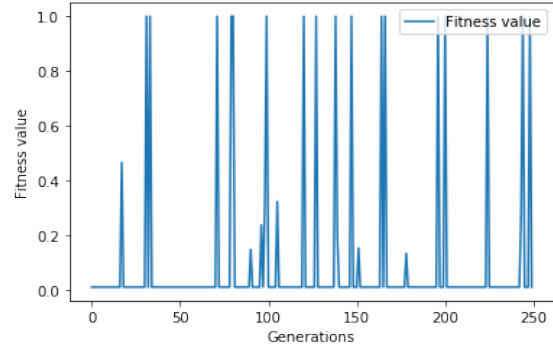


Fig. 5. Fitness values of all 250 individuals in 1<sup>st</sup> generation for objective based approach

Since, Fig. 4 is empty but task has been solved so I can expect that the task has been solved in first generation. When the fitness values of all individuals were analyzed, 17 individuals got maximum fitness i.e 1.0 within the first generation as shown in Fig. 5.

Similar trend is observed for the case of Novelty search optimization. However, 51 individuals reached the goal with Novelty Search technique.

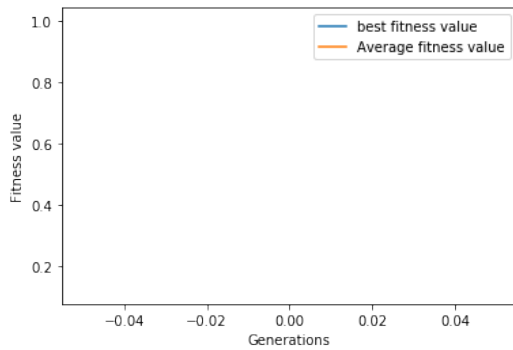


Fig. 6. Fitness values across generations for Novelty Search optimization

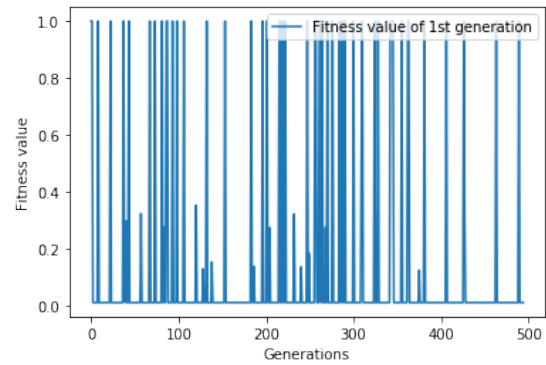
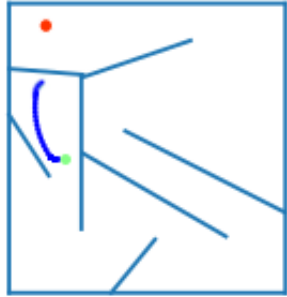


Fig. 7. Fitness values of all 250 individuals for Novelty Search Optimization

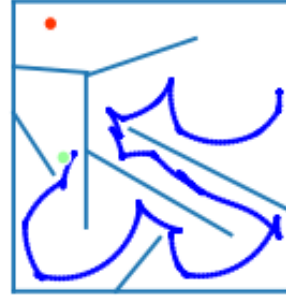
## Start Location (40, 110)

Goal Oriented Objective based approach

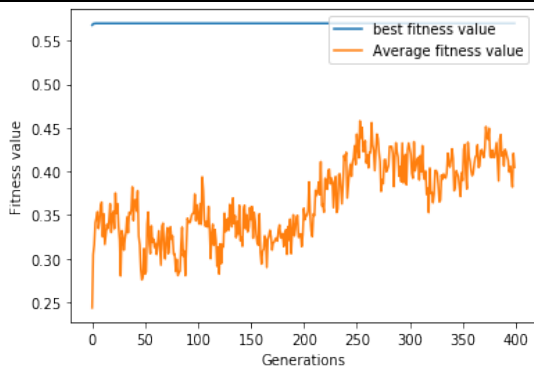


Path taken by best solving agent

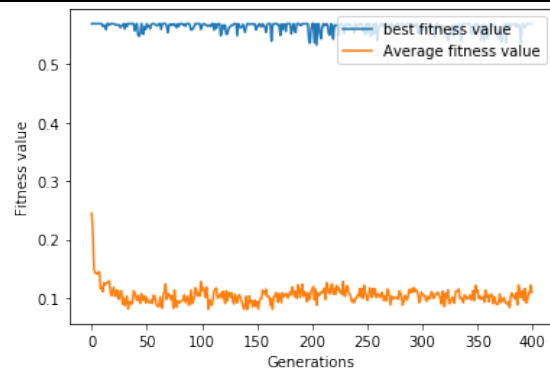
Novelty Search Optimization



Path taken by best solving agent



Fitness scores of best individuals from each generation



Fitness scores of best individuals from each generation

**Start Location (40, 180)**

Goal Oriented Objective based approach

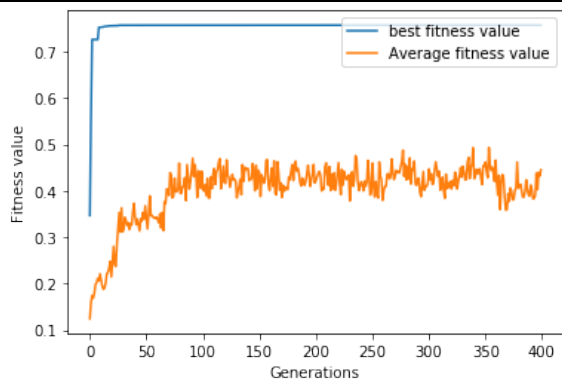


Path taken by best solving agent

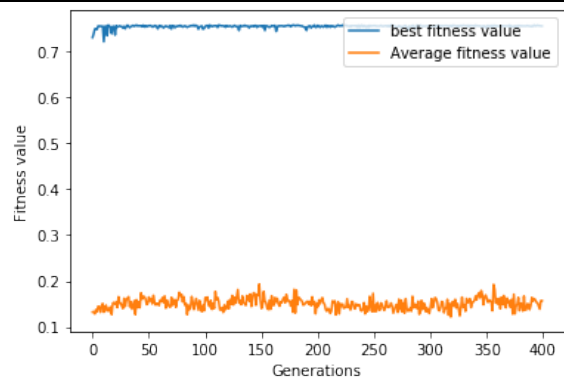
Novelty Search Optimization



Path taken by best solving agent



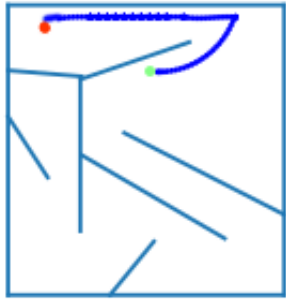

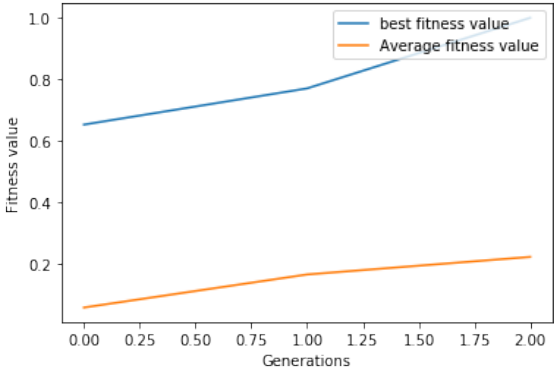
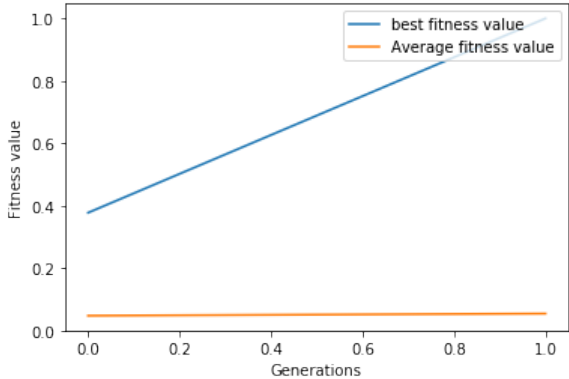
Fitness scores of best individuals from each generation





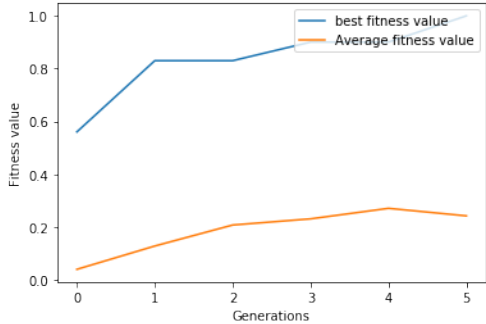
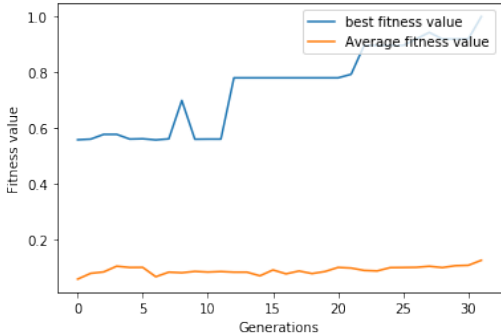
Fitness scores of best individuals from each generation



## Start Location (110, 50)

Goal Oriented Objective based approach	Novelty Search Optimization
 <p data-bbox="290 825 711 856">Path taken by best solving agent</p>	 <p data-bbox="911 821 1330 852">Path taken by best solving agent</p>
 <p data-bbox="219 1367 781 1440">Fitness scores of best individuals from each generation</p>	 <p data-bbox="839 1375 1401 1449">Fitness scores of best individuals from each generation</p>

Start Location (110, 110)

Goal Oriented Objective based approach	Novelty Search Optimization
 <p>Path taken by best solving agent</p>	 <p>Path taken by best solving agent</p>
 <p>Fitness scores of best individuals from each generation</p>	 <p>Fitness scores of best individuals from each generation</p>

## Start Location (110, 180)

Goal Oriented Objective based approach

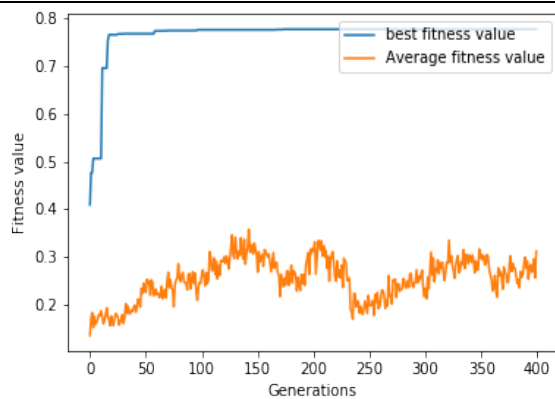


Path taken by best solving agent

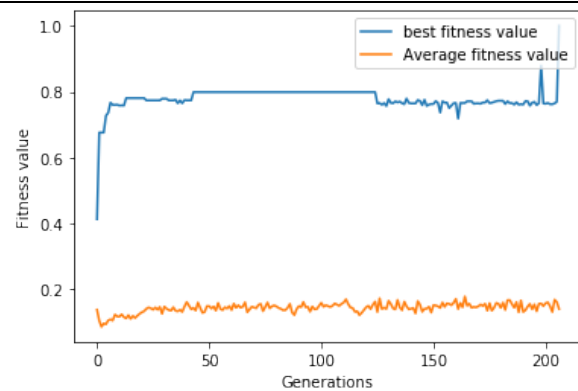
Novelty Search Optimization



Path taken by best solving agent



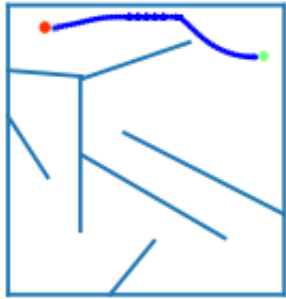
Fitness scores of best individuals from each generation



Fitness scores of best individuals from each generation

Start Location (180, 40)

Goal Oriented Objective based approach

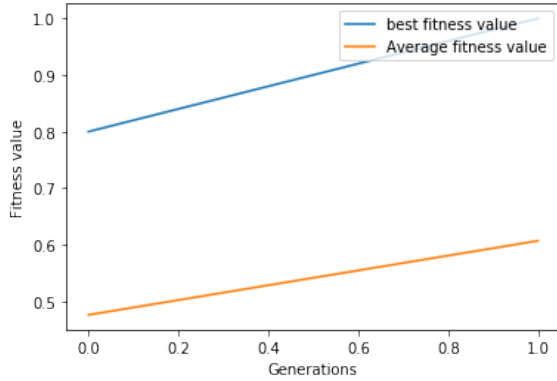


Path taken by best solving agent

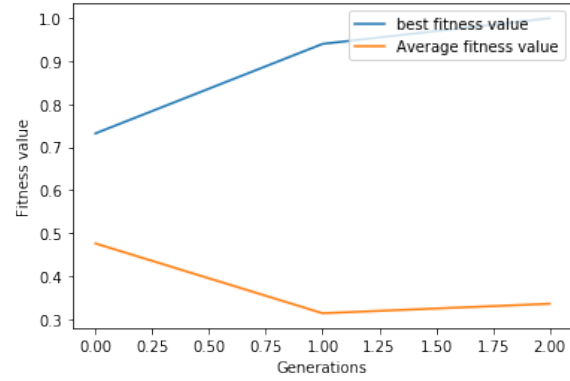
Novelty Search Optimization



Path taken by best solving agent



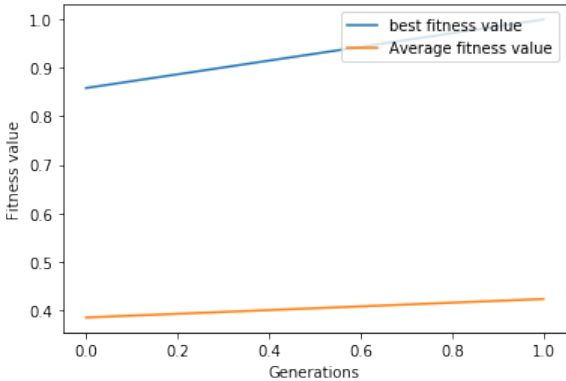
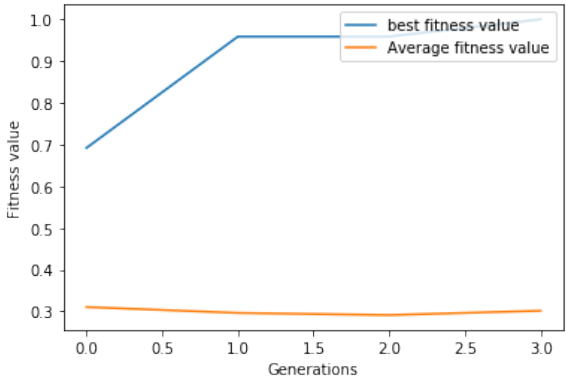


Fitness scores of best individuals from each generation

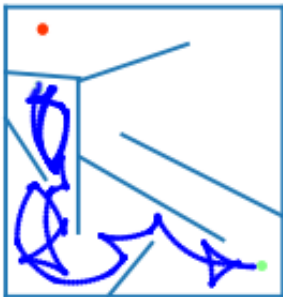

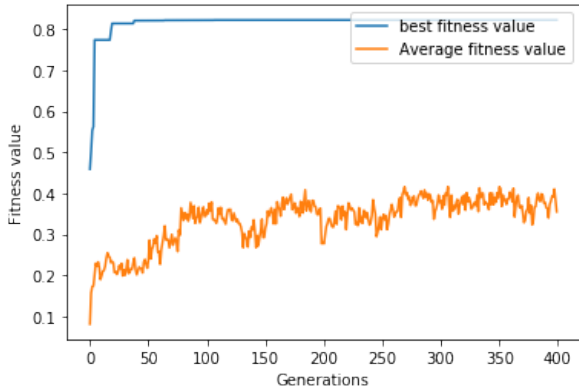
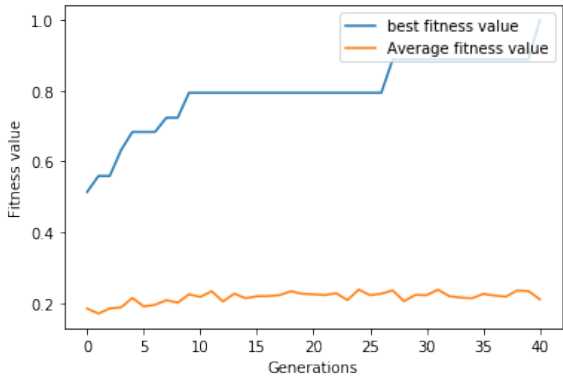


Fitness scores of best individuals from each generation

Start Location (180, 110)

Goal Oriented Objective based approach	Novelty Search Optimization
 <p>Path taken by best solving agent</p>	 <p>Path taken by best solving agent</p>
 <p>Fitness scores of best individuals from each generation</p>	 <p>Fitness scores of best individuals from each generation</p>

Start Location (180, 180)

Goal Oriented Objective based approach	Novelty Search Optimization
 <p>Path taken by best solving agent</p>	 <p>Path taken by best solving agent</p>
 <p>Fitness scores of best individuals from each generation</p>	 <p>Fitness scores of best individuals from each generation</p>

## Results and Discussions:

- Observing the performance of best solver agent for start locations (40, 110), (40, 180), (110, 180), (180, 180), we see that Novelty search optimization brings the robot to the correct path and does not stuck in local optima.
- For other starting locations, robots evolved with both techniques reach the goal but the path length and number of generations to evolve successful agent are lower for objective based approach as compared to Novelty search. Hence , it will take lesser time to evolve the successful agent and lesser time for it to reach goal.
- The results of the experiments are present in *results\_fixed\_positions* directory.

## Experiments with Random Positions:

I have modified the code such that now it generates a random x and y location in the suitable range and then checks whether the random point collides with wall or not. If it collides then it generates random point again till we get a valid starting position. The code has been modified in `read_environment()` function of `maze_environment.py`

## References:

- [1] M. Karl, J. Duggan, E. Howley, "Maze Navigation using Neural Networks evolved with Novelty Search and Differential Evolution", Adaptive and learning agents workshop at ICML-AAMAS 2018, 2018.
- [2] I.Omelianenko, "Creation of Autonomous Artificial Intelligent Agents using Novelty Search Method of fitness function optimization", August, 2018.
- [3] PacktPublishing, "Hands on Neuroevolution with Python", <https://github.com/PacktPublishing/Hands-on-Neuroevolution-with-Python.git>, 2019.