ARTIFICIAL EVOLUTION

A report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE DEPARTMENT

By
PRIYANK KUMAR GUPTA (IIITU17146)
KRITAGYA KHANDELWAL (IIITU17145)

SCHOOL OF COMPUTING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA
HIMACHAL PRADESH

DEC 2020

BONAFIDE CERTIFICATE

This is to certify that the project titled ARTIFICIAL EVOLUTION is a bonafide record of the work done by

PRIYANK KUMAR GUPTA (IIITU17146)

KRITAGYA KHANDELWAL (IIITU17145)

in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science of the INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH, during the year 2020- 2021.

under the guidance of

Ms. Himani Joshi

Project viva-voce held on: 19/12/20

Internal Examiner External Examiner

ORIGINALITY / NO PLAGIARISM DECLARATION

We certify that this project report is our original report and no part of it is copied from any published reports, papers, books, articles, etc. We certify that all the contents in this report are based on our personal findings and research and we have cited all the relevant sources which have been required in the preparation of this project report, whether they be books, articles, reports, lecture notes, and any other kind of document. We also certify that this report has not previously been submitted partially or as whole for the award of degree in any other university in India and/or abroad.

We hereby declare that, we are fully aware of what constitutes plagiarism and understand that if it is found at a later stage to contain any instance of plagiarism, our degrees may be cancelled.

PRIYANK KUMAR GUPTA (IIITU17146)
KRITAGYA KHANDELWAL (IIITU17145)

ABSTRACT

Nature is full of miracles, just a little genetic shift can promote the inception of a new species. In this new era of artificial intelligence, we are keen to replicate it on artificial species to thrive in their artificial environment. Ant is a small yet complex creature on earth, different species have different traits and they possess some of the most complex behaviour due to natural selection and survival of the fittest. In our artificial environment, we have produced complex behaviour in ants with the help of artificial intelligence. In our experimental investigations, we have successfully trained the brain (ANN) of ant to feed on food present in our artificial Environment and is now capable of surviving in our artificial environment.

***Keywords*:** Genetic Algorithm, Artificial Neural Network, Evolution, Natural Selection.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

We would like to express our sincere gratitude and heartfelt thanks to Ms. Himani Joshi for their unflinching support and guidance, valuable suggestions and expert advice. Their words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

We also take the opportunity to thank our Director and all the faculty of the School of Computing, IIIT Una for helping us by providing necessary knowledge base and resources.

We would also like to thank our parents and friends for their constant support.

PRIYANK KUMAR GUPTA (IIITU17146)
KRITAGYA KHANDELWAL (IIITU17145)

TABLE OF CONTENTS

LIST OF ACRONYMS

NN	Neural Network
GA	Genetic algorithm
ANN	Artificial neural network
EA	Evolutionary Algorithms
DNN	Deep Neural Network

LIST OF FIGURES

Chapter 1
Introduction

1.1 Project Idea

Nature is stuffed with miracles, just a touch genetic shift can promote the inception of a replacement species. During this new era of AI, we are keen to duplicate it on artificial species to thrive in their artificial environment. Ant may be a small yet complex creature on earth, different species have different traits and that they possess a number of the foremost complex behaviours because of action and survival of the fittest. In our artificial earth, we wish to supply complex behaviour in ants with the assistance of computing.
Algorithms used for artificial earth:

1) Genetic Algorithm
2) Artificial Neural Network

1.1.1 Genetic algorithm

In engineering and research, a genetic algorithm (GA) is a metaheuristic algorithm, inspired by the method of survival that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are preferred algorithms to generate high-quality solutions to optimization problems and search problems by using biologically inspired operators like mutation, crossover and selection.

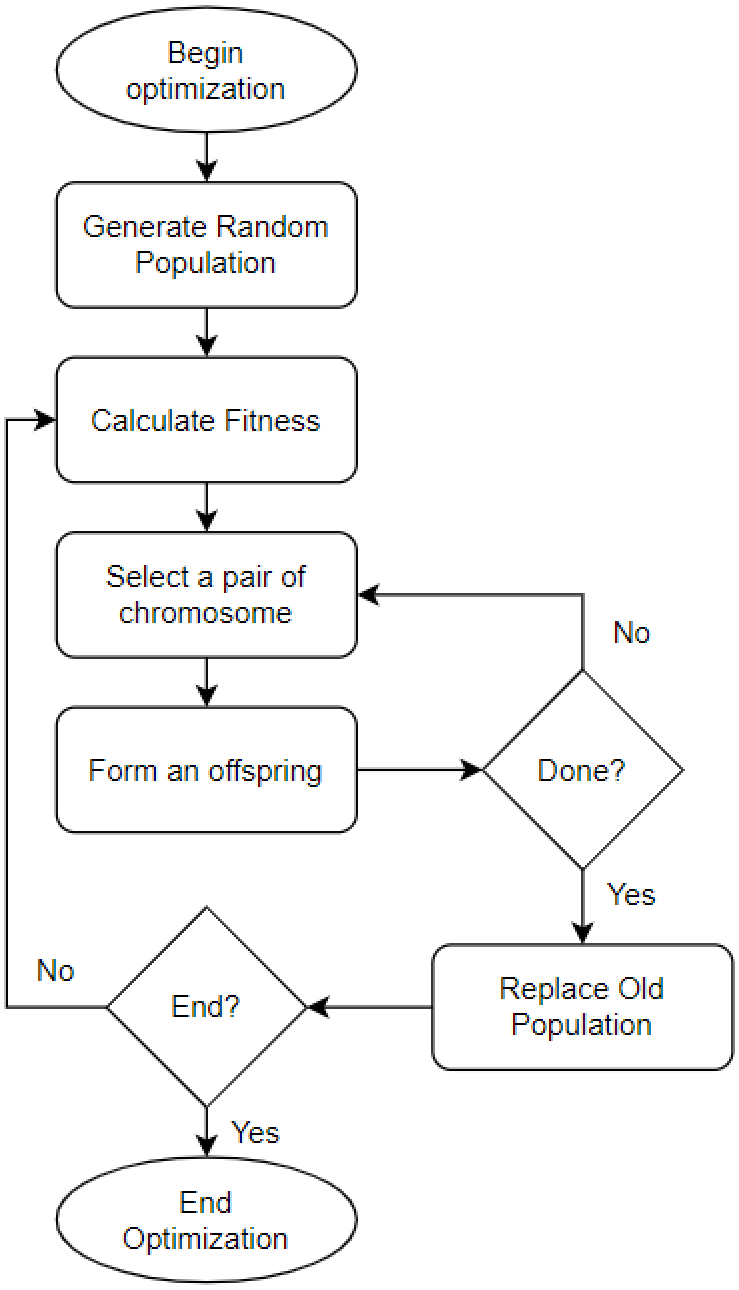


Figure 1.1: Flowchart of genetic algorithm

1.1.2 Artificial Neural Network

An Artificial Neural Network (ANN) is a type of algorithm that is designed to simulate the neurons in the human brain. This computing system is found to solve some of the most complex problems which have not been solved by simple algorithms. The artificial neural network has self-learning capabilities. The ANN algorithms consist of forward propagation and backward propagation. The forward propagation helps the algorithm to find the output through the neurons and backward propagation helps the algorithm to learn from the errors that are the mistake done by it.

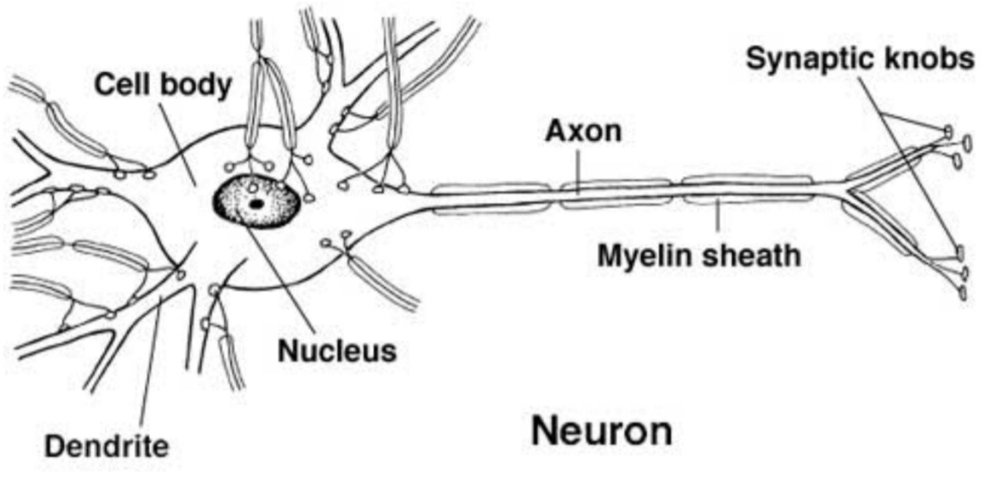


Figure 1.2: Detailed diagram of biological Neuron.[2].

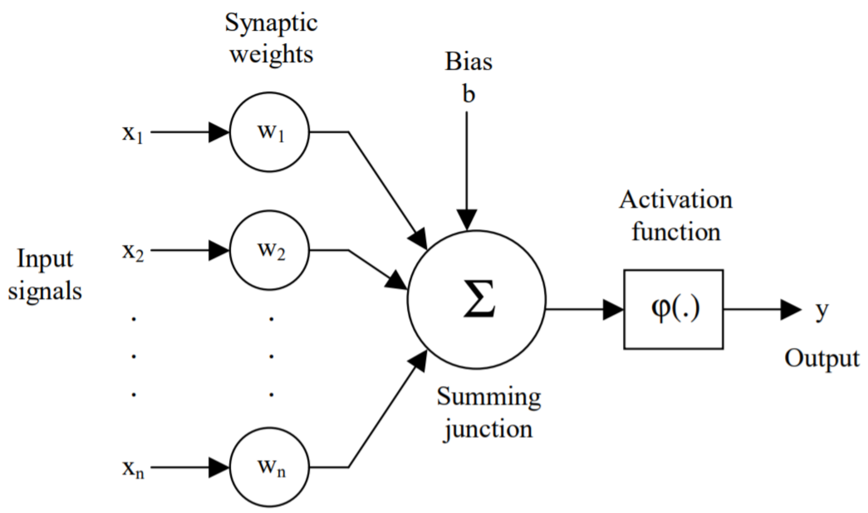


Figure 1.3: Artificial Neuron Architecture.[2].

Chapter 2
Literature Survey

2.1 Emergent Tool Use From Multi-Agent Interaction [3]

In this paper, they provided various situations to the hide-and-seek agents, Under which they reacted differently and very much humanly and also reflected some out of the box thinking ability like humans.

In this environment, agents play a team-based hide-and-seek game. Hiders are tasked with avoiding line-of-sight from the seekers, and seekers are tasked with keeping the vision of the hiders. There are objects scattered throughout the environment that hiders and seekers can grab and lock in situ, furthermore as randomly generated immovable rooms and walls that agents must learn to navigate. Before the sport begins, hiders are given a preparation phase where seekers are immobilized to provide hiders with an opportunity to run away or change their environment.

As agents train against one another in hide-and-seek, as many as six distinct strategies emerge. Each new strategy creates a previously nonexistent pressure for agents to achieve the following stage. Note that there aren't any direct incentives for agents to interact with objects or to explore; rather, the emergent strategies shown below are a result of the auto curriculum induced by multi-agent competition and therefore the simple dynamics of hide-and-seek.

Chapter 3
Motivation

Our model is inspired by nature which is quite random. This randomness helps the species to grow in different ways due to which some organisms have dissimilar looking species. The natural selection of survival of the fittest helps the organism to survive in the harsh environment and learn complex behaviour which helps them to survive and thrive. In this new age of artificial intelligence, we want to produce similar conditions and results in our artificial environment.

In the research mentioned above agents reacted very intelligently in accordance with the situations. They were able to strategize and execute to win the sport. We want to replicate it on artificial species to thrive in their artificial environment.

Chapter 4

Proposed Work

4.1 Single Ant Interaction and Evolution

In our artificial environment, there is a single ant. The ant's motive is to eat food which is present in the environment.

4.1.1 Environment

The artificial environment consists of a grid of 20 x 20. The ant can't go out of the grid. The environment provides food on the grid, the ant has to eat the food to survive.

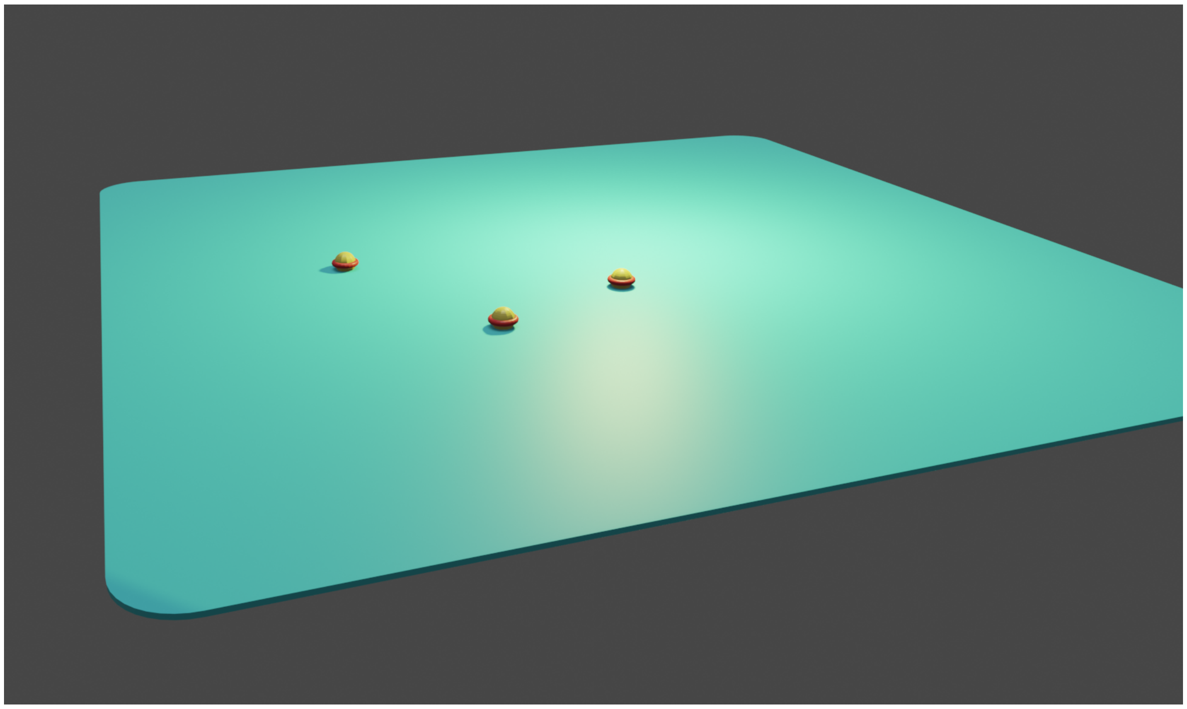


Figure 4.1: Artificial Environment

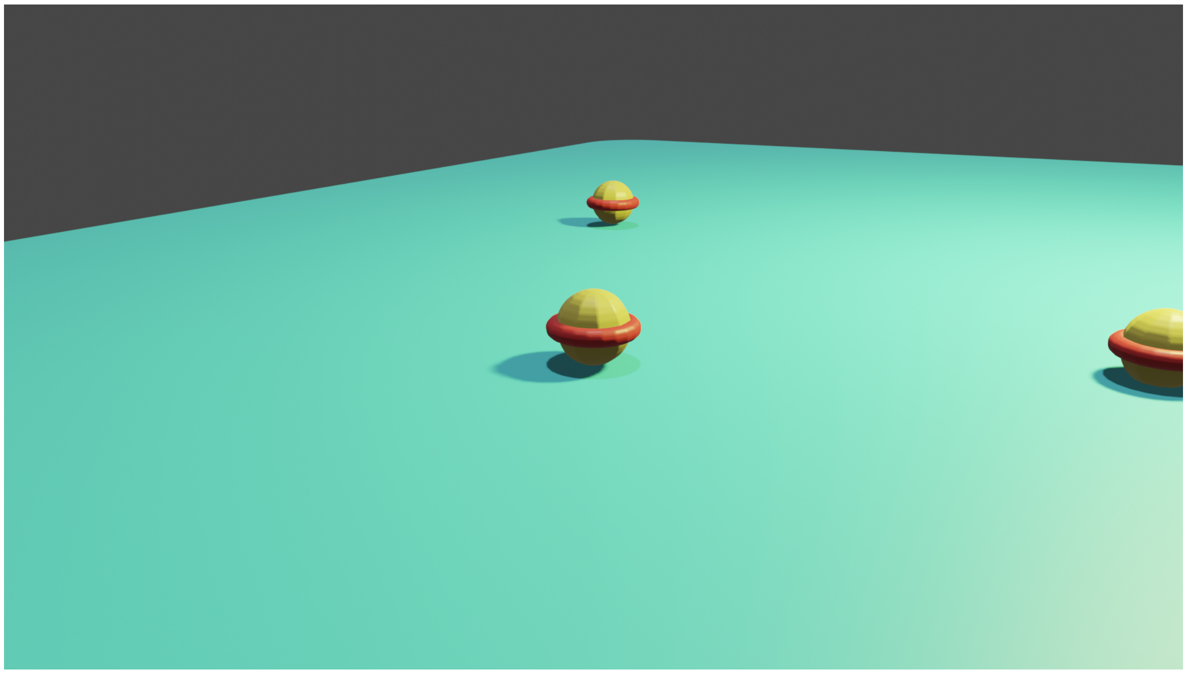


Figure 4.2: Food for ant

These images show the Environment that we provide our artificial ant. The Environment comprises the land area(Grid) on which our ant can roam around and foods which our ant can eat and gain some fitness points.

4.1.2 Ant

The ant resides in the grid in the environment. The ant can move in three directions - left, right and front. The ant can roam freely in the environment, the constraint for the ant is that if it goes out of board it will die. The ant has to eat the food which the environment provides. The ant has some energy in the start, after moving some distance if the ant does not eat the food then it will have no energy left and it will die.

The ant has a vision, it can look in front of it and the ant has an artificial brain which helps it to decide the direction to go taking vision as input and direction as output.

The brain of ant consists of a deep neural network with the following specifications -

1. Input has 20 neurons.
2. It has two hidden layers having 20 neurons in the first layer and 12 neurons in the second layer.
3. Output has 4 neurons.

Figure 4.3: Ant

4.1.3 Training Deep Neural Network

We have used a genetic algorithm to train the neural network. The description of every phase of genetic algorithm is as follows -

1. The population of genetic algorithm is taken as a deep neural network with random weights.
2. The fitness of the deep neural network is calculated by simulating the decisions made by the deep neural network in the environment, the fitness function is proportional to the number of food eaten in the life of the ant.
3. The fitness of the brain (DNN) is sorted in decreasing order, the top fifty per cent of the best performing brain of an ant is selected as the parent for making offspring.
4. In our approach, we have not used a crossover operator due to bad performance.
5. The parent is selected from the top fifty per cent and mutation is performed over the weights of the neural network. The mutation rate is taken as five per cent.
6. In our approach, we have taken top one per cent of parent to be passed to next generation so that the fitness in the next generation is not decreased and the complex behaviour learnt in previous generations is retained.
7. The old generation is replaced and the whole procedure is repeated to get maximum fitness.

Chapter 5
Conclusion, Future Work and Learning Outcomes

5.1 Conclusion

In our experimental investigations, we have successfully trained the brain (ANN) of ant to get the highest fitness value as 45592083. The maximum food eaten by the ant is 45590. Our trial is based on running the genetic algorithm to 1190 generations after which we have got the best fitness value.
The following graph shows the fitness of the best performing ant in each generation.

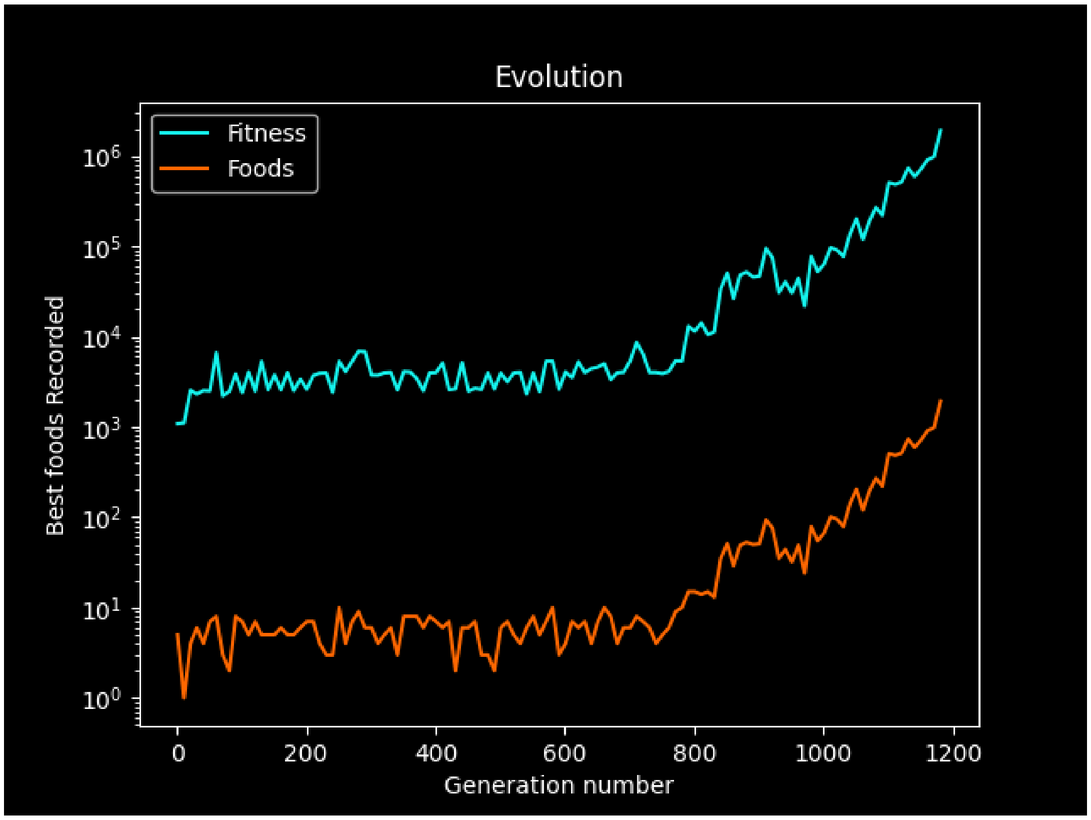


Figure 5.1: Plot of fitness and food eaten by the best performing ant in each generation.

5.2 Future Work

In our model we have taken a single ant to learn the direction it needs to go to get the food, the next goal is to make an artificial environment of multiple ants to interact with each other and the environment to learn some complex behaviours like teamwork, self-organizing, ant colony building, dead reckoning. For training, we have used a genetic algorithm and it has slow learning due to its brute force nature, so for learning reinforcement learning can be used to optimize.

5.3 Learning Outcomes

To accomplish the project's objective we learned about different fields of Artificial Intelligence, Main Algorithms in our project are Genetic Algorithm and Artificial Neural Networks.

Both the Algorithms are combined to replicate the natural selection in our artificial ant. These Algorithms are implemented using python and a 3-D simulation of the outcome is built with the help of blender software.

8.

References

[1] Mantas Paulinas, Andrius Ušinskas, "A Survey Of Genetic Algorithms Applications For Image Enhancement and Segmentation", Electronic Systems Department, Faculty of Electronics, Vilnius Gediminas Technical University.

[2] Csáji, B.C., 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*(48), p.7.

[3] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B. and Mordatch, I., 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.

[4] Omar M. Sallabi and Younis EI-Haddad, "An Improved Genetic Algorithm to Solve the Traveling Salesman Problem", World Academy of Science, Engineering and Technology Volume 3, 2009.

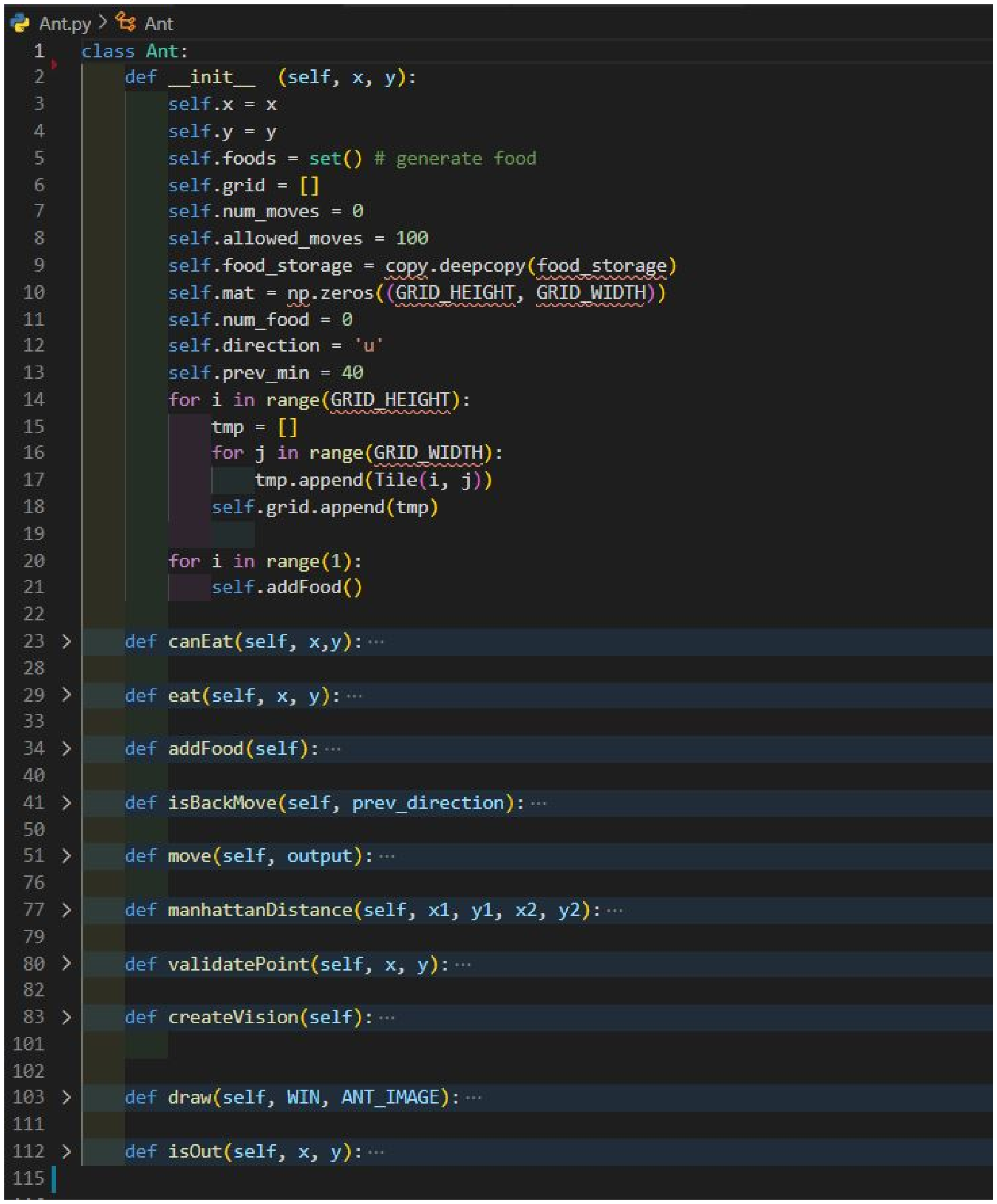
Appendices

Appendix A

Code Attachments

The following is the partial / subset of the code. Code of some module(s) have been wilfully attached.

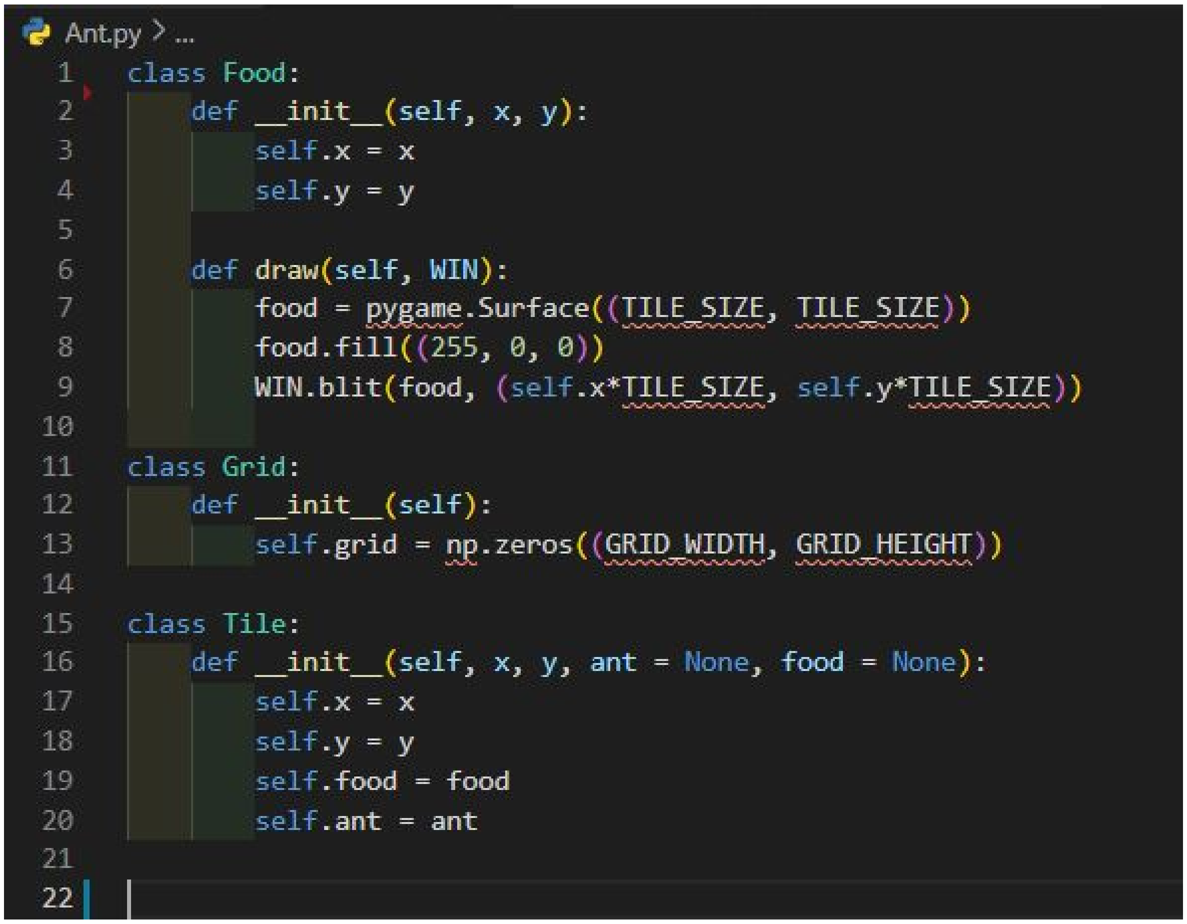
A.1 Ant Class



Ant.py > Ant

```python
class Ant:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.foods = set() # generate food
        self.grid = []
        self.num_moves = 0
        self.allowed_moves = 100
        self.food_storage = copy.deepcopy(food_storage)
        self.mat = np.zeros((GRID_HEIGHT, GRID_WIDTH))
        self.num_food = 0
        self.direction = 'u'
        self.prev_min = 40
        for i in range(GRID_HEIGHT):
            tmp = []
            for j in range(GRID_WIDTH):
                tmp.append(Tile(i, j))
            self.grid.append(tmp)

        for i in range(1):
            self.addFood()

    def canEat(self, x,y): ...

    def eat(self, x, y): ...

    def addFood(self): ...

    def isBackMove(self, prev_direction): ...

    def move(self, output): ...

    def manhattanDistance(self, x1, y1, x2, y2): ...

    def validatePoint(self, x, y): ...

    def createVision(self): ...


    def draw(self, WIN, ANT_IMAGE): ...

    def isOut(self, x, y): ...

```

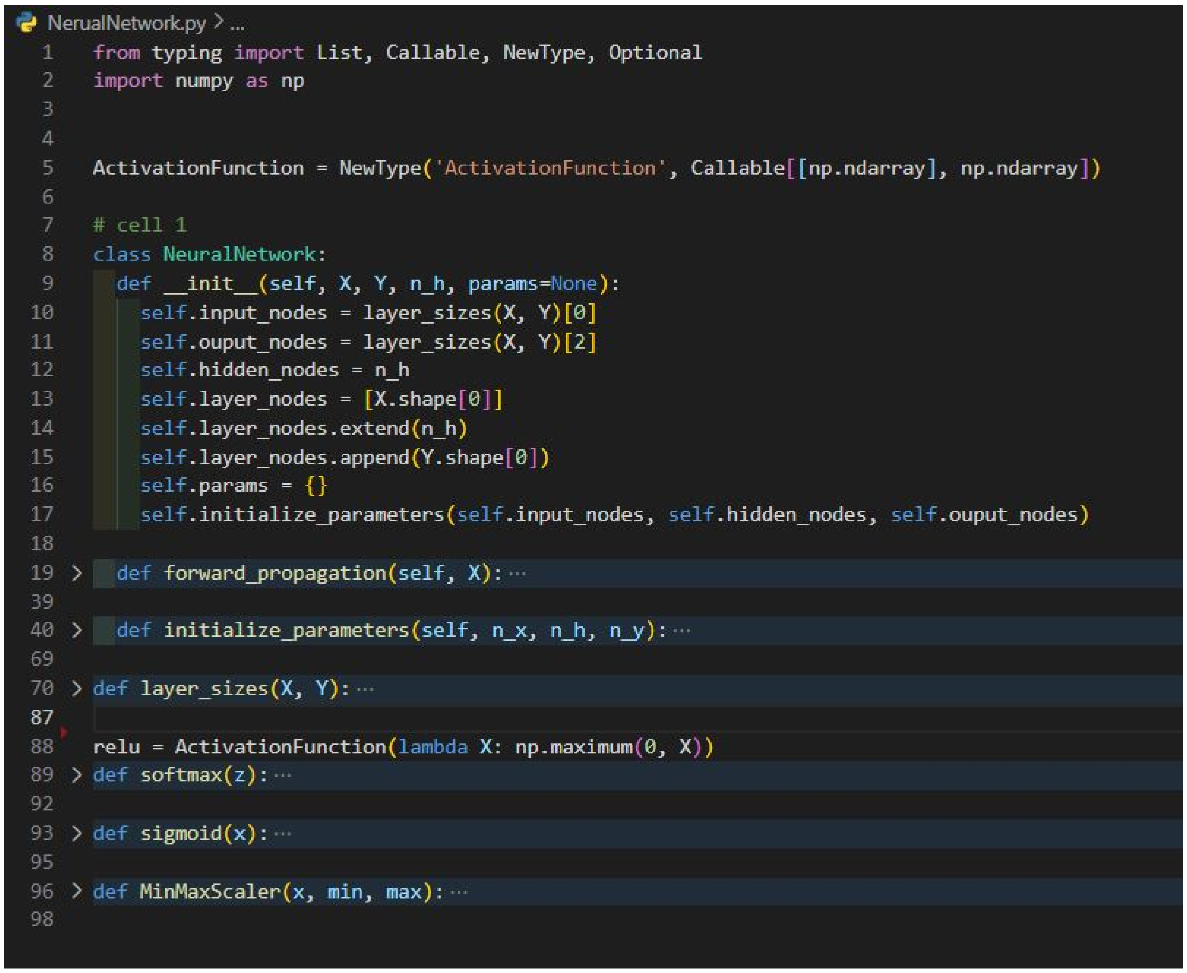
A.2 Food and Grid Class



Ant.py > ...

```python
class Food:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def draw(self, WIN):
        food = pygame.Surface((TILE_SIZE, TILE_SIZE))
        food.fill((255, 0, 0))
        WIN.blit(food, (self.x*TILE_SIZE, self.y*TILE_SIZE))

class Grid:
    def __init__(self):
        self.grid = np.zeros((GRID_WIDTH, GRID_HEIGHT))

class Tile:
    def __init__(self, x, y, ant = None, food = None):
        self.x = x
        self.y = y
        self.food = food
        self.ant = ant

```

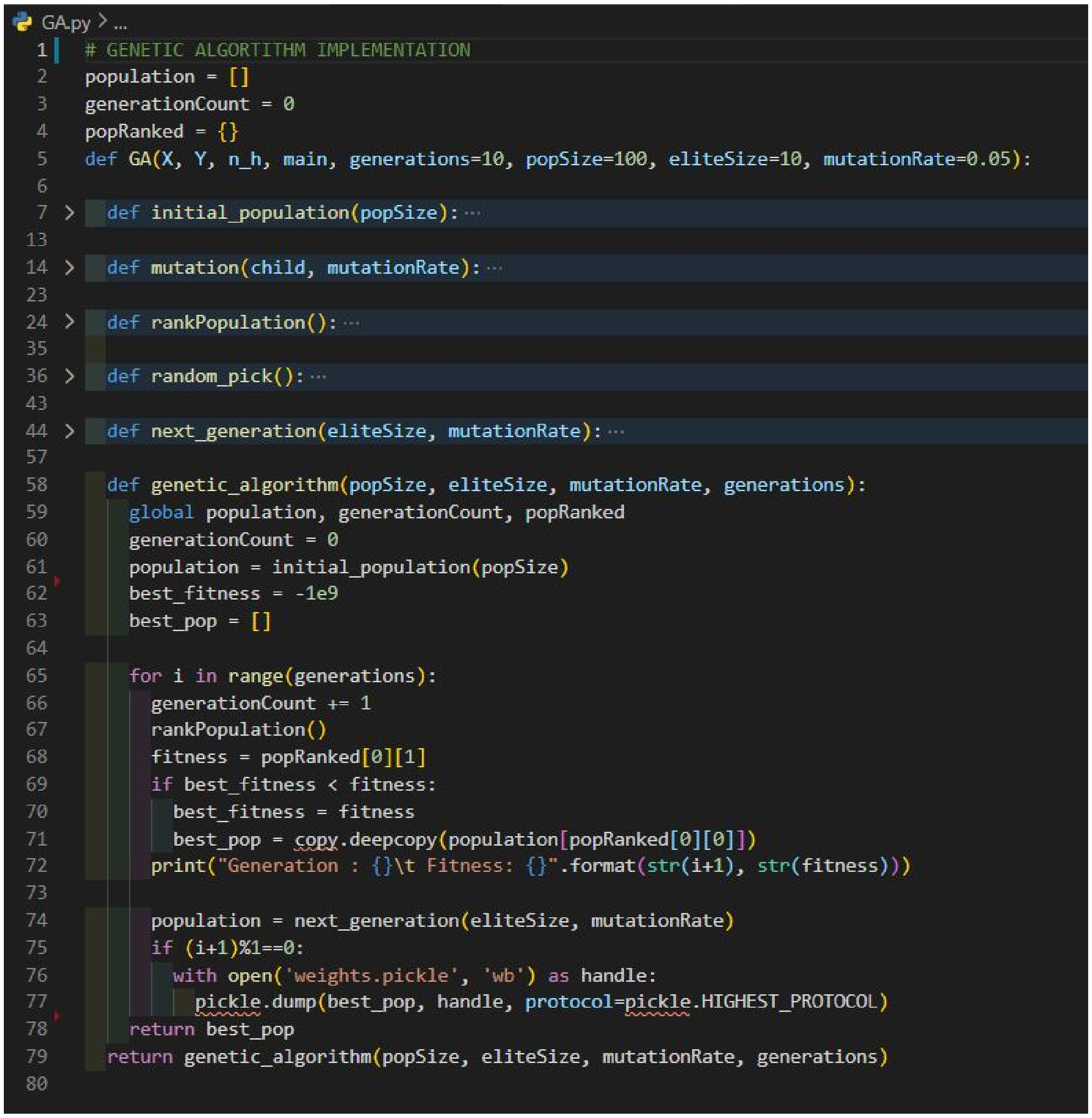
A.3 Neural Network Class



NerualNetwork.py > ...

```python
from typing import List, Callable, NewType, Optional
import numpy as np


ActivationFunction = NewType('ActivationFunction', Callable[[np.ndarray], np.ndarray])

# cell 1
class NeuralNetwork:
    def __init__(self, X, Y, n_h, params=None):
        self.input_nodes = layer_sizes(X, Y)[0]
        self.ouput_nodes = layer_sizes(X, Y)[2]
        self.hidden_nodes = n_h
        self.layer_nodes = [X.shape[0]]
        self.layer_nodes.extend(n_h)
        self.layer_nodes.append(Y.shape[0])
        self.params = {}
        self.initialize_parameters(self.input_nodes, self.hidden_nodes, self.ouput_nodes)

    def forward_propagation(self, X): ···

    def initialize_parameters(self, n_x, n_h, n_y): ···

def layer_sizes(X, Y): ···

relu = ActivationFunction(lambda X: np.maximum(0, X))
def softmax(z): ···

def sigmoid(x): ···

def MinMaxScaler(x, min, max): ···

```

A.4 Genetic Algorithm


GA.py > ...

```python
# GENETIC ALGORTITHM IMPLEMENTATION
population = []
generationCount = 0
popRanked = {}
def GA(X, Y, n_h, main, generations=10, popSize=100, eliteSize=10, mutationRate=0.05):

    def initial_population(popSize):···

    def mutation(child, mutationRate):···

    def rankPopulation():···

    def random_pick():···

    def next_generation(eliteSize, mutationRate):···

    def genetic_algorithm(popSize, eliteSize, mutationRate, generations):
        global population, generationCount, popRanked
        generationCount = 0
        population = initial_population(popSize)
        best_fitness = -1e9
        best_pop = []

        for i in range(generations):
            generationCount += 1
            rankPopulation()
            fitness = popRanked[0][1]
            if best_fitness < fitness:
                best_fitness = fitness
                best_pop = copy.deepcopy(population[popRanked[0][0]])
            print("Generation : {}\t Fitness: {}".format(str(i+1), str(fitness)))

            population = next_generation(eliteSize, mutationRate)
            if (i+1)%1==0:
                with open('weights.pickle', 'wb') as handle:
                    pickle.dump(best_pop, handle, protocol=pickle.HIGHEST_PROTOCOL)
        return best_pop
    return genetic_algorithm(popSize, eliteSize, mutationRate, generations)

```