CS 451: Computational Intelligence

Assignment 02

Syed Hasan Faaz Abidi (sa06195), Syed Hammad Ali (sa04324)

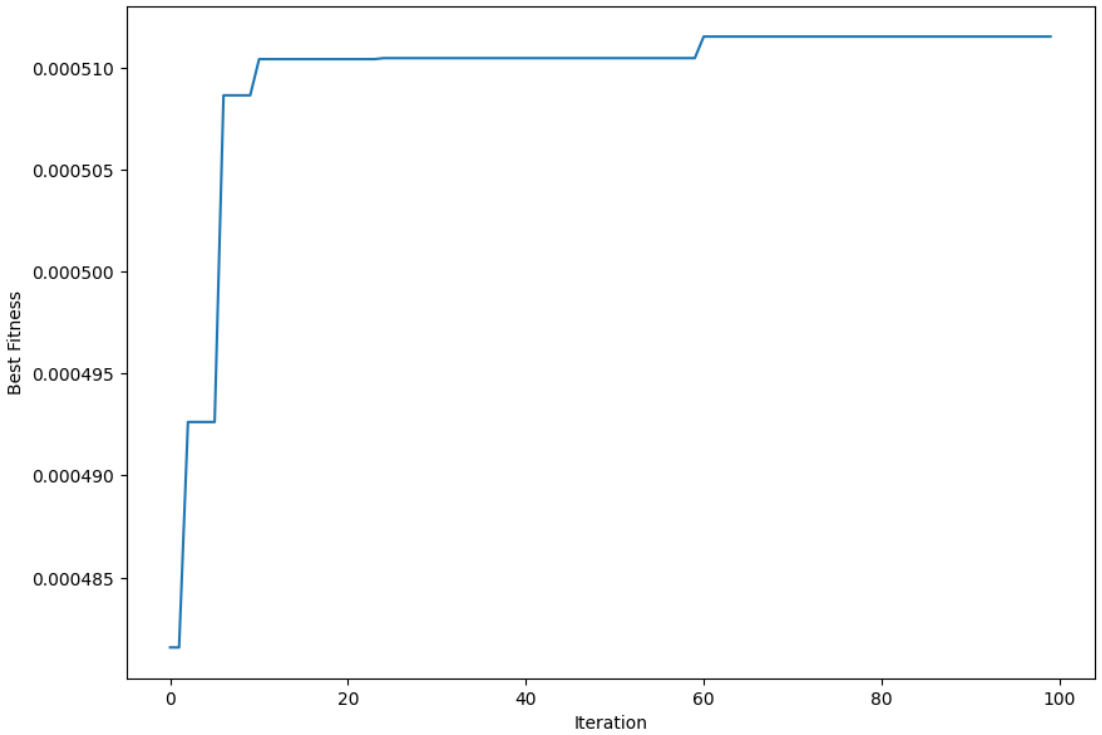
17 March 2022

1. Capacitated Vehicle Routing using ACO

A population of Ants are generated. They have a capacity to deliver around 100 valued items. Each ant will be visiting all the nodes once and coming back to the warehouse when their load is finished to reload. Each ant starts with randomly selecting a node to deliver to as there is no pheromone in the start. Only the distance from the current position has affect on the choice. When all the nodes are visited by an ant, they put pheromones on the path they took depending on the cost of the path. And best solution of a generation is selected and more pheromone is put on its path.

Best So Far

Best Cost: **2004.5034815581962**
Best Fitness: **0.0004988766590830026**
Best Cost Overall: **1954.9990120430932**



2. Evolving a Tic-tac-toe Agent

We were required to create and evolve a Tic-tac-toe agent using any evolutionary algorithm. I decided to go with Particle Swarm Optimation. To train an agent I used a neural network with only one hidden layer, for the training, instead of conventional backpropagation, I implemented Particle Swarm Optimization for optimizing the weights of the network.

Fitness Function

For the fitness of each particle, I played my model with 3 different strategies that were already implemented: Intelligent, Naive, and Unintelligent agent. My model played 500 matches with each of the mentioned strategies and then the sum of all the wins and ties was my final fitness score. The range of my fitness score was [0, 1500], where a fitness score of 1500 means it has a perfect score.

Accuracy

Here, by accuracy, we mean number of times our agent won/tied out of the total played games.

Best So Far Fitness: **1419** (out of 1500)

Accuracy against Intelligent Agent: **100%**
Accuracy Against Naive Agent: **84%**
Accuracy Against unIntelligent Agent: **100%**

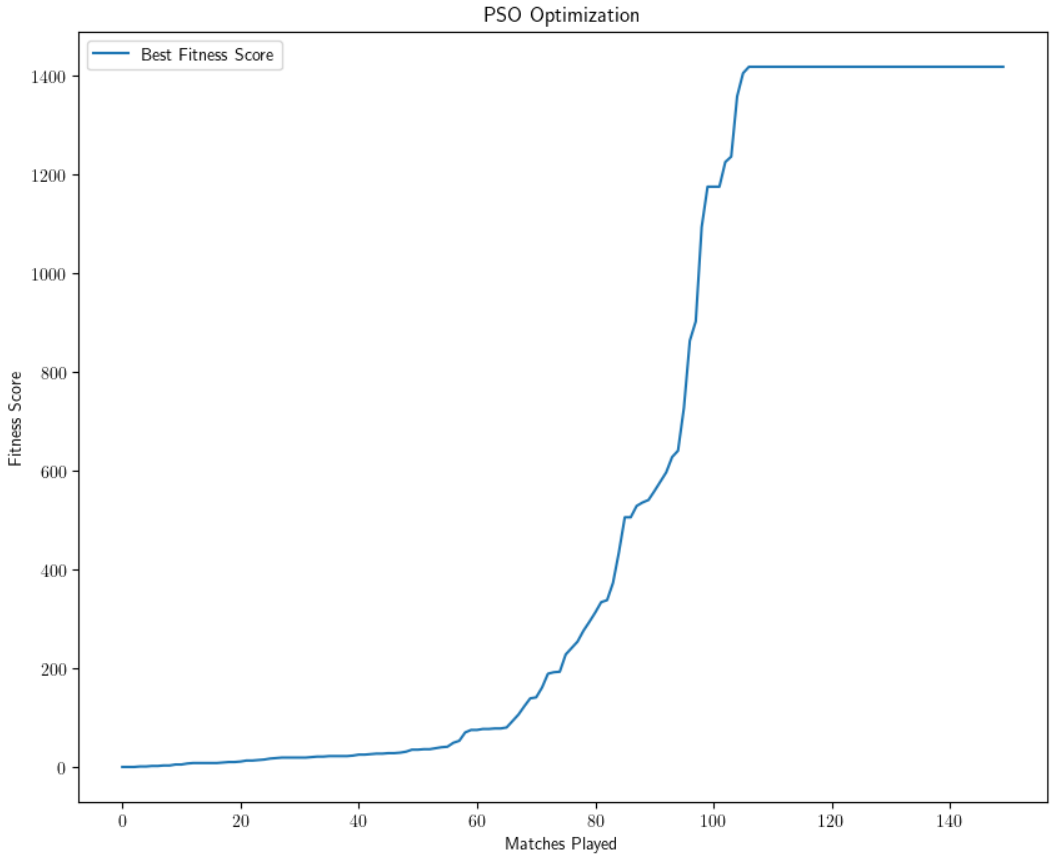
Parameters for Best So Far

Below are the parameters that generated best results:

w = 1.3
c1 = 0.6
c2 = 0.6
Input size = 9
Hidden layer size = 36
Output size = 36
Particles = 25
Iterations = 50
Maximum Position = 1
Maximum Velocity = 0.6

Learning Curves

Below are the learning curves while our model was optimizing using PSO.



Challenges

One of the main challenges I faced was to optimize agent for multiple adversaries. One thing I noticed and that wasted a lot of my time was activation function. I was trying to optimize using sigmoid activation but it was performing poorly. ReLU, on the other hand, worked great at fitting. So, I decided to go for ReLU activation. Moreover, I tried changing the hidden parameters sizes, inertia, c1, and c2. And the best combination that I got is mentioned above. Initially I trained my model by training it against a single adversary but it was overfitting and was not performing well with other adversaries. Therefore, I created a different fitness function that would make our agent more general and would cater other adversaries as well.

Saved Model

In jupyter notebook, there's an option to load already trained model. So, you can either the model from random weights or load already optimized weights.