Swarm Optimization

Olof Harrysson

December 2016

0.1 Resources

The algorithms were implemented in Python 3.5.1. The code was run on a 2,3 GHz Intel Core i7, MAC OSX Yosemite. The library Numpy was used to create the graphs presented.

1 Algorithms

1.1 Swarm

A swarm is initialized with random positions and velocities in the search space. The creatures position is evaluated and given a cost. Every creature keeps track of their own best cost while also knowing about the best position in the whole swarm. In every iteration the individual changes their velocity to go towards a mix of their personal best and the swarms best.

1.2 Random Search

Random positions are generated and evaluated for a cost. The best cost is stored.

2 Results

When the simulation ran ten times it created the results shown in the graphs below. For the swarm algorithm, the best individual in the population is used to create the best, worst and average result when the simulation runs multiple times.

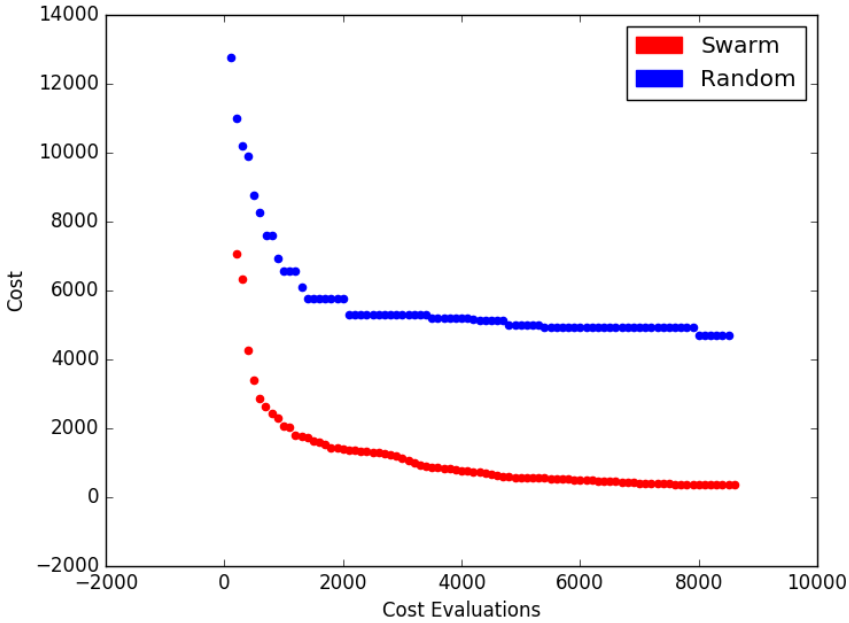

Figure 1: The Average Case

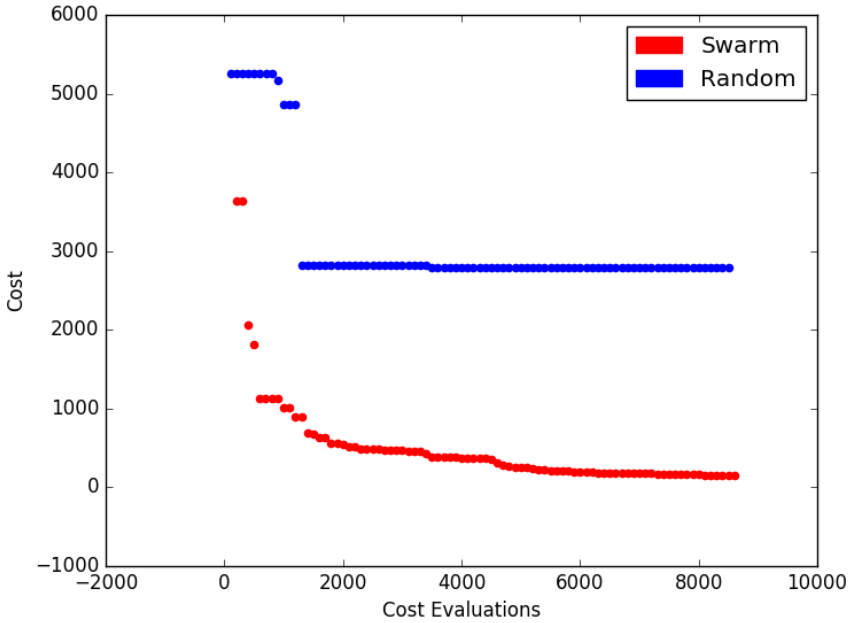

Figure 2: The Best Case

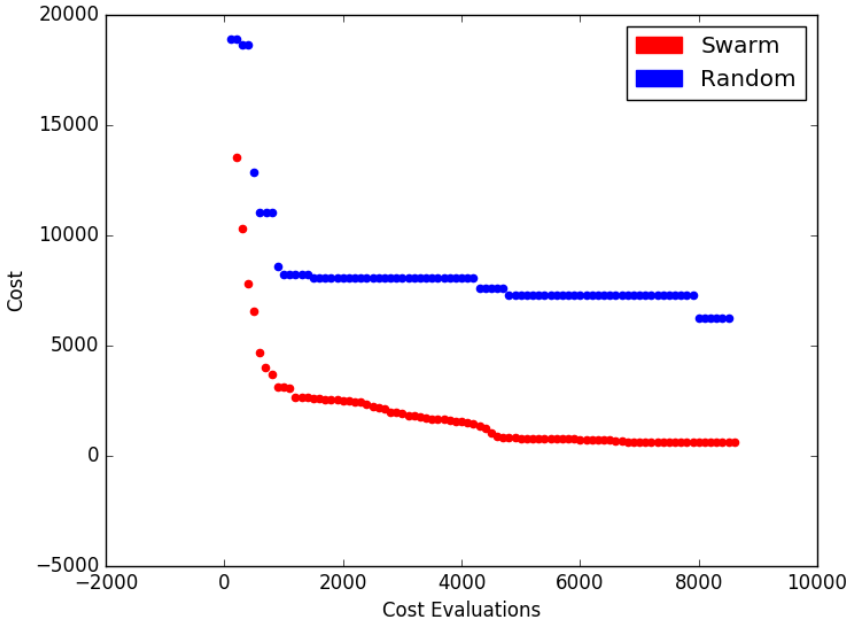

Figure 3: The Worst Case

3 Conclusion

The swarm optimization algorithm outperforms the random search. The swarm algorithm doesn't quite converge to the global minimum in the given time it is allowed to run. This could be due to non-optimum parameters which controls how much the individual should change velocity towards the swarms best or the individuals best. A possible improvement could be to change the parameters at run time, starting with individuals more prone to follow their own optima and later on converge towards the global optima. Another idea could be to set the parameters differently for different individuals.