*Heriot-Watt University*

*School of Mathematical and Computer Science*

**Title**:

# *"Implementing and Comparing Optimisation Algorithms"*

**By**:

Jordan Stuart Walker

(H00222194)

***Biologically Inspired Computation Coursework 3***

## Implementing PSO and a GA

For this assignment we were asked to implement a PSO (Particle Swarm Optimisation) and GA (Genetic Algorithm). Both Algorithms have unique and interesting features and for what could be argued complex topics the implementations of both are relatively simple. Following are what I believe to be the most interesting parts of each algorithm.

### Genetic Algorithm – Selection Method

When selecting the individuals from the initial population when implementing a GA a variety of techniques are available to be implemented. For this assignment it was given in the specification our method of selection was to be a technique referred to as "*Tournament Selection*".

As a short summary tournament selection works by choosing individuals randomly from the initially set population. The winning individual (The one with the best fitness) is then selected for crossover and eventually mutation to the next generation.

### PSO – How it Works

The PSO algorithm is an interesting form of optimisation as what is being created i.e. the swarm works together collectively to find the most optimal solution by utilizing communication between the particles within the storm.

The particles themselves do not search the full search space but each particle essentially has its own search space and if it finds a new "best" solution will update the rest of the swarm which can change its position rapidly when needed towards the swarms best.

## Benchmark Functions Used

To test the performance of the implemented PSO and GA, it is recommended that we perform our evaluation on some form of mathematical functions. With this recommendation, also suggested that we research popular benchmark functions that have been well established as tests for Evolutionary Algorithms.

For this assignment functions from the 2005 CEC Test Functions[1] have been used and are a set of 25 mathematical functions 5 of which are Unimodal and the remaining are Multimodal split into three categories Basic, Expanded and Hybrid. Asked in the assignment brief was that we pick a small number of functions to test our implementations of a GA and PSO on.

I have then chosen to pick four functions to test and compare my implementations these are as follows:

1. F1 – Unimodal Function
2. F6 – Basic Multimodal Function
3. F13 – Expanded Multimodal Function
4. F17 – Multimodal Hybrid Composition Function
5. F5 – Unimodal Function

I believe by choosing these four as means of testing my implementations that I can cover all forms of highlighted mathematical functions and can obtain a fair a fair assessment of how each algorithm performs on a various task and an unbiased evaluation and conclusion can be formed.

## Results of Benchmark Functions Using PSO and GA

Following are the results obtained from the implemented GA and PSO algorithms:

**PSO set parameters:** particles = 100, alpha = 0.85, beta = 1.2, gamma = 1, delta = 1, jumpsize = 1, informants = 20

**GA set parameters:** Iterations = 10, crossover probability = 0.6, mutation rate = 0.1, population size = 50 , max generations = 150

GA

| Benchmark Function | **Optimal** | **Avg. Fitness** | **Best Fitness** | **Difference From Benchmark Optimal (AVG)** | **Difference From Benchmark Optimal (Best)** |
|---|---|---|---|---|---|
| ***F1*** | **-450** | -447.158 | -447.99 | -0.63% | -0.45% |
| ***F6*** | **390** | 5515.663 | 1148.59 | +1314.27% | +194.51% |
| ***F13*** | **-130** | Implementation Did Not Work | | | |
| ***F17*** | **120** | Implementation Did Not Work | | | |
| ***F5*** | **-310** | 1781.27 | -148.63 | -674.6% | -52.05% |

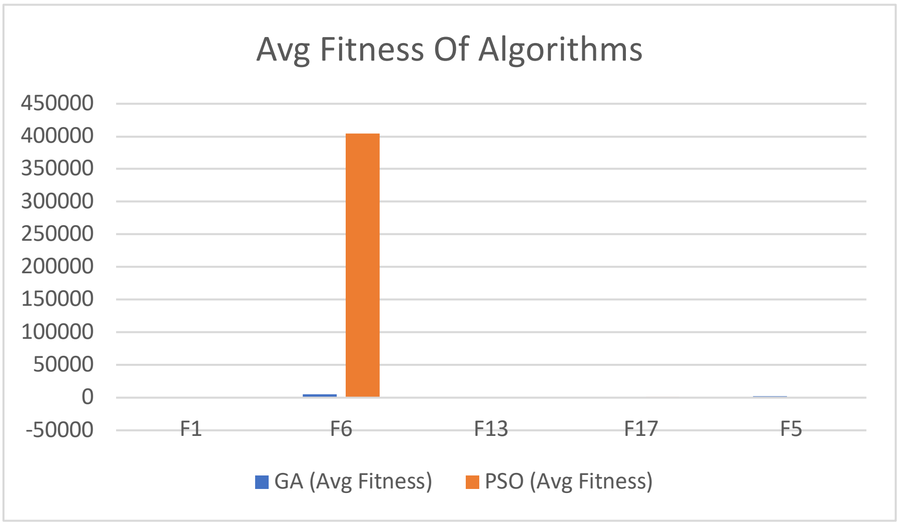
*PSO*

| Benchmark Function | **Optimal** | **Avg. Fitness** | **Optimal Found** | **Difference From Benchmark Optimal** | **Difference From Benchmark Optimal (Best)** |
|---|---|---|---|---|---|
| ***F1*** | -450 | -349.854 | -449.82 | -28.63% | -0.04% |
| ***F6*** | 390 | 404242.96 | 486.64 | +103552.04% | +24.78% |
| ***F13*** | -130 | -127.299 | -128.43 | -2.08% | -1.21% |
| ***F15*** | 120 | 238.417 | 210.95 | +98.68% | +75.79% |
| ***F5*** | -310 | -308.541 | -309.80 | -0.47% | -0.06% |

## Note:

**There was an issue with the GA implementation that would not let me asses functions with bounds that were between [-100, 100].**

## Comparisons on Benchmarks Between PSO and GA



From observation we can deduce that the algorithms both performed well and the average fitness was not too far away from the optimal. This however was not the case for the PSO when searching through Function 6's search space. From an average fitness calculated over 10 runs the PSO received some interestingly adverse results.

However this was not the case for all runs as initialization received a fitness score of 486.64 which is a +24.78% on the optimal solution given in the CEC 2005 Benchmark Documentation.

This then highlights what could be classed as both the advantage and disadvantage of biological programming. What is meant by this is the algorithms are stochastic and are vastly influenced by the initial parameters set and also the "luck" associated with their random placement within the boundaries of a search space.

This in turn can produce optimal results however it can also produce results that are far away from the optimal which in the case of the PSO run on Function 6 was discovered. It is because of results like these that we use benchmark functions to test implementations of Biologically Inspired Algorithms such as PSO and GA.

## Wider Discussion of Changes to Hyper Parameters and Resulting Performance

Because of the nature of PSO and GA the algorithms are never guaranteed to find the optimal. This is due to both algorithms being stochastic which is due to their implementation and the performance they give is greatly influenced by the hyper parameters set before initialisation.

Essentially all stochastic algorithms can be defined as a random search as the hyperparameters set initially determine how far a particle in PSO or Genome in a GA can differentiate from its initial placement into the problem [2].

This essentially means that an optimal solution can be found in the first iteration or the algorithm will return the "best" solution it has found however there is no guarantee that this is the optimum hoped for and as an overall summary each iteration provides better characteristics for the other particles/ genomes to follow.

Because of this is the reason we use benchmarks to test the performance of the set hyperparameters and the implemented algorithm used to solve the benchmark function.

## References

[1] - Suganthan, P., Hansen, N., Liang, J., Deb, K., Y. -P., Y., Auger, A. and Tiwari, S. (2005). Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. 1st ed. [ebook] pp.1 - 50. Available at: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2005/Tech-Report-May-30-05.pdf [Accessed 16 Nov. 2018].

[2] - Collet, P. (2006). Stochastic Optimization Algorithms. 1st ed. [ebook] Sémard, pp.3-10. Available at: https://arxiv.org/pdf/0704.3780.pdf [Accessed 21 Nov. 2018].

[3] – Lones, M 2018, Lecture 7: Particle Swarm Optimization, lecture slides, F20BC, Heriot Watt University, Delivered Week 7

[4] - Turing Finance. (2018). Portfolio Optimization using Particle Swarm Optimization. [online] Available at: http://www.turingfinance.com/portfolio-optimization-using-particle-swarm-optimization/ [Accessed 22 Nov. 2018].

[5] - Luke, S. (2013). Essentials of metaheuristics. Lulu.com.

[6] - Rooy, N. (2018). Particle Swarm Optimization from Scratch with Python. [online] nathanrooy.github.io. Available at: https://nathanrooy.github.io/posts/2016-08-17/simple-particle-swarm-optimization-with-python/ [Accessed 21Nov. 2018].

[7] - Mirjalili, S. (2018). [online] Udemy. Available at: https://www.udemy.com/geneticalgorithm/learn/v4/t/lecture/10322200?start=0 [Accessed 24 Nov. 2018].

[8] - De Rainville, F. (2018). DEAP/deap. [online] GitHub. Available at: https://github.com/DEAP/deap/blob/master/deap/tools/crossover.py?fbclid=IwAR0bUGtAvt9eeSD2JJdb912vr4CGppyOB5Z9fTpJjFgv3HQZu0iUUFHYDXk [Accessed 24 Nov. 2018].

[9]- Cℓinton's Blog. (2018). Genetic Algorithm Hello World with Python. [online] Available at: https://handcraftsman.wordpress.com/2015/06/09/evolving-a-genetic-solver-in-python-part-1/ [Accessed 25 Nov. 2018].