Biologically Inspired Computation: *Assignment 2*

JACK ROME[1]

[*] *Student Number: H00219766, email: jar7@hw.ac.uk*

1. INTRODUCTION

The genetic algorithm and particle swarm optimization algorithm were written to be similar in execution. They share many of the same variables such as number of iterations, population size and the number of dimensions contained in each object (number of genes per chromosome and number of positional dimensions in each particle of the swarm). Due to both algorithms being stochastic, data is averaged after 10 runs and saved to a CSV file. The GA and PSO are tested on problems with 8 dimensions and set boundaries.

2. GENETIC ALGORITHM

A. Implementation

The Genetic Algorithm was implemented in accordance to the set criteria. The implementation features Two-Point Crossover, Tournament Selection and fixed-length vectors of real values. The genetic algorithm was implemented following the pseudo code on page 37 and then expanded upon by using the 'elitism' method on page 46 (Luke, 2015). The Elitism method is identical to the standard genetic algorithm, however, it will add the chromosome with the best fitness to the next population set. This was slightly modified to include the best 2 of the population due to the algorithm being looped N/2 times.

B. Tests

The algorithm was tested with the boundary method and Gaussian mutation. The boundary method involves setting a chosen gene to the upper or lower limits of the equation at random and the Gaussian method involves randomly adding to or subtracting from the gene according to a Gaussian distribution. The results show that the Gaussian gave better results but were still not perfect. A third method was created which involves using the Gaussian method proportionately. The Gaussian method is multiplied by 1% of the gene's current value. The result was not a linear trend with an order of 5,2,3,4. There seems to be no relationship however, the results are contrary to the book's recommendation of t=2 (Luke,2015). A size of t=5 was selected for the rest of the testing. The boundary mutation performed terribly in these tests whilst the customized Gaussian formula, which takes into account the current position of the gene, performed the best. This will be used as the mutation method in future tests.

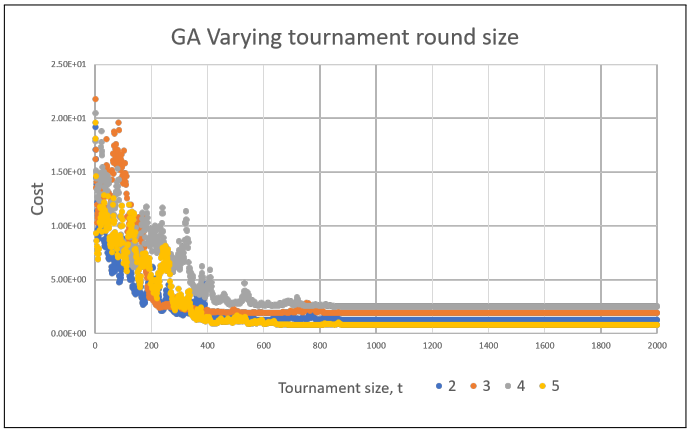

Fig. 1. Results of adjusting the tournament size of the Genetic Algorithm

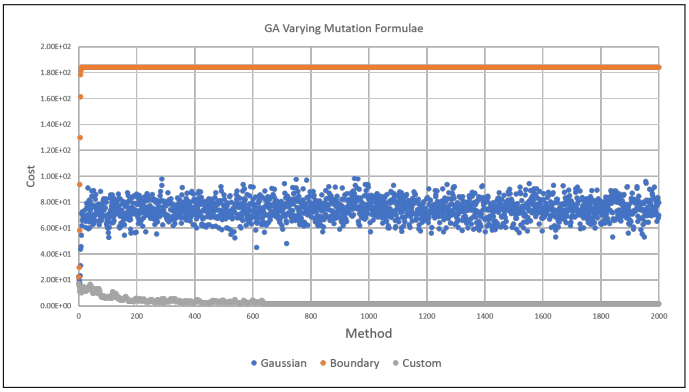

Fig. 2. Results of GA on varying mutation formulae

3. PARTICLE SWARM OPTIMIZATION

A. Implementation

The Particle Swarm Optimization algorithm was implemented in accordance to the set criteria. The implementation feature fixed-length vectors of real values as well as each particle having a group of 6 informants (Garcia-Nieto and Alba, 2012). The informants are randomly assigned at the beginning of the run and the position of the informant with the lowest cost is used for the social weighting of the particle. The formula used for calculating the velocity was the recommended velocity formula found in the lecture slides and Essentials of Metaheuristics (Luke, 2015). This used 4 weights and so experiments were executed to test the effects of varying these weights. It was recommended in the lecture slides on Particle Swarm Optimization that weights Beta, Gamma and Delta (Cognitive, Social and Global weights respectively) added to 4.

B. Tests

Testing was done to observe the effect of the weightings in the velocity formula. The PSO algorithm was ran with the Sphere function with the following weight setups: heavily weighting cognitive, heavily weighting social, heavily weighting global and evenly distributed weights. In all tests, the cognitive, social and global weights all add up to 4 while the inertia weight remains unchanged. By observing the graph, it can be concluded that weighting towards the Cognitive weight yields the best result as it has the lowest cost when the algorithm flat-lines. Weighting towards the global weight or weighting equally gives a poor result from the algorithm. The algorithm is weighted heavily towards the cognitive weight (2,1,1) when used in future.

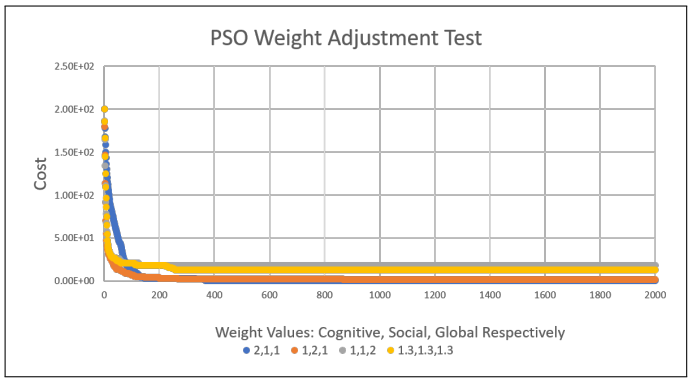

Fig. 3. Results of adjusting the values of cognitive, social and global weight on velocity equation

4. BENCHMARK FUNCTIONS

I chose to write the code for standard benchmark functions as I could not get cec2005real benchmarks to function on Python correctly. All chosen as they are multi-dimensional (KRAMER, 2016). Differing tests with local minimal in most functions. Sphere used for initial testing as it is the simplest and, therefore, quickest to run. The Rastrigin and Griewank have many local minimal pitfalls and are used as examples of difficult tests for the algorithms. The fitness returned from these functions is the result of using the particle's/chromosome's dimensional values as the inputs to the respective function. This is a cost that tells the algorithms how far they are to the optimal point; low costs are, therefore, optimal.

A. Sphere

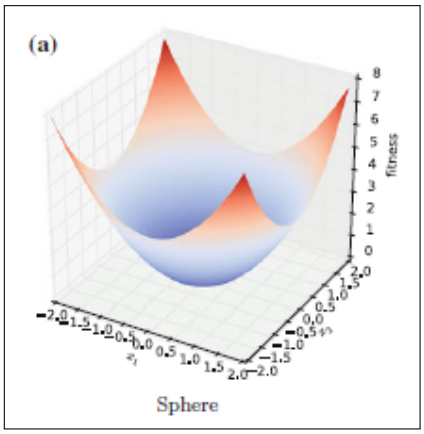

Fig. 4. 3-dimensional representation of Sphere function. (KRAMER, 2016)

$$f(x) = \sum_{i=0}^{N} (x_i{}^2)$$

Global minimum at f(0,...0) = 0. x ∈ [-5.12, 5.12]

B. Cigar

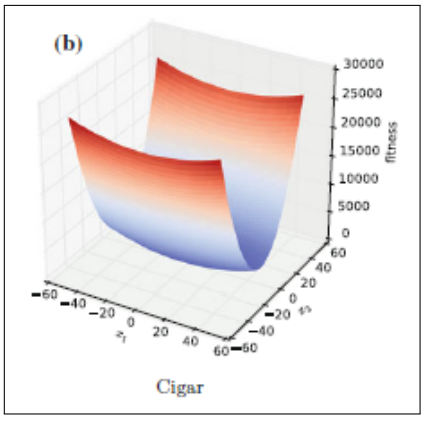

Fig. 5. 3-dimensional representation of Cigar function. (KRAMER, 2016)

$$f(x) = x_i{}^2 + 10 * \sum_{i=0}^{N} (x_i{}^2)$$

Global minimum at f(0,..0) = 0. x ∈ [-5.12, 5.12]

C. Rosenbrock

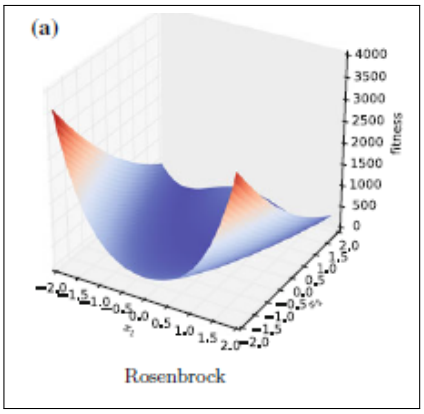

Fig. 6. 3-dimensional representation of Rosenbrock function. (KRAMER, 2016)

$$f(x) = \sum_{i=0}^{N-1} (100((x_i{}^2 - x_{i+1})^2 + (x_i - 1)^2)$$

Global Minimum at f(1,...1) = 0. Typically set x ∈ [-5, 10] however, limitless function

D. Rastrigin

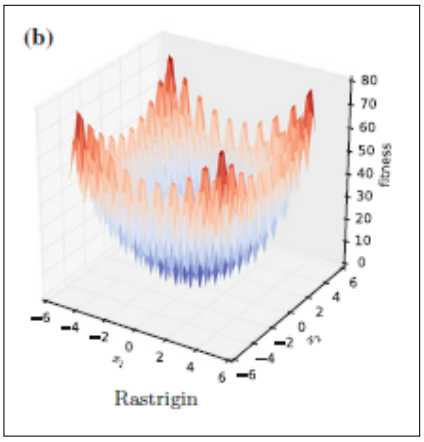

Fig. 7. 3-dimensional representation of Rastrigin function. (KRAMER, 2016)

$$f(x) = \sum_{i=0}^{N} (x_i{}^2 - 10cos(2\pi x_i) + 10))$$

Global Minimum at f(0,...0) = 0. x ∈ [-5.12, 5.12]

E. Griewank

$$f(x) = \sum_{i=0}^{N} (\frac{x_i{}^2}{4000} - \Pi_{i=0}^{N} cos(\frac{x_i}{\sqrt{i}} + 1))$$

Global Minimum at f(0,...0) = 0. x ∈ [-600,600]

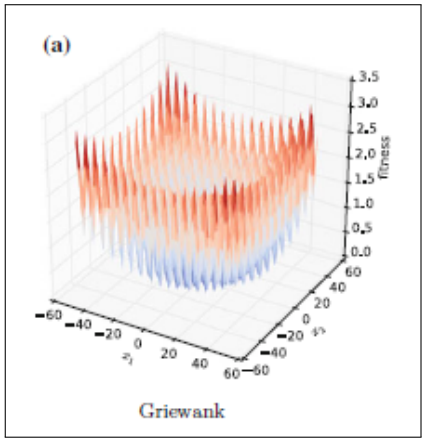

Fig. 8. 3-dimensional representation of Griewank function. (KRAMER, 2016)

5. RESULTS

The algorithms are tested on all of the benchmark functions using the configurations found to make them the most optimal, based upon the previous experiments.

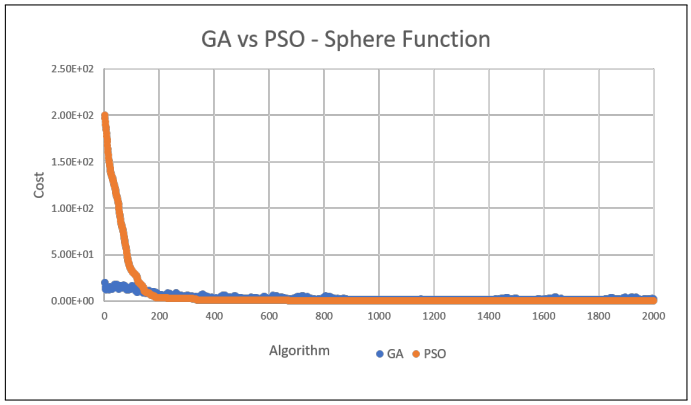

Fig. 9. The PSO started with a randomized set of particles with poor fitness. However, it preformed better than the GA after a rapid readjustment.

The PSO algorithm performed better than the GA in a majority of the benchmark tests, despite it always starting with a higher cost than the GA. This may be because the GA starts at randomized positions whereas the PSO always starts from a set position (all dimensions start at x = 5). Despite this, the PSO finds the solution quickly in the majority of cases. The worst performance for the PSo was on the Rastrigin function which indicates that it takes a few movements for the function to escape local minimums. The GA behaved erratically whereas the PSO would always be descending. I believe this to be because the PSO is more heuristic based than the implemented GA and so it always points towards the optimal or goal. If the GA was implemented with a heuristic, such as a decaying mutation rate, then the amount of times the cost increases during run time is likely to decrease. Whilst implementing the PSO, I found that better results could

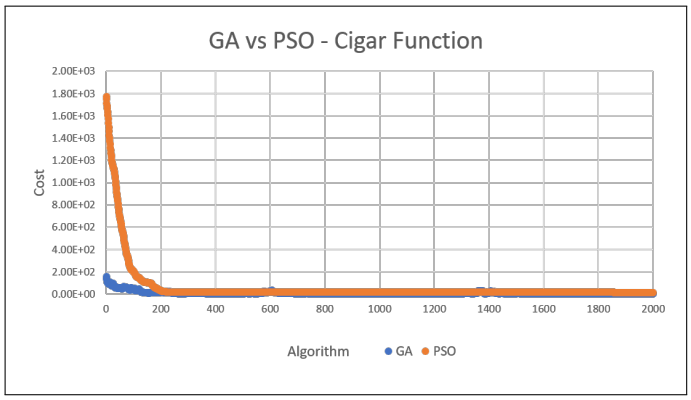

Fig. 10

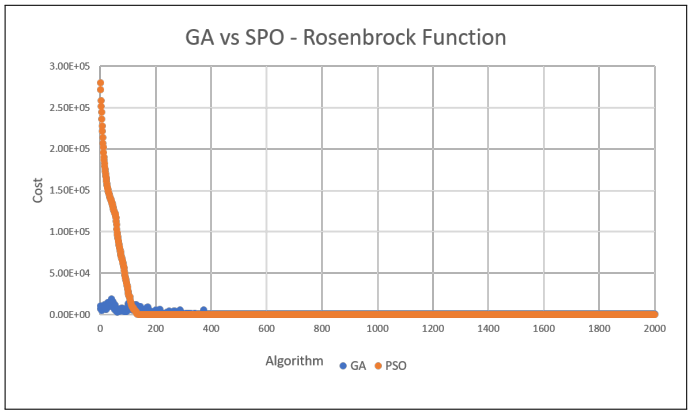

Fig. 11. 3

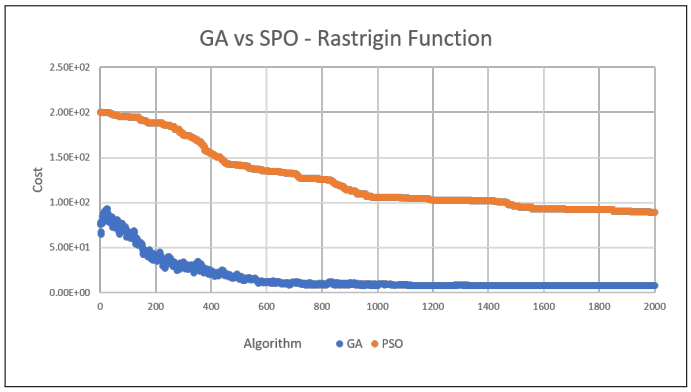

Fig. 12. The PSO performed poorly with this function. This indicates that the algorithm has diffuculty escaping local minimums. It started off with a high cost and did not adapt quick enough for a good result after the 2000 iterations.

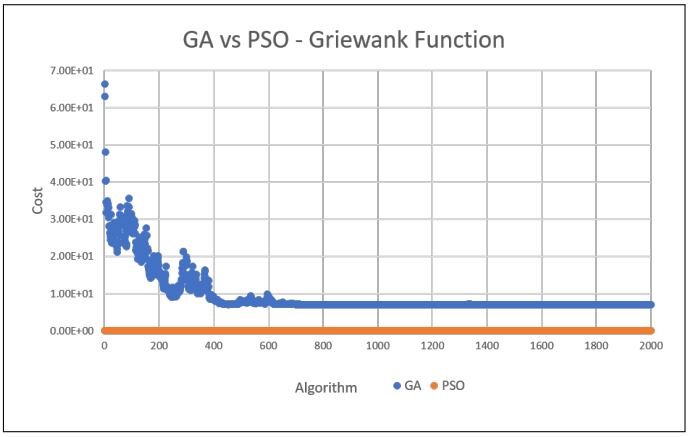

Fig. 13

be achieved without using informants, instead keeping track of the previous best position and weighting using that. The GA could be improved using less randomized methods. For example, replacing tournament selection with systematic sampling, or roulette wheel'. This method distributes the samples based upon their weights. The weights are calculated as the samples contribution to the overall fitness and so the candidates with better fitness have a higher chance of being selected.

6. CITATIONS

- KRAMER, O. (2016). MACHINE LEARNING FOR EVOLUTION STRATEGIES. [S.l.]: SPRINGER, pp.119 - 121.
- Luke, S. (2015). Essentials of metaheuristics. 2nd ed. Lulu.com, pp.37-57.
- Garcia-Nieto, J. and Alba, E. (2012). Why six informants is optimal in PSO. GECCO '12 Proceedings of the 14th annual conference on Genetic and evolutionary computation Pages 25-32. [online] Available at: https://www.researchgate.net/publication/254462427_Why_six_informants_is_optimal_in_PSO [Accessed Nov. 2018].