HERIOT WATT UNIVERSITY

BIOLOGICALLY INSPIRED COMPUTATIONS

COURSEWORK 2

Implementing and Comparing Optimisation Algorithms

Mohit VAISHNAV,

Supervisor
Dr. Michael LONES

Submission Last Date 26^{th}, Nov.

Contents

List of Tables

1 Introduction

Five most known paradigms of evolutionary computations are: evolutionary programming, genetic algorithms, evolution strategies, particle swarm optimization and genetic programming.

Darwin Theory is very critical in developing genetic algorithm (GA). These are very likely to act as optimizer of functions along with other uses like signal processing, pattern discovery and training neural network. While on the other hand, bird folks and fish school kind of scenario were studied to understand their synchronized movements without any collision. This gave birth to Particle Swarm Optimization algorithms. This was initially used as optimization method.

In this coursework, first of all basic version of genetic algorithm and particle swarm optimization are implemented. They are tested over various benchmarks and then genetic algorithms performance is compared with PSO algorithms. Goal of these benchmark is to obtain the global optimum.

2 Working of Genetic Algorithm:

Initially the problem is represented as a system containing chromosomes of fixed length, select the chromosomes population size, their mutation probability and cross over probabilities. Then a fitness function is defined to measure the performance of each chromosome in a system. This function decides the selection procedure of the chromosomes for mating during reproduction. Initial population is then generated randomly and fitness for each chromosomes is carried out. From this current population, a mating pair is selected based on the probability of their fitness which gives priority to highly fit ones. A pair of off-springs are created after applying genetic operators and are placed in new population. This mating steps is repeated till the time new population reaches the size of original and this replaces the parent population. After this again the steps are repeated from calculating the fitness and terminating at a point when the criterion is satisfied.

This is an iterative process with each iteration known as a generation which could range typically from 50 to 500. Being a stochastic process if the solutions is not reached then the whole process is restarted.

3 Working of Particle Swarm Optimization

This optimizes the function on the basis of population based search. Population here consists of particles which are probable solutions. Like a system of flying folks of birds each particles are randomly initialized and allowed to freely fly during which these updates their velocity and position. This update makes the swarm towards high objective function point hence optimization is achieved. They are described as:

- **Initialization** Within a predefined range velocity and position of all particles are set.
- **Velocity Update** Velocities are updated based on the present ones and the best found so far for the entire population. Nevertheless to mention other factors are also present.

- **Position Updating** Within the allowed range positions are updated.
- **Memory Updating** Best positions and objective functions are updated accordingly.
- **Termination** Steps 2-4 are repeated till the termination condition.

4 Differences

Genetic Algorithm have crossover while PSO doesn't but something similar to it. Behaviour of PSO can be taken to be in between evolutionary programming and GA. In PSO any particle can go anywhere and may not reach to a particular point in the problem space for an iteration. Whereas GA is ergodic by definition and can go to a specific point in multiple steps. GA selects the best on the other side PSO do not work that way.

5 Implementation

This coursework is carried out in *Python* framework with one of the most used library **deap** related to *Evolutionary Computation Framework*. Various functions are to be imported here such as: *benchmarks, base, creator, tools and algorithms*.

In the case of GA, individuals are created consisting of random integers in the range desired where chosen value is between 0 and 1. This set is allowed to evolve according to various benchmark functions such as: *h1, Sphere, Cigar, Plane, Fonseca, Kursawe, zdt1, Dent, Griewank*. In Table. 1 various parameters are mentioned to compare their performance over 2 particle, 40 generation system trying to maximize the score using various benchmarks. Attributes generated while simulating are average, mean, standard deviation, minimum & maximum. Python file explains most part of the code in comments section. For clear visualization, different graphs are plotted along with each of the benchmarks solutions to see the average, minimum, maximum and standard deviation of the data as seen in Fig.1. Another visualization is tried with GA to understand the geneology tree with benchmark h1 as seen in Fig. 3. This is carried out with 5 generations and populations size of 10 otherwise a bigger set is different to percieve and looks like the one presented in Fig.

Various evolutionary tools which are listed in [6] for controlling parameters like *Initialization, Crossover, Mutation and Selection*. This library provides many different types of functionality in each section related to genetic programming.

6 Analysis

In GA all three types of operation as crossover, selection and mutations can be implemented in number of ways whereas PSO does not have any such kind of operations. Even within a run, various alterations of GA operation gives varied result. Termination is reached usually because of maximum number of generation limitations.

While Crossover has a significant effect in a run. It moves the chromosomes to a large distance in the space which are initially randomly distributed. At the end of a run, they are converged and crossover has a lesser effect having smaller movements. Even the crossover probability is large at the beginning and becomes smaller at the

Benchmark	Generations	Std_PSO	Max_PSO	Std_GA	Max_GA
h1	40	0.033	0.175	0.0127	0.132
Sphere	40	38.54	3149.24	0.189	2
Cigar	40	3.6e07	3.17e09	189121	2e06
Plane	40	0.38	37.23	0.14	1
Fonseca	40	0	1	.016	0.486
Kursawe	40	9.7e-06	-0.000135	0.1719	-7.563
zdt1	40	0.348	39.919	0.14	1
Dent	40	0.384	41.24	0.16	2.46
Griewank	40	0.195	1.588	0.0568	0.589

Table 1: Comparison of GA and PSO

Benchmark	TS 3		TS 30		TS 100	
	Std_GA	Max_GA	Std_GA	Max_GA	Std_GA	Max_GA
h1	0.0127	0.132	0.0055	0.132	0.0128	0.132
Sphere	0.189	2	0.08	2	0.187942	2
Cigar	189121	2e06	81377	2e06	187942	2e06
Plane	0.14	1	0.05	1	0.14	1
Fonseca	.016	0.486	.0069	0.400	.0161	0.400
Kursawe	0.1719	-7.563	0.0529	-7.536	0.122	-8.187
zdt1	0.14	1	0.0576	1	0.14	1
Dent	0.16	2.46	0.0728	2.46	0.175	2.46
Griewank	0.0568	0.589	0.588	0.240	0.588	0.240

Table 2: Comparison of GA varying tournament size (TS)

ending. However PSO has no such associated process and stochastically each particles is moved towards the best known position and eventually to the globally or locally best based on the version of PSO.

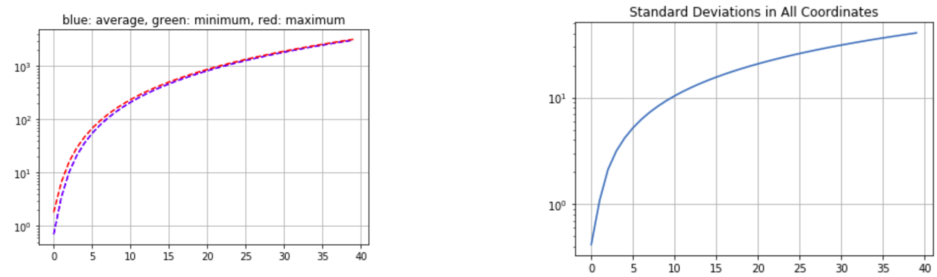
In case of PSO, crossover can be believed to be existing in middle of swarms of particle which are clustering around the local best position. Mutation effect on GA can be seen more at the end of a run instead at the beginning where the populations is randomized. Any changes in the mutation rate is seen at the end. Via mutation, we can theoretically say that GA chromosomes can reach any point in space. But at the end of a run, distant points cannot be covered because of more number of mutations.

Mutations leads to less fitted chromosomes near the end of any run. Because at this point, fitness level if high and populations is converged. So even though mutations will bring the chromosomes into the picture, but they are not well suited enough to make their existence because of high fitness region. So it can be said that even though GA is theoretically ergodic but not in true sense as multiple steps are required to reach. While in case of PSO, for a particle to reach any position in just one run, large velocity is required, probably at the beginning of a run.

Survival of the fittest strategy is used in GA using various selection methods like wheel/tournament selection etc which makes highest fitness chromosomes to pass on to the next generation. In case of PSO, there is no such method and ancestry is defined by the path of particle and all particles are the part of population in a run. In Table. 2 standard deviation and maximum are listed for various benchmarks wrt varying number of tournaments. It can be observed that there is no change in the maximum value but only in standard deviation.

7 Conclusion

The performance of a chosen problem is tested over various well known benchmarks of evolutionary algorithms. Populations size is taken as 2 which helps in reducing the calculation time. This implementations helps in gaining insights related to both the algorithms and their working. Nine different types of benchmarks are chosen from different category like single objective continuous, multi objective continuous in terms of *deap* library or to say uni modal and multi-modal functions.



(a) Min, Max and Average

(b) Standard Deviation

Figure 1: Graphical Presentation for PSO with benchmark h1

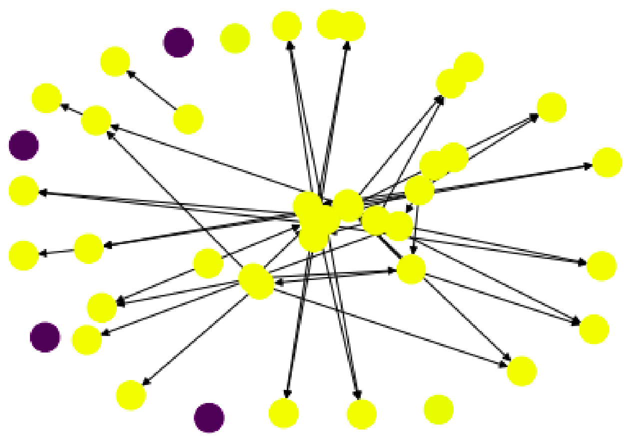


Figure 2: Geneology tree for GA with benchmark h1

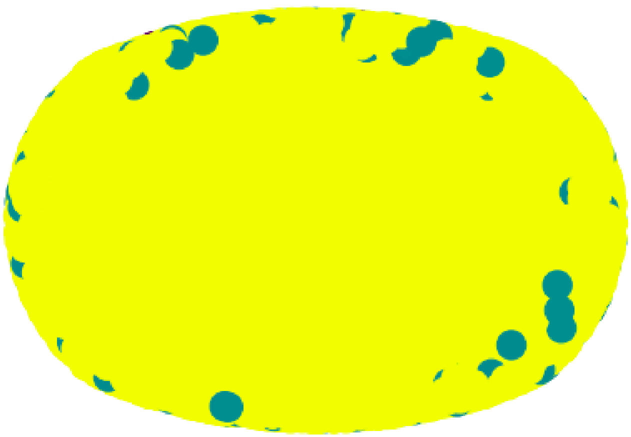


Figure 3: Geneology tree for bigger population

References

[1] https://pdfs.semanticscholar.org/c6d7/300bac7894c5b77d7abcab08a14c4632775c.pdf

[2] https://www.rose-hulman.edu/class/cs/csse453/schedule/day31/GAandPSO.pdf

[3] https://www.researchgate.net/publication/220801072_Comparison_between_Genetic_Algorithms_and_Particle_Swarm_Optimization

[4] https://anaconda.org/conda-forge/deap

[5] https://deap.readthedocs.io/en/master/api/benchmarks.html#module-deap.benchmarks.tools

[6] https://deap.readthedocs.io/en/master/api/tools.html#deap.tools.initRepeat

[7] https://www.rose-hulman.edu/class/cs/csse453/schedule/day31/GAandPSO.pdf

[8] https://www.sciencedirect.com/science/article/pii/S0377042710004796

[9] https://deap.readthedocs.io/en/master/examples/pso_basic.html

[10] https://deap.readthedocs.io/en/master/examples/ga_onemax_short.html one-max-problem-short-version