Optimisation Assignment -2

After thorough understanding and implementing various problems I decided to go with the benchmark function 'schaffer'. Mainly the thought process behind it was that this problem is continuous and as required is non-convex along with unimodal , differential and non-separable. The function mathematically represents as below and was converted as following for implementation -

((x2+x12)*0.25 * ((np.sin(50(x2+x12)**0.1))**2+1.0) for x, x1 in zip(x[1:], x[:-1]))

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} (x_i^2 + x_{i+1}^2)^{0.25} \cdot \left[\sin^2(50 \cdot (x_i^2 + x_{i+1}^2)^{0.10}) + 1.0\right]$$

Note : -(A lot of details about the use, installation and reason for using the library is specified in the code attached.)

For the specific problem I considered and implemented various individual programs and libraries and different parameters , in some cases I even tried to tweak the source code of the library and implement the algorithm based upon my requirement, but failed miserably. After thorough implementation and time spent I decided to go with the following algorithms.

- The Evolutionary algorithms are not meant for large scale optimisation problems and are weak algorithms that work fine with unimodal functions and are easy in objective handling i.e for few parameter and and equations.

- **For all the below types until swarm based algorithms, with the increase in the runs there won't be a notable difference with every increase of run and so doing that won't prove to be much productive.**

Random Search -

- The first algorithm that I chose was this, the implementation was very simple and no built in library was used , the code was tweaked according to the requirement and after 4 runs the attached graph was achieved.
- Based on the plot we can say that the objective was much higher initially and at the end of iterations we can see notable downfall in term of objective function. (refer fig 1)

CMA - Variant (Changes in mean and sigma) -

- This algorithm was chosen as this evolutionary algorithm with the randomised approach of hyperparameter seemed very promising.
- We can see a very notable change of optimised objective in the iterations, this algorithm has its own stopping condition which made it stop at 400th iteration. (refer fig 2)

Original CMA -

- As a requirement for this assignment this basic version of the algorithm was chosen.
- We can see some fluctuations in the graph over first 100 iteration but then after that this becomes quiet stable after that and no notable change was observed.
  Compared to other algorithms this one has initial lower value which showed a strong . (refer fig 3)

CMA -Bipop

- The cma variation using bipop strategy which progressively increments the population .
- This reached its optimum much before in the iterations as compared to other algorithms. (refer fig 4)

- **The algorithms are all swarm based algorithms and all are chosen because they are strong in nature i.e working good with uni-modal, multi-modal, some hybrid and some composite functions along with that they handle large scale data extremely well.**

PSO -

- This is the original version of pso.
- With increase in each run he objective function will move towards the origin more and hence will show a notable difference after more runs. (refer fig 5)

PPSO - Phaser PSO

- Simple and effective variant of PSO: Phaser Particle Swarm Optimisation - in this while creating list instead of uniform nature we take many multipe random geometric constants and add them to the best positions form the list.
- Compared to PSO this variant gives much more optimised output for the problem and with increase in runs it will give much more stable output and efficient one after every run. (refer fig 6)

PSO_W -

- Another interesting variant of Phaser PSO where for the initial list another temporary variant is taken and that is multiplied with each element list to get the final list.
- Based on the trend in the graph and the program it is very hard for this variant and also the blow one (HPSO) to give better result after more runs , so I strongly feel that the result will depreciate after more runs.(refer fig 7)

HPSO -

- This one was by far the most interesting implementation for PSO , it is a hierarchical PSO with jumping time- varyin (refer fig 8)

GWO -

- Grey wolf optimisation is nature based optimisation improved recently that mimics the hunting nature of wolfs.
- Based on the graph and results this algorithm is already well optimised so more runs won't make any changes in the objective function. (refer fig 9)

RW_GWO -

- This is the grey wolf variant where a random values are appended to initial positions.
- This won't show much results after more runs and even though the graph is similar to basic gwo the results (sigma) show notable differences.. (refer fig 10)

Plots -

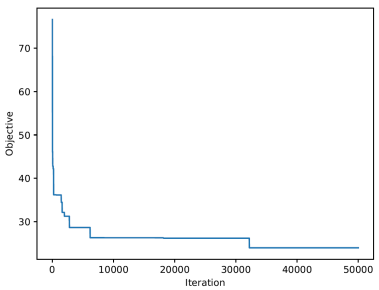

Fig 1: Random Search

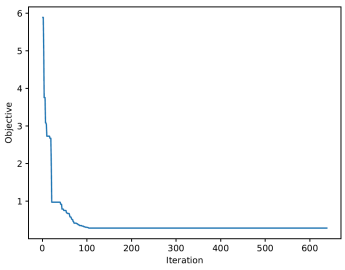

Fig 3 : Base CMA

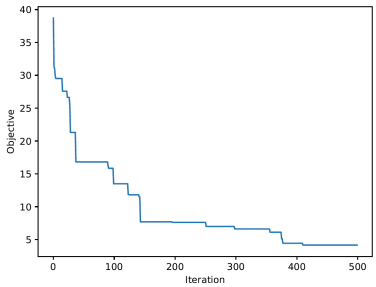

Fig 5 : Base PSO

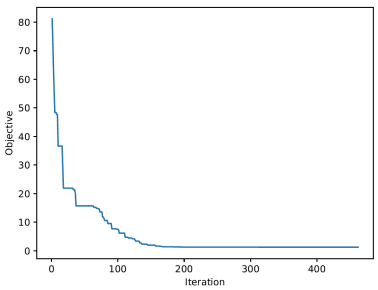

Fig 2 : CMA Variant

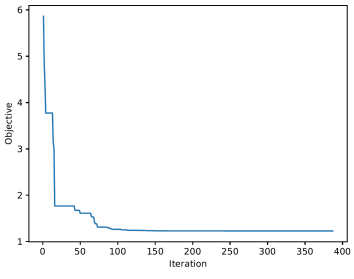

Fig 4 : Bipop CMA

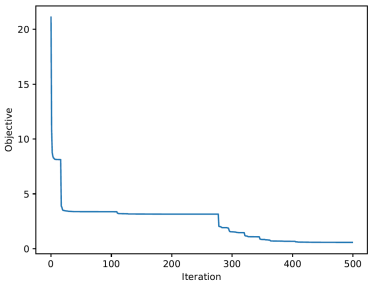

Fig 6 : PPSO

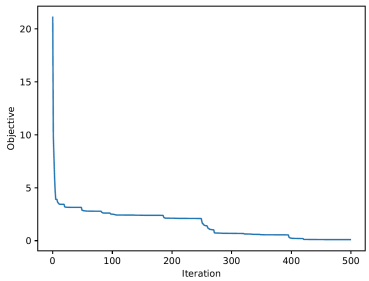

Fig 7 : PPSO_W

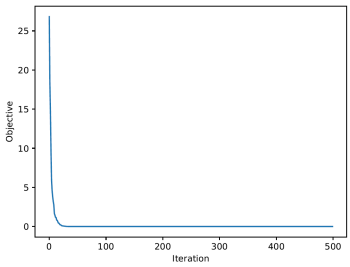

Fig 9 : GWO

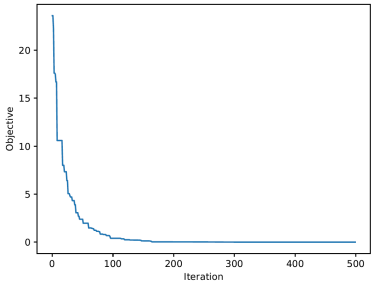

Fig 8 : HPSO

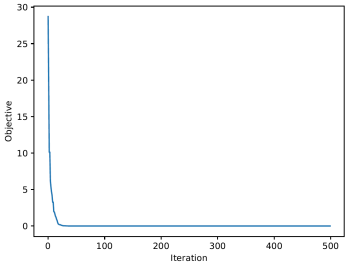

Fig 10 : RW_GWO

Results

Aa Algorithm Name	# Mean	Standard Deviation
Random Search	26.083825412335756	6.748382133033724
CMA - Variant	0.07136533816515618	2.9522623764858023e-9
Basic CMA	0.2669162600920321	2.8651307973104343e-12
Bipop CMA	1.970458	2.026246340971358e-12
RW_GWO	7.716527445746953e-39	3.0683551968645326
Untitled		

Results

Aa Algorithm Name	# Mean	# Standard Deviatio
PPSO	0.6180069809599109	3.323596370535317
PPSO_W	0.2580183093835583	2.266512041373265
HPSO	0.000001123740583462688	8.363942091114216
GWO	1.6697216754058995e-38	2.906289186909805
PSO	5.186172378288352	40.81281936733696

- Based on the the above results we can conclude that the most **optimised algorithm** is the **Bipop CMA** which has the **lowest sd value** and the **best objective value** was achieved with **Random Walk Grey Wolf Optimisation algorithm**.