2022

PARTICLE SWARM OPTIMIZATION

REPORT

IZMA AZIZ , HASNAIN ALI

CONTENTS

PARTICLE SWARM OPTIMIZATION

A SWARM INTELLIGENCE BASED OPTIMISATION ALGORITHM

INTRODUCTION:

Particle Swarm Optimization (PSO) is a stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, that uses the motion of bird flocks and schooling fishing as inspiration. It iteratively tries to improve the solution by using given measure of quality i.e., fitness function.

In this method the particles move with a certain velocity which is calculated in every iteration. Each particle's movement has the influence of his own best-known position and also the best-known position in the space search. The final result expected is that the particle swarm converge to the best solution.

ALGORITHM:

Pseudo code for PSO is as follows:

FOR each particle i
  FOR each dimension d
    Initialize position x_{id} randomly within permissible range
    Initialize velocity v_{id} randomly within permissible range
  End FOR
END FOR
Iteration k=1
DO
  FOR each particle i
    Calculate fitness value
    IF the fitness value is better than p_best_{id} in history
      Set current fitness value as the p_best_{id}
    END IF
  END FOR
  Choose the particle having the best fitness value as the g_best_d
  FOR each particle i
    FOR each dimension d
      Calculate velocity according to the equation
      $v_{id}(k+1) = w\, v_{id}(k) + c_1\, rand_1 (p_{id} - x_{id}) + c_2\, rand_2\, (p_{gd} - x_{id})$
      Update particle position according to the equation
      $x_{id}(k+1) = x_{id}(k) + v_{id}(k+1)$
    END FOR
  END FOR
k=k+1
WHILE maximum iterations or minimum error criteria are not attained

There are two key steps when applying PSO to optimization problems: the representation of the solution and the fitness function. One of the advantages of PSO is that PSO take real numbers as particles. It is not like Genetic Algorithm, which needs to change to binary encoding, or special genetic operators must be used. For example, we try to find the solution for f(x) = x1^2 + x2^2+x3^2, the particle can be set as (x1, x2, x3), and fitness function is f(x). Then we can use the standard procedure to find the optimum. The searching is a repeat process, and the stop criteria are that the maximum iteration number is reached or the minimum error condition is satisfied. Moreover, PSO can be applied to nonlinear functions, it is able to find a minimum or maximum point of highly complex functions having many local minima's and maxima's.

CONTROL PARAMETERS OF PSO:

1. Fitness Function:

The fitness function is a function that maps the values in your particles to a real value that must reward those particles that are close to your optimisation criterion. All of particles have fitness values which are evaluated by the fitness function to be optimized and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. In code we use '**func'** for fitness function.

2. Population Size:

Size of population is the number of Particles. We use '**pop**' in our code to represent the population size.

A good rule of thumb is to set POP = 3x or 4x the problem dimensionality. The population size always depends on the nature of the problem, as expressed in the objective function. I.e., convergence is more problematic for obj. func. that use threshold operators (min, max, etc.) since the search space has discontinuities.

The bigger the population, the better is the performance of the algorithm. However, the computational cost also increases.

3. Inertia:

'**w'** is used for inertia in our code. 'w' controls the inertia of the swarm's movement in Particle Swarm Optimization.

4. Acceleration Coefficients:

'c1' and **'c2'** represent acceleration coefficients. c1 and c2 are the cognitive and social parameters respectively. They control the particle's behaviour given two choices: (1) to follow its personal best or (2) follow the swarm's global best position. Usually, c1 equals to c2 and ranges from [0, 4]

5. Dimension of particles:

Number of dimensions is number of parameters of function. **'n_dim'** is determined by the problem to be optimized.

6. Range of particles:

It is also determined by the problem to be optimized, you can specify different ranges for different dimension of particles.

In code,

lb is the lower bound of every variable of function

ub is the upper bound of every variable of function

7. Vmax:

It determines the maximum change one particle can take during one iteration. Usually, we set the range of the particle as the Vmax for example, the particle (x1, x2, x3) X1 belongs [-10, 10], then Vmax = 20

8. Stopping Condition:

The maximum number of iterations the PSO execute and the minimum error requirement. The stop condition depends on the problem to be optimized.

In code we use '**max_iter'** for maximum number of iterations and this is used as the stopping criteria.

SOURCE CODE:

Scikit-opt [6] is used for implementing the Particle Swarm Optimization which is a Powerful Python module for Heuristic Algorithms. The PSO module is imported from the SKO library. The file PSO.py contains the PSO class which is used for particle swarm optimization.

The rosenbrock.py file contains the code for Rosenbrock's function.

The griewank.py file contains the code for Griewank's function.

TASK

"Use PSO to search for the minimum value of Rosenbrock's and Griewank's functions. Find optimal value of hyperparameters c1, c2, w and population size and determine the fitness function, particle encoding topology and stopping criterion."

OPTIMAL VALUE OF HYPERPARAMETERS:

According to the paper by M. Clerc and J. Kennedy [4] to define a standard for Particle Swarm Optimization, the best static parameters are w=0.72984 and c1 + c2 > 4. More exactly c1 = c2 = 2.05. Additionally, the linear decay of the parameter w was initially proposed by Yuhui and Russ Y. H. Shi and R. C. Eberhart [5]. We used these parameters for initial optimization.

For choosing more optimal value of hyperparameters **w**, **c1**, and **c2** 'HIT AND TRIAL METHOD' is used by tweaking some optimal values from research paper by "Magnus Erik Hvass Pedersen" [3].

A method of generating the optimal coefficients automatically is also used which is inspired from the paper by "G. Sermpinis" [1].

The formulae used in the **update_coef ()** method of class **PSO** to update the coefficients automatically are,

$$w^t = 0.4\frac{(t-N)}{N^2} + 0.4$$

$$c_1^t = -3\frac{t}{N} + 3.5$$

$$c_2^t = +3\frac{t}{N} + 0.5$$

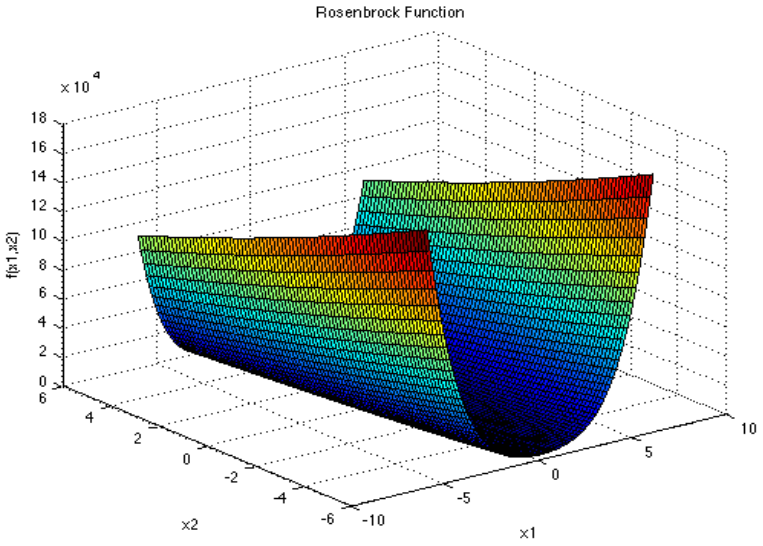
Fitness Function	Rosenbrock's function and Griewank's function
Particle Encoding	Real number encoding
Topology Type	Global topology
Stopping Criterion	Maximum number of iterations

ROSENBROCK'S FUNCTION

The Rosenbrock's function, also referred to as the Valley or Banana function, is a popular test problem for gradient-based optimization algorithms.

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Graphical Representation:



Result Analysis at D=20

Initial Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
60	0.72984	2.05	2.05

Optimal Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
60	-0.4736	-0.9700	3.7904

Coefficients Optimized During Iterations:

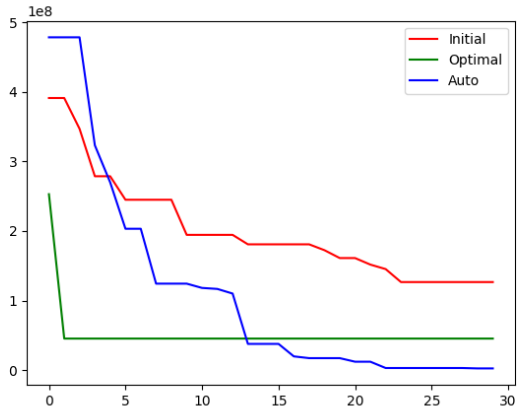
In this technique we put **auto_coef=True** while calling the PSO, the optimal coefficients are generated using the **update_coef ()**.

MEAN AND STANDARD DEVIATION:

```
FOR 20 ITERATIONS

Name       Global Best        Mean    Standard Deviation
-------  -------------  ----------  --------------------
Initial    1.26607e+08  -0.0706658               13.0292
Optimal    4.53229e+07  -1.2806                   9.96956
Auto       2.40044e+06   1.39386                  8.2984
```

ANALYZING RESULT:



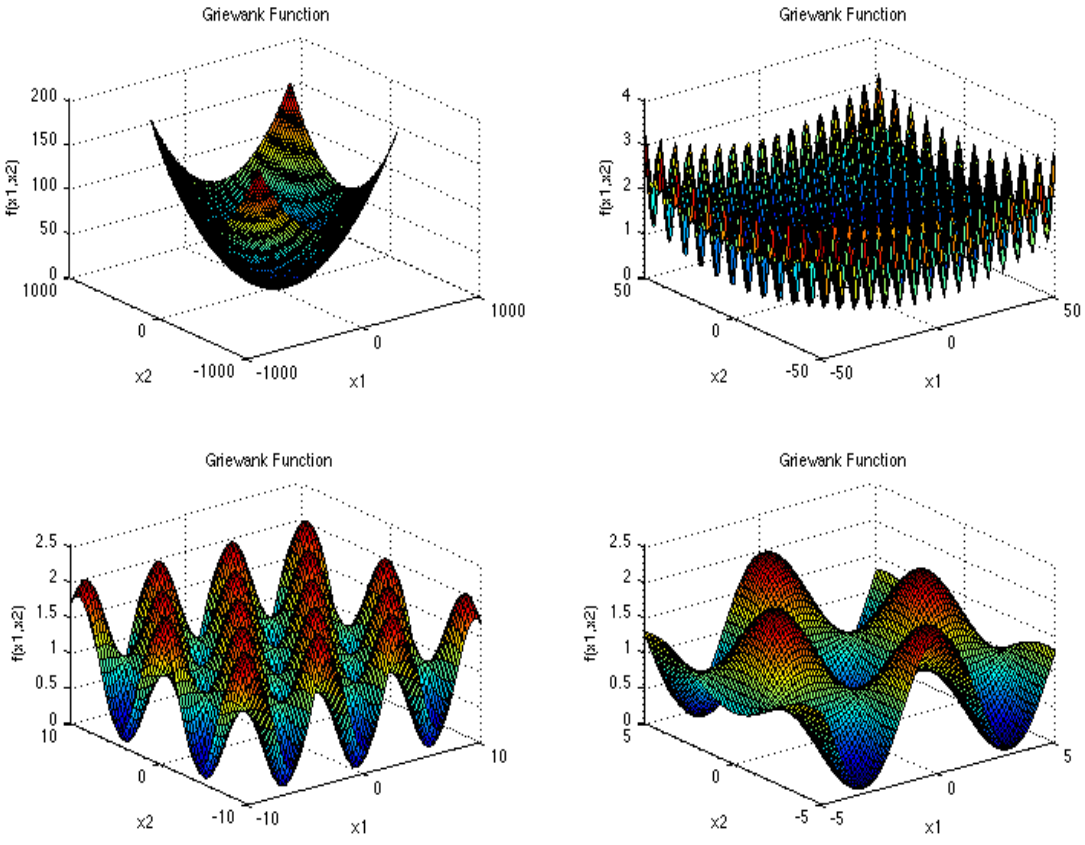
From the above graph it can be concluded that the coefficient generated during Iterations using formulas used in method **update_coef ()** are the most optimal coefficients as they result in minimum value of standard deviation.

GRIEWANK'S FUNCTION

The Griewank's function has many widespread local minima, which are regularly distributed. It is one of the benchmark functions used to analyse PSO.

$$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Graphical Representation:



Result Analysis at D=20

Initial Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
90	0.72984	2.05	2.05

Optimal Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
90	0.8	0.5	0.5

Coefficients Optimized During Iterations:

In this technique we put **auto_coef=True** while calling the PSO, the optimal coefficients are generated using the **update_coef ()**.

MEAN AND STANDARD DEVIATION:

```
FOR 20 ITERATIONS

Name       Global Best       Mean    Standard Deviation
-------  -------------  ---------  --------------------
Initial        1.45386   0.532688                9.51255
Optimal         1.0663   -1.34136                3.40021
Auto           1.08923   0.846927                4.13866
```

Result Analysis at N=50

Initial Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
150	0.72984	2.05	2.05

Optimal Coefficients:

Population Size(pop)	Inertia (w)	Acceleration Coefficient (c1)	Acceleration Coefficient (c2)
150	0.8	0.5	0.5

Coefficients Optimized During Iterations:

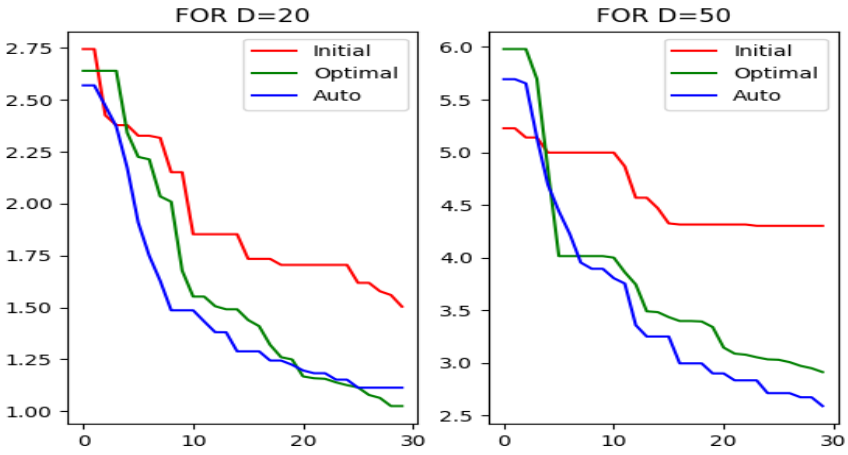
In this technique we put **auto_coef=True** while calling the PSO, the optimal coefficients are generated using the **update_coef ()**.

MEAN AND STANDARD DEVIATION:

```
FOR 50 ITERATIONS

Name       Global Best      Mean    Standard Deviation
-------  -------------  --------  --------------------
Initial        4.30236  1.34924                16.1978
Optimal        2.91405  4.23554                11.6269
Auto           2.59138  0.669801               11.2633
```

ANALYZING RESULT:



From the given result it can be concluded that as the dimension increases it get more and more difficult to reach the global minima as increasing the number of dimensions exponentially increases the number of local minima and maxima of this function. However, the most optimal coefficients are still generated using auto coefficient method.

References:

[1] G. Sermpinis, K. Theofilatos, A. Karathanasopoulos, E. F. Georgopoulos, & C. Dunis, *Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and Particle Swarm Optimization*, European Journal of Operational Research.

[2] R. Eberhart & J. Kennedy, *A New Optimizer Using Particle Swarm Theory*, Sixth International Symposium on Micro Machine and Human Science.

[3] Pedersen, M. E. H., 2010. Good parameters for particle swarm optimization. Tech. Rep. HL1001, Hvass Lab.

[4] Clerc, M., and J. Kennedy. *The Particle Swarm — Explosion, Stability, and Convergence in a Multidimensional Complex Space*. IEEE Transactions on Evolutionary Computation 6, no. 1 (February 2002): 58–73.

[5] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in Proceedings of the IEEE International Conferences on Evolutionary Computation, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[6] https://scikit-opt.github.io/scikit-opt/