**SWARM INTELLIGENCE:PySwarms**

**Using Python**

**Mrs.S.VISHALATCHI**,

**Assistant Professor of Computer Science**,

P.K.R Arts College for Women (Autonomous),

Gobichettipalayam-638476

**Mail ID:** vishalatchispkr@gmail.com

**ABSTRACT**

There is a trend in the scientific community to model and solve complex optimization problems [1]by employing natural metaphors. This is mainly due to inefficiency of classical optimization algorithms in solving larger scale combinatorial and/or highly non-linear problems. One of the main characteristics of the classical optimization algorithms is their inflexibility to adapt the solution algorithm to a given problem. Inorder to overcome these limitations more flexible and adaptable general purpose algorithms are needed. It should be easy to tailor these algorithms to model a given problem as close as to reality. Based on this motivation many nature inspired algorithms were developed. A branch of nature inspired algorithms which are known as swarm intelligence is focused on insect behaviour in order to develop some meta-heuristics which can mimic insect's problem solution abilities .A branch of nature inspired algorithms which are known as swarm intelligence is focused on insect behavior in order to develop some meta-heuristics which can mimic insect's problem solution abilities.[1]On the other hand, modern trends in solving tough optimization problems tend to use evolutionary algorithms[2] and nature-inspired meta heuristic algorithms, especially those based on swarm intelligence (SI).algorithms and ﬁreﬂy algorithm (FA) use the behavior of the so-called swarm intelligence (SI). All SI-based algorithms use the real-number randomness and some form of communication among agents or particles. These algorithms are usually easy to implement as there is no encoding or decoding of the parameters into binary strings as those in genetic algorithms where real-number strings can also be used. In addition, SI-based algorithms are very ﬂexible and yet eﬃcient to deal with a wide range of problems.[2]There are several approaches that can be taken to maximize or minimize a function to find the optimal value. Particle Swarm Optimization (PSO)[3] is also an optimization technique belonging to the field of nature-inspired computing. It is an algorithm that searches for the best solution in space in a straightforward way. In this article, we will discuss Particle Swarm Optimization in detail along with its working and different variants. We will also learn the hands-on implementation of PSO using the python package PySwarms.[3]

**Keywords**

Swarm Intelligence, PySwarms, Particle Swarm Optimization(PSO),Nature Inspired Algorithms

**INTRODUCTION**

There are a number of techniques, developed for optimization, inspired by the behaviours of natural systems (Pham & Karaboga, 2000). In this study, we employ swarm intelligence, a natural optimization technique for optimizing both K-means and SCL algorithms. Optimization algorithms play a significant role in solving the real-world optimization problems. Especially, these algorithms can be compartmentalized different categories using different descriptions. Common names are evolutionary algorithm (EA),nature-inspired algorithm (NIA), meta-heuristic algorithm (MA), and swarm intelligence (SI) algorithm, however, some of the algorithms included are the same. Thus, a challenge of the algorithm is that searching for the optima in the search space with higher convergence speed.

**Genetic Algorithm** [4,5] combines evolution and natural selection, which are applied to its population over generations, and it was proposed in the 1970s. The best chromosomes in the previous generation or generated by crossover and mutation constitutes the next population in the optimization process. The crossover is to inherit a part of the value of two chromosomes from each parent and produces one offspring, which can direct to the exploitation. The mutation is randomly changing some values in a chromosome and responsible for the exploration. Overall, highly random operations make GA avoid falling into local optimum, and slow convergence is its disadvantage at the same time.[4,5]

**Simulated Annealing**[7] was proposed in 1983, one of the most well-known physics-based methods, which is inspired by the annealing in metallurgy. It starts to find the global optimal solution at a high "temperature" and becomes more sensitive as the temperature decreases, that is, the ratio of the difference solution decreases. Thus, the initial temperature and annealing speed are the key indicators that affect whether it can reach the optimum.[7]

**Particle Swarm Optimization** (PSO)[8] algorithm was proposed in 1995 one of the most popular SI methods, which inspired by the bird flocking behaviour. The movements of the particles are affected by the position and speed of the previous generation and the surrounding particles. PSO algorithm has a clearer direction than GA and SA, because it is easy to implement, and the parameters are rarely the outstanding advantages of the PSO. However, it tends to converge to the local optimum prematurely when optimizing multi-modal functions, because it uses the static finite predecessor and group of linear motion.[8]

Above the three methods, their variations have been proposed, such as Quantum PSO, Adaptive PSO, and Hybrid GA with SA, etc.

**METAHEURISTICS**

The first main division of meta-heuristics is evolution-based methods. Such evolutionary algorithms normally mimic evolutionary rules in nature, some of the most well-known techniques are Genetic algorithm (GA) , Genetic programming(GP), Differential evolution (DE), Evolutionary programming (EP) , Biogeography-based optimizer (BBO) ,Gradient evolution algorithm (GEA) , and Tree-seed algorithm (TSA).

The second main division of meta-heuristics is swarm-based approaches. These SI algorithms currently mimic swarm behaviors in animals. Some of the most popular algorithms are Particle swarm optimization (PSO) , Ant colony optimization,(ACO), Firefly algorithm (FA), Cuckoo search (CS), Grey wolf optimizer (GWO), Salp swarm algorithm,(SSA) , and Marine-predators algorithm (MPA) . In addition, some well-known SI algorithms are Whale optimization algorithm (WOA) inspired by the foraging and hunting of the whales in the ocean, Moth-flame

optimization (MFO) inspired by the navigation approach of moths, and Butterfly optimization algorithm (BOA) inspired by the foraging and mating behaviors of butterflies, etc.

The third main division of meta-heuristics is physics-based methods. These optimization algorithms usually mimic physical principle. Some of the well-known methods are Simulated annealing (SA), Gravitational search algorithm (GSA) , Water cycle algorithm (WCA), Sine cosine algorithm (SCA), Henry gas solubility optimization (HGSO) algorithm, and Archimedes optimization algorithm (AOA) . It is worth mentioning that AOA is proposed in 2021 by Fatma A. et al, which inspired from the phenomenon explained by Archimedes’ principle. Also, Equilibrium optimizer (EO) and Gradient-based optimizer (GBO) are proposed for solving the numerical optimization problems inspired by the physical rules.

The fourth main division of meta-heuristics is human social behavior-based tools. Such optimization algorithms typically mimic social behavior rules in humans. Some of the popular algorithms like Harmony search (HS), Imperialist competitive algorithm (ICA), Teaching learning-based optimization (TLBO), Socio evolution and learning optimization (SELO) and Political optimizer (PO). We divide HS algorithm into social behavior is based on the harmony that only humans can sing, and its principle include the description of the propagation of musical sound. For a more detailed review, different categories can refer to the literature.

Overall, various SI methods have been proposed recently. Most of these approaches is inspired by foraging, mating, hunting and searching behaviors of animals in nature.

**SWARM INTELLIGENCE**

Swarm intelligence is a collective intelligence of groups of simple agents [9]. Swarm intelligence deals with collective behaviors of decentralized and self-organized swarms, which result from the local interactions of individual components with one another and with their environment [9]. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions among such agents often lead to the emergence of global behavior.

The concept of individual–organization [10] has been widely used to understand collective behavior of animals. The principle of individual–organization indicates that simple repeated interactions between individuals can produce complex behavioral patterns at group level [10]. The agents of these swarms behave without supervision and each of these agents has a stochastic behavior due to its perception from,and also inﬂuence on, the neighborhood and the environment. The behaviors can be accurately described in terms of individuals following simple sets of rules. The existence of collective memory in animal groups.

Grouping individuals often have to make rapid decisions about where to move or what behavior to perform, in uncertain or dangerous environments. Groups are often composed of individuals that differ with respect to their informational status,and individuals are usually not aware of the informational state of others. Some animal groups are based on a hierarchical structure according to a ﬁtness principle known as dominance.

Unike EAs, which are primarily competitive among the population, PSO and ACO adopt a more cooperative strategy. They can be treated as ontogenetic, since the population resembles a multicellular organism optimizing its performance by adapting to its environment

**GROUP BEHAVIORS**

The organization of collective behaviors in social insects can be understood as a combination of the four functions of organization: coordination, cooperation, deliberation, and collaboration [11]. The coordination function regulates the spatiotemporal density of individuals, while the collaboration function regulates the allocation of their activities. The deliberation function represents the mechanisms that support the decisions of the colony, while the cooperation function represents the mechanisms that overstep the limitations of the individuals. Together, the four functions of organization produce solutions to the colony problems.

The extracted general cooperative group behaviors, search strategies, and communication methods are useful within a computing context [11]

**Cooperation and group behavior** Cooperation among individuals of the same or different species must beneﬁt the cooperators, whether directly or indirectly. Socially, the group may be individuals working together for mutual beneﬁt, or individuals each with their own specialized role. Competition for the available resources may restrict the size of the group.

**Search strategies** The success of a species depends on many factors, including its ability to search effectively for resources, such as food and water, in a given environment. Search strategies can be broadly divided into sit and wait (for ambush) and foraging widely (for active searchers). Compared to the latter, the former has a lower opportunity to get food, but with a low energy consumption.

**Communication strategies** Inter-group communication is necessary for group behavior. Communication strategies are often multimodal and can be either direct or indirect.

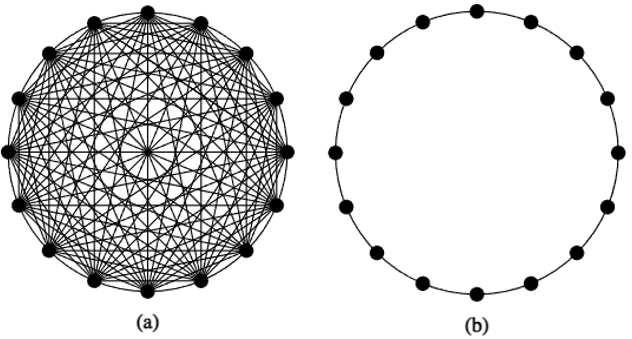
**FORAGING THEORY**

Natural selection has a tendency to eliminate animals having poor foraging strategies and favor the ones with successful foraging strategies to propagate their genes. After many generations, poor foraging strategies are either eliminated or shaped into good

ones. Foraging can be modeled as an optimization process where an animal seeks to maximize the energy obtained per unit time spent in foraging, or to maximize the long-term average rate of energy intake, under constraints of its own physiology and environment. Optimization models are also valid for social foraging where groups of animals cooperatively forage.In general, a foraging strategy involves ﬁnding a patch of food, deciding whether to proceed and search for food, and when to leave the patch. There are predators and risks, energy required for travel, and physiological constraints (sensing, memory,cognitive capabilities). Foraging scenarios can be modeled and optimal policies can be found using dynamic programming.

**BASIC PSO ALGORITHMS**

PSO was developed by Kennedy and Eberhart in 1995 and has become one of the most widely used SI-based algorithms. The PSO algorithm[13] searches the space of an objective function by adjusting the trajectories of individual agents, called particles, as the piecewise paths formed by positional vectors in aquasi-stochastic manner. The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best ***g*** ∗ and its own best location ***x***∗*i* in history, while at the same time it has a tendency to move randomly.



When a particle ﬁnds a location that is better than any previously found loca-tions, then it updates it as the new current best for particle *i*. There is a current best for all *n* particles at any time *t* during iterations. The aim is to ﬁnd the global best among all the current best solutions until the objective no longer improves or after a certain number of iterations. The movement of particles is schemat-ically represented in where ***x***∗*i* is the current best for particle *i*, and ***g*** ∗ ≈ min{*f* (***x****i*)} for (*i* = 1*,* 2*,* … *, n*) is the current global best at *t*.

The essential steps of the PSO can be summarized as the pseudocode shown in Algorithm.

Particle i

-*xi\**

-g\*

Schematic representation of the motion of a particle in PSO, moving toward the global best ***g***∗ and the current best ***x*** ∗*i* for each particle *i*.

**Algorithm-** Pseudocode of particle swarm optimization.

Objective function *f* (***x***)*,* ***x*** = (*x*1*,* … *, xD*)T

Initialize locations ***x****i* and velocity ***v****i* of *n* particles.

Find ***g*** ∗ from min{*f* (***x***1 )*,* … *, f* (***x****n*)} (at *t* = 0)

**while** (criterion)

**for** loop over all *n* particles and all *D* dimensions

Generate new velocity ***v****ti*+1 using Eq. (14.2)

Calculate new locations ***x****ti*+1 = ***x****ti* + ***v****ti*+1

Evaluate objective functions at new locations ***x****ti*+1

Find the current best for each particle ***x***∗*i*

**end for**

Find the current global best ***g*** ∗

Update *t* = *t* + 1 (pseudo time or iteration counter)

**end while**

Output the ﬁnal results ***x***∗*i* and ***g*** ∗

Let ***x****i* and ***v****i* be the position vector and velocity for particle *i*, respectively.

The new velocity vector is determined by the following formula

***v****ti*+1 = ***v****ti* + *𝛼****𝝐***1 [***g*** ∗ − ***x****ti* ] + *𝛽* ***𝝐***2[***x***∗(*i t*) − ***x****ti* ]*,*

where ***𝝐***1 and ***𝝐***2 are two random vectors, and each entry taking the values between 0 and 1. The parameters *𝛼* and *𝛽* are the learning parameters or acceleration constants, which can typically be taken as, say, *𝛼* ≈ *𝛽* ≈ 2.[13]

As stated in the study, fish or a flock of birds moving in a group “may benefit from the experience of all other members.” In other words, if a bird is flying around randomly looking for food, all of the birds in the flock can share their discoveries and help the entire flock get the best hunt. While we may imitate the movement of a flock of birds, we can also assume that each bird is assisting us in finding the optimal solution in a high-dimensional solution space, with the best solution found by the flock being the best solution in the space.

#### ****INNER WORKING OF PSO****

Researchers believe that swarm behaviour differs between exploratory behaviour (searching a larger part of the search space) and exploitative behaviour seeking a smaller region of the search space to get closer to a (potentially local) optimum. Since PSO’s inception, according to researchers, the PSO algorithm[14] and its parameters must be designed to strike an appropriate balance between exploration and exploitation in order to avoid early convergence to a local optimum while ensuring a good rate of convergence to the optimum.

###### Convergence

In PSO convergence, Regardless of how the swarm operates, convergence to a local optimum occurs when all personal bests P or, alternatively, the swarm’s best-known position G approaches a local optimum of the problem.

The ability of a PSO algorithm to explore and exploit may be affected by its topology structure; that is, with a different structure, the algorithm’s convergence speed and ability to avoid premature convergence on the same optimization problem will be different because a topology structure determines the search information sharing speed or direction for each particle. The two most common topological structures are the global star and the local ring.

 A PSO with a global star structure, in which all particles are connected, has the shortest average distance in the swarm, while a PSO with a local ring structure, in which every particle is connected to two nearby particles, has the highest average distance in the swarm.The experimental study examines two commonly utilized architectures, the global star structure (figure 1a) and the local ring structure (figure 1b). There are 16 particles in each group. It should be emphasized that the closest particle in the local structure is primarily determined by the particle index.

###### Adaptive Mechanism

An adaptive mechanism can be implemented without the necessity for a trade-off between convergence (‘exploitation’) and divergence (‘exploration’). Adaptive particle swarm optimization ([APSO](http://eprints.gla.ac.uk/7645/1/7645.pdf)) outperforms regular particle swarm optimization (PSO). With a faster convergence time, APSO can execute global searches across the entire search space.

It allows for real-time modification of the inertia weight, acceleration coefficients, and other computational factors, resulting in increased search efficacy and efficiency. APSO can also operate on the best particle globally to jump out of the most likely local optima.

#### ****Variants of PSO****

Even a simple PSO algorithm can have a lot of different variations. There are various ways to initialize the particles and velocities (for example, start with zero velocities), only update Pi and G after the entire swarm has been updated, and so on.

###### Gradient PSO

To construct gradient-based PSO algorithms, the ability of the PSO algorithm to efficiently explore many local minimums can be combined with the ability of gradient-based local search algorithms to effectively calculate an accurate local minimum.

The PSO algorithm is used in gradient-based PSO algorithms to explore several local minima and discover a location in the basin of attraction of a deep local minimum. The deep local minimum is then properly located using efficient gradient-based local search techniques.

###### Hybrid PSO

In order to increase optimization performance, new and more advanced PSO variations are being introduced. There are certain developments in that study, such as developing a hybrid optimization approach that combines PSO with other optimizers, such as combining PSO with biogeography-based optimization and including an effective learning mechanism.

#### ****Implementing Particle Swarm Optimization using PySwarms****

PySwarms is a Python-based tool for particle swarm optimization. It is used by swarm intelligence researchers, practitioners, and students who want to use a declarative high-level interface to apply PSO to their issues. PySwarms offers interaction with swarm optimizations and basic optimization with PSO.PySwarms implements many-particle swarm optimization techniques at a high level. As a result, it aspires to be user-friendly and adaptable. Supporting modules can also be employed to assist you with your optimization problem.

In this section, we will implement the global-best optimizer using PySwarms’s functional API pyswarms.single.GBestPSO. We will plot the functions in the 2D and 3D manner as well.PySwarm can be straightway installed by pip install pyswarms

###### Optimizing the Function of Sphere

# Import PySwarms

import pyswarms as ps

from pyswarms.utils.functions import single\_obj as fx

We’ll work on improving the sphere function. Let’s put some arbitrary settings in our optimizers for now. At the very least, there are three steps to optimization:

* To configure the swarm as a dict, set the hyperparameters.
* Pass the dictionary along with the relevant inputs to create an instance of the optimizer.
* Invoke the optimize() method, and tell it to save the best cost and position in a variable.

# Set-up hyperparameters

options = {'c1': 0.5, 'c2': 0.3, 'w':0.9}

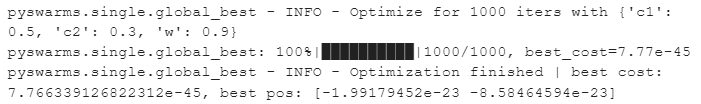
# Call instance of PSO

optimizer = ps.single.GlobalBestPSO(n\_particles=10, dimensions=2, options=options)

# Perform optimization

cost, pos = optimizer.optimize(fx.sphere, iters=1000)

This will run the optimizer for 1000 iterations before returning the swarm’s best cost and position.



###### Visualizing the Function

PySwarms includes tools for visualizing your swarm’s behaviour. These are constructed on top of matplotlib, resulting in user-friendly and highly customizable charts. The plotters module has two animation methods: plot contour() and plot surface(). These approaches, as the name implies, plot the particles in a 2-D or 3-D space.

In order to plot the sphere function, we should add meshes to our swarm. This allows us to see where the particles are in relation to our objective function graphically. Using the Mesher class, we can achieve this.

import matplotlib.pyplot as plt

from pyswarms.utils.plotters import plot\_contour, plot\_surface

from pyswarms.utils.plotters.formatters import Designer

from pyswarms.utils.plotters.formatters import Mesher

The pyswarms.utils.plotters.formatters module contains many formatters for customizing your plots and visualizations. Aside from Mesher, there’s a Designer class for modifying font sizes, figure sizes, and so on, as well as an Animator class for setting animation delays and repeats.

###### 2-D Plot

m = Mesher(func=fx.sphere)

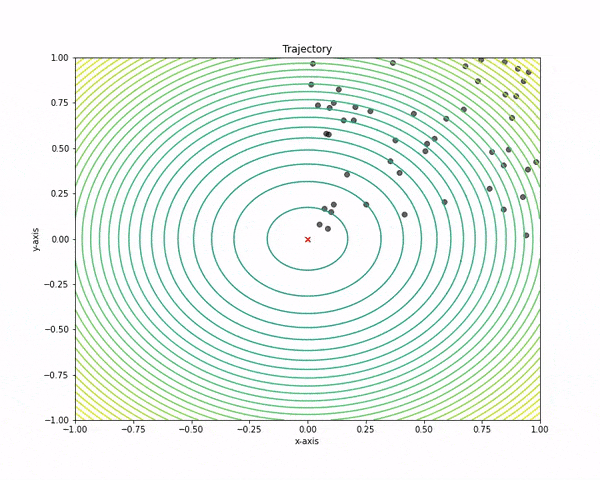
# Make animation

animation = plot\_contour(pos\_history=optimizer.pos\_history,

mesher=m,

mark=(0,0)) # Mark minima

animation.save('mymovie.mp4')



###### 3-D Plot

# preprocessing

pos\_history\_3d = m.compute\_history\_3d(optimizer.pos\_history)

# adjust the figure

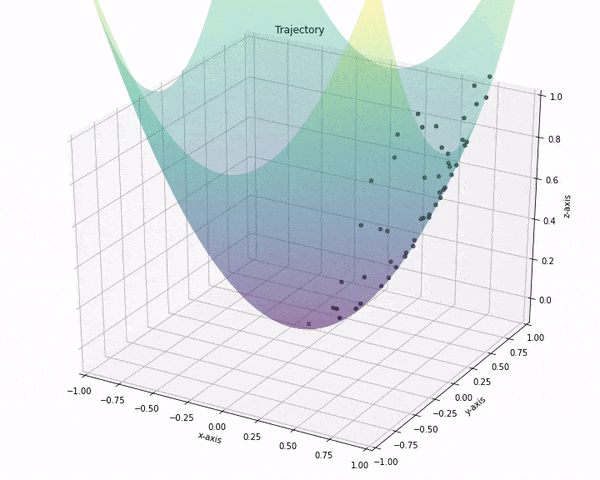
d = Designer(limits=[(-1,1), (-1,1), (-0.1,1)], label=['x-axis', 'y-axis', 'z-axis'])

# Make animation

animation3d = plot\_surface(pos\_history=pos\_history\_3d,

mesher=m, designer=d,

mark=(0,0,0)) # Mark minima[14]

**CONCLUSIONS**

PSO is the first and foremost algorithm which is the most successful algorithm especially for the solution of the continuous optimization problems. Subsequent algorithms using swarm intelligence nearly have the same ideas. All of them are population based, and all particles are distributed in the solution space. Particles have initial locations and velocities. They converge to the optimum solution using their swarm intelligence.

Like all other SI-based optimization approaches, PSO has some drawbacks like premature, high computational complexity, slow convergence, sensitivity to parameters, and so forth. The reasons behind the problems are complicated. One potential reason is that PSO does not utilize the crossover operator as employed in GA or DE; hence the distribution of good information between candidates is not at a required level. Another reason may fall within the fact that PSO does not appropriately handle the relationship between exploitation (local search) and exploration (global search), so it usually converges to a local minimum quickly.

Now there are over hundreds of both various PSO variants and test functions at present. It is impossible for each newly proposed PSO variant to compare with all other variants and to go through all test functions. Therefore, it is difficult to proclaim which type of modification is better or promising.

Although there have been improvements in the optimization algorithms using swarm intelligence, there is not a unique algorithm which is successful in all types of optimization problems. So the efforts trying to simulate the animal behaviors and swarm intelligence will continue. At the same time, developing hybrid algorithms will also continue until the best combination of the algorithms would be found.

**REFERENCES**

1. Felix T.S.Chan and Manoj Kumar Tiwari, “Swarm intelligence Focus on ant and particle swarm optimization”, I-Tech Education and Publishing,PP.165-169
2. Xin-She Yang, “Optimization Techniques and Applications with Examples”,Wiley., PP.297-338,2018
3. A tutorial on Particle swarm Optimization using python by vijay sinh lendave

<https://analyticsindiamag.com/category/developers_corner>

1. Holland J H (1992) Genetic algorithms. Scientific American 267(1): 66-72.
2. Sastry K, Goldberg D, Kendall G (2005) Genetic algorithms. Search Methodologies. Springer, Boston, MA
3. Kirkpatrick S, Gelatt J C D, Vecchi M P (1983) Optimization by simulated annealing. Science 220(4598): 671-680
4. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: the IEEE International Conference on Neural Networks, Perth, Australia, pp 1942-1948.
5. Mengjian Zhang1, Guihua Wen1, \*, Jing Yang2,” Duck swarm algorithm: a novel swarm intelligence algorithm” School of Electrical engineering, Guizhou University, Guizhou 550025, China-link.PP.167-168
6. Bonabeau E.Doriga M.Theraulaz G.Swarm Intelligence:Form Natural to artificial systems,New York:Oxford University Press:1999
7. Sumper D.The principles of collective animal behaviour,philos Trans R Soc B.2006:36(1465):5-22
8. Banks A, Vincent J, Phalp K. Natural strategies for search. Nat Comput. 2009;8:547–70.
9. Felix T.S.Chan and Manoj Kumar Tiwari, “Swarm intelligence Focus on ant and particle swarm optimization”, I-Tech Education and Publishing,PP.103,113-117

13.K.-L. Du and M.N.S. Swamy, “Search and Optimization by Metaheuristics,”,DOI 10.1007/978-3-319-41192-7\_9,Springer International Publishing Switzerland,2016PP:154-156

14. A tutorial on Particle swarm Optimization using python by vijay sinh lendave

<https://analyticsindiamag.com/category/developers_corner>