Particle swarm optimization Algorithm

Instructor: Samer El Zant

Group:

Gurbanzade Orkhan

Murshudov Ramiz

Imamverdiyev Yusif

Project Work Date: 18/12/2021

Introduction

Particle Swarm Optimization (PSO) 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. This is the method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. This algorithm was presented in 1995 by Kennedy and Eberhard. The algorithm was inspired by the concept of swarm intelligence, which is commonly exhibited in animal groupings such as flocks and shoals.

In this particular work we will review PSO as a whole, it's implementation, application and resources related to this algorithm. We will also discuss particle swarms concept, optimization problem example and algorithm details such as search strategy and pseudo code.

Algorithm's theoretical description

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 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.

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.

An adaptive mechanism can be implemented without the necessity for a trade-off between convergence ('exploitation') and divergence ('exploration'). Adaptive particle swarm optimization (APSO) 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 optimum.

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.

Search in PSO

This may be the most important slide of this presentation, for it summarizes the core of the method.

1. Use Case Diagram

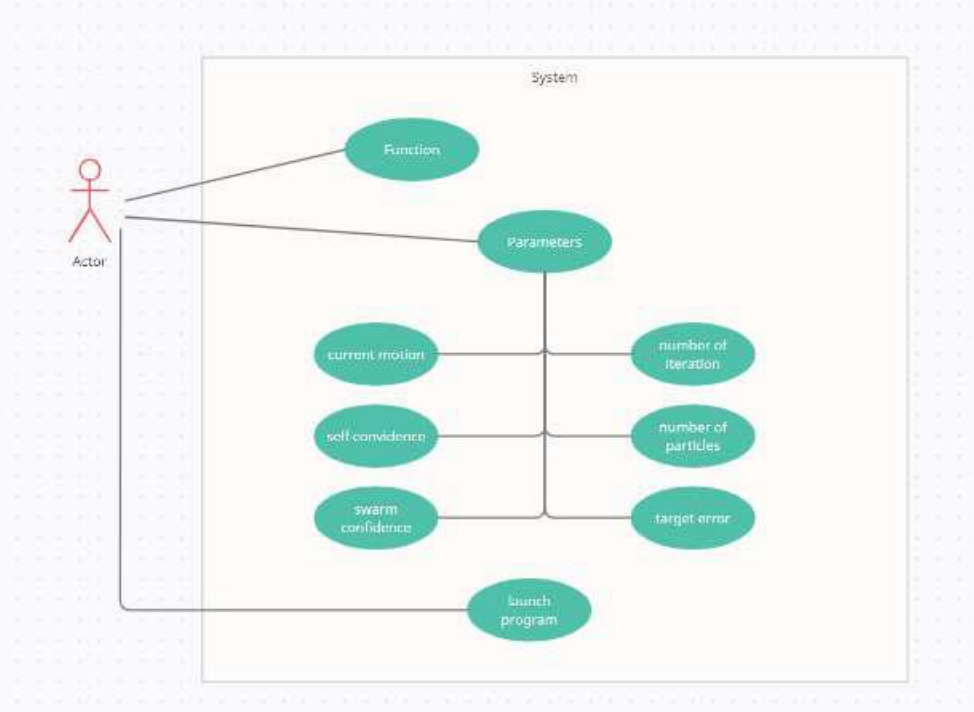

Diagram 1

Our team, which is represented as "System" in Diagram 1, constructs the algorithm as defined in the Use Case Diagram. To utilize the program that our team has prepared, the "user" just inputs the inertia factor range, self-confidence range, and swarm confidence range. During the flow description, the algorithm will process the input and perform all of the procedures listed above. After all, it will provide the user with the optimum result that they are seeking. The main idea is particle swarm optimization (PSO), a computer approach for optimizing a problem by iteratively trying to improve a candidate solution in terms of a specific quality measure in computational science. It solves a problem by generating a population of possible solutions, which are referred to as particles, and spreading them around in the search space using a simple mathematical formula based on their position and velocity.

2. Sequence Diagram

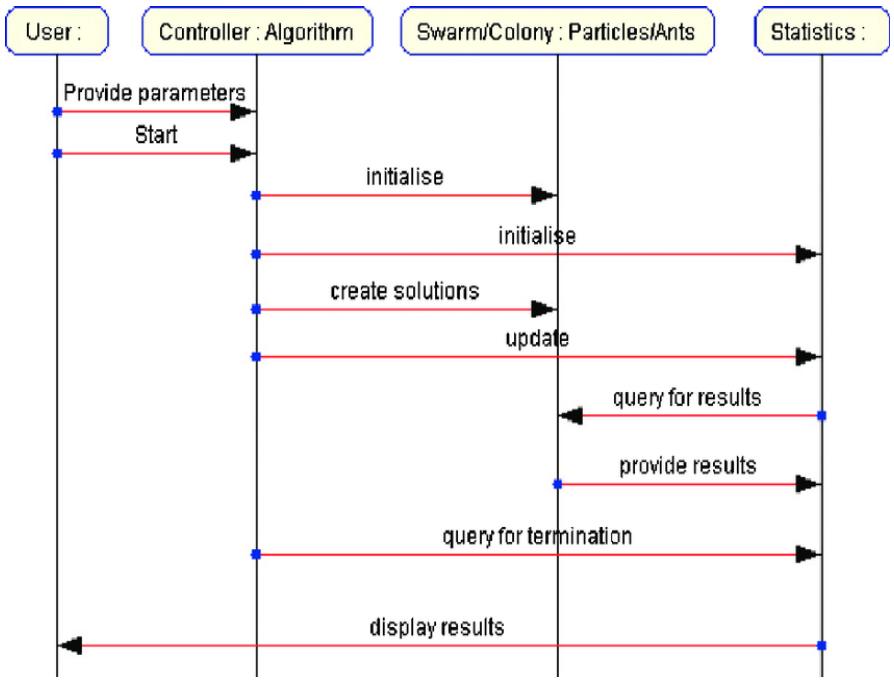

Diagram 2

Let's take it a step forward and explain what's stated above on the Sequence diagram. As previously said, the user only needs to run the application and set up the swarms. After the user has sent a request to From the controller, program. The request is delivered to the software, and the logic is begun to be applied. After each iteration, the user is notified, and the final result is shown. As previously stated, the idea underpinning particle swarm optimization will handle the majority of the activities.

3. Class Diagram

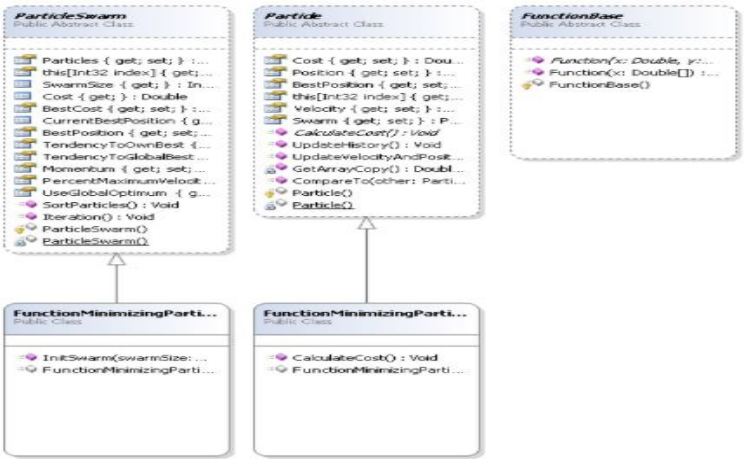

Diagram 3

A class diagram will be discussed here. Important sections of our software, together with their classes and methods, have been presented in the figure above. There is an algorithm operator in

the code that will manage all operations, as you can see. Particle is another essential class. Let's speak about the operations that our class diagram has depicted. One of the most significant parts of this optimization process, as we've already mentioned, is FunctionMinimizingParticle, which is depicted in this class diagram. As may be seen in the class diagram.

Implementation

```
Initialize the controlling parameters (N, c1, c2, Wmin, Wmax, Vmax, and MaxIter)

Initialize the population of N particles

do
    for each particle
        calculate the objective of the particle
        Update PBEST if required
        Update GBEST if required
    end for

    Update the inertia weight
    for each particle
        Update the velocity (V)
        Update the position (X)
    end for
white the end condition is not satisfied

Return GBEST as the best estimation of the global optimum
```

After watching a lot of Indian guys' PSO algorithm implementations, we finally wrote our algorithm. First, we implemented our libraries such as random, NumPy, and matplotlib libraries. And then, after receiving three inputs from our user, W, which means inertia factor, C1 self-confidence, and C2 swarm confidence, And we have some constant values: the number of iterations, particles, and our target error. Following that, we have two classes: one for space and the other for particles. In the Particles class, we use one function to update the position of our particle. The "init" method roughly represents a constructor in Python. And the self variable represents the instance of the object itself. After implementing our iterations and our algorithm, we have a particle class. Which has the fit function for knowing positions— Set_Pbest for setting the particle's best value. And if particle best is better than the best candidate, we will change the best position to particle best. And, of course, we have to update our particles. We also update velocity. We are finally showing particle function. I think it is understandable since it is the name). Finally, by carrying out our iterations, we created space and particle vectors. After setting the particle and global best, we visualized them. Checking conditions Then print the solutions and interations.

Conclusion

The most exciting feature of PSO is that it has a stable topology, allowing particles to communicate with one another and improve their learning rate in order to achieve a global optimum. Because it optimizes a problem by iteratively trying to improve a potential solution, the metaheuristic nature of this optimization method offers up a number of options. Its usefulness will expand much more as a result of the present Ensemble Learning work.

References:

https://analyticsindiamag.com/a-tutorial-on-particle-swarm-optimization-in-python/

https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

https://pyswarms.readthedocs.io/en/latest/

https://nathanrooy.github.io/posts/2016-08-17/simple-particle-swarm-optimization-with-python/

https://towardsdatascience.com/particle-swarm-optimization-visually-explained-46289eeb2e14

https://www.geeksforgeeks.org/implementation-of-particle-swarm-optimization/

https://www.researchgate.net/publication/3903911_Particle_swarm_optimization_Development_applications_and_resources

https://www.researchgate.net/publication/3810335_Empirical_Study_of_Particle_Swarm_Optimization

https://www.hindawi.com/journals/mpe/2021/5574501/