Particle Swarm Optimization

Particle swarm optimization (PSO) has been successfully applied to wide variety of problems in both of scientific and industrial fields alongside with other swarm intelligence based techniques. It undergoes beneath the umbrella of meta-heuristics where the global solution is not guaranteed to be found for some class of problems , but optimal or local solution can be easily found with few assumptions and lower computation resources. Also, PSO is stochastic in its nature which means the solution is not going to be the same for each time the algorithm gets executed ; however, based on the bench-marking and some handful testing, the final result fall contained within small region inside the solution space.

PSO proved to be so efficient in terms of computational time and memory compared to other methods. It also could be parallelized, but it is out of this project's scope. The downside, however, is that PSO has some hyper-parameters that have to be tuned to get reasonable solution within small margin of error. This section will provide a mathematical and a computational model of PSO.

0.1 Introduction

To understand the basic idea behind PSO, we have to know its origin. This stochastic method was proposed by Dr. Eberhart and Dr. Kennedy in 1995 inspired by the social behavior of birds or groups of fish.

Imagine that there is a group of birds trying to find food which is located only in one spot, none of the birds know exactly where the food is ; however, they have an idea of how far away might the food be. So, they have to cooperate intelligently to find the food spot.

In the same way, we have a bunch of particles (population), each one represents

a potential solution. None of the particle knows where is the global minima of the objective function, but each particle has memory of best visited location based on the value of the cost (objective) function. Within each generation (new population), the best known position to the population is updated for each iteration based on the best fitness evaluated among this particular population.

0.2 Mathematical Model of PSO

Each particle has its own position vector $\vec{x_i}(t)$ and a velocity vector $\vec{v_i}(t)$ to update the position each iteration along side with the particle's best known position $\vec{P_i}(t)$ and population's best know position $\vec{g}(t)$. This can be modeled with simple node has the aforementioned vectors and and another node representing the new state of the particle after one iteration.

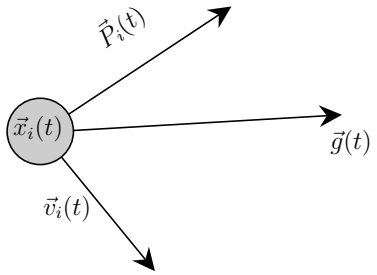


Figure 1: Initial state of particle at point of time t

Each iteration, the position of the particle has to be updated, to do so, the velocity vector must be updated first.

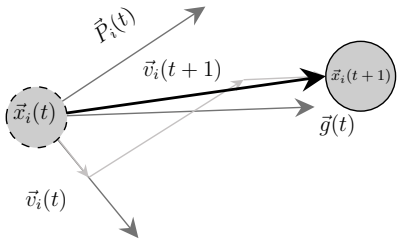


Figure 2: Initial state of particle at point of time t

As the figure shows, the velocity of each particle with a population can be updated as follows:

$$\vec{v_i}(t+1) = w.\vec{v_i}(t) + r_1.c_1.[\vec{P_i}(t) - \vec{x_i}(t)] + r_2.c_2.[\vec{g}(t) - \vec{x_i}(t)]$$

Where:

- w is the inertia weight (a scalar multiple of the velocity vector)
- r_1 and r_2 are uniformly random numbers between 0 and 1 $\rightarrow U[0,1]$
- c_1 is cognitive weight (how the particle tends to be positioned towards the its best location)
- c_2 is social weight (how the particle tends to be positioned towards the global best location)

As we have a finite search space, the velocity vector has to fall within a range:

$$\vec{v}_i(t+1) \in [\vec{v}_{min}\ ,\ \vec{v}_{max}]$$

If the condition does not prevail, the velocity is set to the extreme value it has surpassed weather minimum or maximum.
After updating the velocity, the position vector can be update as follows :

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$$

Following the last step, the best known position for the i^{th} particle can be improved if the the following condition holds :

$$f(\vec{x}_i(t+1)) < f(\vec{P}_i(t)) \implies \vec{P}_i(t+1) = \vec{x}_i(t+1)$$

The same way applies for the global best known position:

$$f(\vec{P}_i(t+1)) < f(\vec{g}(t)) \implies \vec{g}(t+1) = \vec{P}_i(t+1)$$

Surely, this is a simplified yet descriptive explanation of PSO; however, some additional features can be added that can assist in fast convergence to solution like simulating the death of a particle , which can be described by :

$$\vec{x}_i(t+1) \notin [x_{max}\ ,\ x_{min}] \implies \vec{x}_i(t+1) = rand([x_{max}\ ,\ x_{min}])$$

Fairly simple but still powerful to consider.

0.3 PSO PseudoCode

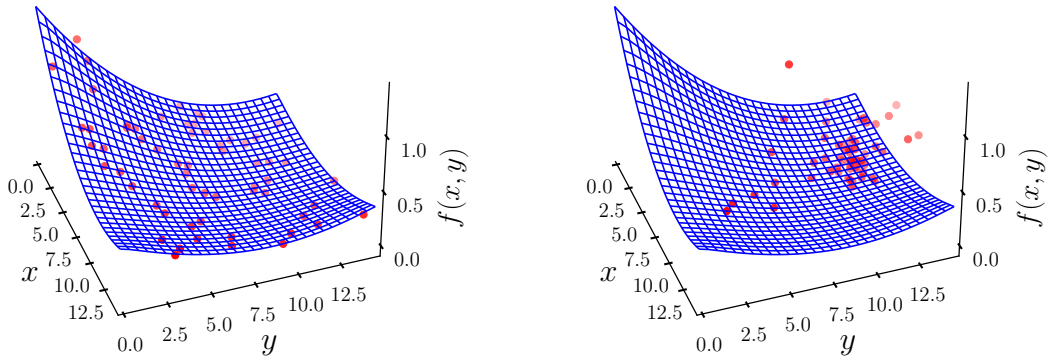
The pseudo-code of this algorithm is quite easy, we just have to compile the equations stated above, all together yield :

Randomly initialize Swarm population of N particles $\vec{x}_i, 1 \leq i \leq N$
Select hyperparameter values w, c_1 and c_2
for $k \leftarrow 1$ to *generations* **do**
  for $i \leftarrow 1$ to N **do**
    a. Compute new velocity of i^{th} particle
    $\vec{v_i}(t+1) = w.\vec{v_i}(t) + r_1.c_1.[\vec{P_i}(t) - \vec{x_i}(t)] + r_2.c_2.[\vec{g}(t) - \vec{x_i}(t)]$
    b. Check new velocity bound
    if $\vec{v}_i(t+1) < \vec{v}_{min}$ **then**
      $\vec{v}_i(t+1) = \vec{v}_{min}$
    else if $\vec{v}_i(t+1) > \vec{v}_{max}$ **then**
      $\vec{v}_i(t+1) = \vec{v}_{max}$
    end if
    c. Update the position of i^{th} particle
    $\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$
    d. Update new best of this particle and new best of Swarm
    if $f(\vec{x}_i(t+1)) < f(\vec{P}_i(t))$ **then**
      $\vec{P}_i(t+1) = \vec{x}_i(t+1)$
    end if
    if $f(\vec{P}_i(t+1)) < f(\vec{g}(t))$ **then**
      $\vec{g}(t+1) = \vec{P}_i(t+1)$
    end if
  end for
end for

0.4 PSO Implementation

PSO can be easily implemented in python relying on OOP paradigm. It is very suitable for the purpose and and the algorithm can be further extended to adopt more techniques. I have implemented a particle class which contains the particle's position, velocity, fitness, best position , and best fitness. That is besides a solver

class to properly transfer the data between different routines and export the necessary data for visualization. The following two figures are snapshots from the animation. In the first sub figure, the particle positions had been assigned randomly; after some iterations (generations), the particles tended to collaborate until they reached the minima of the objective function as clearly shown in the second sub-figure.



(a) Intial state of particles

(b) Final state of particles

Figure 3: PSO Animation