Particle Swarm intelligence (PSO)

The PSO algorithm is population based optimization algorithm which can be used to solve such problems. It is a direct search (non-gradient) algorithm where a population of particles "search" in parallel for the minimum of a given function in a multi-dimensional (*n*-dimensional) space (or region/domain)without using gradient information. Below we describe the basic PSO iteration.

Basic PSO Iteration

In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself (i. e. its own experience) and the position of the best particle in its neighborhood (i. e. the experience of neighboring particles). When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as a *gbest* PSO. When smaller neighborhoods are used, the algorithm is generally referred to as a *lbest* PSO. The performance of each particle (i. e. how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem. Each particle in the swarm is represented by the following characteristics:

$\boldsymbol{x}_i$: The *current position* of the particle;

$\boldsymbol{v}_{i:}$ The *current velocity* of the particle;

$\boldsymbol{y}_i$: The *personal best position* of the particle.

$\hat{\boldsymbol{y}}_i$**:** The *neighborhood best position* of the particle.

The personal best position of particle *i* is the best position (i. e. the one resulting in the best fitness value) visited by particle *i* so far. Let *f* denote the objective function. Then the personal best of a particle at time step *t* is updated as

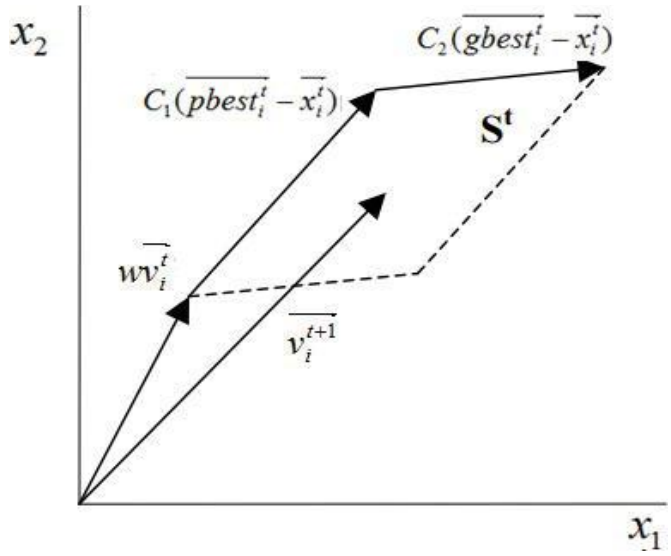
$$v_t(t+1) = \begin{cases} y_t(t) & \text{if } f(x_i(t+1) \geq f(y_t(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_t(t)) \end{cases} .$$

For the *best* model, the best particle is determined from the entire swarm by selecting the best personal best position. If the position of the global best particle is denoted by the vector $\boldsymbol{y}^{\wedge}$, then

$$y^{\wedge}(t) \in \{y_0, y_1, \ldots\ldots y_s\} = \min\{f(y_0(t)), f(y_1(t)), \ldots\ldots f(y_s))\} \ldots\ldots\ldots\ldots\ldots\ldots (3.26)$$

Where *s* denotes the size of the swarm. The velocity update step is specified for each dimension $j \in 1,\ldots,N_d$, hence, $v_{i,j}$ represents the j_{th} element of the velocity vector of the i_{th} particle. Thus the velocity of particle *i* is updated using the following equation *j* . there are three components, namely :

$$v_i(t+1) = v_i(t) + c_1 * rand() * (p_i(t) - x_i(t)) + c_2 * rand()(g_i(t) - x_i(t)) \ldots\ldots\ldots (3.29)$$



• The *inertia weight* term, *w* . This term serves as a memory of previous velocities. The inertia weight controls the impact of the previous velocity: a large inertia weight favors exploration, while a small inertia weight favors exploitation

• The *cognitive component*, $y_i(t) - x_i$, which represents the particle's own experience as to where the best solution.

• The *social component*, $y_i\hat{}(t) - x(t)$, which represents the belief of the entire swarm as to where the best solution is the relation between the inertia weight and acceleration constants should satisfy the following equation in order to have guaranteed convergence:

$$\frac{c_1 + c_2}{2} - 1 < w$$

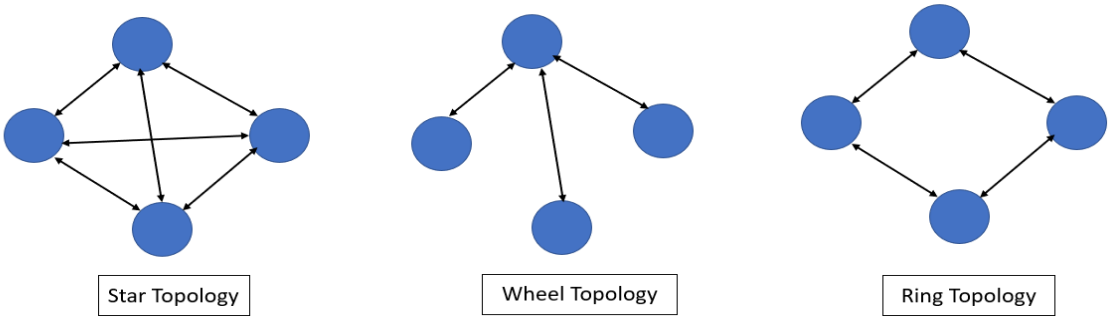
Otherwise, the PSO particles may exhibit divergent of cyclic behavior. Velocity updates can also be clamped with a user defined maximum velocity, V_{max}, which would prevent them from exploding, thereby causing premature convergence. The position of particle *i*, x_i, is then updated using equation:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

The PSO updates the particles in the swarm. This process is repeated until a specified number of iterations is exceeded, or velocity updates are close to zero. The quality of particles is measured using a fitness function which reflects the optimality of a particular solution.

• PSO Neighborhood Topologies

A neighborhood must be defined for each particle. This neighborhood determines the extent of social interaction within the swarm and influences a particular particle's movement. Less interaction occurs when the neighborhoods in the swarm are small. For small neighborhoods, the convergence will be slower, but it may improve the quality of solutions. The convergence will be faster for more prominent neighborhoods, but the risk that sometimes convergence occurs earlier



- For Star topology, each particle is connected with other particles. It leads to faster convergence than other topologies, Easy to find out gbest. But it can be biased to the pbest.
- For wheel topology, only one particle connects to the others, and all information is communicated through this particle. This focal particle compares the best performance of all particles in the swarm, and adjusts its position towards the best performance particle. Then the new position of the focal particle is informed to all the particles.
- For Ring Topology, when one particle finds the best result, it will make pass it to its immediate neighbors, and these two immediate neighbors pass it to their immediate neighbors until it reaches the last particle.

PSO algorithm

Lets us assume a few parameters first. You will find some new parameters, ***f:*** Objective function, ***Vi***: Velocity of the particle or agent, A: Population of agents, ***C1***: cognitive constant, ***C2***: social constant, ***Xi:*** Position of the particle or agent, P_b**:** Personal Best, g_b: global Best

The PSO algorithm can be summarize in the following steps**:**

Step 1: Initialize the particle position by assigning location

$p = (p_0, p_1 \ldots\ldots, p_N)$.

Initial weight (w_1, w_2) and velocities .

$$v = (v_0, v_1, \ldots, v_N).$$

Determine the fitness value of all the particles:

$$f(p) = (f(p_0), f(p_1), \ldots., f(p_N)).$$

Step 2: Evaluate the location where each individual has the highest fitness value so far:

$$p = (p_0^{best}, p_1^{best}, \ldots, p_N^{best}).$$

Step 3: Evaluate global fitness value which is best of all $p^{best:}$

$$G(p) = \max(f(p)).$$

Step 4: The particle velocity is updated based on the p^{best} and g^{best}.

$$v_i(t+1) = v_i(t) + c_1 * rand() * (p_i(t) - x_i(t)) + c_2 * rand()(g_i(t) - x_i(t)) \quad \ldots\ldots\ldots(3.29)$$

where the i value (1<i<N), c1 and c2 are constants known as acceleration coefficients and rand () are two separately generated uniformly distributed random numbers in the range [0, 1].

Step 5: Update the particle location by

$$p_i(t+1) = p_i + v_i(t+1) \quad for\ i < i \leq N \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.30)$$

Step 6: Terminate if maximum number of iterations is attained or minimum error criteria is met.

Step 7: Go to step 2

Gray Wolf Optimization

Grey Wolf Optimization (GWO) algorithm and how it works is a nature-inspired metaheuristic that is based on the behavior of a pack of wolves. The inspiration for this algorithm is the behavior of the grey wolf, which hunts large prey in packs and relies on cooperation among individual wolves. There are two interesting aspects of this behavior:

- social hierarchy
- hunting mechanism

The grey wolf is a highly social animal with the result that it has a complex social hierarchy. This hierarchical system, where wolves are ranked according to strength and power, is called "dominance hierarchies". Therefore, there are the alphas, betas, deltas, and omegas.

1. The alpha male and females are at the top of the hierarchy, and they lead the pack. All members of the pack have ordered within a specific rank. The wolf's hierarchical system is not just about dominance and aggression; it also provides assistance to vulnerable members of the pack who cannot hunt for themselves.
2. Afterward is the beta wolf that supports the alpha wolf's decisions and helps keep discipline within the pack.
3. The delta wolf is below the beta wolf in rank. They are often strong but lack leadership skills or confidence in themselves to take on leadership responsibilities.
4. And last, the omega wolf does not have any power at all and other wolves will quickly chase him. Omega wolf is also responsible for watching over younger wolves.

Social hierarchy in wolf pack

Besides social hierarchy, grey wolves have a very specific way of hunting with a unique strategy. They hunt in packs and work together in groups to separate the prey from the herd, then one or two wolves will chase after and attack the prey while the others chase off any stragglers. Describe the hunting strategy of the wolf pack and it includes:

- approach, track and chase the prey
- pursuit, harass, and encircling maneuver around the prey until it stops moving
- attack the prey when it is exhausted.

At each step, we'll denote respectively the three best solutions by alpha, beta, and delta, and other solutions by omega. Basically, it means the optimization process goes with the flow of the three best solutions. Also, the prey will be the optimal solution of the optimization.

Most of the logic follows the equations:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \qquad (1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \qquad (2)$$

where t denotes current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\overrightarrow{X_p}$ is the position vector of the prey and $\vec{X}$ is the position of wolf . Vectors $\vec{A}$ and $\vec{C}$ are equal to:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \qquad (3)$$

$$\vec{C} = 2\vec{r}_2, \qquad (4)$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 through iterations and $\vec{r}1$, $\overrightarrow{r2}$ are random vectors with values from [0 1], calculated for each wolf at each iteration.

Vector $\vec{A}$ controls the trade-off between exploration and exploitation while $\vec{C}$ always adds some degree of randomness. This is necessary because our agents can get stuck in the local optima and most of the metaheuristics have a way of avoiding it.

Since we don't know the real position of the optimal solution, $\overrightarrow{X_p}$ depends on the 3 best solutions and the formulas for updating each of the agents (wolfs) are:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \qquad (5)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot, \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot, \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \qquad (6)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}, \qquad (7)$$

where $\vec{X}$ is the current position of the agent and $\vec{X}(t+1)$ is the updated one. The formula above indicates that the position of the wolf will be updated accordingly the to best three wolves from the previous iteration. Notice that it won't be exactly equal to the average of the three best wolves

but because of the vector , it adds a small random shift. This makes sense because, from one side, we want that our agents are guided by the best individuals and from another side, we don't want to get stuck in the local optima.

Finally, the pseudocode of GWO is:

Algorithm 1: Grey Wolf Optimizer

```
Initialize the grey wolf population X_i, i = 1,n
Initialize a, A and C
Calculate the fitness of each search agent
X_α = the best search agent
X_β = the second best search agent
X_δ = the third best search agent
while t < max number of iteration do
    for each search agent do
        Randomly initialize r_1 and r_2
        Update the position of the current search agent by the
         equation (7)
    Update a, A and C
    Calculate the fitness of all search agents
    Update X_α, X_β and X_δ
    t = t + 1
return X_α
```