

Particle Swarm Optimization

EEE882 - Evolutionary Computation

Frederico Gadelha Guimarães¹

¹ **Machine Intelligence and Data Science (MINDS) Lab**
Department of Electrical Engineering
Universidade Federal de Minas Gerais

9 mai 2019

Summary

- 1 Swarm Intelligence
- 2 Basic PSO
- 3 Social Network Structures
- 4 Basic Variations
- 5 Hyper-parameters
- 6 Binary PSO

Emergent Complexity

Emergent Complexity : *“a phenomenon whereby larger entities arise through interactions among smaller or simpler entities”*

Necessary ingredients :

- Fundamental units with simple behavior ;
- An external force for cooperation ;

Implication :

- Complex behavior resulting from an optimization process

Swarm Intelligence

Unfortunately : complexity \nRightarrow usefulness

However : *natural* complexity \Rightarrow usefulness ?

- Complexity in nature is quite expensive.
- Complex systems arise in nature only if they are useful and efficient.

Swarm intelligence : useful applications of emergent complexity

Design goals

- Accurately simulate the behavior of a large flock of birds.
- Emergent complexity – keep it simple, stupid.
- $O(1)$ local behavior – high distributivity

A simple flocking model

- Simple model of bird-like objects (call them *boids*).
- Each boid is defined by a position and a velocity vector.
- Boids have total awareness of events in some local neighborhood (and cannot see outside of that).

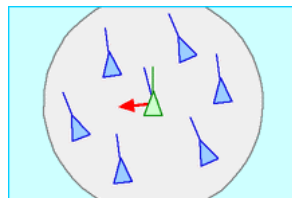


FIGURE 1 – A flock of boids

A simple flocking model

Rule 1 : Separation

- **Motivation** : Boids should avoid collisions with those around them.
- **Mechanism** : at each timestep, the boids will steer away from anyone in their personal space.

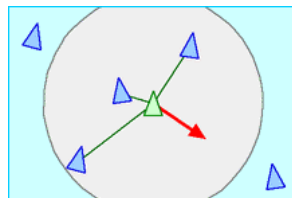


FIGURE 2 – The separation behavior

A simple flocking model

Rule 2 : Alignment

- **Motivation** : Boids should steer towards the direction of those around them.
- **Mechanism** : at each timestep, the boids will correct their alignment by some factor towards the average alignment of the boids in their vicinity.

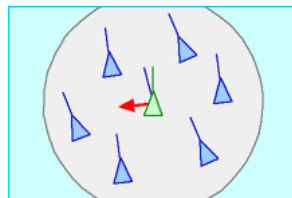


FIGURE 3 – The alignment behavior

A simple flocking model

Rule 3 : Cohesion

- **Motivation** : Boids should move alongside their neighbors.
- **Mechanism** : at each timestep, add a direction component towards the *center of gravity* of the boids in the local neighborhood.

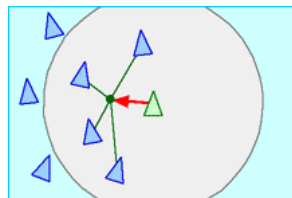


FIGURE 4 – The cohesion behavior

Particle Swarm Optimization

- Particle Swarm Optimization (PSO) is a nature-inspired evolutionary and stochastic optimization method to solve computationally hard optimization problems.
- PSO is based on the movement and intelligence of swarms.
- Individuals follow a very simple behavior : to emulate the success of neighboring individuals.
- It was developed in 1995 by James Kennedy and Russel C. Eberhart.

Particle Swarm Optimization

Terminology and notation

Swarm : Population or set of candidate solutions $\mathbf{x}_i(t)$, $i = 1, \dots, N_s$

Particle : Individual or candidate solution or potential solution to a problem

Let $\mathbf{x}_i(t)$ denote the position of particle i in the search space at time step t :

$$\mathbf{x}_i(t) = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{id} \end{pmatrix}$$

with $\mathbf{x}_i(0) \sim U(\mathbf{x}_{\min}, \mathbf{x}_{\max})$.

Particle Swarm Optimization

General update equation

The position of each particle is changed at every iteration by adding an update :

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

The position update depends on three simple rules :

Rule-1 Inertia : Particles tend to move in the previous direction.

Rule-2 Memory : Particles tend to move towards the best point found in its individual experience (personal best position).

Rule-3 Cooperation : Particles tend to move towards the best point found by the group or neighborhood (group best position).

Global Best PSO

For the global best PSO, or simply **gbest PSO**, the neighborhood of each particle is **the entire swarm**. The update equation is given by :

$$r_{1j}(t), r_{2j}(t) \sim U(0, 1) \quad (2)$$

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t) [p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t) [g_j(t) - x_{ij}(t)] \quad (3)$$

$i = 1, \dots, N_s, j = 1, \dots, d$. Or :

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2 \mathbf{r}_2(t) \circ [\mathbf{g}(t) - \mathbf{x}_i(t)] \quad (4)$$

$i = 1, \dots, N_s$.

- \circ is the Hadamard product or the entry-wise product.
- Constants c_1 and c_2 are positive constants used to scale the contribution of the cognitive and social components respectively.
- \mathbf{p}_i is the best position the particle has visited since the first time step.
- \mathbf{g} is the global best position.

Global Best PSO

Algorithm 1: gbest PSO

Data: Cost function $f(\cdot)$, swarm size N_s , \mathbf{x}_{\min} , \mathbf{x}_{\max}

```

1 Create and initialize a  $d$ -dimensional swarm  $\{\mathbf{x}_1(t), \dots, \mathbf{x}_{N_s}(t)\}$ ;
2 while stopping condition is false do
3   foreach particle  $i = 1, \dots, N_s$  do
4     Evaluate  $f(\mathbf{x}_i)$ ;
4     /* Set the personal best position */
5     if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then  $\mathbf{p}_i(t) = \mathbf{x}_i(t)$  ;
6   end
6   /* Set the global best position */
7    $\mathbf{g}(t) = \text{find } \mathbf{p}_k \in \{\mathbf{p}_1, \dots, \mathbf{p}_{N_s}\} \text{ corresponding to } \min f(\mathbf{p}_k)$ ;
8   foreach particle  $i = 1, \dots, N_s$  do
9      $\mathbf{r}_1(t), \mathbf{r}_2(t) \sim U(0, 1)$ ;
10     $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2 \mathbf{r}_2(t) \circ [\mathbf{g}(t) - \mathbf{x}_i(t)]$ ;
11     $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$ ;
12  end
13 end

```

Global Best PSO

Previous velocity : $\mathbf{v}_i(t)$ serves as a memory of the previous direction, which can be seen as a momentum, preventing the particle from drastically changing direction. Also referred to as the inertia component.

Cognitive component : $c_1 \mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)]$. This is an individual memory of particle's past performance. The cognitive component makes the particles move back to their own best positions.

Social component : $c_2 \mathbf{r}_2(t) \circ [\mathbf{g}(t) - \mathbf{x}_i(t)]$. This quantifies the performance of particle i relative to a group of particles. It resembles a group norm or standard that individuals seek to attain. Each particle is also drawn towards the best position found in the group.

Local Best PSO

For the local best PSO, or simply **lbest PSO**, the neighborhood of each particle is defined by a **ring social network topology**. The social component reflects information within the neighborhood of the particle, which has local knowledge of the environment. The update equation is given by :

$$r_{1j}(t), r_{2j}(t) \sim U(0, 1) \quad (5)$$

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t) [p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t) [g_{ij}(t) - x_{ij}(t)] \quad (6)$$

$i = 1, \dots, N_s, j = 1, \dots, d$. Or :

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2 \mathbf{r}_2(t) \circ [\mathbf{g}_i(t) - \mathbf{x}_i(t)] \quad (7)$$

$i = 1, \dots, N_s$.

- \circ is the Hadamard product or the entry-wise product.
- Constants c_1 and c_2 are positive constants used to scale the contribution of the cognitive and social components respectively.
- \mathbf{p}_i is the best position the particle has visited since the first time step.
- \mathbf{g}_i is the best position found in the neighborhood of particle i .

Local Best PSO

Algorithm 2: lbest PSO

Data: Cost function $f(\cdot)$, swarm size N_s , \mathbf{x}_{\min} , \mathbf{x}_{\max}

```

1 Create and initialize a  $d$ -dimensional swarm  $\{\mathbf{x}_1(t), \dots, \mathbf{x}_{N_s}(t)\}$ ;
2 while stopping condition is false do
3   foreach particle  $i = 1, \dots, N_s$  do
4     Evaluate  $f(\mathbf{x}_i)$ ;
4     /* Set the personal best position */
5     if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then  $\mathbf{p}_i(t) = \mathbf{x}_i(t)$ ;
5     /* Set the group best position */
6      $\mathbf{g}_i(t) = \text{find } \{\mathbf{p}_k \in \mathcal{N}_i\} \text{ corresponding to } \min f(\mathbf{p}_k)$ ;
7   end
8   foreach particle  $i = 1, \dots, N_s$  do
9      $\mathbf{r}_1(t), \mathbf{r}_2(t) \sim U(0, 1)$ ;
10     $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2 \mathbf{r}_2(t) \circ [\mathbf{g}_i(t) - \mathbf{x}_i(t)]$ ;
11     $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$ ;
12  end
13 end

```

Local Best PSO

The best position found in the neighborhood \mathcal{N}_i is defined as :

$$\mathbf{g}_i(t) = \text{find } \{\mathbf{p}_k \in \mathcal{N}_i\} \text{ corresponding to } \min f(\mathbf{p}_k) \quad (8)$$

The neighborhood using ring social network topology is given by :

$$\mathcal{N}_i = \{\mathbf{p}_{\text{mod}(i-1, N_s)}, \mathbf{p}_i, \mathbf{p}_{\text{mod}(i+1, N_s)}\} \quad (9)$$

that is, the neighborhood is defined based on particle indices.

- Neighborhoods based on indices are computationally inexpensive, not requiring the calculation of distances ;
- It helps to promote the spread of information regarding good particles to all particles, regardless of their location ;
- The gbest PSO is a special case of lbest PSO, when \mathcal{N}_i is equal to the entire swarm.

gbest versus lbest PSO

The two basic versions of PSO are similar in the sense that the social component points towards the global best. The main differences are :

- Due to the complete connectivity of gbest PSO, **it converges faster** than the lbest PSO ;
- However, the fast convergence of gbest PSO comes at the cost of **rapid decrease of diversity** ;
- lbest PSO is less susceptible to being trapped in local minima, hence improving global convergence.

Geometric interpretation

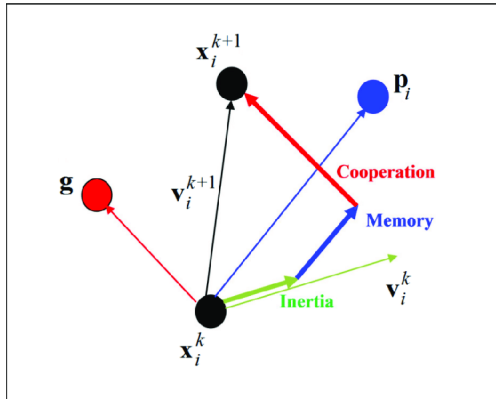


FIGURE 5 – Geometrical illustration of velocity and position updates.

Algorithm details

Initialization

- Initialize the hyper-parameters c_1 and c_2 (see more details ahead)
- Initial positions of the particles are defined at random, i.e.

$$\mathbf{x}_i(t = 0) \sim U(\mathbf{x}_{\min}, \mathbf{x}_{\max})$$

- Initial velocities can be set equal to zero :

$$\mathbf{v}_i(t = 0) = \mathbf{0}$$

- Although it is possible to start the velocities to random values, it is not necessary and it must be done with care, avoiding large initial velocities.
- The personal best position of each particle is equal to its initial position :

$$\mathbf{p}_i(t = 0) = \mathbf{x}_i(0)$$

Algorithm details

The following stopping conditions have been used :

- Terminate when a maximum number of iterations or function evaluations has been achieved.
- Terminate when an acceptable solution has been found, that is, when a particle \mathbf{x}_i is found such that $f(\mathbf{x}_i) \leq |f(\mathbf{x}^*) - \epsilon|$. Useful when the optimum value is known or when dealing with benchmark functions to validate the method.
- Terminate when no improvement is observed over a number of iterations. There are different ways in which improvement can be measured :
 - average change in particle positions is small ;
 - average particle velocity is close to zero ;
 - no significant update in the global best position.

This introduces two new hyper-parameters (window of iterations and threshold).

- Terminate when the volume of the swarm is close to zero. There are different ways in which the volume of the swarm can be defined.

Social network structures

Particles in the same neighborhood communicate with one another by exchanging local information. All particles then move (stochastically) towards some quantification of what is believed to be a better position.

- The information flow in the swarm affects the movement of the particles.
- **Therefore, the performance of PSO depends strongly on the structure of the social network (neighborhood structure).**

Social network structures

Different social network structures have been studied for PSO.

Complete graph : All particles are interconnected. gbest PSO is equivalent to using complete graph as social structure.

Ring : Each particle communicates with its n_R immediate neighbors. Typical case is with $n_R = 2$, which is the case of lbest PSO. Neighborhoods overlap, facilitating the flow of information and convergence to a single solution. Convergence is slower but diversity is promoted.

Clusters : Clusters of fully connected neighbors are formed with few connections between clusters.

Spatial : or distance-based neighborhood. In this case the neighborhood is dynamic and defined in terms of proximity between particles. Additional cost of $O(N_s^2)$ per iteration.

Other topologies : have been studied. In general, more connected structures perform better for unimodal problems while less connected structures are better for multimodal problems.

Social network structures

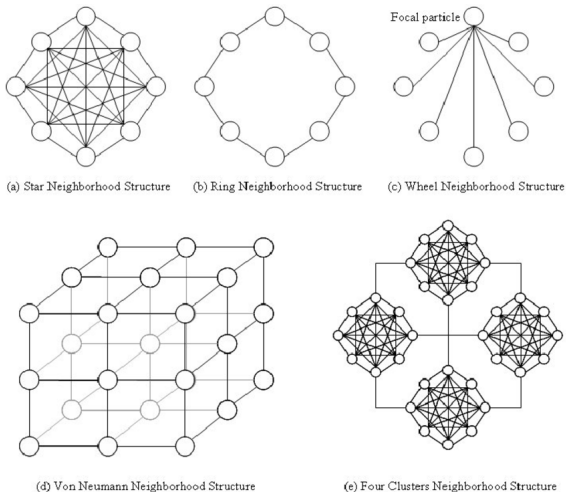


FIGURE 6 – Neighborhood structures commonly used in PSO.

Basic Variations

A number of modifications to the basic PSO have been developed to improve speed of convergence and the quality of solutions found by the PSO. These modifications include :

- velocity clamping ;
- the introduction of an inertia weight ;
- velocity constriction ;
- different ways of determining the personal best and global best (or local best) positions ;
- different velocity models.

Velocity clamping

- In the early applications of the basic PSO, it was found that the position update may quickly explode to large values, especially for particles far from the group best and personal best positions.
- Large position updates may cause the particles to leave the boundaries of the search space or oscillating trajectories.
- To control this, velocities are clamped to stay within boundary constraints (EBERHART ; SIMPSON ; DOBBINS, 1996).

Velocity clamping

If a particle's position update exceeds a specified maximum velocity it is limited to the maximum value :

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } v_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{if } v_{ij}(t+1) \geq V_{\max,j} \end{cases} \quad (10)$$

- Large values of $V_{\max,j}$ facilitate global exploration but increase oscillatory behavior, since the particle risks missing a good region or jumping over optimum.
- When $V_{\max,j}$ is too small, the swarm may not explore sufficiently and it may increase the number of steps to reach an optimum.

So, it depends on the problem :

$$V_{\max,j} = \delta \times (x_{\max,j} - x_{\min,j}) \quad (11)$$

with $\delta \in [0.2, 1.0]$.

Velocity clamping

Some important remarks about velocity clamping :

- Velocity clamping does not confine the positions of particles, only the step sizes as determined from the particle velocity.
- One might still need to use **reflection** if the particles leave the boundaries.
- The maximum value depends on the dimension j .
- Some ideas are available to decrease V_{\max} over time, see (FAN, 2002; SCHUTTE ; GROENWOLD, 2003).

Inertia weight PSO

The inertia weight PSO was introduced in (SHI ; EBERHART, 1998) to control the exploration and exploitation abilities of the swarm, and as a mechanism to eliminate the need for velocity clamping¹. For the gbest PSO the update equation is modified to :

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2\mathbf{r}_2(t) \circ [\mathbf{g}(t) - \mathbf{x}_i(t)] \quad (12)$$

$0 < w < 1, i = 1, \dots, N_s$.

- The inertia weight controls the momentum of the particle by weighing the contribution of the previous velocity.
- Large values for w (close to 1) facilitate exploration. Particles are more influenced by the previous direction.
- For small values for w (close to 0), particles are more influenced by the cognitive and social components in their position updates.
- For $w > 1$, velocities increase over time and the swarm diverges.
- For $w < 1$, particles reduce their velocities until convergence.

1. However, it could not completely eliminate the need for velocity clamping.

Inertia weight PSO

Approaches to dynamically adjusting the inertia weight have been proposed :

Random adjustments : a different inertia weight is randomly selected at each iteration, sampling from a Gaussian distribution, e.g.

$$w(t) \sim N(0.7, \sigma)$$

Linear decreasing : an initially large inertia weight (usually 0.9) is linearly decreased to a small value (usually 0.4).

$$w(t) = (w(0) - w(T)) \frac{T - t}{T} + w(T)$$

Inertia weight PSO

Nonlinear decreasing : Nonlinear decreasing methods allow a shorter exploration time than the linear decreasing methods, with more time spent on refining solutions (exploiting).

$$w(t) = \alpha w(t')$$

where $\alpha = 0.975$ and t' is the time step when the inertia weight last changed. The average fitness of the swarm is used to trigger variation.

Fuzzy adaptive : where the inertia weight is dynamically adjusted on the basis of fuzzy sets and rules.

if normalized best fitness is LOW, and current inertia weight value is LOW then the change in weight is MEDIUM

Constricted PSO

Clerc developed an approach very similar to the inertia weight to balance the exploration–exploitation trade-off, where the velocities are constricted by a **constriction coefficient** [133, 136]. The velocity update equation changes to :

$$\mathbf{v}_i(t+1) = \chi [\mathbf{v}_i(t) + \phi_1 (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \phi_2 (\mathbf{g}(t) - \mathbf{x}_i(t))] \quad (13)$$

where :

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi(\phi - 4)}|} \quad (14)$$

$\phi = \phi_1 + \phi_2$, $\phi_1 = c_1 r_1$, $\phi_2 = c_2 r_2$, $\phi > 4$, $\kappa \in [0, 1]$, $i = 1, \dots, N_s$.

Constricted PSO

The parameter κ controls the exploration and exploitation abilities of the swarm.

- For $\kappa \approx 0$, fast convergence is obtained with local exploitation. The swarm exhibits an almost hill-climbing behavior.
- For $\kappa \approx 1$, slow convergence is obtained with a high degree of exploration.

Usually, κ is set to a constant value. Typically :

$$\kappa = 1, \phi_1 = \phi_2 = 2.05 \rightarrow \chi \approx 0.73$$

However, an initial high degree of exploration with local exploitation in the later search phases can be achieved using an initial value close to one, then decreasing it to zero.

Constricted PSO

Constricted PSO (C-PSO) versus Inertia weight PSO (IW-PSO) :

- Both approaches have the goal of balancing exploration and exploitation, improving convergence time and the quality of solutions found.
- Low values of w and χ result in exploitation, while large values result in exploration.
- Velocity clamping is not necessary for the constriction model².
- The constriction model guarantees convergence under the given constraints.
- any ability to regulate the change in direction of particles must be done via the constants ϕ_1 and ϕ_2 for the constriction model.

A common model used in the literature is :

$$\mathbf{v}_i(t+1) = \chi [\mathbf{w}\mathbf{v}_i(t) + \phi_1 (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \phi_2 (\mathbf{g}(t) - \mathbf{x}_i(t))] \quad (15)$$

2. While it is not necessary to use velocity clamping with the constriction model, Eberhart and Shi showed empirically that if velocity clamping and constriction are used together, faster convergence rates can be obtained [226].

Synchronous versus Asynchronous Updates

The gbest and lbest PSO algorithms presented in Algorithms 1 and 2 perform synchronous updates of the personal best and global (or local) best positions.

- asynchronous PSO calculates the new best positions after each particle position update (very similar to a steady state GA).
- asynchronous updates are more important for lbest PSO where immediate feedback will be more beneficial in loosely connected swarms, while synchronous updates are more appropriate for gbest PSO [108].

Asynchronous Local Best PSO

Algorithm 3: Asynchronous lbest PSO

Data: Cost function $f(\cdot)$, swarm size N_s , \mathbf{x}_{\min} , \mathbf{x}_{\max}

```

1 Create and initialize a  $d$ -dimensional swarm  $\{\mathbf{x}_1(t), \dots, \mathbf{x}_{N_s}(t)\}$ ;
2 while stopping condition is false do
3     foreach particle  $i = 1, \dots, N_s$  do
4         Evaluate  $f(\mathbf{x}_i)$ ;
4         /* Set the personal best position */
5         if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then  $\mathbf{p}_i(t) = \mathbf{x}_i(t)$ ;
5         /* Set the group best position */
6          $\mathbf{g}_i(t) = \text{find } \{\mathbf{p}_k \in \mathcal{N}_i\} \text{ corresponding to } \min f(\mathbf{p}_k)$ ;
7          $r_1(t), r_2(t) \sim U(0, 1)$ ;
8          $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 r_1(t) [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2 r_2(t) [\mathbf{g}_i(t) - \mathbf{x}_i(t)]$ ;
9          $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$ ;
10    end
11 end

```

Selection-based PSO

- In gbest PSO, the global best is defined as :

$$\mathbf{g}(t) = \text{find } \mathbf{p}_k \in \{\mathbf{p}_1, \dots, \mathbf{p}_{N_s}\} \text{ corresponding to } \min f(\mathbf{p}_k) \quad (16)$$

- In lbest PSO, the local best position found in the neighborhood \mathcal{N}_i is defined as :

$$\mathbf{g}_i(t) = \text{find } \{\mathbf{p}_k \in \mathcal{N}_i\} \text{ corresponding to } \min f(\mathbf{p}_k) \quad (17)$$

One can select the best positions randomly (or biased random selection) from the neighborhood [448]. The random selection was specifically used to address the difficulties that the gbest PSO experience on highly multi-modal problems :

$$\mathbf{g}(t) = \text{stochastic selection of } \mathbf{p}_k \in \{\mathbf{p}_1, \dots, \mathbf{p}_{N_s}\} \quad (18)$$

$$\mathbf{g}_i(t) = \text{stochastic selection of } \mathbf{p}_k \in \mathcal{N}_i \quad (19)$$

Fully Informed PSO

In the standard PSO, each particle's new position is influenced by the particle itself (via its personal best position) and the best position in its neighborhood. In Fully Informed PSO (FI-PSO), each particle is influenced by the successes of all its neighbors :

$$\mathbf{v}_i(t+1) = \chi \left(\mathbf{v}_i(t) + \mathbf{r}(t) \sum_{m \in \mathcal{N}_i} \frac{[\mathbf{p}_m(t) - \mathbf{x}_i(t)]}{|\mathcal{N}_i|} \right) \quad (20)$$

\mathcal{N}_i is the set of particles in the neighborhood of particle i :

$$\mathcal{N}_i = \{ \mathbf{p}_{\text{mod}(i-N_i, N_S)}, \dots, \mathbf{p}_i, \dots, \mathbf{p}_{\text{mod}(i+N_i, N_S)} \} \quad (21)$$

and $\mathbf{r}(t) \sim U(0, c_1 + c_2)^n$.

Barebones PSO

Formal proofs [851, 863, 870] have shown that each particle converges to a point that is a weighted average between the personal best and neighborhood best positions. If it is assumed that $c_1 = c_2$, then a particle converges, in each dimension to :

$$\frac{p_{ij}(t) + g_{ij}(t)}{2} \quad (22)$$

Based on this, we can in principle replace the entire velocity equation by random numbers sampled from a Gaussian distribution as given by :

$$x_{ij}(t+1) \sim N\left(\frac{p_{ij}(t) + g_{ij}(t)}{2}, |p_{ij}(t) - g_{ij}(t)|\right) \quad (23)$$

One can use $\mathbf{g}_i(t)$ (local best position) or instead :

- the global best position (gbest PSO) ;
- a randomly selected neighbor ;
- or the center of best positions in a neighborhood (FI-PSO) ;

Life Cycle PSO

Life Cycle PSO. Krink and Løvberg [490, 534] used the life-cycle model to change the behavior of individuals.

- Using the life-cycle model, an individual can be in any of three phases : a PSO particle, a GA individual, or a stochastic hill-climber.
- Initial population starts as PSO particles.

Life Cycle PSO

Algorithm 4: Life Cycle PSO

```
1 Initialize a population of individuals;
2 while stopping condition is false do
3   foreach individual  $i = 1, \dots, N$  do
4     Evaluate fitness  $f(\mathbf{x}_i)$ ;
5     if fitness did not improve over last iterations then Switch to next phase ;
6   end
7   foreach PSO particle do
8     Calculate new velocity;
9     Update particle's position;
10  end
11  forall GA individuals do
12    Perform selection;
13    Apply crossover and mutation;
14  end
15  foreach LS profile do
16    Apply stochastic local search;
17  end
18 end
```

Hyper-parameters

The basic PSO is influenced by a number of control parameters, namely :

- number of particles : N_s
- acceleration coefficients c_1, c_2
- inertia weight w
- neighborhood size $|\mathcal{N}_i|$
- number of iterations T_{\max}
- and the random values that scale the contribution of the cognitive and social components $\mathbf{r}_1, \mathbf{r}_2$

Additionally, if velocity clamping or constriction is used, the maximum velocity and constriction coefficient also influence the performance of the PSO.

Hyper-parameters

- **Swarm size** : the more particles in the swarm, the larger the initial diversity of the swarm – provided that a good uniform initialization scheme is used to initialize the particles. However, this N_s function evaluations are required per iteration. It has been shown in a number of empirical studies that the PSO has the ability to find optimal solutions with small swarm sizes of 10 to 30 particles [89, 865]. The optimal swarm size is problem-dependent.
- **Neighborhood size** : The neighborhood size defines the extent of social interaction within the swarm. While smaller neighborhoods are slower in convergence, they have more reliable convergence to optimal solutions. Start the search with small neighborhoods and increase the neighborhood size proportionally to the increase in number of iterations [820].
- **Number of iterations** : The number of iterations to reach a good solution is also problem-dependent. Too few iterations may terminate the search prematurely. A too large number of iterations has the consequence of unnecessary added computational complexity (provided that the number of iterations is the only stopping condition).

Hyper-parameters

- **Acceleration coefficients** : The acceleration coefficients, c_1 and c_2 , together with the random vectors \mathbf{r}_1 and \mathbf{r}_2 , control the stochastic influence of the cognitive and social components.

$c_1 > 0, c_2 = 0$ (Cognitive-only model). All particles are independent hill-climbers. Each particle converges to the best position in its neighborhood.

$c_1 = 0, c_2 > 0$ (Social-only model). The entire swarm is attracted to a single point³. The swarm turns into one stochastic hill-climber.

$c_1 = c_2$ Particles are attracted towards the average of \mathbf{p}_i and \mathbf{g} (or \mathbf{g}_i).

Usual values

Usually c_1 and c_2 are static, with suggested values ranging in $[1, 2]$. Adaptive methods have been proposed in the literature (ENGELBRECHT, 2007; WANG; HAN, 2009).

3. Even in lbest PSO, eventually knowledge about gbest will propagate to all particles.

Hyper-parameters

- **Random vectors** : In basic PSO, the contributions of cognitive and social components are randomly combined :

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2\mathbf{r}_2(t) \circ [\mathbf{g}_i(t) - \mathbf{x}_i(t)] \quad (24)$$

If the same random number is used for all variables, PSO update equation becomes rotational invariant, which is a desirable property :

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + \underbrace{c_1 r_1(t)}_{\phi_1} [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + \underbrace{c_2 r_2(t)}_{\phi_2} [\mathbf{g}_i(t) - \mathbf{x}_i(t)]$$

Binary PSO

Although PSO was originally developed for continuous-valued search spaces, Kennedy and Eberhart developed the first discrete PSO to operate on binary search spaces (KENNEDY ; EBERHART, 1997; EBERHART ; SHI ; KENNEDY, 2001). Some adaptations need to be done :

- Particles represent solutions in the binary space

$$\mathbf{x}_i \in \mathbb{B}^n$$

- The update equation changes bits in the solution, leading to a new position in the binary space :

$$x_{ij}(t+1) = \begin{cases} 1, & \text{if } U(0, 1) \leq s(v_{ij}(t+1)) \\ 0, & \text{otherwise} \end{cases} \quad (25)$$

A sigmoid function is used to normalize the values of v_{ij} to the interval $[0, 1]$:

$$s(v_{ij}(t+1)) = \frac{1}{1 + e^{-v_{ij}(t+1)}} \quad (26)$$

Binary PSO

With this modification in the update equation, the velocity vectors are still real-valued, with the same velocity calculation as given by :


$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1(t) \circ [\mathbf{p}_i(t) - \mathbf{x}_i(t)] + c_2\mathbf{r}_2(t) \circ [\mathbf{g}_i(t) - \mathbf{x}_i(t)] \quad (27)$$


Therefore, $\mathbf{x}_i, \mathbf{p}_i, \mathbf{g}_i \in \mathbb{B}^n$ and $\mathbf{v}_i \in \mathbb{R}^n$.

$v_{ij} = 0$: the probability $P\{x_{ij}(t+1) = 1\} = 0.5$


$v_{ij} > 0$: the probability $P\{x_{ij}(t+1) = 1\} > 0.5$


$v_{ij} < 0$: the probability $P\{x_{ij}(t+1) = 1\} < 0.5$


 EBERHART, R. C. ; SHI, Y. ; KENNEDY, J. **Swarm Intelligence**. Burlington, MA, USA : Morgan Kaufmann, 2001.


 EBERHART, R. C. ; SIMPSON, P. K. ; DOBBINS, R. W. **Computational Intelligence PC Tools**. first edition. San Diego, CA, USA : Academic Press Professional, 1996.

 ENGELBRECHT, A. P. **Computational Intelligence : An Introduction**. 2nd. ed. USA : Wiley, 2007.

 FAN, H.-Y. A modification to particle swarm optimization algorithm. **Engineering Computations**, v. 19, n. 7-8, p. 970–989, 2002.

 KENNEDY, J. ; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In : **1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation**. [S.l.] : IEEE, 1997. v. 5, p. 4104–4108. ISSN 1062-922X.

 SCHUTTE, J. ; GROENWOLD, A. Sizing design of truss structures using particle swarms. **Structural and Multidisciplinary Optimization**, v. 25, n. 4, p. 261–269, 2003.

 SHI, Y. ; EBERHART, R. A modified particle swarm optimizer. In : **1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)**. Anchorage, USA : IEEE, 1998. p. 69–73.



WANG, G. ; HAN, D. Particle swarm optimization based on self-adaptive acceleration factors. In : **2009 Third International Conference on Genetic and Evolutionary Computing**. [S.l.] : IEEE, 2009. p. 637–640.