Particle Swarm Optimisation

Dr H.K. Lam

Department of Engineering
King's College London
Office S2.14, Strand Building, Strand Campus
Email: hak-keung.lam@kcl.ac.uk
https://nms.kcl.ac.uk/hk.lam

Nature-Inspired Learning Algorithms (7CCSMBIM)

Outline

1. Introduction

Assignment Project Exam Help

2. Basic Particle Swarm Optimisation
   - Global Best PSO
   - Local Best PSO
   - Velocity Components
   - Particle Initialisation
   - Stopping Criteria
   - Social Network Structure
   - Velocity Clamping
   - Inertia Weight

https://powcoder.com

Add WeChat powcoder

3. Examples

Learning Objectives

King's College London
University of London

- To get an idea of swarm intelligence.
- To get the concept of particle swarm optimisation and know how it works.
- To apply the particle swarm optimisation to optimisation problems.

Introduction

Swarm Intelligence

- Swarm intelligence is an artificial intelligence technique based on the study of collective behaviour in decentralised, self-organised systems.
- Swarm intelligence systems are typical made up of a population of simple agents interacting locally with one another and with their environment.
- Although there is no centralised control structure, local interactions between such agents often lead to the emergence of global behaviour.
- Examples of systems in nature, including social insect colonies, bird flocking, fish schooling and animal herding.

Self-organisation

- A set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower-level components.

Self-organisation

- A set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower-level components.

- **Four basic ingredients:**
  - **Positive feedback (amplification)**: To show the right of direction to the food source (optimal solution); to reinforce those portions of good solutions that contribute to the quality of these solutions.

  - **Negative feedback:** to introduce a time scale into the algorithm through pheromone evaporation, to prevent premature convergence (stagnation), for counter-balance and stabilisation.

  - **Amplification of fluctuation:** Randomness or errors, e.g., lost ant foragers can find new food sources. An element moves more randomly to search for a solution and then amplified by a positive feedback loop.
  - **Multiple interactions:** Direct or indirect communication (e.g., modification of the environment).

Why do Animals Swarm?

- Defence against predators.
  - Enhance the detection of predators.
  - Minimise the chance of being captured.
- Enhance success in foraging.
- Better chances to find a mate.
- Decrease of energy consumption.

KING'S College LONDON
University of London

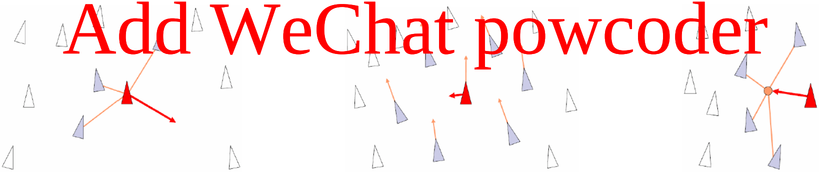
Coordinated Collective Behaviour

- Reynolds (1987) proposed a behaviour model to interpret bird flocking, fish schooling and animal herds.
- Biologically and physically sound assumption:
  1. Individual has only local knowledge.
  2. Has certain cognitive capabilities.
  3. Bound by laws of physics.
- Comply with only three simple rules.

KING'S College LONDON
University of London

Coordinated Collective Behaviour Reynolds (1987) Behaviour Model

1. **Separation**: each agent tries to move away from its neighbours if they are too close.
2. **Alignment**: each agent steers towards the averaging heading of its neighbours.
3. **Cohesion**: each agent tries to go towards the average position of its neighbours.



(a) Separation (b) Alignment (c) Cohesion

KING'S College LONDON
University of London

Building a Metaheuristic

- They expand the swarm behavioural model to n-dimensional psychosocial space without constraints of physical laws.
- A swarm consists of a set of particles, where each particle represents a potential solution for an n-dimensional optimisation problem.
- Particles are flown through the hyperspace, where position of each particle is changed according to its own experience and that of its neighbours.

Basic Particle Swarm Optimisation

KING'S College LONDON
University of London

- **Particle Swarm Optimisation (PSO):** A numerical population-based optimisation techniques discovers optimal regions of a high dimensional search space through collective behaviour of individuals.

Basic Particle Swarm Optimisation

- **Particle Swarm Optimisation (PSO):** A numerical population-based optimisation techniques discovers optimal regions of a high dimensional search space through collective behaviour of individuals.

- **Ingredients:**
  - *A particle*: an individual (a potential solution)
  - *A swarm*: a population.
  - *Update rules*.

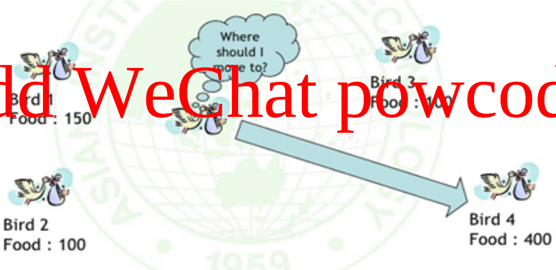


Image from the Internet

Image from the Internet

KING'S College LONDON
University of London

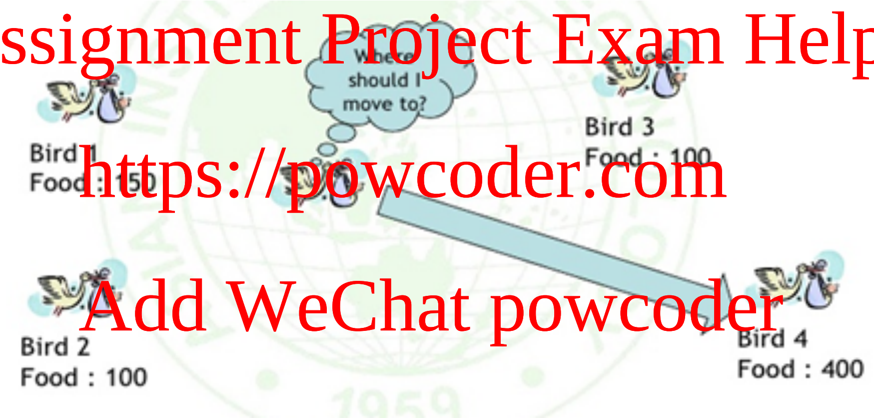
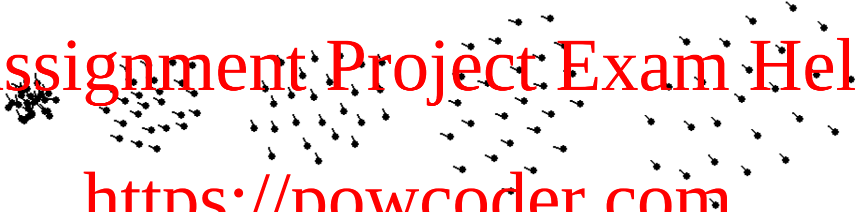


Image from the Internet



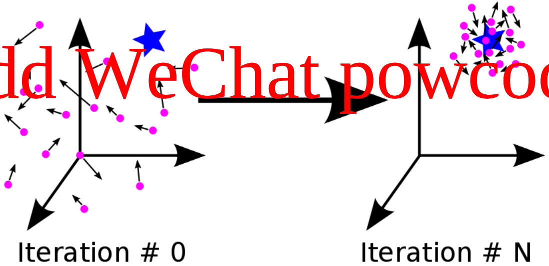


Image from the Internet

Notation

- $\mathbf{x}_i(t) = [x_{i1}, \cdots, x_{in_x}]$: the i^{th} particle (individual).
- n_x: number of elements in each particle.
- n_s: size of swarm (number of particles in the swarm).
- n_t: maximum number of iterations.
- $\hat{\mathbf{y}}(t) = [\hat{y}_1(t), \cdots, \hat{y}_{n_x}(t)]$: the global best position since the first generation.
- $\hat{\mathbf{y}}_i(t) = [\hat{y}_{i1}(t), \cdots, \hat{y}_{in_x}(t)]$: the local best position since the first generation.
- $\mathbf{y}_i(t) = [y_{i1}(t), \cdots, y_{in_x}(t)]$: the personal best position since the first generation.
- $\mathcal{N}_i$: the set of neighbourhoods of particle i.
- $\mathbf{x}_{\min} = [x_{1_{\min}}, \cdots, x_{n_{x\min}}]$: a vector of constants denoting the lower bound of $\mathbf{x}_i(t)$.
- $\mathbf{x}_{\max} = [x_{1_{\max}}, \cdots, x_{n_{x\max}}]$: a vector of constants denoting the upper bound of $\mathbf{x}_i(t)$.
- $\mathbf{v}_i = [v_{i1}, \cdots, v_{in_x}]$: a velocity vector.
- $r_{1j}(t), r_{2j}(t) \in [0, 1]$: a random number .

- Two basic PSO algorithms:
  - The main difference is the size of their neighbourhoods.
- **Global best PSO** (*gbest* PSO):
  - Social network: star topology.
  - Neighbours of a particle: the whole swarm.
  - Particles are updated based on the social information from all particles in the swarm.
  - Social information: the best position (solution) found by the swarm.
- **Local best PSO** (*lbest* PSO):
  - Social network: ring topology.
  - Neighbours of a particle: a small number of particles in the swarm
  - Particles are updated based on the social information exchanged within the neighbourhood of the particle.
  - Social information: the local best position (solution) within the neighbourhood.

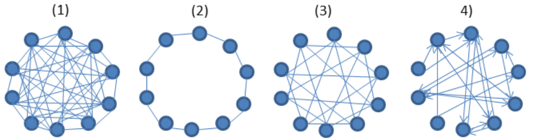


Image from the Internet

Graphical representation of (1) fully connected, (2) ring, (3) von Neumann and (4) random topology

Global Best PSO

KING'S College LONDON
University of London

- Size of Swarm: n_s particles.
- The (**global**) best position found in the swarm:

$$\hat{\mathbf{y}}(t) = [\hat{y}_1(t), \cdots, \hat{y}_{n_x}(t)]$$

- The personal best position since the first generation:

$$\mathbf{y}_i(t) = [y_{i1}(t), \cdots, y_{in_x}(t)]$$

- **Velocity update rule:**

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\big(y_{ij}(t) - x_{ij}(t)\big) + c_2 r_{2j}(t)\big(\hat{y}_j(t) - x_{ij}(t)\big) \quad (1)$$

  where $c_1 \geq 0$ and $c_2 \geq 0$ are acceleration constants; $r_{1j}(t)$ and $r_{2j}(t) \in [0, 1]$ are random numbers.

- **Particle update rule:**

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \; i = 1, \cdots, n_s$$

where $\mathbf{x}_i(t) = [x_{i1}, \cdots, x_{in_x}]$ and $\mathbf{v}_i = [v_{i1}, \cdots, v_{in_x}]$.

Global Best PSO

- Update of personal best position:

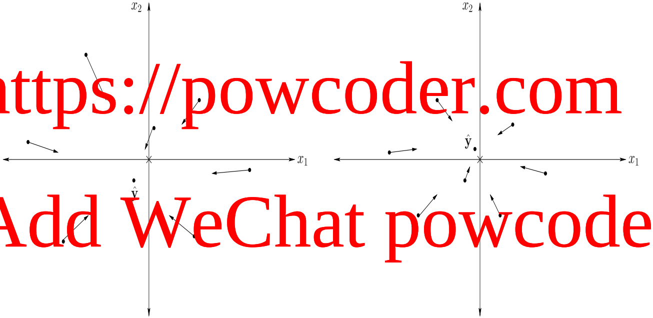
$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{otherwise} \end{cases}, i = 1, \cdots, n_s$$

- The global best position:

$$\hat{\mathbf{y}}(t+1) \in \{\mathbf{y}_1(t+1), \cdots, \mathbf{y}_{n_s}(t+1)\} | f(\hat{\mathbf{y}}(t+1)) = \min\{f(\mathbf{y}_1(t+1)), \cdots, f(\mathbf{y}_{n_s}(t+1))\}$$

Global Best PSO

- What is the swarm size?
- How many elements does each particle have?
- How many elements does the velocity vector have?
- What is the initial velocity, $\mathbf{v}(0)$?



(a) At time $t = 0$

(b) At time $t = 1$

Figure 1: Multi-particle *gbest* PSO illustration. '$\times$' indicates the optimum.

KING'S College LONDON
University of London

***gbest* PSO algorithm**

Create and initialise an n_x-dimensional swarm, $\mathbf{y}_i(0)$ and $\hat{\mathbf{y}}(0)$;
Set $t = 0$, $\mathbf{v}_i(0) = \mathbf{0}$; Choose values for c_1 and c_2;

while STOP-CRIT **do**
  for *each particle* $i = 1, \cdots, n_s$ **do**
    Update the velocity, $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\big(y_{ij}(t) - x_{ij}(t)\big) + c_2 r_{2j}(t)\big(\hat{y}_j(t) - x_{ij}(t)\big)$;
    Update the position of particles, $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$;
  end
  for *each particle* $i = 1, \cdots, n_s$ **do**
    if $f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t))$ **then**
      $\mathbf{y}_i(t+1) = \mathbf{x}_i(t+1)$;
    else
      $\mathbf{y}_i(t+1) = \mathbf{y}_i(t)$;
    end
    if $f(\mathbf{y}_i(t+1)) < f(\hat{\mathbf{y}}(t))$ **then**
      $\hat{\mathbf{y}}(t+1) = \mathbf{y}_i(t+1)$;
    end
  end
  $t \leftarrow t + 1$;
end

Table 1: Pseudocode of *gbest* PSO algorithm.

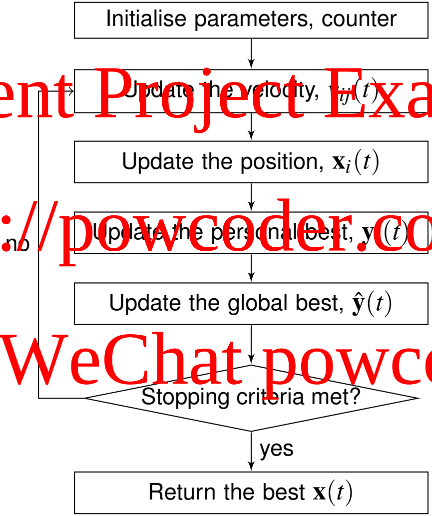
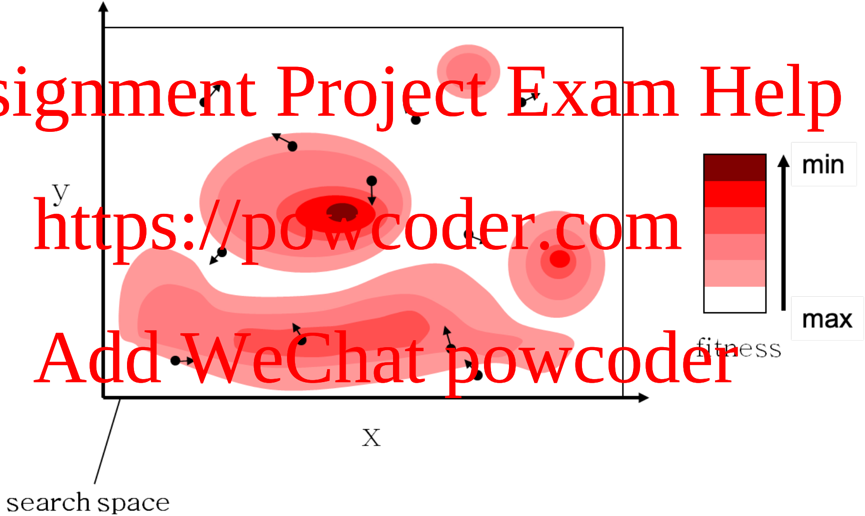
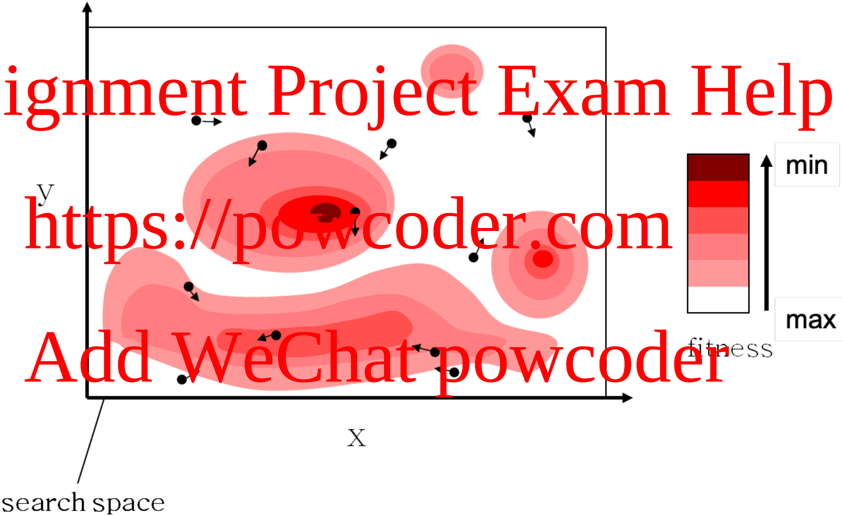


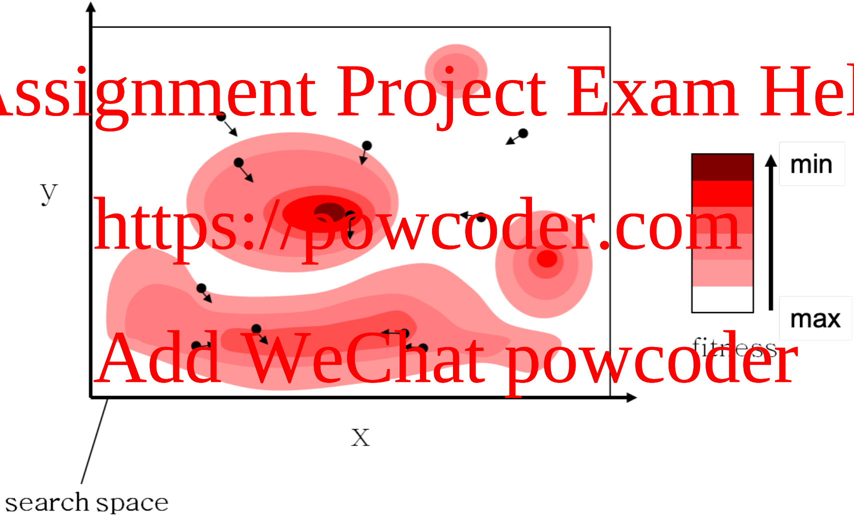
Figure 2: Flowchart of *gbest* PSO algorithm.

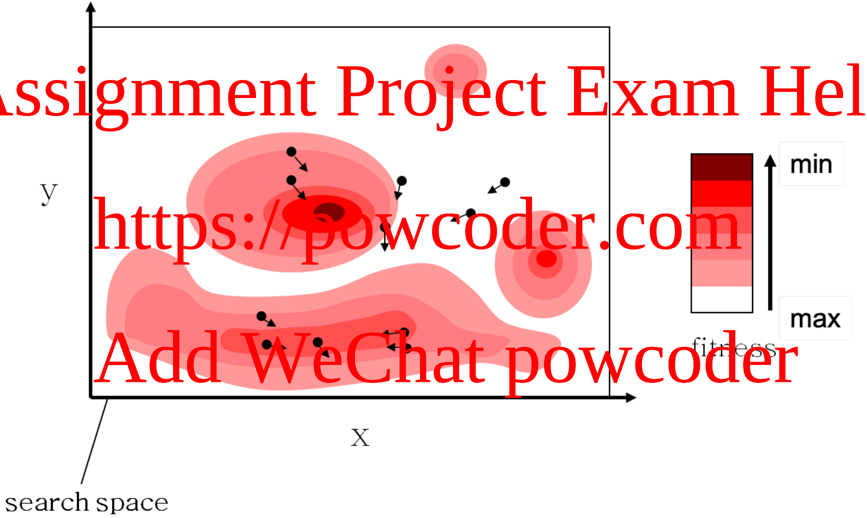

y
x
search space
min
max
fitness

Global Best PSO

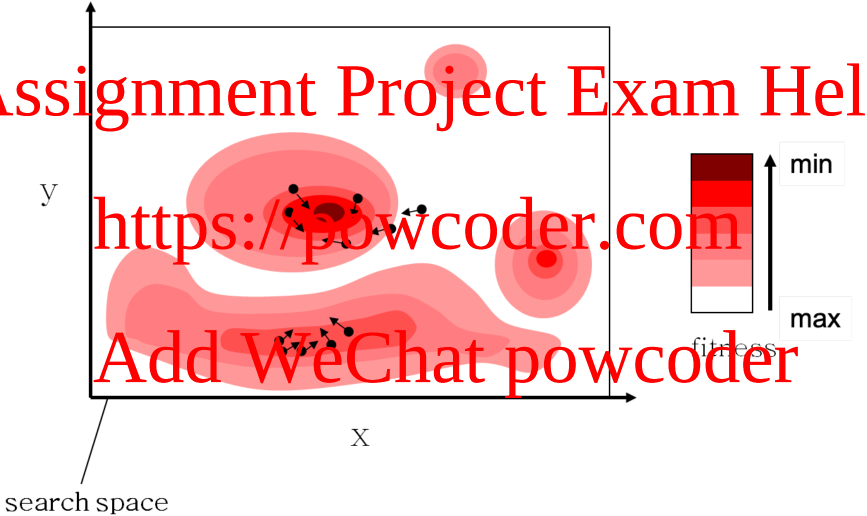


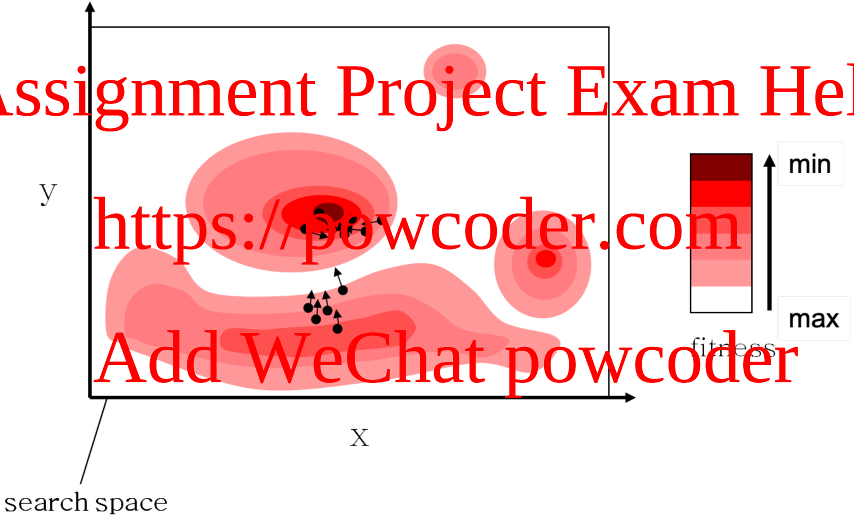
KING'S College LONDON
University of London

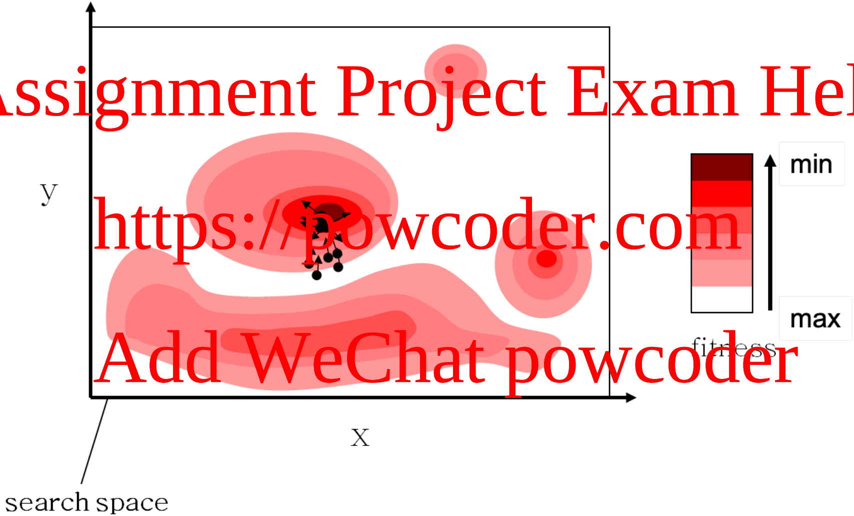


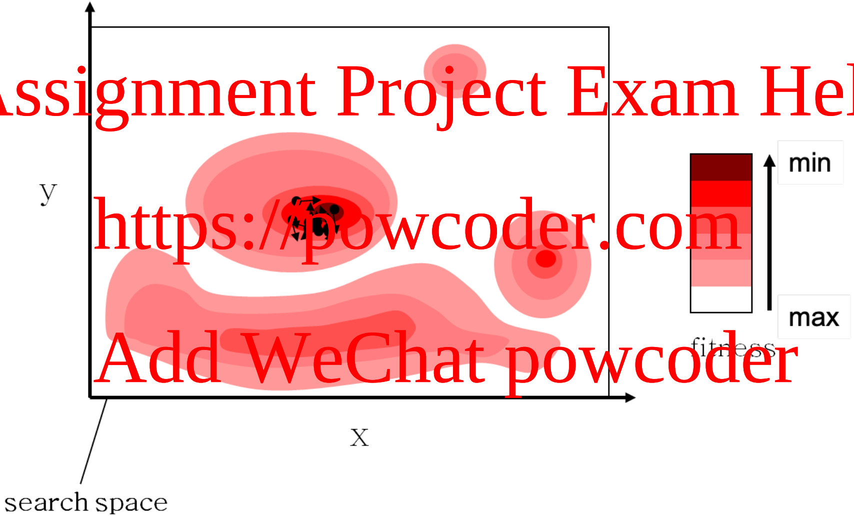

y
x
min
max
fitness
search space

Global Best PSO




y
x
search space
min
max
fitness


y
min
max
fitness
x
search space


y
x
search space
min
max
fitness

Local Best PSO

KING'S College LONDON
University of London

- Size of Swarm: n_s particles.
- The (**local**) best position found in the swarm (found by the neighbourhood of particle i):

$$\hat{\mathbf{y}}_i(t) = [\hat{y}_{i1}(t), \cdots, \hat{y}_{in_x}(t)]$$

- The personal best position since the first generation:

$$\mathbf{y}_i(t) = [y_{i1}(t), \cdots, y_{in_x}(t)]$$

- **Velocity update rule:**

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\big(y_{ij}(t) - x_{ij}(t)\big) + c_2 r_{2j}(t)\big(\hat{y}_{ij}(t) - x_{ij}(t)\big) \quad (2)$$

where $c_1 \geq 0$ and $c_2 \geq 0$ are acceleration constants; $r_{1j}(t)$ and $r_{2j}(t) \in [0, 1]$ are random numbers.

- **Particle update rule:**

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \; i = 1, \cdots, n_s$$

KING'S College LONDON
University of London

- The (**local**) best position found in the swarm (found by the neighbourhood of particle i):

$$\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \min\{f(\mathbf{x}(t))\}, \forall \mathbf{x}(t) \in \mathcal{N}_i\}$$

$\mathcal{N}_i$: the set of neighbourhoods of particle i.

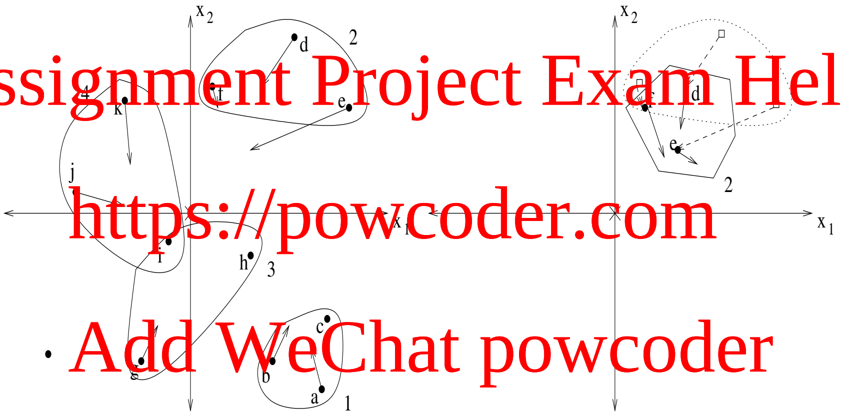
- The local best position is the neighbourhood best position.
- The personal best position since the first generation:

$$\mathbf{y}_i(t) = [y_{i1}(t), \cdots, y_{in_x}(t)]$$

- **Particle update rule:**

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \; i = 1, \cdots, n_x$$

- Note: When the neighbourhoods of all particles are the entire swarm, the *lbest* PSO becomes the *gbest* PSO.



(a) Local Best Illustrated – Initial Swarm

(b) Local Best – Second Swarm

Figure 3: Illustration of *lbest* PSO. '$\times$' indicates the optimum. $v_{ij}(0) = 0$ except for particle f.

KING'S College LONDON
University of London

***lbest* PSO algorithm**

Create and initialise an n_x-dimensional swarm, $\mathbf{y}_i(0)$ and $\hat{\mathbf{y}}_i(0)$;
Set $t = 0$, $\mathbf{v}_i(0) = \mathbf{0}$; Choose values for c_1 and c_2;

while STOP-CRIT **do**
  for *each particle* i = 1, $\cdots$, n_s **do**
    Update the velocity, $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t)\left(\hat{y}_{ij}(t) - x_{ij}(t)\right)$;
    Update the position of particles, $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$;
  end
  for *each particle* i = 1, $\cdots$, n_s **do**
    if $f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t))$ **then**
      $\mathbf{y}_i(t+1) = \mathbf{x}_i(t+1)$;
    else
      $\mathbf{y}_i(t+1) = \mathbf{y}_i(t)$;
    end
  end
  for *each particle* i = 1, $\cdots$, n_s **do**
    $\hat{\mathbf{y}}_i(t+1) \in \{\mathscr{N}_i | f(\hat{\mathbf{y}}_i(\mathbf{x}(t))) = \min\{f(\mathbf{x}(t))\}, \forall \mathbf{x}(t) \in \mathscr{N}_i\}$
  end
  $t \leftarrow t+1$;
end

$n_{\mathscr{N}_i}$: number of neighbourhoods

Table 2: Pseudo Code of *lbest* PSO algorithm.

Local Best PSO

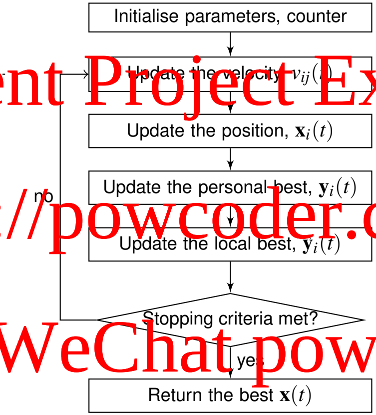


Figure 4: Flowchart of *lbest* PSO algorithm.

Selection of Neighbourhoods:

- **Particles indices:**
  - Computationally inexpensive.
  - Promotion of the spread of information irrespective of the position of the particles.
- **Spatial similarity:**
  - Computationally expensive.
  - Information of similar particles can be used for a local search.
- **Overlapping of neighbourhoods:** A particle can be a member of a number of neighbourhoods. The interconnection of neighbourhoods can promote information sharing such that all particles come to consensus faster (converge faster to a single solution).

KING'S College LONDON
University of London

***lbest* vs *gbest* PSO:**

	lbest PSO	*gbest* PSO
computational demand	higher	lower
diversity	larger	smaller
convergence speed	slower	faster
trapped in local minima	less likely	more likely

Velocity Components

King's College London
University of London

- **Velocity Components:** previous velocity, cognition component and social component

- ***gbest* best PSO:**

$$v_{ij}(t+1) = \underbrace{v_{ij}(t)}_{\text{previous velocity}} + \underbrace{c_1 r_{1j}(t)\big(y_{ij}(t) - x_{ij}(t)\big)}_{\text{cognitive component}} + \underbrace{c_2 r_{2j}(t)\big(\hat{y}_j(t) - x_{ij}(t)\big)}_{\textit{social component}}$$

- ***lbest* best PSO:**

$$v_{ij}(t+1) = \underbrace{v_{ij}(t)}_{\text{previous velocity}} + \underbrace{c_1 r_{1j}(t)\big(y_{ij}(t) - x_{ij}(t)\big)}_{\text{cognitive component}} + \underbrace{c_2 r_{2j}(t)\big(\hat{y}_j(t) - x_{ij}(t)\big)}_{\textit{social component}}$$

- **Velocity Components:** previous velocity, cognition component and social component
- ***gbest* best PSO:**

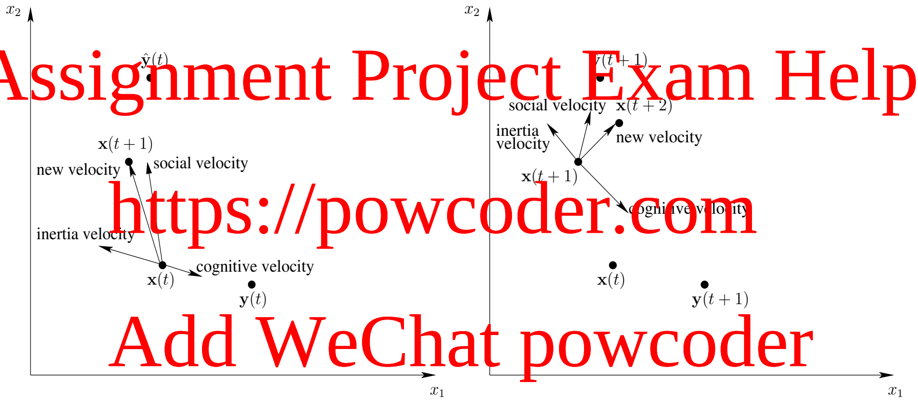
$$v_{ij}(t+1) = \underbrace{v_{ij}(t)}_{\text{previous velocity}} \underbrace{+c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right)}_{\text{cognitive component}} \underbrace{+c_2 r_{2j}(t)\left(\hat{y}_j(t) - x_{ij}(t)\right)}_{\textit{social component}}$$

- ***lbest* best PSO:**

$$v_{ij}(t+1) = \underbrace{v_{ij}(t)}_{\text{previous velocity}} \underbrace{+c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right)}_{\text{cognitive component}} \underbrace{+c_2 r_{2j}(t)\left(\hat{y}_{ij}(t) - x_{ij}(t)\right)}_{\textit{social component}}$$

- Previous velocity: it is an inertia making the particle move in the same direction as in the t^{th} generation.
- Cognitive component: It is a personal influence which attempts to improve the individual by making the particle return to a previous good position.
- Social component: It is social influence which makes the particle follow the best neighbour's direction.
- Previous velocity is for exploration: Its searches new regions (new solutions) for potentially better solutions.
- Cognitive component and social component are for exploitation: It searches the previous regions (previous solutions) for better solutions.

Velocity Components



(a) Time Step t (b) Time Step $t+1$

Figure 5: Illustration of Velocity and Position Update for *gbest* PSO.

Particle Initialisation

- **Particle position:** $x_{ij}(0) = x_{\min,j} + r_j(x_{\max,j} - x_{\min,j})$, $i = 1, \cdots, n_s$; $j = 1, \cdots, n_x$.
  - $r_j \in [0, 1]$: a random number
  - $[x_{\min,j}, x_{\max,j}]$: boundary of the j-th element of particle, for all j
- **Initialised velocity:** $\mathbf{v}_i(0) = \mathbf{0}$, $i = 1, \cdots, n_s$.
- **Personal best position:** $\mathbf{y}_i(0) = \mathbf{x}_i(0)$, $i = 1, \cdots, n_s$.

Stopping Criteria

KING'S College LONDON
University of London

- Maximum number of iterations has been reached.
- Acceptable solution has been found.
- No improvement has been observed over a period of iterations.
- The swarm radius is close to zero.
- The objective function slope is approximately zero.
    - Slope: $f'(t) = \frac{f(\hat{\mathbf{y}}(t)) - f(\hat{\mathbf{y}}(t-1))}{f(\hat{\mathbf{y}}(t))}$
    - algorithm terminates if $|f'(t)| \le \varepsilon$ for a number of consecutive iterations.
    - $\varepsilon > 0$: a user-specified parameter

KING'S College LONDON University of London

Swarm Radius:

1. **Maximum swarm radius:** The algorithm terminates if $R_{\max}(t) \leq \varepsilon$.
   - $\varepsilon > 0$: a user-specified parameter.
   - $R_{\max}(t) = \max_{m \in \{1,\cdots,n_s\}} ||\mathbf{x}_m(t) - \hat{\mathbf{y}}(t)||$.
   - $\hat{\mathbf{y}}(t)$ is the global best position of the *gbest* or *lbest* PSO.
2. **Particle clustering algorithm:** The algorithm terminates if $\frac{|C|}{n_s} \geq \delta$.
   - $0 < \delta \leq 1$: a user-specified parameter.
   - C: a cluster.
   - $|C|$: number of elements in cluster C.

KING'S College LONDON
University of London

Particle Clustering Algorithm

Initialise cluster $C = \{\hat{\mathbf{y}}(t)\}$;

for *about 5 times* **do**

  Calculate the centroid of cluster C: $\bar{\mathbf{x}}(t) = \dfrac{\sum_{i=1,\mathbf{x}_i(t)\in C}^{|C|} \mathbf{x}_i(t)}{|C|}$;

  for each particle i, $i = 1, \cdots, n_s$ **do**

    if $||\mathbf{x}_i(t) - \bar{\mathbf{x}}(t)|| \le \varepsilon$ **do**

      $C \leftarrow C \cup \{\mathbf{x}_i(t)\}$

    end

  end

end

$\varepsilon > 0$: a user-specified parameter

Table 3: Pseudo Code of Particle Clustering Algorithm.

Social Network Structure

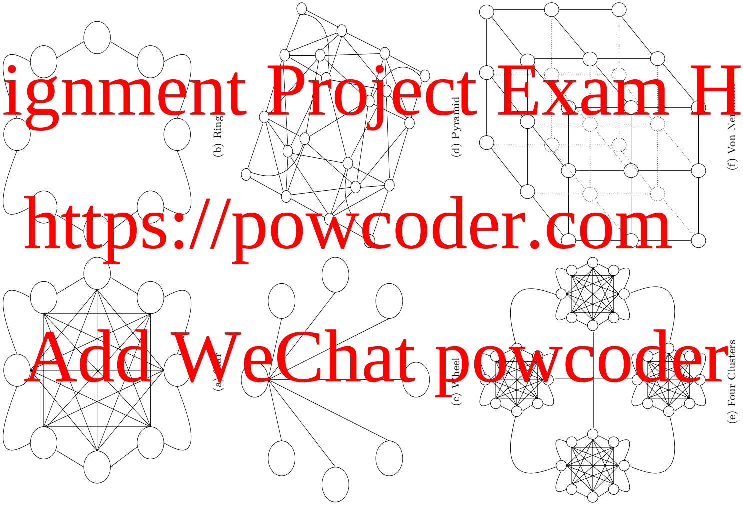


Figure 6: Examples of Social Network Structure.

Velocity Clamping

- **Exploration:** the ability to explore different regions to locate good optima.
- **Exploitation:** the ability to concentrate the search around a region to refine a solution.
- **Potential problem of *gbest* and *lbest* PSO:** Velocity explosion – velocities are updated quickly to large values.
  - It pushes particles to boundaries.

- **Velocity Clamping:** Particle velocity is adjusted before updating the particles' positions.

$$v'_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } v_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{otherwise} \end{cases}$$

  - $V_{\max,j} > 0$: a user-specified maximum velocity of the $j^{th} element$ of the particle.
  - $V_{\max,j}$ controls 1) particle moving speed 2) ability of exploration and exploitation.
  - When $v_{ij}(t+1)$ is of negative, the above condition needs to be modified accordingly.

- **Advantage:** Explosion of velocity is controlled.
- **Disadvantages:** 1)The searching direction of particle is changed. 2) When velocities reach maximum, particles will search on a hypercube defined by $[x_{ij}(t) - V_{\max,j}, x_{ij}(t) + V_{\max,j}]$ for all i.

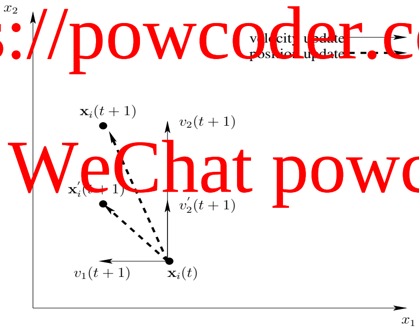


Figure 7: Effect of Velocity Clamping.

KING'S College LONDON
University of London

Dynamic Velocity Approaches:

- Change of the maximum velocity if the global best position does not improve over τ consecutive iterations:

$$V_{\max,j}(t+1) = \begin{cases} \gamma V_{\max,j}(t) & \text{if } f(\hat{\mathbf{y}}(t)) \geq f(\hat{\mathbf{y}}(t-t')) \ \forall\, t' = 1, \cdots, \tau \\ V_{\max,j}(t) & \text{otherwise} \end{cases}$$

  γ decreases linearly or exponentially from 1 to 0.01.

- Exponentially decay the maximum velocity:

$$V_{\max,j}(t+1) = \left(1 - \left(\frac{t}{n_t}\right)^{\alpha}\right) V_{\max,j}(t)$$

  $\alpha \geq 0$: a user-specified parameter, n_t is the maximum number of iterations.

- Velocities are updated:

$$v_{ij}(t+1) = V_{\max,j}(t+1) \tanh\left(\frac{v'_{ij}(t+1)}{V_{\max,j}(t+1)}\right)$$

  where $v'_{ij}(t+1)$ is calculated from the *gbest* velocity update rule (1) or *lbest* velocity update rule (2).

Inertia Weight

Inertia Weight

King's College London, University of London

- **Inertia Weight** - a mechanism 1) to control the exploration and exploitation abilities of the swarm, 2) to eliminate the need for velocity clamping.
- ***gbest* PSO:**

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t)\left(\hat{y}_j(t) - x_{ij}(t)\right)$$

- ***lbest* PSO:**

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t)\left(\hat{y}_{ij}(t) - x_{ij}(t)\right)$$

- $w > 0$
  - $w \geq 1$: Velocities increase over time.
  - $w < 1$: Velocities decrease over time.

KING'S College LONDON
University of London

- **Random adjustments:** $w \in [0, 1]$: a random number
- **Linear decreasing:**
  - $w(t) = (w(0) - w(n_t))\frac{n_t - t}{n_t} + w(n_t)$.
  - $w(0)$: initial inertia weight.
  - $w(n_t)$: final inertia weight.
  - $w(0) \geq w(n_t)$.
  - n_t : maximum number of iterations.
- **Nonlinear decreasing:**
  - $w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n_t + 0.4}$ with $w(0) = 0.9$.