Swarm Intelligence

Particle Swarm Optimization

Christian Camacho Villalón
christian.camacho.villalon@ulb.ac.be

IRIDIA – Université Libre de Bruxelles (ULB)
Bruxelles, Belgium

Particle Swarm Optimisation

- Perturbative algorithm
  - A set of solutions are input to the algorithm from the beginning
- Originally proposed for continuous problems
  - Adaptations have been propose to tackle discrete and mixed variable problems
- Particles communicate among them and share information about the solutions they have found.
  - Topologies and models of influence

Particle Swarm Optimisation

- Solutions: particle moving through the search space
  - Position $x[] = (x_1, x_2, \ldots, x_n)$
  - Velocity $v[] = (v_1, v_2, \ldots, v_n)$
  - Personal best position $p[] = (p_1, p_2, \ldots, p_n)$
  - Global best position $g[] = (g_1, g_2, \ldots, g_n)$

$$v_i(t+1) = v_i(t) + \psi_1 * U_1(p_i(t) - x_i(t)) + \psi_2 * U_2(g(t) - x_i(t))$$

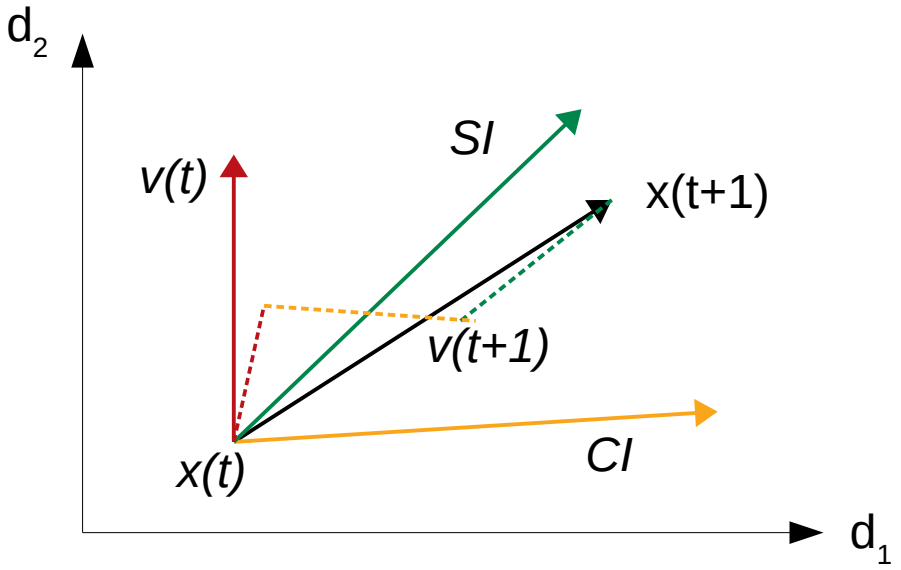
$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Particle Swarm Optimisation

Inertia | Cognitive influence (CI) | Social Influence (SI)

$$v_i(t+1) = \omega * v_i(t) + \psi_1 * U_1(p_i(t) - x_i(t)) + \psi_2 * U_2(g(t) - x_i(t))$$

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

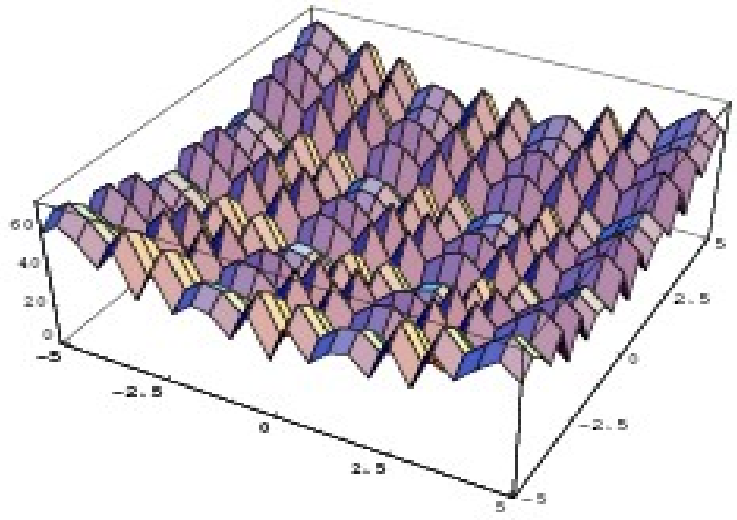


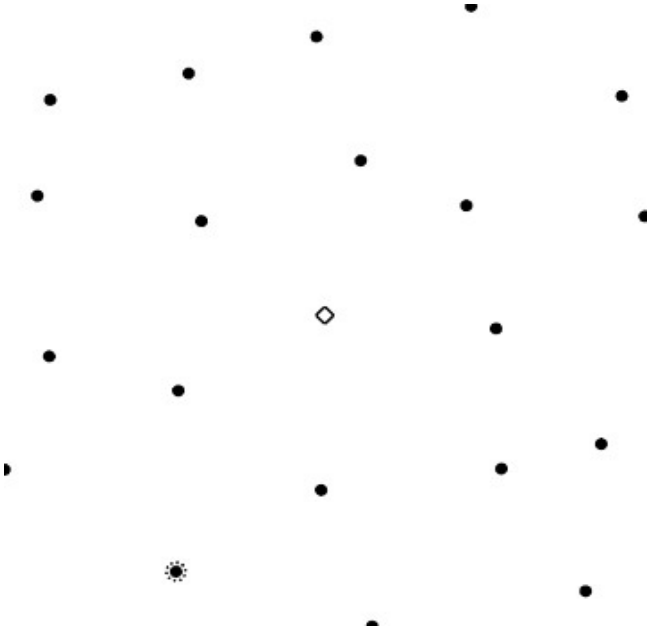
Particle Swarm Optimisation

```
1  Initialize particles
2  While(!termination)
3    Update global best        #Topology dependent
4    Update velocity
5    Update current position
6    Update personal best
7  End while
8  Return best solution
```

Particle Swarm Optimisation

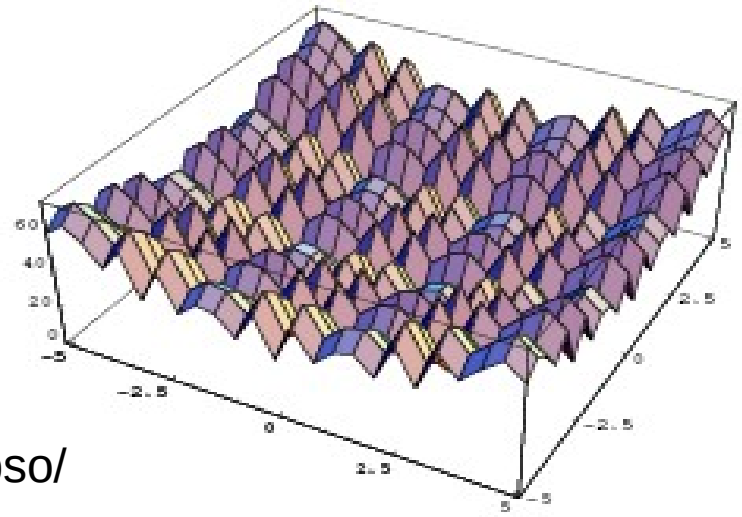
$$F_{\text{Rastrigin}} = \sum_{i=1}^{m} ({x_i}^2 - 10\cos(2\pi x_i) + 10)$$





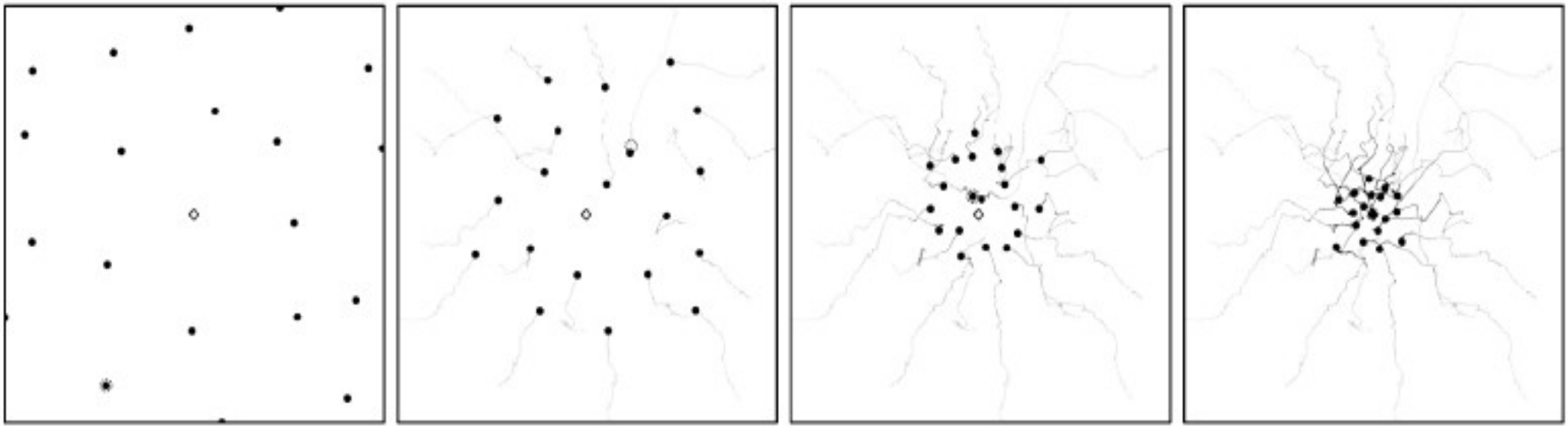
http://cg.kw.ac.kr/kang/pso/

Particle Swarm Optimisation



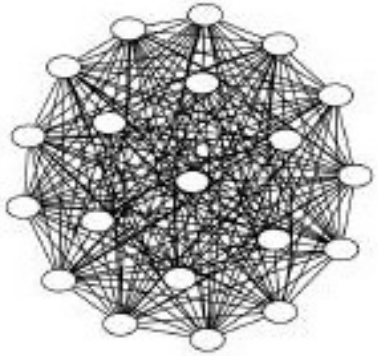
$$F_{\text{Rastrigin}} = \sum_{i=1}^{m} ({x_i}^2 - 10\cos(2\pi x_i) + 10)$$

http://cg.kw.ac.kr/kang/pso/

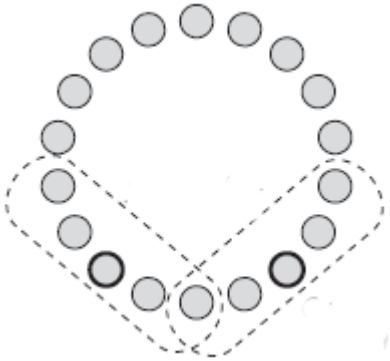


PSO: Topologies

- Define a neighbourhood for the particles.
  - Fully-connected (gBest): all particles are neighbours.



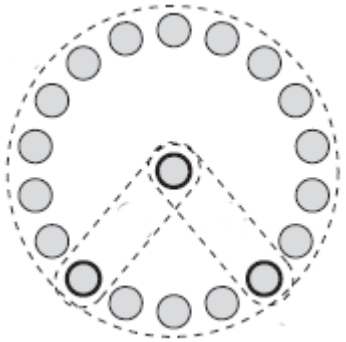
  - Ring: Each particle has n other neighbours.



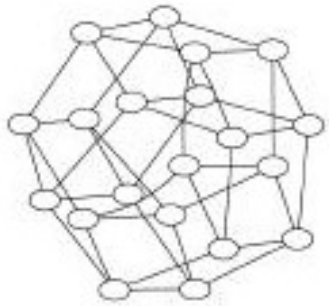
m=4

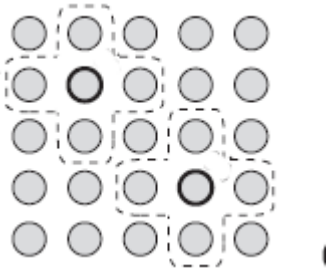
PSO: Topologies

- Wheel: one central particle.



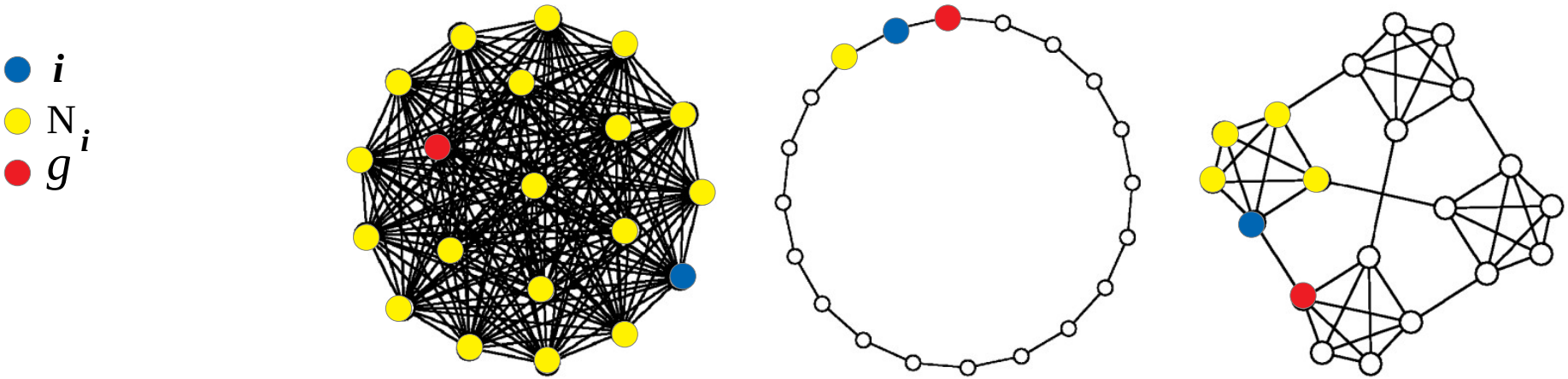
- Von Neumann: Network of 2 dimensions, each particle connected up and down, left and right.





PSO: Models of influence

- The model of influence determines which neighbor(s) will contribute to update the velocity vector of particle ***i***



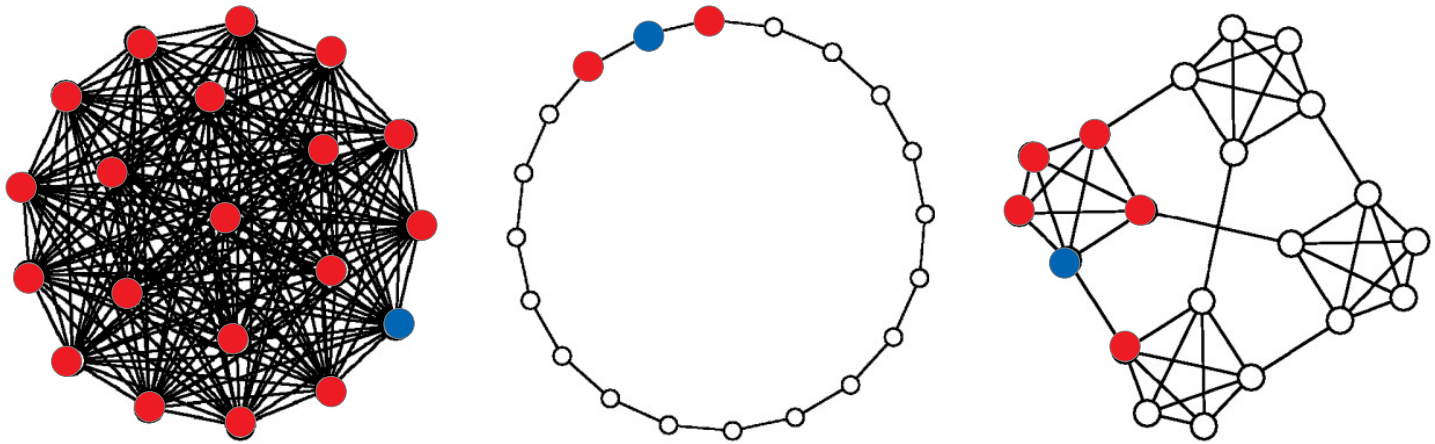
  - **Best of neighborhood**: only *g*best or *l*best

$$v_i(t+1) = \omega * v_i(t) + \psi_1 * U_1(p_i(t) - x_i(t)) + \psi_2 * U_2(g(t) - x_i(t))$$

PSO: Models of influence

- The model of influence determines which neighbor(s) will contribute to update the velocity vector of particle ***i***

p_i
N_i



  - Best of neighborhood: only *g*best or *l*best
  - **Fully informed**: all neighbors

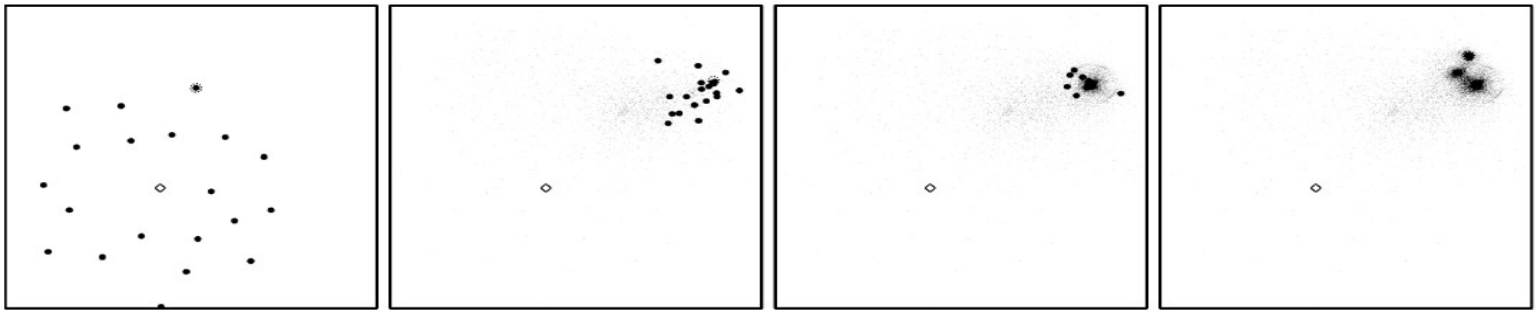
$$v_i(t+1) = \chi[v_i(t) + \sum_{j=1}^{N} \psi_j * U_j(p_j(t) - x_i(t))]$$

$$\psi_j = \frac{\psi}{|N_i|} \qquad \chi = 0.7298$$

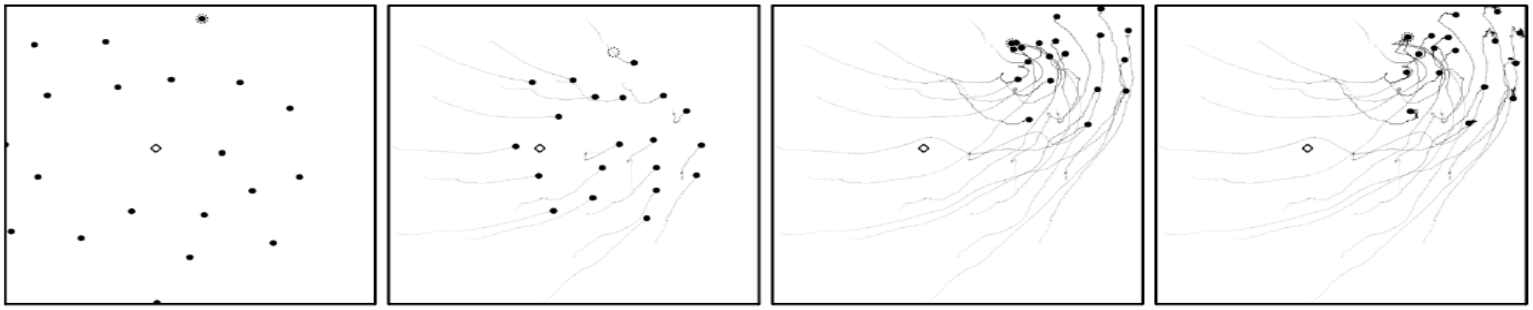
PSO: Inertia

$$v_i(t+1) = \omega * v_i(t) + \psi_1 * U_1(p_i(t) - x_i(t)) + \psi_2 * U_2(g(t) - x_i(t))$$

- Controlling the balance of the search
  - Small inertia → more exploitative



  - Large inertia → more exploratory



PSO: Inertia

- The value of inertia can vary during the search:
  - Example: Privilege exploration in initial iterations and exploitation at the end.

$$v_i(t+1) = \omega(t) * v_i(t) + \psi_1 * U_1(p_i(t) - x_i(t)) + \psi_2 * U_2(g(t) - x_i(t))$$

  - Linear decreasing:

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

  - No linear decreasing:

$$\omega(t+1) = \omega(T) - (\omega(T) - \omega(0))\left(\frac{T-t}{T}\right)^{\alpha}$$