Swarm Intelligence

Particle Swarm Optimization

Christian Camacho Villalón
christian.camacho.villalon@ulb.ac.be
Federico Pagnozzi
federico.pagnozzi@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 $g[\,]=(g_1, g_2, \ldots, g_n)$

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

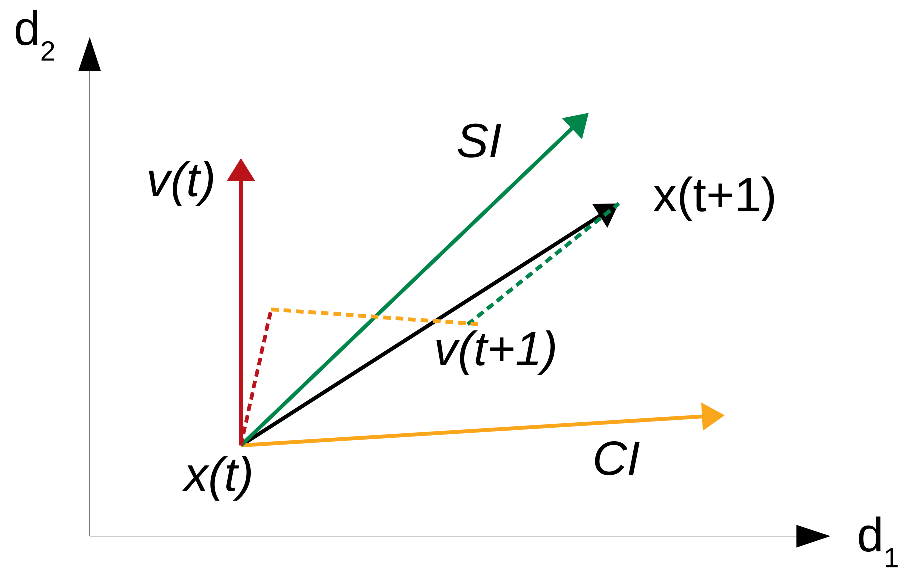
$$x_i(t+1)=x_i(t)+v_i(t+1)$$

Particle Swarm Optimisation

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

$$v_i(t+1) = \boxed{\omega * v_i(t)} + \boxed{\psi_1 * U_1(p_i - x_i(t))} + \boxed{\psi_2 * U_2(g_i - x_i(t))}$$

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

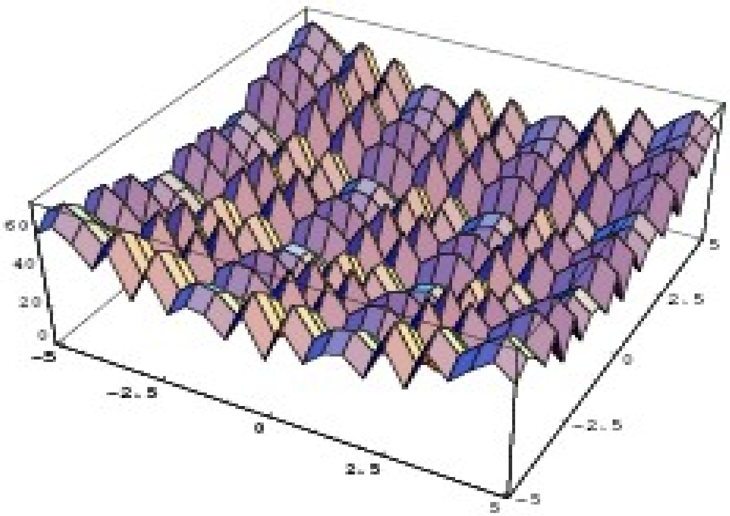


Particle Swarm Optimisation

```
Initialize particles
While(!termination)
  Update global best        #Topology dependent
  Update velocity
  Update current position
  Update personal best
End while
Return best solution
```

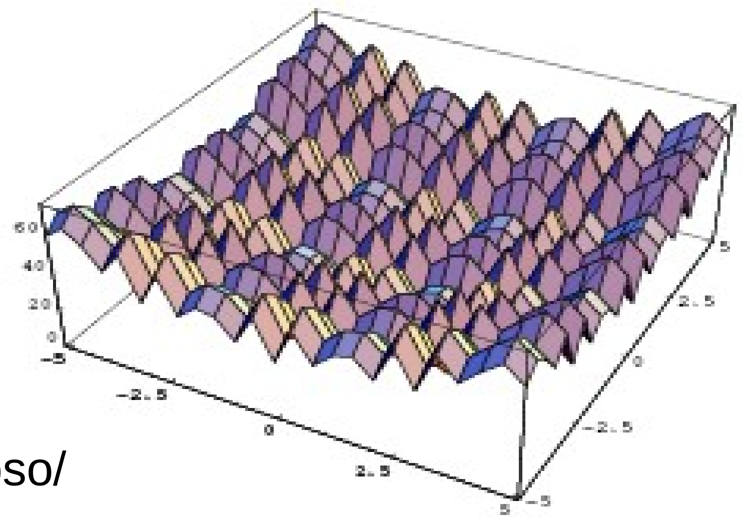
Particle Swarm Optimisation

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



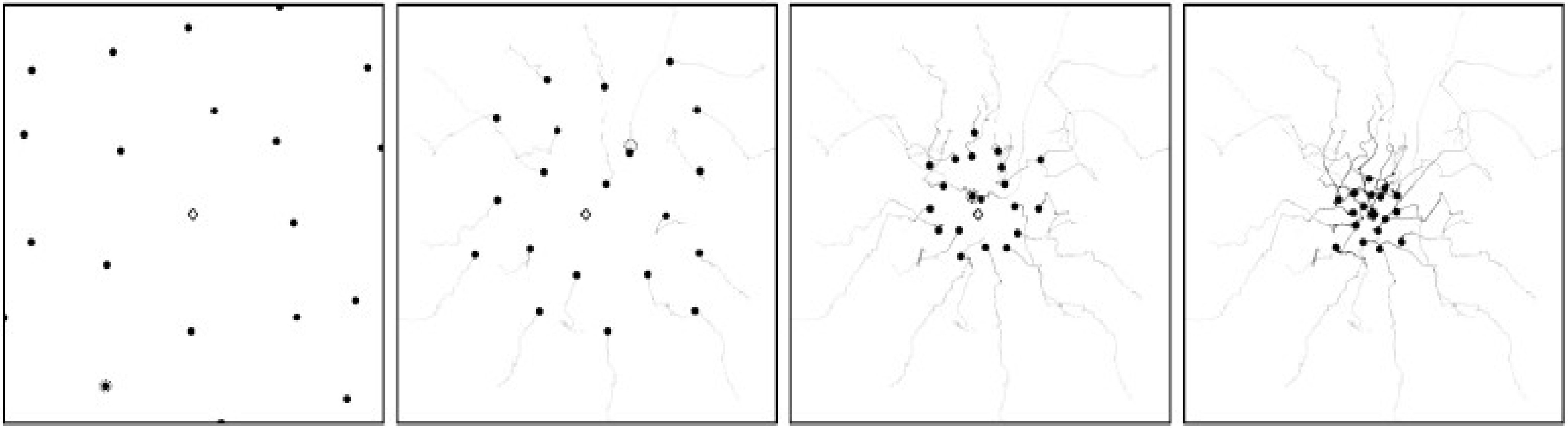
http://cg.kw.ac.kr/kang/pso/

Particle Swarm Optimisation



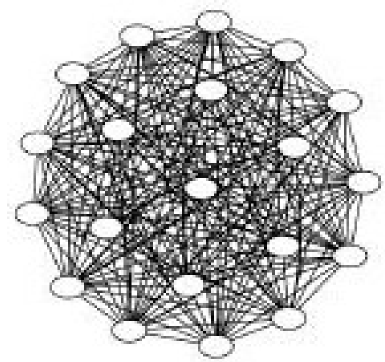
$$F_{\mathrm{Rastrigin}} = \sum_{i=1}^{m} (x_i{}^2 - 10\cos(2\pi x_i) + 10)$$

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

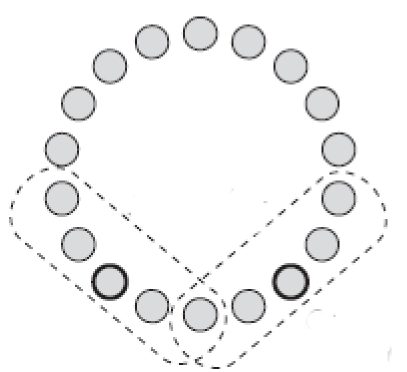


Particle Swarm Optimisation: Topologies

- Define a neighbourhood for the particles.
  - Gbest: all particles are neighbours.



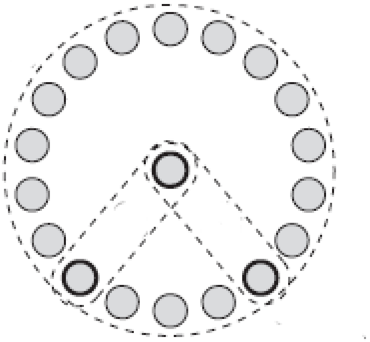
  - Ring: Each particle has n other neighbours.



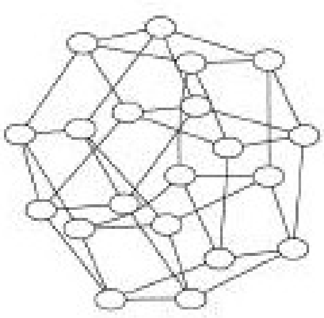
m=4

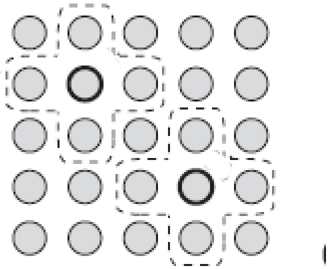
Particle Swarm Optimisation: Topologies

– Wheel: one central particle.



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

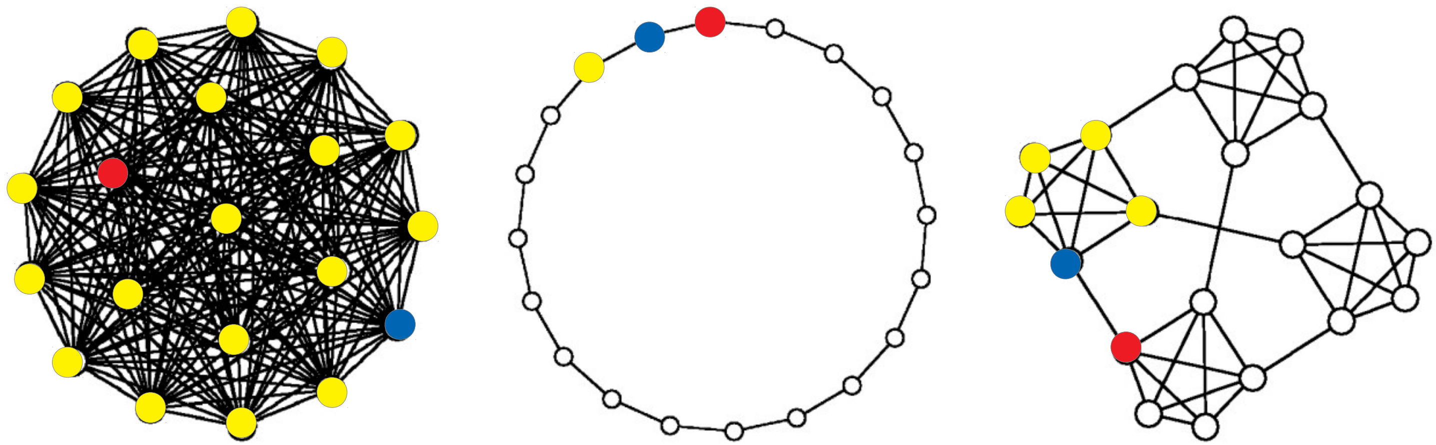




Particle Swarm Optimisation: Models of influence

- The model of influence determines which particle(s) will contribute to update the velocity vector of $\boldsymbol{p_i}$ (aka its informants)

$\boldsymbol{p_i}$
N_i
g_i

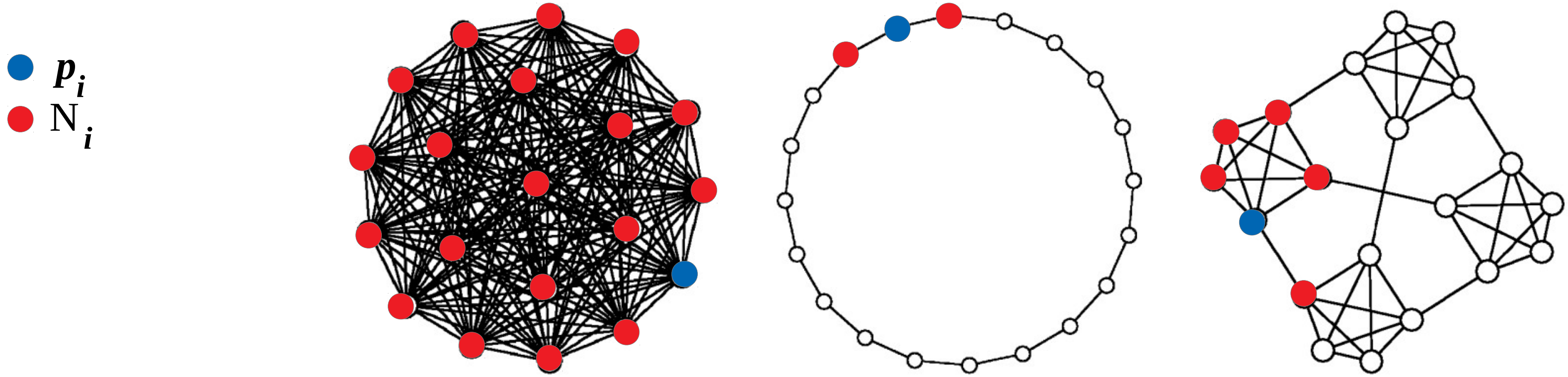


  - **Best of neighbourhood**: only *g*best or *l*best

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

Particle Swarm Optimisation: Models of influence

- The model of influence determines which particle(s) will contribute to update the velocity vector of $\boldsymbol{p_i}$ (aka its informants)



  - Best of neighbourhood: only *g*best or *l*best
  - **Fully informed**: all neighbours

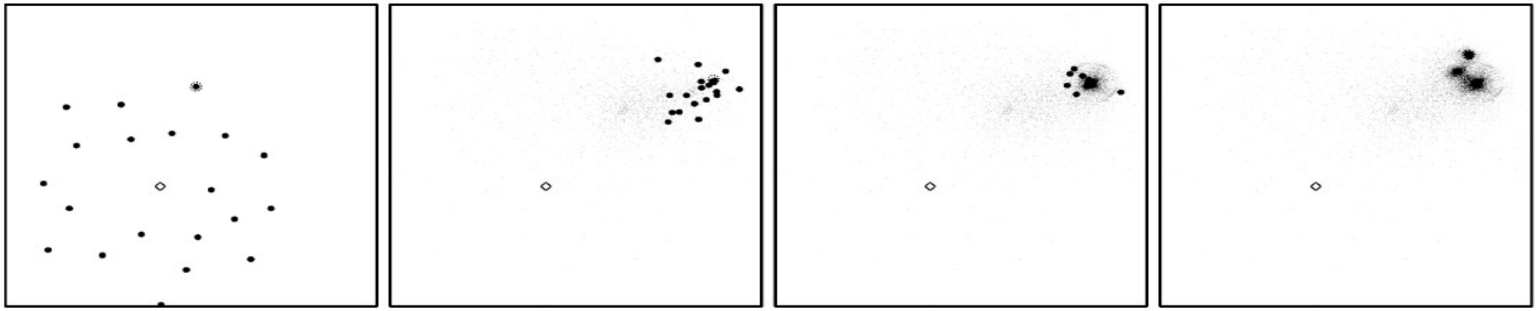
$$v_i(t+1)=\chi\left[v_i(t)+\sum_{j=1}^{N}\psi_j * U_j(p_j-x_i(t))\right]$$

$$\psi_j=\frac{\psi}{|N_{\boldsymbol{i}}|} \qquad \chi=0.7298$$

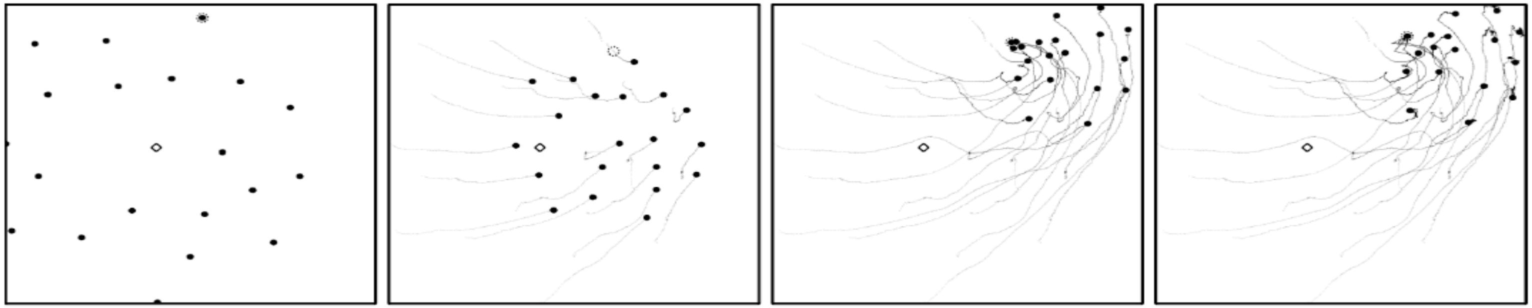
Particle Swarm Optimisation: Inertia

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

- Controlling the balance of the search
  - Small inertia $\rightarrow$ more exploitative



  - Large inertia $\rightarrow$ more exploratory



Particle Swarm Optimisation: Inertia

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

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

  - Linear decreasing:

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

  - No linear decreasing:

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