Swarm Intelligence

Particle Swarm Optimization

Leslie Pérez Cáceres
leslie.perez.caceres@ulb.ac.be
Hayfa Hammami
haifa.hammami@ulb.ac.be

IRIDIA – Université Libre de Bruxelles (ULB)
Bruxelles, Belgium
Particle Swarm Optimisation

- Perturbative algorithm
  - Initial solutions must be provided
- Originally proposed for continuous problems
  - Adaptations have been propose to tackle mixed and discrete problems
- Particles share information
  - Topologies
Particle Swarm Optimisation

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

\[
\begin{align*}
  v_i(t+1) &= v_i(t) + \psi_1 * u_1(p_i - x_i(t)) + \psi_2 * u_2(g_i - x_i(t)) \\
  x_i(t+1) &= x_i(t) + v_i(t+1)
\end{align*}
\]
Particle Swarm Optimisation

\[ v_i(t+1) = \omega \cdot v_i(t) + \psi_1 \cdot u_1(p_i - x_i(t)) + \psi_2 \cdot u_2(g_i - x_i(t)) \]

Inertia

Personal influence

Social Influence

\[ x_i(t+1) = x_i(t) + v_i(t+1) \]
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: Topologies

- Define a neighbourhood for the particles.
  - Gbest: all particles are neighbours.
  - Ring: Each particle has $n$ other neighbours.
Particle Swarm Optimisation: Topologies

- **Star**: one central particle.

- **Von Neumann**: Network of 2 dimensions, each particle connected up and down, left and right.
Particle Swarm Optimisation: Inertia

\[ v_i(t+1) = \omega \ast v_i(t) + \psi_1 \ast u_1(p_i - x_i(t)) + \psi_2 \ast u_2(g_i - x_i(t)) \]

- Controlling the balance of the search
  - Inertia is small \( \rightarrow \) more intensification
  - Inertia is big \( \rightarrow \) more exploration
Particle Swarm Optimisation: Inertia

- Inertia can be varied during the search:
  - Example: Privilege exploration in initial iterations and exploitation at the end.

\[ v_i(t+1) = \omega(t) \cdot v_i(t) + \psi_1 \cdot u_1(p_i - x_i(t)) + \psi_2 \cdot u_2(g_i - x_i(t)) \]

- Linear:

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

- No lineal

\[ \omega(t+1) = \alpha \omega(t') \]