<u>Swarm Intelligence</u>

Particle Swarm Optimization

Based on slides by Thomas Bäck, which were based on: Riccardo Poli, James Kennedy, Tim Blackwell: Particle swarm optimization. Swarm Intelligence 1(1): 33-57 (2007)

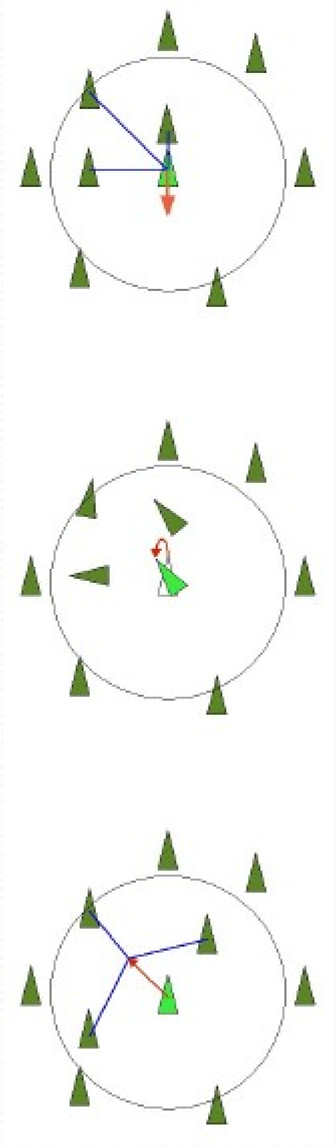
Particle Swarm Optimisation

- Optimization strategy inspired on bird flocking or fish schooling

Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. Proceedings of the Fourth IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center 1942-1948, 1995.

Origins

- PSO started from a behavioral model (by Reynolds) in which an agent follows three rules:
  - **Separation:** agents move away from neighbors that are too close
  - **Alignment:** agents steer towards the average heading of neighbors
  - **Cohesion:** agents steer towards the average position of neighbors



Origins - Roosts

- Kennedy and Eberhart included a roost in a simplified Reynolds-like simulation so that:
  - agents are attracted towards the roost
  - agents remember where they were closest to the roost
  - agents share information with neighbors about the closest location to the roost

General Ideas

- PSO simulates a swarm of particles
- Each particle has
  - a current position ~ genotype
  - a memory of its best position till now
  - a fitness ~ fitness
  - a velocity ~ strategy parameters
- The velocity of a particle is influenced by
  - its own best position so far
  - the best position of its neighbors so far

Original PSO Algorithm

```
initialize particles (positions, velocities)
for each iteration do
        for k = 1 to number of particles do
                evaluate fitness
                determine particles closeby
                if fitness at current position is better
                  than at best position then
                        update best position
                end if
                update velocity
                update position
        end do
end do
return best solution found
```

(Asynchronous)

Original PSO Algorithm

```
initialize particles (positions, velocities)
for each iteration do
      for k = 1 to number of particles do
            evaluate fitness
            determine particles closeby
            if fitness at current position is
              better than at best position then
                  update best position
            end if
            update velocity
            update position
      end do
end do
return best solution found
```

Asynchronous

```
initialize particles (positions, velocities)
for each iteration do
      for k = 1 to number of particles do
            evaluate fitness
            determine particles closeby
            update velocity
      end do
      for k = 1 to number of particles do
            update position
            if fitness at current position is
              better than at best position then
                  update best position
            end if
      end do
end do
return best solution found
```

Synchronous

Original PSO Algorithm

- For particle *i*, let
  - $\vec{x}_i$ be its current position
  - $\vec{v}_i$ be its current velocity
  - $\vec{p}_i$ be the best position that it has found till now
  - $\vec{g}_i$ be the best position that has been found in its neighborhood till now
  - $U(0,\varphi)$ be a sample from a uniform distribution in range $[0,\varphi]$
- Update rules:

$$v_{id} \leftarrow v_{id} + U(0,\varphi_1)(p_{id} - x_{id}) + U(0,\varphi_2)(g_{id} - x_{id})$$

$$x_{id} \leftarrow x_{id} + v_{id}$$

where φ_1 and φ_2 are acceleration coefficients

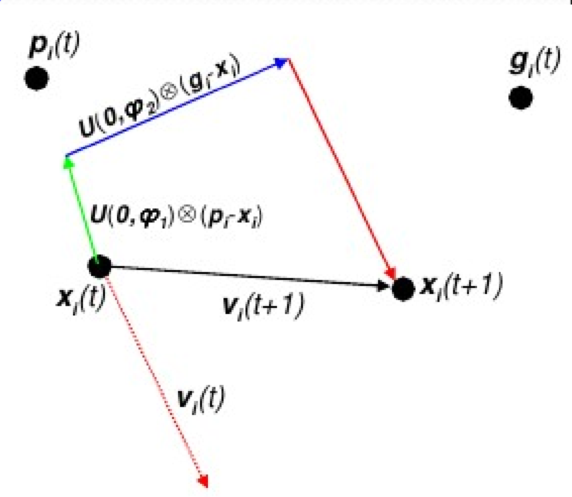
Original PSO Algorithm

$$v_{id} \leftarrow \underbrace{v_{id}} + U(0,\varphi_1)\underbrace{(p_{id} - x_{id})} + U(0,\varphi_2)\underbrace{(g_{id} - x_{id})}$$

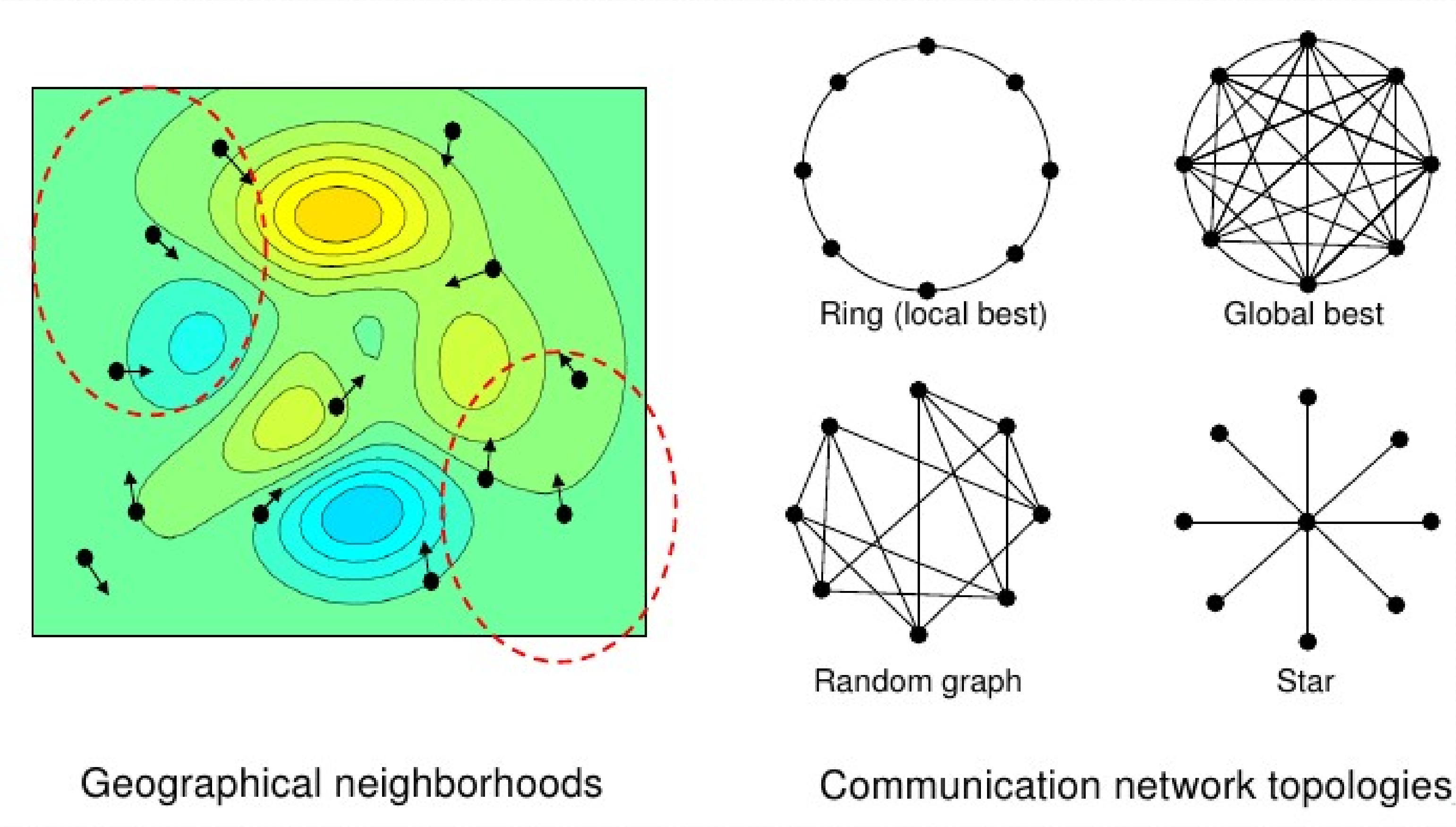
Momentum: pull particle in its current direction

Cognitive component: a tendency to return to its own best solution found so far

Social component: a tendency to move towards the best solution found so far in the neighborhood

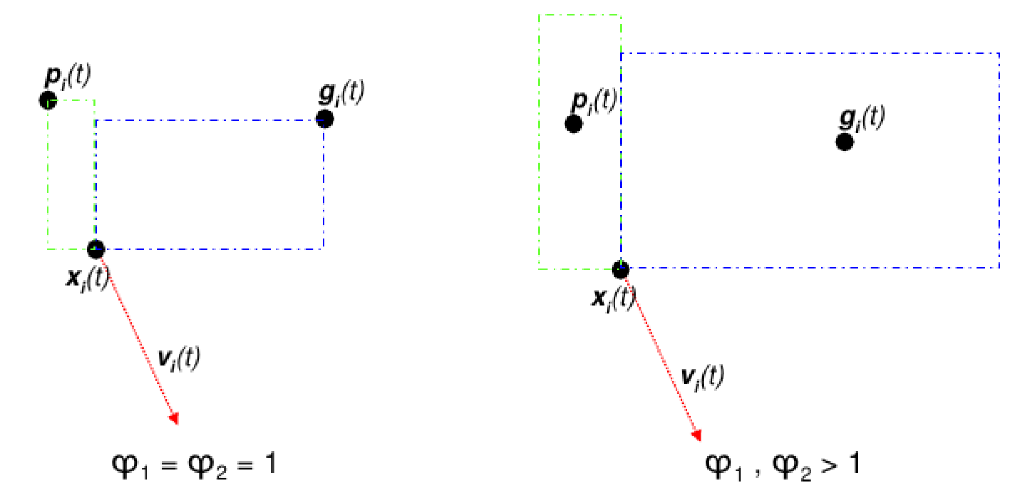


Neighborhoods



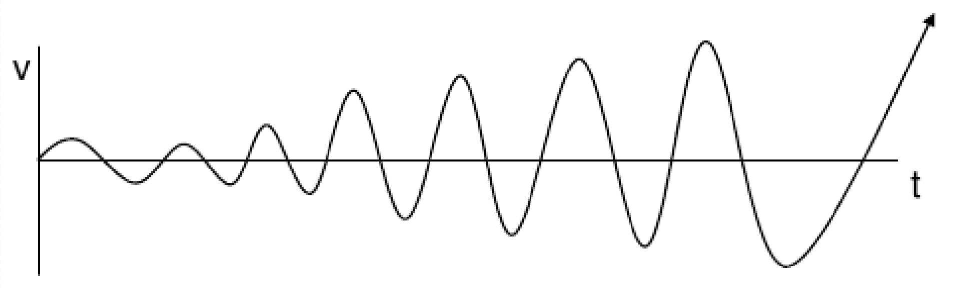
Acceleration Coefficients

- The acceleration coefficients determine the relative influences of the social and cognitive components
  - $\varphi_1 > \varphi_2$: independent particles → beneficial for multimodal problems (many optima)
  - $\varphi_1 < \varphi_2$: collaborating particles → beneficial for unimodal problems (one optimum)



Original PSO Algorithm – Oscillation

- Sufficiently high acceleration coefficients are needed, but can lead to increasing oscillation due to the randomness of the velocity updates (no proof given)



- Basic solution: limit the minimum and maximum velocity

Inertia Weighed PSO

- Velocity update includes inertia weight ω:

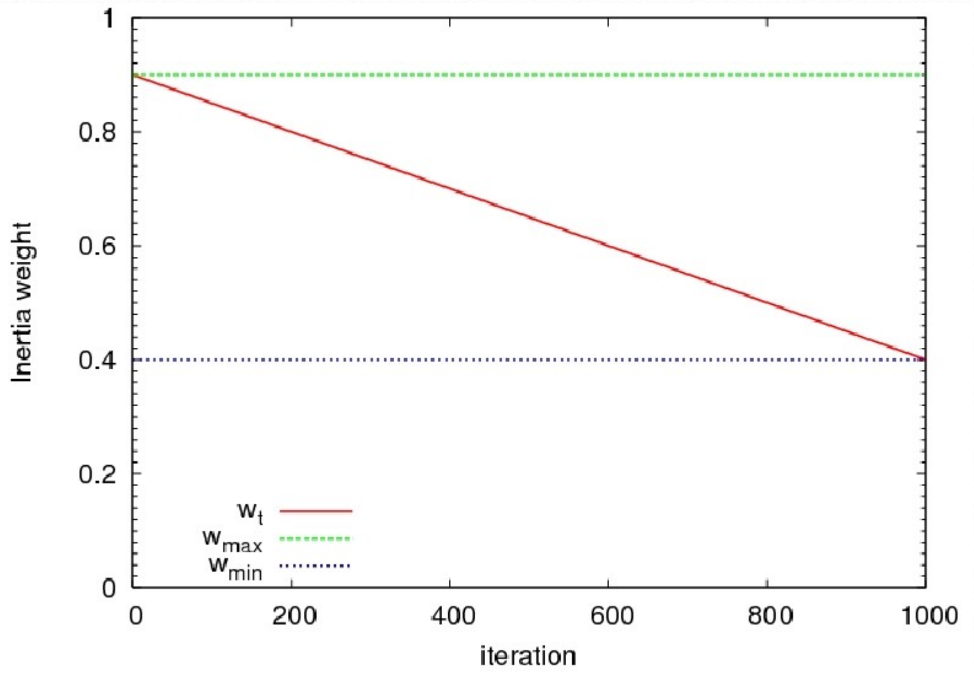
$$v_{id} \leftarrow \omega v_{id} + U(0, \varphi_1)(p_{id} - x_{id}) + U(0, \varphi_2)(g_{id} - x_{id})$$

  - if properly set, strong increases in velocity are avoided
  - $\omega > 1$: particles accelerate; exploration
  - $\omega < 1$: particles decelerate; exploitation
  - Rule-of-thumb settings: $\omega = 0.7298$ and $\phi_1 = \phi_2 = 1.49618$

Shi, Y. Eberhart, R., 'A modified particle swarm optimizer', in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on , pp. 69-73 (1998).

Inertia Weighed PSO

- Eberhart & Shi suggested to decrease inertia over time



Binary/Discrete PSO

- A simple modification for discrete search spaces

$$x_{ij} = \begin{cases} 1 & \text{if } 1/(1 + exp(-v_{ij})) > \tau \\ 0 & \text{otherwise} \end{cases}$$

- Velocity hence expresses a probability that a coordinate is 0/1

- Velocity updates as usual

J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 4104-4108, IEEE Press, 1997

Variants

- Other PSO variants
  - Binary Particle Swarms
  - PSO for noisy fitness functions
  - PSO for dynamical problems
  - PSO for multi-objective optimization problems
  - Adaptive particle swarms
  - PSO with diversity control
  - Hybrids (e.g. with evolutionary algorithms)

Conclusions

- PSO is applicable for the optimization of hard multi-dimensional non-linear functions
- PSO is competitive to other known global optimization methods
- Using the recommended parameter settings it allows for off-the-shelf usage
- Among others, applications for and in:
  - Training of Neural Networks
  - Control applications
  - Video analysis applications
  - Design applications
  - ….