Metaheuristic Optimization Methods: Particle Swarm Optimization

Swarm Intelligence

- Study of collective behavior in decentralized, self-organized systems
- Originated from the study of colonies, or swarms of social organisms
- Collective intelligence arises from interactions

Particle Swarm Optimization (PSO)

- Introduced by Kennedy & Eberhart 1995
- Inspired by social behavior of birds and shoals of fish
- Swarm intelligence-based optimization
- Nondeterministic
- Population-based optimization
- Performance comparable to Genetic algorithms

PSO vs. GA

Similarity

- Start with a group of a randomly generated population
- Use fitness values to evaluate the population
- Update population and use stochastic techniques
- Neither guarantee success

Dissimilarity

- PSO has no evolution operators (e.g. crossover)
- Particles update themselves with an internal velocity
- Particles have memory
- PSO works best (naturally) on continuous space

Advantages

- PSO is easy to implement with few parameters to adjust
- PSO tends to converge to 'best' solution quickly

Particle Swarm Optimization

- Simple algorithms for movement
- Movement is influenced by three factors:
    1. **Inertia**
    2. **Cognitive influence**
    3. **Social influences**
- Particles try to improve themselves and often achieve this by (2) and (3)
- Overall population moves towards “better” areas of the problem space

Particle Swarm Optimization

- Swarm: a set of particles (S)
- Particle: a potential solution
  - Position, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$
  - Velocity , $V_i = (v_{i1}, v_{i2}, \ldots, v_{in}) \in \mathbb{R}^n$
- Each particle maintains
  - Individual best position: $P_i = (p_{i1}, p_{i2}, \ldots, p_{in}) \in \mathbb{R}^n$

    $pbest_i = f(P_i)$
- Swarm maintains its global best: $P_g \in \mathbb{R}^n$

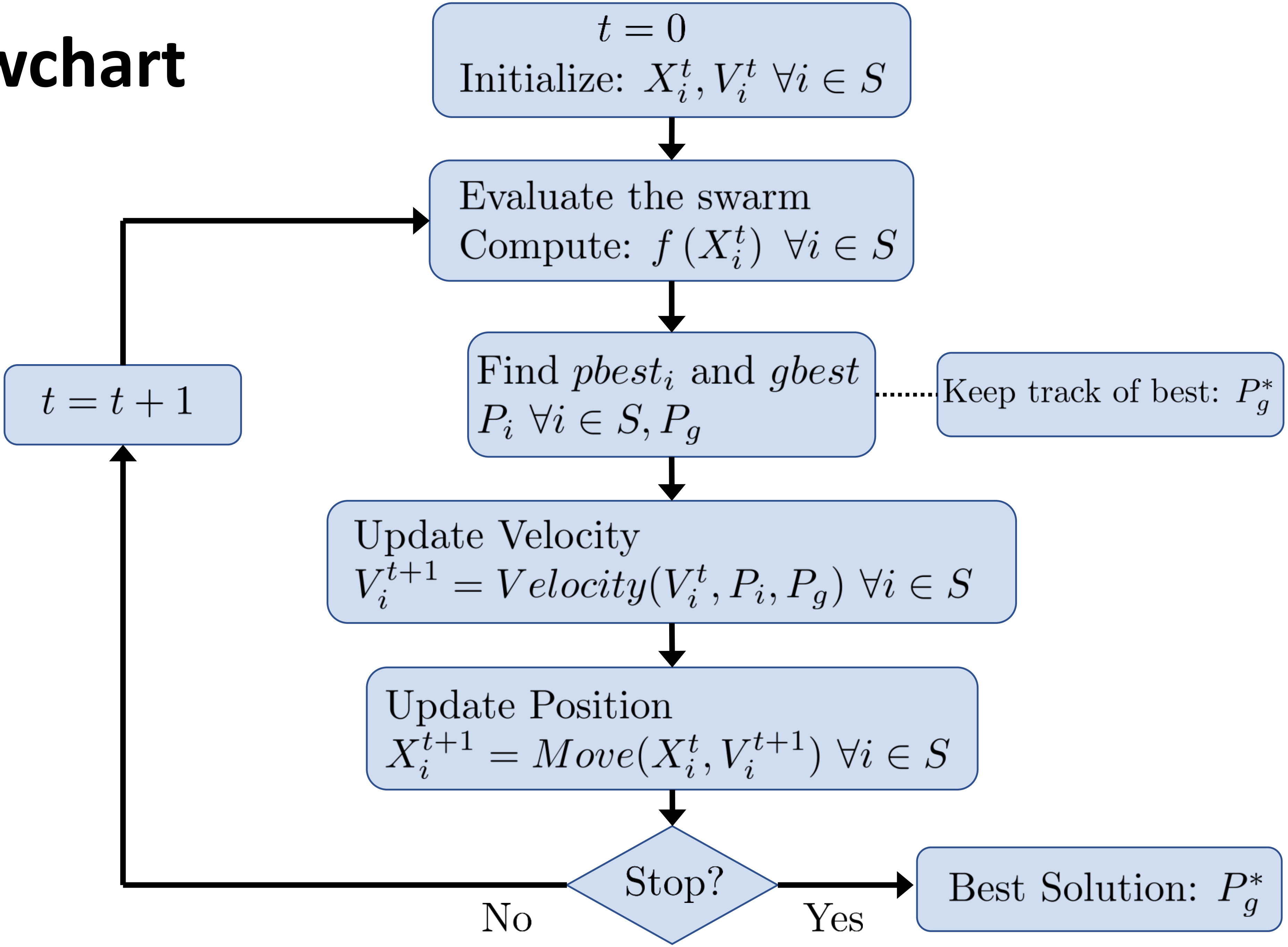
  $gbest = f(P_g)$

PSO Algorithm

Basic process:

1. Initialize the swarm from the solution space
2. Evaluate fitness of each particle
3. Update individual and global bests
4. Update velocity and position of each particle
5. Go to step 2, and repeat until termination condition

PSO flowchart



Particle Swarm Optimization

- **Original velocity update equation:**

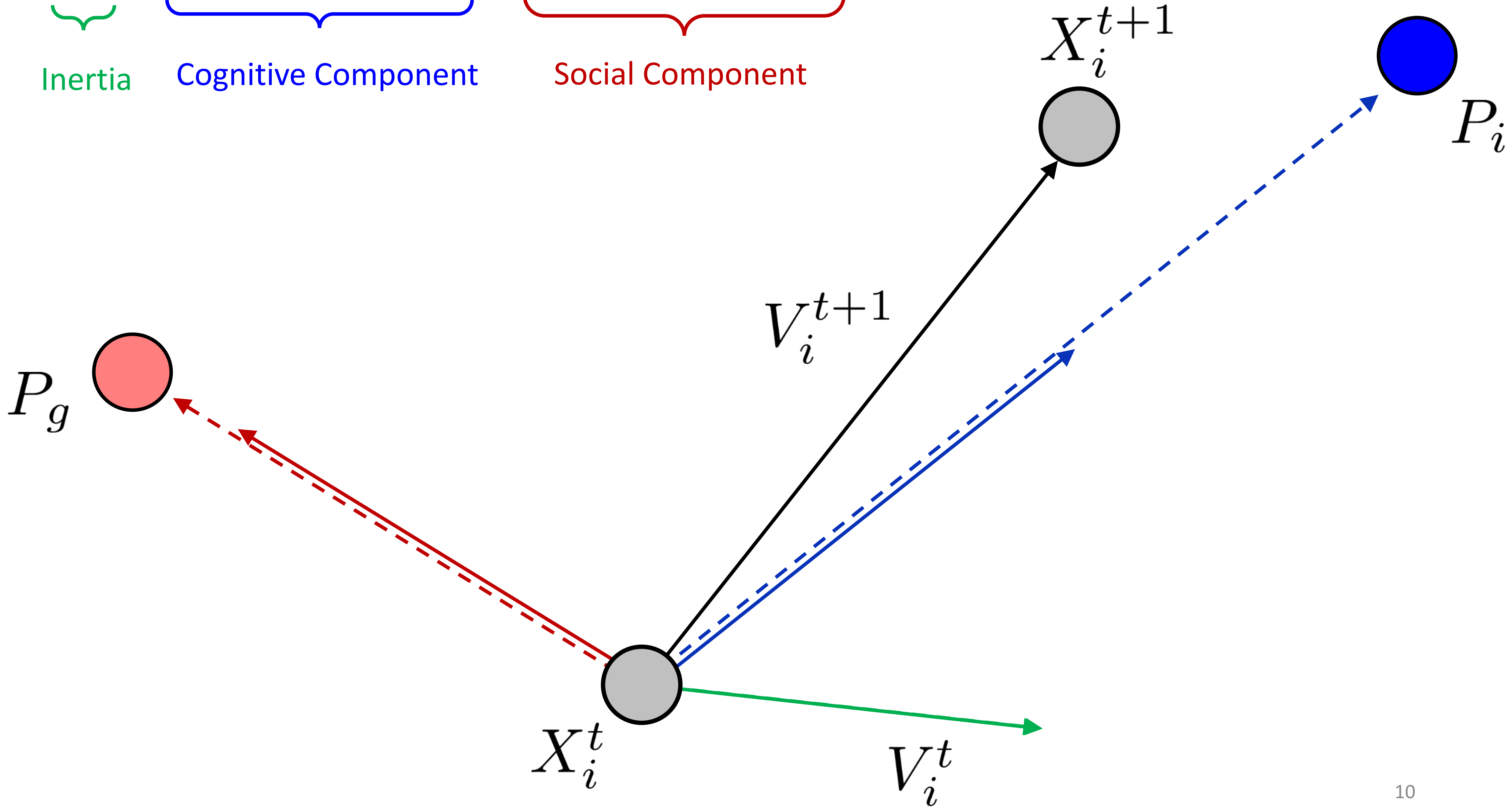
$$V_i^{t+1} = \underbrace{V_i^t}_{\text{Inertia}} + \underbrace{\varphi_1 . r_1 (P_i - X_i^t)}_{\text{Cognitive Component}} + \underbrace{\varphi_2 . r_2 (P_g - X_i^t)}_{\text{Social Component}}$$

where $r_1, r_2 \sim U(0,1)$

and acceleration constants φ_1, φ_2

- **Position Update:** $X_i^{t+1} = X_i^t + V_i^{t+1}$

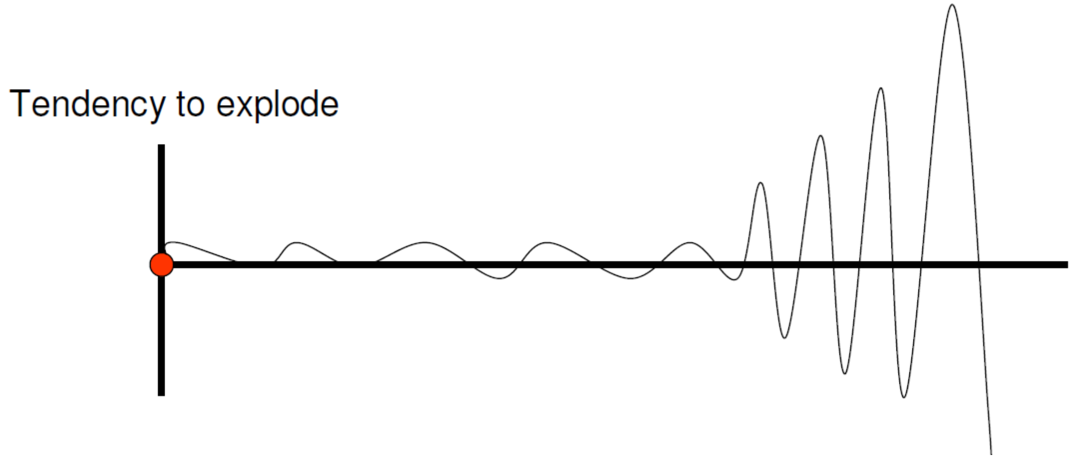
$$V_i^{t+1} = \underbrace{V_i^t}_{\text{Inertia}} + \underbrace{\varphi_1 . r_1 (P_i - X_i^t)}_{\text{Cognitive Component}} + \underbrace{\varphi_2 . r_2 (P_g - X_i^t)}_{\text{Social Component}}$$



PSO Algorithm - Parameters

- Acceleration constants: φ_1, φ_2
  - small values limit the movement of the particles
  - large values : tendency to explode toward infinity
  - In general,

$$\varphi_1 + \varphi_2 \leq 4$$

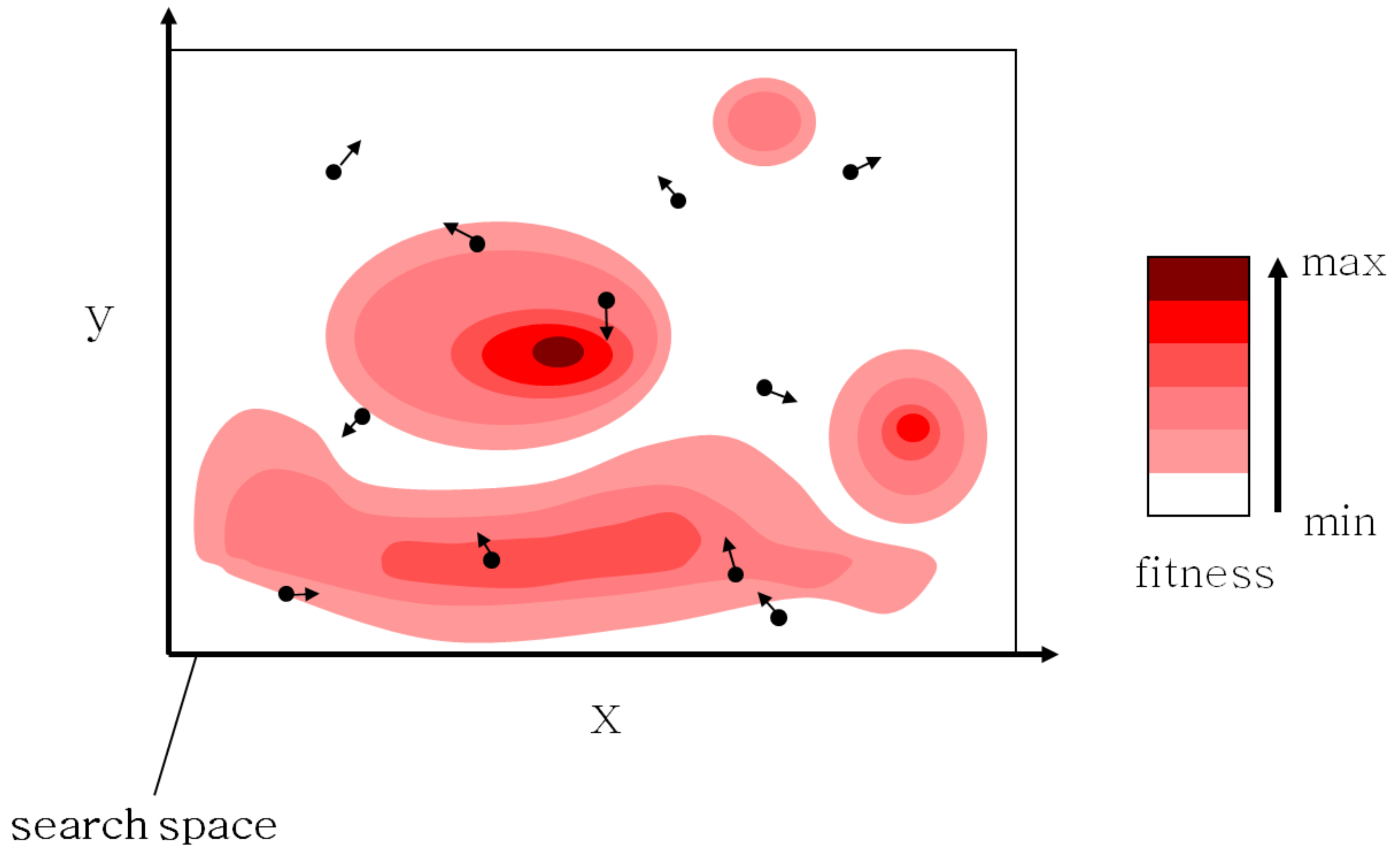


- Maximum velocity

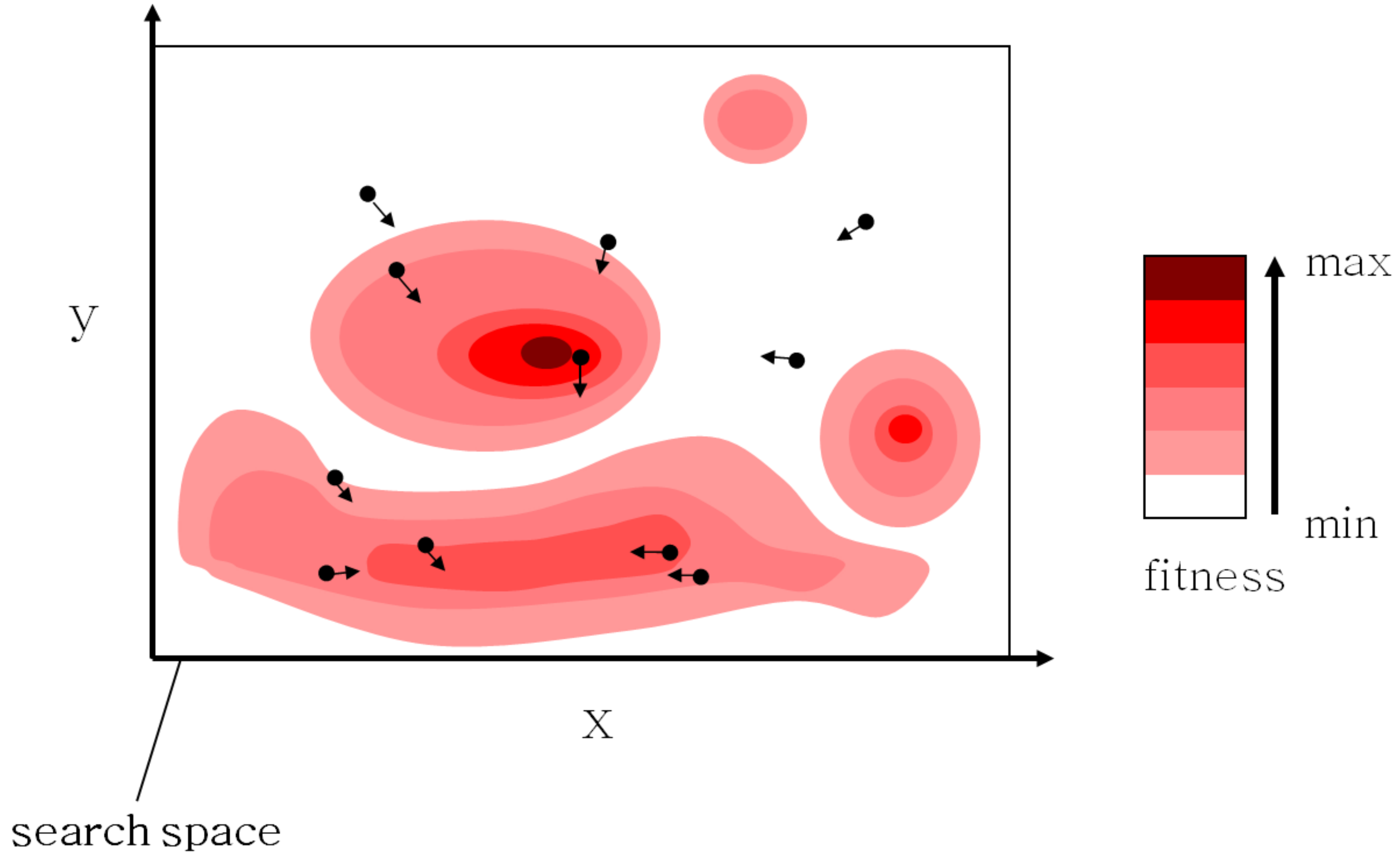
$$\text{If } v_{ij} > v_{\max} \text{ then } v_{ij} = v_{\max}$$

$$\text{else if } v_{ij} < -v_{\max} \text{ then } v_{ij} = -v_{\max}$$

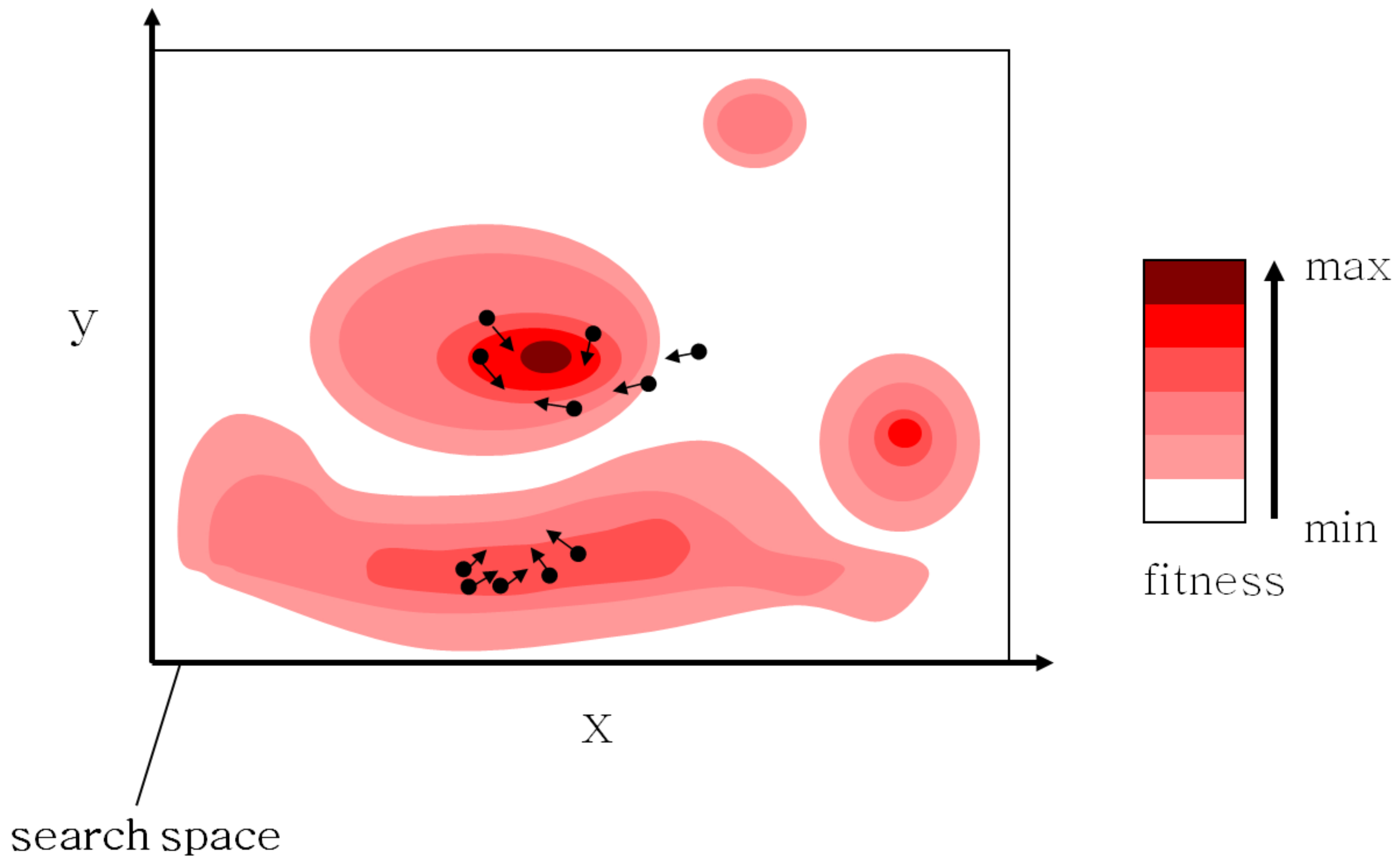
PSO: 2d example animation



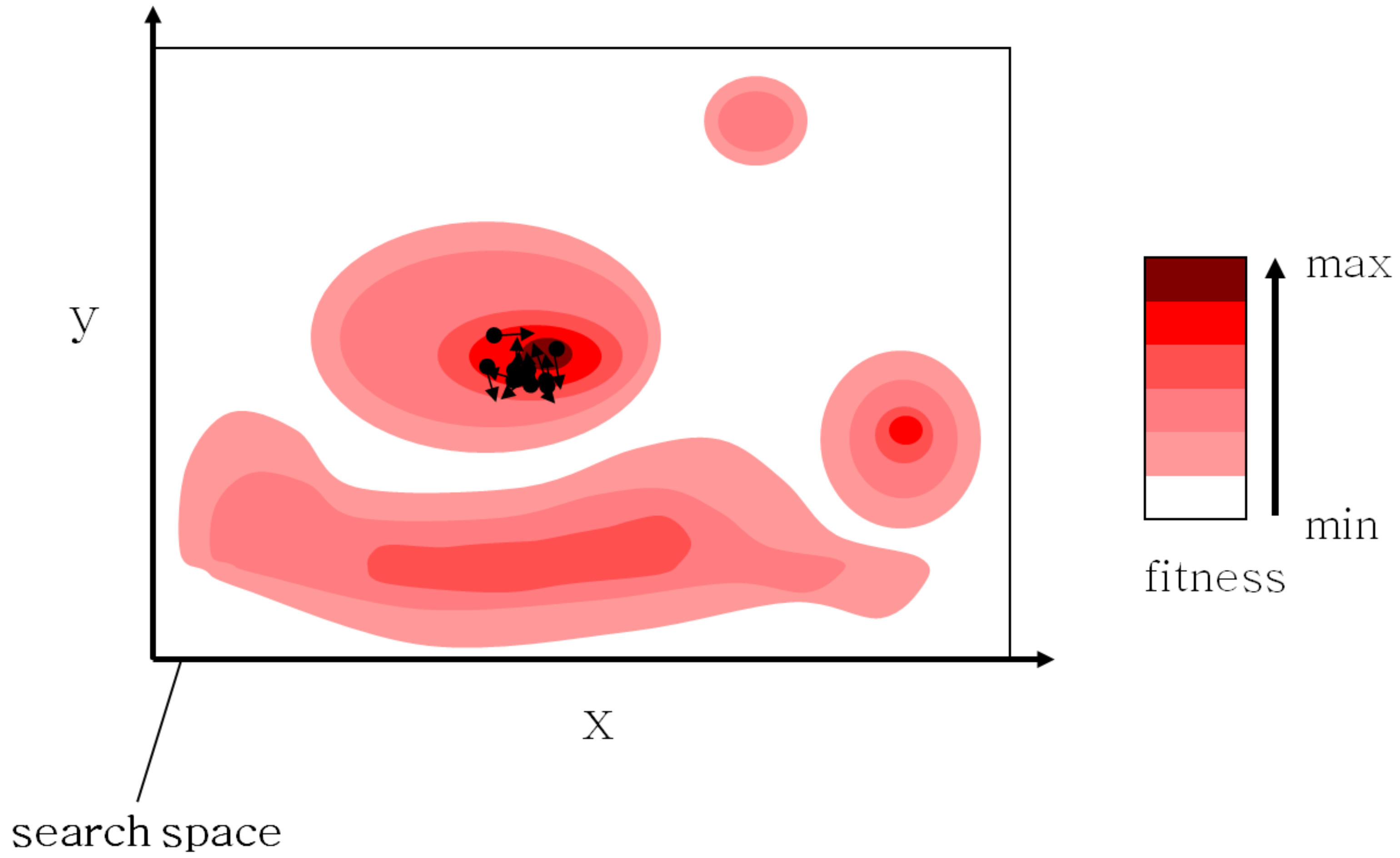
PSO: 2d example animation



PSO: 2d example animation



PSO: 2d example animation



Rate of Convergence Improvement

- Inertia weight:

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$

- Scales the previous velocity
- Control search behavior
  - High values → exploration
  - Low values → exploitation

PSO with Inertia weight

Can be decreased over time

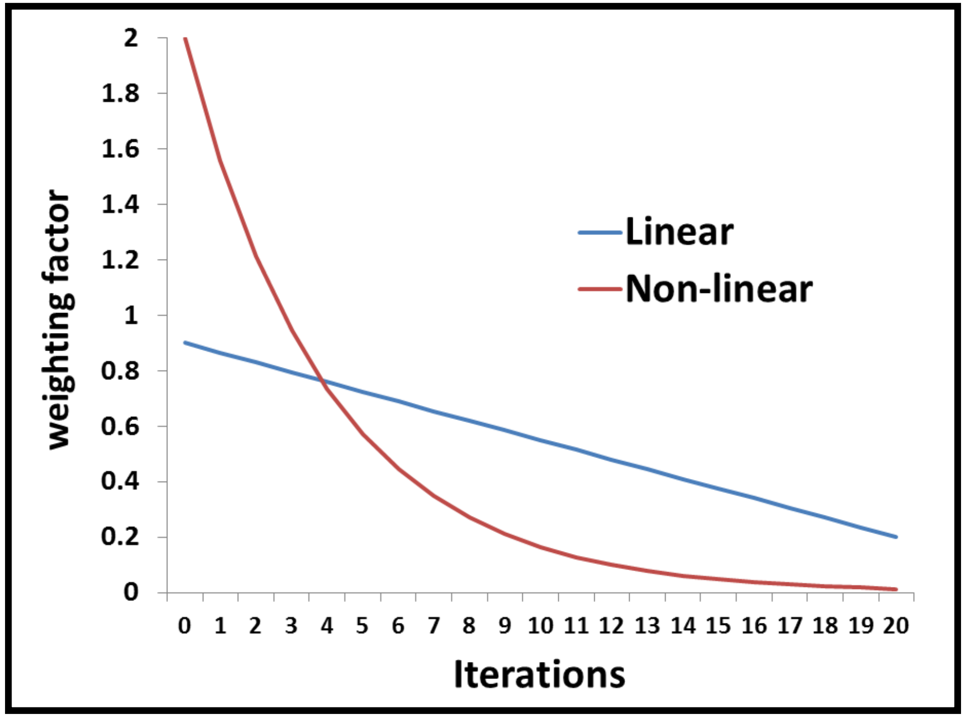
- Linearly:

$$w(t) = w_{\max}\left(w_{\max} - w_{\min}\right)\frac{t}{t_{\max}}$$

e.g. 0.9 to 0.2

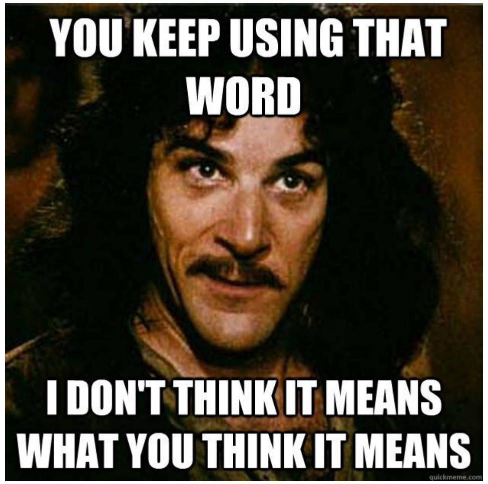
- Nonlinearly, e.g.:

$$w(t) = \frac{A}{e^{kt}}$$



- main disadvantage:
  - once the inertia weight is decreased, the swarm loses its ability to search new areas (can not recover its exploration mode).

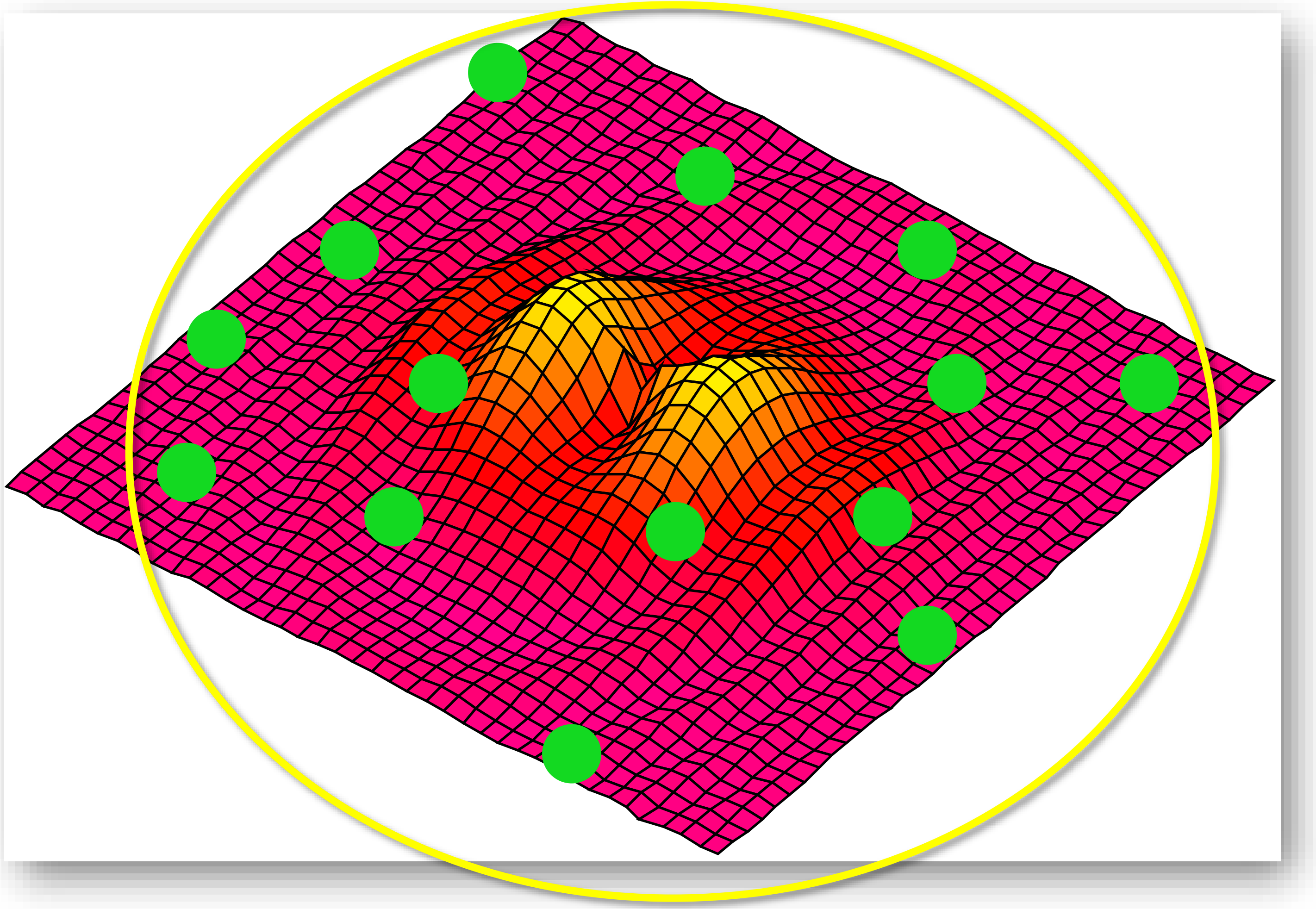
PSO Neighborhoods



- The neighborhood concept in PSO is not the same as the one used in other metaheuristics search
- Neighborhoods do not depend on particle position in the search space, but on "external" relationships that are not problem dependent
- The original PSO had two types of neighborhoods defined: **global** and **local**

PSO Neighborhoods

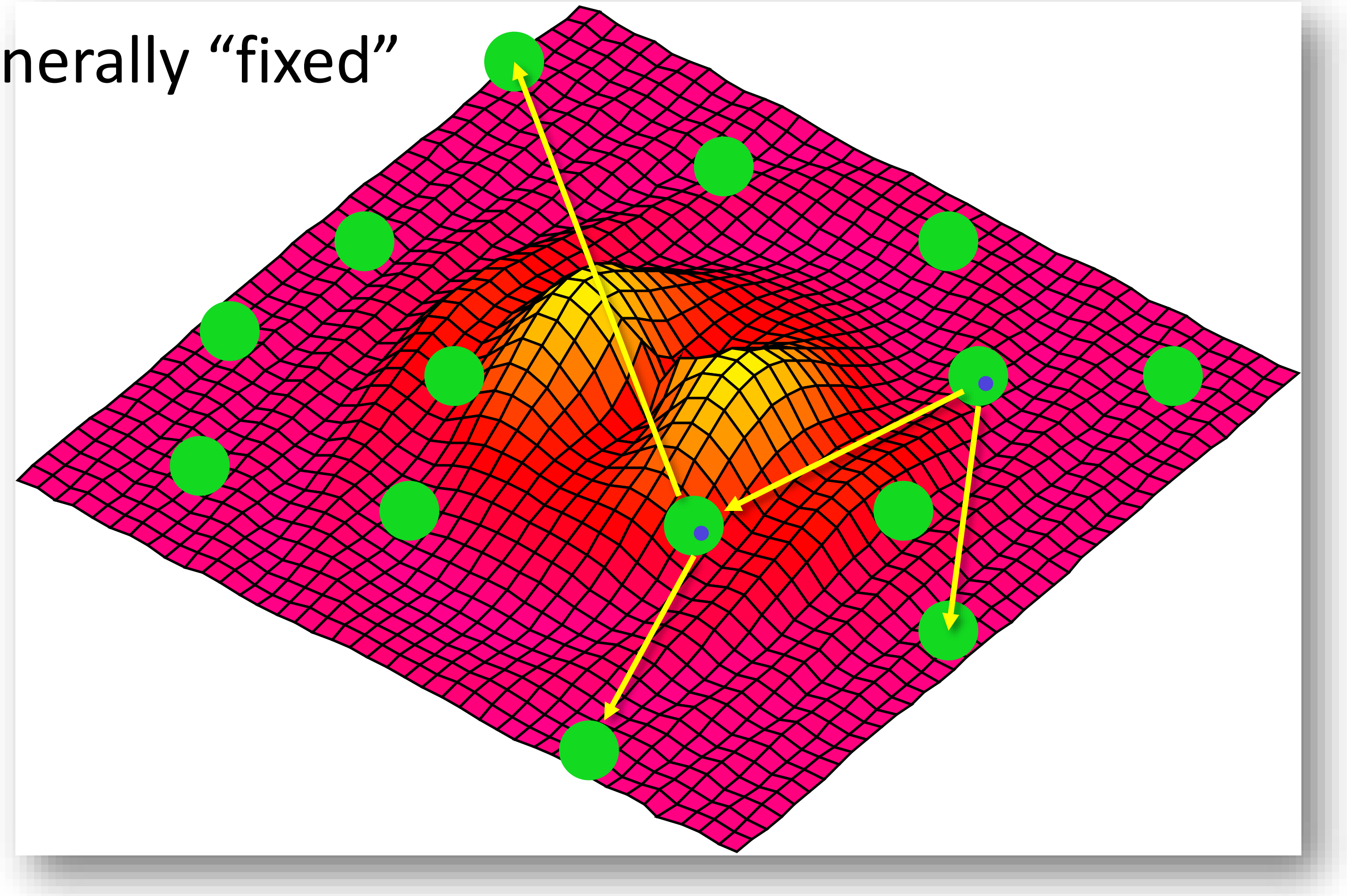
Global neighborhood → use “gbest”



PSO Neighborhoods

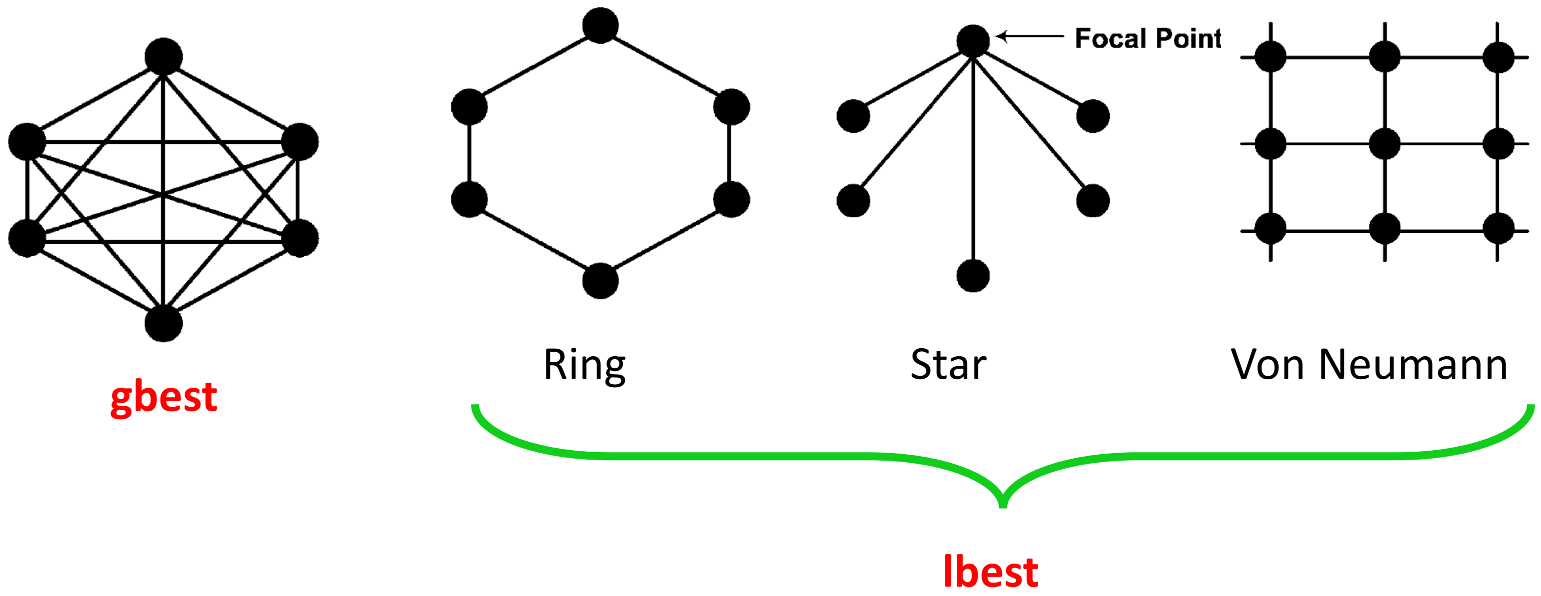
Local neighborhood → use "lbest" as social component

Neighbors are a subset of swarm and generally "fixed"



Swarm Topologies

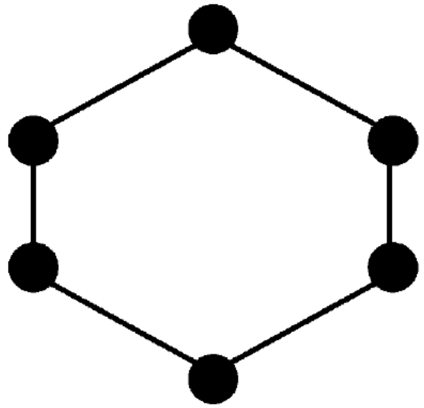
- Two general types of neighborhoods:
  - Global best (*gbest*) : fully connected network
  - Local best (*lbest*) : according to a topology



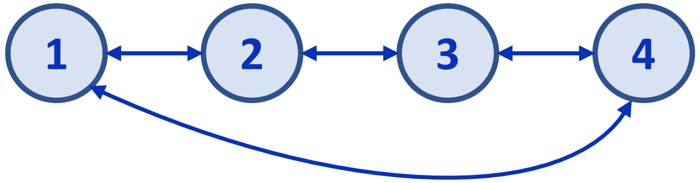
Toscano-Pulido, G., Reyes-Medina, A.J., Ramírez-Torres, J.G.: A statistical study of the effects of neighborhood topologies in particle swarm optimization. SCI, vol. 343, pp. 179–192 (2011)

Swarm topologies

Ring



- each particle has $k/2$ neighbors on each "side"
- usually $k = 2$
- e.g.,

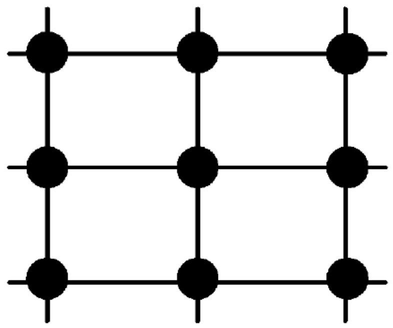


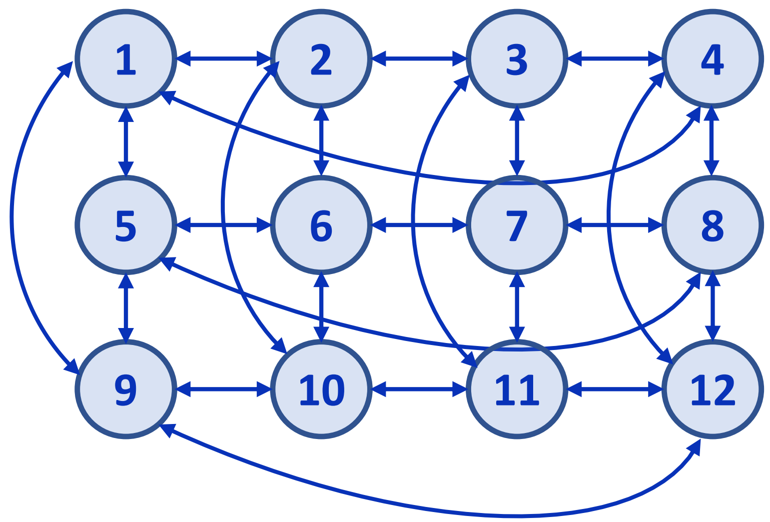
Particle 2 is neighbors with particles 1 and 3

Particle 3 is neighbors with particles 2 and 4

Swarm topologies

- von Neumann
  - basically a ring with 2 neighbors on each side
  - each particle has 4 neighbors



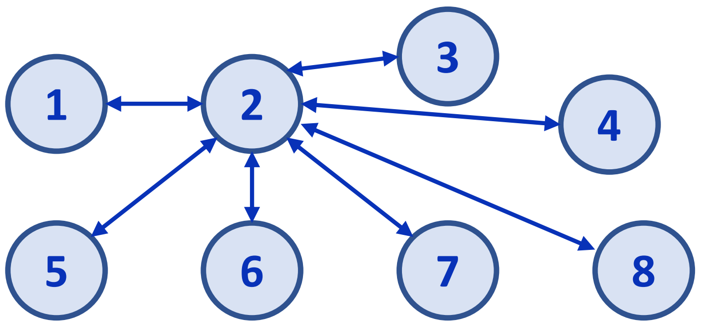


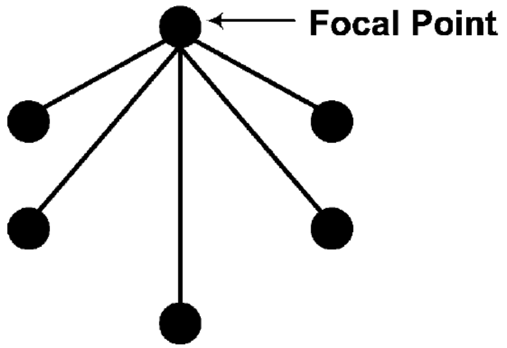
Particle 2 is neighbors with particles 1, 3, 6, and 10

Particle 3 is neighbors with particles 2, 4, 7, and 11

Swarm topologies

- Star
  - All particles are connected to one “central” particle
  - All “information” passes through the central node





Particle 2 is neighbors with particles all other particles

Particle 3 is neighbors with particle 2

PSO Variants

- Discrete PSO: discrete or binary space instead of continuous space
- Hybrid PSO: incorporate capabilities of evolutionary techniques, e.g.,
  - GA-PSO: add “natural selection” and breeding to PSO
  - EPSO: stochastic tournament selection, parameters and “gbest” mutate
- Adaptive PSO
  - Swarm size is dynamic
  - Parameters adapt (e.g., cognitive component influence increases if “pbest” is very good)

Schwefel Function

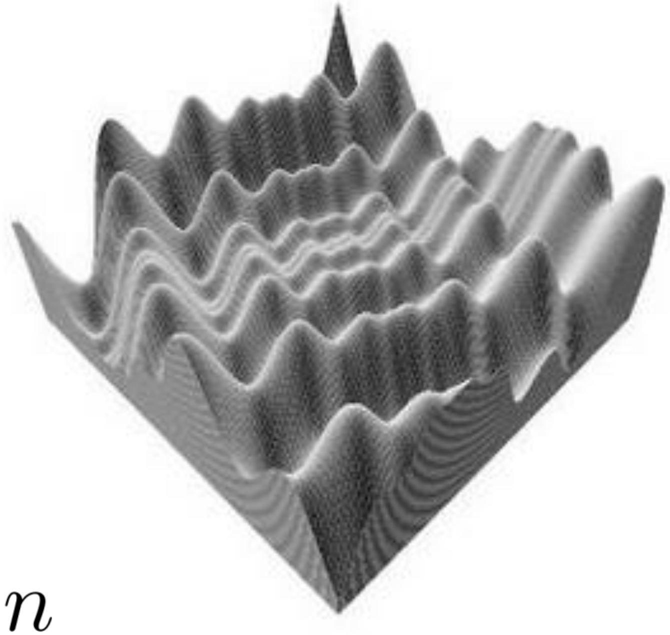
$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

- Number of variables: n
- Several local minima.
- Optimal solution: known
- Search domain:

$$-500 \leq x_i \leq 500 \text{ for } i = 1 \ldots n$$

- Global minima:

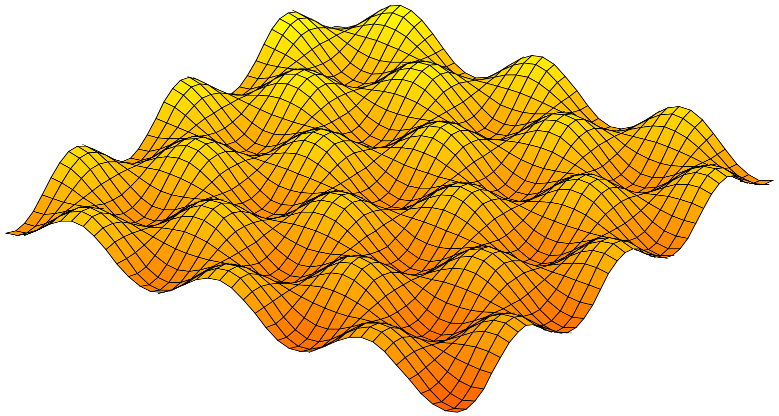
$$\mathbf{x}^* = (420.9687, \ldots, 420.9687)$$
$$f(\mathbf{x}^*) = -418.9829n$$



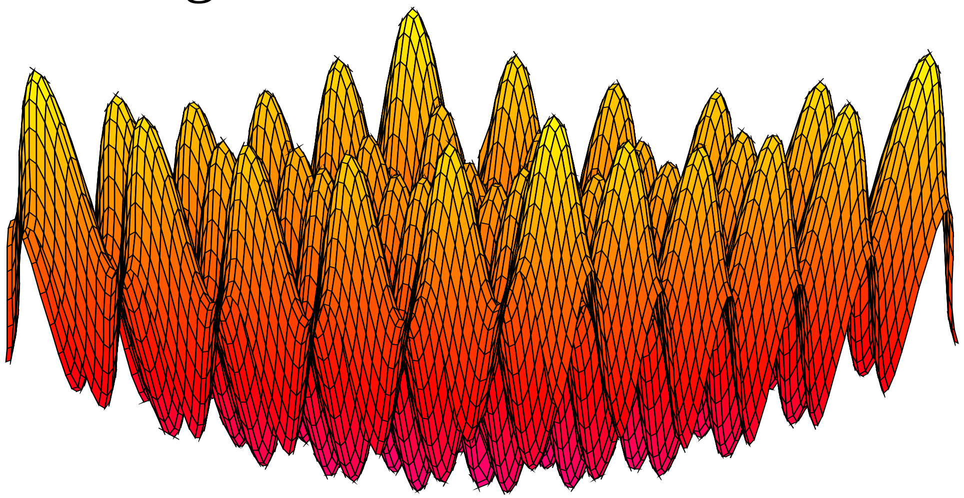
2D Schwefel Problem

Some functions often used for testing real-valued optimization algorithms

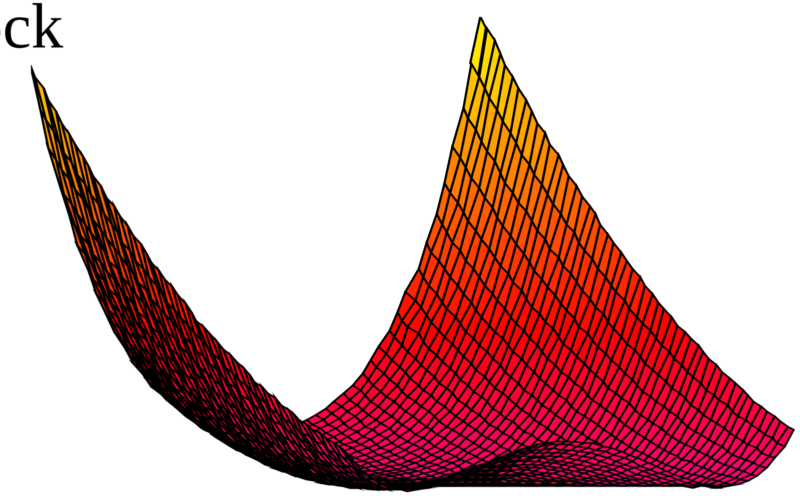
Griewank



Rastrigin



Rosenbrock



… and some typical results

Optimum=0, dimension=30

Best result after 40 000 evaluations

30D function	PSO	Evolutionary algorithm
Griewank [±300]	0.003944	0.4033
Rastrigin [±5]	82.95618	46.4689
Rosenbrock [±10]	50.193877	1610.359