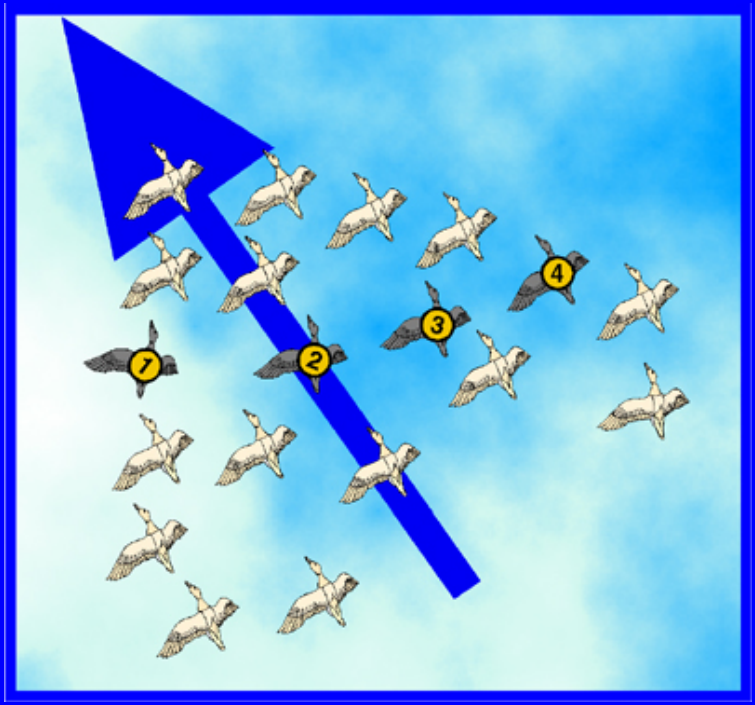
Particle Swarm Optimization (PSO)



C++ Programming
2019 Fall

Haitao Yuan
Email: htyuan@bjtu.edu.cn

Final Project

- Implement PSO with C++
  - Higher scores if you implement advanced PSO, *e.g.*, Genetic Learning Particle Swarm Optimization (Optional)
- Solve several optimization problems
  - At most 15 problems
  - Solve more problems, get higher scores
- Required materials

  Document to describe your final project (!!!!!!Important)
  - Analysis of your codes
  - Solved problems
  - Figures of final results
  - Comments about of PSO

  Slides (PPTs) for presentation (!!!!!!Important)

  Source codes

Final Project

- Deadline (!!!!!!Important)
  - 18:00 pm on Nov. 6, 2019
- Email (!!!!!!Important)
  - c_p_pl@163.com
  - Confirmation messages if I receive your materials
  - Resend them if you do not receive my messages

Final Project

- 15 Optimization problems (Minimization)

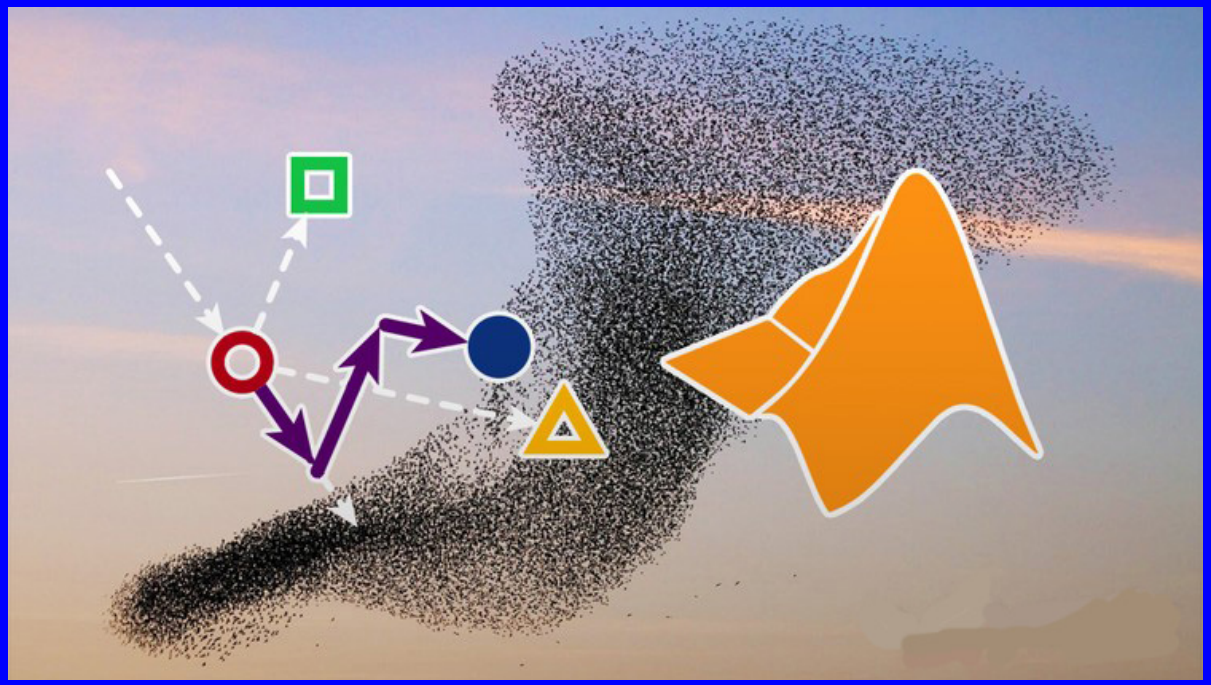
Table 1 Benchmark functions

ID	Equation	Lower	Upper	Dimension	type
F1	$f(x)=\sum_{i=1}^{n} x_i^2$	-100	100	10	Unimodal
F2	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \Pi_{i=1}^{n} \lvert x_i\rvert$	-10	10	10	Unimodal
F3	$f(x)=\sum_{i=1}^{n} (\sum_{j-1}^{i} x_i)^2$	-100	100	10	Unimodal
F4	$f(x)=max_i\{\lvert x_i\rvert, 1 \leqslant i \leqslant n\}$	-100	100	10	Unimodal
F5	$f(x)=\sum_{i=1}^{n-1} [100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	-30	30	10	Unimodal
F6	$f(x)=\sum_{i=1}^{n} (\lvert x_i+0.5\rvert)^2$	-100	100	10	Unimodal
F7	$f(x)=\sum_{i=1}^{n} ix_i^4+random[0,1]$	-1.28	1.28	10	Unimodal
F8	$f(x)=\sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i\rvert})$	-500	500	10	Multimodal
F9	$f(x)=\sum_{i=1}^{n} [x_i^2-10cos(2\pi x_i)+10]$	-5.12	5.12	10	Multimodal
F10	$f(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i))+20+e$	-32	32	10	Multimodal
F11	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\Pi_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})+1$	-600	600	10	Multimodal
F12	$f(x)=\frac{\pi}{n}\{10sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$	-50	50	10	Multimodal
	$u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m, & x_i>a \\ 0, & -a\leqslant x_i\leqslant a \\ k(-x_i-a)^m, & x_i<-a \end{cases}$				
F13	$f(x)=0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	-50	50	10	Multimodal
F14	$f(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	-65.536	65.536	2	Multimodal*
F15	$f(x)=(\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}]^2$	-5	5	4	Multimodal*

* Fixed-dimension multimodal

Summary

- Particle Swarm Optimization (PSO)
  - Origins
  - Concept
  - PSO Algorithm
- Example

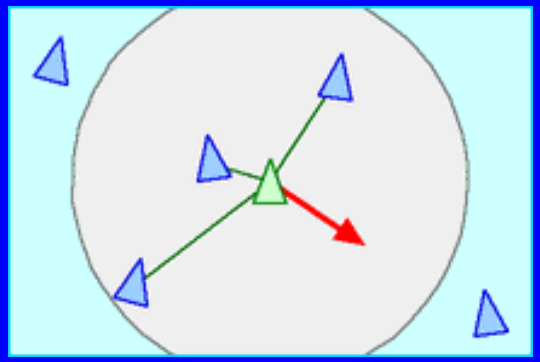


Introduction to PSO: **Origins**

- Inspired from the nature: social behavior and dynamic movements with communications of insects, birds and fish

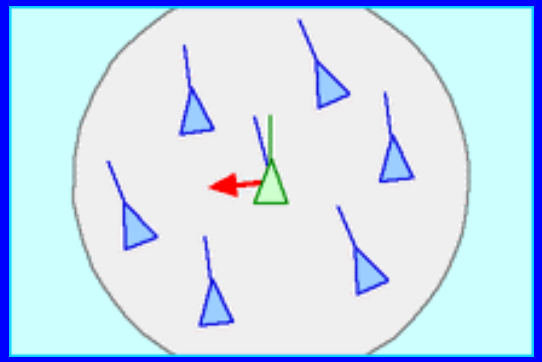
Introduction to PSO: **Origins**

- In 1986, Craig Reynolds described this process in 3 simple behaviors:



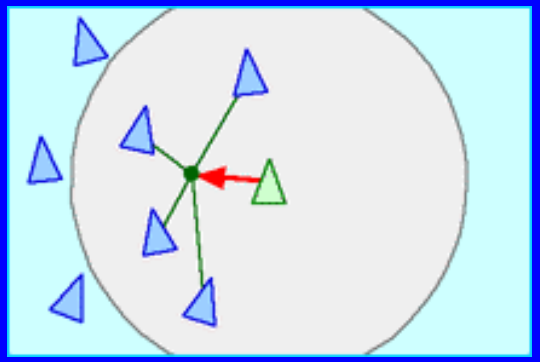
Separation

avoid crowding local flockmates



Alignment

move towards the average heading of local flockmates



Cohesion

move toward the average position of local flockmates

Introduction to PSO: **Origins**

- Application to optimization: Particle Swarm Optimization
- Proposed by James Kennedy & Russell Eberhart (1995)
- Combines self-experiences with social experiences

Introduction to PSO: **Concept**

- Uses a number of agents (**particles**) that constitute a swarm moving around in the search space looking for the best solution

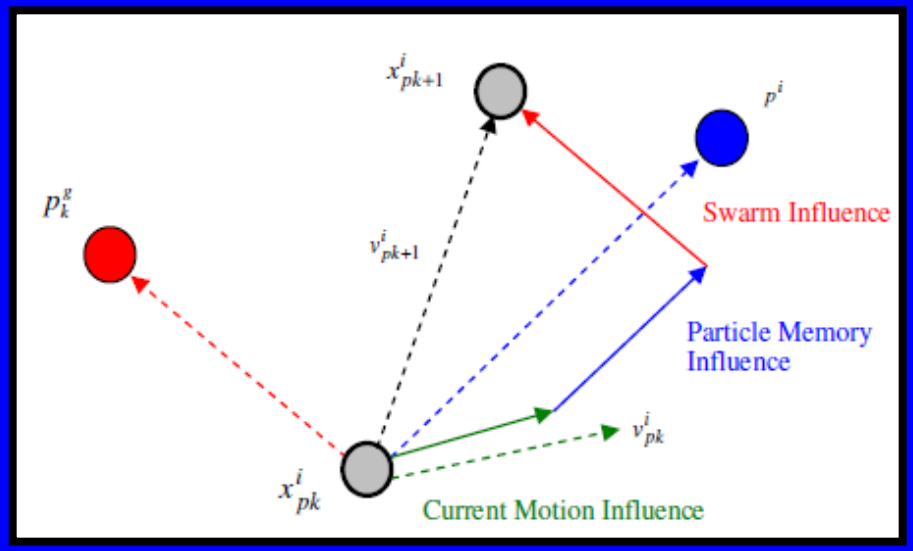
- Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles

Introduction to PSO: **Concept**

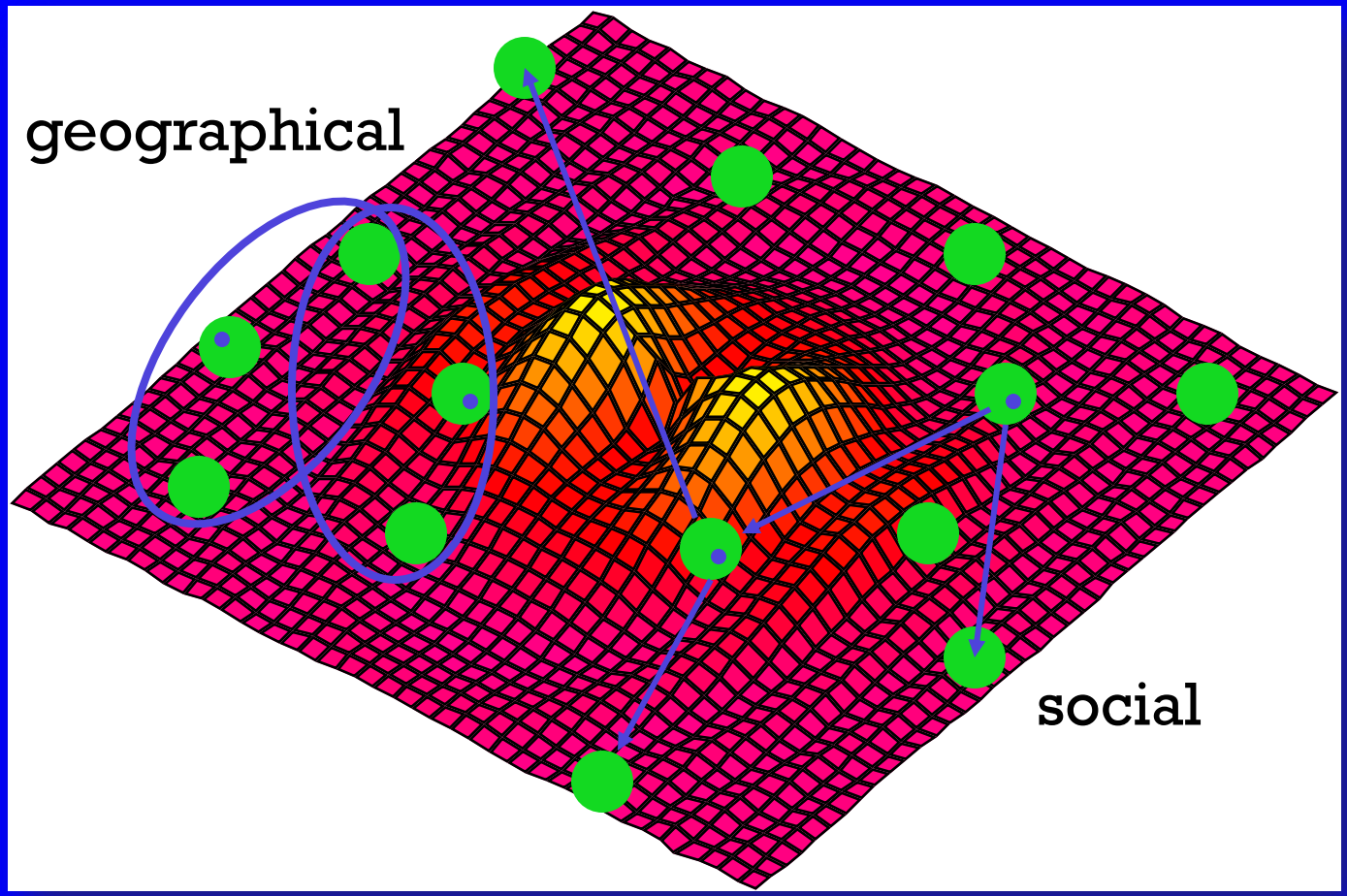
- Collection of flying particles (swarm) - Changing solutions
- Search area - Possible solutions
- Movement towards a promising area to get the global optimum
- Each particle keeps track:
  - its best solution, personal best, *pbest*
  - the best value of any particle, global best, *gbest*

Introduction to PSO: **Concept**

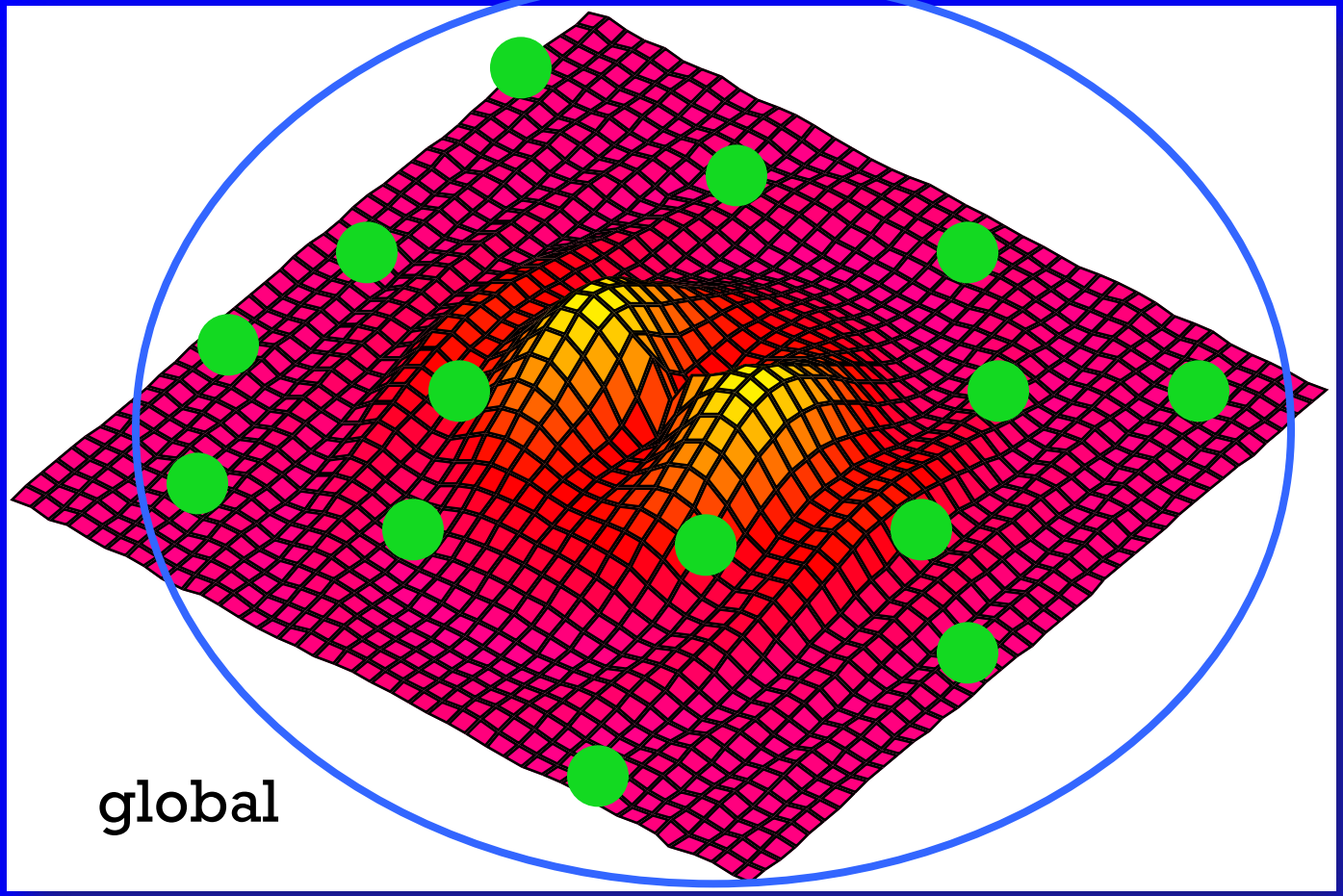
- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and *pbest*
  - the distance between its current position and *gbest*



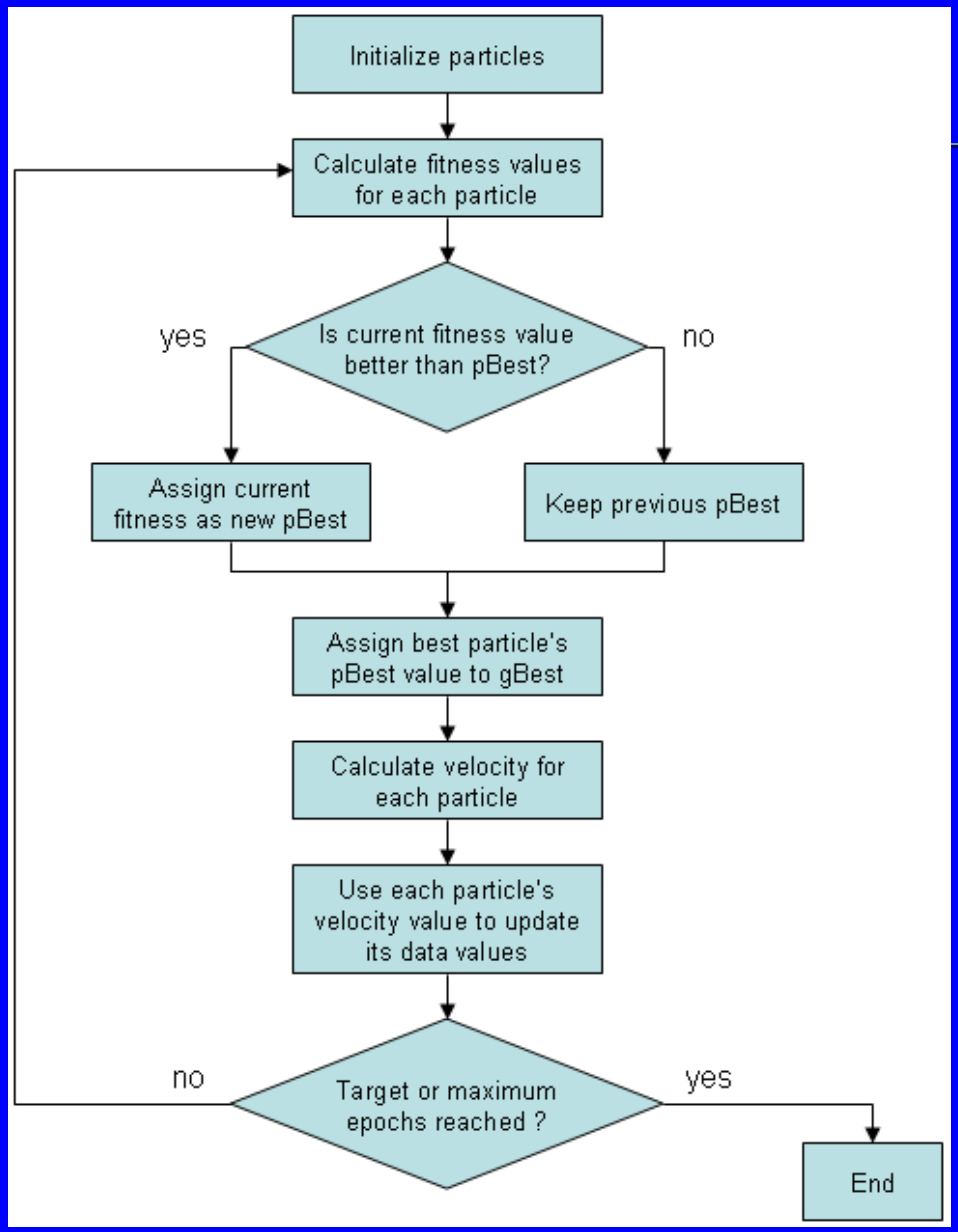
Introduction to PSO:
Algorithm - Neighborhood



Introduction to PSO:
Algorithm - Neighborhood



Introduction to PSO: **Flow diagram**



Introduction to PSO: **Algorithm - Parameters**

- Algorithm parameters
  - A : Population of agents
  - p_i : Position of agent a_i in the solution space
  - f : Objective function
  - v_i : Velocity of agent a_i
  - $V(a_i)$: Neighborhood of agent a_i (fixed)
- The neighborhood concept in PSO is not the same as the one used in other meta-heuristic search, since in PSO each particle's neighborhood never changes (fixed)

Introduction to PSO: **Algorithm**

```
[gBest] = PSO()
P = Particle_Initialization();
For i=1 to it_max
  For each particle p in P do
    fp = f(p);
    If fp is better than f(pBest)
        pBest = p;
    end
  end
  gBest = best p in P;
  For each particle p in P do
      v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);
      p = p + v;
  end
end
Output gBest
```

Introduction to PSO: **Algorithm**

- Particle update rule

$$p = p + v$$

  - with

$$v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$$

  - where

- p: position of a particle
- v: path direction (velocity)
- c_1: weight of local information
- c_2: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

Introduction to PSO: **Algorithm - Parameters**

- Number of particles is usually between 10 and 50
- c_1 is the importance of personal best value
- c_2 is the importance of global best value
- Usually $c_1 + c_2 = 4$ (empirically chosen value)
- If velocity is too low → algorithm too slow
- If velocity is too high → algorithm too unstable

Introduction to PSO: **Algorithm**

1. Create a 'population' of agents (particles) uniformly distributed over X
2. Evaluate each particle's position according to the objective function
3. If a particle's current position is better than its previous best position, update it
4. Determine the best particle (according to the particle's previous best positions)

Introduction to PSO: **Algorithm**

5. Update particles' velocities:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\textit{inertia}} + \underbrace{c_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{\textit{personal influence}} + \underbrace{c_2 \mathbf{U}_2^t (\mathbf{gb}^t - \mathbf{p}_i^t)}_{\textit{social influence}}$$

6. Move particles to their new positions:

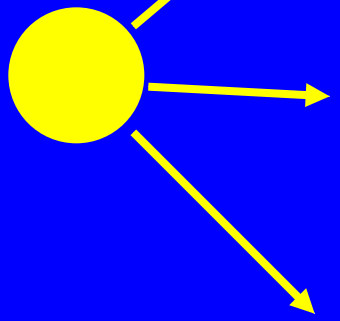
$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

7. Go to step 2 until stopping criteria are satisfied

Introduction to PSO: **Algorithm**

Particle's velocity:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{inertia}} + \underbrace{c_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{\text{personal influence}} + \underbrace{c_2 \mathbf{U}_2^t (\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{social influence}}$$



1. Inertia

- Makes the particle move in the same direction and with the same velocity

2. Personal Influence

- Improves the individual
- Makes the particle return to a previous position, better than the current
- Conservative

3. Social Influence

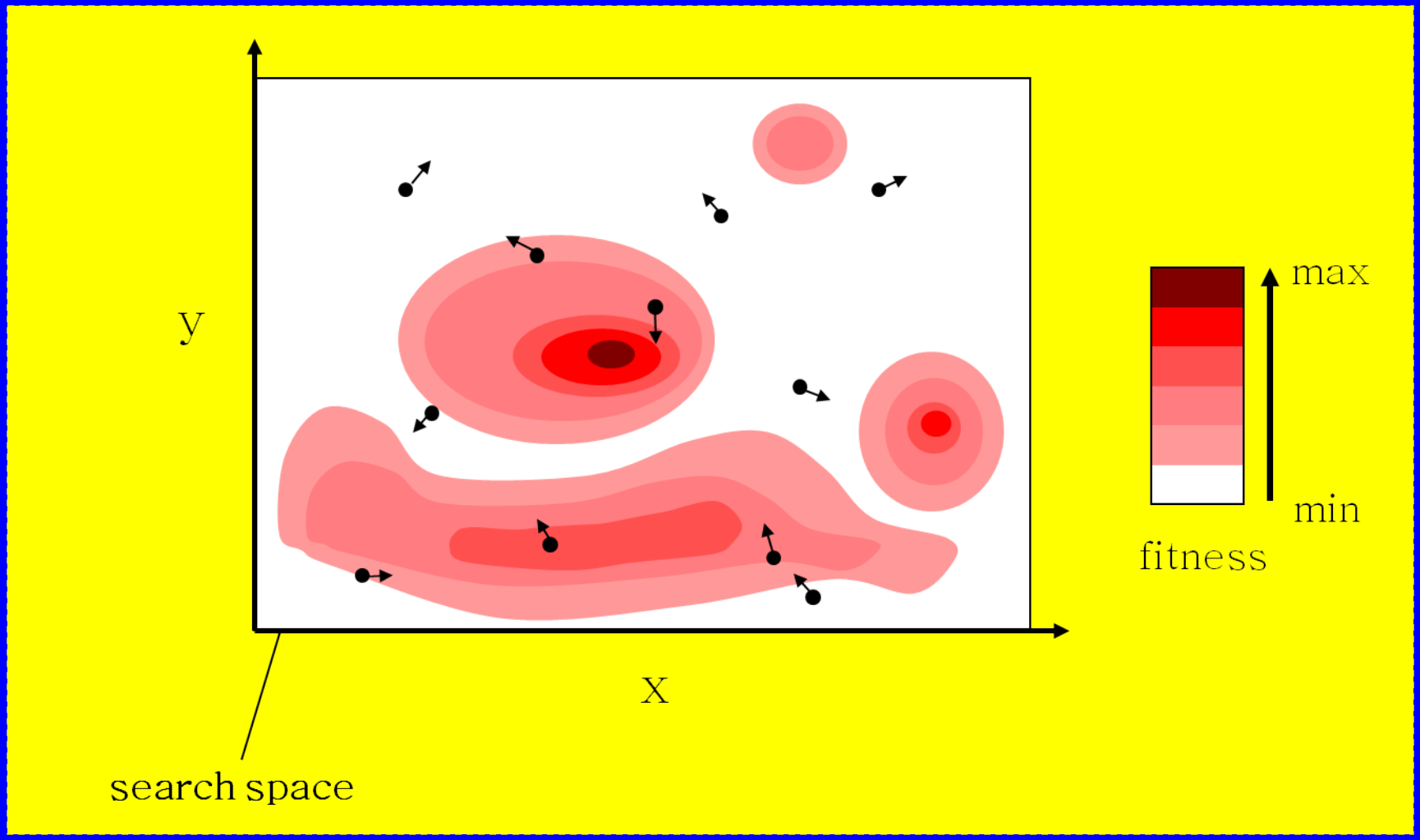
- Makes the particle follow the best neighbor' direction

Introduction to PSO: **Algorithm**

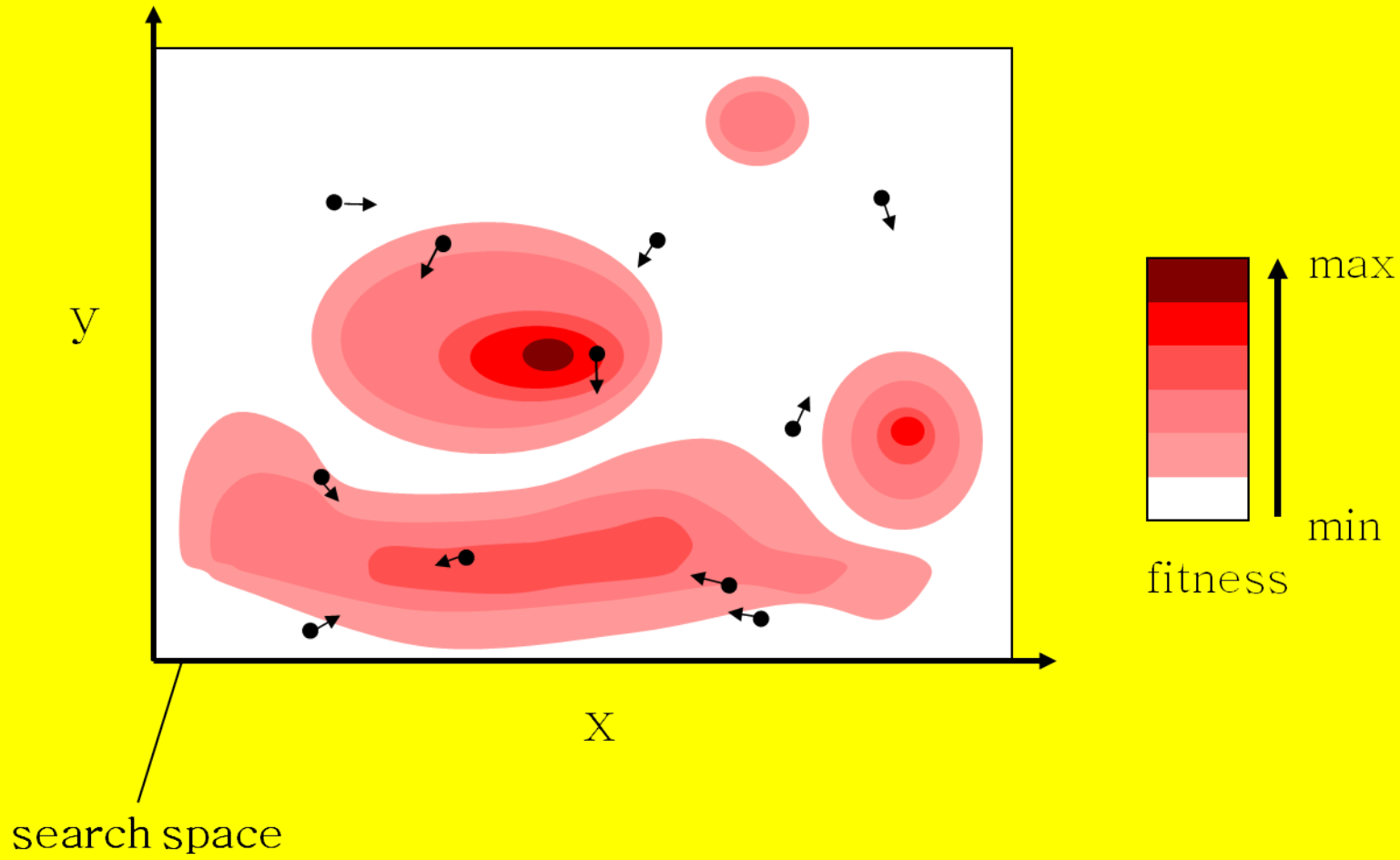
- Intensification: explores the previous solutions, finds the best solution of a given region
- Diversification: searches new solutions, finds the regions with potentially the best solutions
- In PSO:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{Diversification}} + \underbrace{c_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + c_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{Intensification}}$$

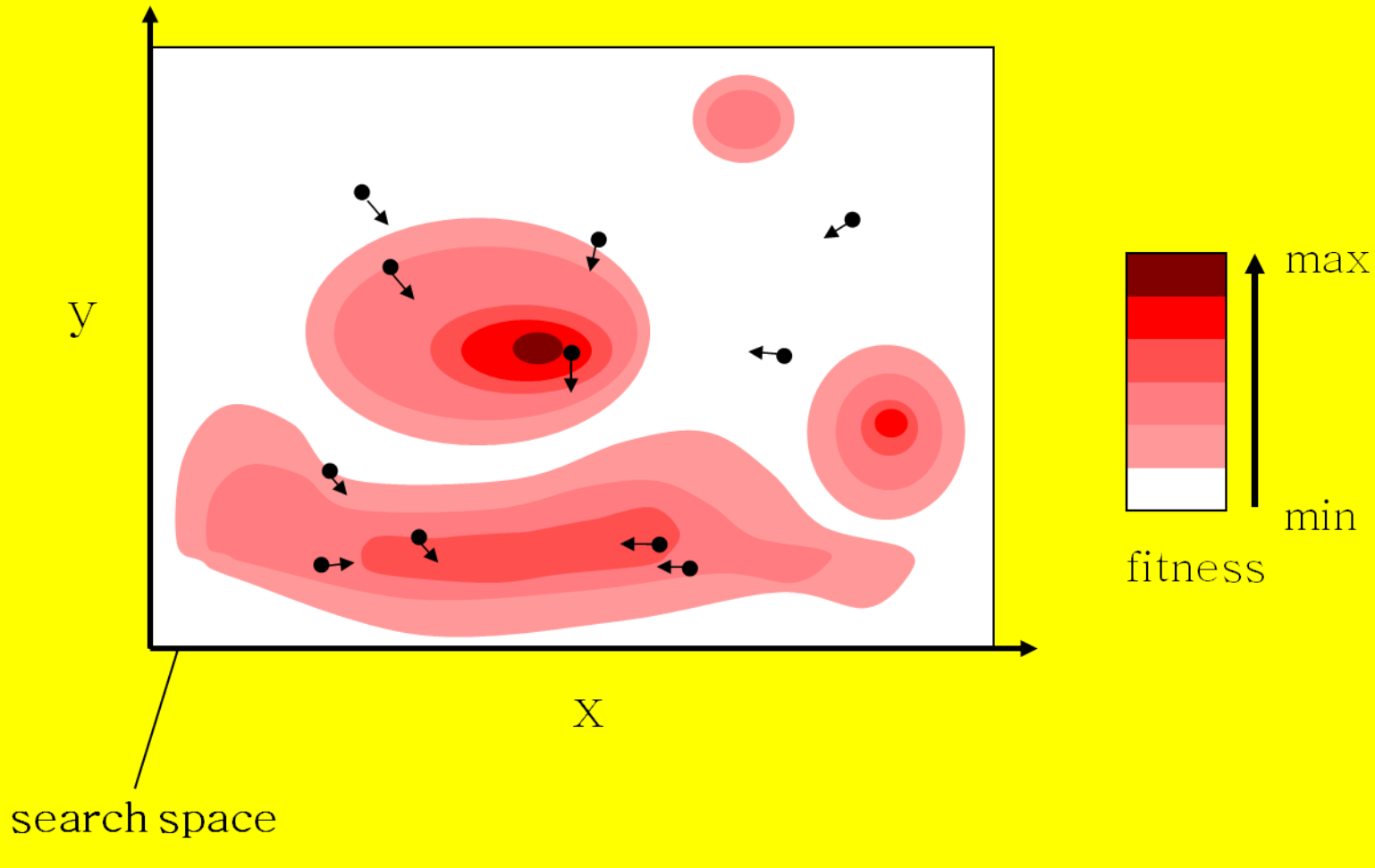
Introduction to PSO:
Algorithm - Example



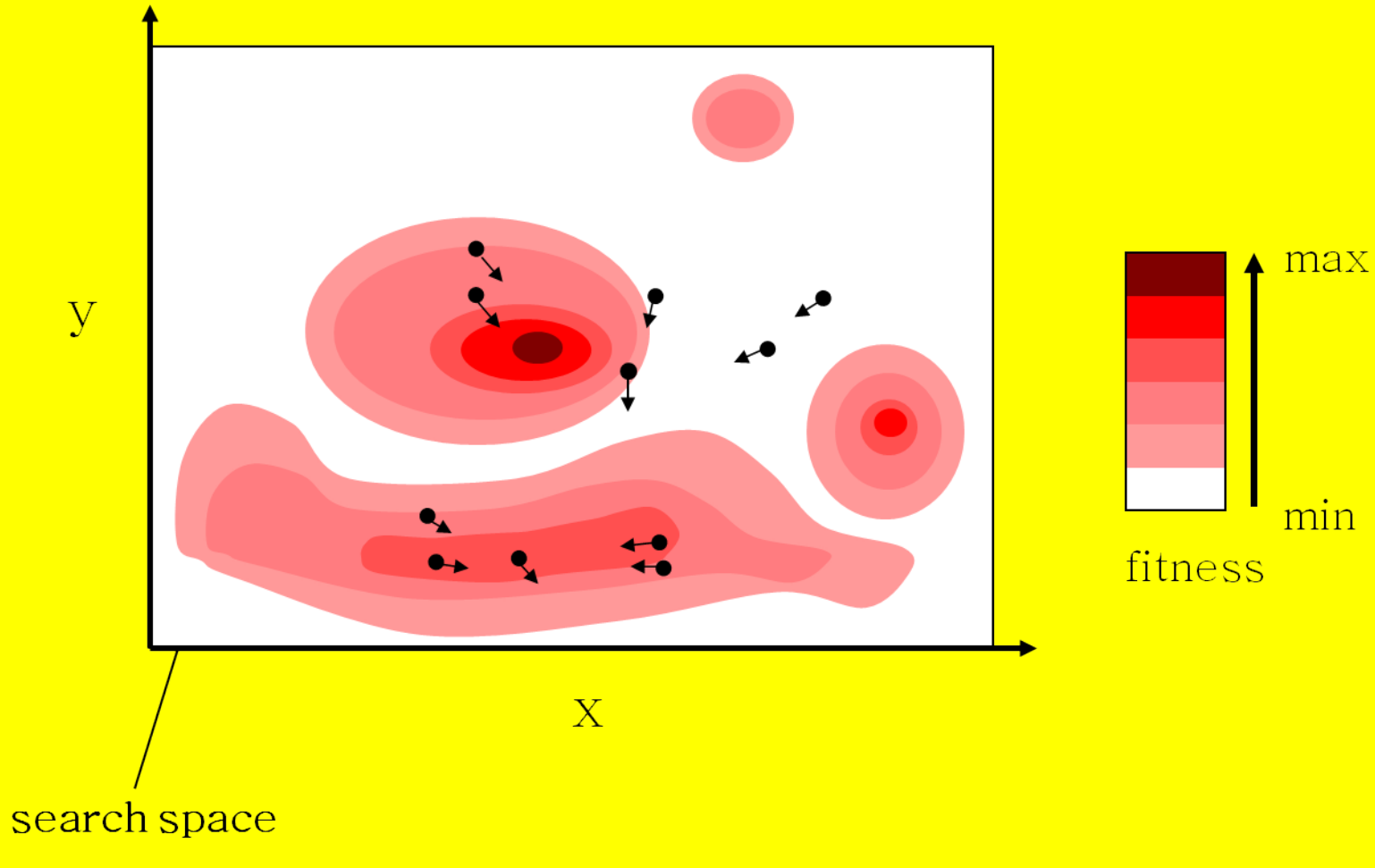
Introduction to PSO:
Algorithm - Example



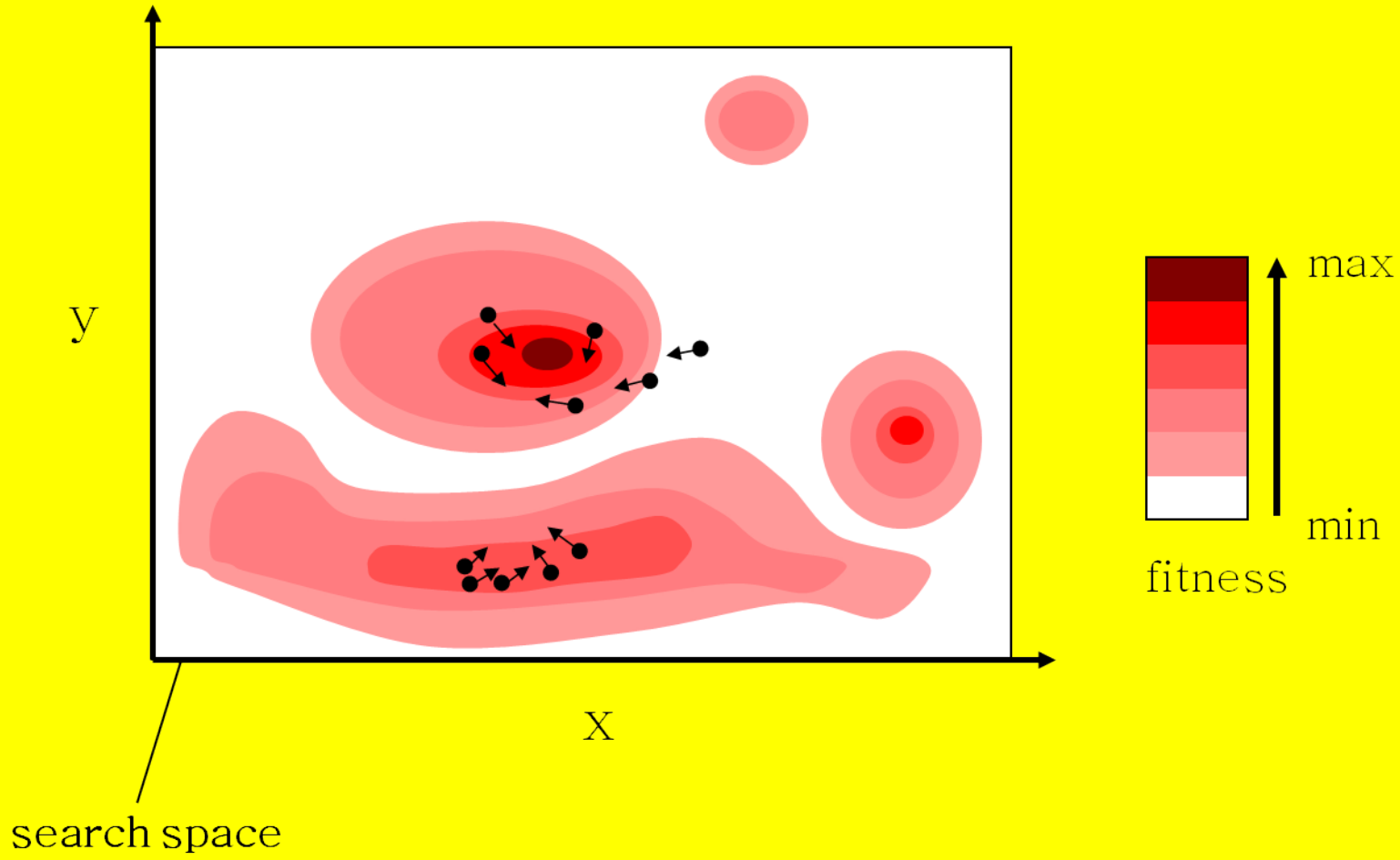
Introduction to PSO:
Algorithm - Example



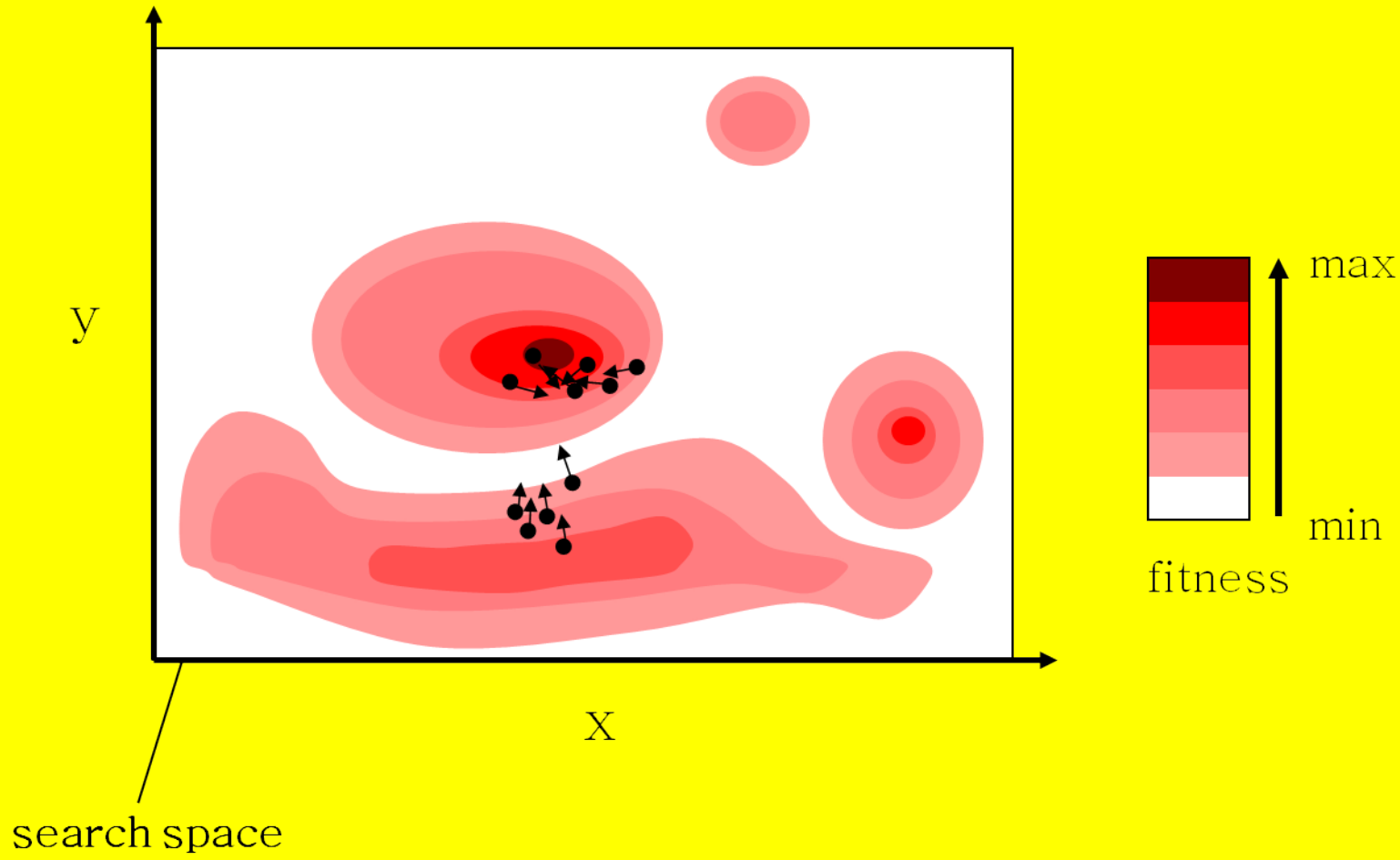
Introduction to PSO:
Algorithm - Example



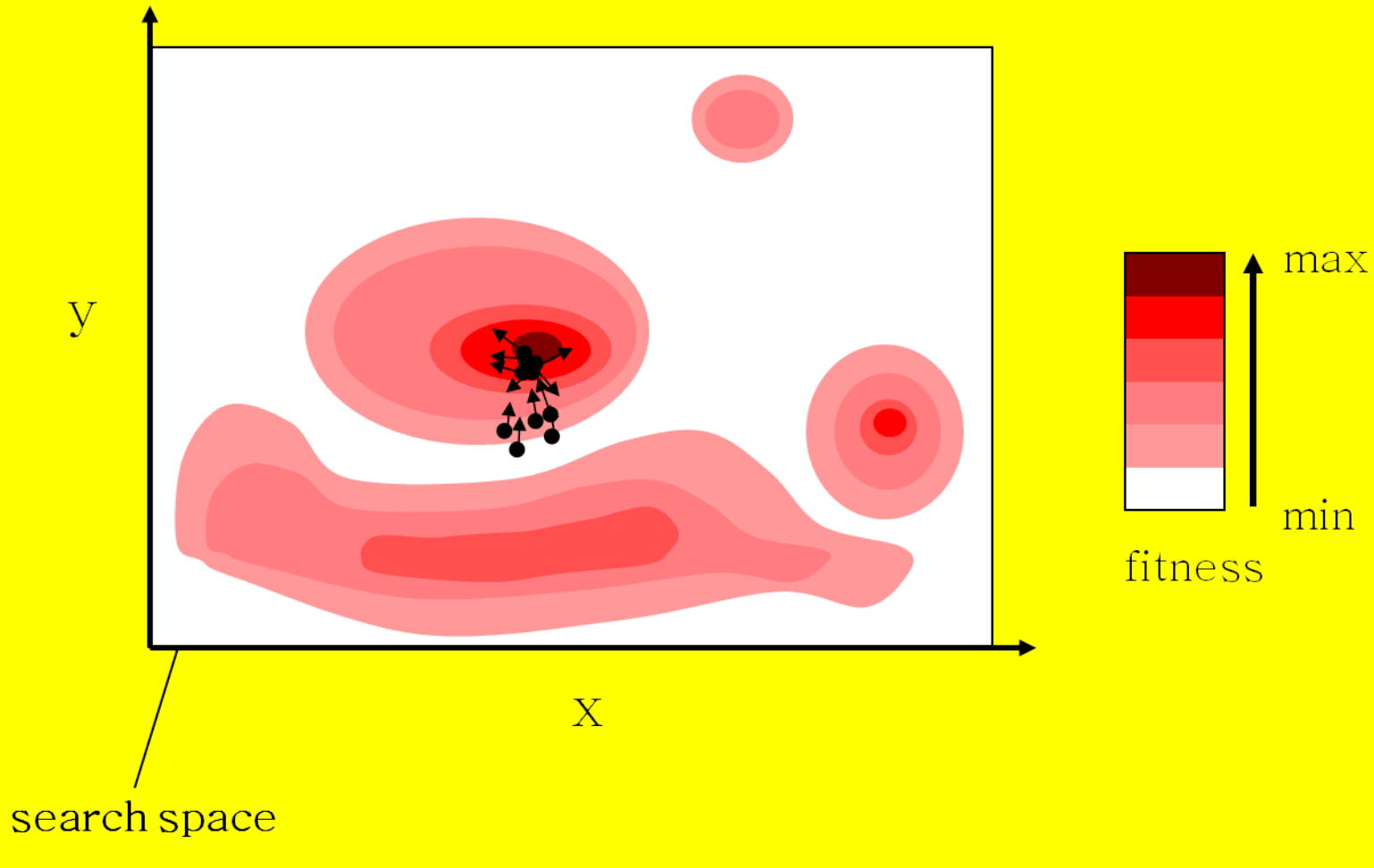
Introduction to PSO:
Algorithm - Example



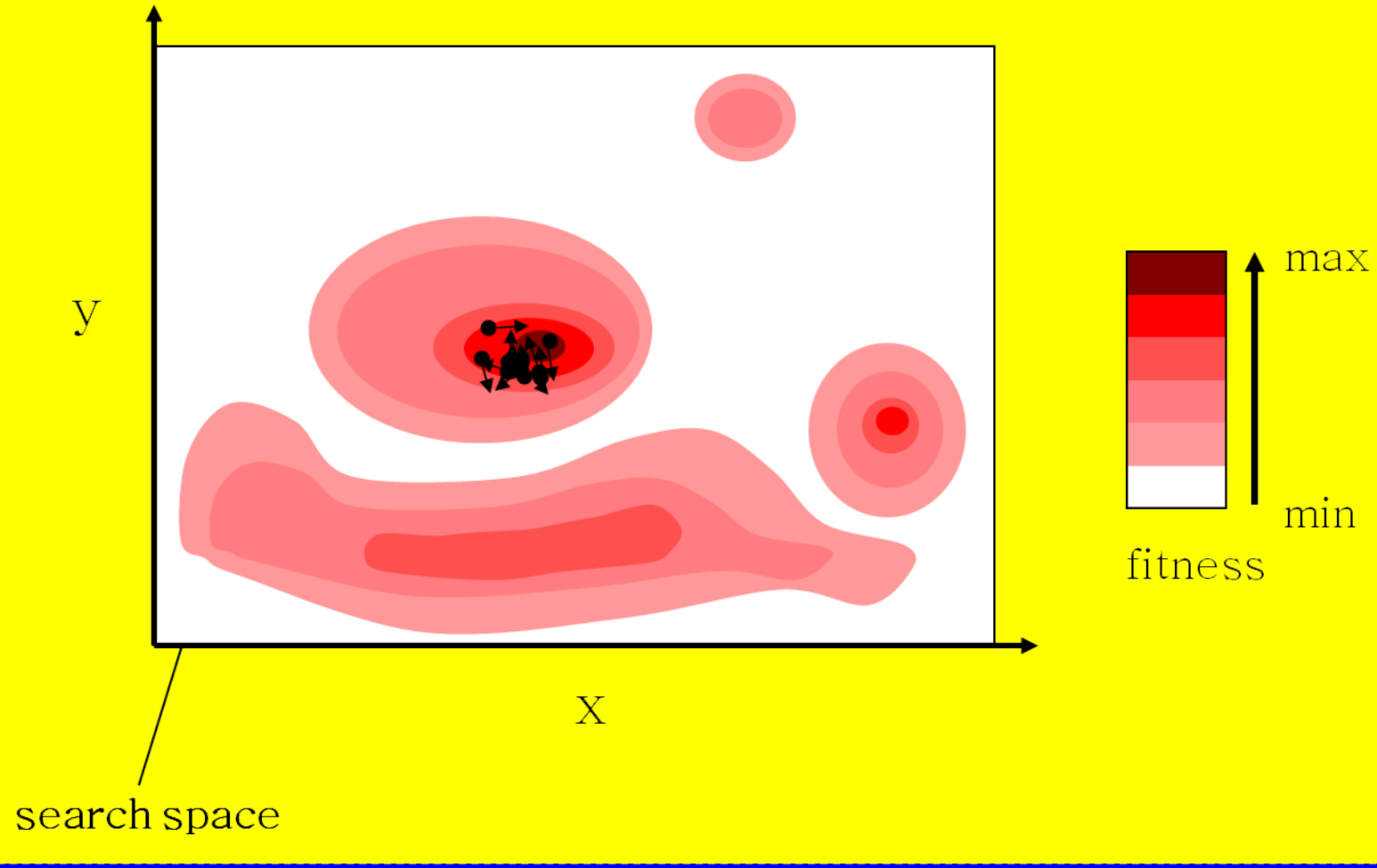
Introduction to PSO:
Algorithm - Example



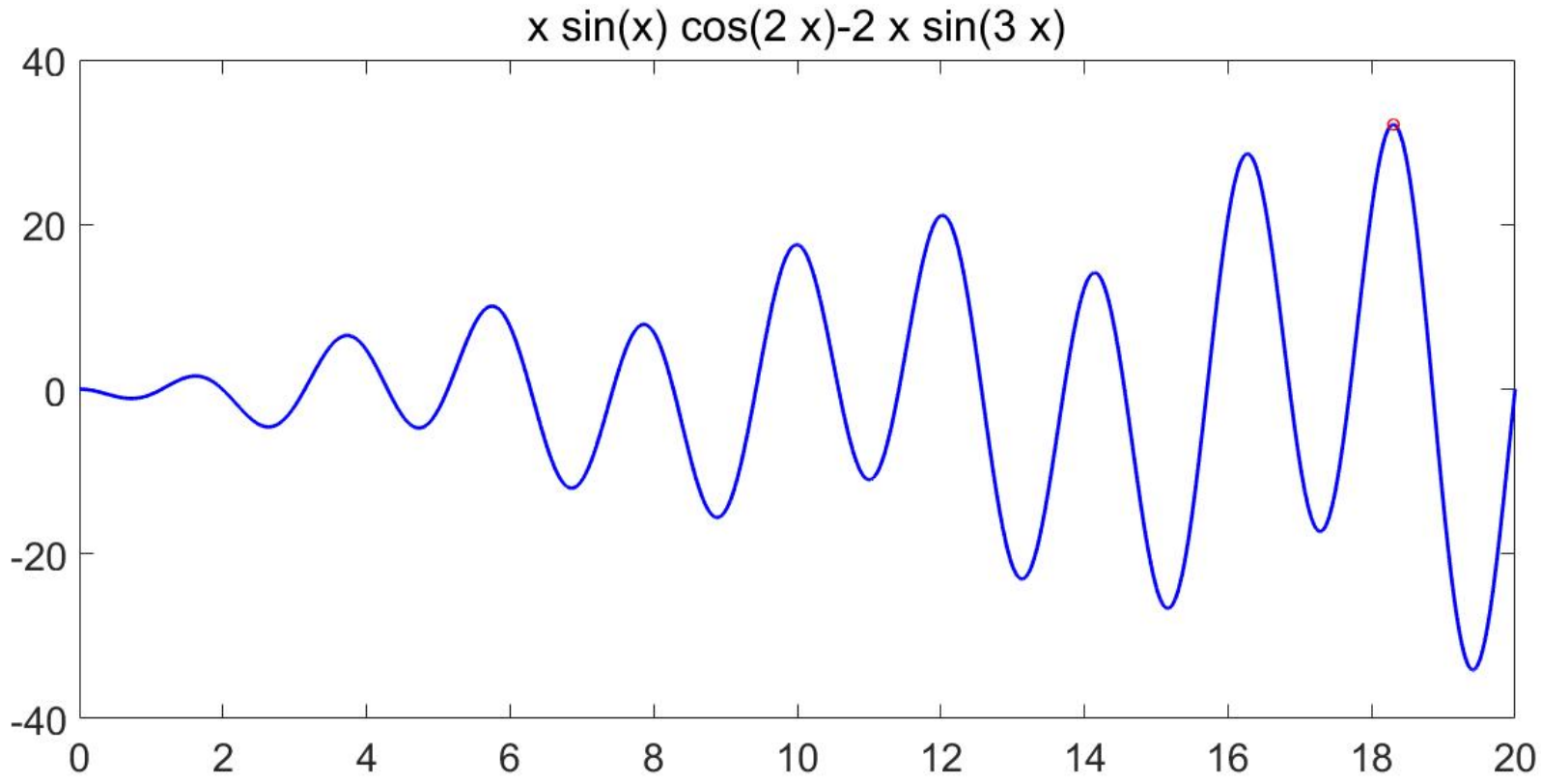
Introduction to PSO:
Algorithm - Example



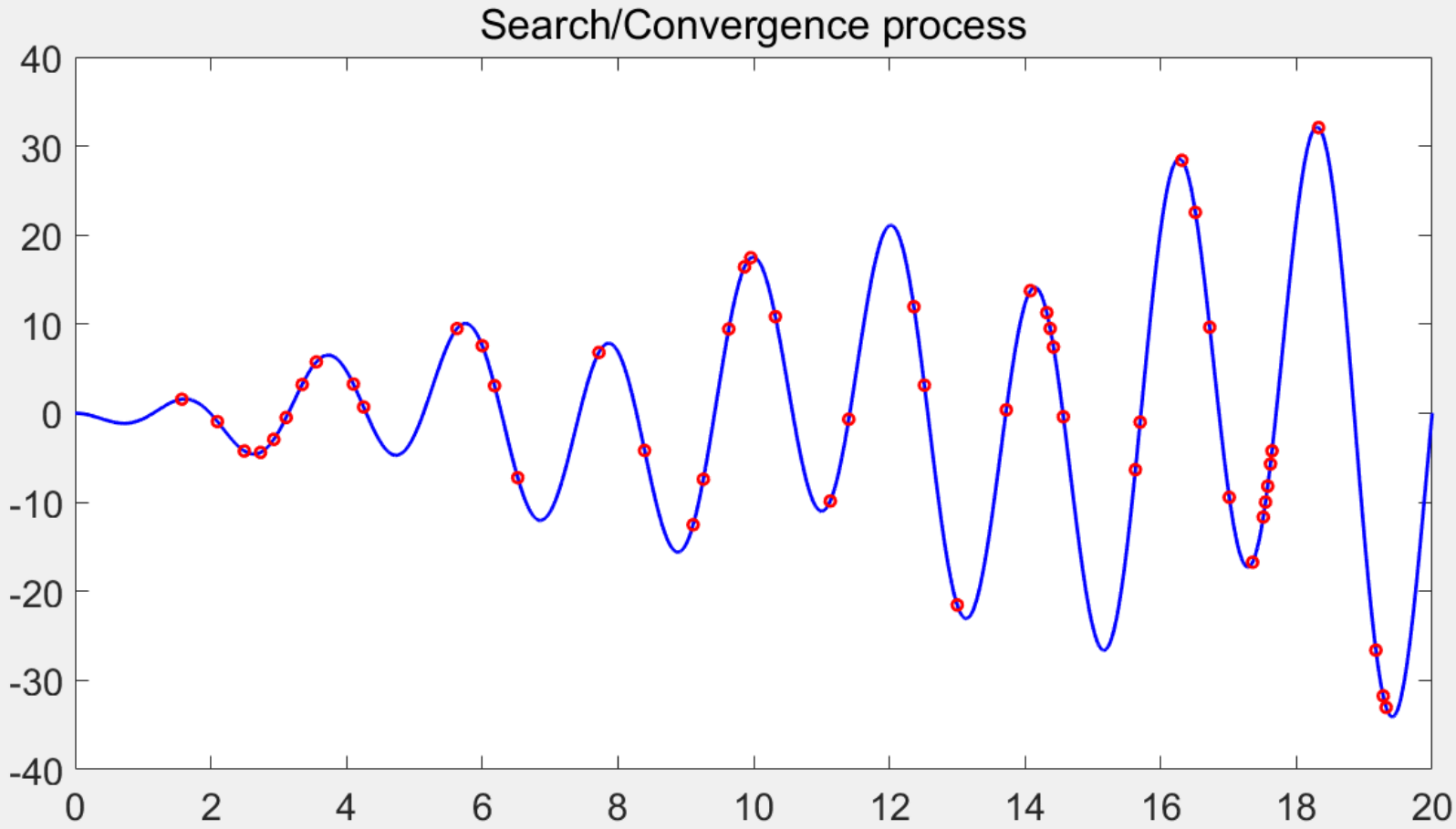
Introduction to PSO:
Algorithm - Example



Introduction to PSO:
Algorithm - Example



Introduction to PSO:
Algorithm - Example



Introduction to PSO: **Algorithm Characteristics**

- **Advantages**
  - Insensitive to scaling of design variables
  - Simple implementation
  - Easily parallelized for concurrent processing
  - Derivative free
  - Very few algorithm parameters
  - Very efficient global search algorithm
- **Disadvantages**
  - Tendency to a fast and premature convergence in local optimum points
  - Slow convergence in refined search stage (weak local search ability)

Introduction to PSO: **Different Approaches**

- **Several approaches**
  - *2-D Otsu PSO*
  - *Active Target PSO*
  - *Adaptive PSO*
  - *Adaptive Mutation PSO*
  - *Adaptive PSO Guided by Acceleration Information*
  - *Attractive Repulsive Particle Swarm Optimization*
  - *Binary PSO*
  - *Cooperative Multiple PSO*
  - *Dynamic and Adjustable PSO*
  - *Extended Particle Swarms*
  - *…*

Particle Swarm Optimization