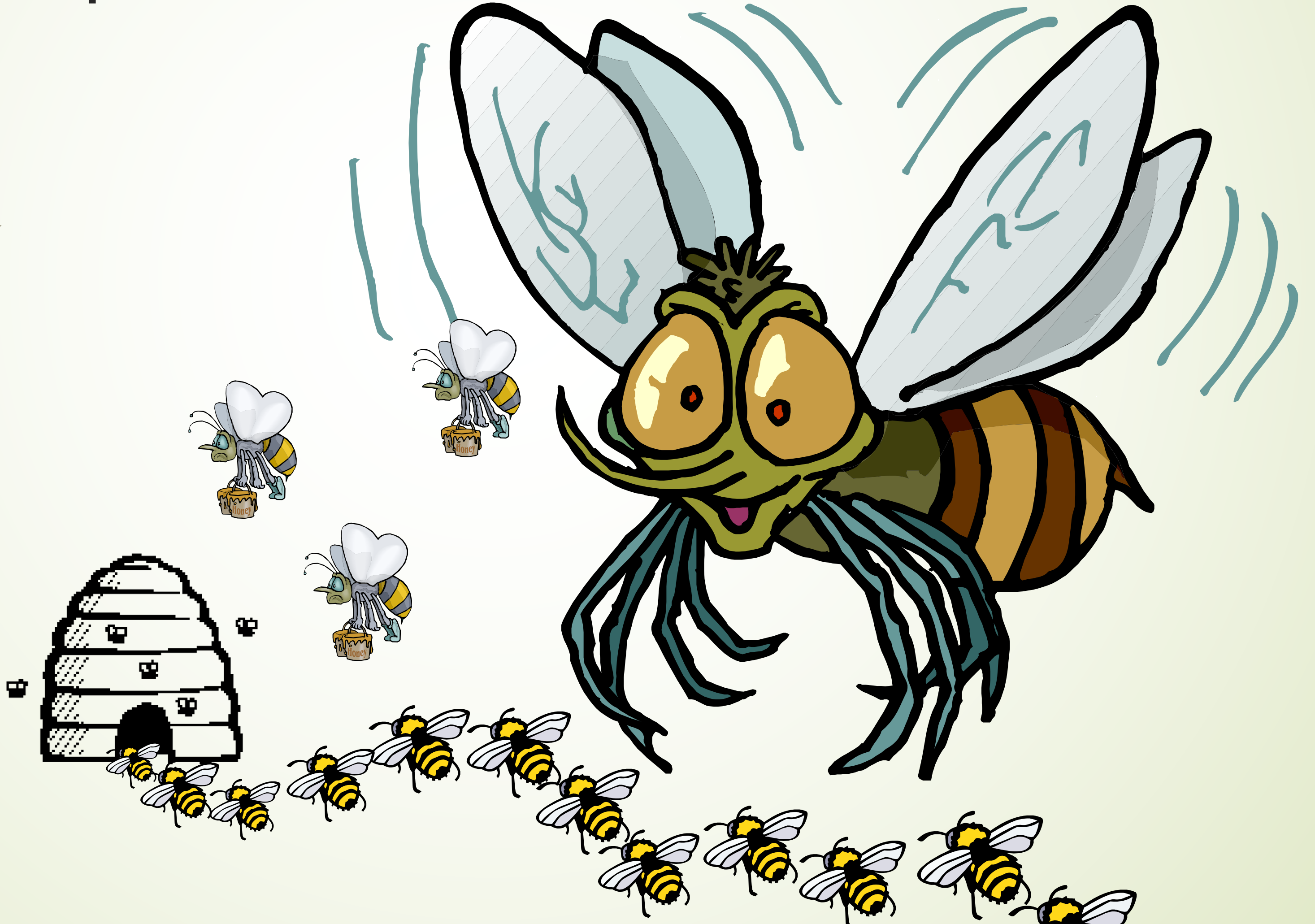
Particle Swarm optimisation



Search Techniques

- Deterministic Search Techniques
  - Branch and Bound
  - Steepest Descent
  - Newton-Raphson
  - Simplex based Technique
- Stochastic or Random Search Techniques
  - Swarm Intelligence
  - Genetic Algorithm
  - Differential Evolution
  - Simulated Annealing

Components of Search Techniques

- Initial solution
- Search direction
- Update criteria
- Stopping criteria
- All above elements can be either
  - Deterministic or Stochastic
  - Single points or population based

Swarm Intelligence

"The emergent collective intelligence of groups of simple agents."

(Bonabeau et al, 1999)

Swarm Intelligence Algorithms

- **Particle Swarm Optimization (PSO)**
- **Ant Colony Optimization (ACO)**
- Artificial Bee Colony Algorithm
- Artificial Immune Systems Algorithm

Global Optimization

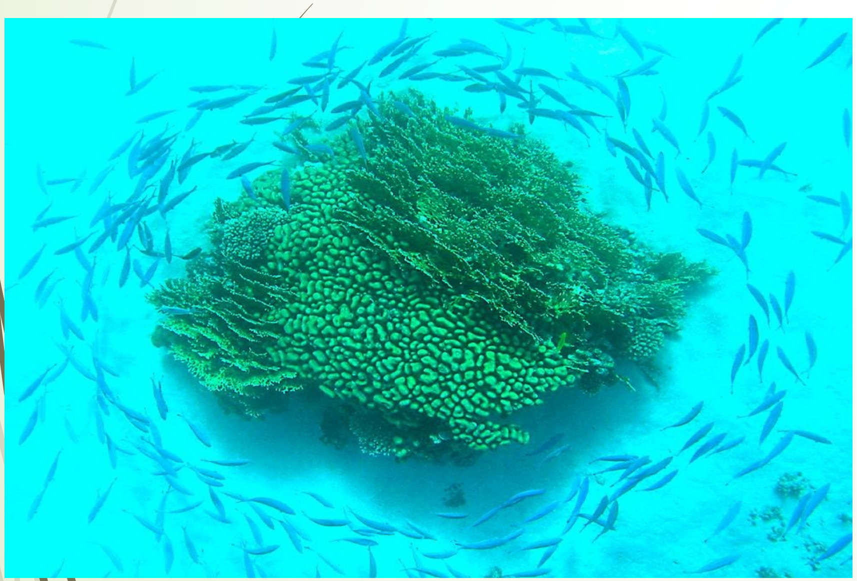
- The goal is to find the Global minimum   inside a bounded domain:

$$\min f(x), \;\; x \in S \subset R^N$$

- One way to do that, is to find all the local minima and choose among them the global one (or ones).
- Popular methods of that kind are Multistart, MLSL, TMLSL, etc.

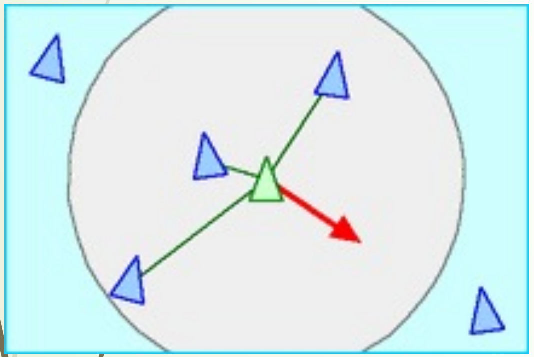
Introduction to the PSO: **Origins**

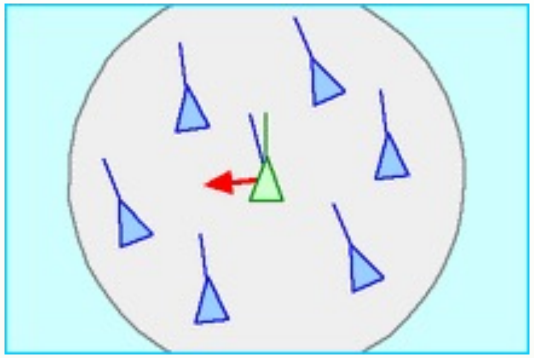
- Inspired from the nature social behavior and dynamic movements with communications of insects, birds and fish

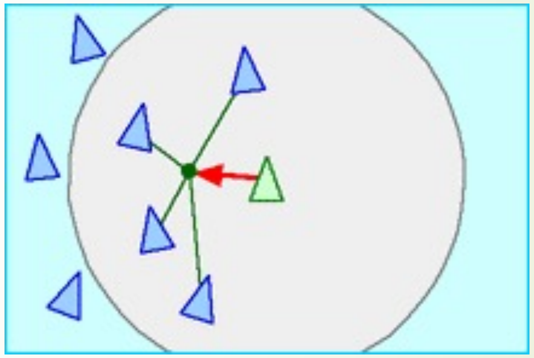


Introduction to the PSO: **Origins**

- 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

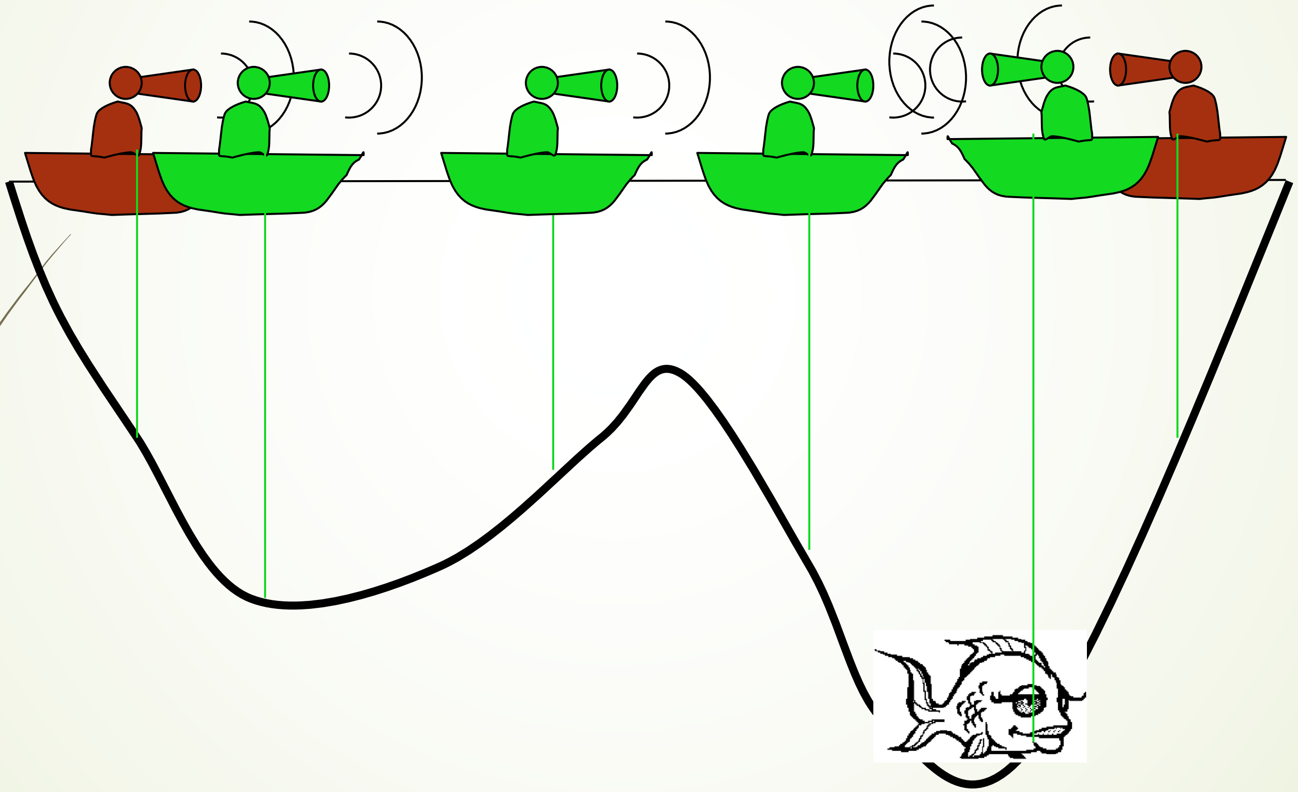
Cohesion

move toward the average position of local flockmates

Introduction to the PSO: **Origins**

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

Cooperation example



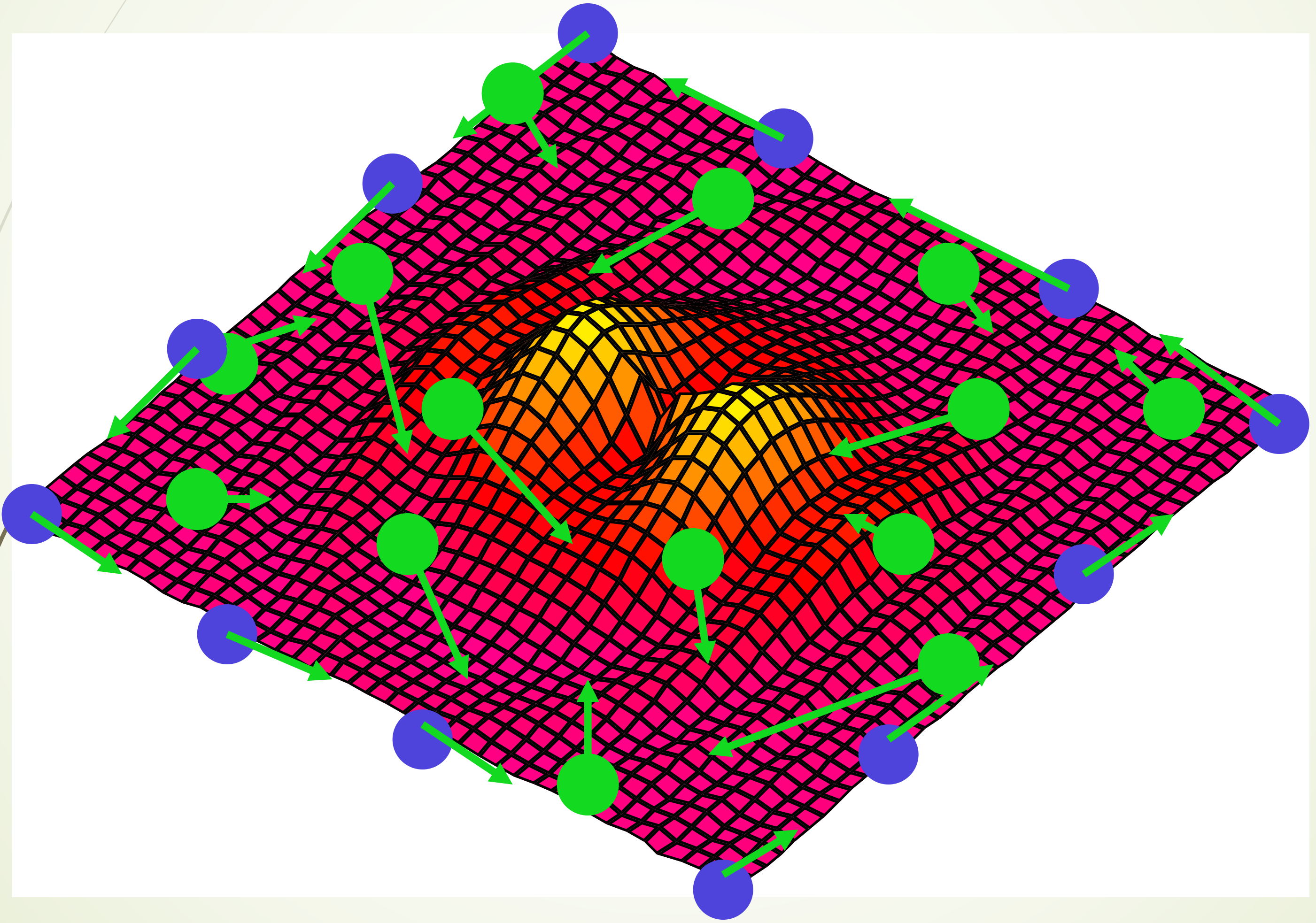
The basic idea

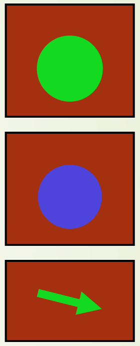
- Each particle is searching for the optimum
- Each particle is *moving* and hence has a *velocity*.
- Each particle remembers the position it was in where it had its best result so far (its *personal best*)
- *But this would not be much good on its own; particles need help in figuring out where to search.*

The basic idea II

- The particles in the swarm co-*operate*. They exchange information about what they've discovered in the places they have visited
- The co-operation is very simple. In basic PSO it is like this:
    - A particle has a *neighbourhood* associated with it.
    - A particle knows the fitnesses of those in its neighbourhood, and uses the *position* of the one with best fitness.
    - This position is simply used to adjust the particle's velocity

Initialization. Positions and velocities

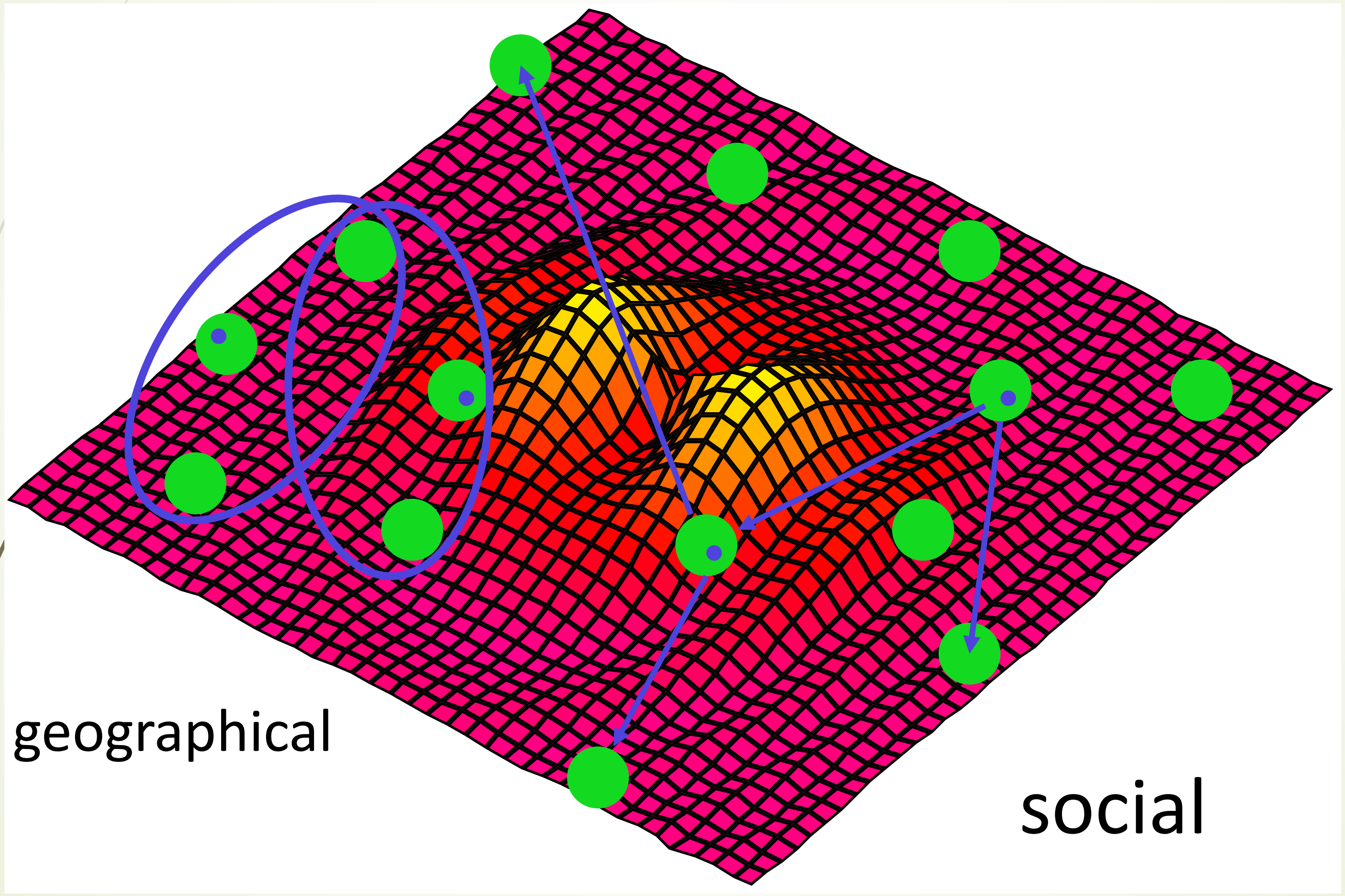




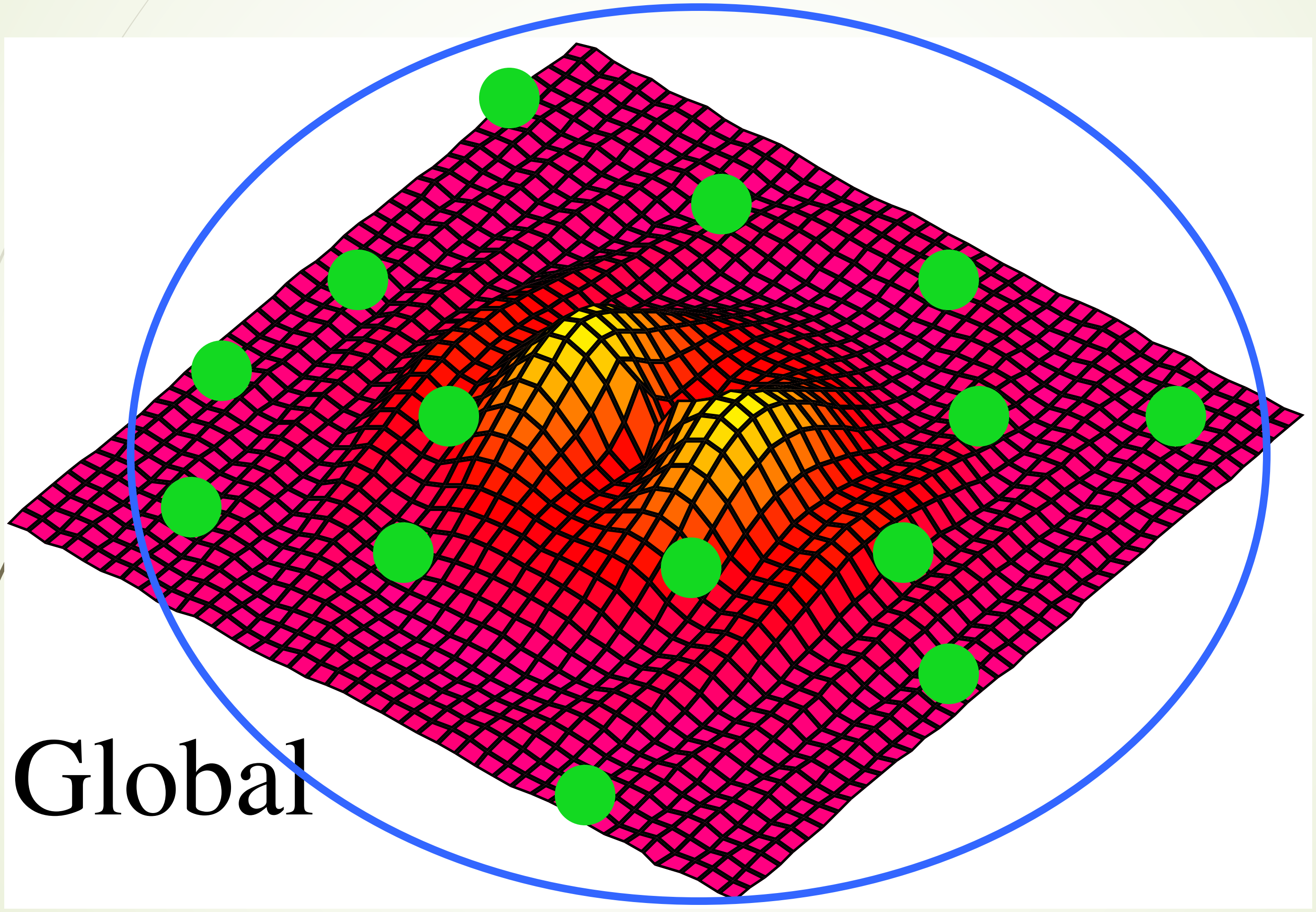
What a particle does

- In each timestep, a particle has to move to a new position by adjusting its *velocity*.
  - *The adjustment is essentially this:*
  - *The current velocity PLUS*
  - *A weighted random portion in the direction of its personal best PLUS*
  - *A weighted random portion in the direction of the neighbourhood best.*
- *Having worked out a new velocity, its position is simply its old position plus the new velocity.*

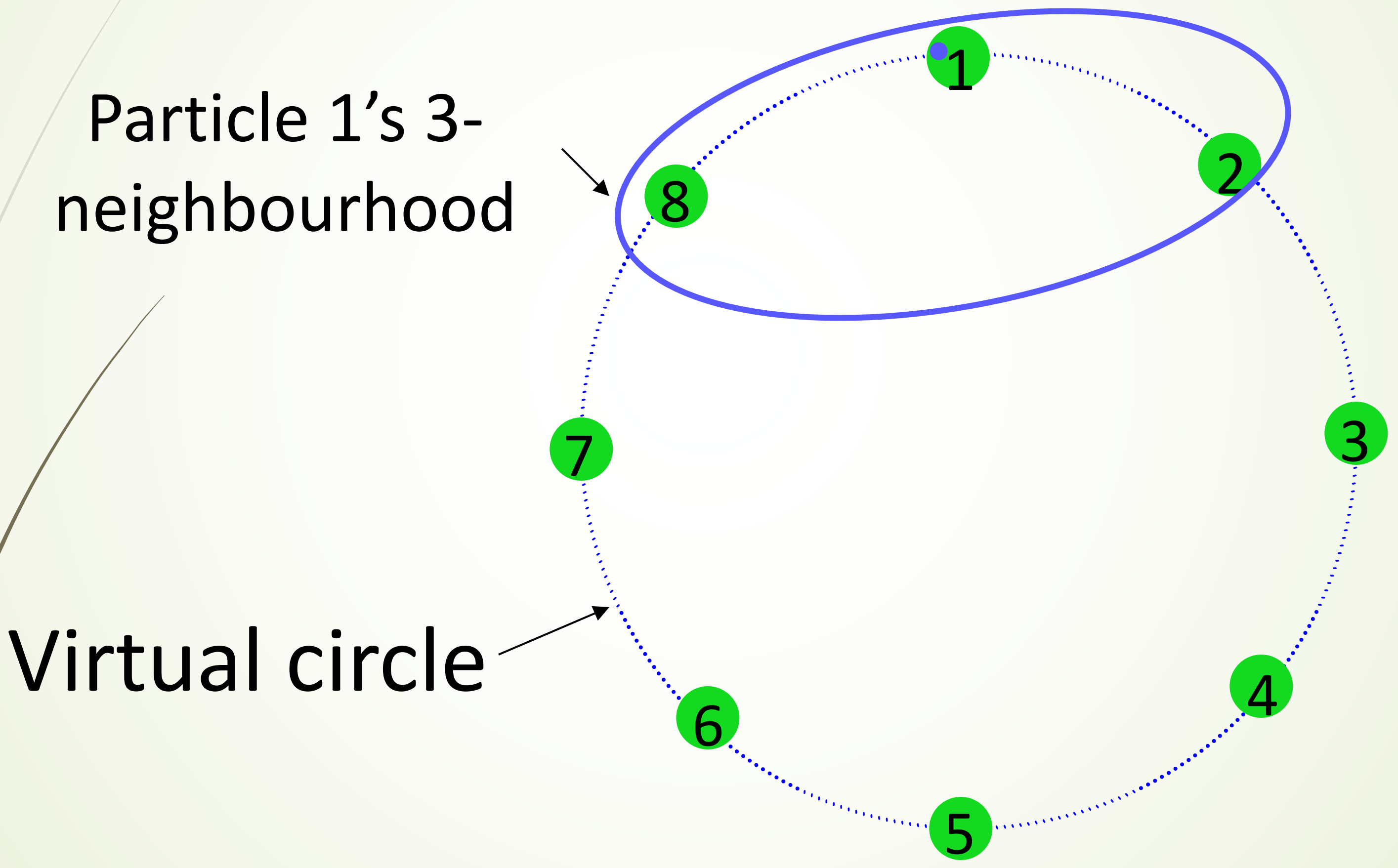
Neighbourhoods



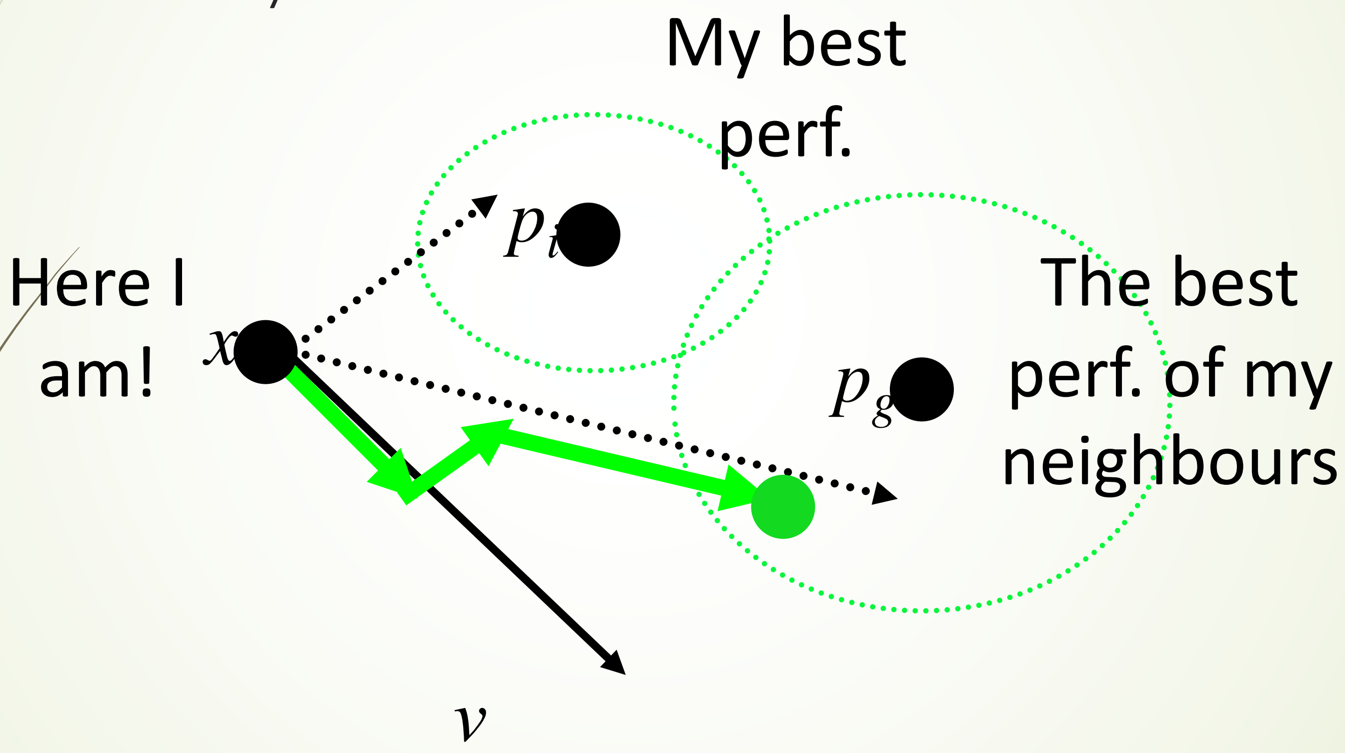
Neighbourhoods



The circular neighbourhood



Particles Adjust their positions according to a ``Psychosocial compromise'' between what an individual is comfortable with, and what society reckons



Introduction to the PSO: **Algorithm - Parameterss**

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

Introduction to the PSO: **Algorithm**

- Particle update rule

$$p = p + v$$

with

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

where

- *p*: particle's position
- v: path direction
- 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 the PSO: **Algorithm - Parameters**

1. Number of particles usually between 10 and 50
2. C1 is the importance of personal best value
3. C2 is the importance of neighborhood best value
4. Usually C1 + C2 = 4 (empirically chosen value)
5. If velocity is too low → algorithm too slow
6. If velocity is too high → algorithm too unstable

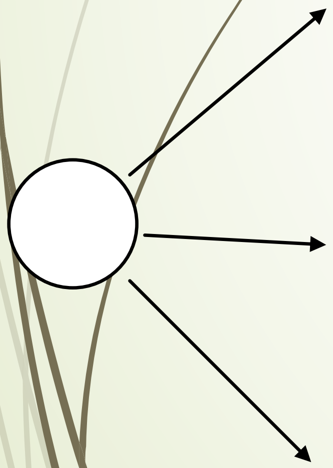
Introduction to the PSO: **Algorithm**

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 the 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**

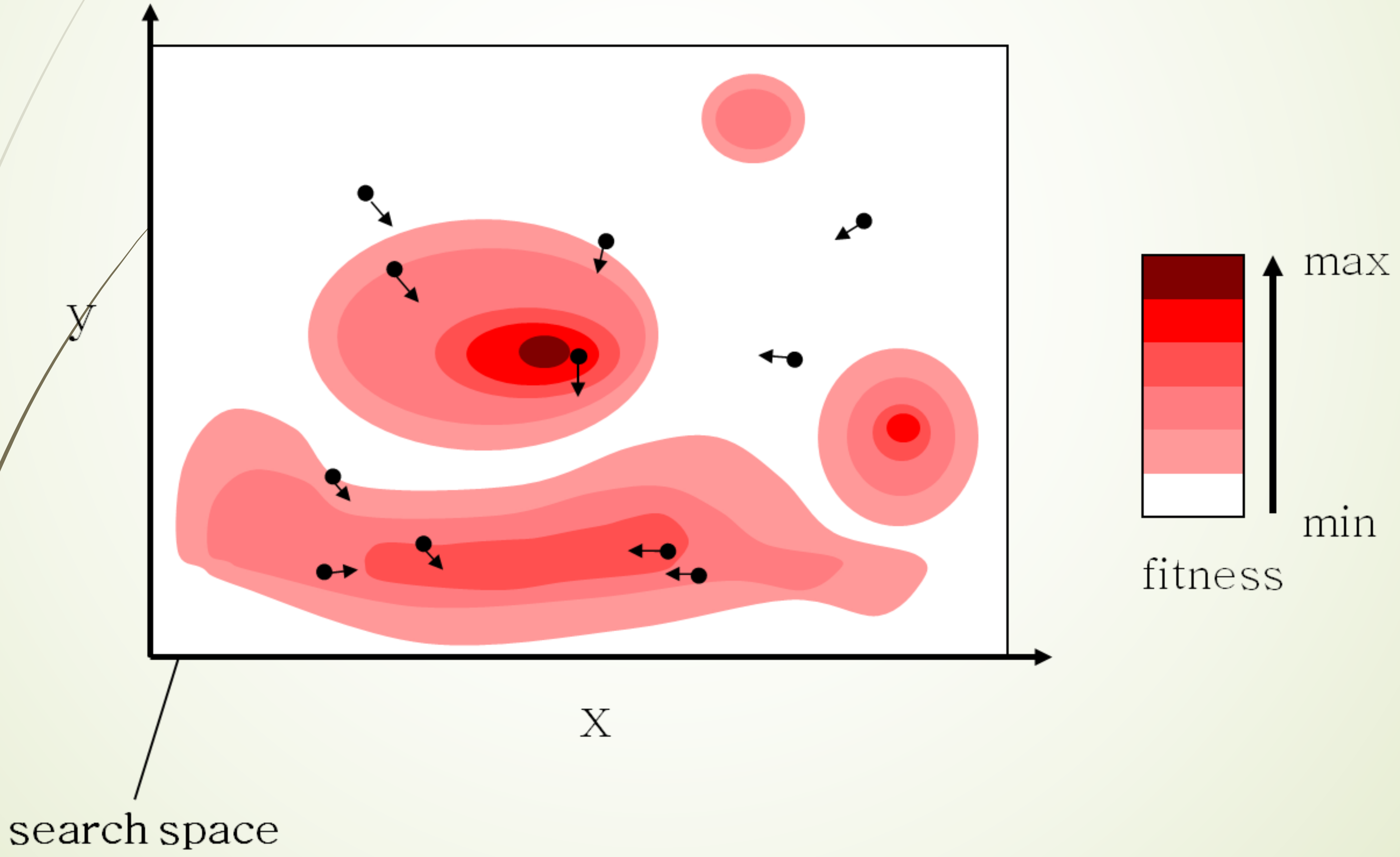
3. Social Influence

- **Makes the particle follow the best neighbors direction**

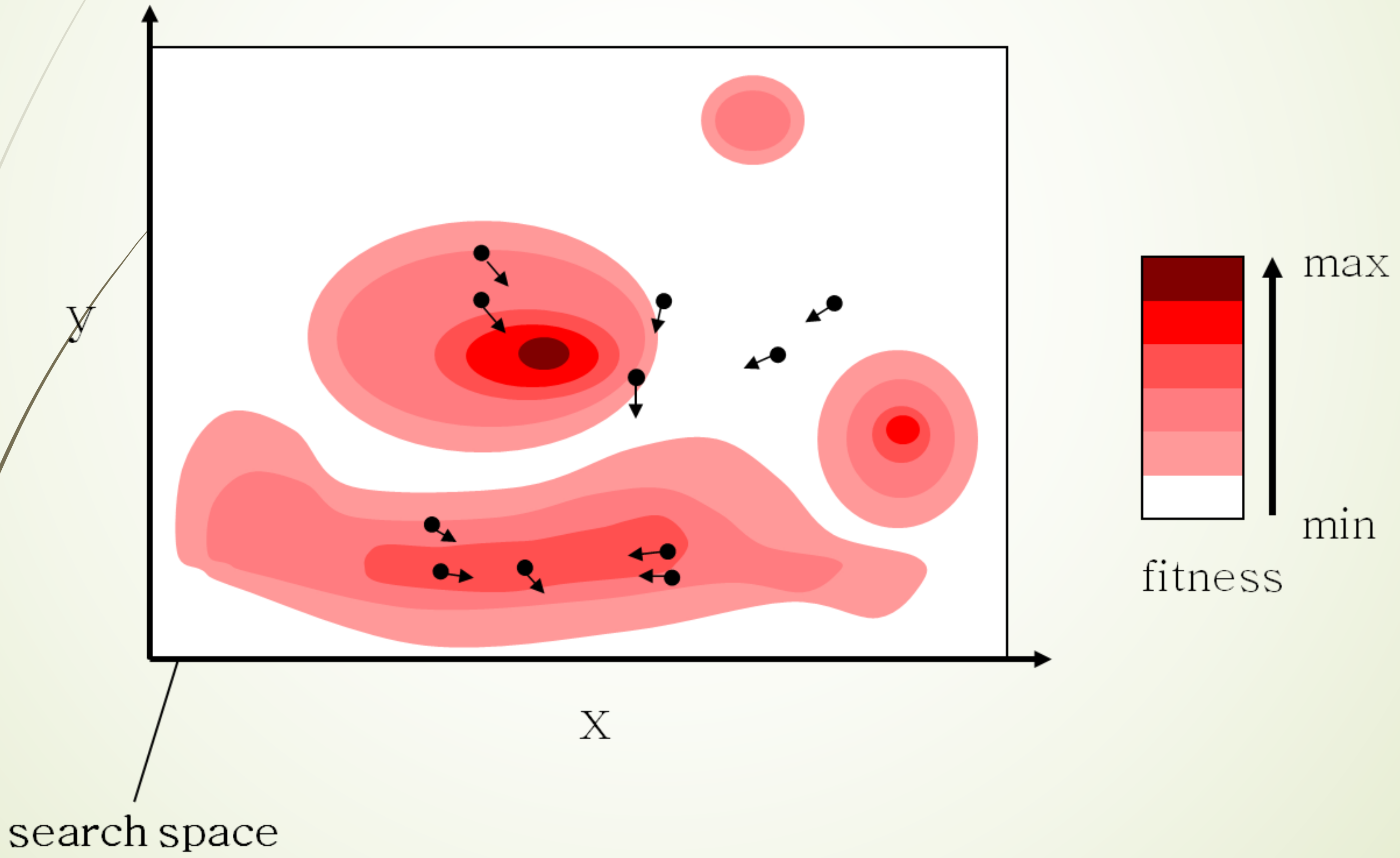
Pseudocode http://www.swarmintelligence.org/tutorials.php

```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than its peronal best
        set current value as the new **pBest**
    End

    Choose the particle with the best fitness value of all as **gBest**
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained
```

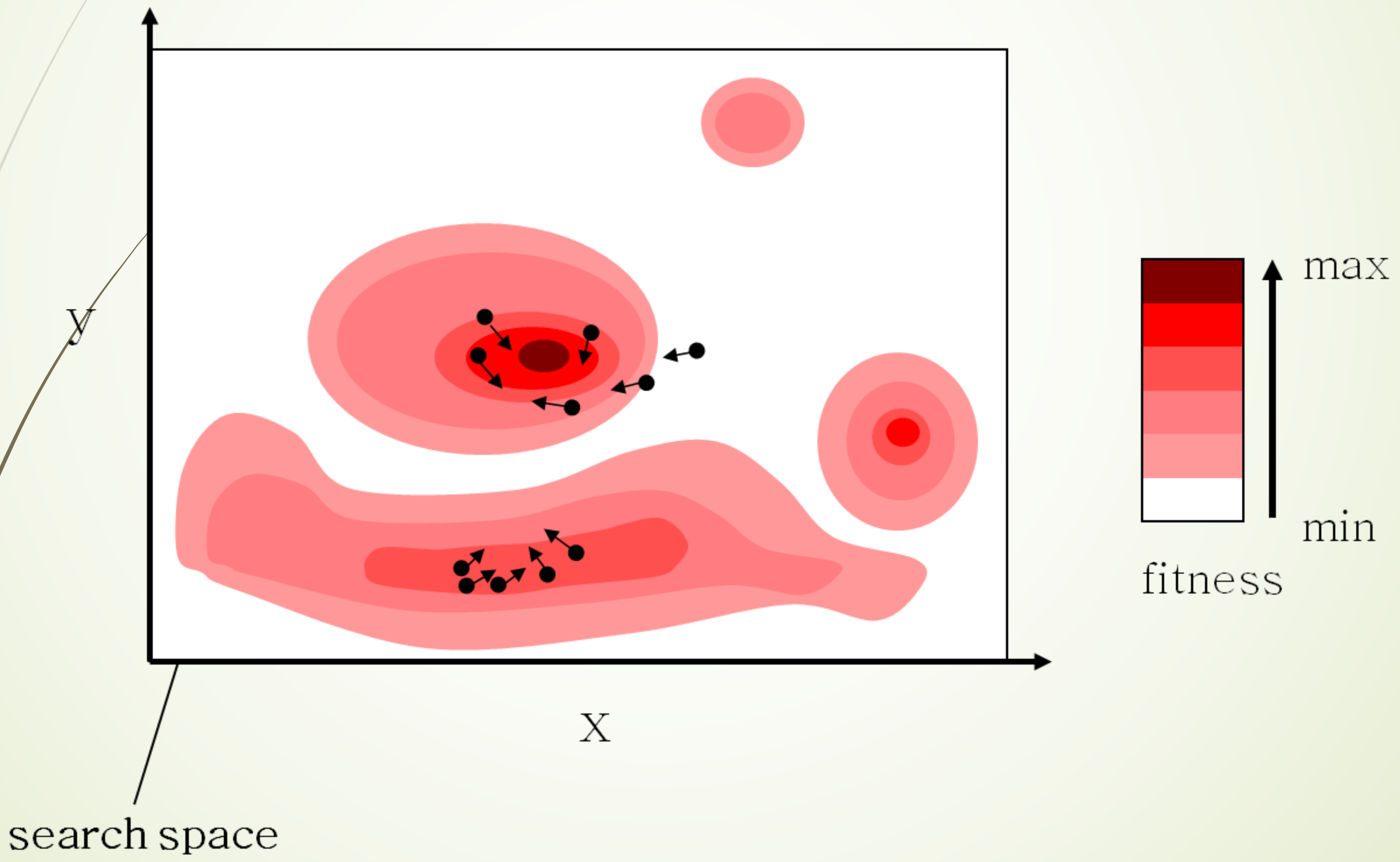
Introduction to the PSO: **Algorithm - Example**



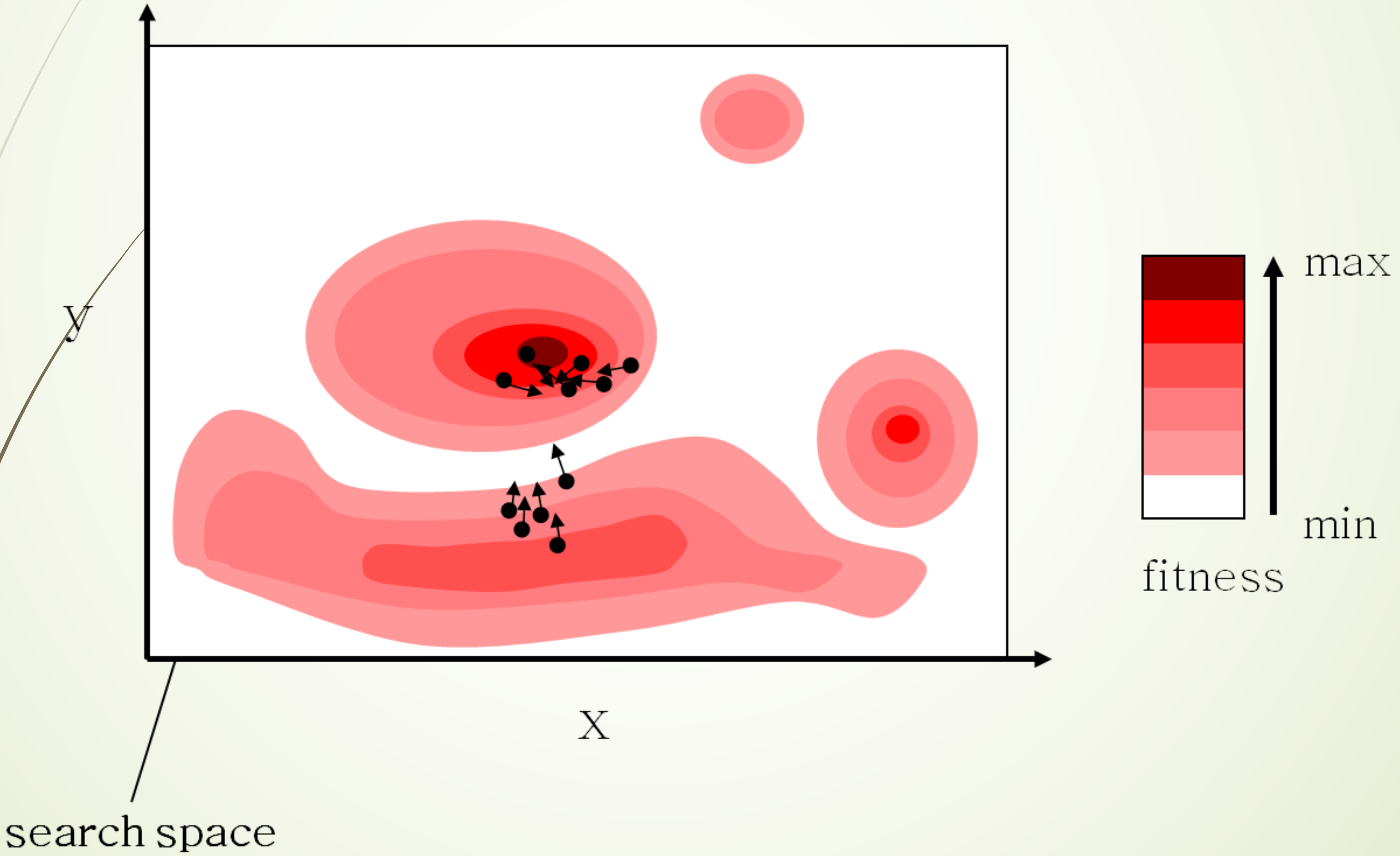
Introduction to the PSO: **Algorithm - Example**



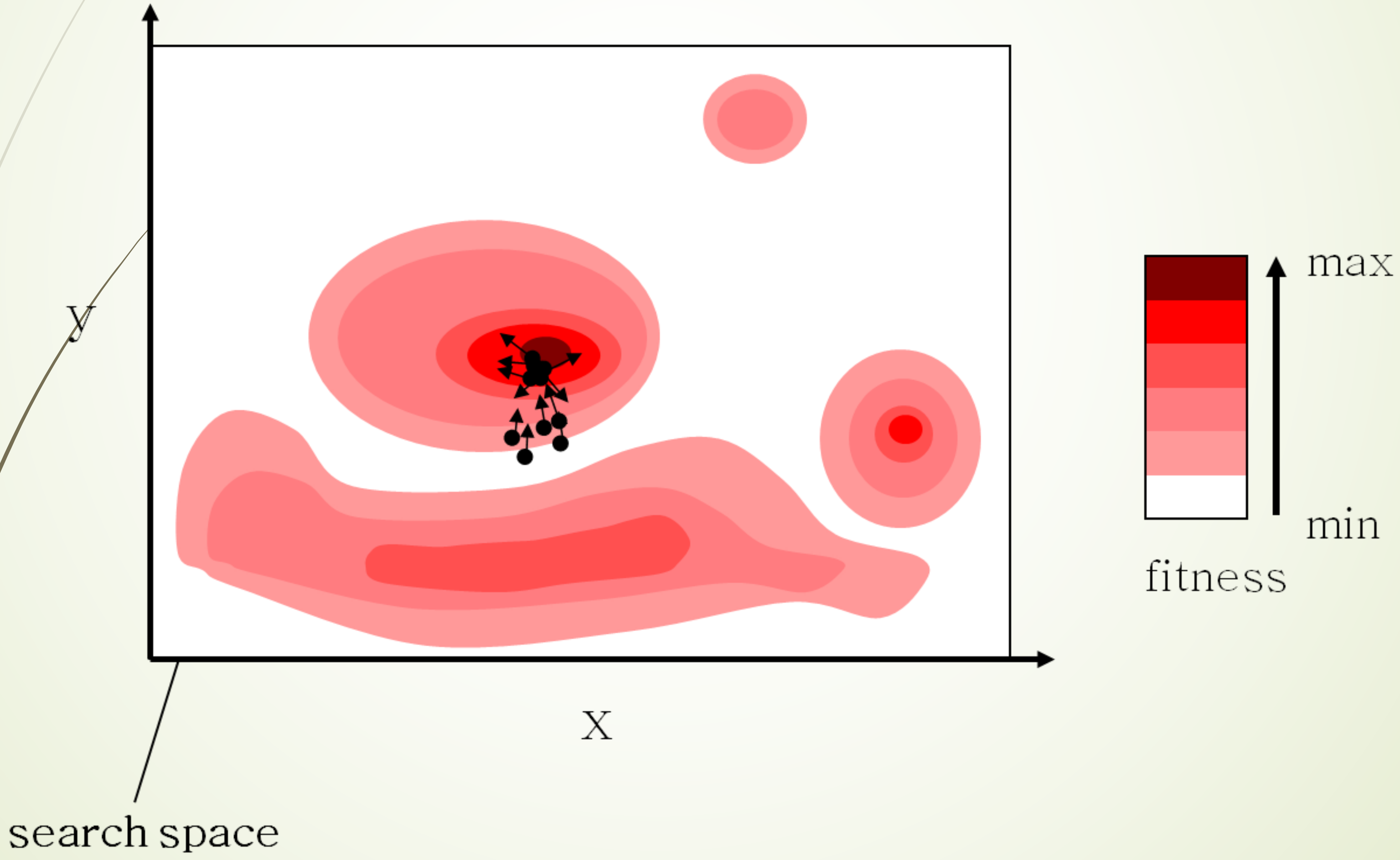
Introduction to the PSO: **Algorithm - Example**



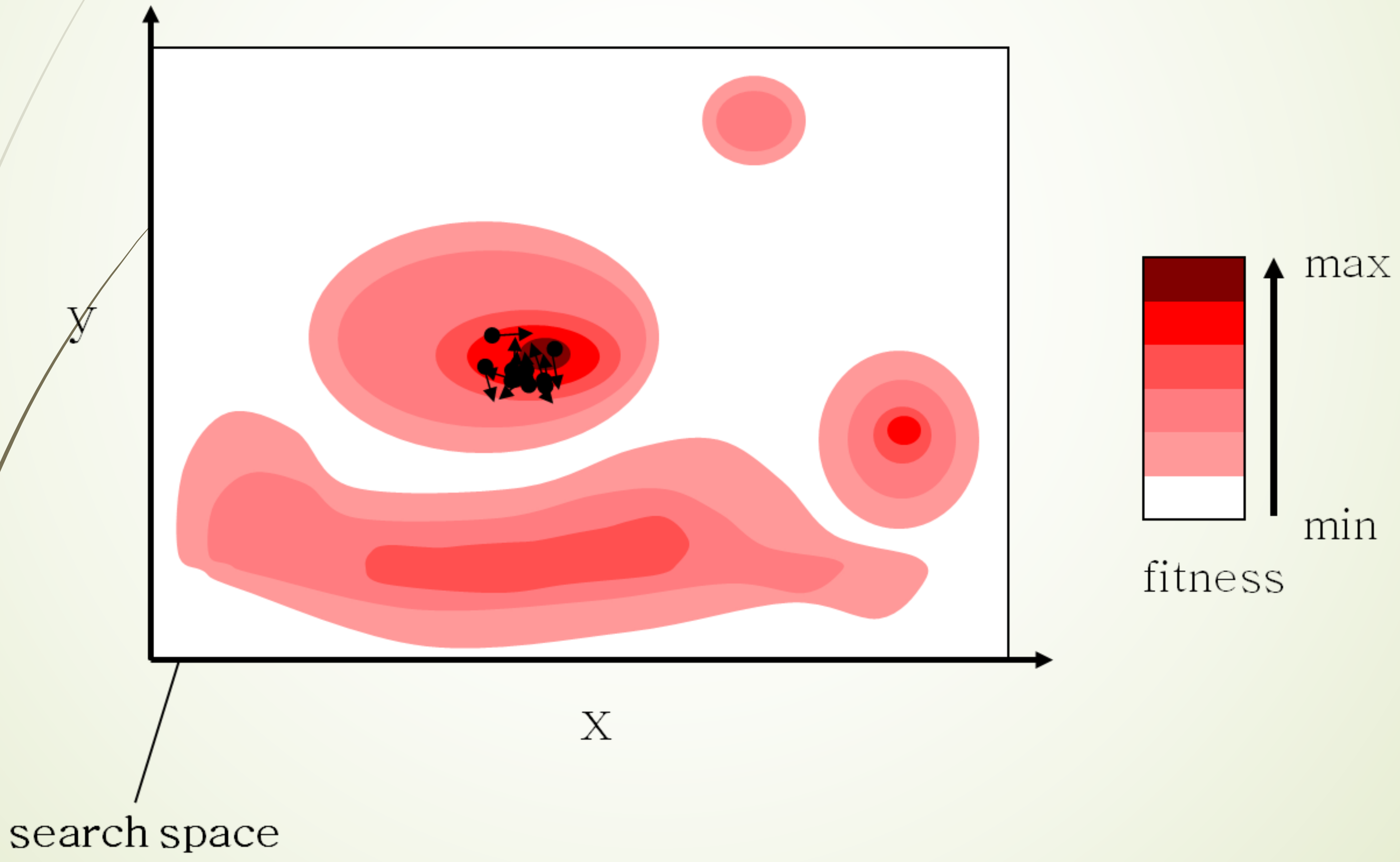
Introduction to the PSO: **Algorithm - Example**



Introduction to the PSO: **Algorithm - Example**



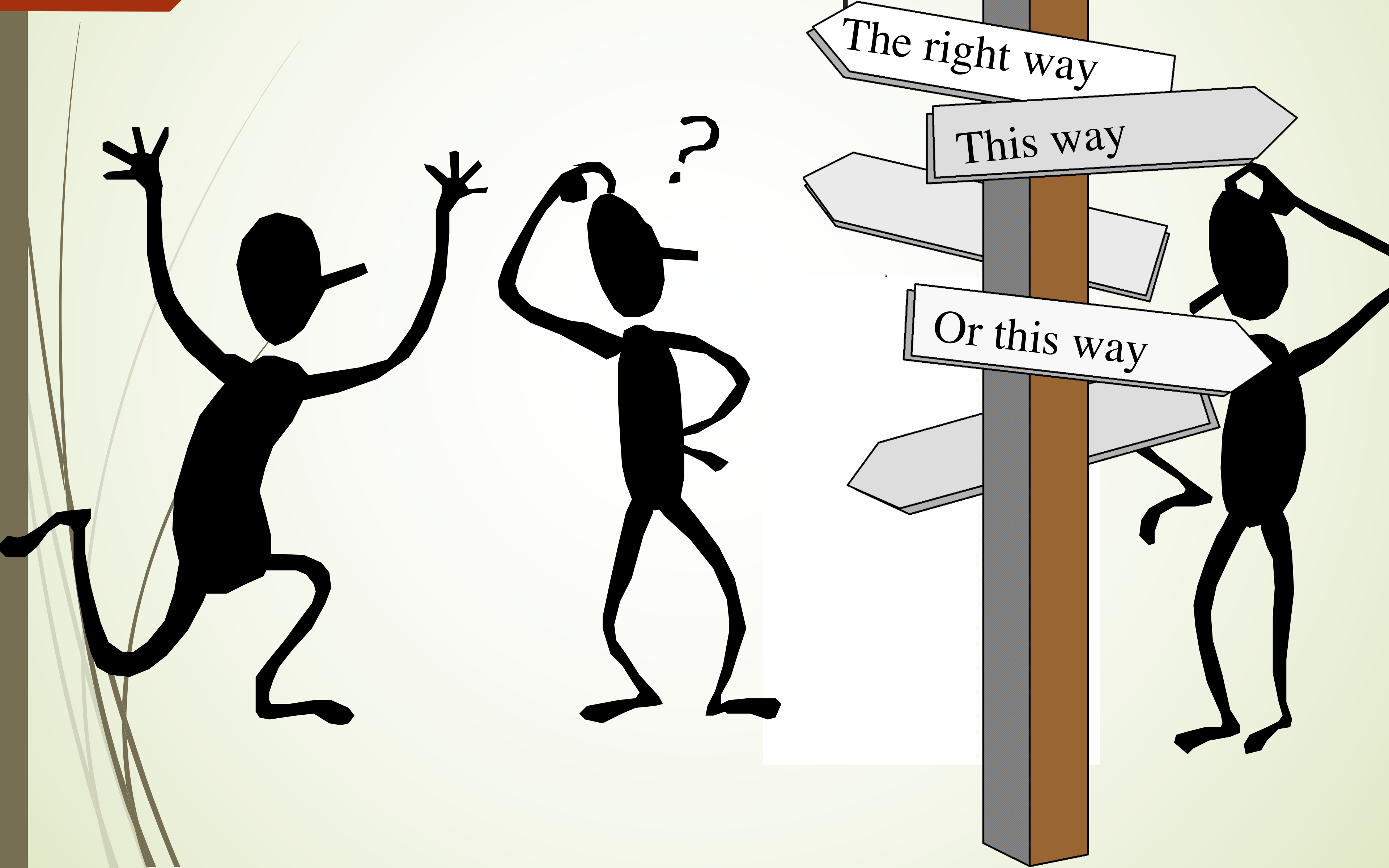
Introduction to the PSO: **Algorithm - Example**



Parameters

- Number of particles (swarmsize)
- C1 (importance of personal best)
- C2 (importance of neighbourhood best)
- Vmax: limit on velocity

How to choose parameters



Parameters

- Number of particles
- (10—50) are reported as usually sufficient.
- C1 (importance of personal best)
- C2 (importance of neighbourhood best)
- Usually C1+C2 = 4. No good reason other than empiricism
- Vmax – too low, too slow; too high, too unstable.

Introduction to the 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 mid optimum points
  - Slow convergence in refined search stage (weak local search ability)

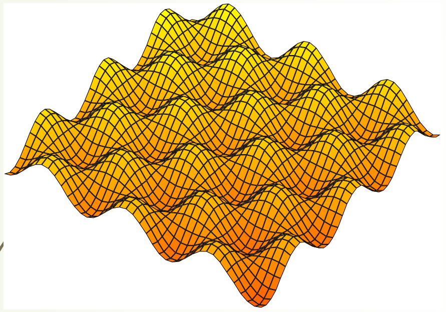
Introduction to the PSO: **Different Approaches**

- **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*
  - …

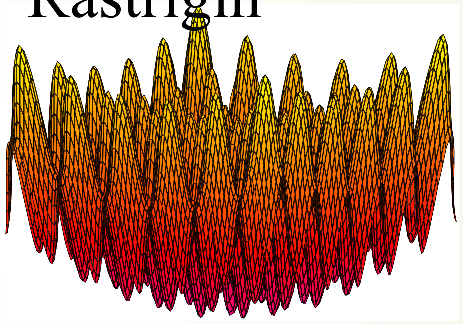
Davoud Sedighizadeh and Ellips Masehian, "Particle Swarm Optimization Methods, Taxonomy and Applications". International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December 2009

Some functions often used for testing real-valued optimisation algorithms

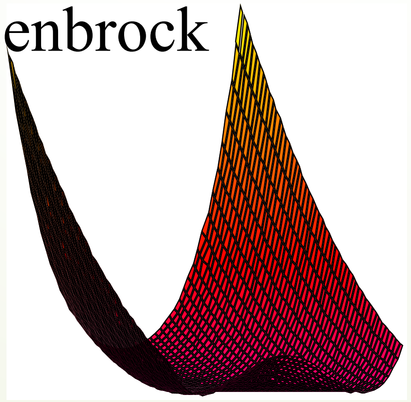
Griewank



Rastrigin



Rosenbrock



Play with DWC’s app for a while

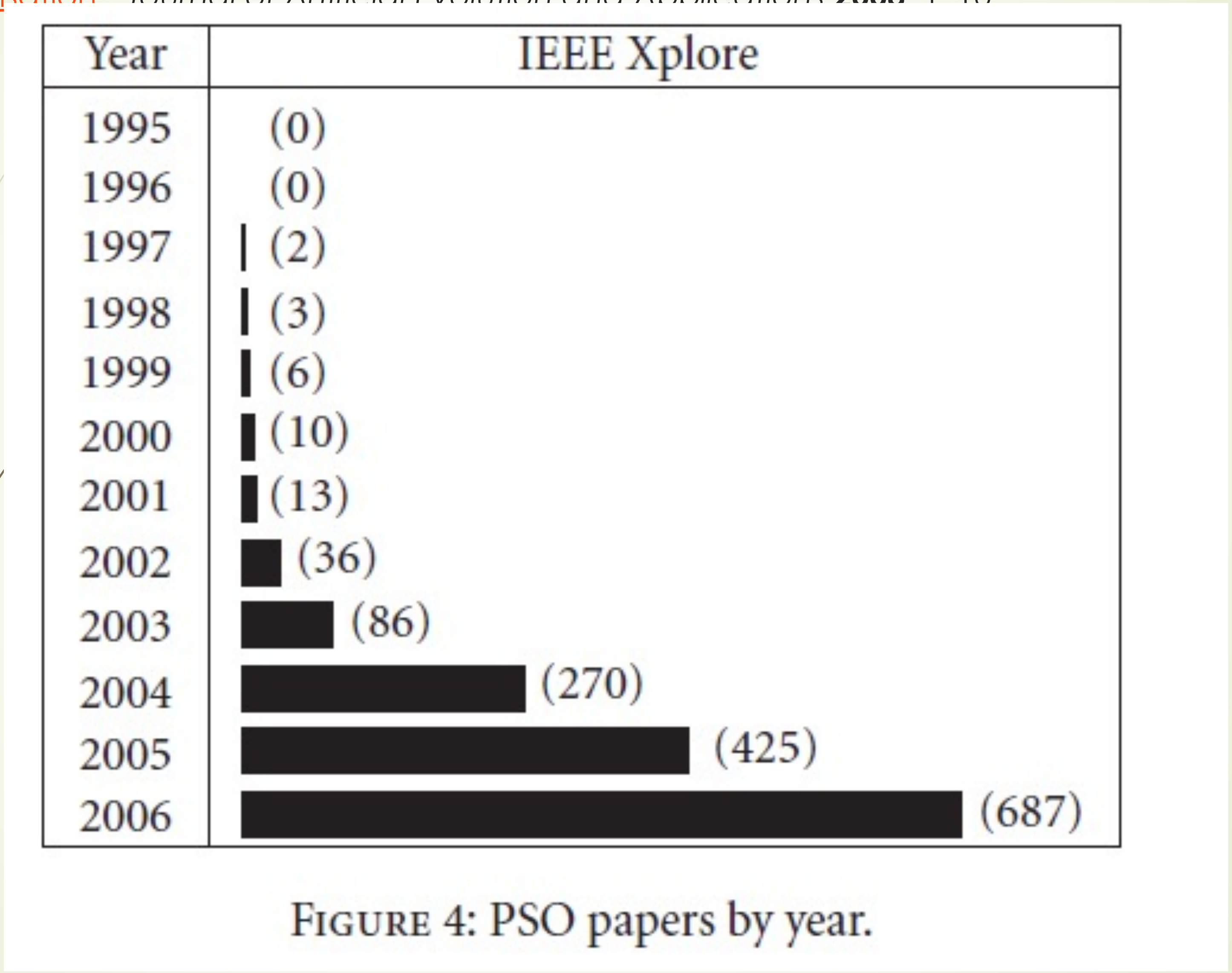
… and some typical results

Optimum=0, dimension=30

Best result after 40 000 evaluations

30D function	PSO Type 1"	Evolutionary algo.(Angeline 98)
Griewank [±300]	0.003944	0.4033
Rastrigin [±5]	82.95618	46.4689
Rosenbrock [±10]	50.193877	1610.359

This is from Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation". *Journal of Artificial Evolution and Applications* **2008**: 1–10

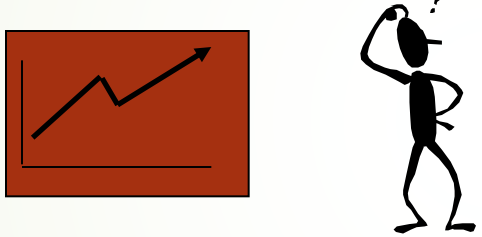


Year	IEEE Xplore
1995	(0)
1996	(0)
1997	(2)
1998	(3)
1999	(6)
2000	(10)
2001	(13)
2002	(36)
2003	(86)
2004	(270)
2005	(425)
2006	(687)

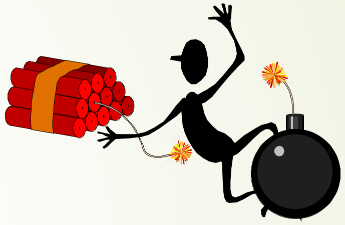
FIGURE 4: PSO papers by year.

Adaptive swarm size

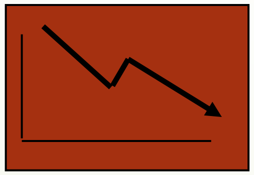
There has been enough improvement
although I'm the worst



I try to kill myself

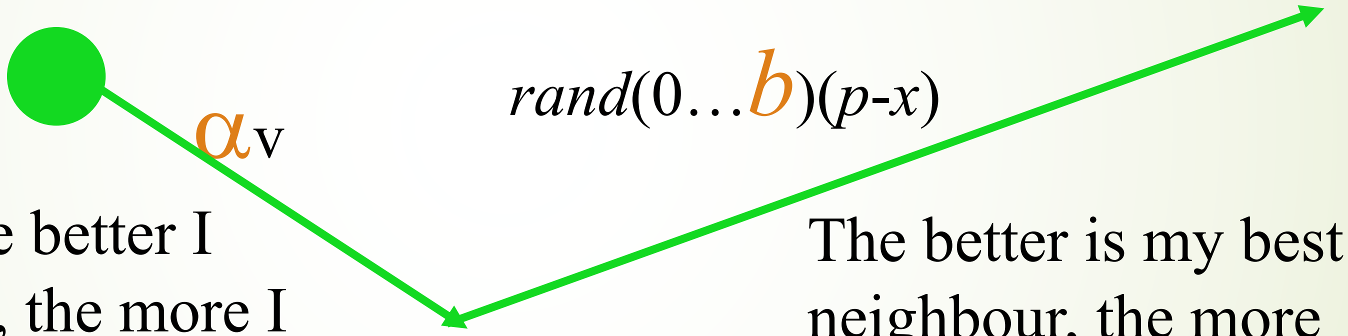


I'm the best
but there has been not enough improvement



I try to generate a new particle

Adaptive coefficients



The better I am, the more I follow my own way

The better is my best neighbour, the more I tend to go towards him