UNIVERSITY OF LIVERPOOL

COMP 532
Machine Learning and BioInspired Optimization

Lecture 23: Swarm Intelligence

Dr. Shan Luo

Department of Computer Science

shan.luo@liverpool.ac.uk

Admin

Reading material Swarm Intelligence (on VITAL)

- On **Particle Swarm Optimisation**:
  Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, *1*(1), 33-57.

- On **Ant Colony Optimisation**:
  Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, *2*(4), 353-373.

- On **Bee-Inspired Systems**:
  Lemmens, N., & Tuyls, K. (2012). Stigmergic landmark optimization. *Advances in Complex Systems*, *15*(08), 1150025.

Overview

- What is Swarm Intelligence?
- Particle Swarm Optimization
  - Origins of PSO: flocking
  - The canonical algorithm
  - Examples & extensions
- Ant Systems and Ant Colony Optimization
- Bee Colonies and Swarm Robotics

Swarm Intelligence

- As we have seen, **Swarm Intelligence** combines **local search** with **global information sharing** through communication.

- E.g. in **Ant Systems**
  - Each ant explores individually (local search)
  - Deposits pheromone to share knowledge to recruit others (information sharing)

Origins of Particle Swarm Optimisation

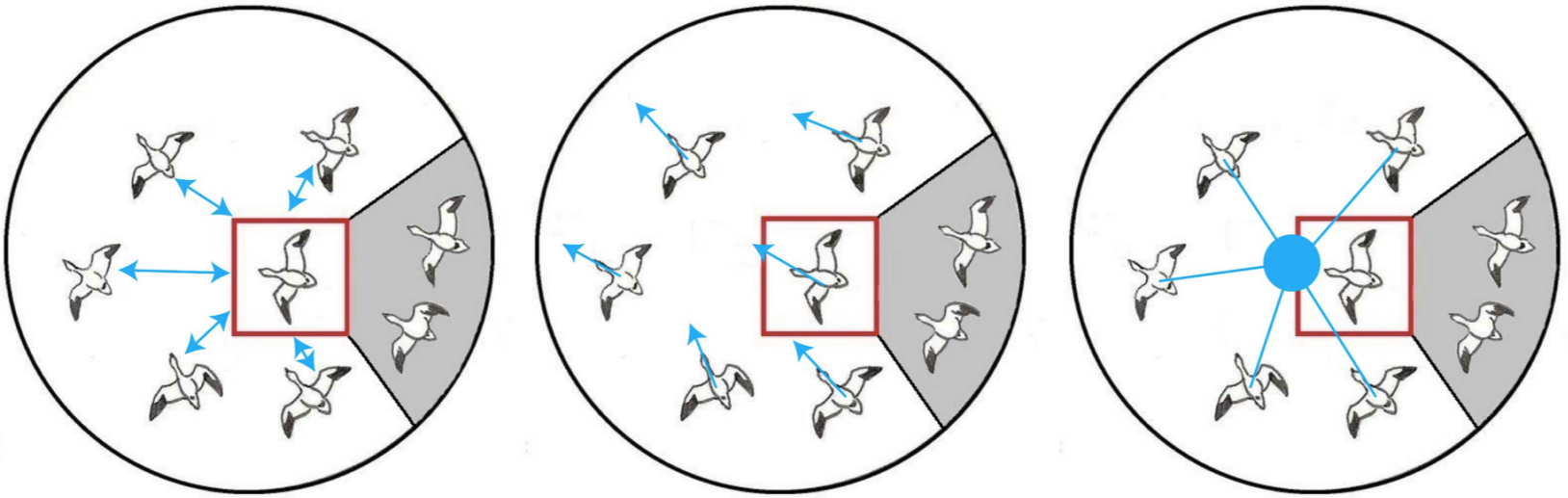
Flocking

How can birds or fish exhibit such incredible coordinated behaviour?

Origins of Particle Swarm Optimisation

In 1986 Craig Reynolds developed a model of flocking (Boids model) based on three simultaneous dynamics.

Separation **Alignment** **Cohesion**



Reynolds, C.W.. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH 1987*.

Origins of Particle Swarm Optimisation

- In 1995, Kennedy & Eberhart added a “roost” to the Reynolds model
  - Served as a global attractor for the flock
  - Each bird remembered where it was closest to the roost and tried to get back
  - Birds shared their knowledge
  - Eventually the whole flock “landed” on the roost

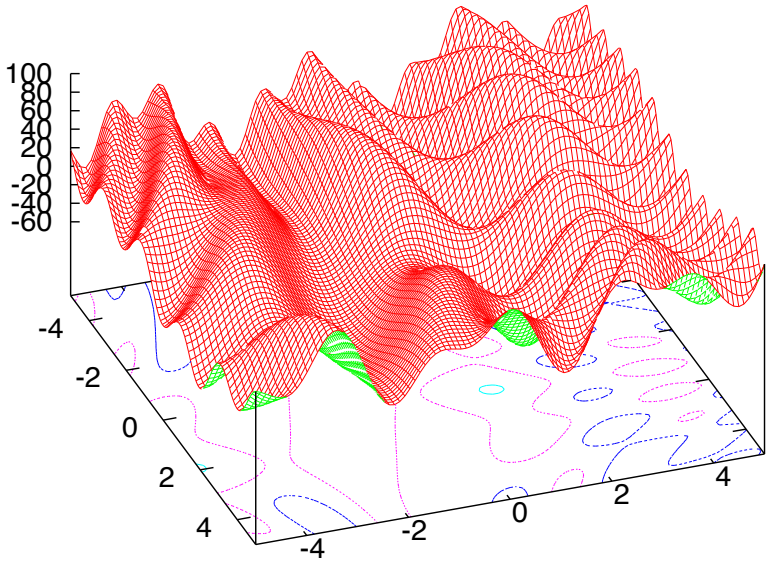
- **Breakthrough**:
  what if the distance to the roost is replaced with an arbitrary function? Will the flock find the minimum?

Eberhart, R. and Kennedy, J., October. A new optimizer using particle swarm theory. *MHS 1995*.

Particle Swarm Optimisation

How to find the optimum?

(Typically: minimise error function)



Particle Swarm Optimisation:

- Particles start in random locations
- Particles remember own best location so far
- Swarm shares global best so far
- Particles always move towards a combination of own best and global best (with some exploration)

The Basic Algorithm

Ingredients:

- Swarm is a set of particles P

- Each particle i has a
  - current position x_i
  - current velocity v_i
  - personal best position found so far: pb_i
  - shared global best found so far: gb

- Particle i has a current fitness $f(x_i)$ which represents the quality of its position

The Basic Algorithm

Algorithm:

- Randomly initialize particle positions and velocities
- While not terminated:
  - For each particle i:
    - Evaluate its fitness $f(x_i)$
    - If $f(x_i)$ is better than $f(\mathrm{pb}_i)$ then update $\mathrm{pb}_i \leftarrow x_i$
    - If $f(x_i)$ is better than $f(\mathrm{gb})$ then update $\mathrm{gb} \leftarrow x_i$
  - For each particle i:
    - Update position as $x_i^{t+1} = x_i^t + v_i^{t+1}$ where

$$v_i^{t+1} = v_i^t + U(0, \beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\mathrm{gb} - x_i^t\right]$$

The Velocity Update

The velocity update rule has three components:

$$v_i^{t+1} = v_i^t + U(0, \beta_1)\left[\text{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\text{gb} - x_i^t\right]$$

momentum

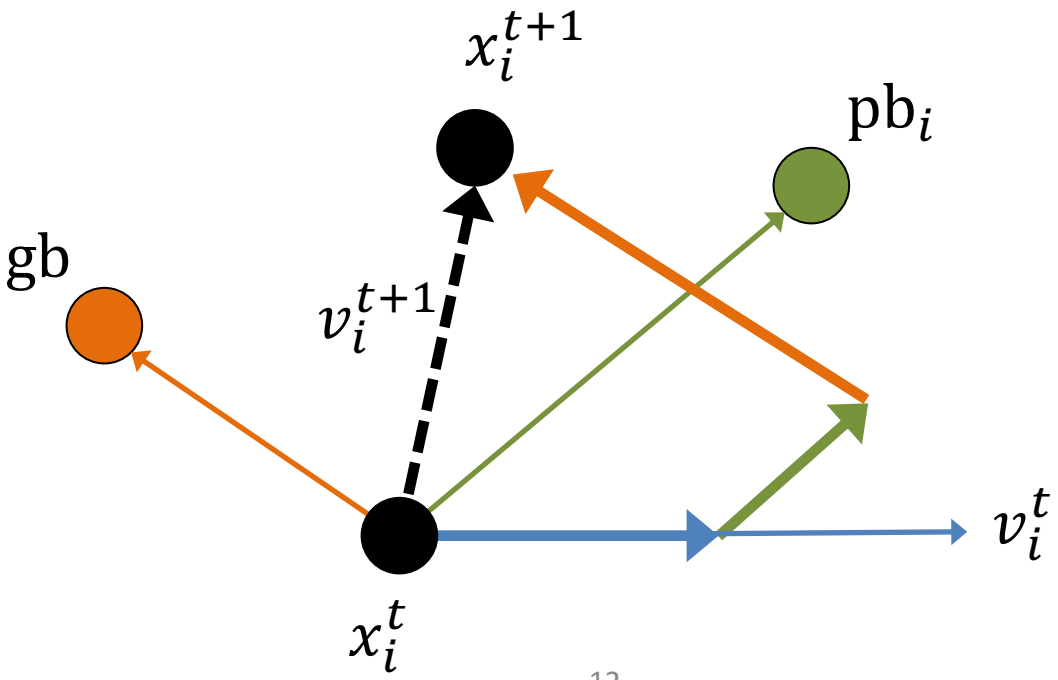
personal influence

social influence

The Velocity Update

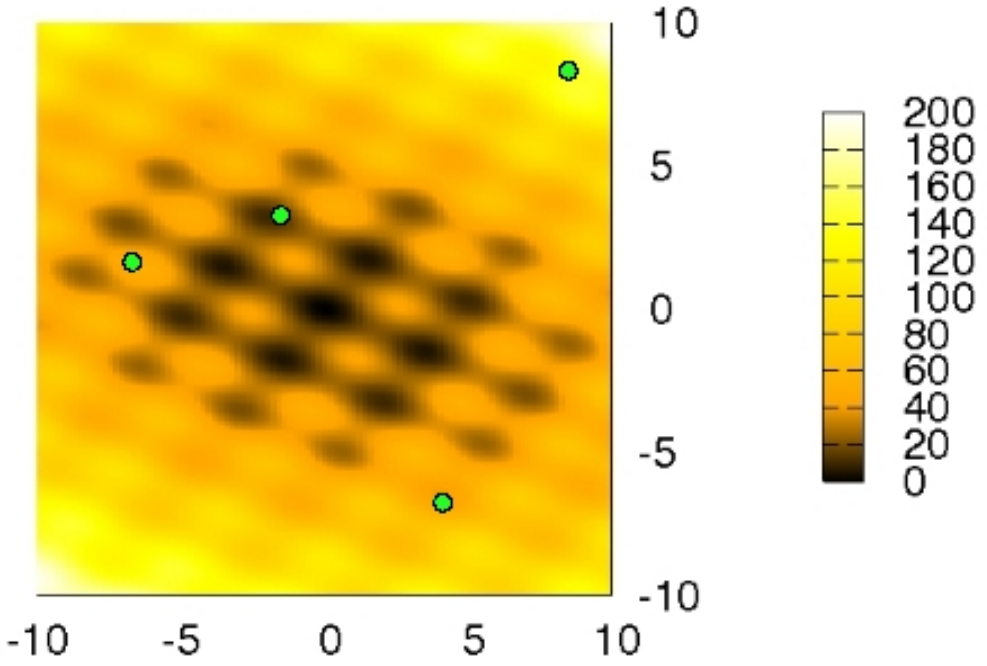
$$v_i^{t+1} = v_i^t + U(0, \beta_1)\left[\text{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\text{gb} - x_i^t\right]$$

In two-dimensional space, the update looks like this:



PSO Example

- Step 1: initialize particles

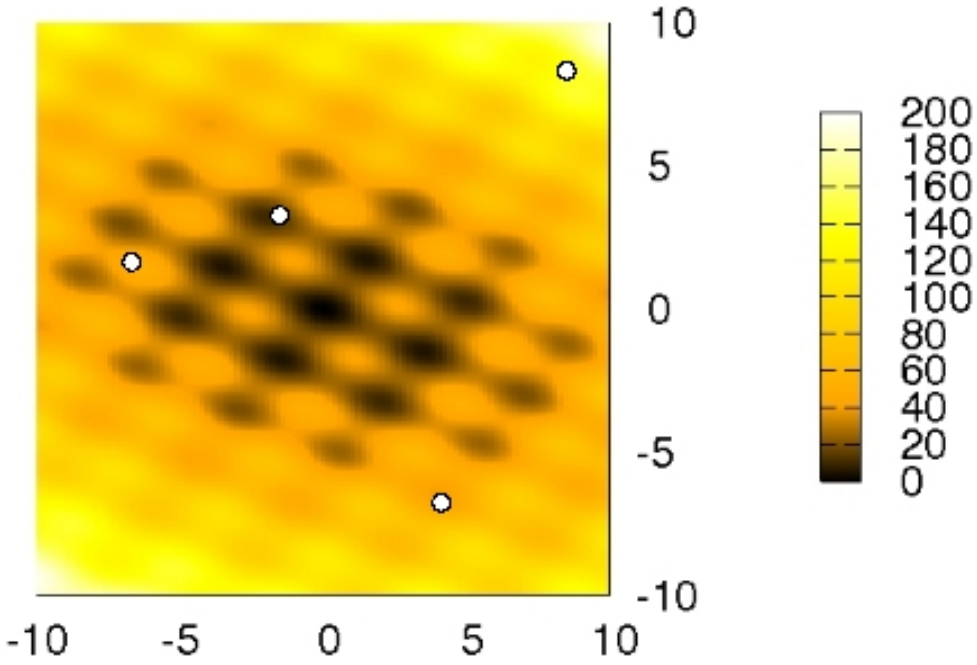


Create a population of agents (called particles) uniformly distributed over X

Example by Marco A. Montes de Oca
IRIDIA-CoDE, Universite Libre de Bruxelles

PSO Example

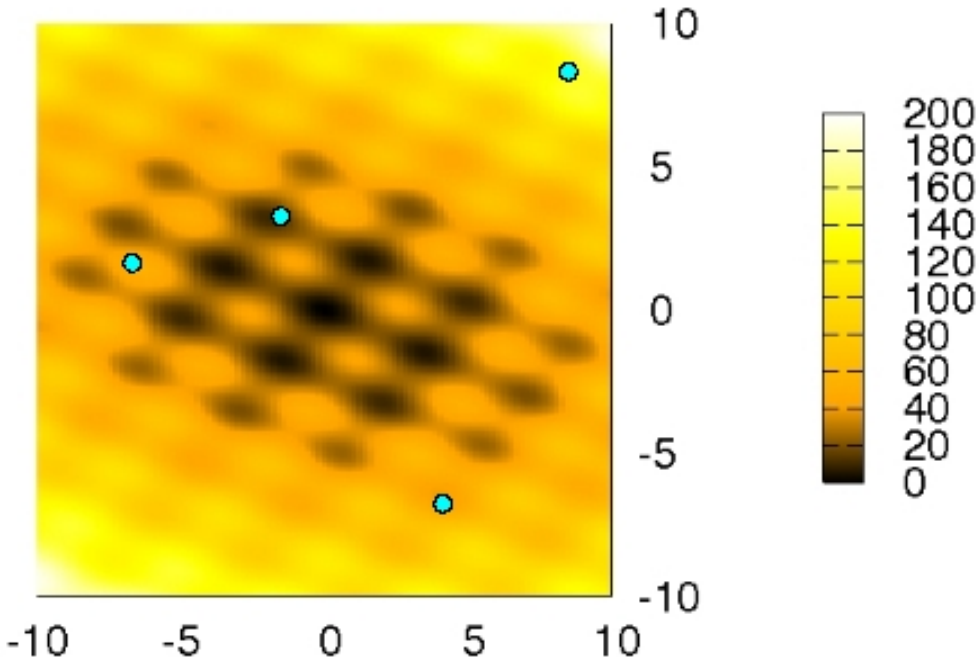
- Step 2: compute fitness



Evaluate each particle's position according to the objective function

PSO Example

- Step 3: update personal best

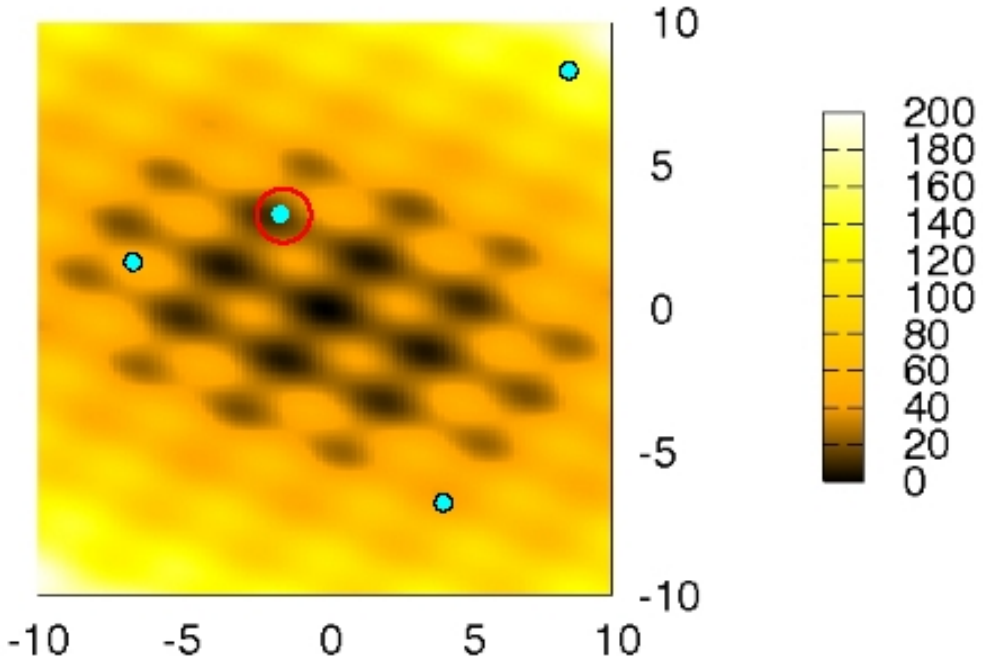


A particle compares its current position with its personal best position.

If there is improvement, the particle‘s current position becomes its new personal best position

PSO Example

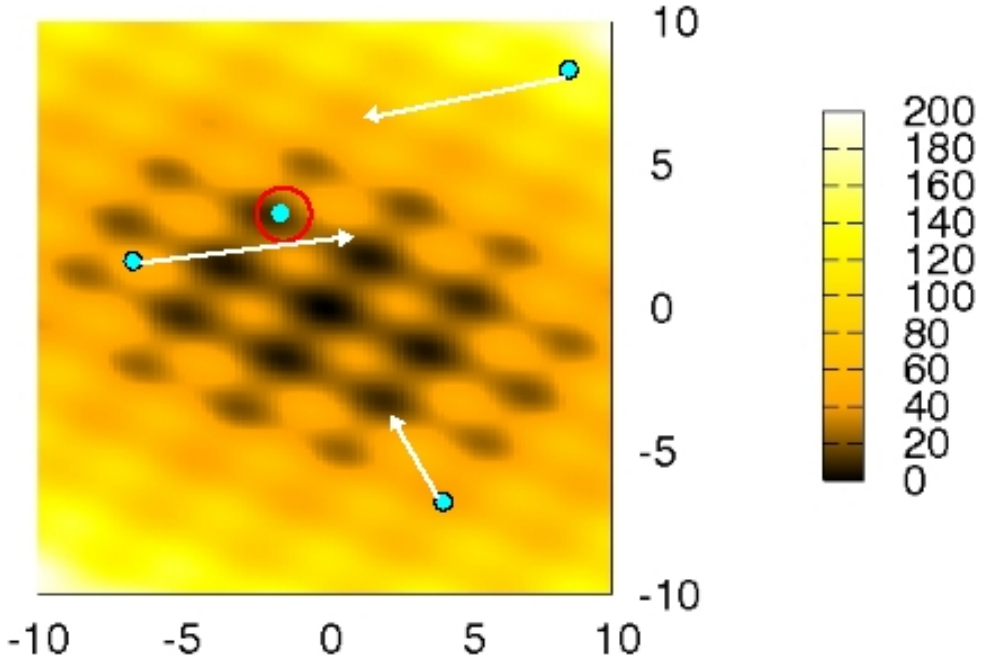
- Step 4: update global best



Determine the best particle

PSO Example

- Step 5: compute new velocities

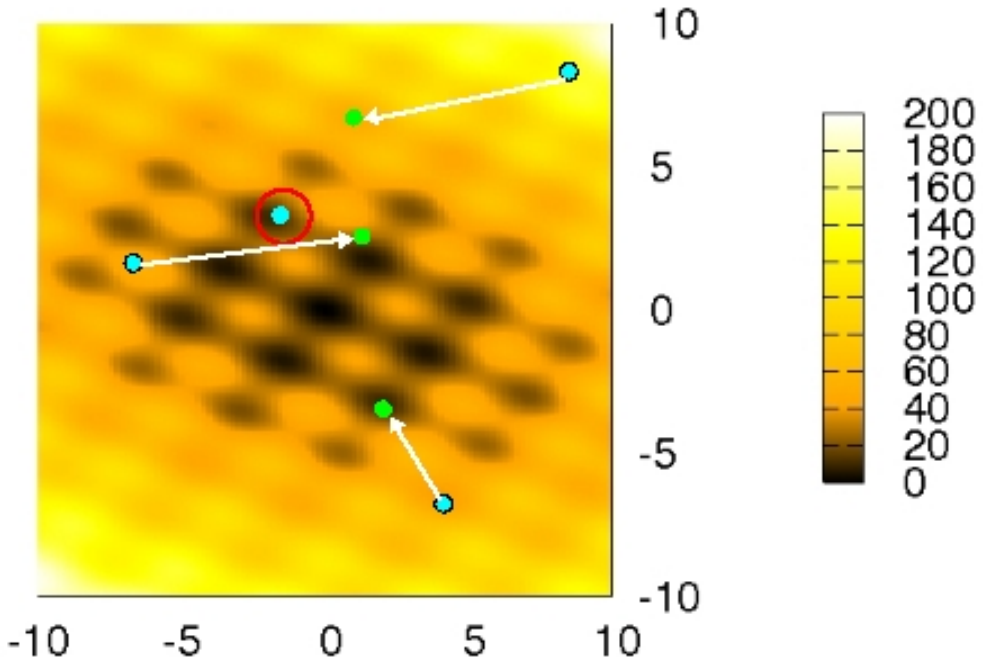


Update particles' velocities according to

$$v_i^{t+1} = v_i^t + U(0,\beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0,\beta_2)\left[\mathrm{gb} - x_i^t\right]$$

PSO Example

- Step 6: move particles to new location

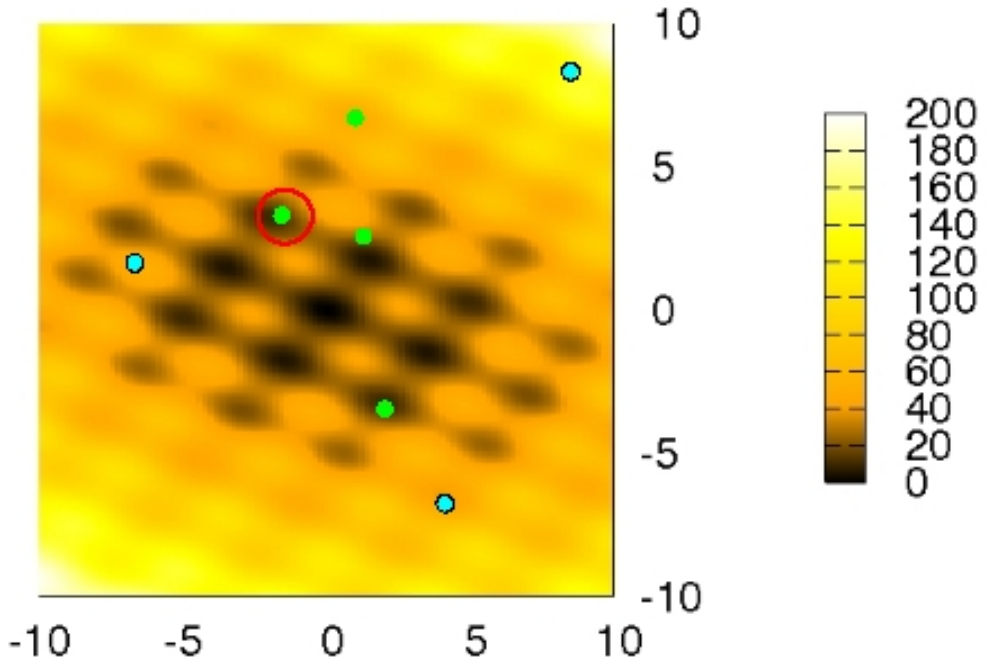


Move particles to their new positions according to

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

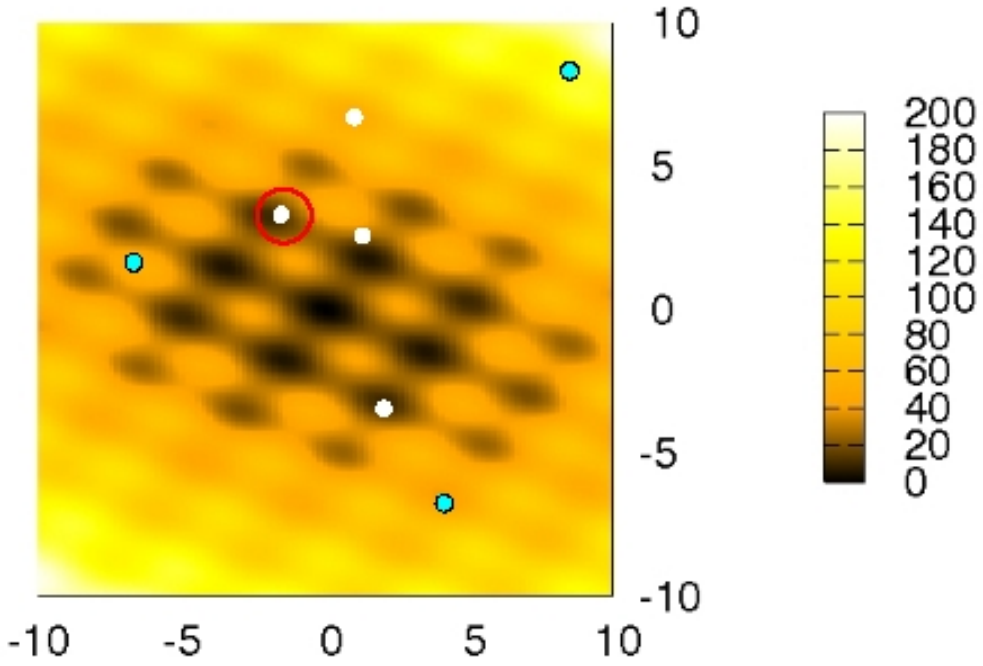
PSO Example

- Go to Step 2 until stopping criteria are met



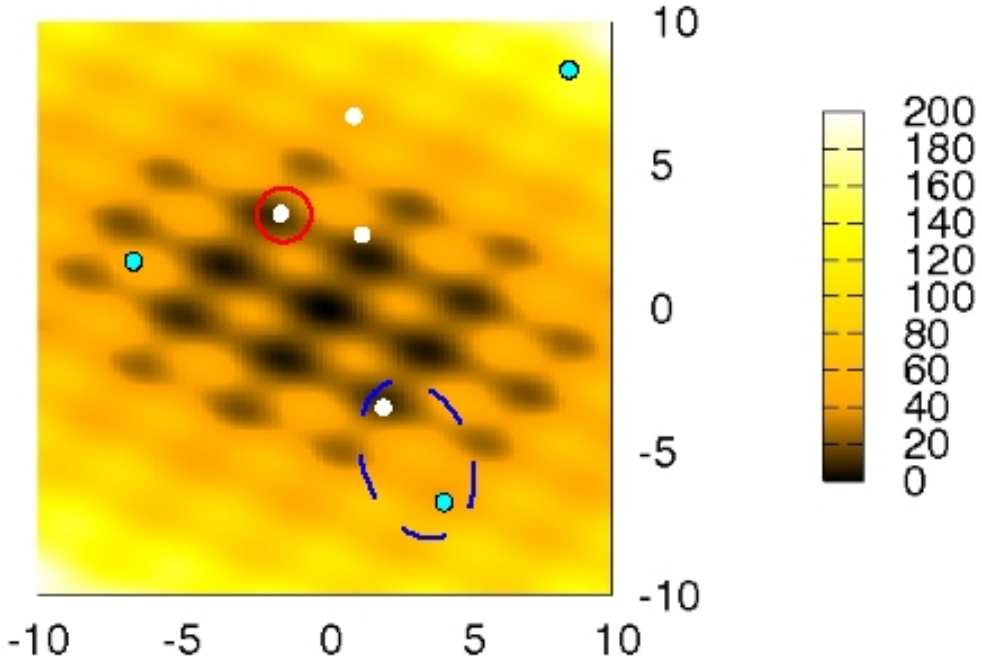
PSO Example

- Step 2: compute fitness



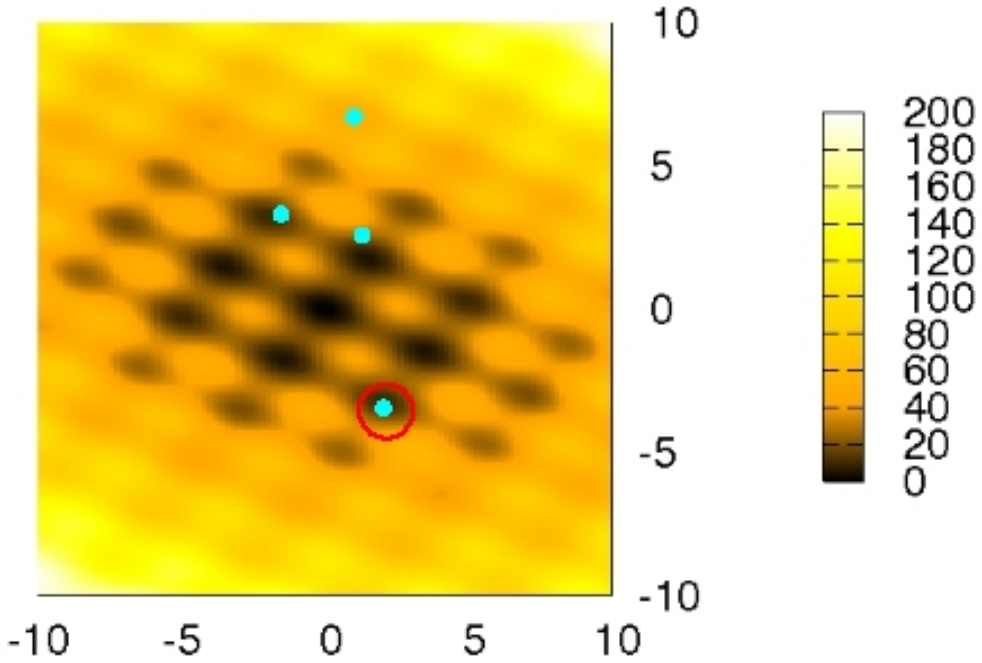
PSO Example

- Step 3: if current fitness is better than personal best, update it



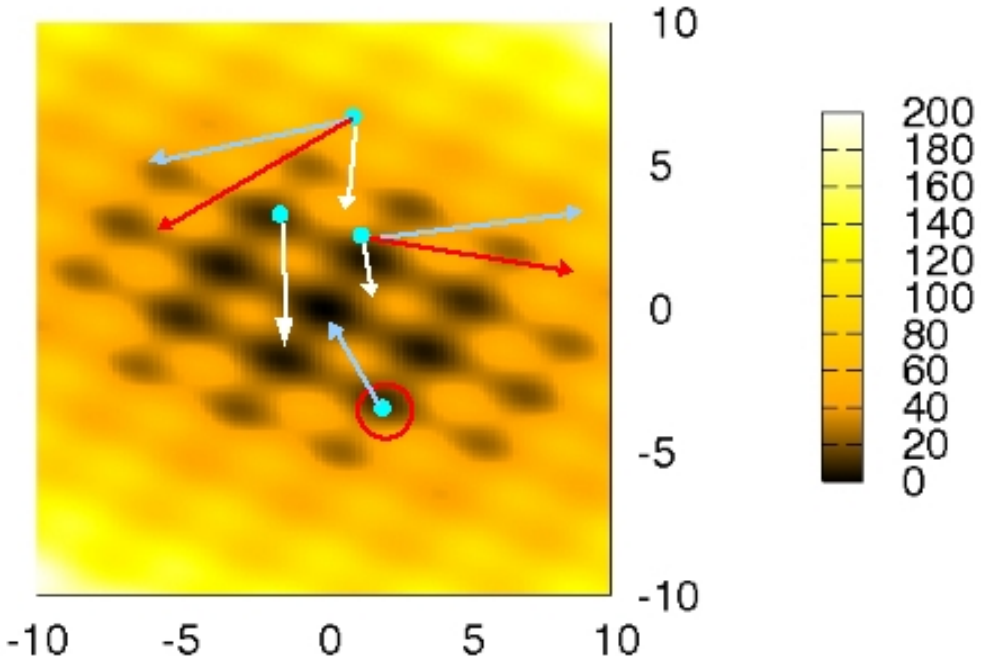
PSO Example

- Step 4: determine the new global best



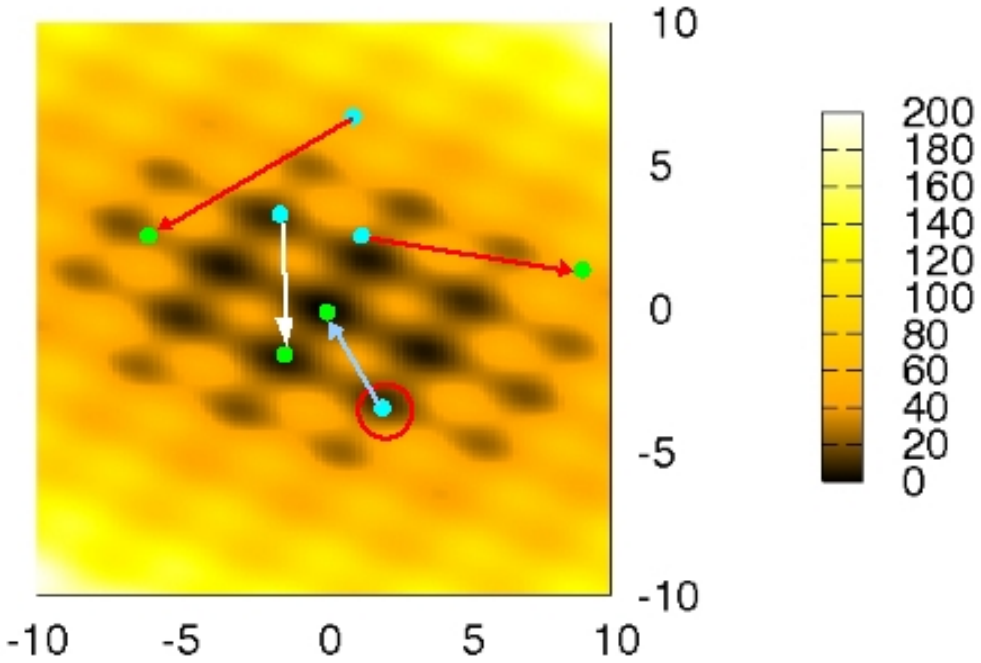
PSO Example

- Step 5: compute new velocities based on inertia, personal best, and global best



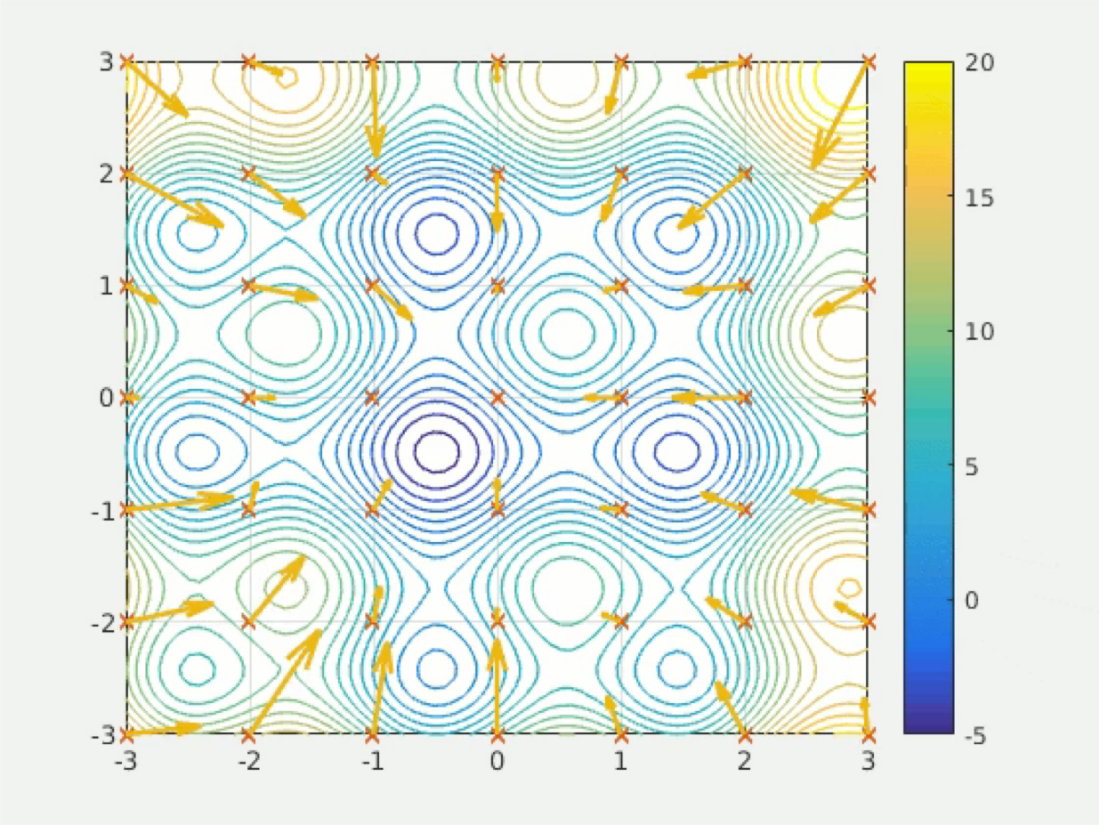
PSO Example

- Step 6: move particles to new location, go back to Step 2, etc…



PSO Example

The end result looks something like this..

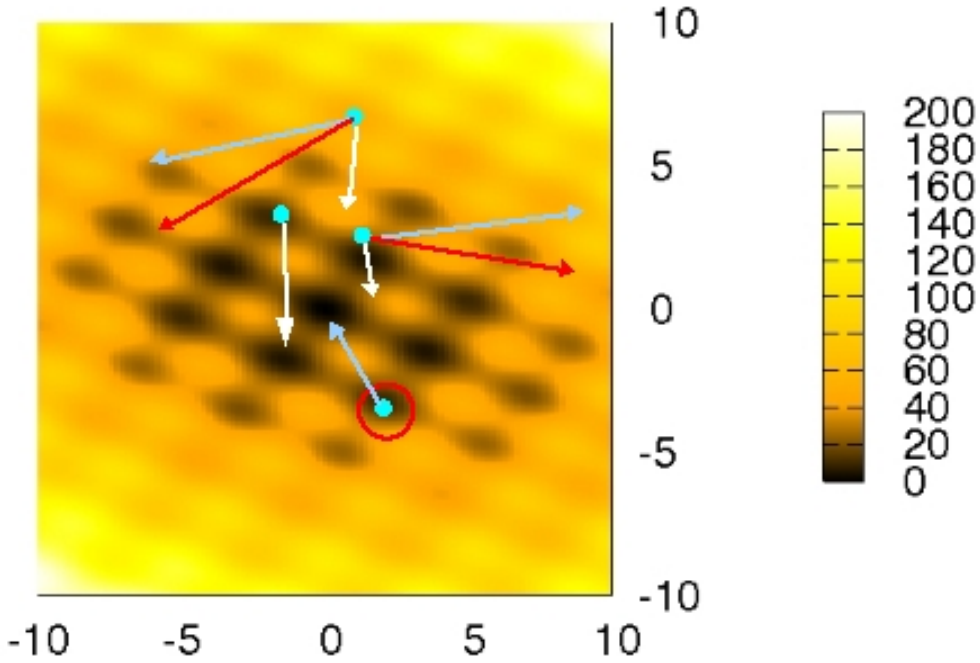


Animation taken from Wikipedia

PSO Example

Algorithm

1. initialize particles
2. compute fitness
3. update personal best
4. update global best
5. compute new velocities based on inertia, personal best, and global best
6. move particles to new location
7. Go to Step 2 until stopping criteria are met



Step 5

Population Topology

- Maintaining only one **global best** may lead the swarm to converge to a local optimum

- Alternative: particles interact only within their neighbourhood
  - Information is shared only locally: **local best**
  - Neighbourhoods can be overlapping

- Helps to maintains diversity
  - slower convergence..
  - but better able to find global optimum!

Population Topology

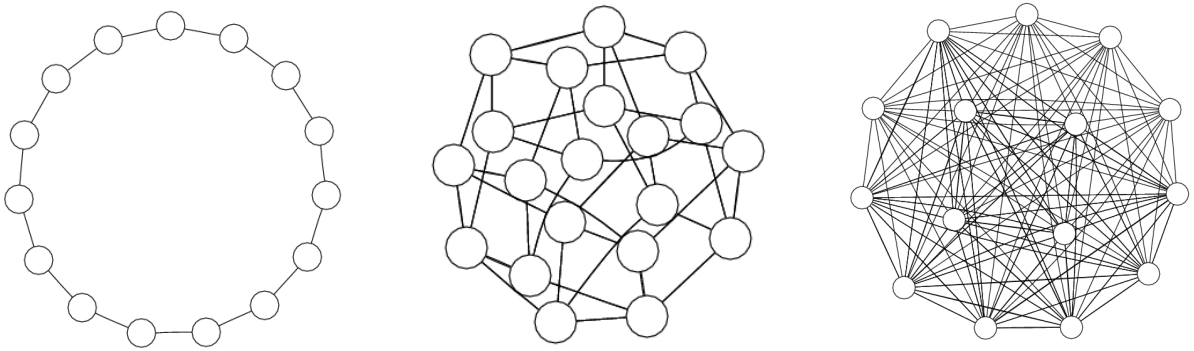
Global best vs **Local best**

$$v_i^{t+1} = v_i^t + U(0, \beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\mathrm{gb} - x_i^t\right]$$

$$v_i^{t+1} = v_i^t + U(0, \beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\mathrm{lb} - x_i^t\right]$$

Neighbourhood Structure

- Different structures can be used



- **Degree** (#neighbours) of particles is important
  - Low degree: slow information spread, slow convergence, but higher chance of finding global optimum
  - High degree: fast information spread, faster convergence, but may get stuck in local (sub)optimum

Momentum Weight

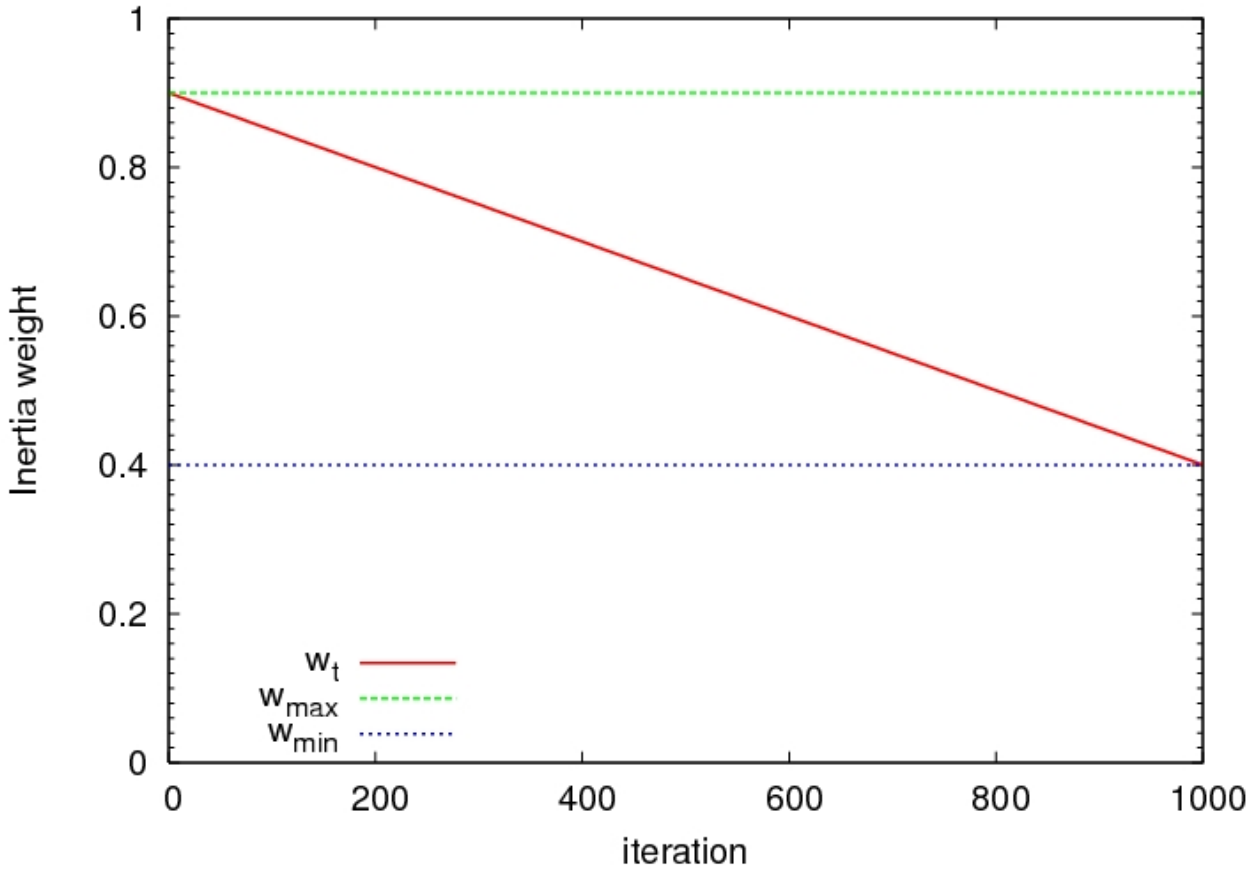
- Momentum serves as a way of insuring **exploration**
- Exploration – exploitation balance using a weight on the momentum term (inertia)

$$v_i^{t+1} = \omega v_i^t + U(0, \beta_1)\left[\text{pb}_i - x_i^t\right] + U(0, \beta_2)\left[\text{gb} - x_i^t\right]$$

- A higher weight tends to cause more exploration, a lower weight causes exploitation
- Typically weight is decreased over time

Momentum Weight

Typical way of decreasing inertia over time



Proposed by Shi & Eberhart

Constriction Coefficient

- Similar to momentum weight, now the whole velocity term is weighted

$$v_i^{t+1} = \chi\left(v_i^t + U(0,\beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0,\beta_2)\left[\mathrm{gb} - x_i^t\right]\right)$$

- Typically χ is chosen together with β_1 and β_2 to ensure convergence
  - Common choice: $\chi = 0.7298$, and $\beta_1 = \beta_2 = 2.05$

- The two approaches are mathematically equivalent

Constriction Coefficient

- Equivalence of Constriction and Inertia methods

$$v_i^{t+1} = \chi\left(v_i^t + U(0,\beta_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0,\beta_2)\left[\mathrm{gb} - x_i^t\right]\right)$$

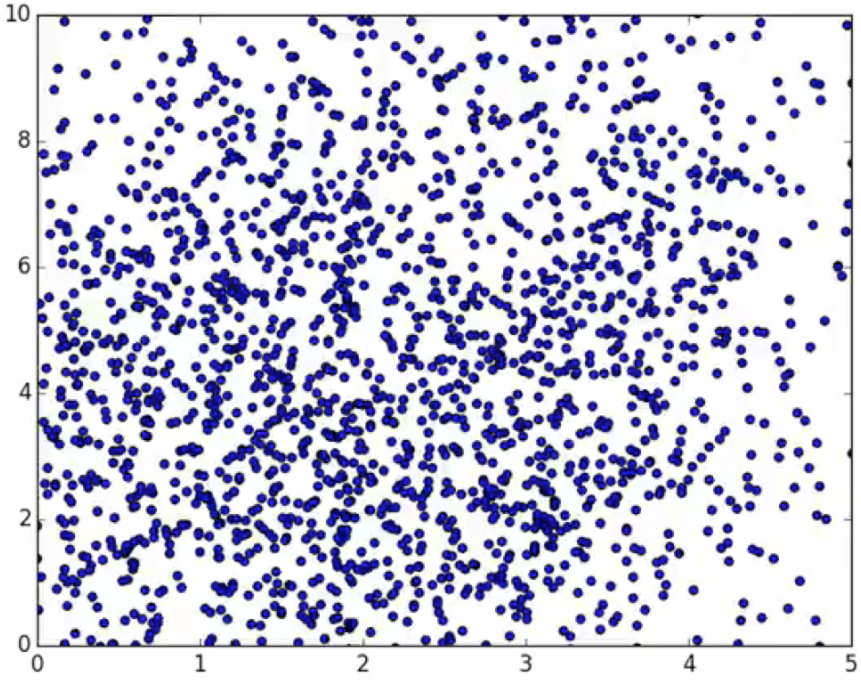
equals

$$v_i^{t+1} = \omega v_i^t + U(0,\beta'_1)\left[\mathrm{pb}_i - x_i^t\right] + U(0,\beta'_2)\left[\mathrm{gb} - x_i^t\right]$$

  - with $\omega = \chi$, $\beta'_1 = \chi\beta_1$, and $\beta'_2 = \chi\beta_2$

PSO in Action

- Example: learning the function $f(x, y) = -|x^2 - y|$
  - That means: finding the squares!



Video by Chase Stevens
University of Edinburgh

Finally..

- Many more variation exist..
  - with dynamic neighborhood
  - with enhanced diversity
  - with different velocity update
  - for discrete optimization problems
  - …
- In many ways related to ideas from learning, gradient ascent, genetic algorithms, ...