Lecture 07 - Real-valued optimisation

Optimisation CT5141

James McDermott

NUI Galway

OÉ Gaillimh
NUI Galway

2020

Overview

1. **Real-valued optimisation: big picture and applications**
2. Covariance Matrix Adaptation
3. Particle Swarm Optimisation

Real-valued optimisation

In real-valued optimisation we have a vector of real DVs. We already saw some black-box algorithms for this:

- In HC, LAHC, SA, GA, we can use `init`, `nbr` (and `crossover` for GA) suitable for real vectors.

LP and gradient descent

If a stronger algorithm is available, we won't use a black-box algorithm:

- Some applications with real DVs have linear objective and constraints, so we would use LP;
- In some applications with real DVs, we can find the gradient, so we would use gradient descent.

Black-box real-valued optimisation

But if we have to, we can use a metaheuristic for real optimisation.

Black-box real-valued optimisation

But if we have to, we can use a metaheuristic for real optimisation.
Now, a confession:
LAHC, SA and GA are not the best metaheuristics for this

- Covariance Matrix Adaptation
- Particle Swarm Optimisation

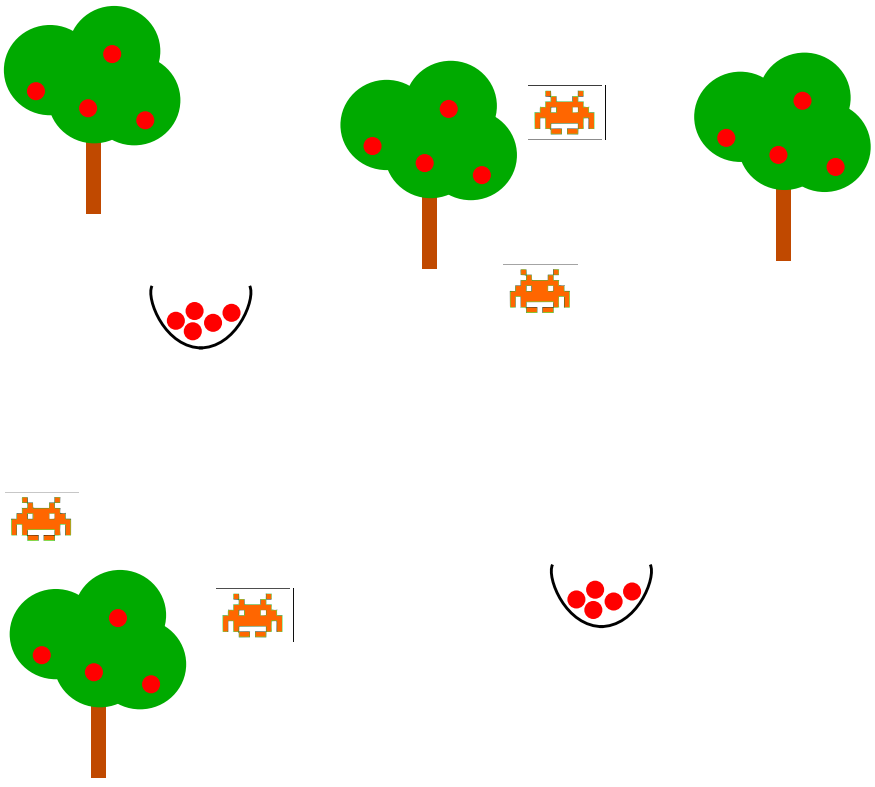
Black-box real-valued optimisation

But if we have to, we can use a metaheuristic for real optimisation.
Now, a confession:
LAHC, SA and GA are not the best metaheuristics for this

- Covariance Matrix Adaptation
- Particle Swarm Optimisation

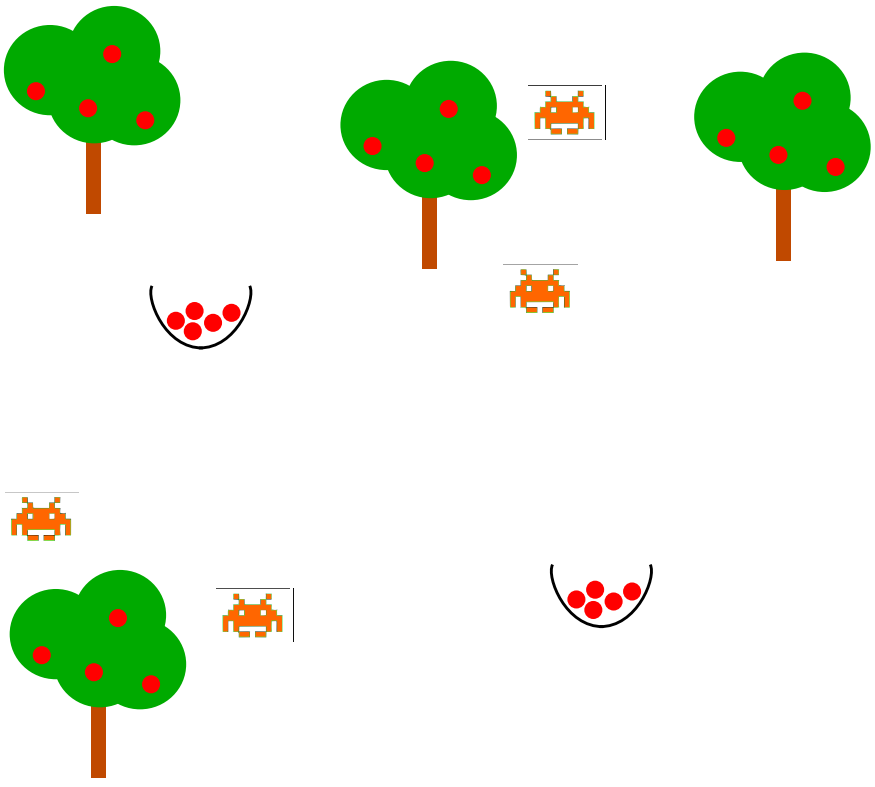
First let's consider applications.

Facility location



Large field with n fruit trees at locations t_i. Tree i has w_i fruit. We have a swarm of tiny flying robot fruit pickers. Each robot picks a fruit and flies to a bin. We can place m bins b_j each with unlimited capacity. Where in the field should we put them?

Facility location

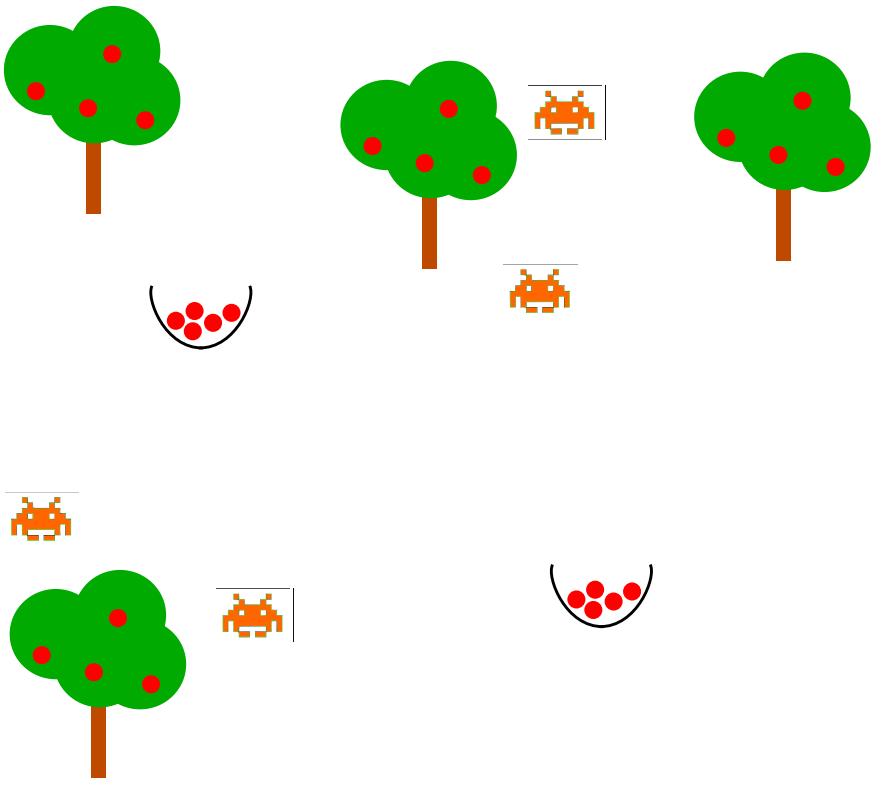


Large field with n fruit trees at locations t_i. Tree i has w_i fruit. We have a swarm of tiny flying robot fruit pickers. Each robot picks a fruit and flies to a bin. We can place m bins b_j each with unlimited capacity. Where in the field should we put them?

Minimise $f(x_1, x_2, \ldots, x_{2m}) = \sum_i w_i \min_j(d(t_i, b_j))$

where d is Euclidean distance.

Facility location



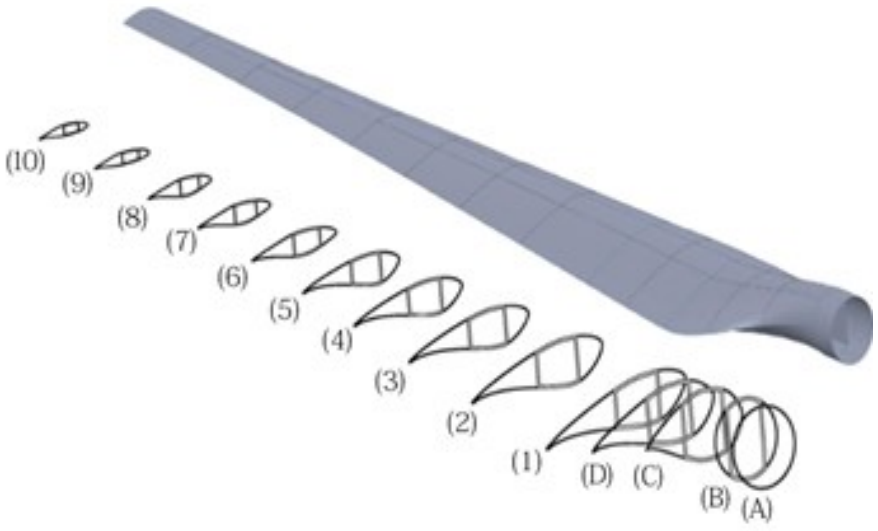
Large field with n fruit trees at locations t_i. Tree i has w_i fruit. We have a swarm of tiny flying robot fruit pickers. Each robot picks a fruit and flies to a bin. We can place m bins b_j each with unlimited capacity. Where in the field should we put them?
Minimise $f(x_1, x_2, \ldots, x_{2m}) = \sum_i w_i \min_j(d(t_i, b_j))$
where d is Euclidean distance.

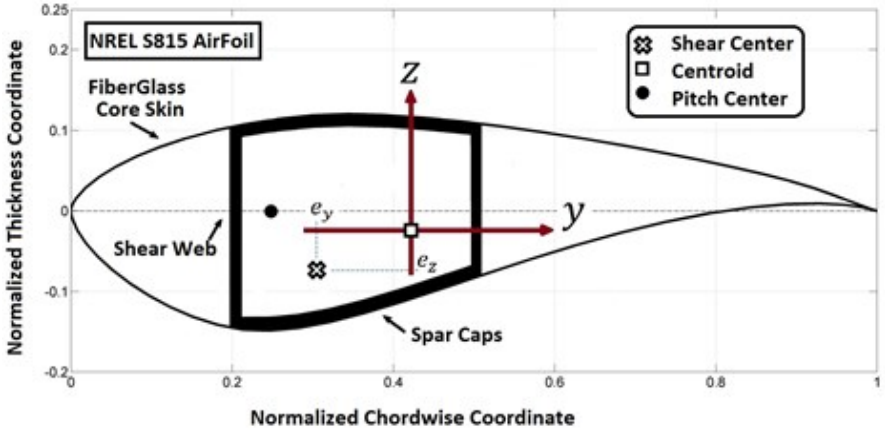
Exactly the same problem arises when choosing where to locate a supermarket to serve several cities.

Wind turbine engineering

In a wind turbine, what **shape** should the sails be?

Several real-valued parameters:



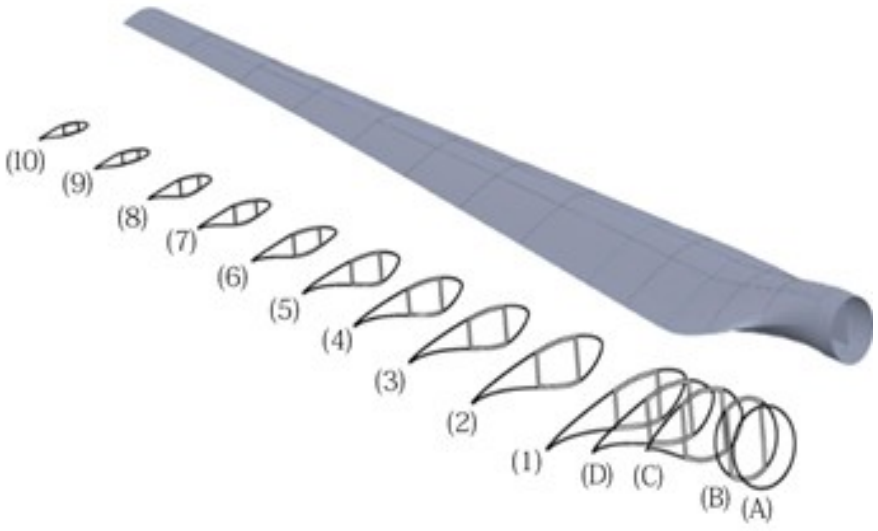


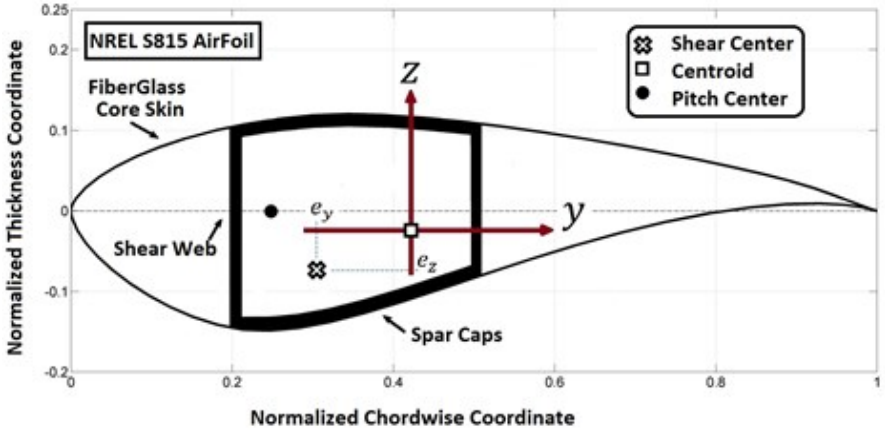
Sheibani and Akbari

Wind turbine engineering

In a wind turbine, what **shape** should the sails be?

Several real-valued parameters:

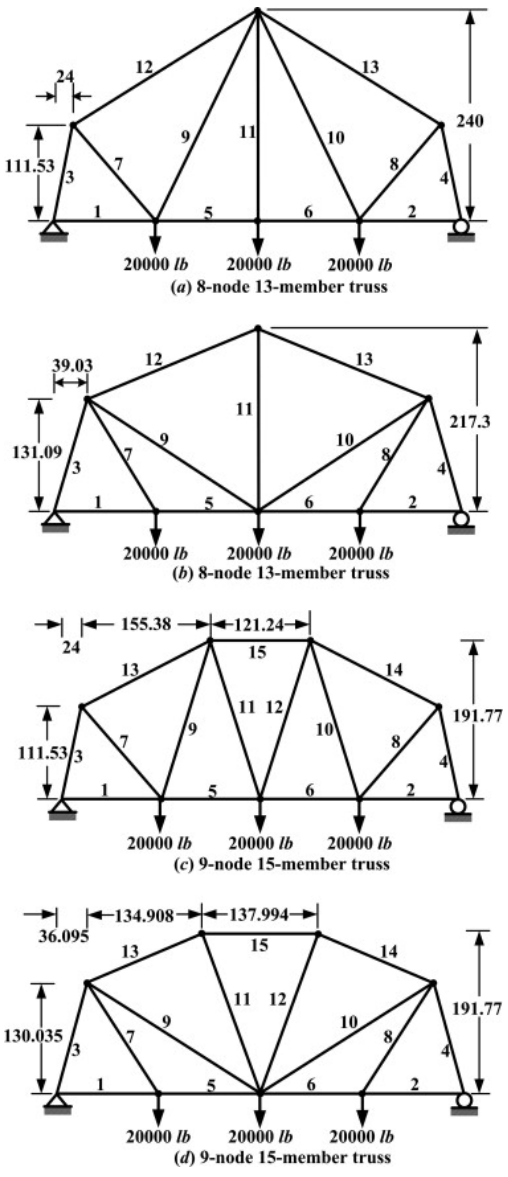




Sheibani and Akbari

We program a finite-element simulation to estimate the **annual energy production** for a given shape.

A-frame truss design



From Luh and Lin

- **Truss**: a 2D or 3D engineering design of beams or rods, attached at nodes – like in the attic of a house.
- Calculate behaviour under gravity, taking account of beam width/weight etc.
- Numerical parameters: node positions and beam widths.
- Objective: minimise weight, minimise deflection, prevent collapse.
- Simple example: https://pythonhosted.org/pyswarm/
- An interactive setting: https://www.polybridge2.com/

FM Synthesis

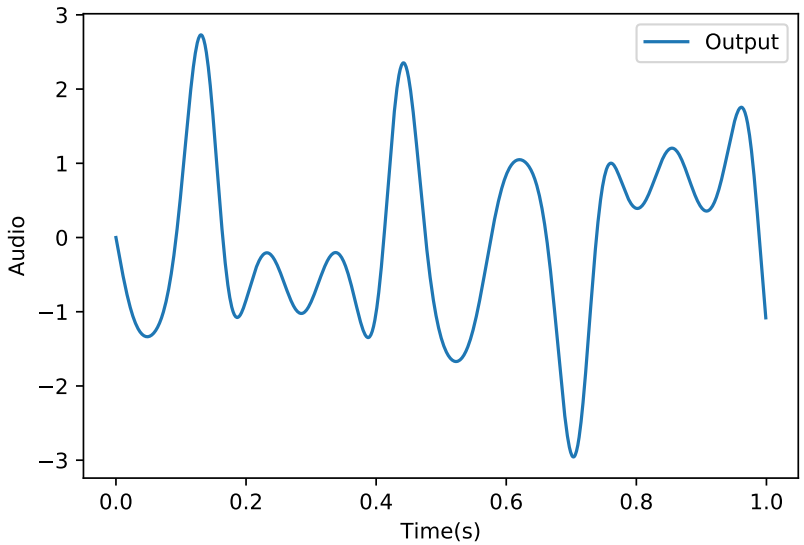
- Frequency-modulation synthesis is an approach to sound synthesis for music
- Invented by Chowning and popularised by Yamaha
- Common in 1980s synth music, e.g. Blade Runner
- Core idea: to take the sine of a sine, giving a sound spectrum with many frequencies.

FM Synthesis control problem

The output of an FM Synth at time t is:

$$\hat{y}(t) = a_1 \sin(\omega_1 t\theta) + a_2 \sin(\omega_2 t\theta + a_3 \sin(\omega_3 t\theta))$$

where $\theta = 2\pi/f_s$, and f_s is sampling frequency, e.g. 44.1kHz.

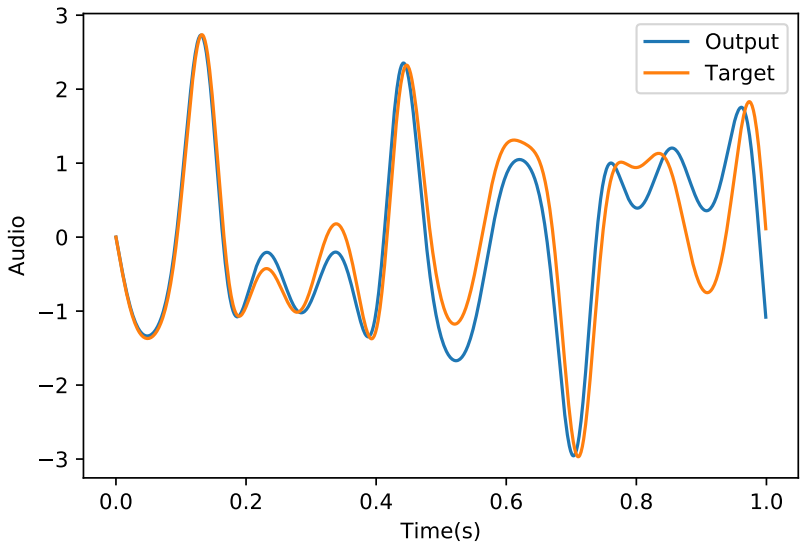


FM Synthesis control problem

We are given some target sound y which we wish to match using FM Synthesis. We **minimise**:

$$f(a_1, \omega_1, a_2, \omega_2, a_3, \omega_3) = \text{RMSE}(y, \hat{y}).$$

(The parameters are constrained, e.g. to $[-6.4, 6.35]$)

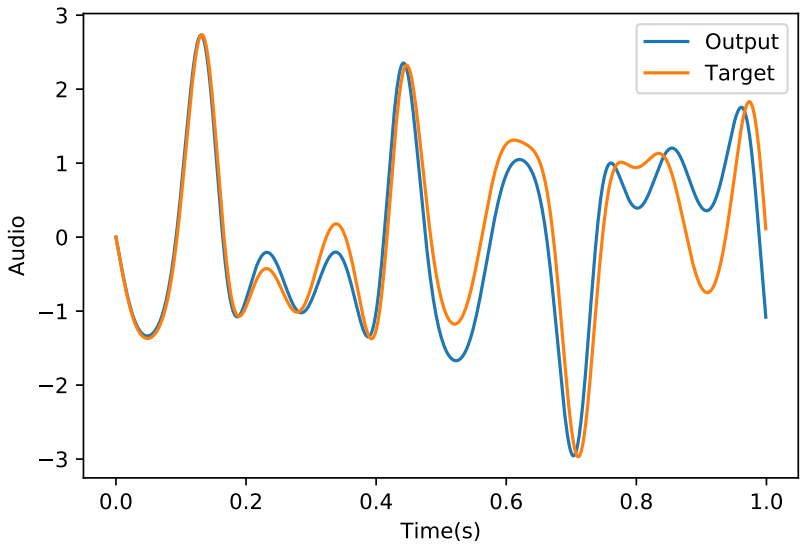


FM Synthesis control problem

We are given some target sound y which we wish to match using FM Synthesis. We **minimise**:

$$f(a_1, \omega_1, a_2, \omega_2, a_3, \omega_3) = \text{RMSE}(y, \hat{y}).$$

(The parameters are constrained, e.g. to $[-6.4, 6.35]$)



(Formulation from Das and Suganthan, 2011, in Blackboard.)

FM Synthesis control problem

(My PhD was in this area, but with a different type of synthesizer. E.g., I experimented with other objectives that work better than RMSE.)

Some more applications

Again see Das and Suganthan (2011) in Blackboard:

- Radar pulse code optimisation
- Antenna array
- Economic dispatch (like unit commitment)
- Spacecraft trajectory optimisation

Overview

1. Real-valued optimisation: big picture and applications
2. **Covariance Matrix Adaptation**
3. Particle Swarm Optimisation

How does a GA work? Another view

- In a GA, the population implicitly represents our knowledge of f
- Think of the population as samples from a distribution
- We draw new samples using crossover and mutation
- These new samples are concentrated in high-fitness regions thanks to selection.

Genetic operators

- Mutation takes in **one** individual and returns a new one

Genetic operators

- Mutation takes in **one** individual and returns a new one
- Crossover takes in **two** individuals and returns one or two new ones

Genetic operators

- Mutation takes in **one** individual and returns a new one
- Crossover takes in **two** individuals and returns one or two new ones
- Could we have 3-parent operators, or higher?

Genetic operators

- Mutation takes in **one** individual and returns a new one
- Crossover takes in **two** individuals and returns one or two new ones
- Could we have 3-parent operators, or higher?
- These do exist, but are rarely used

Genetic operators

- Mutation takes in **one** individual and returns a new one
- Crossover takes in **two** individuals and returns one or two new ones
- Could we have 3-parent operators, or higher?
- These do exist, but are rarely used
- But taking a blend of **all parents at once** is quite common: it is called **Estimation of Distribution**.

Estimation of Distribution Algorithms

The **Estimation of Distribution Algorithm** (EDA):

1. Create an initial population
2. Discard the worst according to f
3. **Estimate** a statistical model of the remainder
4. **Draw samples** from the model to create new population
5. Go to 2.

Estimation of Distribution Algorithms

The **Estimation of Distribution Algorithm** (EDA):

1. Create an initial population
2. Discard the worst according to f
3. **Estimate** a statistical model of the remainder
4. **Draw samples** from the model to create new population
5. Go to 2.

Think of this as a **crossover** which blends **all parents at once**.

Simple real EDA

1. Create population using `init`
2. Evaluate and discard worst to give "parents" P
3. Create one vector representing the **mean**

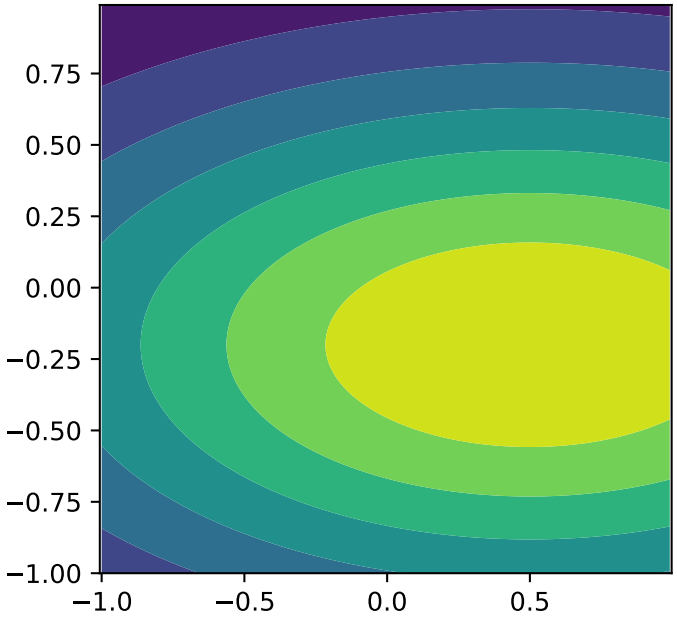
$$\mu \in \mathbb{R}^n : \mu_i = \frac{1}{|P|} \sum_{x \in P} x_i$$

and one representing the **standard deviation**

$$\sigma \in \mathbb{R}^n : \sigma_i = \sqrt{\frac{1}{|P|} \sum_{x \in P} (x_i - \mu_i)^2}$$

4. Draw new individuals x by sampling: $x_i \sim N(\mu_i, \sigma_i)$
5. Go to 2.

What will happen?



$$\mu = (0.5, -0.25), \sigma = (2.0, 0.5)$$

- μ_i will gradually move towards the best value for DV x_i
- But this is **independent** of the values at other i
- σ_i will gradually decrease (the distribution narrows).

Reminder: linkage

Linkage between DVs makes problems harder.

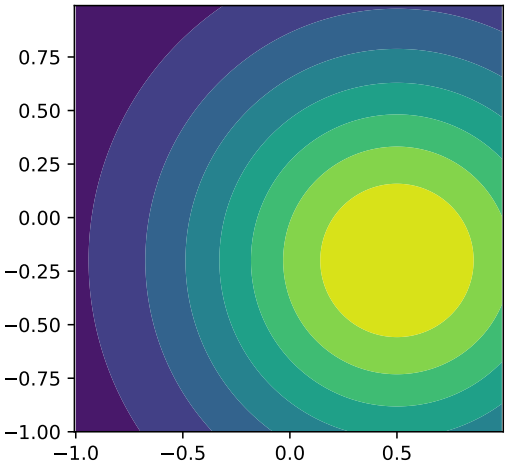
Other words for linkage:

- dependency
- epistasis
- synergy
- “more than the sum of its parts”.

Problem: our EDA is too simple

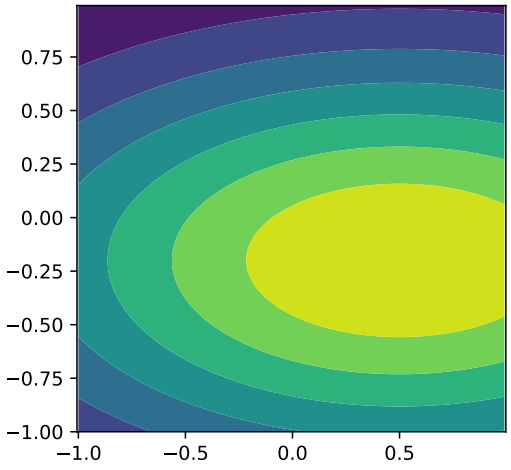
- All positions x_i are considered **independently**
- Cannot learn about linkage between two variables
- But the **covariance** between two variables would give that information
- (Covariance is just **correlation** in disguise)
- **Solution**: use a more complex distribution with covariance

Multivariate Normal: $N(\mu, \Sigma)$



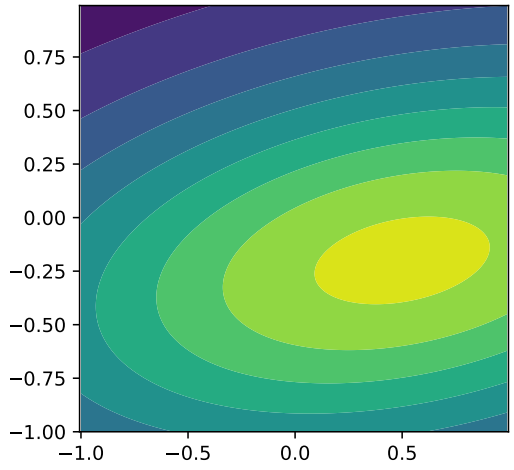
Spherical

$$\Sigma = \begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{bmatrix}$$



Diagonal

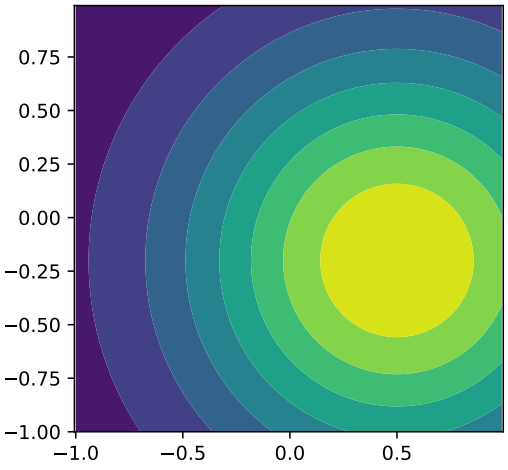
$$\Sigma = \begin{bmatrix} 2.0 & 0.0 \\ 0.0 & 0.5 \end{bmatrix}$$



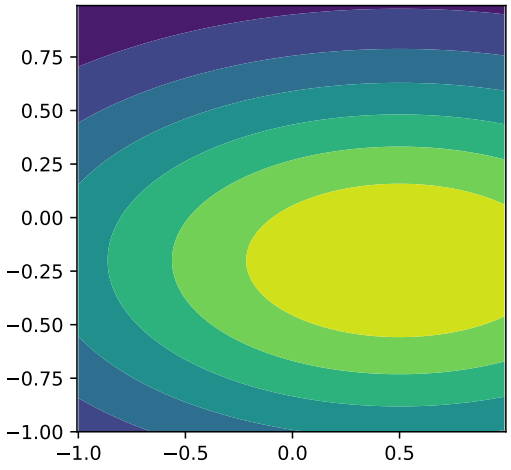
Full

$$\Sigma = \begin{bmatrix} 2.0 & 0.3 \\ 0.3 & 0.5 \end{bmatrix}$$

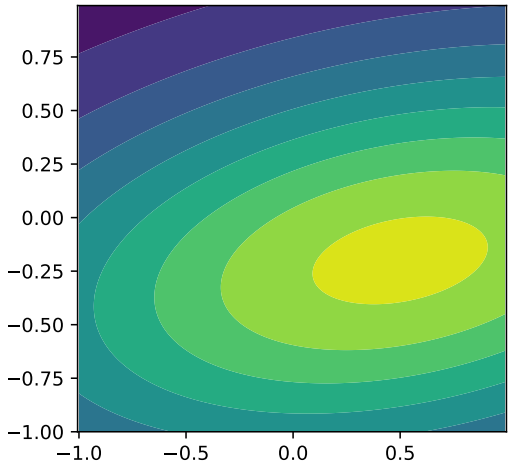
Multivariate Normal: $N(\mu, \Sigma)$



Spherical



Diagonal



Full

$$\Sigma = \begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 2.0 & 0.0 \\ 0.0 & 0.5 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 2.0 & 0.3 \\ 0.3 & 0.5 \end{bmatrix}$$

Don't get confused: **diagonal** means Σ is diagonal. The distribution is then **axis-aligned**. A **full** matrix is more flexible (still must be symmetric): distribution can be diagonal.

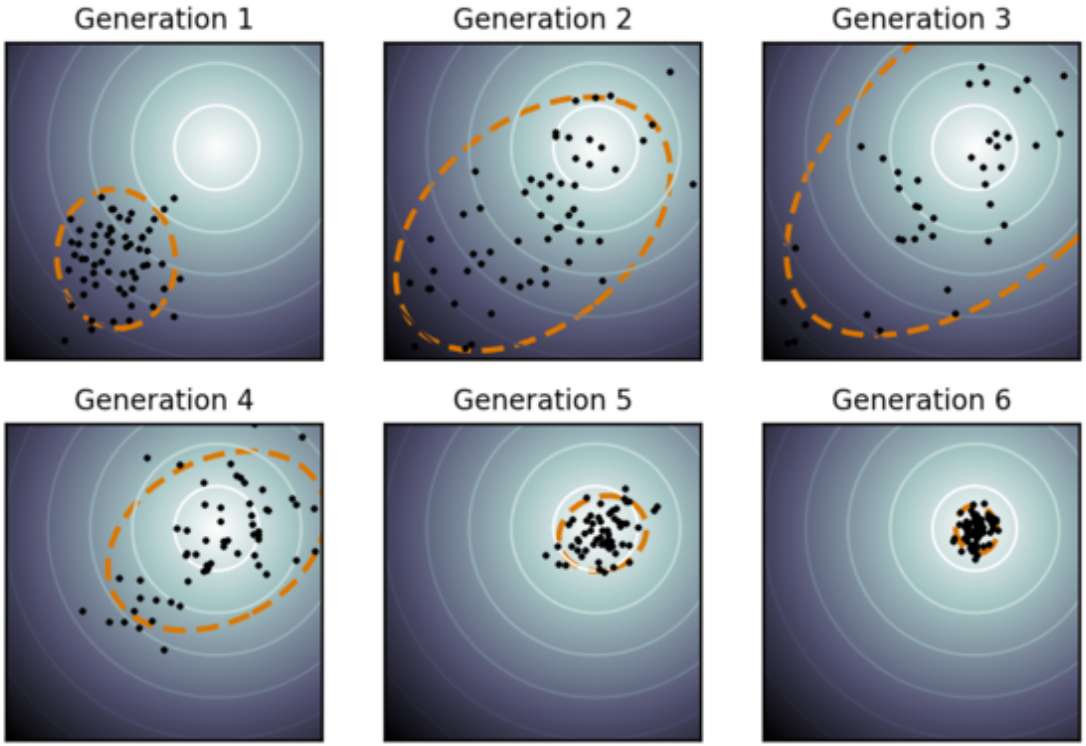
Covariance matrix

The covariance matrix Σ gives the **shape** of the distribution as shown.
In our simple EDA we had a **diagonal** Σ: no linkage.

Covariance Matrix Adaptation

- In CMA, we model the population as a **multivariate normal**
- For each decision variable we have a **mean**
- We also have a **full covariance matrix**.
- (Think of the distribution as an ellipse, non-axis aligned)

CMA: illustration of a run



Wiki

In this picture, fitness is shown as grey-scale. The current population is modelled using a multivariate normal with full covariance. This distribution changes location **and shape** to model the population.

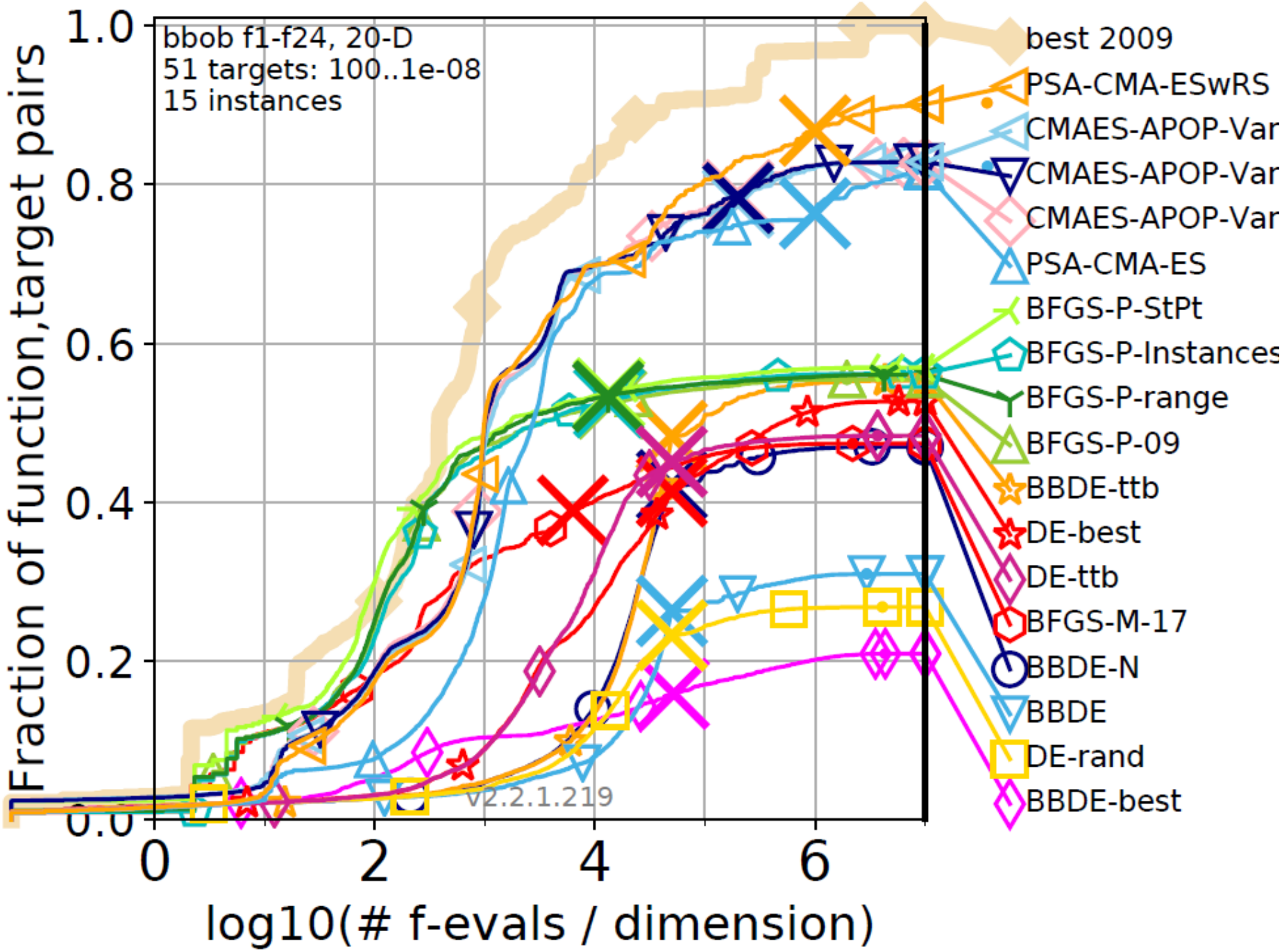
CMA

- Uses **standard** statistical methods to draw samples from the multivariate normal, giving the new population
- Also use some **sophisticated** methods to update the distribution based on new population at each step
- (i.e. don't just re-learn it from scratch)
- It also **self-adapts** many hyperparameters
- Further details are omitted and will not be examinable.

CMA: performance?

CMA and variants often win the “Black-box optimisation benchmarking” competition (BBOB)

See https://coco.gforge.inria.fr/.


bbob f1-f24, 20-D
51 targets: 100..1e-08
15 instances
Fraction of function,target pairs
log10(# f-evals / dimension)
v2.2.1.219
best 2009
PSA-CMA-ESwRS
CMAES-APOP-Var
CMAES-APOP-Var
CMAES-APOP-Var
PSA-CMA-ES
BFGS-P-StPt
BFGS-P-Instances
BFGS-P-range
BFGS-P-09
BBDE-ttb
DE-best
DE-ttb
BFGS-M-17
BBDE-N
BBDE
DE-rand
BBDE-best

Using CMA in practice

```
$ pip install cma
```

```
import cma
 ...
# pass in initial mean and sigma - just a guess
es = cma.CMAEvolutionStrategy(mu0, sigma0,
                              {'bounds': (lb, ub)
                               'seed': seed})
es.optimize(f, iterations=maxits / es.popsize)
print(es.result)
```

Using CMA in practice

```
$ pip install cma
```

```
import cma
 ...
# pass in initial mean and sigma - just a guess
es = cma.CMAEvolutionStrategy(mu0, sigma0,
                              {'bounds': (lb, ub)
                               'seed': seed})
es.optimize(f, iterations=maxits / es.popsize)
print(es.result)
```

It chooses good hyperparameter values by itself :)

Overview

1. Real-valued optimisation: big picture and applications
2. Covariance Matrix Adaptation
3. **Particle Swarm Optimisation**

Bio-inspired optimisation

- Evolutionary algorithms: inspired by Darwinian evolution
- **Particle swarm optimisation**: inspired by bird flocking behaviour
- Ant colony optimisation: inspired by ant foraging

Flocking

Murmuration: https://www.youtube.com/watch?v=eakKfY5aHmY

Flocking seagulls

Boids and Crowds

Reynolds' Boids website: http://www.red3d.com/cwr/boids/

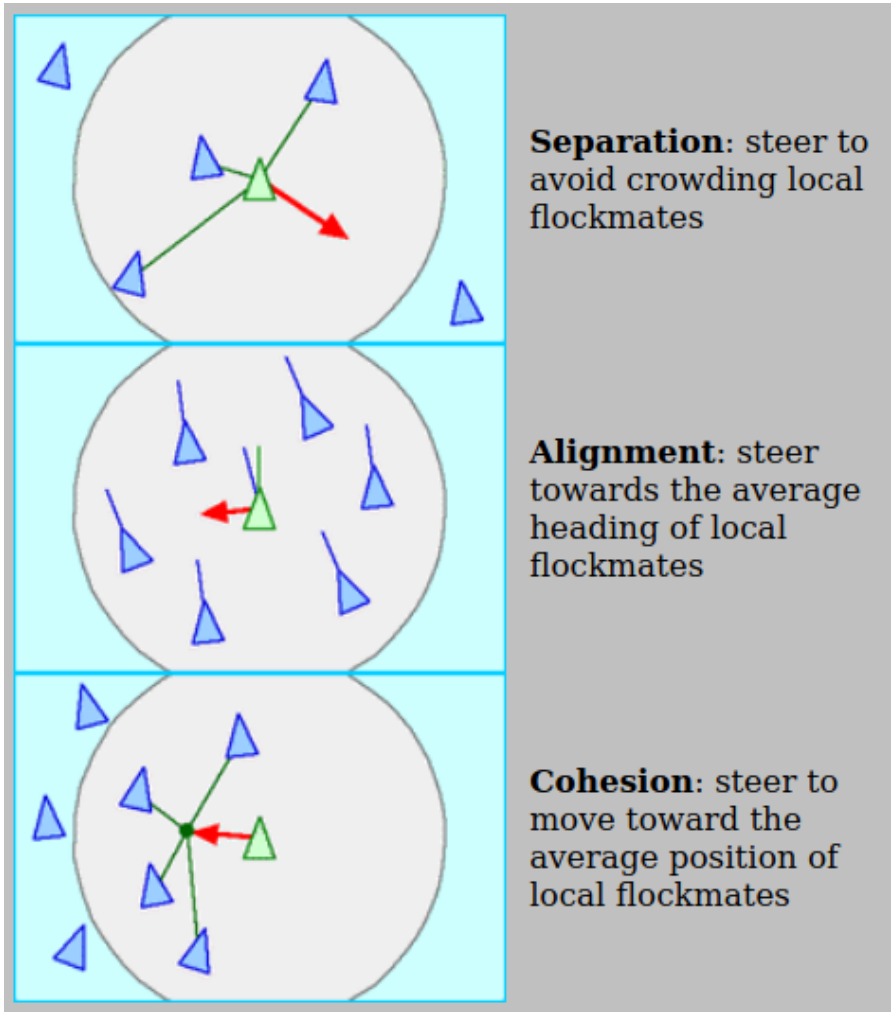
Reynolds' paper (SIGGRAPH 1987)

Boids video: https://www.youtube.com/watch?v=QbUPfMXXQIY

Use in movies: https://www.youtube.com/watch?v=042YjQ9aZNk

Boids rules

A **boid** is a “bird” (a point moving in 2D or 3D space) obeying three rules



Reynolds

Boids properties

- **Emergence**: “complex global behavior can arise from the interaction of simple local rules” (Reynolds)
- Some **ordered** behaviour
- Also some **chaotic** behaviour: small changes in initial conditions lead to large changes later
- A system **at the edge of chaos** (Langton).

Particle Swarm Optimisation

- PSO is “a population based stochastic optimization technique”
- “inspired by social behavior of bird flocking or fish schooling.”
- (from http://www.swarmintelligence.org/tutorials.php)
- Proposed by Eberhart and Kennedy (1995).

PSO and GA

Similarities:

- Population of randomly-initialised points
- Many iterations
- Each generation formed by variation of the previous

Differences:

- Think of each point as **moving**, not being replaced by a new one
- Points have **velocity**.

PSO algorithm

- We store the best point ever observed, called `gbest`

  All particles move towards `gbest`. This is "social influence". Think of a seagull observing **another** seagull finding food.

- Each particle stores the best point it has visited, called `pbest`

  Each particle moves towards `pbest`. This is "personal influence". Think of a seagull **remembering** where it found food.

PSO algorithm

- We store the best point ever observed, called `gbest`

  All particles move towards `gbest`. This is "social influence". Think of a seagull observing **another** seagull finding food.

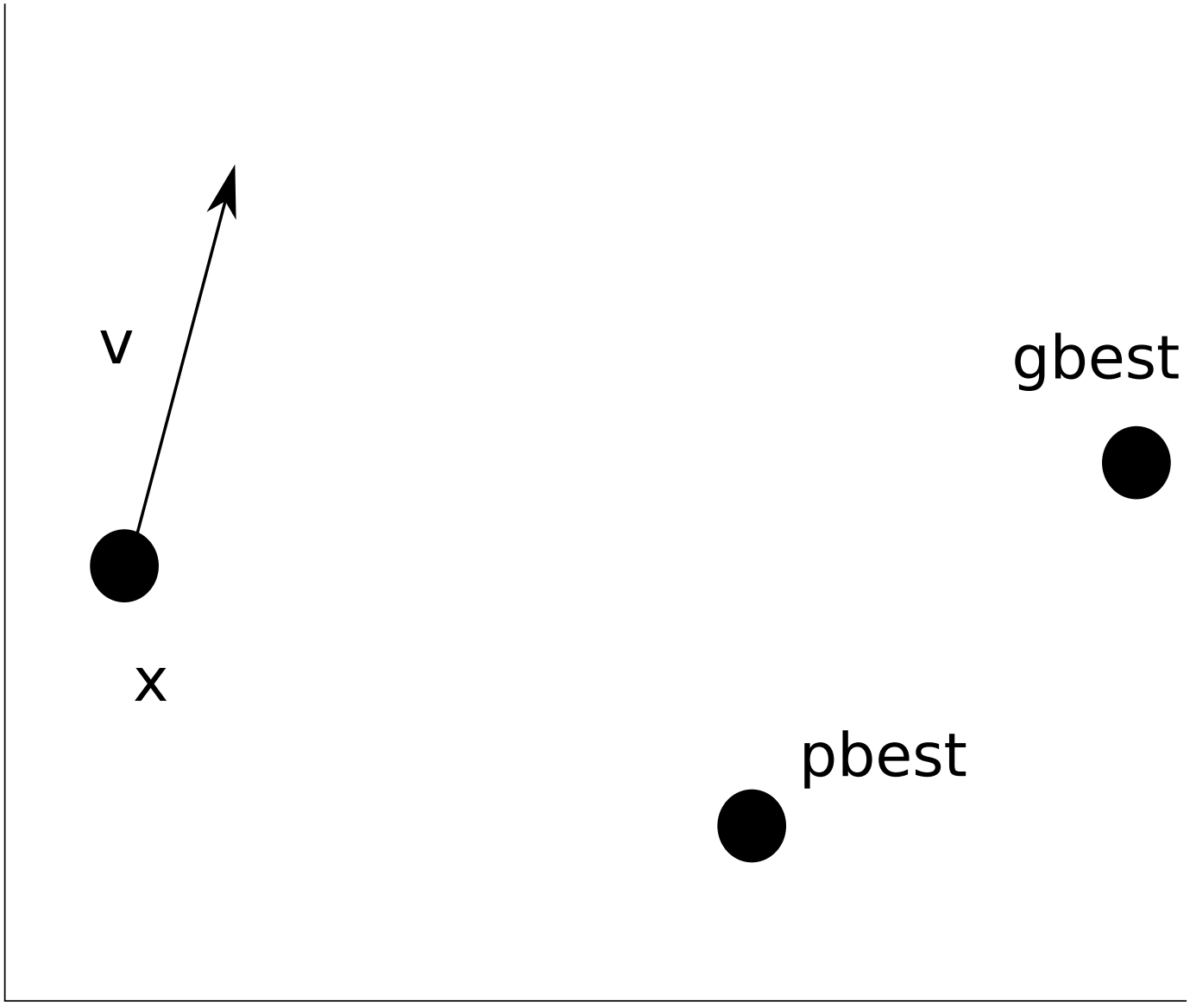
- Each particle stores the best point it has visited, called `pbest`

  Each particle moves towards `pbest`. This is "personal influence". Think of a seagull **remembering** where it found food.

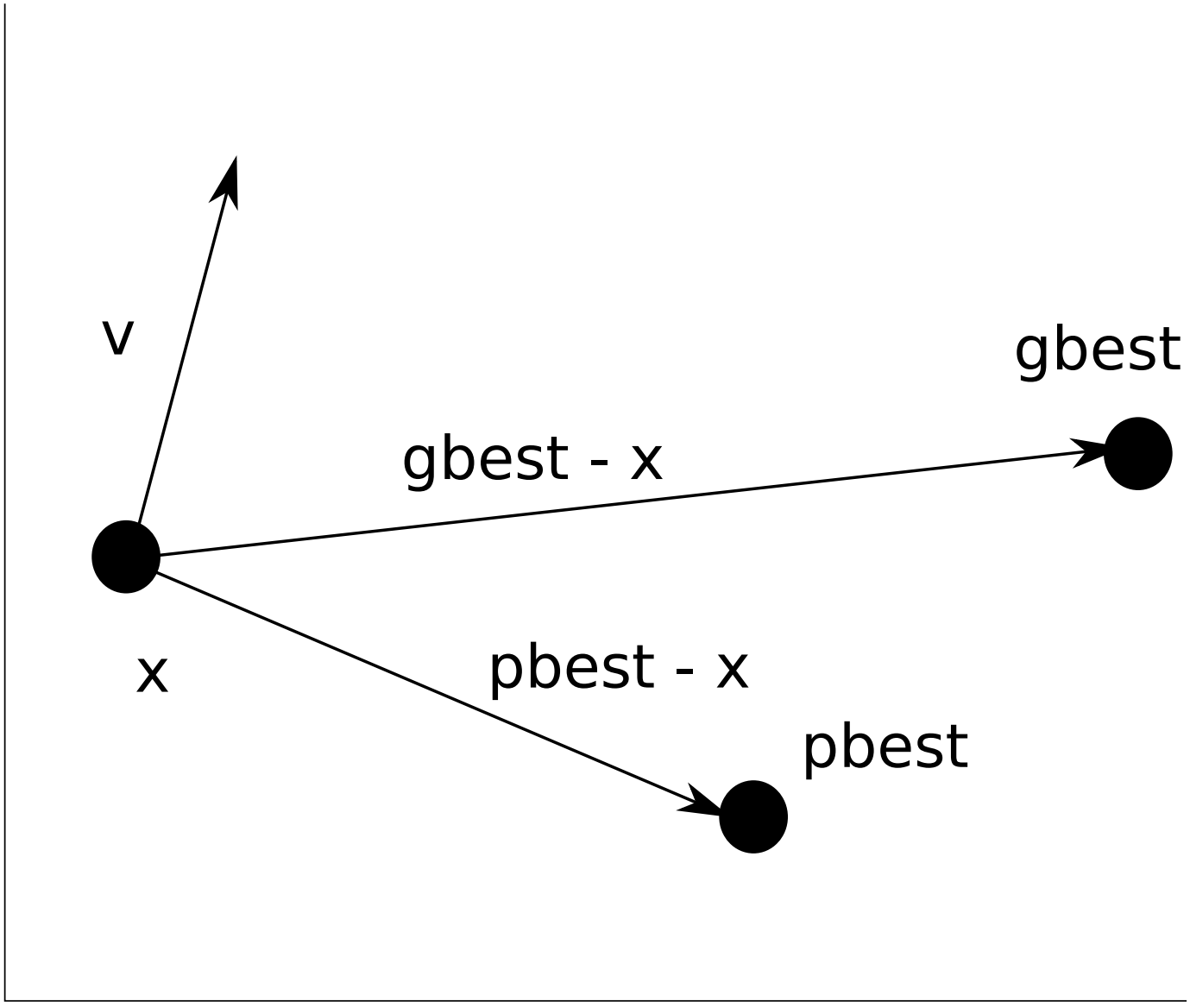
- At each "generation", we update each particle's velocity v and then position x.

1. $v := wv + c_p$ `rand()` (`pbest` $-x$) $+ c_g$ `rand()` (`gbest`$-x$)
2. $x := x + v$

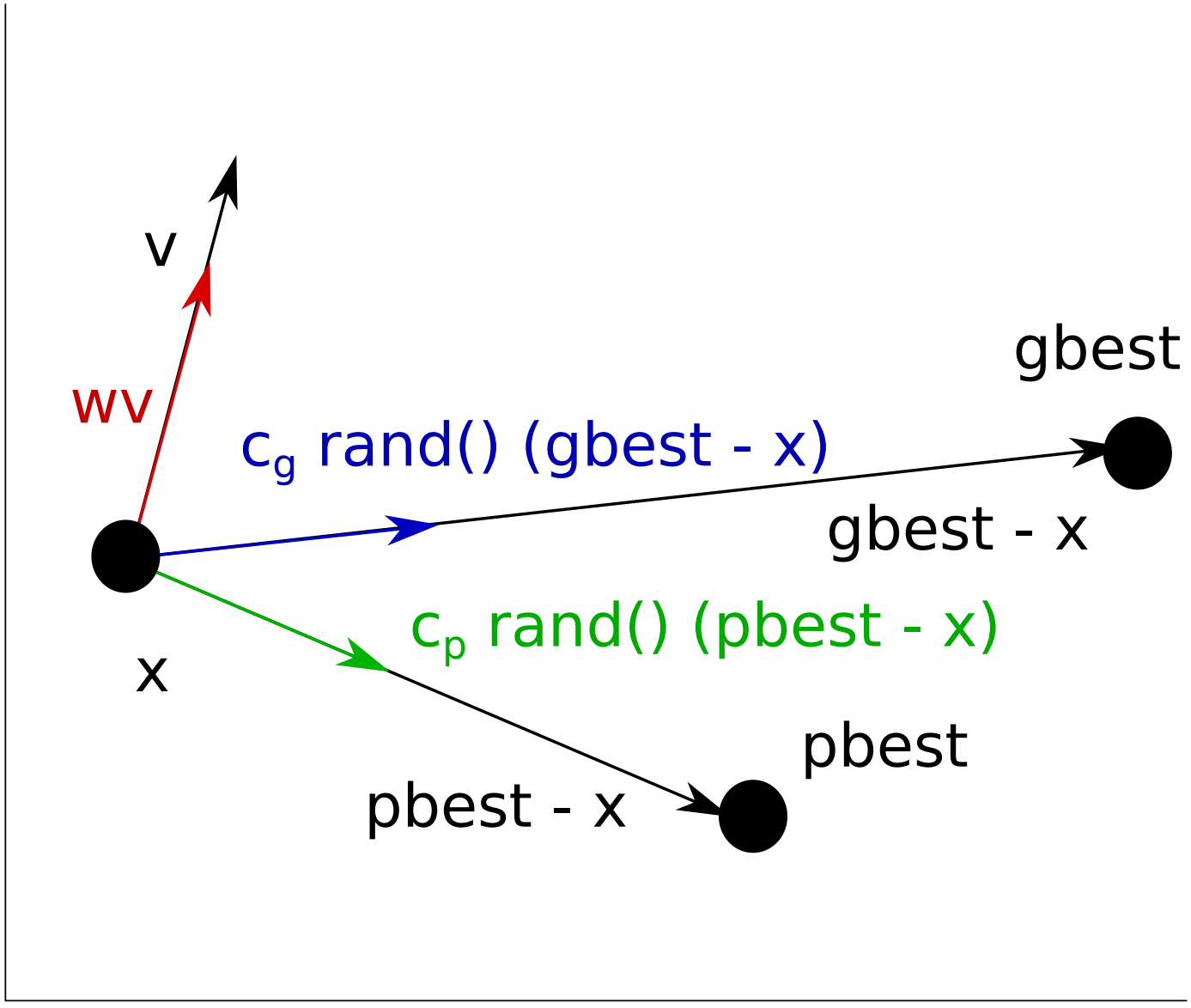
Adding vectors



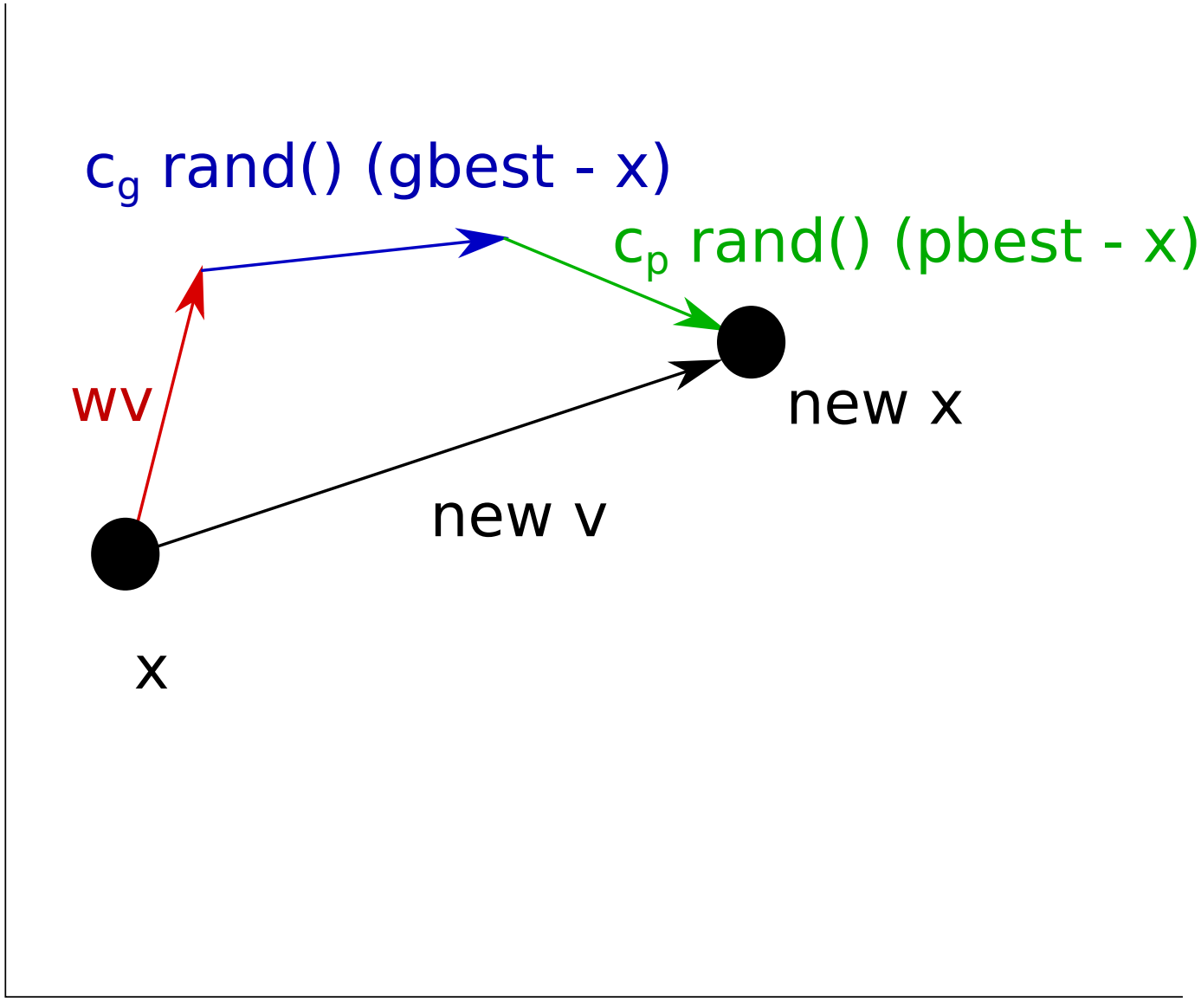
Adding vectors



Adding vectors



Adding vectors



Control parameters

c_p and c_g control the strength of the two rules. Often $c_p = c_g = 2$.

Inertia

- w controls **inertia** (i.e. momentum)
- Large w: velocity more influential: exploration
- Small w: `gbest` and `pbest` more influential: exploitation
- Sometimes w is on an **schedule**, e.g. gradually reduce w from 0.9 to 0.4 over 1000 generations.

Inertia

- w controls **inertia** (i.e. momentum)
- Large w: velocity more influential: exploration
- Small w: `gbest` and `pbest` more influential: exploitation
- Sometimes w is on an **schedule**, e.g. gradually reduce w from 0.9 to 0.4 over 1000 generations.

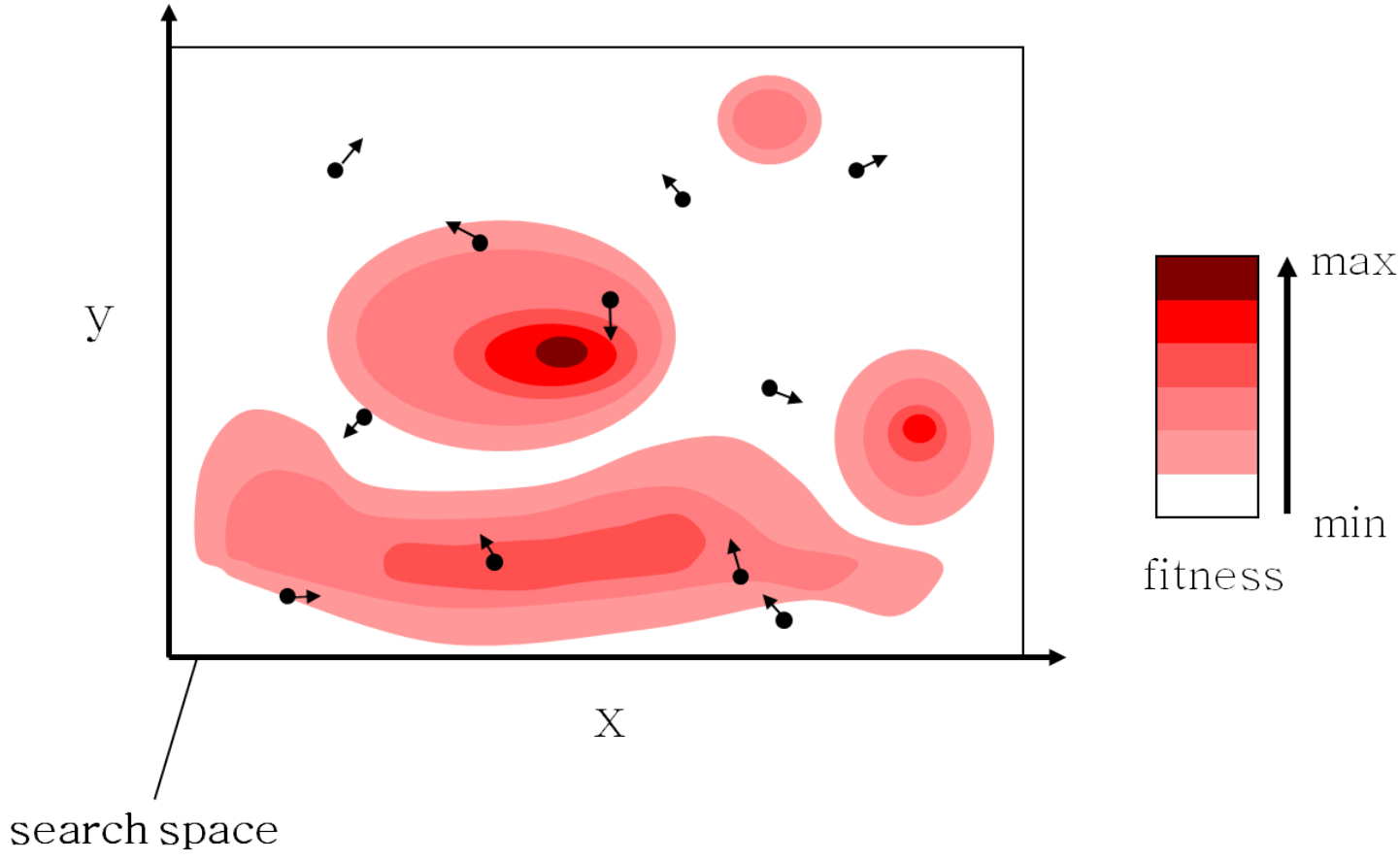
Gradually move from exploration to exploitation over time: just like SA

PSO animation

Wiki:

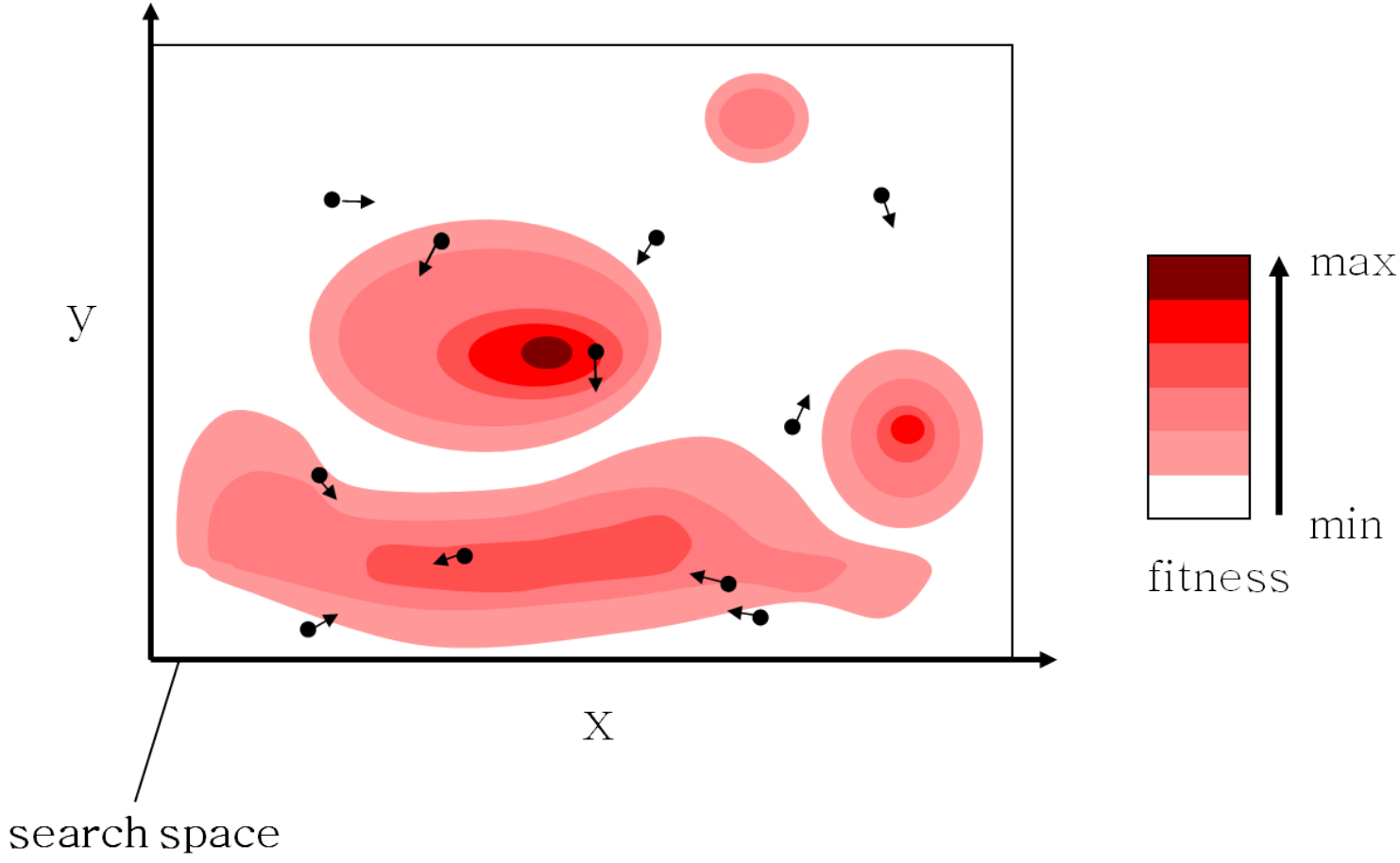
https://commons.wikimedia.org/wiki/File:ParticleSwarmArrowsAnimatio
gif

PSO AT WORK (MAX OPTIMIZATION PROBLEM)

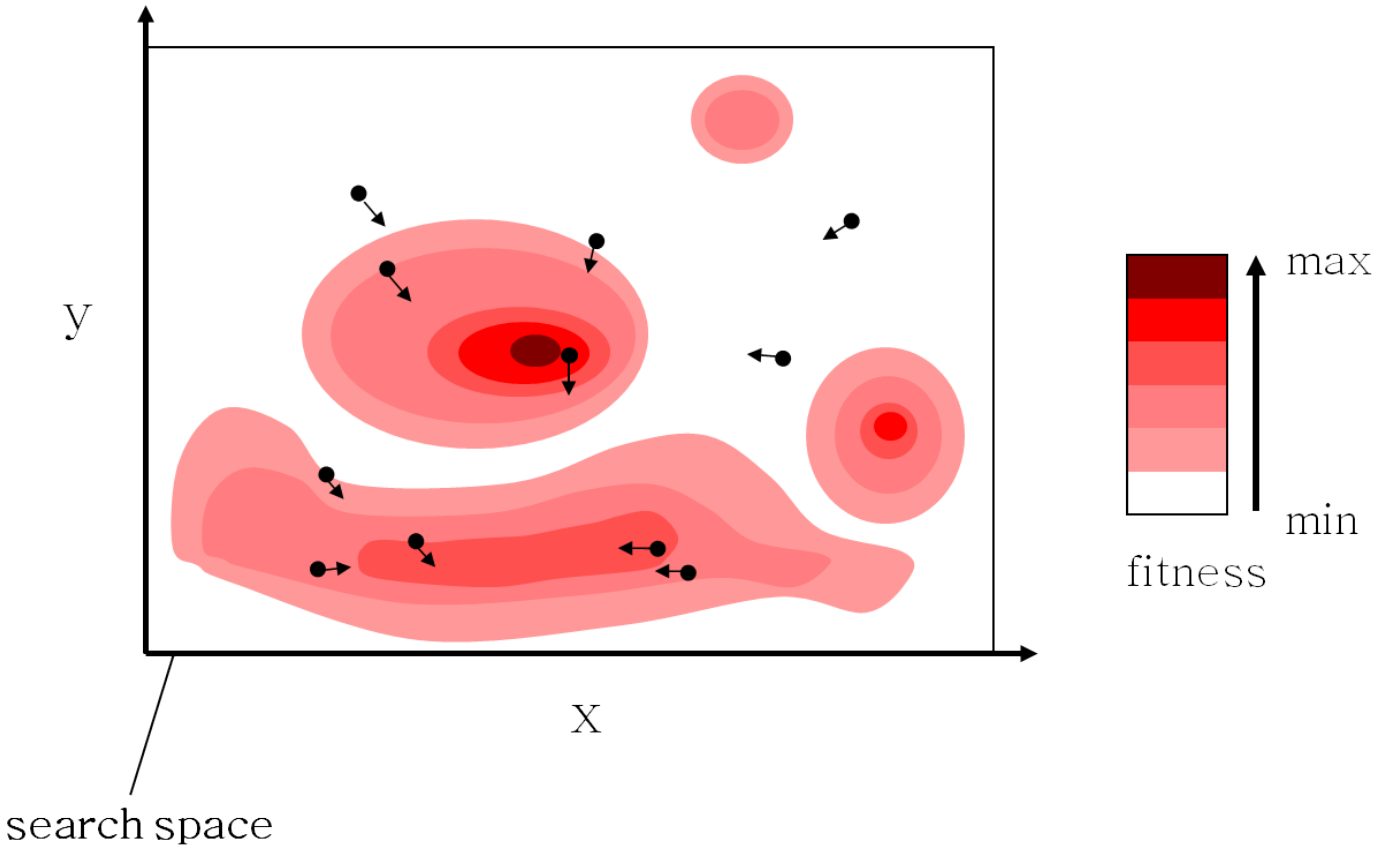


Example slides from Pinto et al.

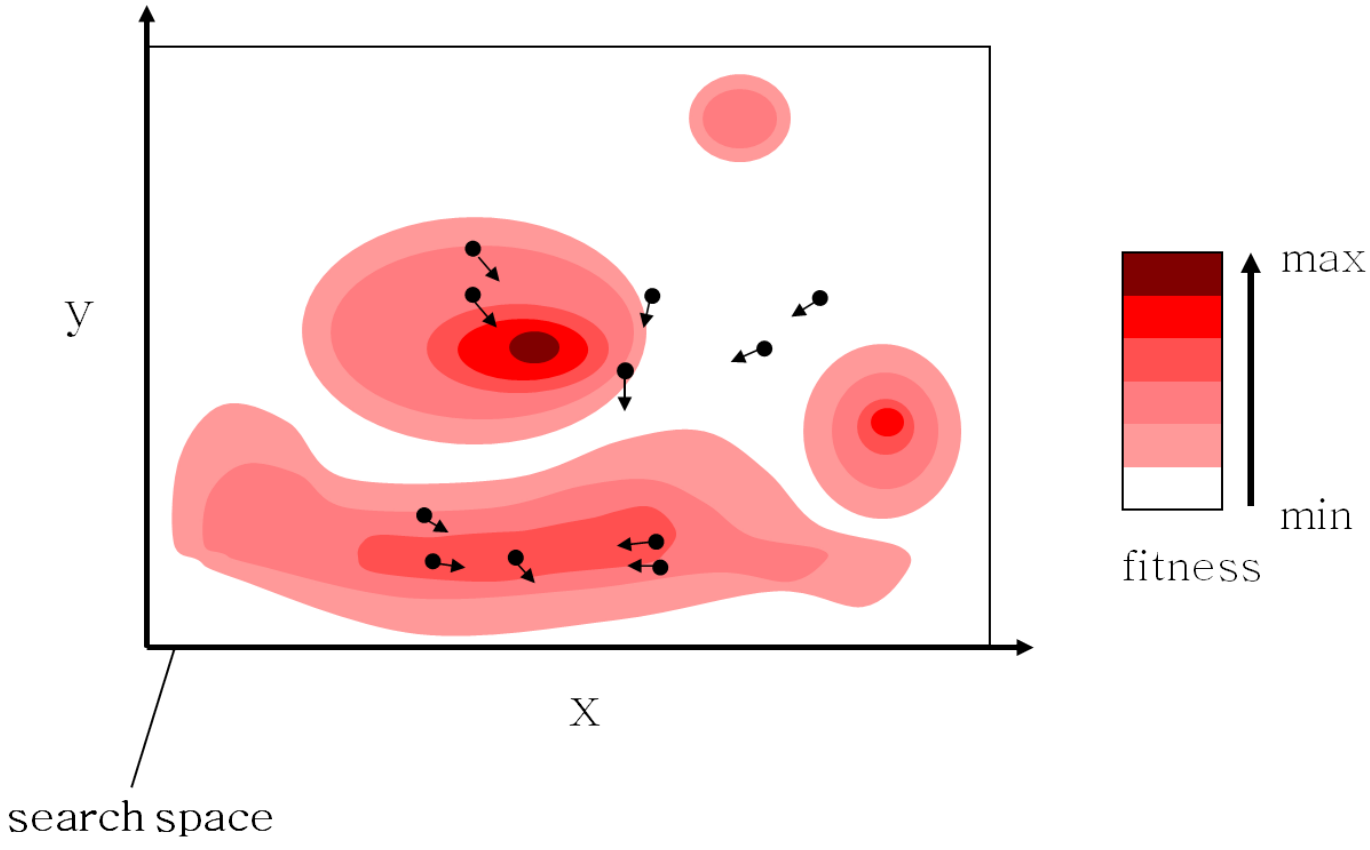
PSO AT WORK



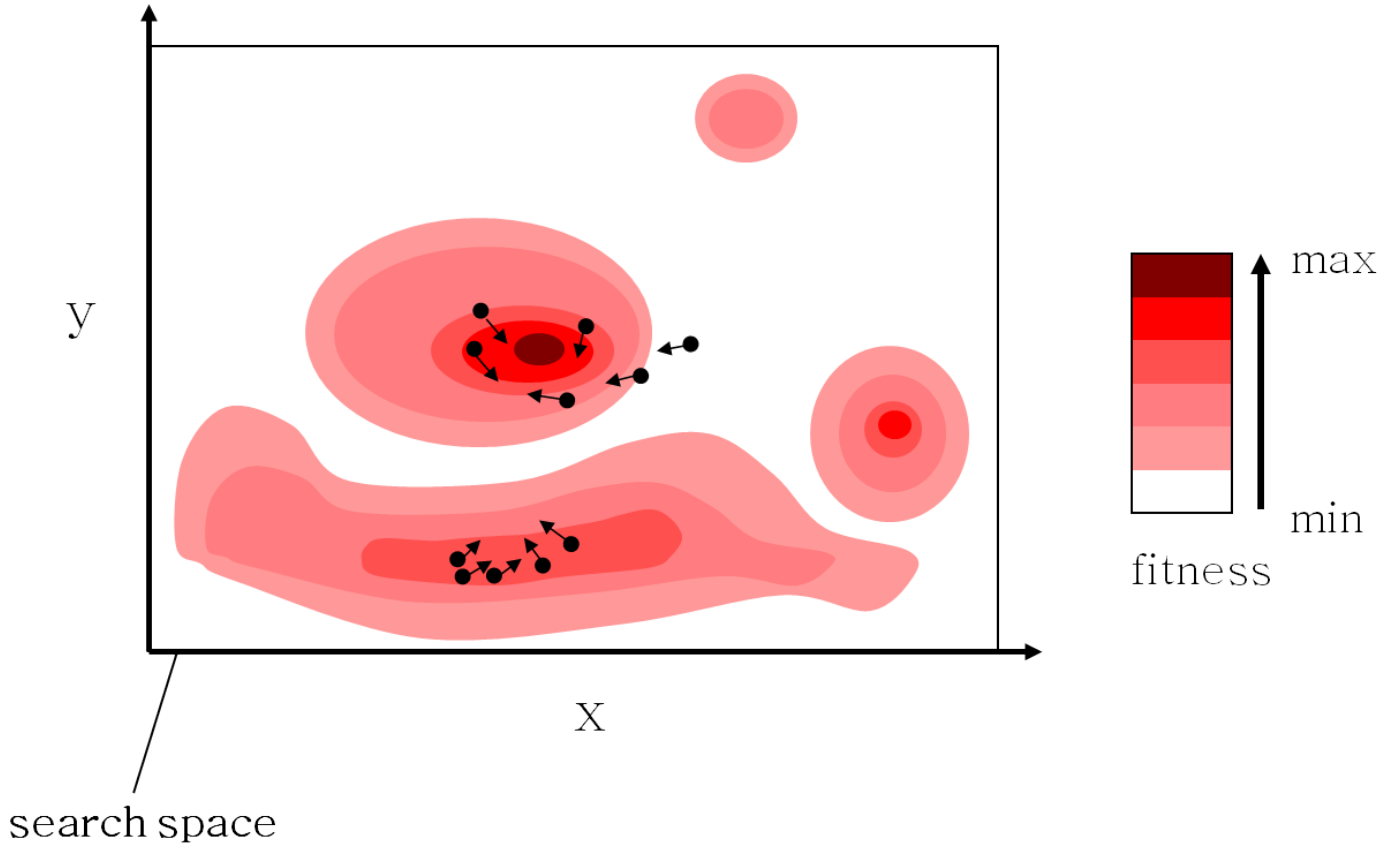
PSO AT WORK



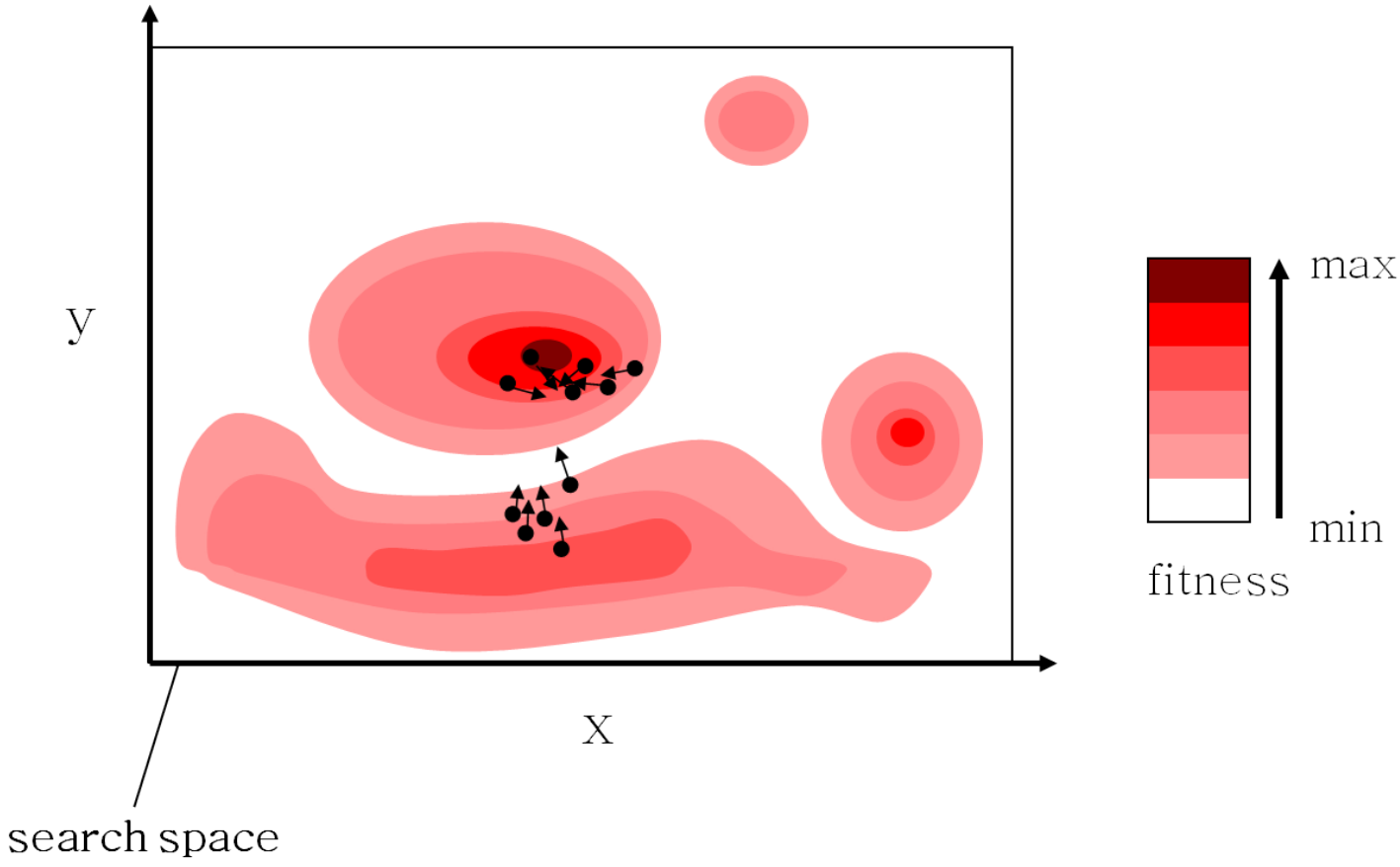
PSO AT WORK



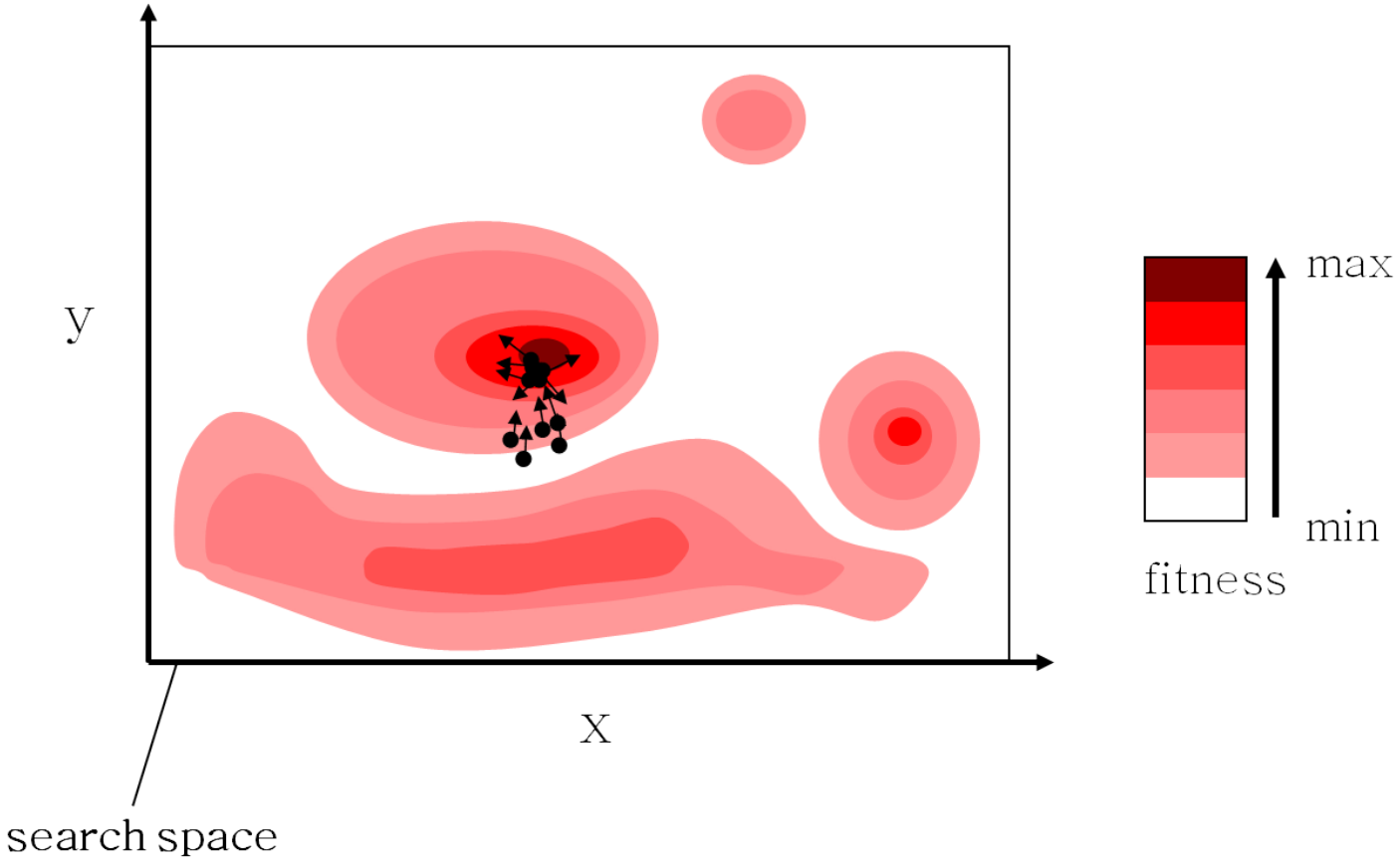
PSO AT WORK



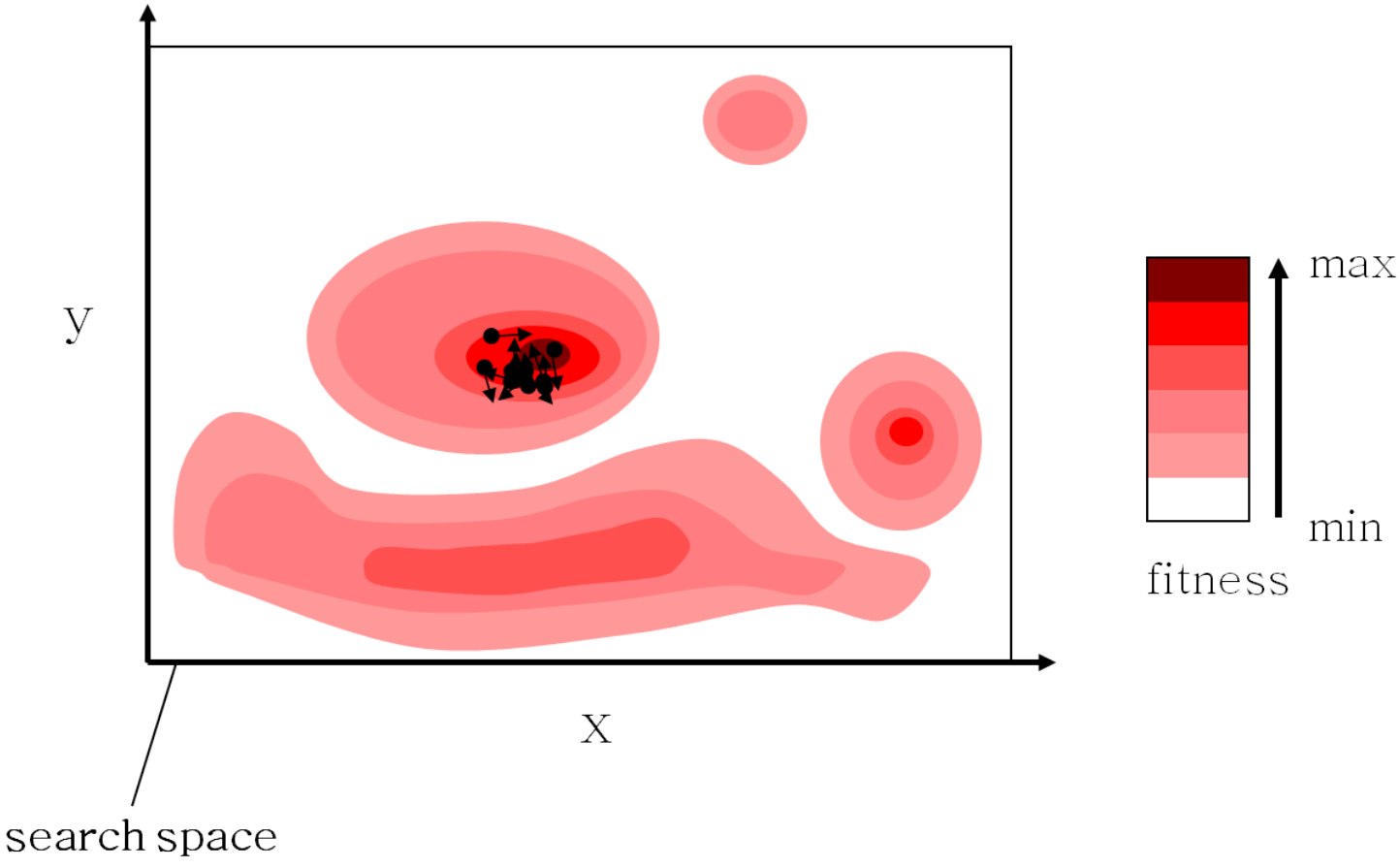
PSO AT WORK



PSO AT WORK



PSO AT WORK



(These images are from Di Caro, credited there to Pinto, I can’t find Pinto!)

Why does it work?

Velocity tends to help because a particle sometimes just “flies past” local optima.

`gbest` means there is **information-sharing** among particles.

Using a PSO in practice

We can program PSO without too much difficulty. But there are some nice libraries, e.g.:

- PySwarms https://github.com/ljvmiranda921/pyswarms
- DEAP https://deap.readthedocs.io/en/master/examples/pso_basic.html

Real-valued optimisation: summary

- Real-valued **GA**
- **CMA** is not really bio-inspired
  - Multivariate normal with full covariance matrix
  - Smart hyperparameters
  - Wins competitions.
- **PSO** is another large branch of bio-inspired computing
  - Inspired by flocking behaviour, not evolution
  - Steer towards personal best and global best
  - But often grouped together with EC.

Overview

1. Real-valued optimisation: big picture and applications
2. Covariance Matrix Adaptation
3. Particle Swarm Optimisation