HERIOT WATT UNIVERSITY

Particle Swarm Optimisation (PSO)

Dr. Michael Lones
Room EM.G31
M.Lones@hw.ac.uk

What is PSO?

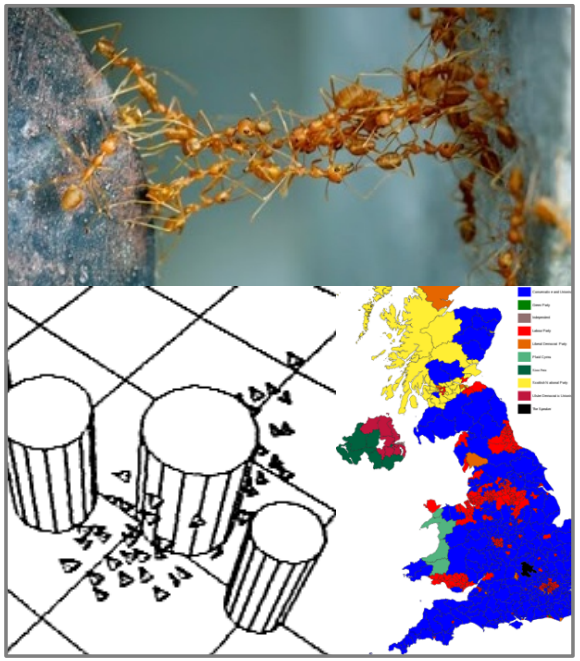
HERIOT WATT UNIVERSITY

- **Particle swarm optimisation** is another well-known optimisation algorithm
  - Invented by Kennedy (a social scientist) and Eberhart (an engineer) in 1995
  - It has been widely applied to real world problems
  - It has inspired lots of other optimisation algorithms
  - It has a lot in common with evolutionary algorithms

What is PSO?

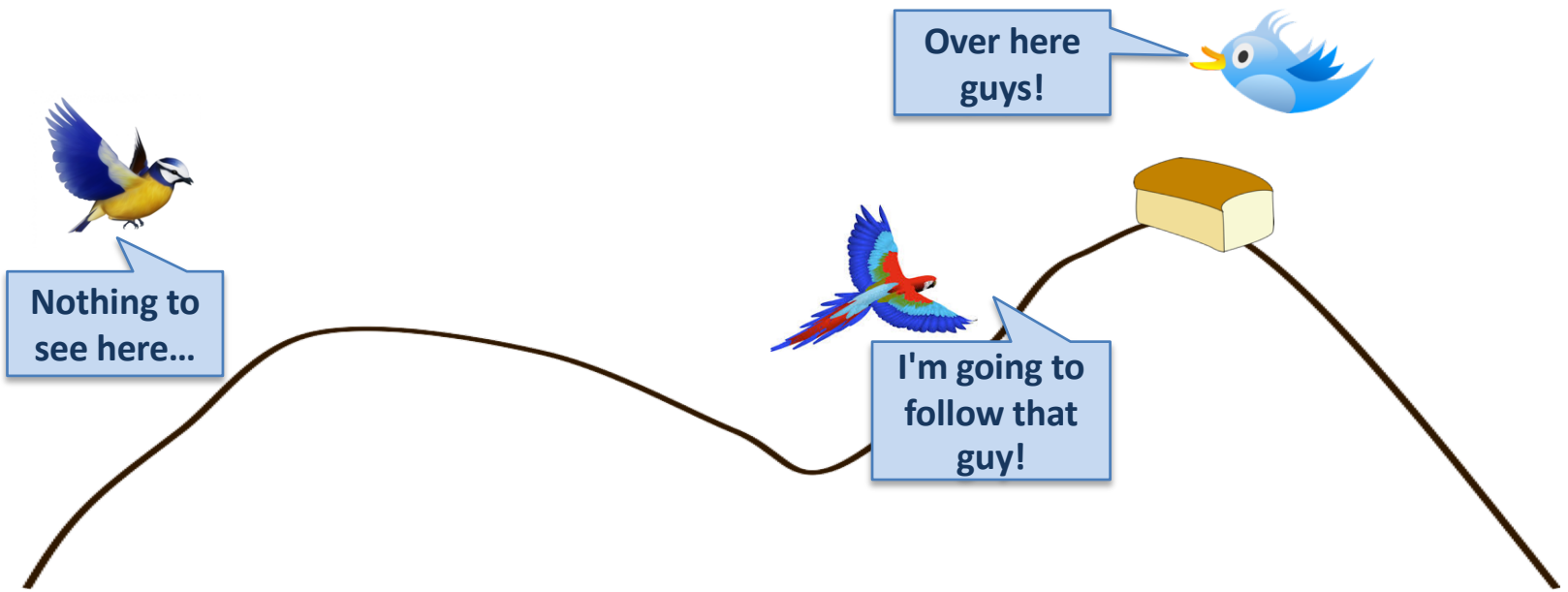
HERIOT WATT UNIVERSITY

- Particle swarm optimisation was inspired by previous work on swarm intelligence:
  - How animal flocking and swarming behaviour is used in **foraging** and exploration,
  - the simple rules used by **boids** to simulate swarms,
  - and the **adaptive culture model** of information sharing
  - In a nutshell: "Can we use swarms to explore fitness landscapes? Yes, we can!"



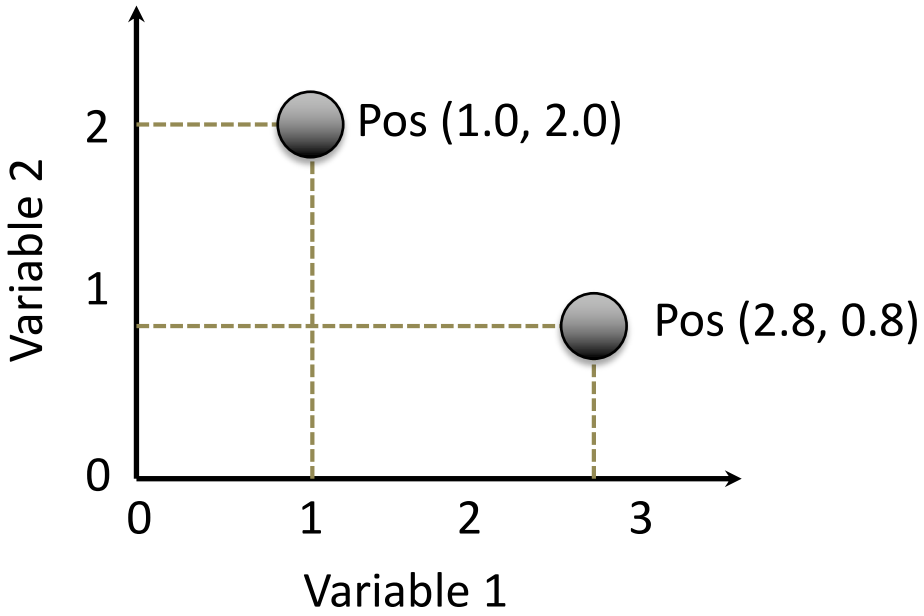
The General Idea

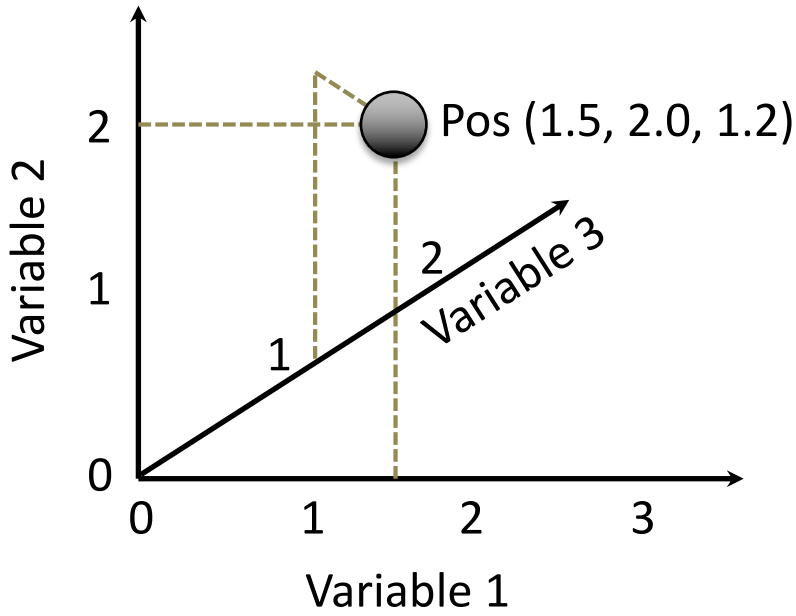
- A group of **particles** fly around a fitness landscape
  - In much the same manner as a flock of birds
  - They are looking for a global optimum
  - They are influenced by one another's experiences



Population

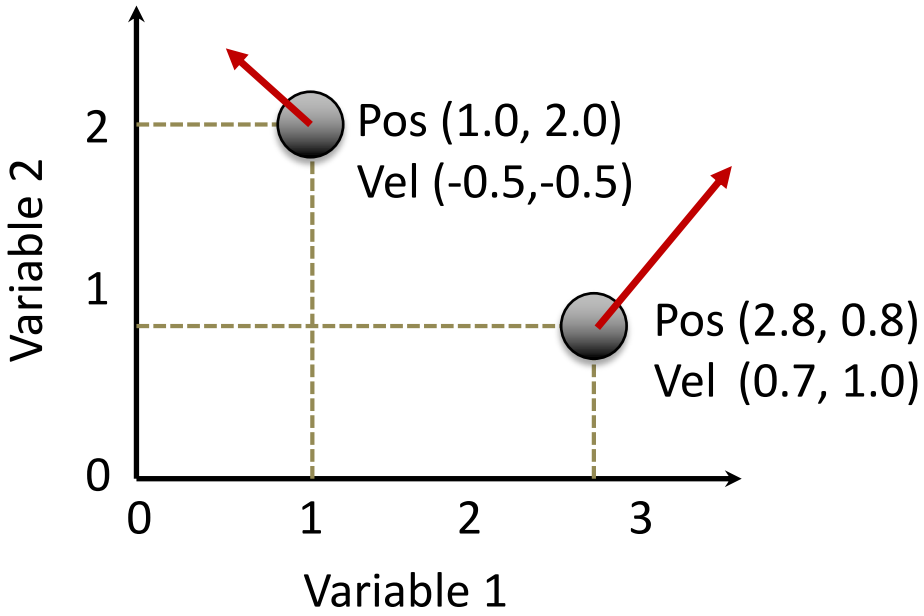
- PSO maintains a **population**, or swarm, of particles
  - Each particle has a position in a search space
  - This is expressed as a vector (i.e. a list), with a value for each dimension (i.e. each variable) of the search space
  - In this respect, PSO is similar to an EA

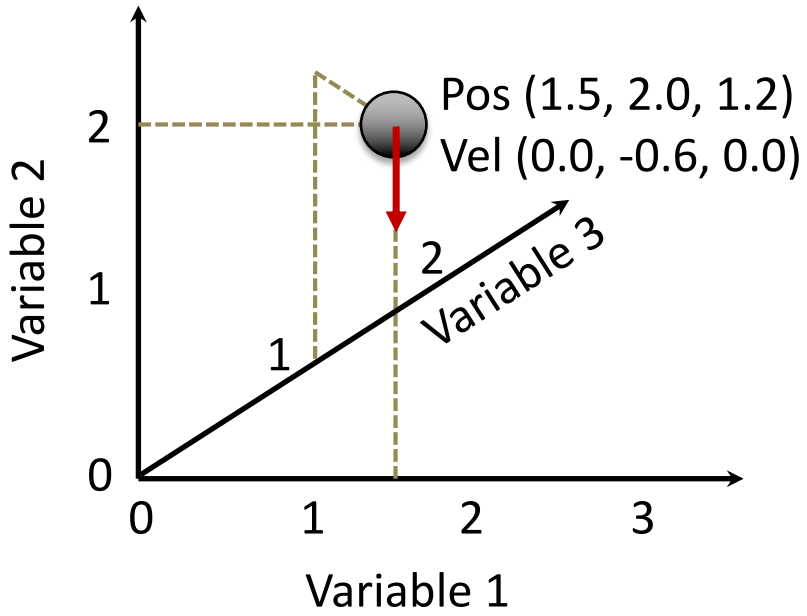




Velocity

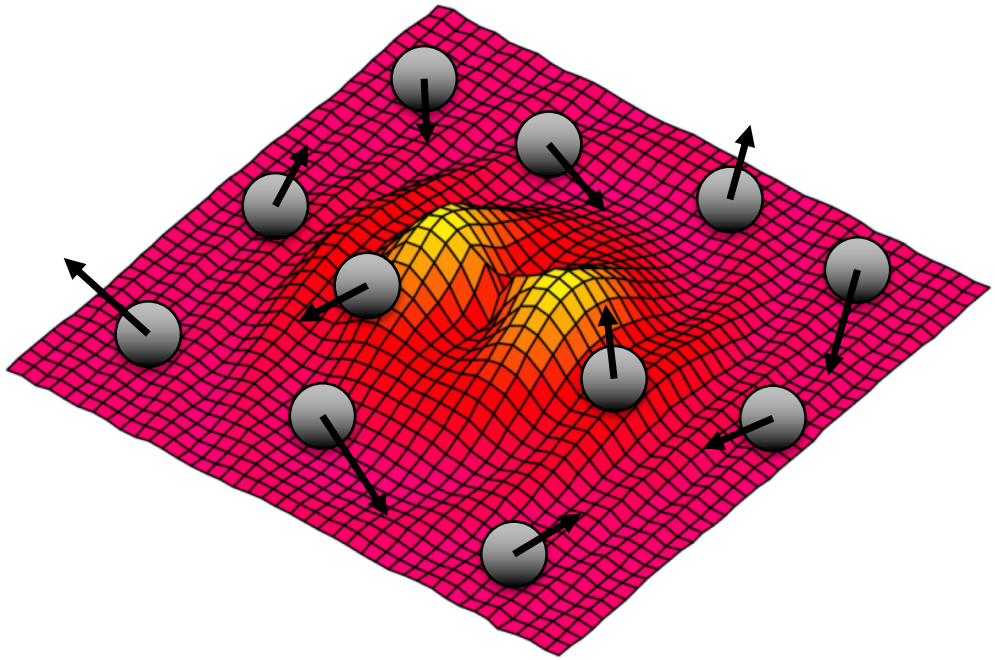
- Each particle also has a **velocity**
  - Specifying how fast it is moving through the search space
  - Also a vector, with a component for each dimension
  - In this respect, PSO is quite different to an EA, where search points do not have an explicit direction of search





Initialisation

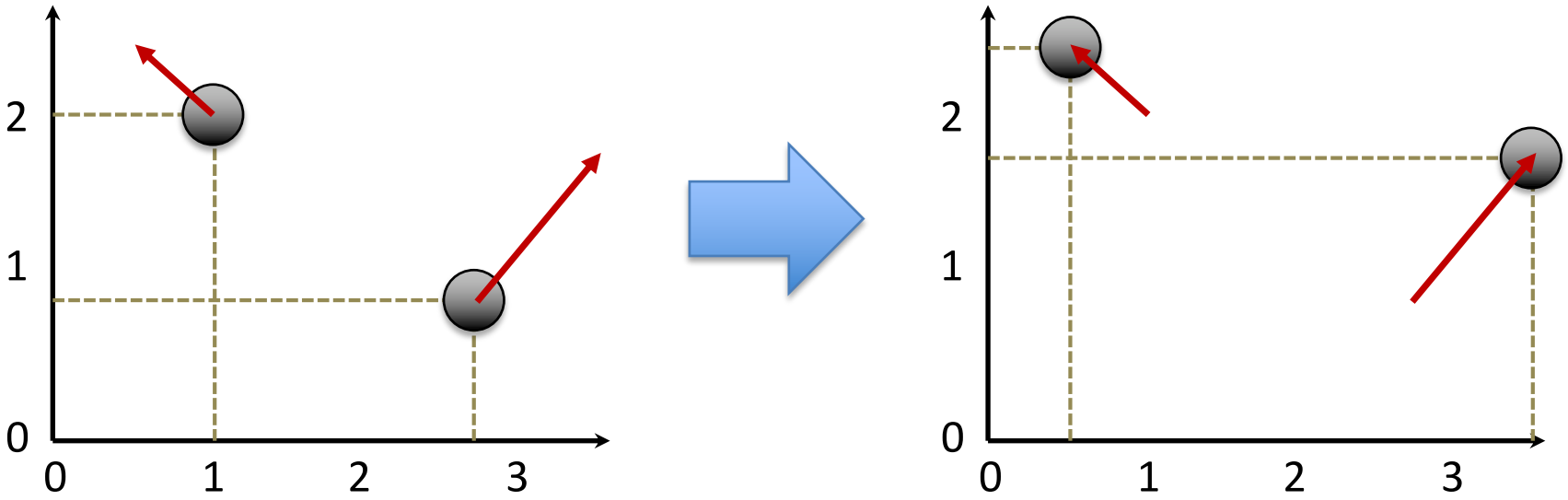
- The initial population is created randomly; after this, particles are neither created nor destroyed
  - Both position and velocity are initially random
  - This is another difference to an EA, where search points are created and destroyed by selection and variation



Initially, the search points will be scattered uniformly across the fitness landscape

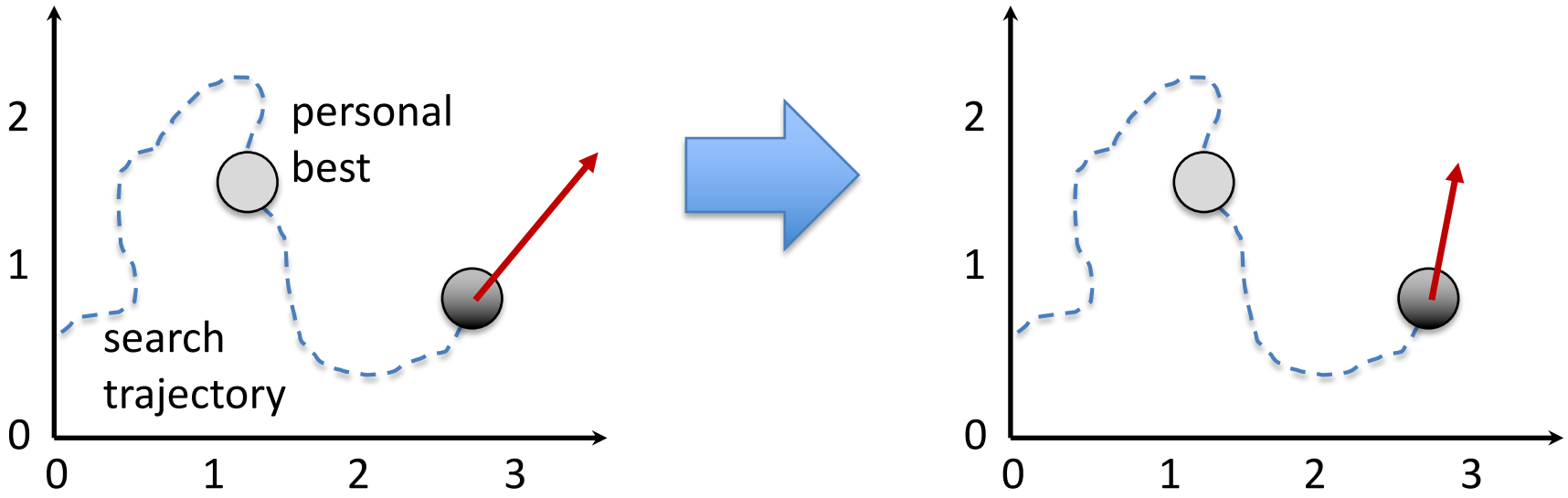
Moving Particles

- At each iteration of the algorithm, each particle's position is updated according to its velocity
  - The velocity is added to its current position
  - This is known as a particle's **inertia**, since it tends to keep it moving in the same general direction



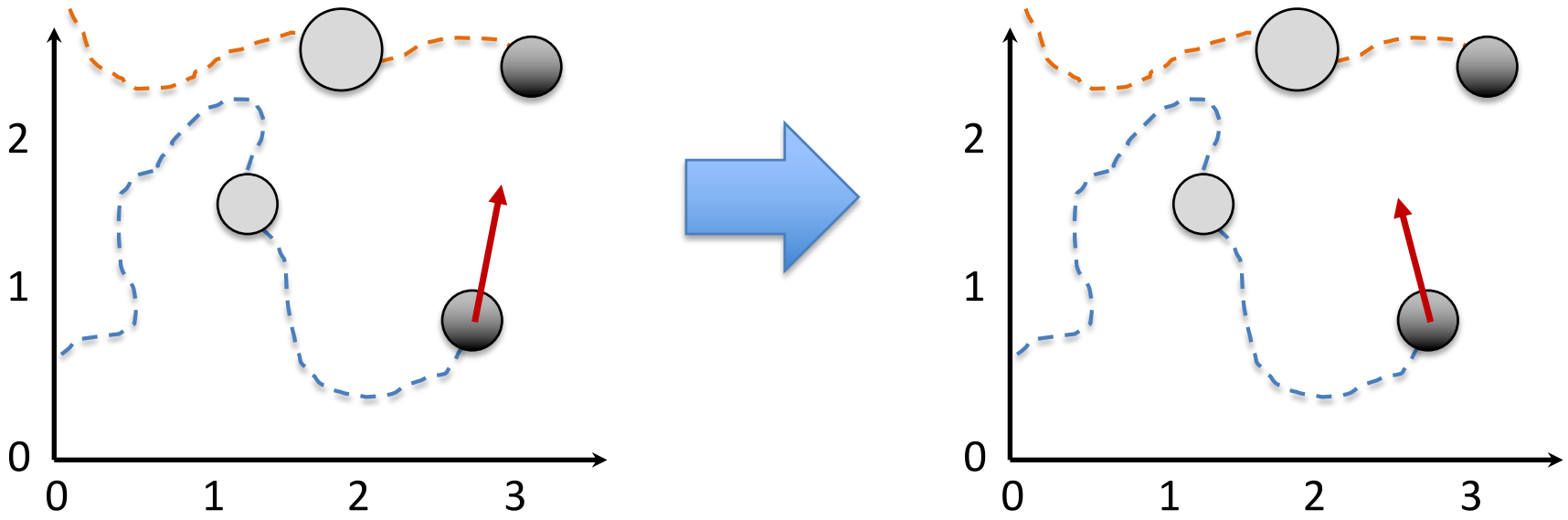
Changing Velocities

- Also at each iteration of the algorithm, each particle's velocity is tweaked in two ways:
  - ***First***, it is tweaked back towards its previous best search point, i.e. the fittest point the particle has seen so far
  - This is known as the **cognitive component** of search, using its own memory to guide its search process



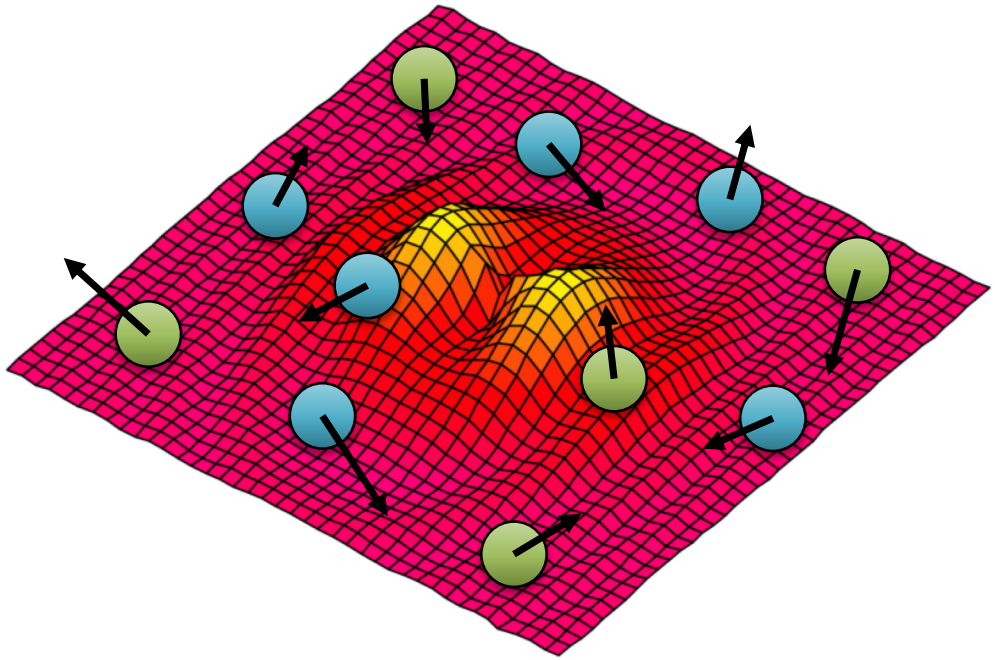
Changing Velocities

- Also at each iteration of the algorithm, each particle's velocity is tweaked in two ways:
  - ***Second***, it is tweaked towards **good search points** which have been seen by other particles
  - This is known as the **social component** of search, using other particles' memories to guide its search process



Changing Velocities

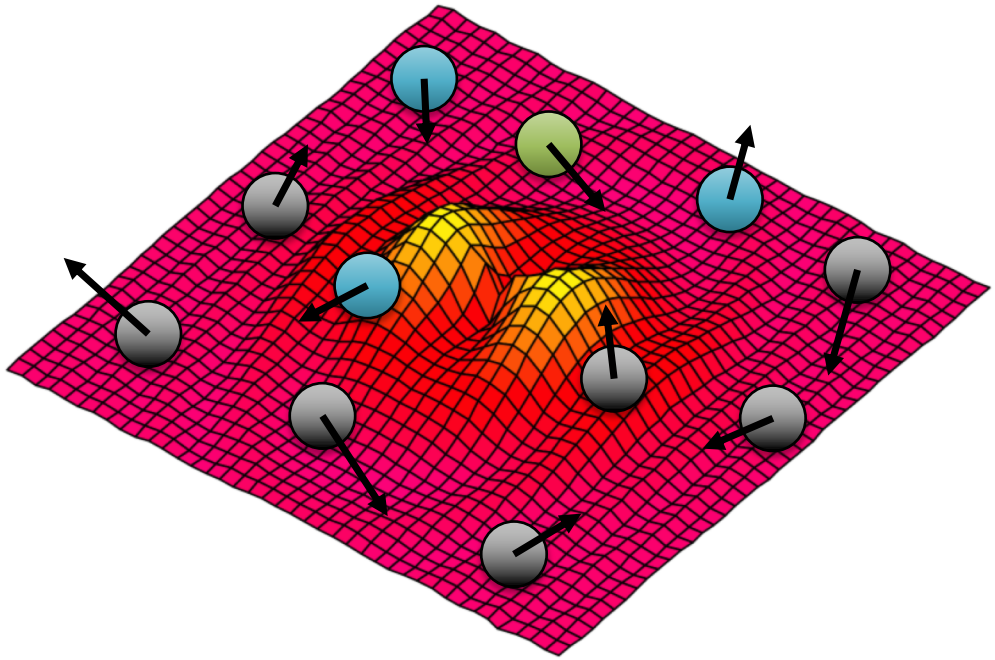
- These other particles are known as **informants**
  - In the simplest case, all particles are informants, so this is the best search point seen so far by the population
  - In many PSO implementations, a group of particles are randomly selected to be informants of each other



Here, there are two groups of mutually-informing particles. These behave as two separate swarms.

Changing Velocities

- Sometimes, a local **neighbourhood** is used:
  - This breaks the swarm up into separate sub-swarms, with information spreading gradually between them
  - The idea being that this will prevent the population from rapidly converging to a local optimum



Here, the green particle is only influenced by the three closest (blue) particles.

This is a similar to interactions in the boids swarm model and the adaptive culture model.

Changing Velocities

- All this is expressed by an equation:

$$v_i \leftarrow \alpha v_i + b(x_i^* - x_i) + c(x_i^+ - x_i)$$

  - v_i is the velocity of a particle
  - α is the inertia weight
  - b and c are weights for the cognitive and social components, respectively
  - x_i is the particle's current position
  - x^*_i is the particle's best position so far
  - x^+_i is its informants' best position so far

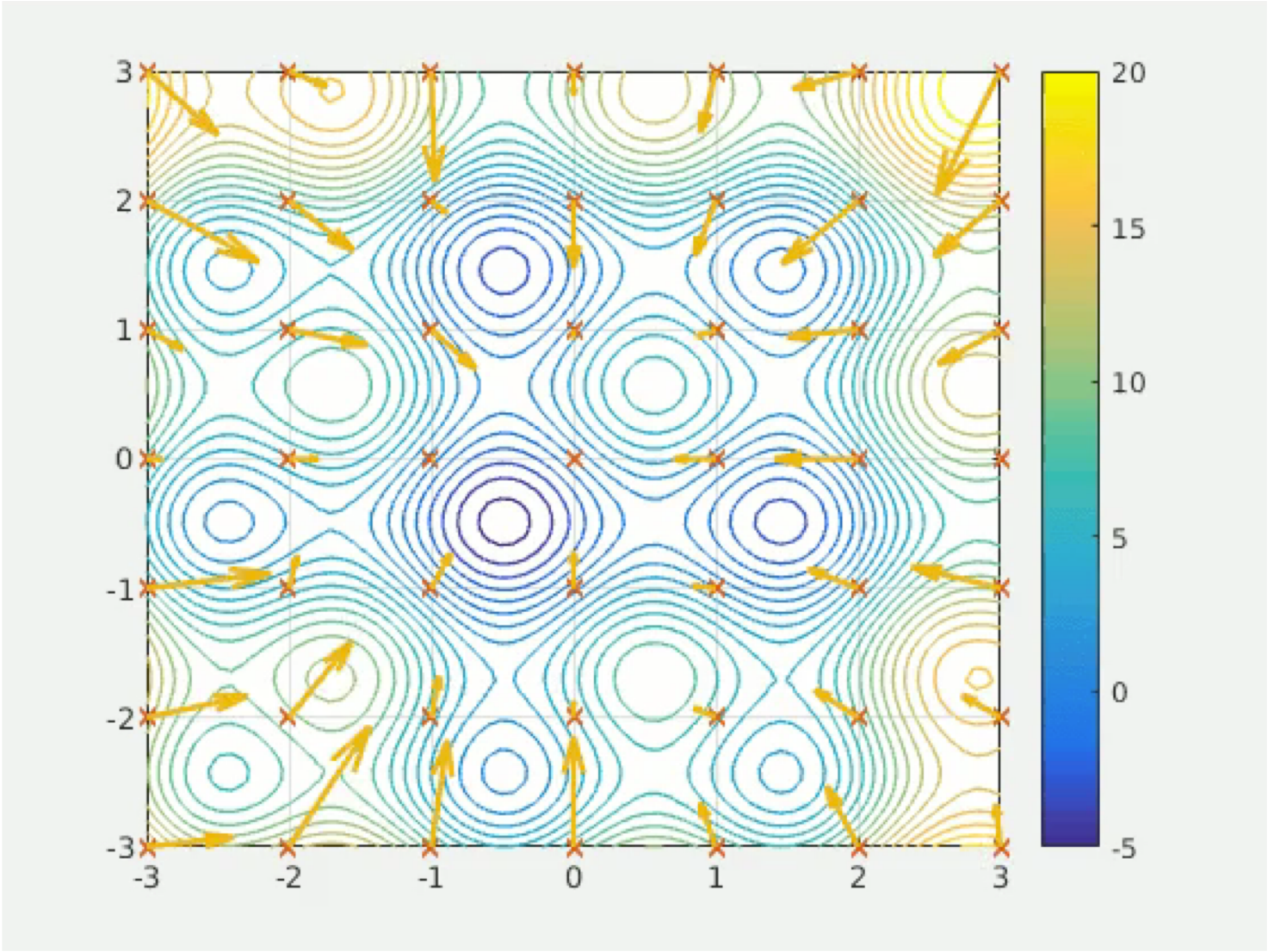
Parameters

- The amount the velocity is weighted towards the cognitive and social components is important
  - Weighting too much or too little towards the cognitive component can stop a particle's search from progressing
  - Weighting too much towards the social component can cause the population's progress to stagnate
  - Weighting too little towards the social component means communication between search processes breaks down
  - So, in short, these weights have to be chosen carefully!

A Random Element

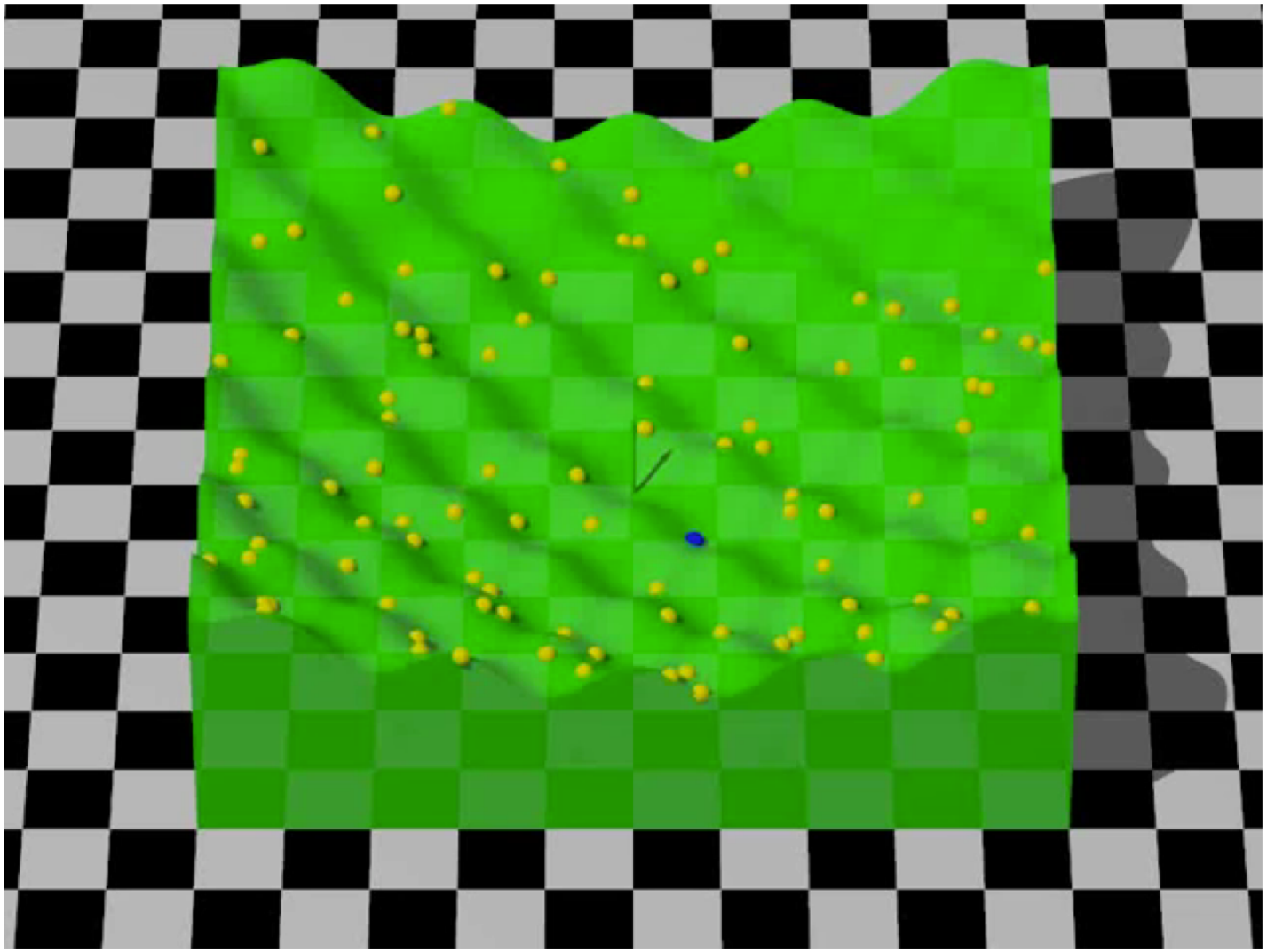
- PSO is stochastic: the weights are in part random
  - Each time the velocity is updated, random numbers are used to determine the contribution of each component
  - A random number weights the contribution of each dimension of each velocity vector
  - Given the same initial population, this means that search will proceed slightly differently each time it is run

Visual Example 1



https://en.wikipedia.org/wiki/File:ParticleSwarmArrowsAnimation.gif

Visual Example 2



https://www.youtube.com/watch?v=3LdgzUIwvXU

Implementations

Pseudocode

- This is for a fairly standard implementation
  - Note: velocity update uses global best and informants

Algorithm 39 *Particle Swarm Optimization (PSO)*

Essentials of Metaheuristics, p57

1: $swarmsize \leftarrow$ desired swarm size
2: $\alpha \leftarrow$ proportion of velocity to be retained
3: $\beta \leftarrow$ proportion of personal best to be retained
4: $\gamma \leftarrow$ proportion of the informants' best to be retained
5: $\delta \leftarrow$ proportion of global best to be retained
6: $\epsilon \leftarrow$ jump size of a particle

These are all of PSO's parameters

7: $P \leftarrow \{\}$
8: **for** $swarmsize$ times **do**
9: $\quad P \leftarrow P \cup \{\text{new random particle } \vec{x} \text{ with a random initial velocity } \vec{v}\}$
10: $\overrightarrow{Best} \leftarrow \square$

Initialise state

Pseudocode

11: **repeat**
12: **for** each particle $\vec{x} \in P$ with velocity $\vec{v}$ **do**
13: AssessFitness($\vec{x}$)
14: **if** $\overrightarrow{Best} = \square$ or Fitness($\vec{x}$) > Fitness($\overrightarrow{Best}$) **then**
15: $\overrightarrow{Best} \leftarrow \vec{x}$
16: **for** each particle $\vec{x} \in P$ with velocity $\vec{v}$ **do**
17: $\vec{x}^* \leftarrow$ previous fittest location of $\vec{x}$
18: $\vec{x}^+ \leftarrow$ previous fittest location of informants of $\vec{x}$
19: $\vec{x}^! \leftarrow$ previous fittest location any particle
20: **for** each dimension i **do**
21: $b \leftarrow$ random number from 0.0 to β inclusive
22: $c \leftarrow$ random number from 0.0 to γ inclusive
23: $d \leftarrow$ random number from 0.0 to δ inclusive
24: $v_i \leftarrow \alpha v_i + b(x_i^* - x_i) + c(x_i^+ - x_i) + d(x_i^! - x_i)$
25: **for** each particle $\vec{x} \in P$ with velocity $\vec{v}$ **do**
26: $\vec{x} \leftarrow \vec{x} + \epsilon\vec{v}$
27: **until** $\overrightarrow{Best}$ is the ideal solution or we have run out of time
28: **return** $\overrightarrow{Best}$

Update global best?

Gather information

Update particle velocity

Update positions

Parameter Values

- This version of PSO has 6 parameters
  - Swarm size, the weights α–δ, and the step size ε
  - (and also the number and choice of informants)
  - The optimal values for these depends upon the problem being solved, and the form of the fitness landscape
  - A **rule of thumb** is that the weights β–δ sum to 4; this discourages the swarm from exploding or imploding
  - A swarm size of 10–100 is typical
  - A step size of 1 is typical

PSO Variants

- There are **a lot** of variants of PSO. These are some of the ways in which they vary:
  - **Adaptive swarm sizes**: this works like selection in an EA – unfit particles die off and fit particles are replicated
  - **Adaptive weights**: higher fitness particles pay less attention to the search experiences of other particles
  - **Multi-swarm**: used to search multimodal landscapes or to avoid premature convergence in deceptive landscapes
  - **Learning**: incorporating algorithmic elements that learn about the landscape and inform search
  - **Simplification**: people value algorithms that are easy to understand and implement, e.g. bare-bones PSO

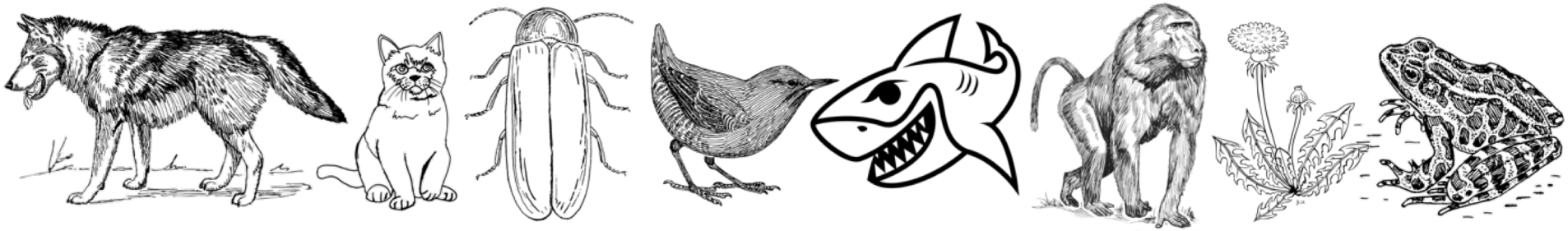
Standard PSO

- Given the large number of possible design choices, there have been attempts to standardise PSO
  - The imaginatively named "Standard PSO", see: http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf
  - Each version (2011 is the latest) is based around the best theoretical knowledge available at the time
  - Worth a read, if only to see all the considerations that need to be made when choosing an optimal PSO!

More Bio-Inspiration?

More Bio-Inspiration

- Another way of extending PSO is to mimic more ways in which biological swarms work
  - By looking at how particular organisms form swarms, and the kind of problems these swarms solve
  - The motivation is that biology has already solved the problem of swarms that do useful things
  - However, biologists do not always understand these swarms, and there's the question of how well biological foraging behaviours translate to fitness landscapes



More Bio-Inspiration

- This has led to an *interesting* phenomenon:
  - Inexperienced researcher reads a dummies guide about a particular biological organism
    This can lead to quite shallow knowledge!

More Bio-Inspiration

- This has led to an *interesting* phenomenon:
  - Inexperienced researcher reads a dummies guide about a particular biological organism
    This can lead to quite shallow knowledge!
  - Then invents a new algorithm based on their understanding of this organism's behaviour
    Which is often the same as an existing algorithm

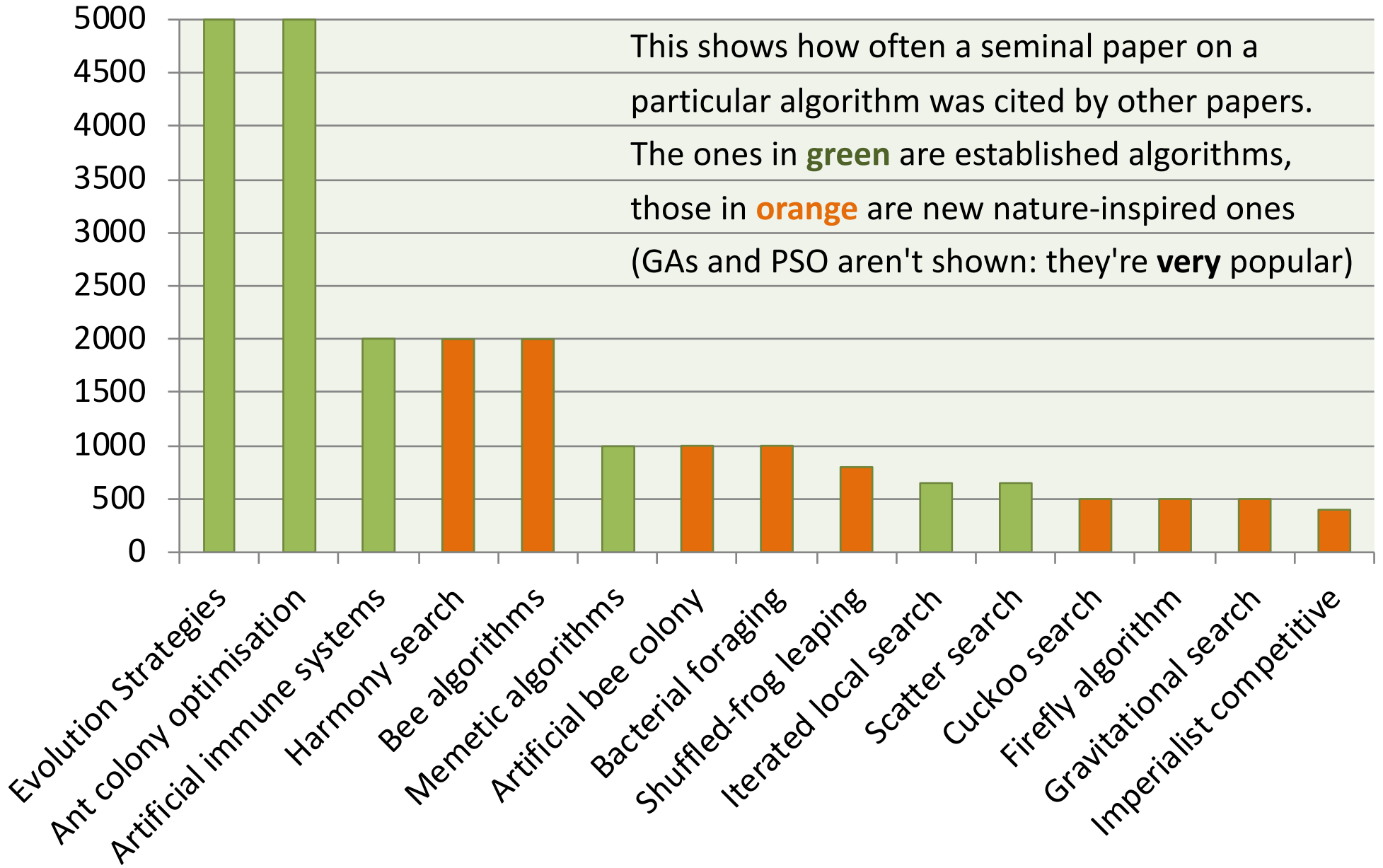
More Bio-Inspiration

- This has led to an *interesting* phenomenon:
  - Inexperienced researcher reads a dummies guide about a particular biological organism
    This can lead to quite shallow knowledge!
  - Then invents a new algorithm based on their understanding of this organism's behaviour
    Which is often the same as an existing algorithm
  - Then writes a paper saying that their algorithm is better than all other algorithms
    Their knowledge of statistics is not great

More Bio-Inspiration

- This has led to an *interesting* phenomenon:
  - Inexperienced researcher reads a dummies guide about a particular biological organism
    This can lead to quite shallow knowledge!
  - Then invents a new algorithm based on their understanding of this organism's behaviour
    Which is often the same as an existing algorithm
  - Then writes a paper saying that their algorithm is better than all other algorithms
    Their knowledge of statistics is not great
  - Fame and fortune!
    Since people who use algorithms don't know any better

Google Scholar Citations



A Typical Example

Cat Swarm Optimization

Shu-Chuan Chu[1], Pei-wei Tsai[2], and Jeng-Shyang Pan[2]

[1] Department of Information Management,
Cheng Shiu University
[2] Department of Electronic Engineering,
National Kaohsiung University of Applied Sciences

Abstract. In this paper, we present a new algorithm of swarm intelligence, namely, Cat Swarm Optimization (CSO). CSO is generated by observing the behaviors of cats, and composed of two sub-models, i.e., tracing mode and seeking mode, which model upon the behaviors of cats. Experimental results using six test functions demonstrate that CSO has much better performance than Particle Swarm Optimization (PSO).

A Community Response

HERIOT WATT UNIVERSITY

Metaheuristics—the metaphor exposed

Kenneth Sörensen

University of Antwerp, Operations Research Group ANT/OR, Prinsstraat 13 – B5xx, 2000 Antwerp, Belgium
E-mail: kenneth.sorensen@ua.ac.be [Sörensen]

Received 2 November 2012; received in revised form 12 November 2012; accepted 13 November 2012

Abstract

In recent years, the field of combinatorial optimization has witnessed a true tsunami of "novel" metaheuristic methods, most of them based on a metaphor of some natural or man-made process. The behavior of virtually any species of insects, the flow of water, musicians playing together – it seems that no idea is too far-fetched to serve as inspiration to launch yet another metaheuristic. In this paper, we will argue that this line of research is threatening to lead the area of metaheuristics away from scientific rigor. We will examine the historical context that gave rise to the increasing use of metaphors as inspiration and justification for the development of new methods, discuss the reasons for the vulnerability of the metaheuristics field to this line of research, and point out its fallacies. At the same time, truly innovative research of high quality is being performed as well. We conclude the paper by discussing some of the properties of this research and by pointing out some of the most promising research avenues for the field of metaheuristics.

Things you should know

- How PSO works
  - Understand the underlying concepts
  - Understand velocity updates
  - Be able to describe a general PSO algorithm
  - I don't expect you to know about all the different variants of PSO

Further Reading

HERIOT WATT UNIVERSITY

- Kennedy & Eberhart, Particle Swarm Optimization http://ieeexplore.ieee.org/document/488968/
  - The seminal paper on PSO. Fairly short and readable
- http://www.scholarpedia.org/article/Particle_swarm_optimization
  - An brief overview of PSO by some well known researchers in the field
- Jordehi & Jasni, Particle swarm optimisation for discrete optimisation problems, https://link.springer.com/article/10.1007/s10462-012-9373-8
  - A weakness of PSO is that it only works in continous search spaces. Many real world fitness landscapes are discrete. This paper discusses variants for such problems.

Coursework 2

Coursework 2

HERIOT WATT UNIVERSITY

- You choose between two coursework variants
- It is recommended you do Variant 1:
  - Implement a basic genetic algorithm (GA)
  - Implement a basic version of PSO
  - Compare them using some benchmark problems
  - Write a 3 page report which discusses the results and shows some understanding of GAs and PSO

Coursework 2

- However, you can do Variant 2 if you prefer
  - This is designed for non-programmers
  - Do a literature review on comparing optimisers using mathematical benchmark problems
  - Compare GA and PSO using a software tool
  - Write a 5 page report which includes the literature review, the results, and shows some understanding of GAs and PSO

Variant 1

- The main aim is to give you some experience of implementing and using two popular optimisers
  - You get a lot of marks for just implementing them
  - 60% (for F20BC), 50% (for F21BC)
  - You're only required to implement basic versions, but you can add bells and whistles if you want
  - You can use any sensible language, but should use the same language for both implementations

Variant 1

- The second aim is to give you some idea of how different optimisers perform on different problems
  - You will be using the CEC 2005 benchmarks for this
  - These are standardised mathematical functions where the aim is to find the global optimum
  - You should choose 5 of these
  - Code is available in Java, Python, Matlab and C, so you shouldn't need to implement these functions yourself

Variant 2

- In Variant 2, you're not asked to implement algorithms, so:
  - I expect you to spend more time studying the literature
  - In particular, you're asked to summarise previous literature on using benchmarks to compare optimisers
  - You're also expected to spend more time on the comparative study (reflected by a larger CW %age) and the discussion of the results

Comparing Algorithms

- Both variants involve doing experiments to compare GA and PSO
  - It's important that you do this in a fair way, for example using the same population size
  - There's information about this in the coursework spec. **Please make sure you read the spec thoroughly!**
  - You need to do multiple runs, because both algorithms are stochastic

Comparing Algorithms

HERIOT WATT UNIVERSITY

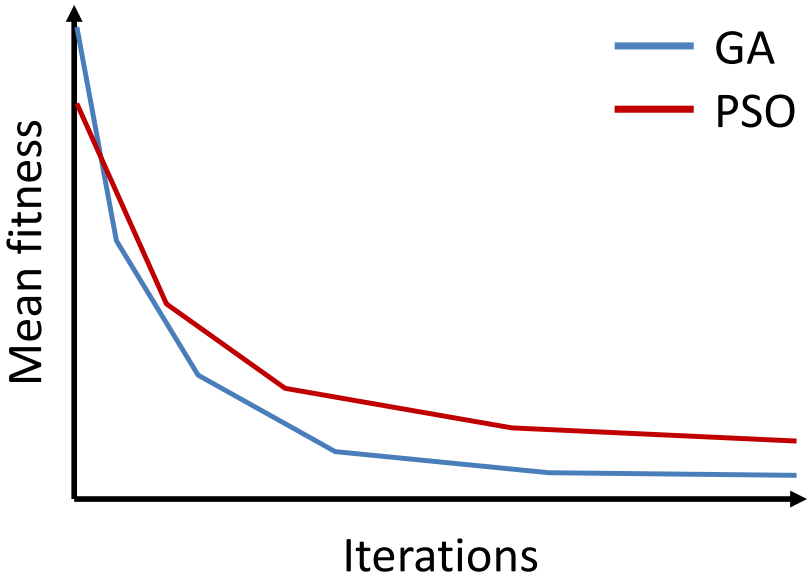
- There are various ways you could present the results of your comparative study:

e.g. mean* fitness of 10 solutions from 10 runs:

Problem	GA	PSO
1	**0**	0.5
2	-50	**-100**
3	**0.04**	0.05
4	10	**9**
5	100	**75**

* You could also add standard deviations or statistical tests

e.g. fitness-iteration plots (average of 10 runs)



Optimising Hyperparameters

- Both variants also require you to gain some insights into the role of hyperparameters
  - e.g. the effects of varying population size, mutation rate, acceleration coefficients etc.
  - But don't spend too long on this. A few examples and a bit of discussion would suffice.

F20BC vs F21BC

- Those of you studying F21BC are also asked to relate your observations to the wider literature:
  - e.g. if you observe a particular effect from varying hyperparameters, or see that GA/PSO perform well on particular problems, try to relate this to what other people have reported in the literature