Course: Natural Computing

# 3. Metaheuristic Optimisation Algorithms for Continuous Domains


J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Course: Natural Computing

# Evolutionary Algorithms (EA)


J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

# Evolutionary Computation (EC)

1. Genetic algorithms: Solution of a problem in the form of strings of numbers using recombination and mutation

   we'll come back to GAs soon

2. Genetic programming: Evolution (of the structure) of computer programs

   GP (next week)

3. Evolution strategies: Vectors of real numbers as representations of solutions

   this is what we are going to look at now

4. Evolutionary programming: For parametrised programs, i.e. mainly parameters evolve, see also differentiable programming

   together with GP and later

Search space: discrete (1, 2) or (in prinicple) continuous (3, 4)
Search space dimension: finite (1, 3) or (in prinicple) unbounded (2, 4)
Alphabet: numeric (1, 3), (numeric and) symbolic (2, 4)

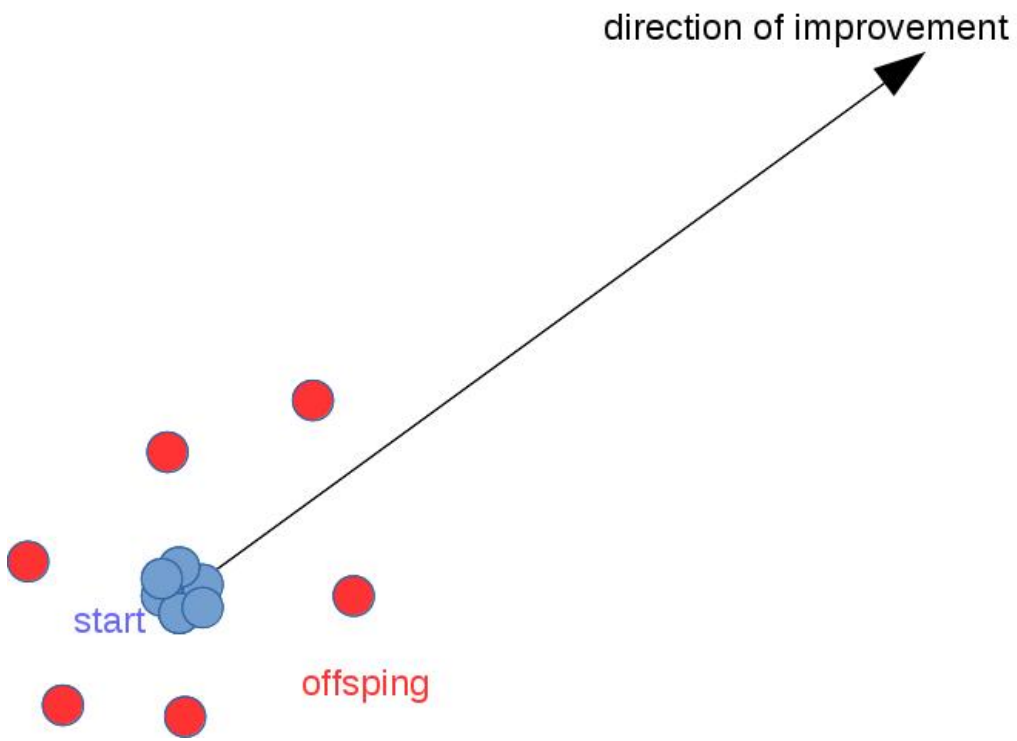
# Evolution Strategies

## Evolution with continuous representations

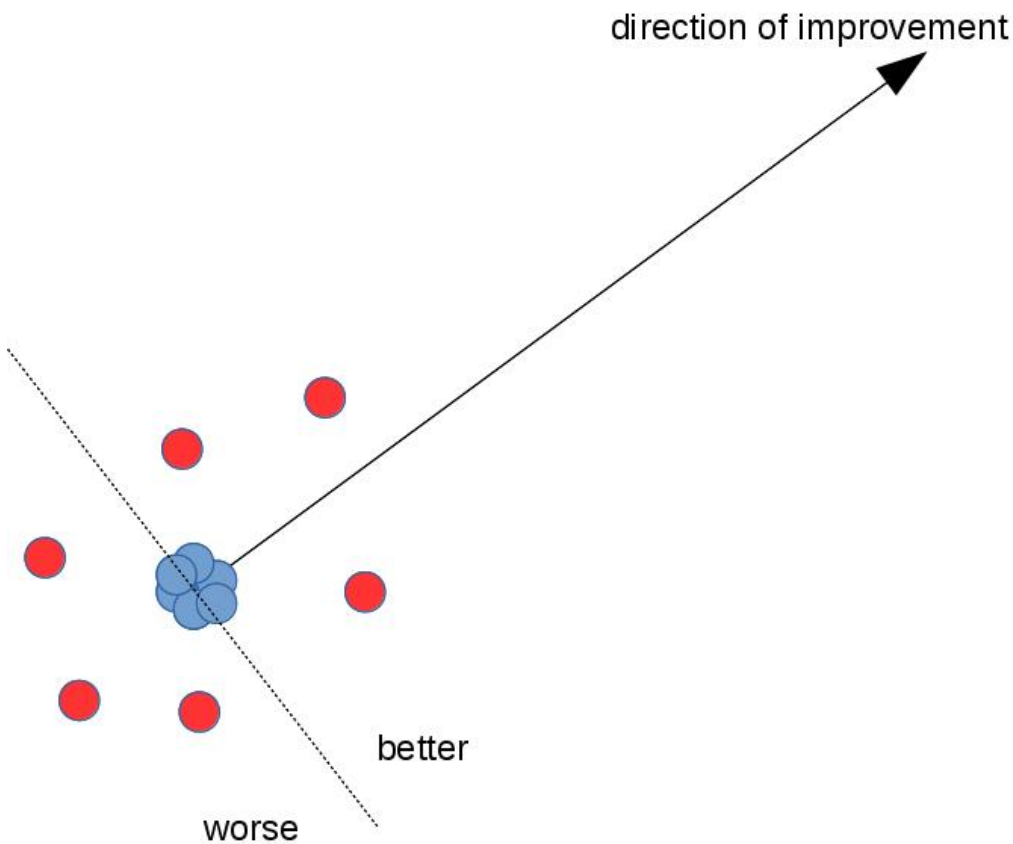
- Problem-dependent continuous representation for search and optimisation (what is a good representation?)
- In contrast to simulated annealing, a population of solutions is used
- Individuals are vectors of real numbers which describe current solutions of the problem
- Selection by fitness from various "parent" sets
- Mutation in continuous steps (similar to simulated annealing) with adaptation of the mutation rate to account for different scales and correlations of the components

1964: Ingo Rechenberg; Hans-Paul Schwefel
web.archive.org/web/20180422161252/http://www.bionik.tu-berlin.de/

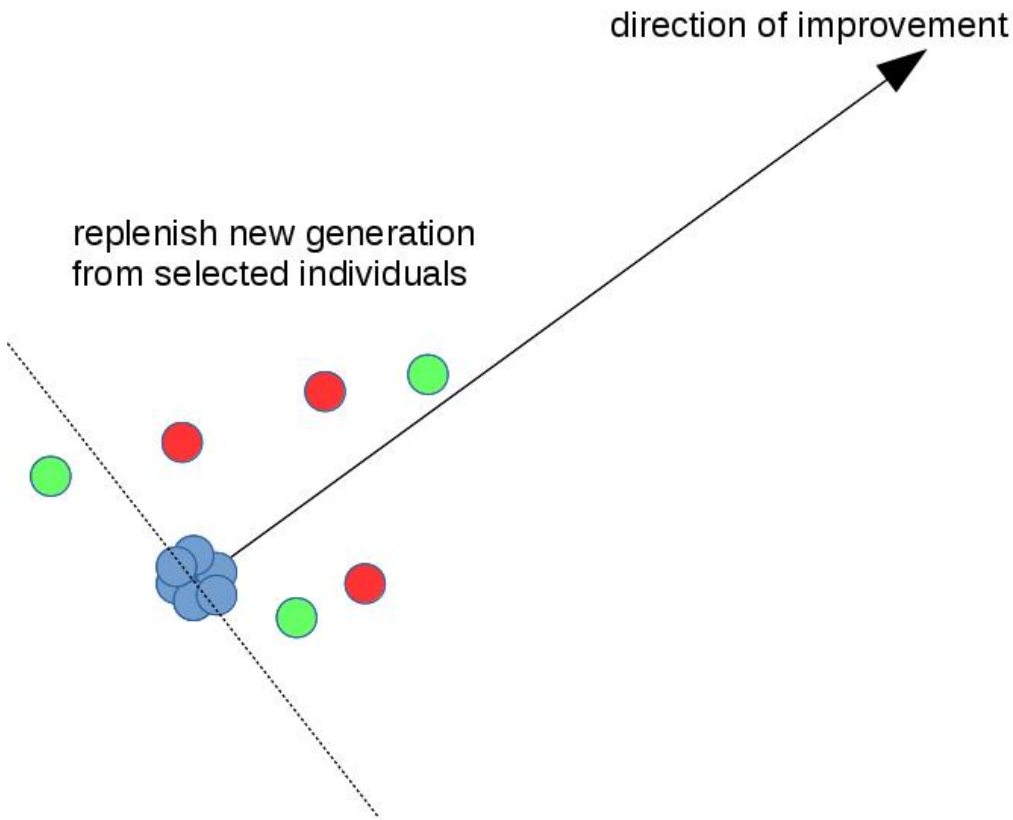
# Evolution Strategies



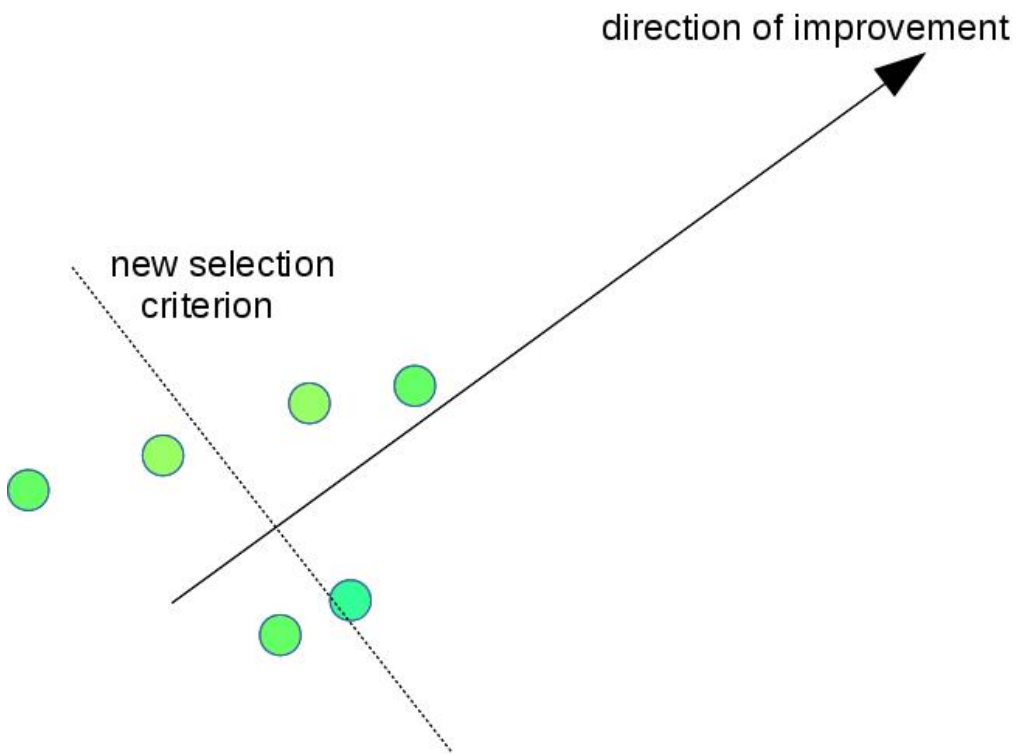
# Evolution Strategies



# Evolution Strategies



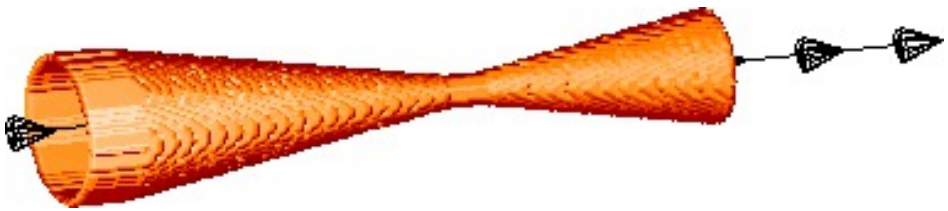
# Evolution Strategies



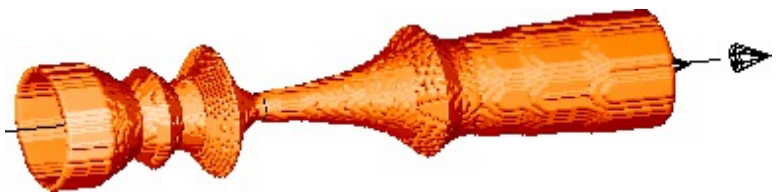
# An early example of artificial evolution

Experimental contour optimization of a supersonic flashing flow nozzle (1967-1969) Hans-Paul Schwefel

Initial shape

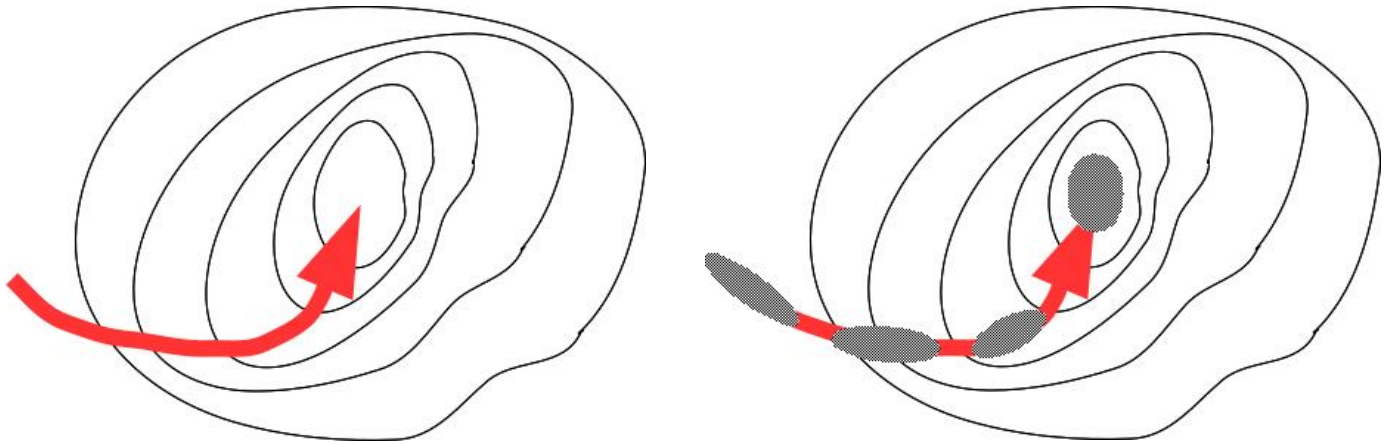


Evolved shape



Klockgether, J.; Schwefel, H.P. (1970) Two-phase nozzle and hollow core jet experiments. Engineering Aspects of Magnetohydrodynamics. (not available online, image credit: H.-P. Schwefel)

# Multidimensional Mutations in ES



Generation of offspring: $x = m + \mathcal{N}(0, C)$, i.e. "directional" noise!

$m$ stands for the vector $(m_1, \ldots, m_L)^\top$ describing a parent

Matrix $C$ describes preferred directions, e.g. one of the following:

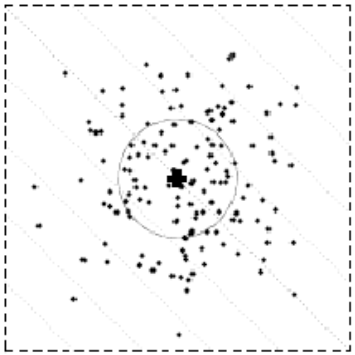
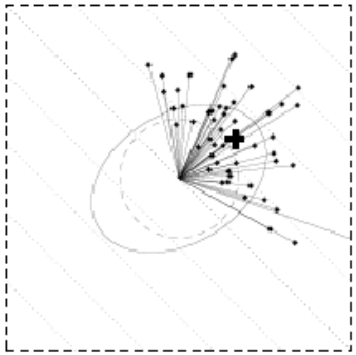
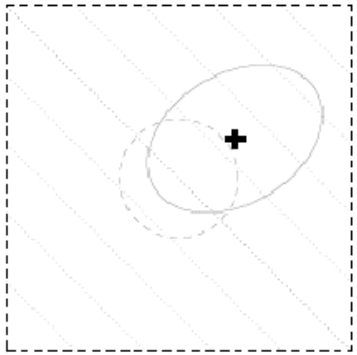
- $C = \mathrm{diag}(\sigma, ..., \sigma)$ for homogeneous uncorrelated mutations,
- $C = \mathrm{diag}(\sigma_1, ..., \sigma_L)$ for gene-dependent mutation rates or
- $C = (C_{ij})$ full covariance matrix for correlated mutations

$C$ needs to be updated as well (after selection!), see next slide

A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, 2008.

# Correlation matrix adaptation (CMA)

(Used also in hybrid algorithms beyond ES)

Off-spring vectors for one parent $m$: $x_i := m + z_i, \quad z_i \sim \mathcal{N}(0, C)$

Select $\lambda$ best individuals from the $x_i$ [i.e. $(1, \lambda)$ - ES, see below]

New state vector by averaging: $m' := \frac{1}{\lambda} \Sigma_{i \text{ selected}} x_i$ to

Smoothen fitness fluctuations; alternatively: $m' =$ best (i.e. $\lambda = 1$)

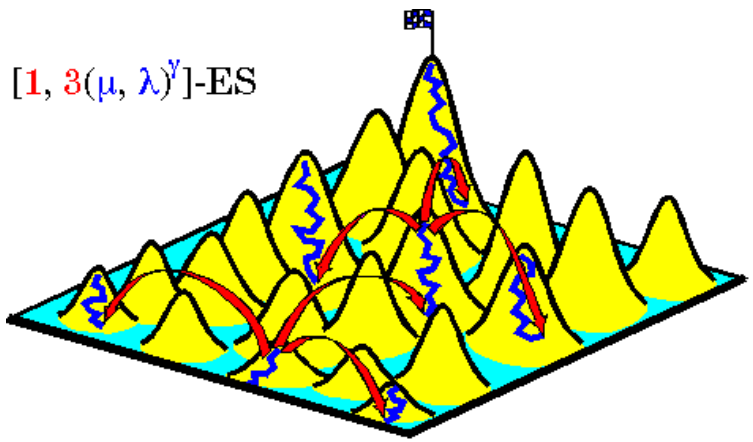
Correlations among offspring: $\Xi := \frac{1}{\lambda} \Sigma_{i \text{ selected}} (x_i - m')(x_i - m')^\top$

Update correlation matrix: $C := (1 - \varepsilon) C + \varepsilon \Xi$ (for $\lambda \geq 1$)

Heuristic 1/5 rule: If less than 1/5 of the children are better than their parents then decrease size of mutations, i.e. $C := \rho\, C$, $\rho < 1$

# Nested Evolution Strategy*

- Hills are not independently distributed (hills of hills)
- Find a local maximum as a start state
- Generate 3 offspring populations (founder populations) that then evolve in isolation
- Local hill-climbing (if convergent: increase diversity of offspring populations)
- Select only highest population
- Walking process from peak to peak within an "ordered hill scenery" named Meta-Evolution
- Takes the role of crossover in GA




web.archive.org/web/20180422161252/http://www.bionik.tu-berlin.de/

# Evolution strategies*

## Naming convention for variants

- $(\mu, \lambda)$: From $\mu$ parents $\lambda$ children (mutants) are generated. Selection only from the set of the $\lambda$ children
- $(\mu + \lambda)$: Same as above, but selection from the set of $\mu$ parents plus $\lambda$ children
- $(\mu', \lambda'(\mu, \lambda)^\gamma)$: Hierarchical (nested) variant: From $\mu'$ parent sub-populations, $\lambda'$ child-populations are generated. Then the children are isolated for $\gamma$ generations where each time $\lambda$ children are created (total population is $\lambda\lambda'$) and $\mu$ are selected. Then the best $\mu'$ subpopulations are selected and become parents for the new cycle of again $\gamma$ generations
- Analogous: $(\mu' + \lambda'(\mu, \lambda)^\gamma)$, $(\mu' + \lambda'(\mu + \lambda)^\gamma)$, $(\mu', \lambda'(\mu + \lambda)^\gamma)$

This convention can be adapted also for the algorithms that we will discuss during the next weeks. Try to remember the options, not the details.

# Evolution Strategies: Conclusion

- Evolution Strategies create noisy versions of initial solutions and select the best ones
- What strategy (last slide) performs best is problem-dependent
- The way the solutions are represented as vectors is crucial, but a genetic encoding is not attempted here
- More about encoding in the context of genetic algorithms, ... next weeks.
- Many variations of ES exist: Elitism, islands, adaptation of parameters, ... to be discussed later as well
- Recent overview: Z.H. Zhan et al. (2022) A survey on evolutionary computation for complex continuous optimization. *Artif. Intell. Rev.* **55**, 59-110.

Course: Natural Computing (week 3)

# Particle Swarm Optimisation (PSO)


J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

# Swarm intelligence / Collective intelligence

- Using "The whole is more than the sum of its parts" as a motivation is agreeable, but it's even less clear than natural evaluation for GA
- Mechanisms: Cooperation and competition self-organisation, and communication
- May show abilities that are not present in the individuals ('intelligence')
- "Super-organism" emerging from the interaction of individuals
- Examples: Social animals (incl. ants), smart mobs, immune system, biological neural networks, internet, swarm robotics

Beni, G., Wang, J.: Swarm Intelligence in Cellular Robotic Systems, Proc. NATO Adv. Workshop on Robots and Biological Systems, Tuscany, Italy, 26–30/6 (1989)

# Swarm intelligence: Application areas

- Biological and social modelling
- Movie effects
- Dynamic optimisation
  - routing optimisation
  - structure optimisation
  - data mining, data clustering
- Organic computing
- Swarm robotics

# Swarms in robotics and biology

AI/Robotics

- Main interest in pattern synthesis
  - Self-organization
  - Self-reproduction
  - Self-healing
  - Self-configuration
- Construction

Biology/Sociology

- Main interest in pattern analysis
  - Recognizing best pattern
  - Optimizing path
  - Minimal conditions
  - "what" as well as "why"
- Modelling

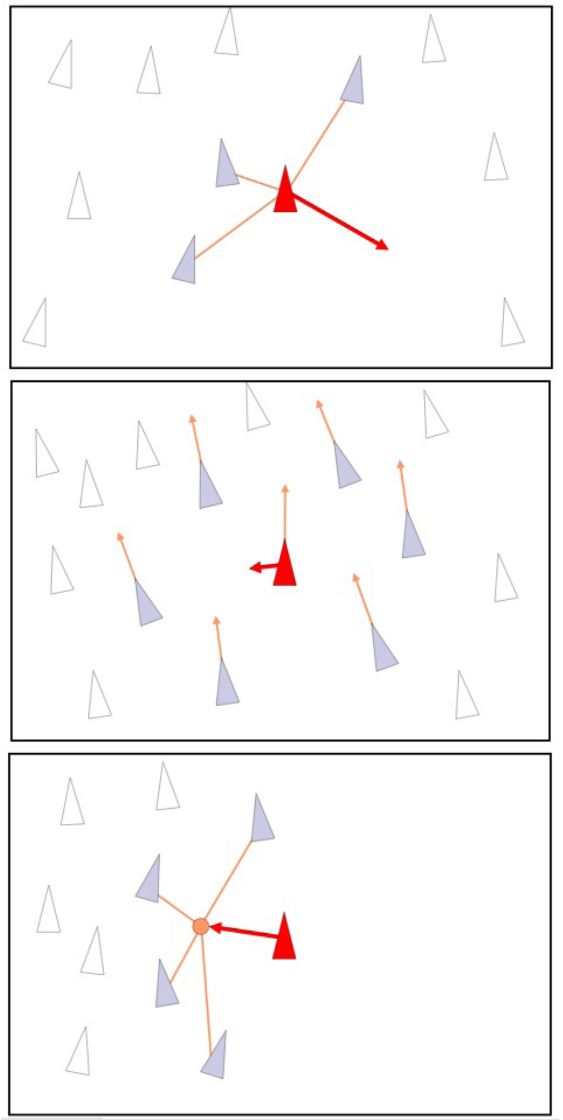
This course: optimisation, computation, dynamical systems

# Complex behaviour from simple rules (Reynolds' rules)

Rule 1: **Separation**
Avoid Collision with neighbouring agents

Rule 2: **Alignment**
Match the velocity of neighbouring agents

Rule 3: **Cohesion**
Stay near neighbouring agents



Craig W. Reynolds: Flocks, Herds and Schools: A distributed behavioral model, 1987

# Towards a computational principle

- Evaluate present position (i.e. its fitness)
- Compare it to own previous better positions and better ones the among neighbouring agents
- Keep moving, and imitate self and others

- **Hypothesis:** There are two major sources of cognition, namely, own experience and communication from others.

Leon Festinger, 1954/1999, Social Communication and Cognition

# The canonical PSO algorithm

Each particle $\mathsf{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ represents a potential solution

- Determine velocities based on own experience & good example

$$\mathsf{v}_{i,t+1} = \omega\, \mathsf{v}_{i,t} + \alpha_1\, \mathsf{r}_1 \circ (\mathsf{p}_i - \mathsf{x}_{i,t}) + \alpha_2\, \mathsf{r}_2 \circ (\mathsf{g} - \mathsf{x}_{i,t})$$

$\circ$: component-wise multiplication

- Using random vectors $r_1$, $r_2$ with components in $U = [0, 1]$
- Update positions (i.e. time unit equals "1")

$$x_{i,t+1} \leftarrow x_{i,t} + v_{i,t+1}$$

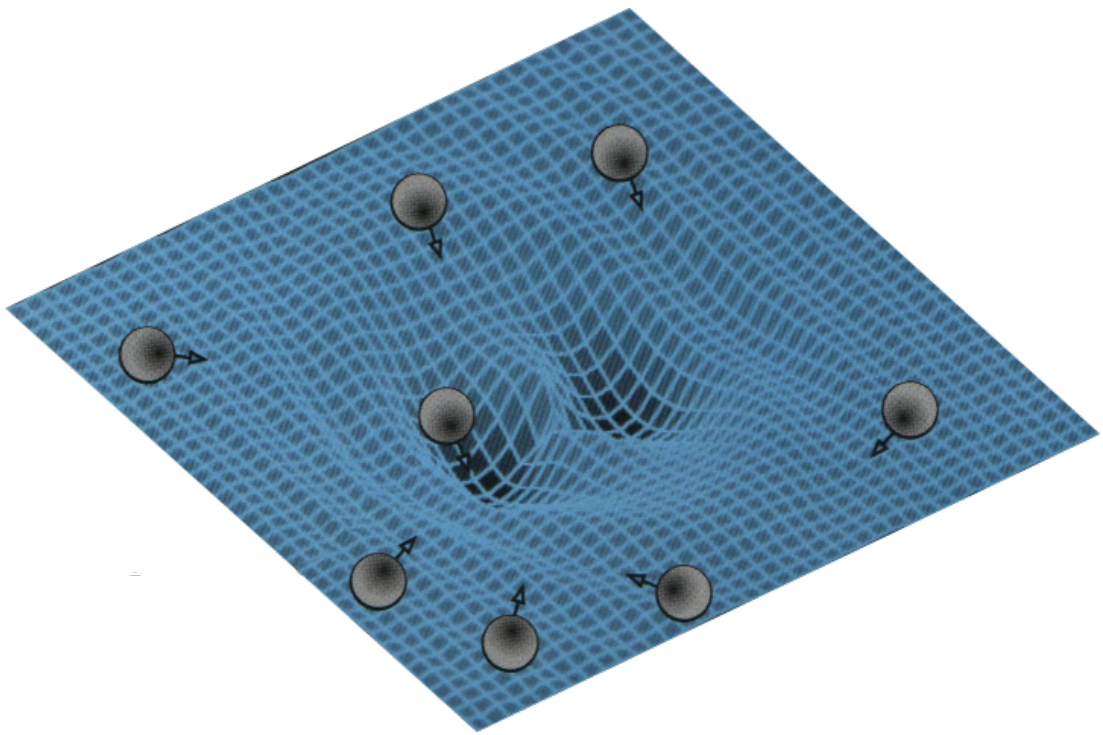
Update fitnesses $f_i = f(\mathsf{x}_i)$ for the new particle positions:

- Update local bests (for a minimisation problem)
  $\mathsf{p}_i \leftarrow \mathsf{x}_i$ if $f(\mathsf{x}_i) < f(\mathsf{p}_i)$
- Update global best (for a minimisation problem)
  $\mathsf{g} \leftarrow \mathsf{x}_i$ if $f(\mathsf{x}_i) < f(\mathsf{g})$

# Particle Swarm Optimisation

$N$ particles at positions $x_1, \ldots, x_N$ with velocities $v_1, \ldots, v_N$

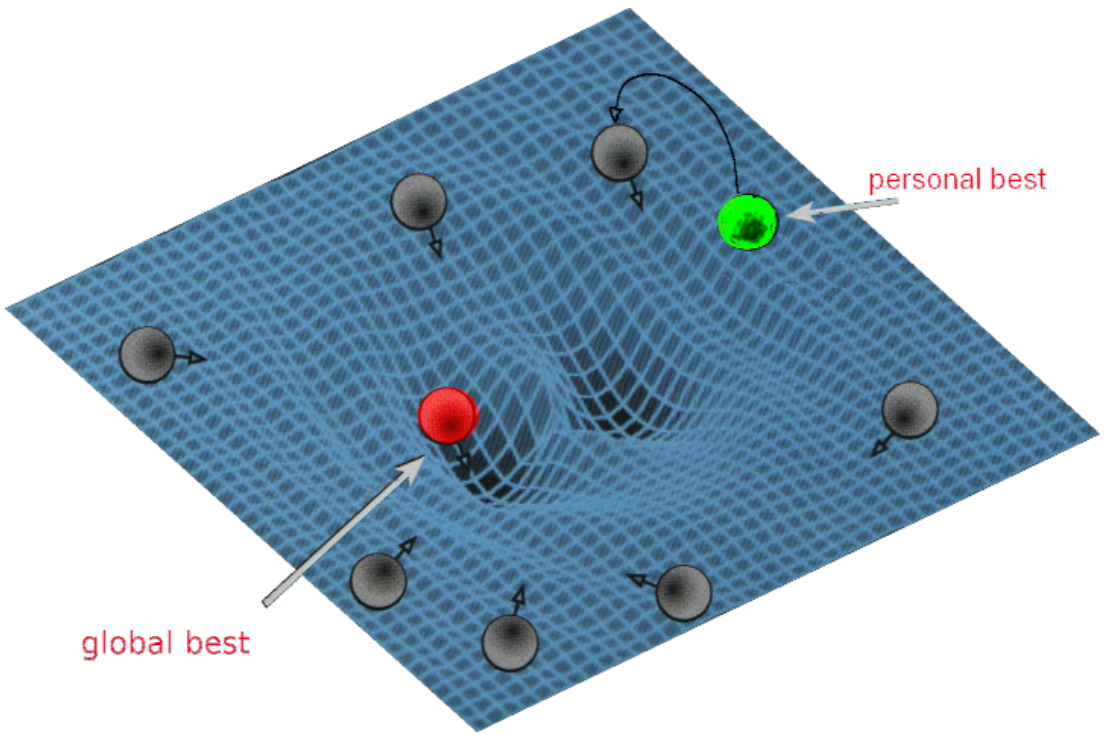
$$
\begin{aligned}
v_{i,t+1} &= \omega v_{i,t} + \alpha_1 r_1 \circ (p_i - x_{i,t}) + \alpha_2 r_2 \circ (g - x_{i,t}) \\
x_{i,t+1} &= x_{i,t} + v_{i,t+1} \qquad\qquad i = 1, \ldots, N
\end{aligned}
$$



# Particle Swarm Optimisation

$N$ particles at positions $x_1, \ldots, x_N$ with velocities $v_1, \ldots, v_N$

$$
\begin{aligned}
v_{i,t+1} &= \omega v_{i,t} + \alpha_1 r_1 \circ (p_i - x_{i,t}) + \alpha_2 r_2 \circ (g - x_{i,t}) \\
x_{i,t+1} &= x_{i,t} + v_{i,t+1}
\end{aligned}
$$



# PSO algorithm: Initialization

| | |
|---|---|
| Objective function | $f : \mathbb{R}^m \to \mathbb{R}$ |
| Number of particles | $n = 20 \ldots 200$ |
| Particle positions | $x_i \in \mathbb{R}^m,\ i = 1 \ldots n$ |
| Particle velocities | $v_i \in \mathbb{R}^m,\ i = 1 \ldots n$ |
| Best so-far for each particle ("simple nostalgia") | $\hat{x}_i$ |
| Global best ("group norm")[1] | $\hat{g}$ |
| Parameters | $\omega,\ \alpha_1,\ \alpha_2$ |

[1] this can be the best solution present within the current swarm or the best solution found by any particle so far. The former seems to make more sense, but the latter usually works better.

# Particle Swarm Optimization (PSO)

- Methods for finding optimal solutions of an objective function
- Direct search, i.e. no explicit gradient
- Simple and quasi-identical units
- Can be used asynchronously, centralized control not needed
- 'Medium' number of units: $\sim 10^1 - 10^3$ (practically 20 - 30)
- Redundancy for reliability and convergence (should be small)
- PSO is one of the computational algorithms in the field that is "inspired" by swarm intelligence (another one is ACO)

J. Kennedy, and R. Eberhart, Particle swarm optimization, in Proc. IEEE. Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

# How does it work?

Exploitation occurs, if a new (better) position is found in the search space, but also the particle dynamics shows both tendencies:

- **Exploratory behaviour**: Search a broad region of space
- **Exploitative behaviour**: Locally oriented search to approach a (possibly local) optimum

Parameters to be chosen to properly balance between exploration and exploitation, i.e. to avoid premature convergence to a local optimum yet still ensure a good rate of convergence to the optimum.

# Preliminary conclusions

- The standard PSO is biologically inspired, but it perhaps more similar to a physical system reminiscent of the 3-body problem
- The local best introduces a (non-physical) force towards a past state
- Both bests have a similar effect as a gradient
- For large values of $\alpha_1 + \alpha_2$ the behaviour is zig-zagging (which is non-physical, but very good for exploration)
- Values of $\omega \approx 0$ are usually not good, all other $\omega \in (-1, 1)$ can be used, but $\omega > 0$ can be more easily tuned
- Particles do not interact with each other, except via the global best (which changes often only initially)
- The behaviour depends on the parameters $\alpha_1$, $\alpha_2$, and $\omega$: Larger values tend to imply less stability.

# PSO Applications

- Evolving structure and weights of neural networks
- Robot path planning, localisation
- Electrical distribution systems under uncertainty in load
- Automatic control systems, e.g. tuning PID control
- Communication Systems: E.g. channel equalisation by PSO designed ANN
- Operations research: Real-time task assignment in heterogeneous multiprocessor systems
- Mechanical Engineering: Optimisation of composite structures
- Sampling of probabilistic models

See: Zhang, Wang and Ji: A comprehensive review on PSO algorithm and its applications, Math. Probl. Eng. 2015
(See also more recent more specialised reviews)

Course: Natural Computing (week 3)

Particle Swarm Optimisation (PSO)

# II. The algorithm and some variants


J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

# The PSO algorithm: Initialisation

- Initialize the particle positions and their velocities

$x = \text{lower}_{\text{limit}} + (\text{upper}_{\text{limit}} - \text{lower}_{\text{limit}}) \times \text{rand}(n_{\text{particles}}, m_{\text{dimensions}})$

assert $x$.matrix == $(n_{\text{particles}}, m_{\text{dimensions}})$

$v = \text{zeros}(x.\text{matrix})$

- Initialize the global and local fitness to the worst possible

$f^g = -\infty$ (or $\infty$ for minimisation problem)

$f_i^p = f^g \ \forall i$

- Initialize parameters

$\omega = 0.7$ (range 0.1 ... 0.9)

$\alpha_1 + \alpha_2 = 4$ (range 1 ... $\approx$5, usually both equal)

$n = 25$ (range 20 ... 200, more for more complex problems)

max velocity (not larger than, typically 10-20% of the range of $x$)

# The PSO algorithm

initialise: $\mathsf{g} \in \mathbb{R}^d$, $f^g \in \mathbb{R}$, $n \in \mathbb{N}$, $\alpha_1, \alpha_2, \omega \in \mathbb{R}$

$\forall i = 1, \ldots, n : \mathsf{v}_i, \mathsf{x}_i, \mathsf{p}_i \in \mathbb{R}^d$, $f_i^p \in \mathbb{R}$

loop: maximisation

fitness: Calculate $\forall i$: $f_i = \text{fitness}(\mathsf{x}_i)$ (it's actually a "cost")

personal best: $\forall i$: if $f_i^p < f_i$ then $\mathsf{p}_i = \mathsf{x}_i$, $f_i^p = f_i$

global best: $\forall i$: if $f^g < f_i$ then $\mathsf{g} = \mathsf{x}_i$, $f^g = f_i$

or (best ever): $\forall i$: if $f^g < f_i^p$ then $\mathsf{g} = \mathsf{x}_i$, $f^g = f_i^p$

update velocities: $\forall i$: $\mathsf{v}_i = \omega \mathsf{v}_i + \alpha_1[R](\mathsf{p}_i - \mathsf{x}_i) + \alpha_2[R](\mathsf{g} - \mathsf{x}_i)$

update positions: $\forall i$: $\mathsf{x}_i = \mathsf{x}_i + \mathsf{v}_i$

check: terminate / continue

Note: Use each time a different realisation for random $[R] \in [0, 1]$.
Update of x and v can be synchronous or asynchronous.

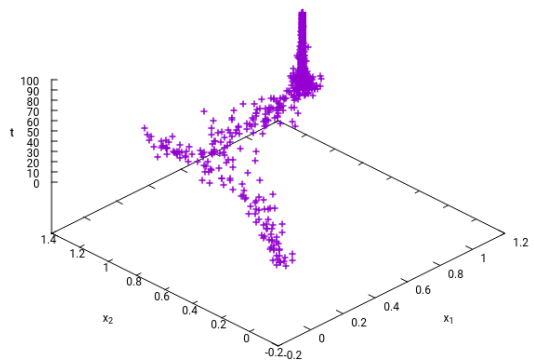
# Convergence

- **Failure**: Swarm diverges or remains itinerant away from optima
- **Optimally**: Global best approaches global optimum (swarm may still oscillate, even divergence would be tolerable)
- **Typically**: Global best approaches a local optimum (premature collapse of the swarm)

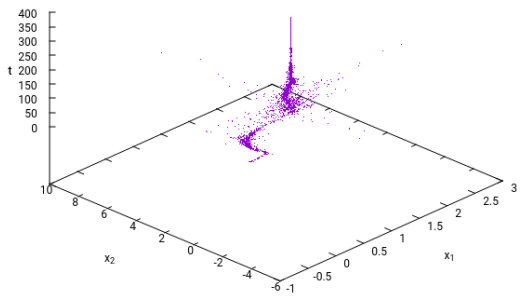
- Convergence is not necessary (global or local bests remember previous good solutions)
- Convergence can be useful to search the space around a good solution more carefully (a.k.a. "constriction")
- Alternatively, add a hill-climbing stage to the PSO algorithm

# How does it work: Solving the all-ones problem by PSO

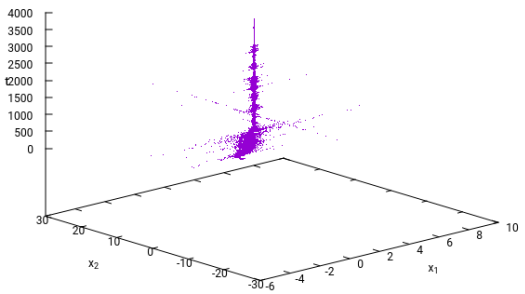
- Fitness: Distance to $(1, 1, \ldots, 1) \in \mathbb{R}^D$, continuous search space; shown is projection to two coordinates
- Initialisation: Randomly near origin, $N = 10$ particles, $\alpha_1 = \alpha_2 = 2$, $\omega = 0.6$ (mildly convergent dynamics)




$D = 2$




$D = 10$




$D = 100$

Search seems to proceed with one (or a few) dimensions at a time, which seems similar to mutations in GA.

# The axis bias in PSO

**Observation**: Particles can often be seen to stick to the coordinate axes (if global optimum is at origin)

The noise terms $r_1$ and $r_2$ are ineffective if $x_{i,t}$ coincides with p or g in some of the dimensions, i.e. these subspaces cannot be further explored.

This bias may be useful to obtain sparse solutions or building blocks, but was not recognised when the PSO algorithm was new.

The axis bias exists only, if the algorithm is convergent, and disappears for critical parameters.

For information on the axis bias see Spears, Green and Spears, IJSIR, 2010.

# How does it work? Exploration

- Particles are attracted by the personal and global "bests", but for good exploration not just a bit.
- Consider e.g. the term $\alpha_2 \mathsf{r}_2 \circ (\mathsf{g} - \mathsf{x}_{i,t})$
  - includes a random number $\mathsf{r}_2 \sim U[0,1]$ with mean $\langle \mathsf{r}_2 \rangle = \frac{1}{2}$
  - for $\alpha_2 = 4$ (say $\alpha_1 = \omega = 0$) the velocity averages to twice the distance to g, so that the position tends to be reflected to the other side of g, and the randomness guarantees exploration
  - any value $\alpha_2 \in [0, 4)$ leads to convergence (with probability 1)
- $\alpha_1$ and $\alpha_2$ have similar effect: Consider $\alpha = \alpha_1 + \alpha_2$ ($\alpha_2 > 0$)
- $\alpha = 4$ is a critical parameter for a simplified algorithm (Clerc & Kennedy, 2002).
- For a more realistic theory the critical parameters are different, $\alpha \geq 4$: stochastic convergence still possible ($\alpha_1$, $\alpha_2$, $\omega \neq 0$), but this depends also on runtime. (Erskine, Joyce, H., 2017)

# Additional features to control the swarm*

- **Update** of the global best:
  - synchronous: after all particles have moved
  - asynchronous: each time a particle finds a good solution
  - temporarily: only within generation (default is "best ever")
- **Velocity control** to counteract convergence/divergence
  - clamping: make sure all particles' velocity components remain in a certain range
  - control: add a term $\kappa\,(v - v_{\text{target}})$ and reduce $v_{\text{target}}$ over time
  - "constriction" to force convergence before termination
- **Limited search space**: Reflect particles when they hit the boundary of the search space (see initialisation)
- **Detection of premature convergence**: Add noise, increase parameters, or restart

Note that in this way additional parameters are introduced!

# PSO variant with separation*

Include a repulsive term

$$\begin{aligned} v_i &= \omega v_i + \alpha_1 r_1 \circ (p_i - x_i) + \alpha_2 r_2 \circ (g - x_i) + \alpha_3 Z \\ x_i &= x_i + v_i \end{aligned}$$

with

$$Z = -\sum_{k \neq i} \frac{r_3 \circ (x_k - x_i)}{\varepsilon + \|x_k - x_i\|^2}$$

Typically, $Z$ is small if particles are widely dispersed, but is large if another particle gets close. For small $\varepsilon$, the dynamics can become unstable. Usually, $\alpha_3 > 0$. What would happen for $\alpha_3 < 0$?

More terms can be added such as an alignment term, attraction to other particles, velocity control, two different global best terms etc.

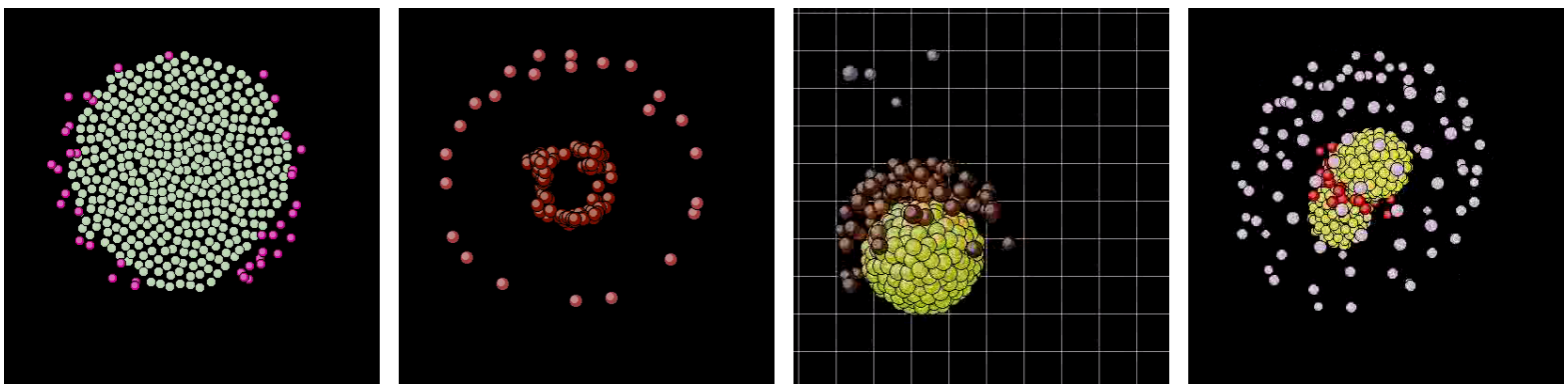
# Predator-Prey PSO*

- Two types of particles: "Predators" and "prey"
  - Predators are attracted to prey (or to the global-best particle)
  - Prey is repulsed by predators
- Can take different roles in the exploration-exploitation dilemma

For details see Silva, Neves & Costa AICS, 2002

# Heterogeneous swarms* ("artificial chemistry")

- The complex dynamics of heterogeneous swarms is interesting beyond the application in optimisation (i.e. no "bests" here but attraction/repulsion w.r.t. other particles)
- A swarm that consists of several species of interacting particles can serve as a model for natural phenomena
- Each species is characterised by different set of parameters and interacts with particles from other species in a different way.



See Hiroki Sayama's papers on artificial (swarm) chemistry.

Images form Erskine & Herrmann (2015) Artificial Life **21**(4) 481-500.

Course: Natural Computing (week 3)

Particle Swarm Optimisation (PSO)

# III. Swarm topology*


J. Michael Herrmann

School of Informatics, University of Edinburgh

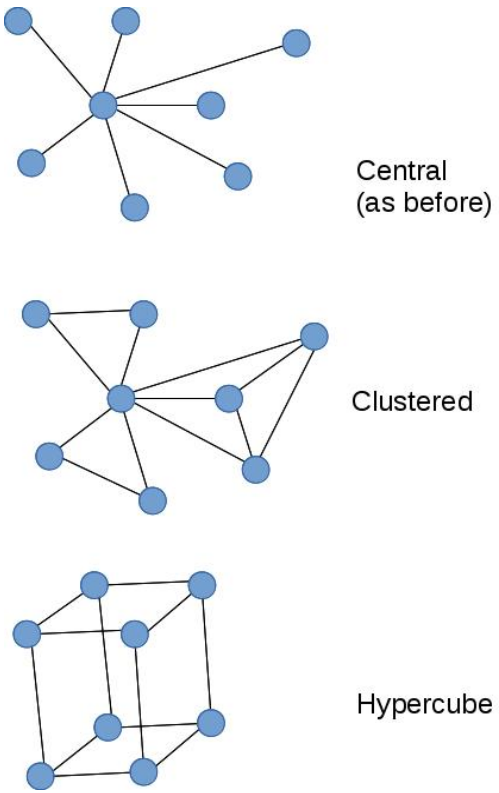
michael.herrmann@ed.ac.uk, +44 131 6 517177

# Innovative topologies*

Specified by:

- Mean degree
- Degree distribution
- Clustering
- Heterogeneity
- Small-world-ness

etc.



# Topology: Restricted competition/coordination*

- Topology: Restricted competition/coordination
- Topology determines with whom to compare and thus how solutions spread through the population
- $\boldsymbol{g}_{\text{best}}$ is determined only among neighbours; $\boldsymbol{p}_{\text{best}}$ as usual
- Global version is faster but might converge to local optimum for some problems.
- Local version is a somewhat slower, not easily trapped into local optimum.
- Combination: Use global version to get rough estimate. Then use local version to refine the search.
- For some topologies analogous to islands in GA

# Fully Informed Particle Swarm (FIPS)*

- Rui Mendes et al. (2004): "Simpler, maybe better"
- Distribute total $\phi$ across neighbours using weights $\mathcal{W}(k)$ which are chosen according to quality

- All neighbours contribute to the velocity adjustment
- Best neighbour is not selected, all contribute with prob. $p_m$
- Individual $m$ itself is not included in its own neighbourhood $\mathcal{N}_m$

$$r_k = U\left[0, \frac{r\max}{|\mathcal{N}|}\right] \quad \forall k \in \mathcal{N}$$

$U$ equi-distr. random vector

$w \in \mathbb{R}^{|\mathcal{N}|}$ weight vector:

swarm follows weighted average rather than best

$$P_m = \frac{\sum_{k\in\mathcal{N}} w_k r_k \circ \hat{x}_k}{\sum_{k\in\mathcal{N}} w_k r_k}$$

- $w$ can be constant or fitness-dependent
- Fails quite often, but results are, if successful, good (strongly dependent on good topology)

Mendes, R., Kennedy, J. Neves, J., 2004. The fully informed particle swarm: Simpler, maybe better. IEEE TA Evol. Comput. 8(3), 204-210.

# Parameters, Conditions, & Tweaks*

- Initialization methods
- Population topology (e.g. "tribes")
- Variable population size:
  Births, deaths, migration
- Adaptive parameters
- Population diameter
  Limiting domain ($X_{MAX}$, $V_{MAX}$)
  (e.g. reflection from walls)
- Multiobjective optimization
- "Subvector" techniques
- Norms other than Euclidean
- Comparison over problem spaces
- Hybrids, see e.g. differential evolution (DE)

Eberhart, Y Shi, J Kennedy (2001) Swarm Intelligence. Morgan Kaufmann.

# Conclusion: Why is PSO interesting?

- Attraction to better states is similar to a gradient.
- Axis bias could support a mechanism similar to building blocks in GA, especially for non-trivial topologies.
- Velocities incorporate the current knowledge about the problem (similar to CMA?)
- Dynamics is non-trivial: A particle can be unstable, if it is the best, i.e. if $p = g$, but stable, if $p \neq g$. This means that the currently best particle moves on, while g memorises its state, and the other particles continue with their local search.
- PSO seems to be a prototype for the majority of existing MHO algorithms (just replace a term or add another term)
- 650,000 papers* referring to PSO can't all be wrong.

*Google Scholar