Lecture

Organic Computing II

Summer term 2019

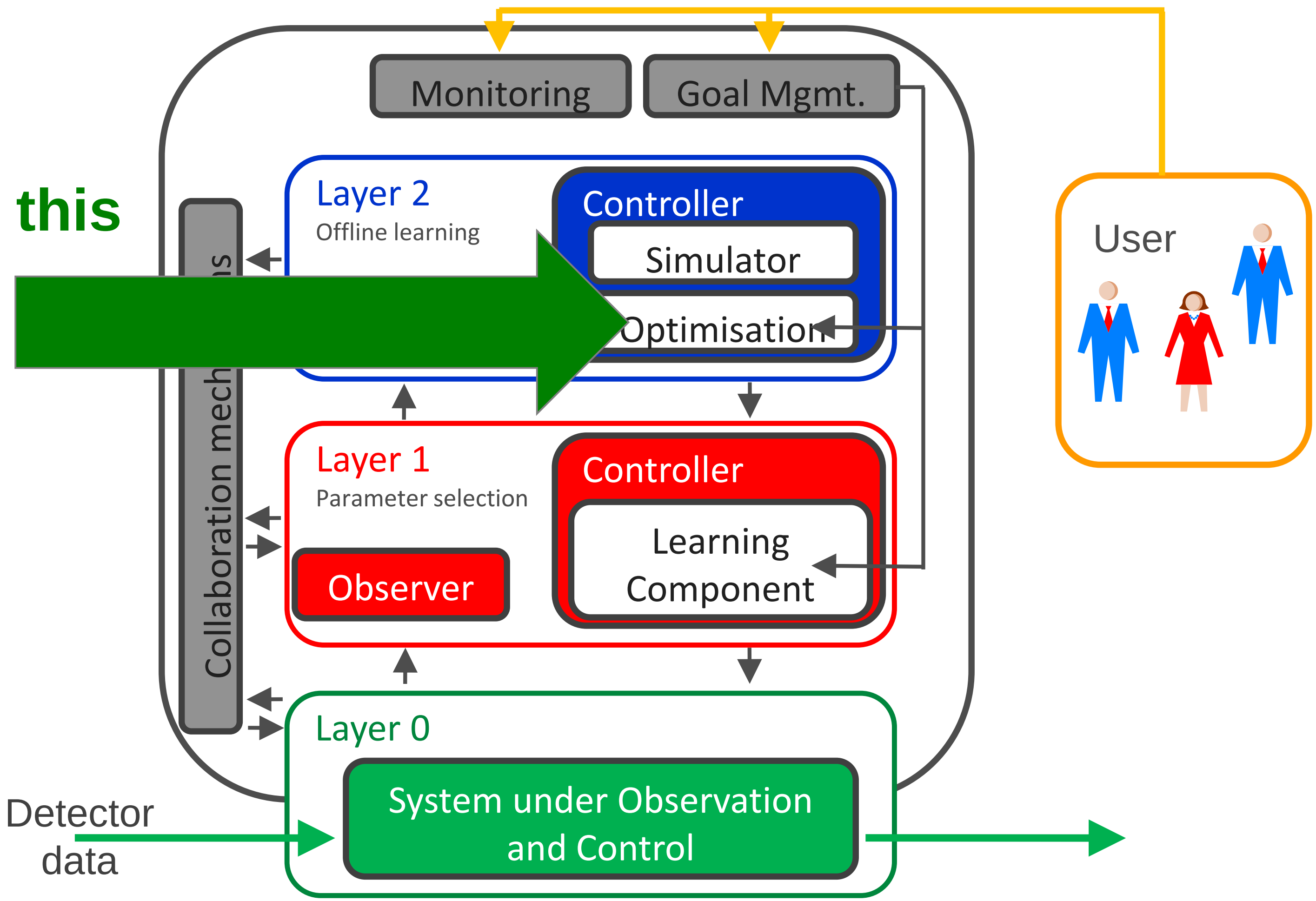
Chapter 5: Optimisation

Lecturer: Jörg Hähner

Slides have been created in cooperation with Prof. Dr. Sven Tomforde (Uni Kassel),
Prof. Dr. Müller-Schloer (Uni Hannover) and Prof. Dr. Jörg Hähner (Uni Augsburg)

ORGANIC COMPUTING

Focus of this chapter!



Content

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings

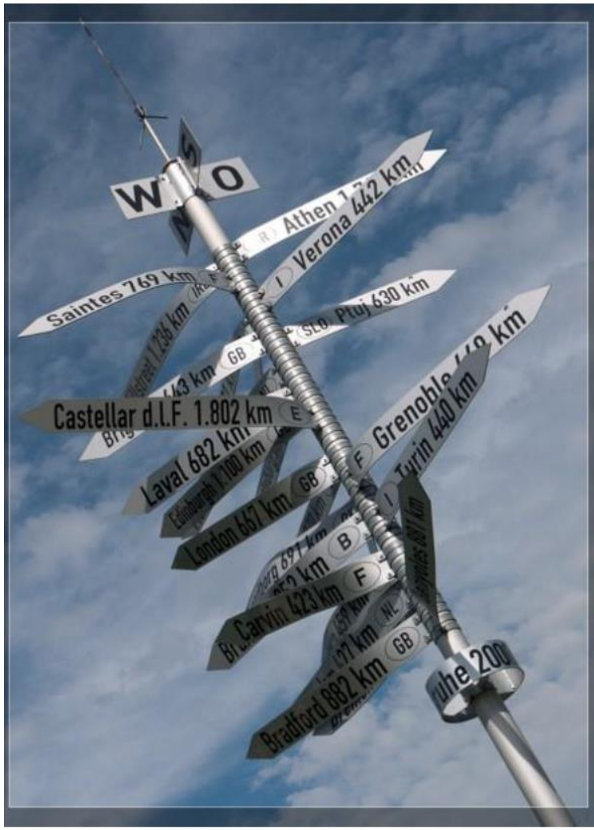
Goals

Students should be able to:

- Define what an optimisation problem is
- Outline different concepts to solve optimisation problems
- Explain nature-inspired techniques, especially Evolution Strategies, Particle Swarm Optimisation, and Simulated Annealing
- Apply the RBI algorithm
- Compare the different concepts in the context of OC problems

Agenda

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings



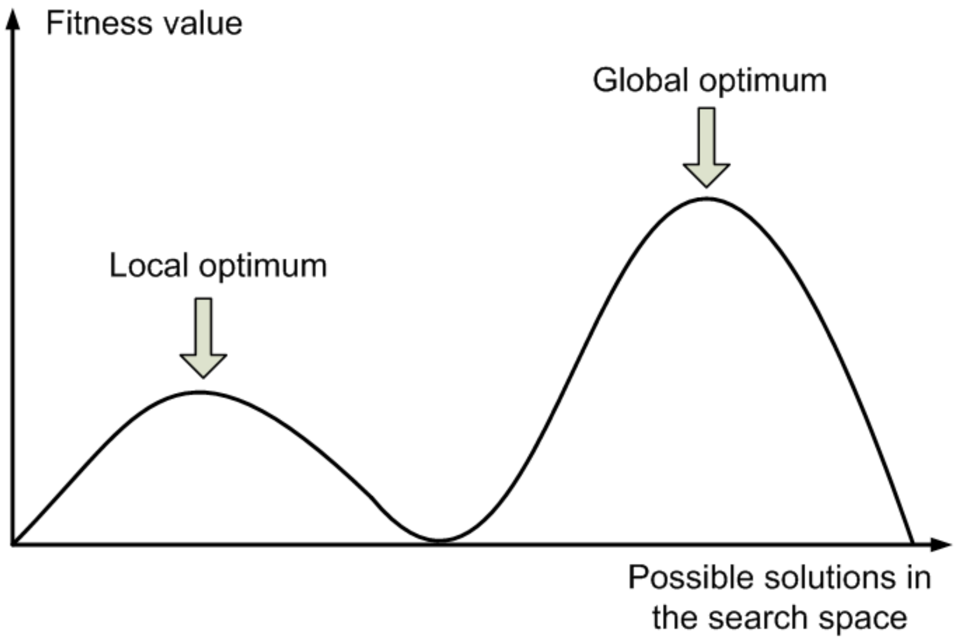
What is optimisation?

- Selecting the best element from a set
- Usually: no analytic solution possible
  ⇒ search for a good element

ORGANIC COMPUTING

Term definition: (OC) optimisation problem

- Each system configuration is a *solution (S).*
- The set of all possible system configurations is the *search space (X).*
- The fitness function *(f)* defines the quality (i.e. fitness) of the solutions in *X.*
- A fitness landscape defines the mapping between solutions in *X* and their corresponding fitness values.
- Goal: Finding the global optimum!



Properties of optimisation problems

Simple

- Few decision variables
- Differentiable
- Single modal
- Objective easy to calculate
- No or light constraints
- Feasibility easy to determine
- Single objective
- Deterministic

Hard

- Many decision variables
- Discontinuous, combinatorial
- Multi modal
- Objective difficult to calculate
- Several constraints
- Feasibility difficult to determine
- Multiple objectives
- Stochastic

Classes of fitness landscapes

- Static: The fitness landscape is fixed and does not change over time.
- Time-varying: The fitness landscape changes as a function of time.
- Self-referential: The fitness landscape changes as a function of agent behaviour.

OC systems contain agents …

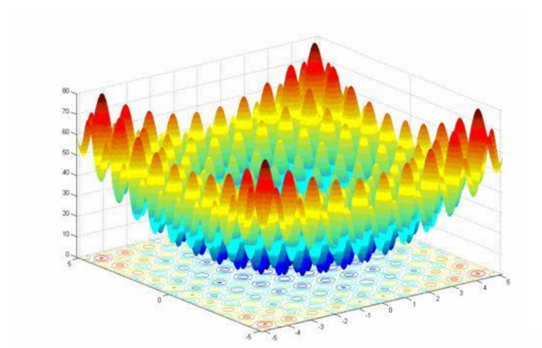
… which interact with their environment thus changing it. This changes the fitness landscape!

⇒ In many cases, OC systems have self-referential landscapes!

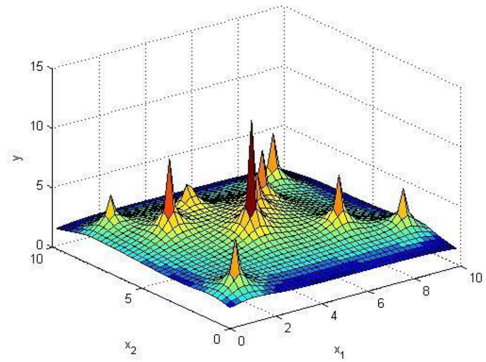
Static fitness landscapes

- Typical for problems with continuous or discrete parameter spaces.
- Problems with a continuous parameter space are
  e.g. the benchmark problems in function optimisation literature.
- A problem with a discrete parameter space is
  e.g. the Travelling Salesman Problem (TSP).

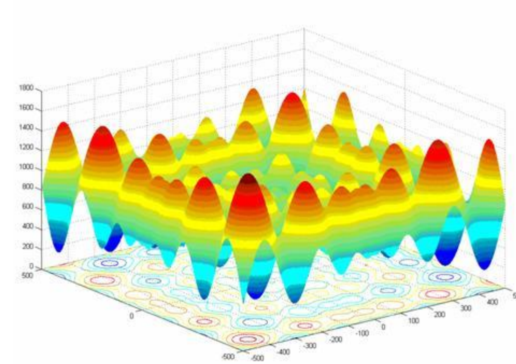
Examples for static fitness landscapes



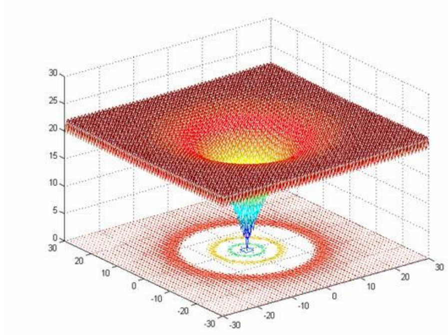
Rastrigin function



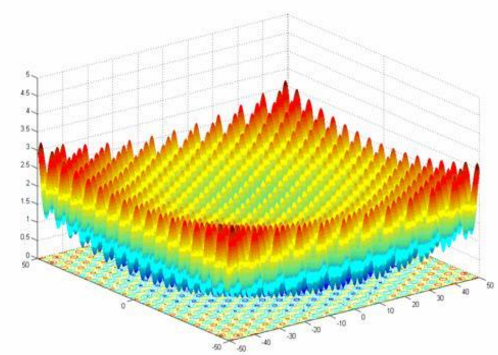
Shekel function



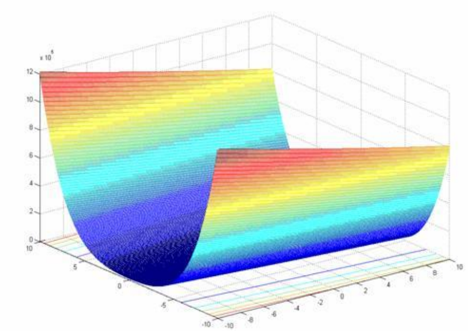
Schwefel function



Ackley function



Griewank function



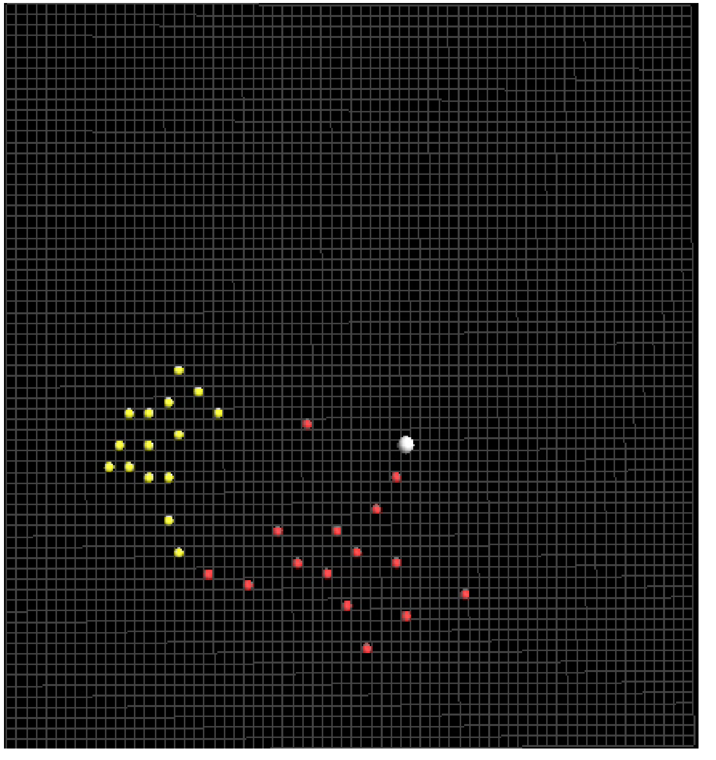
Rosenbrock function

Time-varying fitness landscapes

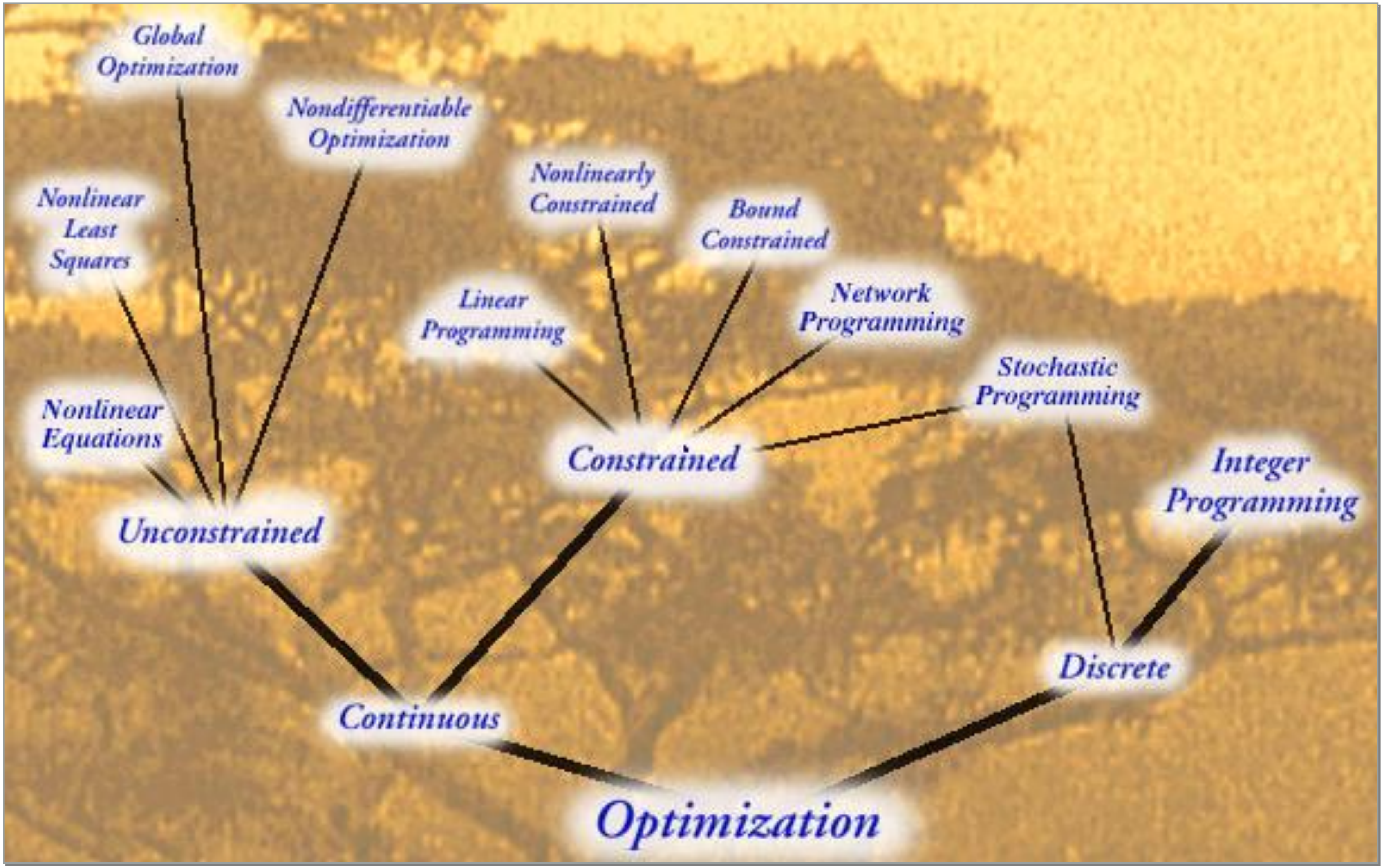
- Global optimum moves over time.
- Optimisation has to find and follow the optimum.
- Example:
  - Search space: Different locations on the earth
  - Fitness values: Temperature of the given location
  - Optimum: Locations with a temperature between 20 and 25 centigrade
  - Characteristic: Fitness landscape changes as a function of time and the optimal locations in the landscape move according to the change of seasons.

Self-referential fitness landscapes

- Global optimum moves according to the behaviour of agents.
- Optimisation has to find and follow the optimum.
- Example 1: Minority game
  - Odd number of players
  - Each must choose one of two alternatives independently at each turn.
  - The players who end up on the minority side win.
- Example 2: Predator/Prey scenario



Another classification of optimisation problems



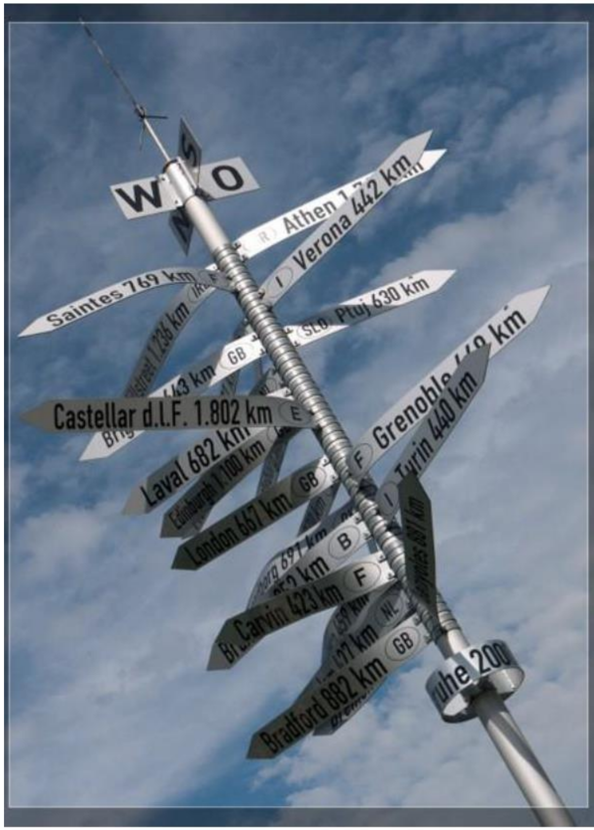
Source: Optimisation technology Center – http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/

Choosing optimisation methods

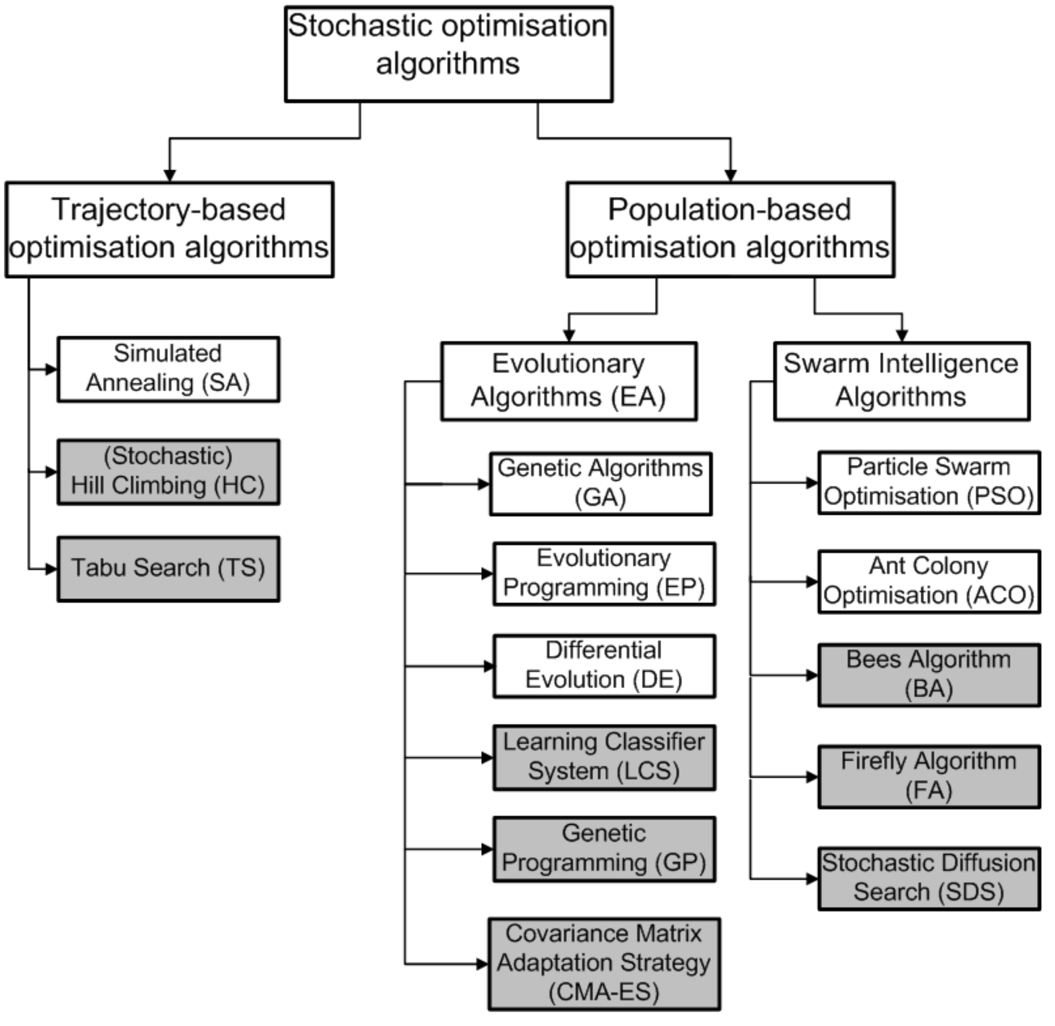
- Algorithms have very different flavour depending on the specific problem.
  - Closed form vs. numerical vs. discrete
  - Local vs. global minima
  - Running times ranging from O(1) to NP-hard
- In OC systems, optimisation at runtime ⇒ Specific requirements!
  - Fast convergence, minimised effort
  - Finding a good solution instead of the optimal one is often OK
  - Focus: Stochastic techniques

ORGANIC COMPUTING

- Motivation
- Term definition
- **Stochastic approaches**
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings



Stochastic optimisation algorithms



- Simulated Annealing (Kirkpatrick et al., 1983)

- Genetic Algorithms (Holland, 1975)
- Evolutionary Programming (Fogel, 1964)
- Differential Evolution (Storn et al., 1995)

- Particle Swarm Optimisation (Eberhart et al., 1995)
- Ant Colony Optimisation (Dorigo et al., 1996)

A general stochastic optimisation algorithm

1. Generate an initial configuration.
2. Repeat (until some termination criterion is fulfilled):
   1. Search the neighbourhood and choose a new neighbour c as candidate.
   2. Evaluate some criterion f (fitness function)
   3. $c_0 \leftarrow c$ if $f(c) > f(c_0)$ (where c_0 is the best candidate currently known – the current solution).

Examples for termination criteria:

- No improvement can be found anymore
- Fixed iteration count
- The solution's quality (the value of f) is sufficiently high

Components of a stochastic optimisation method

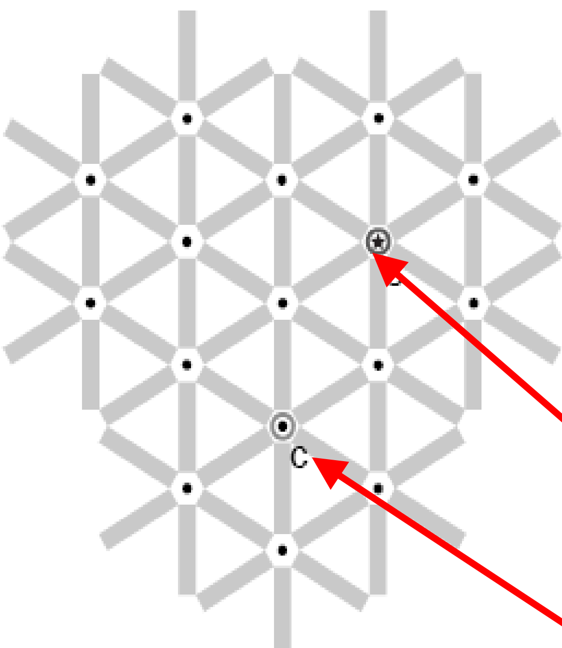
What do we need to build such a process?

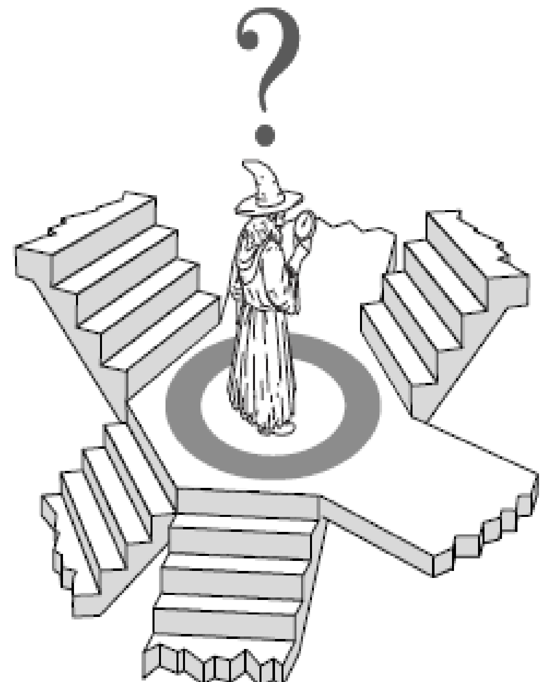
1. A method to generate the initial configuration
2. A transition or generation function to find and select a neighbour as next candidate
3. A cost or fitness function f
4. A stop criterion

This differs the most between different optimisation techniques!

Local search algorithms

Idea: Greedily select the best candidate in some neighbourhood.



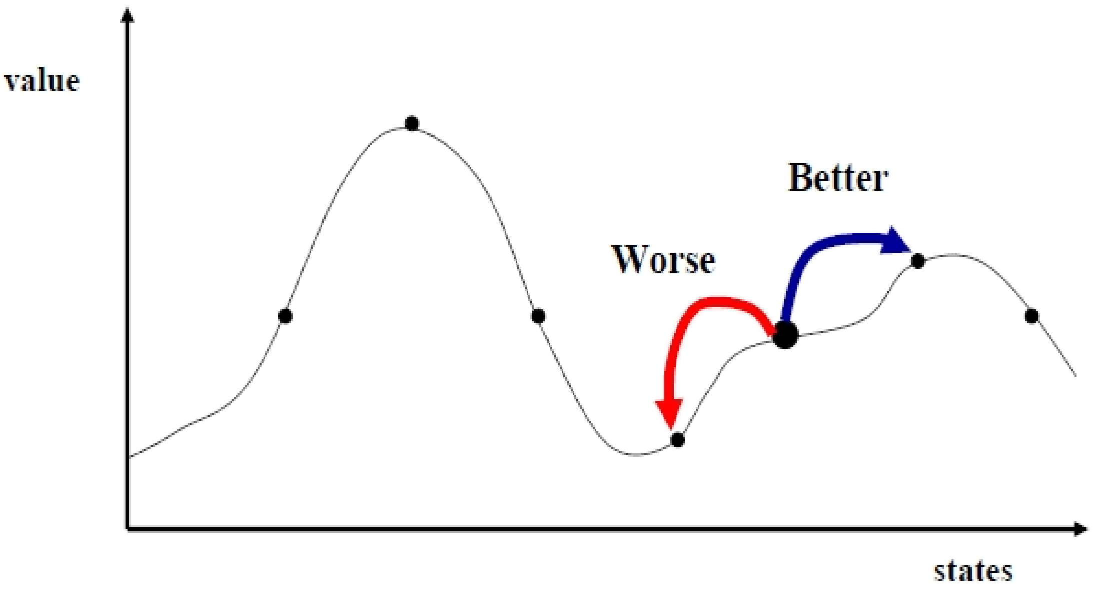


Goal position

Current position

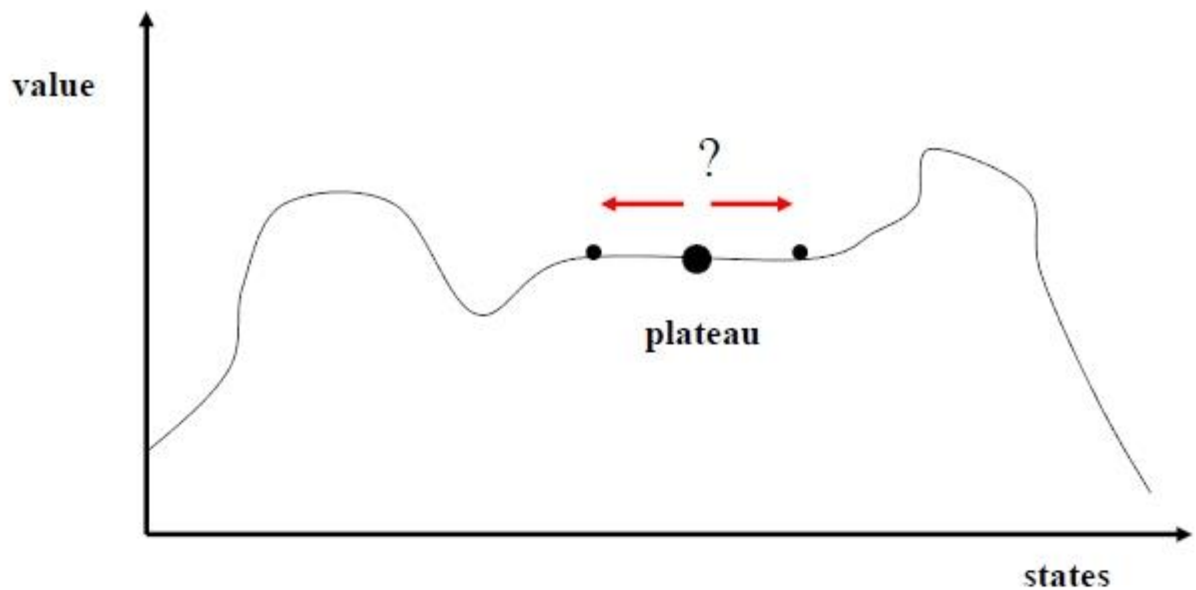
Hill climbing

- Local search technique
- Simple iterative improvement
- Accept candidate only if fitness is higher than current solution
- Process stops when no better neighbour can be found



Disadvantages of Hill Climbing

- Gets stuck in local optima quickly
- Which one depends on
  - The initial configuration
  - Step size
- In general, no upper bound for iteration length



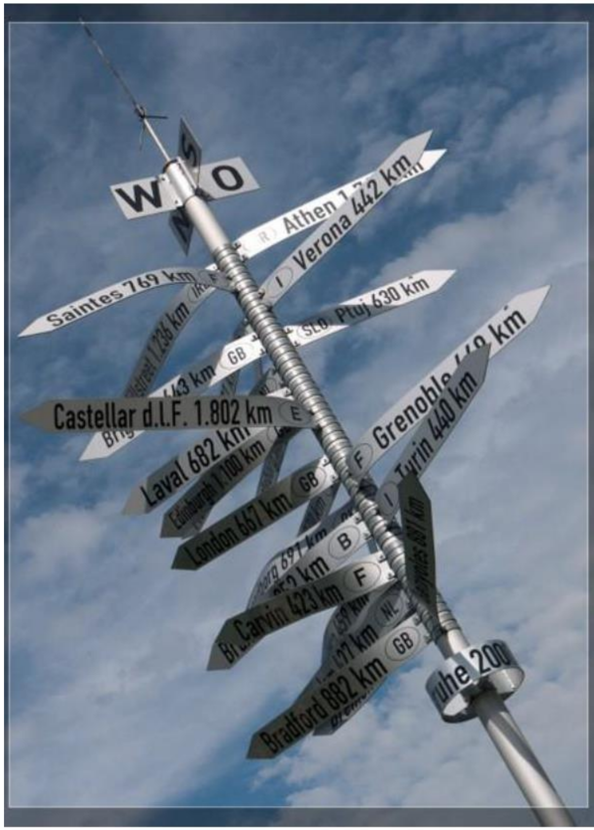
Improving Hill Climbing

- Repeat the algorithm many times with different initial configurations
- Re-use information gathered in previous runs
- Use more complex neighbourhood/generation functions to jump out of local optima
- Use more complex evaluation criteria that sometimes (randomly) accept solutions away from the (local) optimum

⇒ Better techniques are needed for complex problems!

Agenda

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings



Nature-inspired optimisation heuristics

ORGANIC COMPUTING

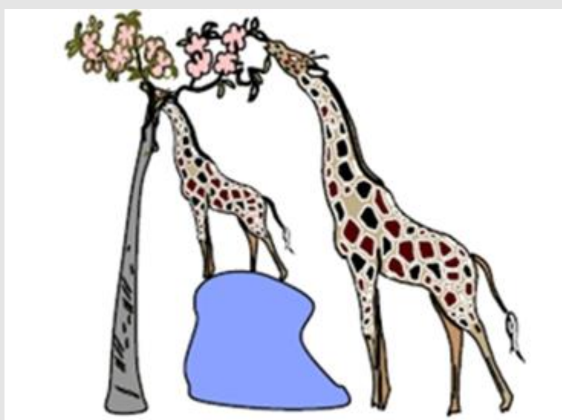
a) Physical processes

e.g. cooling process of metals

b) Genetic Algorithms

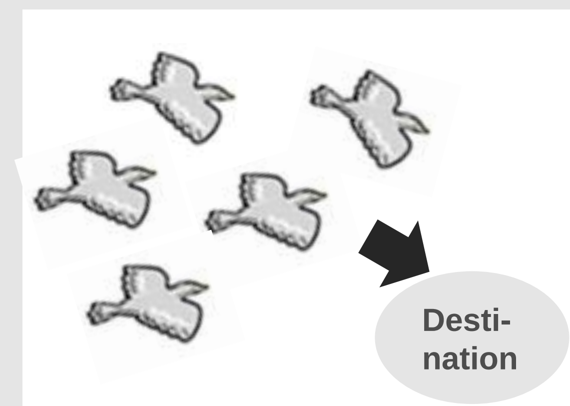
"survival of the genetically fittest"

c) Memetic Algorithms



„survival of the fittest & experienced"

d) Swarm behaviour



particle swarm: flock migration

Classes of nature-inspired optimisation heuristics

ORGANIC COMPUTING

- Physical processes: mimic the cooling process of material in the physical world (e.g. Simulated Annealing)
- Evolution: mimic the reproduction cycle of individuals in nature (i.e. Evolutionary or Genetic Algorithms)
- Memetics: combine evolutionary search with classic local search techniques (Memetic Algorithms)
- Swarms: mimic swarm-behaviour (i.e. Particle Swarms)

However, several other analogies and combinations of techniques (hybrid approaches) have been discussed, e.g. search for harmonies in music.

Simulated Annealing (SA)

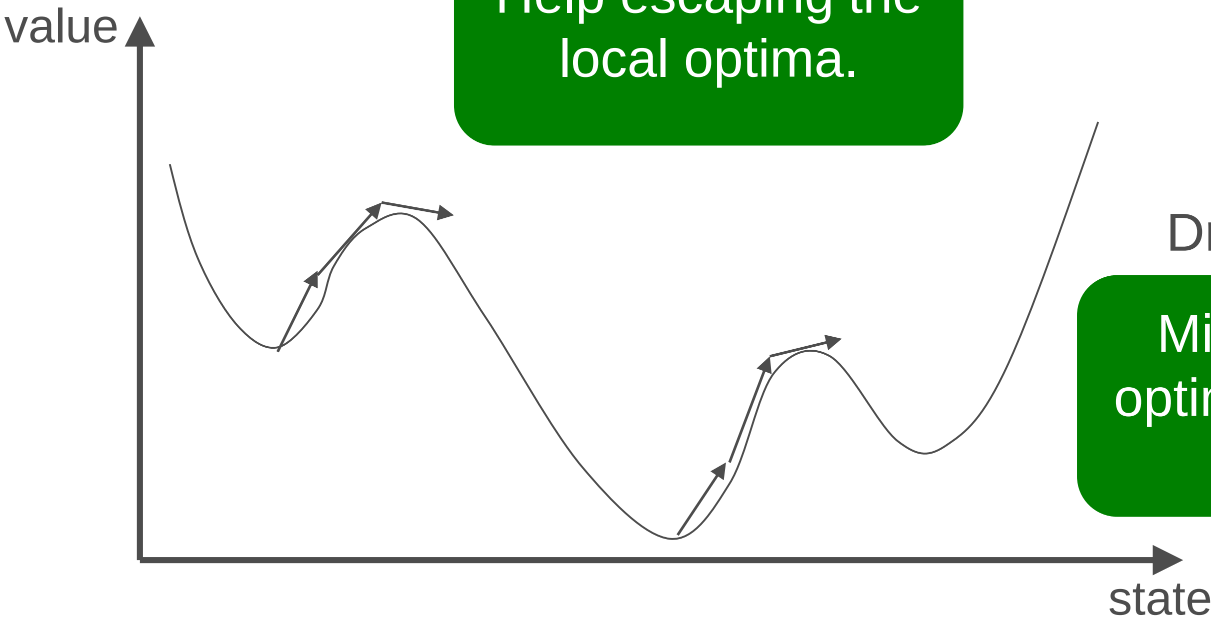
- Is a probabilistic search technique and imitates physical processes.
- Observation in nature:
  - At high temperatures, molecules move freely
  - At low temperatures, molecules “get stuck”
  - This is how crystals are formed in a thermodynamic process
- Other names:
  - Monte Carlo Annealing
  - Statistical Cooling
  - Probabilistic Hill Climbing
  - …

Idea of SA

- Solution candidates ~ states of (some quantity of) a metal
- Random initialization ~ heat the metal to a high temperature
- Next candidate ~ next state of the metal (more probabilistic if temperature is higher)
- Narrowing down the search ~ cooling down the metal

(Un-)Desired effects of SA

Help escaping the local optima.



Drawback:

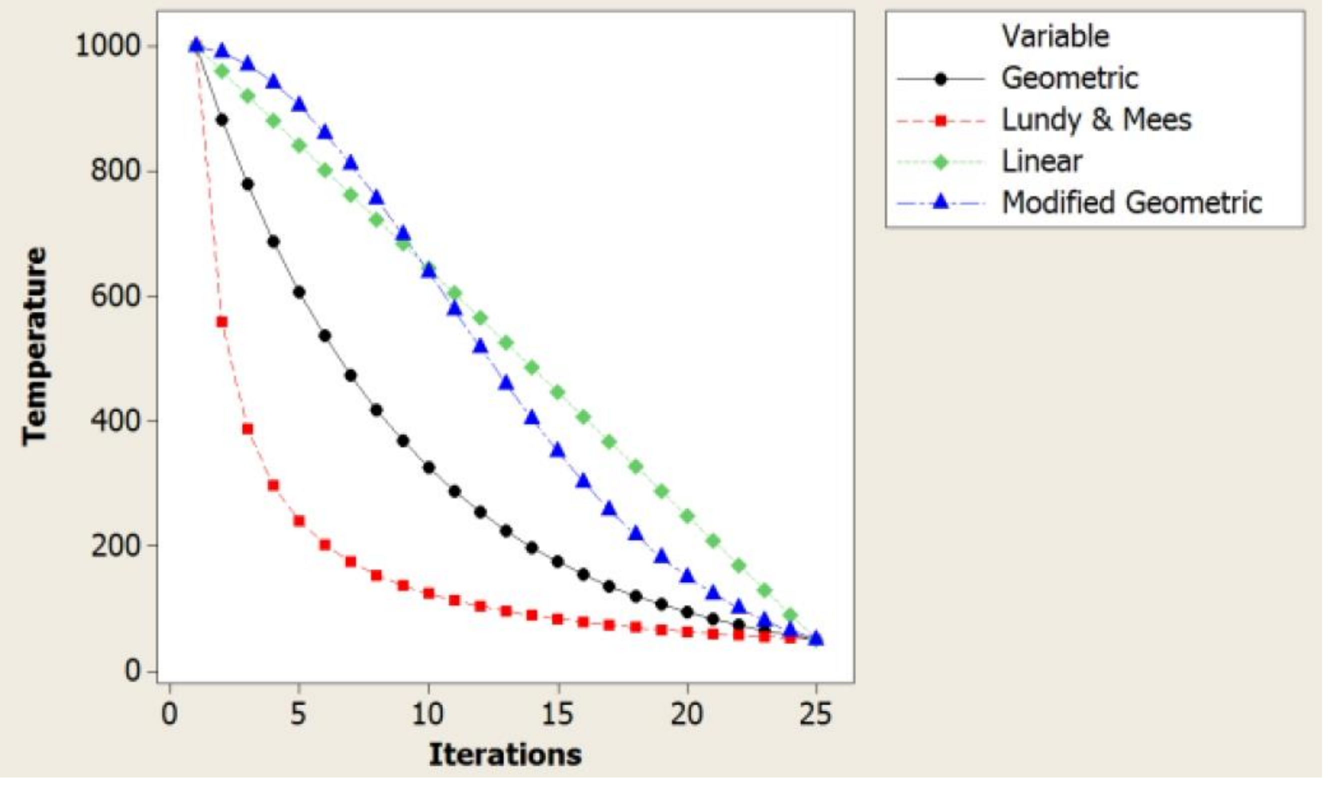
Might pass global optima after reaching them.

SA: Algorithm

1. Initialisation.
   Start with a random initial placement. Initialise a very high "temperature".
2. Movement.
   Perturb the placement through a defined move.
3. Score calculation.
   Calculate the change in the score due to the move made.
4. Selection.
   Depending on the change in score, accept or reject the move. The probability of acceptance depends on the current "temperature".
5. Update.
   Update the temperature value by lowering the temperature. If freezing point is reached, terminate; otherwise, go back to 2.

Parameters of SA

Main parameter of SA is the used cooling scheme.



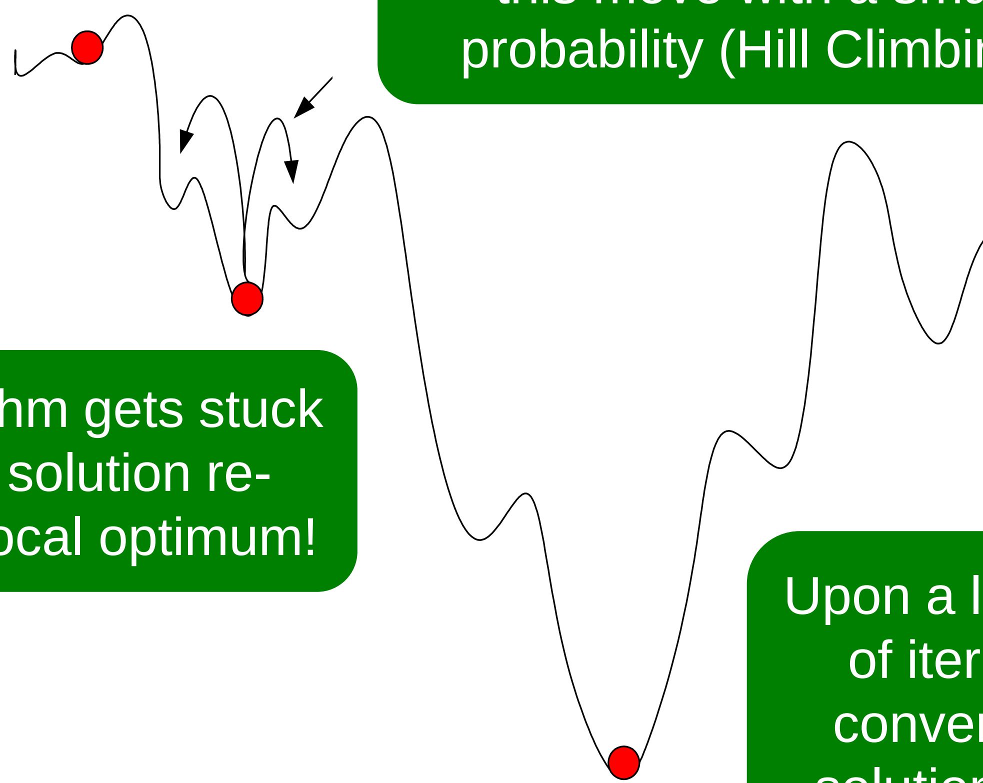
Example: Ball on the terrain

- Comparison between SA and greedy techniques (i.e. Hill Climbing)
- Process:
  - The ball is initially placed at a random position on the terrain.
  - From the current position, the ball should be fired such that it can only move one step left or right.
- What algorithm should we follow for the ball to finally settle at the lowest point on the terrain?

Example: Ball on the terrain (2)

Initial position of the ball

SA explores more. Chooses this move with a small probability (Hill Climbing)

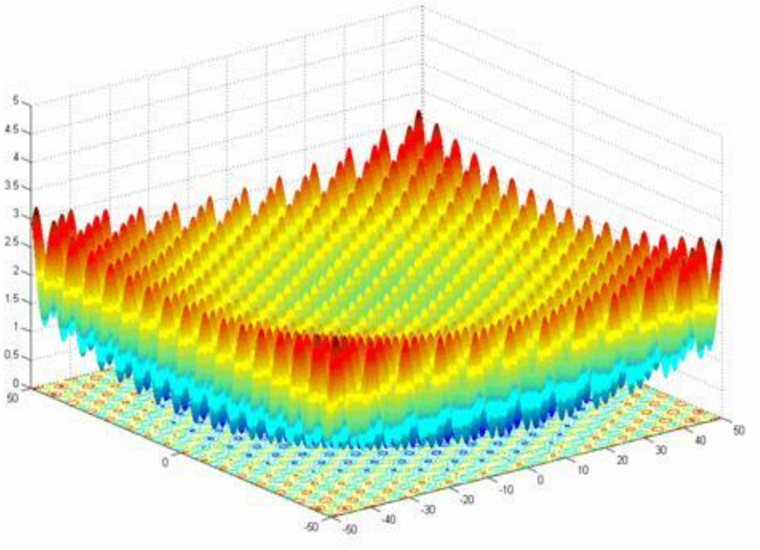


Greedy algorithm gets stuck here! This a solution re-presenting a local optimum!

Upon a large number of iterations, SA converges to this solution (optimum)

SA: Discussion

- SA guarantees convergence upon running a sufficiently large number of iterations.
- The configuration of its parameters is crucial for the success.
- The technique likely fails if the fitness landscape is highly multi-modal, e.g. the Griewark function:



ORGANIC COMPUTING

Tabu Search (TS)

- TS is very similar to SA but uses a deterministic acceptance/rejection criterion.
- Maintain a tabu list of solution changes:
  - A move made is entered at top of the tabu list
  - Fixed length (5-9) of the list
  - Neighbours are restricted to those solutions that do not require a tabu move.
- The tabu list
  - Rational: avoid returning to a local optimum
  - Disadvantage: tabu move could lead to a better schedule.
  - Fixed length:
too short → cycling ("stuck") or
too long → the search is too constrained!

Tabu Search: Algorithm

- Step 1:
  - Set $k = 1.$ Select an initial solution S_1 and set $S_0 = S_{1.}$
- Step 2:
  - Select a candidate solution S_c from $N(S_k)$.
  - If S_c on tabu list set $S_{k+1} = S_k$ and go to Step 3.
  - Set $S_k = S_c$
  - Enter S_k on tabu list.
  - Push all the other entries down (and delete the last one).
  - If $G(S_c) < G(S_0)$, set $S_0 = S_c$.
  - Go to Step 3
- Step 3:
  - Let $k=k+1$. If $k=N$ STOP; otherwise go to Step 2.

Drawbacks and limitations of SA and TS

- Limitations (of Simulated Annealing and Tabu Search)
  - Pursues one state configuration at the time.
  - Changes to configurations are typically local.
- Is there a more promising approach?
  - Assume we have two configurations with good values that are quite different.
  - We expect that the combination of the two individual configurations may lead to a configuration with higher value.
- Results of these considerations:

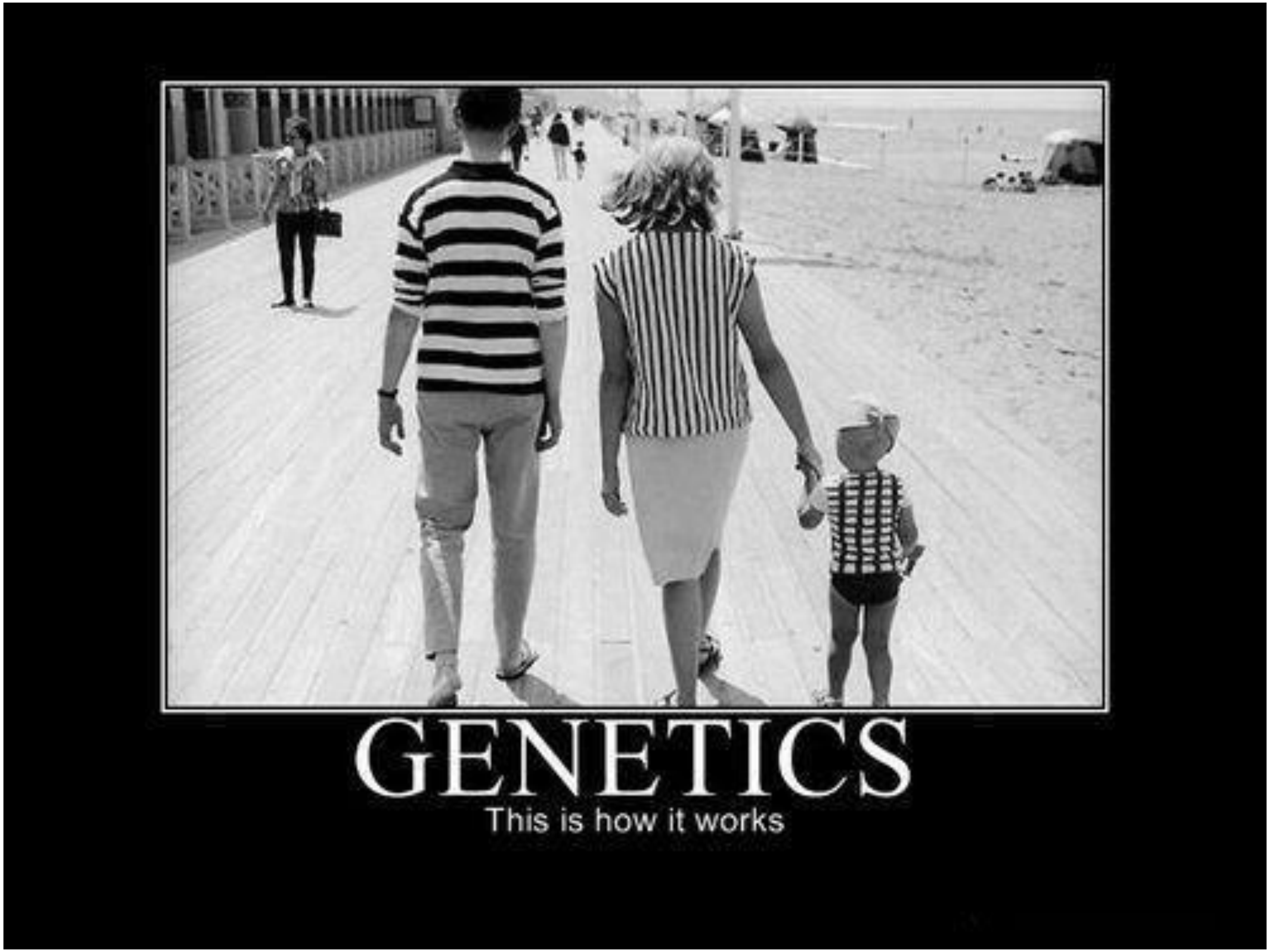
Genetic Algorithms!

Genetic / Evolutionary Algorithms

- Basic question:

What would Darwin do?

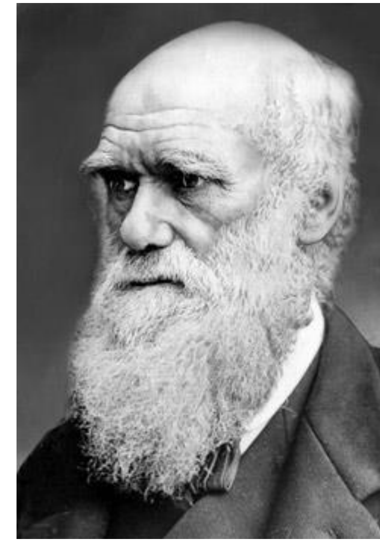
Nature-inspired optimisation heuristics (15)

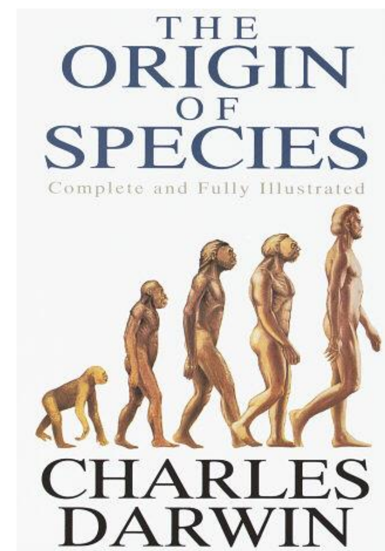


Nature-inspired optimisation heuristics (16)

Genetic / Evolutionary Algorithms

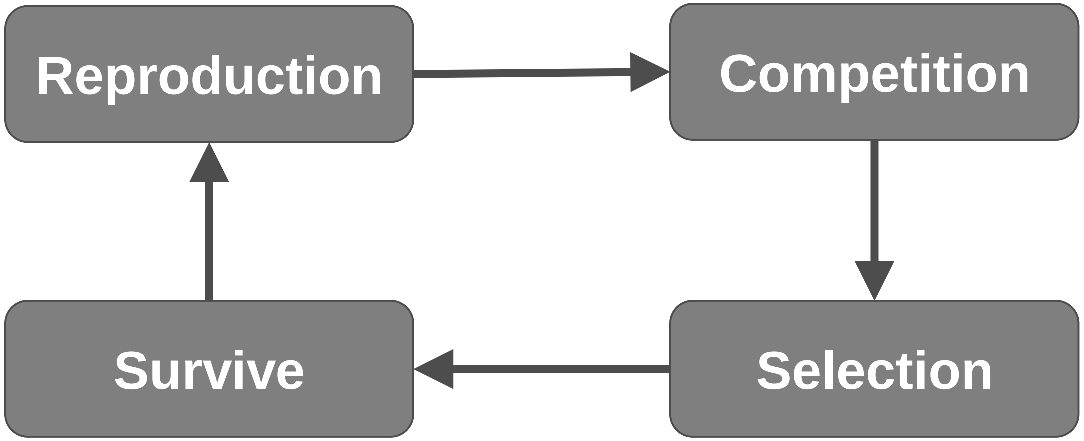
- Charles Robert Darwin
  - * 12.02.1809 (Shrewsbury); † 19.04.1882 (Downe)
  - English naturalist
- Scientific work:
  - Established that all species of life have descended over time from common ancestors.
  - Proposed the scientific theory that this branching pattern of evolution resulted from a process that he called natural selection.





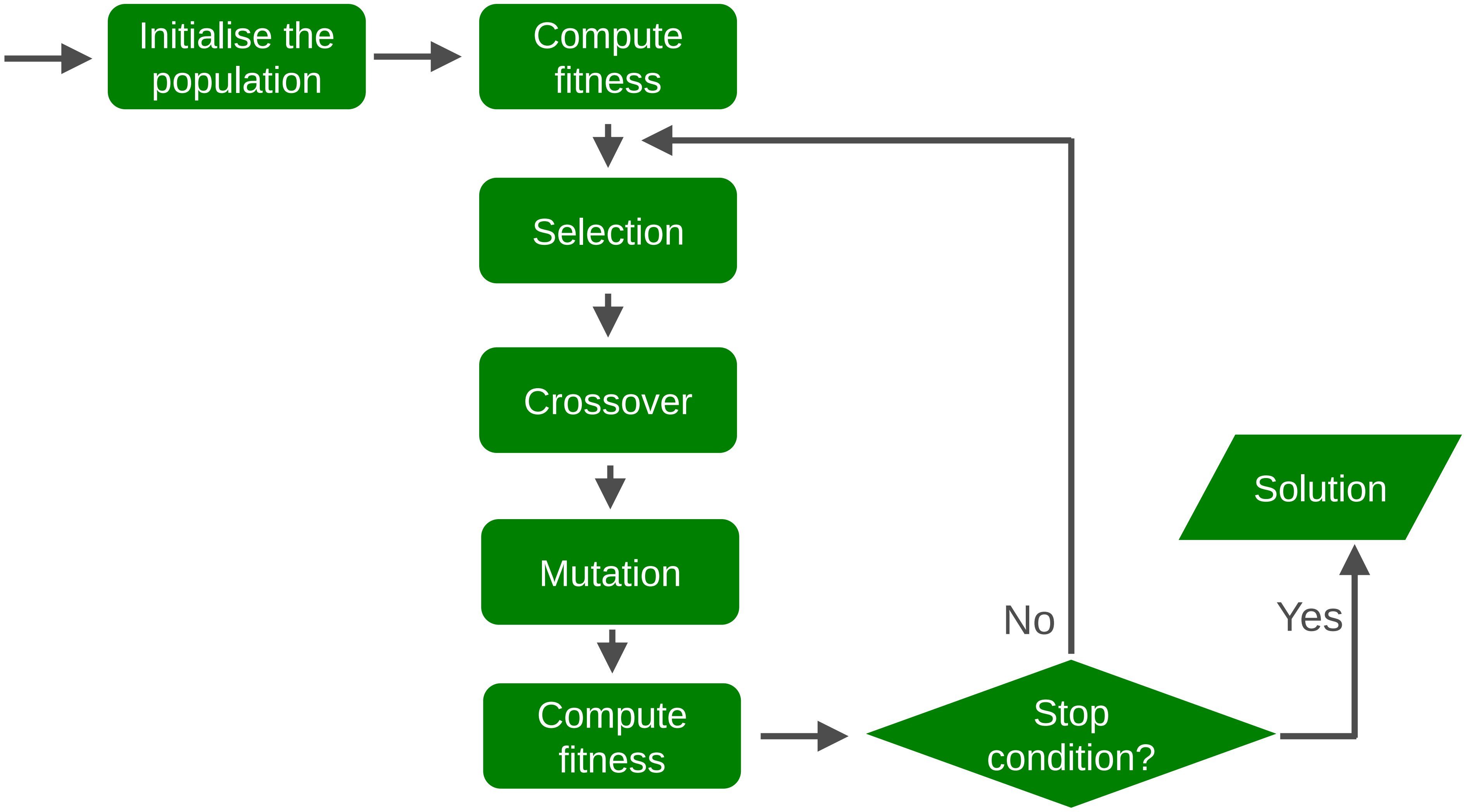
Genetic / Evolutionary Algorithms

- Natural selection (according to Darwin):
  - „Survival of the fittest“
  - Reproduction loop:



- Novel aspect in a technical sense: the population.
  - Not one element searches the space, but a large set of individuals is distributed over the space and searches it as result of genetic processes.

Nature-inspired optimisation heuristics (18)



Nature-inspired optimisation heuristics (19)

Genetic operator: 1-point crossover

- One position in the chromosomes is chosen randomly.
- Child 1 is head of the chromosome of parent 1 with tail of the chromosome of parent 2.
- Child 2 is head of 2 with tail of 1.

Randomly chosen position

Parents: **1010001110** **0011010010**

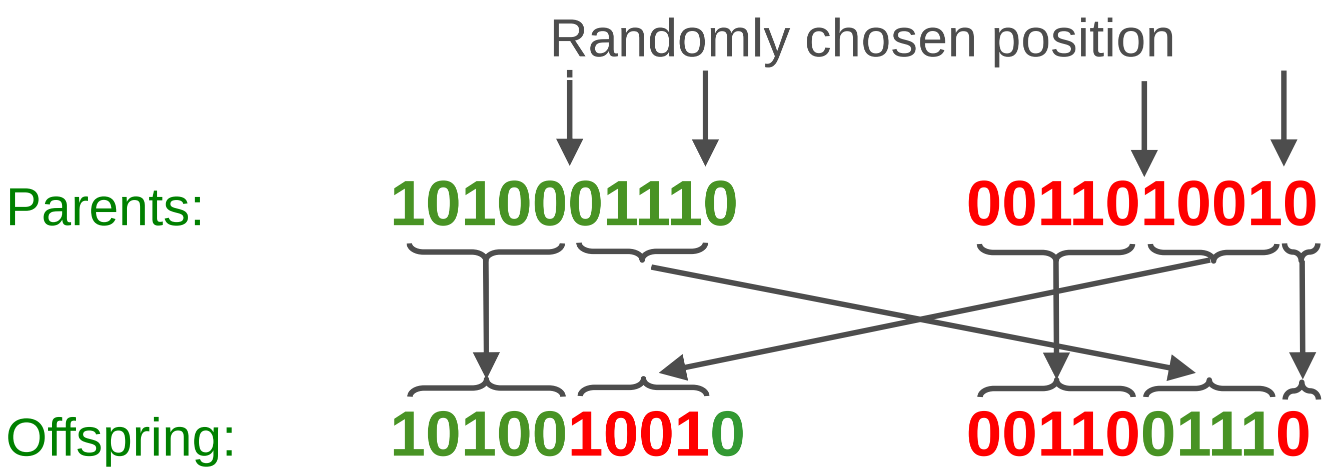
Offspring: **1010010010** **0011001110**

Nature-inspired optimisation heuristics (20)

ORGANIC
COMPUTING

Genetic operator: 2-point crossover

- Two positions in the chromosomes are chosen randomly.
- Avoids that genes at the head and genes at the tail of a chromosome are always split when recombined.



ORGANIC COMPUTING

Genetic operator: uniform crossover

- A random mask is generated.
- The mask determines which bits are copied from one parent and which from the other parent.
- The bit density in mask determines how much material is taken from the other parent (takeover parameter).
- Mask: ABAABABBAB

Parents:	**1010001110**	**0011010010**
Offspring:	**1010000010**	**0011011110**

Crossover strategy

- Is uniform crossover better than single-point crossover?
- Trade off between:
  - Exploration: introduction of new combinations of features
  - Exploitation: keep the good features in the existing solution

Genetic operator: mutation

- Generation of new offspring from single parents
- Alternatively: modify generated offspring

Parent: **1010001110**

↓

Offspring: **1011010010**

- Maintaining the diversity of the individuals
- Crossover can only explore combinations of the current gene pool.
- Mutation can “generate” new genes

Genetic operator: selection

- Selection strategies for survivors
  - Always keep the best one.
  - Elitist: deletion of the *k* worst.
  - Probability selection:
    - Same probability for each individual
    - Inverse probability to their fitness values
    - Hybrid approaches

ORGANIC COMPUTING

Genetic operator: selection

- Selection strategies for parents
  - Always take the best ones.
  - Uniformly randomised selection
  - Probability selection:
    - Same probability for each individual
    - Inverse probability to their fitness values
  - Tournament selection (multi objectives)
    - Build a small comparison set.
    - Randomly select a pair:
      → The higher ranked one beats the lower one.
      → The non-dominated one beats the dominated one.
  - Niche count: the number of points in the population within a certain distance → the higher the niche count, the lower the rank.
    - Strategy to maintain population diversity!

Genetic operator: selection

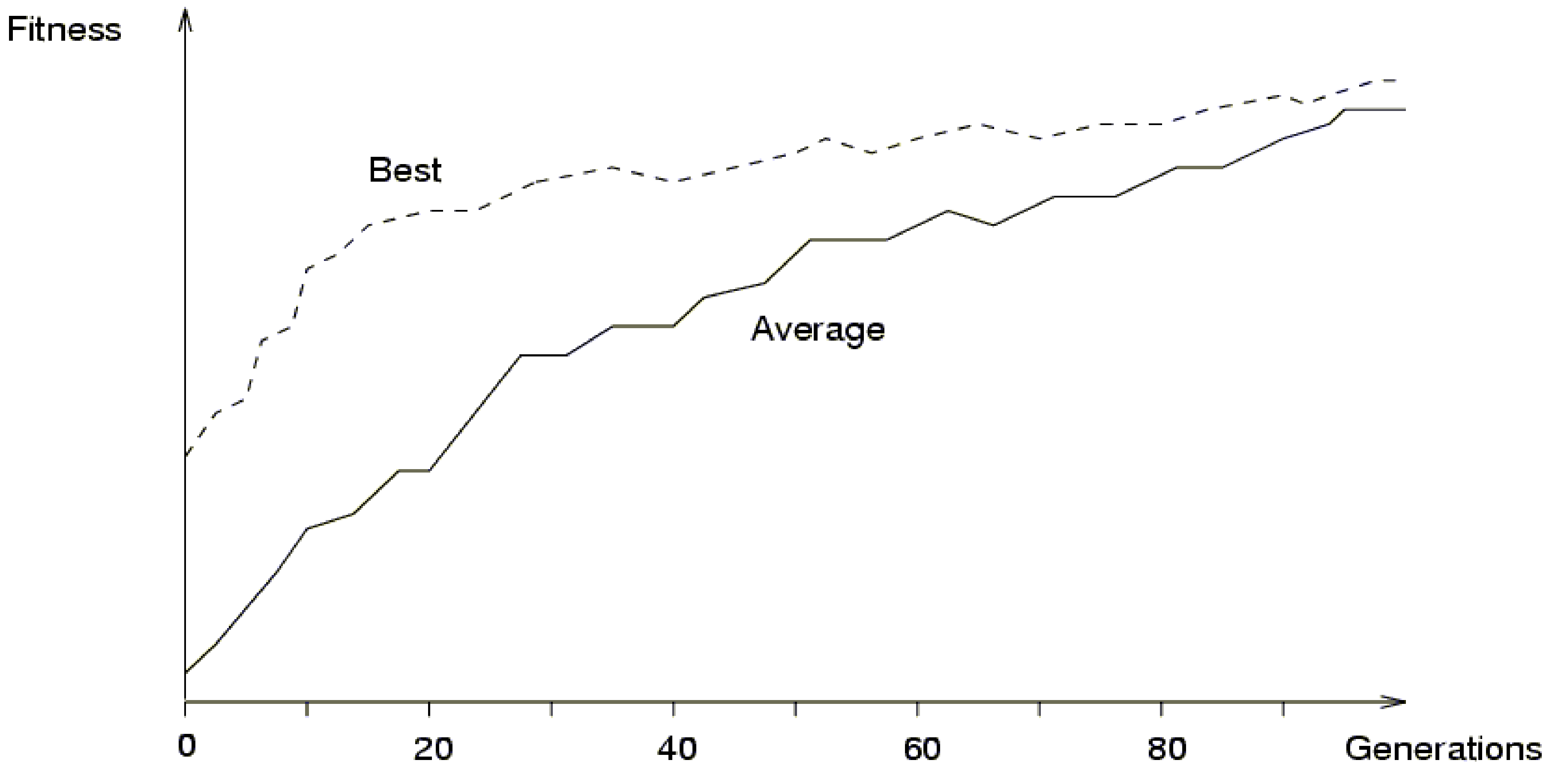
- Discussion of impact:
  - Too strong fitness selection bias can lead to sub-optimal solutions.
  - Too little fitness bias selection results in an unfocused and meandering search behaviour.

- Strategies:
  - Increase the probability to be selected as parent (proportional to the fitness value):
    Roulette wheel approach!
  - Avoiding problems with the fitness function:
    Tournament approach!

Parameters of Evolutionary / Genetic Algorithms

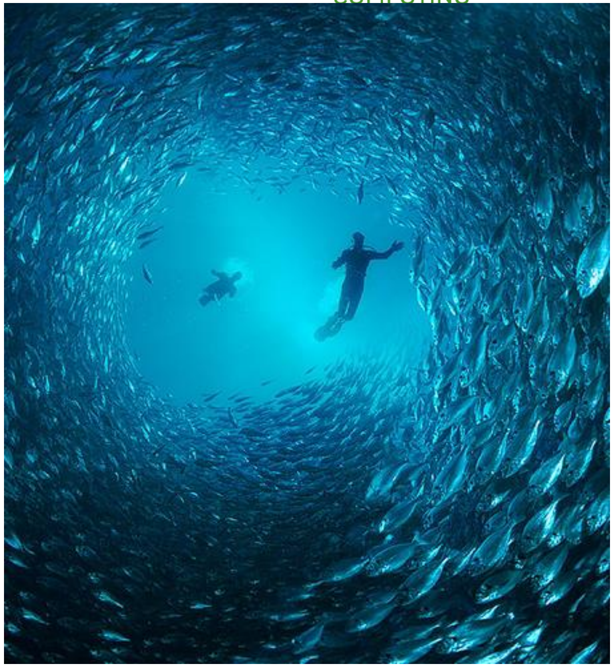
- Variable control parameters, i.e.:
  - Population size
  - Mutation / crossover / selection probabilities
- Impact:
  - Problem specific
  - Increase population size
    → Increase diversity and computation time for each generation
  - Increase crossover probability
    → Increase the opportunity for recombination but also disruption of good combinations
  - Increase mutation probability
    → Closer to random search
    → Help to introduce new genes or reintroduce the lost ones

Evolutionary / Genetic Algorithms: Convergence



Swarm-based optimisation

- Origins: How can birds or fish exhibit such a coordinated collective behaviour?

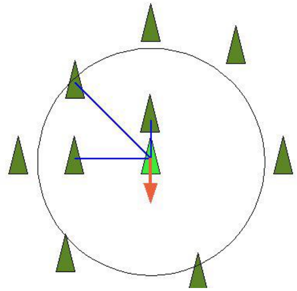


Pictures for swarm-based optimisation are taken from Marco A. Montes de Oca (IRIDIA-CoDE, Université Libre de Bruxelles, Belgium)

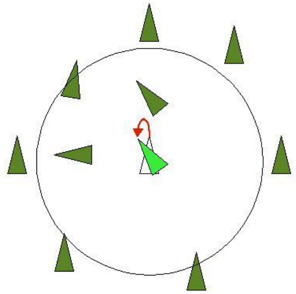
Nature-inspired optimisation heuristics (34)

Basic agent behaviour in swarms modelled by three simple rules:

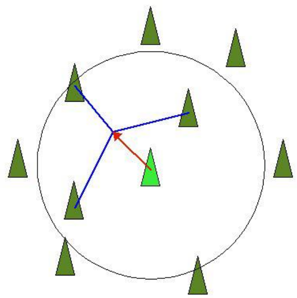
- Separation: Each agent tries to move away from its neighbours if they are too close.



- Alignment: Each agent st… …e average heading of … .



- Cohesio… …ach agent tries to go towards the average position of its neighbours.

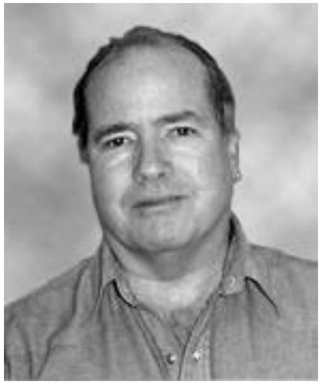


This simple model is enough to simulate the complex behaviour in natural swarms!

Craig W. Reynolds: "Flocks, herds, and schools: A distributed behavioural model". ACM Computer Graphics, 21(4):25–34, 1987.

Particle Swarm Optimisation (PSO)

- Application of swarm behaviour to optimisation problems:
  - Initial concept by James Kennedy and Russell Eberhart in 1995
  - Applies the concept of social interaction to problem solving.
  - Has been applied successfully to a wide variety of search and optimisation problems.



Kennedy

Eberhart

- In PSO, in a swarm the n contained individuals communicate either directly or indirectly with one another about their search directions (gradients).

James Kennedy and Russell Eberhart: “Particle swarm optimization”. In Proceedings of IEEE International Conference on Neural Networks, pages 1942–1948, Piscataway, NJ, USA, 1995. IEEE Press.

Nature-inspired optimisation heuristics (36)

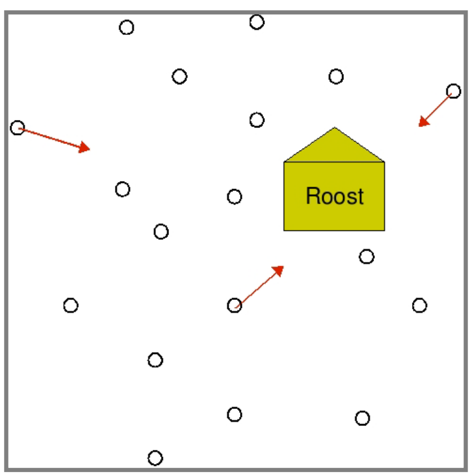
Kennedy and Eberhart introduced a “roost“ (in German „Schlafplatz“):

a) Each agent is attracted towards the location of this roost.

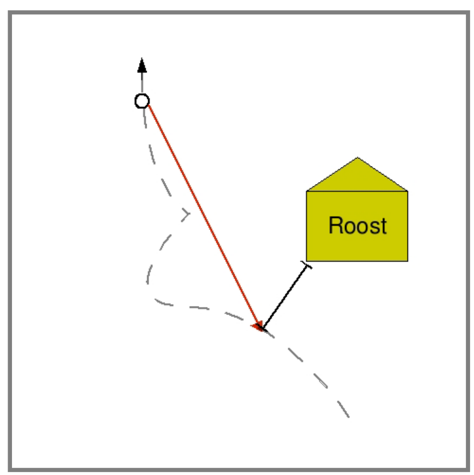
b) Each agent remembers where it was closest to the roost.

c) Each agent shares its information about the closest location to the roost with its neighbours.

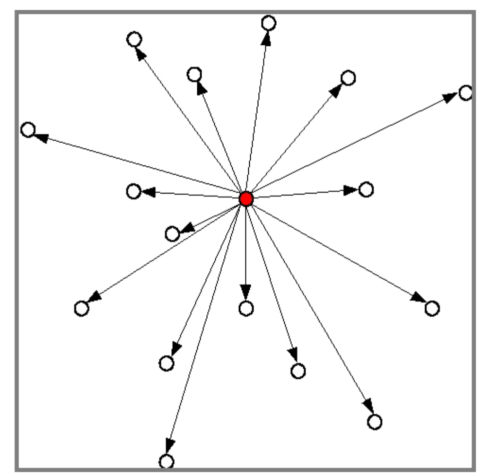
a)



b)



c)



Nature-inspired optimisation heuristics (37)

ORGANIC COMPUTING

PSO: Anatomy of a particle - a particle (individual) is composed of:

- Three vectors:
  - The x-vector records the current position (location) of the particle in the search space,
  - The p-vector records the location of the best solution found so far by the particle, and
  - The v-vector contains a gradient (direction) for which particle will travel in if undisturbed.
- Two fitness values:
  - The x-fitness records the fitness of the x-vector, and
  - The p-fitness records the fitness of the p-vector.

$$\mathbf{I_k}$$

$$X = <x_{k0}, x_{k1}, \ldots, x_{kn-1}>$$

$$P = <p_{k0}, p_{k1}, \ldots, p_{kn-1}>$$

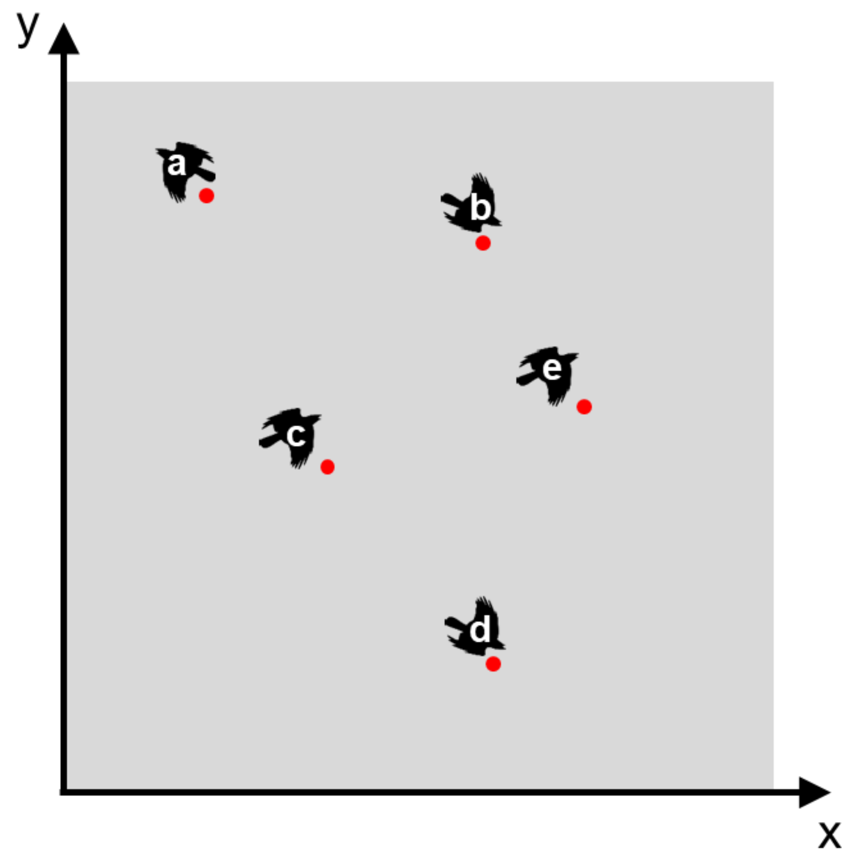
$$V = <v_{k0}, v_{k1}, \ldots, v_{kn-1}>$$

x_fitness = ?

p_fitness = ?

PSO: Algorithm

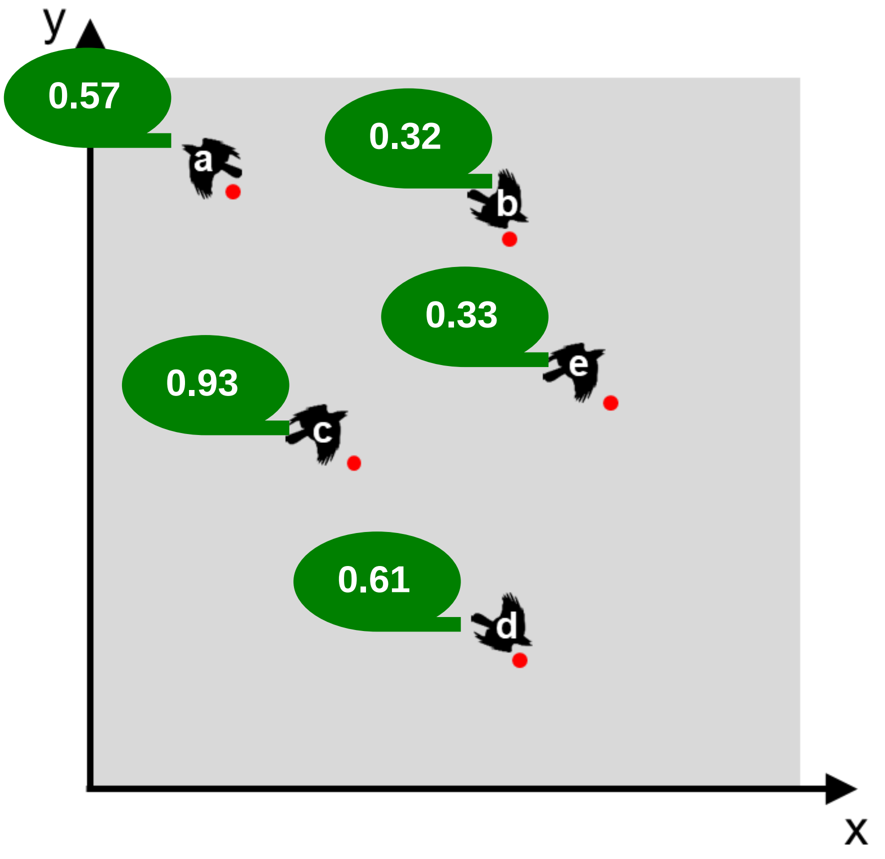
- Step 1: Create a population of agents (called particles) uniformly distributed over X.



ORGANIC COMPUTING

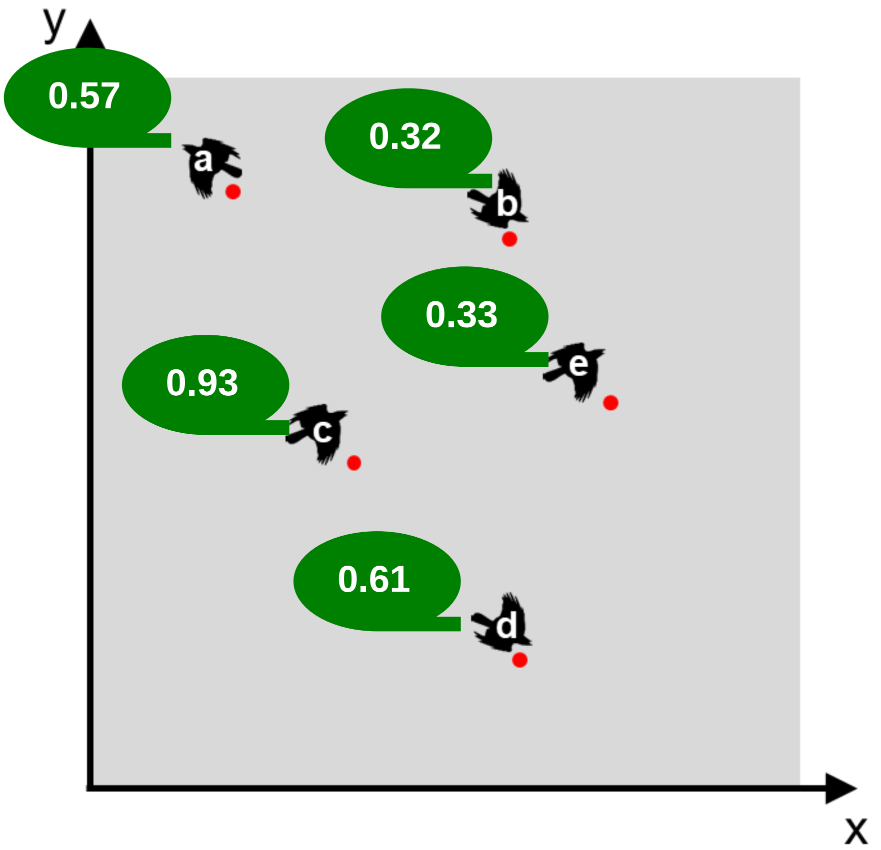
PSO: Algorithm

- Step 2: Evaluate each particle‘s position according to the objective function.



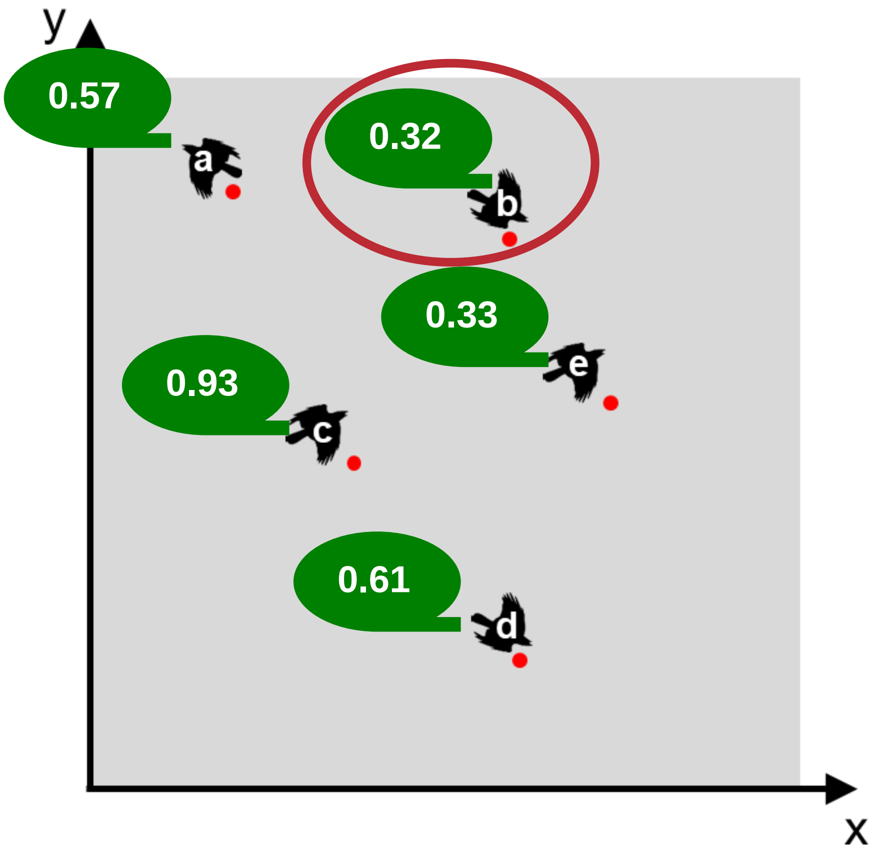
PSO: Algorithm

- Step 3: If a particle‘s current position is better than its previous best one → update it.
  - a: position xa/ya
    old: 1.0; new: 0.57
  - b: position xb/yb
    old: 1.0; new: 0.32
  - c: position xc/yc
    old: 1.0; new: 0.93
  - d: position xd/yd
    old: 1.0; new: 0.61
  - e: position xe/ye
    old: 1.0; new: 0.33



PSO: Algorithm

- Step 4: Determine the currently best particle.



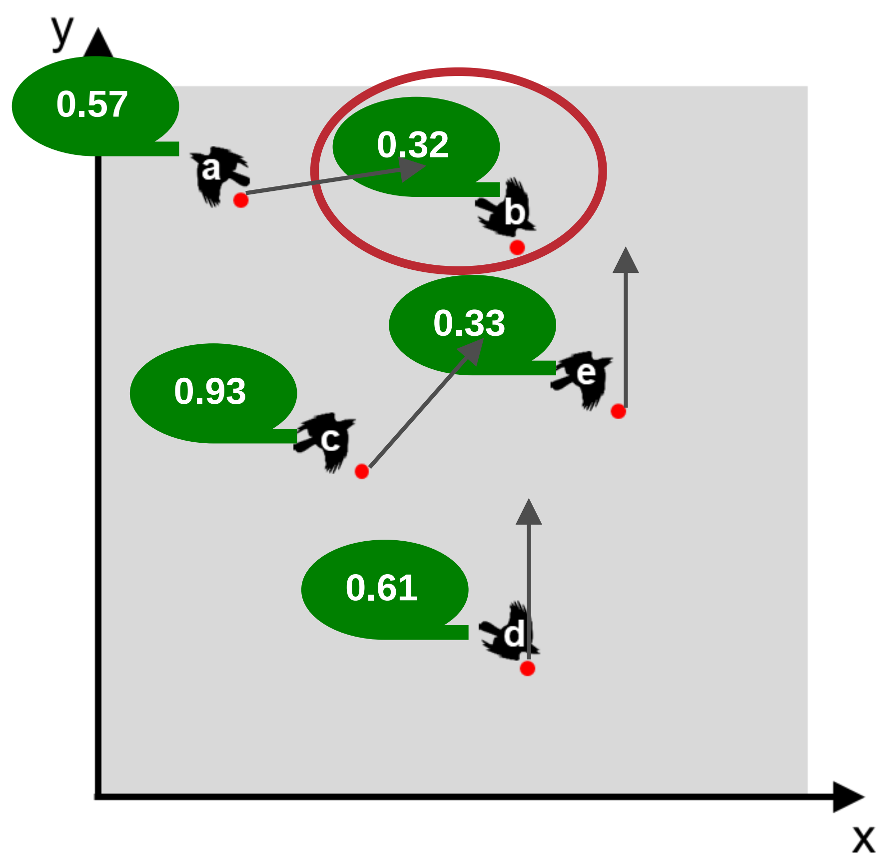
ORGANIC COMPUTING

PSO: Algorithm

- Step 5: Update the particles' velocities according to the PSO update formula

$$v_i^{t+1} = v_i^t + \varphi_1 U_1^t (pb_i^t - x_i^t) + \varphi_2 U_2^t (gb_i^t - x_i^t)$$

→ We can simplify the presentation of the formula for a better understanding.



What do all these variables mean?

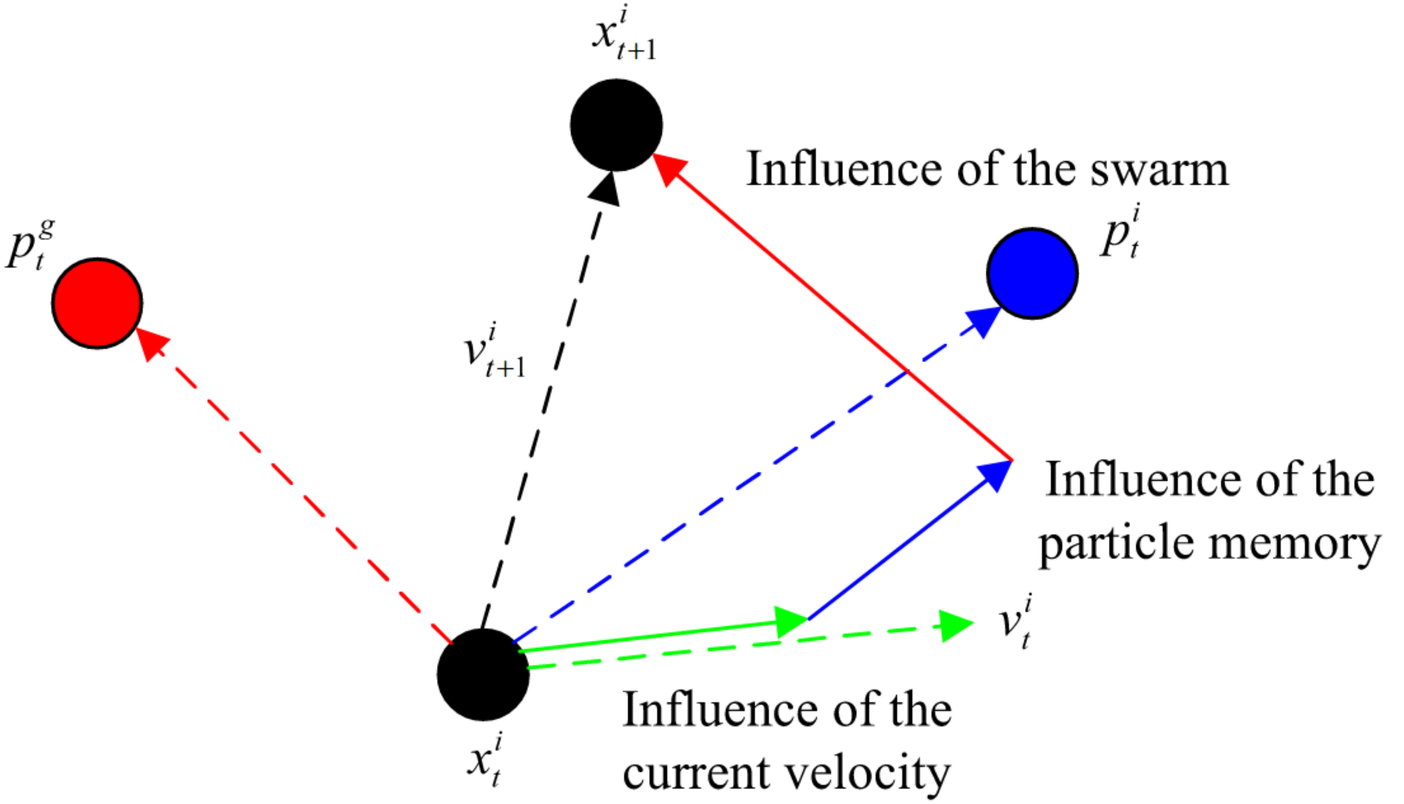
- Actually, we must adjust the v-vector before adding it to the x-vector as follows:

$$v_{id} = v_{id} + \varphi_1 \cdot \mathrm{rand}() \cdot (p_{id} - x_{id}) + \varphi_2 \cdot \mathrm{rand}() \cdot \left(p_{gd} - x_{id}\right)$$

$$x_{id} = x_{id} + v_{id}$$

- Where
  - i is the particle
  - φ_1 and φ_2 are learning rates governing the cognition and social components
  - g represents the index of the particle with the best p-fitness
  - d is the d-th dimension

PSO update intuition

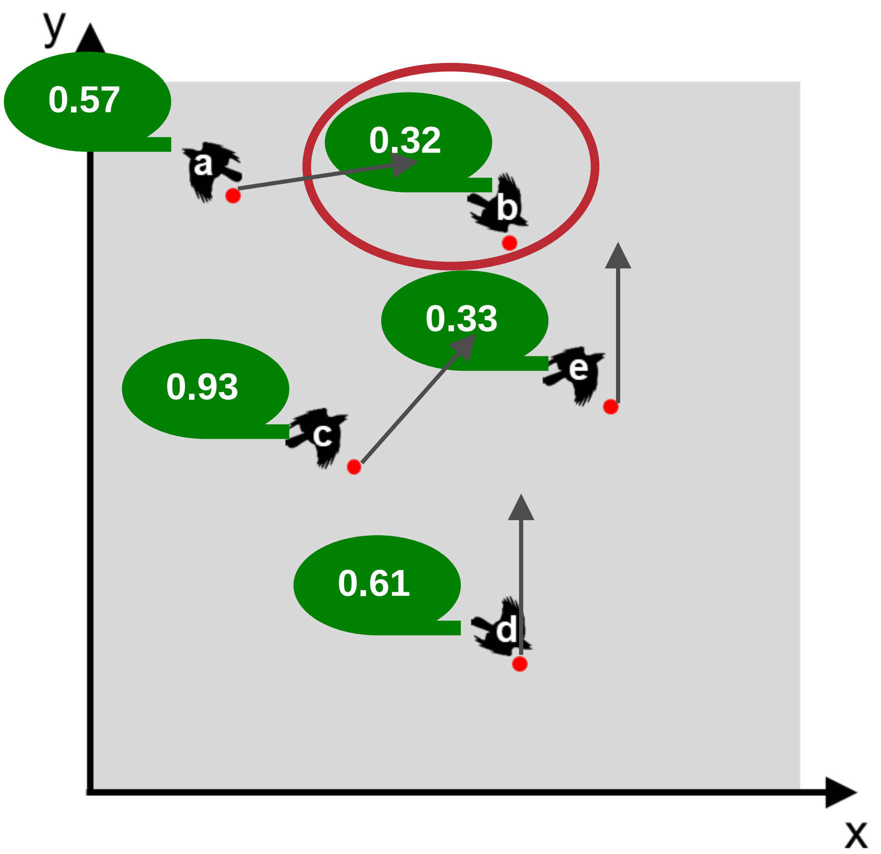


Wang, D., Tan, D. & Liu, L. Soft Comput (2018) 22: 387.
https://doi.org/10.1007/s00500-016-2474-6

PSO: Algorithm

- Step 6: Move the particles to their new positions

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

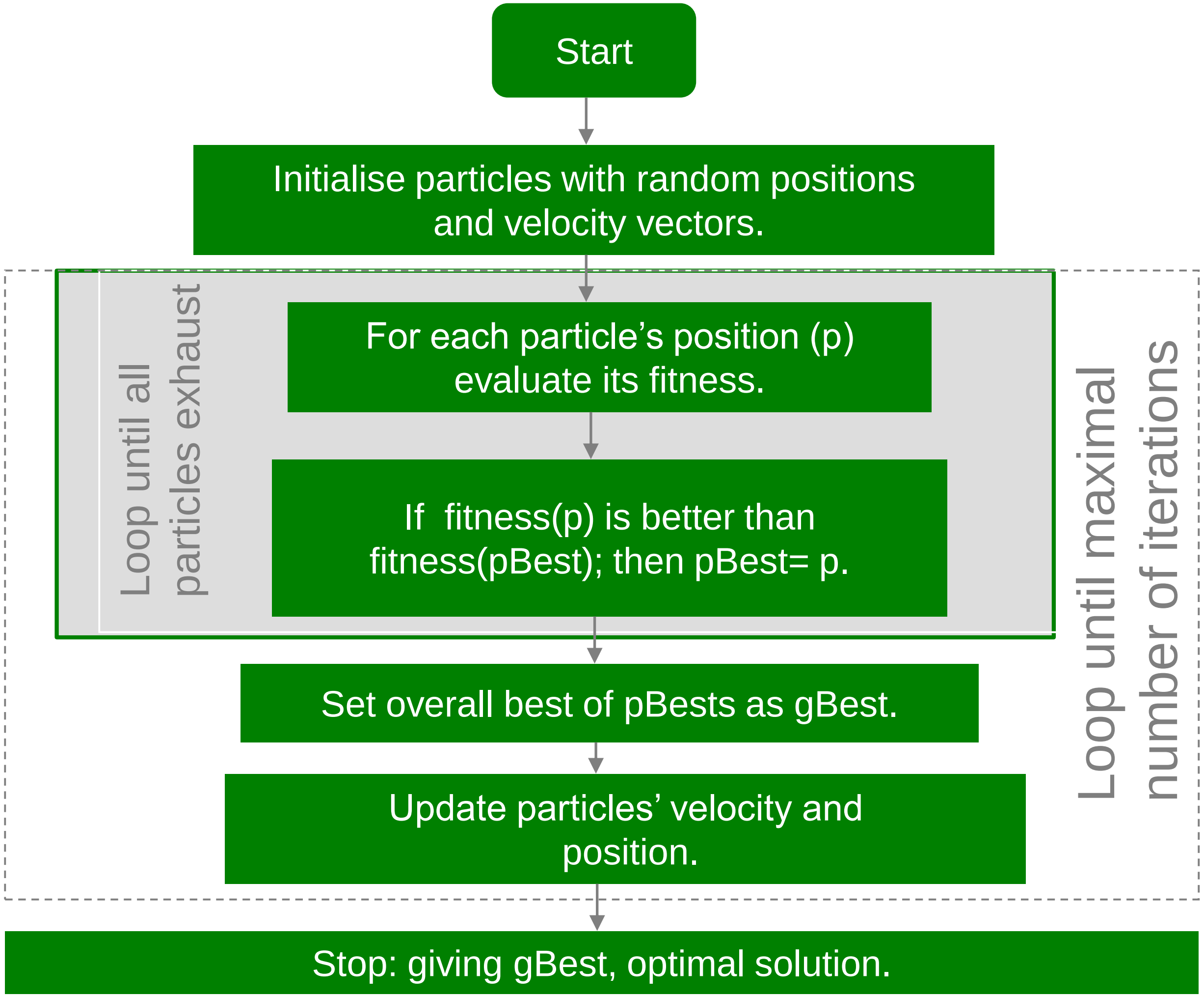


ORGANIC COMPUTING

PSO: Algorithm

- Step 7: Go to step 2 until the stop criteria are satisfied

- Algorithm summary:

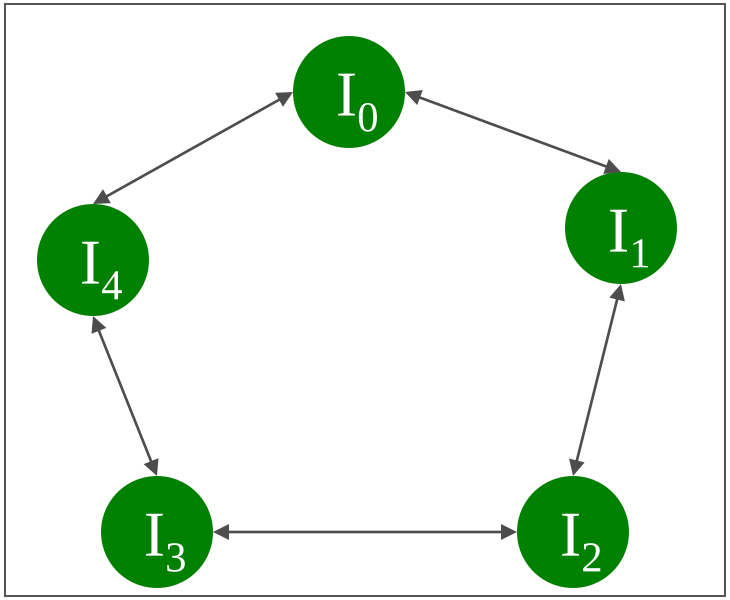


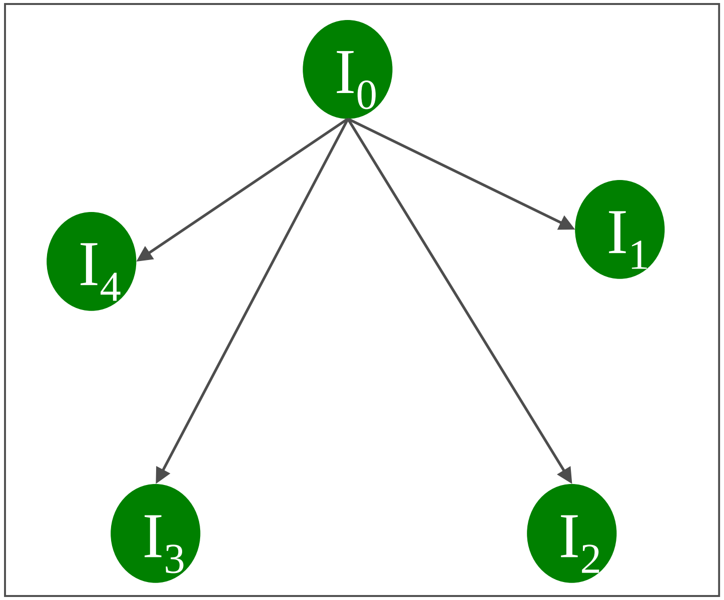
PSO: Algorithm summary

- In PSO, particles never die!
- Particles can be seen as simple agents that fly through the search space and record (and possibly communicate) the best solution that they have discovered.
- Most important question in this context: “How does a particle move from one location in the search space to another?”
- This is done by simply adding the v-vector to the x-vector to get another x-vector ($X_i = X_i + V_i$).
- After computing the new Xi, each particle immediately evaluates its new location. If the x-fitness is better than the p-fitness, then $P_i = X_i$ and p-fitness = x-fitness.

Parameters of PSO

- Swarm topology
- Two basic topologies can be found in the literature:
  - Ring topology (neighbourhood of 3)
  - Star topology (global neighbourhood)





ORGANIC COMPUTING

Parameters of PSO

- Velocity vectors
  - Randomly generated at the beginning
  - Within the range [-Vmax, Vmax] where Vmax is the maximum value that can be assigned to any v_{id}.
  - Continuously updated during the optimisation process

- Impact
  - When using PSO, the magnitude of the velocities might become very large.
  - The performance can suffer if Vmax is set inappropriately.
  - Two methods were developed for controlling the growth of velocities:
    → A dynamically adjusted inertia factor, and
    a constriction coefficient

Parameters of PSO

- The intertia factor (dt. Trägheit)
  - When the inertia factor is used, the equation for updating velocities is changed to:

$$v_{id} = w \cdot v_{id} + \varphi_1 \cdot \mathrm{rand}() \cdot (p_{id} - x_{id}) + \varphi_2 \cdot \mathrm{rand}() \cdot \left(p_{gd} - x_{id}\right)$$

  - Where w is initialised to 1.0 and gradually reduced over time (measured by cycles through the algorithm).

- Swarm and neighbourhood size
  - Trade-off between solution quality and cost (in terms of function evaluations)
  - Global neighbourhoods seem to be better in terms of computational costs. The performance is similar to the ring topology (or neighbourhoods greater than 3).

ORGANIC COMPUTING

Parameters of PSO

- Update strategies: synchronous vs. asynchronous
  - Asynchronous update allows for newly discovered solutions to be used more quickly.
- Futher important parameters:
  - Controlling velocities (determining the best values for Vmax),
  - size of the swarm,
  - neighbourhood size and topology,
  - robust settings for $\varphi 1$ and $\varphi 2$.
- Current research:
  - Trying to find and establish an Off-The-Shelf PSO.

Carlisle, A. and Dozier, G. (2001). “An Off-The-Shelf PSO”, *Proceedings of the 2001 Workshop on Particle Swarm Optimization*, pp. 1-6, Indianapolis, IN. (http://antho.huntingdon.edu/publications/Off-The-Shelf_PSO.pdf)

ORGANIC COMPUTING

Types of swarms

- Given:

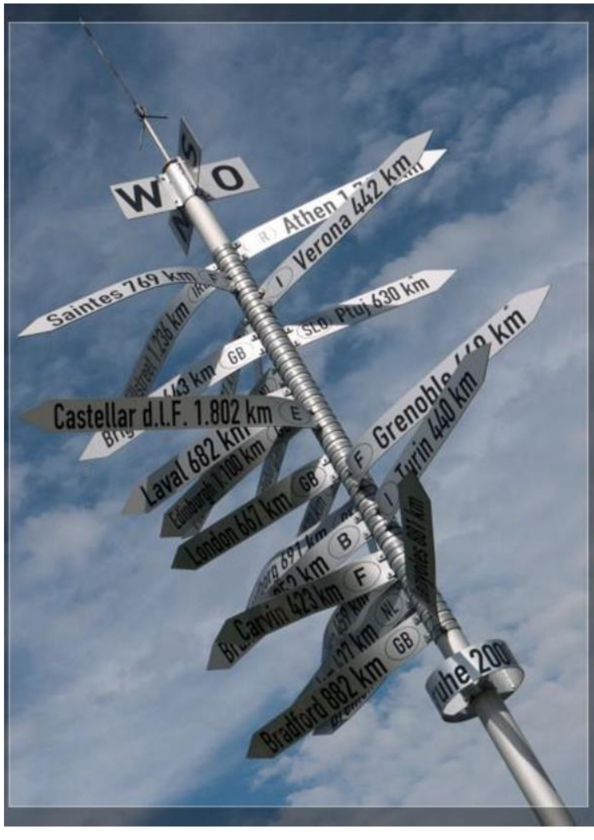
$$v_{id} = w \cdot v_{id} + \varphi_1 \cdot \mathrm{rand}() \cdot (p_{id} - x_{id}) \\ + \varphi_2 \cdot \mathrm{rand}() \cdot \left(p_{gd} - x_{id}\right)$$

- Full Model (φ_1, $\varphi_2 > 0$)
- Cognition Only ($\varphi_1 > 0$ and $\varphi_2 = 0$),
- Social Only ($\varphi_1 = 0$ and $\varphi_2 > 0$)
- Selfless ($\varphi_1 = 0$, $\varphi_2 > 0$, and $g \neq i$)

Details: [Kennedy, J. (1997), “The Particle Swarm: Social Adaptation of Knowledge”, Proceedings of the 1997 International Conference on Evolutionary Computation, pp. 303-308, IEEE Press.]

Agenda

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings

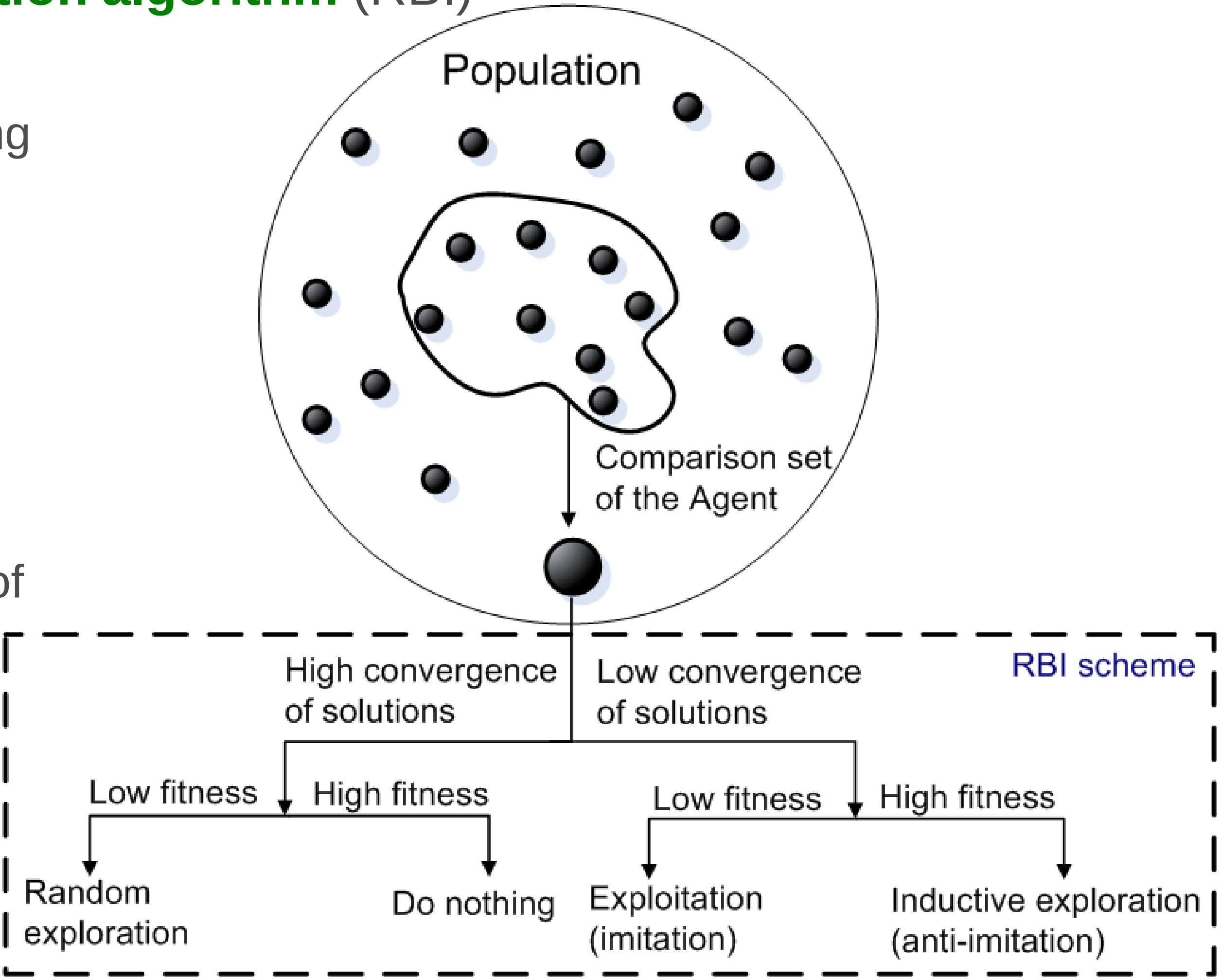


The Role-based Imitation algorithm (RBI)

- Clear distinction of exploring and exploiting agents (individuals) according to:

  a) the current degree of convergence of a (sub-)population and

  b) the relative quality of the agent's solution.



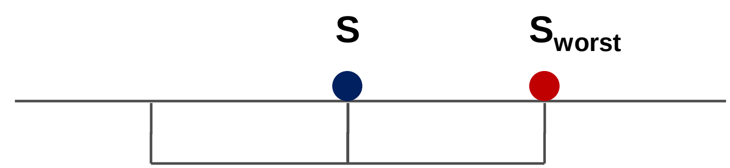
Cakar, Emre, Sven Tomforde, and Christian Müller-Schloer. "A role-based imitation algorithm for the optimisation in dynamic fitness landscapes." Swarm Intelligence (SIS), 2011 IEEE Symposium on. IEEE, 2011.

The Role-based Imitation Algorithm (2)

a) Imitation

$$S = S + |S_{best} - S| * U[0,1)$$

b) Anti-imitation

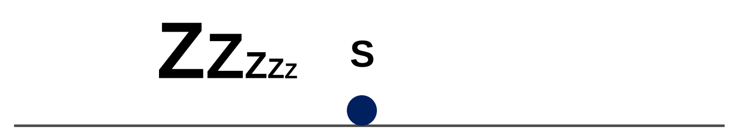


$$S = S - |S_{worst} - S| * U[0,1)$$

c) Random exploration

$$S = S * (2 - U[0,4])$$

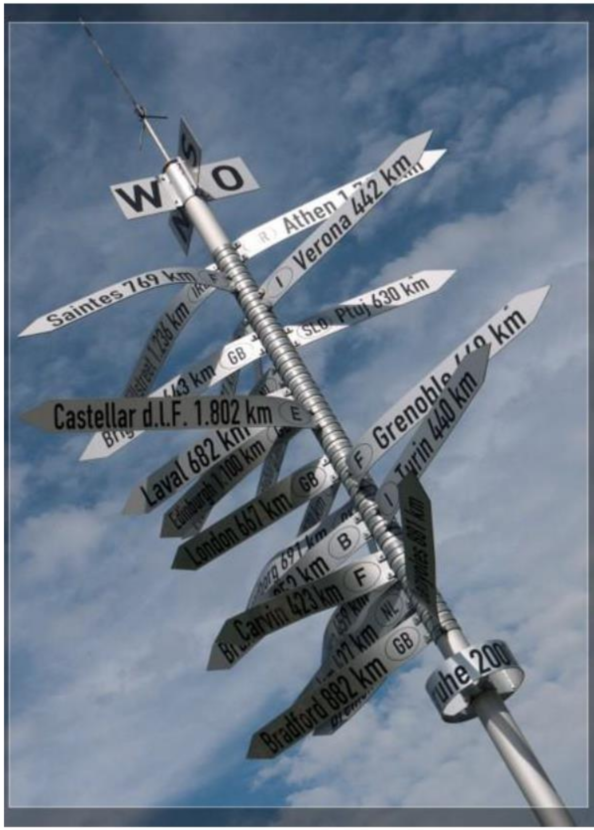
d) Do nothing



Agenda

ORGANIC COMPUTING

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings

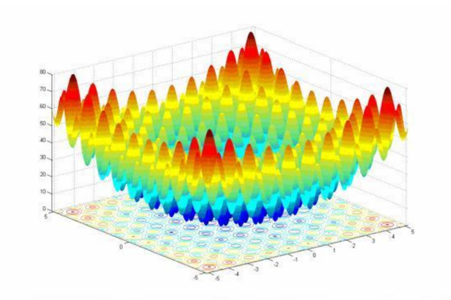


Comparison of optimisation heuristics

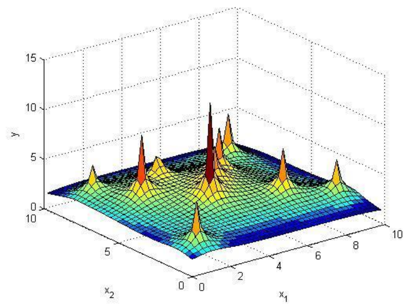
- Basic question: Which technique shall we use in OC systems?
- Comparison in:
  - static environments
  - dynamic environments
- Candidates:
  - Role-based Imitation Algorithm (RBI)
  - Differential Evolution (DE)
  - Genetic Algorithm (GA)
  - Particle Swarm Optimisation (PSO)
  - Simulated Annealing (SA)

A brief evaluation in OC systems (2)

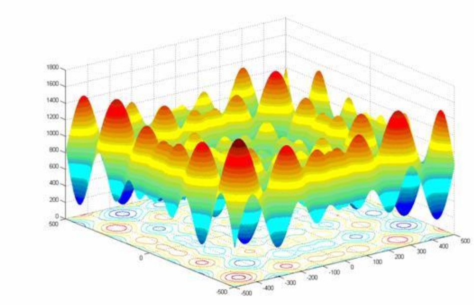
Reminder: static fitness landscapes



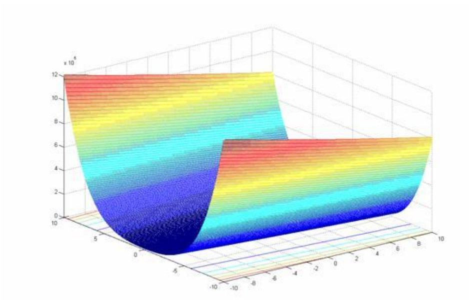
Rastrigin function [F9]



Shekel function [F14]



Schwefel function [F2]



Rosenbrock function [F5]

- Benchmark: "*A comparative Study of Differential Evolution, Particle Swarm Optimisation and Evolutionary Algorithms on Numerical Benchmark Problems*", Vesterstrom et al., CEC 2004

A brief evaluation in OC systems (3)

Evaluation setup

- Functions from Vesterstrom et al.:
  - F1 - F13 are high-dimensional functions each with 30 dimensions.
  - F14 - F21 are low-dimensional functions with 2 or 4 dimensions.
- Criteria:
  - Quality of the solution
    (after 500,000 calls of the evaluation function)
  - Convergence speed

- Best results are highlighted in grey!

ORGANIC COMPUTING

Results of the evaluation

- Low-dimensional functions

	RBI	DE	PSO	GA	SA	Optimum
F14	9.9800384 e-01	9.9800384 e-01	9.9800384 e-01	1.6892421 e+00	9.9800384 e-01	9.9800384 e-01
F15	3.0748593 e-04	4.6010056 e-04	3.0748599 e-04	3.0814498 e-04	6.1205233 e-04	3.0748593 e-04
F16	-1.0316285 e+00	-1.0316285 e+00	-1.0316285 e+00	-1.0316285 e+00	-1.0316285 e+00	-1.0316285 e+00
F17	3.9788735 e-01	3.9788735 e-01	3.9788735 e-01	3.9788735 e-01	3.9788735 e-01	3.9788735 e-01
F18	3.0 e+00	3.0 e+00	3.0 e+00	3.0 e+00	3.0 e+00	3.0 e+00
F19	-9.486763 e+00	-10.1532 e+00	-9.735774 e+00	-8.410121 e+00	-9.984785 e+00	-10.1532 e+00
F20	-10.402941 e+00	-10.402941 e+00	-10.402941 e+00	-10.402941 e+00	-10.049961 e+00	-10.402941 e+00
F21	-10.536409 e+00	-10.536409 e+00	-10.536409 e+00	-9.772439 e+00	-10.536409 e+00	-10.536409 e+00

- All algorithms produce similar results for the low-dimensional functions so that they are all on the same level regarding the quality of solutions

A brief evaluation in OC systems (5)

Results of the evaluation

- High-dimensional functions
- The functions from F1 to F7 are unimodal and the functions from F8 to F13 are multimodal functions

	RBI	DE	PSO	GA	SA	Optimum
F1	0.000000 e+00	0.000000 e+00	0.000000 e+00	1.6828756 e-06	2.390598 e-10	0.000000 e+00
F2	0.000000 e+00	8.142916 e-37	0.000000 e+00	5.466097 e-03	5.140537 e-05	0.000000 e+00
F3	0.000000 e+00	0.000000 e+00	0.000000 e+00	6.52916 e-04	1.29546415 e-05	0.000000 e+00
F4	3.223638 e-15	3.7110183 e-08	3.7296683 e-11	1.9280297 e+00	9.731591 e-01	0.000000 e+00
F5	2.620019 e+01	4.03334 e-22	2.2833145 e+01	3.3183784 e+01	6.4433184 e+00	0.000000 e+00
F6	0.000000 e+00	0.000000 e+00	0.000000 e+00	0.000000 e+00	0.000000 e+00	0.000000 e+00
F7	3.419498 e-04	3.215471 e-03	1.6625043 e-03	8.144887 e-04	3.7914343 e-02	0.000000 e+00
F8	-1.2569486 e+04	-1.2569486 e+04	-1.0557845 e+04	-7.4431504 e+03	-1.2558608 e+04	-1.2569486 e+04
F9	0.000000 e+00	0.000000 e+00	2.527239 e+01	3.0669328 e-04	1.9462983 e-01	0.000000 e+00
F10	8.970602 e-15	3.996803 e-15	7.312669 e-15	9.4749196 e-04	1.9690224 e-04	4.4408 e-16
F11	0.000000 e+00	0.000000 e+00	1.0678519 e-03	8.698255 e-03	5.5567506 e-07	0.000000 e+00
F12	1.5705448 e-32	1.5705448 e-32	3.4549083 e-03	1.2341902 e-08	3.499239 e-10	1.5705448 e-32
F13	-1.1504403 e+00	-1.1504403 e+00	-1.1504403 e+00	-1.1504400 e+00	-1.1504403 e+00	-1.1504403 e+00

DE and RBI are better than GA, PSO and SA.

Results of the evaluation

- Static functions are not realistic for OC systems
- Most important aspect: noise
  - Noise is caused by e.g. error-prone sensor information
  - F7 is a noisy quartic function:

$$\left[\sum_{i=0}^{n-1} (i+1) x_i^{\ 4} \right] + rand[0,1)$$

	RBI	DE	PSO	GA	SA	Optimum
F7	1.7144777 e-02	2.4342616 e-01	2.0309965 e-01	2.5565637 e-02	1.3845968 e+00	0.000000 e+00

- Best technique: RBI

ORGANIC COMPUTING

Results of the evaluation

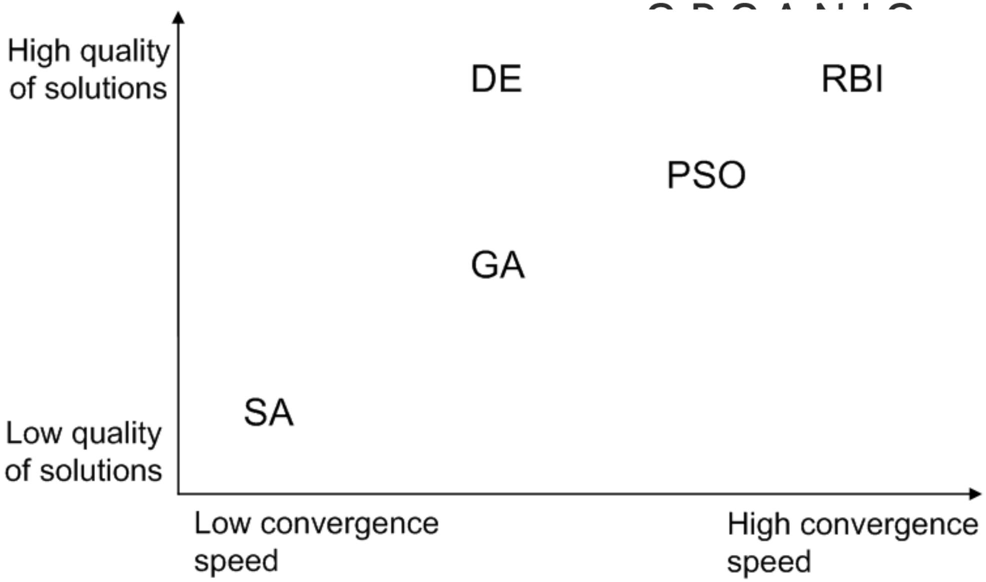
- Criterion: convergence speed
- Required: Success Criterion (SC)

	RBI	DE	PSO	GA	SA	Optimum	SC
F1	9750.00	29640.00	14826.66	12800.00	151601.67	0.000000 e+00	F_{best_1} < 0.01
F2	20546.66	54993.33	28173.33	296393.34	281790.84	0.000000 e+00	F_{best_2} < 0.01
F3	29106.66	63510.00	34376.66	161586.67	344800.84	0.000000 e+00	F_{best_3} < 0.01
F4	6230.00	64683.33	29533.33	149236.67	248167.50	0.000000 e+00	F_{best_4} < 5
F5	12253.33	36496.66	23623.33	35003.33	197495.00	0.000000 e+00	F_{best_5} < 100
F6	10973.33	32563.33	18430.00	17343.33	172253.33	0.000000 e+00	F_{best_6} < 1
F7	1836.66	22916.66	6840.00	2680.00	198515.83	0.000000 e+00	F_{best_7} < 0.1
F8	4150.00	12233.33	6240.00	50790.00	8122.50	-1.2569486 e+04	F_{best_8} < -6500
F9	19626.66	28873.33	17023.33	4743.33	66582.50	0.000000 e+00	F_{best_9} < 100
F10	6873.33	26440.00	12470.00	8963.33	178097.50	4.4408 e-16	F_{best_10} < 2.5
F11	17406.66	45456.66	23443.33	54460.00	218437.50	0.000000 e+00	F_{best_11} < 0.1
F12	6853.33	34973.33	34746.66	58576.66	136327.50	1.5705448 e-32	F_{best_12} < 0.2
F13	7510.00	30746.66	19363.33	14510.00	137881.67	-1.1504403 e+00	F_{best_13} < 0

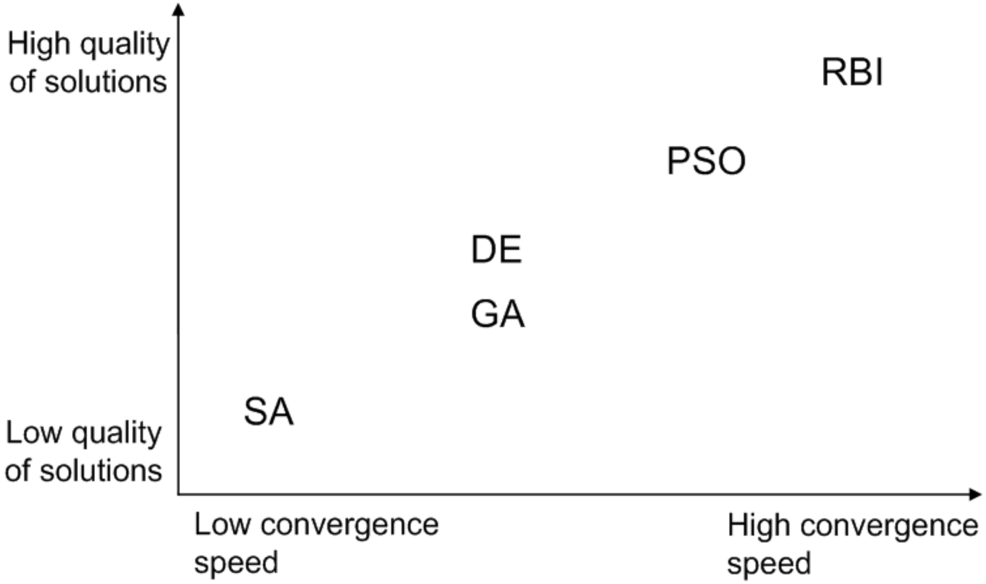
A brief evaluation in OC systems (8)

Summary (static)

- noiseless



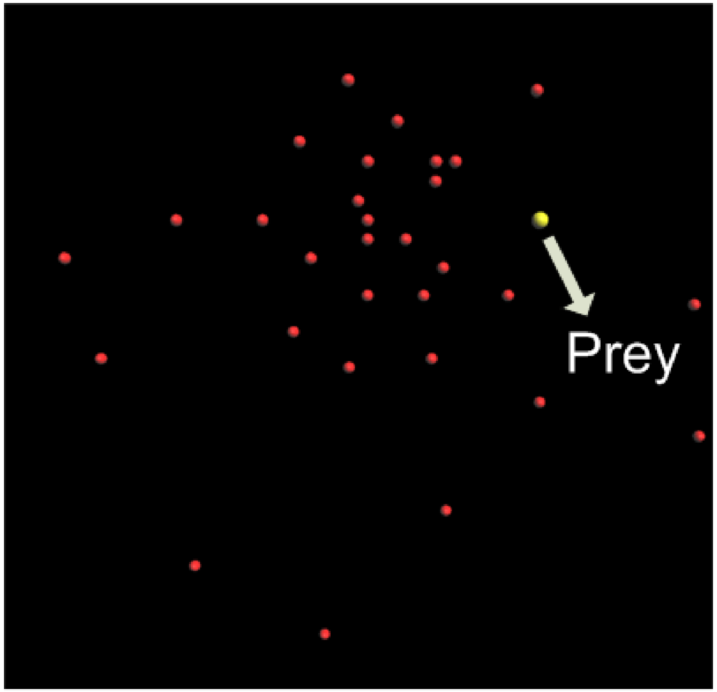
- noisy



A brief evaluation in OC systems (9)

Comparison in dynamic environments

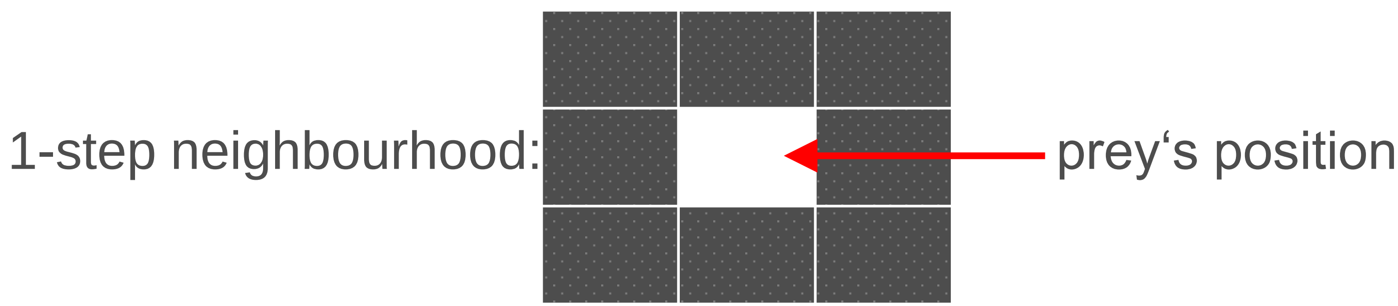
- Scenario from the pursuit (predator-prey) domain
  - Implementation with RePast
  - A time step (iteration) in RePast is called a “tick”.
  - The predators follow and observe the prey and the prey evades the predators.
  - The prey is twice as fast as a predator.



A brief evaluation in OC systems (10)

Comparison in dynamic environments: evaluation criterion

- Each predator counts its “number of observations” (NofOBS).
  - NofOBS is increased each time the prey is in the 1-step neighbourhood of the predator.
  - Goal of a predator: <u>maximise</u> the value of its NofOBS.
  - System performance: the sum of all NofOBS.

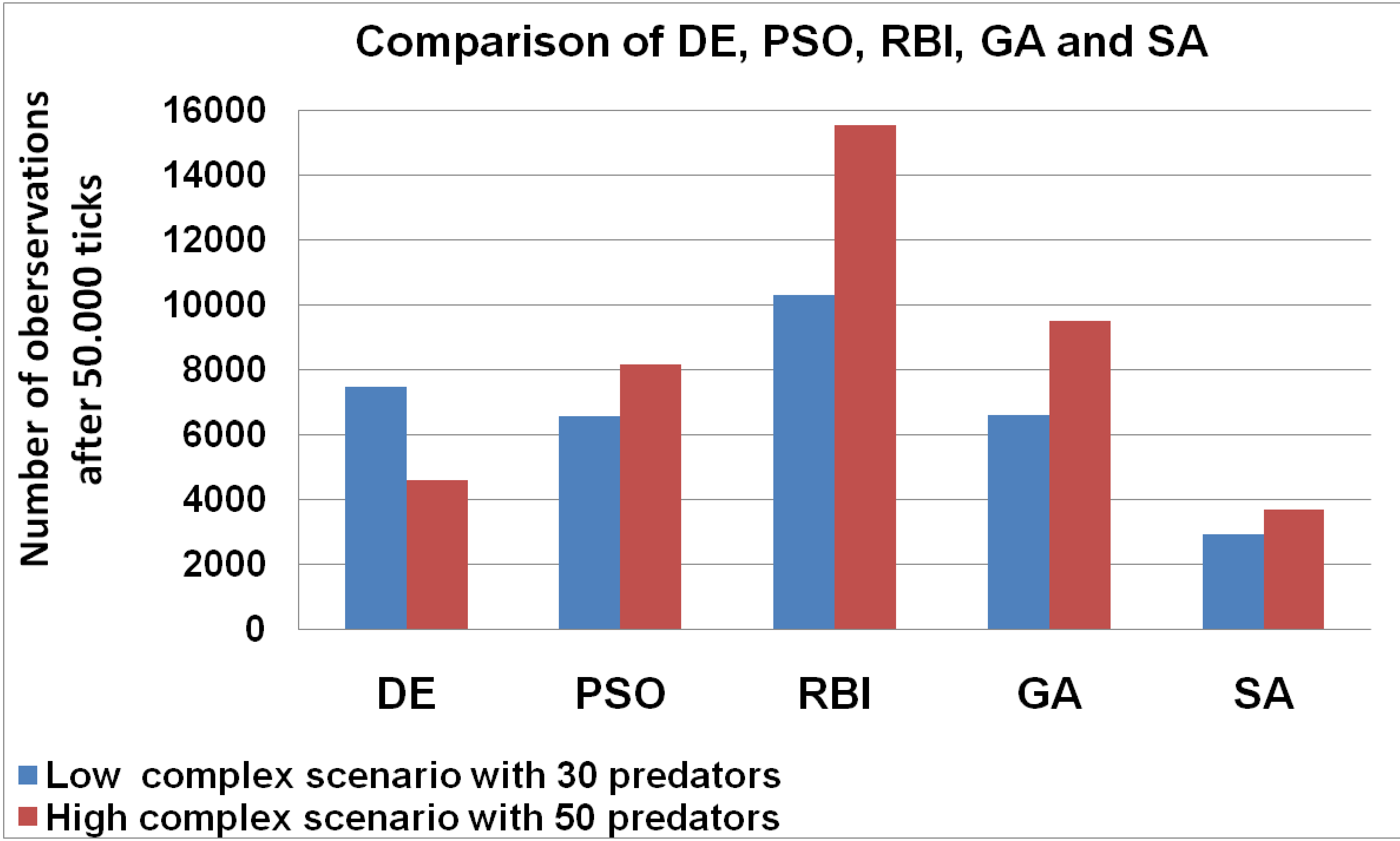


- The prey evades the predators to stay unobserved.
  - Goal of the prey: <u>Minimise</u> the sum of all NofOBS.

ORGANIC COMPUTING

Evaluation setup

- The prey is twice as fast as a predator.
  - The predators work together to collectively observe the prey.
  - The success (i.e., the optimal behaviour) of a predator depends on the behaviour of other predators in the system resulting in a self-referential fitness landscape.
- Experimental setup:
  - Each predator optimises a single parameter *Pi* individually to change (i.e., adapt) its behaviour.
  - The search range for *Pi* values is set to [-10, 10].
  - The total number of observations is measured after **50,000** ticks.
  - Each predator optimises its behaviour every **100** ticks.
    - The number of function evaluations for a single predator is limited to **500** (50.000 / 100).
    - Predator's fitness is calculated using its NofOBS in the last 100 ticks.


Comparison of DE, PSO, RBI, GA and SA
Number of oberservations after 50.000 ticks
16000
14000
12000
10000
8000
6000
4000
2000
0
DE
PSO
RBI
GA
SA
Low complex scenario with 30 predators
High complex scenario with 50 predators

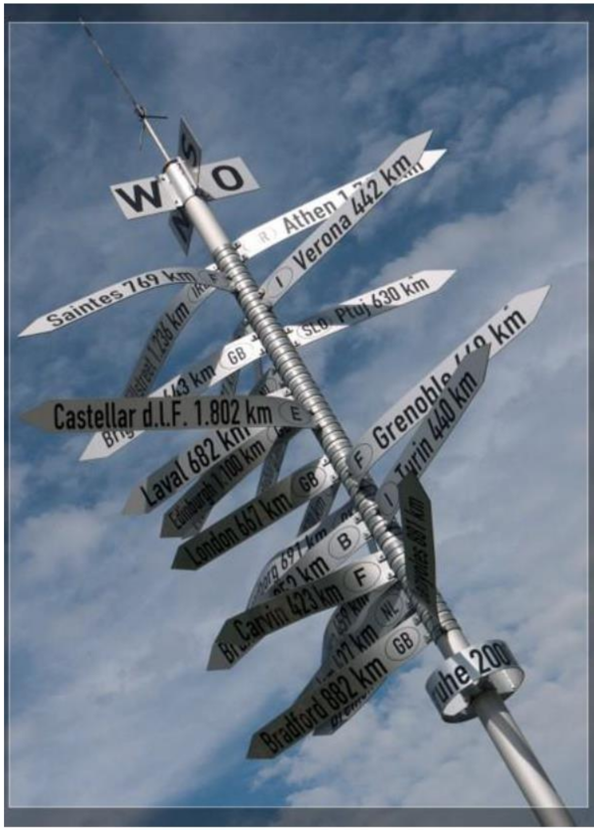
A brief evaluation in OC systems (13)

Conclusion

- Which technique should be applied to the runtime optimisation problem in OC systems?
  - Optimisation is slow!
    → We need techniques that converge extremely fast!
  - Optimisation needs performance!
    → We need techniques that can be stopped and resumed.
    → We need techniques that find the best possible solution as fast as possible.
  - OC systems are real-world systems!
    → We need techniques that can cope with noisy data and error-prone sensor information.
- In most cases, the RBI algorithm will be the most promising solution!

Agenda

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings



Conclusion

This chapter:

- Motivation
- Term definition
- Stochastic approaches
- Nature-inspired techniques
- Role-based imitation algorithm
- A brief evaluation in OC systems
- Conclusion and further readings

By now, students should be able to:

- Define what an optimisation problem is
- Outline different concepts to solve optimisation problems
- Explain nature-inspired techniques, especially Evolution Strategies, Particle Swarm Optimisation, and Simulated Annealing
- Apply the RBI algorithm
- Compare the different concepts in the context of OC problems

References

Optimisation technology Center – http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/

Craig W. Reynolds: “Flocks, herds, and schools: A distributed behavioural model”. ACM Computer Graphics, 21(4):25–34, 1987

James Kennedy and Russell Eberhart: “Particle swarm optimization”. In Proceedings of IEEE International Conference on Neural Networks, pages 1942–1948, Piscataway, NJ, USA, 1995. IEEE Press

Carlisle, A. and Dozier, G. (2001). “An Off-The-Shelf PSO”, Proceedings of the 2001 Workshop on Particle Swarm Optimization, pp. 1-6, Indianapolis, IN. (http://antho.huntingdon.edu/publications/Off-The-Shelf_PSO.pdf)

Kennedy, J. (1997), “The Particle Swarm: Social Adaptation of Knowledge”, Proceedings of the 1997 International Conference on Evolutionary Computation, pp. 303-308, IEEE Press

A comparative Study of Differential Evolution, Particle Swarm Optimisation and Evolutionary Algorithms on Numerical Benchmark Problems”, Vesterstrom et al., CEC 2004

Cakar, Emre, Sven Tomforde, and Christian Müller-Schloer. "A role-based imitation algorithm for the optimisation in dynamic fitness landscapes." Swarm Intelligence (SIS), 2011 IEEE Symposium on. IEEE, 2011.

Questions …?