Lecture

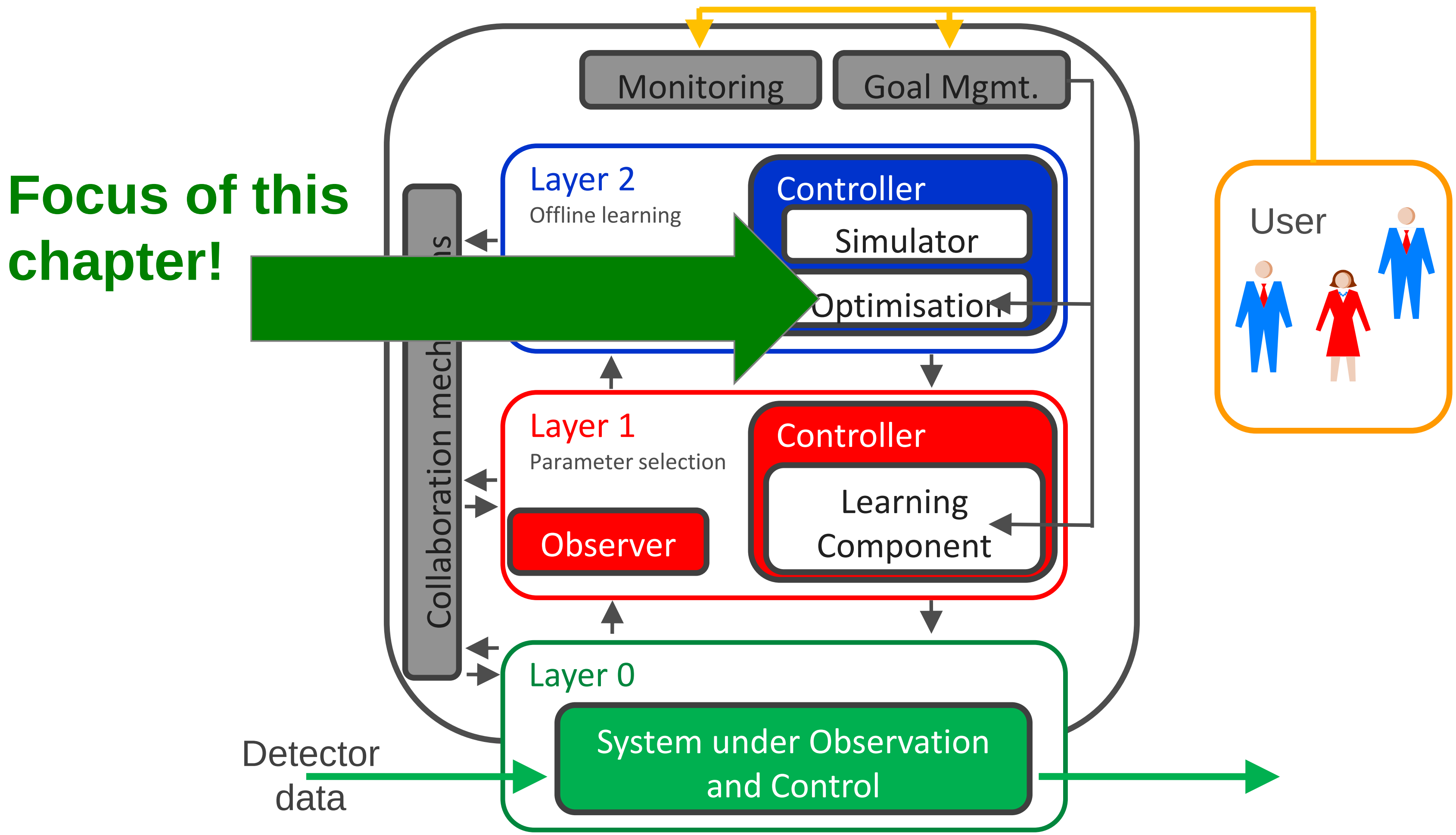
Organic Computing II

Summer term 2019

Chapter 5: Optimisation

Lecturer: Anthony Stein, M.Sc.

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)



Monitoring
Goal Mgmt.
Layer 2
Offline learning
Controller
Simulator
Optimisation
Collaboration mechanisms
Layer 1
Parameter selection
Controller
Observer
Learning Component
Layer 0
System under Observation and Control
Detector data
User
Focus of this chapter!

Agenda

Content

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

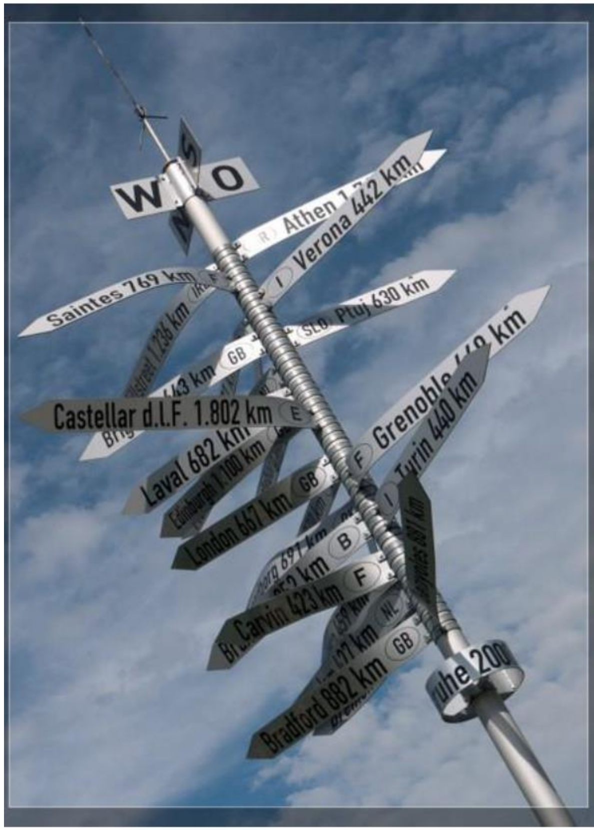
Goals

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

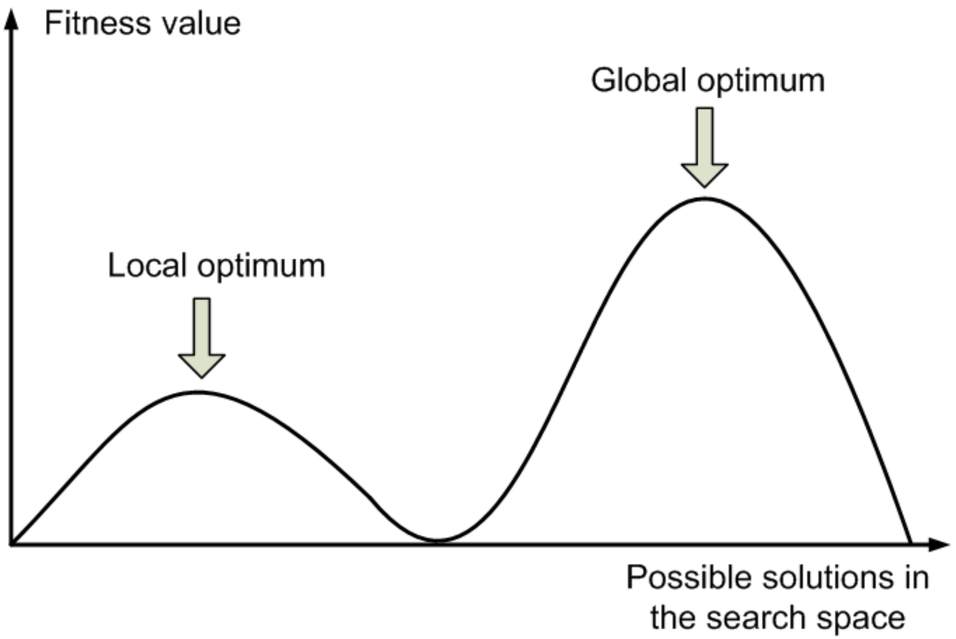


What is optimisation?

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

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!

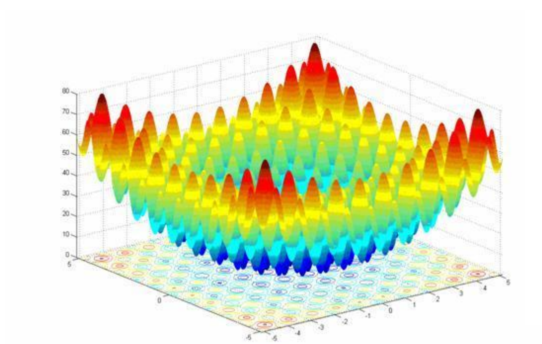
⇒ In many cases, OC systems have self-referential landscapes!

Static fitness landscapes

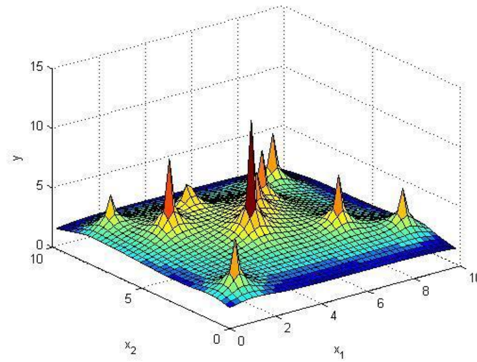
- Typical for problems with continuous or discrete parameter spaces.
- Problems with a continuous parameter space are
  e.g. the benchmark problems in function optimisation literature.
- A problem with a discrete parameter space is
  e.g. the Travelling Salesman Problem (TSP).

Examples: Static fitness landscapes

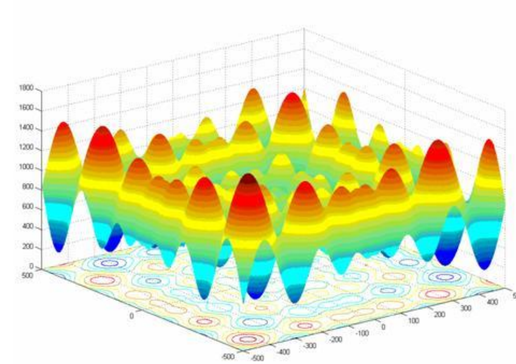
Examples for static fitness landscapes



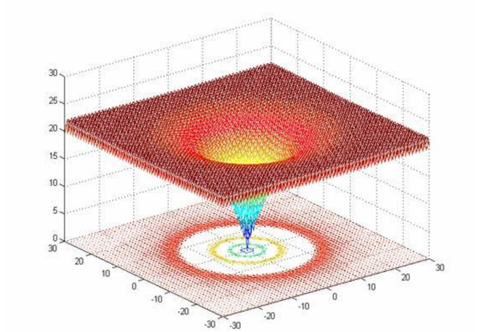
Rastrigin function



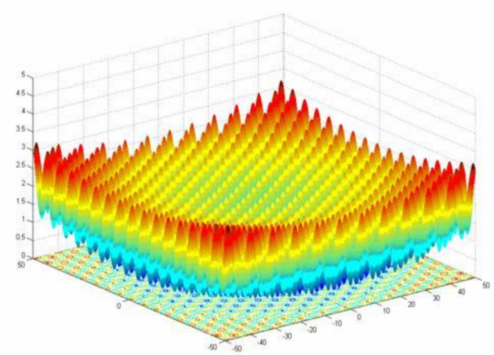
Shekel function



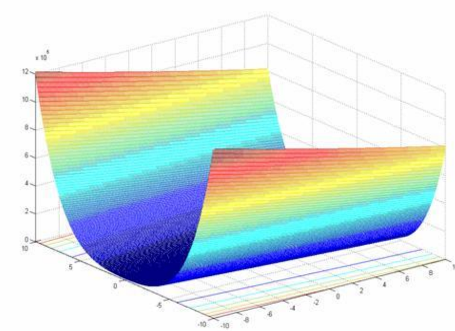
Schwefel function



Ackley function



Griewank function



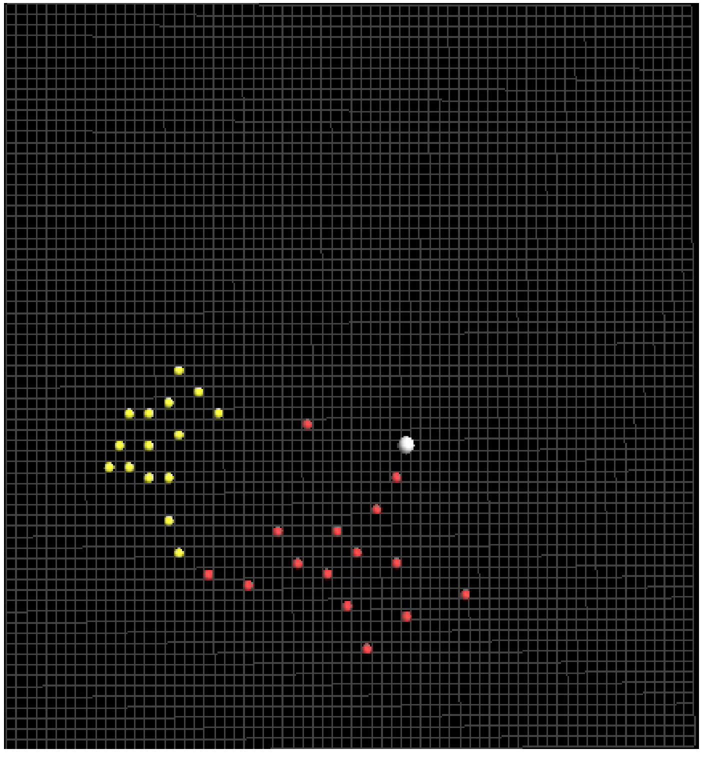
Rosenbrock function

Time-varying fitness landscapes

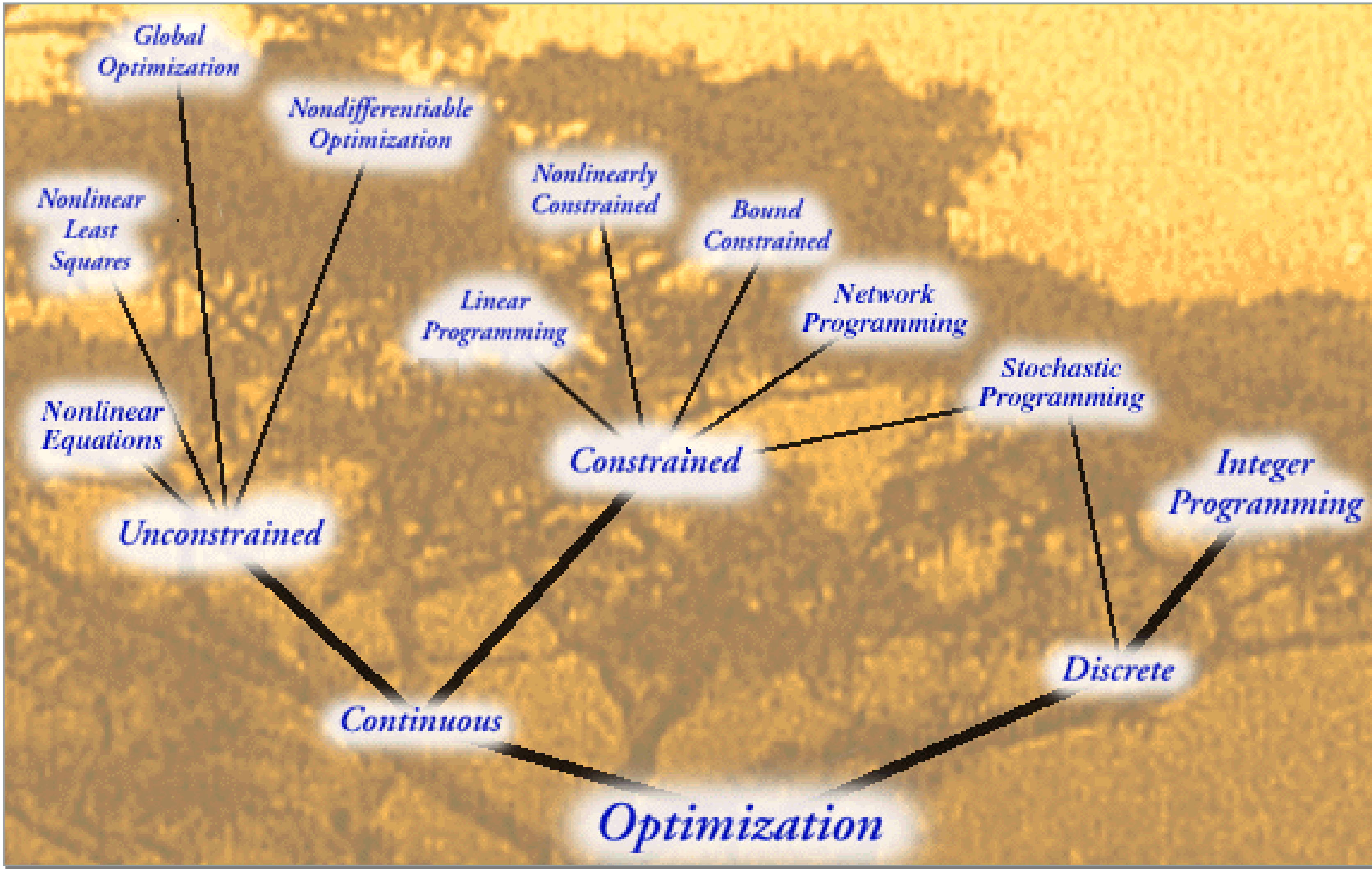
- Global optimum moves over time.
- Optimisation has to find and follow the optimum.
- Example:
  - Search space: Different locations on the earth
  - Fitness values: Temperature of the given location
  - Optimum: Locations with a temperature between 20 and 25 centigrade
  - Characteristic: Fitness landscape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



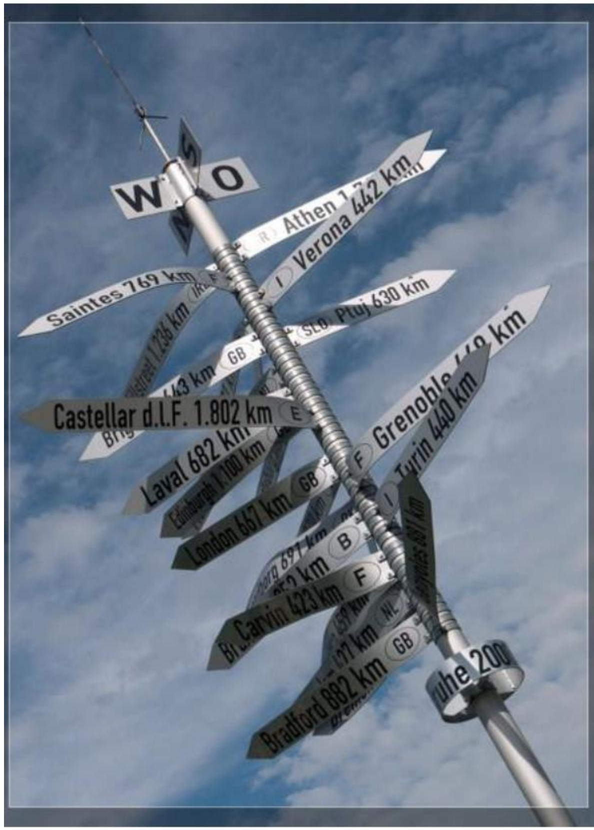
Source: Optimisation technology Center – http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/

Choosing optimisation methods

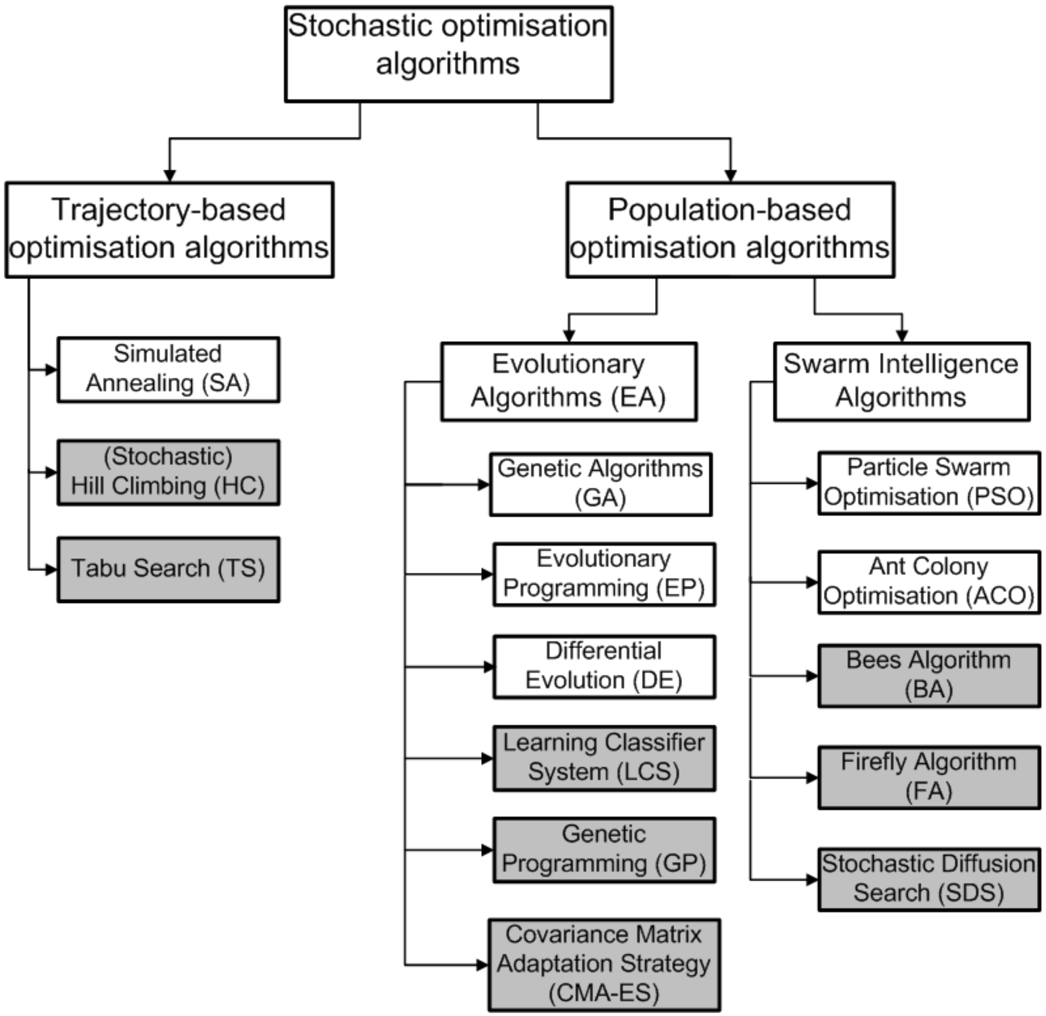
- Algorithms have very different flavour depending on the specific problem.
  - Closed form vs. numerical vs. discrete
  - Local vs. global minima
  - Running times ranging from O(1) to NP-hard
- In OC systems, optimisation at runtime ⇒ Specific requirements!
  - Fast convergence, minimised effort
 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

Agenda

- 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

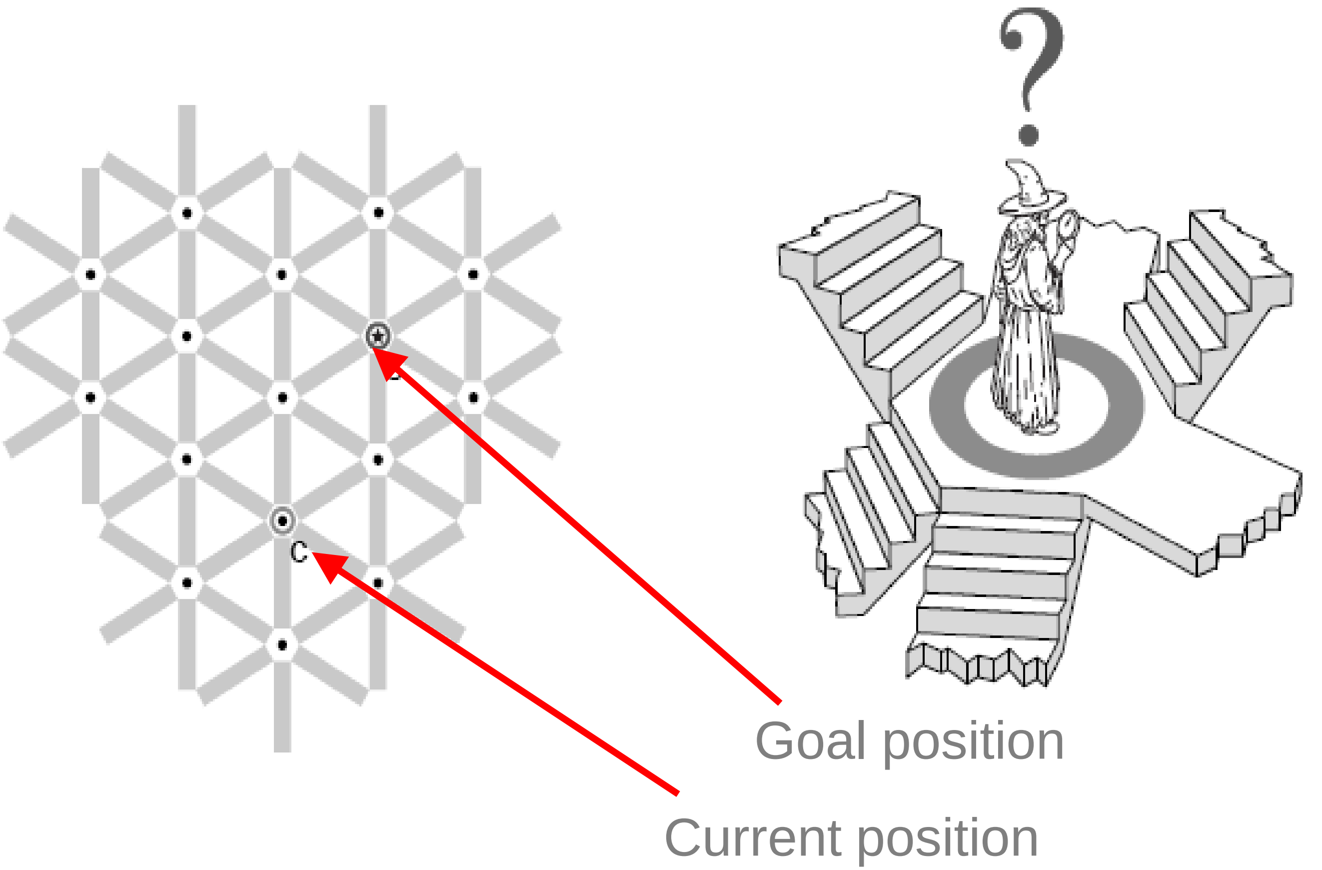
Components of a stochastic optimisation method

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

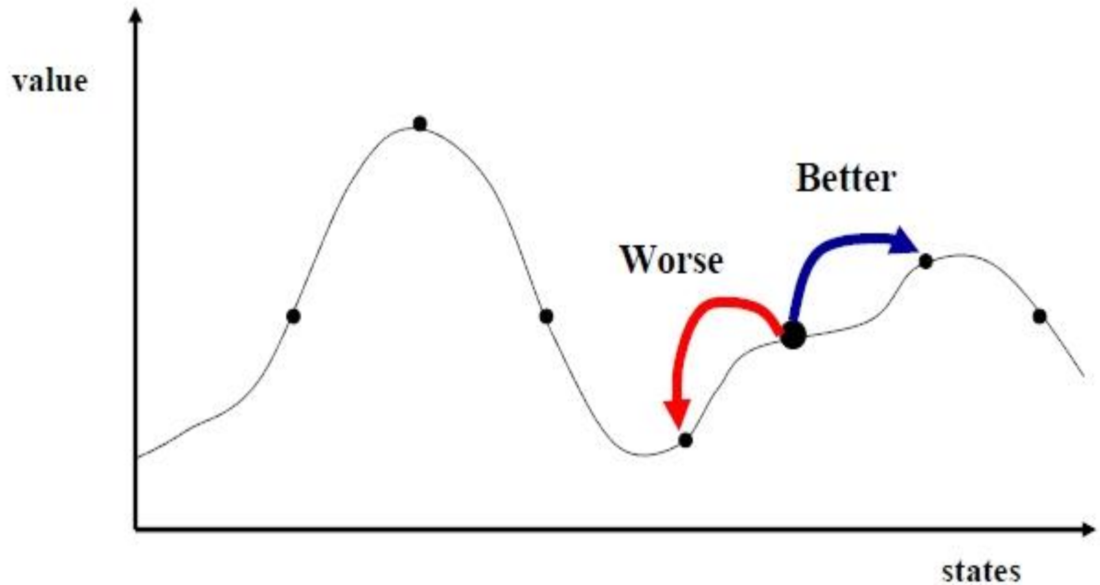
This differs the most between different optimisation techniques!

Idea: Greedily select the best candidate in some neighbourhood.



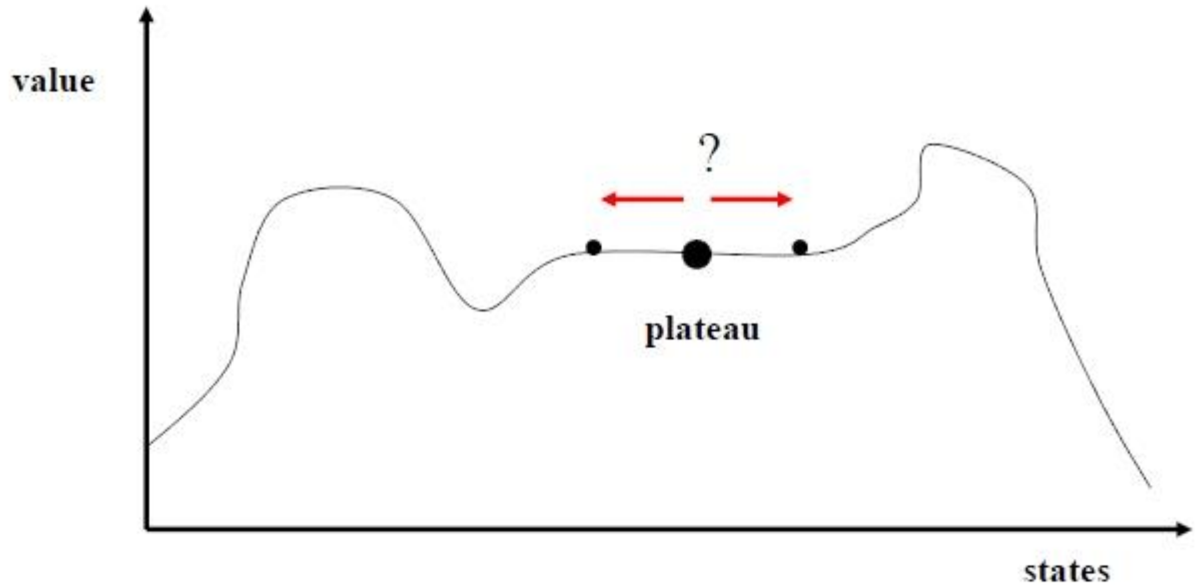
Hill climbing

- 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



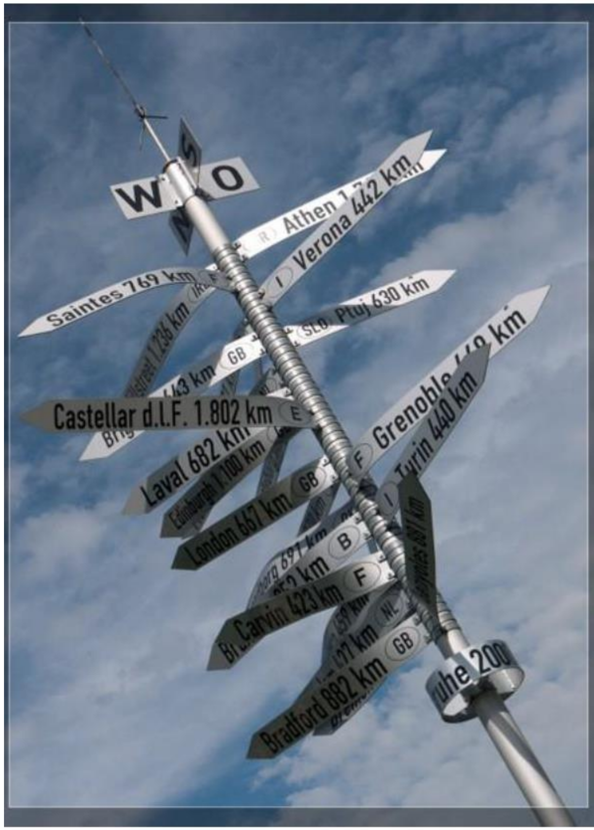
Improving Hill Climbing

- 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

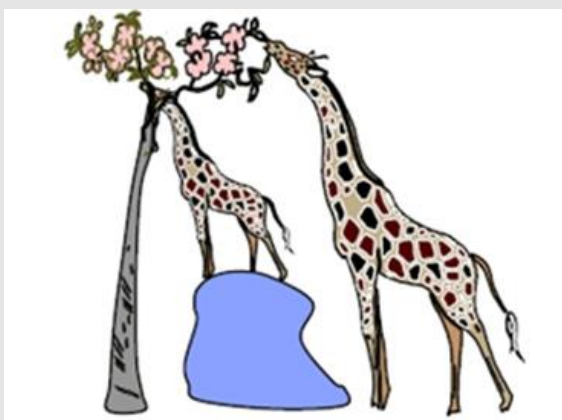
a) Physical processes

e.g. cooling process of metals

b) Genetic Algorithms

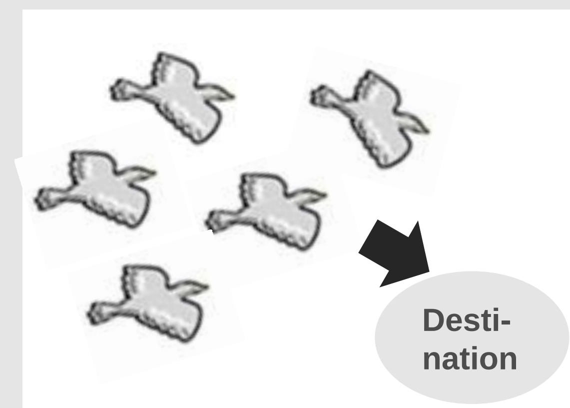
“survival of the genetically fittest”

c) Memetic Algorithms



„survival of the fittest & experienced“

d) Swarm behaviour



particle swarm: flock migration

Classes of nature-inspired optimisation heuristics

- 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)

ORGANIC COMPUTING

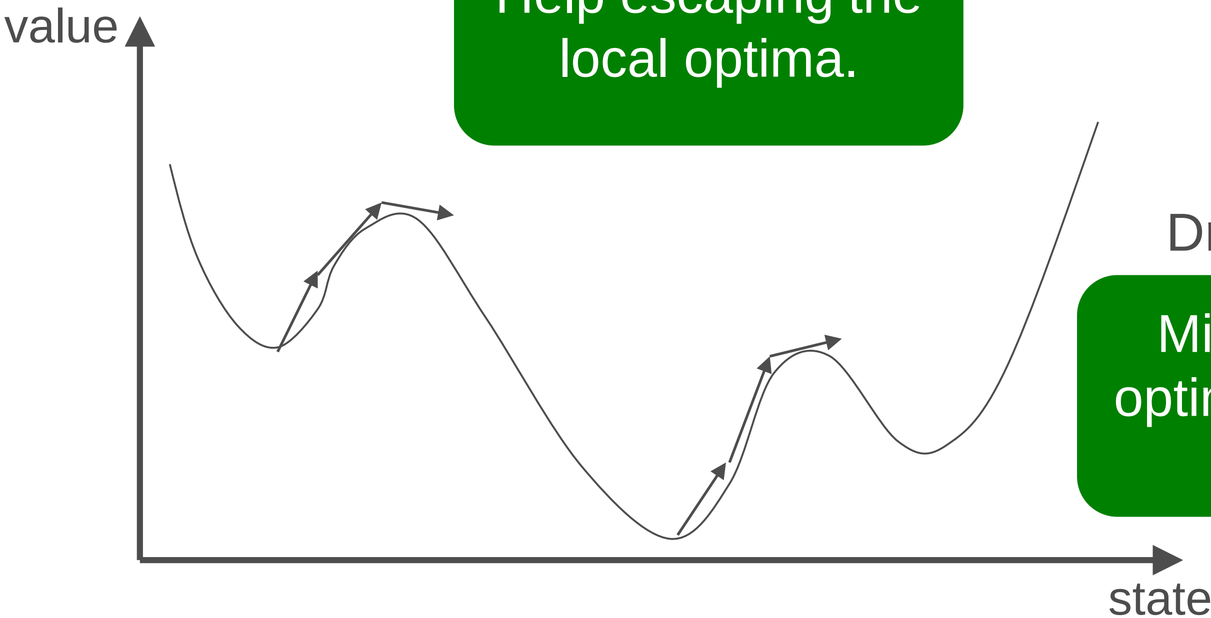
- Is a probabilistic search technique and imitates physical processes.
- Observation in nature:
  - At high temperatures, molecules move freely
  - At low temperatures, molecules “get stuck”
  - This is how crystals are formed in a thermodynamic process
- Other names:
  - Monte Carlo Annealing
  - Statistical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

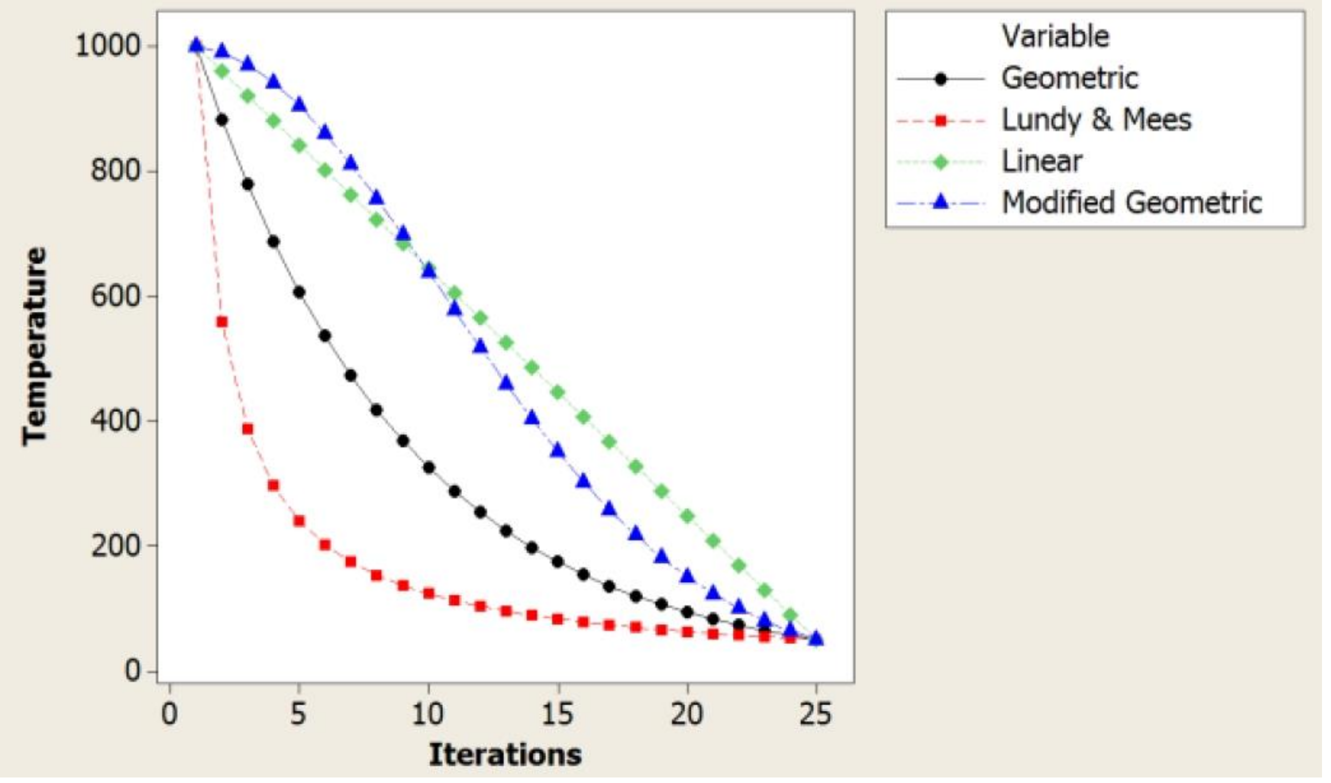
Might pass global optima after reaching them.

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.



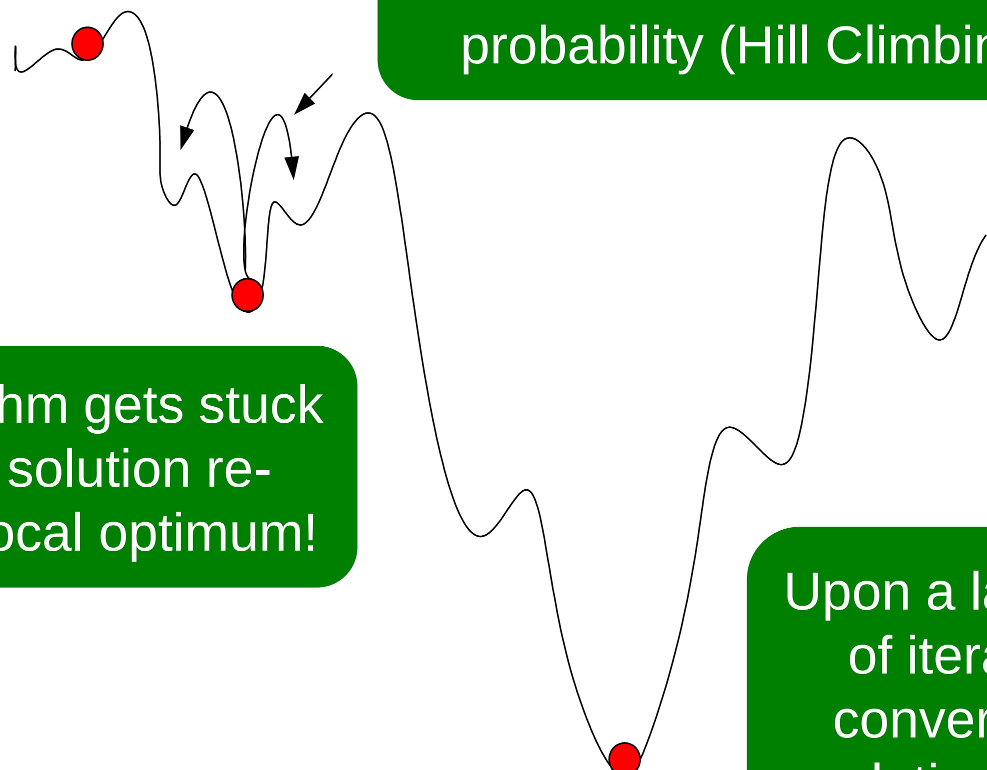
Example: Ball on the terrain

- 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: