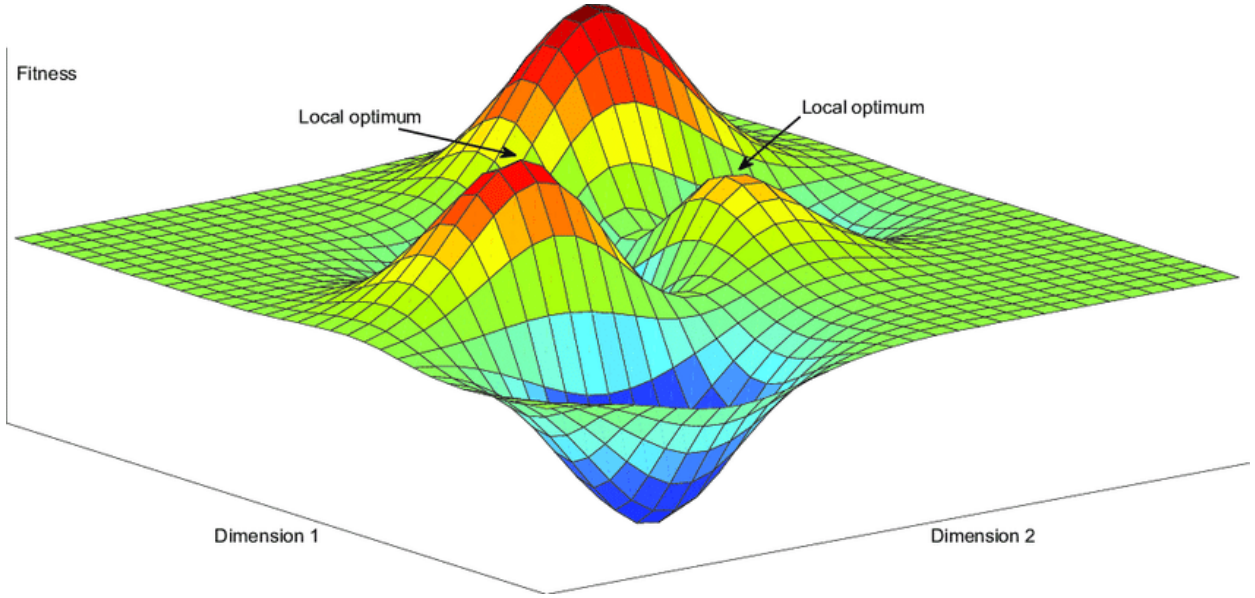
Continuous Optimization: Overview

Alberto TONDA, Senior Researcher (DR)

UMR 518 MIA-PS (Applied Mathematics and Computer Science)
INRAE, AgroParisTech, Université Paris-Saclay
Institut des Systèmes Complexes, Paris-Ile-de-France

Outline

- Gradient-based
  - Gradient descent
  - Successors of gradient descent
- Stochastic/Approximate
  - Basin-hopping
  - Evolutionary algorithms

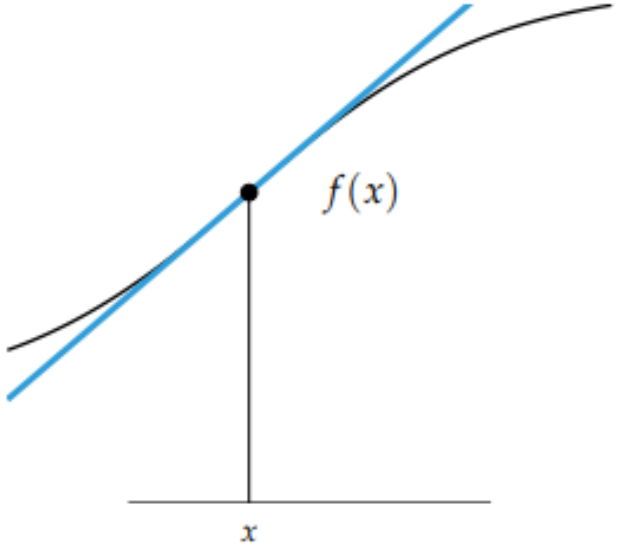


Introduction

- Continuous optimization
  - Solution is a point in $x \in \mathbb{R}^d$, with *d* number of dimensions
  - E.g. $x = \{0.2, 1.3, \dots, -0.9\}$
  - Usually bounded (different bounds by dimension are possible)
- For the moment, we don't consider other constraints
- Different techniques make different assumptions
- Trade-offs efficiency/applicability
- **Overview of a few, selected algorithms**

Gradient-based techniques

- Derivative $f'(x)$ of a function $f(x)$ in a single point
  - Rate at which the value of *f* changes at *x*
  - Can be visualized as the slope of a tangent line



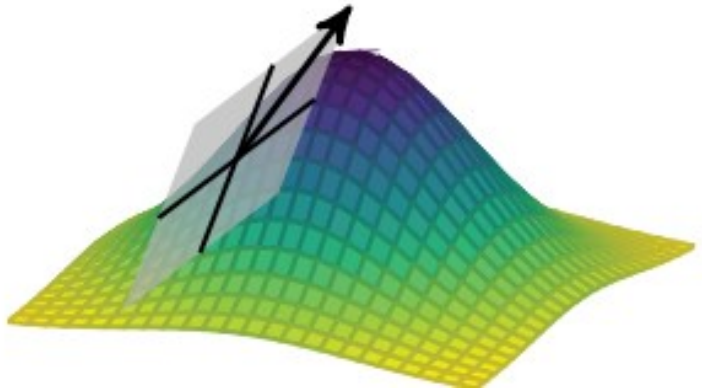
Gradient-based techniques

- Different interchangeable notations

$$f'(x) \equiv \frac{df(x)}{dx}$$

- Derivative in multiple dimensions is the **gradient (Jacobian)**

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \quad \frac{\partial f(\mathbf{x})}{\partial x_2}, \quad \ldots, \quad \frac{\partial f(\mathbf{x})}{\partial x_n}\right]$$



Gradient-based techniques

- Simple intuition: always follow steepest descent/ascent
- Assumptions
  - Gradient can be computed
  - Or approximated
  - Function is convex (guarantees global optimum)
  - Also runs on non-convex functions

Algorithm: Gradient Descent

Input: convex function f, step size γ_t, initial point x_0

1 **for** $t = 0 \dots T-1$ **do**

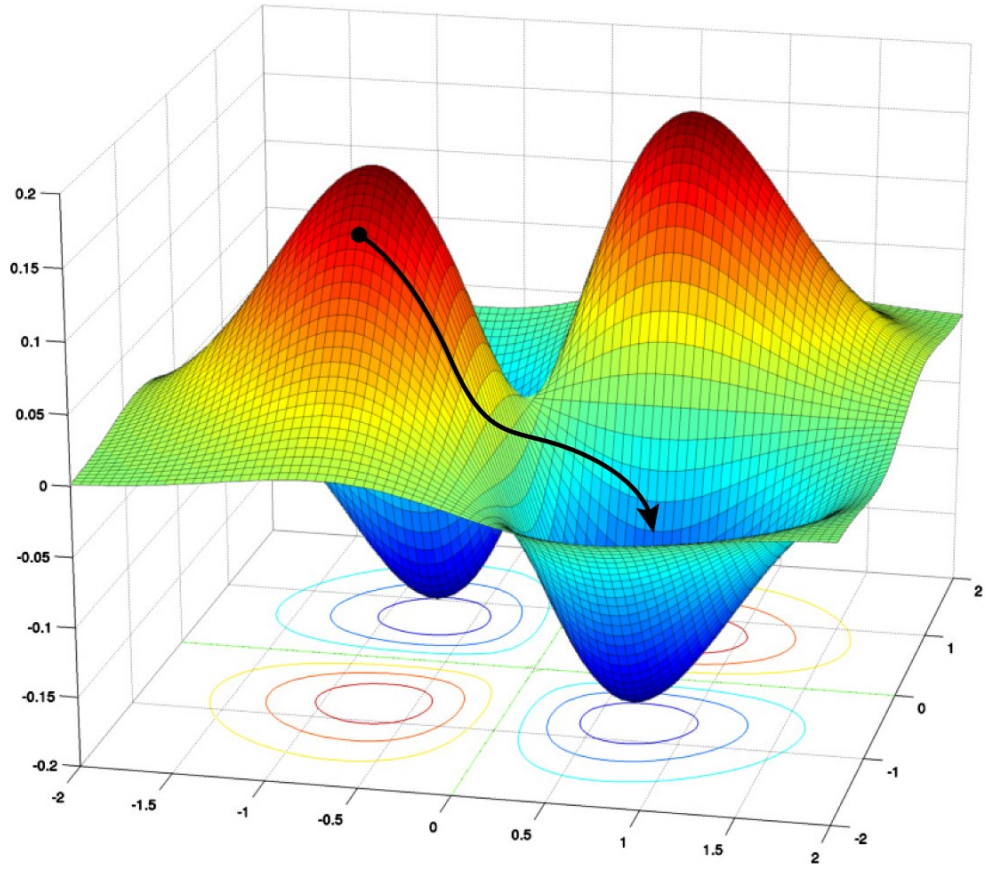
2 Compute $g_t \in \partial f(x_t)$

3 $x_{t+1} \leftarrow x_t - \gamma_t g_t$

4 **return** x_T

Gradient-based techniques

- Simple intuition: always follow steepest descent/ascent
- Assumptions
  - Gradient can be computed
  - Or approximated
  - Function is convex (guarantees global optimum)
  - Also runs on non-convex functions



Issues with gradient-based techniques

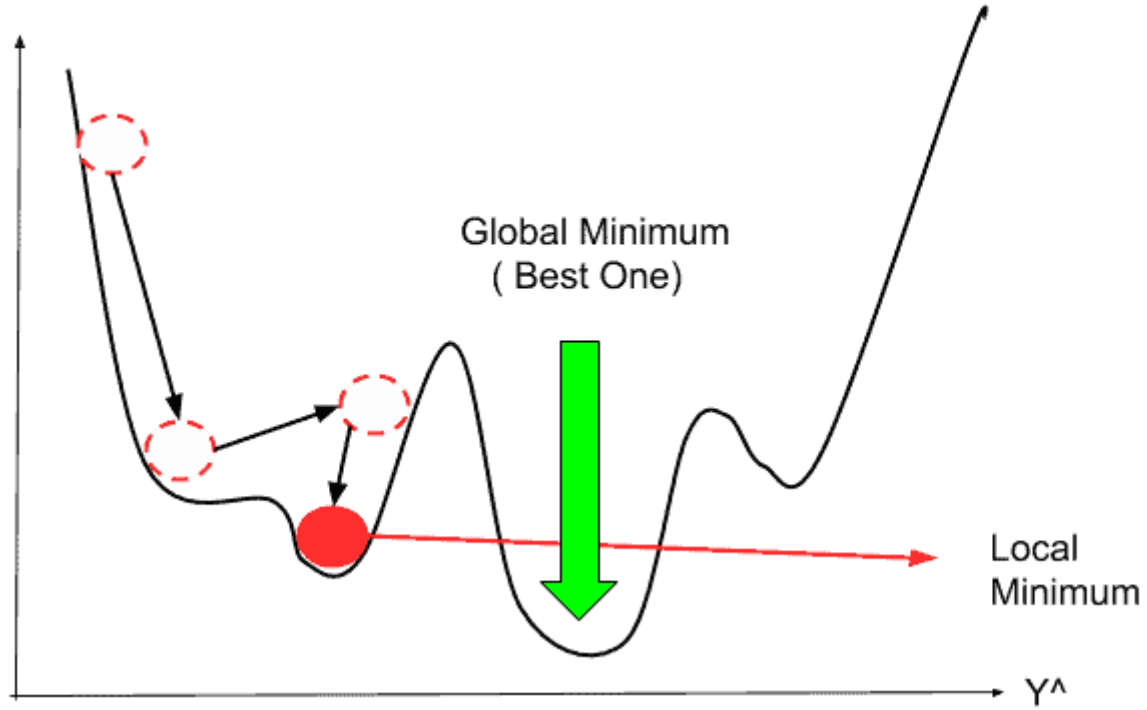
- Brainstorming!

Issues with gradient-based techniques

- Hard/impossible to get out of local optima
- Starting point of exploration matters
- Step size? Too small / too large leads to convergence issues

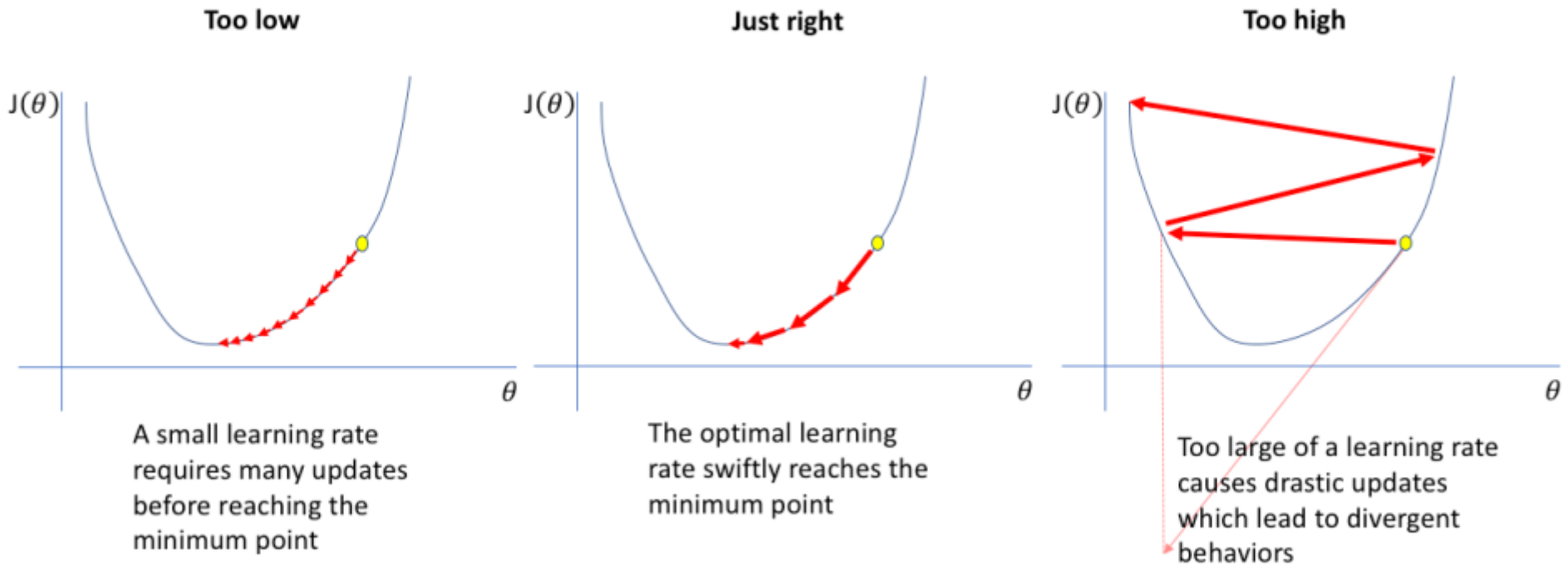
Issues with gradient-based techniques

- Starting point of exploration matters



Issues with gradient-based techniques

- Step size? Too small / too large lead to convergence issues



Gradient-based techniques for neural networks

- Why are gradient-based techniques so successful?
  - Actually, *stochastic gradient descent* (SGD) & descendants
  - Techniques made popular by (deep) neural networks
  - State-of-the-art for NNs, not very used elsewhere
  - Can escape local optima!

Modern gradient-based techniques

- Momentum

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} - \alpha \mathbf{g}^{(k)}$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{v}^{(k+1)}$$

- Accumulates “velocity” like a ball rolling down an incline
- Much faster at traversing nearly flat areas of search space
- However, it adds an extra parameter (α, β)

Modern gradient-based techniques

- And now it cumulates *too much* momentum!
- Nesterov momentum

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)} + \beta \mathbf{v}^{(k)})$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{v}^{(k+1)}$$

- I evaluate the gradient at the point I plan to end in
- And rescale my velocity accordingly

Modern gradient-based techniques

- Further issues motivated further advances (**Adagrad**)
  - The step size could be different in each dimension
  - Maintain a “memory” of the gradient values in each direction

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}} g_i^{(k)}$$

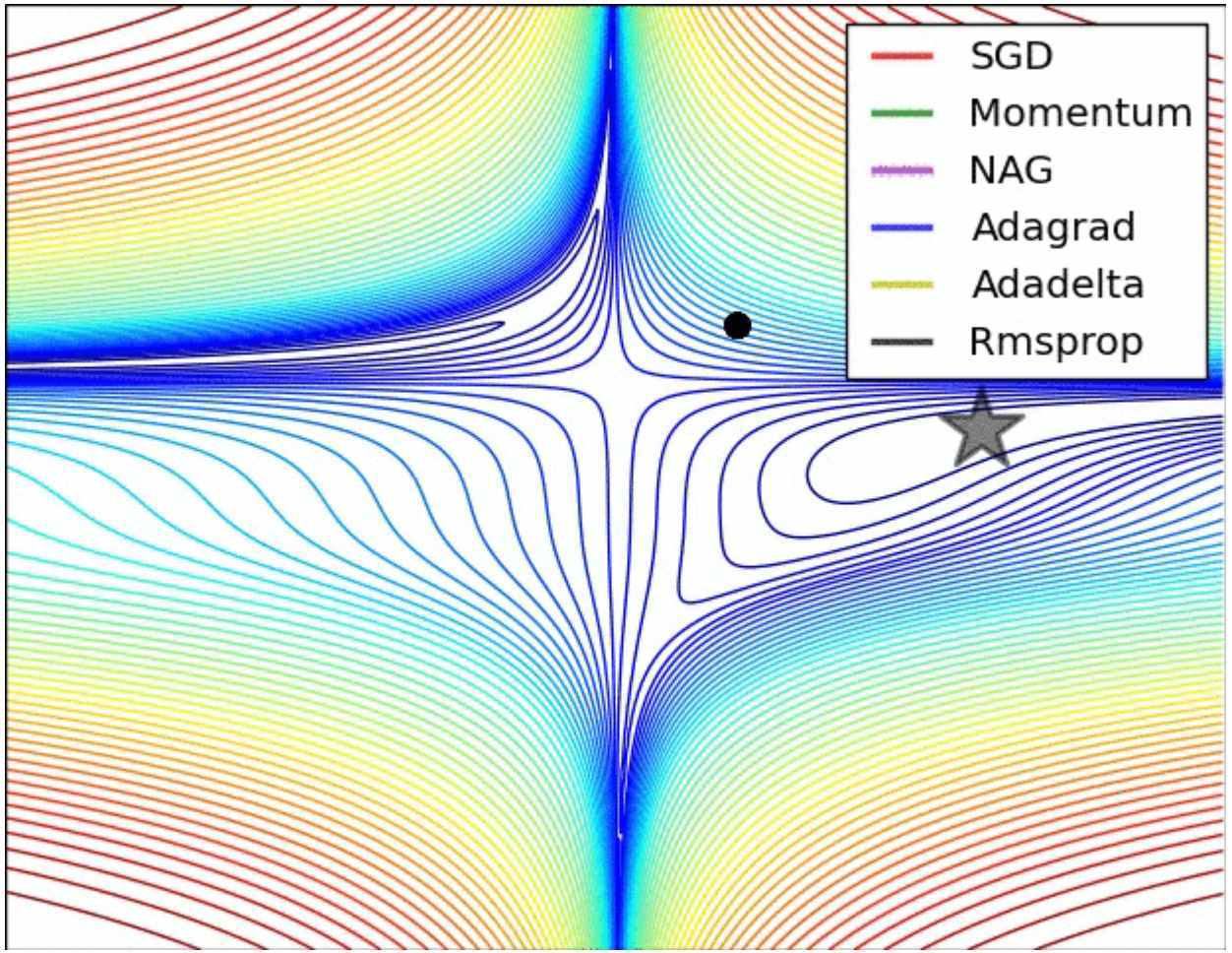
$$s_i^{(k)} = \sum_{j=1}^{k} \left(g_i^{(j)} \right)^2$$

- However, this led to the step size always decreasing

Modern gradient-based techniques

- Solutions to issues of previous algorithms create new issues
- Latest (**Adam** and **RMSProp**) have 3-4 (hyper)parameters
- Default values work reasonably well (but not always)

Modern gradient-based techniques



Stochastic/Approximate techniques

- Limitations of gradient-based techniques were evident
- Practical problems had objective functions with local optima
- You can randomly sample the search space…
- …but exploiting feedback/function characteristics is better!

PSEUDO-random number generation

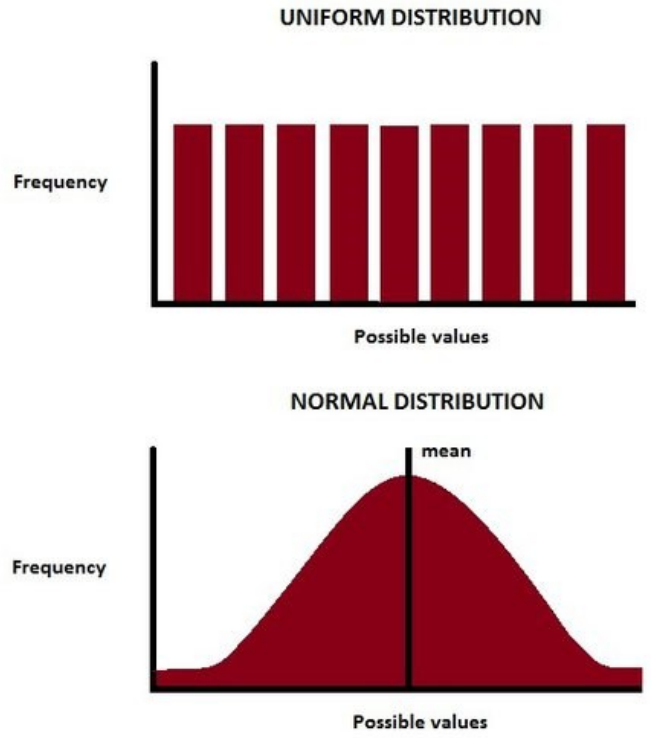
- Important caveat!
  - “Random” numbers in a computer are NOT random
  - Specialized algorithms can produce long sequences of numbers
  - …but in the end they repeat (!!), they have a period
  - Details are complex and out of scope: see **Mersenne Twister**
- Concrete consequences
  - PRNG algorithms need to be initialized with *random seed*
  - All sequences that start from same random seed are identical
  - Storing and setting random seed ensures repeatability of experiments

PSEUDO-random number generation

STORE THE RANDOM SEED

PSEUDO-random number generation

- Most of the continuous values we extract
  - Either sampling a Gaussian distribution
  - Or uniform distribution

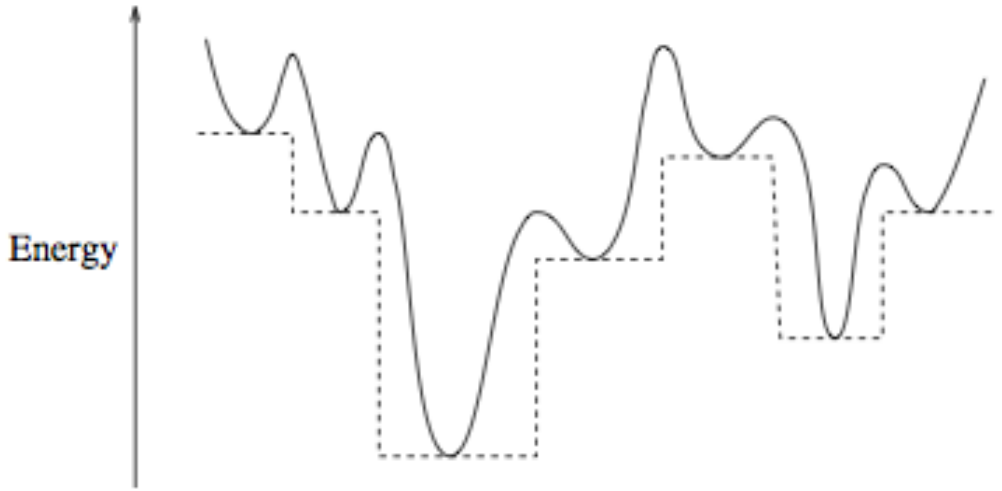


Brainstorming!

- How to use stochasticity to go beyond limitations of gradient-based techniques?

Basin-hopping

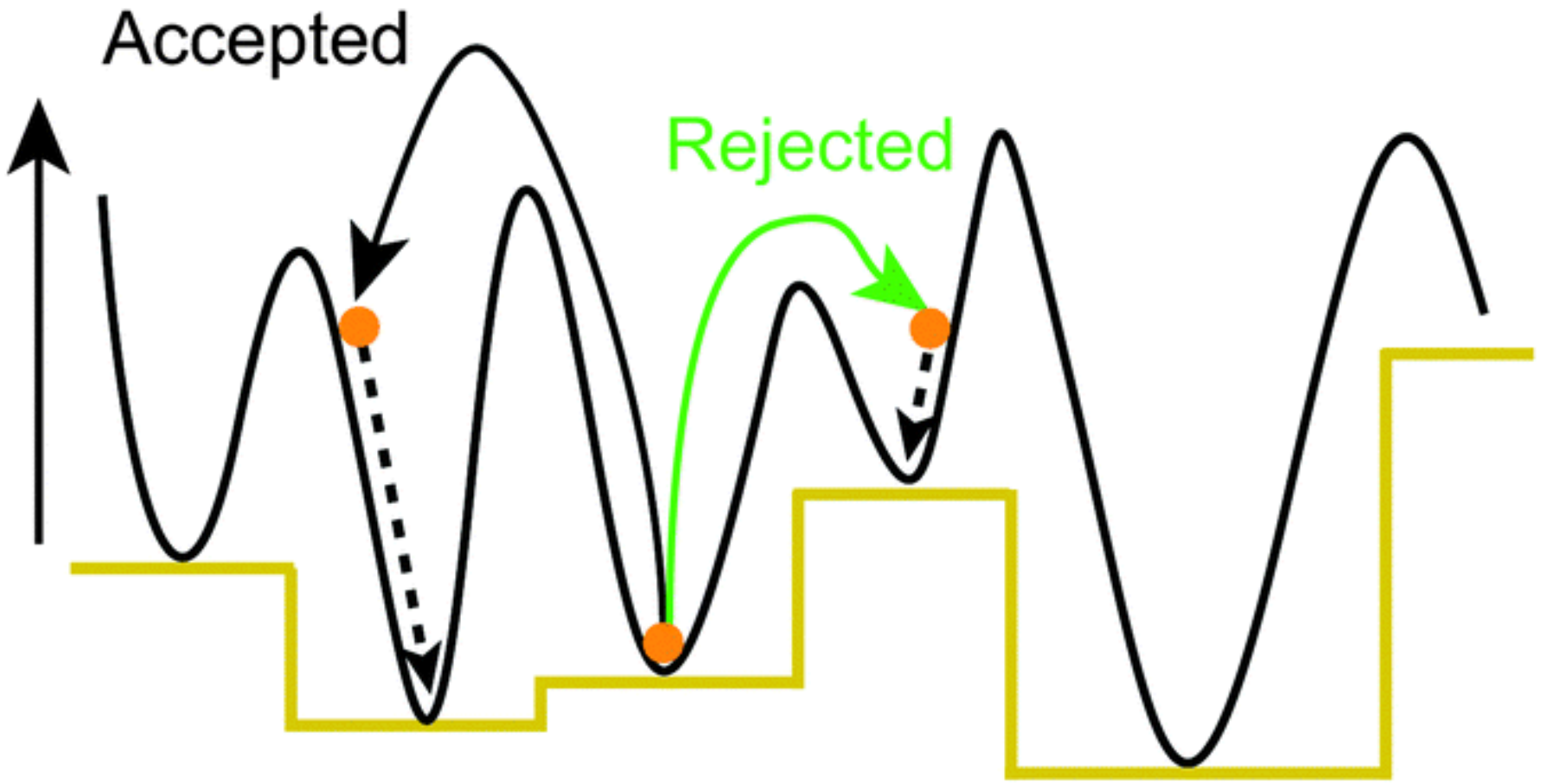
- Developed starting from a practical application
- Protein folding, can compute derivative of objective function
- However, it does contain several local optima



Basin-hopping

- Start from an initial user-defined point x_o
- Apply gradient descent, end up in local optimum x_c
- For a user-defined number of iterations T
  - Move from x_c in a random direction by a random step s in $(0, S)$
  - Starting from the new point x_t , apply gradient descent
  - End up in (another? the same?) local optimum $x_c{}'$
  - If $f(x_c{}')$ has a better value than $f(x_c)$, $x_c = x_c{}'$

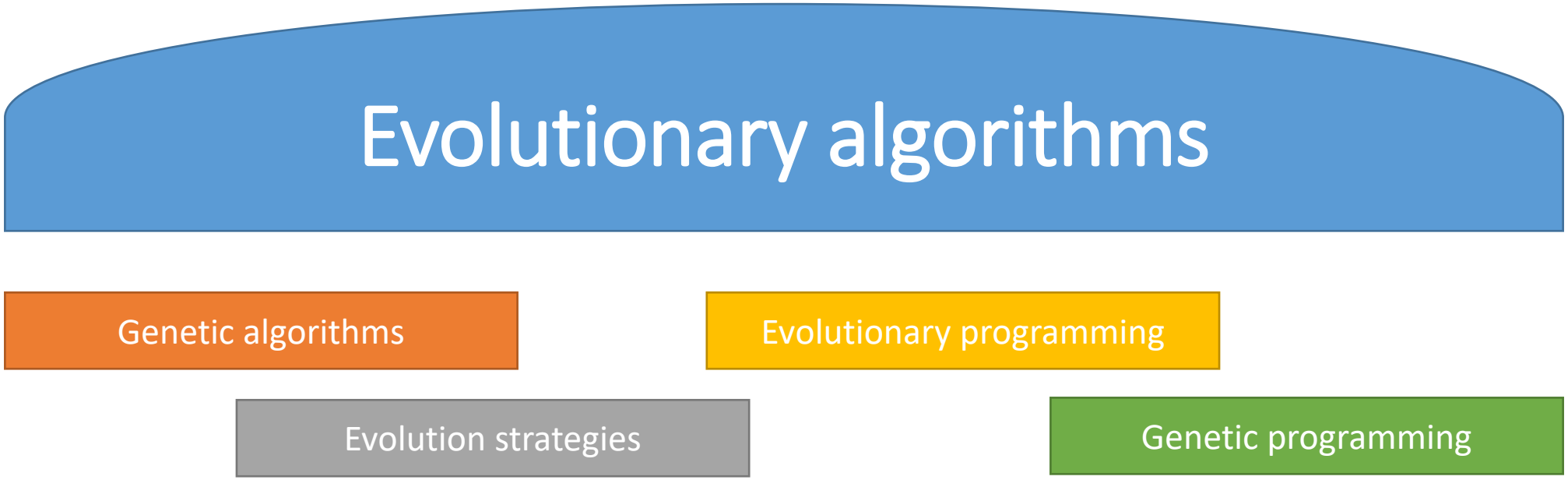
Basin-hopping



Evolutionary algorithms

- But what if our objective function is **not derivable**?
- Can we use only the feedback from the function…
- …and do better than random?
- Assumption: “good candidate solutions are usually close to other good solutions”

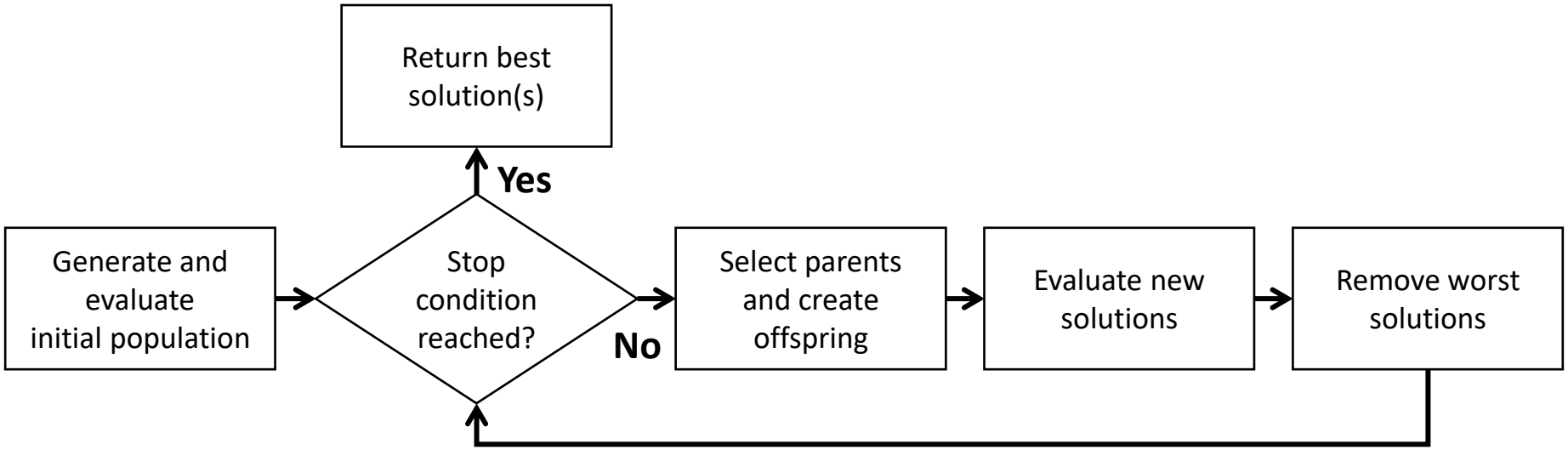
Evolutionary algorithms



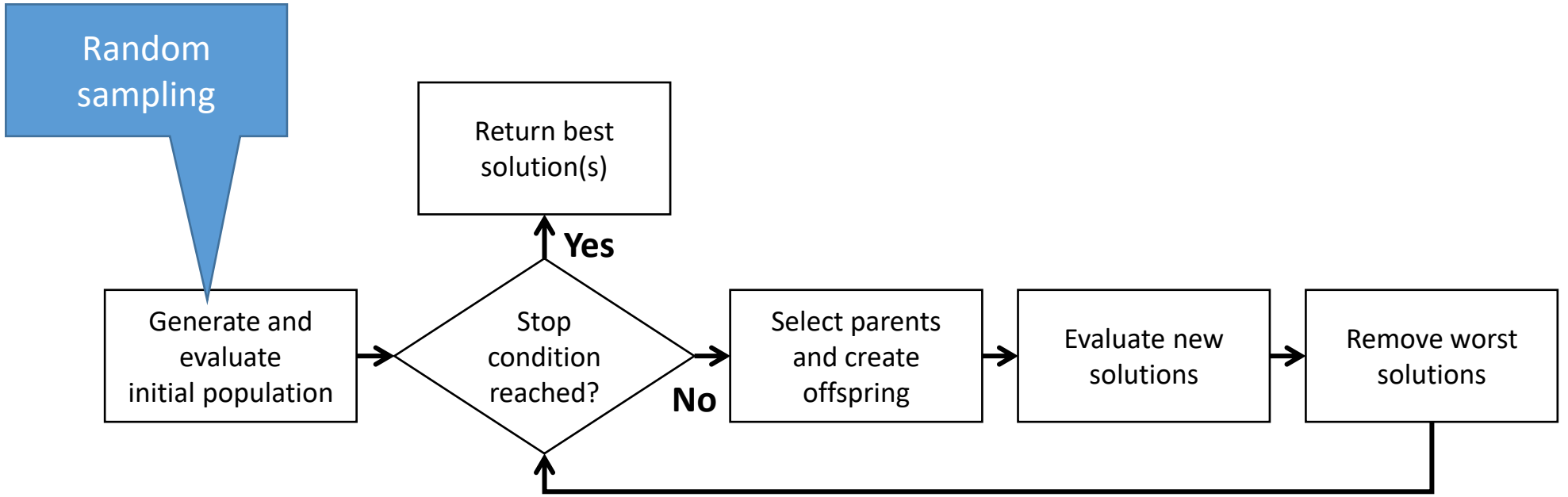
Evolutionary algorithms

- Stochastic optimization techniques
  - Sample search space, proceed with "natural selection" of candidate solutions **Individuals**
  - "Memory" of search space with current set of candidate solutions **Population**
  - Best solutions "reproduce" more often
  - New solutions **Offspring** are a slightly altered version of parents
  - Evaluate solution performance **Fitness**

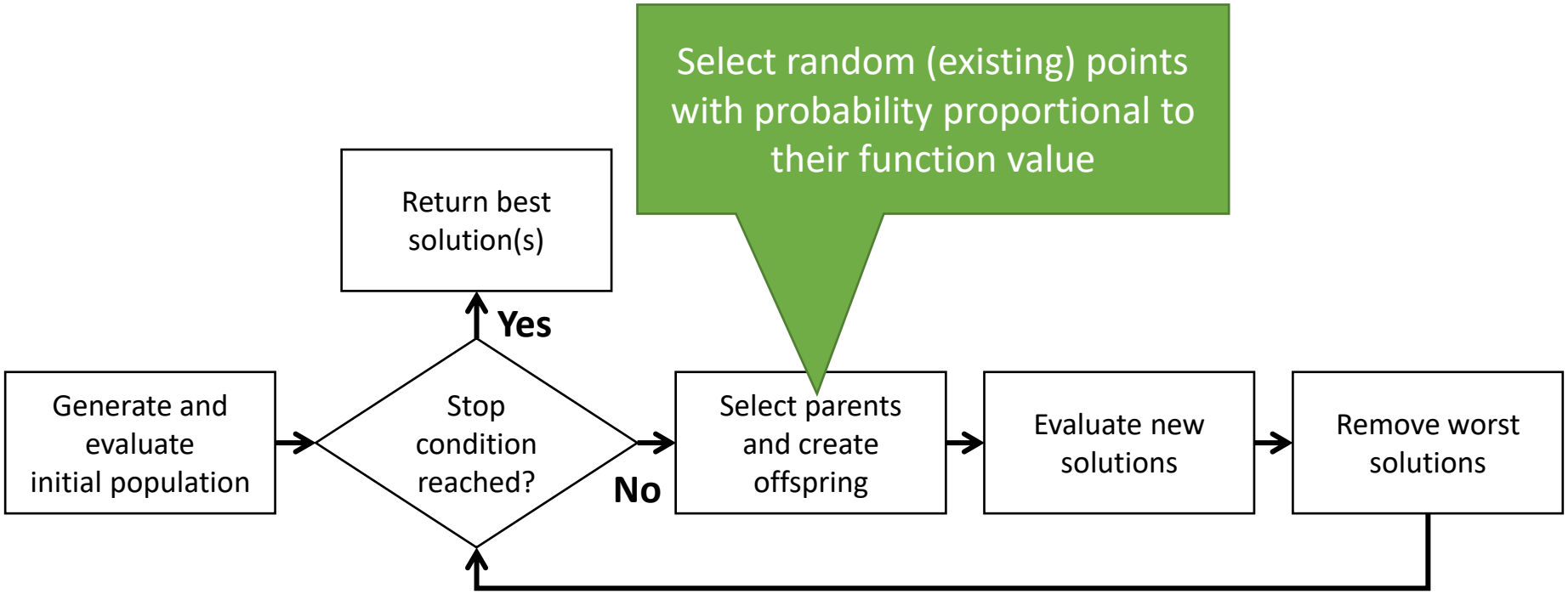
Evolutionary algorithms



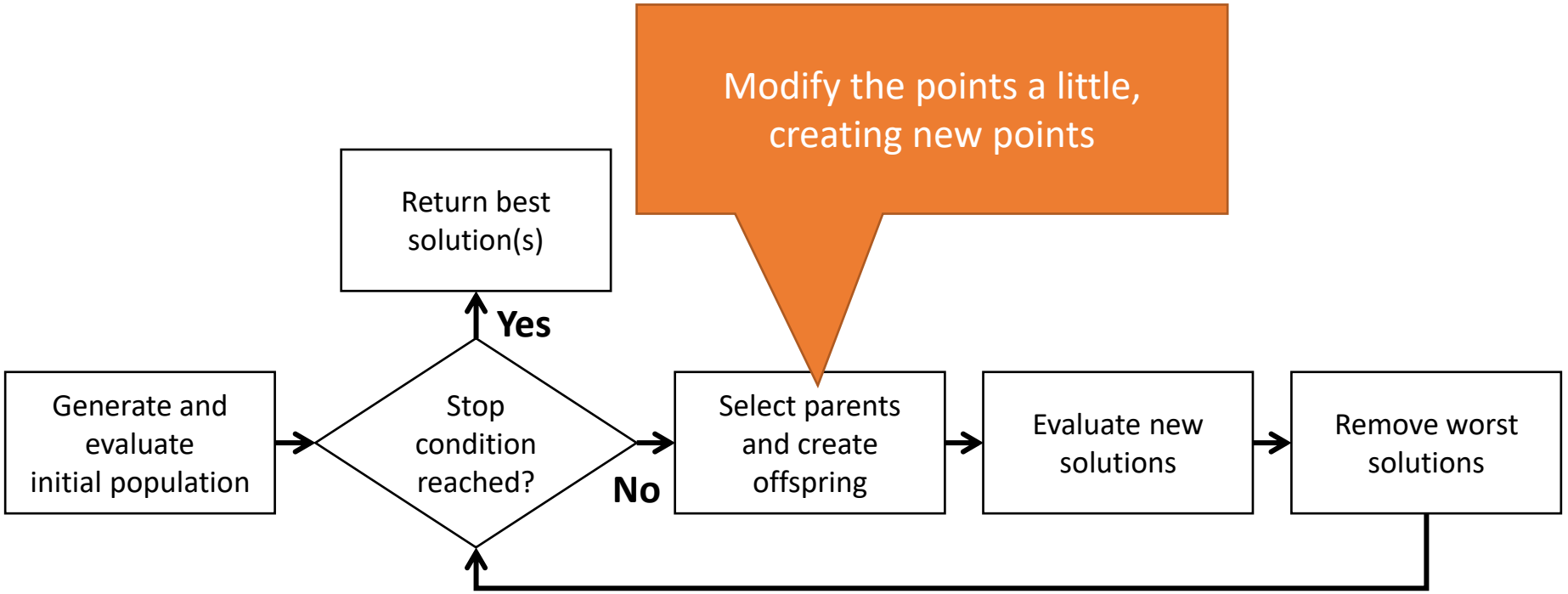
Evolutionary algorithms



Evolutionary algorithms



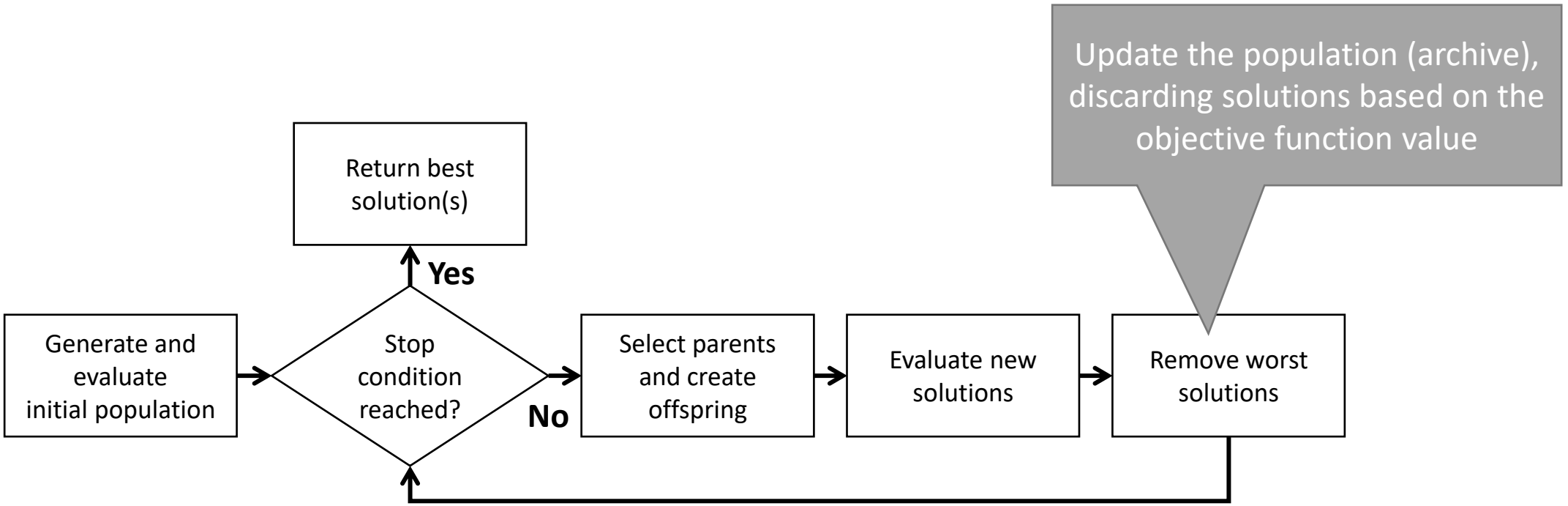
Evolutionary algorithms



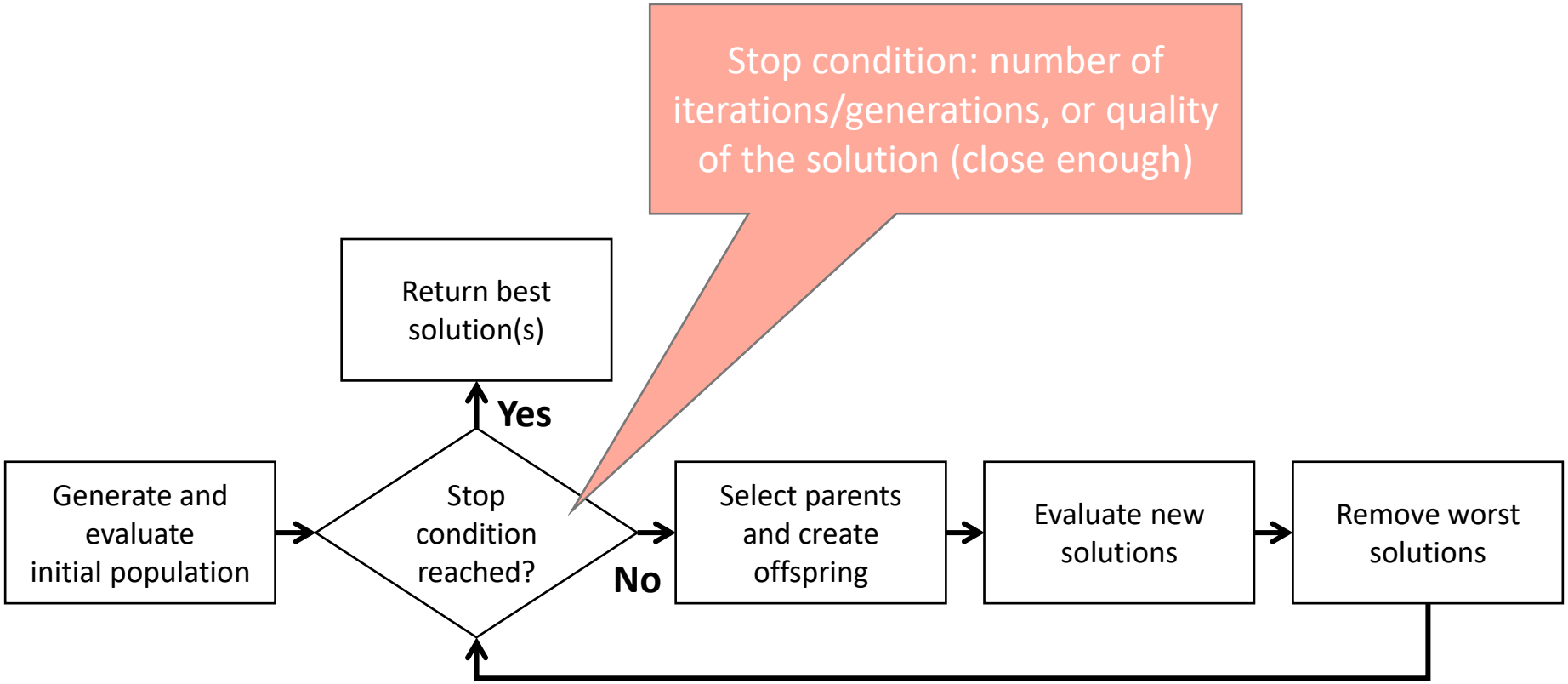
Evolutionary algorithms

- Creating new points (offspring)
- Mutation (e.g. small perturbation of coordinates)
  - [0.52, 0.24] -> [**0.55**, 0.24]
  - [0.52, 0.24] -> [0.49, 0.28]
- Crossover
  - [0.52, 0.24] and [0.19, 0.82] -> [0.52, 0.82] and [0.19, 0.24]
  - [0.52, 0.24] and [0.19, 0.82] ->
    -> [(0.52 + 0.19)/2, (0.82 + 0.24)/2] -> [0.355, 0.530]

Evolutionary algorithms

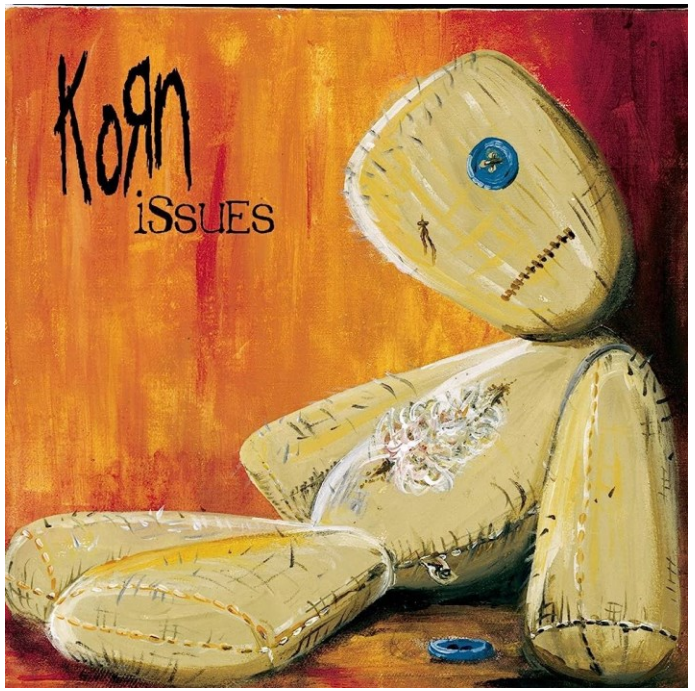


Evolutionary algorithms



Issues with stochastic techniques

- Large number of evaluations required
- Exploration vs exploitation / step size
- In general, LOTS OF (hyper)PARAMETERS

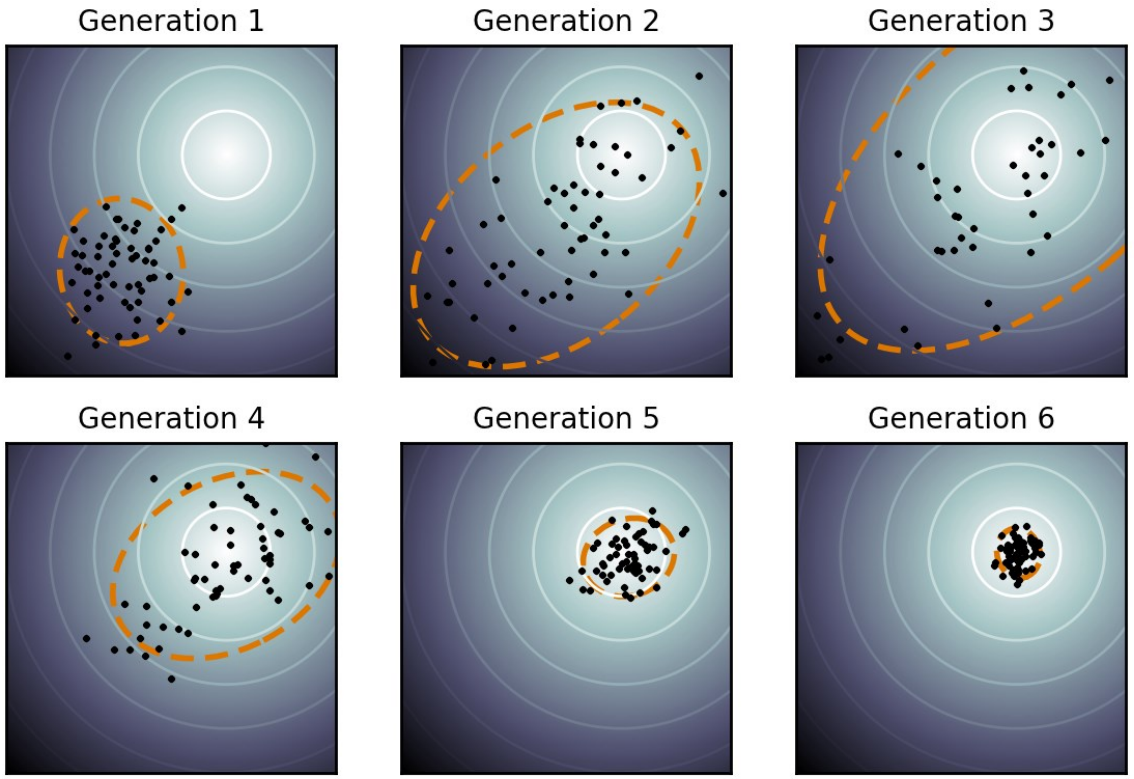


Issues with stochastic techniques

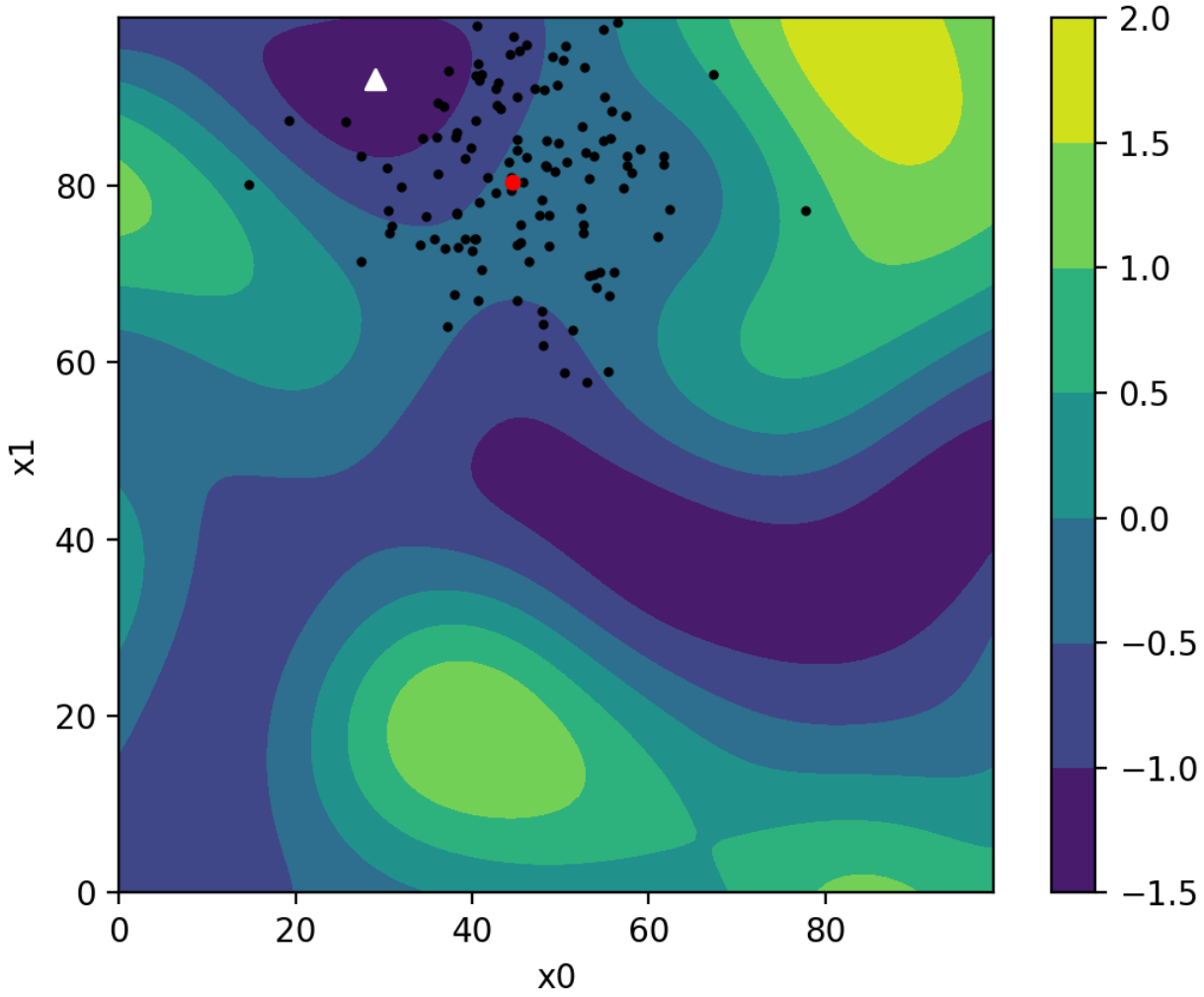
- Basin-hopping
  - All parameters of the gradient descent technique…
  - …plus number of iterations T and step size S
- Evolutionary algorithms
  - Number of iterations/generations
  - Size of the archive/population and offspring
  - How to select individuals for reproduction
  - What kind of modifications to apply, etc., etc. …
- I counted 7 for a standard EA $(G, \mu, \lambda, \tau, p_c, p_m, r)$

Modern evolutionary algorithms

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)



Modern evolutionary algorithms



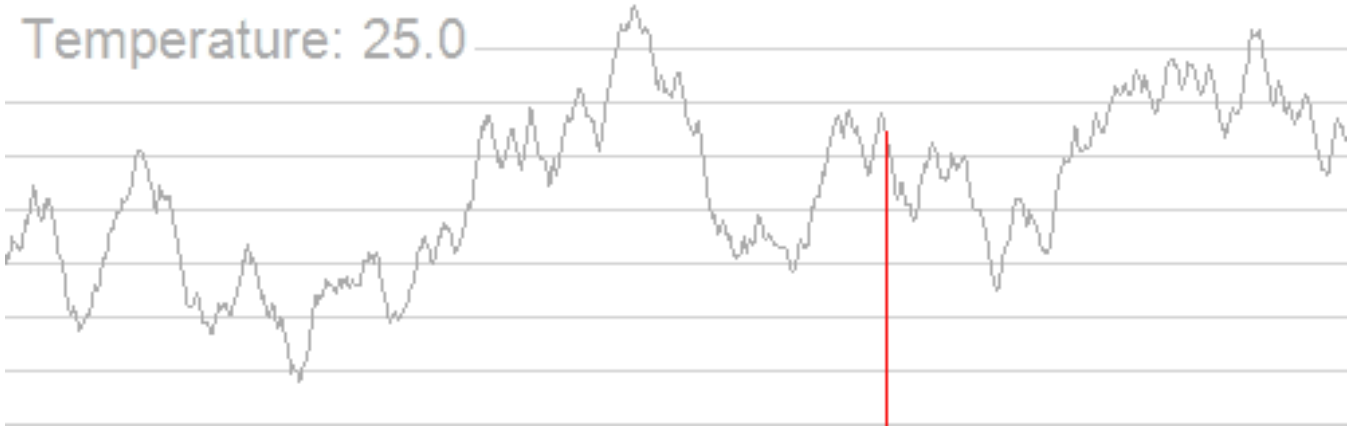
Simulated annealing

- Inspired by physical phenomenon of annealing
  - Basically an EA with a population $\mu = 1$ with mutation only
  - Strength of mutation (Temperature) starts high
  - Decreases over iterations
  - Intuition: Exploration, then Exploitation!

- This idea was copied by EAs, applied to offspring generation

Simulated annealing

- Let $s = s_0$
- For $k = 0$ through k_{max} (exclusive):
  - $T \leftarrow$ temperature(1 - $(k+1)/k_{max}$)
  - Pick a random neighbour, $s_{new} \leftarrow$ neighbour(s)
  - If $P(E(s), E(s_{new}), T) \geq$ random(0, 1):
    - $s \leftarrow s_{new}$
- Output: the final state s



Questions?

Bibliography

- Kochenderfer & Wheeler, *Algorithms for Optimization*, MIT Press, 2019
- De Jong, *Evolutionary Computation: A Unified Approach*, 2016

Images: unless otherwise stated, I stole them from the Internet. I hope they are not copyrighted, or that their use falls under the Fair Use clause, and if not, I am sorry. Please don’t sue me.