Data-driven design & fabrication methods: Design Encoding & Search

Prof. Josie Hughes, CREATE Lab

Data Driven Methods: What to use when

Heuristic Search Techniques

Genetic Algorithms (Holland – 1975)
Inspired by genetics and natural selection – max fitness

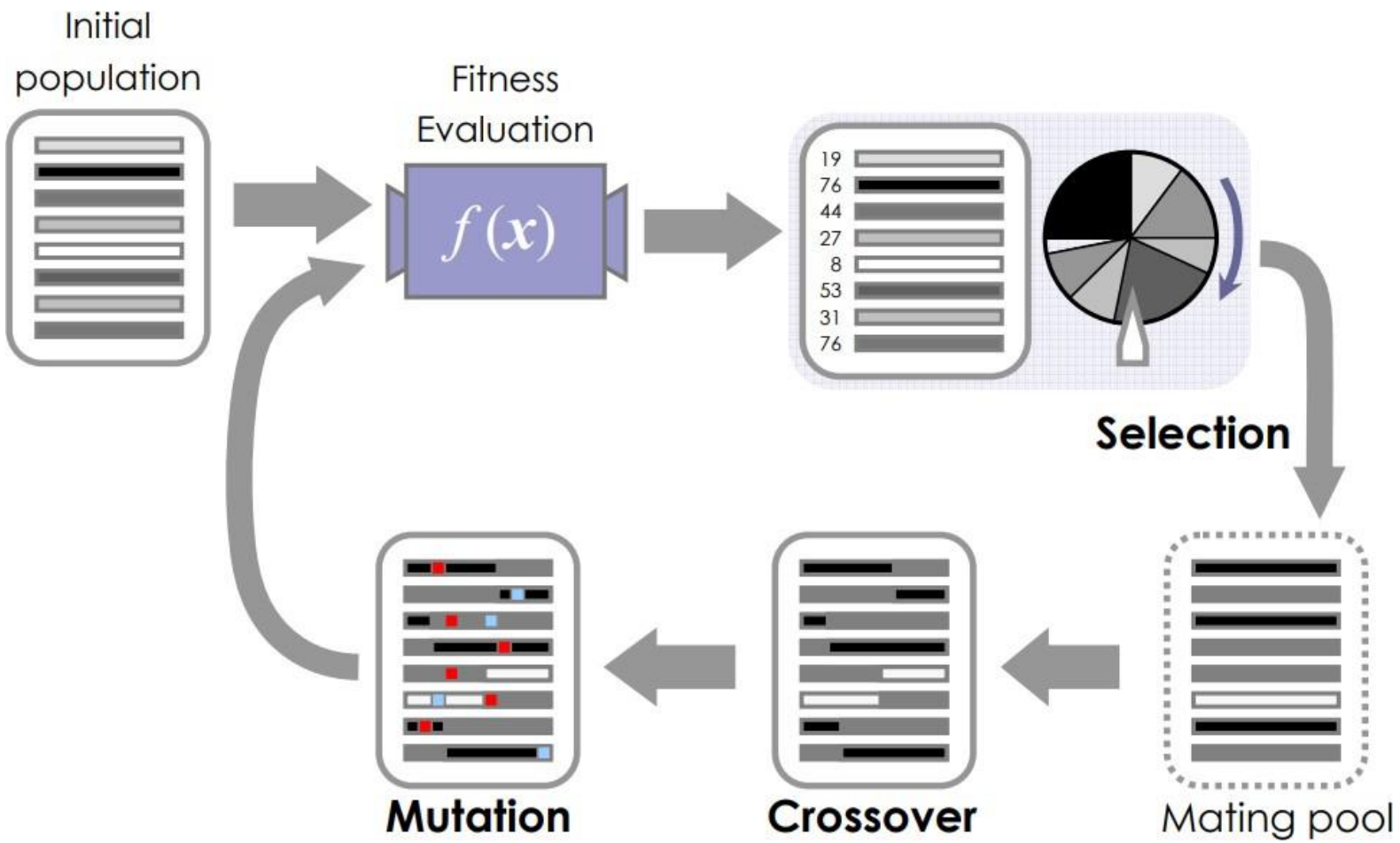
Simulated Annealing (Kirkpatrick – 1983)
Inspired by statistical mechanics– min energy

Partical Swarm Operators - Kennedy - 1995) Inspired by the social behavior of swarms of insects or flocks of birds
– max “food”

Bayesian Optimization

Different Acquisition Functions

GAs: Summary



Specific characteristics:

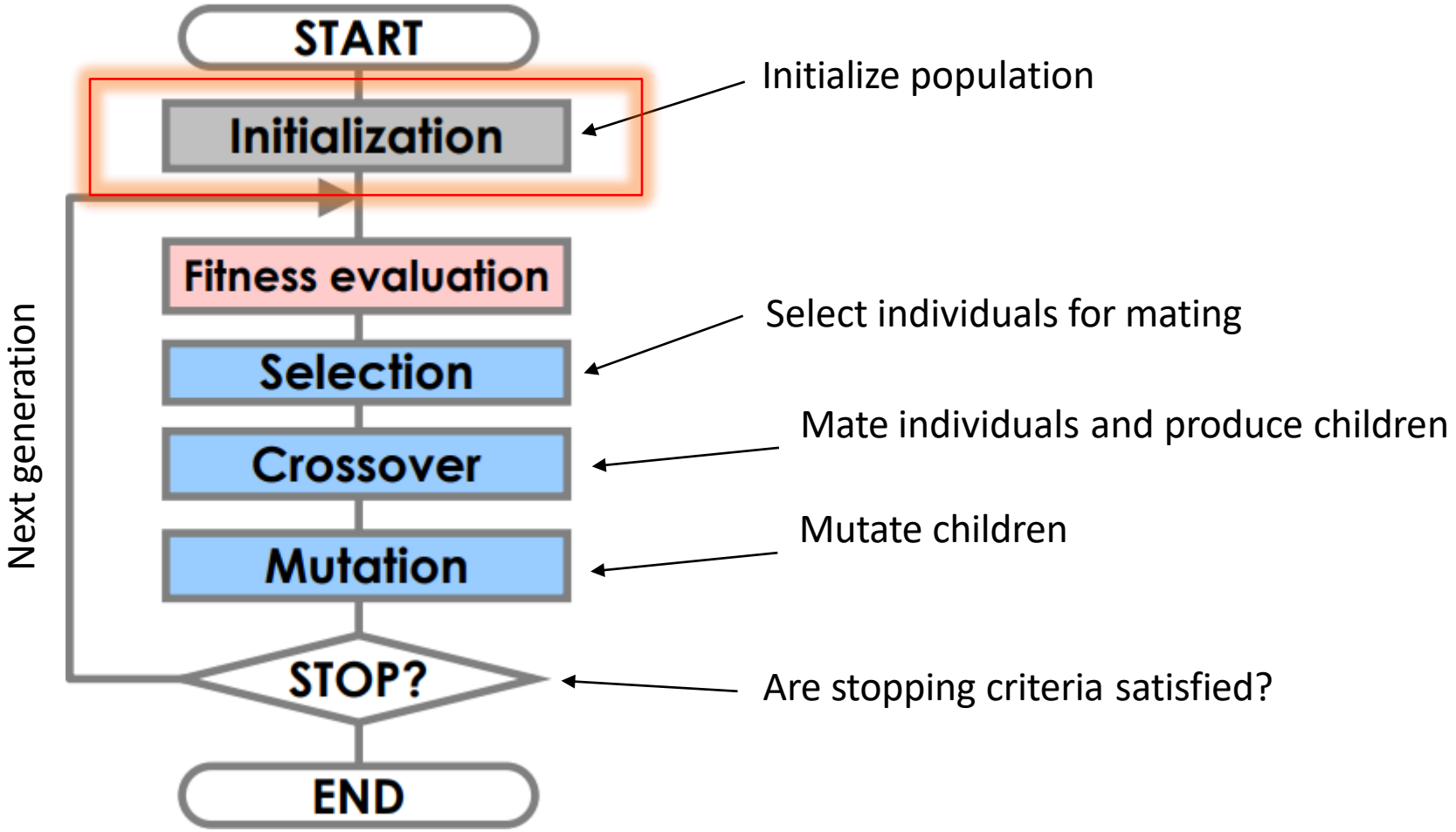
- Binary representation
- Fixed string length
- Fitness proportional selection operator
- Single-point crossover operator
- Gene-wise mutation operator

GAs: Summary

Specifics of GAs versus other 'traditional' methods

- GAs search a population of points in parallel, not only a single point
- GAs use probabilistic transition rules, not deterministic ones
- GAs work on an encoding of the design variable set rather than on the variables themselves
- GAs do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

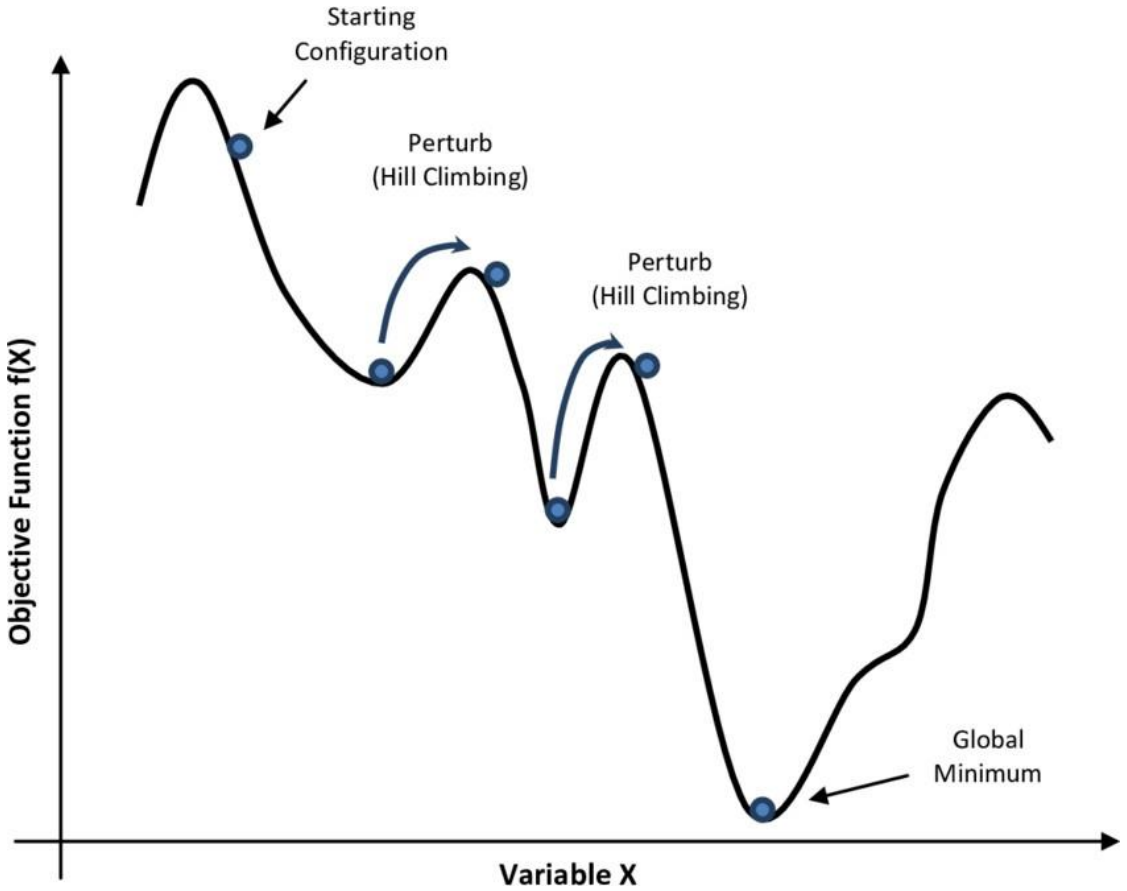
Genetic Algorithm



Simulated Annealing

- Start in a randomized state (T high)
- **Find ‘close’ perturbed solutions** – accepting moves that decrease the energy - find the solution with the minimum energy at this state – equilibrium is reached
- Decrease the temperature
  - Sometimes accepting less bad configurations at each temperature (enables hill-climbing)
  - To decide if good/bad use Boltzmann distribution, P(r,t) =exp(-E/T)

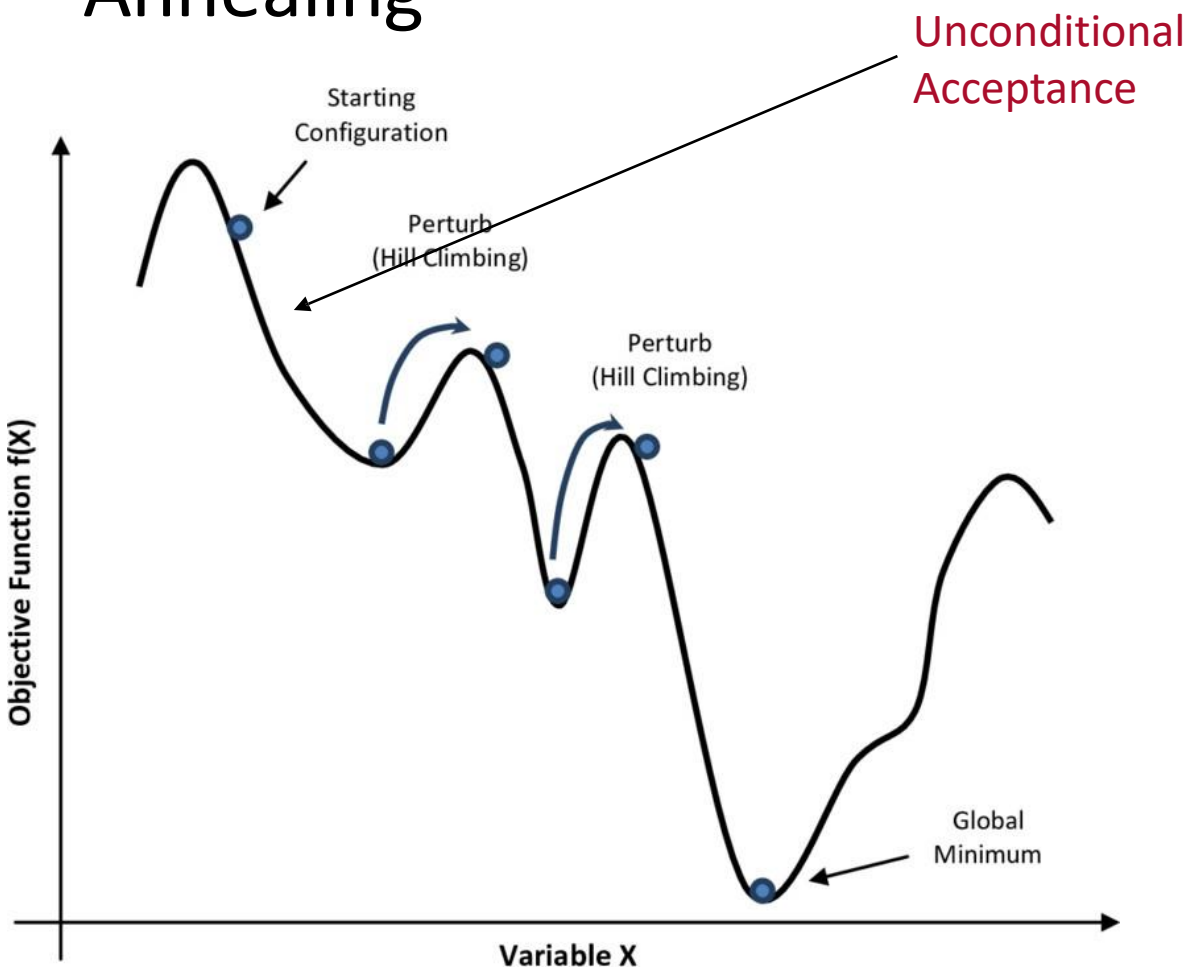
Simulated Annealing



S = solution, R = perbuted solution

- *If R is better than S, we'll always replace S with R as usual. B*
- *ut if R is worse than S, we may still replace S with R with a certain probability*

Simulated Annealing

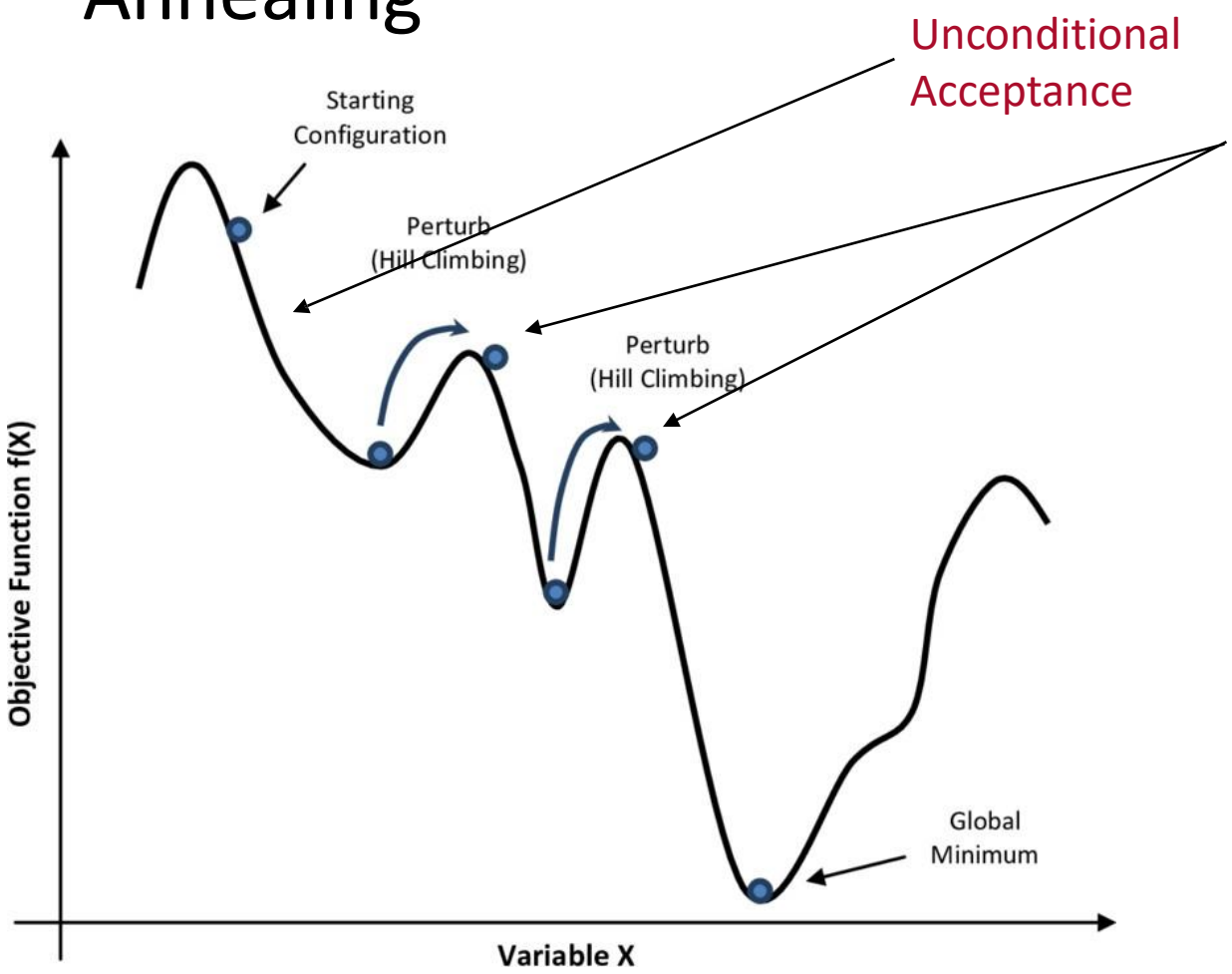


S = solution, R = perbuted solution

- *If R is better than S, we'll always replace S with R as usual.*

  But if R is worse than S, we may still replace S with R with a certain probability

Simulated Annealing

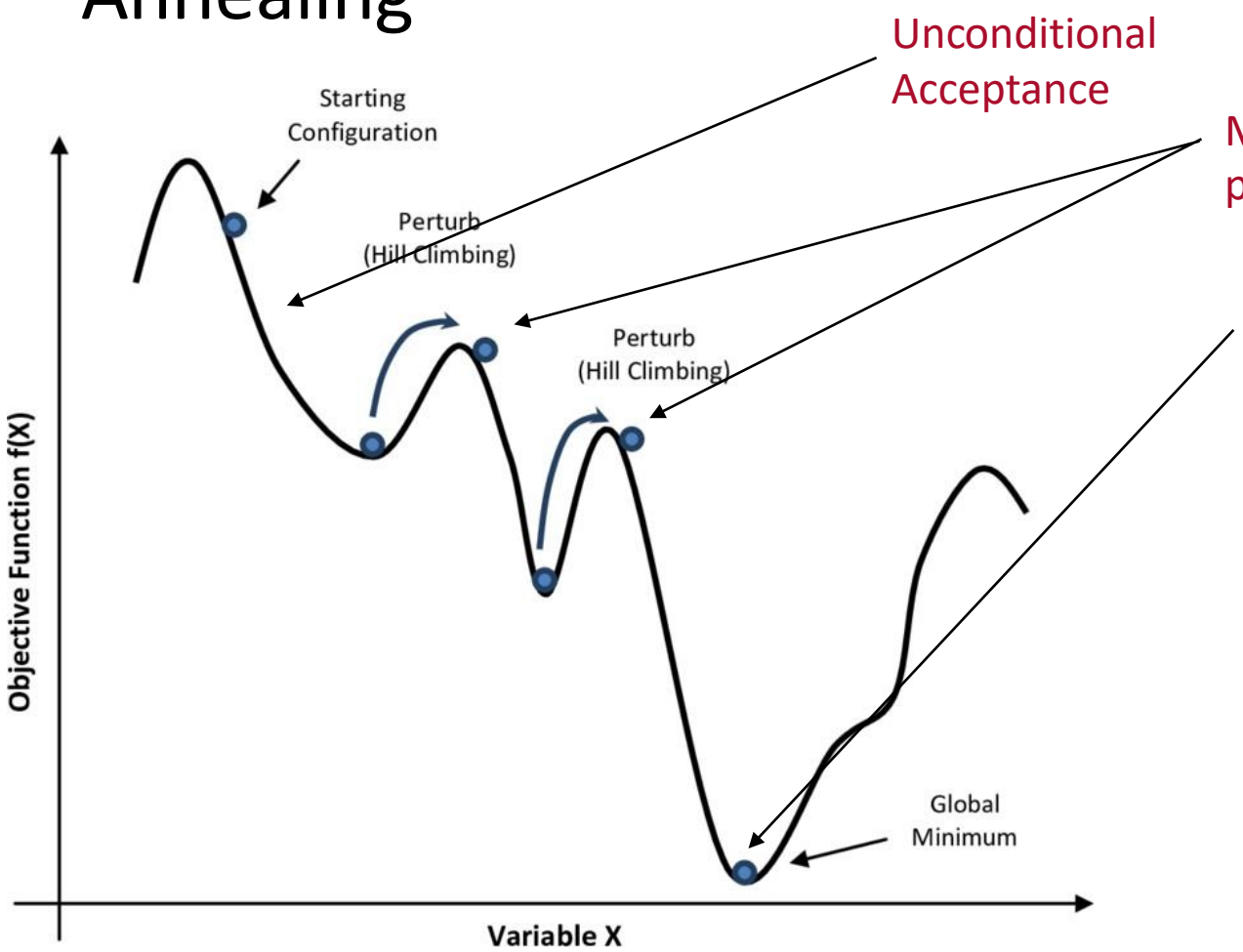


There is a probability it might be a less bad solution

S = solution, R = perbuted solution

- *If R is better than S, we'll always replace S with R as usual. B*
- *ut if R is worse than S, we may still replace S with R with a certain probability*

Simulated Annealing



Unconditional Acceptance

Move accepted with probability exp(-E/T)

Found at final temperature

S = solution, R = perbuted solution

- *If R is better than S, we'll always replace S with R as usual. B*
- *ut if R is worse than S, we may still replace S with R with a certain probability*

Simulated Annealing

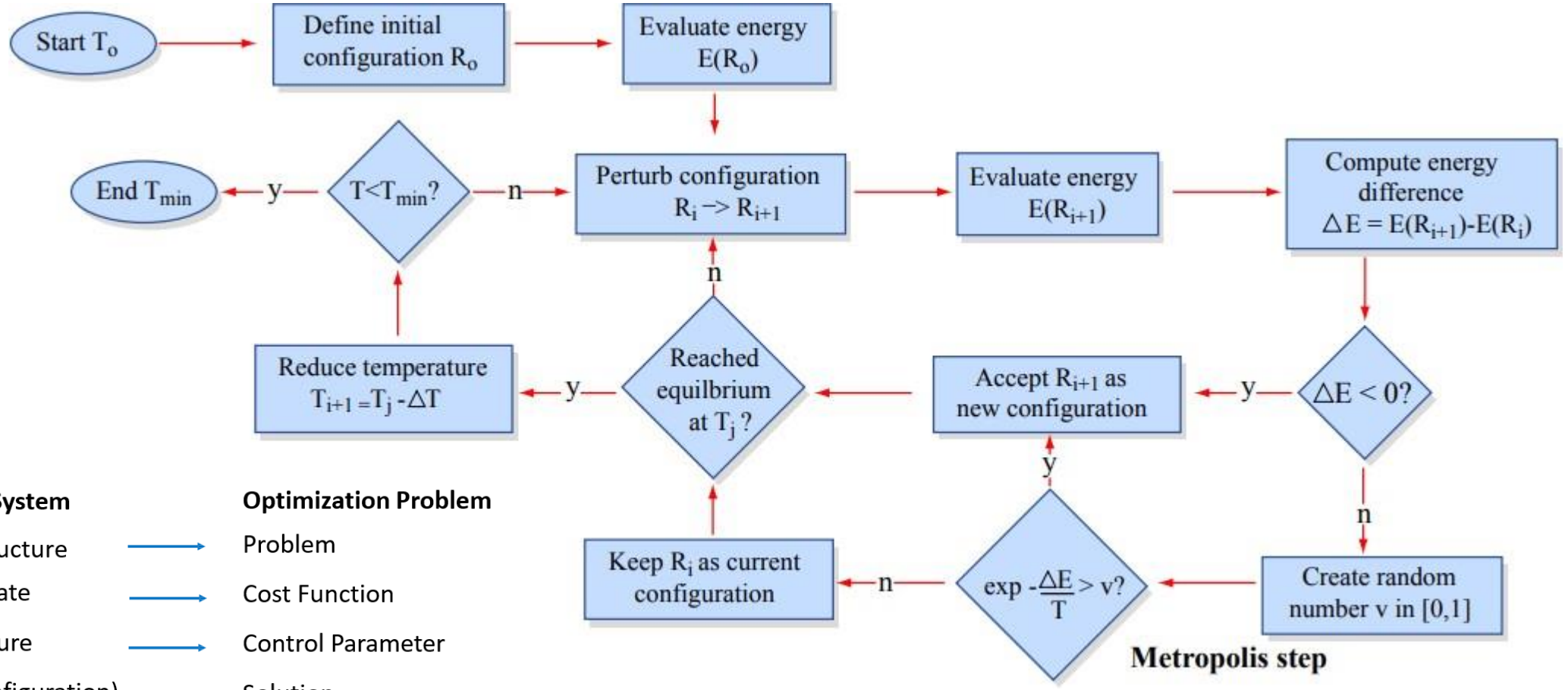
1. How do we define equilibrium?
   - When we cannot yield any significant improvements after certain number of loops
   - A constant number of loops has been performed

1. How do we calculate the new temperature for each step?
   - A constant value is subtracted to get a new temperature, T’ = T –Td
   - A constant scale factor is used to get a new temperature, T’ = T*Rd

   → Scale factor normally can achieve better performance

Problem specific and may need to be tuned

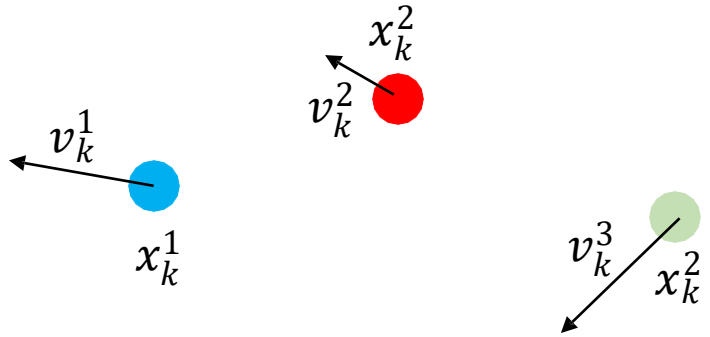
Simulated Annealing: Block Diagram



Physical System		Optimization Problem
Metal Structure	→	Problem
Energy State	→	Cost Function
Temperature	→	Control Parameter
State (configuration)	→	Solution
Careful Annealing	→	Simulated Annealing

Concepts for a PSO Algorithm

- Particle Description: each particle has three features
  - Position $\mathbf{x}_k^i$ (this is the i^{th} particle at time k, notice vector notation)
  - Velocity $\mathbf{v}_k^i$ (similar to search direction, used to update the position)
  - Fitness or objective $f\left(\mathbf{x}_k^i\right)$ (determines which particle has the best value in the swarm and also determines the best position of each particle over time.



Concepts for a PSO Algorithm

- Initial Swarm
  - No well established guidelines for swarm size, normally 10 to 60.
  - particles are randomly distributed across the design space.

$$\mathbf{x}_0^i = \mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)$$

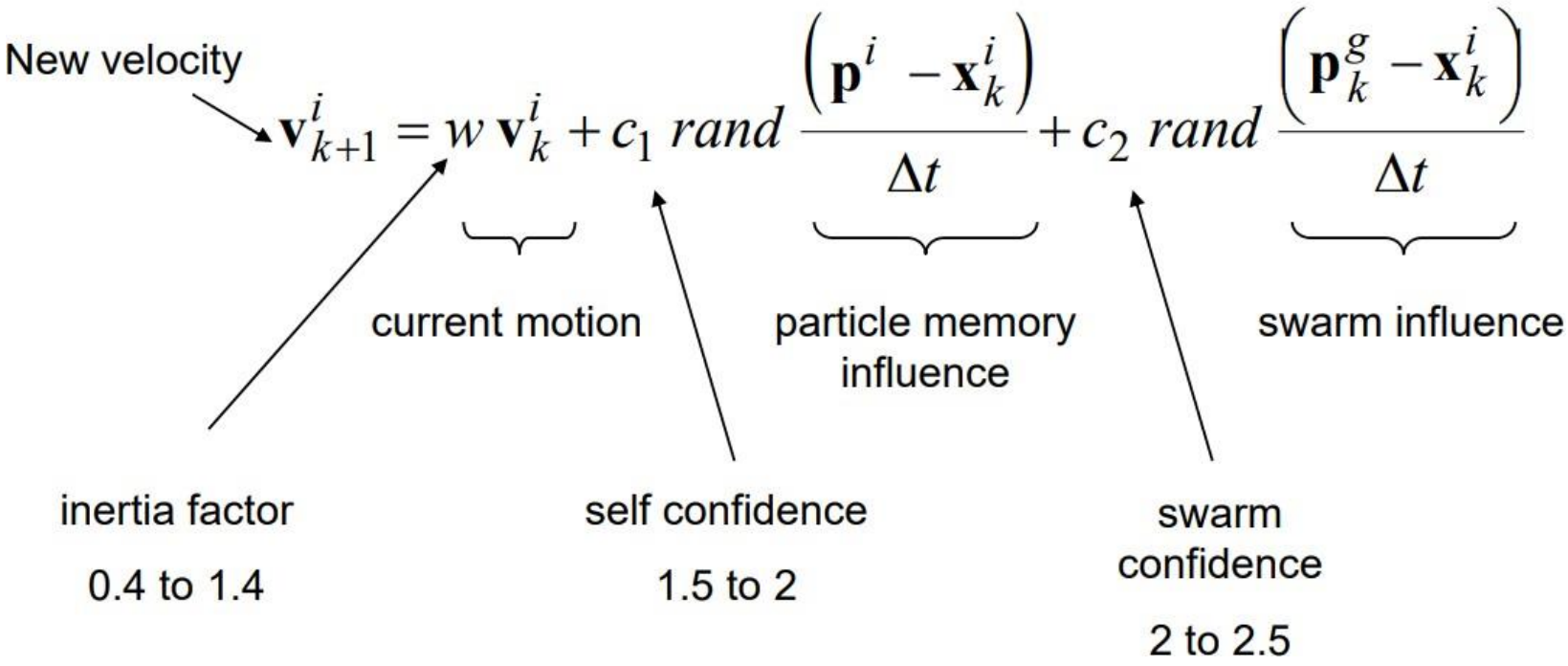
where $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ are vectors of lower and upper limit values respectively.

  - Evaluate the fitness of each particle and store:
    - particle best ever position (particle memory $\mathbf{p}^i$ here is same as $\mathbf{x}_0^i$)
    - Best position in current swarm (influence of swarm $\mathbf{p}_0^g$)
  - Initial velocity is randomly generated.

$$\mathbf{v}_0^i = \frac{\mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)}{\Delta t} = \frac{\text{position}}{\text{time}}$$

Concepts for a PSO Algorithm

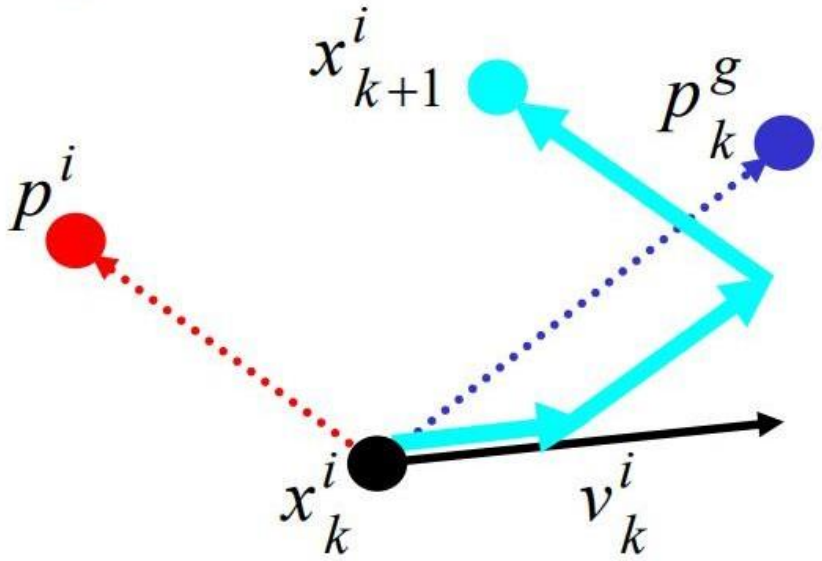
- Velocity Update
  - provides search directions
  - Includes deterministic and probabilistic parameters.
  - Combines effect of current motion, particle own memory, and swarm influence.



Concepts for a PSO Algorithm

- Position Update
  - Position is updated by velocity vector.

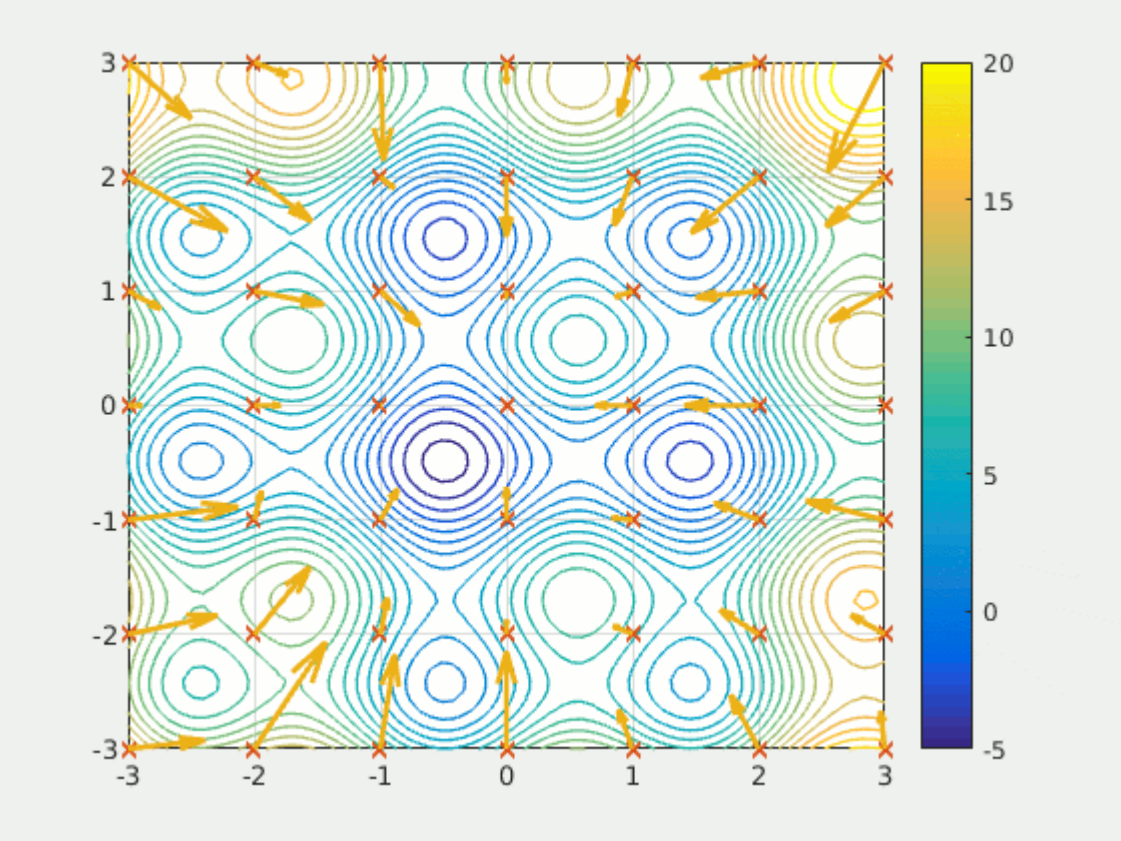
$$\mathbf{x}_{k+1}^{i} = \mathbf{x}_{k}^{i} + \mathbf{v}_{k+1}^{i}\Delta t$$

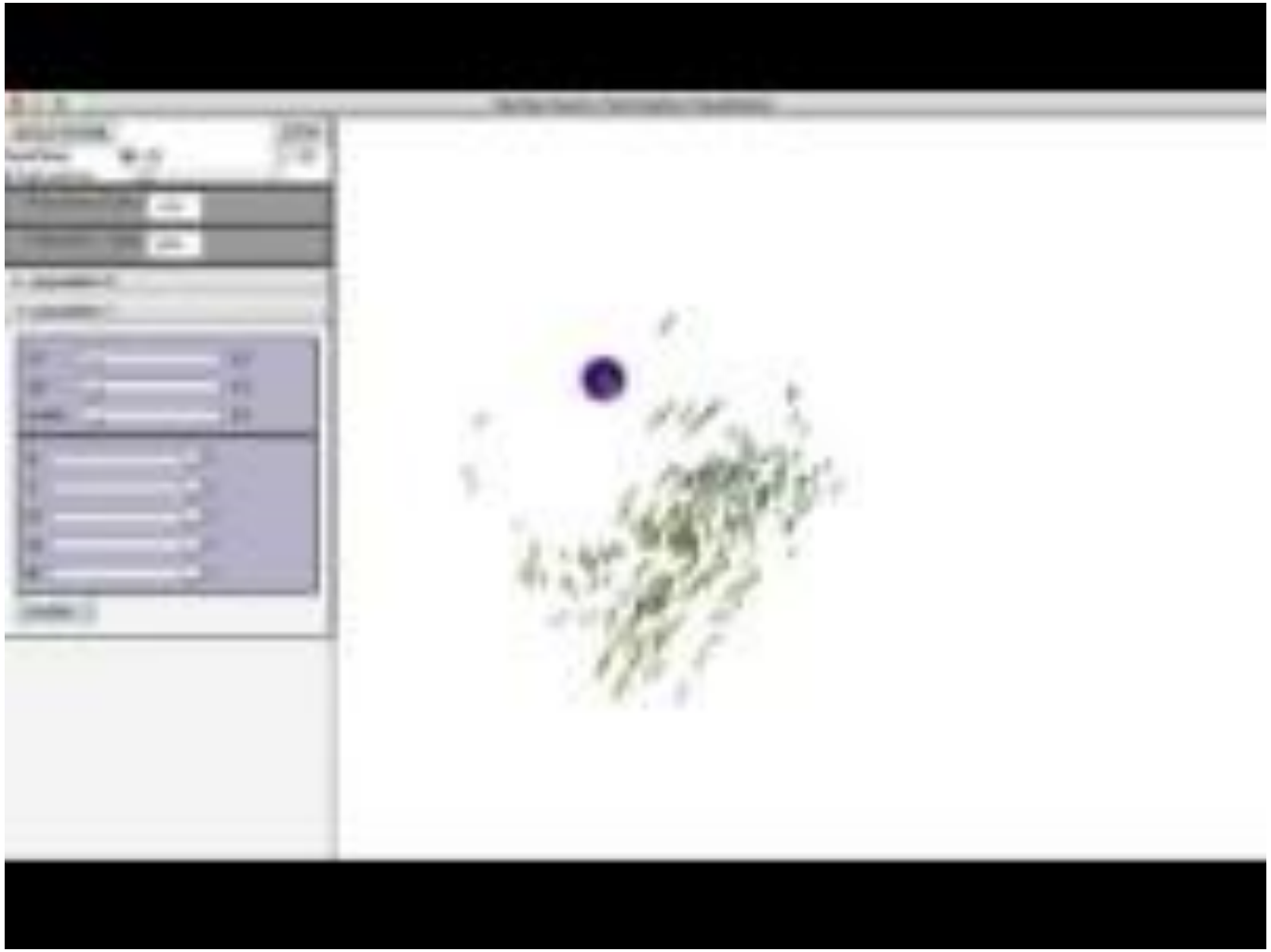


- Stopping Criteria
  - Maximum change in best fitness smaller than specified tolerance for a specified number of moves (S).

$$\left| f\left(\mathbf{p}_{k}^{g}\right) - f\left(\mathbf{p}_{k-q}^{g}\right) \right| \le \varepsilon \quad q = 1,2,..S$$

Demonstration of PSO





Heuristics: Summary

	GAs	SA	PSO
Population or Individual	Population	Individual	Population
Bounds	No clear definition of neighbourhood	Defined and explicit. Not all the search space is assumed to be important	No clear definition of neighbourhood
Data-types	An encode anything	Continous	Continous
Tuning	Many parameters	Few parameters	Many parameters
Memory	No memory, but implicitly keeps track of good performance through elitism	No memory, only used after each temperature reset	Uses velocity and position
Probablistic Operators	Crossover and mutation	Metropolis Step	Deterministic and probabilistic components
Speed	Can get stuck in local minimum, may not yield the global optimum	Typically the fastest, but may only get a solution ‘near’ to the optimal	Hard to tune, can be efficienc

Heuristics: Summary

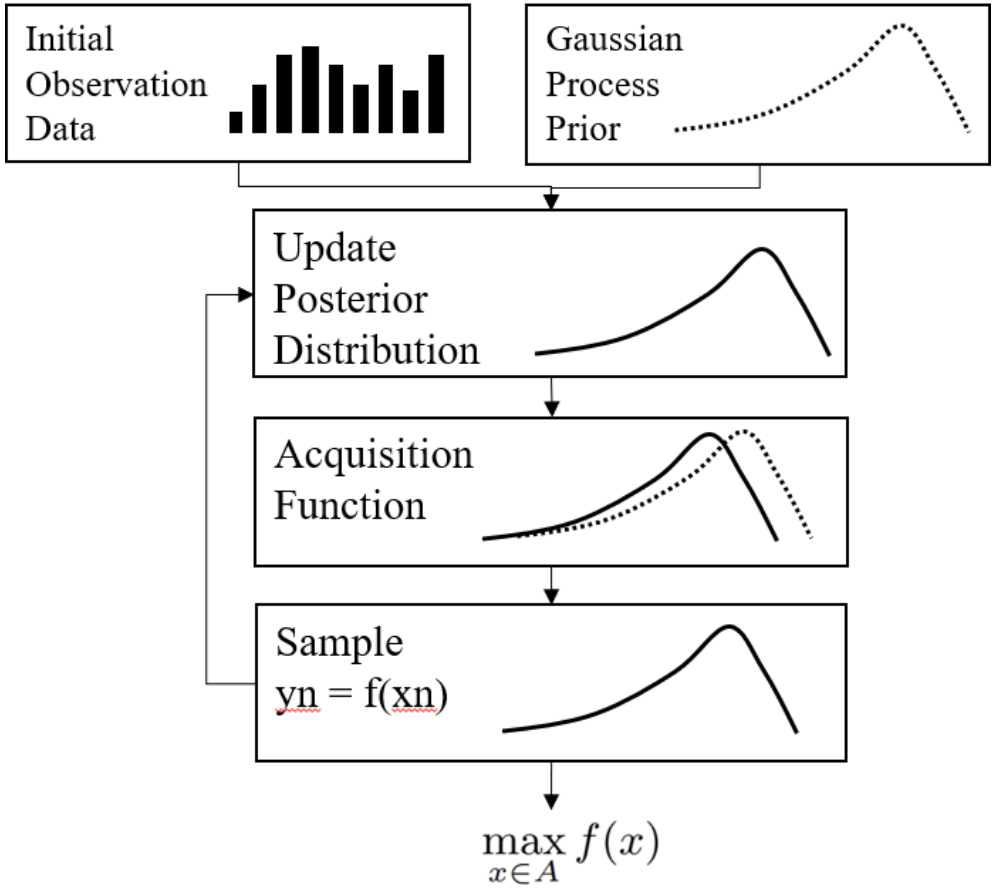
- No model/surrogate built
- No estimate of uncertainty
- No guarantee of optimality

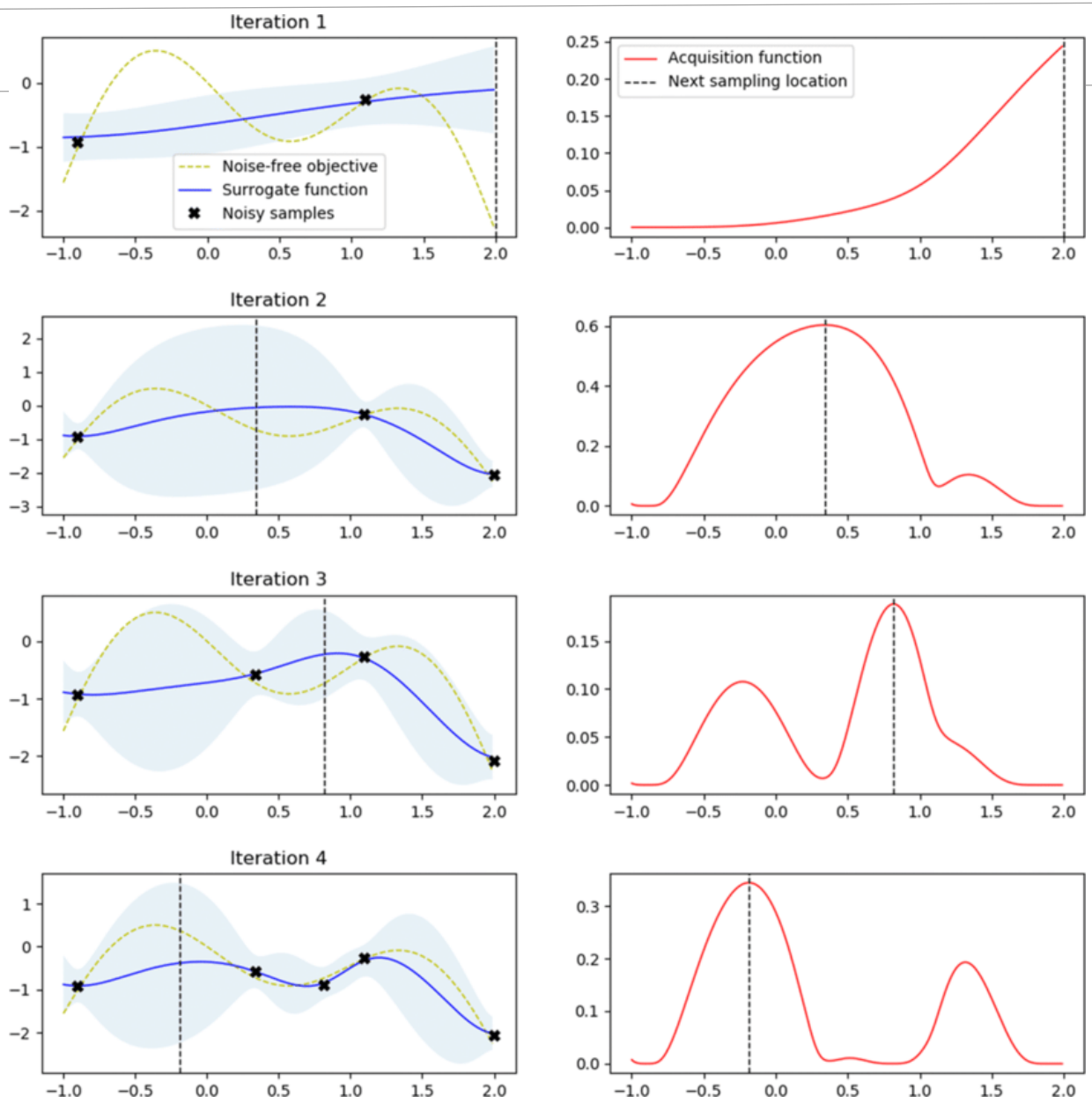
→ But can operate in very high dimensionality search spaces

Bayesian Optimization

Bayesian Optimization

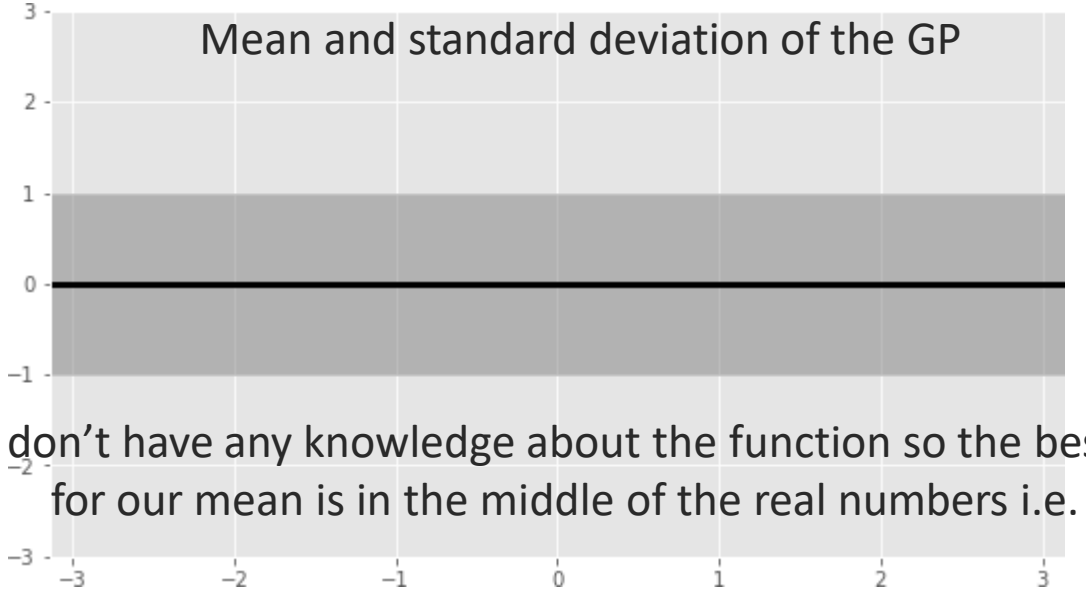
Global optimization of multimodal black-box function.



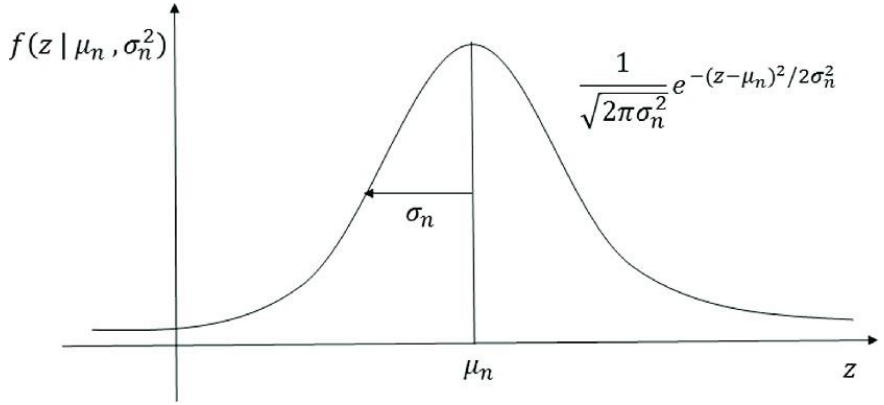


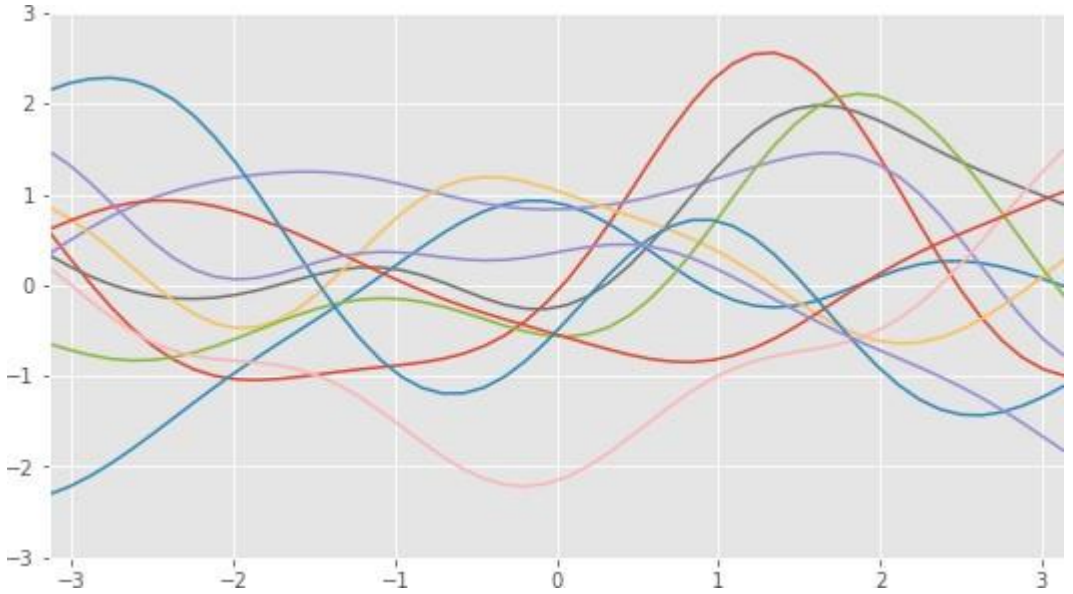
Gaussian Processes

Our **prior belief** about an unknown function



we don't have any knowledge about the function so the best guess for our mean is in the middle of the real numbers i.e. 0.

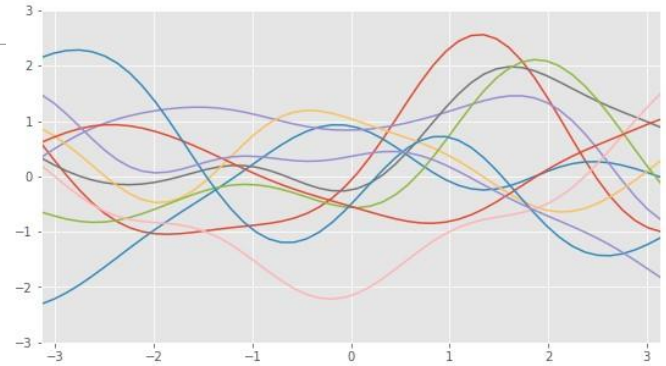


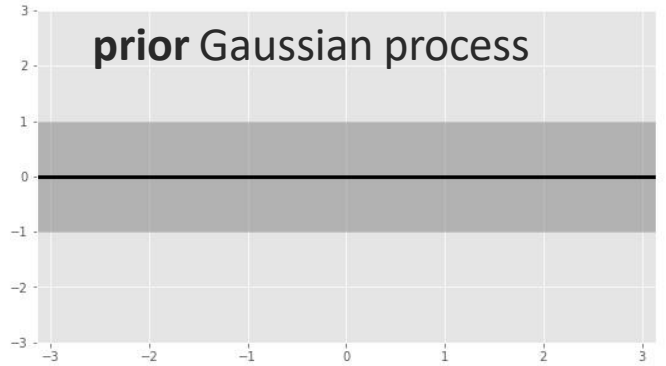


There is a wide range of possible functions and diverse function shapes on display…

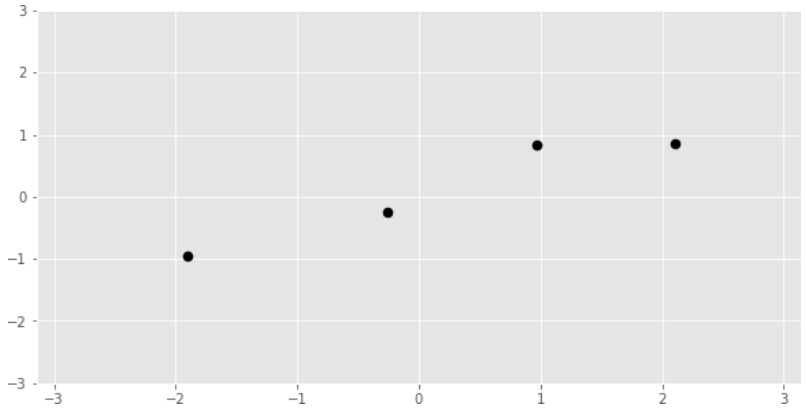
Gaussian Processes

Prior Belief

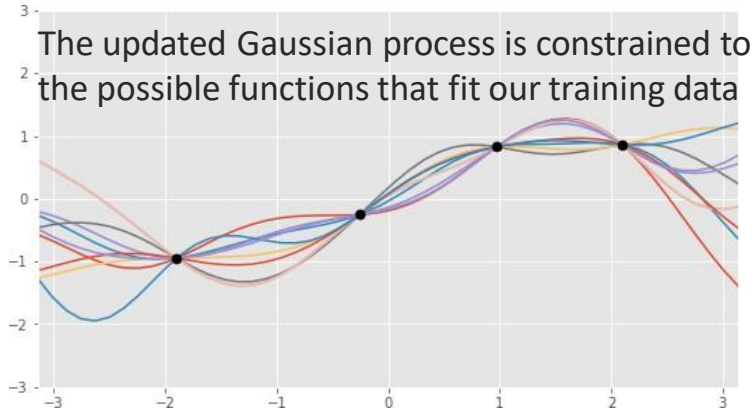




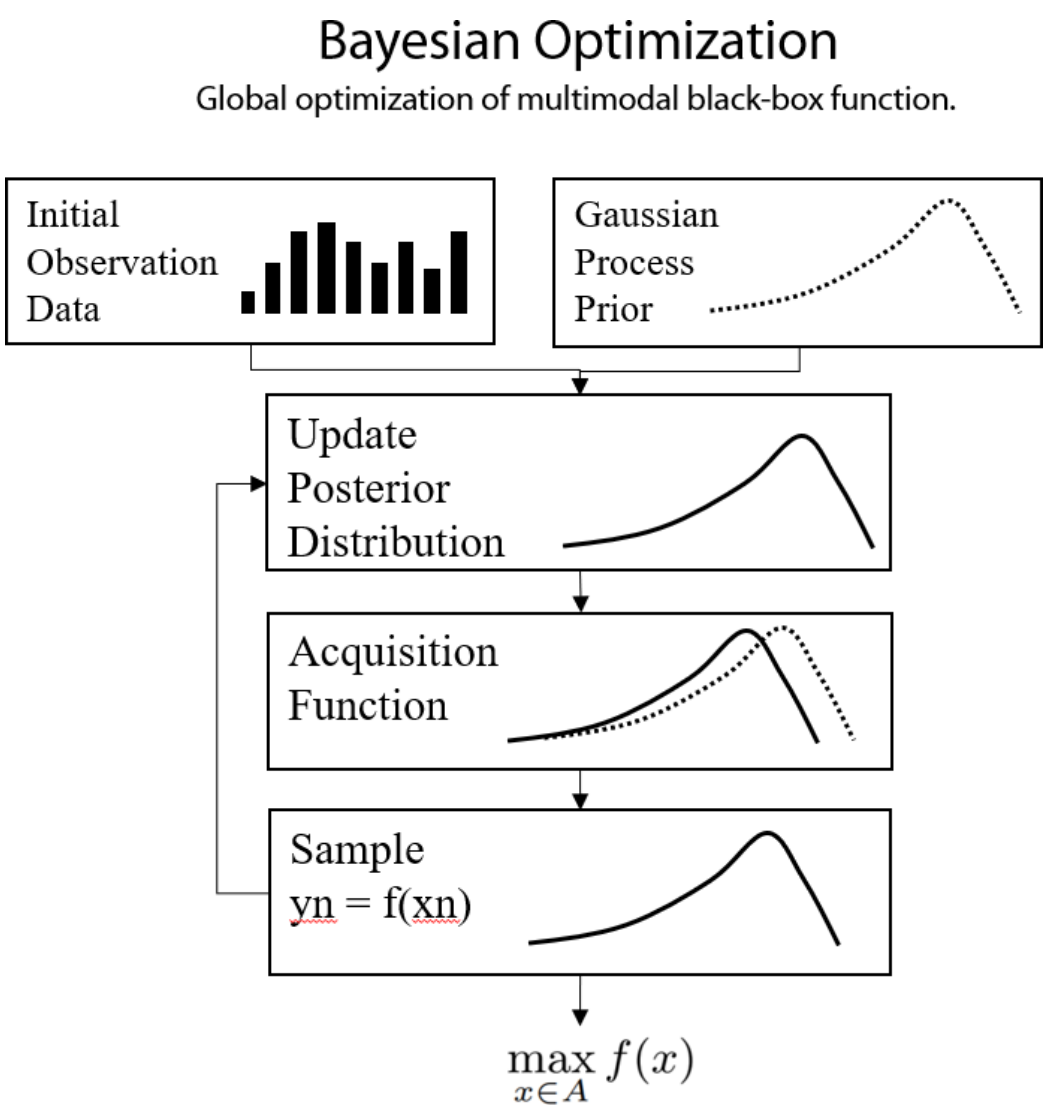
Training Data: observe some outputs of the unknown function at various points



Use Bayes' rule to update our belief about the function to get the **posterior** Gaussian process



Formalizing This Approach: Bayesian Optimization



1. We first choose a surrogate model for modeling the true

- function f*f* and define its **prior**.

2. Given the set of **observations** (function evaluations), use Bayes rule to obtain the **posterior**.

3. Use an acquisition function which is a function of the posterior, to decide the next sample point

- 1. Add newly sampled data to the set of **observations** and goto step #2 till convergence or budget elapses.

The learning/utility/aquisition function

The utility should represent our design goal:.

1. **Active Learning and experimental design**: reduce the uncertainty in the model (prediction or hyper-parameters).

2. **Optimization**: Minimize the loss in a sequence $x_1, \ldots, x_n$

$$r_N = \sum_{n=1}^{N} f(x_n) - N f(x_M)$$

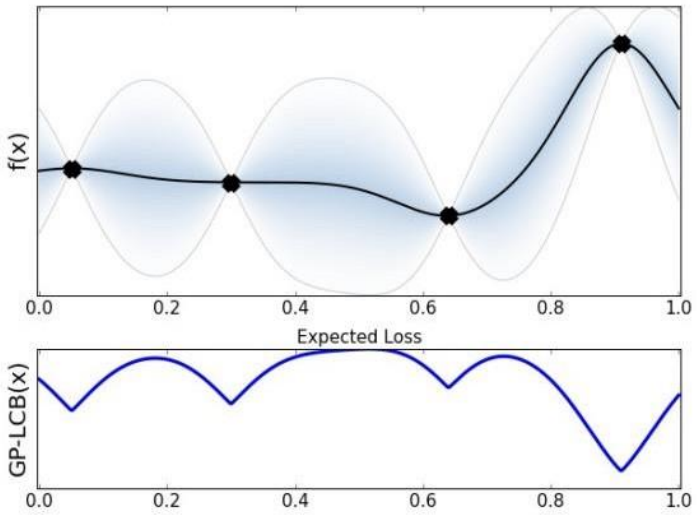
(1) does to a lot exploration whereas (2) encourages exploitation about the minimum of f.

What options do we have for this acquisition function?

Upper (lower) confidence band/

Direct balance between exploration and exploitation:

$$\alpha_{LCB}(\mathbf{x};\theta,\mathcal{D}) = -\mu(\mathbf{x};\theta,\mathcal{D}) + \beta_t \sigma(\mathbf{x};\theta,\mathcal{D})$$



- In noiseless cases, it is a lower bound of the function to minimize.
- This allows to computer a bound on how close we are to the minimum.
- Optimal choices available for the 'regularization parameter'.

Theorem 1 *Let* $\delta \in (0,1)$ *and* $\beta_t = 2\log(|D|t^2\pi^2/6\delta)$. *Running GP-UCB with* β_t *for a sample* f *of a GP with mean function zero and covariance function* $k(\boldsymbol{x},\boldsymbol{x}')$, *we obtain a regret bound of* $\mathcal{O}^*(\sqrt{T\gamma_T \log|D|})$ *with high probability. Precisely, with* $C_1 = 8/\log(1+\sigma^{-2})$ *we have*

$$\Pr\left\{R_T \leq \sqrt{C_1 T \beta_T \gamma_T} \quad \forall T \geq 1\right\} \geq 1-\delta.$$

Probability of Improvement

This acquisition function chooses the next query point as the one which has the highest *probability of improvement* over the current max $f(x^+)$. Mathematically, we write the selection of next point as follows,

$$x_{t+1} = argmax(\alpha_{PI}(x)) = argmax(P(f(x) \geq (f(x^+) + \epsilon)))$$

where,

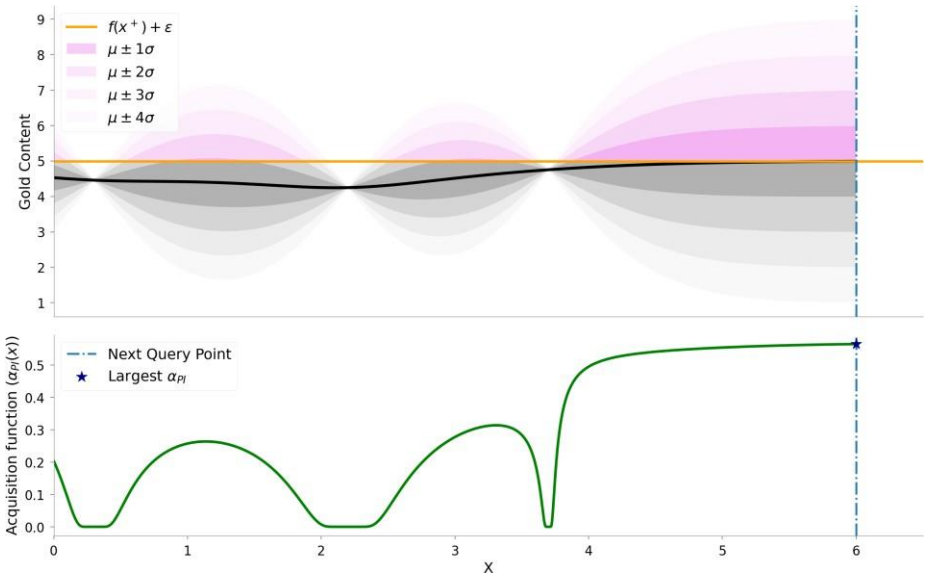
$P(\cdot)$ indicates probability

ϵ is a small positive number

And, $x^+ = \text{argmax}_{x_i \in x_{1:t}} f(x_i)$ where x_i is the location queried at i^{th} time step.

Looking closely, we are just finding the upper-tail probability (or the CDF) of the surrogate posterior. Moreover, if we are using a GP as a surrogate the expression above converts to,

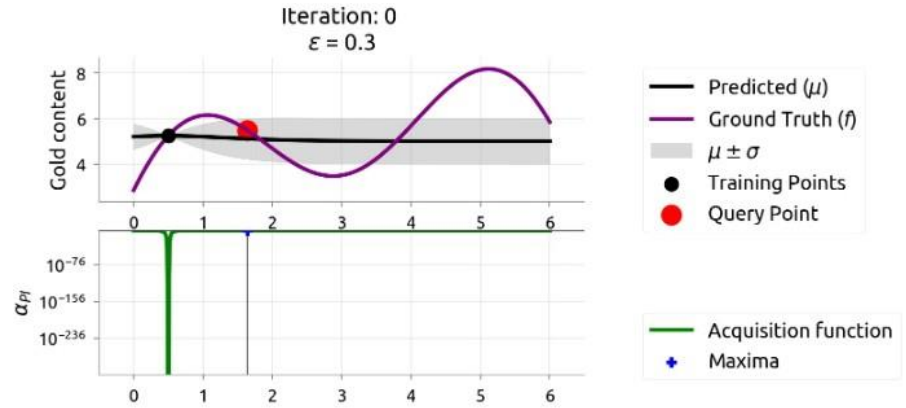
$$x_{t+1} = argmax_x \Phi\left(\frac{\mu_t(x) - f(x^+) - \epsilon}{\sigma_t(x)}\right)$$

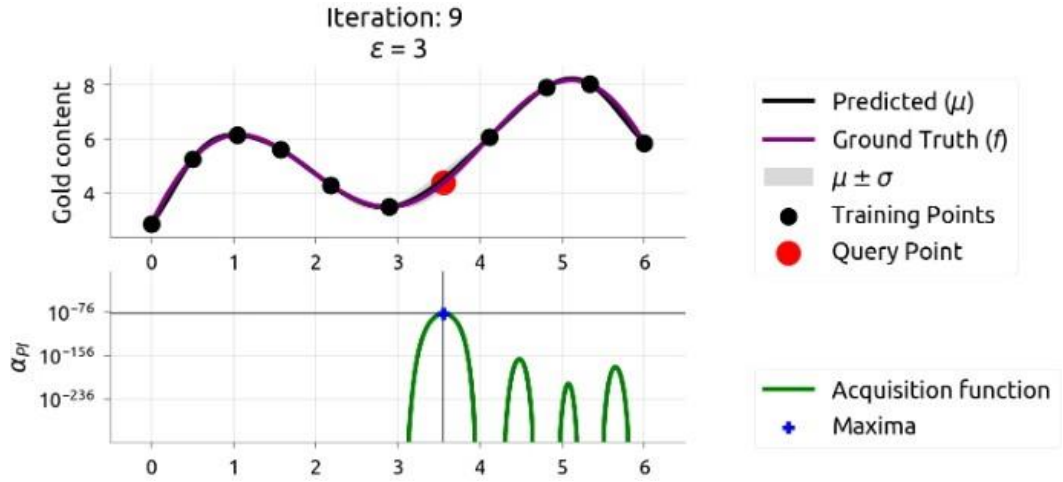


The grey regions show the probability density below the current max. The “area” of the violet region at each point represents the “probability of improvement over current maximum”.

Probability of Improvement

- PI uses ϵ to strike a balance between exploration and exploitation.
- Increasing ϵ results in querying locations with a larger σ as their probability density is spread.



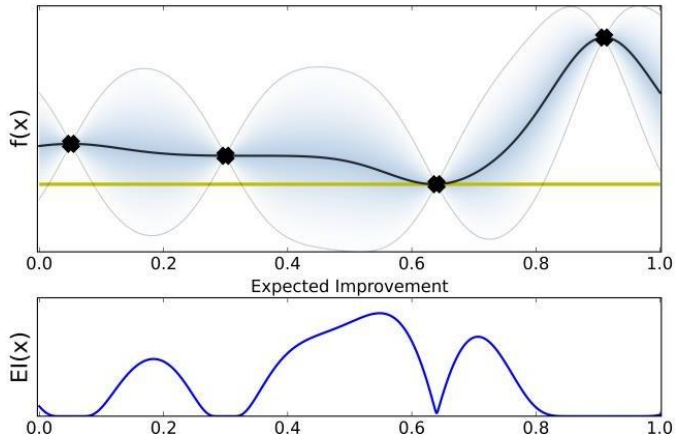


We are unable to exploit when we land near the global maximum. Moreover, with high exploration, the setting becomes similar to active learning.

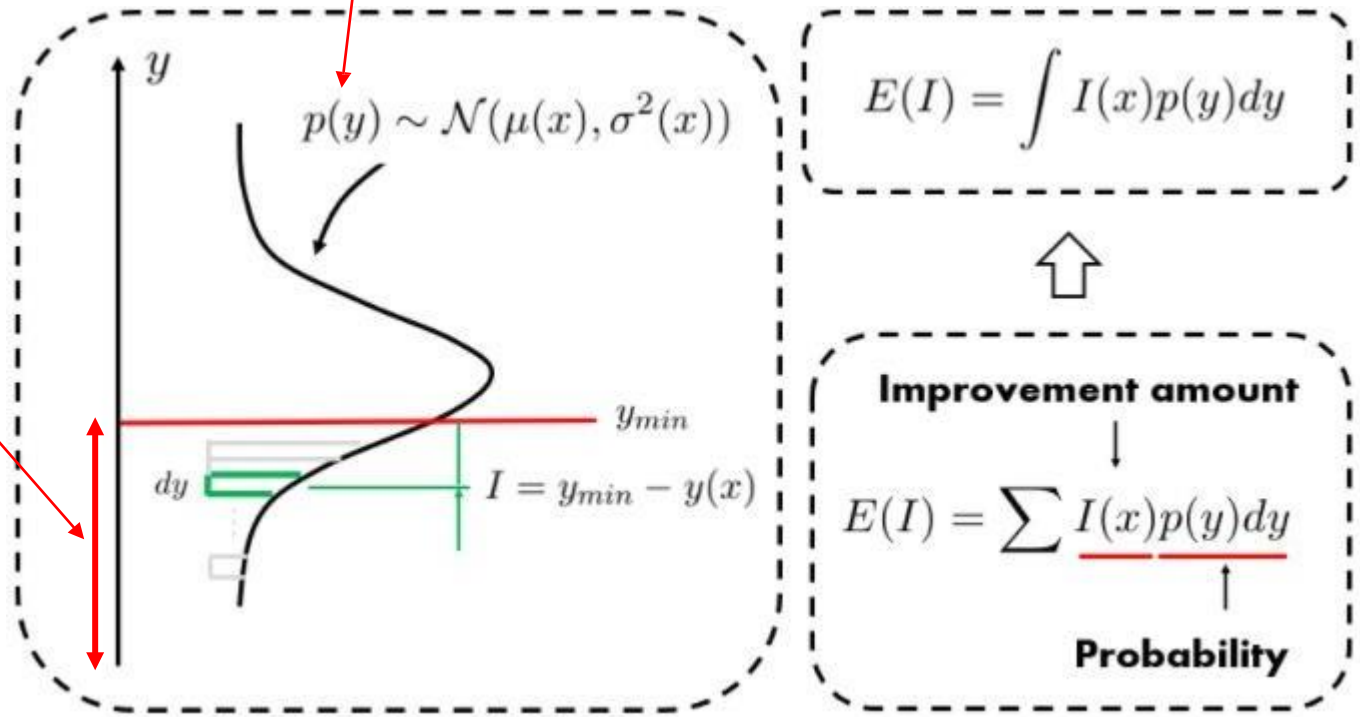
Expected Improvement Function

Improvement is where the y(x) is better (lower) than the previous best minimum

$$I(x) = \begin{cases} y_{min} - y(x) & \text{if } y(x) < y_{min} \\ 0 & \text{if } y(x) \geq y_{min} \end{cases}$$



$p(y)$ is the probability density of the normal distribution $N(\mu(x), \sigma^2(x))$



Expected Improvement Function

- Perhaps the most used acquisition.
- Explicit for available for Gaussian posteriors.
- It is too greedy in some problems. It is possible to make more explorative adding a 'explorative' parameter

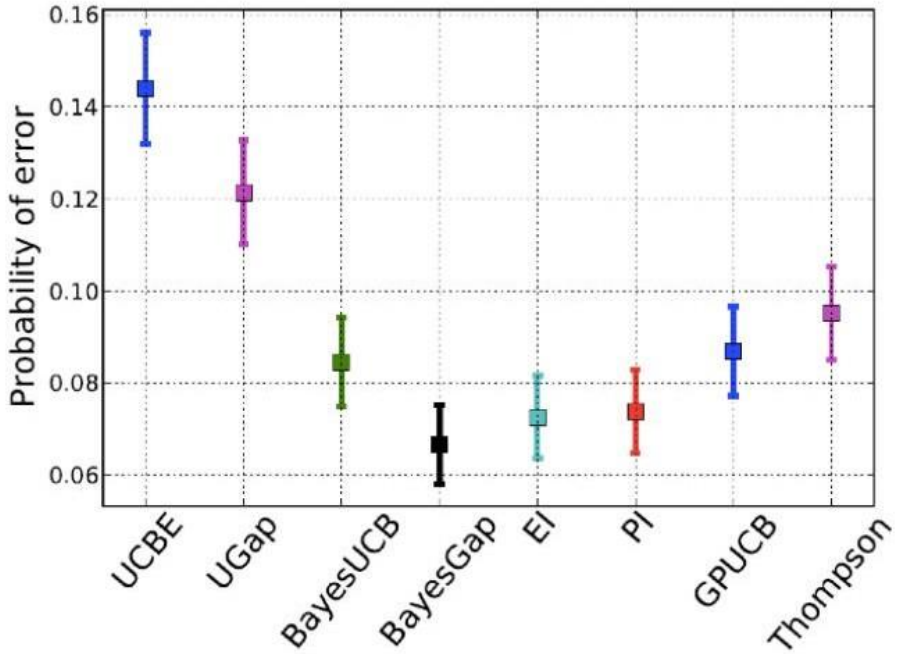
$$\alpha_{EI}(\mathbf{x};\theta,\mathcal{D}) = \sigma(\mathbf{x};\theta,\mathcal{D})(\gamma(x)\Phi(\gamma(x))) + \mathcal{N}(\gamma(x);0,1).$$

where

$$\gamma(x) = \frac{f(x_{best}) - \mu(\mathbf{x};\theta,\mathcal{D}) + \psi}{\sigma(\mathbf{x};\theta,\mathcal{D})}.$$

The choice of utility matters…

The choice of the utility may change a lot the result of the optimisation.



Why Doesn’t Everyone Use This?

These ideas have been around for *decades*.
Why is Bayesian optimization in broader use?

- **Fragility and poor default choices.**
  Getting the function model wrong can be catastrophic.
- **There hasn’t been standard software available.**
  It’s a bit tricky to build such a system from scratch.
- **Experiments are run sequentially.**
  We want to take advantage of cluster computing.
- **Limited scalability in dimensions and evaluations.**
  We want to solve big problems.

Rules of Thumb

Bayesian optimization

+ is a powerful algorithm for optimizing functions that are expensive to evaluate, noisy, and have a small number of dimensions. It can find the optimum with relatively few function evaluations, which can be a huge advantage when evaluating the function is computationally expensive.

- The main disadvantage is that it can be slow to converge when the function is highly non-linear or has many dimensions.

Genetic algorithms

+ are well-suited for problems with a large number of potential solutions, they can search a large space of potential solutions and are good at finding global optima.

- They can be slow to converge and can suffer from premature convergence.

Simulated Annealing algorithms

+ are well-suited for problems with a large number of local optima. They can escape local optima and find the global optimum.

- The main disadvantage is that they can be slow to converge and require careful tuning of the temperature schedule.

Particle swarm optimization

+ is well-suited for problems with a large number of dimensions, it can search a large space of potential solutions and is good at finding global optima.

- It can be slow to converge and can suffer from premature convergence.

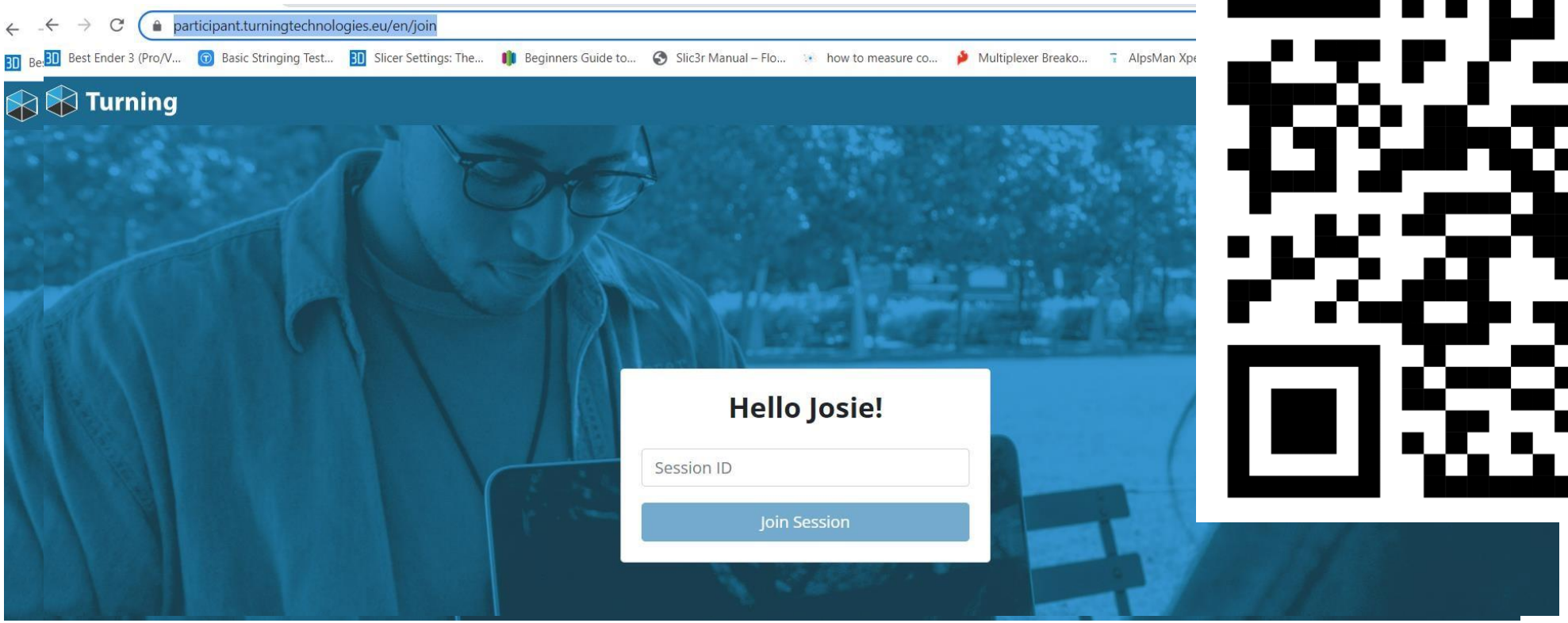
Recap Quiz!!

https://participant.turningtechnologies.eu/en/join

https://participant.turningtechnologies.eu

SessionID: datadriven

https://ttpoll.eu/p/datadriven

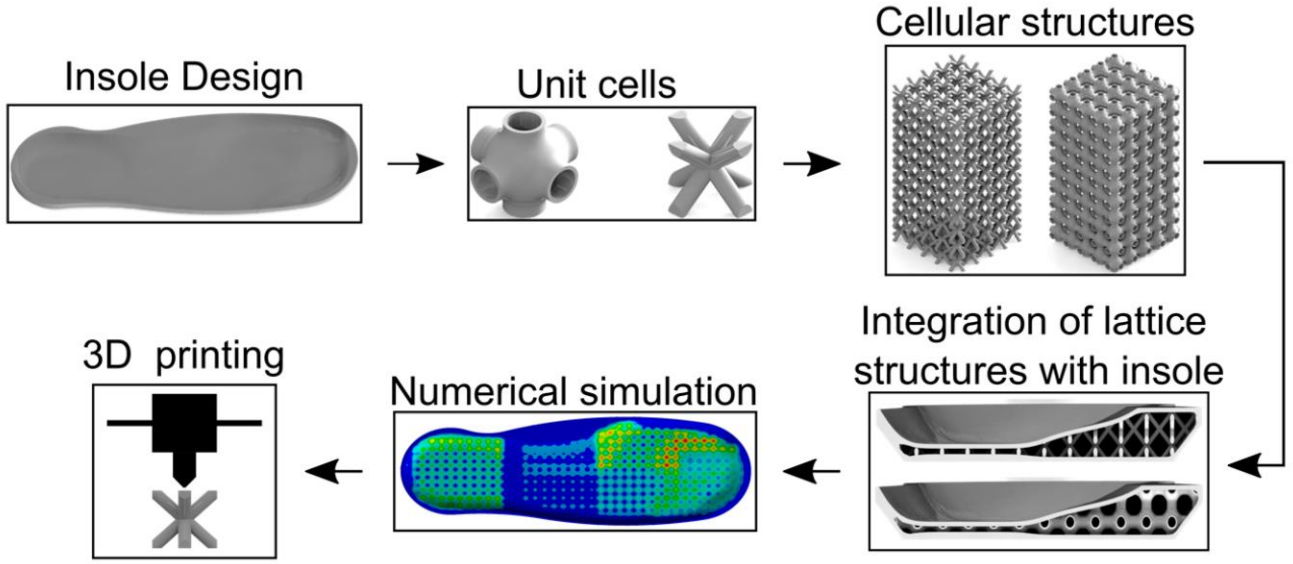


Data Driven Methods:

What to use when

Scenario 1: Optimization of foam for insole

Goal: Maximize stiffness whilst minimizing material



Q1: How should I encode?

Q2: Should I consider dimensionality reduction?

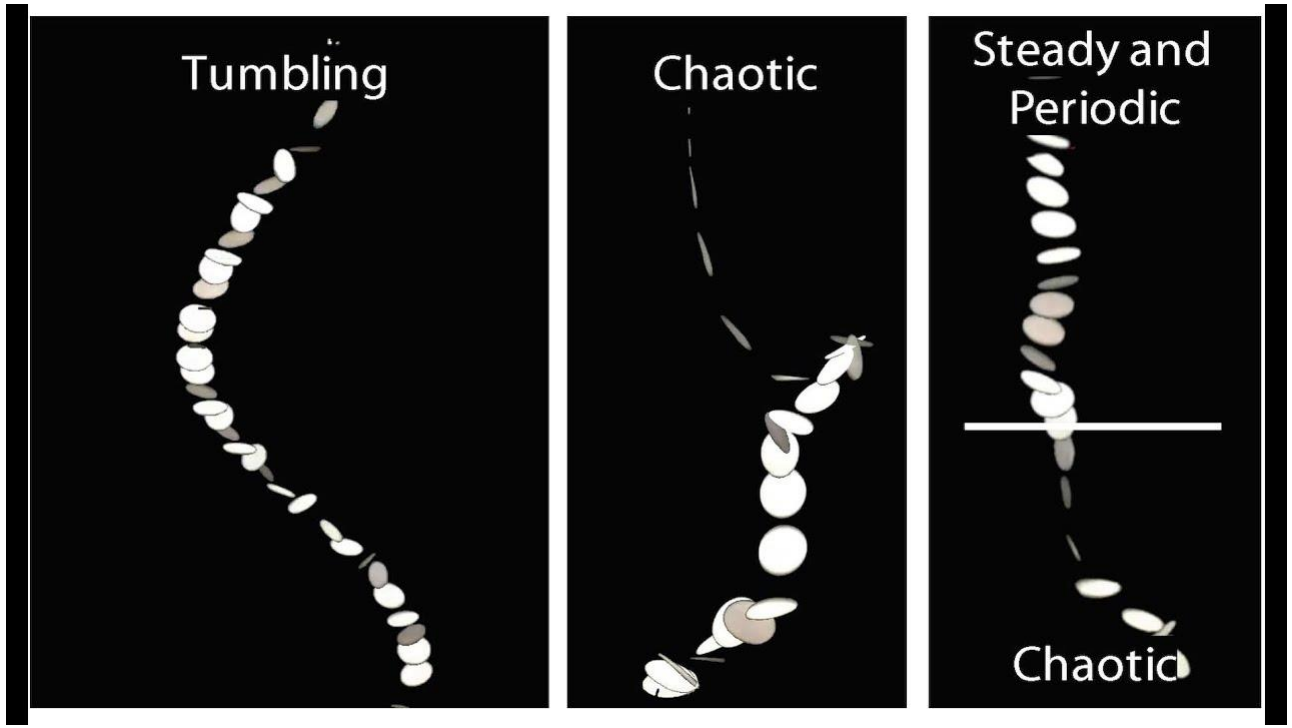
Q3: What DD algorithm should I use and why?

Function Evaluation:

- Can be simulated
- Takes approximately 1s per simulation evaluation
- Goal is to optimize for many different materials

Scenario 2: Falling Paper Shapes

Goal: Identify what paper shape falls the fastes



Q1: How should I encode?

Q2: Should I consider dimensionality reduction?

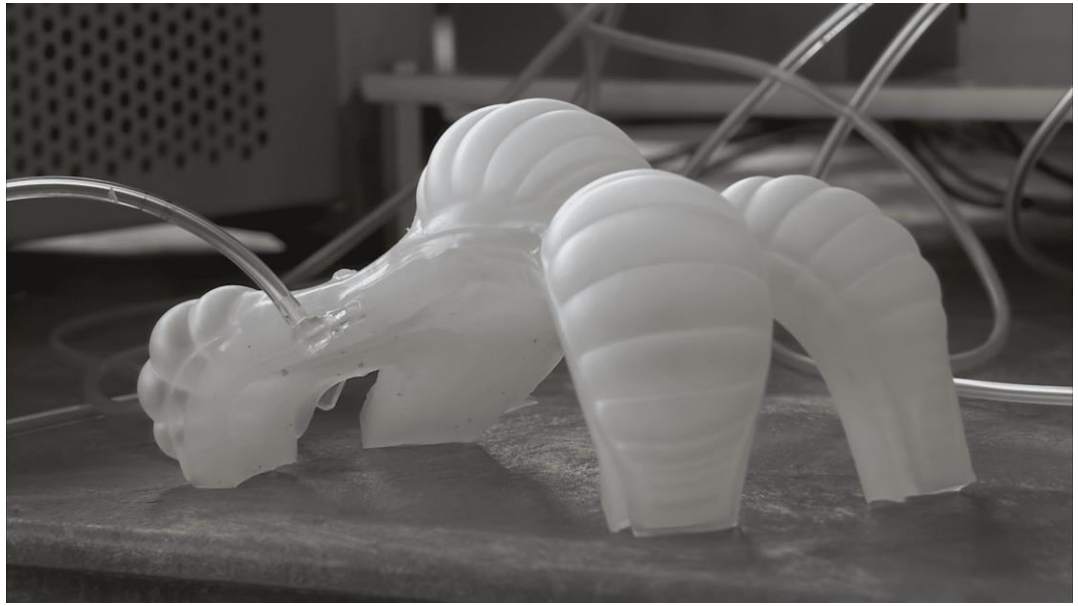
Q3: What DD algorithm should I use and why?

Function Evaluation:

- Can’t be simulated
- Takes approximately 90 seconds per trial, must be human supervised
- Stocastic behaviours

Scenario 3: Optimization of soft robot

Goal: Maximize stiffness whilst minimizing material



Function Evaluation:

- Can be simulated, takes 1 hr per simulation
- Can be experimentally evaluated, take 90 minutes to build and test

Q1: How should I collect data?
Q2: How should I encode?
Q3: What DD algorithm should I use and why?

Scenario 4: Optimization of 3D printing of structures

Goal: Maximize overhang with least material

Function Evaluation:

- Can be simulated, take 60 seconds, but very low accuracy
- Can be experimentally evaluated by takes 5 minutes per design

Q1: How should I collect data?
Q2: How should I encode?
Q3: What DD algorithm should I use and why?