C477: Zeroth Order Methods

Ruth Misener
r.misener@imperial.ac.uk

Panos Parpas
p.parpas@imperial.ac.uk

Computational Optimisation Group
Department of Computing

November 4, 2016

Outline

- **Topics**
  - Using Global Information
  - DIRECT
  - Probabilistic Search Methods (will not be tested)
    - ⋆ Bayesian Optimisation, Simulated Annealing, Genetic Algorithms
- **Examples**
  - Microalgae metabolism
  - Making a robot walk quickly
- **Reading**
  - Chapters 14 (Global Search Algorithms), Chong & Żak, Third Edition.
- **Acknowledgements**
  - Parts of these slides were originally developed by Benoit Chachuat and Panos Parpas. LaTeX design by Miten Mistry. Mistakes by Ruth Misener.

Global Information

Level of Information

Without global information, no algorithm can provide a certificate of global optimality, unless it generates a dense sample

Examples of Global Information:

- Number of local optima
- Global optimum value
- Convexity of the objective function and feasible region
- Lipschitz constant L,

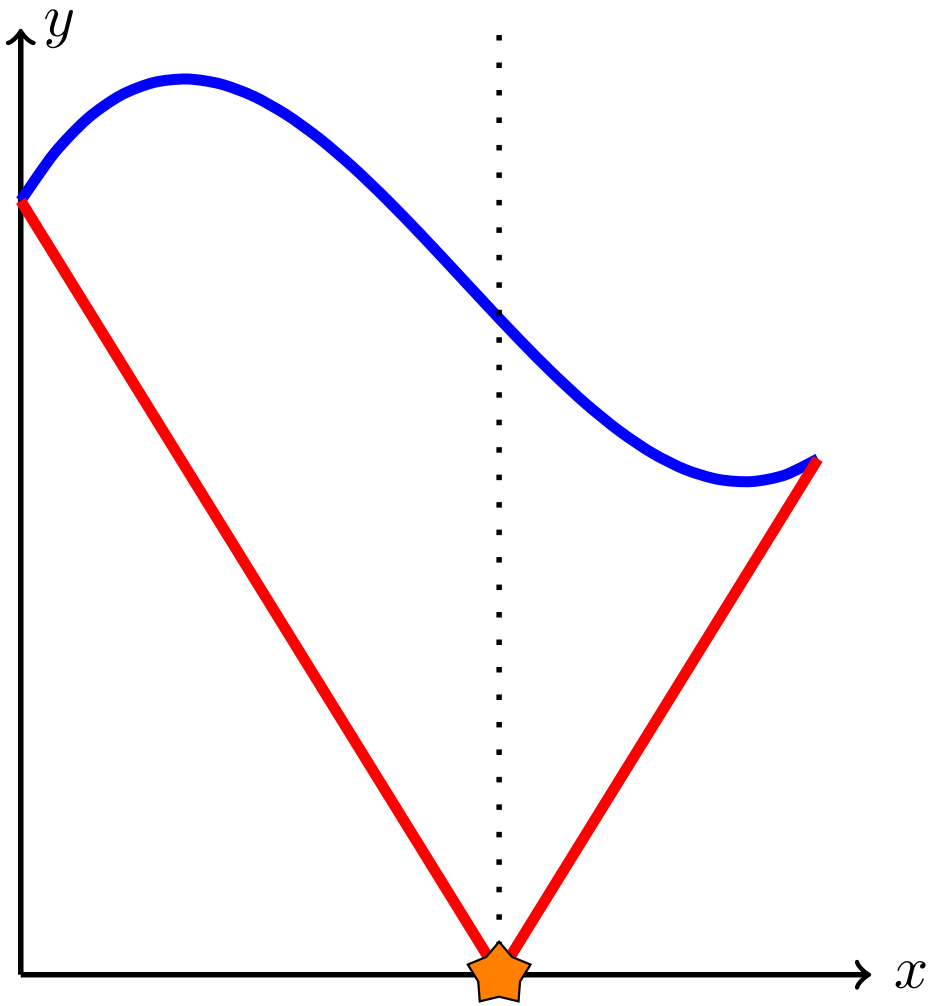
$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|_2, \quad \forall \mathbf{x}, \mathbf{y} \in S$$

Algorithmic Strategies:

1. Unstructured Problems: Global information unavailable
   ➥ Global optimum certificate is hopeless!
2. Structured Problems: Global information available
   ➥ Global optimality may be certified, but...

Use Lipschitz Constant for **Global Optimisation**?

Initiate divide and conquer algorithm similar to **Branch & Bound**



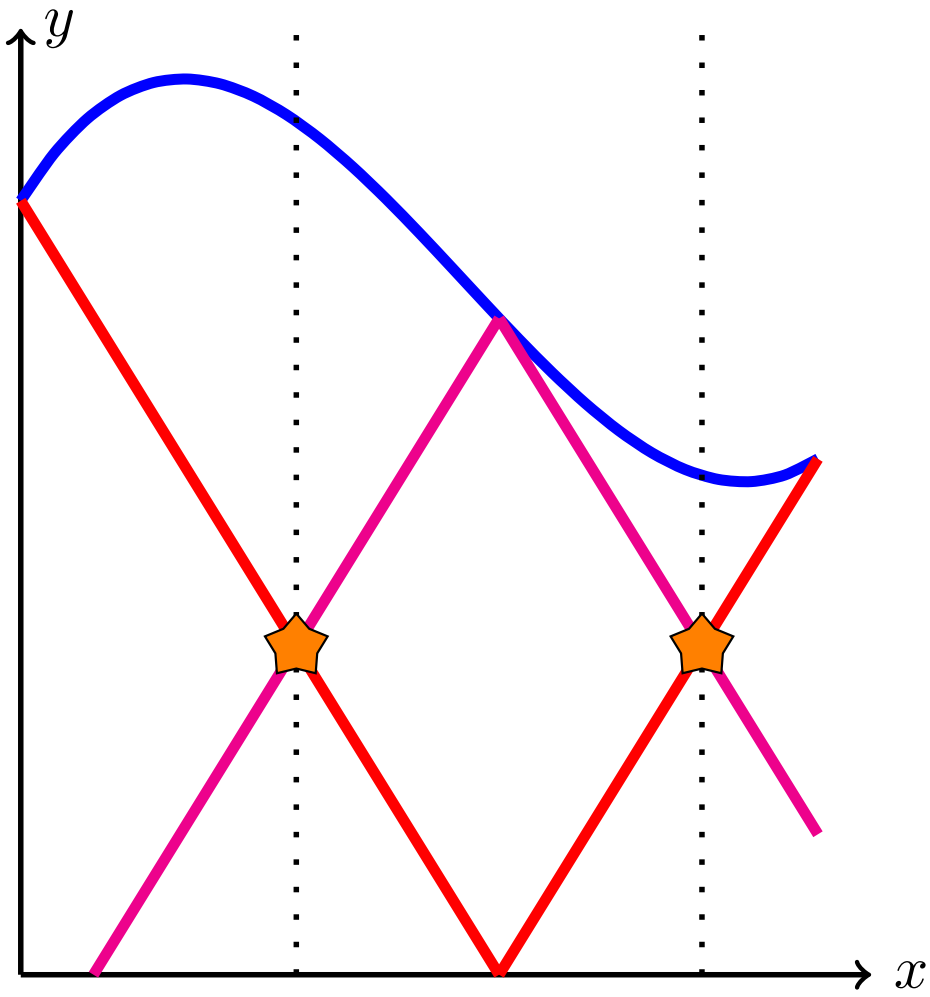
Key Observations

- This is Shubert's algorithm;
- Easy to find intersection of two lines!
- Typical convergence criteria? $f^{\mathsf{UB}} - f^{\mathsf{LB}} < \epsilon$
- Is this as bad as complete search?

Function Evaluation Points: $\{0.8,\ 0.0368,\ 1.5632,\ -0.5740,\ 0.6476\}$

Use Lipschitz Constant for **Global Optimisation**?

Initiate divide and conquer algorithm similar to **Branch & Bound**



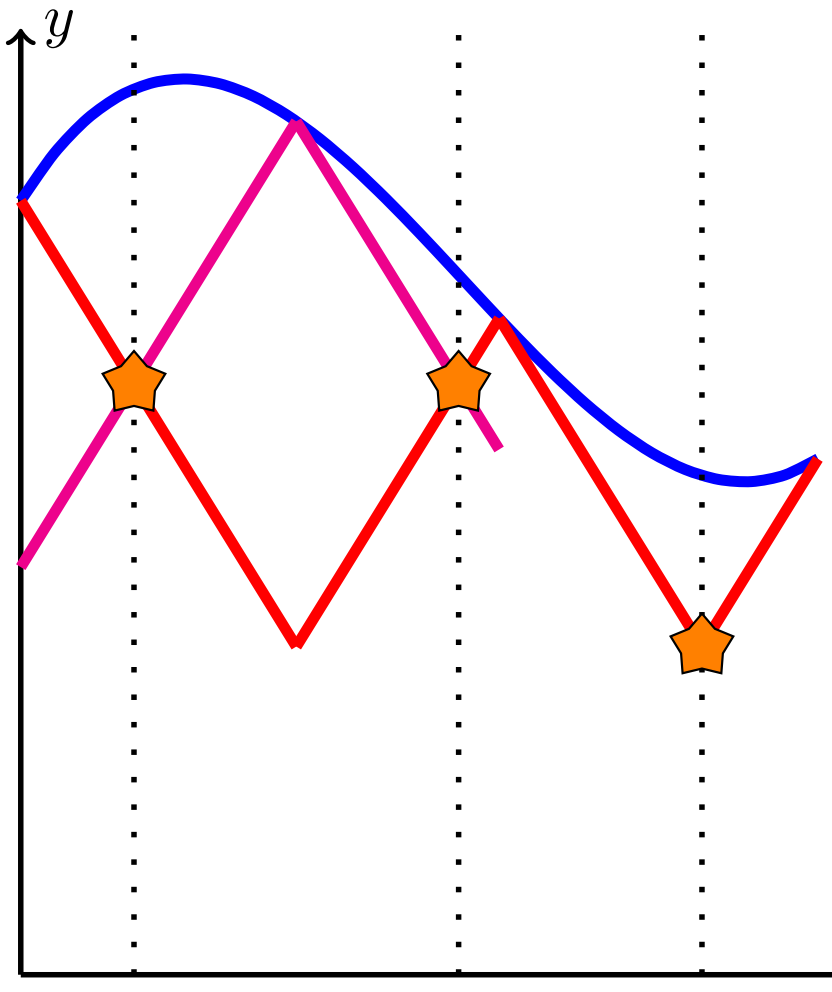
Key Observations

- This is Shubert's algorithm;
- Easy to find intersection of two lines!
- Typical convergence criteria? $f^{\mathsf{UB}} - f^{\mathsf{LB}} < \epsilon$
- Is this as bad as complete search?

Function Evaluation Points: $\{0.8,\ 0.0368,\ 1.5632,\ -0.5740,\ 0.6476\}$

Use Lipschitz Constant for **Global Optimisation**?

Initiate divide and conquer algorithm similar to **Branch & Bound**



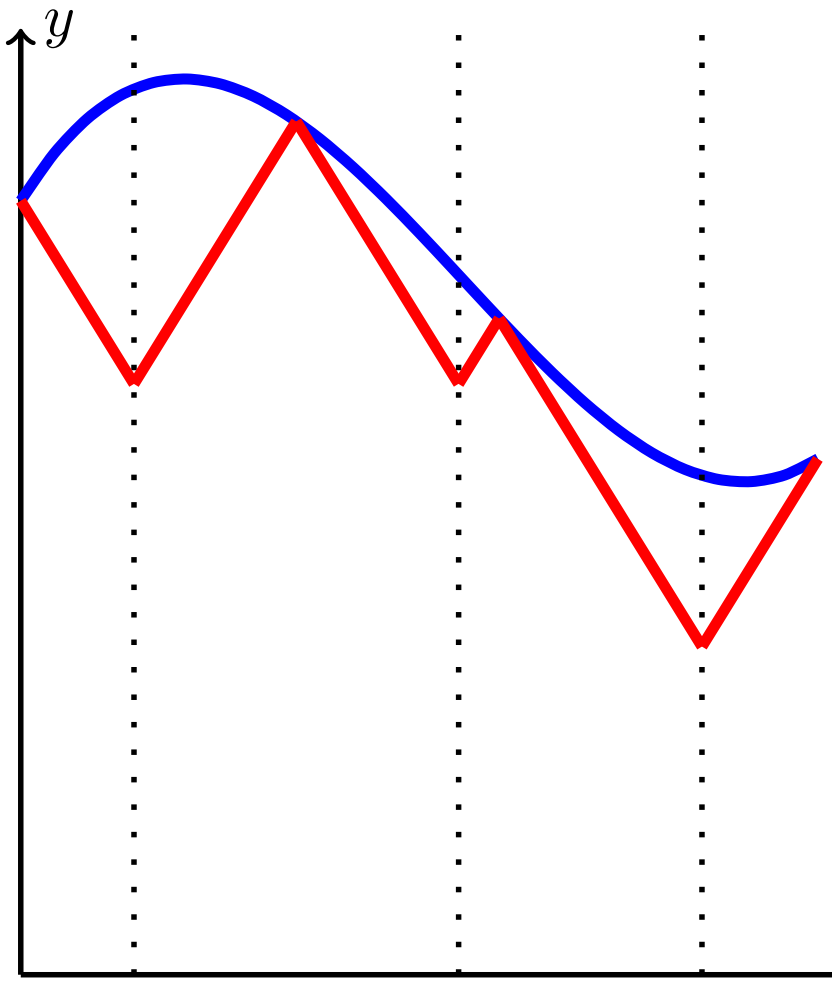
Key Observations

- This is Shubert's algorithm;
- Easy to find intersection of two lines!
- Typical convergence criteria? $f^{\mathsf{UB}} - f^{\mathsf{LB}} < \epsilon$
- Is this as bad as complete search?

Function Evaluation Points: $\{0.8,\ 0.0368,\ 1.5632,\ -0.5740,\ 0.6476\}$

Use Lipschitz Constant for **Global Optimisation**?

Initiate divide and conquer algorithm similar to **Branch & Bound**



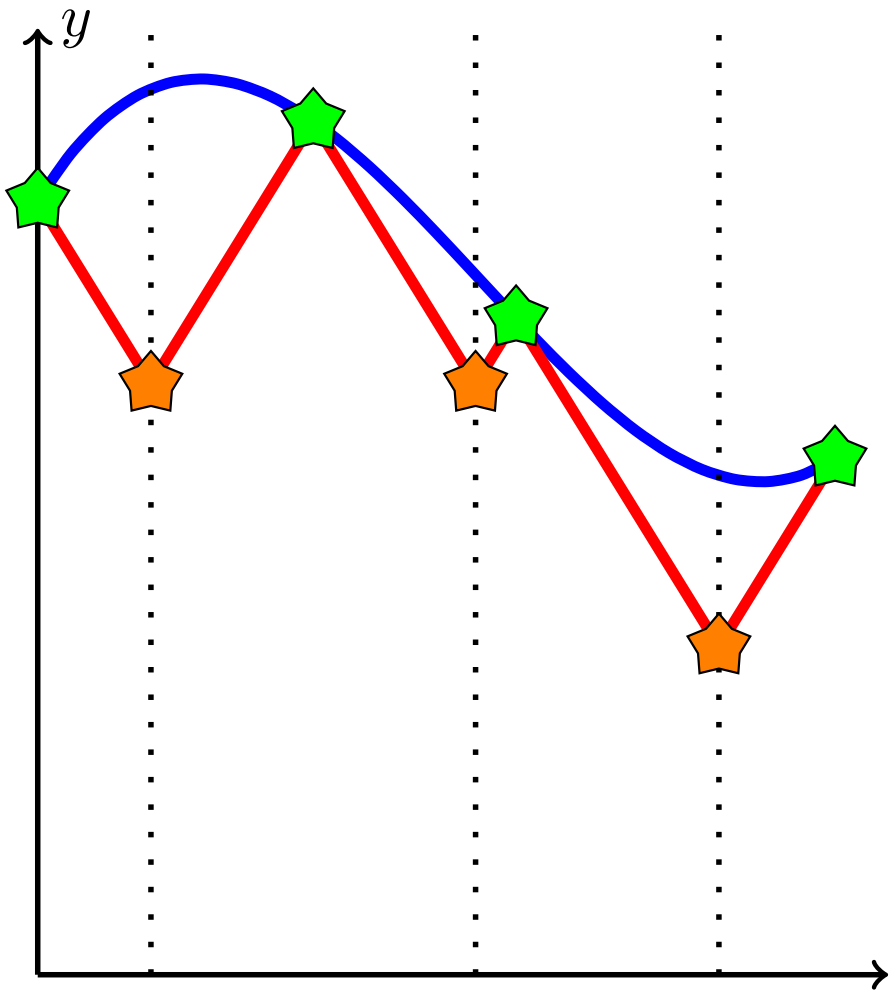
Key Observations

- This is Shubert's algorithm;
- Easy to find intersection of two lines!
- Typical convergence criteria? $f^{\mathsf{UB}} - f^{\mathsf{LB}} < \epsilon$
- Is this as bad as complete search?

Function Evaluation Points: $\{0.8,\ 0.0368,\ 1.5632,\ -0.5740,\ 0.6476\}$

Use Lipschitz Constant for **Global Optimisation**?

Initiate divide and conquer algorithm similar to **Branch & Bound**



Key Observations

- This is Shubert's algorithm;
- Easy to find intersection of two lines!
- Typical convergence criteria? $f^{\mathsf{UB}} - f^{\mathsf{LB}} < \epsilon$
- Is this as bad as complete search?

Function Evaluation Points: $\{0.8,\ 0.0368,\ 1.5632,\ -0.5740,\ 0.6476\}$

Problems of Liptschitzian Optimisation

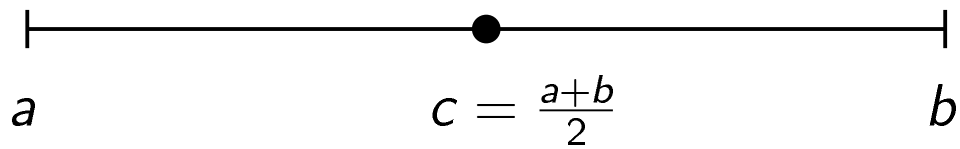
- L may not be easily accessible. What constant should we use?
- The parameter L is a trade-off between global and local search. If we could use all possible L, we could balance better between global and local search.
- **Combinatorial Complexity in Higher Dimensions** Lipschitzian Optimisation is initialized by evaluating the function at the corners of a hyperrectangle. We have to make $\mathcal{O}(2^n)$ evaluations.

DIRECT in 1D

Jones, Perttunen, Stuckman (1993)

"Lipschitzian Optimization Without the Lipschitz Constant"

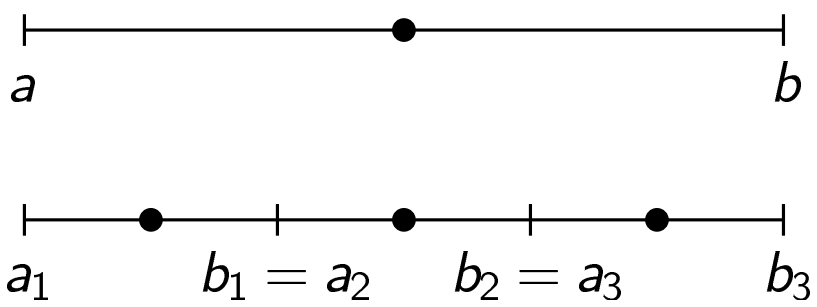
- The name DIRECT stands fro DIviding RECTangles, but also captures the fact that it is a direct search technique.
- Key idea: Sample the function at center of rectangle.



Dividing Intervals

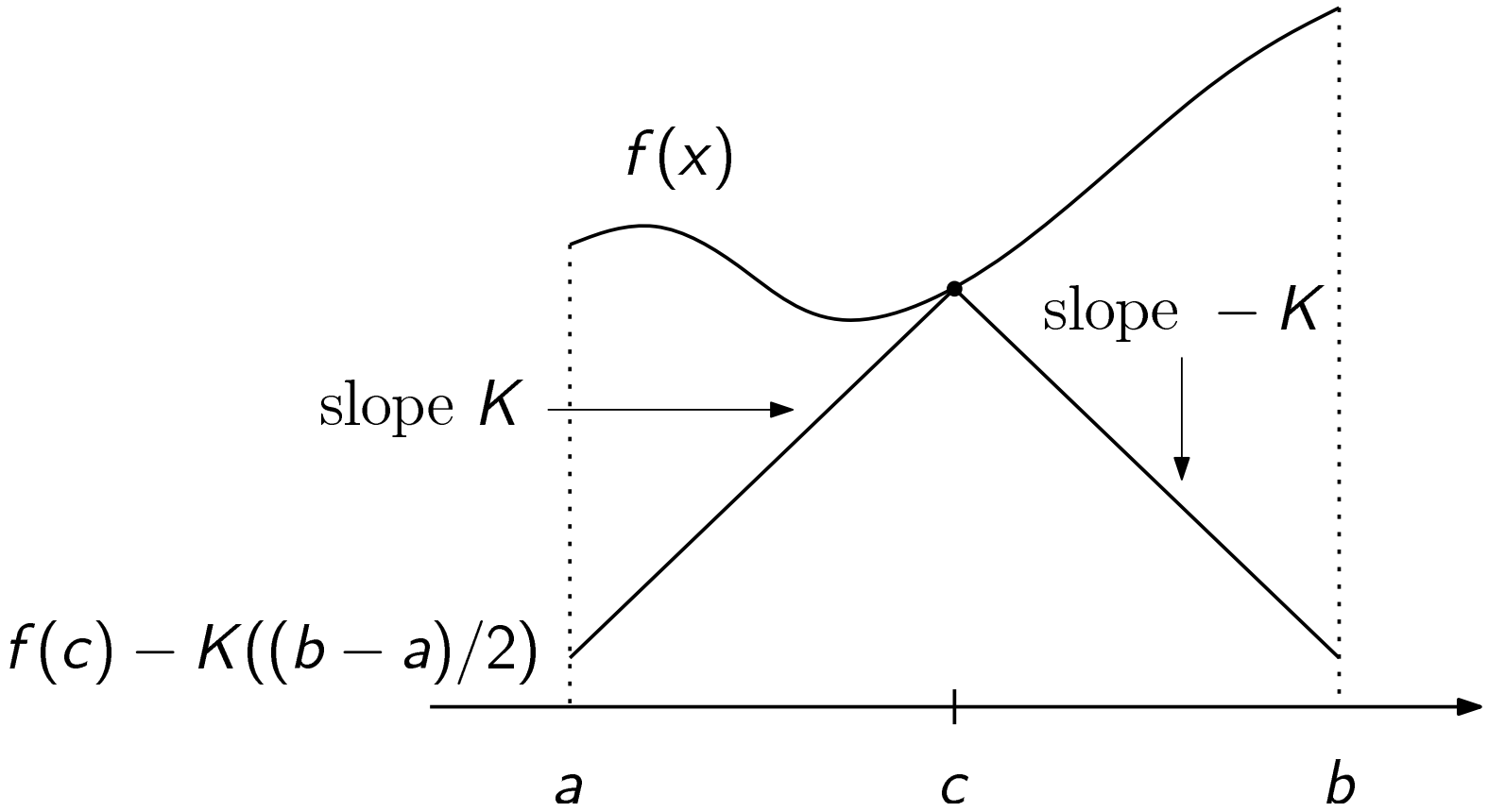
When dividing the search space we have to make sure that previous function evaluations are not lost, i.e. they are still at the center of some interval.

$\Rightarrow$ Instead of a bisection we do a trisection.



Lipschitz Bound

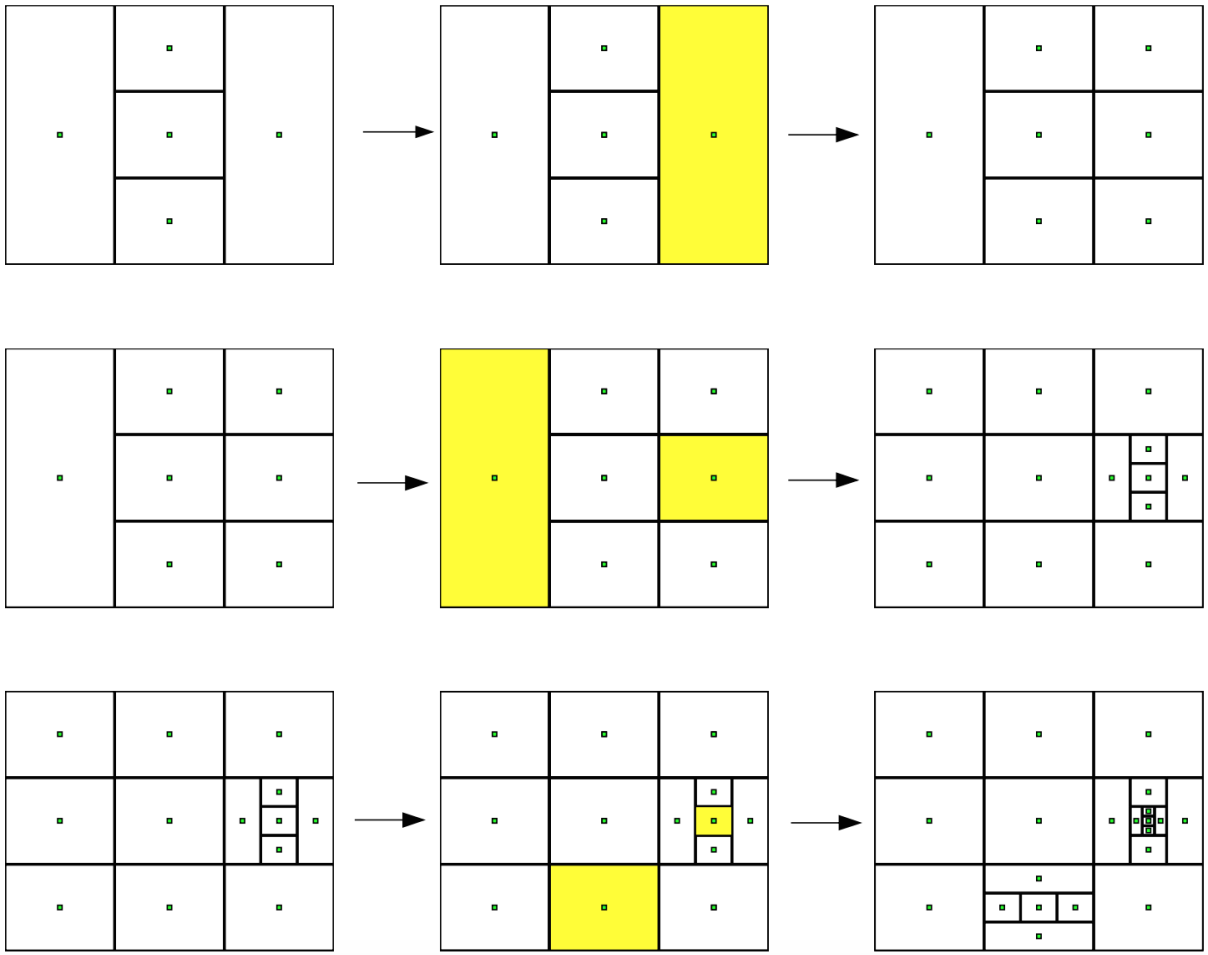
$$f(x) \geq f(c) + K(x - c) \quad \text{for } x \leq c,$$
$$f(x) \geq f(c) - K(x - c) \quad \text{for } x \geq c.$$



Generalisation: **DIRECT** Algorithm

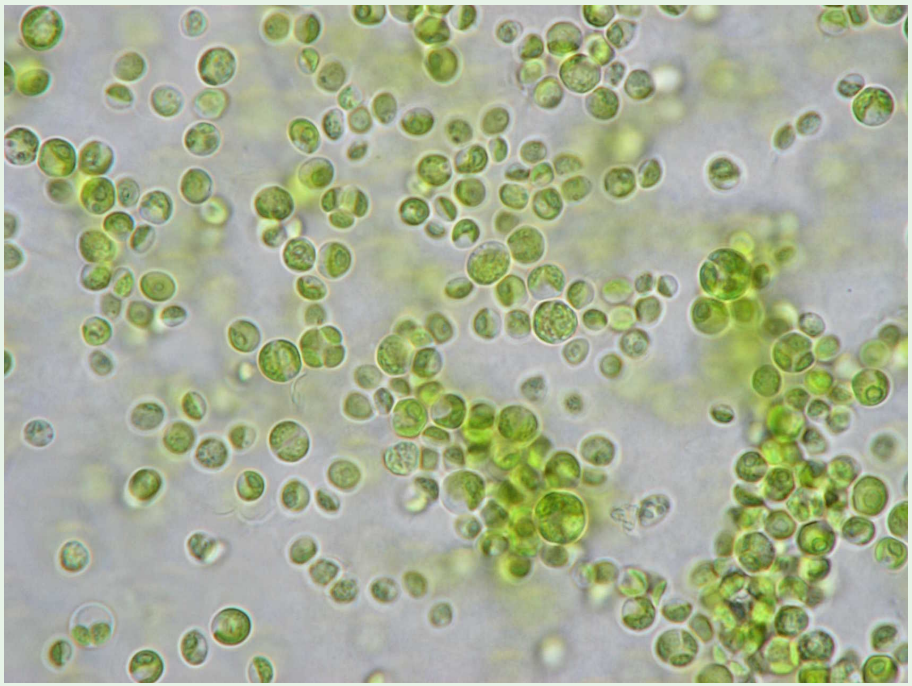
DIvide RECTangles

Converge on the global solution by dividing the hypercube into smaller rectangles; each time we only take a function evaluation.



Example: Microalgae Metabolism

Challenge: How can we test biological hypotheses for microalgae metabolism without doing too many costly, real-world experiments?

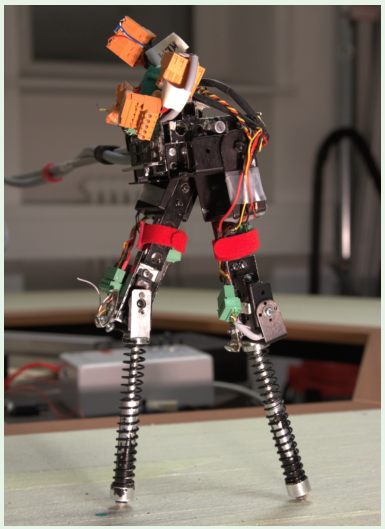


Use **Bayesian optimisation**, a type of **black-box optimisation**, to estimate parameters by minimising the squared error between model & experimental data points.

[Ulmasov, MSc Thesis, 2015]

C477 does not require probability or statistics as a prerequisite, so we cannot cover all of BayesOpt. But we will cover several **zeroth order** optimisation algorithms including DIRECT which is used within BayesianOpt frameworks.

Example Make a robot walk



Objective: optimise performance, e.g., as measured by velocity, within the range of reasonable gait parameters. But there is no closed-form mathematical formulation for the velocity as a function of the input parameters. To solve this problem, the authors use a type of **black-box optimisation** called **Bayesian optimisation**.

`http://wp.doc.ic.ac.uk/sml/project/bayesian-optimization/`

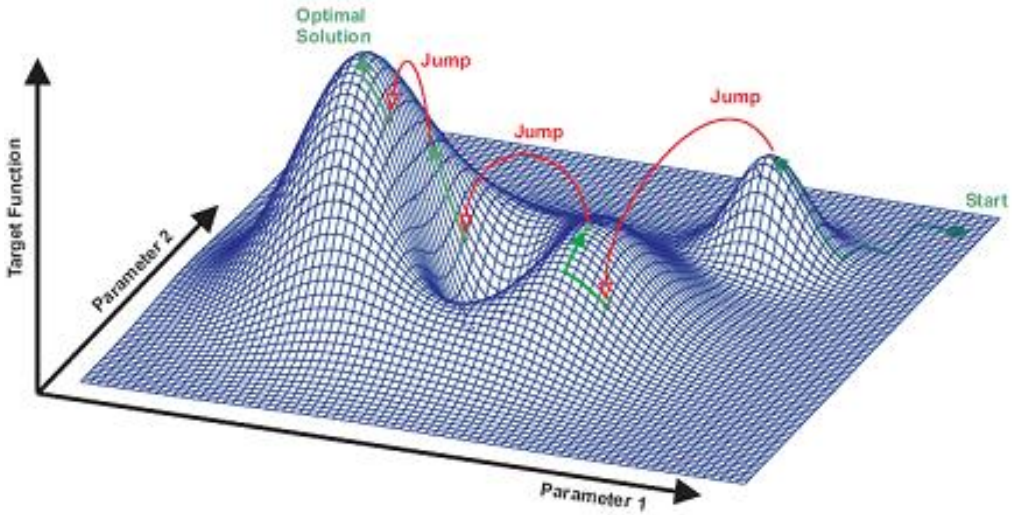
State-of-the-Art Tool for Bayesian Optimisation

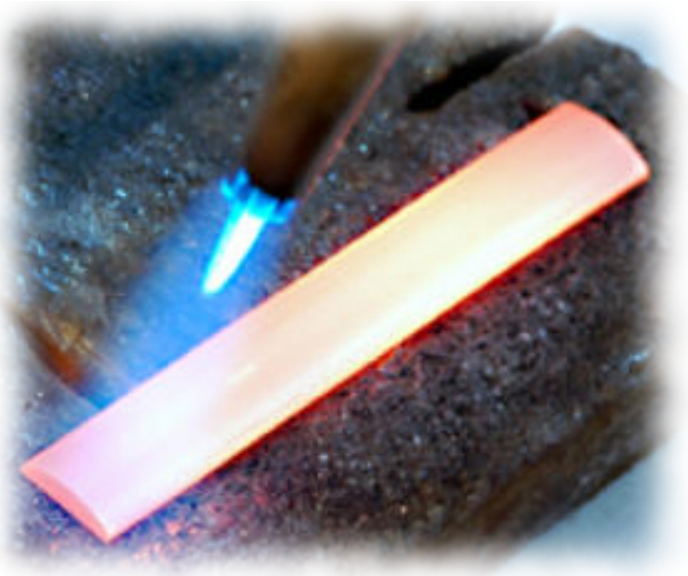
`https://github.com/mwhoffman/pybo`

Simulated Annealing

Idea

"When optimizing a complex system, instead of always going downhill/uphill, try to do downhill/uphill *most of the time*" — Haykin, 1999





Analogy: Annealing in Metallurgy

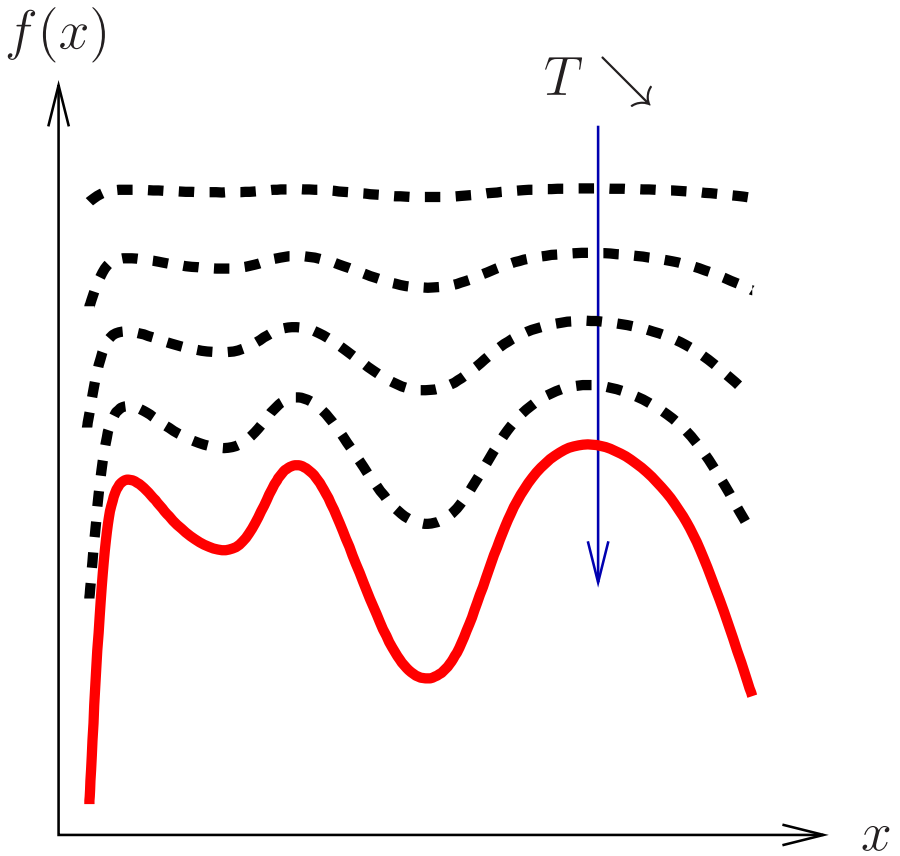
A technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects

- Heat causes the atoms to become unstuck from their initial positions
- Slow cooling increases chances of finding configurations with lower internal energy

Simulated Annealing [cont'd]

- Adaptive search method using Markov-chain sampling
  - Accept improvements unconditionally
  - Accept deteriorations with a probability that slowly decreases to zero

$$\mathcal{P}^k = \min\left\{1, \exp\left(\pm\frac{f(x^k) - f(x^{\text{best}})}{T^k}\right)\right\}, \quad T^k \xrightarrow{k\to\infty} 0$$

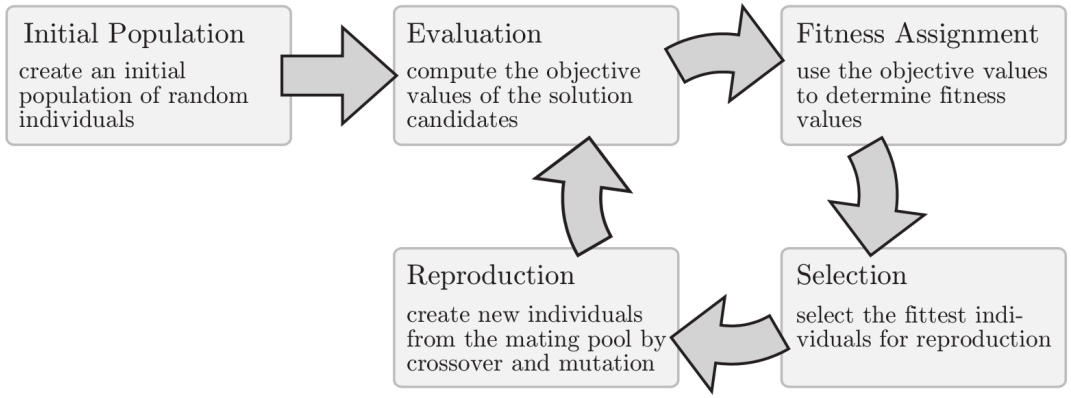



```
PROCEDURE simulated annealing()
    InputInstance();
    Set y = −∞;
    Set T = ∞;
    Choose x ∈ S;
    DO
        Generate a point z according to some
        Markov chain transition distribution;
        With probability min{1, e^((f(z)−f(x))/T)},
        set x = z;
        If f(x) > y set y = f(x);
        Adjust T;
    OD;
    RETURN(y);
END simulated annealing;
```

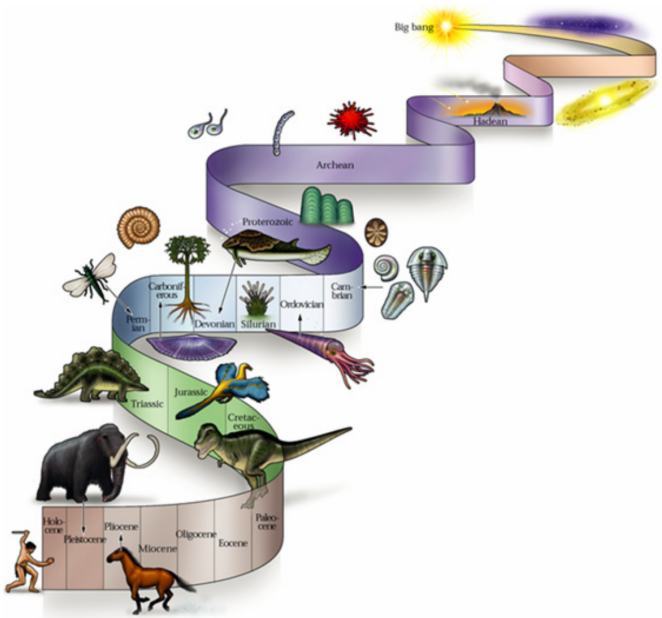
Population-based Methods

Idea

Keep a population of solution points as a base to generate new points, instead of a single iterate.



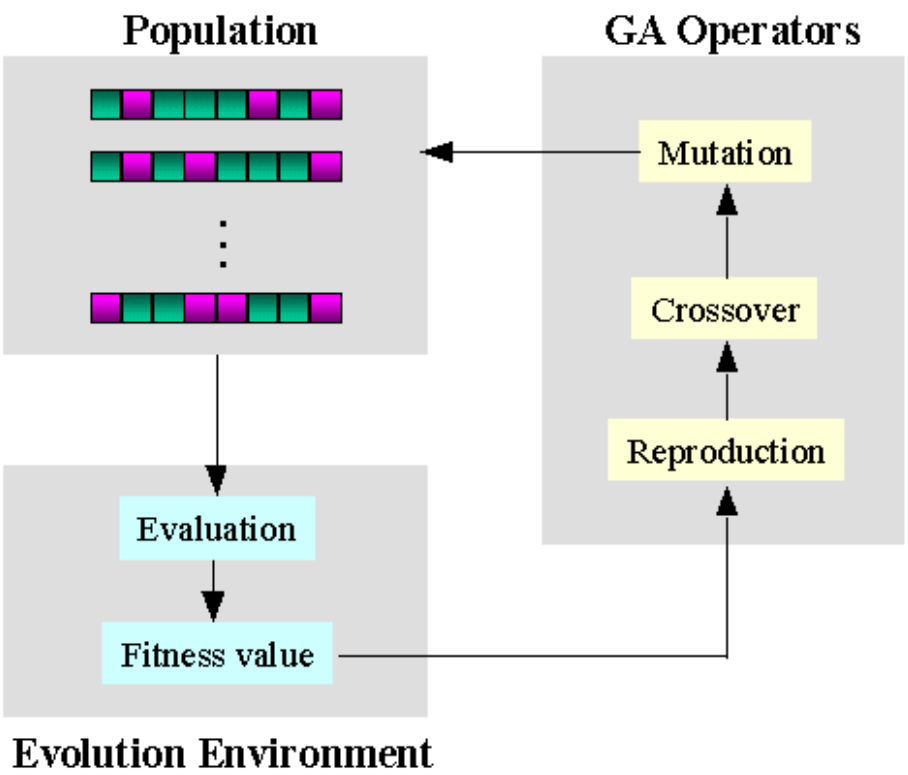
Examples: Controlled random search; Genetic Algorithms; Particle swarms



Analogy: Evolution in Biology

- Organisms produce offsprings similar to themselves, but with variations due to crossover & mutation.
- Some offsprings survive and reproduce, others do not, leading to adaptation

(Vanilla) Genetic Algorithms




```
PROCEDURE genetic algorithm()
    Initialize population;
    FOR (g = 1 to N_gen generations) DO
        FOR (i = 1 to N_pop individuals) DO
            Evaluate fitness of individual i: f_i(g):
        END FOR;
        Save best individual to population g + 1;
        FOR (i = 2 to N_pop) DO
            Select 2 individuals;
            Crossover: create 2 new individuals;
            Mutate the new individuals;
            Move new individuals to population g+1;
        END FOR;
    END FOR;
END genetic algorithm;
```

Fitness: For a population $\mathcal{P}$ of N individuals,

$$\mathcal{F}(\mathbf{x}) = \frac{\max_{\mathbf{y}\in\mathcal{P}} f(\mathbf{y}) - f(\mathbf{x})}{\max_{\mathbf{y}\in\mathcal{P}} f(\mathbf{y}) - \min_{\mathbf{y}\in\mathcal{P}} f(\mathbf{y})}$$

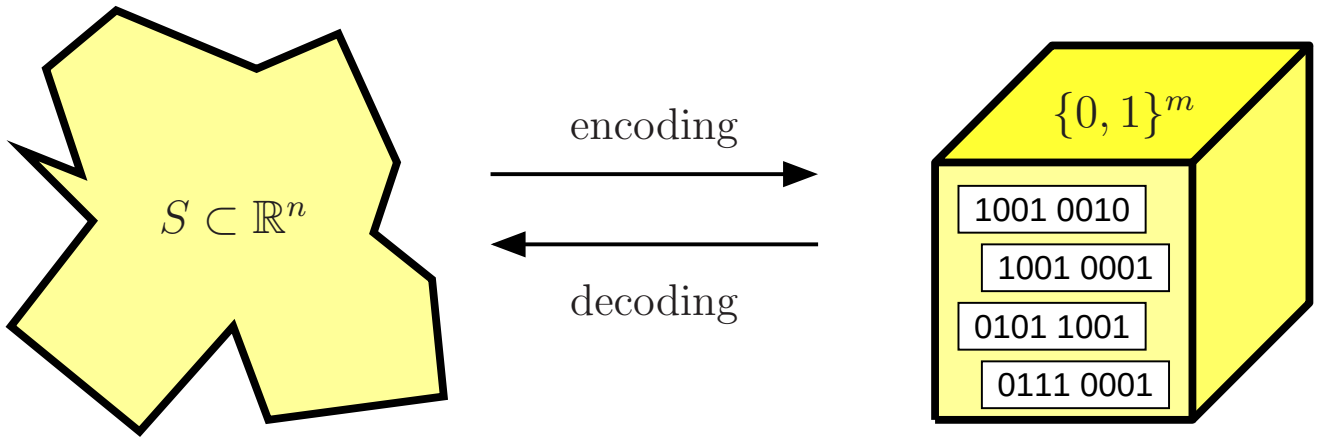
Reproduction: Probability for $\mathbf{x} \in \mathcal{P}$ to be selected, $\dfrac{\mathcal{F}(\mathbf{x})}{\sum_{\mathbf{y}\in\mathcal{P}} \mathcal{F}(\mathbf{y})}$

Higher chance for better individuals!

Genetic Algorithms [cont'd]

Coding: Representation of the individuals by fixed-length binary strings

- Same coding used by computers to manipulate data!



- Other representations: Real numbers; Gray (binary) coding

Overwhelming terminology from biology and nature!

- Gene – A single encoding of part of the solution space, e.g. a bit $\boxed{1}$
- Chromosome – A string of genes that represent a solution $\boxed{1001}$
- Population – All of the chromosomes available to test

Genetic Algorithms: Outlook

Pros:

- Easiness of concept
- Handling of discontinuous/ nonsmooth functions
- Multiobjective optimization support

Extensions:

- More operators!
- Alternative representations

Cons:

- Tuning – Too many parameters!
  - Population size
  - Selection policies
  - Crossover/mutation operators
  - Crossover/mutation probabilities
  - Termination criteria
- Performance
  - Covers a large space, yet slow!
  - Numerous function evaluations
- Handling of Constraints

Software

- Rather easy implementation
- Numerous available codes in Fortran, C, Matlab, etc.
  check: `http://www.mat.univie.ac.at/~neum/glopt/software_g.html`

Summary