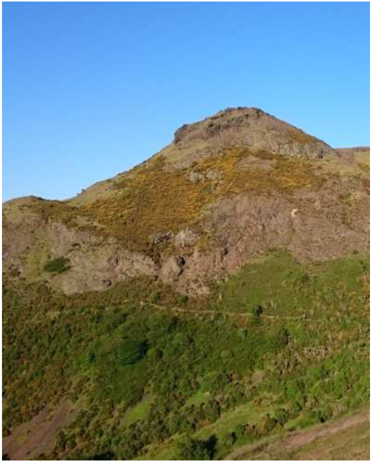
Course: Natural Computing

1. Optimisation

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Optimisation: Searching for a maximum or minimum

Optimisation by "hill-climbing" (or: "valley-diving")



Optimisation: Searching for a minimum or maximum

States: $x \in X$ with $X \subset \mathbb{R}^d$ or $X \subset \mathbb{Z}^d$

Objective function $F : X \to \mathbb{R}$

For which $x \in X$ the value $F(x)$ is smallest/largest?

In order to avoid mathematical difficulties, we will assume that

- F is bounded from below/above,
  i.e. $\exists c \; \forall x \;\; F(x) > c \; / \; F(x) < c$
- X is finite and closed (*compact*),
  i.e. at least one minimum will always exist

In natural computing, we usually do not assume the existence of derivatives or the availability an analytical form of F.

A few relevant terms

- Optimum:
  - Maximum (in maximisation problems)
  - Minimum (in minimisation problems)
- Extremum: a point that is either a minimum or a maximum
- Gradient: generalisation of the derivative to higher dimensions; a vector pointing in the upward direction (if there is an upward direction)
- Stationary point: any point where there is no upward direction (gradient is the zero vector) can be a
  - maximum,
  - minimum or
  - saddle point (i.e. a maximum w.r.t. some directions, minimum w.r.t. to others)
- Hessian: a matrix that contains second derivatives

Optimisation methods

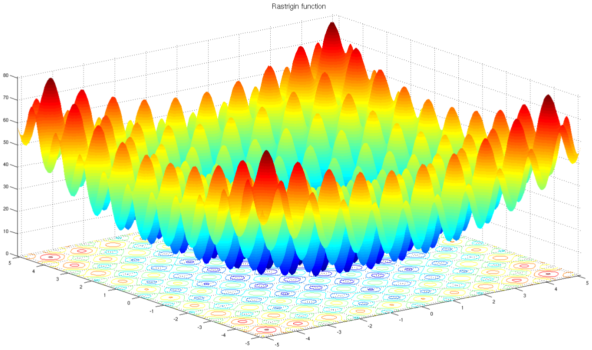
- Hill-climbing (with step width control)
- Downhill simplex (Nelder–Mead) Method
- Gradient descent
- Newton methods
- BFGS method (Broydon-Fletcher-Goldfarb-Shanno)
- Conjugate gradient
- Stochastic optimisation
  - stochastic problems
  - stochastic search

Metaheuristic Optimisation

- Conceptually and computationally simple
- Parallel search by a population
- Easily adaptable to particular problems
- Can be used, .e.g., for optimisation of hyperparameters of other algorithms
- Focus on exploration-exploitation dilemma

Optimisation by “hill-climbing” (or “valley-diving”)

Searching for improvement among near-by states?



https://en.wikipedia.org/wiki/Rastrigin_function

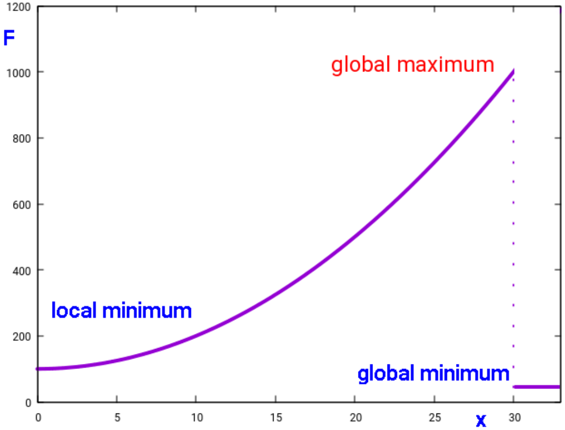
Local optima

How important is it in practice to avoid local optima?

- An objective function over a high-dimensional spaces is expected to have few minima (or maxima), instead the stationary points are more likely to be saddle points
- Nevertheless, an agent moving in this landscape will spend most of the time at or near the local optima, and saddle points can cause similar problems
- High-dimensional data is usually sparse, i.e. not all local optima may be revealed
- Benchmark functions tend to have many local optima (or completely flat regions)

Optimisation by "Hill-Climbing" (rather: "valley-diving")

Searching for improvement among near-by states?

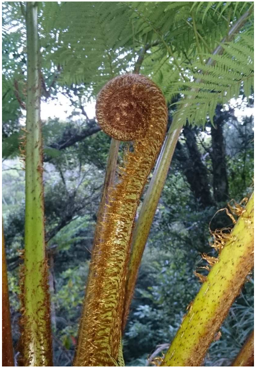


Evolutionary algorithms

Natural evolution achieves improvement by

- Small random changes
- Selection of “good” individuals
- Replenishing the population

We’ll focus on these (and leave the genotype-phenotype relations for later)



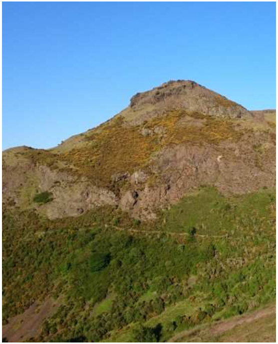
Nature-inspired optimisation

- The natural phenomenon that is referred to, is usually merely an analogy
- Nevertheless, nature has found astonishing solution to difficult problems, so it is interesting to learn how this happens
- Nature-inspired algorithms are usually stochastic optimisation algorithms, i.e. randomness is added in order to escape local minima
- Different types of noise characterise different algorithms

A simple example: Hill-climbing

Consider first a single individual $x \in \mathbb{R}^d$:

- Apply a small change
- Selection consists in accepting or rejecting the change based on an evaluation function F: Maximisation task:
  - Reject if $F(x') < F(x)$
  - accept otherwise
- Continue with one individual

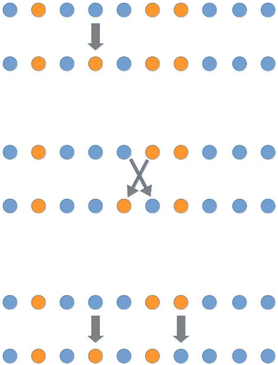


Obviously, this is not a sophisticated and efficient algorithm. However, because of its simplicity, it seems good for a start.

Hill-climbing: Operator formalism

Possible changes:

- add noise to a component of x
- swap two of its components
- change two components simultaneously
- ...



More formally: $x \in X \subset \mathbb{R}^d$
Consider operators

$$O : X \to X$$

$$O(x) = x'$$

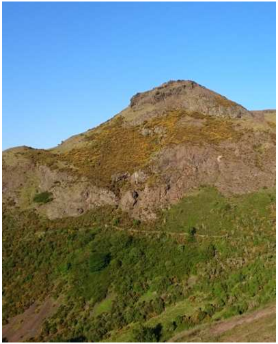
Let $\mathcal{O}$ be a class of such operators

e.g. all operators $O_k^{+\varepsilon}$ and $O_k^{-\varepsilon}$ that add or subtract ε to or from the k-th component of x, i.e. $\mathcal{O}_\varepsilon = \left\{ O_k^{+\varepsilon}, O_k^{-\varepsilon} | k \in \{1, \ldots, d\} \right\}$

Hill-climbing

Hill-climbing

- choose a class of operators on x
- apply any of them and accept or reject bases on evaluation function F
- until no further improvements can be achieved



E.g. All-Ones problem ($x \in \{0,1\}^d$):
$F(x) = \sum_{k=1}^{d} x_k$, number of "1"s in x

Operators: $\mathcal{O} = \{O_k | k = 1, \ldots, d\}$, O_k flips k-th component of x

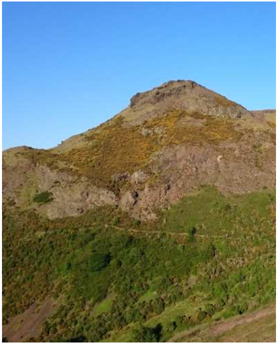
Result:

For any starting point $x_0 \in \{0,1\}^d$, hill-climbing finds $x = (1, \ldots, 1)$

Hill-climbing

Hill-climbing

- choose a class of operators on x
- apply any of them and accept or reject bases on evaluation function F
- until no further improvements can be achieved



E.g. All-Ones problem ($x \in \{0,1\}^d$):
$F(x) = \sum_{k=1}^{d} x_k$, number of "1"s in x

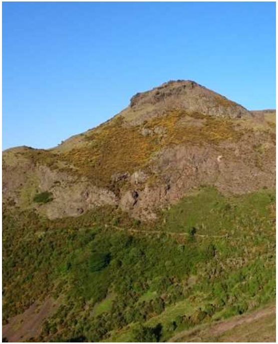
Operators: $\mathcal{O} = \{O_{kl} | k, l = 1, \ldots, d\}$, O_{kl} swaps the k-th and the l-th component of x

Result: None of the operators in the set $\mathcal{O}$ have any effect on F
$\Rightarrow$ no improvement. (Single-flip operators would work better in this case.)

Hill-climbing: Conclusion

Hill-climbing

- – tends to get stuck local optima
- – does not acquire information about the problem while running
- + is widely applicable
- + can be used as a baseline or
- + can be used for final or intermittent improvement of the solution



Outlook: Natural Computing

- Algorithms
- Theory
- Extensions
- Applications
- Recent trends