Course: Natural Computing (2020/21)

1. Introduction

1.1. Problem Solving in Nature

J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

Evolution of Computing

- Abacus
- Slide ruler
- Gear-driven calculating machines
- Relays, vacuum tubes, transistors ...
- Turing machines
- Analogue computers
- Social computing

Inspired by processes from

- Biology
- Physics
- Chemistry
- Neuroscience
- Social science

Natural Computation

comprises three classes of methods:

1. Those that employ natural materials (e.g., molecules) to compute (media)
2. Those that are based on the use of computers to synthesise natural phenomena (models) and
3. Those that take inspiration from nature for the development of novel problem-solving techniques (methods)

adapted from http://en.wikipedia.org/wiki/Natural_computing

(1) Computing with natural materials

Natural phenomena are used for information processing

- Analogue computers
- Smart matter
- Neural networks
- Membrane computing
- Molecular computing (DNA computing)
- Quantum computing

(2) Synthesising Natural Phenomena in a Computer

Revealing computational principles that underlie natural phenomena

- Computational systems
  - L-systems
  - Fractals
  - Cellular automata
- Mechanical artificial life
- Artificial chemistry
- Synthetic biology
- Computational neuroscience

(3) Inspiration from nature

Methods, algorithms and paradigms for computing

- Cellular automata inspired by self-reproduction
- Artificial neural networks by the functioning of the brain,
- Evolutionary computation by Darwinian evolution,
- Swarm intelligence by the behaviour of groups of organisms,
- Artificial immune systems by the natural immune system,
- Artificial life by properties of life in general,
- Membrane computing by the compartmentalised organisation of the cells, and
- Amorphous computing by morphogenesis.

adapted from L. Kari, G. Rozenberg, 2008. The many facets of natural computing. Comm. ACM 51, 10, 72-83.

Problem solving

How to find a good solution to a given problem?

- Direct calculation, straight-forward recipe
- Solution by analogy, generalisation
- Iterative solution, continuous improvement
- Cartesian method: Divide and conquer
- Heuristics and meta-heuristic algorithms
- Trial and error, random guessing

specific, exact, reliable

general, sufficing, sloppy

General problem solving

Utility is measurable, and risk affects utility (Milton Friedman, 1948)

Utility theorem: If a utility satisfies a set of axioms, then this utility can be expressed by a function (John von Neumann and Oskar Morgenstern, 1945)

Utility hypothesis: Every problem can be encoded by a utility function whose maxima are the admissible solutions (see e.g. Sutton and Barto, 1999, 2017)

We will not discuss here whether the latter is true, but will assume instead that we have already an objective function or are able to construct one.

Problem Solving as Minimisation of a Cost Function

Choosing the best option from some set of available alternatives

- Minimise energy, time, cost, risk, ...
- Maximise gains, acceptance, turnover, ...
- Averaging may be necessary (over what time scale?)
- Environmental dynamics may have effects as well
- Discrete cost (incorporating constraints):
  - admissible state: maximal gain
  - anything else: no gain
- Secondary costs for:
  - acquisition of domain knowledge, modelling, determining costs
  - testing alternatives
  - doing nothing

Problem Solving as Optimisation of an Objective Function

Objective function*:

- Cost (min)
- Energy (min)
- Risk (min)
- Quality (max)
- Fitness (max)

May be analytical, observational, relative, qualitative, compositional

---

*Note of caution:

We will sometimes aim at *minimisation*, sometimes at *maximisation*, sometimes we will speak of *optimisation* or the search for the *best* solution. It will usually be clear from context whether high or low values are to be preferred.

Problem Solving as Minimisation of a Cost Function

Example: Evolution of a walking machine

$$
\begin{aligned}
\text{Cost} \;=\; &+\alpha \;\times\; \text{weight} \\
&-\beta \;\times\; \text{speed} \\
&-\gamma \;\times\; \text{(number of steps before falling over)} \\
&+\delta \;\times\; \text{(energy consumed)} \\
&-\varepsilon \;\times\; \text{(measure of similarity to human gait)}
\end{aligned}
$$

- Costs $\rightarrow$ fitness function (low costs = high fitness) can be calculated for each candidate solution
- Use a set (population) of candidate solutions
- Evolution scheme for the candidate solutions to produce more of the good ones and discover better ones
- For different choices of α, β, γ, δ, ε the optimal solutions will be different

Meta-heuristic algorithms

- Similar to stochastic optimisation
- Iteratively trying to improve a possibly large set of candidate solutions
- Few or no assumptions about the problem (need to know what is a good solution)
- Usually finds good rather than optimal solutions
- Adaptable by a number of adjustable parameters
- On-line identification of the structural elements that can be composed into candidate solutions
- General problem: more "greedy" or more "rambling"? Exploitation or exploration?

http://en.wikipedia.org/wiki/Metaheuristic

A Selection of Topics from Natural Computation

Themes

- Evolutionary Computation
- Artificial immune systems
- Neural computation
- Amorphous computing
- Swarm intelligence
- Harmony search
- Cellular automata
- Artificial life
- Membrane computing
- Molecular computing
- Quantum computing

Methods

- Genetic algorithms
- Genetic programming
- Gravitational search
- Central force optimisation
- Electromagnetism-like optimisation
- Water drop algorithm
- Spiral dynamics algorithm
- Simulated annealing
- Chemical reaction optimisation
- Artificial physics optimisation
- Ant colony optimization

Course: Natural Computing (2018/19)

1. Introduction

1.2. Optimisation

J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

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

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

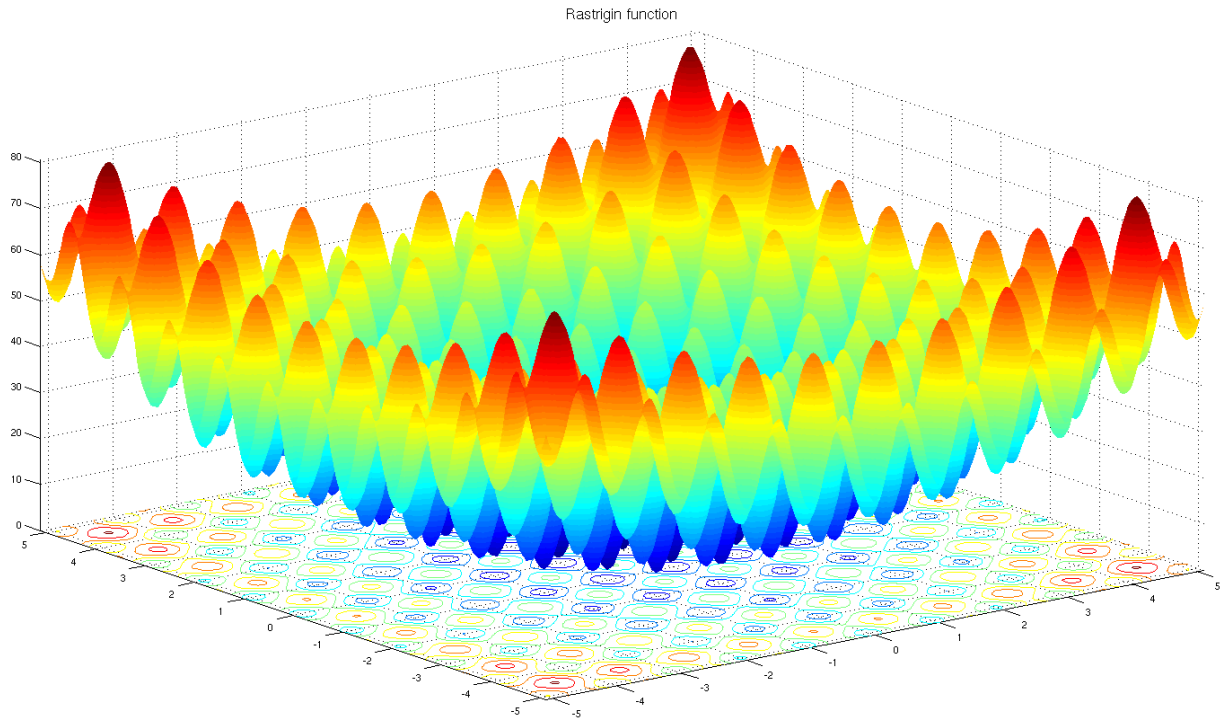
For which $x \in X$ the value $F(x)$ is smallest?

In order to avoid mathematical difficulties, we will assume that

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

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

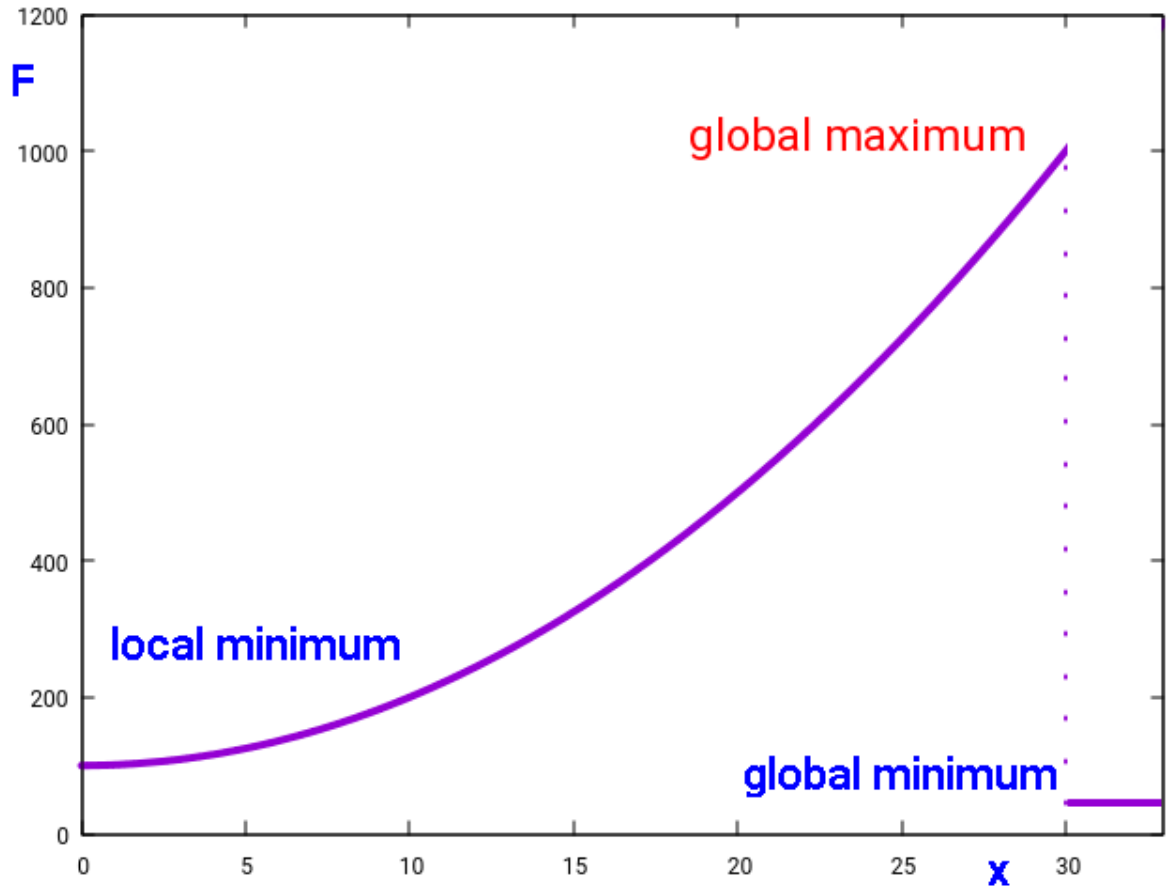
Searching for improvement among near-by states?



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

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

Searching for improvement among near-by states?



A few relevant terms

- Optimum:
  - Maximum (in maximisation problems)
  - Minimum (in minimisation problems)
- Extremum: a point that its 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

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

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
- 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)

Course: Natural Computing (2018/19)

1. Introduction

1.3. Examples from other contexts

J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

Hard satisfiability problems: The 3SAT problem

- Consider random 3-Conjunctive Normal Form sentences
- Example:

$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

m: number of clauses, e.g. $(B \vee \neg A \vee \neg C)$, $(E \vee \neg D \vee B)$

n: number of symbols, e.g. A, B, C

- Assign to each symbol a value (TRUE or FALSE) such that the sentence is true (or show that this is impossible)
- (un)satisfiability is easy to show, if m/n is small (large)
- Problems with $m/n \approx 4.3$ are really hard.

The WalkSAT algorithm

- Choose a clause that is wrong
- with probability p
  - flip any symbol that occurs in this clause
- with probability $1 - p$
  - flip the symbol that leads to the maximal number of correct clauses in the sentence.
- Repeat unless MAX_ITERATIONS is reached

The WalkSAT algorithm

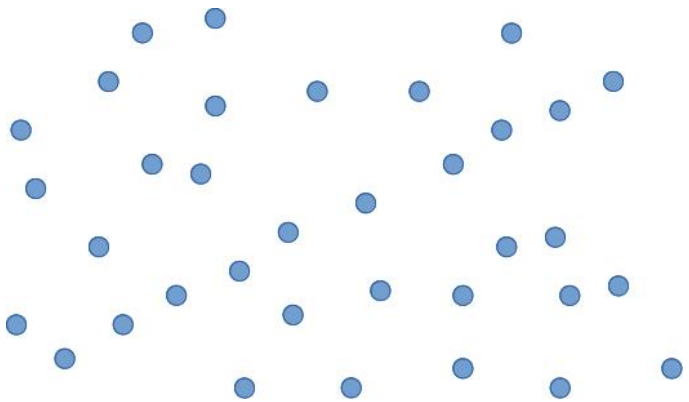
- Incomplete, local search algorithm
- Works well for $m/n < 3$ (and for problems with small n)
- Cannot show unsatisfiability
- Evaluation function: Minimize number of unsatisfied clauses
- WalkSAT uses the following approach:
  - hill-climbing w.r.t. the evaluation function and w. prob. $1 - p$
  - mutations w. prob. p
- A good balance between greediness and randomness is needed

Random sample consensus (RANSAC)

- Often used in computer vision
- General framework for model fitting in the presence of outliers
- Question: How to fit a model to points
- Answer:
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are "close" to the model and reject the rest as outliers
  - Do this many times and choose the best model

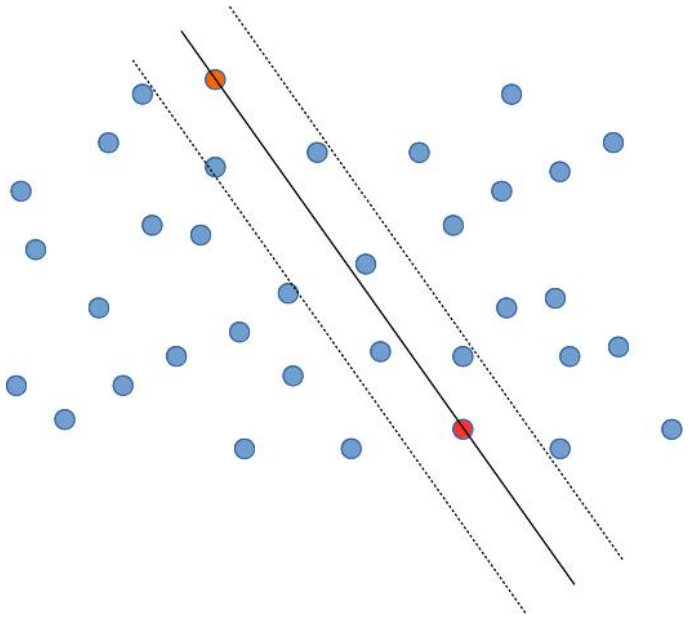
M.A. Fischler, R.C. Bolles. Random Sample Consensus: A paradigm for model fitting with appli- cations to image analysis and automated cartography. Comm. of the ACM 24, 381-395, 1981.

RANSAC: Example



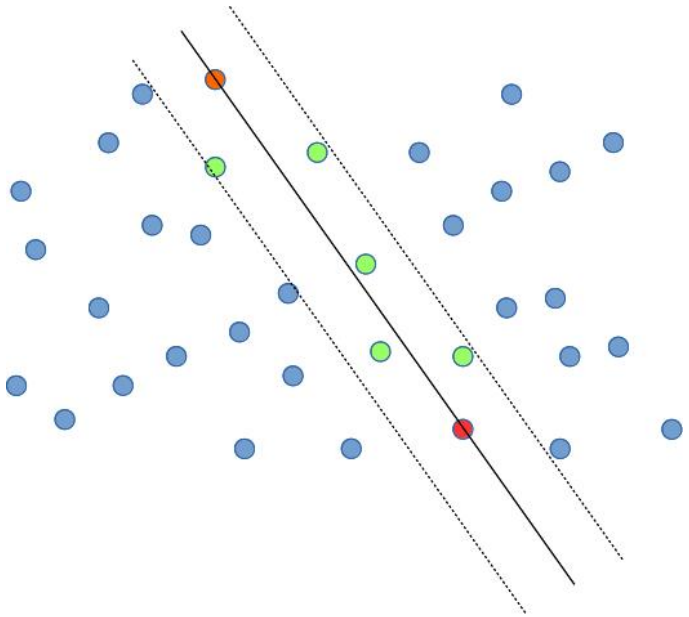
Does a given point cloud contain a line?

RANSAC: Example



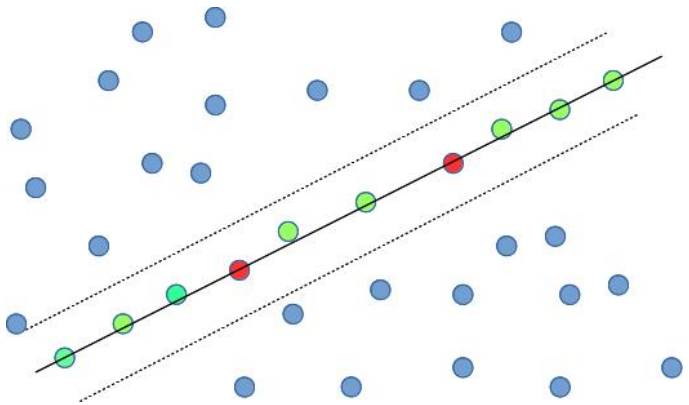
Choose randomly two fitting points and a threshold.
Fit the model to these two points.

RANSAC: Example



Check how many points appear within
the threshold range near the model

RANSAC: Example



If there really is a line, then for some starting points there will be many “inliers”.

Random sample consensus (RANSAC)

- Simple and general, applicable to many different problems
- Often works well in practice, but needs at least 50% inliers (modern versions do much better)
- Several parameters to tune
- RANSAC uses the following approach:
  - random selection
  - evaluation [break if good enough]
  - repeat [unless MAX_ITERATIONS are reached]
- No need to remember RANSAC for this course, but keep the trial-and-evaluation approach in mind.

Conclusions from WalkSAT and RANSAC

- Random search may just work, in particular if
  - problems are not too big and too complex
  - it is combined with greediness
- We need an evaluation function (fitness, cost, ...)
- The gradient of this function may not be defined or not be indicative of the global optimum

Problems may be intrinsically difficult

- A satisfying assignment may not exist: Continue to search?
- Not every line-like arrangement of points is meaningful: Can the algorithm avoid false positives?

Outlook

The next unit will consider

- Hill climbing
- Simulated annealing
- Evolutionary algorithms and evolutionary strategies

After this we will move on to a choice of the metaheuristic algorithms