Course: Natural Computing

*1. Introduction
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

$\Uparrow$

$\Downarrow$

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.

see also: D. Silver e.a. (2021) Reward is enough. *Artif. Intell*. 103535

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 timescale?)
- 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

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

- 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 optimisation

Outlook

The next unit will consider discrete metaheuristic optimisation algorithms, in particular

- Simulated annealing
- Genetic algorithms