Natural Computing

Lecture 1

Michael Herrmann
mherrman@inf.ed.ac.uk
phone: 0131 6 517177
Informatics Forum 1.42

Problem Solving in Nature

Natural Computation

- Physics
- Chemistry
- Biology
- Geology
- Astronomy

- Abacus
- Slide ruler
- Gear-driven calculating machines
- Relays, vacuum tubes, transistors ...
- Turing machines

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 synthesize 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) Synthesizing 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

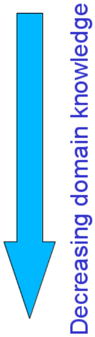
(3) Inspiration from nature

- Cellular automata inspired by self-reproduction, Neural computation by the functioning of the brain,
- Evolutionary computation by the Darwinian evolution of species,
- 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 compartmentalized organization of the cells, and
- Amorphous computing by morphogenesis.

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

Inspiration for problem solving
How to find good solutions?

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



Problem Solving is Optimisation of an Objective Function

Objective function:

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

May be analytical, observational, relative, qualitative, compositional

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

Problem Solving as Minimisation of a Cost Function

Choosing the best option from some set of available alternatives

- Minimize energy, time, cost, risk, . . .
- Maximize gains, acceptance, turnover, . . .
- Discrete cost:
  - admissible goal state: maximal gain
  - anything else: no gain
- Secondary costs for:
  - acquisition of domain knowledge
  - testing alternatives
  - doing nothing
  - determining costs

Optimization by “Hill-Climbing”
Searching for better options among near-by states

Meta-heuristic algorithms

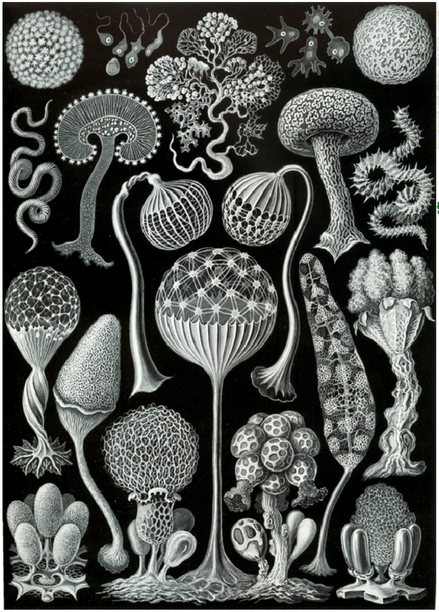
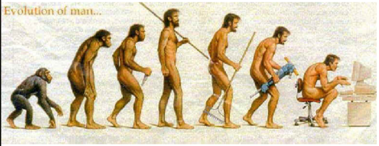
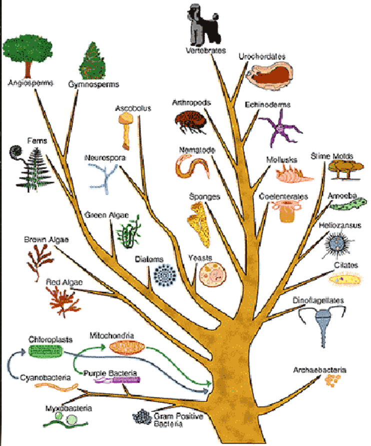
- Similar to stochastic optimization
- Iteratively trying to improve a possibly large set of candidate solutions
- Few or no assumptions about the problem (need to know what is a good solution)
- Usually finds good rather than optimal solutions
- Adaptable by a number of adjustable parameters
- On-line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

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

Syllabus

- Part 1. Genetic algorithms (GAs)
  - The canonical genetic algorithm
  - The schema theorem and building block hypothesis
  - Formal analysis of genetic algorithms
  - Methodology for genetic algorithms
  - Designing real genetic algorithms
  - Multi-objective optimization
- Part 2. Genetic programmingEvolving programs and intelligent agents
- Part 3. Ant colony optimisation (ACO) & Particle swarm optimisation (PSO)
- Part 4. Artificial life
- Part 5. Material computing

Genetic algorithms



Evolution of man...

Vertebrates
Urochordates
Angiosperms
Gymnosperms
Ascobolus
Arthropods
Echinoderms
Ferns
Neurospora
Nematode
Mollusks
Slime Molds
Sponges
Coelenterates
Amoeba
Green Algae
Heliozansus
Brown Algae
Diatoms
Yeasts
Cilates
Red Algae
Dinoflagellates
Chloroplasts
Mitochondria
Cyanobacteria
Purple Bacteria
Archaebacteria
Myxobacteria
Gram Positive Bacteria

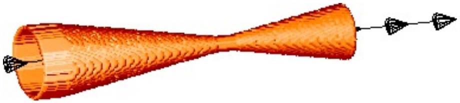
Paralipomena

- Theory of natural evolution: Genetics, genomics, bioinformatics
- The Philosophy of Chance (Stanislaw Lem, 1968)
- Memetics (R. Dawkins: The Selfish Gene, 1976)
- Neural Darwinism – The Theory of Neuronal Group Selection (Gerald Edelman, 1975, 1989)
- (artificial) Immune systems
- Evolution of individual learning abilities, local heuristics
- Computational finance, markets, agents

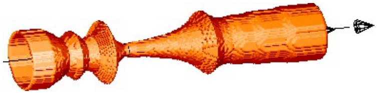
An early example of artificial evolution

Experimental contour optimization of a supersonic flashing flow nozzle (1967-1969)
Hans-Paul Schwefel

Initial shape



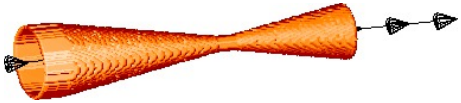
Evolved shape



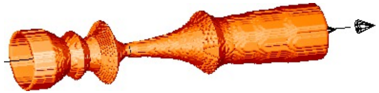
An early example of artificial evolution

Experimental contour optimization of a supersonic flashing flow nozzle (1967-1969) Hans-Paul Schwefel

Initial shape



Evolved shape

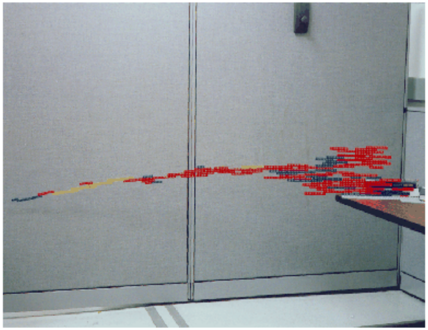


Genetic Algorithms

- Global search heuristics
- Technique used in computing
- Find exact or approximate solutions to optimization problems

Applications in

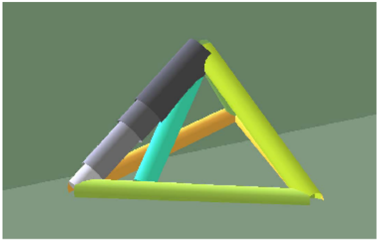
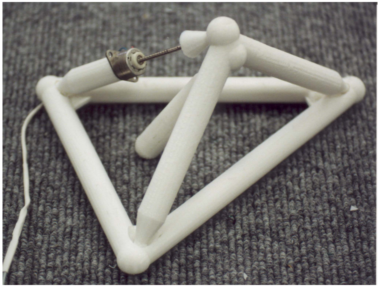
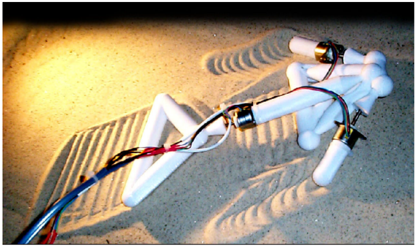
- Bioinformatics
- Computational science
- Engineering
- Robotics
- Economics
- Chemistry
- Manufacturing
- Mathematics
- Physics



Problem Solving by Genetic Algorithms

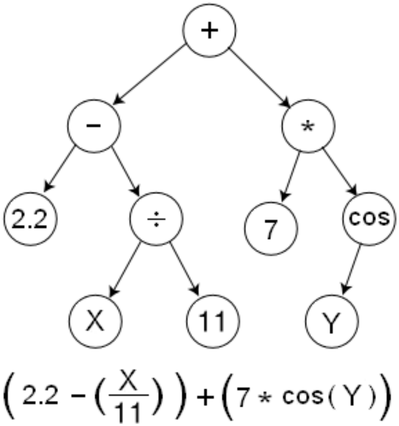
- Find a representation of solutions
  e.g.: $S =$ (lengths of legs, angle limits, motor power, ...)
  [expressed as a bit string]
- Define an objective function $\rightarrow$ fitness function (to be maximised)
  that can be calculated for each S
- Generate a population of candidate solutions (bag of strings)
- Evolution scheme for the solutions:
  - Evaluate the candidate solutions
  - Choose (preferentially) high-fitness solutions
  - Modify some of them
  - Start again unless a termination criterion is met

The Golem Project
Hod Lipson & Jordan B. Pollack (2000)

Genetic Programming (GP)

- Evolutionary algorithm-based methodology inspired by biological evolution
- Finds computer programs that perform a user-defined task
- Similar to genetic algorithms (GA) where each individual is a computer program
- Optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.

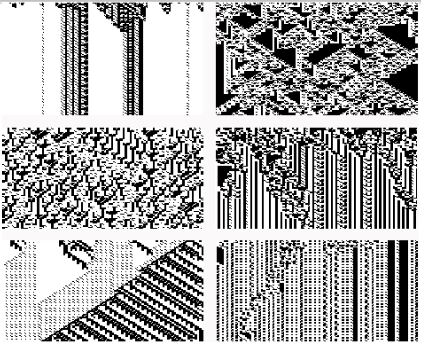

+
-
*
2.2
÷
7
cos
X
11
Y
$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$

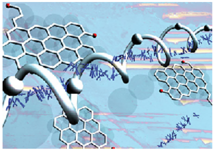
Evolutionary Computation (EC)

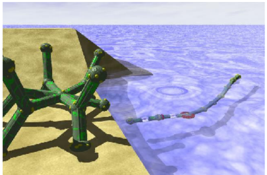
- Genetic algorithms: Solution of a problem in the form of strings of numbers using recombination and mutation
- Genetic programming: Evolution of computer programs
- Evolutionary programming: Like GP, but only the parameters evolve
- Evolution strategies: Vectors of real numbers as representations of solutions

Natural Computation (NC)

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







Reading suggestions

- Leandro Nunes de Castro (2007). Fundamentals of Natural Computing: An Overview. Physics of Life Reviews 4: pp.1–36.
- L. N. de Castro: Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications. Chapman & Hall/CRC, 2006.
- G. Rozenberg, T. Bäck, J. N. Kok (Editors) Handbook of Natural Computing Springer Verlag, 2010. [for reference only, don't even think of buying it]
- Melanie Mitchell: An Introduction to Genetic Algorithms. MIT Press, 1996.
- Xin-She Yang: Nature-Inspired Metaheuristic Algorithms. Luniver Press 2010.
- J. R. Koza: Genetic Programming: On the programming of computers by means of natural selection, MIT Press, 1992.
- Wolfgang Banzhaf et al.: Genetic Programming: An Introduction. Morgan Kaufmann, 1988.
- Eric Bonabeau, Marco Dorigo and Guy Theraulez: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.

Course organization

Tuesday & Friday 15:00 – 15:50 at AT LT2 Assignments: two assignment together worth 30% (10% + 20%) of the course mark, to be handed in at the end of Week 5 and the end of Week 9. on 27 Oct / 24 Nov (both: Thursdays 4pm)

Exam: worth 70% of the course mark, taken at the end of Semester 2. (Visiting students can take the exam at the end of Semester 1.)

michael.herrmann@ed.ac.uk

phone: 0131 6 517177

Informatics Forum 1.42

Tutorials (to be organised until next week)