OPTIMIZATION PROBLEMS

An instance $I = (F, c)$, where:

* F is the *set of feasible solutions,* and
* c is a *cost function,* assigning a cost value to each feasible solution; $c : F \rightarrow R$

The solution of the optimization problem is the feasible solution with optimal (minimal/maximal) cost.

A feasible solution can be described by specific settings of *problem parameters* or *variables.*

* If the variables can assume *discrete* values, the problem is called a *combinatorial optimization problem*.
* If the variables are continuous, the problem is a *continuous optimization problem.*

COMPUTATIONAL COMPLEXITY (1)

* *Computational complexity:* an abstract measure of the time and space necessary to execute an algorithm as function of its "input size".
* *Time complexity* is expressed in *elementary computational steps.* For example: an addition (or multiplication, pointer indirection, etc.) is one step.
* *Space complexity* is expressed in *memory locations* (e.g. bits, bytes, words).
* *Big-O notation:*

  $f = O(g)$, if two constants n_0 and K can be found such that:

  $$\forall n \geq n_0 : f(n) \leq K \cdot g(n)$$

  Example:

  $$2n^2 = O(n^2)$$

COMPUTATIONAL COMPLEXITY (2)

Relevant growth rates for the time complexity:

* *polynomial* vs. *exponential*
* *linear* vs. *quadratic*
* *sublinear*

INTRACTABLE PROBLEMS

A complexity class that is often encountered in optimization, is the class of *NP-complete* (NP = nondeterministic polynomial) problems. A precise definition involves several subtle issues. Here, it suffices to know that for these problems only exponential-time algorithms are known and that polynomial-time algorithms are very unlikely to be found.

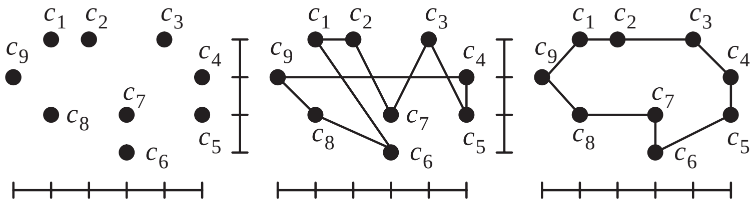
Problems for which only exponential-time algorithms are known, are called *intractable*. For intractable problems:

* Exact solutions can only be found when the problem size is small.
* For larger problem sizes, one can use:
  + *approximation algorithms:* they can e.g. guarantee a solution within 20% of the optimum.
  + *heuristics:* nothing can be said a priori about the quality of the solution. Heuristics could be based on computational intelligence.

THE TRAVELING SALESMAN PROBLEM (1)

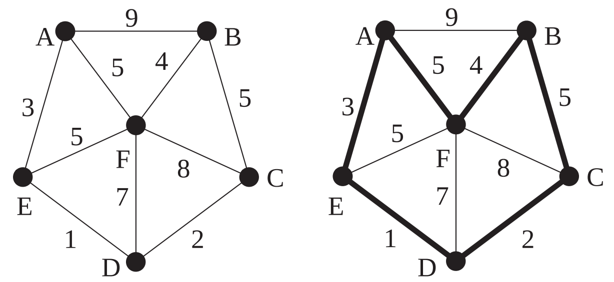
* A typical example of an NP-complete problem.

PROBLEM DEFINITION:
Find the shortest *tour* that visit all cities in a given set exactly once.



Euclidean version

THE TRAVELING SALESMAN PROBLEM (2)



Graph version

COMPUTABILITY

* A function is called *computable* if it can be computed by a *Turing machine.*
* A Turing machine is a theoretical model of a computer; it consists of a tape (data memory), a simple instruction set (read or write a bit on the tape, advance tape, test current bit value) and a program.
* Any electronic computer can be emulated by a Turing machine at the cost of a polynomial-time overhead. So, problems that are intractable for a Turing machine will be intractable for any other computer.
* Some problems are incomputable: no program can be written to solve it, not even one with an exponential or higher time complexity (e.g. the *halting problem*).
* Some people (e.g. Roger Penrose) claim that computation is insufficient to model the brain.

COMPUTATIONAL INTELLIGENCE

* Special case of *artificial intelligence* (AI), nowadays often called *classical AI.*
* Classical AI is more oriented towards models of reasoning. Think of expert systems, playing games like chess, etc. A medical expert system will e.g. assist a doctor in determining a patient's disease by starting from symptoms and using if-then-else-type rules.
* *Computational intelligence* (CI) refers to techniques inspired by phenomena in nature. It is also called *soft computing.*
* CI is much more involved with numerical computations rather than symbolic ones. It is, in a sense, closer to signal processing/electrical engineering.

COMPUTATION IN LIFELESS NATURE

* Question in book: does a table execute an algorithm because it organizes its atoms in some specific way?
* There is a parallel between the tendency of physical systems to move towards a state of minimal energy and optimization problems. *Simulated annealing* is based on this principle.
* Learning in neural networks can be based on simulated annealing. These type of neural networks are called *Boltzmann machines.*

SIMULATED ANNEALING

Inspired by material cooling down slowly and settling in minimal energy state.

Analogies:

* Energy ↔ cost function
* Molecule movement ↔ movement in search space.
* Temperature ↔ control parameter *T.*

Move strategy for f and $g = m(f)$:

* $\Delta c = c(g) - c(f)$
* If $\Delta c \leq 0$, always accept transition to *g.*
* If $\Delta c > 0$, accept with probability $e^{\frac{-\Delta c}{T}}$.

SIMULATED ANNEALING: CODE

```
int accept(struct feasible_solution f, g)
{
  float Δc;

  Δc ← c(g) − c(f);
  if (Δc ≤ 0)
    return 1;
  else return (e^(−Δc/T) > random(1));
}
```

```
simulated_annealing()
{
  struct feasible_solution f, g;
  float T;

  f ← initial_solution();
  do {
     do {
        g ← "some element of N(f)";
        if (accept(f, g))
          f ← g
     while (!thermal_equilibrium());
     T ← new_temperature(T);
  while (!stop());
  "report f";
}
```


COMPUTATION IN LIVING NATURE (1)

Three types of developments take place in living nature[1]:

1) *Phylogeny*, the temporal evolution of the genetic code.

  + The genetic code changes from generation to generation through *mutation* (asexual reproduction) and *recombination* (sexual reproduction).
  + A *survival-of-the-fittest* principle leads to improved performance of the species and adaptation to the environment.
  + The principle is used in the branch of computational intelligence called *evolutionary computation* (EC). A well-known EC technique is the use of *genetic algorithms* (GAs).

[1] Mange, D. and M. Tomassini (Eds.), *Bio-Inspired Computing Machines, Towards Novel Computational Architectures*, Presses Polytechniques et Universitaires Romandes, Lausanne, (1998).

COMPUTATION IN LIVING NATURE (2)

2) *Ontogeny*, the growth of a multicellular organism from a single cell, including *differentiation*. This phenomenon was a source of inspiration for *hardware evolution.*

3) *Epigenesis,* the ability of an organism to modify parts of its system as a result of interaction with its environment. Think e.g. of the nervous system and the immune system. This aspect of natural computation has clearly led to *neural networks.*

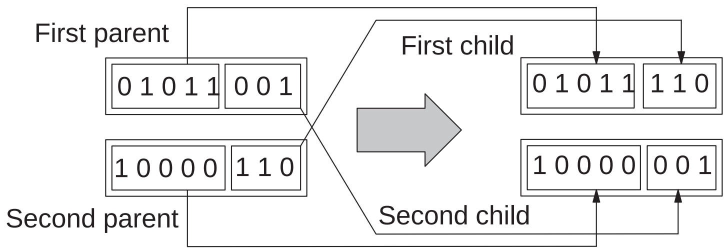
Additionally, there is the actual "information processing", the use of the nervous system in various situations (from reflexes, to well-planned behavior).

GENETIC ALGORITHMS

Principles:

* Based on analogy with evolution process in nature.
* Works with a *population* of feasible solutions, instead of a single feasible solution.
* Each feasible solution is encoded in a linear data structure, usually a bit string, called a *chromosome.*
* Two *parent* chromosomes are combined by *crossover* to form one/two *child* chromosomes.
* Optimization based on "survival of the fittest": prefer parents with better costs for mating.

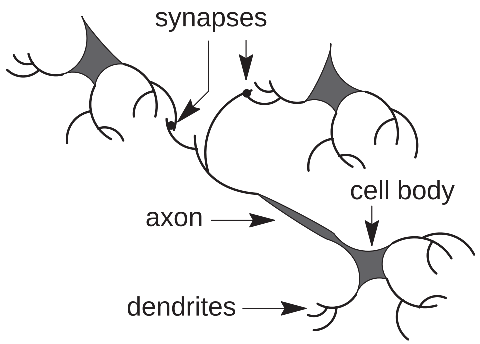
GENETIC ALGORITHMS: ILLUSTRATION



GENETIC ALGORITHMS: CODE

```
genetic()
{
   pop ← ∅;
   for (i ← 1; i ≤ pop_size; i ← i + 1)
      pop ← pop ∪ {"chromosome of random feasible solution"};
   do {
      newpop ← ∅;
      for (i ← 1; i ≤ pop_size; i ← i + 1) {
         parent1 ← select(pop);
         parent2 ← select(pop);
         child ← crossover(parent1, parent2);
         newpop ← newpop ∪ {child};
      }
      pop ← newpop;
   } while (!stop());
   "report best solution";
}
```

NATURAL NEURONS (1)



NATURAL NEURONS (2)

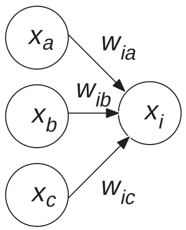
* The human brain has about 10^{11} neurons.
* Each neuron connects to about 10^4 other neurons.
* A cell body has a diameter of about 10 microns; an axon can be as long as 1 cm.
* Signals from one neuron to another are carried by *neurotransmitters* which result in a charge transfer.
* A neuron will *spike* when its charge passes some value. The spiking rate is a measure of neuron activity.
* A key issue in *learning* is the modification in signal transfer degree at a synapse.

ARTIFICIAL NEURONS

A neuron is modeled by an element:

* It has an output value in a limited range, e.g. from –1 to 1 or from 0 to 1; sometimes the output can only have a discrete value (0 and 1); note that the spiking rate is modeled by a level.
* It takes its inputs from other neuron outputs or from external inputs;
* Its output is calculated as the weighted sum of the inputs that is passed through a *limiting* (or activation) *function f*:

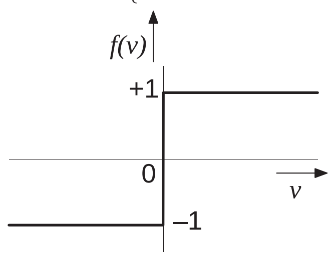
$$x_i = f(v_i) \quad v_i = \sum_j w_{ij} x_j$$



LIMITING FUNCTION EXAMPLES

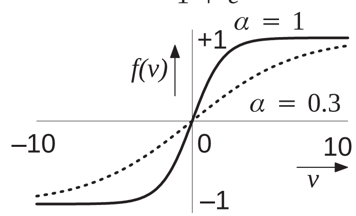
* *Threshold function:*

$$f(v) = \begin{cases} -1, & \text{if } v < 0 \\ 1, & \text{if } v \geq 0 \end{cases}$$



* *Sigmoid function:*

$$f(v) = \frac{1 - e^{-\alpha v}}{1 + e^{-\alpha v}}$$



ARTIFICIAL NEURAL NETWORKS

* Given the model of the artificial neuron, many different types of neural networks can be built: without and with feedback loops (*feedforward* resp. *recurrent* networks), with some specific structure (layered, hierarchical), etc.
* Neural networks are mainly used for *learning* some function by means of examples. This amounts to finding the correct weights.
* There are two types of learning: *supervised* and *unsupervised*.
* In some applications the network weights are fixed a priori; the learning property is not used. Instead, the final state after execution of the network with given weights has a special meaning (it e.g. represents the solution to some combinatorial optimization problem).

FUZZY LOGIC (1)

* Consider a *universe of discourse*, e.g. the set of all positive real numbers. Indicate this set by X.
* A subset $A \subset X$ can be specified by a Boolean *membership function* $\mu_A : X \to \{0,1\}$:

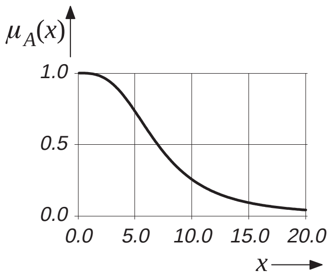
$$\mu_A(x) = \begin{cases} 0, & \text{if } x \notin A \\ 1, & \text{if } x \in A \end{cases}$$

* Such sets are clearly defined; they are called *crisp.*

FUZZY LOGIC (2)

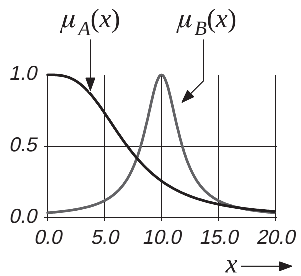
* There is a necessity to define sets whose membership is not clearly defined. Think e.g. of the “set of small numbers”, which in some context may mean numbers more-or-less smaller than 5. Is 4.99 small and 5.01 no longer small?
* Fuzzy logic solves this issue by having membership functions that map to the entire interval [0,1] instead of to 0 or 1. The value of $\mu_A(x)$ for some x expresses the degree of membership.

Set of small numbers:

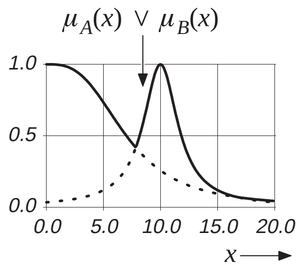


FUZZY LOGIC (3)

* Consider a second membership function, μ_B for “all numbers that are close to 10”:



* Then a logic system can be used to combine membership functions to obtain new ones. For example, the “set of numbers that are either small or close to 10”:



FUZZY LOGIC (4)

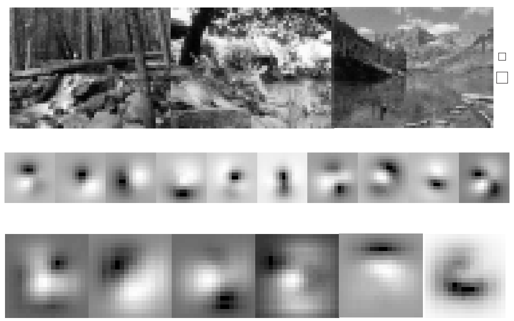
* Fuzzy logic can be used to build systems for:
  + approximate reasoning (fuzzy if-then-else rules)
  + control applications (e.g. capture the behavior of a human operator by means of fuzzy rules)
* Fuzzy logic can be combined in many ways with neural networks:
  + the neurons themselves can be made to have a fuzzy behavior;
  + a neural network can be used for preprocessing signals to be input to a fuzzy system;

FITNESS

* *Fitness* is seen as the ability of an organism to solve problems that it encounters.
* The book uses fitness in relation with the ability to compress data; a better compression is associated with higher fitness scores. However, optimal solutions are not always necessary: approximate solutions may be sufficient.
* Suppose that a collection of data is given.
  + If the data is random, the most compact description of the data is the data itself.
  + Otherwise, a better description may be given by a combination of a compaction program and a reduced data set.
  + The *minimum description length* gives the lower bound on the description.

RECEPTIVE FIELD EXAMPLE (1)

* Take 16×16 and 20×20 patches from some images can be used as filters:

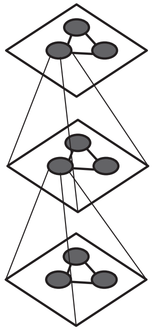


RECEPTIVE FIELD EXAMPLE (2)

* The output of 45 filters give a *feature vector* for some observation (white cross);
* Filter: multiply pixel value in patch with value in image and sum.

ARCHITECTURES

* *Locality* is important in a "massively parallel system" such as a brain.
* On the other hand, long distance communication is required as well.
* A *hierarchical architecture* satisfies both requirements.
* The time to accomplish a task increases with increasing levels of abstraction in hierarchy.



TIME SCALES

Temporal scale	Primitive	Example
1 ms	Neuron spike	
10 ms	Neural circuit	
50 ms	Neural act	Noticing a stimulus
300 ms	Physical act	Moving the eyes
2 sec	Simple task	Saying a sentence
10 sec	Complex task	Moving in speed chess

OVERVIEW

* *Core concepts:* fitness (information theory, minimum description length), programs (heuristic search techniques), data (data compression, eigen vectors), dynamics (linear and nonlinear systems), optimization (continuous and discrete).
* *Memories:* content-addressable memories (Hopfield, Kanerva), supervised learning (multilayer perceptron, recurrent networks), unsupervised learning (principal components, Kohonen networks).
* *Programs:* (hidden) Markov models, reinforcement learning.
* *Systems:* genetic algorithms, genetic programming.