Evolutionary Algorithms

Mihai Oltean
moltean@cs.ubbcluj.ro
www.cs.ubbcluj.ro/~moltean

Structure

- Simulating the nature
- Overview of EAs
  - What are
  - Why
  - Darwin’s theories
  - Search space / Local and global solutions
  - Ingredients
  - Example – *n*-queens problem
  - What you should expect from them
  - Strength and weaknesses
  - No Free Lunch

Simulating the Nature

- We can build:

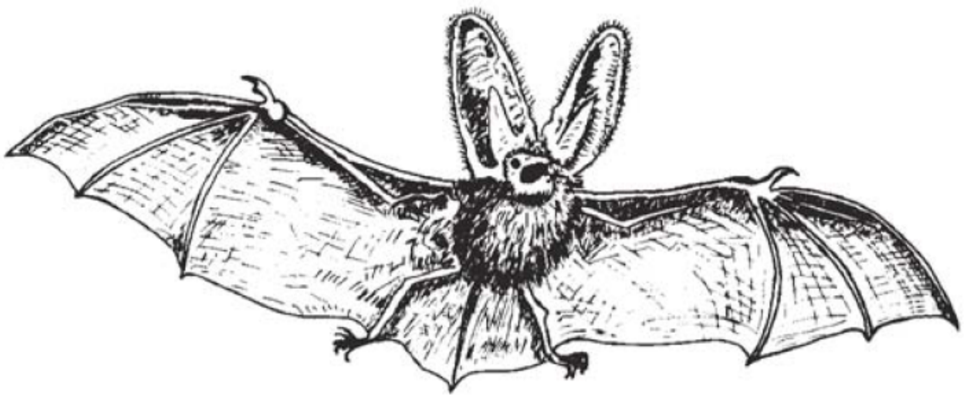
- Machines that simulate the nature (ANNs simulate brain).

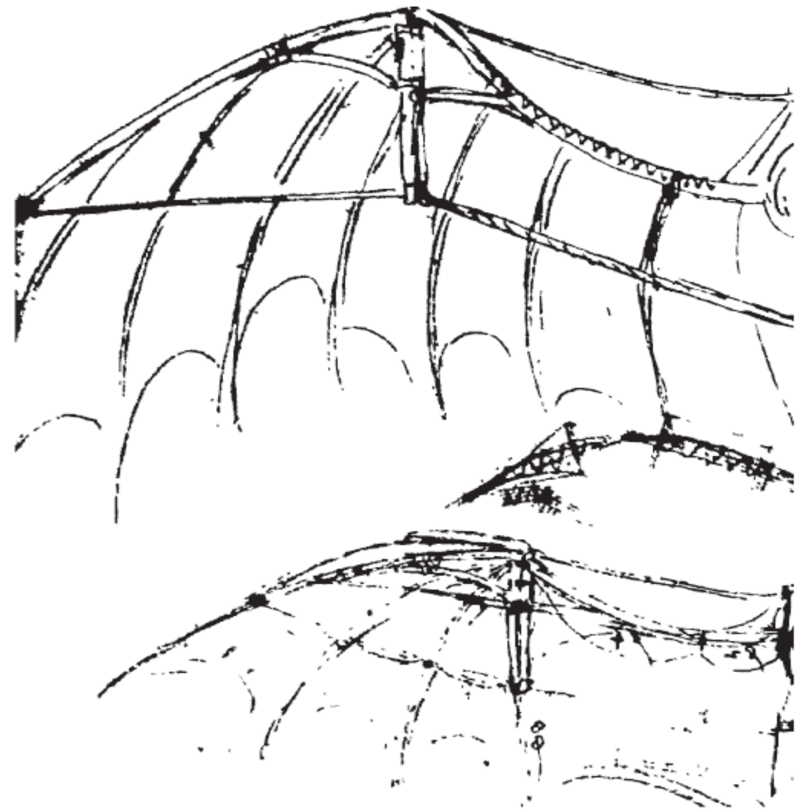
- Algorithms that simulate the nature (EAs simulate evolution).

Machines mimicking the nature

- Flying machines
- DNA computers
- Membrane Computers
- …

The bat - Biological model simply to be mimicked.

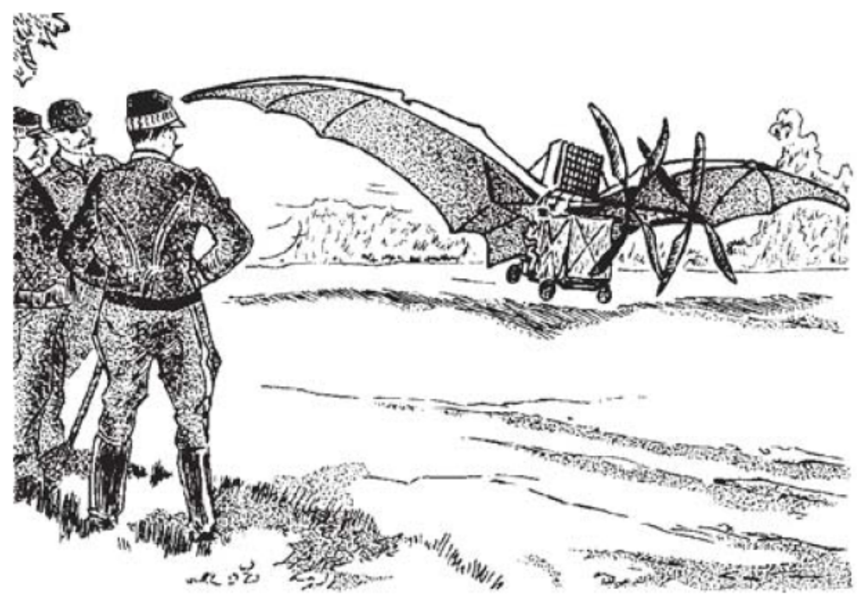




Leonardo da Vinci – part of a sketch for a flying machine.

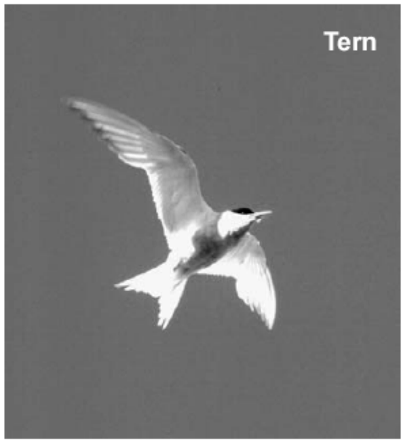
Otto Lilienthal on August 16th, 1894

The glider mimics a bird with spread wing tips.



Avion III makes only jumps!

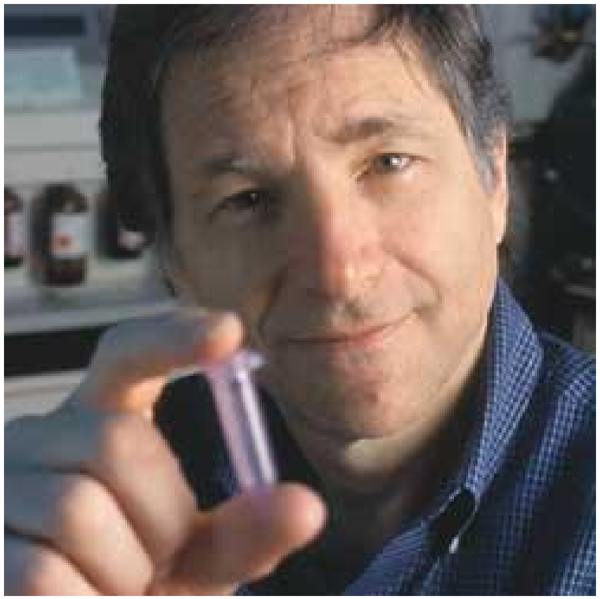
Up to date simulations

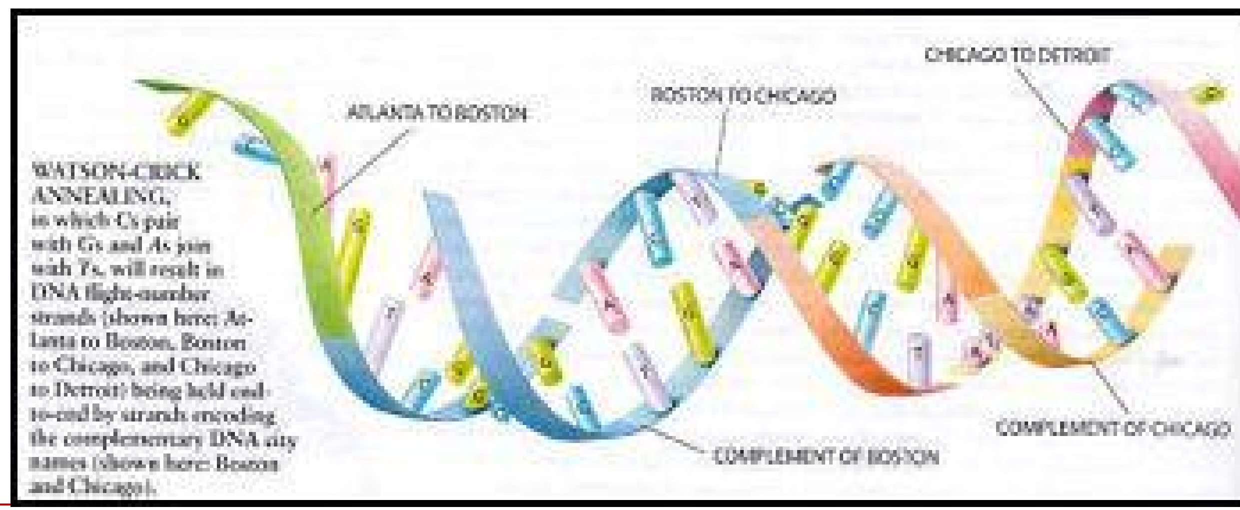


Festo – smart bird

- http://www.youtube.com/watch?v=nnR8fDW3Ilo

LEONARD ADLEMAN
FIRST DNA-BASED COMPUTER (1994)





Simulating Algorithms

- Brain – ANN
- Evolution - EAs

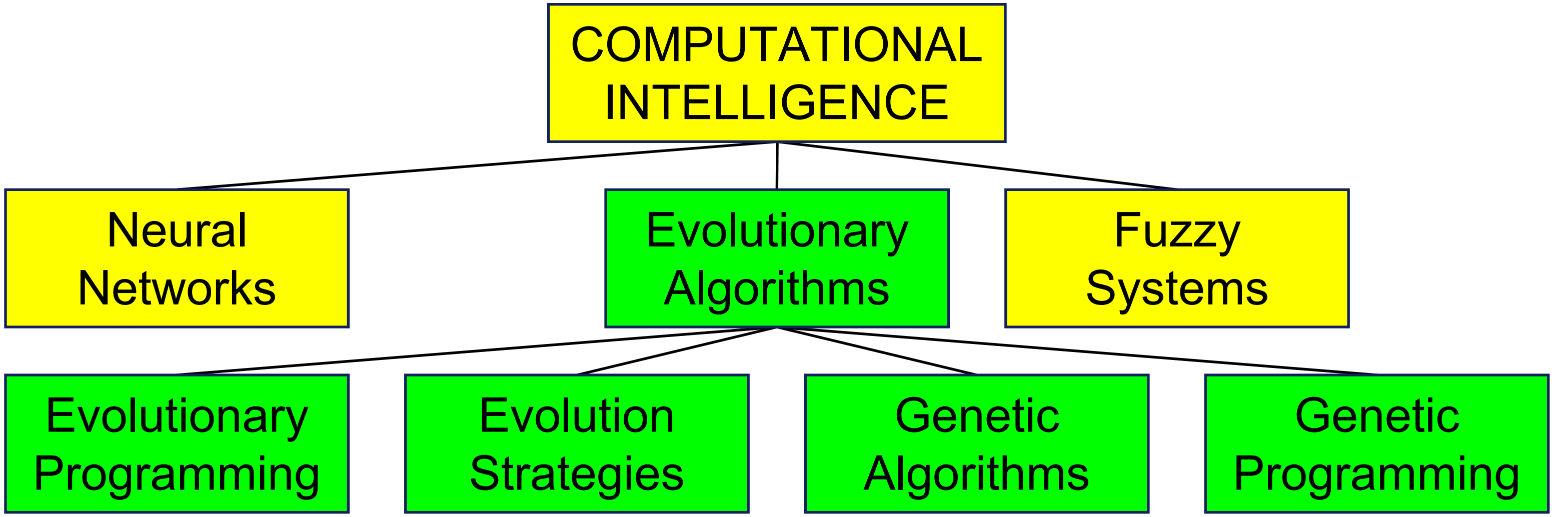
What are Evolutionary Algorithms?

- = subfield of AI.
- Can be recognized from:
  - iterative progress, growth or development
  - population based
  - guided random search
  - parallel processing
  - often biologically inspired

Darwin + Modern Genetics

- Fittest survive longest.
- In the reproduction the chromosomes of offspring are a mix of their parents.
- Characteristics, encoded into genes are transmitted to offspring and tend to propagate into new generations.
- An offspring’s characteristics are partially inherited from parents and partially the result of new genes created during the reproduction process.

Computational Intelligence Taxonomy



EVOLUTIONARY COMPUTATION HISTORY

- L. Fogel 1962 (San Diego, CA): *Evolutionary Programming*
- J. Holland 1962 (Ann Arbor, MI): *Genetic Algorithms*
- I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany): *Evolution Strategies*
- J. Koza 1989 (Palo Alto, CA): *Genetic Programming*
- *Other ~10^4 paradigms and techniques…*

…

- Evolutionary algorithms generally involve techniques implementing mechanisms such as:
  - reproduction,
  - mutation,
  - recombination,
  - natural selection.

The Metaphor

NATURAL EVOLUTION		PROBLEM SOLVING
Individual	⟷	Candidate Solution
Population	⟷	Set of solutions
Chromosome	⟷	Encoding of a solution
Gene	⟷	Part of the encoding
Fitness	⟷	Quality
Crossover and mutation	⟷	Search operators
Environment	⟷	Problem

Why EC

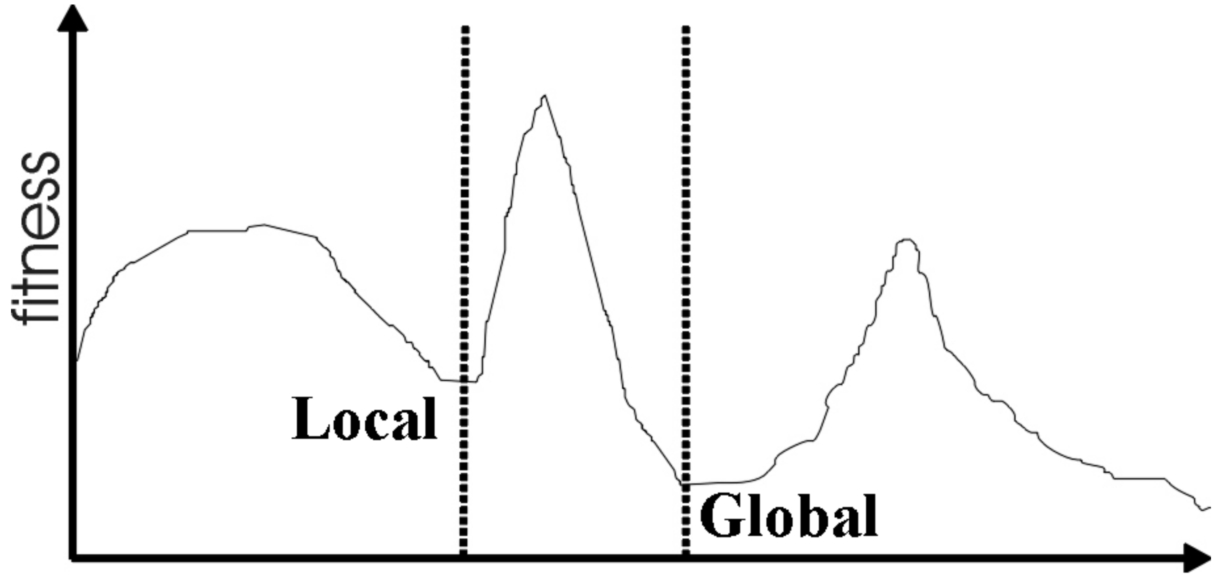
- Nature solved many problems, so any algorithm showing the same behaviour might be good
- EA can handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure.
- EA are very robust to time-varying behavior, even though they may exhibit low speed of convergence

Search space

- The set of all possible solutions
- One measure of the complexity of the problem is the size of the search space
- Crossover and mutation implement a random walk through search space
- Walk is random because the crossover and mutation are non-deterministic
- Adding selection we obtain a direct search aiming to maximize quality of solutions.

Local and global optima

- local optima
  - are better than their neighbors
  - but not as good as the global optimum

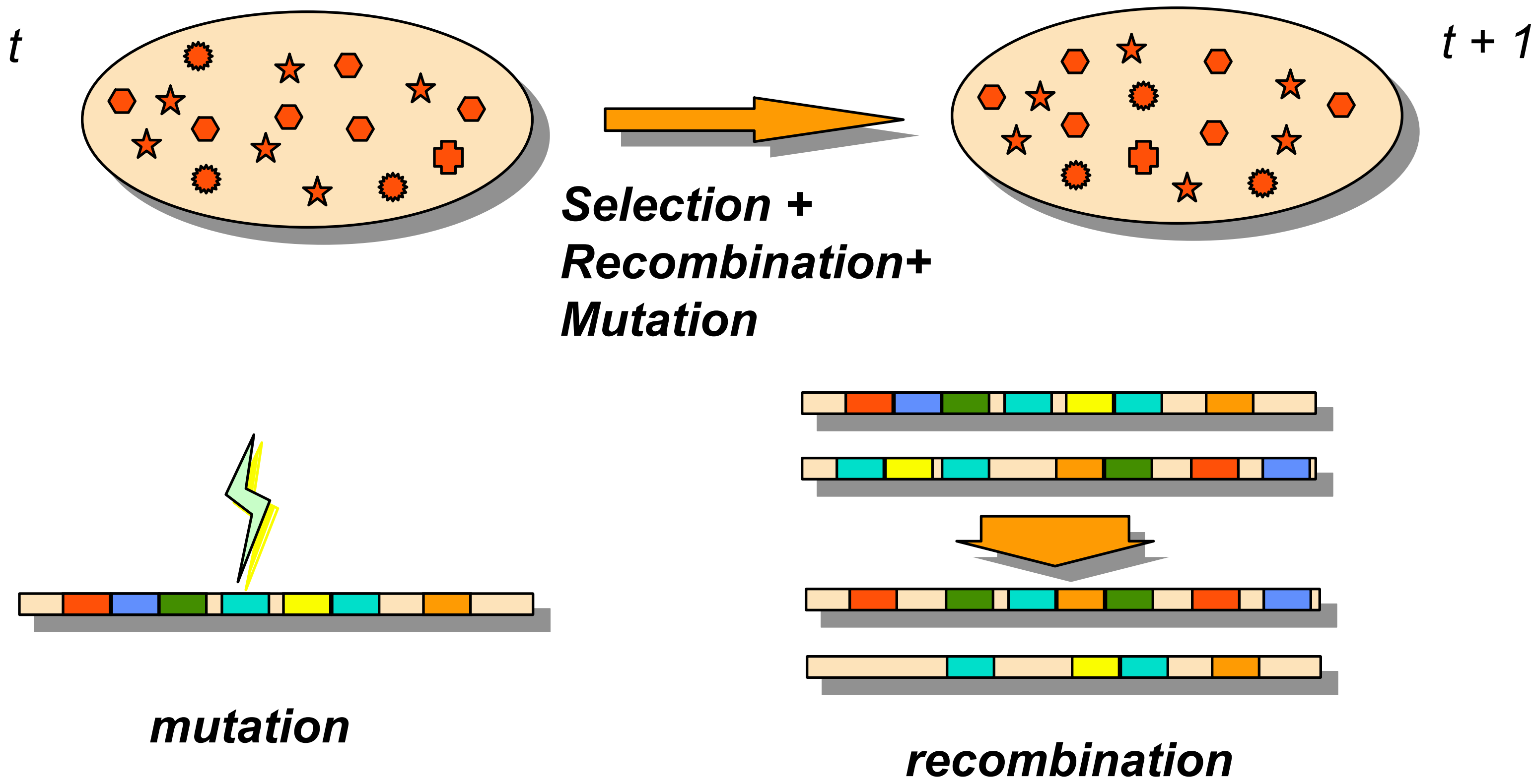


GLOBAL AND LOCAL SEARCH

- Local search
  - Looking for solutions near to the existing solutions in the search space (also called local-optimal solutions).
  - Crossover usually does this (in nature).

- Global search
  - Looking for solutions in the entire search space.
  - Mutation usually does this (in nature).

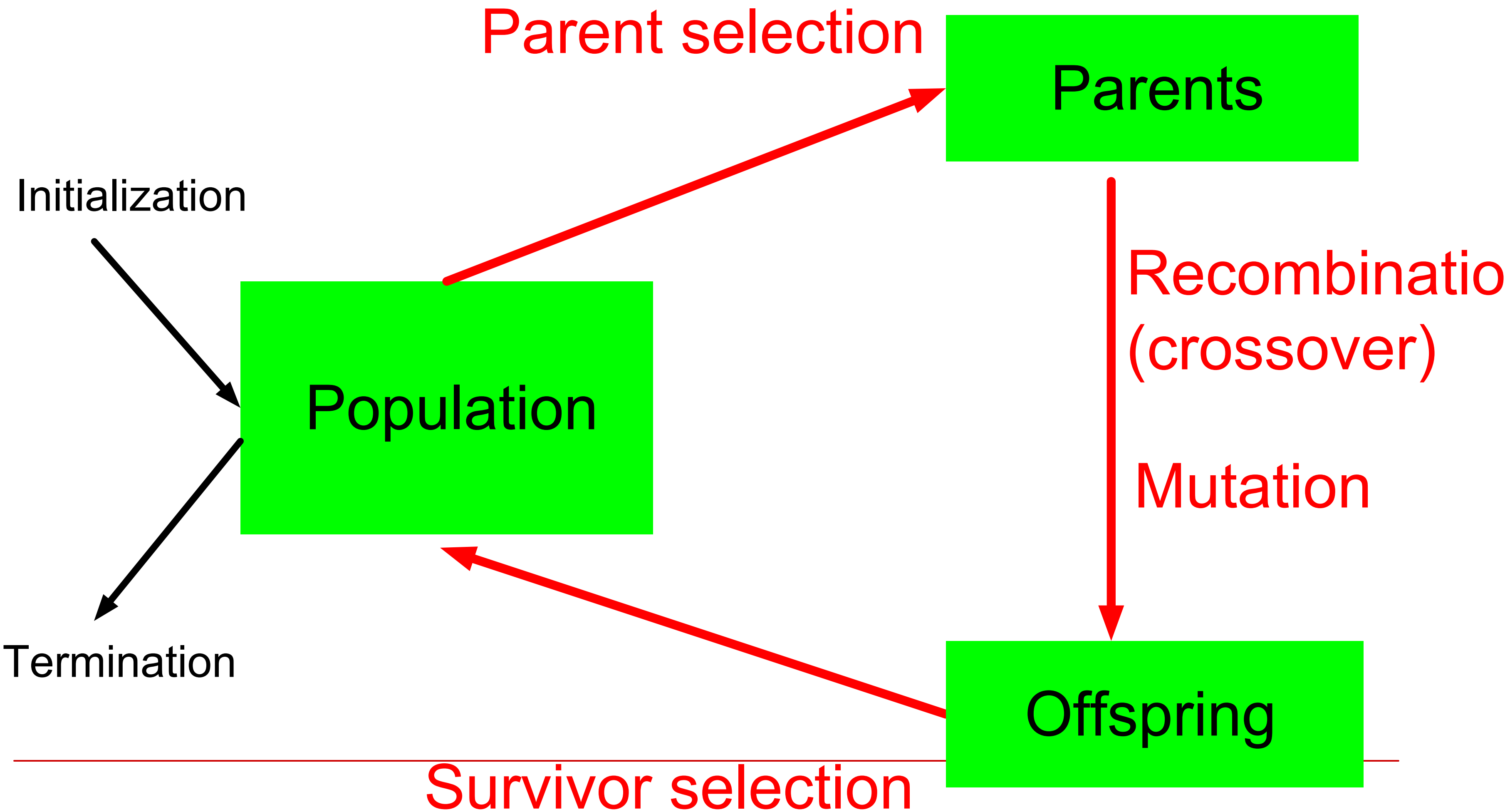
The Ingredients



The ingredients

- The algorithm
- Individuals
- Population
- Fitness
- Genetic operators
  - Crossover
  - Mutation
- Selection

Evolutionary Scheme



Mathematically ...

$$\boxed{\boldsymbol{x}[t+1] = s(v(\boldsymbol{x}[t]))}$$

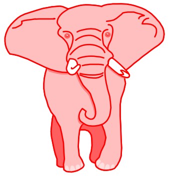
- $x[t]$: the population at time t under representation x
- v : is the variation operator(s)
- s : is the selection operator

Representation / individuals (1)

Individuals have two levels of existence

- phenotype: object in original problem context, the outside
- genotype: code to denote that object, the inside (chromosome, “digital DNA”):

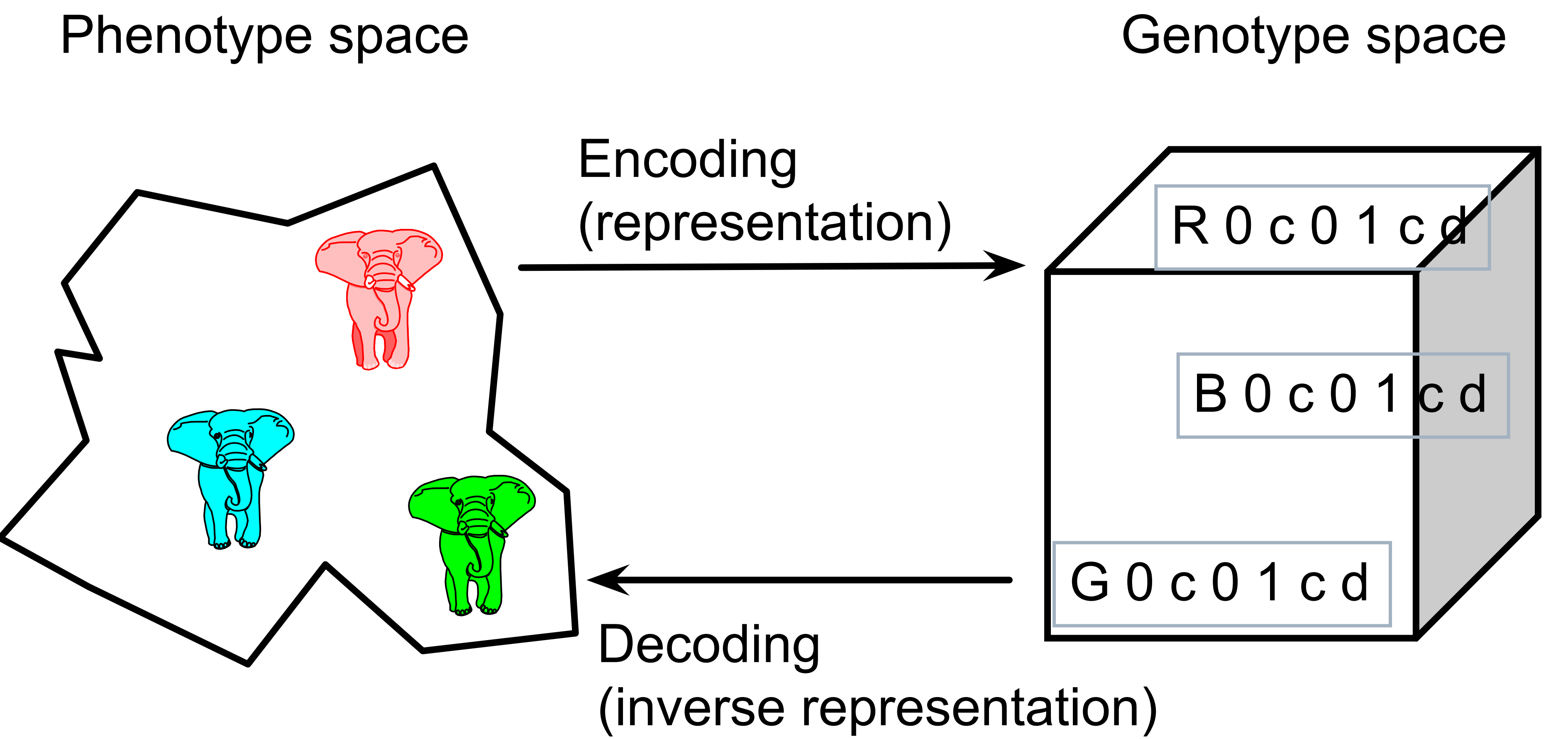
phenotype:



genotype:

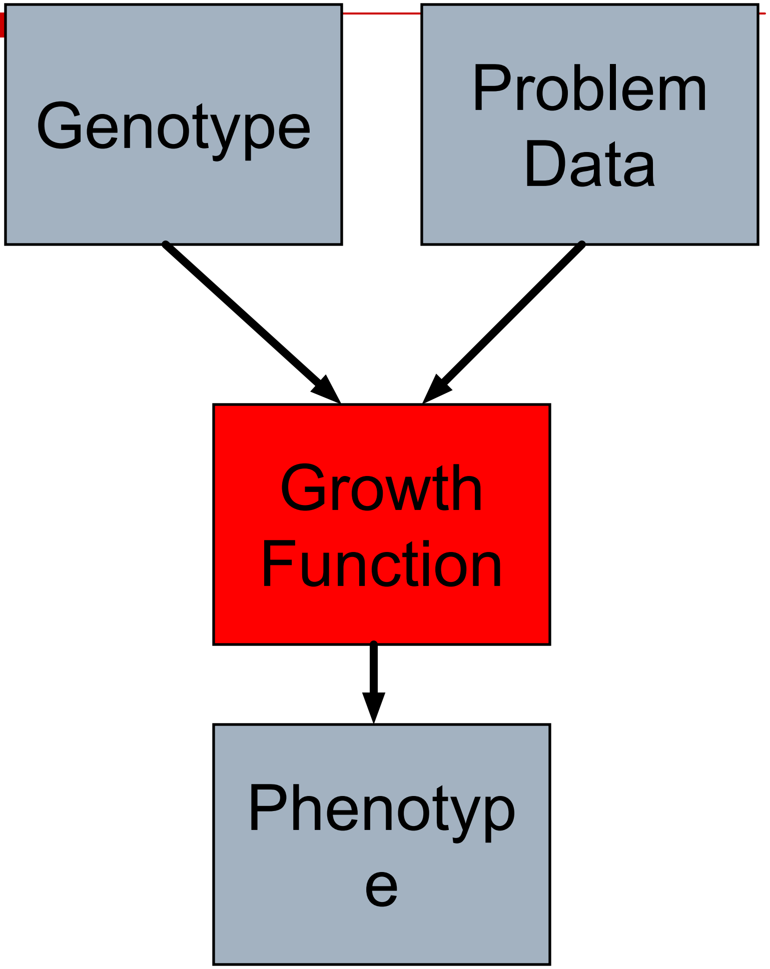
a	d	c	a	a	c	b

Representation / individuals (2)



Representation / individuals (3)

- Sometimes producing the phenotype from the genotype is a simple and obvious process.
- Other times the genotype might be a set of parameters to some algorithm, which works on the problem data to produce the phenotype.



Representation / individuals (4)

- Search takes place in the genotype space
- Evaluation takes place in the phenotype space
- Role of representation: defines objects that can be manipulated by (genetic) operators

Quality – Fitness function

- Each individual has a quality which depends on the environment where that individual act.

Population

- Role: holds the candidate solutions of the problem as individuals (genotypes)
- Formally, a population is a multiset of individuals, i.e. repetitions are possible
- Population is the basic unit of evolution, i.e., the population is evolving, not the individuals
- Selection operators act on population level
- Variation operators act on individual level

SELECTION

Role:

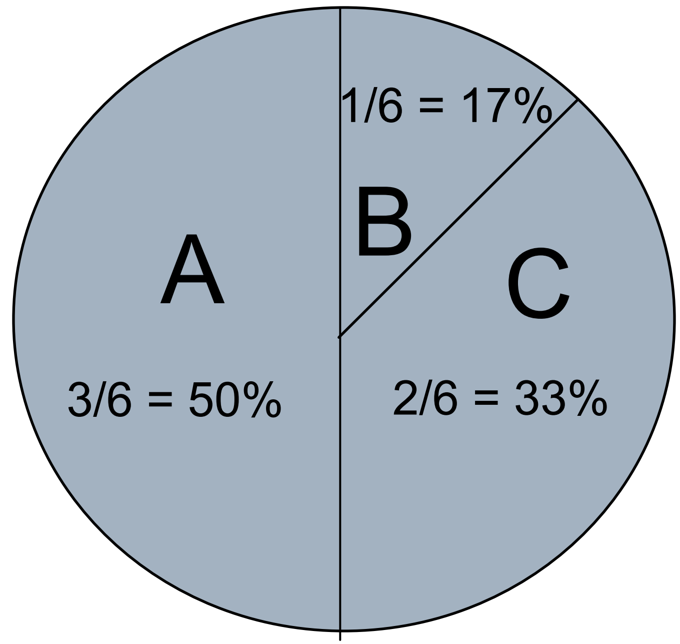
- Gives better individuals a higher chance of
  - becoming parents
  - surviving
- Pushes population towards higher fitness

Roulette selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

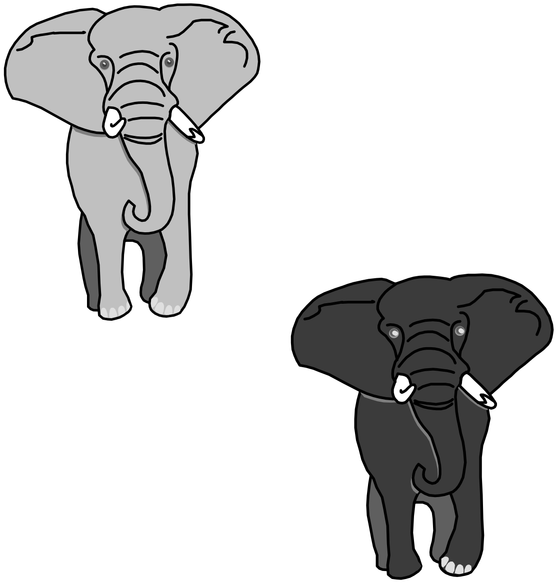


MUTATION

Role: causes small (random) variance

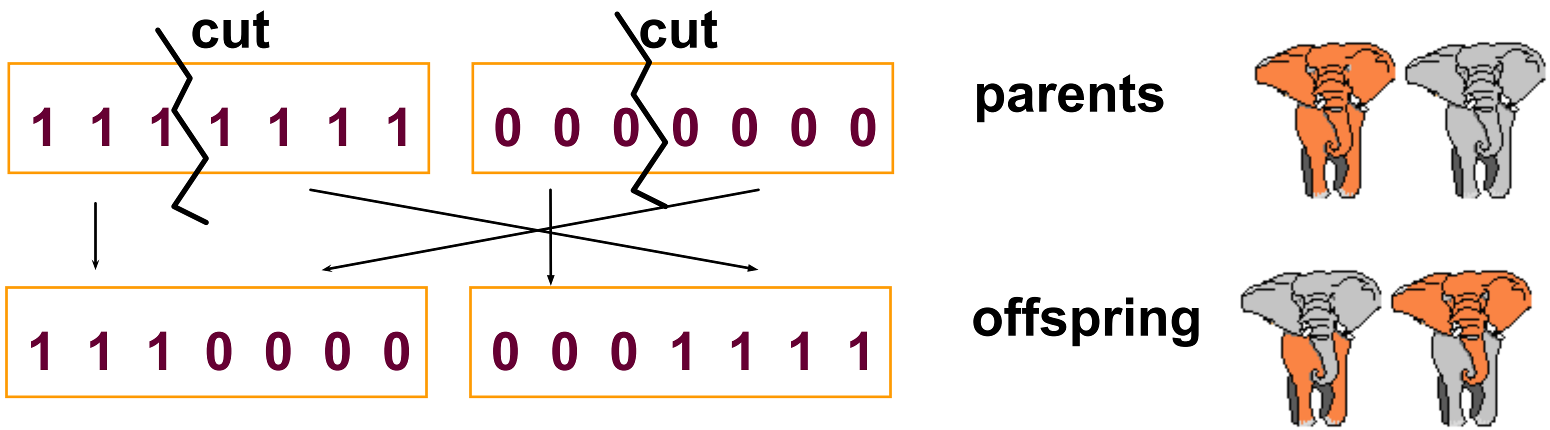
before | 1 1 1 1 1 1 1

after | 1 1 1 0 1 1 1

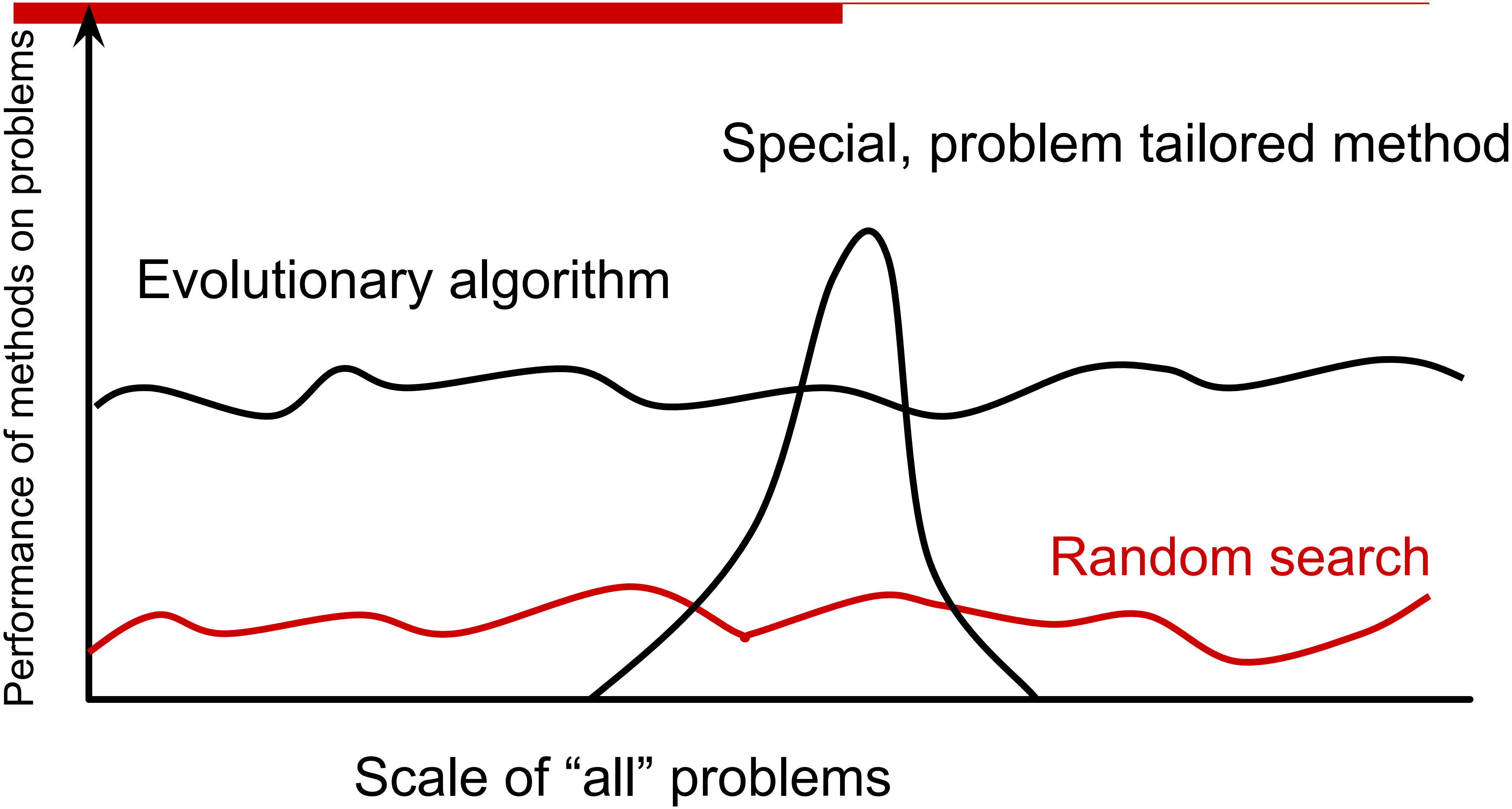


CROSSOVER - RECOMBINATION

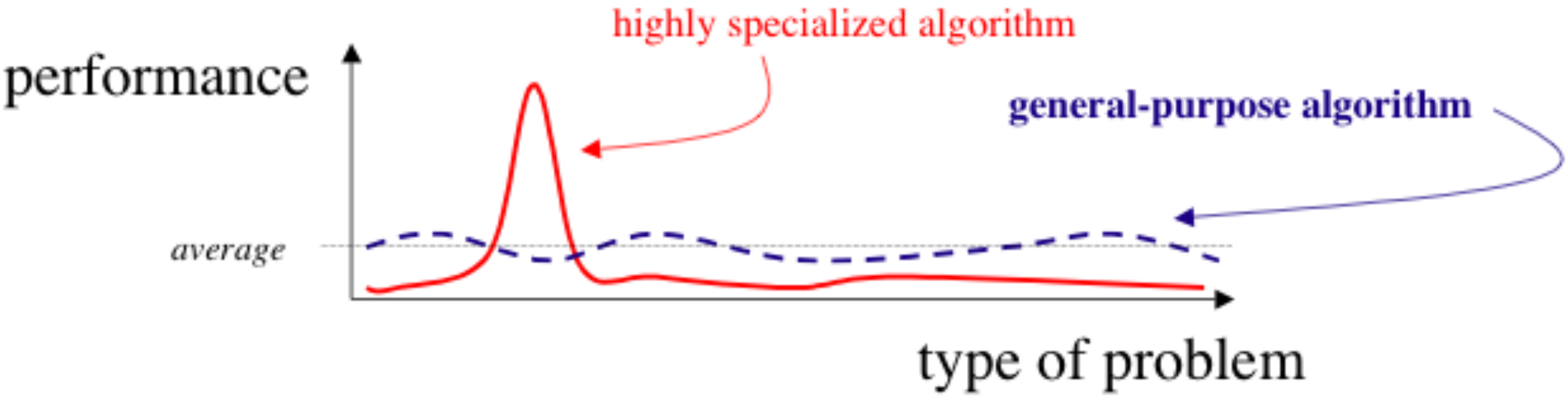
Role: combines features from different sources



Goldberg'89 view



No Free Lunch theorems



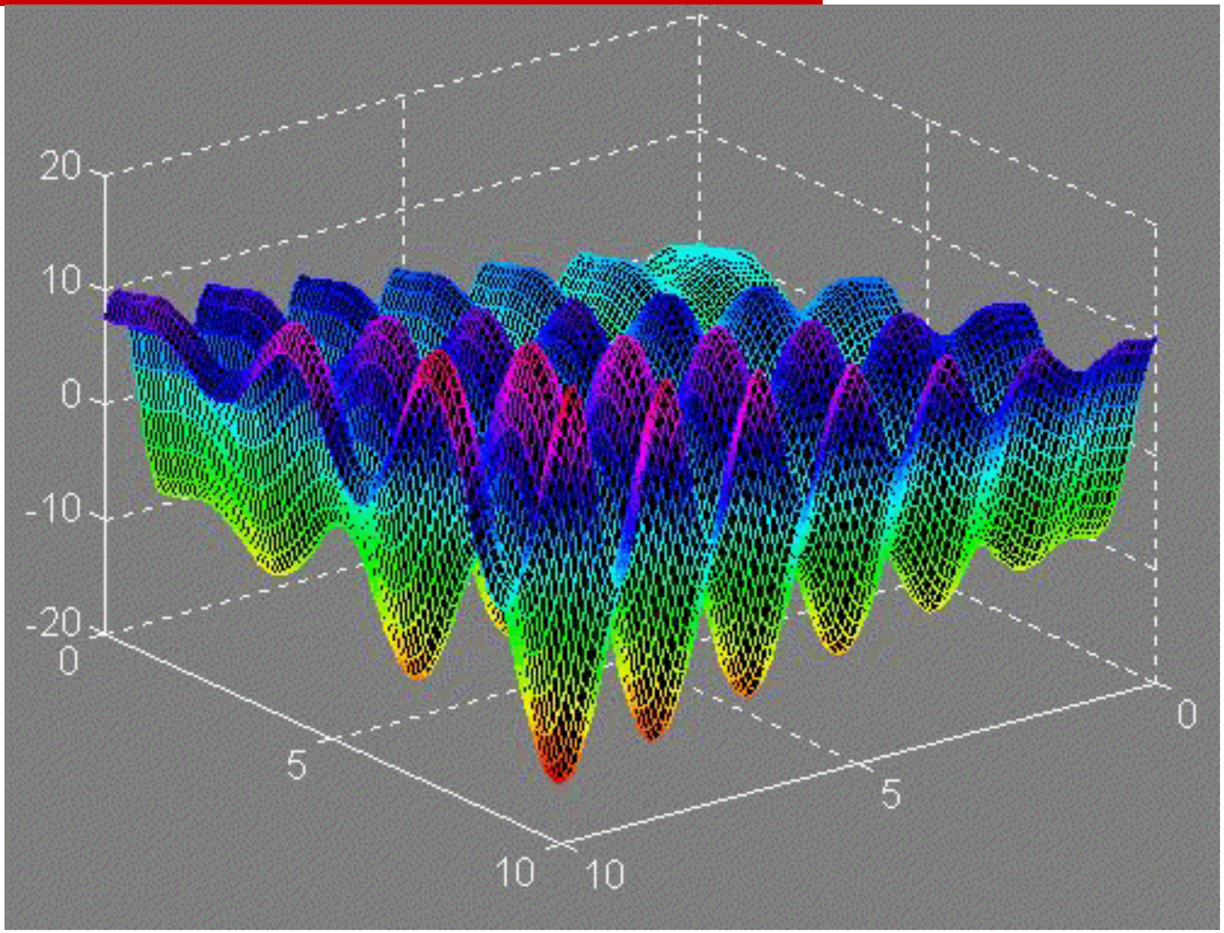
"[...] all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions." (Wolpert and Macready, 1995)

Random search performs better than all other algorithms for some problems.

Domains of Application

- Any optimization problem!!!
- Numerical, Combinatorial Optimization
- Planning and Control
- Engineering Design
- Data Mining
- System Modeling and Identification
- Machine Learning
- Artificial Life

FUNCTION OPTIMIZATION



STRENGTHS

- General algorithm for all problems
- No presumptions on problem space
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provides many alternative solutions
- Intrinsic parallelism, straightforward parallel implementations

Weakness

- No guarantee for optimal solution within finite time
- Weak theoretical basis
- May need parameter tuning
- Sometimes computationally expensive, i.e. slow

GENETIC ALGORITHMS & GENETIC PROGRAMMING

Genetic algorithms (USA, 70's, Holland, DeJong):

- Typically applied to: optimization
- Attributed features:
  - not too fast
  - good solver for combinatorial problems
- Special: many variants, e.g., reproduction models, operators

Genetic programming (USA, 90's, Koza)

- Typically applied to: evolving computer programs
- Attributed features:
  - competes with neural nets and alike
  - slow
  - needs huge populations (thousands)
- Special: non-linear chromosomes: trees, graphs

Evolution strategies & evolutionary programming

- Evolution strategies (Germany, 70's, Rechenberg, Schwefel)
  - Typically applied to:
    - numerical optimization
  - Attributed features:
    - fast & good optimizer for real-valued optimization
    - relatively much theory
  - Special:
    - self-adaptation of (mutation) parameters standard
- Evolutionary programming (USA, 60's, Fogel et al.)
  - Typically applied to: machine learning (old EP), optimization
  - Attributed features:
    - very open framework: any representation and mutation op's OK
  - Special:
    - no recombination
    - self-adaptation of parameters standard (contemporary EP)

BEYOND DIALECTS

- Field merging from the early 1990's
- No hard barriers between dialects, many hybrids, outliers
- Choice for dialect should be motivated by given problem
- Best practical approach: choose representation, operators, population model, etc. pragmatically (and end up with an "unclassifiable" EA)
- There are general issues for EC as a whole

SUMMARY

EVOLUTIONARY COMPUTATION:

- is based on biological metaphors
- has great practical potentials
- is getting popular in many fields
- yields powerful, diverse applications
- gives high performance against low costs