Bio-Inspired Artificial Intelligence

Lecture 6: Evolutionary Computing

Knowledge Technology

http://www.informatik.uni-hamburg.de/WTM/

Motivation

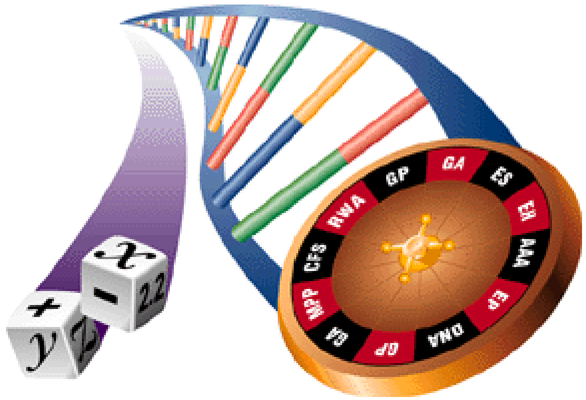
- Problem solving is a central theme in computer science
  - Many new and more ***complex problems***
  - Time for analysis and algorithms development decreases
  - More ***universal*** algorithms with automatic adaptation needed
  - “good” solutions within acceptable time are often satisfying
- Most powerful problem solvers in nature:
  - The (human) brain
    ... that created “the wheel, New York, wars and so on” [Adams 1978]
  - The evolutionary mechanism
    ... that created the human brain [Darwin 1895]

Computational Approach: Evolutionary Algorithms

Motivating Example

Goal of this lecture

- Learn about an approach to ***approximate*** problem solving based on the theory of evolution
- Get a deeper understanding of Evolutionary Algorithms

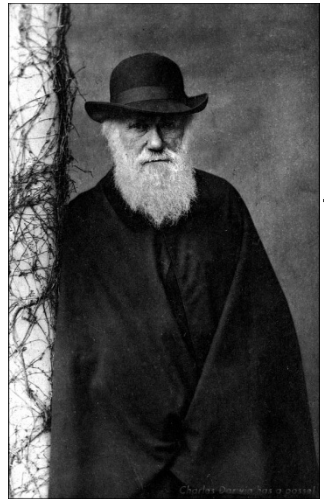


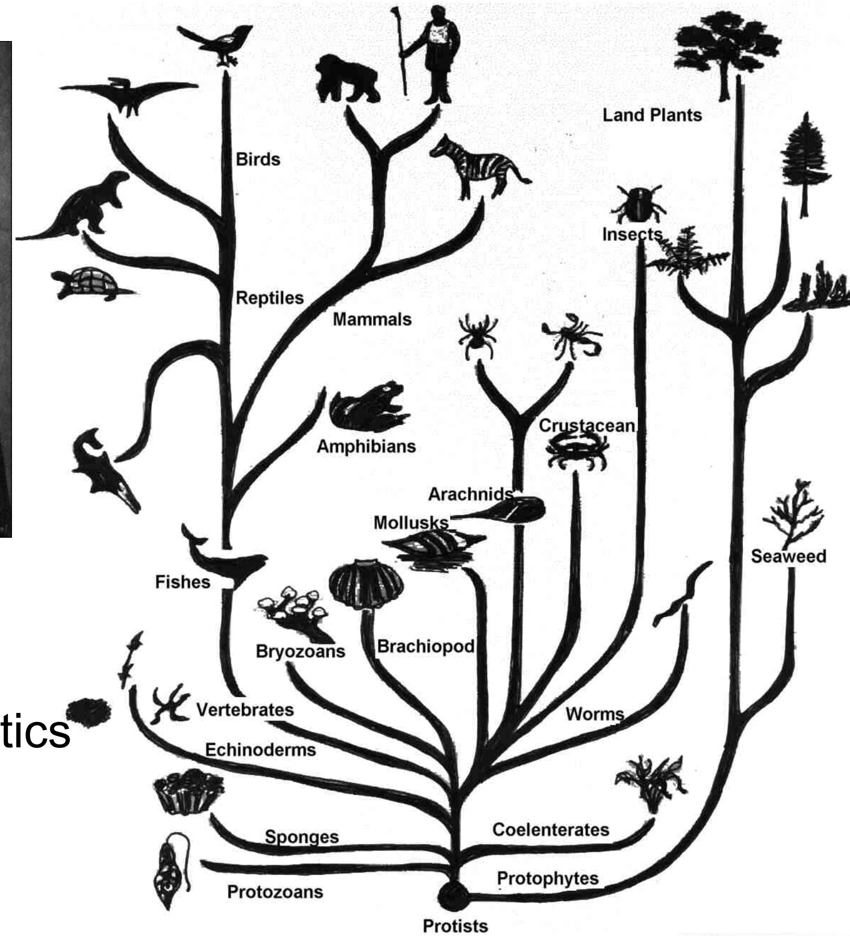
- Slides are mainly based on:
  - *Introduction to Evolutionary Computing* by Eiben and Smith, Springer 2003.
  - *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Floreano & Mattiussi, MIT 2008

The 4 pillars of Evolution

All species derive from common ancestor

Charles Darwin, 1859
On the Origins of Species





- Population
  - Group of several individuals
- Diversity
  - Individuals have different characteristics
- Heredity
  - Characteristics are transmitted over generations
- Selection
  - Individuals produce more offspring than the environment can support
  - Better at food gathering = better at surviving = make more offspring

Evolutionary Computing

- Trial-and-error problem solving method
- Population of individuals with reproduction and mutation
- “Survival of the fittest” (Darwin)
- “Diversity drives change” (Darwin)
- Inspired by natural evolution:

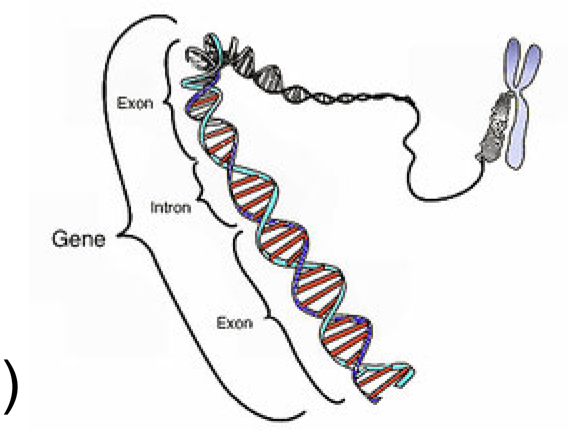
Evolution		**Problem Solving**
Environment	↔	Problem
Individual	↔	Candidate solution
Fitness	↔	Quality

History

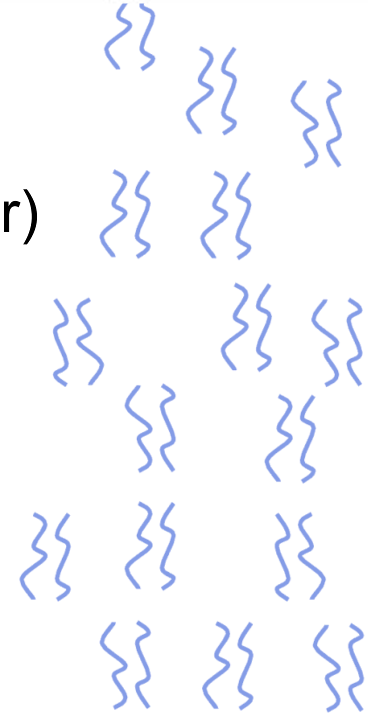
1948	Turing proposes "genetical or evolutionary search"
1962	Bremermann runs first computer experiments on "optimization through evolution and recombination"
1964	Rechenberg and Schwefel introduce *evolution strategies (ES)*
1965	Fogel, Owens and Walsh introduce *evolutionary programming (EP)*
1975	Holland introduces *genetic algorithms (GA)*
1992	Koza introduces *genetic programming (GP)*
1997	Launch of European EC Research Network EvoNet

- All techniques use same technology & ideas but differ in details
- Research area *Evolutionary Computing* deals with *Evolutionary Algorithms*

Biological Background

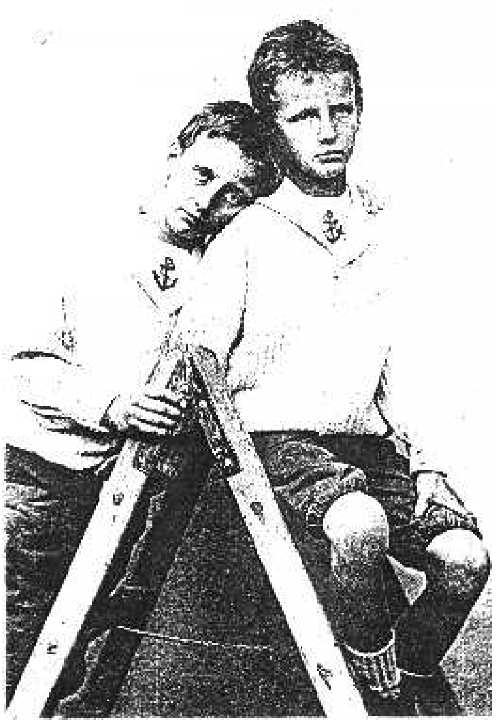


- Genotype determines Phenotype
- Genes: ***complex mapping***
  - One gene may affect many traits (features)
  - Many genes may affect one trait
  - Small changes in the genotype lead to large changes in the organism (e.g. Height, hair colour)
  - Encoded in strands of DNA
- Human DNA is organised into chromosomes
  - Characterised by Alleles (allele is one of two or more forms of a gene or a gene locus)
  - Together define the physical attributes of the individual



Phenotype & Genotype

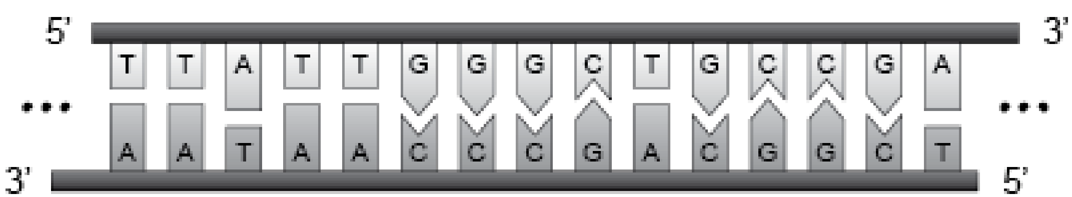
- ***Phenotype***: Manifestation of the organism (appearance, behavior, etc.).
  - Selection operates on the phenotype;
  - It is affected by environment, development, and learning
- ***Genotype***: The genetic material of that organism.
  - It is transmitted during reproduction;
  - It is affected by mutations;
  - Selection does not operate directly on it
- ***Genetics***: Structure and operation of genes
- ***Functional genomics***: Role of genes in the organism
- To what extent are we determined by genotype and phenotype?



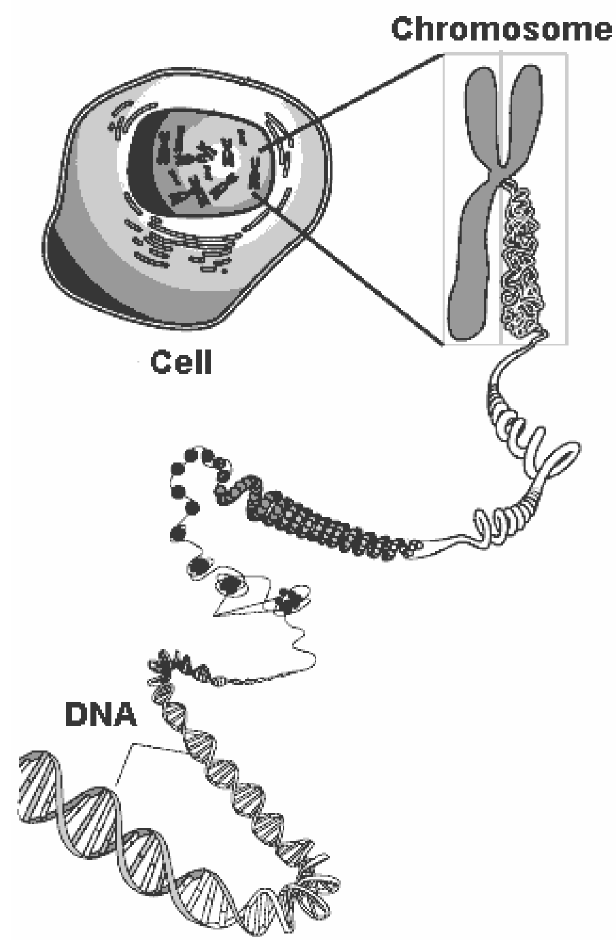
Jean-Felix & Auguste Piccard

DNA *(DesoxyriboNucleic Acid)*

- Long molecule, twisted in a spiral and compressed
- Humans have 23 pairs of DNA molecules (*chromosomes*)
- DNA is composed of 2 complementary sequences (*strands*) of 4 nucleobases (A, T, C, G), which bind together in pairs (A-T and C-G)

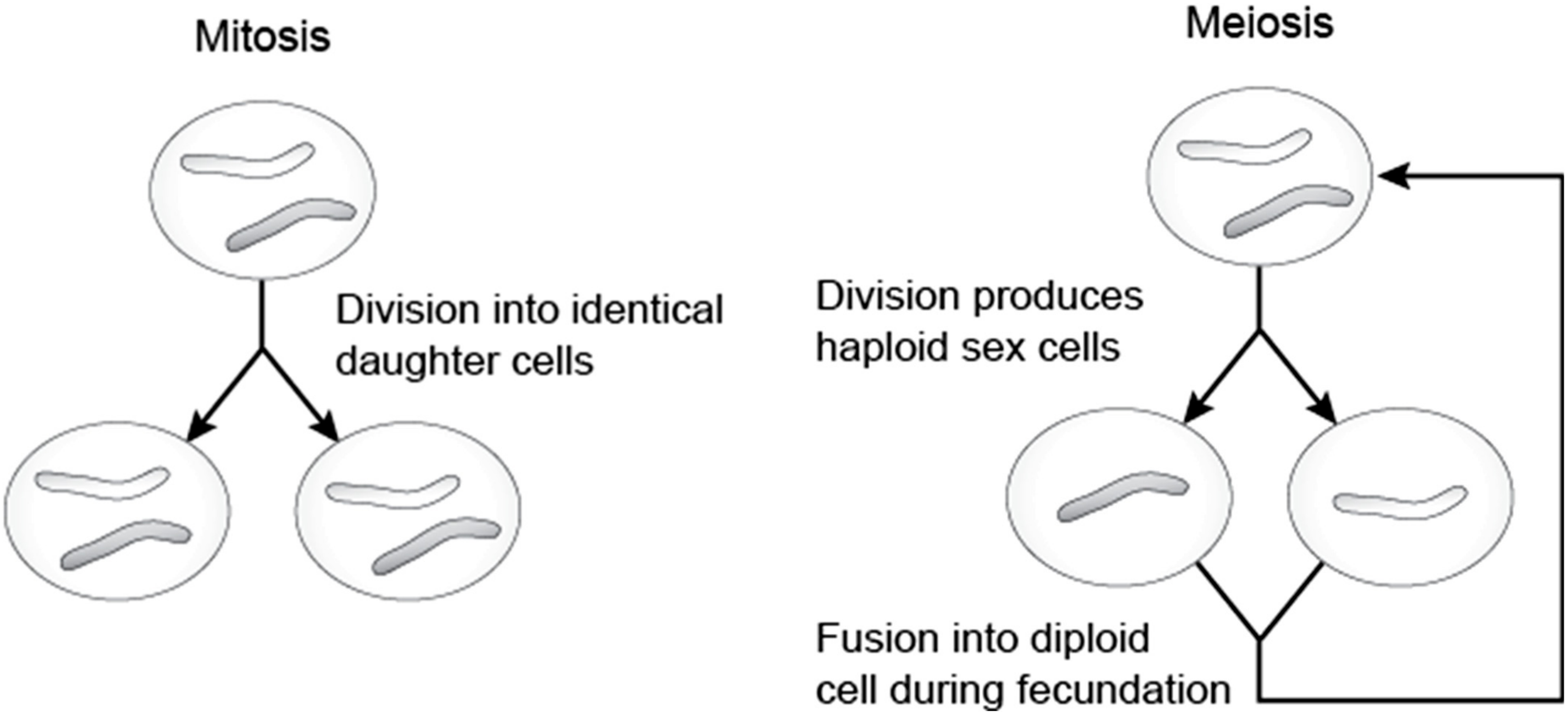


- A **gene** is a sequence of several nucleotides that produce a protein



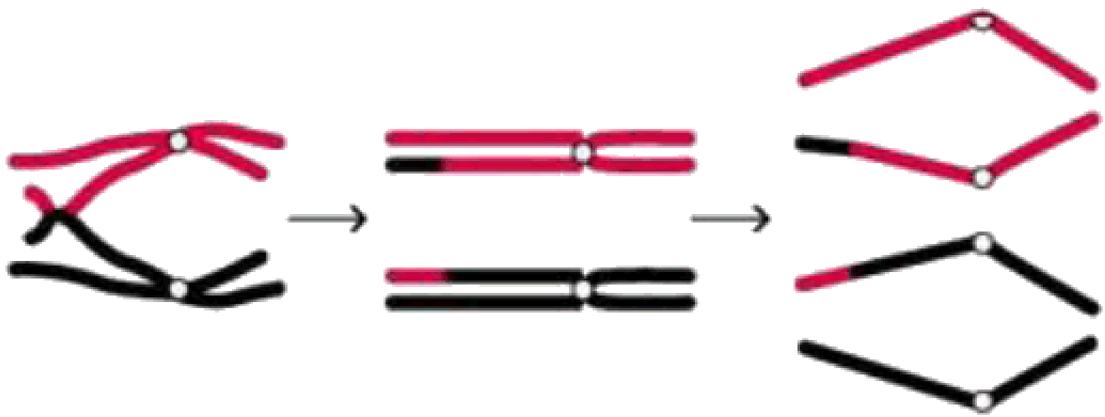
Cell Replication

- Cells replicate in two ways:
  - ***Mitosis***: during growth/maintenance of the organism
  - ***Meiosis***: during production of gametes (egg / sperm cells)



Cell Replication (cont.)

- Gametes: Sperm or egg cell
  - Contain only one single chromosome complement of chromosomes
  - Formed by a special form of cell splitting: ***Meiosis***
  - During meiosis, pairs of chromosomes undergo ***crossing-over***



- Occasionally some of the genetic material changes very slightly (replication error): ***Mutations***

Example: Evolution of Camouflage

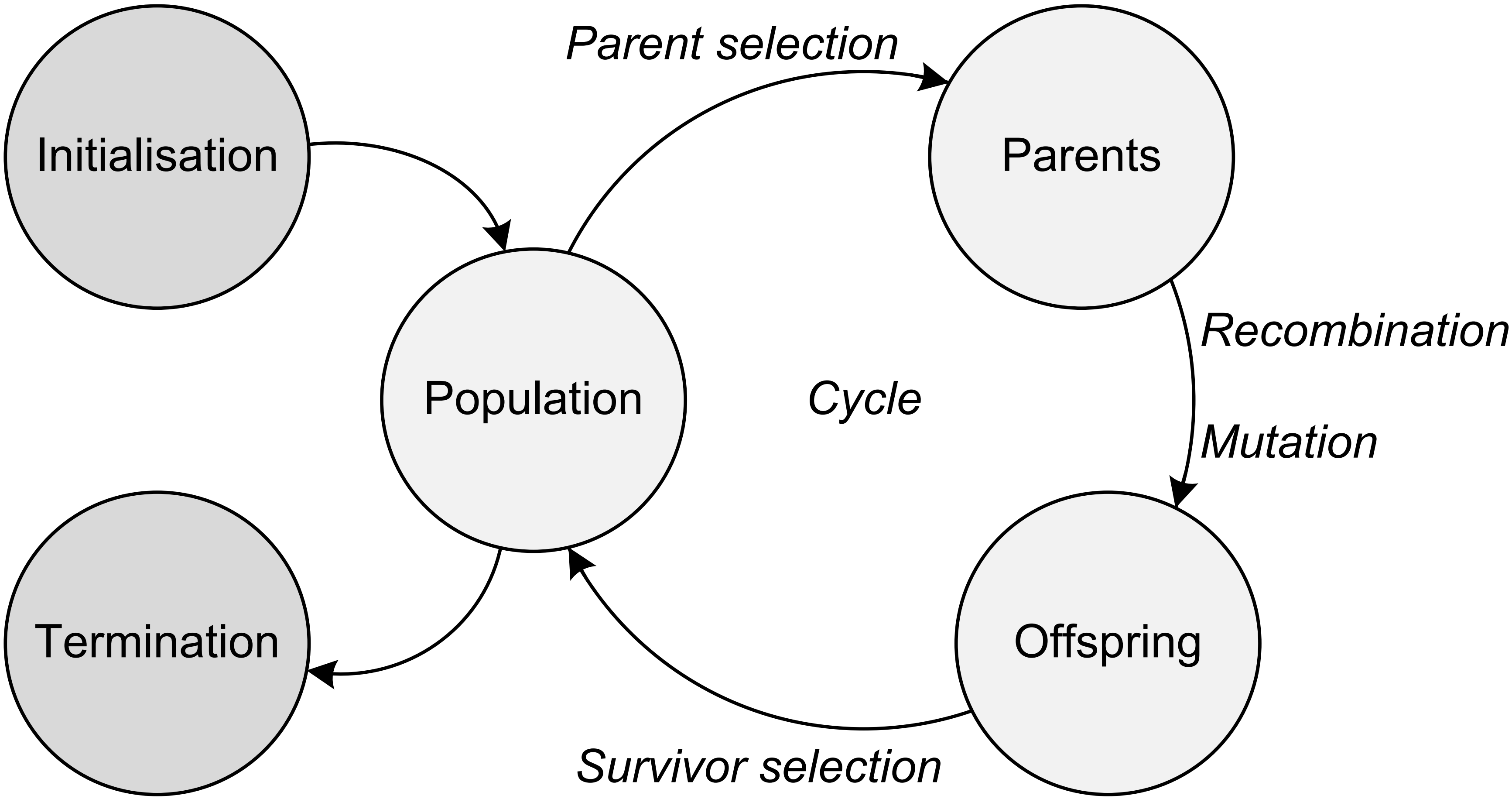
Chris Schneider, Boston University

General Characteristics of Evolutionary Algorithms (EA)

Evolutionary Algorithms are:

- generate-and-test algorithms
- population-based
- stochastic
- inspired by natural evolution
- search procedures:
  - guided by fitness
  - driven by variation and selection operators
- anytime algorithms: stop at any time with a (suboptimal) solution

General Scheme of EAs



General Categories of EAs

- Variation operators (recombination & mutation) depend on candidate representation
- Selection operators only require fitness value
  - *“universal” operators*
- EAs follow same basic scheme
  - differ mainly in technical details, e.g. representation of candidate solutions:

Algorithm Class	**Candidate Representation**
Genetic Algorithm	string over finite alphabet, e.g. 01110
Evolution Strategies	real-valued vectors, e.g. (0.1,0.4,12.3)
Evolutionary Programming	application specific, e.g. FSM
Genetic Programming	trees

Representation

- Cover all possible solutions
- Encoding should only allow valid solutions (not always possible)
- Choose ***appropriate representation***:
  - Bit-string can decode integers or real numbers
  - Problems: e.g. mutation changes values significantly (work-around: *Gray coding:* two successive values differ by 1 bit
  - Better direct representation of numbers
  - Bit-string often fine
- Two meanings of representation:
  - Mapping between phenotype & genotype space: de-/encoding
  - Data structure used in genotype space

Fitness Function

- Represents requirements to adapt to
- Basis for selection
- Assigns quality measure to genotypes
- Synonyms: evaluation or objective function

Example:

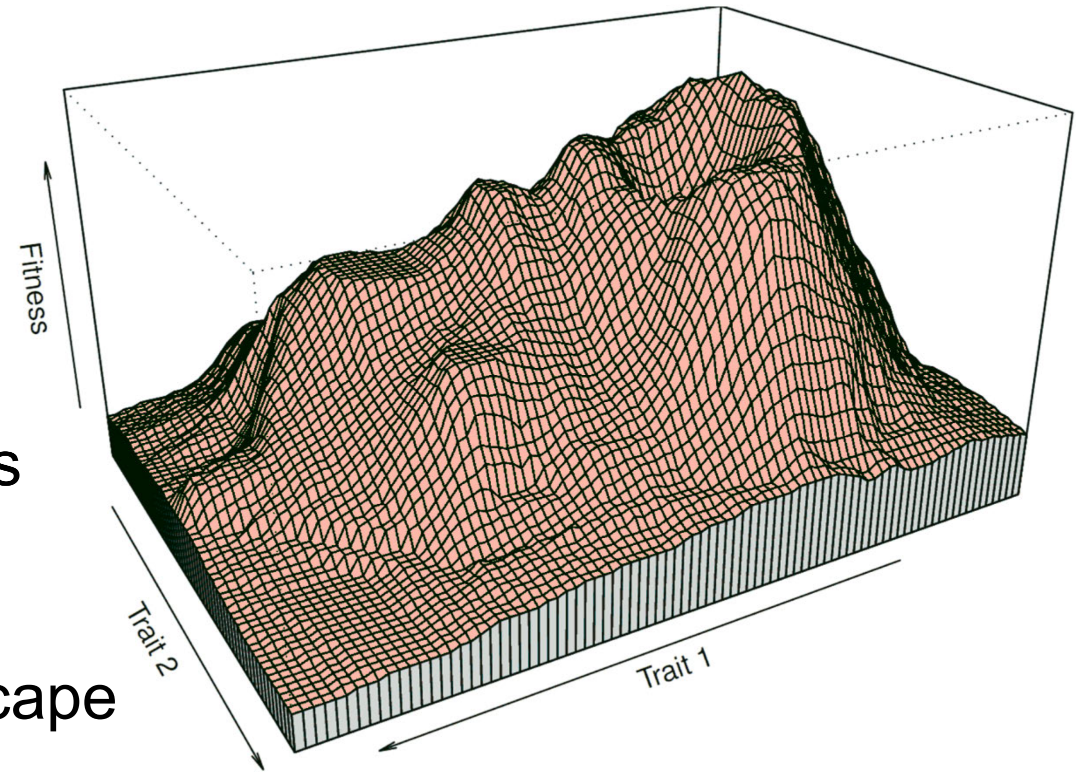
- ***Context***: Minimize x^2
- ***Phenotype***: $x \in \mathbb{N}$
- ***Genotype***: z: binary representation of x
- ***Fitness Function***: fitness $f(z)$ of genotype z is defined as 1 divided by square of its corresponding phenotype, e.g.
  $z = 0010 \rightarrow$ phenotype: $x = 2 \rightarrow f(z) = 1/x^2 = 0.25$

Fitness Landscape Metaphor

- Usually of a very
  high dimension

- Local optimum:
  solution better than
  neighbouring solutions

- Global optimum:
  best solution in landscape



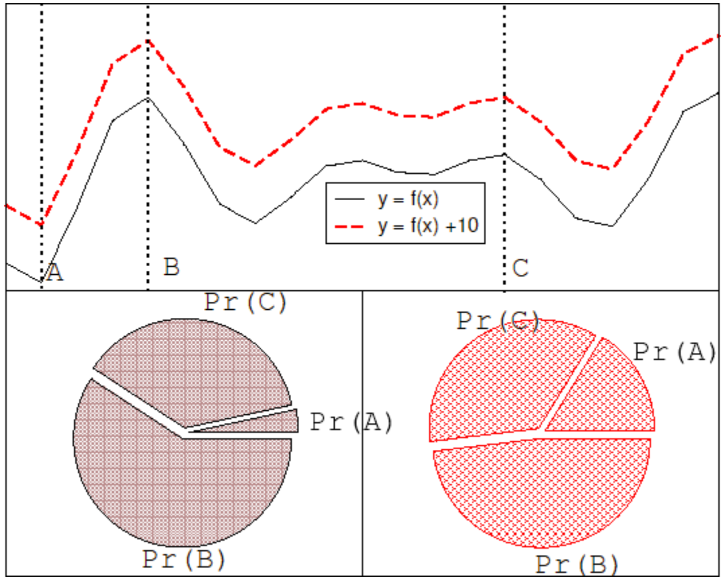
- Uni-modal problem: only one local optimum

- Multi-modal problem: several local optima

Parent Selection

- Select individuals to create offspring
- Population level
- Often ***probabilistic***:
  - Individuals with high fitness more likely to become parents
  - “Weak” individuals might also become parents (with low probability) to avoid local optima
  - Sum over all probabilities is 1.0
- Parent selection in ES: often chosen randomly from parent population
- Parent selection supports process of evolving better solutions over time

Parent Selection – Fitness Proportional Selection (FPS)



- Absolute fitness of individual vs. absolute fitness of population
- Probability to select *i* from population of size $\mu : \Pr(i) = f_i / \sum_i^{\mu} f_i$
- Problems:
  - ***Premature convergence***
  - Nearly equal fitness values result in no selection pressure
  - Selection probabilities change for transposed fitness values
  - Fitness scaling, e.g. f$(i) - \beta^t$ with $\beta = min_{y \in P^t}$ f(y) (subtract minimum fitness in current population)

Parent Selection – Ranking Selection

- Inspired by problems of Fitness Proportional Selection
- Rank individuals by fitness
- Assign selection probabilities based on rank
- Different implementations of *Pr(i)*
- Problem: Can lead to slower convergence

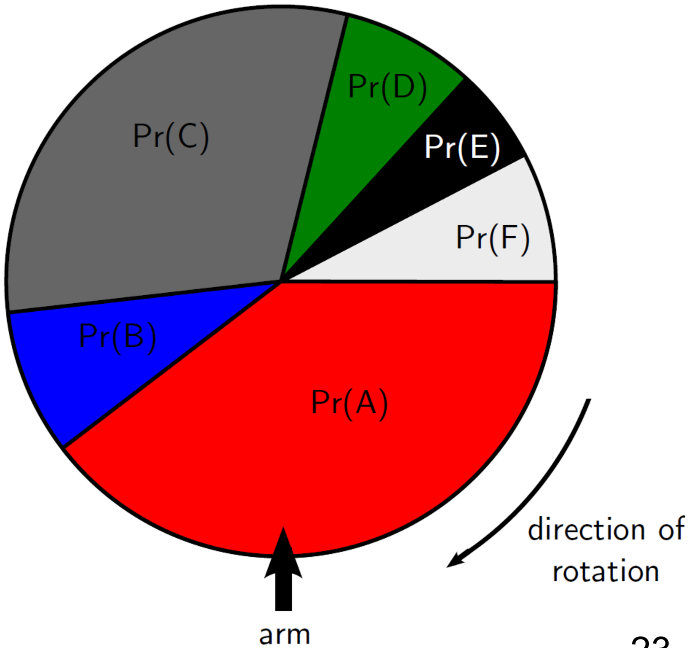
Example: Linear Rank (LR)

$$P_{LR}(i) = \frac{(2 - s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

	Fitness	Rank	P_{selFP}	P_{selLR} (s=2)	P_{selLR} (s=1.5)
A	1	1	0.1	0	0.167
B	5	3	0.5	0.67	0.5
C	4	2	0.4	0.33	0.33
Sum	10		1.0	1.0	1.0

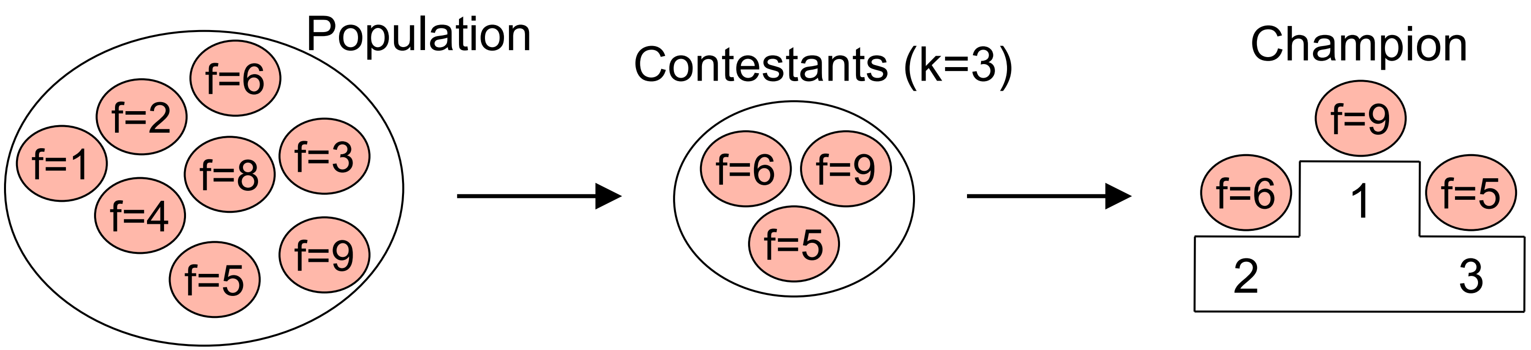
Parent Selection – Roulette Wheel

- FPS and Ranking Selection define probability distributions for selecting individuals
- How to sample these distributions?
  - *Roulette Wheel (RW)!*
- Sampling quality of RW in practice:
  - ***High variance*** from theoretical distribution
  - ***Expensive*** to calculate in
    - distributed systems
    - if number of parents μ is large



Parent Selection – Tournament Selection

- Tournament Selection only requires order between individuals (relative fitness) and no global knowledge
  - Previous methods use knowledge over entire population
- Idea:
  - randomly select *k* individuals into a group *G* of contestants
  - individual *i* with $f(i) = max_{i \in G} f(i)$ wins the tournament
  - add *i* to parent set
  - repeat until μ parents are selected



k is the tournament size (larger size = larger selection pressure)

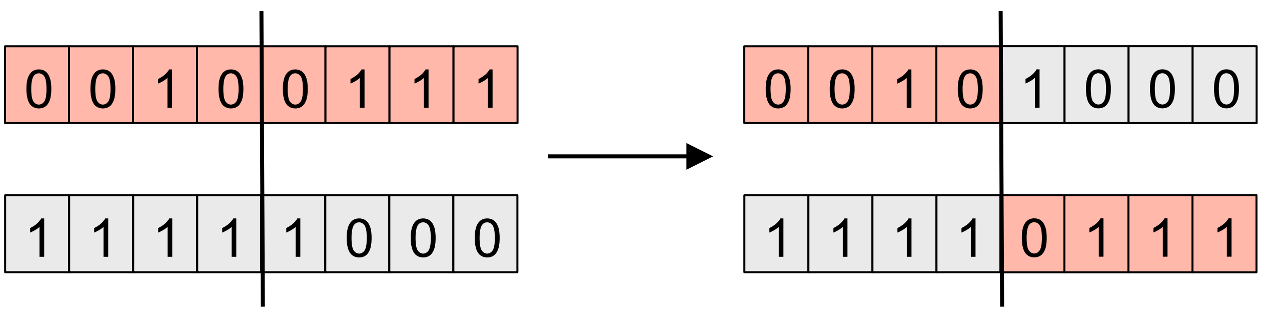
Variation: Recombination

- Variation operator: create new individual(s) from parents
  - Synonym: ***crossover*** operator
  - Distinguishes EA from other optimization techniques
  - Merge information from parents into offspring
  - Aims for diversity,
    but sometimes a destructive jump in fitness landscape
- Crossover applied with probability p_c, e.g. $p_c \in [0.5, 1.0]$;
  otherwise parents are copied
- Implementation depends on representation form

Variation: Recombination – *n*-Point Crossover

- Split parents at *n* points and recombine segments
- ***Positional*** bias:
  - *n*-Point Crossover tends to keep together genes located close to each other
  - One-Point can never keep together genes from opposite ends
  - Knowledge on problem structure often not available

Example One-Point Crossover



Variation: Recombination – Uniform Crossover

- Swap genes based on vector of random values
  - Process generates first child
  - Second child: inverse process
- ***Distributional*** bias:
  - Genes are distributed among children instead of transferring larger sets of co-adapted genes to one child

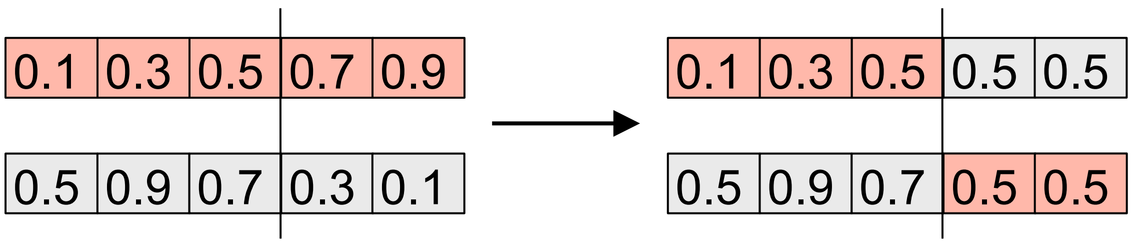
Example: p=0.5, x = (0.3, 0.7, 0.4, 0.2, 0.6, 0.9, 0.1, 0.8)

0	0	1	0	0	1	1	1		0	1	1	0	1	0	1	0
								→								
1	1	1	1	1	0	0	0		1	0	1	1	0	1	0	1

Variation: Recombination – Further operators

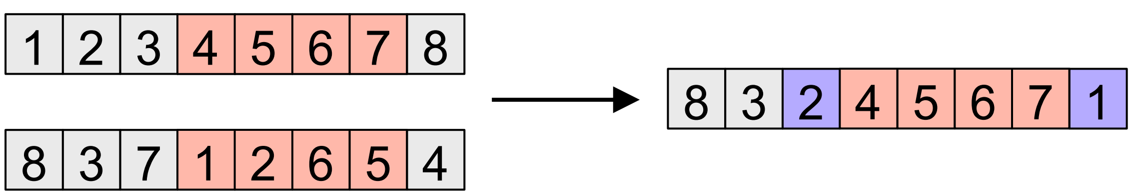
- Arithmetic Recombination:
  - Powerful for floating-point representations

    Example: Simple arithmetic recombination



- Permutation:
  - Powerful if exchange of substrings is a constraint

    Example: Partially Mapped Crossover (PMX)



Variation: Mutation

- Variation operator: create new individual(s) from old ones
- Slightly mutates one individual
- Always ***stochastic***: random and unbiased changes
- Implementation depends on representation form
- Many different implementations
- Like recombination, mutation plays different roles in different EAs

Variation: Mutation – Change of Allele Values

- Bitwise mutation
  - For every position: flip bit with probability p_m

1	0	1	0	0	1	0	1

$\longrightarrow$

1	0	0	0	0	1	1	1

- Random resetting / Uniform mutation
  - For every position: change value to random value from the corresponding domain with probability p_m

2	3	4	5	4	6	7	1

$\longrightarrow$

2	3	4	5	2	6	7	1

$x \in \mathbb{N}$

a	b	a	c	b	a	c	a

$\longrightarrow$

a	c	a	c	b	a	c	b

$x \in \{a,b,c\}$

0.3	0.1	0.8	0.2	0.6

$\longrightarrow$

0.3	0.7	0.5	0.2	0.6

$x \in]0,1[$

Variation: Mutation – Permutation

- Swap mutation
  - Randomly choose two positions and swap their allele values

2	3	4	5	4	6	7	1	→	2	4	4	5	3	6	7	1

- Insert mutation
  - Shuffle value of a second position towards the first position

2	3	4	5	4	6	7	1	→	2	3	4	5	7	4	6	1

- Scramble mutation
  - For a chosen subset scramble their positions

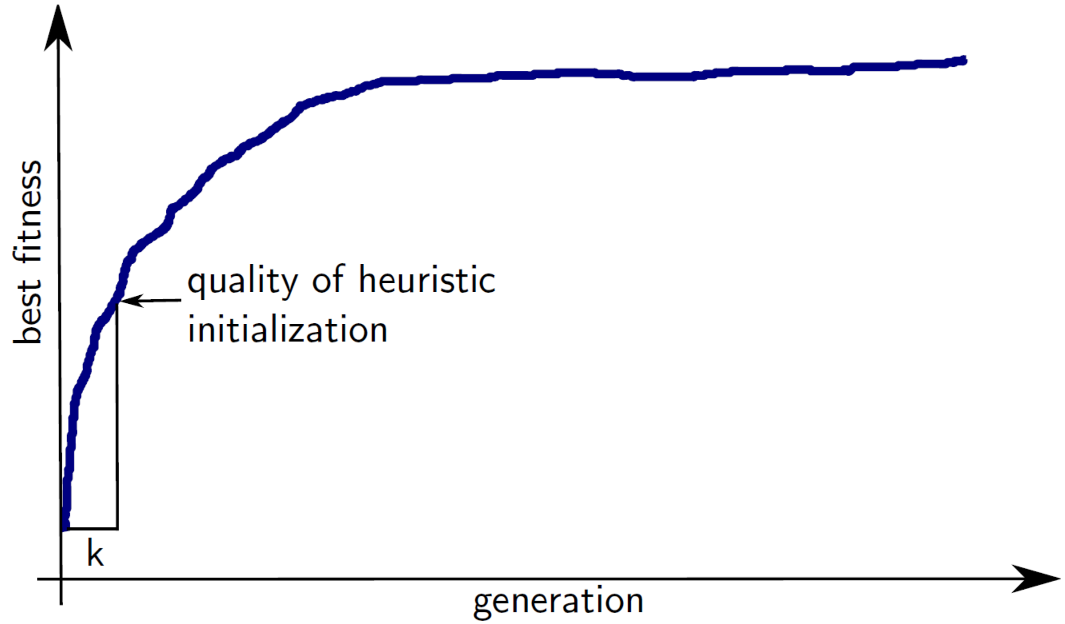
2	3	4	5	4	6	7	1	→	2	3	4	6	5	4	7	1

Survivor Selection

- Similarly to parent selection, survivor selection selects individuals based on quality
- Role: reduce μ parents and λ offspring to μ individuals that constitute the next generation
  - Fitter individuals more likely to survive
  - Weak individuals may also survive
- Selection often ***deterministic*** based on age and/or fitness
  - Age-based: Choose μ best from *offspring* only
  - Fitness-based: Choose μ best from *parents* and *offspring*
  - Elitism: Keep the $\kappa < \mu$ best, replace the rest by offspring
- Synonyms: environmental selection, replacement

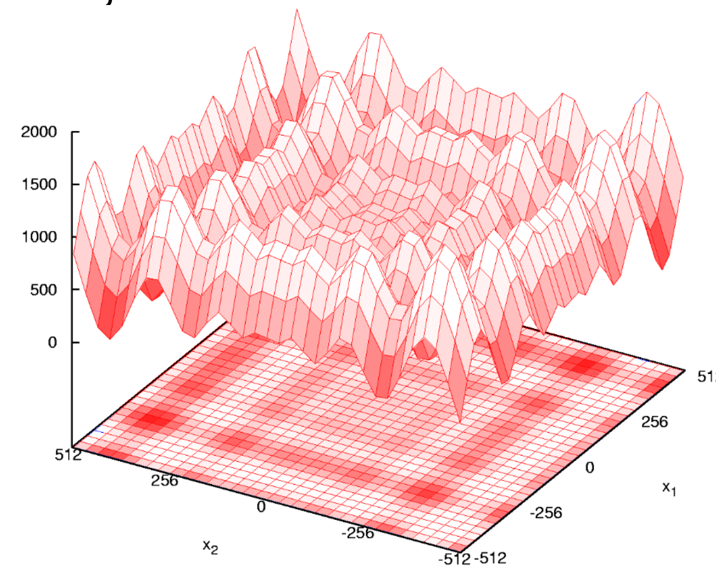
Initialization

- Random initialization
- Heuristics:
  - might be used for higher quality initialization
  - additional time for implementation & computational effort
- EA should reach level of heuristic-based initialization after some *k* generations



Termination Condition

- Optimum value known: stop if reached, or if only a small $\varepsilon > 0$ away
- Problem: EAs are stochastic and may never fulfill that condition
- Other possible criteria include:

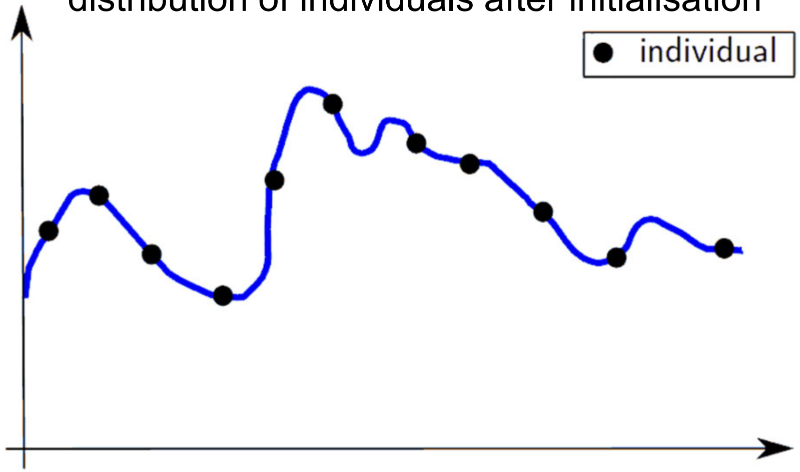


  - CPU time elapsed
  - Maximum number of iterations or fitness calculations reached
  - No fitness improvement within last *t* generations
  - Diversity of population too small

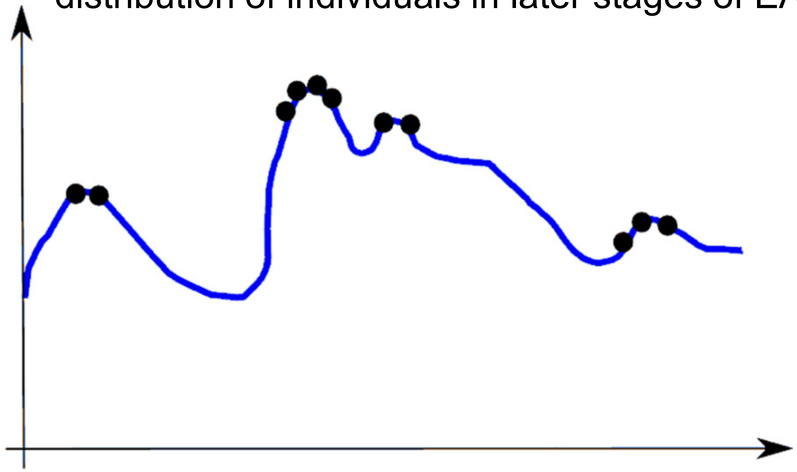
Exploration vs. Exploitation

- ***Exploration***: generate individuals in unexplored regions of the solution space
- ***Exploitation***: generate individuals in regions with high fitness
- Trade-off:
  - too much exploration: inefficient search
  - too much exploitation: risk of getting stuck in local optima

distribution of individuals after initialisation



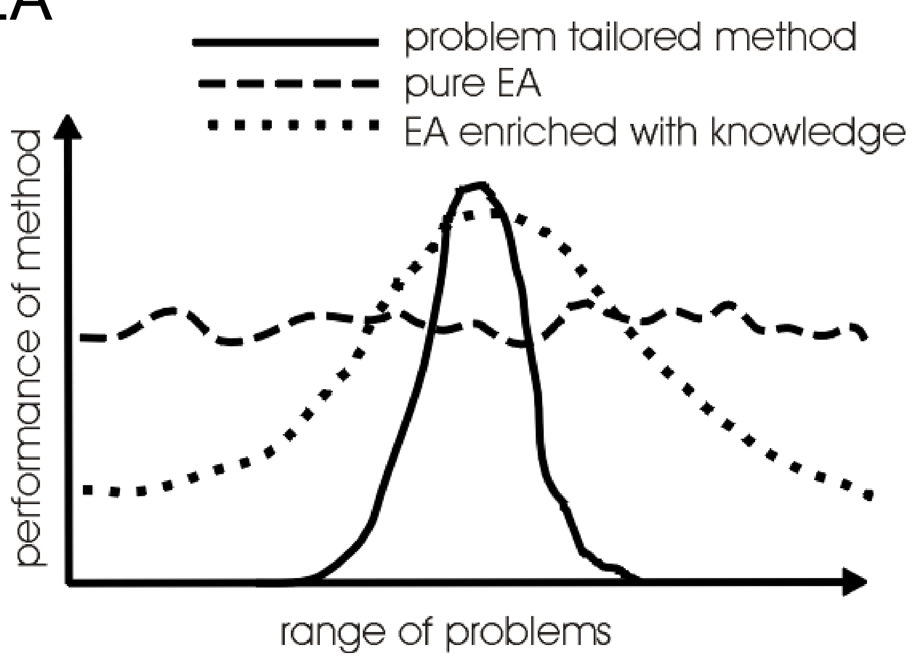
distribution of individuals in later stages of EA



Hybrid Evolutionary Algorithms

- Issue: For some problems an EA converges very slowly to a good solution
- But: Some knowledge about solutions is available
- Idea: Looking to improve EA search for good solutions
  - Hybridise EAs with ***Local Search*** operators that work within the EA loop

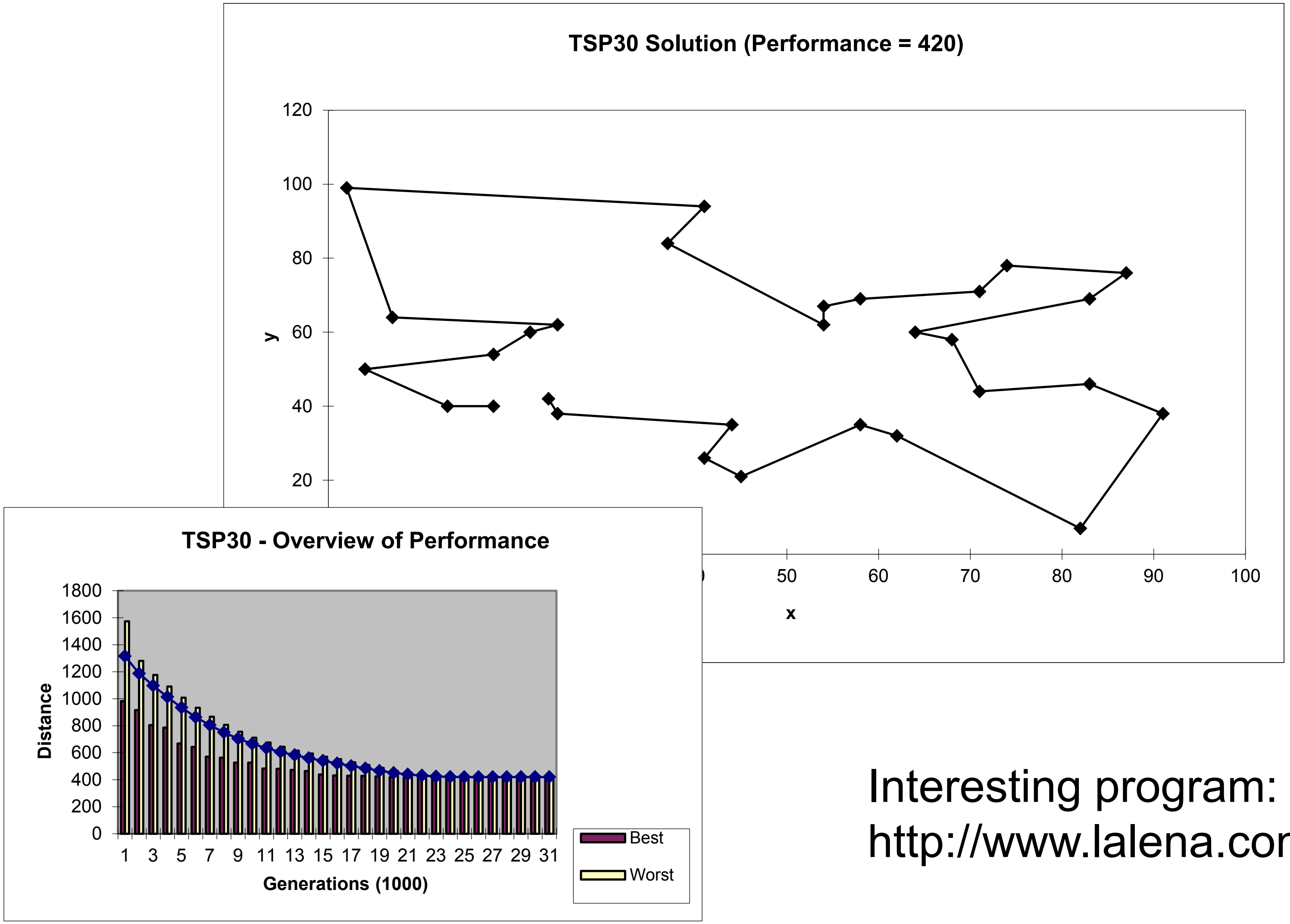
→ Memetic Algorithm



Applications

- Civil, mechanical, and industrial engineering
- Optimal cutting
- Power systems
- Control systems
- Signal processing
- Rule for artificial player in computer games
- Machine learning
- … and many more

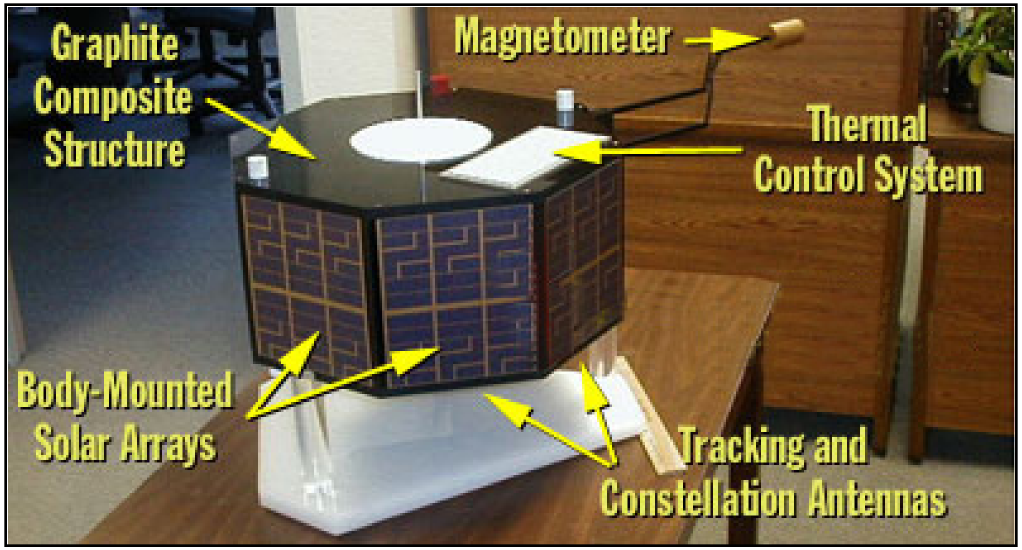
Example: Travelling Salesman Problem

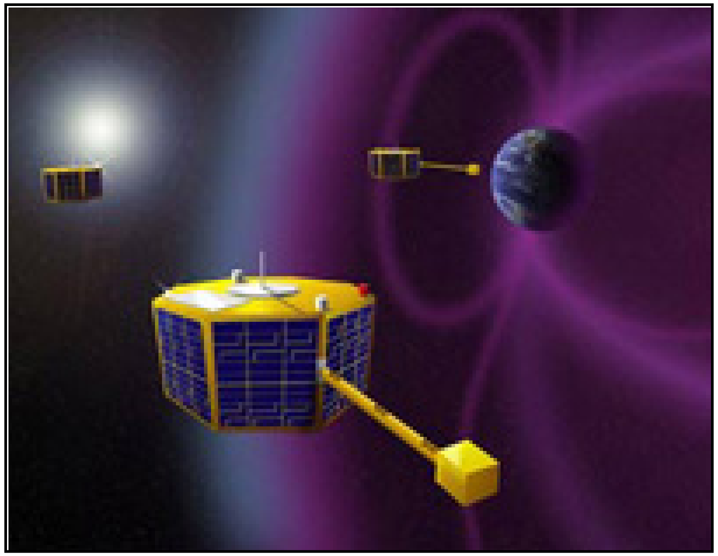


Interesting program:
http://www.lalena.com/AI/Tsp/

Example: Antenna for Nanosatellites

- ST5 mission: Measure effect of solar activity on the Earth's magnetosphere
  - 3 nanosatellites (50 cm)
  - Design of antenna to send data to ground station



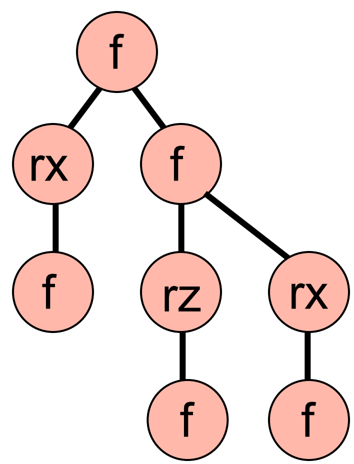


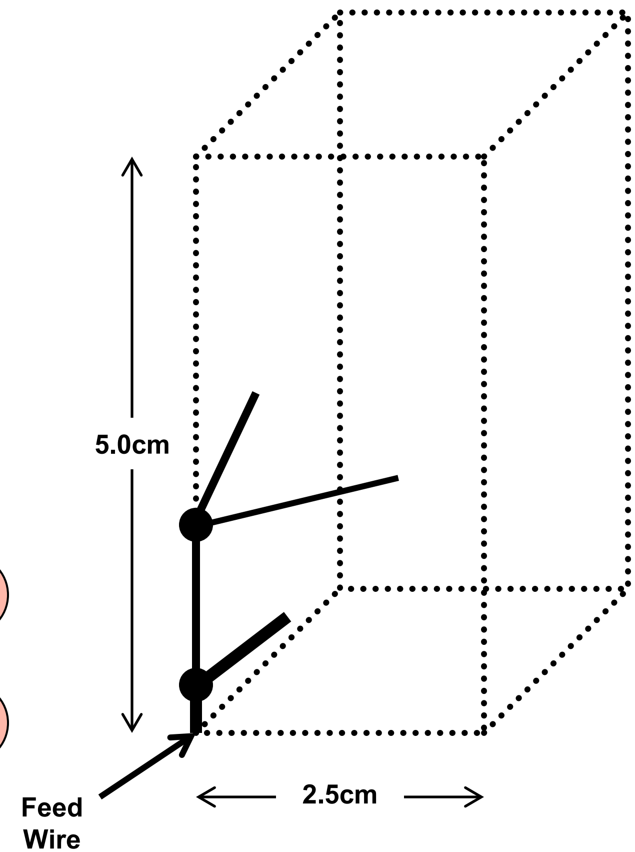
[Lohn, Hornby, Linden, 2004]

Genetic Representation

- Tree-based Encoding
- Execute instructions from root to leaves
- Evaluate fitness according to specs in simulation
- Build best and test in anechoic chamber

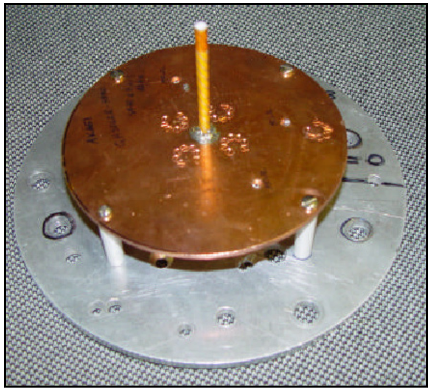
Function Set
f = forward (length, radius)
r x/y/z = rotate x/y/z
Terminal Set
Length, radius, x, y, z
Constraint:
max. 3 branches for each f node



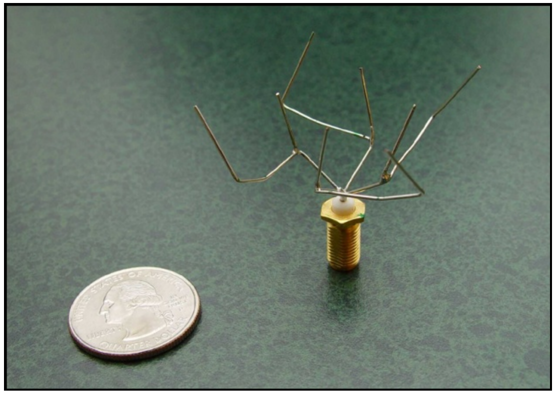


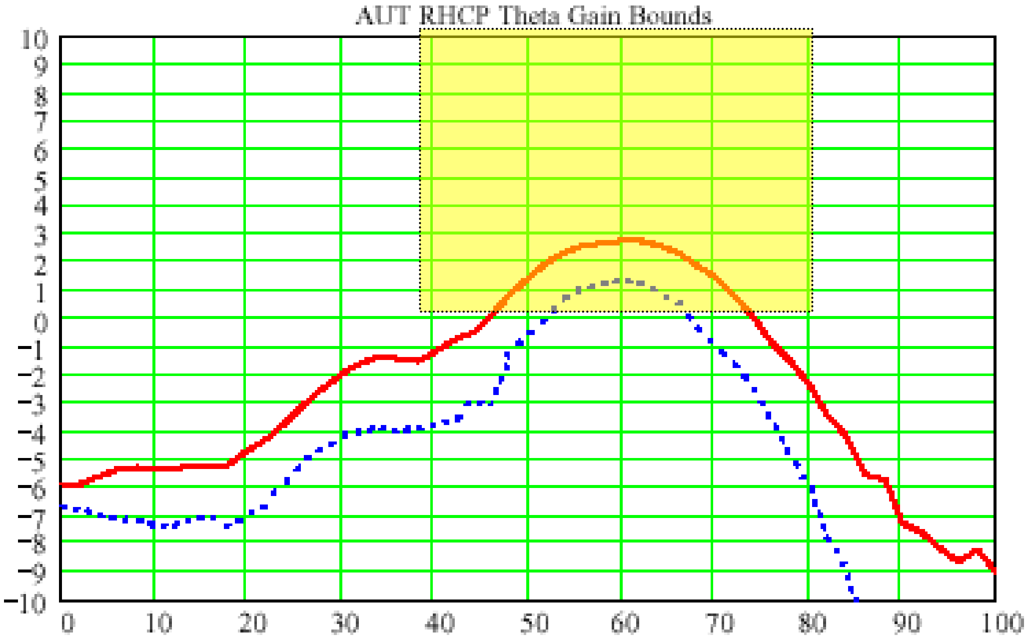
Comparison human/evolved

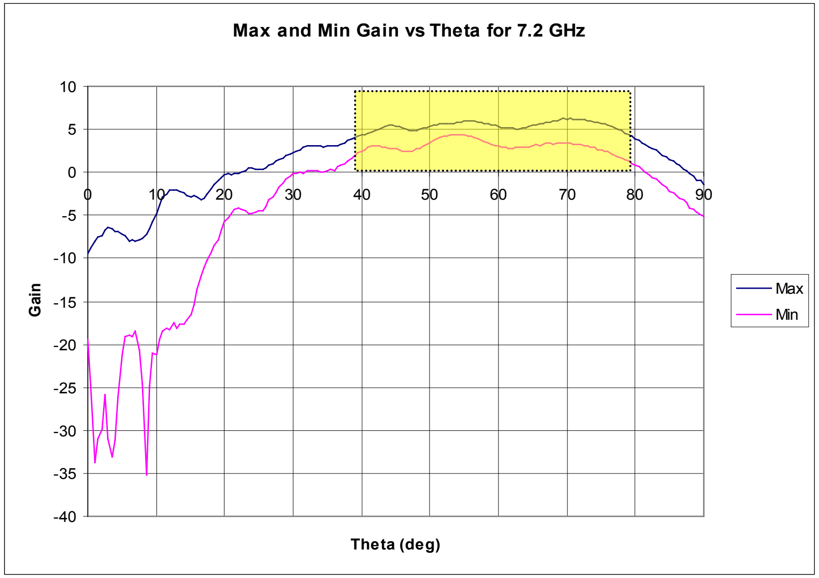
Human



Evolved







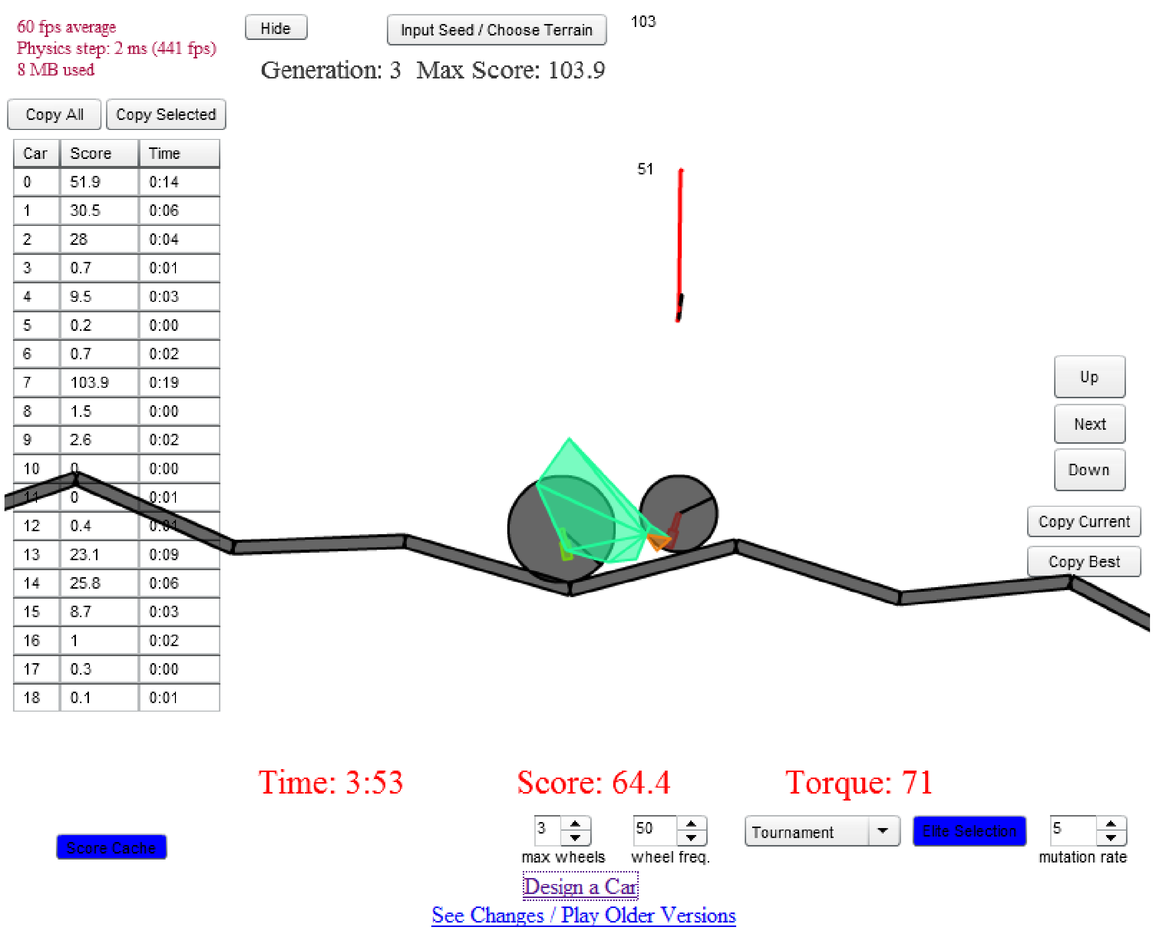
Example: Evolving of Vehicle Structures

BoxCar 2D

http://boxcar2d.com/

Home | Designer | Best Cars | Forum | News | FAQ | The Algorithm | Versions | Contact

Computation Intelligence Car Evolution Using Box2D Physics (v3.2)



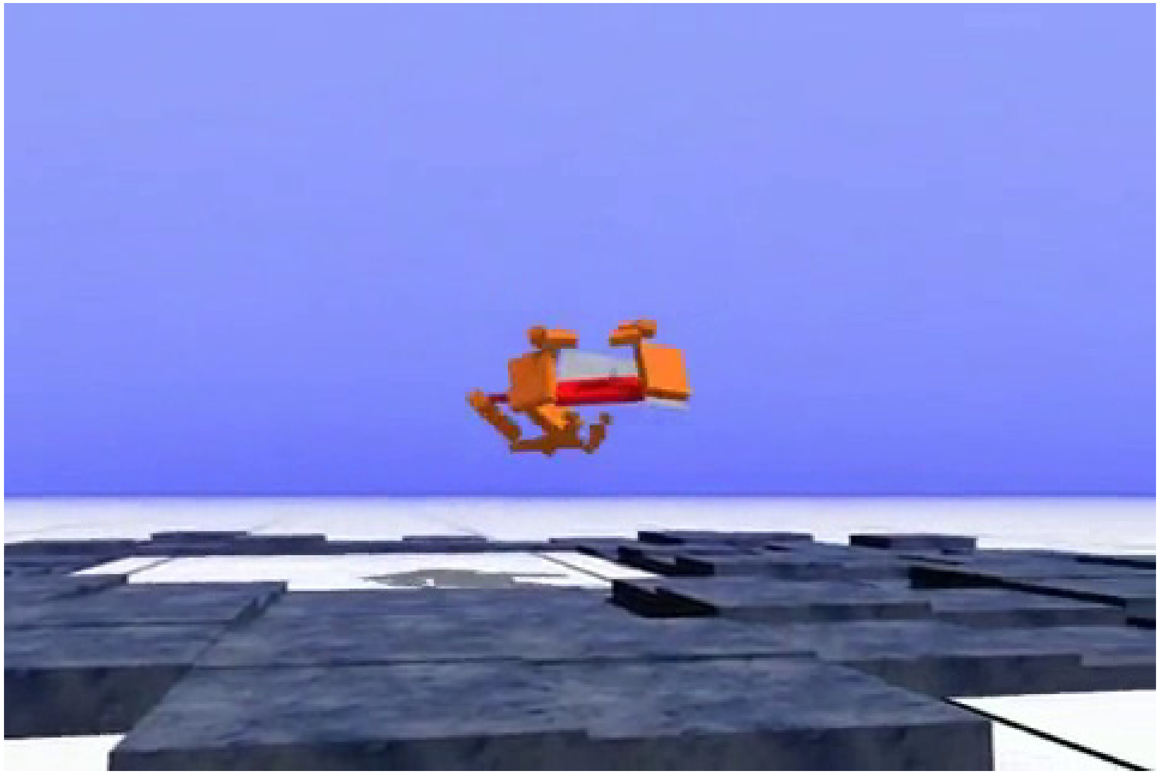
Discussion

- Pros:
  - General heuristic
  - Able to find good solutions in feasible computing time
  - Distributed execution possible
- Cons:
  - Cannot guarantee optimal solutions
  - Long runtimes
  - Success depends also on used parameters

Further Readings:

- Eiben, A.E., Smith, J.E., *Introduction to Evolutionary Computing*, Springer, 2003

Outlook: Evolving of Creatures



Motivating Example – new Perspective