THE BIOROBOTICS INSTITUTE

Scuola Superiore Sant'Anna

Introduction to Evolutionary Systems

Robotics course - MSc program in *Computer Science*, 2016
Teacher: Prof. Cecilia Laschi

Teaching Assistant: **Francesco Corucci**
PhD Student in BioRobotics
M.Eng. Computer Engineering
f.corucci @sssup.it
Website

April 28h, 2016

Outline

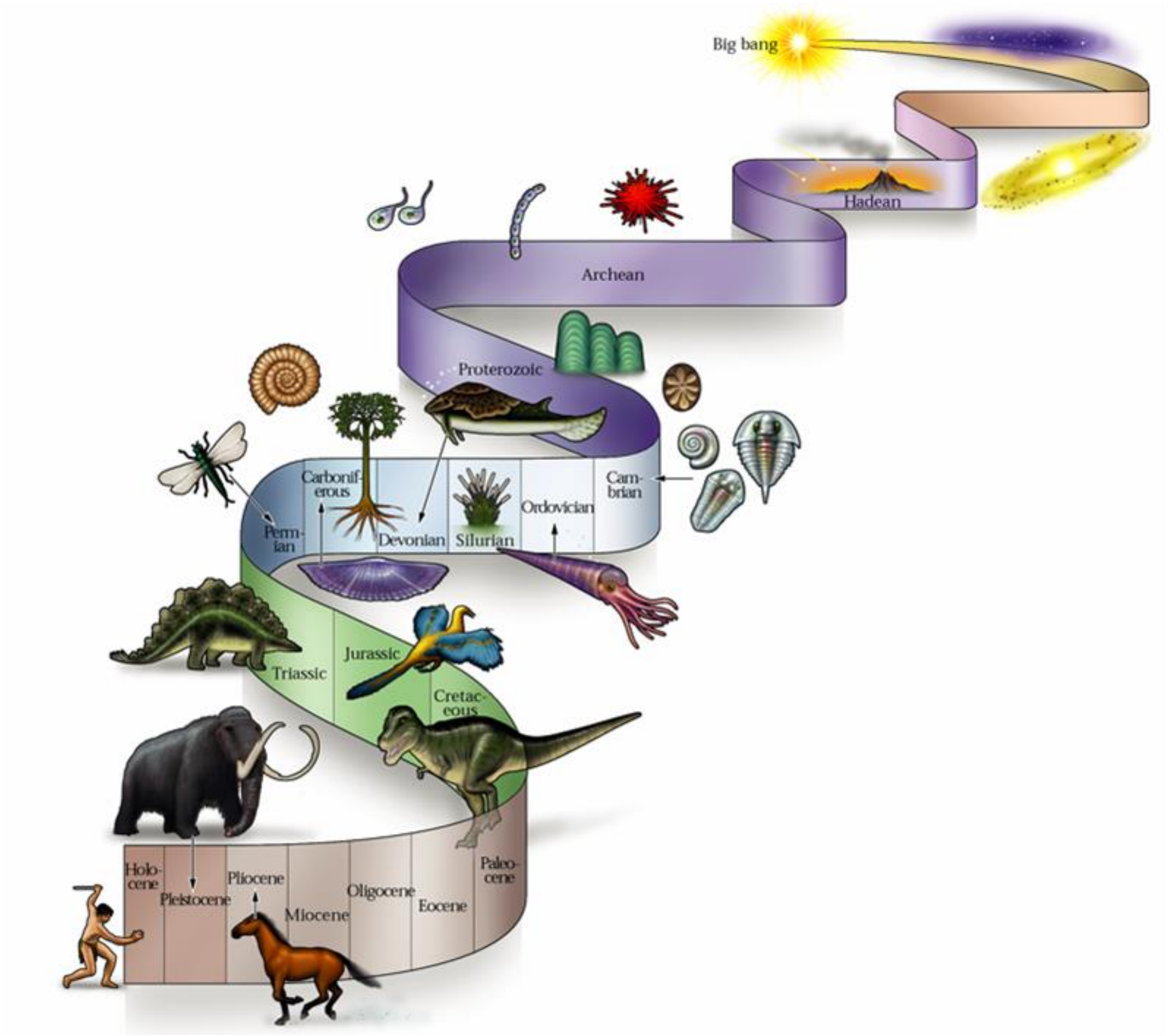
1. **Natural Evolution**
2. **Artificial Evolution**
3. **Some applications of Artificial Evolution**
4. **Neuroevolution**: how to evolve neural networks
5. **Evolutionary Robotics**: how to evolve complete robots

Natural Evolution

Natural evolution

- All biological systems are the result of an ***evolutionary process***
- Those systems are highly:
  - Robust
  - Complex
  - Adaptive
  - Extremely sophisticated
- Robots and artificial systems in general typically lack of these characteristics

→ Source of inspiration



Let's take a look at some of the products of ***evolution****….*

Evolved biomechanics

Cheetah

Peregrine falcon

Manta ray

Evolution and adapation to the *ecological niche*

- Adaptation to the environment: <u>body coverings</u> (mimicry), body parts, behaviors

Leaf-tailed gecko

Walking stick

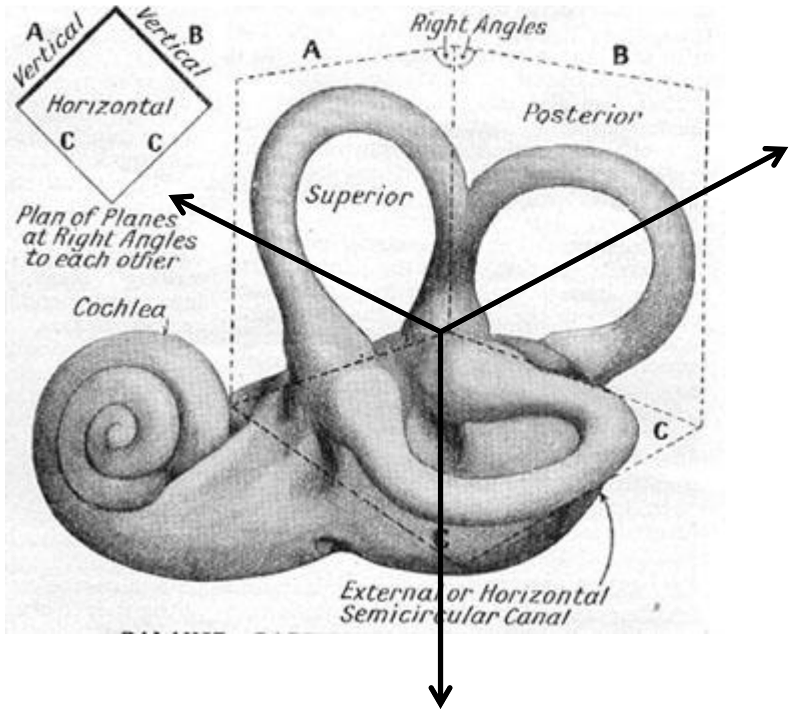
Green leaf Katydid

Chaetodon capistratus

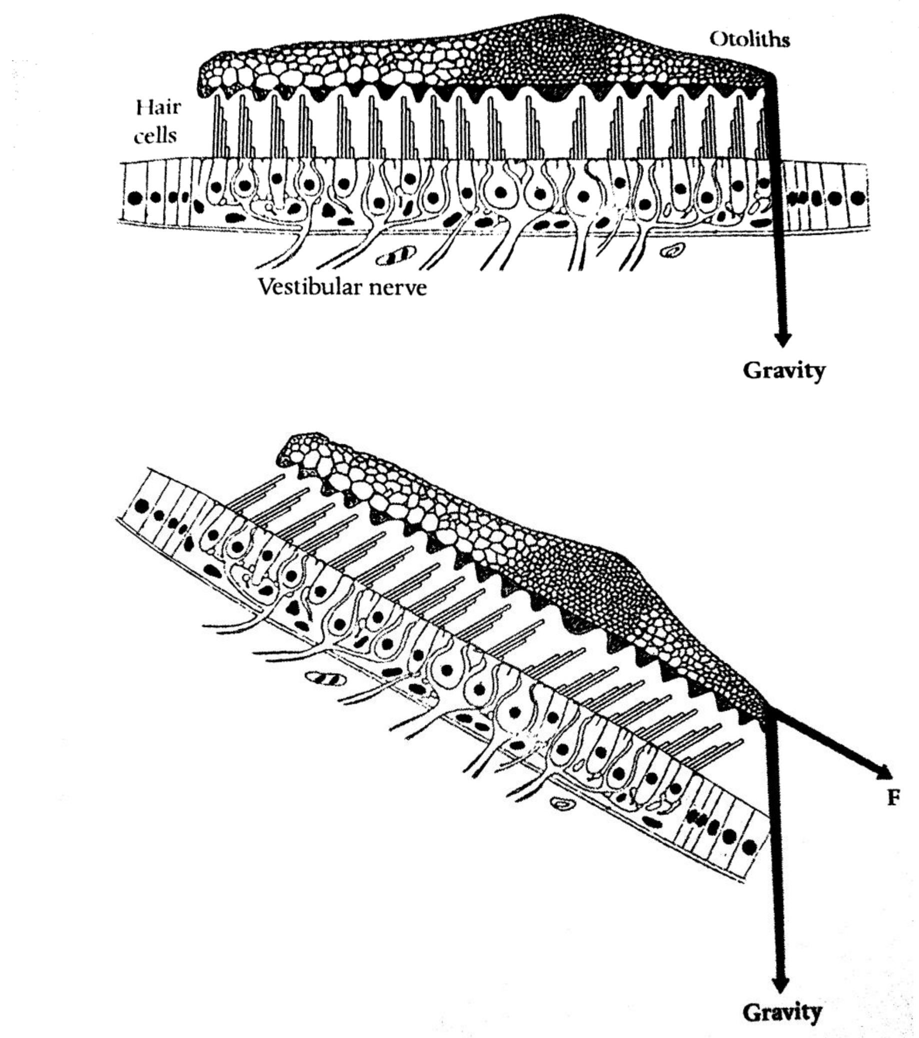
Non toxic butterfly mimics a toxic one

Evolved Sensors – vestibular system

Semicircular canals, detecting angular accelerations



Otoliths, detecting linear accelerations and tilting.
In some animals (e.g. insects) adapted to also detect vibrations

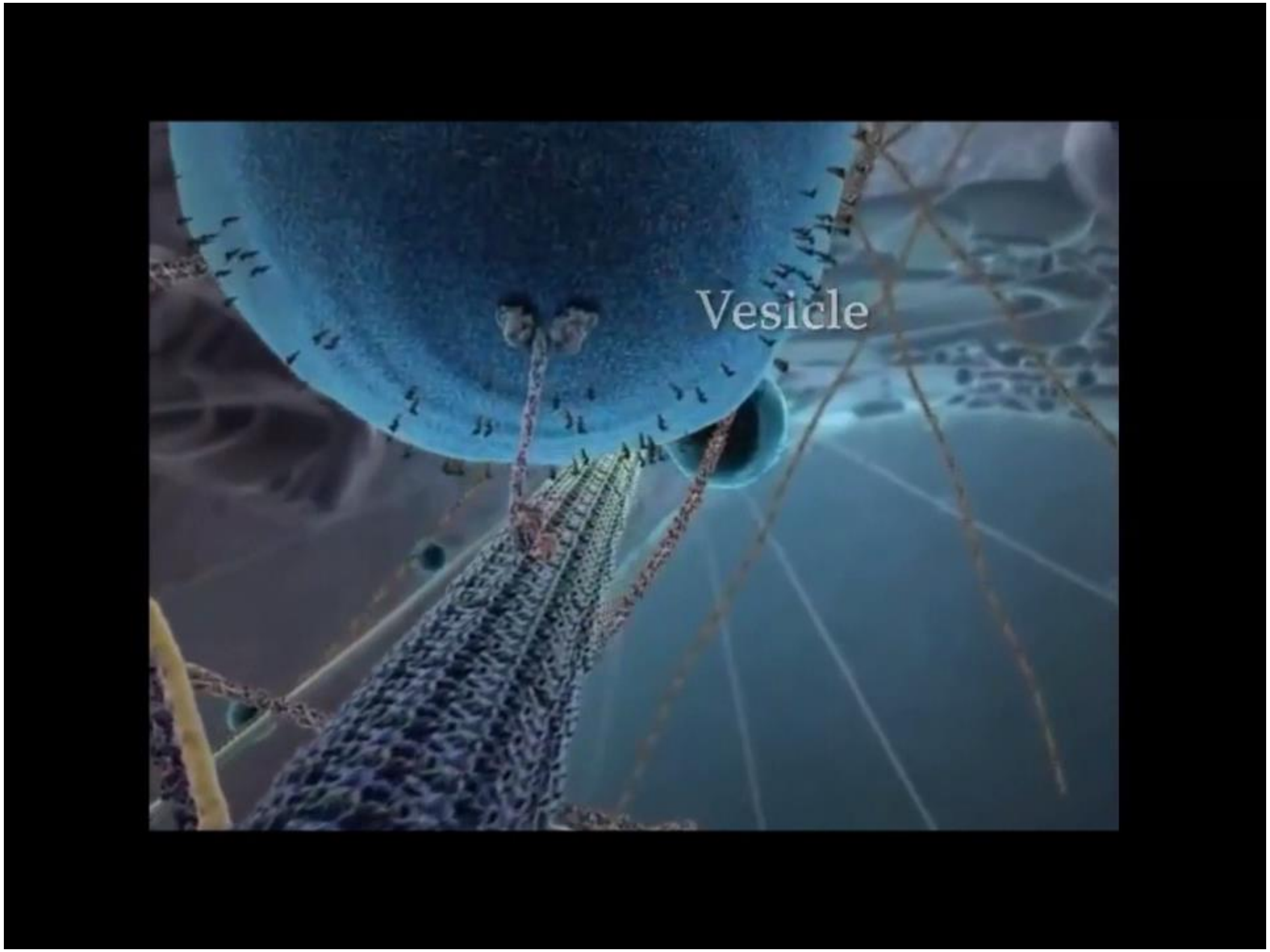


Remarkably sophisticated solutions!

Evolved Complexity at the micro scale

ATP Synthase *– a protein-based micro rotational motor*

Evolved Complexity at the micro scale



The inner life of the Cell - BioVisions, Harvard University – http://multimedia.mcb.harvard.edu

Another product of Evolution…

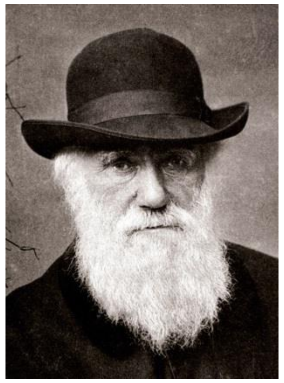
Biological Inspiration

- Some of the features exhibited by biological creatures are desirable also for artificial ones (e.g. robots)
- Since these features have been produced by natural evolution it makes sense to try emulate such a process in an artificial way

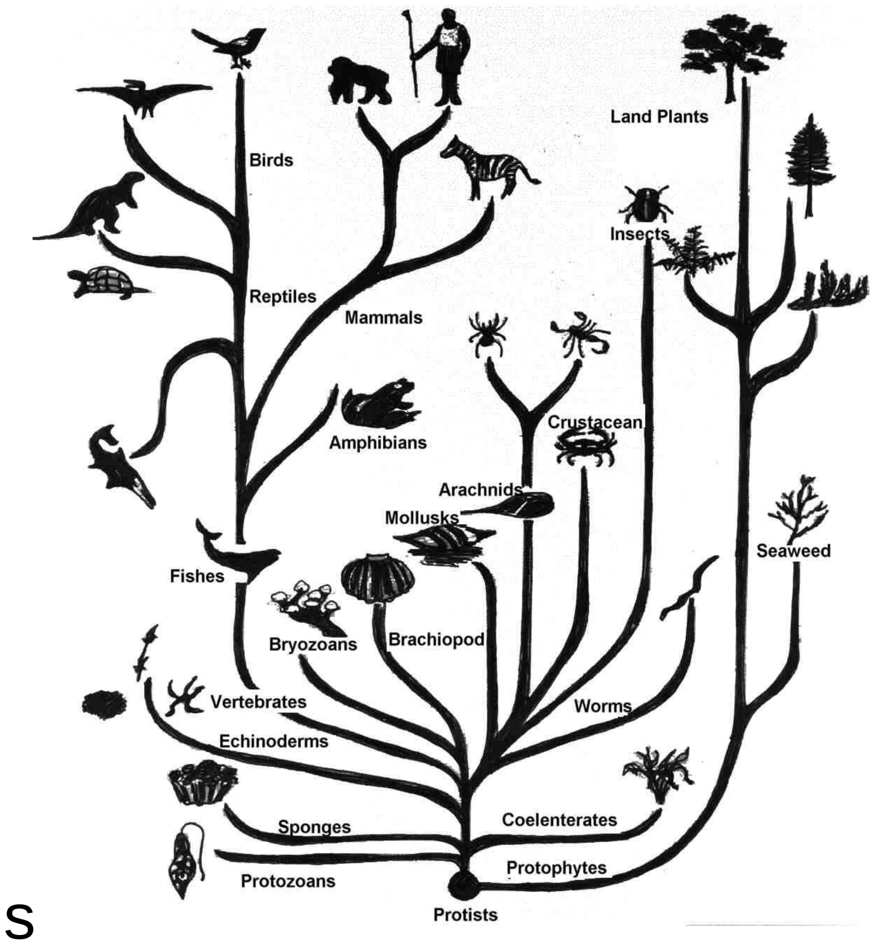
→ we'll talk about **Artificial Evolution**

Let's first take a look at how biological evolution works…

Biological Evolution



“All species derive from a common ancestor”, Charles Darwin, “On the Origins of Species”, 1859



The four pillars of Evolution:

1. **Population**: Evolution is based on groups of individuals
2. **Diversity**: Individuals in a population have different characteristics
3. **Heredity**: Characteristics are transmitted over generations through reproduction
4. **Selection**: Limited resources in the environment → Not all individuals will survive nor reproduce
   The better an individual (food gathering, mating) → The more chances to survive and reproduce → The more offsprings → The more probable that individual’s traits are propagated.
   Selection depends on many factors

Genotype and phenotype

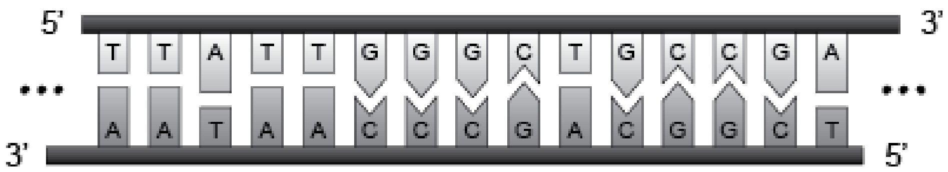
Genotype:

- Genetic material of an organism
- Individual’s traits are encoded there
- It is transmitted during reproduction, and affected by mutations
- Contains the “blueprint” to build the organism

Phenotype:

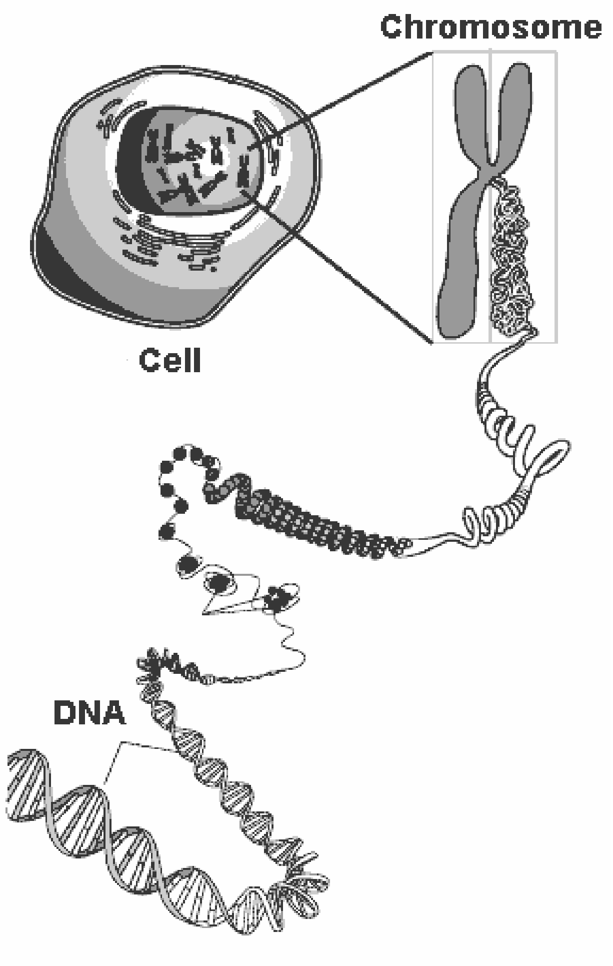
- Manifestation of the organism (appearance, behavior, etc.)
- Selection operates on the phenotype
- Affected by environment, development, learning, …

Genetic material

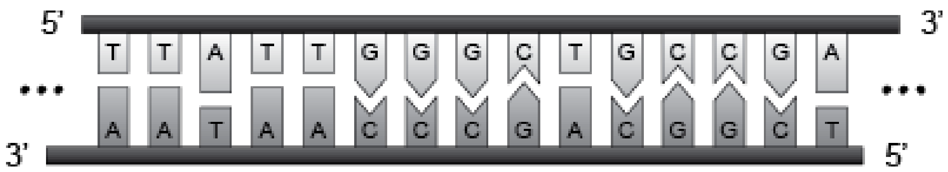


DNA

- Long molecule, twisted in spiral, present in the nucleus of the cells
- All cells have the same genetic material
- Two complementary strands composed of four types of chemical units (nucleotides/bases) “ATCG” → letters of the “genetic alphabet”
- Pairs of complementary nucleotides can bind together (A-T, C-G)
- The DNA string is interpreted via processes called *transcription* and *translation*, that ultimately lead to the expression of encoded traits

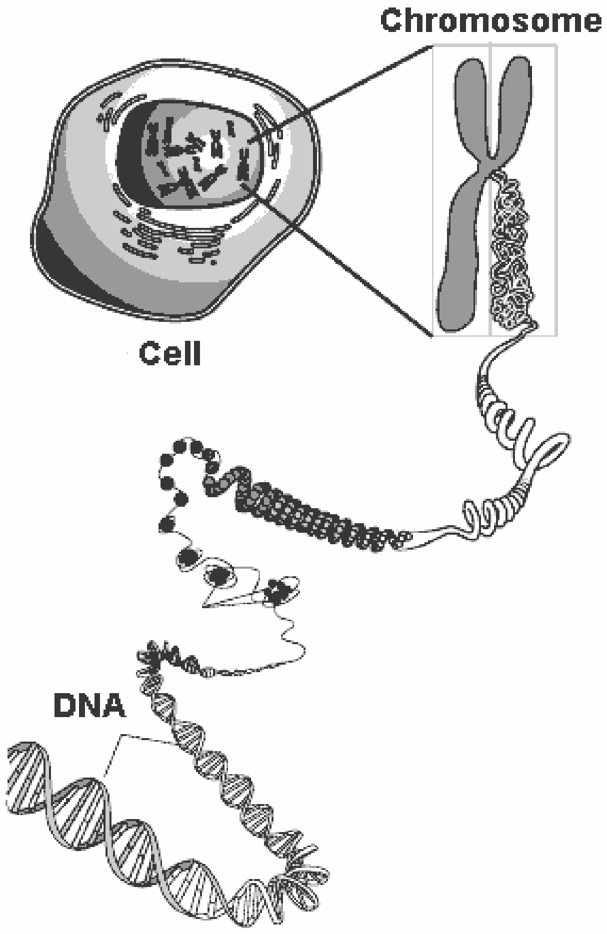


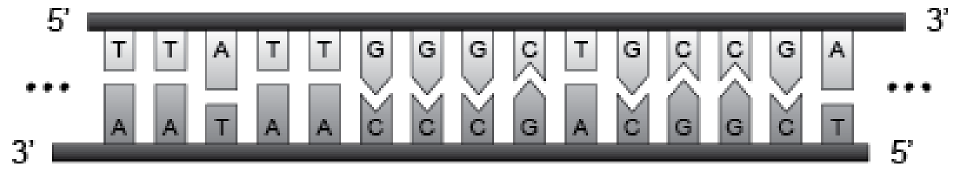
Genetic material



Chromosomes

- The genetic material is organized in several separated DNA molecules called *chromosomes*
- In *diploid* species chromosomes occur in pairs
- Redundancy (2 strands, 2 chromosomes) allows replication of DNA molecules during cell
- During reproduction (in diploid organisms) child cells receive one chromosome from each parent

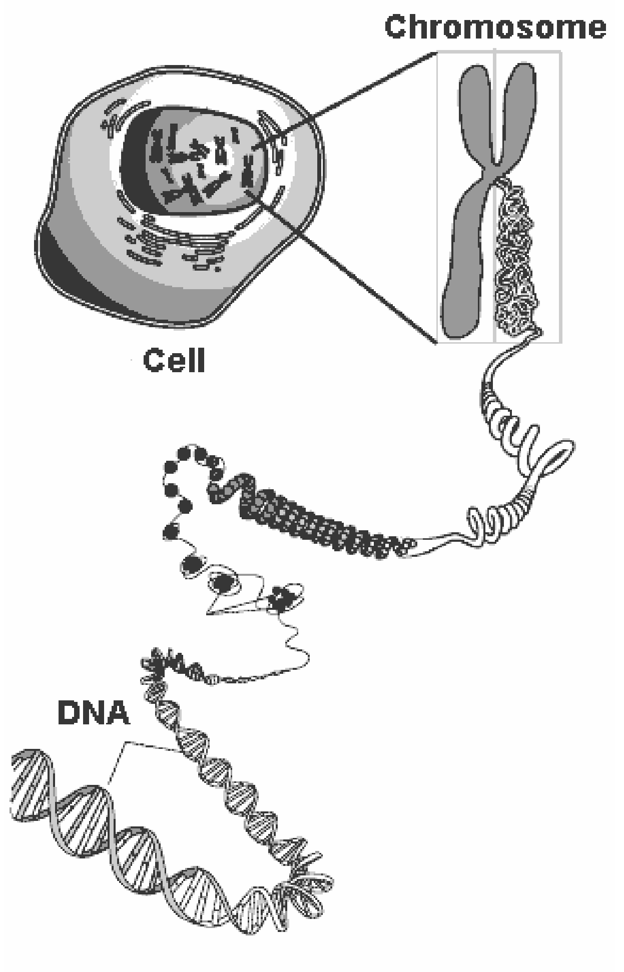




Genetic material

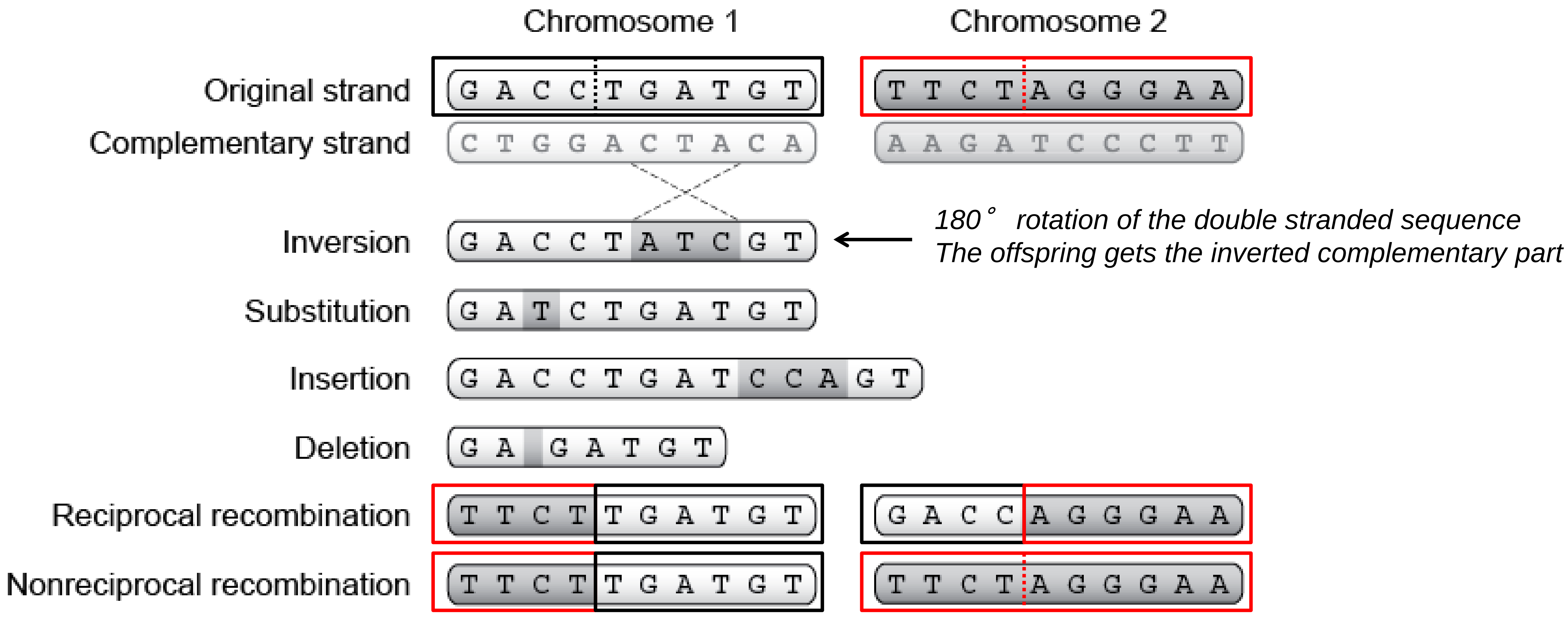
Genes

- Functionally relevant sub-sequences of several nucleotides in the DNA chain (e.g. encode instructions for the production of a protein)
- If nucleotides are letters of the genetic alphabet, genes are words
- The particular sequence of nucleotides in a gene determines (through a process of *gene expression*) the characteristics of the associated gene product (usually proteins), affecting cells' properties and thus specific traits of the phenotype



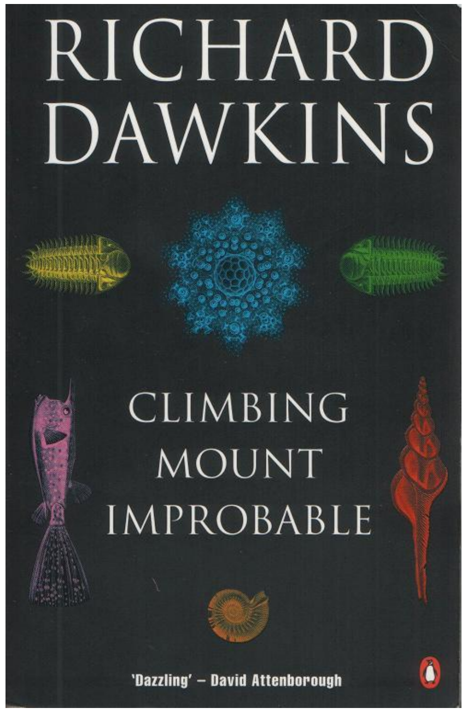
Genetic mutation and recombination

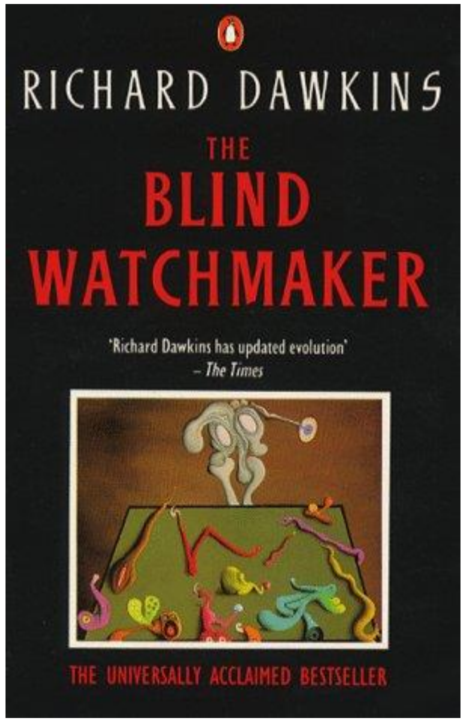
- Error-prone replication mechanisms → <u>Mutations and recombinations</u> → Original traits arise
- Mutations and recombinations occurring during sexual reproduction (meiosis) affect the evolution of the species



A random, blind process

- Natural evolution relies mostly on *random* dynamics
- The only non-random criteria involved are the ones determining *survival* and *reproduction*
- It is *blind (non goal-directed)* and *open-ended* (does not end)
- It's hard, though, to imagine how something sophisticated such a human can *emerge* from such a process
- *Frame-of-reference problem* (or *antropomorphization* risk) also common to AI
    - Projecting our human understanding onto observed phenomena that may in fact be far more simple than they look

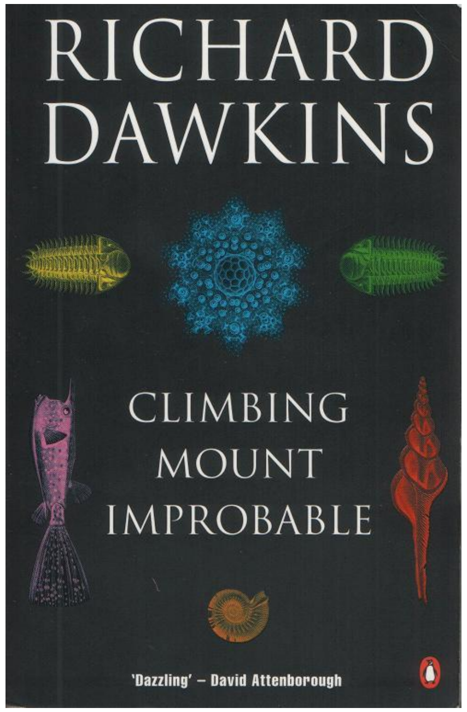


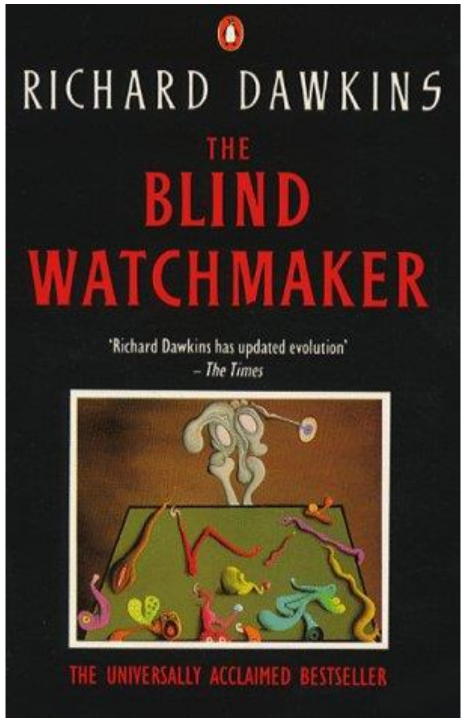


A random, blind process

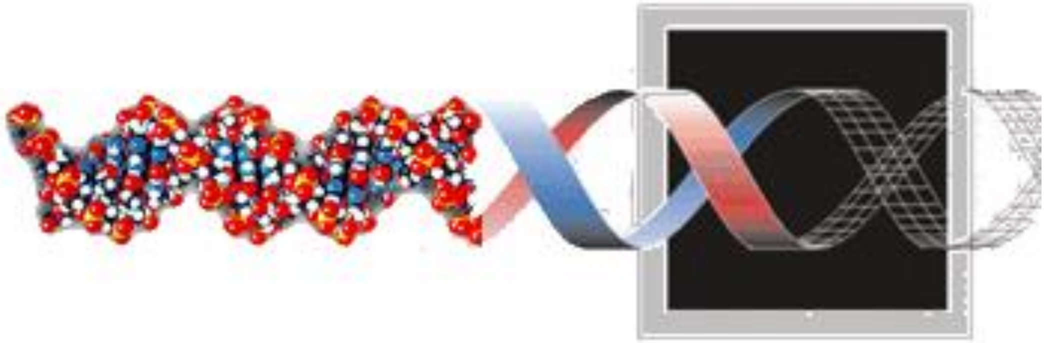
<u>Some insights</u>:

- Evolution proceeds by *<u>gradual adaptation steps</u>*
- Powerful allies: the *<u>self-organization</u>* properties of the physical world
- E.g. (Eggenberger Hotz, 2003): showed in simulation how complex shapes (e.g. a lens, intermediate product of an eye) can easily emerge during evolution exploiting self-organizing phenomena of cells (e.g. cell adhesion)





Artificial Evolution



1001101010001

Artificial evolution

- Includes a wide set of algorithms inspired (to different extents) from the natural evolution
- **Can be used with different goals in different settings, e.g.:**
  - To solve complex optimization problems (e.g. in engineering)
  - To automatically design robots, both in terms of control and morphology (evolutionary robotics)
  - To study properties of biological systems (artificial life, computational biology)
  - To evolve cognitive behaviors (artificial intelligence)
  - …

Some important differences

Natural evolution	Artificial evolution
Is <u>open-ended</u>	Usually* <u>has</u> an end
Does <u>not</u> have an ultimate goal	Usually* <u>has</u> a specific goal
Selection is based on very indirect evaluation criterions, i.e. <u>survival</u> and <u>reproduction</u>. Traits can be selected that are not useful from an engineering perspective (e.g. attract individuals of the opposite sex but make a easier prey, …)	Selection is usually* based on a very precise, task-based function (the <u>*fitness*</u> *function*), quantifying performances
Does <u>not</u> proceed towards an *optimum*, selection occurs in the <u>*here-and-now*</u>: no comparative memory. E.g. a prey is successful with respect to the *current* generation of predators.	Usually* we want it to proceed towards an optimum (during artificial evolution, current solutions are usually <u>better</u> than previous ones)

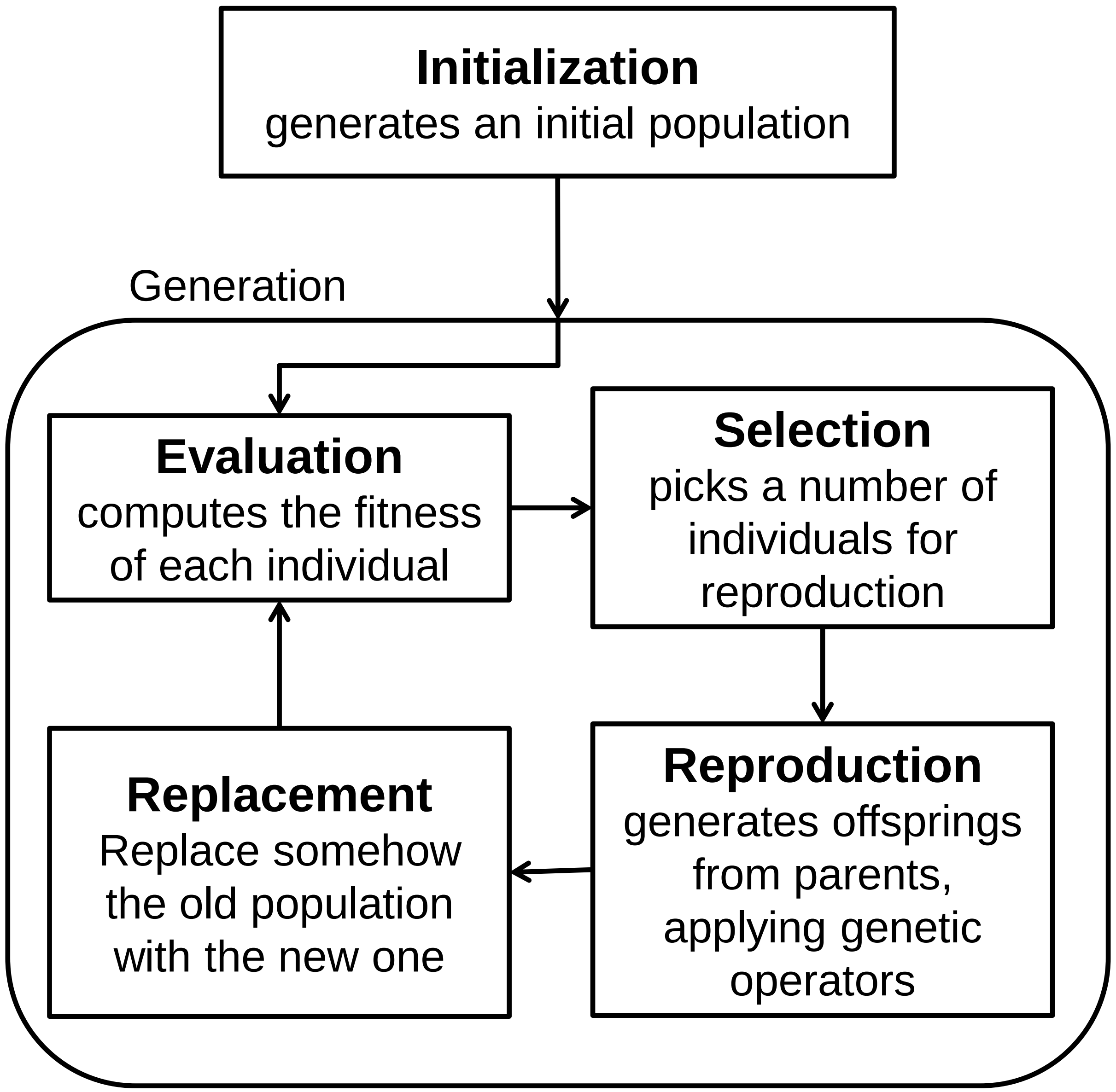
* This is true mostly when artificial evolution is used in engineering contexts. There are approaches to artificial evolution that are open-ended, not goal directed, and that mimic survival and reproduction

Artificial evolution

Ingredients:

1. A **genetic representation** (a way to *encode candidate solutions*)
2. An initial **population** (e.g. random set of candidate solutions)
3. A **fitness** function (quantifies how good each solution is, assigning a scalar *score* to them)
4. A **selection** method (usually selecting with higher probability individuals with high fitness)
5. **Crossover** & **mutation** genetic operators (come into play when offspring-solutions are generated from selected parents)

Artificial evolution



<u>Iterative procedure, termination criteria</u>:

- Fitness reached a given threshold
- Fitness not improving for several generations
- Maximum time, number of generations, ….

Genetic representation - Encoding

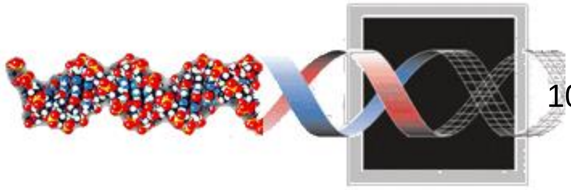
A first distinction:

- Direct encodings: each parameter appears directly and explicitly into the genome, i.e. the genotype directly maps to the phenotype.
- Indirect/generative encodings: the genotype indirectly encodes the phenotype, e.g. it encodes parameters governing a *development process* implementing the genotype-to-phenotype mapping

Example: Image you want to evolve a robot morphology for walking given a fixed activation

- You could decide that the body has a fixed structure, and use artificial evolution to find the lengths of the joints that better allow the robot to walk → Direct encoding
- You could, instead, decide that the body structure is encoded by a developmental process (e.g. based formal grammars such as L-system) and use artificial evolution to find the expression that results in the best body structure → Indirect encoding ("*artificial embryogeny")*

Genetic representation - Encoding



<u>Another example</u>: Image you want to evolve a neural network for controlling a robot

- Fix a priori the structure of the network, encode the weights into the genome, let artificial evolution find networks that perform well for a given task → <u>Direct encoding</u>
- Encode into the genome the parameters governing a distribution, from which neural weights will be sampled, or even parameters governing a <u>process of neural growth</u> → <u>Indirect encoding</u>

<u>Considerations</u>:

- Direct encoding is probably the most adopted, especially in engineering contexts
- Indirect encodings are more <u>compact</u>, and greatly improve <u>scalability</u>, reducing the number of parameters i.e. the dimension of the search space
- Also, generative encodings allow <u>complexification</u> over time

Genetic representation - Encoding

- In a problem-solving setting, domain knowledge is crucial in defining the encoding, which parameters are relevant, how to constrain them, etc.
- In the parallel with natural evolution, usually great simplifications are done:
  - Single stranded sequence of characters
  - Fixed length
  - Haploid structure, just one chromosome
  - Often direct genotype to phenotype mapping
  - …

Discrete representations

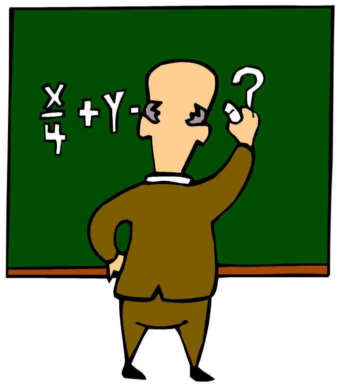
- A sequence of *L* discrete values drawn from an alphabet with cardinality *k*
- E.g. a binary string (*L=8, k=2*) → <u>Population = set of binary strings</u>
- Could represent different things in different settings

to configuration string of
FPGA electronic circuits

01010100

to integer *i* using
binary code

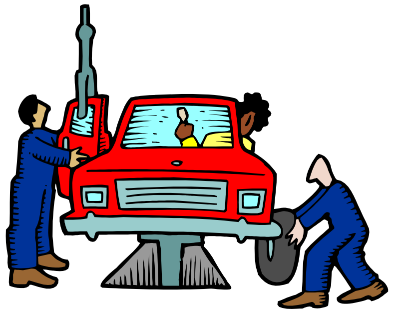
84



0.328125

to real value *r* in range [min, max]:
$r = min + (i/255)(max-min)$

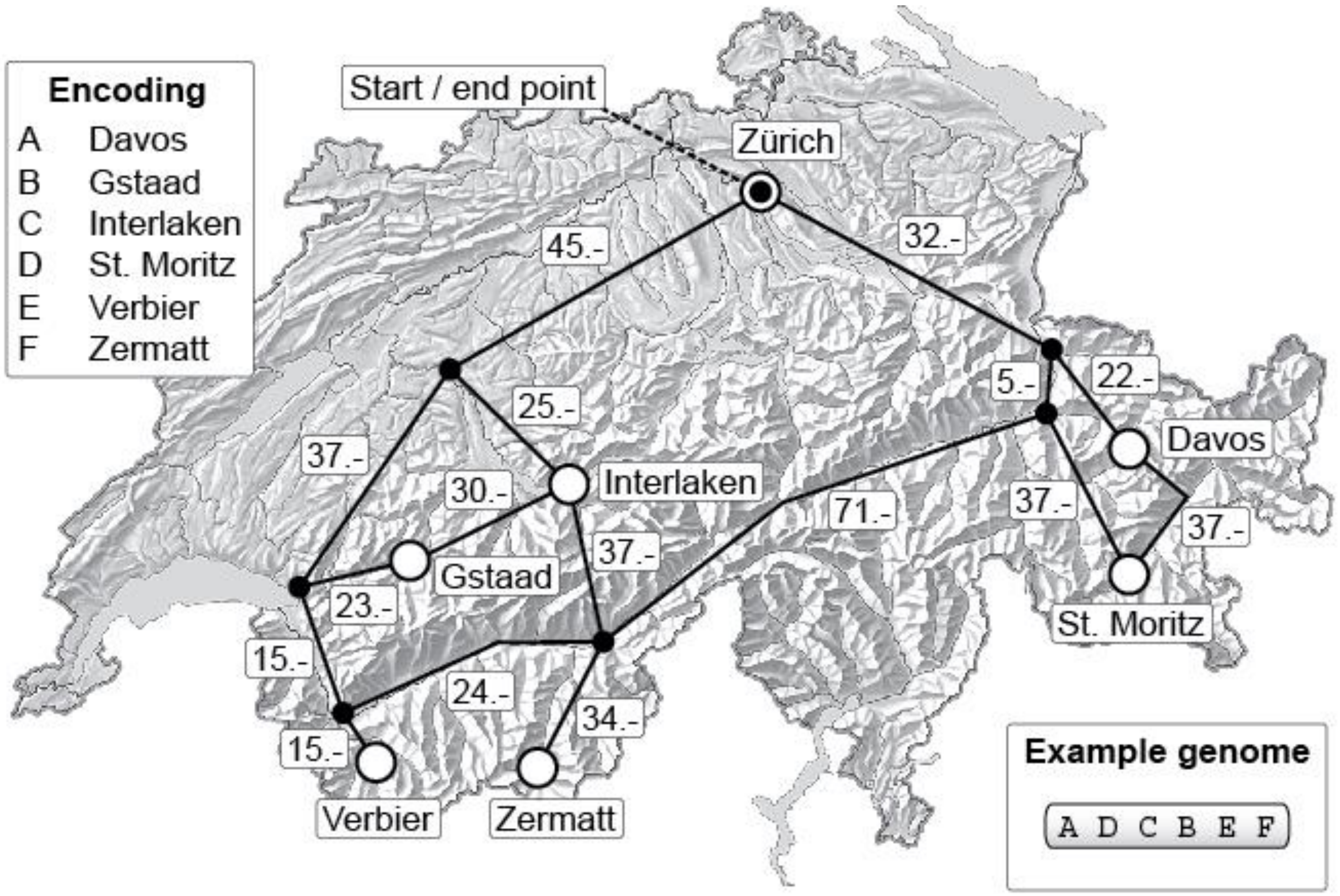
Job	A.M.	P.M.
1	x	
2		x
3	x	
4		x
5	x	
6		x
7	x	
8	x	



to job schedule:
- job=gene position
- time=gene value

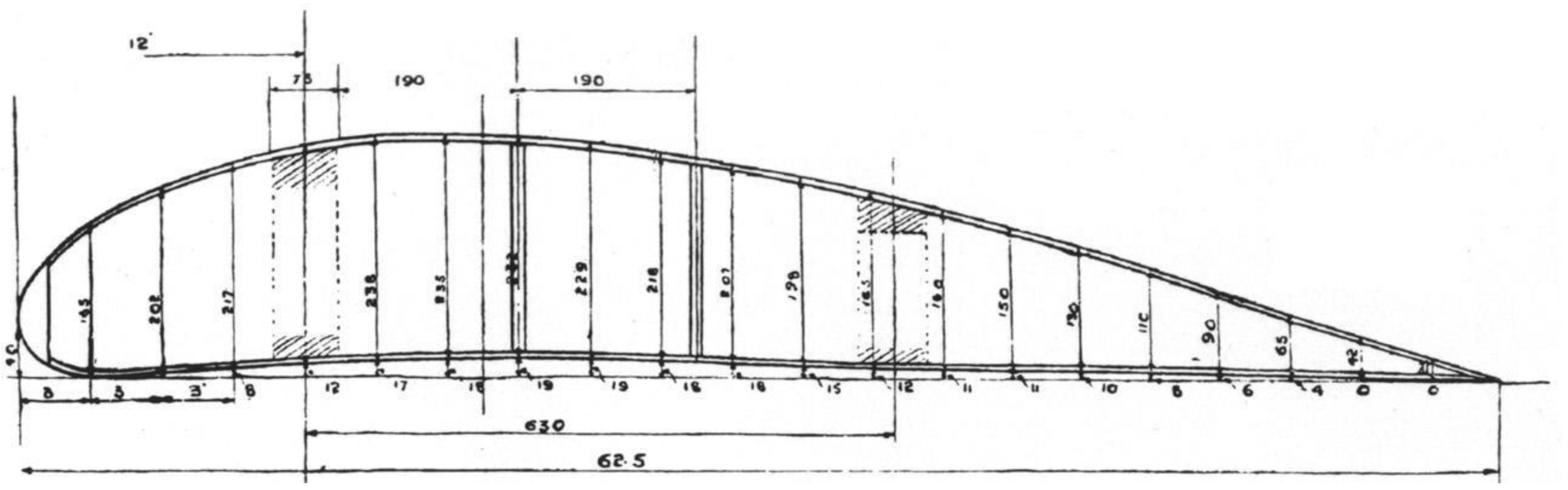
Sequence representation

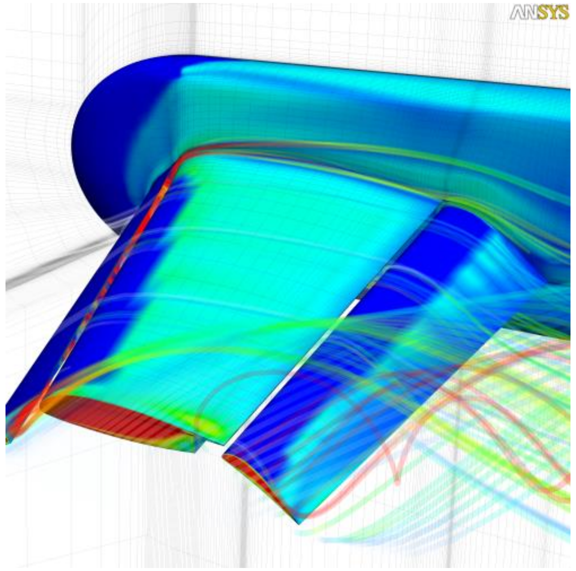
- A particular case of discrete representation for problems in which the solution is a *sequence*
- E.g. if you have to solve a TSP instance, planning a long trip minimizing transformation costs



Real-valued representation

- The genotype is a vector of real number, associated to relevant parameters
- Useful e.g. in parameter optimization for an engineering problem
- Example: evolve the shape of a wing to improve its efficiency. The genotype could encode relevant dimensions



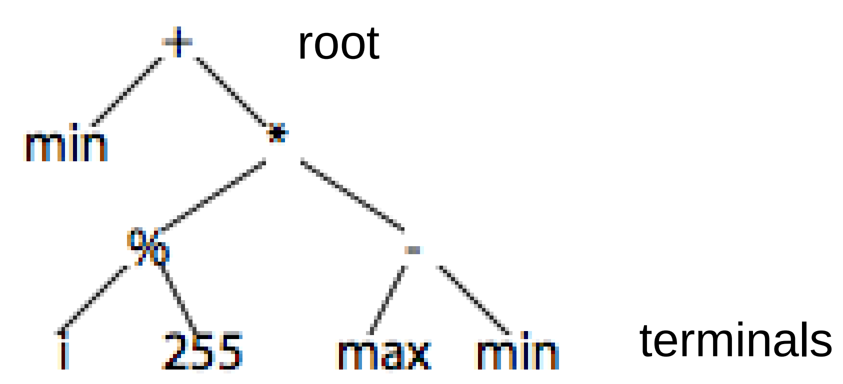


Tree-based representation

- The genotype describes a tree with branching points and terminals
- Suitable to encode <u>hierarchical structures</u>
- Used e.g. to encode and <u>*evolve computer programs*</u> (e.g. <u>*genetic programming*</u>), that can be represented as a tree of operators (from a *functions set*) and operands (from a *terminals set*)

Expression r=min+(i/255)(max-min)

Nested list (+, min, (*, (/, i, 255), (-, max, min)))



- <u>An application of genetic programming:</u> **symbolic regression**
  - An intriguing example: *"The Robot Scientist"* - <u>http://en.wikipedia.org/wiki/Eureqa</u>

Initial population – how big?

Some guidelines:

- The higher the dimension of the search space (i.e. the more parameters in the genome), the bigger should be the population

  → **Risk?** local optima (***premature convergence***)

- When initializing, you should avoid generating homogeneous populations (in terms of fitness scores) to improve *evolvability* (ability of the algorithm to make progresses)
  - The higher this risk, the larger should be the population (E.g. evolve locomoting robots)
- In the end, especially in practical applications, it's often a trade-off constrained by time, related to the computational cost of evaluating the fitness
  - Each generation takes $t_{fitness} \cdot PopulationSize$, and typically hundreds of generations are allowed ($\text{time} = \mathrm{N}_{generations} \cdot t_{fitness} \cdot PopulationSize$)… Usually you know $t_{fitness}$, and you know how patient you are…
- Typically, in the order of thousands individuals

Initial population – how to initialize?

1. Have a starting point? (e.g. fairly good solution achieved with another optimizer, or manually devised)

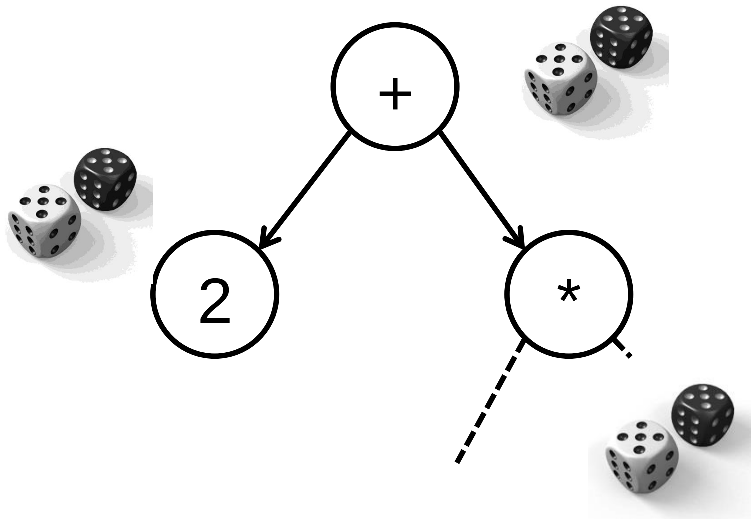
   → Initialize population with «variations» of the starting point

2. If you don't: **random** (most common)

   - You may use option 2) also in case 1) → seeding the algorithm can result in less diversity, and may bias the algorithm towards sub optima

Random initialization:

- Binary representation → $PopulationSize$ random strings of bits of $GenomeLength$
- Real-valued representation → $PopulationSize$ random samples from a given interval
- Tree-based representation → recursive process from the root, expanding each node into randomly sampled branches (until a maximum depth)



Fitness function

- It is a function F associating a scalar (fitness value/*score*) to each phenotype (f)
- Evaluating the fitness function is usually the most time-consuming part of an evolutionary algorithm
  - e.g. entails running a physically realistic simulation, let a robot act for some time, etc.
- The fitness (usually) quantifies individuals' *performance* (in terms of what you do want to optimize) → domain specific objective

Fitness function

- The fitness function can embed one or more components (*multiple objectives*)
- E.g. evolution of a swimming robot in a 2D world, maximum problem:

$$fitness(f) = spaceTraveled_x$$

$$fitness(f) = 0.8 \cdot {spaceTraveled_x}^2 + 0.2 \cdot {spaceTraveled_y}^2$$

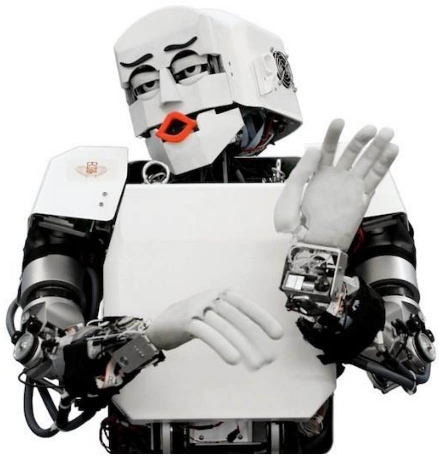
$$fitness(f) = \frac{0.8 \cdot {spaceTraveled_x}^2 + 0.2 \cdot {spaceTraveled_y}^2}{energySpent}$$

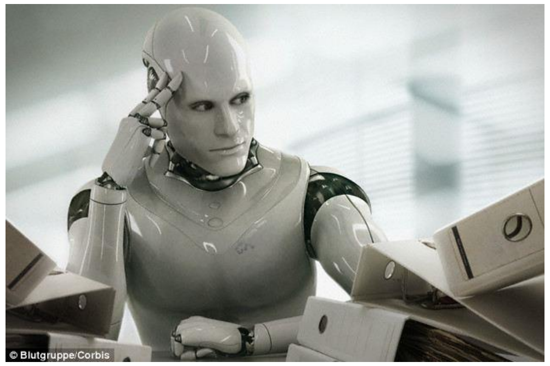
…

- <u>Which component(s) to choose? How to combine/weight them</u>?
  - Again, no precise rules: domain knowledge, trial-and-error, …

Fitness function - Observation

- <u>Note</u>: In general the fitness allows you to specify every kind of <u>high-level performance metric</u>
- Also <u>**subjective**</u> ones, that you are not able to «code»/express (but that are coded in your head!) → <u>**Interactive evolution**</u>
  - «User appreciation for an evolved picture/song» → may produce an artistic agent!
  - «Number of times the robot said/did something funny» → may produce a comedian robot!
  - «Number of times the robot appeared to behave intelligently»
- <u>Interesting from an AI perspective</u>: evolutionary approaches are more general with respect to others that require a strict formulation of the problem
- These algorithms are free to find their way to meet such «blurry» requirements





Subjective fitness and Interactive Evolution

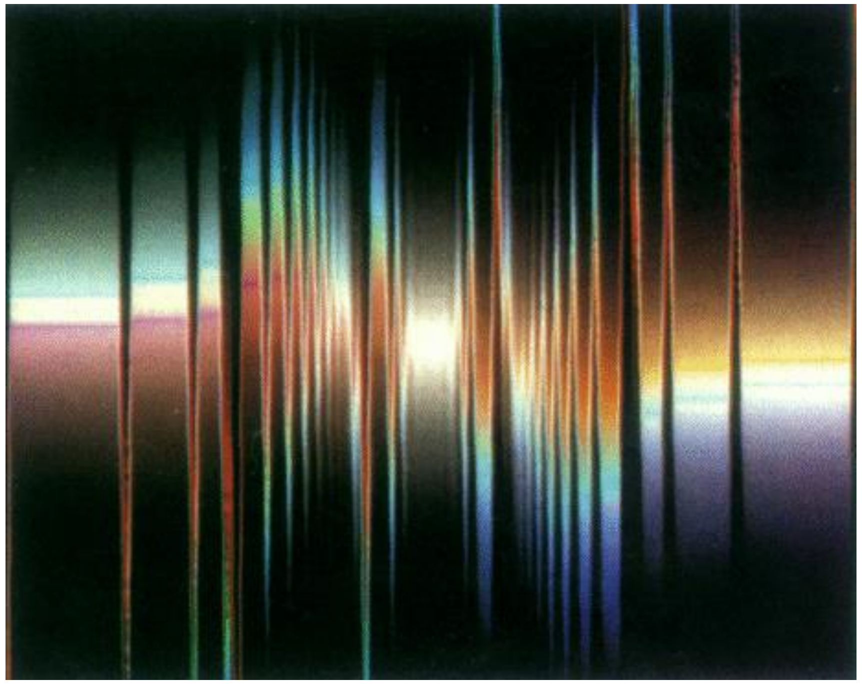
- *Karl Sim's "**Genetic Images**"* (1993) is a media installation in which visitors can <u>interactively "evolve"</u> abstract still images.

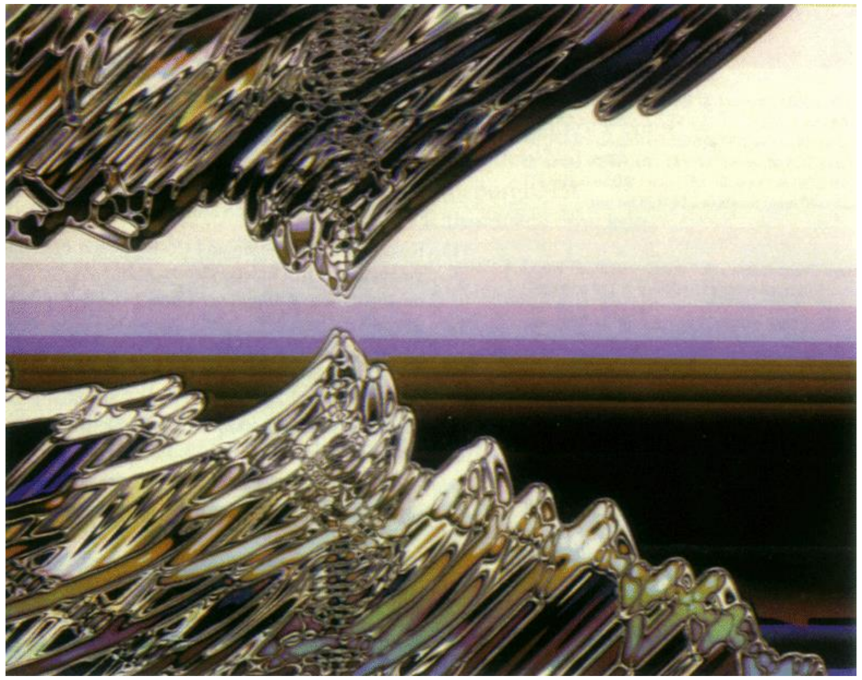
- A supercomputer generates and displays 16 images on an arc of screens.

- Visitors stand on <u>sensors in front of the most aesthetically pleasing images</u> to select which ones will survive and reproduce to make the next generation

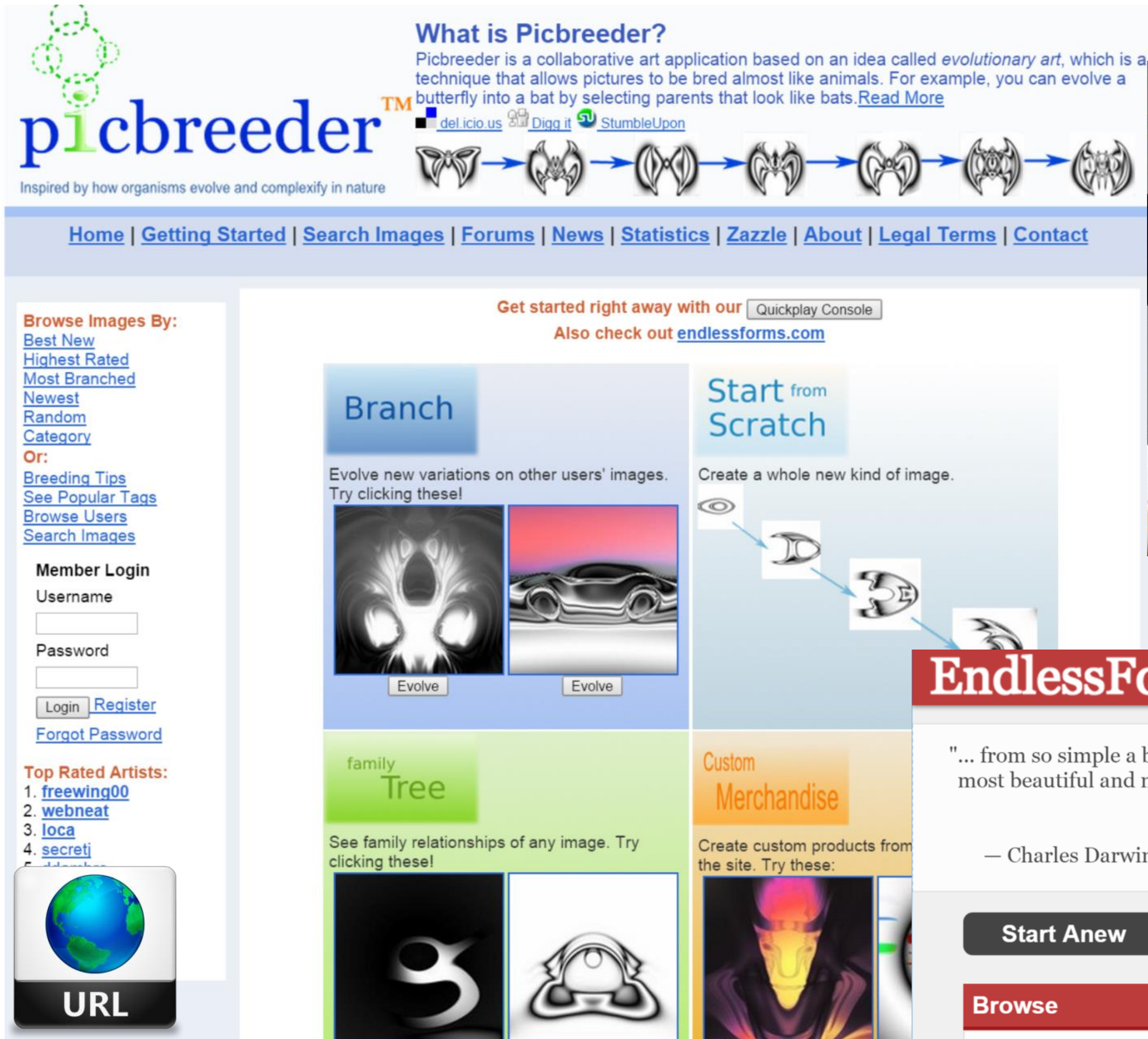
→ <u>Fitness</u>: user appreciation, how long they stared at an image

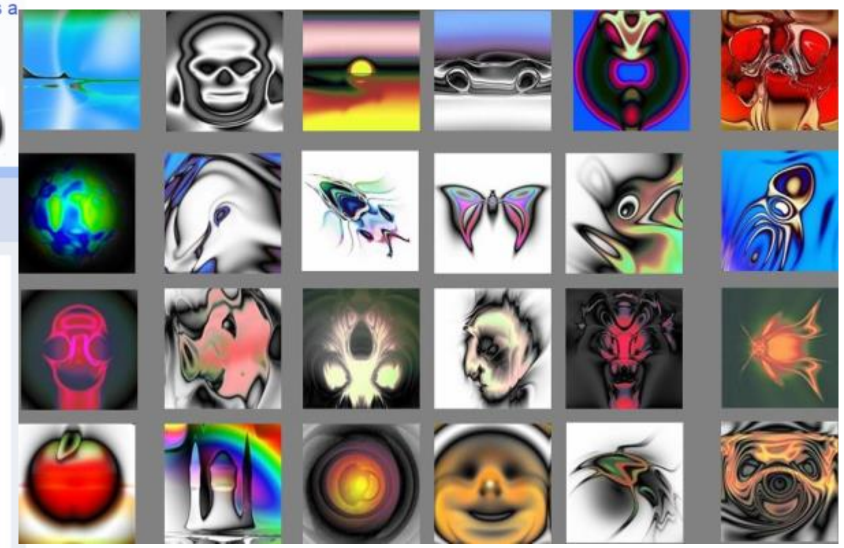
Subjective fitness and Interactive Evolution

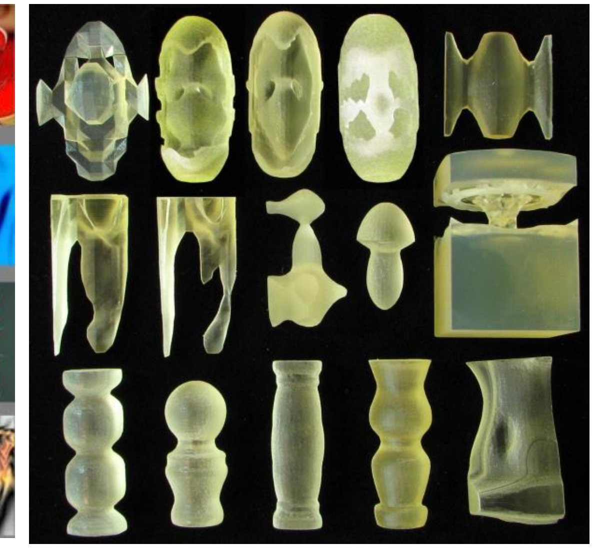


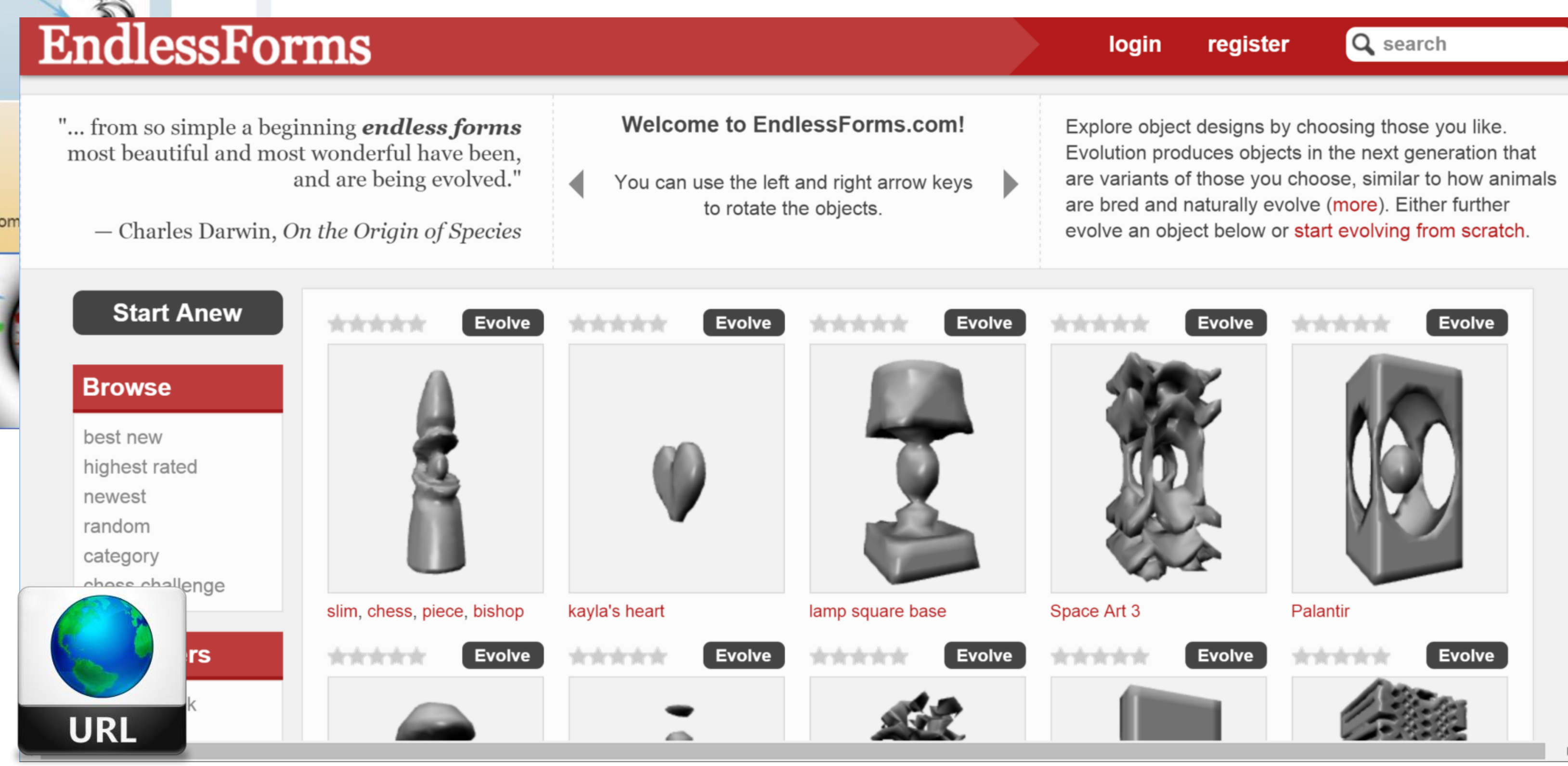


Subjective fitness and Interactive Evolution









Selection and selection pressure

- Rationale: allocate a larger number of offsprings to the best performing individuals of the population
- Selection pressure: % of individuals that will create offspring for the next generation
- High selection pressure: small % of individuals will be selected for reproduction

  → **rapid fitness improvement**, but **rapid loss of diversity**, risk of **premature convergence** to a local optimum

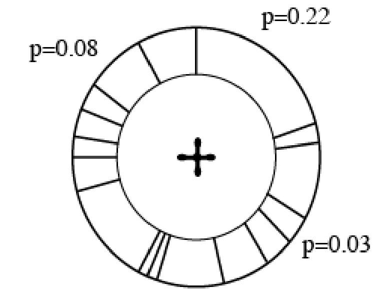
→ A **balance** is needed between selection pressure and factors that instead generate diversity (e.g. mutations, we'll see)

- You should let less fit individuals reproduce too to maintain diversity
  - They may embed traits that will become successfull later on in evolution

Proportional selection (roulette wheel)

- The probability $p(i)$ that an individual i makes an offspring is <u>proportional to its fitness relative to the overall population fitness</u> (N is the population size):

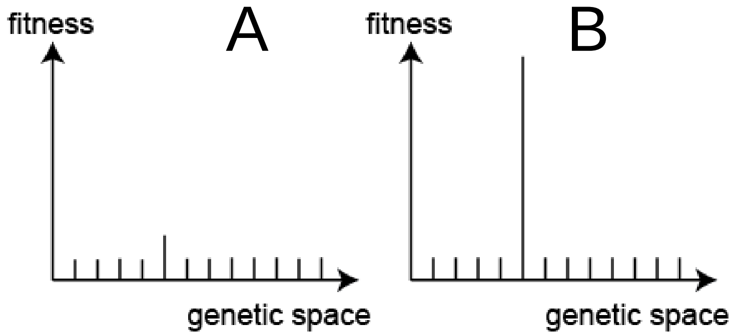
$$p(i) = \frac{f(i)}{\sum_{k=1}^{N} f(k)}$$



- Like a *<u>roulette wheel</u>* where each slot corresponds to one individual of the population, and has a width that is proportional to $p(i)$

- To build the next generation, you spin the wheel N times. Individuals can be selected several times (they are replicated, not moved)

- **Works bad when: A)** all individuals have <u>almost the same score</u> (uniform selection probability → almost **random search** → **"genetic drift"**) **B)** some individuals have <u>remarkably bigger</u> scores (**diversity loss, premature convergence**)



- **<u>A solution</u>**: ***fitness scaling***

Rank-based selection

- Sort individuals on their fitness value, from best to worst
- The place of an individual i in this sorted list is called **rank** $\boldsymbol{r(i)}$
- Instead of the fitness value (*proportionate selection*) use the rank to determine the selection probability of individuals. The *roulette wheel* approach is used.
- A possible *linear ranking* (there are many):

$$p(i) = 1 - \frac{r(i)}{N}$$

→ Solves the problems mentioned for *propotionate selection*, given that the absolute value of the fitness does not determine directly the selection probability

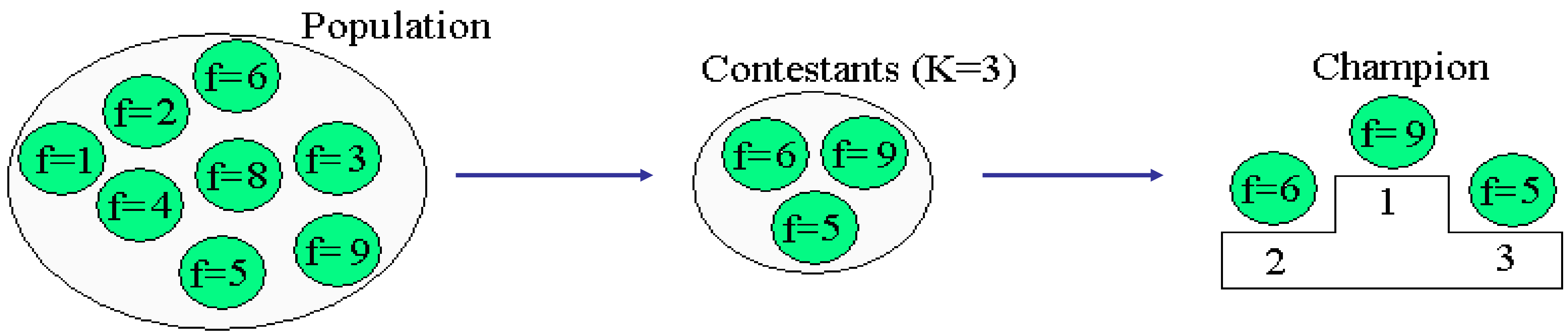
Truncated rank-based selection

- Select only the top n individuals based on their fitness
- Each of them will produce the same number of offsprings (N/n)
- E.g. $N = 100$, select top $n = 20$, $\frac{N}{n} = 5$ copies of each of the selected individuals will be used to form the next generation
- If n is not too small (would entail diversity loss → premature convergence), this method allows less fit individuals to produce the same number of offsprings as the fittest → maintains diversity

Tournament selection

- For each new offspring to be generated:
  - Randomly select a small subset of k individuals (*contestants*) of the current population
  - k is the *tournament size* parameter, the larger, the higher the selection pressure)
  - The individual that has the best fitness among the contestants wins and generates the new offspring
- Contestants can participate to multiple tournaments

→ Good trade off between selection pressure and genetic diversity



Genetic operators

- Capture the biological effect of *mutations* and *recombinations* on the genotype observed in the natural evolution
- Must match the genetic representation
- Introduce *diversity* in the population by altering individuals and combining them

Genetic operators – Crossover/recombination

- Emulates the recombination of genetic material from two parents during meiosis
- After selection, pairs of individuals are randomly formed
- And their genotypes are combined with a given probability p_c
- Crossover should allow to merge successful sub-solutions from the parents into an offspring that will hopefully perform even better

→ *There are plenty of genetic operators, and you can devise your own for your custom encoding and application. We will review some of the most commonly adopted*

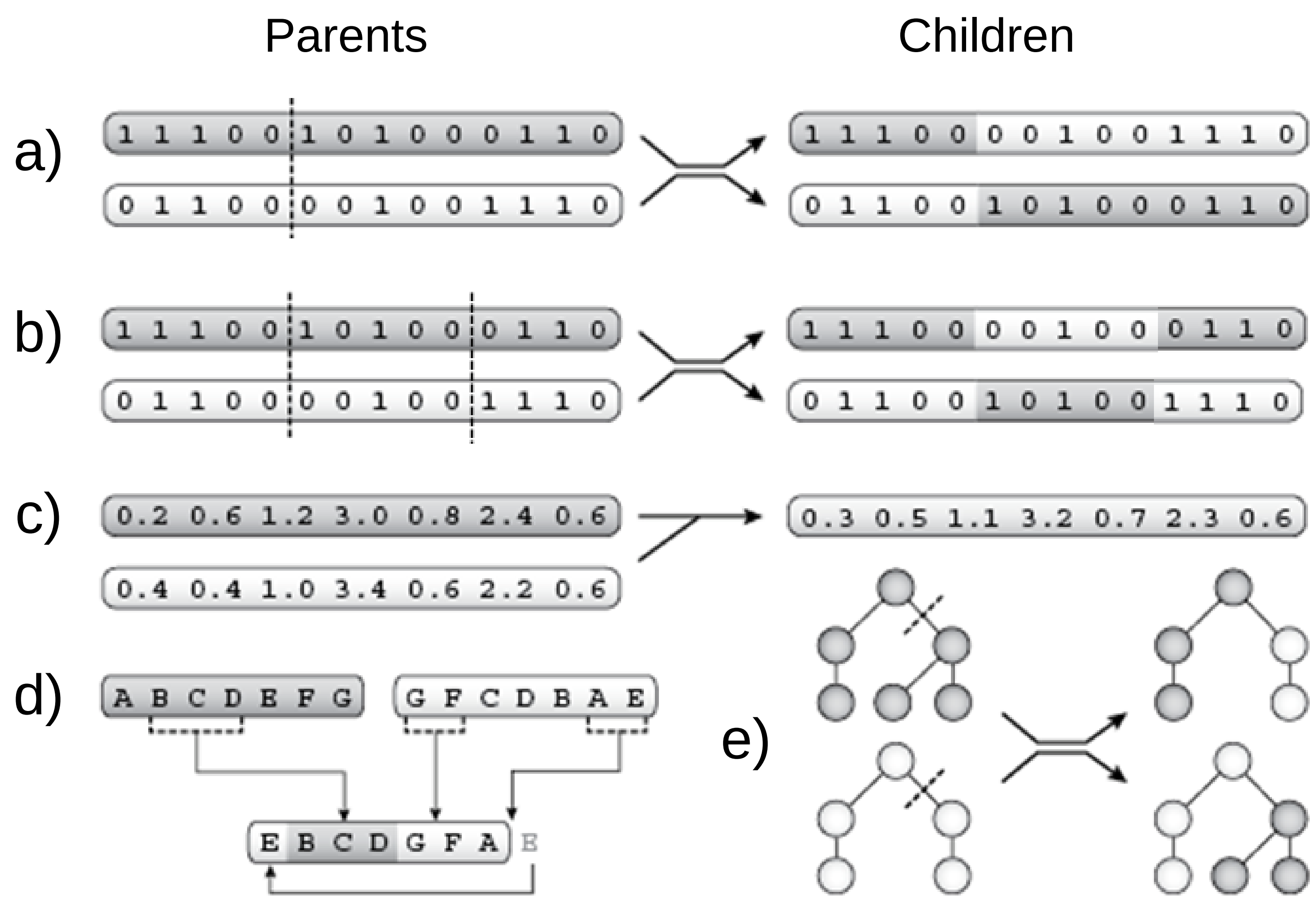
Genetic operators – Crossover/recombination

<u>Discrete/real valued encodings:</u>

a) **one-point**: randomly select a *crossover point* and swap chromosomes around that point

b) **multi-point**: as before, but selecting n *crossover points* (here $n = 2$)

c) **arithmetic**: creates a single offspring by combining the two genomes at n random positions (e.g. AND/OR for binary coded, average, or convex combination for real-coded, etc)



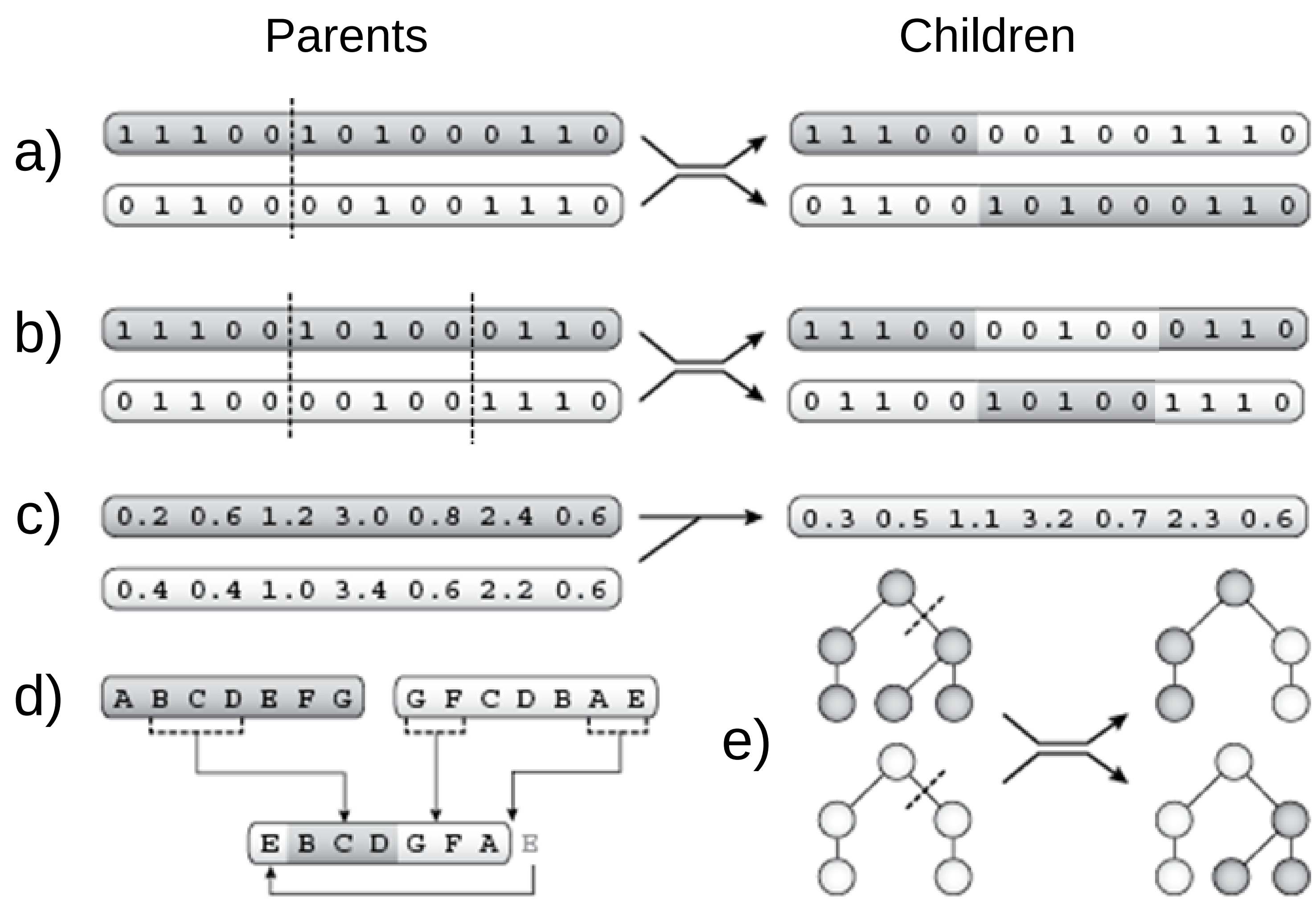
Genetic operators – Crossover/recombination

<u>Crossover for sequence encoding</u> *(all symbols must occur once and only once):*

d) Randomly copy a part of the sequence from one parent, then fill-in with remaining elements in the order in which they appear in the other parent (with wraparound, where necessary)

<u>Crossover for tree encoding:</u>

e) Randomly select a node of each parent, and exchange the two corresponding subtrees



Genetic operators – Crossover/recombination

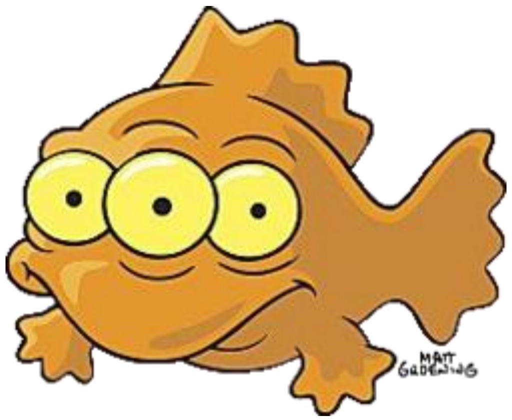
- Crossover is not a trivial operation, as it entails to isolate chunks of two different genomes and recombine them
- In some cases the effect on the fitness may be different from the expected one (i.e. the fitness of the offspring(s) is worse than the ones of the parents)

  → If this happens frequently, crossover may be implemented/tuned in such a way that it acts as a large random mutation

→ Checking the best/average fitness of offsprings obtained through crossover at each generation can help in detecting this situation

→ Solutions: revise the parameters of crossover, change crossover method

Genetic operators – Mutation



- Operates at the level of the individual
- Applies small *random* modifications of the genotype
- Allows evolution to explore *variations* with respect to the current solutions
- **Mutations are useful to**:
  - Produce diversity
  - Escape from local optima
  - Further progress when homogeneous populations are produced, where recombinations do not help to further improve
- However, too mutations may *destroy* previously discovered solutions and make the search too randomic → Proper tuning of mutations, fitness monitoring as already mentioned for the crossover

Genetic operators – Mutation

<u>Mutation</u> = change the content of each gene with probability p_m

e.g. $p_m = 0.01$ (1%, much higher than in biology), but the actual value really depends on the effect of a genotype change on the phenotype and on the characteristics of the problem

a) <u>Binary encoding</u>: toggle bit values

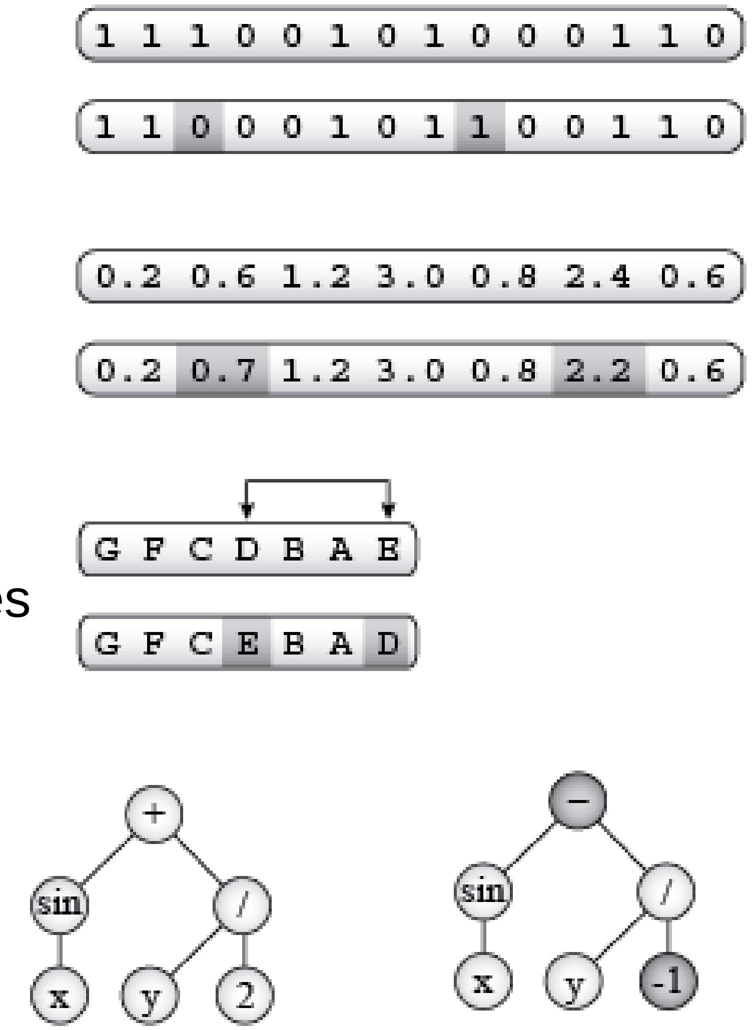
b) <u>Real-valued encoding</u>: add random noise (e.g. from a Gauss distribution $N(0,\sigma)$ → <u>most mutations are small, few are big</u>. Note that σ is an additional parameter)

a) Binary genotypes

b) Real-valued genotypes

c) Sequence genotypes

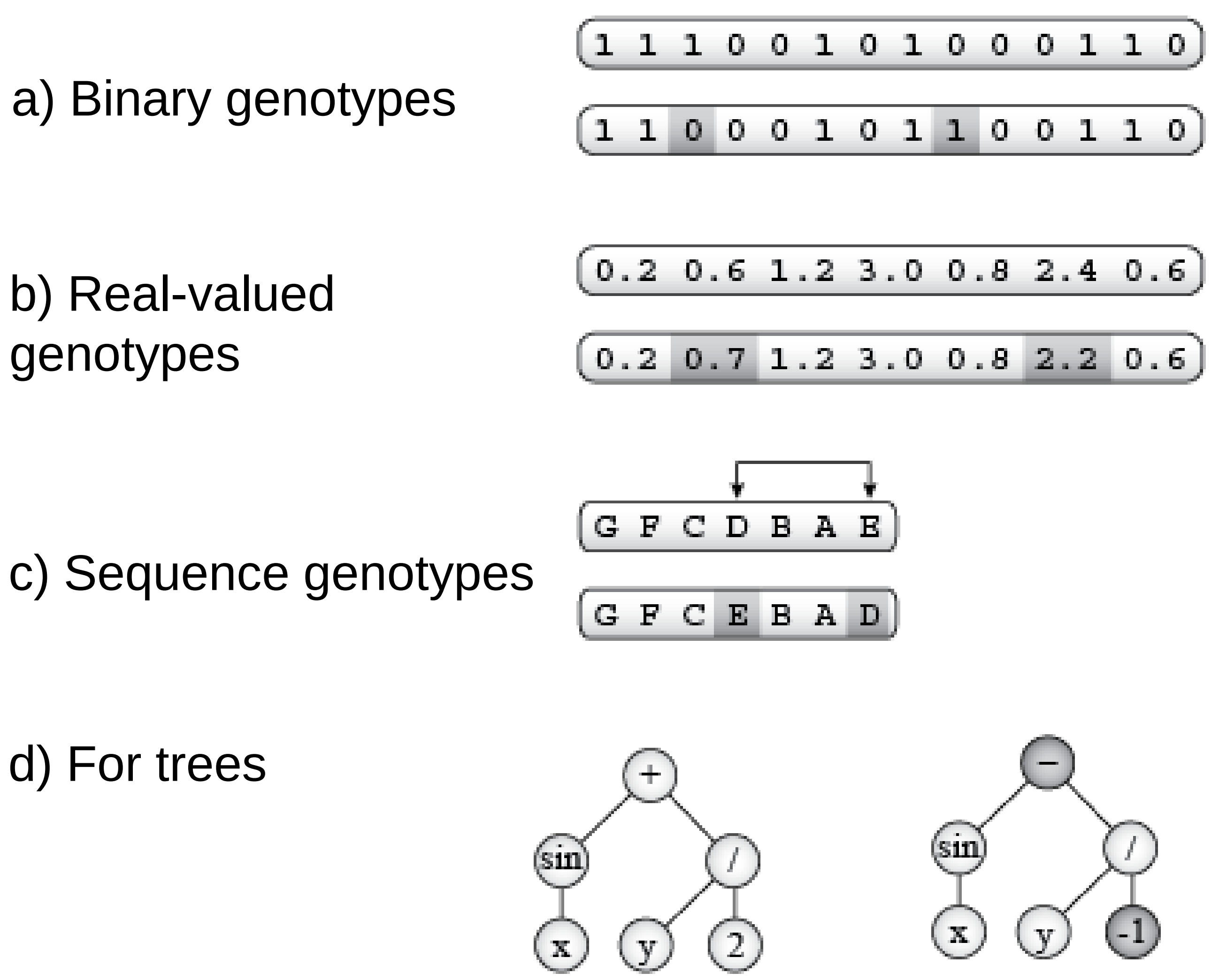
d) For trees



Genetic operators – Mutation

c) Sequence encoding: swap the contents of two randomly chosen genes

d) Tree-based encoding: change the value of a node with another from the same set (functions set/terminals set) with the same number of leaves → tree-structure unchanged



Replacement strategies

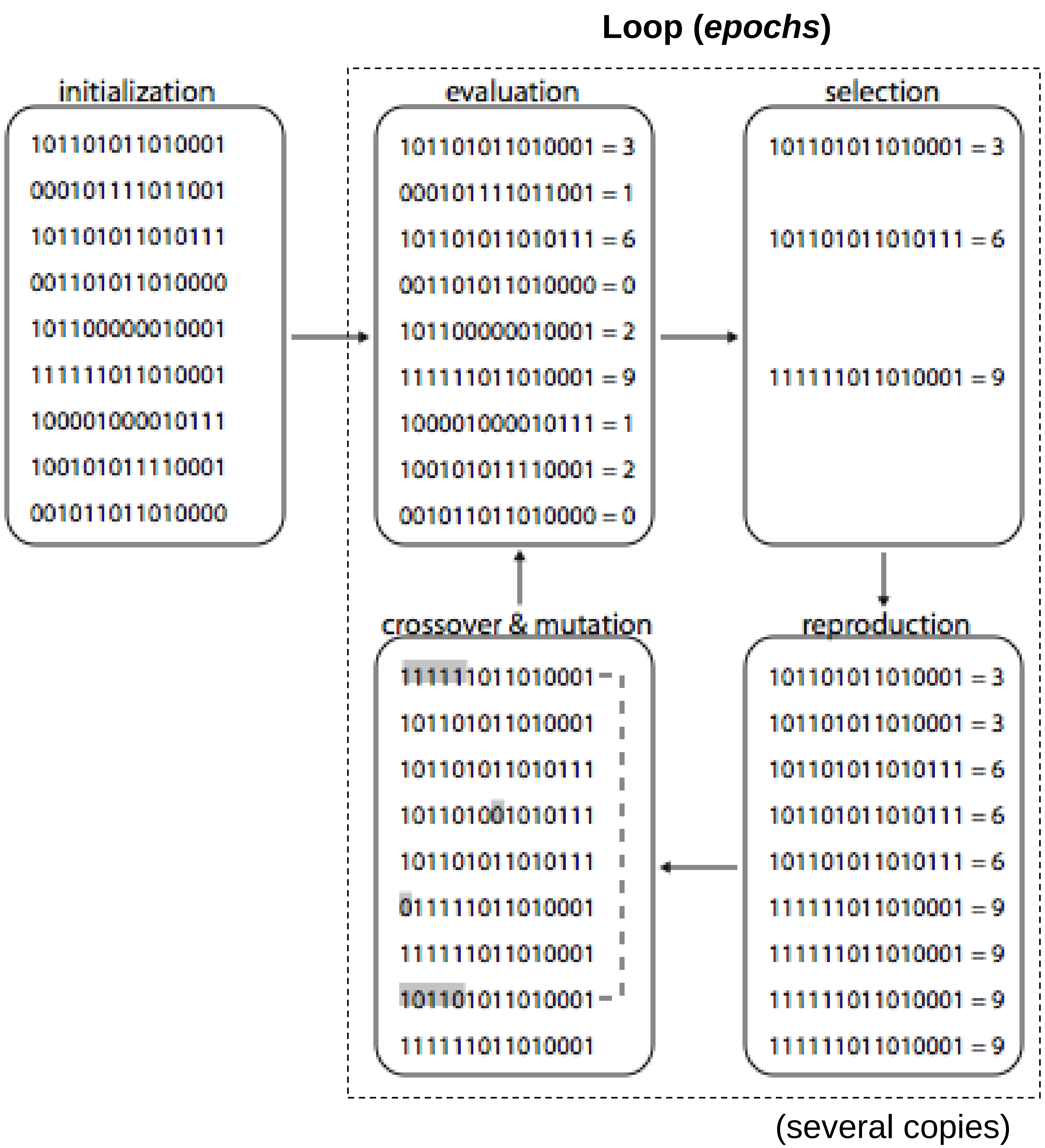
What to do once the new population is produced?

- *Generational replacement*: the new population completely replaces the old one (most adopted)

→ Good individuals can sometimes get lost

- *Elitism:* the best n *individuals of the current population are propagated, unchanged, to the new one.*

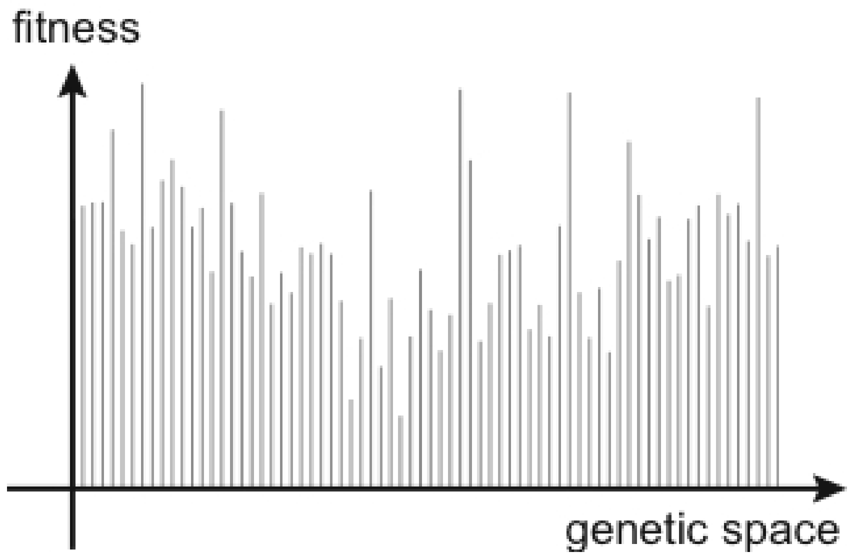
Example



Loop (*epochs*)
initialization
101101011010001
000101111011001
101101011010111
001101011010000
101100000010001
111111011010001
100001000010111
100101011110001
001011011010000
evaluation
101101011010001 = 3
000101111011001 = 1
101101011010111 = 6
001101011010000 = 0
101100000010001 = 2
111111011010001 = 9
100001000010111 = 1
100101011110001 = 2
001011011010000 = 0
selection
101101011010001 = 3
101101011010111 = 6
111111011010001 = 9
reproduction
101101011010001 = 3
101101011010001 = 3
101101011010111 = 6
101101011010111 = 6
101101011010111 = 6
111111011010001 = 9
111111011010001 = 9
111111011010001 = 9
111111011010001 = 9
crossover & mutation
111111011010001
101101011010001
101101011010111
101101001010111
101101011010111
011111011010001
111111011010001
101101011010001
111111011010001
(several copies)

The fitness landscape

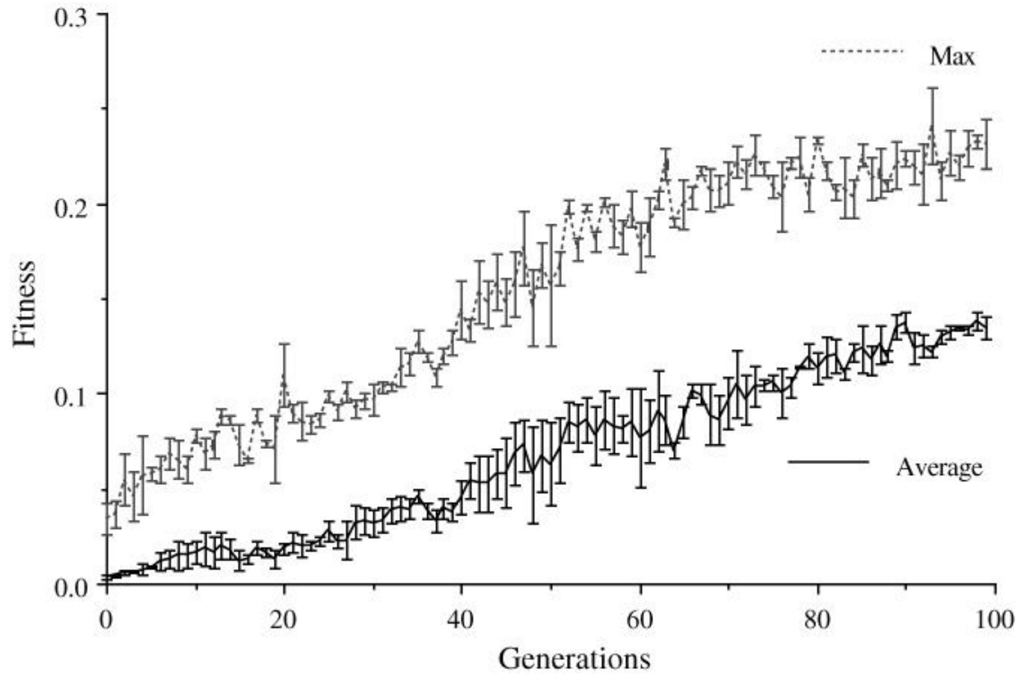
- (unknown) multidimensional surface associating a fitness value to each possible genome (e.g. $\Re^{GenomeLength+1}$)
- Sampled during evolution ("navigation" is guided by the genetic operators: not a straightforward path through the landscape)



- Rough sampling analysis can help to better understand the problem
- E.g. estimating ruggedness of real landscape:
  - Sample random genotypes: if fitness ≈ uniform, use large populations
  - Explore surroundings of an individual by applying genetic operators in sequence for fixed number of steps: the larger the fitness variation observed the easier should be evolution

The fitness graph

- How to show the performance of an evolutionary algorithm across generations: fitness graph
- Usually average and best fitness at each generations are plotted, along with standard deviation across multiple runs
- Given the random components in these algorithms, **several executions** are usually necessary, with different initializations (especially for quantitative comparisons)
- Plateau: have we reached the global optimum or are we stuck in a local one (premature convergence)?

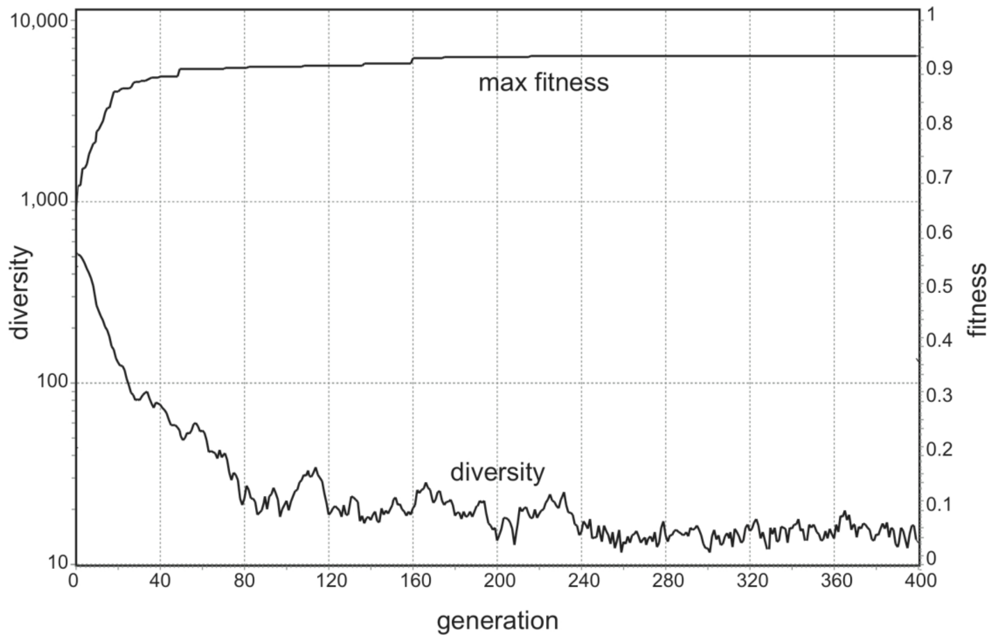


Population diversity

- Further insights can be gained by analysing the *diversity* of the population
- A possible measures of diversity (for *direct encoding, fixed length*):
  - *"All-possible-pairs" diversity*:

$$D_a(P) = \sum_{i,j \in P} d(g_i, g_j)$$

→ Sum of Euclidean or Hamming* distances ($d(.,.)$) among all genomes



* Number of differing elements in corresponding positions. For binary strings: $d_h(a,b) = \sum a \oplus b$. E.g. $d_h(000, 101) = 2$

Stagnation, diversity, *neutral paths*

- Stagnation = evolutionary algorithm does not improve fitness for a long time

In case of stagnation:

- Low diversity → crossover won't help much → we can hope only in mutations → may take long to further improve
- It may also happen (e.g. in case of redundant encodings) that evolution took a *neutral path,* a phase in which despite genetic alterations of the population, the overall fitness does not change
- These neutral paths can sometimes result in rapid progresses after a long stasis (waiting a bit may be rewarding)

Types of evolutionary algorithms

- **Genetic Algorithms** (GA) - Holland, 1975
  Binary genotypes, crossover and mutation
- **Genetic Programming** (GP) - Koza, 1992
  Tree-based genotypes, crossover and mutations
- **Evolutionary Programming** (EP) - Fogel etal., 1966
  Real-valued genotypes, mutations, tournaments, gradual pop. replacement
- **Evolutionary Strategies** (ES) - Rechenberg, 1973
  As EP + mutation range encoded in genotype of individual
- **Island Models** – Whitley et al., 1998
  Parallel evolving populations with rare migration of individuals
- **Steady-State Evolution** – Whitley et al., 1988
  Gradual replacement: Best individuals replace worst individuals
- …

Some practical pros and cons

Pros:

- Evolutionary algorithms can work where other optimization techniques cannot (e.g. discontinuous, noisy fitness functions) → robust
- Can be easily extended to deal with multi-objective, constrained problems
- Inherently parallel structure → can be distributed / parallelized
- It is easy to incorporate knowledge into them, e.g. refine previous solutions

Cons:

- No guarantees regarding the success and/or the time to get a solution
- Weak theoretical basis
- Parameters tuning is needed
- Often computationally expensive

Some applications

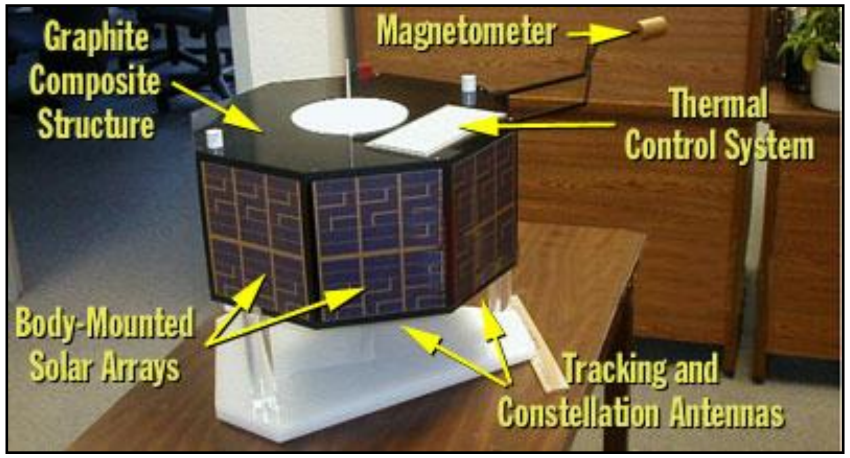
From «How The Body Shapes The Way We Think – A new view of Intelligence» (R. Pfeifer & J. Bongard)

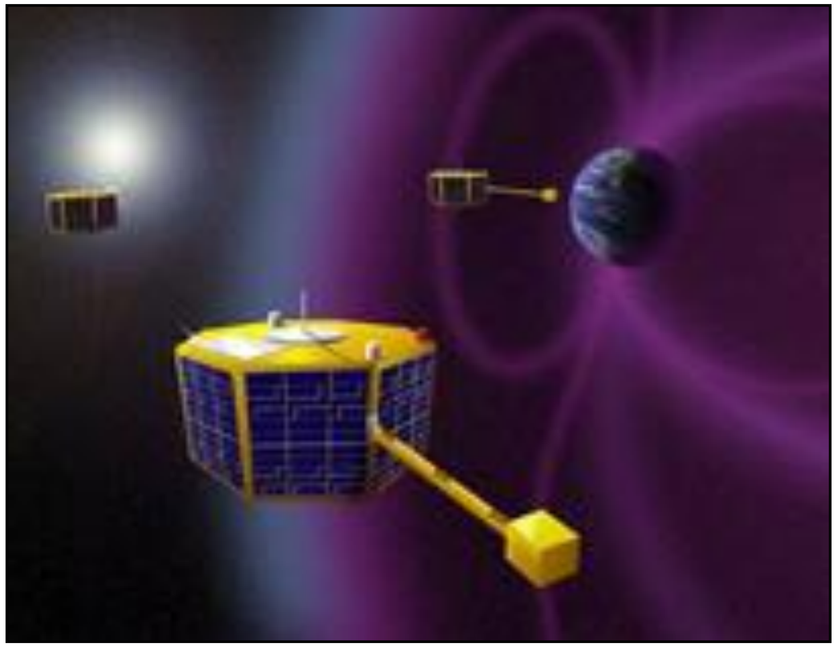
NASA's Antenna

Human-competitive design of an antenna for nanosatellites, NASA [Lohn, Hornby, Linden, 2004]

- Designing an efficient antenna meeting several quality requirements (gain, sizes, operational frequencies, …) is very challenging for humans
- NASA automated its design by using evolutionary techniques

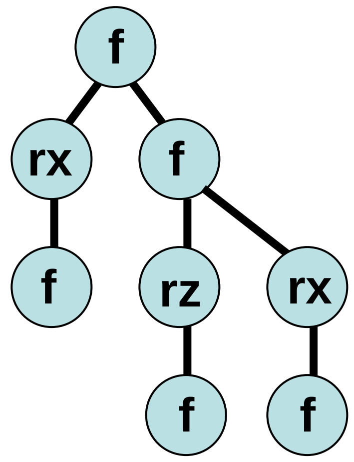
- Wiki page
- Paper

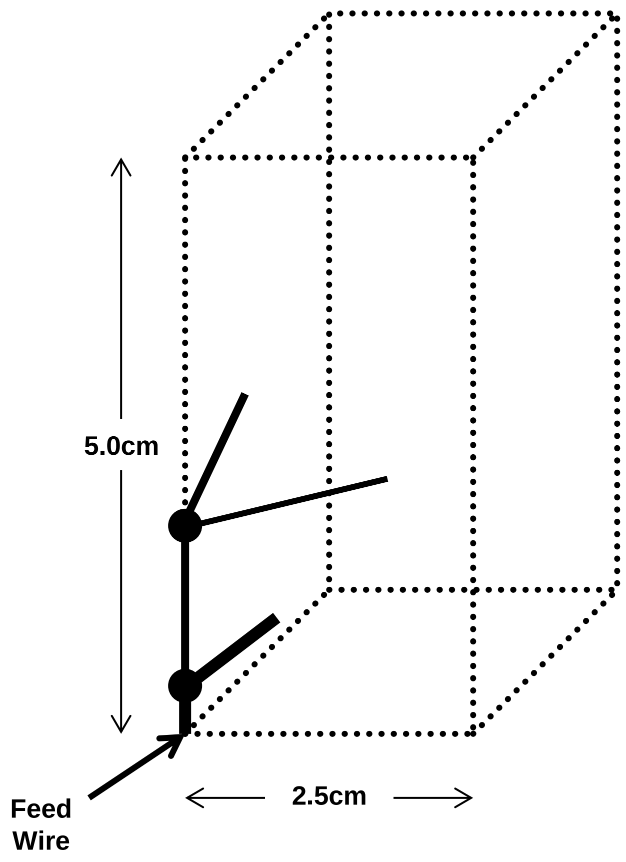




NASA's Antenna

- Tree-based encoding, instructions to "grow" an antenna
- Function set:
  - f=forward(length)
  - rx/y/z(angle)
  - Terminals: length, angles
- Technical specs tested in simulation
- Best designs were built and worked well also in the real world





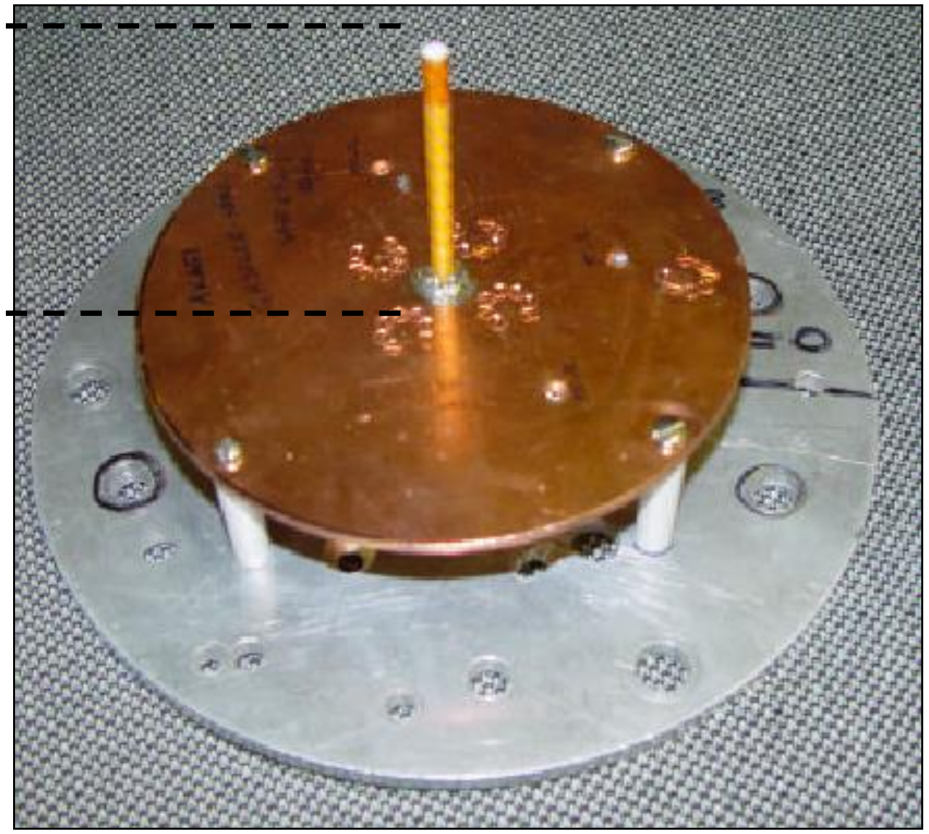
NASA's Antenna

"Weird" designs, <u>considerably smaller</u> and exhibiting <u>far superior performances</u> with respect to human devised ones

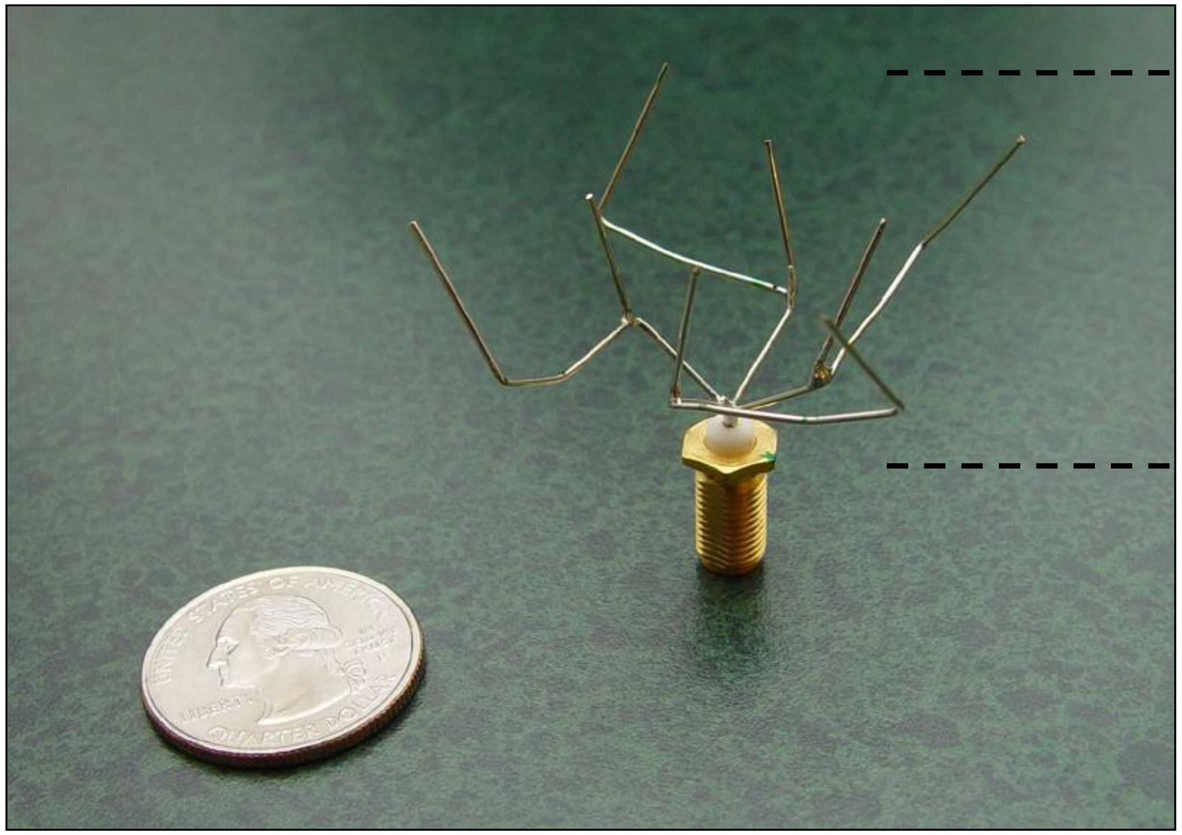
→ <u>Launched on board of the ST-5 satellite in 2006</u>

Human

~12 cm



Evolved

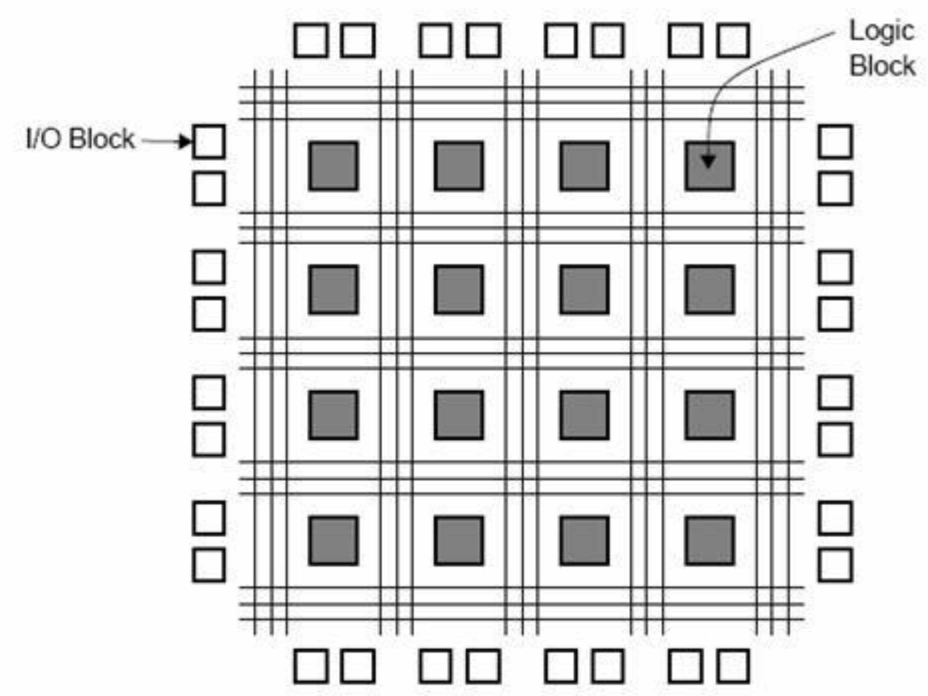


~ 5 cm

Evolvable hardware

Adrian Thompson, Sussex University, 1996

- Experiments on *evolvable hardware* (evolutionary algorithms finding circuits configurations for FPGA)
- <u>Goal</u>: evolve a circuit to distinguish between a low tone from a high tone
- No simulation, <u>evolution in the real world</u>
- <u>An effective circuit was evolved, that worked properly but…</u>
- …once re-created on a custom circuit considering only the FPGA component effectively connected in the design…
  → <u>Did not work anymore!</u>



Evolvable hardware

- It was found out that the original evolved circuit was exploiting weak electromagnetic interactions with the disconnected FPGA components

- The solution devised by evolution broke human-imposed modular design, exploiting to its benefit phenomena of the ecological niche that are usually regarded as undesired

Evolvable hardware

Another experiment in Sussex, by Jon Bird and Paul Layzell

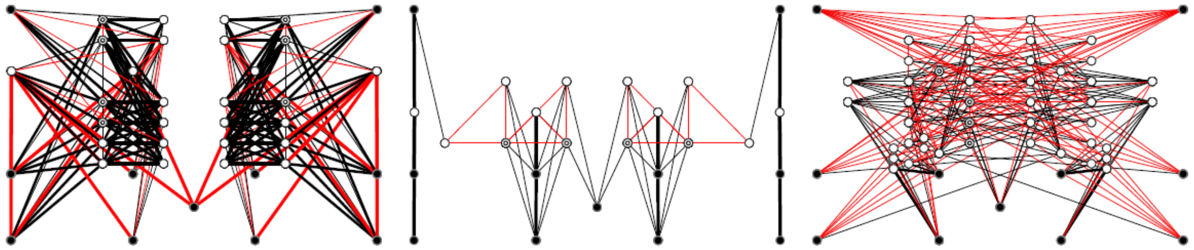
- Goal: evolve a circuit producing an oscillatory signal without having an internal clock

- Evolved solution: instead of an oscillator, something like a *radio receiver* was evolved from scratch

→ The oscillating signal produced by the circuit was indeed coming from electromagnetic interferences caused by a computer nearby: the evolved circuit was "stealing" the clock of that computer!

- Another example of how artificial evolution finds clever way to exploit the ecological niche

→ Even a new sensor modality was evolved from scratch!



Neuroevolution

A biologically inspired path to Artificial Intelligence

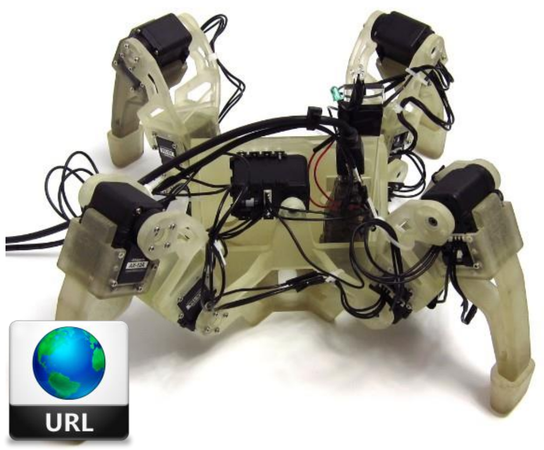
Neuroevolution

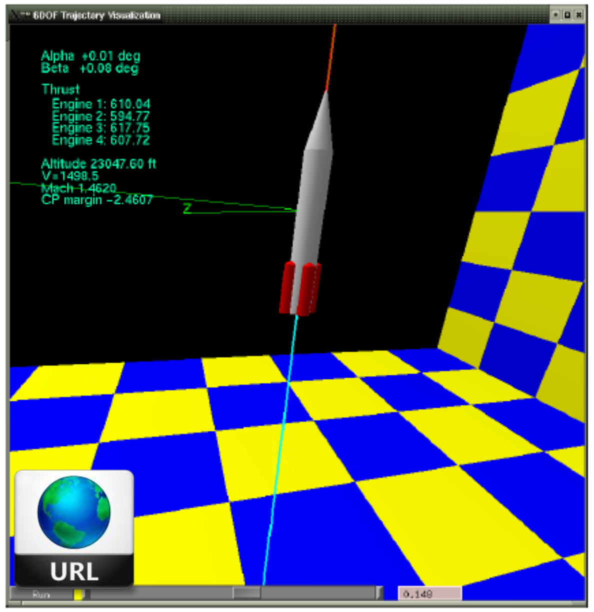
- Use of evolutionary algorithms to *__construct Neural Networks__*

→ __Evolution of *cognitive architectures*__

- __Proved to be effective to:__
  - Evolve cognitively multimodal cognitive behaviors
  - Evolve large scale brain-like structures
  - Evolve effective control policies (e.g. locomotion, guidance, stabilization, …)
  - Evolve human-like game playing in a variety of videogames

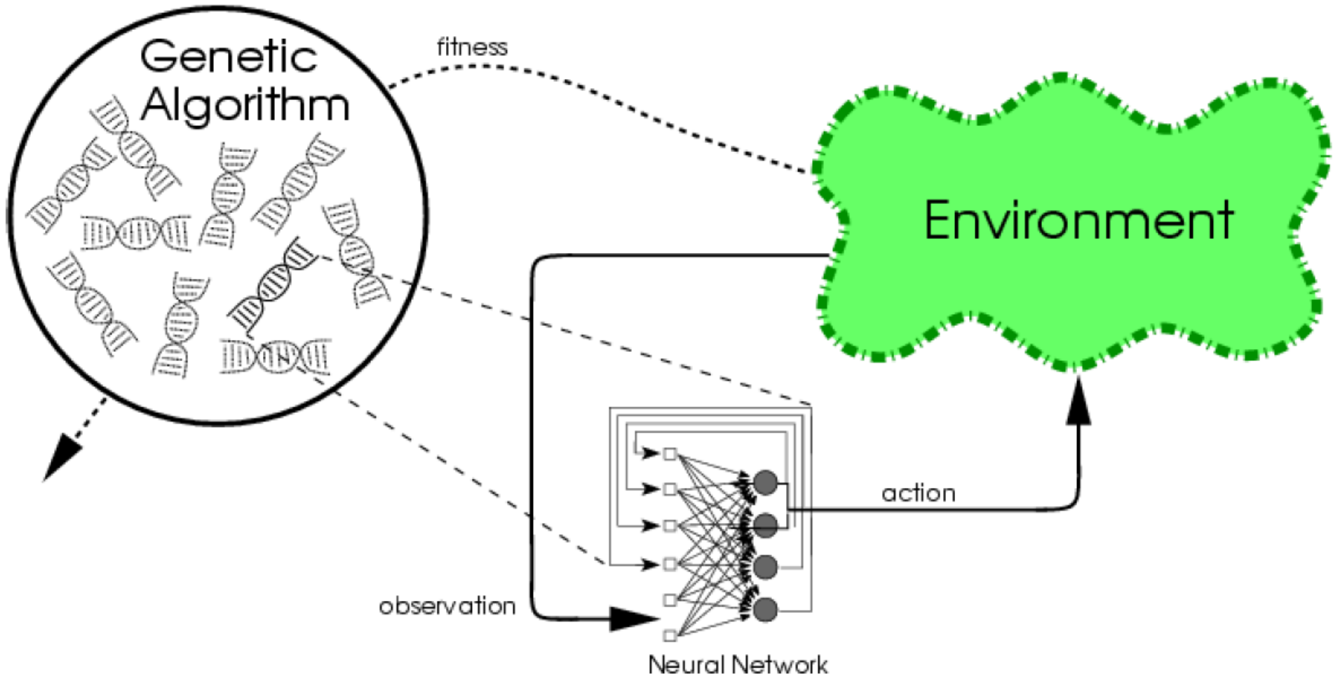
→ __Interesting from several perspectives__: AI, control, evolutionary robotics, ALIFE, …





Conventional NeuroEvolution (CNE)

1. Fix the structure of the NN (usually fully connected)
2. Concatenate synaptic weights and biases into a genome (random init)
3. Use an evolutionary algorithm to evolve the network with respect to a given task
4. Fitness: evaluation of network's performance



→ This can be seen as a way to train neural networks in an unsupervised setting

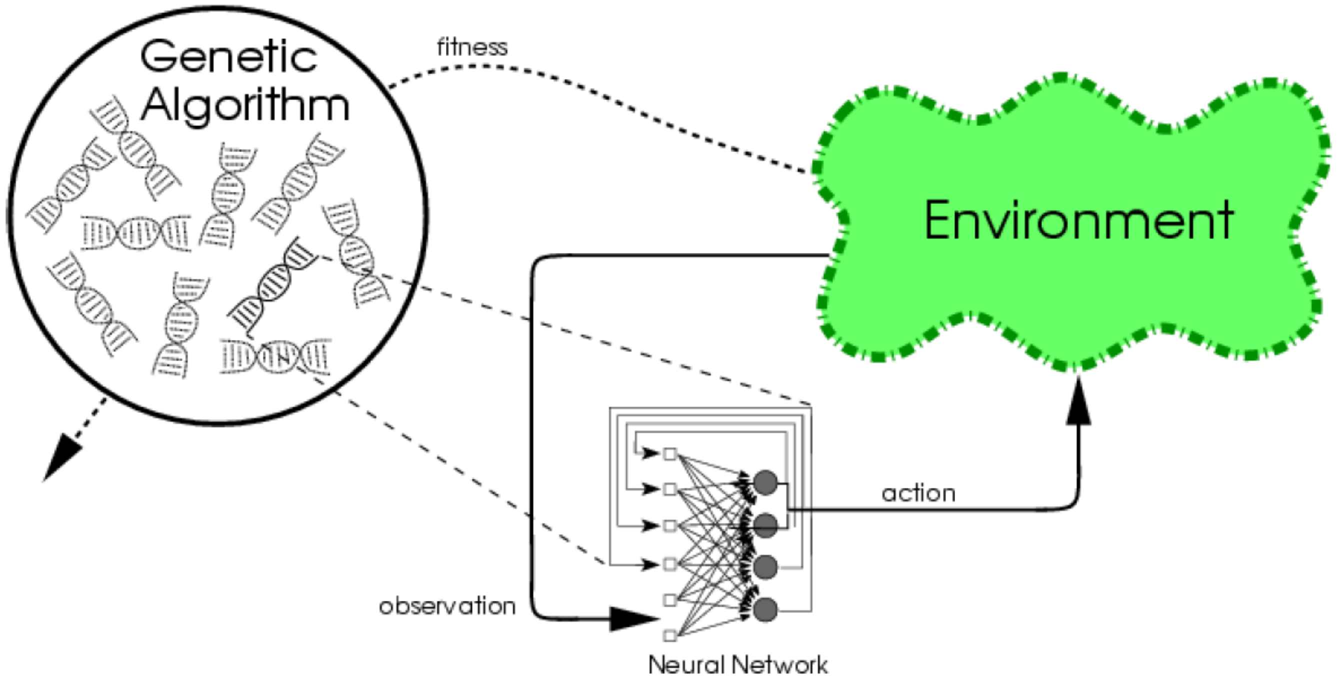
Conventional NeuroEvolution (CNE)

Pros:

- Easy to implement
- Effective in many scenarios

Cons:

- Requires to arbitrarily choose network's size and topology
- In general not very scalable (number of parameters easily reaches thousands)
- Can easily converge to local optima



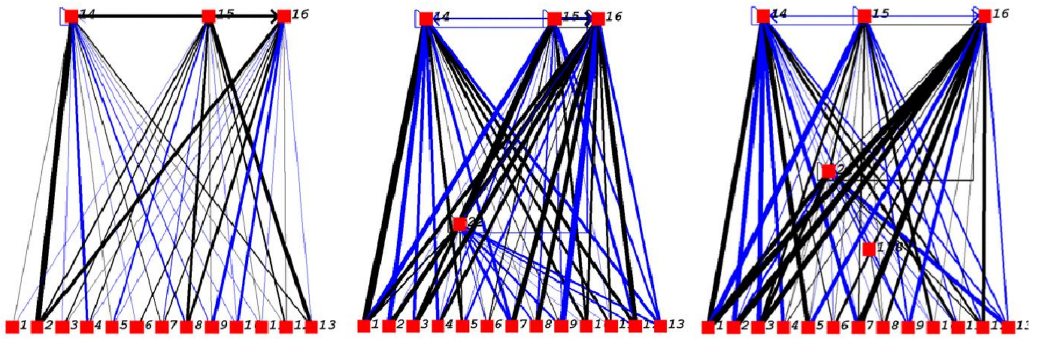
Extensions

- Evolve both the topology and the weights → Better performances
- Evolve the type of activation functions on the nodes
- Evolve plastic networks (or *evolution of learning*) [1, 2, 3]
  - Evolution provides an initial network that then adapts online during the lifetime of the agent through environmental feedback
  - E.g. local Hebbian learning rules ("*fire together, wire together*"). New learning rules can also be evolved!
  - It was postulated for natural evolution (Baldwin effect) and showed in artificial evolution that learning can indeed affect positively evolution

NEAT - NeuroEvolution of Augmenting Topologies [Stanley et al. 2002]

- Features *complexification* (biologically plausible, useful to improve performances)
- **Of networks**
  - Starts with simple networks → Mutations add new nodes and new links when necessary to progress
- **Of behaviors**
  - New networks elaborates on earlier behaviors

→ Helps in reducing the search space



NEAT - NeuroEvolution of Augmenting Topologies [Stanley et al. 2002]

- Additional complexity is retained only if provides a competitive advantage
- Genes are *marked* with a *global innovation number* (chronological order of appearance)
- This allows to measure the "age" of each individual and to identify homogeneous sub-populations
- To protect recent topological innovations (that may be promising later on but still have low fitness at a certain point in time), competition with older solutions (already mature) is avoided
- Diversity maintenance: competition among very different genomes is avoided

HyperNEAT – Hypercube based NEAT [Stanley 2009]

- Goal: To improve scalability and to promote regular structures in the NNs evolved by NEAT
- NEAT is coupled with a powerful indirect encoding method called ***Compositional Pattern Producing Networks (CPPNs)***

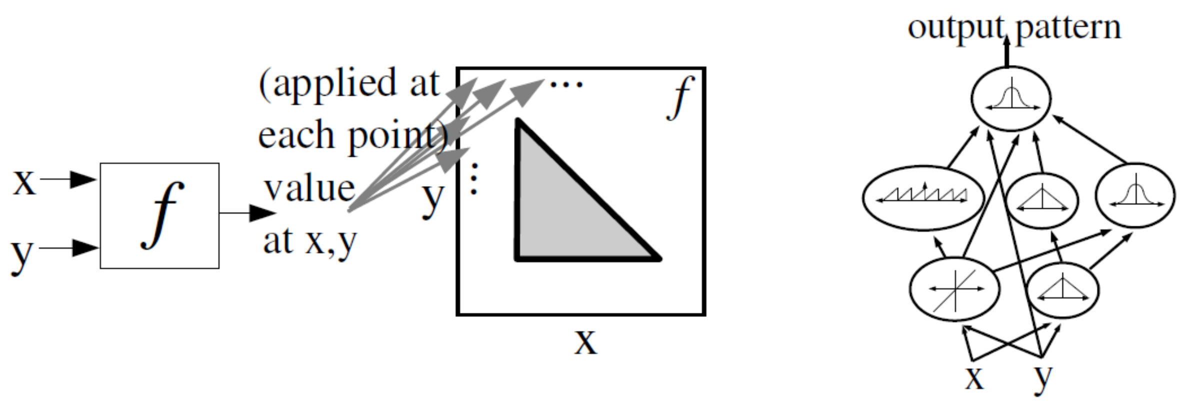
→ HyperNEAT actually evolves CPPNs then used to define NNs

- CPPN: designed to represent spatial patterns with regularities such as symmetry, repetition, and repetition with variation
- Thanks to this encoding:
  - HyperNEAT is able to produce very large networks in an efficient way (millions of connections)
  - Evolved networks present regularities also observed in biological brains

HyperNEAT – Hypercube based NEAT [Stanley 2009]

But what are CPPN?

- Structurally similar to NNs, but mimic a different phenomena: an abstraction of development
- They produce *spatial* patterns by composing basic functions (e.g. sin, cos, gaussians, etc.)
- This composition can produce complex patterns with several regularities



(a) Mapping (b) Composition

HyperNEAT – Hypercube based NEAT [Stanley 2009]

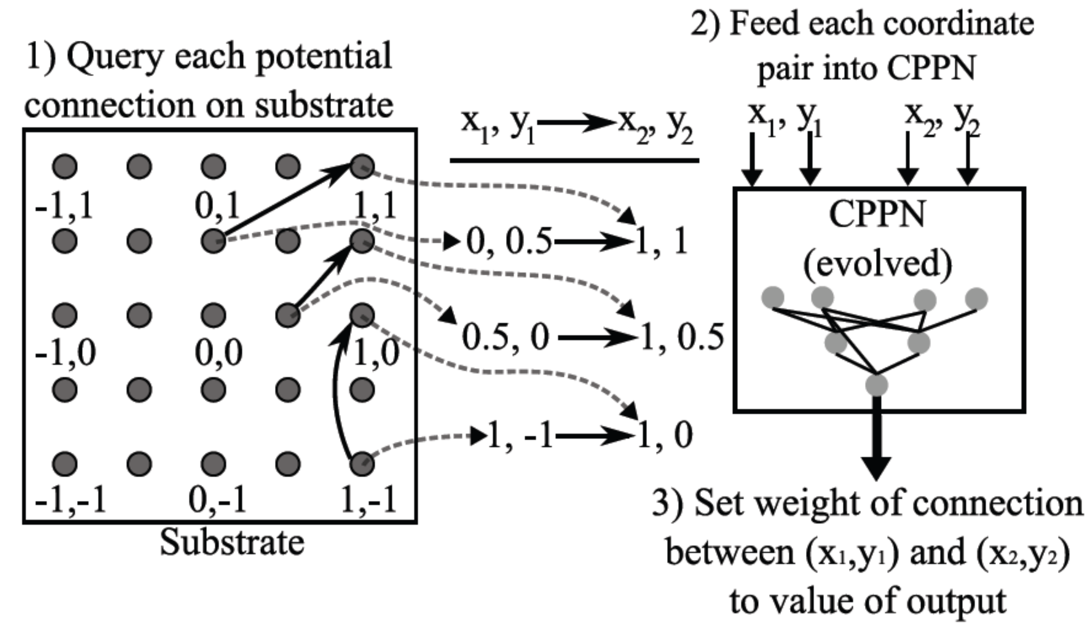
- Remember PicBreeder and EndlessForms?
- Those systems actually evolve CPPNs in an interactive way
- To produce an image or an object, the system asks the CPPN the color of each (x,y) pixel, or the presence of each (x,y,z) voxel

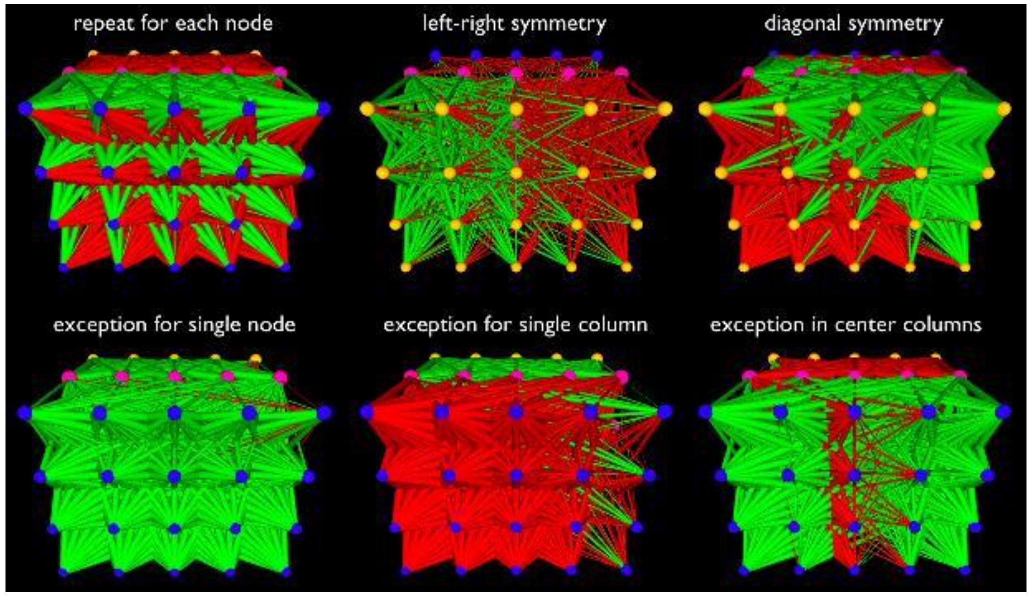
→ You can observe with your eyes the complex spatial patterns those networks are able to encode

HyperNEAT – Hypercube based NEAT [Stanley 2009]

- In HyperNEAT <u>CPPNs are evolved that represent connectivity patterns of hidden nodes of a NN</u>
- CPPNs take as input the coordinates of two points describing each connection
- As a consequence of using CPPN, in HyperNEAT the connectivity of the NN is a <u>function of input's geometry</u>

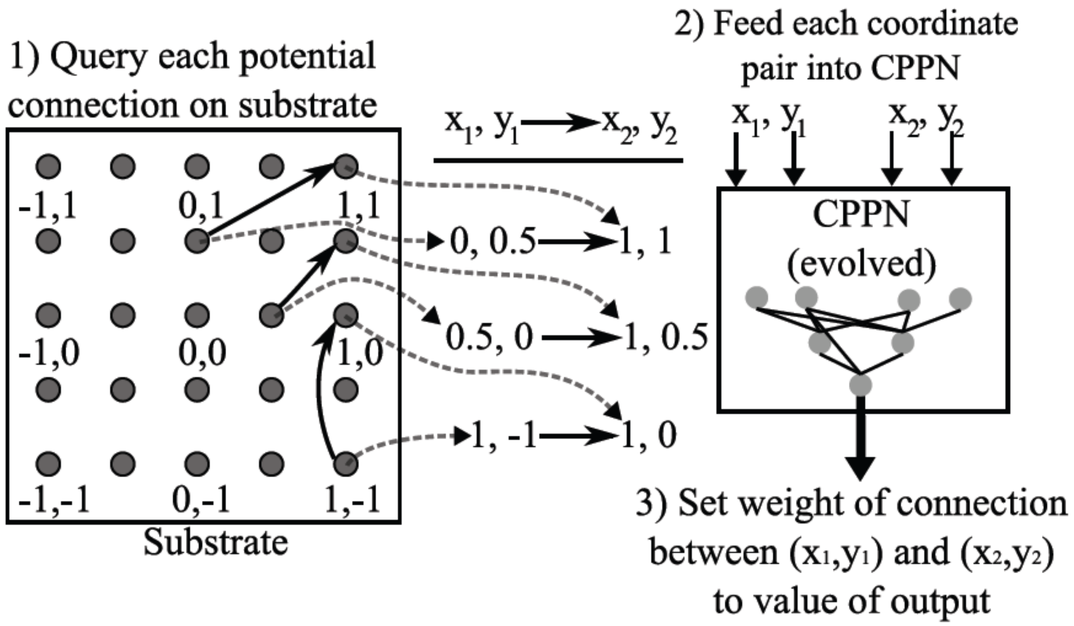
→ HyperNEAT represents and exploits the <u>geometry *(substrate)* of the inputs</u> to enhance learning

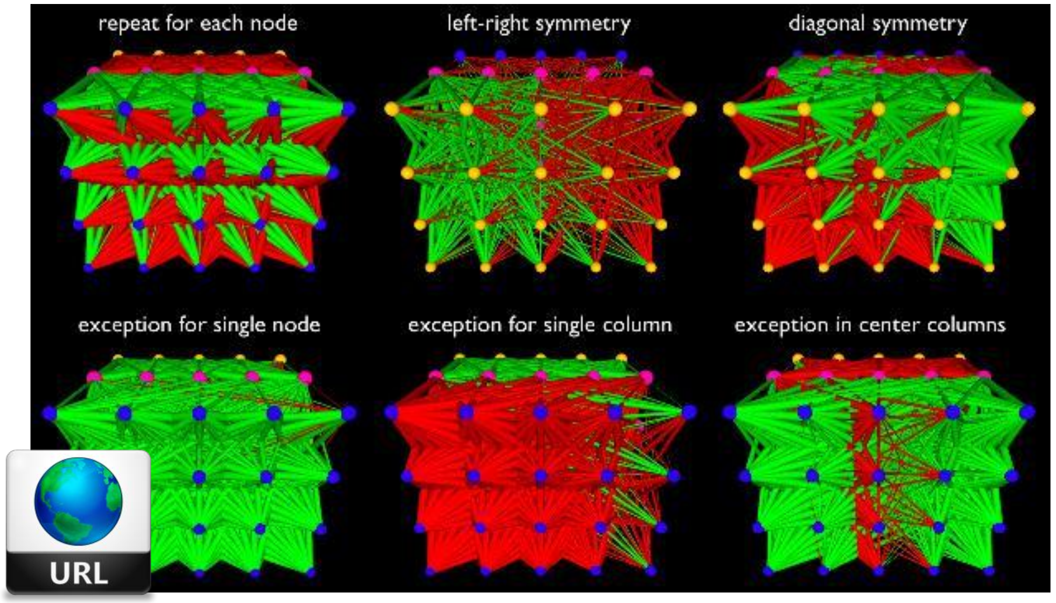




HyperNEAT – Hypercube based NEAT [Stanley 2009]

- The topological arrangement of input nodes is fixed in HyperNEAT (has to be chosen by the user), and it is called *substrate*
- Top figure: square substrate
- Bottom figure: cube substrate
- Input and output nodes are selected from the substrate
- Different substrates are better suited for different problems
- Other extensions of HyperNEAT also evolve the structure of the substrate: evolvable-substrate HyperNEAT (ES-HyperNEAT)





Neuroevolution

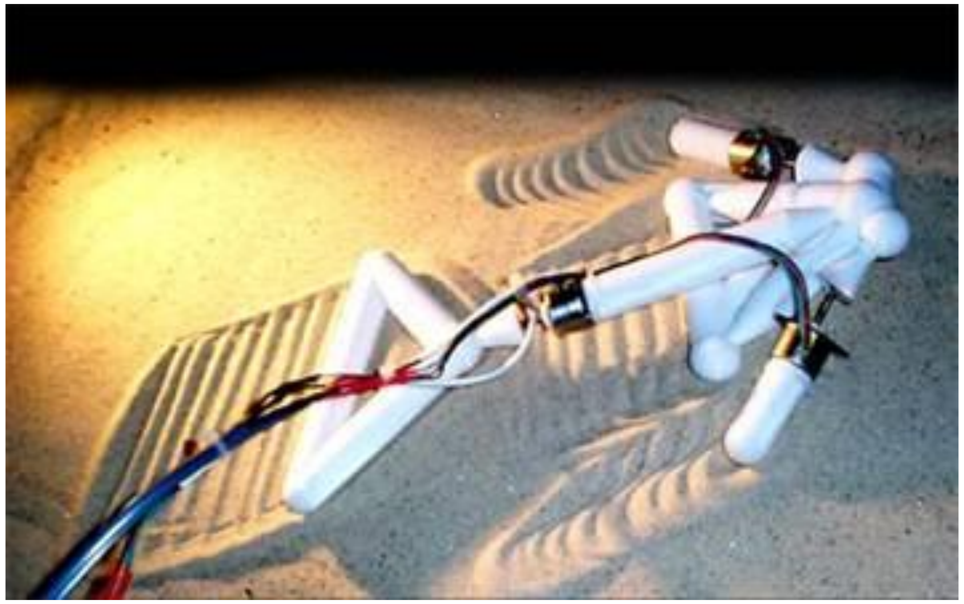
- Neuroevolution techniques are powerful
- Promising for artificial intelligence, evolution of <u>general cognitive behaviors</u> (general, cognitively scalable, does not require much human intervention)
- Sound, bottom-up, biologically inspired approach to AI

Neuroevolution

- Also, more and more a tool for fields such as *artificial life, computational biology, etc.*

- With targeted experiments, is helping in answering questions such as:
  - How specific behaviors evolve? Under which conditions? Foraging, pursuit and evasion, hunting and herding, collaboration, communication → e.g. *competitive coevolution*
  - How modularity evolved in biological body/brains?
  - How development and evolution interact?
- By analyzing evolved neural circuits, insights can be gained regarding biological networks functions

Evolutionary Robotics



Video

Evolutionary robotics

Evolutionary algorithms are used to evolve:

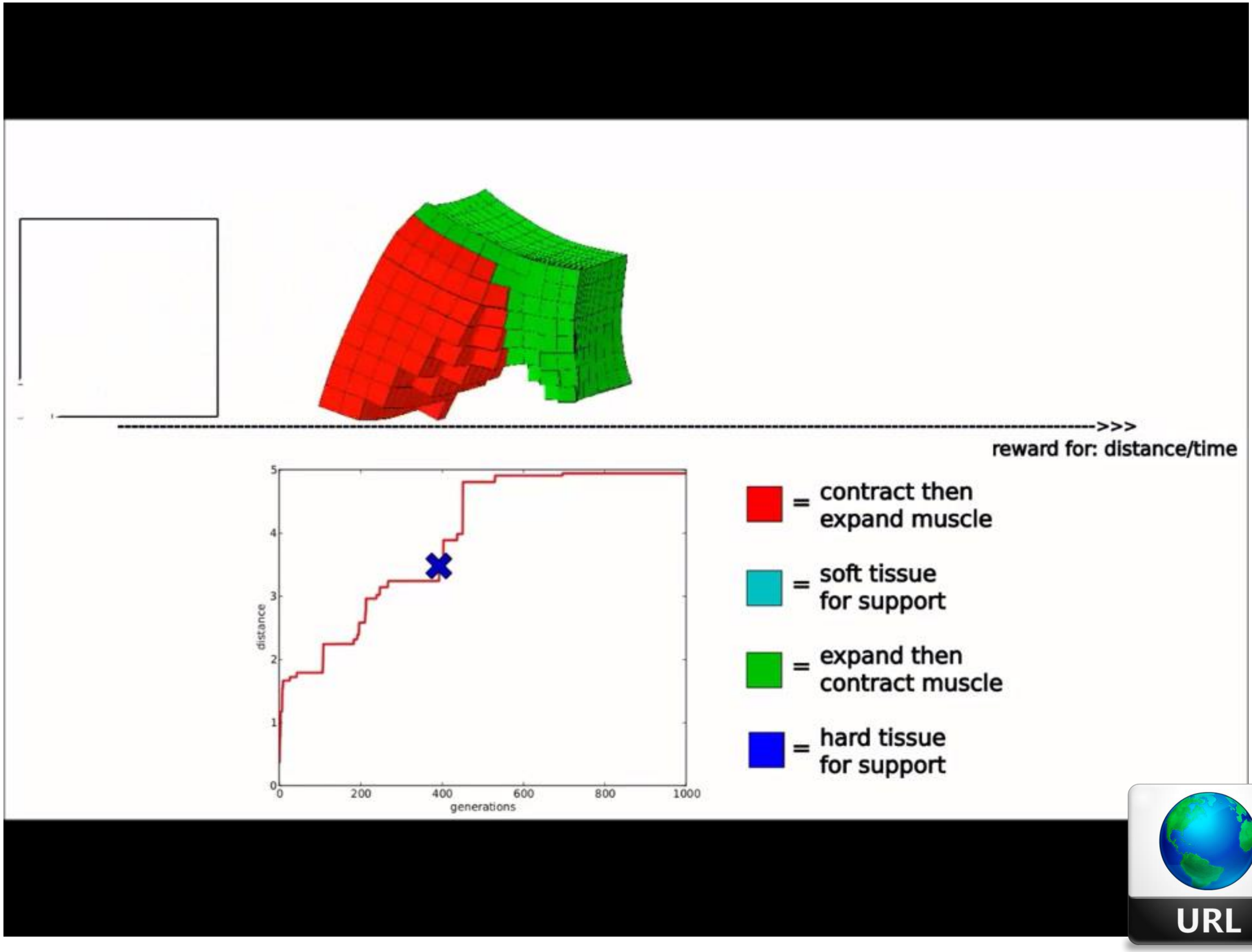
- Brains (typically neural networks)
- Body (some parameters, or the whole structure)
- Both at the same time (*brain-body co-evolution*)

As for evolving bodies:

- Powerful methods to encode the body of a robot in a generative way exist (*artificial embryogeny*) – [a review paper]

Brain-body coevolution – [K Sims, 1994] [1][2]

Evolving Soft Robots – [N Cheney et al., 2013]



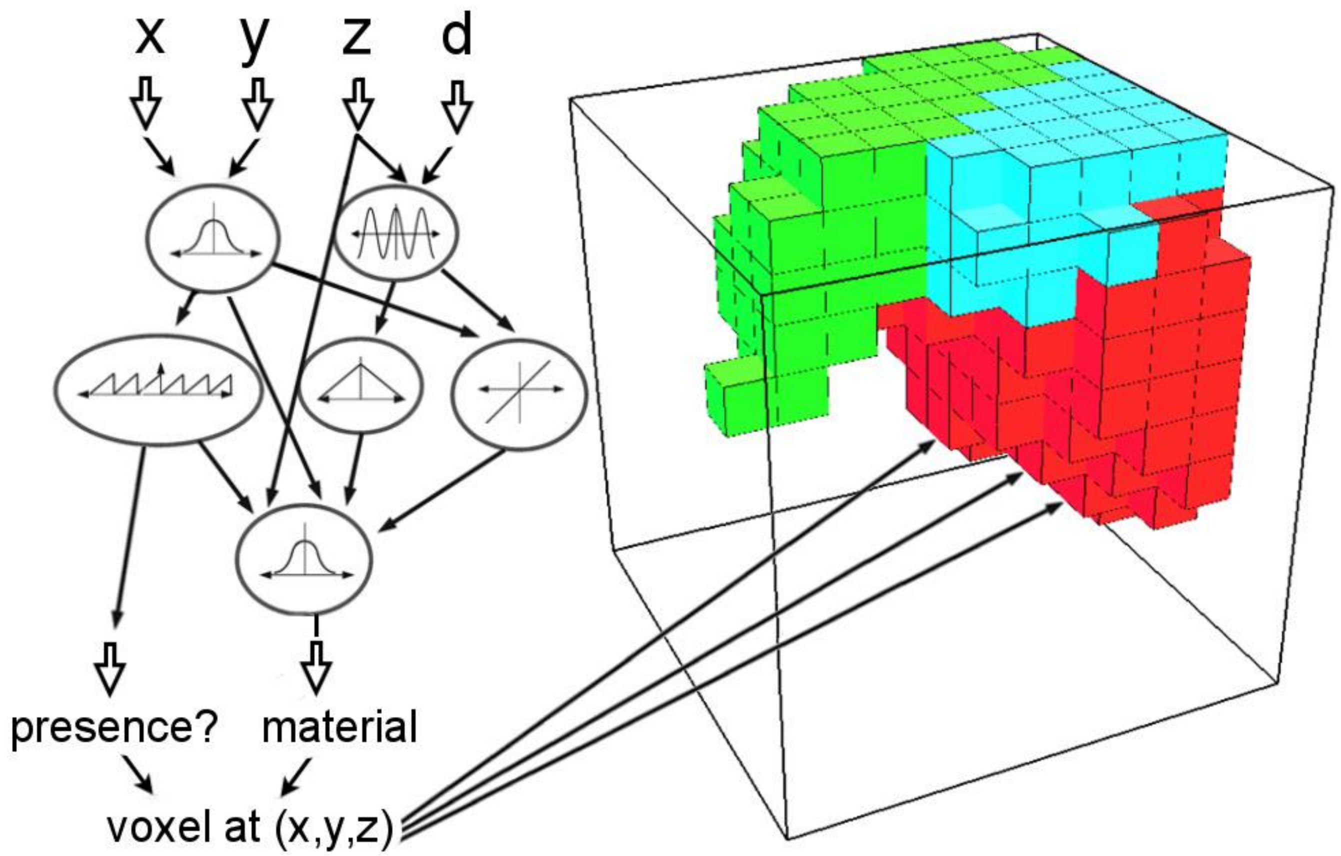
- Virtual creatures evolved at a finer resolution (in terms of morphology and actuation)
- Use of several soft-materials and actuators enriches evolved behaviors

Evolving Soft Robots – [N Cheney et al., 2013]

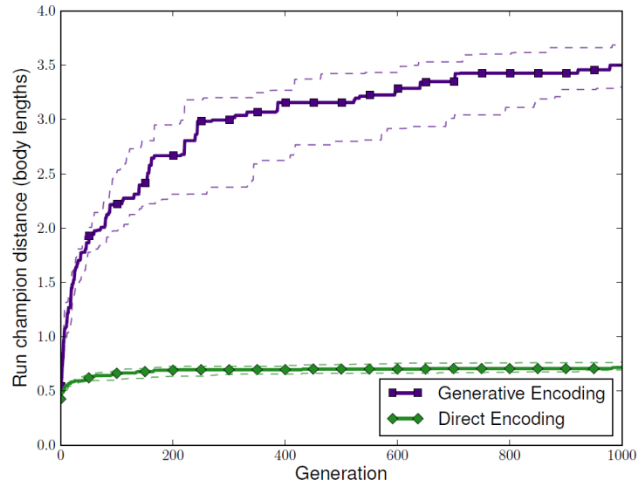
A *Compositional Pattern-Producing Network* is evolved through NEAT (a neuroevolution algorithm often used to evolve neural networks), and sampled to define the morphology

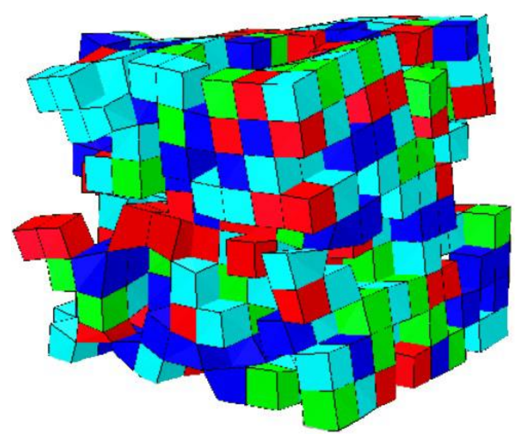
Input: x,y,z position of the voxel, distance d from the center

Output: voxel present, material type



Generative encoding produced *more functional* an *far more regular* morphologies if compared with direct encoding





The *reality gap* / transfer problem

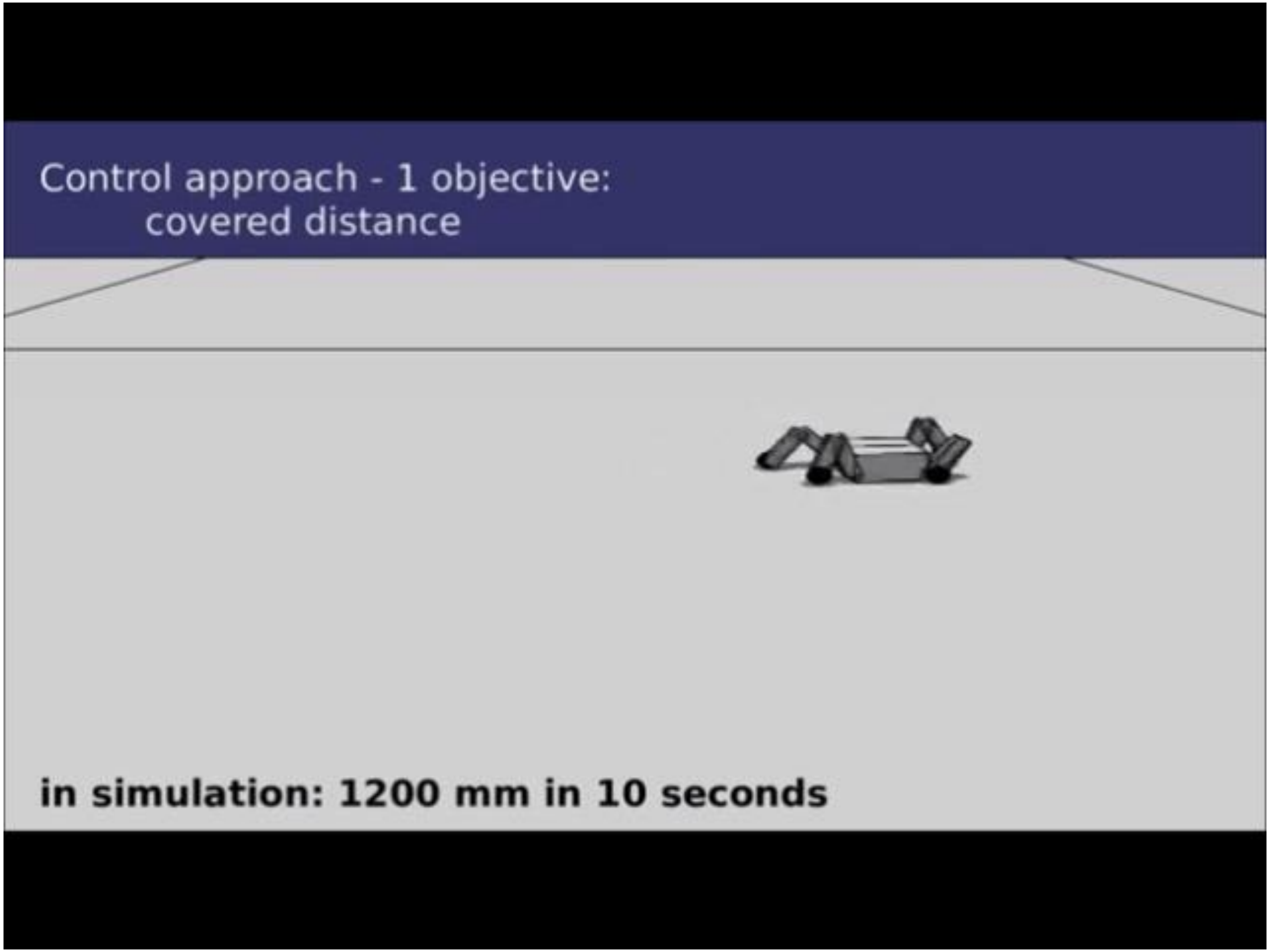
- Can we evolve robots in the real world?
- If we are interested in just the control, yes, although it is impractical (thousand of evaluations!)
- If we are interested also in evolving morphologies, simulation is necessary (at least with the current technology)

→ It has been observed that transferring solutions evolved in simulation to the real world is very hard and often fails

- The smallest discrepancy between the simulated environment and the real world may result in an unsuccessful transfer
- …And no model is as rich as the physical reality. *"There is no better model of the world than the world itself"*, R. Brooks

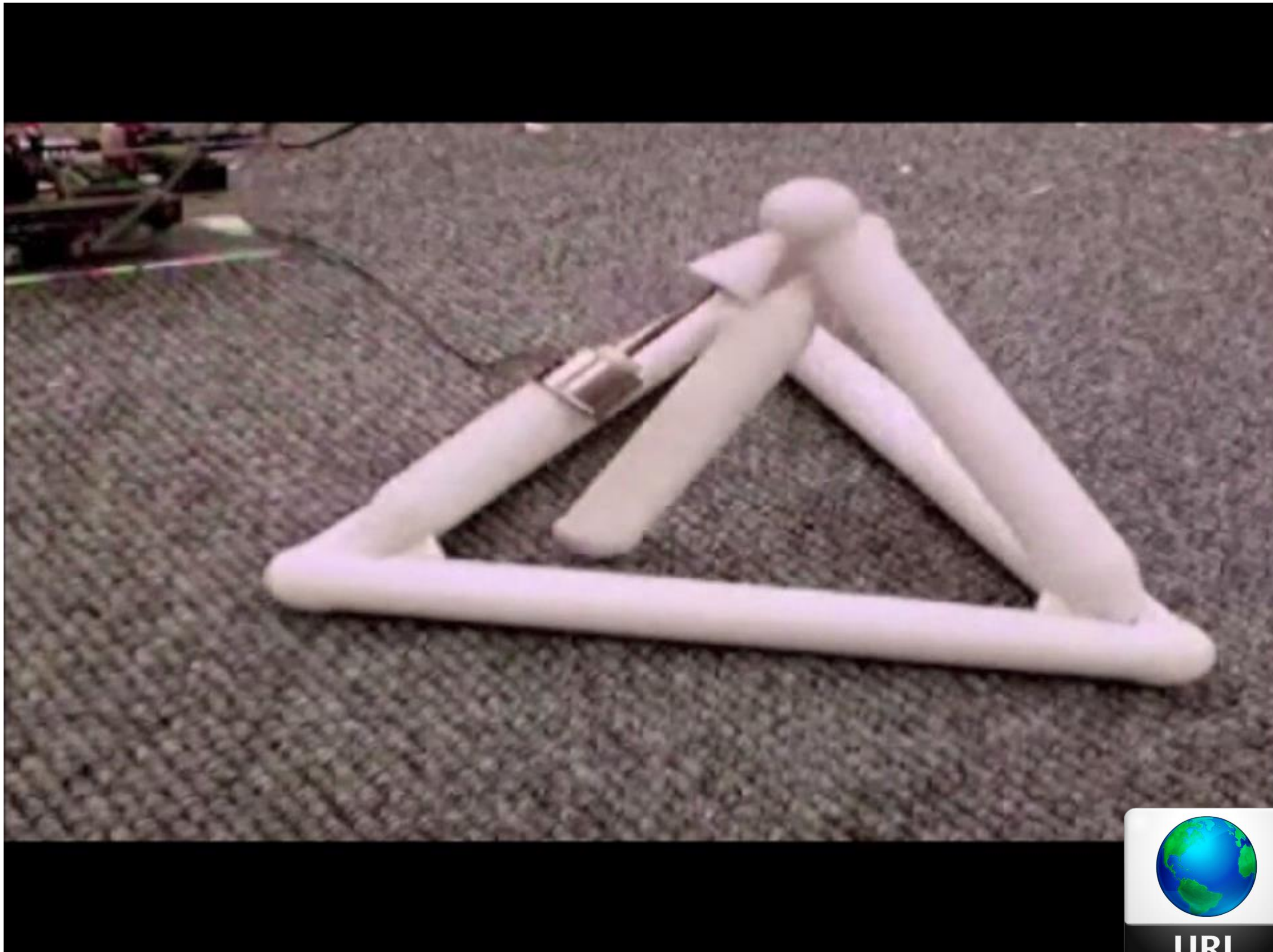
→ **Serious problem for evolutionary robotics**

An example of reality gap – [Koos et al, 2010]



URL

Crossing the gap – [Lipson & Pollack 2000]



Direct encoding:
Robot:= <vertices>
<bars>
<neurons>
<actuators>

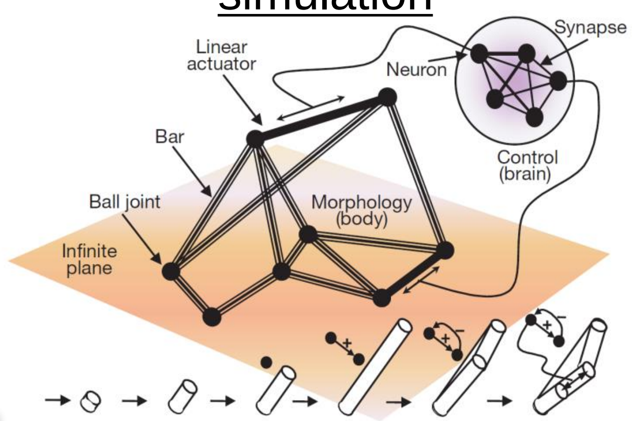
Vertex:=<x,y,z>

Bar:= <vertex 1 index,
vertex 2 index,
relaxed length,
stiffness>

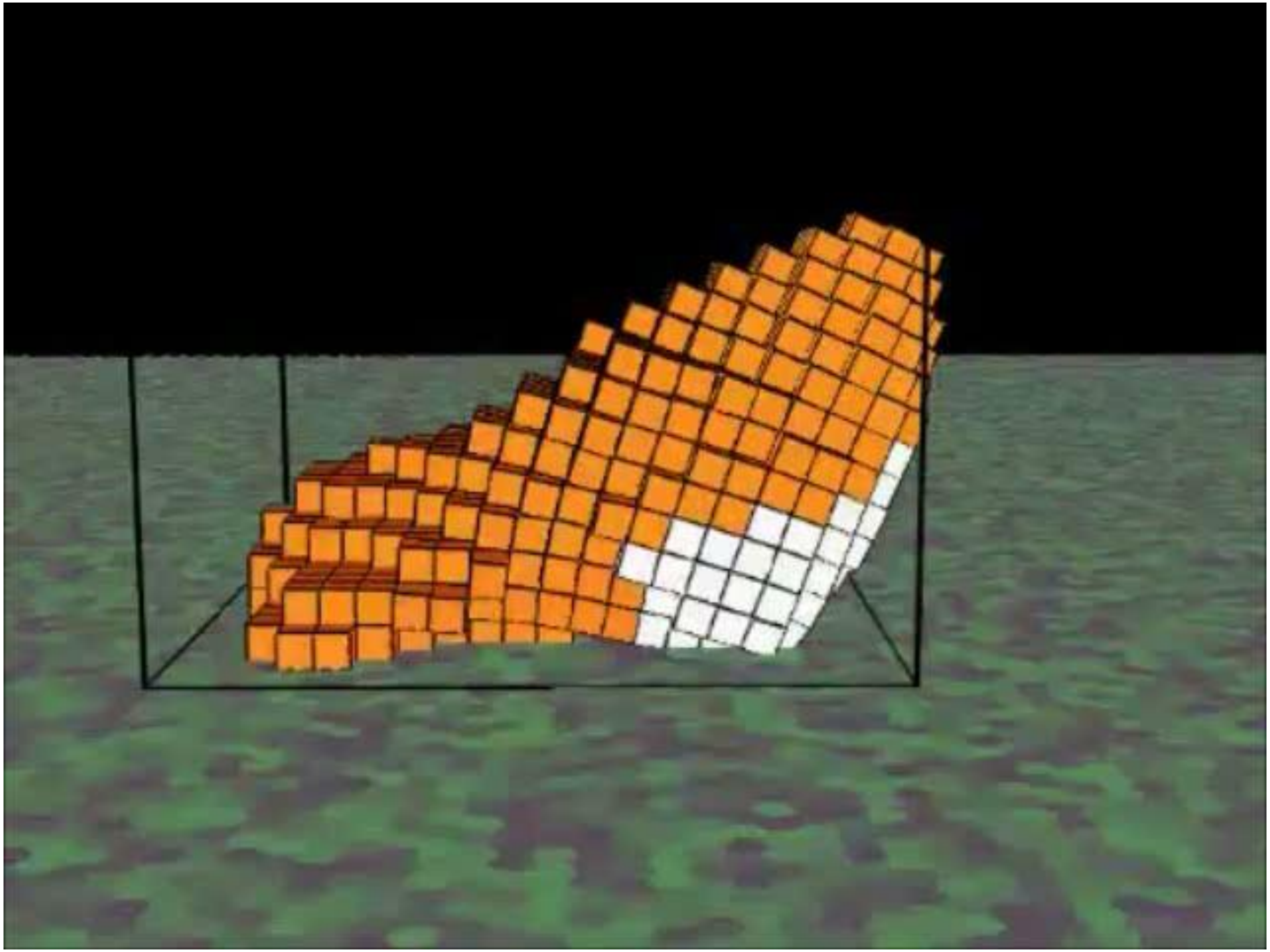
Neuron:=<threshold,
weights>

Actuator := <bar index,
neuron index,
bar range>

Noise added to the simulation

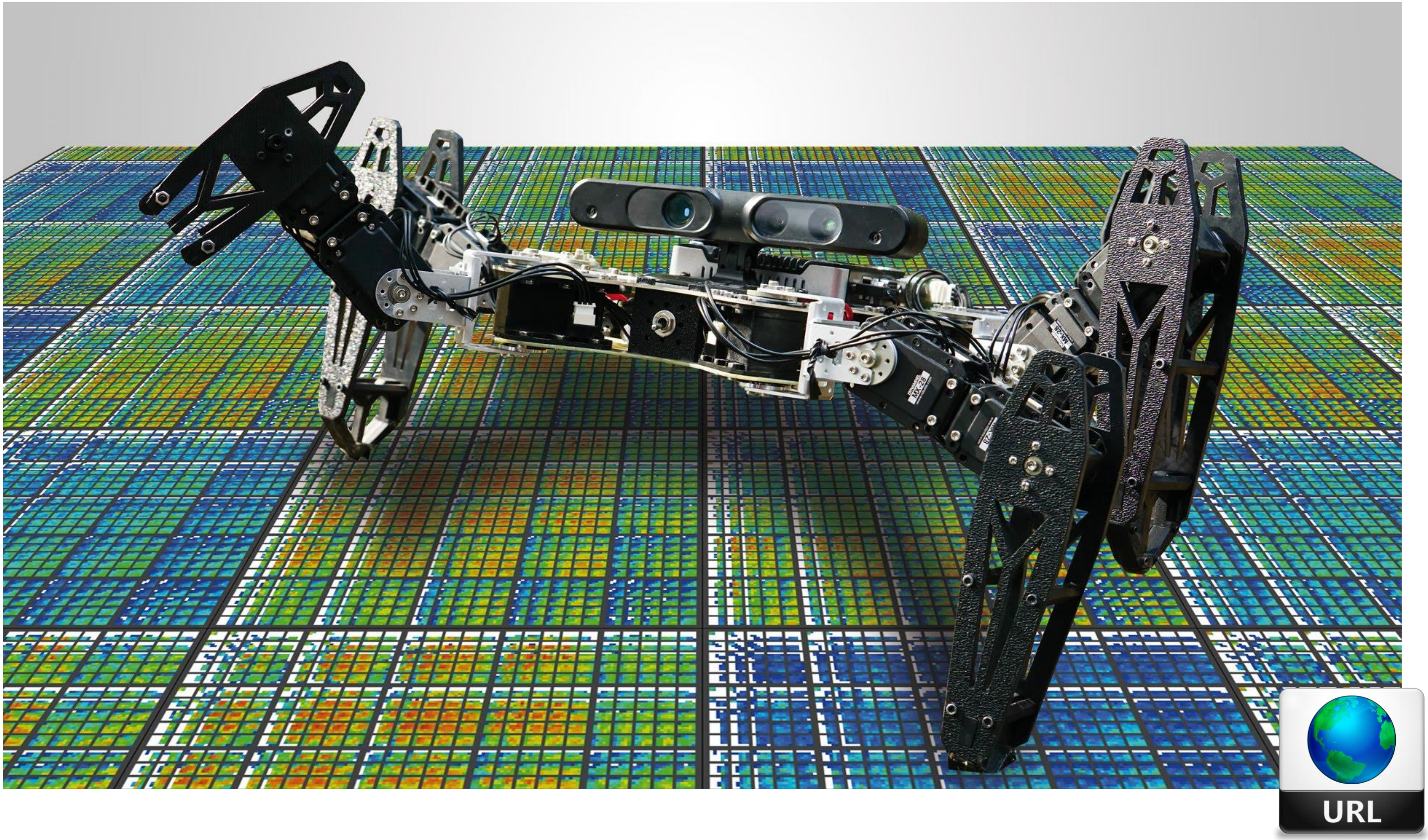


Evolving and fabricating soft robots – [Hiller et al., 2012]



Crossing the gap – [Bongard and Lipson., 2006]

Crossing the gap – [Cully et al, 2015]



Concluding remarks – On Evolution and AI

- Intelligence and life as we know are products of evolution
- If we are interested in better understanding them, and replicating some of their features in artificial form, an evolutionary approach may be the most appropriated
- By translating bio-inspired process into artificial substrates, surprising phenomena can arise, given the different embodiment of the evolving creatures and of the environment

→ **"Intelligence, and life as-it-could-be"**, i.e. alternative forms of intelligence/life

→ Could also help in understanding what is intelligence in general, detached from the biological embodiements we are surrounded with

Concluding remarks – On Evolution and AI

- The *complexity barrier*: some problems are too hard for humans to conceive
- Lot of examples in science and engineering, where we always try to break down the complexity of phenomena in order to be able to manage them (recall the example about evolvable hw and modularity)

→ **Evolutionary approaches can be used to solve very difficult problem with super-human skills (NASA antenna)**

→ ***Human-competitive design* in several fields**

Concluding remarks – On Evolution and AI

Regarding the complexity barrier…

- Some think that <u>the problem of understanding and replicating general machine intelligence may be too hard for us to conceive</u>, and this may be the reason why are struggling on this
- <u>We are biased</u> by how we think, how we are made, what we have experienced
- This may limit our understanding of the phenomena

→ Evolution, even the artificial one, is not biased, and often demonstrates to *“think outside the box”*

→ Moreover, it already produced several life-like phenomena

→ <u>May provide the solution</u>

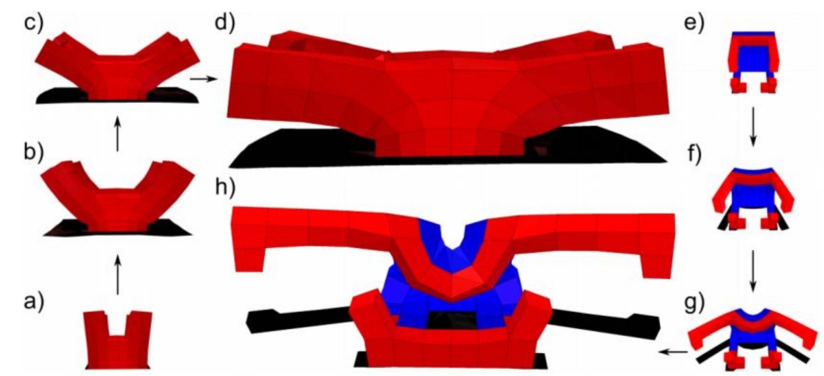
Concluding remarks – On Evolution and AI

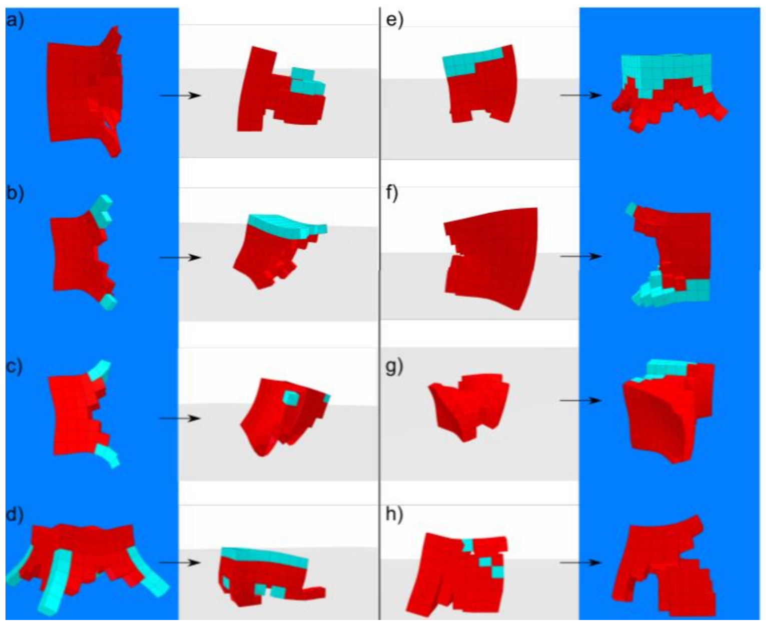
- One genuine form of intelligence (both biological and artificial) consists in the ability of an agent to come up with truly *creative* and *surprising* solutions

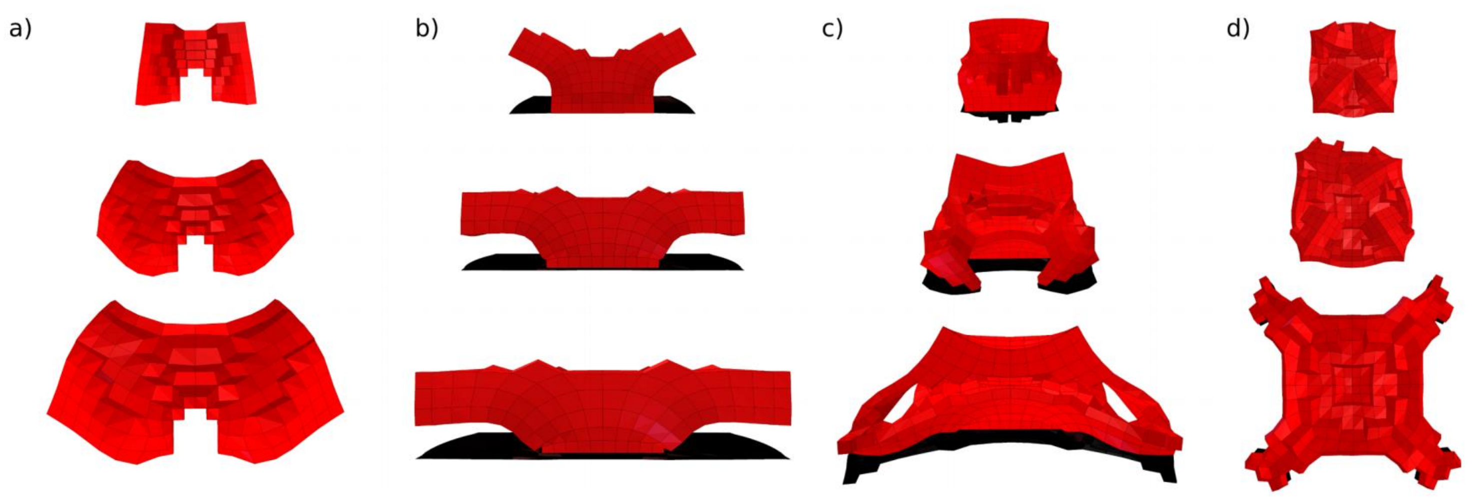
→ <u>Artificial evolution often demonstrates such a creativity</u>

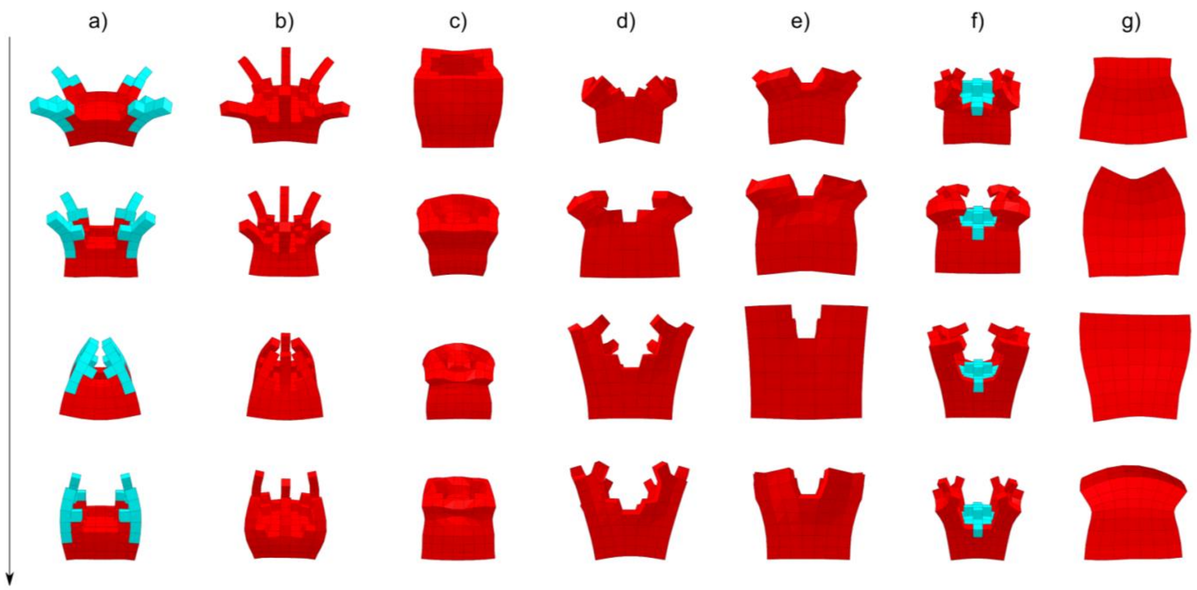
What are we doing

- Using artificial evolution to investigate the emergence of adaptive and intelligent behavior in animals and robots
- Brain-body co-evolution of generic soft-bodied creatures/robots
  - Evolution of growing soft robots
  - Evolution of swimming soft robots





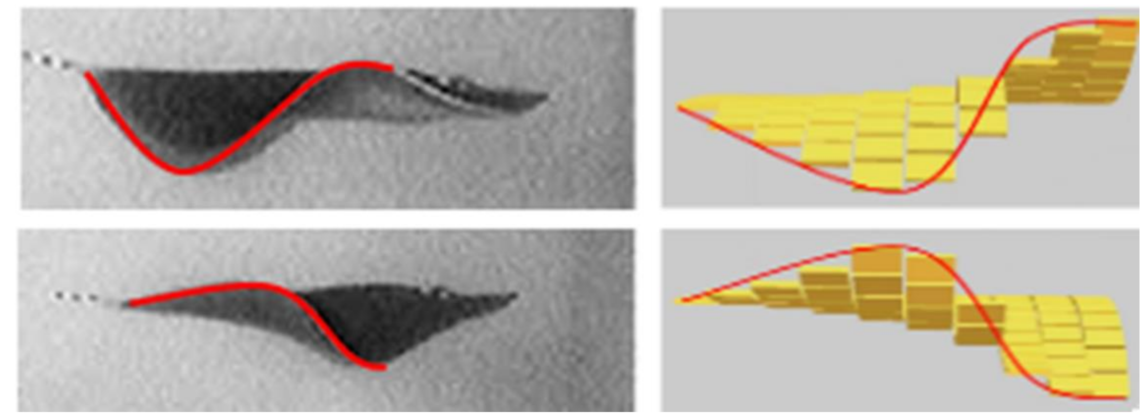


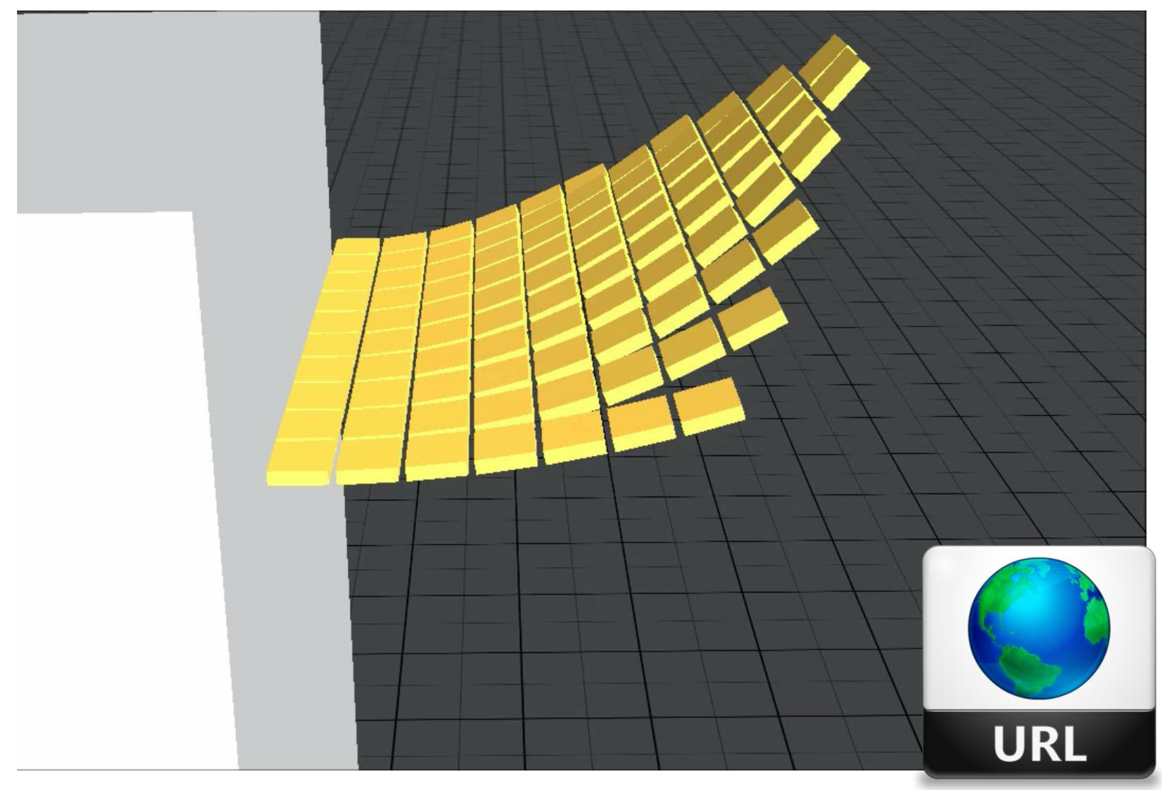


What are we doing

Using artificial evolution to study properties of specific animals

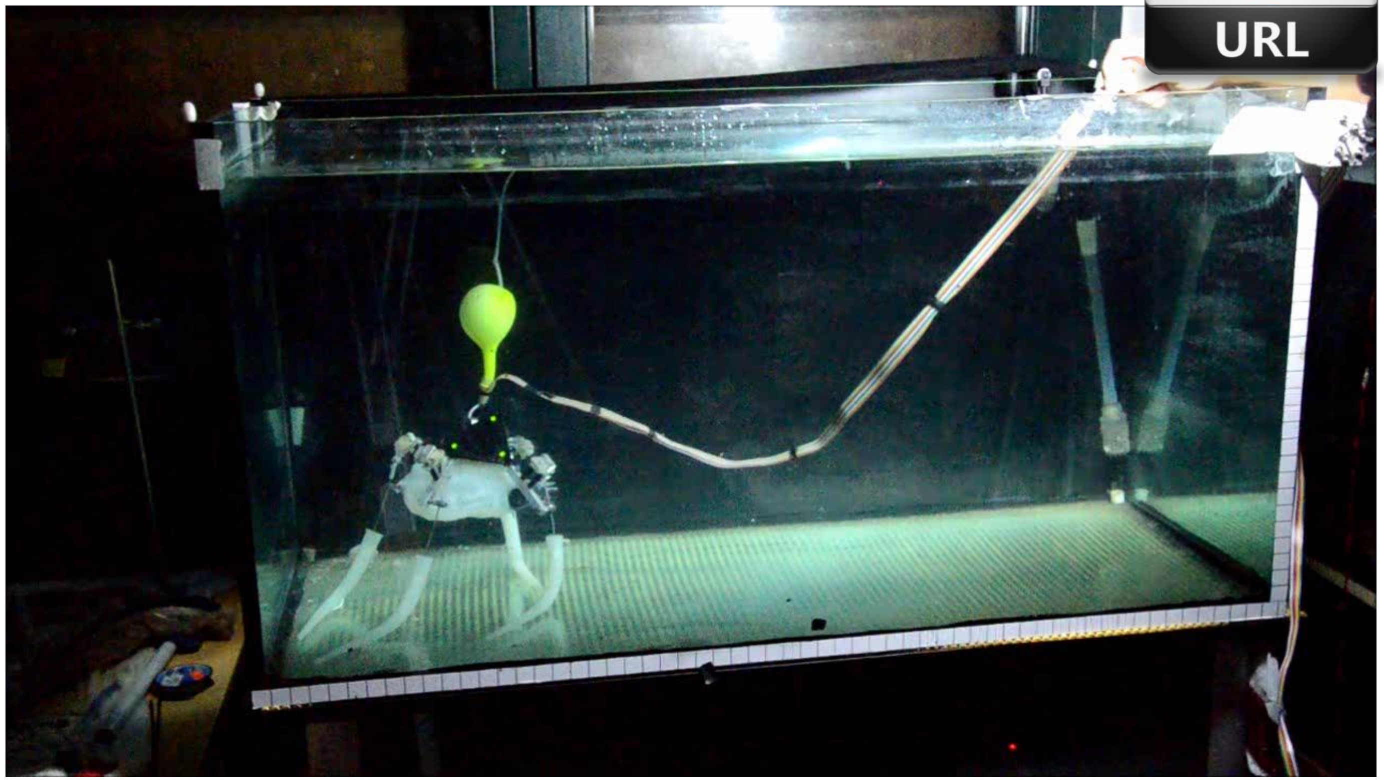
Example: Artificial Evolution and adaptation to different environments of a manta-like fin

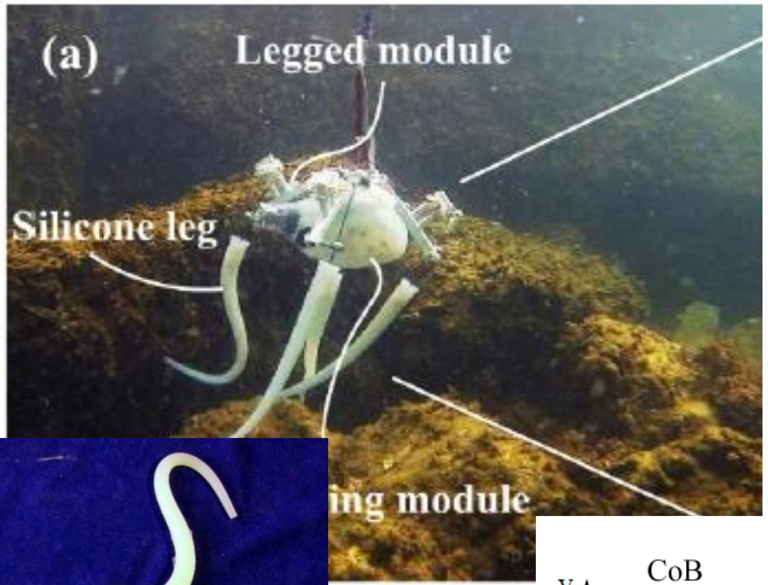


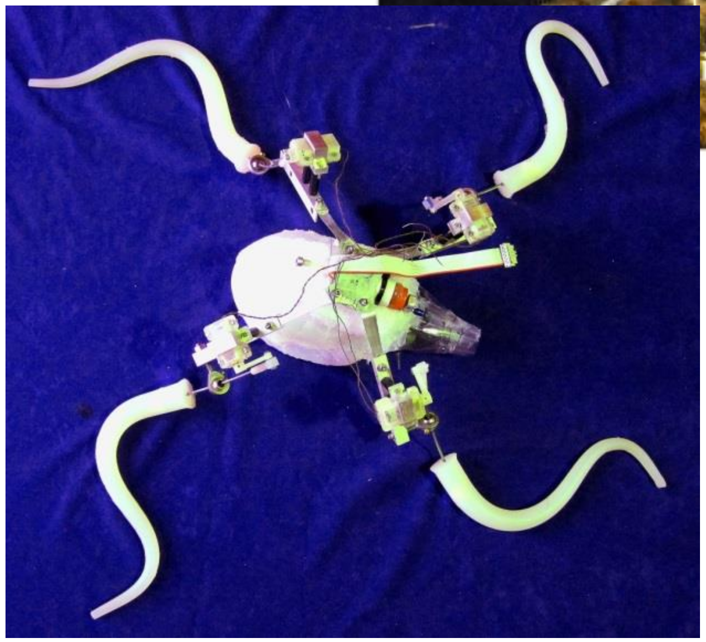


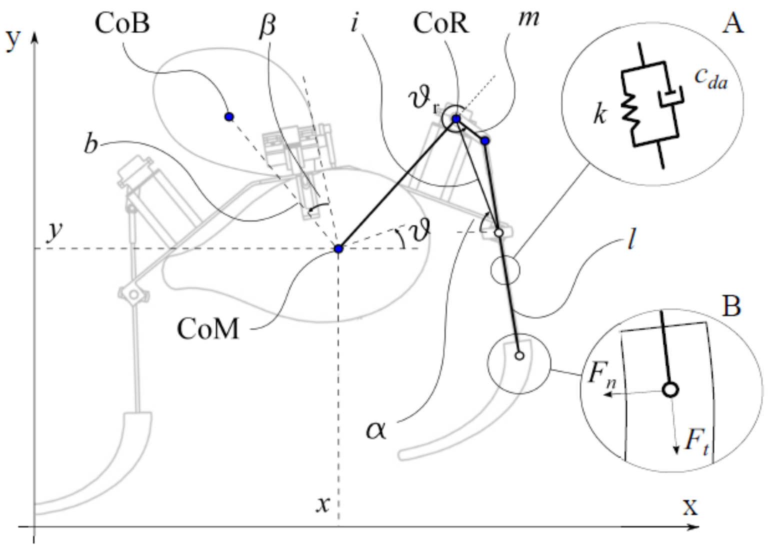
What are we doing

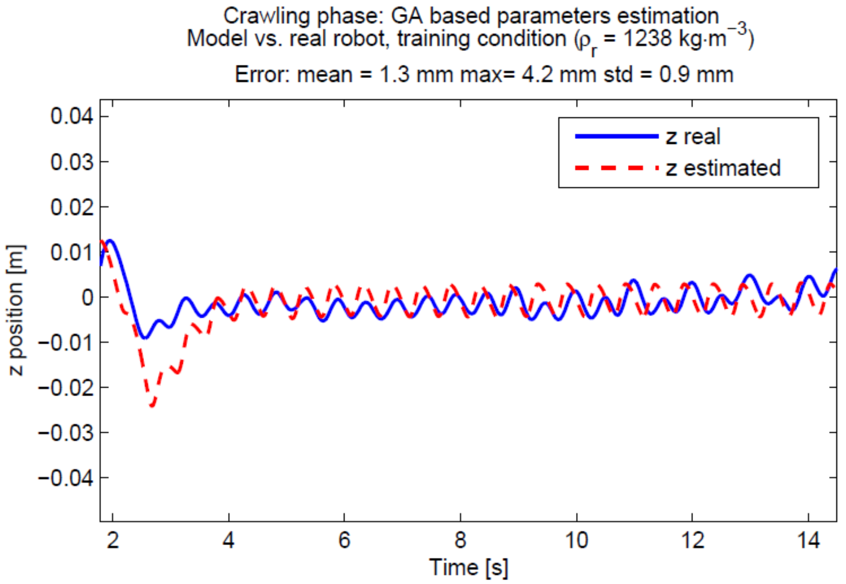
Applying <u>evolutionary design techniques</u> to improve bio-inspired robots





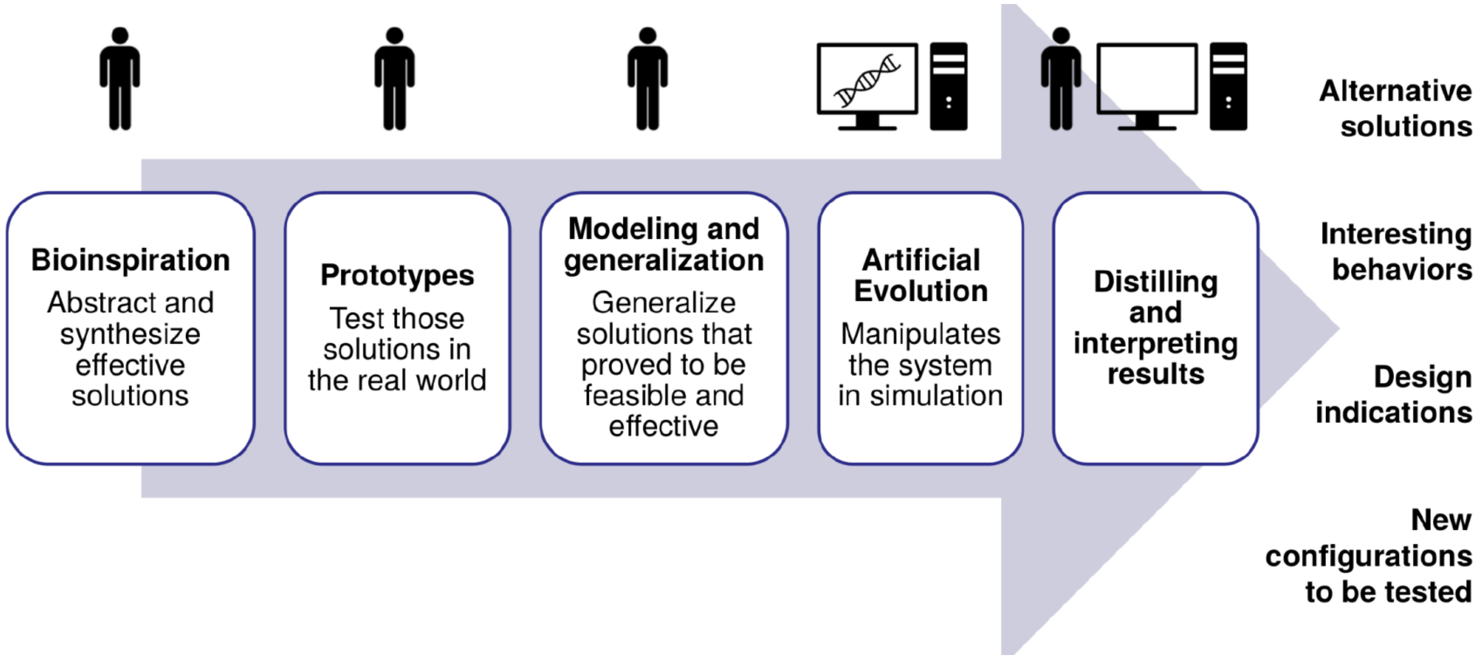


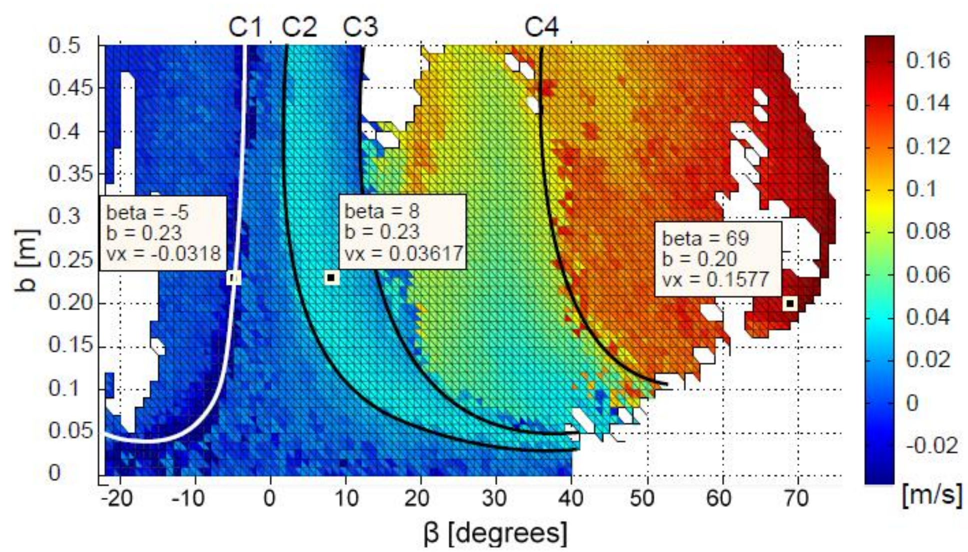




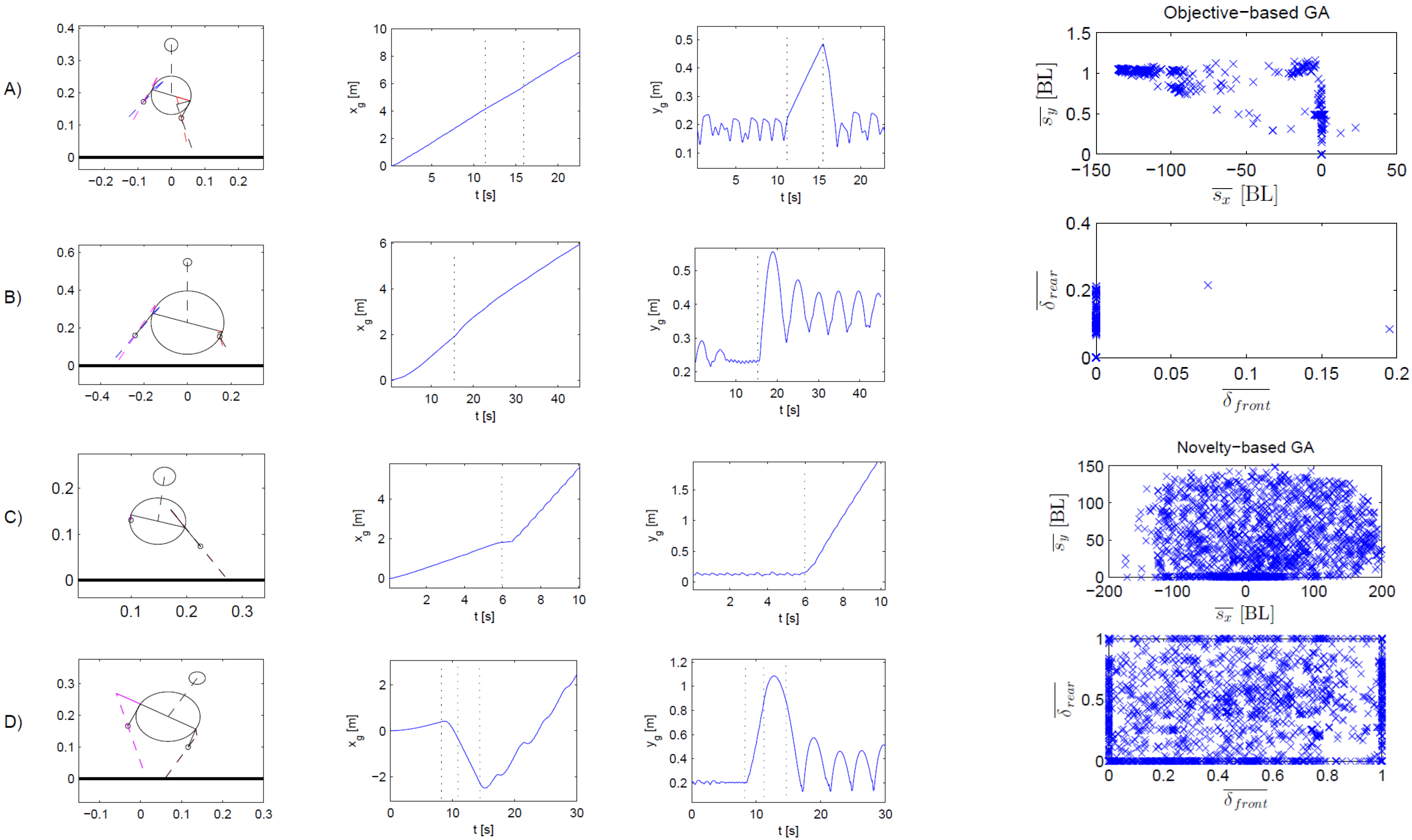
What are we doing

- <u>Human-machine collaborative evolutionary design</u>
- Investigating the feasibility and potential benefits of <u>shape-changing robots (morphing/morphosis)</u>, i.e. robots that can reconfigure their body to adapt to different tasks/environmental conditions





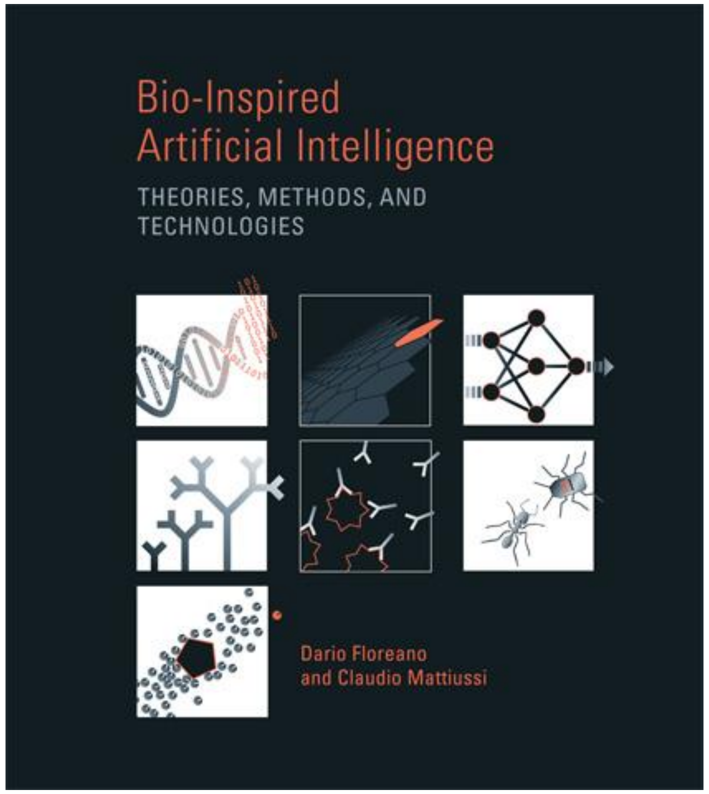
What are we doing

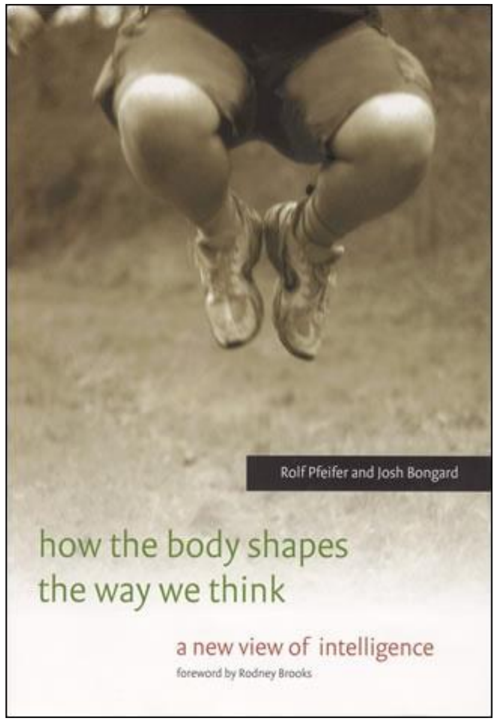


Suggested readings, main references

1. "<u>Bio-Inspired Artificial Intelligence</u>", Theories, Methods and Technology, Dario Floreano and Claudio Mattiussi → *Chapter 1 + part on neuroevolution (starting at p. 238) – but the whole book is very nice*

2. "<u>How the body shapes the way we think</u>", Rolf Pfeifer and Josh Bongard – a new, intriguing view of intelligence → Chapter 6 on Artificial Evolution – *the whole book is a must for everyone in the field of AI and robotics*





Additional readings

3. Neuroevolution

   http://www.scholarpedia.org/article/Neuroevolution

   http://infoscience.epfl.ch/record/112676/files/FloreanoDuerrMattiussi2008.pdf

4. Evolutionary robotics

   http://www.cs.uvm.edu/~jbongard/papers/2013_CACM_Bongard.pdf

5. Plenty of research papers we can provide you, if you are interested (some are linked throughout the presentation)

For doubts, additional material,

projects and theses ideas, suggestions …

Feel free to contact me

f.corucci @ sssup.it

http://sssa.bioroboticsinstitute.it/user/1507