OPTIMIZATION TECHNIQUES

Genetics, evolution and nature-inspired analogies

Prof. Giovanni Iacca

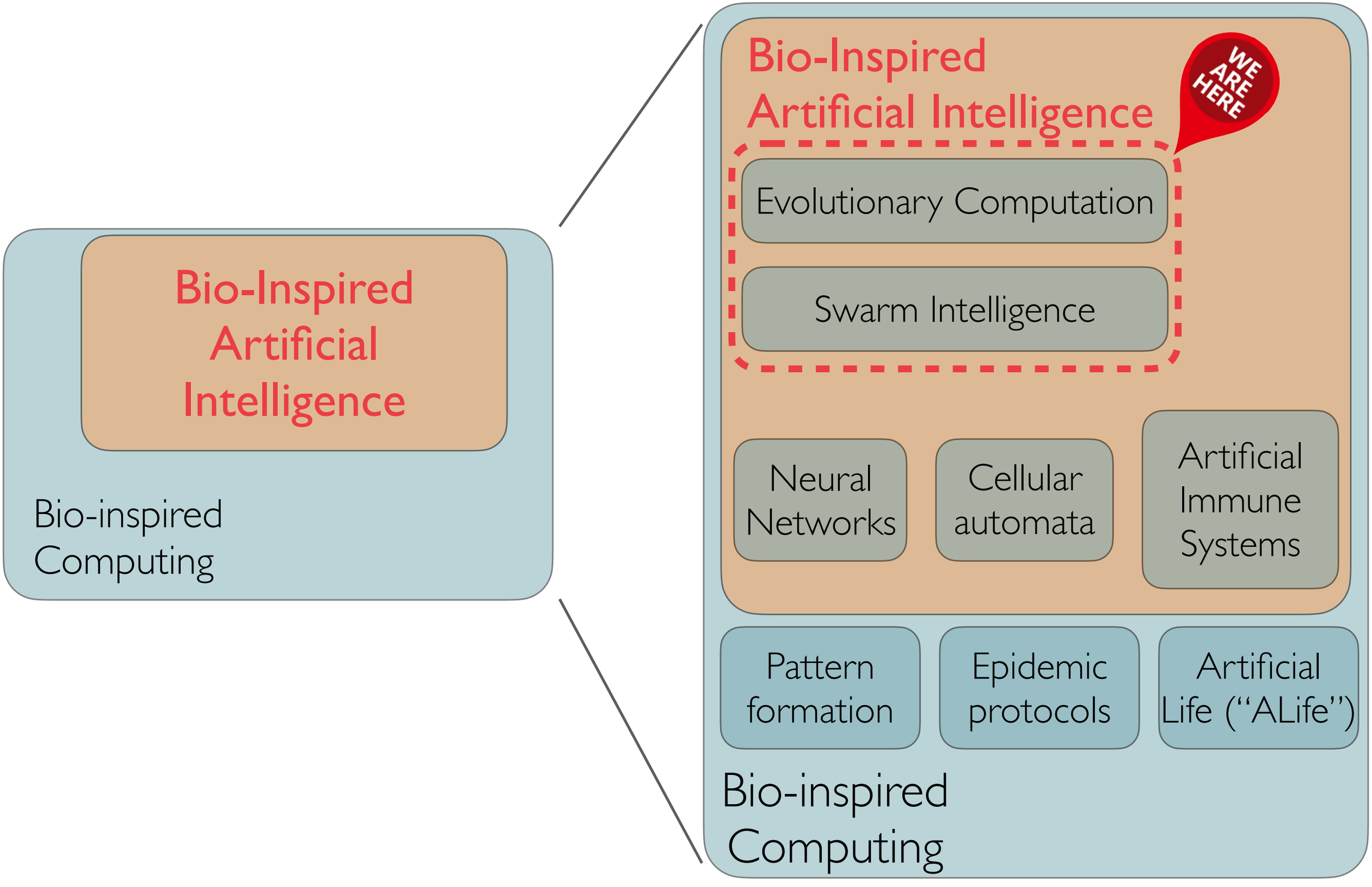
giovanni.iacca@unitn.it

UNIVERSITY OF TRENTO - Italy

Information Engineering
and Computer Science Department

Metaheuristics

METAHEURISTICS



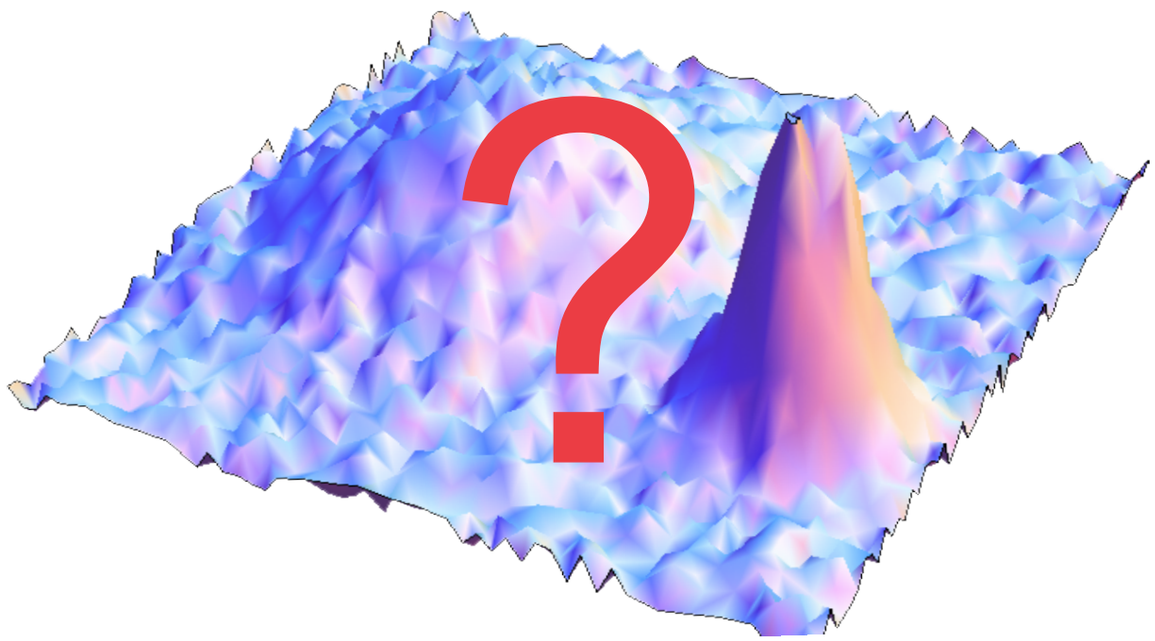
METAHEURISTICS

WHAT CAN WE USE FOR GLOBAL OPTIMIZATION? METAHEURISTICS!

- From the ancient Greek μετα – ευρισκω meta heuriskō, i.e., "I find beyond": algorithms that do not require any assumption on the objective function ("black-box" optimization).
- Especially useful when an analytical expression of the objective function is not even available (e.g., output of simulations), or it is extremely complex (multivariate, noisy, non-differentiable, non-continuous, non-linear, etc.).

COMPUTATIONAL INTELLIGENCE OPTIMIZATION (CIO)

A subfield of CI that studies mathematical procedures to solve optimization problems, especially in the cases when there are no hypotheses and a metaheuristic is the only option.

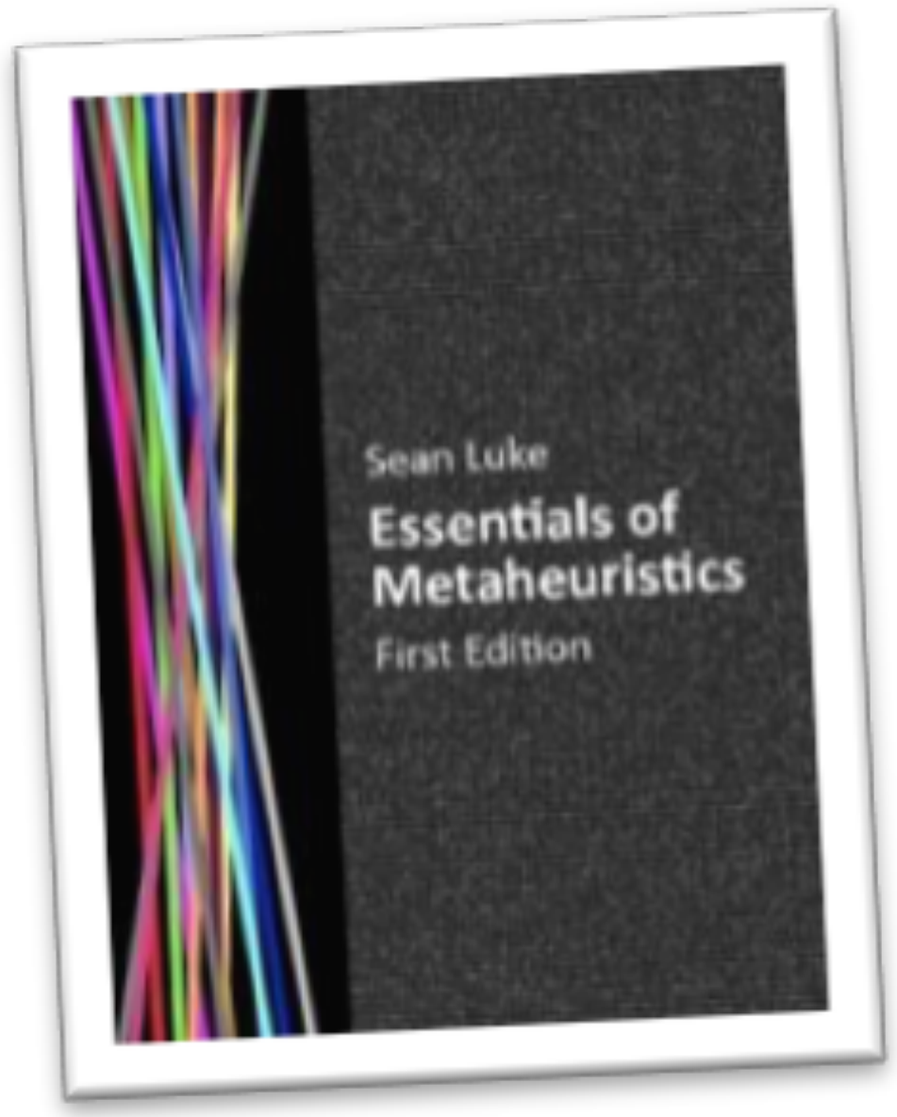


METAHEURISTICS

ONE NAME FOR (TOO) MANY ALGORITHMS

*A common **but unfortunate name** for any stochastic optimization algorithm intended to be the last resort before giving up and using random or brute-force search. Such algorithms are used for problems where you don't know how to find a good solution, but if shown a candidate solution, you can give it a grade.*

Sean Luke, "Essentials of Metaheuristics"

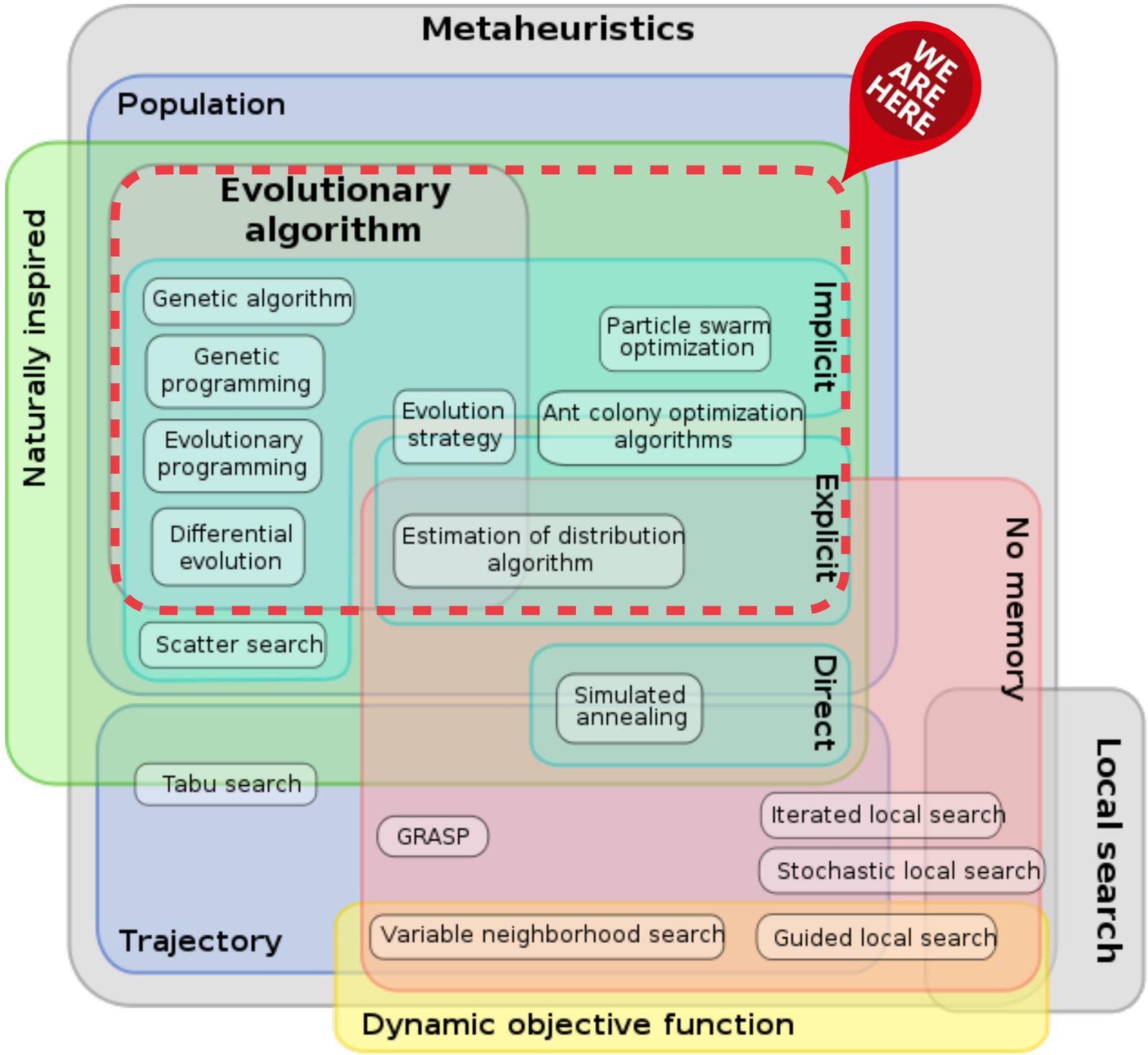


http://cs.gmu.edu/~sean/book/metaheuristics/

METAHEURISTICS

A (POSSIBLE) TAXONOMY

- population metaheuristics vs trajectory-based ones
- nature-inspired methods or not
- dynamic objective function vs static ones
- memory-based algorithms vs memory-less
- implicit, explicit or direct metaheuristics

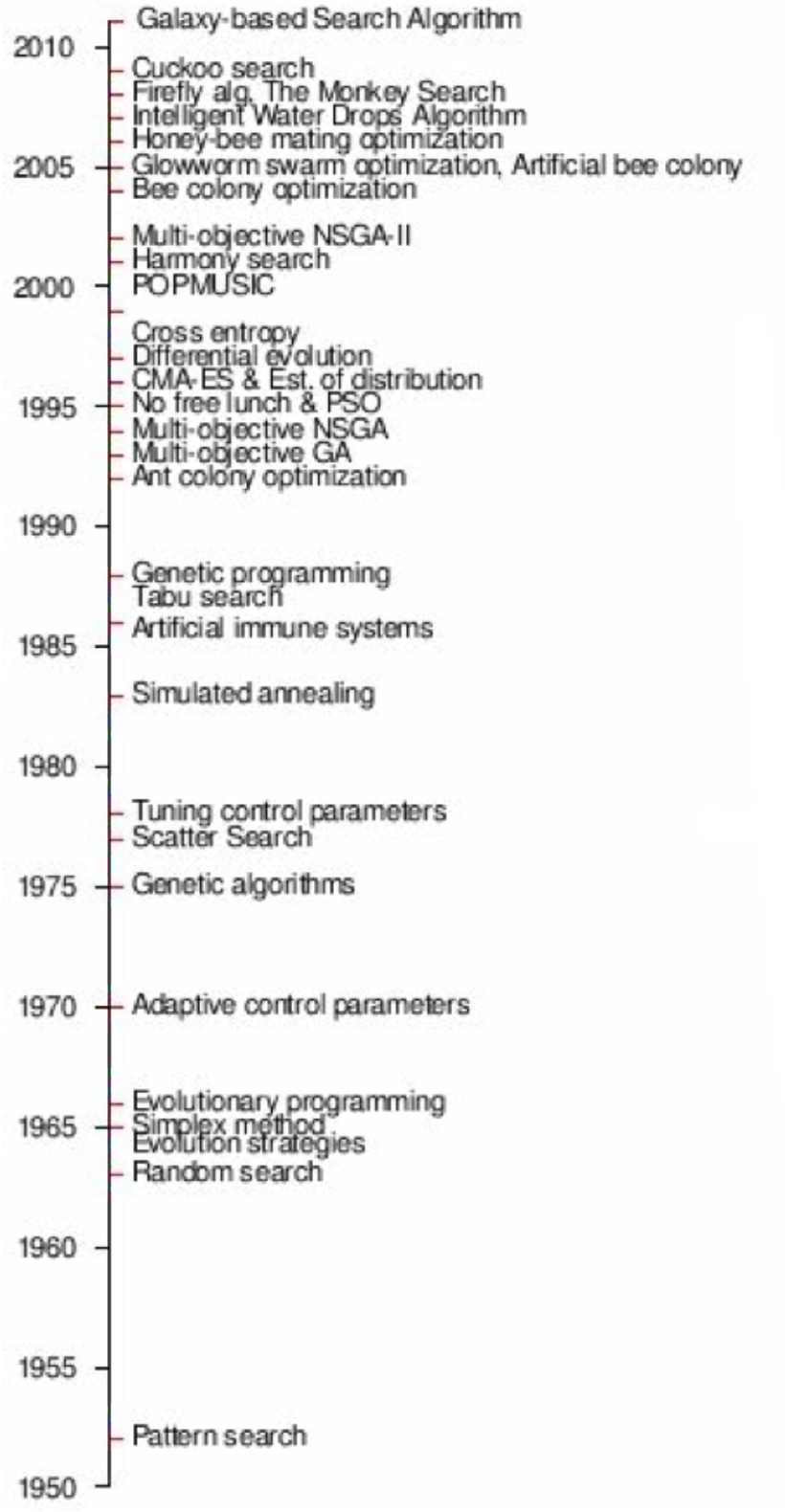


http://metah.nojhan.net/post/2007/10/12/Classification-of-metaheuristics

METAHEURISTICS

SOME EXAMPLES

- Simulated Annealing (SA)
- Evolutionary Algorithms (EAs)
- Genetic Algorithms (GAs)
- Evolutionary Programming (EP)
- Evolution Strategies (ES)
- Particle Swarm Optimization (PSO)
- Ant Colony Optimization (ACO)
- Bacterial Foraging Optimization (BFO)
- Differential Evolution (DE)
- Memetic Algorithms (MA)
- Hybrid Methods



METAHEURISTICS

SOME CRITICISM

Metaphors are not wrong in themselves, and have served the community well as the inspiration for powerful new methods. However, methods whose only novelty is the metaphor they are based upon, should not be considered valuable research. Ants, bees, bacteria, wolves, cats, cuckoos, eagles, fireflies, fish, glowworms, krill, monkeys, bats, dolphins, green herons, Egyptian vultures, and virtually every other type of animal all have been used as the basis for a "novel" metaheuristic, as have imperialist societies, anarchic societies, clouds, consultants, the big bang, black holes, gravity, lightning, electromagnetism, "intelligent" water drops, river formation, the water cycle, musicians playing music, etc. The list is virtually endless.

Kennet Sörensen, "Metaheuristics—the metaphor exposed"

https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12001
http://www.sigevolution.org/issues/SIGEVOlution0901.pdf

METAHEURISTICS

NO FREE LUNCH THEOREMS (NFLT)

- For a given pair of algorithms A and B:

$$\sum_f P(x_m|f,A) = \sum_f P(x_m|f,B)$$

where $P(x_m|f,A)$ is the probability that algorithm A detects the optimal solution x_m for a generic objective function f and $P(x_m|f,B)$ is the analogue probability for algorithm B (D. Wolpert & W. Macready, 1997).

- The performance of every pair of algorithms over all the possible problems is the same.

Q: Which one is the best optimizer?

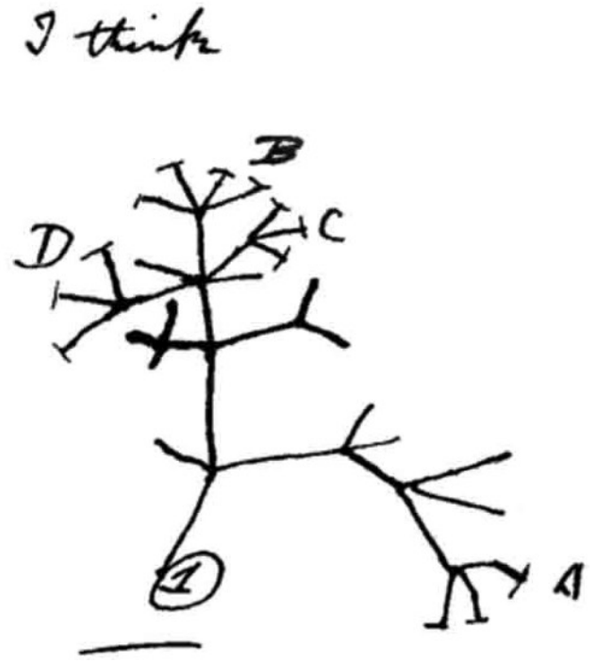
A: The is no best!

IMPLICATIONS

- Metaheuristics are not "magic": albeit general purpose, there is no such thing as a metaheuristic "solving all the problems".

- Every problem should be addressed with a proper algorithm that is tailored around the problem features.

Natural Evolution

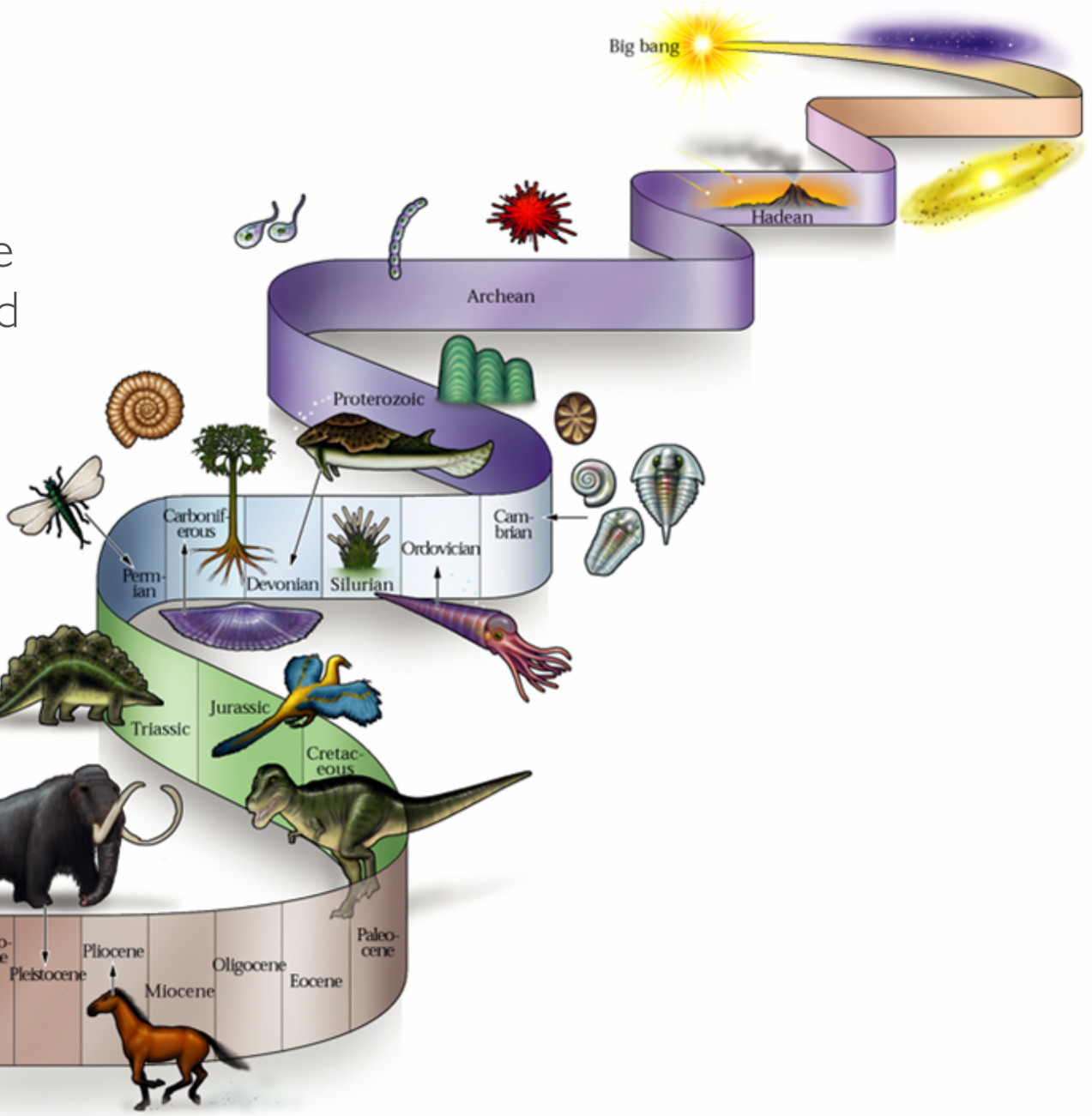


NATURAL EVOLUTION

WHAT IS EVOLUTION?

- Biological systems result from an *evolutionary process*: all types ("species") of plants, animals, and other living things on Earth have their origin in other preexisting types, and the distinguishable differences are due to modifications accumulated in successive generations

- The resulting biological systems are:
  - robust
  - complex
  - adaptive

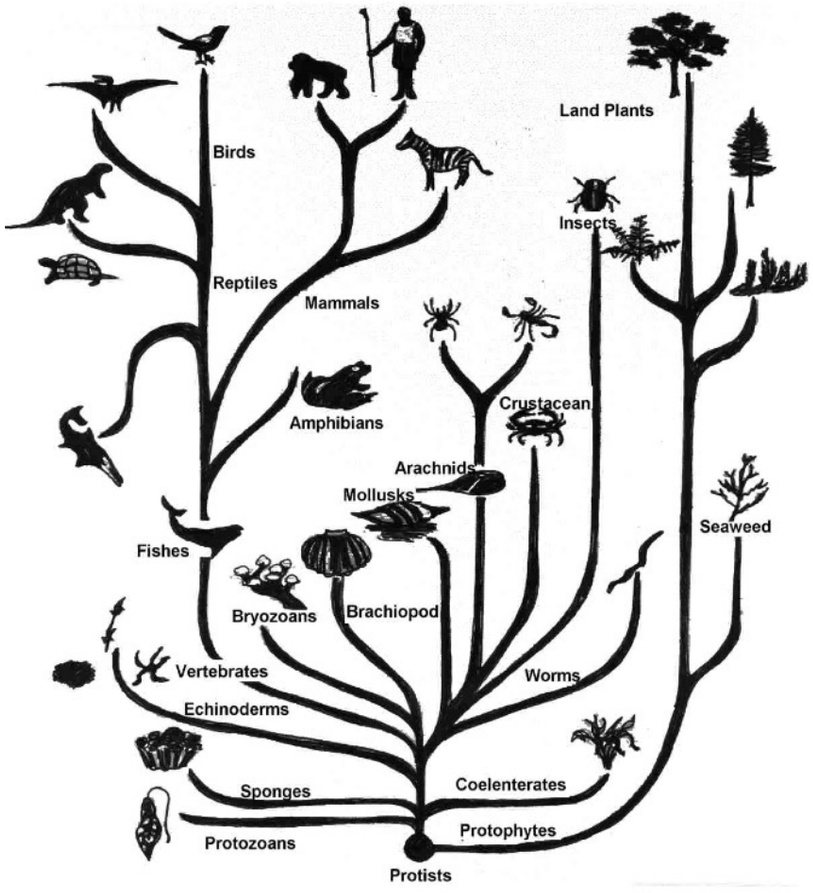


NATURAL EVOLUTION

THE "FOUR PILLARS" OF EVOLUTION

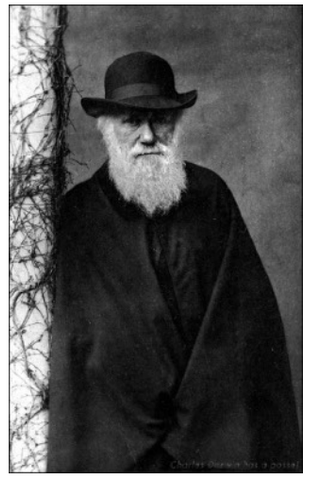
Population
Group of several individuals
Diversity
Individuals have different characteristics, obtained by accumulation of changes (mutations)
Heredity
Characteristics are transmitted over generations
Selection

- Individuals make more offspring than the environment can support
- Better ("fitter") individuals (e.g., better at food gathering, fighting/escaping predators, etc.) make more offspring (higher **fitness**) and/or make offspring that have higher chances of surviving (and, in turn, reproducing)



All species derive from common ancestor

C. Darwin, On the Origins of Species, 1859



NATURAL EVOLUTION

EVOLUTION: NOT A RANDOM PROCESS!

- BUT: randomness affords the materials on which evolution acts

- Result of two effects:
  - Variations (mostly random)
  - Selection (mostly deterministic)

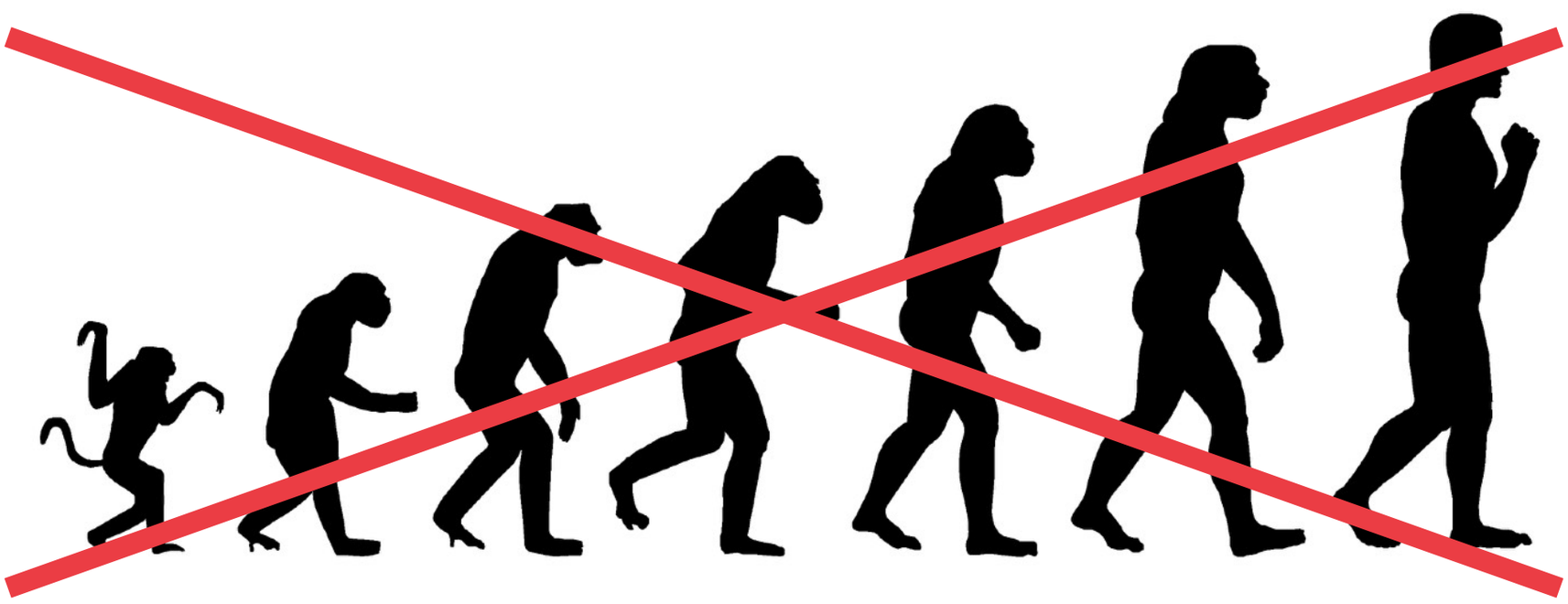
NATURAL EVOLUTION

EVOLUTION: NOT A DIRECTIONAL PROCESS!

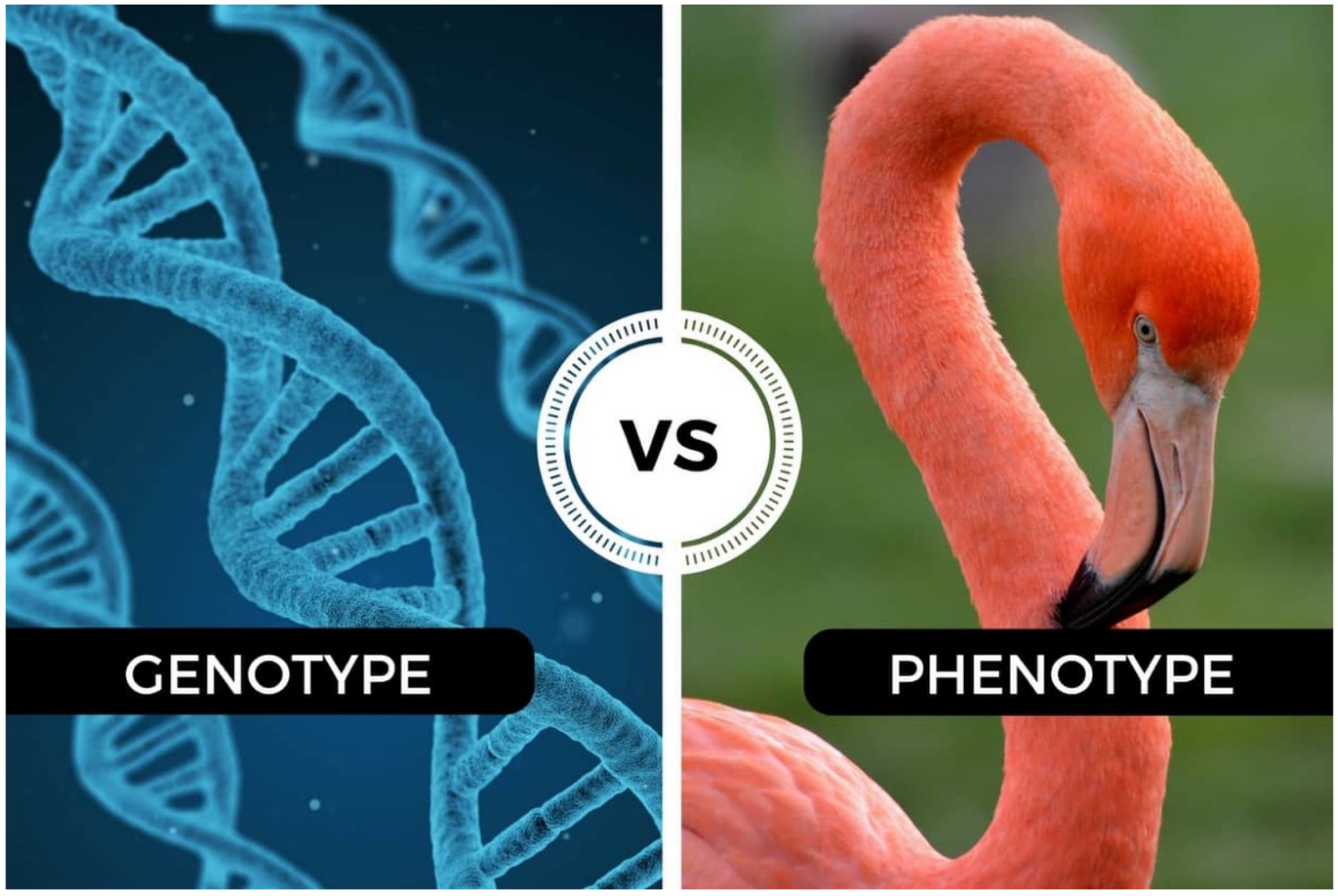
Humans (or any other species) are not the top of the evolutionary ladder (misleading image of evolution with humans at top or end)!

Instead, evolution proceeds "without progress":

- If no competition, no selection of the fittest
- Individuals selected against current environment
- Accumulation of changes with no cost or benefit ("neutral evolution")



NATURAL EVOLUTION



GENOTYPE

- The "genetic material" of an organism, transmitted during reproduction
- Affected by mutations (and crossover)
- Selection does not operate directly on it

PHENOTYPE

- The "manifestation" of an organism (appearance, behavior, etc.)
- Affected by environment, development, learning
- Selection operates on it

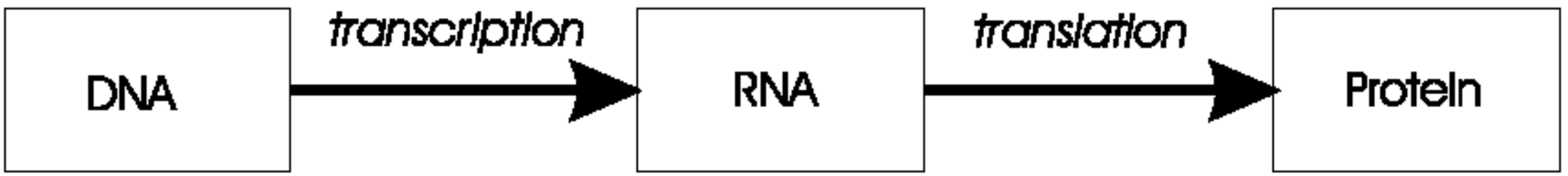
NATURAL EVOLUTION

FROM GENOTYPE TO PHENOTYPE

- A central claim in molecular genetics: only one-way information flow
  genotype → phenotype
  genotype ↛ phenotype

- Phenotypic traits:
  - Behavior/physical differences that affect response to environment
  - Partly determined by inheritance, partly by factors during development
  - Unique to each individual, partly as a result of random changes

- The genotype—phenotype mapping is VERY complex:
  - One gene may affect many phenotypic traits (**pleiotropy**)
  - Many genes may affect one phenotypic trait (**polygeny**)

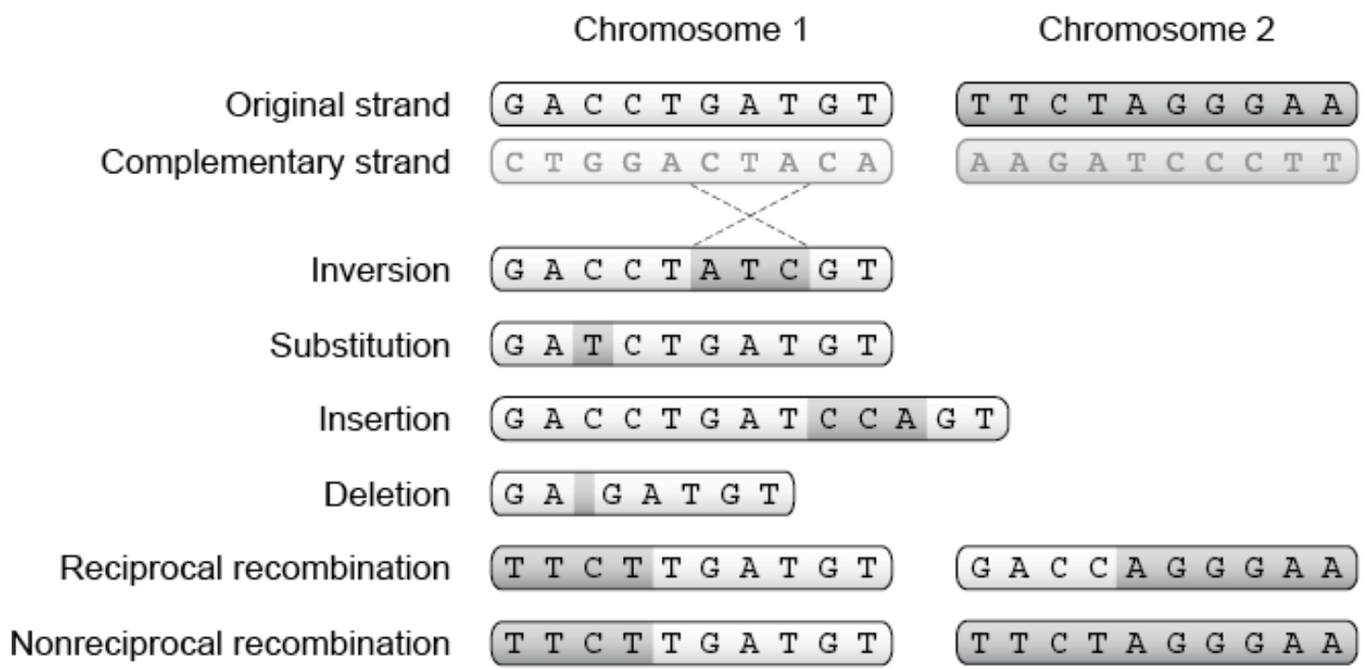


NATURAL EVOLUTION

GENETIC MUTATIONS

- Genetic **mutations** occur during cell replication (replication errors)
- Those that occur in sex cells can affect evolution (transmitted to offspring)
- Mutations can be
  - catastrophic: offspring in not viable (most likely)
  - neutral: new feature does not influence fitness
  - advantageous: "strong" new feature occurs (leads to higher fitness)
- Redundancy in the genetic code forms a good way of error checking

Recombination (or **crossover**) is a "kind of mutation" that affects two homologous chromosomes



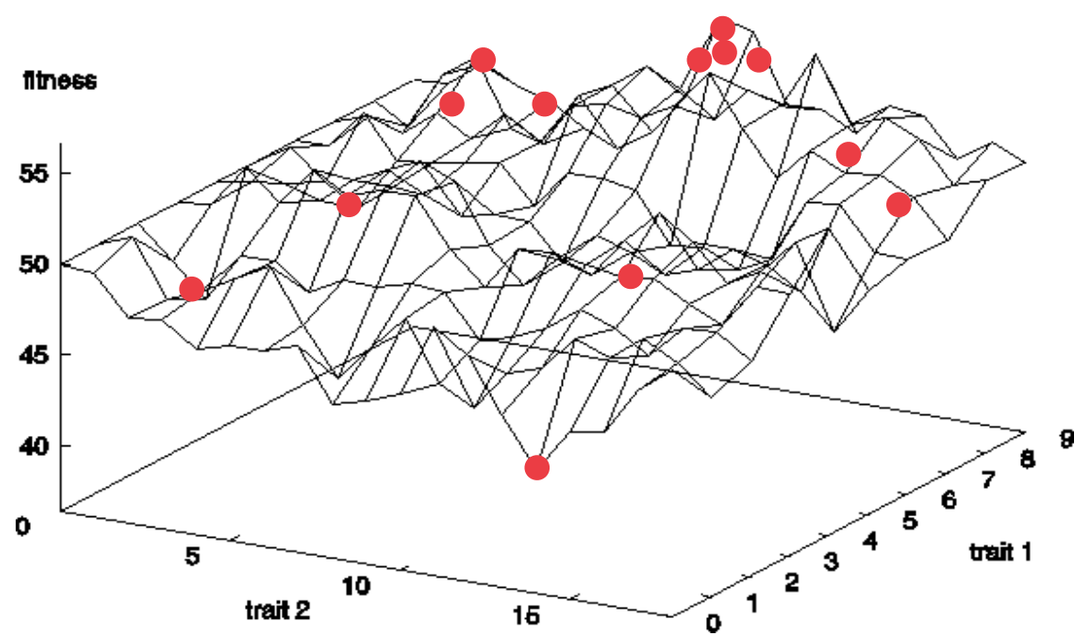
NATURAL EVOLUTION

ADAPTIVE LANDSCAPE METAPHOR

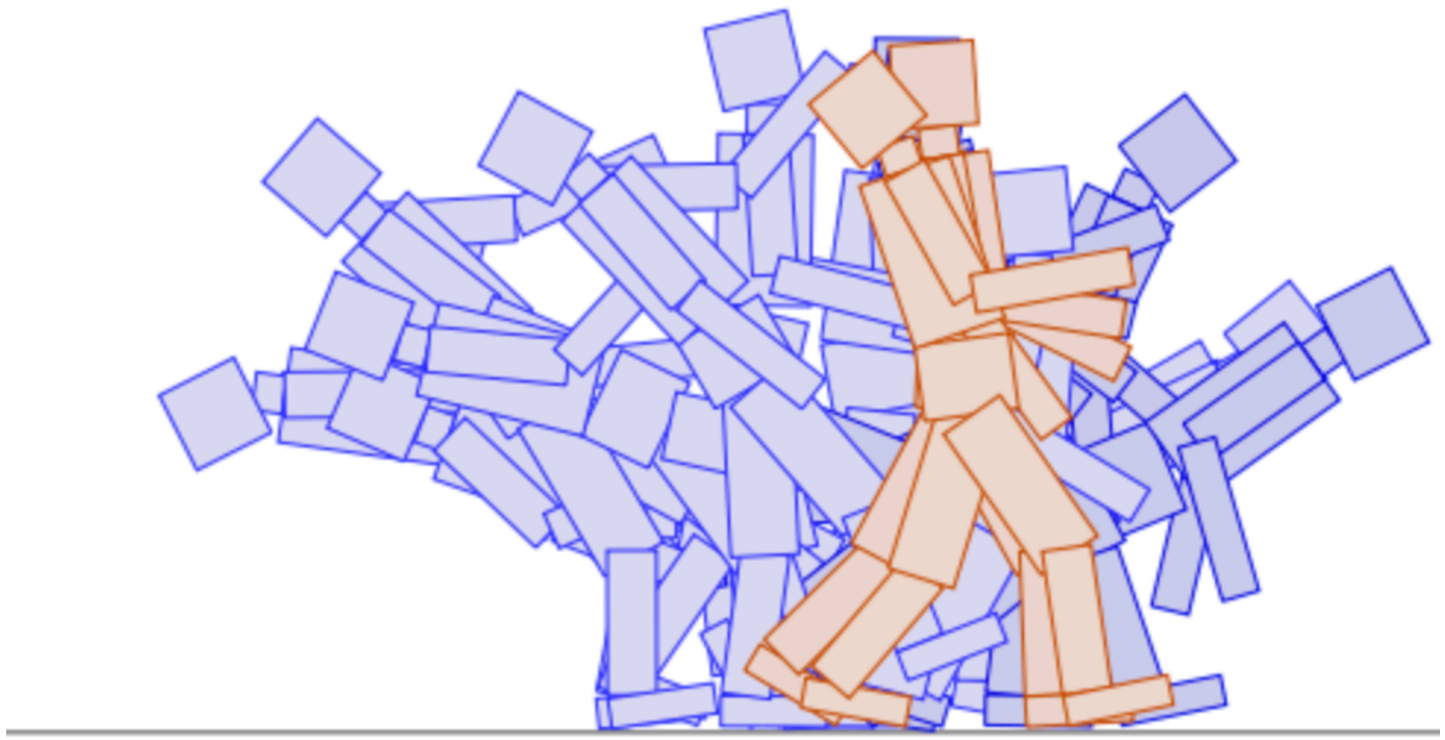
- Can envisage a population with *n* phenotypic traits as existing in a (*n*+1)-dimensional space (**landscape**) whose height corresponds to fitness, i.e., the no. of offspring (S. Wright, 1932)
- Each different individual (phenotype) represents a single point on the landscape
- Population can be seen therefore as a "cloud" of points, moving on the landscape over time as it evolves (by means of mutations) → adaptation
- Selection "pushes" population up the landscape

Genetic drift

- Random variations in feature distribution (+/-) arising from sampling errors
- Can cause the population to "slide over" hills, thus crossing valleys and leaving local optima



Evolutionary Computation



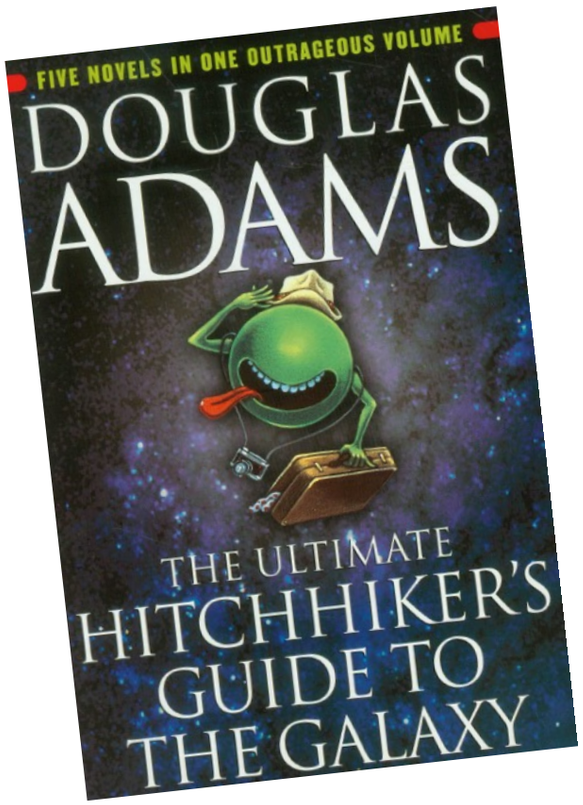
EVOLUTIONARY COMPUTATION

WHAT IS EVOLUTIONARY COMPUTATION?

A broad set of general-purpose computational techniques that attempt to copy the process of natural evolution → generally called **Evolutionary Algorithms** or **Artificial Evolution algorithms**

WHY COPYING NATURAL EVOLUTION?

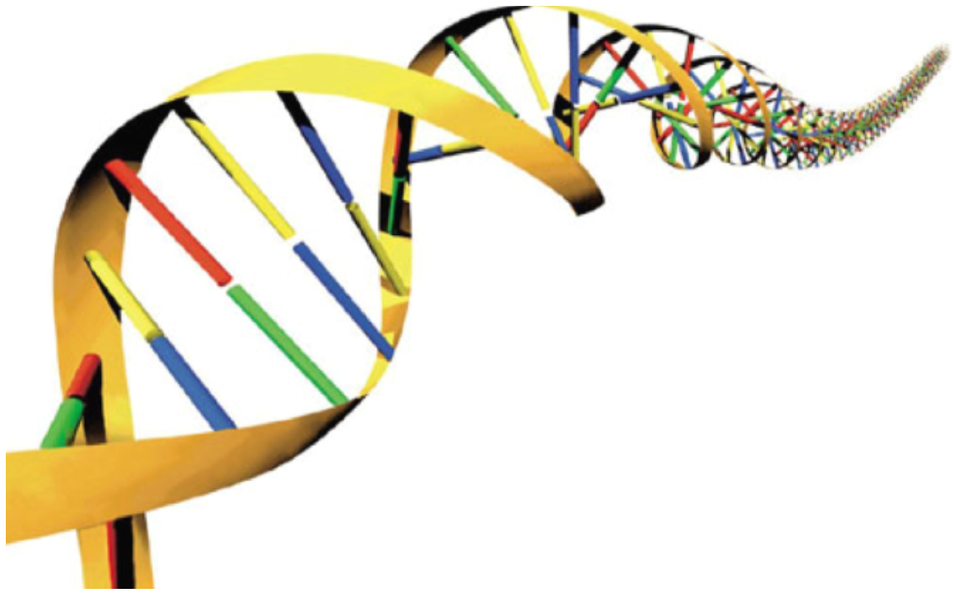
- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solver known in nature is:
  - The (human) brain that created "*the wheel, New York, wars and so on*" → Neural Networks
  - The evolutionary process that created the human brain → Evolutionary Computing



EVOLUTIONARY COMPUTATION

SIMILARITIES BETWEEN NATURAL AND ARTIFICIAL EVOLUTION

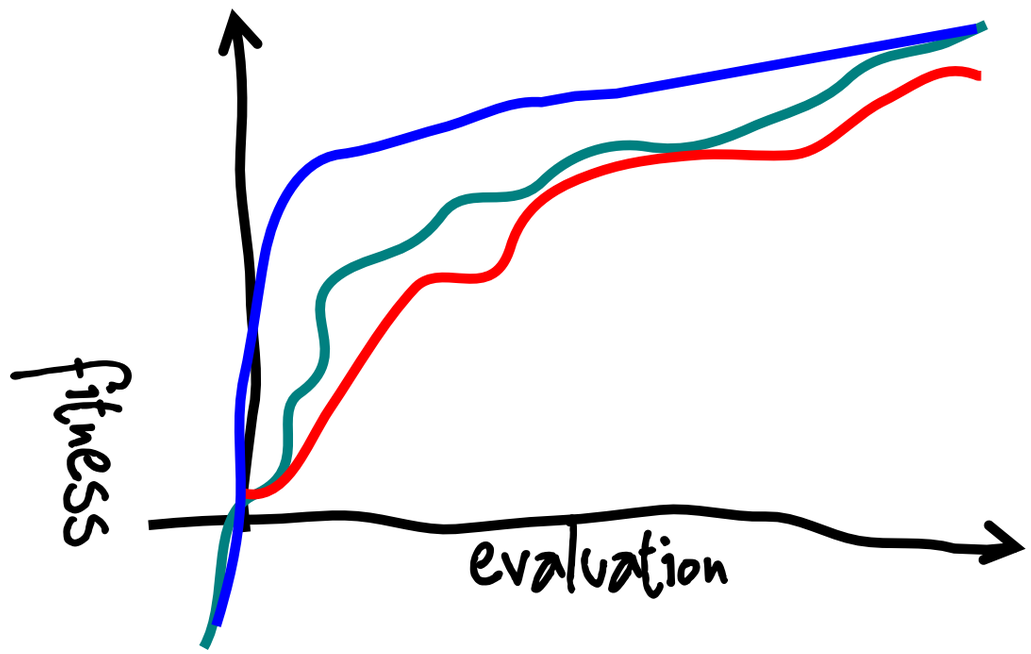
- **Individual**: encodes a potential (candidate) solution for a given problem problem: phenotype (computer program, object shape, electronic circuit, robot, etc.) + genotype (i.e., the genetic representation of the phenotype)
- **Population**: a set of individuals
- **Diversity**: a measure of how individuals in a population differ (at genotype or phenotype level)
- **Selection**: a mechanism to select which individuals "survive" and "reproduce"
- **Inheritance**: a mechanism to (partially) transmit the properties of a solution to another



EVOLUTIONARY COMPUTATION

DIFFERENCES BETWEEN NATURAL AND ARTIFICIAL EVOLUTION

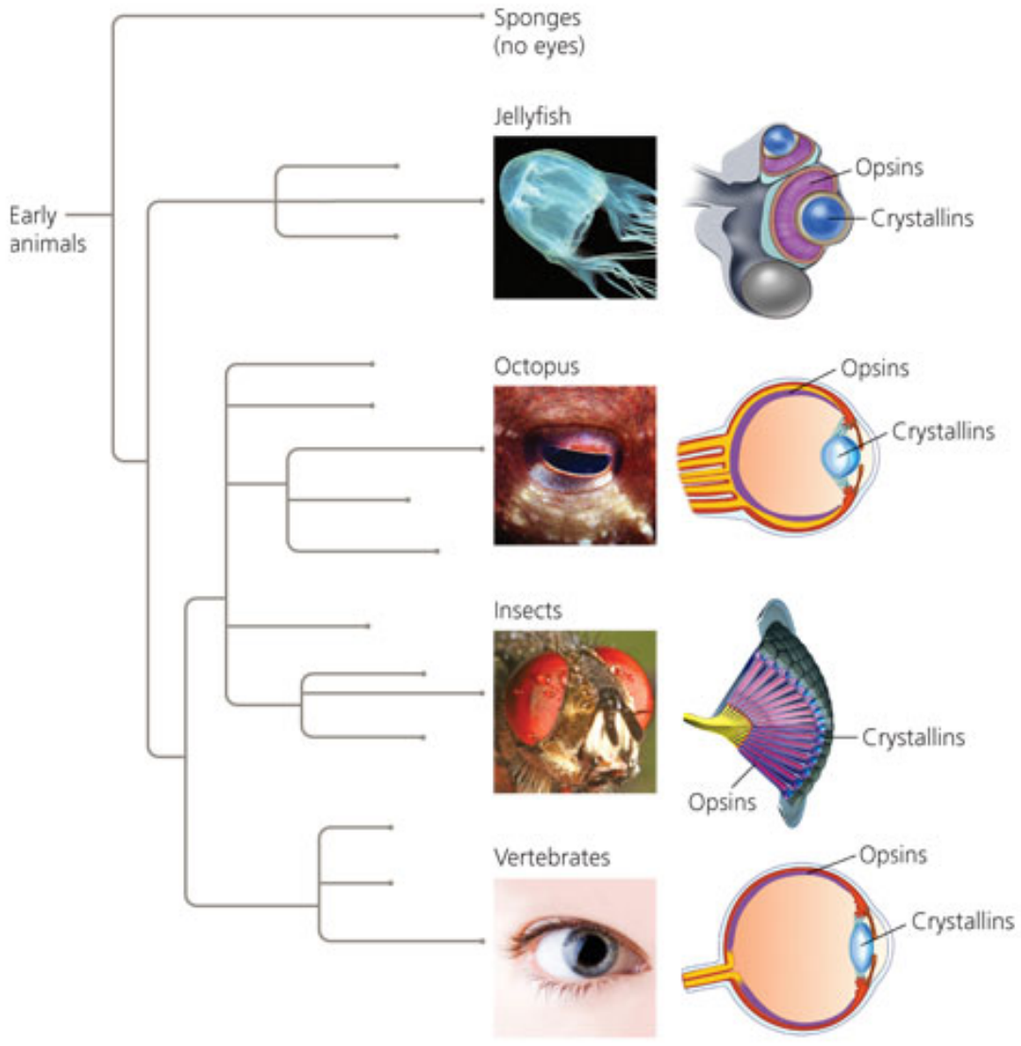
- Fitness is a measure of "how good" is a solution (not the no. of offspring)
- Selection is based on fitness (not on competition and interactions with environment)
- Generations are (typically) non-overlapping (parents and offspring do not exist at the same time)
- Expected improvement between initial and final solution (natural evolution is NOT an optimization process, i.e., it does NOT have a goal)



EVOLUTIONARY COMPUTATION

HOWEVER…

- When all variations are accumulated in one specific direction, the process resembles optimization!
- The final outcome may "look like" the result of an "intelligent design" (but it's NOT), e.g. the eye

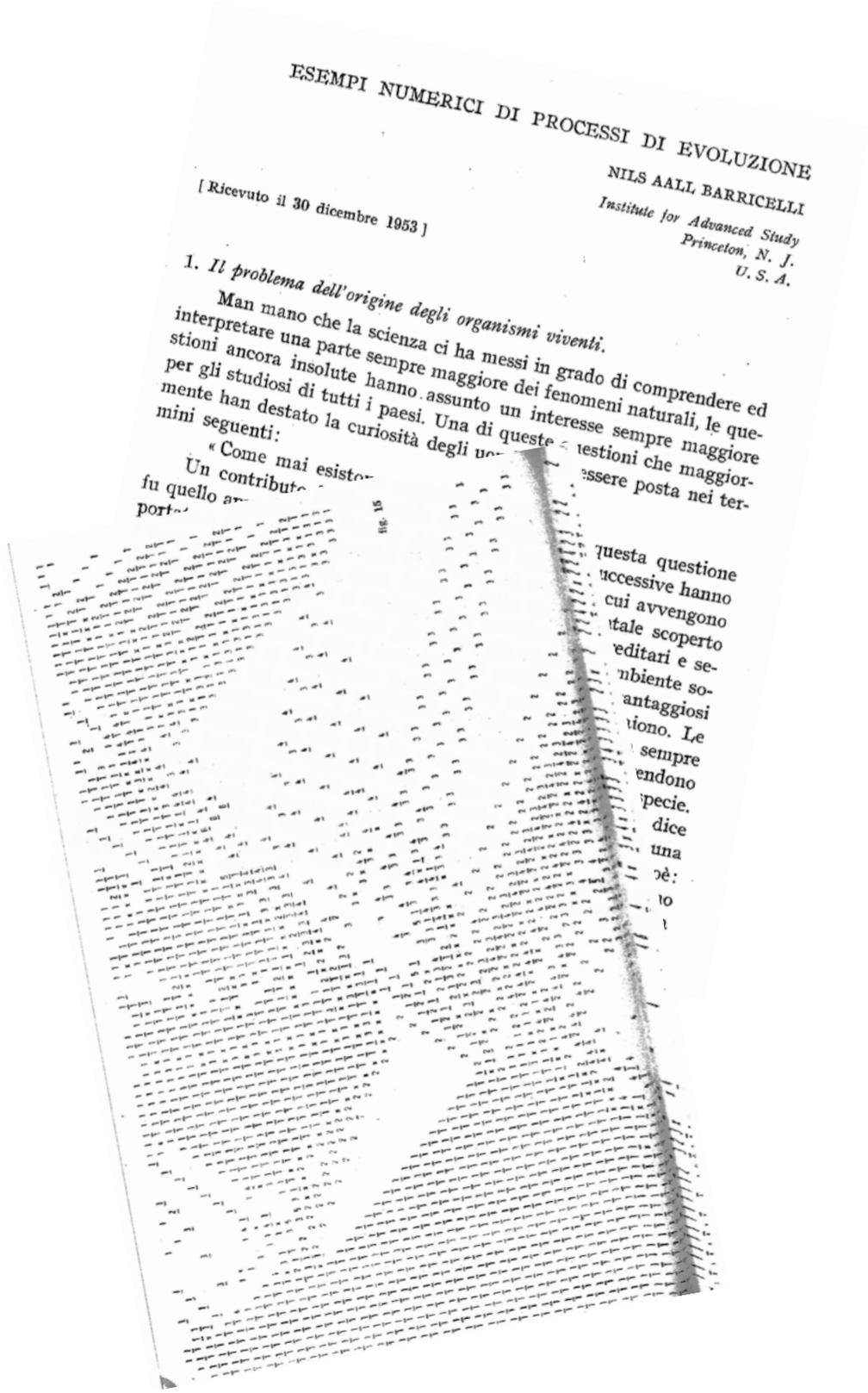


"Cosmos: A Spacetime Odyssey" by Neil deGrasse Tyson: https://www.youtube.com/watch?v=4SXHMm5I-68

EVOLUTIONARY COMPUTATION

A BIT OF HISTORY

- 1948: A. Turing mentions "genetical or evolutionary search" in a National Physical Laboratory internal report (published posthumous in 1968)
- 1954: N. Barricelli publishes his paper "Esempi Numerici di processi di evoluzione" considered nowadays the earliest published record of an evolutionary simulation.
- 1960s: real development of the field starts.
- Currently, a broad plethora of evolutionary algorithms exist (and ever more are created)!


ESEMPI NUMERICI DI PROCESSI DI EVOLUZIONE

NILS AALL BARRICELLI
Institute for Advanced Study
Princeton, N. J.
U. S. A.

[Ricevuto il 30 dicembre 1953]

1. *Il problema dell'origine degli organismi viventi.*

Man mano che la scienza ci ha messi in grado di comprendere ed interpretare una parte sempre maggiore dei fenomeni naturali, le questioni ancora insolute hanno assunto un interesse sempre maggiore per gli studiosi di tutti i paesi. Una di queste questioni che maggiormente han destato la curiosità degli [illegible] essere posta nei termini seguenti:

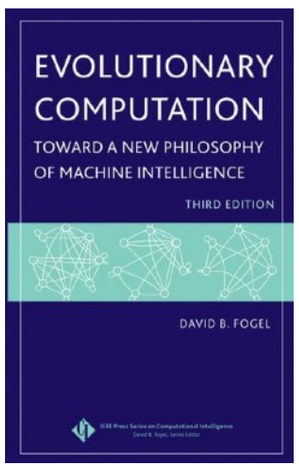
« Come mai esist[illegible]

Un contribut[illegible]

fu quello a[illegible]

port[illegible]

[illegible] questa questione successive hanno cui avvengono tale scoperto editari e sembiente sovantaggiosi tiono. Le sempre endono pecie. dice una è: to



evo*

EVOLUTIONARY COMPUTATION

ONE FRAMEWORK, DIFFERENT PARADIGMS

- **Evolutionary Programming (EP)** - Fogel et al., 1966
  Simulates adaptive behavior in evolution. Mutation only, with self-adaptation.

- **Evolution Strategies (ES)** - Rechenberg & Schwefel, 1973
  Similar to EP. Models the parameters controlling variation in evolution ("evolution of evolution").

- **Genetic Algorithm (GA)** - Holland, 1975
  Models genetic evolution. Binary genotypes, crossover and mutation.

- **Steady-State Evolution** – Whitley et al., 1988
  GA with gradual replacement: best individuals replace replace worst individuals in place.

- **Cultural evolution, or Memetic Algorithms (MA)** – Moscato et al., 1989
  Models the evolution of culture, allowing for competitive or cooperative co-evolution.

- **Genetic Programming (GP)** - Koza, 1992
  Based on GA, but individuals are tree-based representations of computer programs.

EVOLUTIONARY COMPUTATION

ONE FRAMEWORK, DIFFERENT PARADIGMS

- **Differential Evolution (DE)** – Storn & Price, 1995
  Similar to GA, differing in the reproduction mechanism used.

- **Estimation of Distribution Algorithm (EDA)** – Various authors, mid 1990s
  Similar to GA, but uses a probabilistic model of the solutions instead of an actual population.

- **Island Models** – Whitley et al., 1998
  GA with parallel evolving populations and occasional migration of individuals.

- **Covariance Matrix Adaptation Evolution Strategy: CMA-ES** – Hansen & Ostermeier, 2001
  An advanced ES, the current state-of-the-art for single-objective continuous optimization.

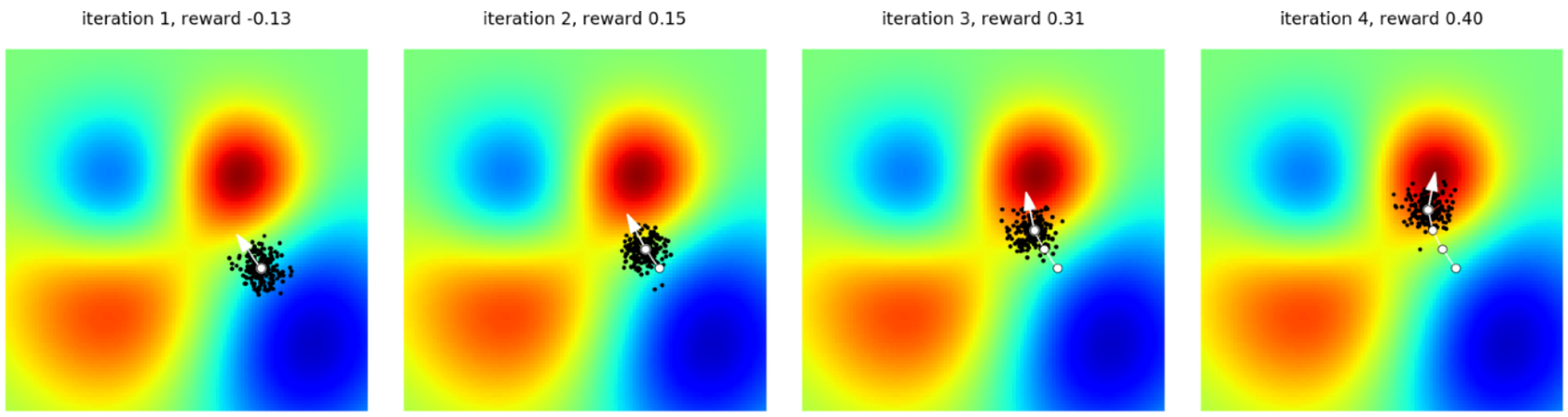
- **Non-Dominated Sorting Genetic Algorithm: NSGA-2** – Deb et al., 2002
  An advanced GA, the current state-of-the-art for multi-objective optimization.

EVOLUTIONARY COMPUTATION

EVOLUTIONARY ALGORITHM

A stochastic population-based metaheuristic that uses some mechanisms inspired by biological evolution:

- Reproduction
- Inheritance
- Mutation
- Selection



REQUIREMENTS
– (a lot of) simulations → natural evolution works over *millions* of generations!
– (inexpensive) computer technology → needed for running thousands of simulations

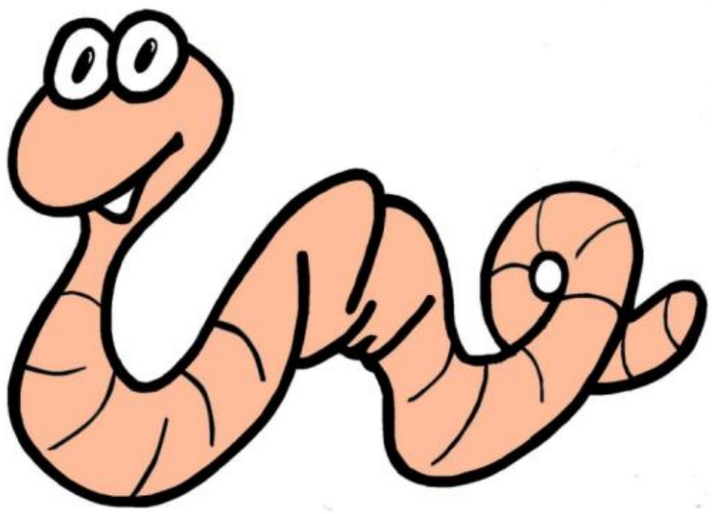
EVOLUTIONARY COMPUTATION

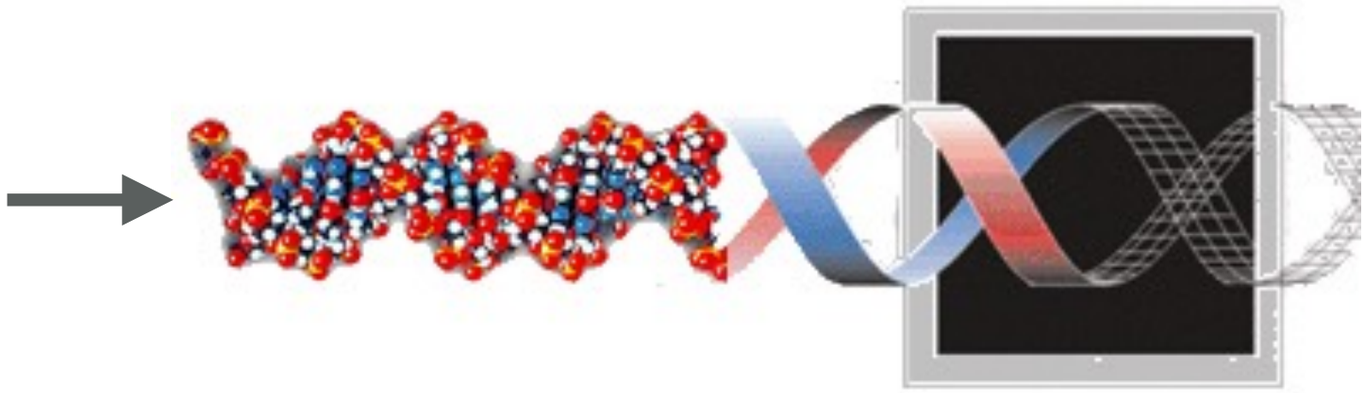
THE KEY ELEMENTS: INDIVIDUAL & FITNESS

An **individual** encodes a potential solution for a given problem, e.g.:
– A list of real numbers
– A sequence of cities
– A bit string
– Etc.

NOTE: mixed representations are also possible! E.g. a part of the genotype is binary, another real-valued, etc.

Each individual has a **fitness**, that is a metric of how good that solution is for a specific problem.





EVOLUTIONARY COMPUTATION

A GENERIC EVOLUTIONARY ALGORITHM FLOW

population

evaluation of fitness

parents selection

reproduction
(genetic operators)

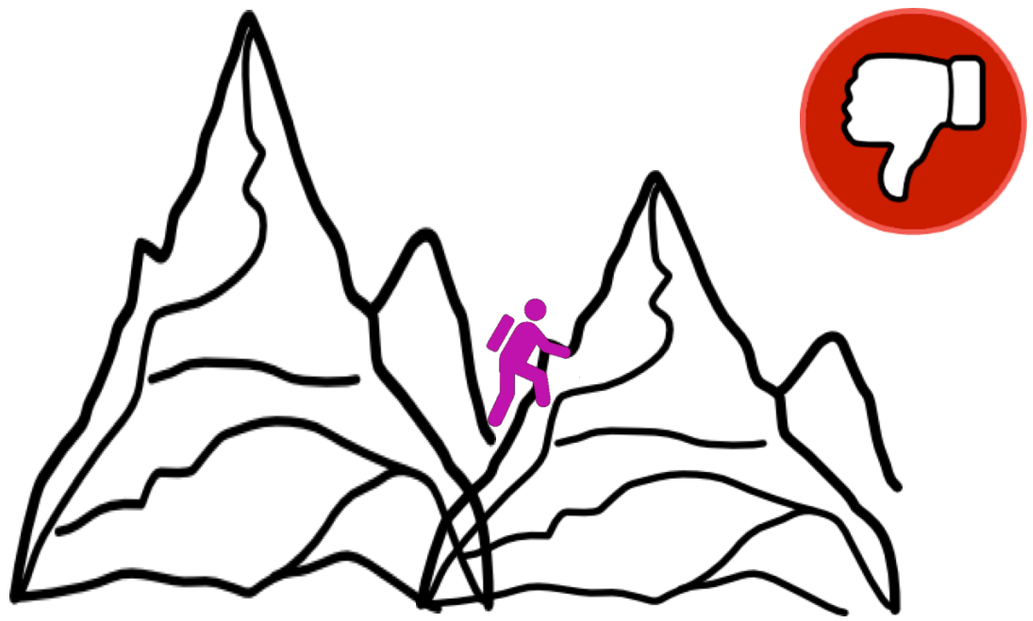
offspring
(new individuals)

slaughtering/replacement

EVOLUTIONARY COMPUTATION

WHY EVOLUTIONARY ALGORITHMS WORK

- Classic optimization algorithms ("local search" methods) work on a single solution at a time, perturbed according to some logics
- They can be very good at exploitation (fast convergence to local optimum close to the starting point), but not at exploration (they may miss the global optimum)
- They are not parallelizable



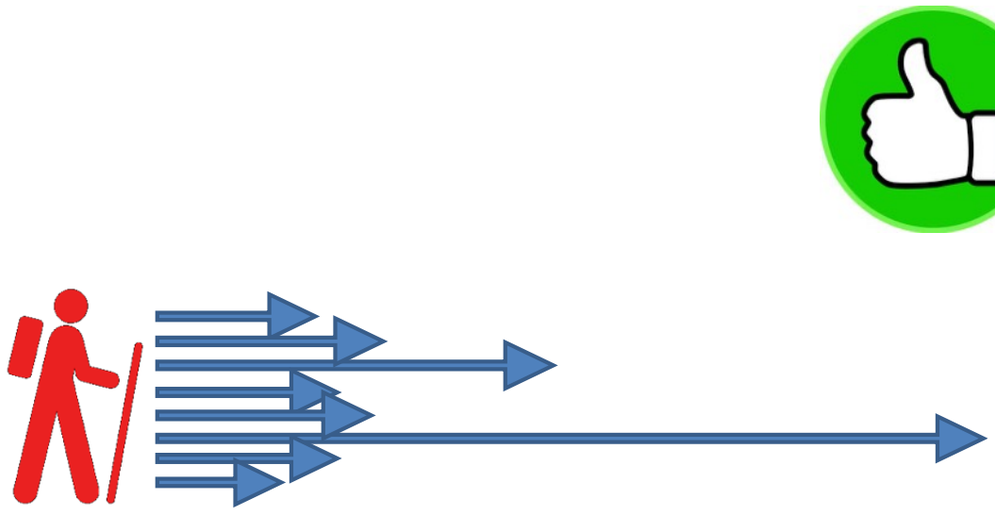
single-solution algorithms: like
a single "hill-climber

EVOLUTIONARY COMPUTATION

WHY EVOLUTIONARY ALGORITHMS WORK

- They are inherently parallel(izable)
- They are both good at exploitation AND exploration
- Each solution may be perturbed differently (different search perspectives/search strategies)
- Interactions (i.e., exchange of information, e.g. crossover) among solutions can be beneficial

population-based algorithms:
multiple, parallel hill-climbers

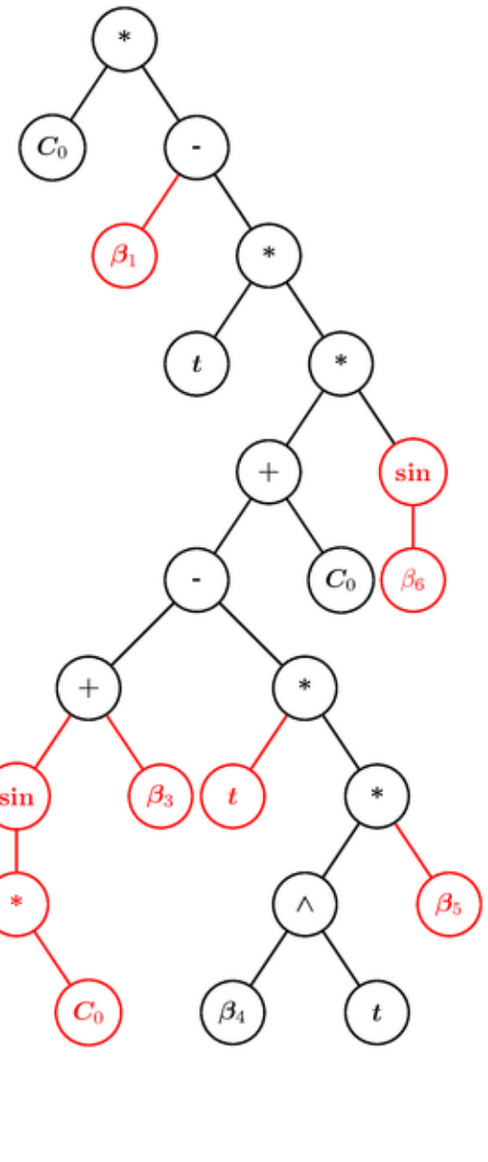


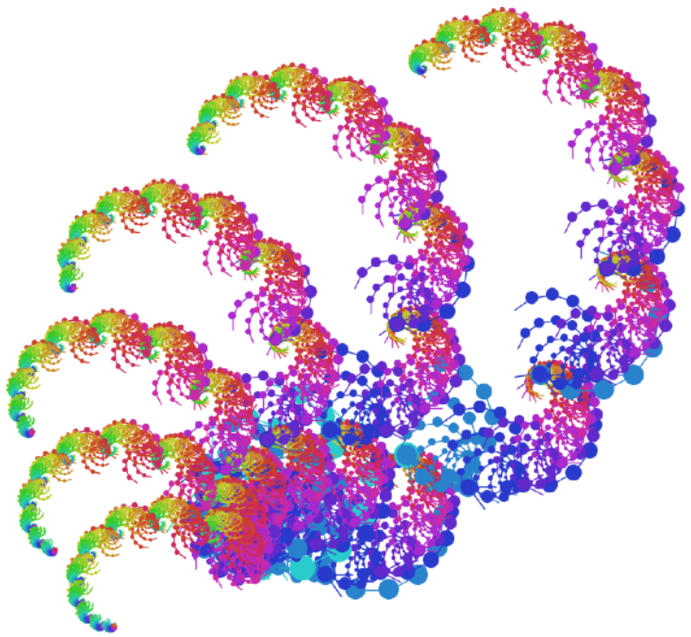
variable step-sizes

EVOLUTIONARY COMPUTATION

SOME APPLICATIONS

- Parameters optimization
- Finance/portfolio optimization
- Model & neural network training
- Forecast
- Scheduling
- Telecom/Networking
- CAD/CAE problems
- Database/Data mining
- Bioinformatics
- Bug identification
- Art (music, design, websites, etc.)
- ALife studies

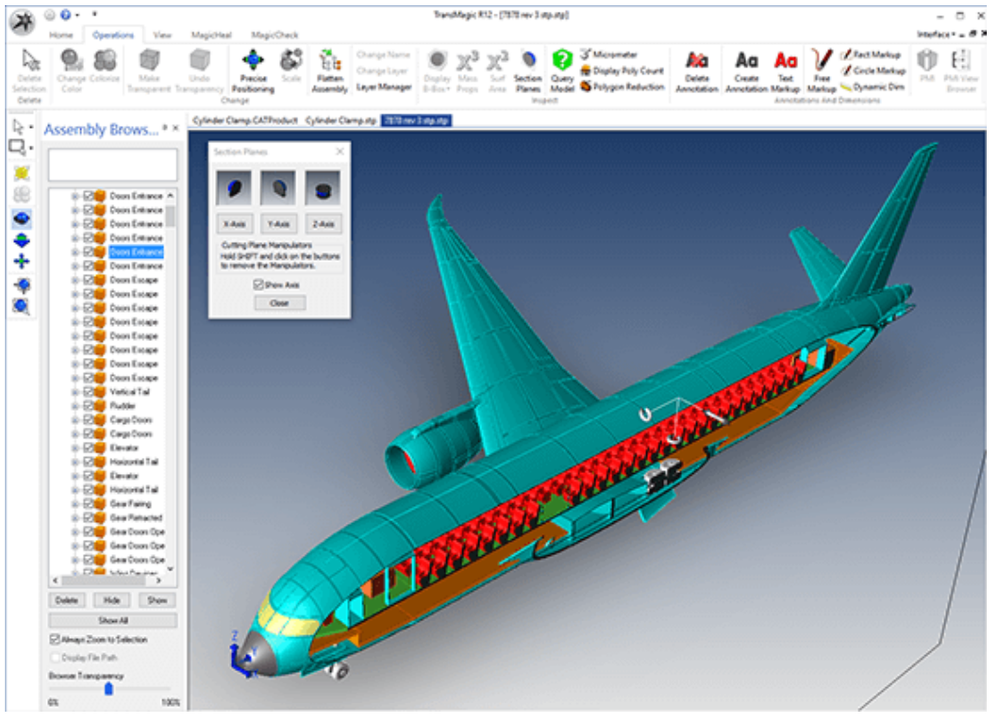


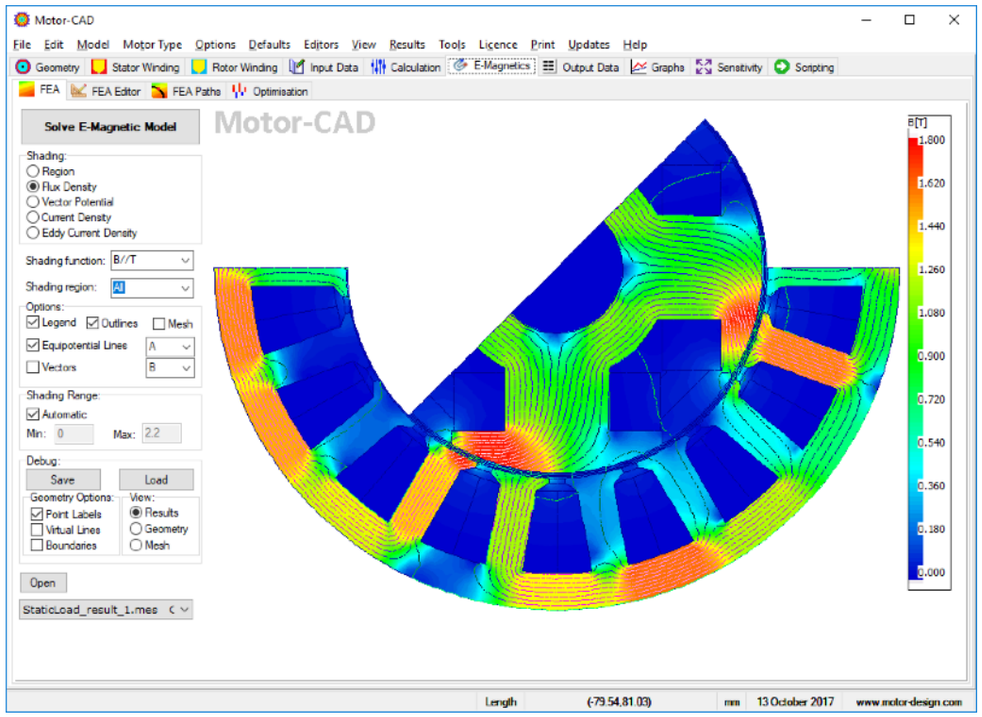


EVOLUTIONARY COMPUTATION

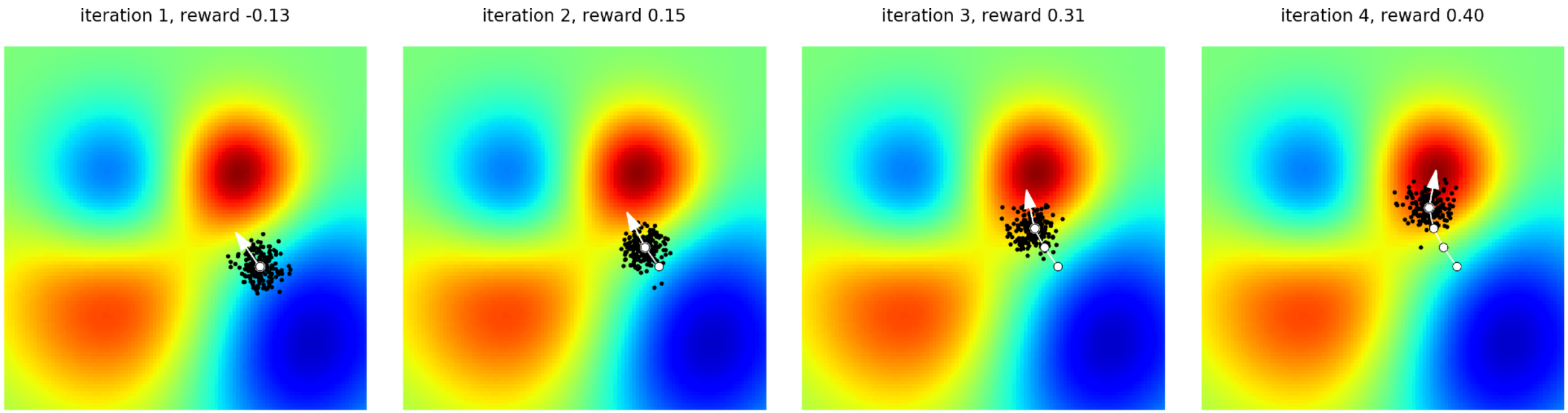
CAD/CAE PROBLEMS

- Computationally intensive
- Unknown fitness landscape
- Vast infeasible regions in solution space
- Additional non-linear constraints
- Practical limitations (e.g. limited HW, limited simulation SW licenses available, etc.)
- Difficulties in comparing infeasible/incomplete solutions
- No clear way to get good solutions (except exhaustive search)





Genetic Algorithms



GENETIC ALGORITHMS

EVOLUTIONARY ALGORITHMS: CHECK LIST

1. Devise genetic representation
2. Build a population
3. Design a fitness function
4. Choose selection method
5. Choose replacement method
6. Choose crossover & mutation
7. Choose data analysis method
8. Repeat generation cycle until
   - maximum fitness value is found
   - solution found is "good enough"
   - maximum wall-clock time is reached
   - a certain convergence condition is met

Evolutionary algorithms are applicable to any problem domain as long as suitable genetic representation, fitness, and genetic operators are chosen.

GENETIC ALGORITHMS

GENETIC REPRESENTATION

- Describes elements of genotype and mapping to phenotype
- Must match genetic operators of recombination and mutation
- Set of possible genotypes should include optimal solution to the problem

- Choice of representation benefits from domain knowledge:
  - Encoding of relevant parameters
  - Appropriate accuracy
  - Completeness: in order to find the global optimum, every feasible solution must be represented in genotype space

- Great simplification of genetics:
  - Single-stranded sequence of symbols (e.g., binary)
  - Often fixed-length, only genic "DNA"
  - Often one chromosome, with haploid structure (one copy only of the chromosome)
  - Often one-to-one direct correspondence between gene and parameter
  - Gene expression and genetic regulation used only in specific situations

GENETIC ALGORITHMS

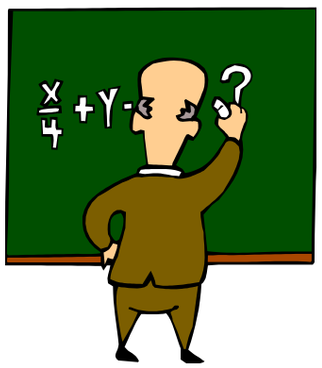
DISCRETE REPRESENTATIONS

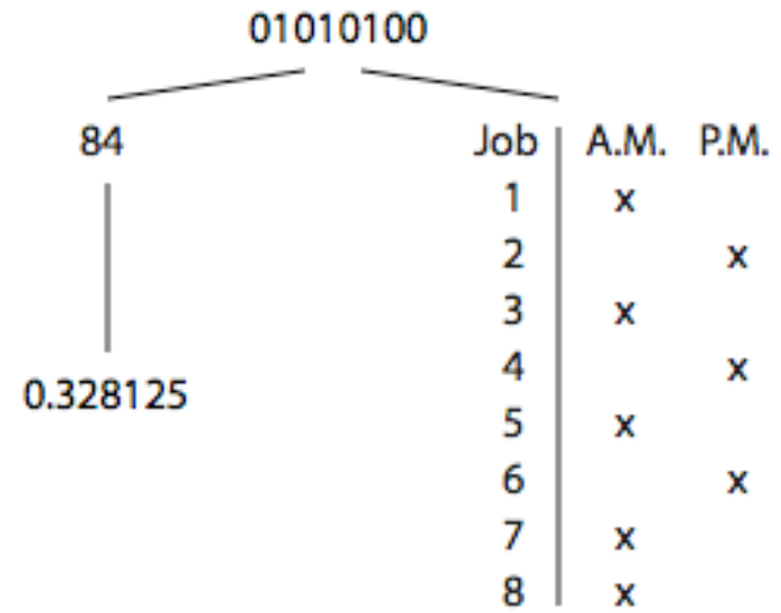
The genotype is a sequence (or an array of sequences) of *n* discrete symbols drawn from alphabet with cardinality *k*.

E.g., a binary string of 8 digits (*n*=8, *k*=2), such as 01010100, can be mapped into different phenotypes, depending on the optimization problem (just to give some examples):

1. To integer *i* using binary code

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





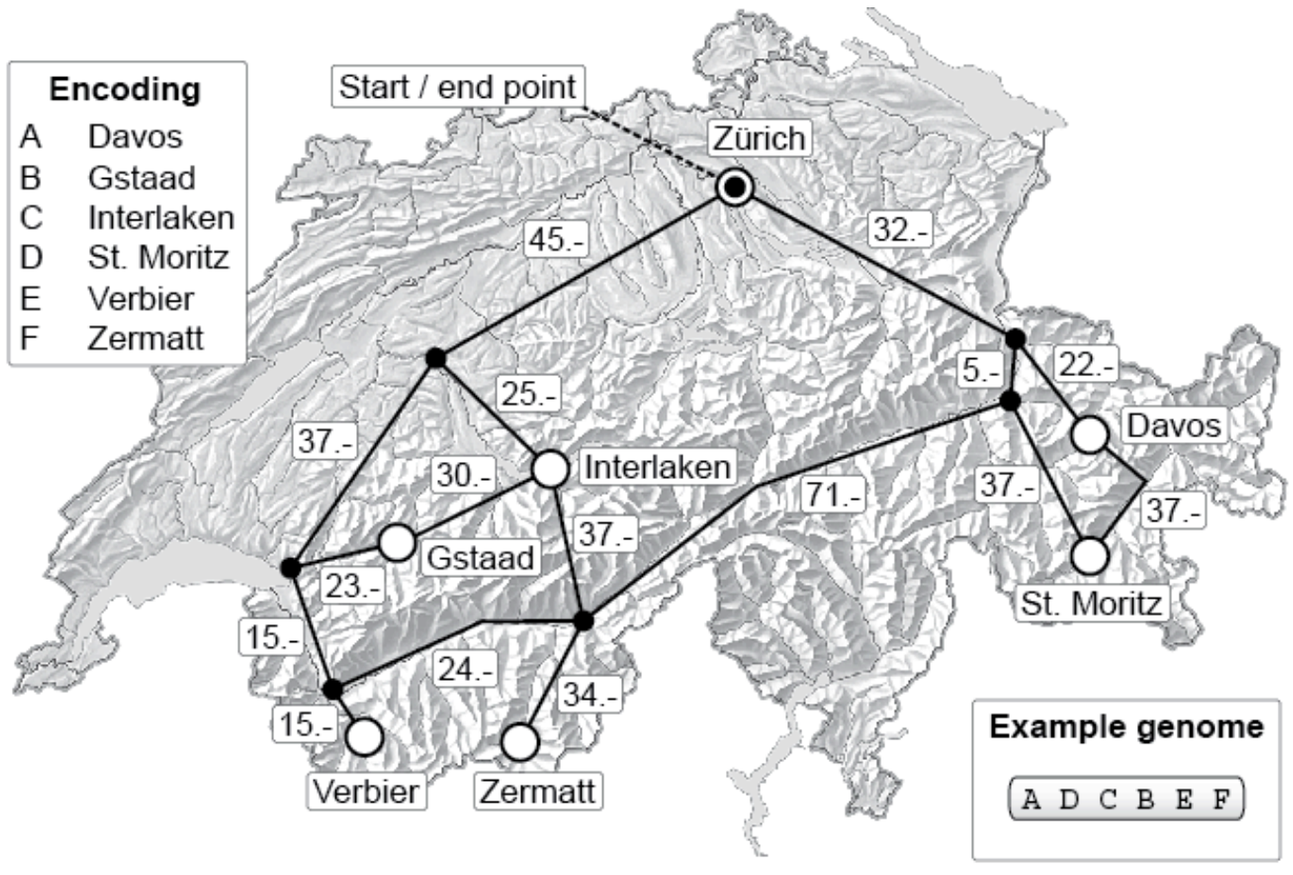
3. A binary assignment, such as a job schedule problem
- job=gene position
- time=gene value

GENETIC ALGORITHMS

SEQUENCE REPRESENTATIONS

It is a particular case of discrete representation used e.g. for Traveling Salesman Problems (TSP), i.e., plan a path to visit *n* cities under some constraints, and problems alike. In this case the individual is a permutation of *n* different symbols (e.g. integers, letters, or labels), each of which occurs exactly once.

E.g., planning ski holidays with lowest transportation costs:

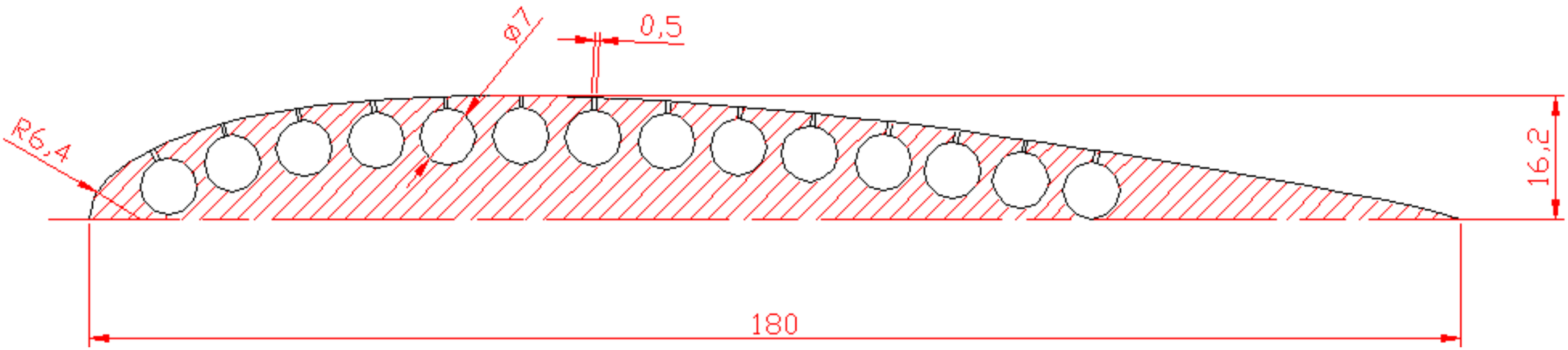


GENETIC ALGORITHMS

REAL-VALUED REPRESENTATIONS

The genotype is a sequence of real values that represent the problem parameters.

- Used when high-precision parameter optimization is required
- For example, genetic encoding of wing profile for shape optimization:



Evolvable wing made of deformable material with pressure tubes. In this case the genotype is a vector representing the pressure values of 14 tubes.

GENETIC ALGORITHMS

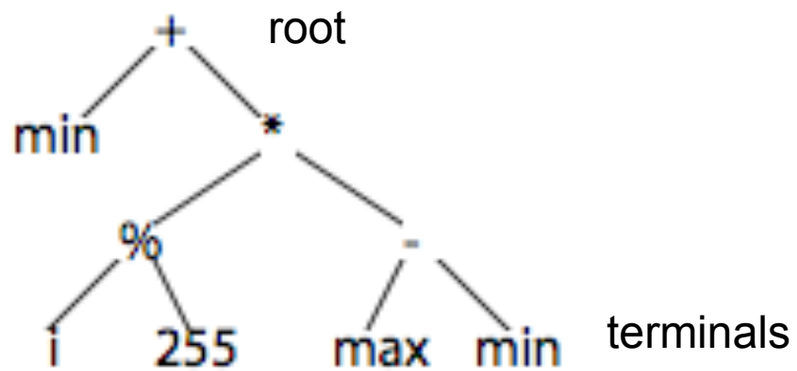
TREE REPRESENTATIONS

The genotype describes a tree with branching points and terminals.

- Suitable for encoding hierarchical structures
- E.g., used to encode computer programs (Genetic Programming, GP) made of:
  - Operators ("Function set": multiplication, If-Then, Log, etc.)
  - Operands ("Terminal set": constants, variables, sensor readings, etc.)

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

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



REQUIREMENTS FOR GP

- Closure: all the "encodable" functions must accept as arguments all terminals in Terminal set, and all possible outputs of all functions in Function set (e.g., protected division %)
- Sufficiency: elements in Function and Terminal sets must be sufficient to generate the program that solves the given problem

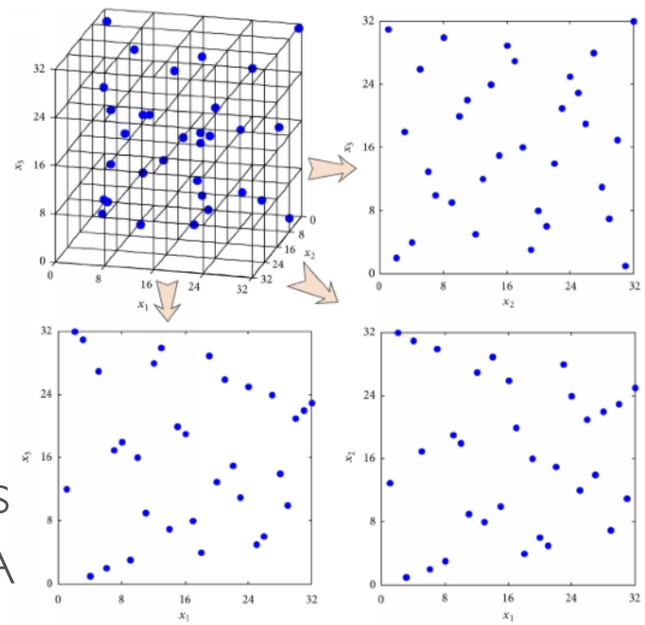
GENETIC ALGORITHMS

INITIAL POPULATION

Should be sufficiently large to cover the search space (!), but sufficiently small in terms of evaluations (typical size: between 10s and 1000s individuals).

Uniform sample of search space:

- Binary strings: 0 or 1 with probability 0.5 (similarly for other discrete representations)
- Real-valued representations: uniform on a given interval if bounded phenotype (e.g., +2.0, -2.0), otherwise "best guess" (based on domain knowledge). Alternatively, use a DoE (Design of Experiments) algorithm.
- Trees are built recursively starting from root: root is randomly chosen from Function set; for every branch, randomly choose among all elements of Function set and of Terminal set; if terminal is chosen, it becomes leaf. A maximum tree depth must be chosen.



IMPORTANT: hand-designed genotypes may cause a loss of genetic diversity (see later) and/or an unrecoverable bias in the evolutionary process!

GENETIC ALGORITHMS

FITNESS FUNCTION = OBJECTIVE FUNCTION

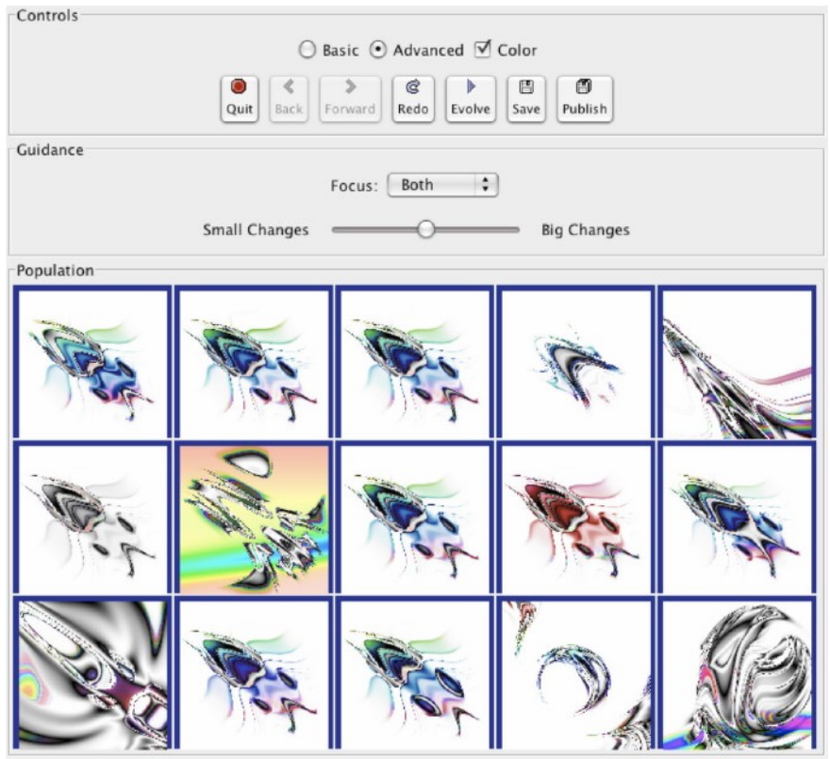
Evaluates the performance of a phenotype with (one or more) numerical scores.

- Choice of components; e.g., lift and drag of a wing
- Combination of components; e.g. (lift + 1.0/drag) or (lift - drag)
- Extensive test of each phenotype (with noise, repeated evaluations are needed)
- The more the fitness function can discriminate (return different values), the better
- Warning! "You Get What You Evaluate"

AN INTERESTING CONCEPT

Subjective fitness: select phenotype by human inspection (basis of IEC, Interactive Evolution Computation)

- Used when aesthetic properties cannot be quantified objectively (e.g. visual art, music, design, etc.)
- Can be combined with an objective fitness function
- E.g., Picbreeder http://picbreeder.org/



GENETIC ALGORITHMS

(PARENT) SELECTION

A method to make sure that better individuals make comparatively more offspring.

- Used in breeding (pedigreed dogs, horses, etc.)
- **Selection pressure** is inversely proportional to no. of selected individuals
- High selection pressure = rapid loss of diversity and premature convergence
- Important that also less performing individuals can reproduce to some extent

GENETIC ALGORITHMS

RANDOM SELECTION

- Each individual has a probability of $1/N$ to be selected, where N is the population size
- No fitness information is used → actually, never used in EA!
- Lowest selection pressure among all possible selection schemes

GENETIC ALGORITHMS

FITNESS-PROPORTIONATE SELECTION (ROULETTE-WHEEL SELECTION)

The probability that an individual makes an offspring is proportional to how good its fitness is with respect to the population fitness: $p(i) = f(i)/\Sigma f(i)$, i is the individual index
→ **Biases selection towards the most-fit individuals!**

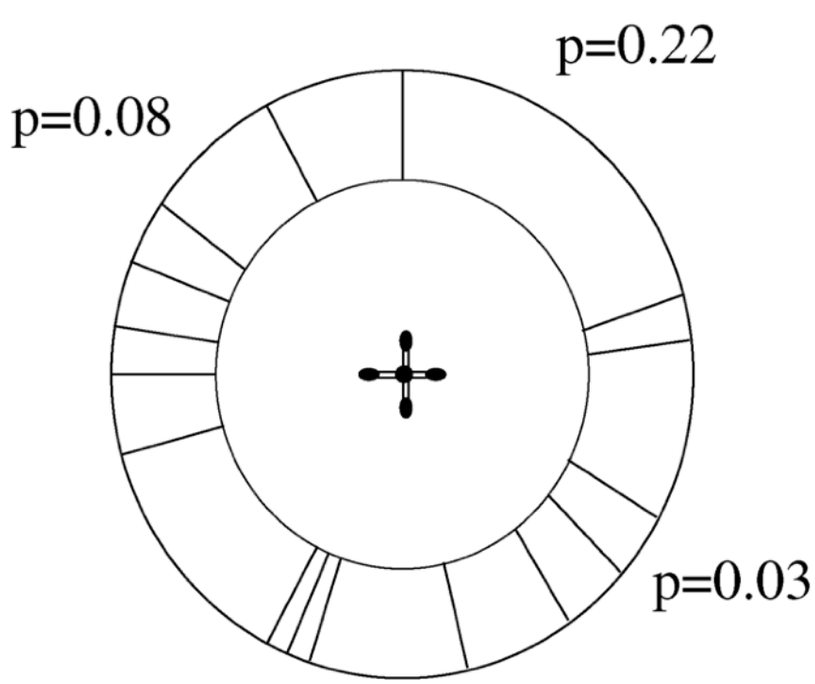
Problems:
- Fitness must be non-negative (otherwise must be shifted to positive values)
- Uniform fitness values (values are too close) = random selection
- Few high-fitness individuals = high selection pressure (low-fitness individuals have close-to-zero chance of reproduction)

GENETIC ALGORITHMS

RANK-BASED SELECTION

- Individuals are sorted on their fitness value from best to worst. The place in this sorted list is called the **rank** *r* (rank=1 for best, rank=N-1 for worst, being N the population size).
- Instead of using the fitness value of an individual, the rank is used to select individuals:
  $p(i) = 1\text{-}r(i)/\Sigma r(i)$ ⟶ lower selection pressure w.r.t. fitness-proportionate selection
- Roulette wheel on rank-based $p(i)$

individual	fitness	rank
A	5	5
B	7	3
C	8	2
D	2	8
E	3	7
F	9	1
G	7	4
H	4	6



GENETIC ALGORITHMS

TRUNCATED RANK-BASED SELECTION

- Only the best *x* individuals are allowed to make offspring and each of them makes the same number of offspring: *N*/*x*, where *N* is the population size.
- E.g., in a population of 100 individuals, make 5 copies of 20 best individuals

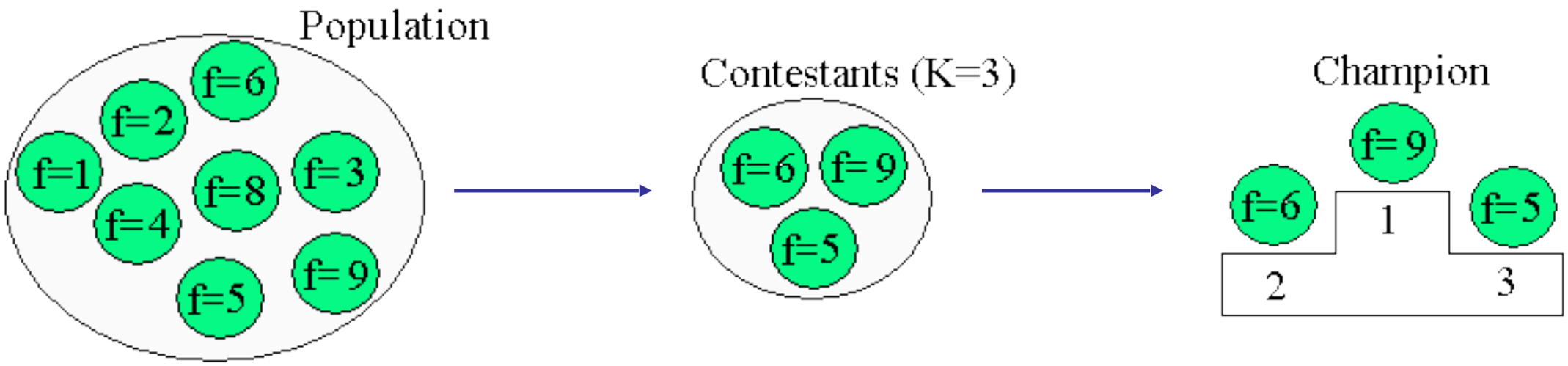
individual	fitness	rank	list
A	5	5	F
B	7	3	C
C	8	2	B
D	2	8	G
E	3	7	A
F	9	1	H
G	7	4	E
H	4	6	D

GENETIC ALGORITHMS

TOURNAMENT SELECTION

For every offspring to be generated:

1. Pick randomly *k* individuals from the population, where *k* is the tournament size < *N*, *N* being the population size (larger k = larger selection pressure, i.e., individual have a higher chance to compete against individuals with higher fitness)
2. Choose the individual with the highest fitness and make a copy
3. Put all individuals back in the population



GENETIC ALGORITHMS

REPLACEMENT (OR "SURVIVOR SELECTION")

- **Generational replacement**: old population is entirely replaced by offspring (also called "aged-based" replacement, or "non-overlapping" replacement, most frequent method)

- **Generational rollover**: insert offspring "in place", i.e., replace worse individuals in the current generation. A special case is the **Steady-State scheme** (Whitley et al., 1988): one offspring is generated per generation, one member of the population (the worst one) is replaced.

- **Elitism**: maintain *n* best individuals from previous generation to prevent loss of best individuals by effects of mutations or sub-optimal fitness evaluation (the higher *n*, the less diversity)

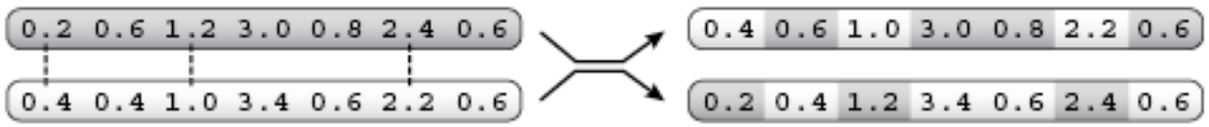
GENETIC ALGORITHMS

RECOMBINATION (CROSSOVER)

- Emulates recombination of genetic material from two parents during meiosis
- **Exploitation** of synergy of sub-solutions (building blocks) from parents
- Applied to randomly paired offspring with a given probability *Pc*

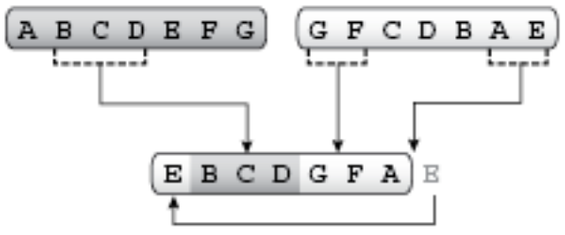
One point

Uniform

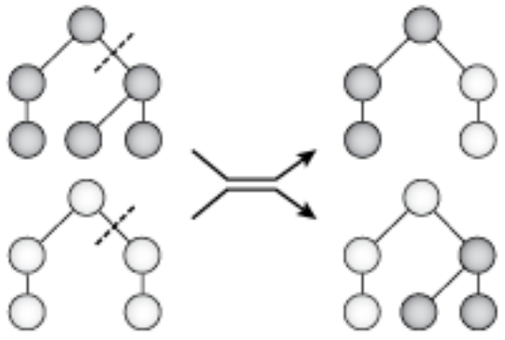


Arithmetic

For sequences



For trees

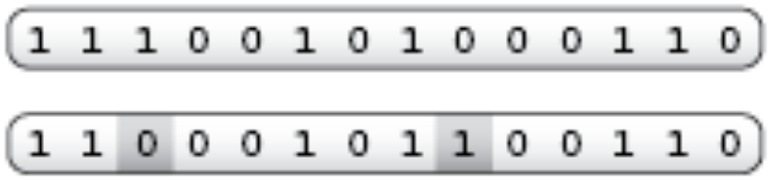


GENETIC ALGORITHMS

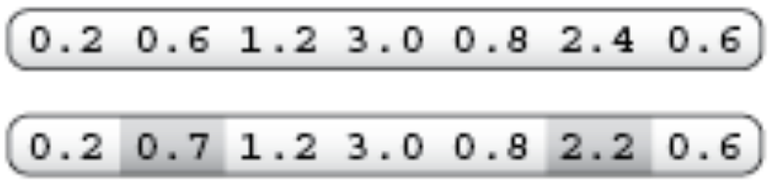
MUTATION

- Emulates genetic mutations
- **Exploration** of variation of existing solutions
- Applied to each gene in the genotype with a given probability *Pm*

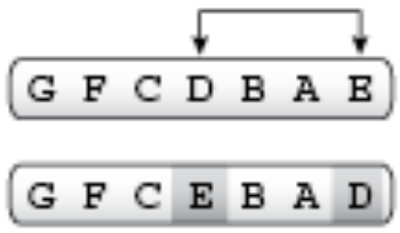
Binary genotypes



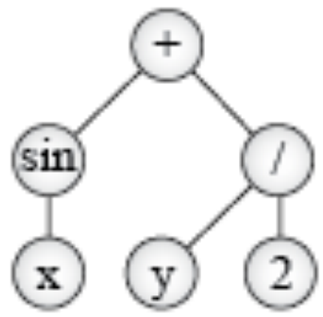
Real-valued genotypes

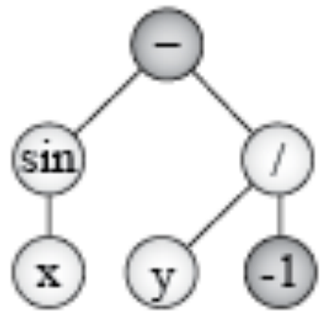


Sequence genotypes



For trees

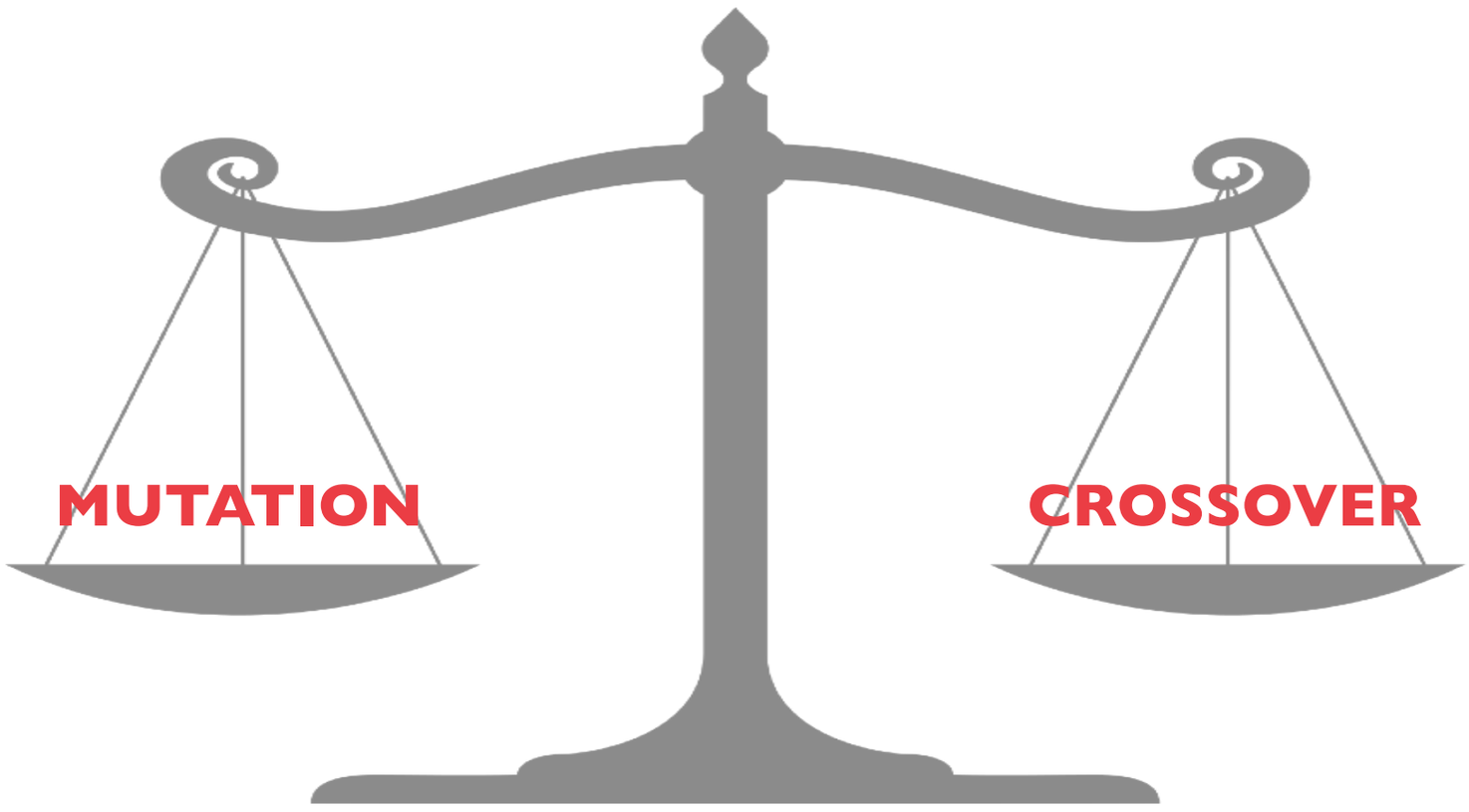




GENETIC ALGORITHMS

CROSSOVER OR MUTATION?

- A long debate: which one is better/necessary?
- Answer (at least, rather wide agreement): it depends on the problem, but, in general, it is good to have both! Mutation provides exploration, while crossover provides exploitation capabilities.
- While mutation-only-EA is possible, crossover-only- EA would not work (no introduction of new genetic material in the population).



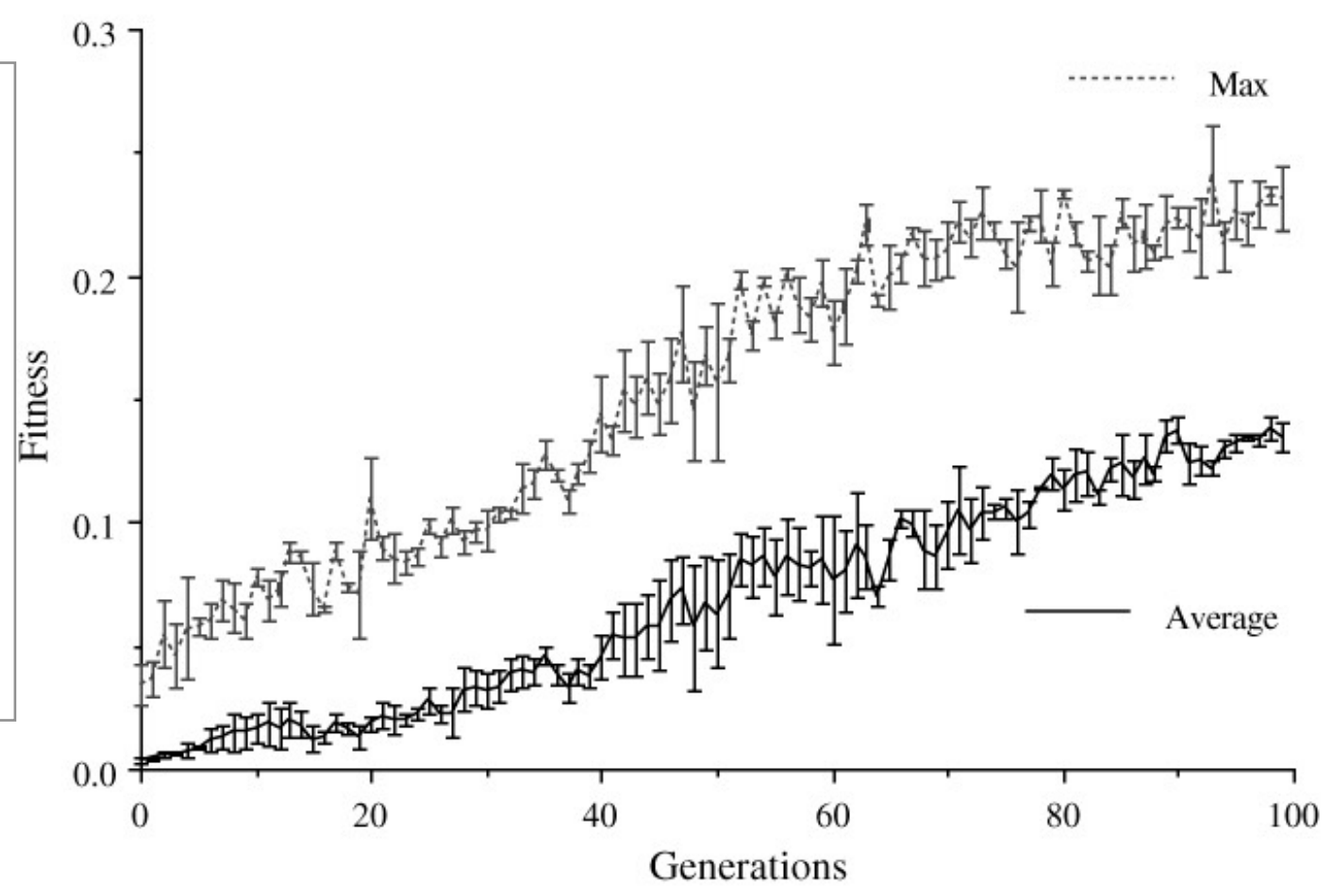
GENETIC ALGORITHMS

DATA ANALYSIS: MONITORING PERFORMANCE

- Track best/worst and/or population average fitness (+/- std. dev.) of each generation
- Multiple runs are necessary ⟶ plot average data and standard error ("anytime behavior")
- Fitness graphs (also called "fitness trends") are meaningful only if the problem is stationary, i.e., the fitness function does not change over time
- These plots can be used to detect if the algorithm stagnated or (prematurely) converged

Stagnation: no further evolution possible even if the population remains diverse (individuals are far from each other, but new individuals are worse).

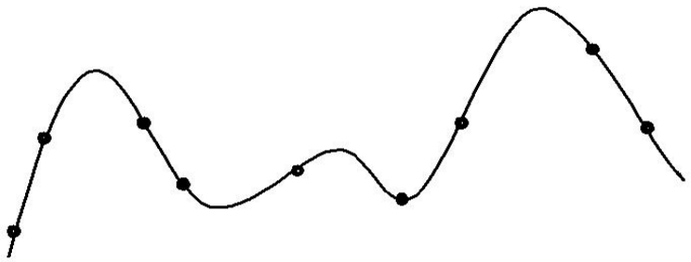
Premature convergence: no further evolution possible since the population lost diversity (individuals are too close to each other.)



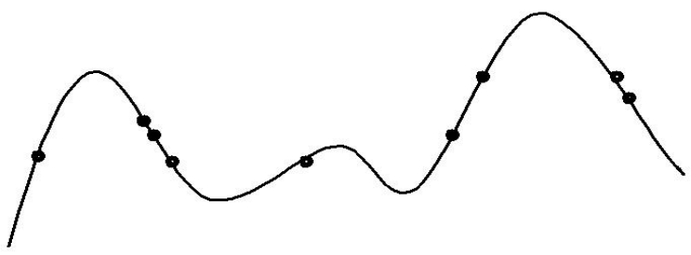
GENETIC ALGORITHMS

DATA ANALYSIS: MONITORING PERFORMANCE

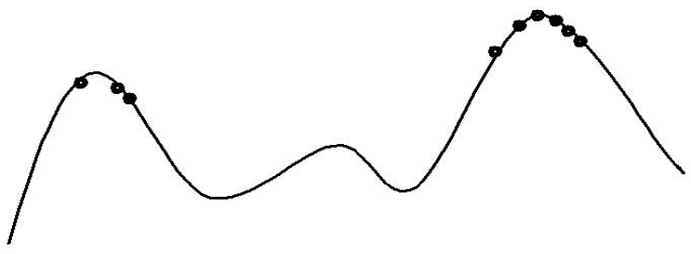
- What is the typical behavior of an EA?
- Can we identify some phases in an evolutionary run?



Early phase:
quasi-random population distribution



Mid-phase:
population arranged around/on hills



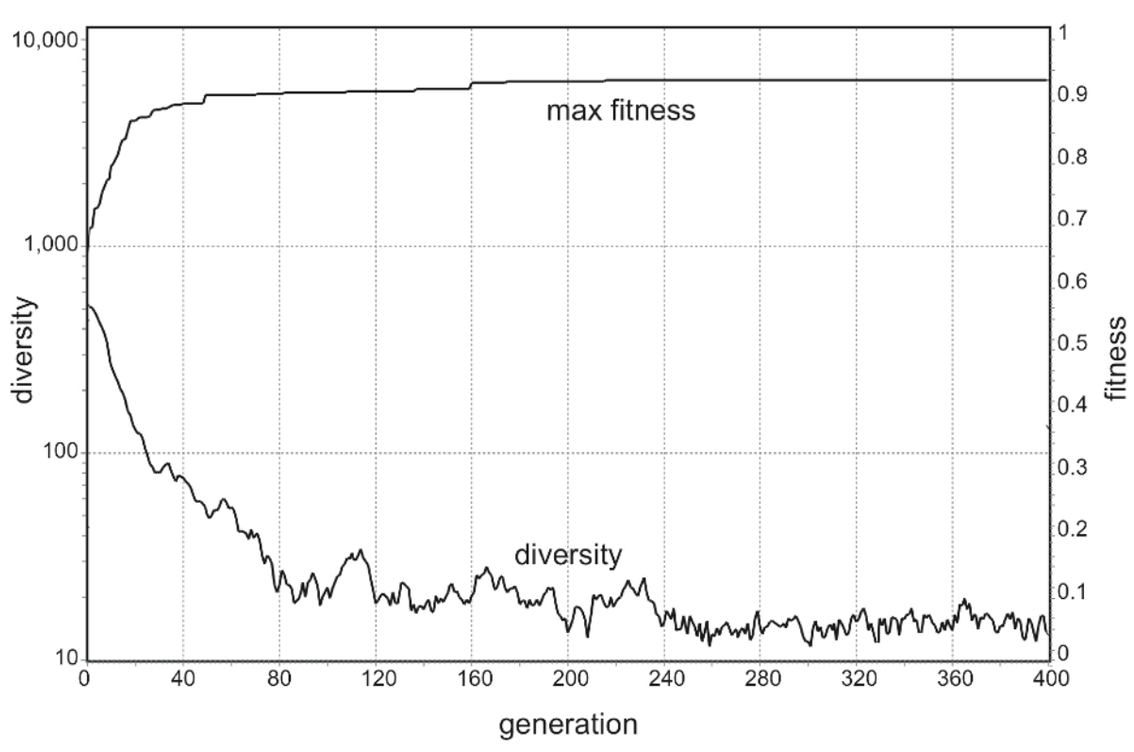
Late phase:
population concentrated on high hills

GENETIC ALGORITHMS

DATA ANALYSIS: MEASURING DIVERSITY

- Diversity tells whether the population has potential for further evolution
- Measures of diversity depend on genetic representation
- E.g., for real and discrete values, sum of Hamming and Euclidean distances, respectively:

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



Swarm Intelligence

SWARM INTELLIGENCE

IMAGINE A TREASURE HUNT GAME

- **Who?**
  You and a group of friends
- **What do you know?**
  The area of the treasure (e.g. an island), but not exactly where it is located
- **What do you have?**
  - A metal detector
  - Signal strength and position of your neighbors' metal detectors
  - Your friends!
- **What is your goal?**

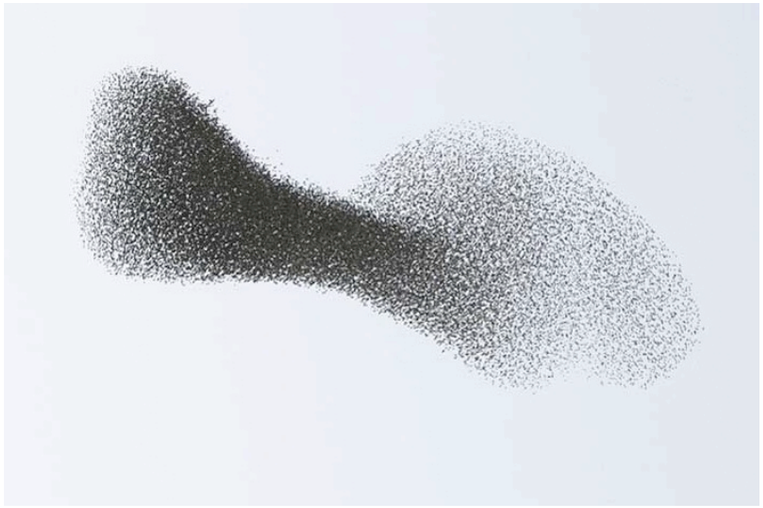
To find that treasure, or at least part of it

- **Reward sharing**
  You and your friends have reached the following agreement: all who have taken part in the search will be rewarded, but the person who finds the treasure will get a higher reward than all others, and the rest will be rewarded based on their distance from the treasure at the time when the treasure is found.
- **What would you do?**
  - Ignore your friends?
  - Use the information from your neighboring friends?

SWARM INTELLIGENCE

WHAT IS SWARM INTELLIGENCE?

Some "social" animals live and operate in groups. When these animals perform some tasks together can make things that wouldn't be able to do when they are alone (showing a form of **collective intelligence**, as opposed to individual intelligence). E.g.: foraging, defending from predators, building complex structures, etc.



Swarm: a group of "simple" agents (not necessarily animals) that communicate with each other (either directly or indirectly), by changing their own state or acting on their local environment.

Swarm Intelligence: the property of a system whereby the collective behaviors of "simple" agents interacting locally cause coherent functional global patterns to emerge (**emergent behavior**).

Main principles

- Agents perceive and act based only on local information
- Agents cooperate by means of local information
- Information propagates through the entire swarm
- This interaction results in distributed collective problem-solving

SWARM INTELLIGENCE

EMERGENT BEHAVIOR (WITHOUT LEADER)

Some examples observed in nature:

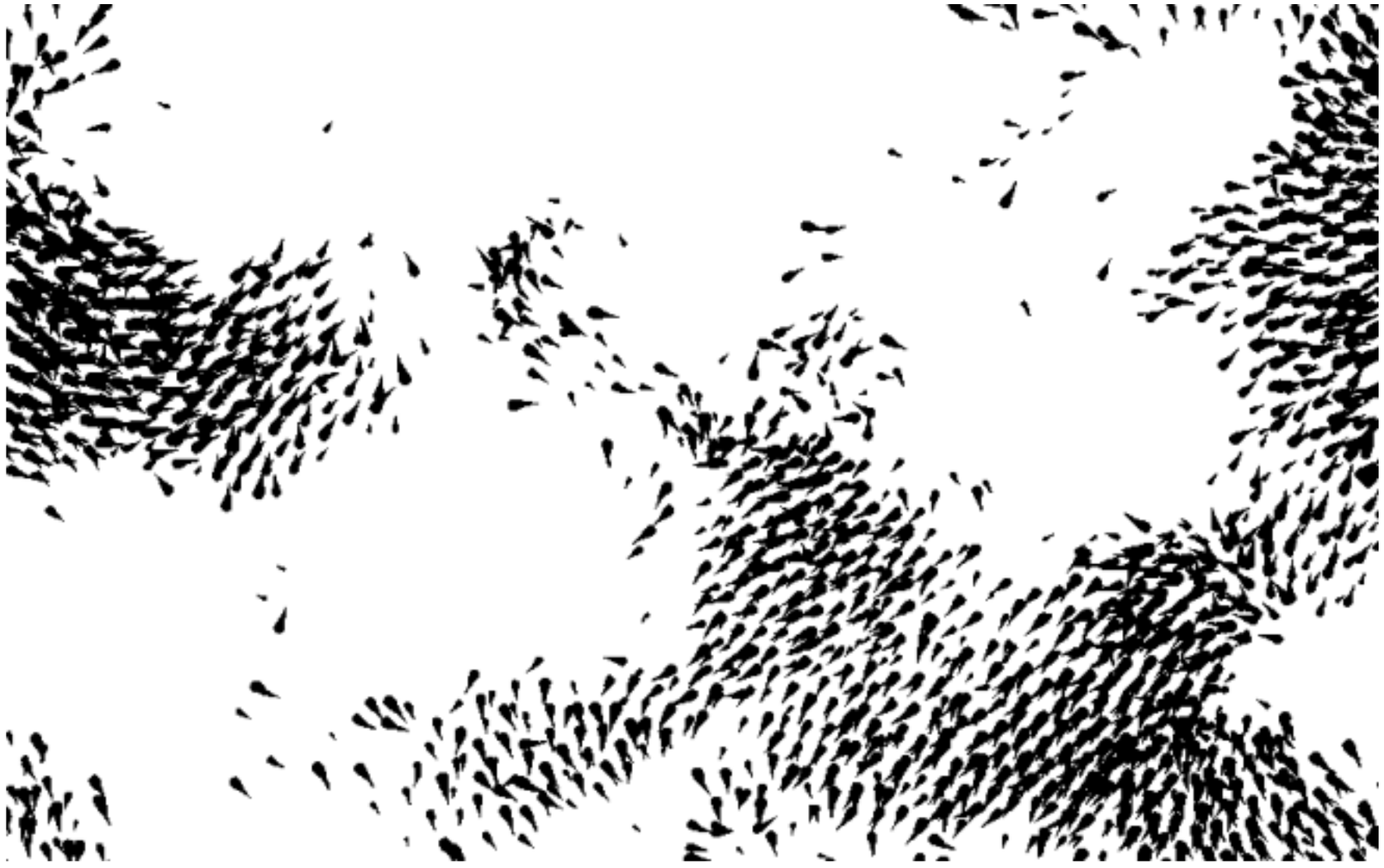
- **Termites:** they build large nest structures with a complexity far beyond the comprehension and ability of a single termite.
- **Ants**: tasks are dynamically allocated within an ant colony, without any central manager or task coordinator. Also, the ant foraging behavior is a result of simple trail-following rules.
- **Bees**: they use the "waggle dance" to communicate, which results in optimal foraging behavior.
- **Birds** in a flock and **fishes** in a school self-organize in optimal spatial patterns.
- **Lions** (as well as wolves and other predators) exhibit hunting strategies to outsmart their prey.
- **Bacteria** communicate using molecules (comparable to pheromones) to collectively keep track of changes in their environment.
- **Slime molds** consist of very simple cellular organisms with limited abilities. However, in times of food shortage they aggregate to form a mobile slug with the ability to transport the assembled individuals to new feeding areas.

SWARM INTELLIGENCE

EMERGENT BEHAVIOR (WITH LEADER)

In some cases there is a **leader** and more restrictive rules on relative motion, but individuals still use local information to decide how to move (e.g. herding, V formation, processions).

Computational Swarm Intelligence



COMPUTATIONAL SWARM INTELLIGENCE

WHAT IS (COMPUTATIONAL) SWARM INTELLIGENCE?

- **Computational Swarm Intelligence** refers to algorithmic models of swarm intelligence behavior observed in nature and, more in general, in groups of agents.

- Main principles
  - **Unity is strength**: the swarm can perform tasks that a single individual cannot perform.
  - **Multiplicity/resilience**: the swarm is composed of several individuals, some of which may be lost or make mistake, but its performance is not affected.
  - **Locality/simplicity**: individuals have "simple" computational/physical abilities, i.e., they have only local sensory information (no knowledge of the global state of the swarm or its "goal"), and little/no memory. Also, they perform "simple" actions (complexity emerges).

- Challenges
  - **Find individual behavioral rules that result in desired swarm behavior (reverse engineering)**: As these rules should be simple, often they are hand-designed by experts. However, they can also be obtained automatically by Evolutionary Algorithms.
  - **Make sure the emergent behavior is stable**: in some simple cases the swarm behavior can be characterized analytically by dynamical systems theory. However, usually agent interactions are hard to model and one can only perform a posteriori observations of the emergent behavior.

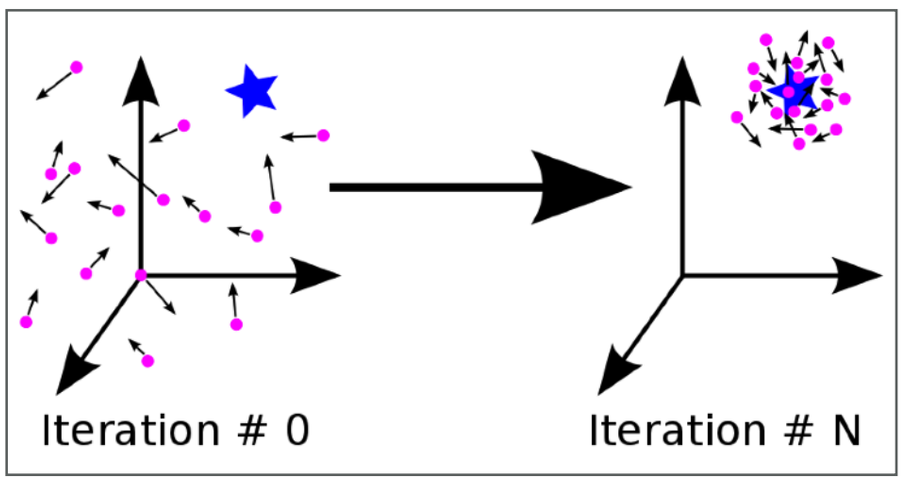
COMPUTATIONAL SWARM INTELLIGENCE

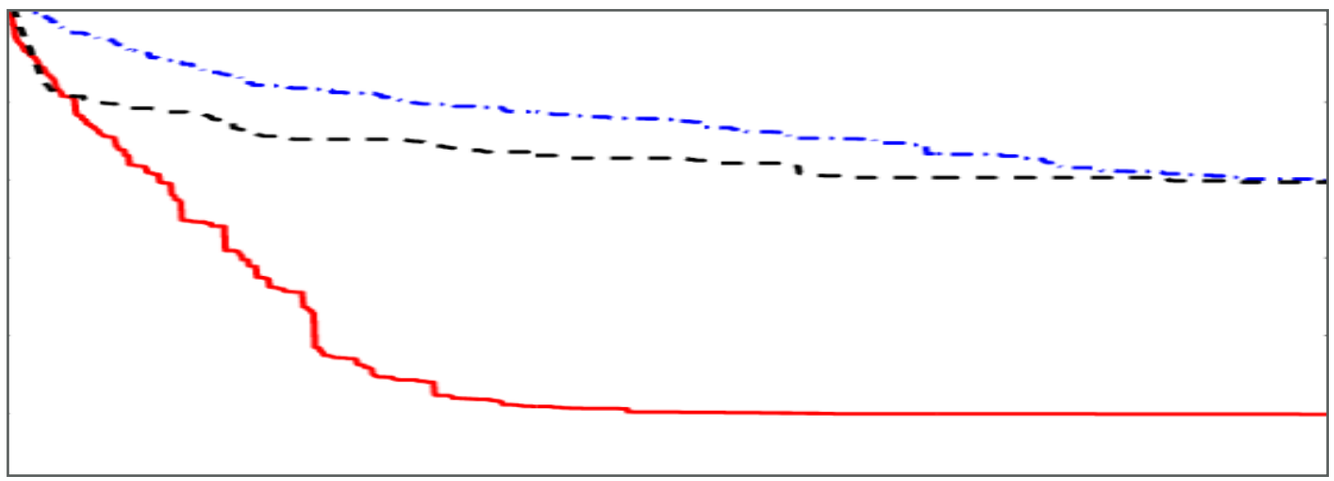
WHAT IS (COMPUTATIONAL) SWARM INTELLIGENCE?

- Virtual models
  - Usually optimization algorithms, e.g. Particle Swarm Optimization, Ant Colony Optimization, etc.
  - Also used for multi-agent simulations (e.g. for predicting mass behavior)

- Embodied models
  - Implemented in hardware, e.g. robots, network nodes, embedded systems, etc.

- Main features
  - Population-based
  - Each individual changes its state according to some strategies
  - Individual strategies use info (e.g.: velocity, relative distance, etc.) from neighboring individuals





COMPUTATIONAL SWARM INTELLIGENCE

PARTICLE SWARM OPTIMIZATION

- Mimics a group of birds exploring an environment in search of food (Kennedy and Eberhart, 1995)
  - Each place in the environment is associated with a reward
  - Each particle has a memory of the most successful place it visited
  - Each particle gets information from its neighbors → **collective problem solving**
- Standard PSO (mostly used for **continuous optimization**)
  - A swarm of particles is initialized with random positions and velocity
  - At each step, each particle updates first its velocity:

$$\overline{v}' = w \cdot \overline{v} + \phi_1 U_1 \cdot (\overline{y} - \overline{x}) + \phi_2 U_2 \cdot (\overline{z} - \overline{x})$$

where:

x and ***v*** are the particle's current position and velocity, respectively

z and **y** are the neighborhood and focal particle's best position, respectively

ω is the inertia (weighs the current velocity)

Φ_1 is the learning rate for the personal influence

Φ_2 is the learning rate for the social influence

U_1 and U_2 are uniform random numbers in [0,1]

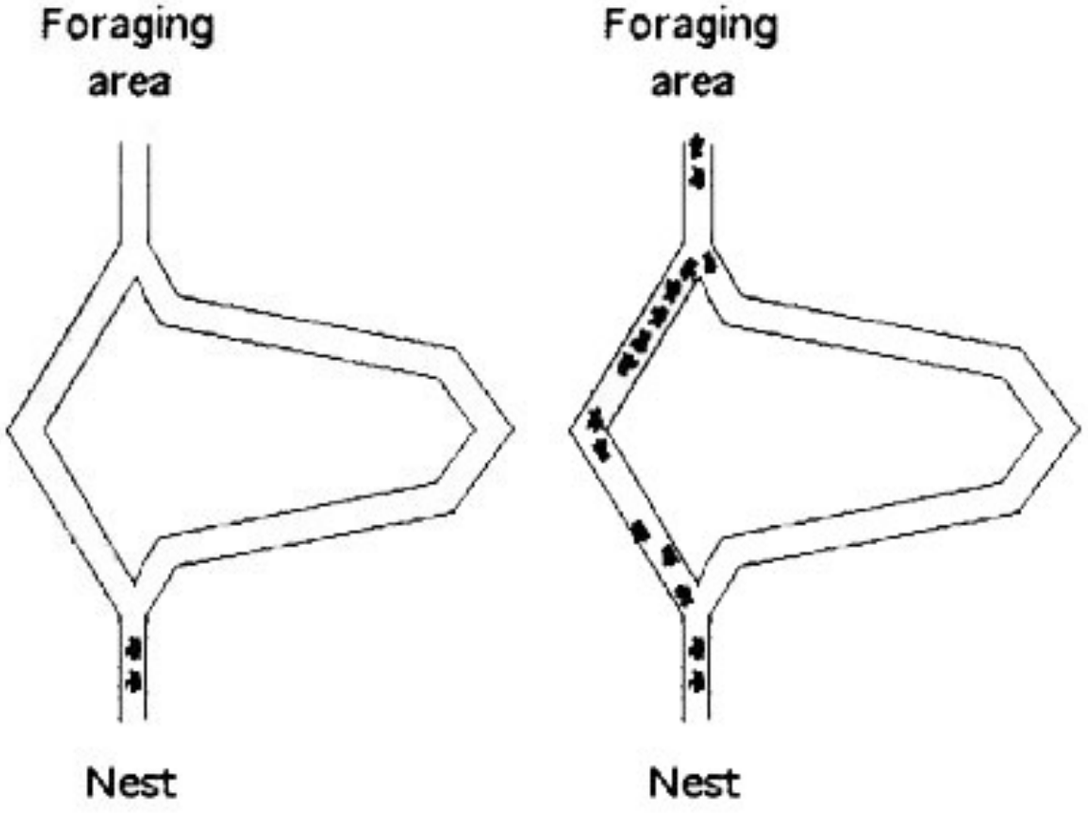
- Then, each particle updates its position ***x'*** = ***x***+***v'***
- In case of improvement, update **y** (and ***z***)
- This loop is iterated until a given stop condition is met

COMPUTATIONAL SWARM INTELLIGENCE

ANT COLONY OPTIMIZATION

- Mimics the navigation strategy used by foraging ants (Dorigo et al., 1991)
  – At first: ants move at random (→exploration)
  – Later, ants deposit pheromones to "reinforce" the path-following along some specific "good" trails
  – Pheromones along a trail evaporate when that trail is not followed anymore
  – Indirect agent communication of search experience via the environment (**stigmergy**)

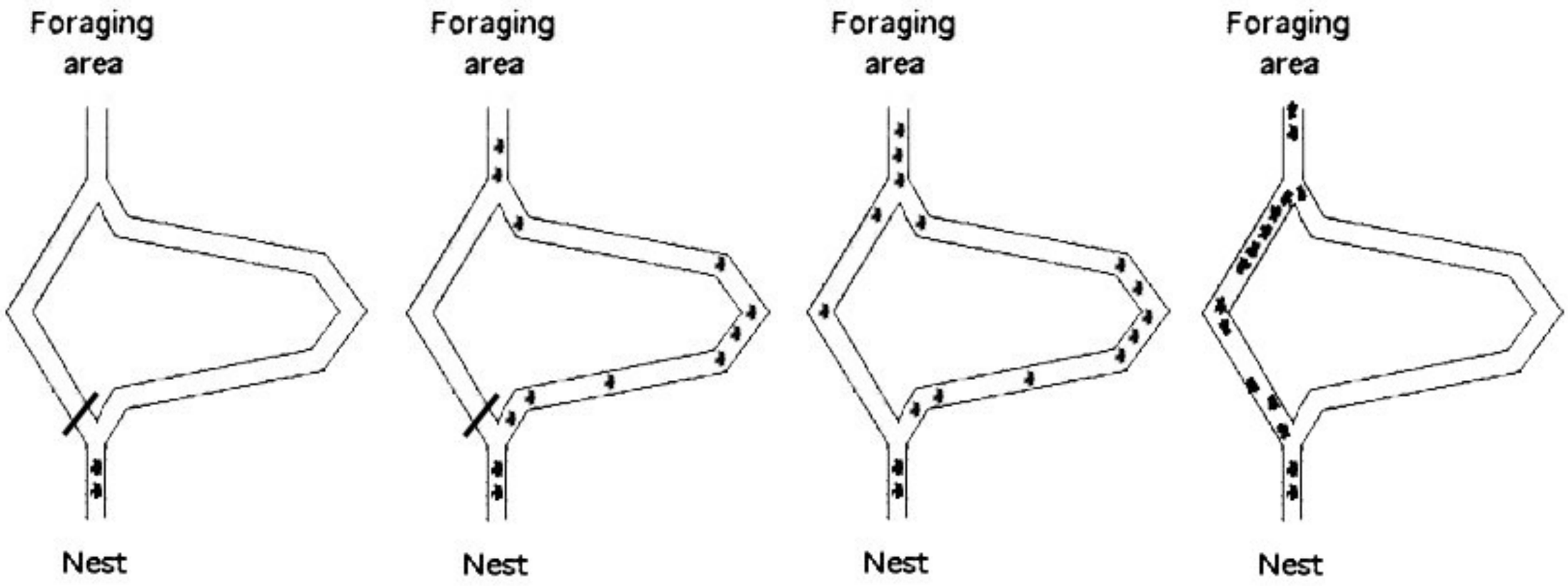
- Typically applied to TSP and other combinatorial (graph-based) problems



COMPUTATIONAL SWARM INTELLIGENCE

ANT COLONY OPTIMIZATION

- Notes on algorithmic performance
  - Finds best solution on "small" problems (e.g. TSP of up to 30 cities)
  - Finds good solutions on large problems compared to other techniques
  - Finds best solution on large problems when coupled with other search techniques
  - Can operate on dynamic problems (e.g., node malfunctioning) that require fast rerouting (ant trails are adaptive!)



COMPUTATIONAL SWARM INTELLIGENCE

OTHER PARADIGMS/METAPHORS

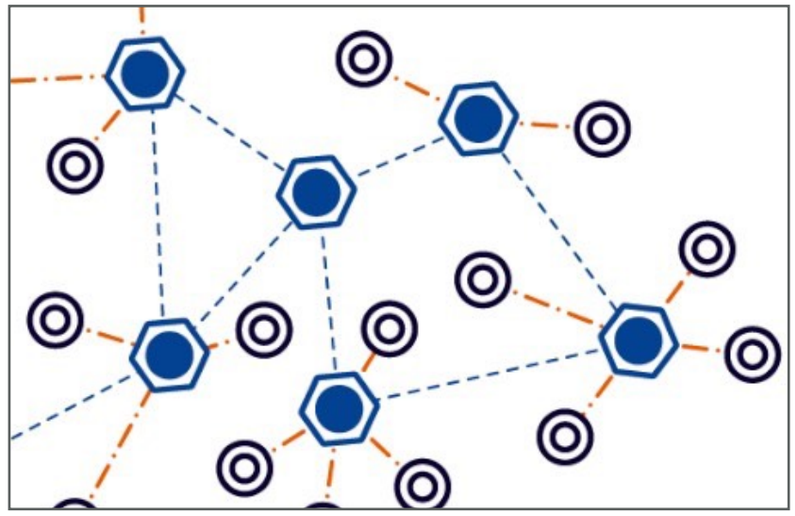
Remember the criticism on metaphors we have seen earlier? There you go:

- Harmony search (Geem, Kim & Loganathan 2001)
- Artificial bee colony algorithm (Karaboga 2005)
- Bees algorithm (Pham 2005)
- Glowworm swarm optimization (Krishnanand & Ghose 2005)
- Shuffled frog leaping algorithm (Eusuff, Lansey & Pasha 2006)
- Imperialist competitive algorithm (Atashpaz-Gargari & Lucas 2007)
- River formation dynamics (Rabanal, Rodríguez & Rubio 2007)
- Intelligent water drops algorithm (Shah-Hosseini 2007)
- Gravitational search algorithm (Rashedi, Nezamabadi-pour & Saryazdi 2009)
- Cuckoo search (Yang & Deb 2009)
- Bat algorithm (Yang 2010)
- Spiral optimization (SPO) algorithm (Tamura & Yasuda 2011,2016-2017)
- Flower pollination algorithm (Yang 2012)
- Cuttlefish optimization algorithm (Eesa, Mohsin, Brifcani & Orman 2013)
- Duelist Algorithm (Biyanto 2016)
- Killer Whale Algorithm (Biyanto 2016)
- Rain Water Algorithm (Biyanto 2017)
- Mass and Energy Balances Algorithm (Biyanto 2017)
- …

COMPUTATIONAL SWARM INTELLIGENCE

APPLICATIONS

- Optimization
- Multi-agent or crowd simulations
- Coordinated robots/drones
- Distributed control systems
- Distribution systems (water, gas, electricity, etc.)
- Telecommunication networks (P2P, sensor networks, etc.)
- Mobile & pervasive computing
- Social networks



Questions?