Workshop I on “Evolutionary Algorithms”

David Jiménez, Diego Reiriz and Javier Guzman

2018/10/11

Agenda

1. **Introduction to Evolutionary Algorithms (EA) ~ 30 min**
    - Biological motivation
    - Basics: *exploration and exploitation*
    - EA components: *representation, evaluation, selection, genetic operators*

2. **Evolutionary Algorithms ~ 45 min**
    - Genetic Algorithms: *intro, mutation, recombination, selection*
    - Evolution Strategies: *motivation*

3. **Neuroevolution ~ 15 min**
    - Introduction and motivation
    - ANN evaluation: *approaches*

4. **Notebook ~ 15 min**

5. **Lightning Talk:** *“LFDL y AcumosAI”* **~ 5 min**

Introduction to Evolutionary Algorithms

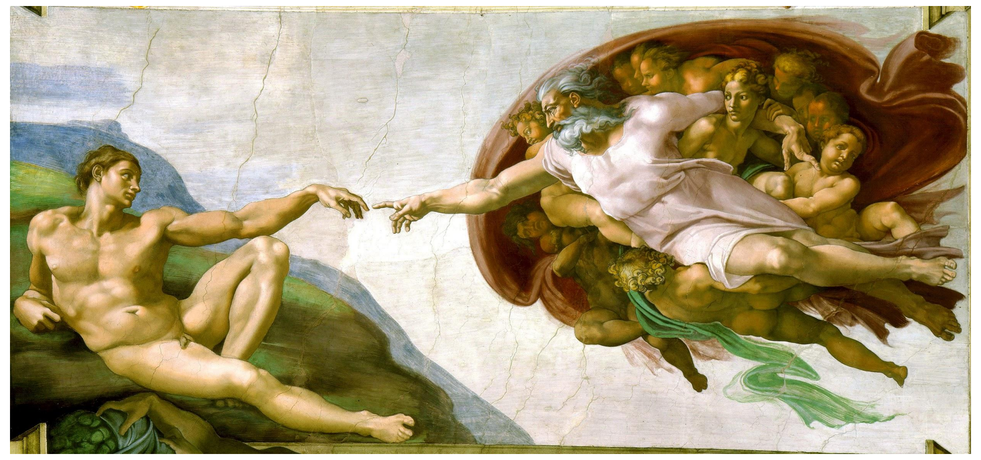
- *Biological motivation*
- *Basics*
- *EA components*

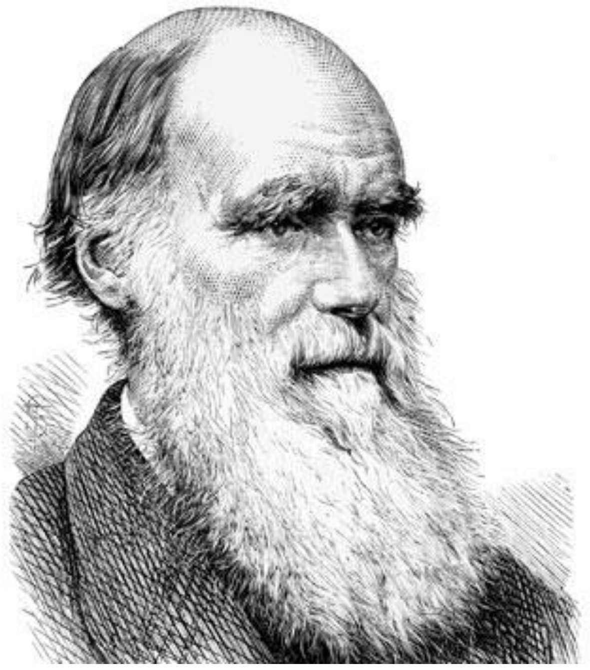
Intro. *Objectives and History*

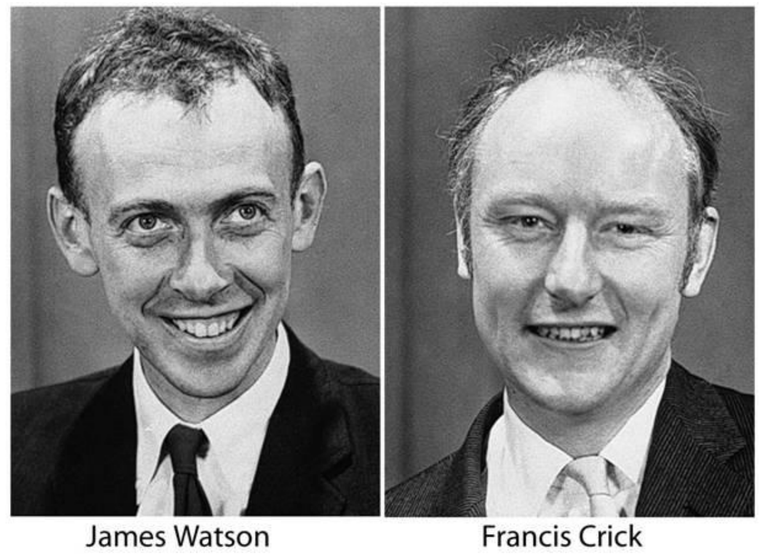
Objectives:

- Introduce *biological vs artificial* evolution
- Justify the utility of artificial evolution from an *engineering* perspective
- Overview the *components* of an Evolutionary Algorithm

History:





Aristotle, Patlo (400 BC)	Creationism (centuries)	Leclerc, Lamark (1830)	Darwin (1882)	Mendel, Weismann (1914)	Watson & Crick (2004)
- Man come from fishes - Spontaneous Generation	- God create the species - Genesis	- Species change - 1st theory of evolution	Natural Selection = Variability + Selection	- Inheritance - Germ and somatic cells	- Discovery of DNA - Molecular biology

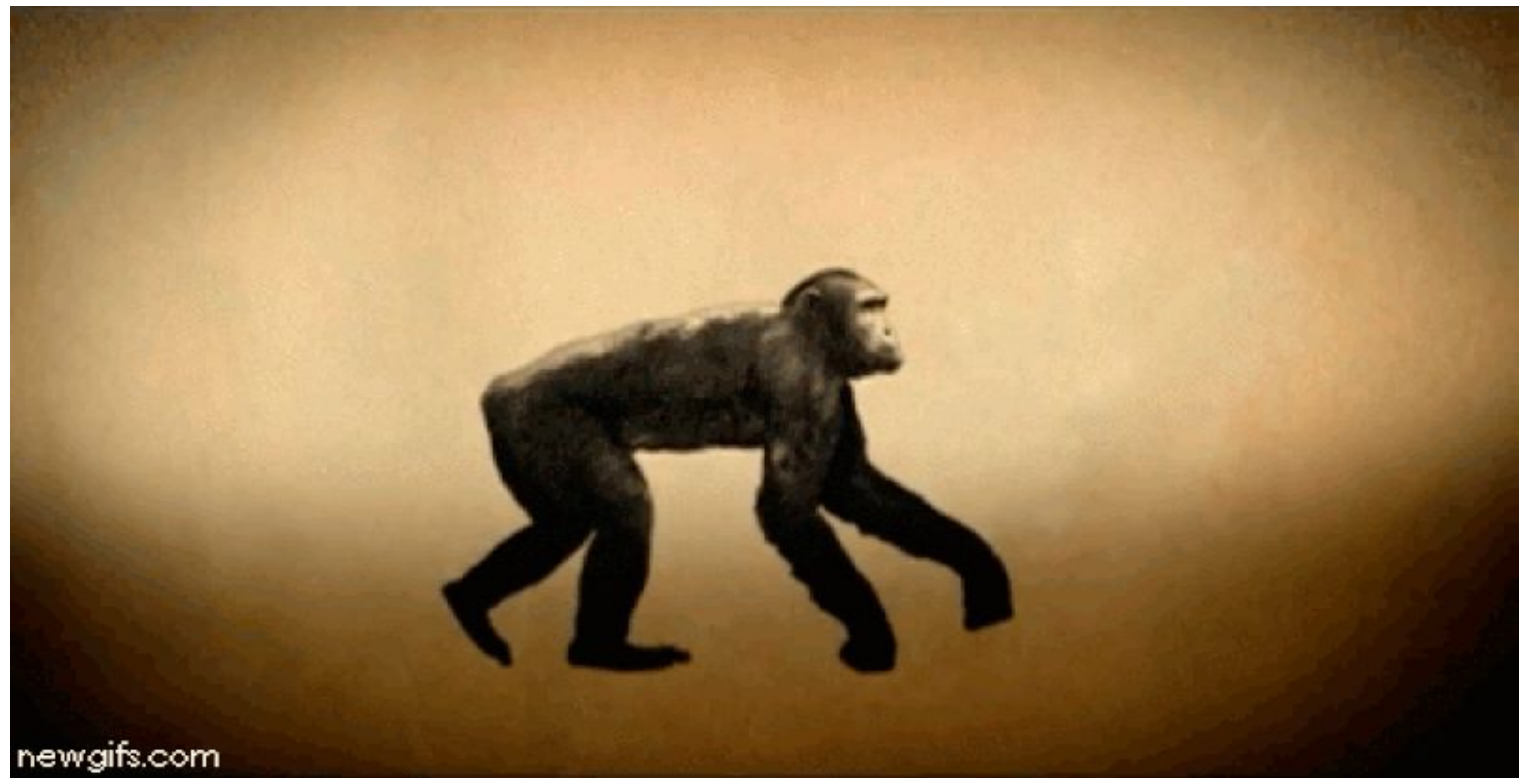
Bibliography:

- Eiben, A.E. and Smith, J.E. *Introduction to Evolutionary Computing*. Springer 2003
- Barrero D.F. *Evolutionary Computation Course*. MsC. Space Science and Technology

Intro. *Theory of evolution*

Neo-Darwinism: Darwin + Mendel + Weismann... also called Theory of Evolution

- *Variability + Selection = Evolution*



Principles:

1. There is *variation* among individuals
   - Sexual reproduction, mutation and gene flow
2. There is a *selection* of those individuals
   - Natural selection
   - Artificial selection
   - Sexual selection
   - Genetic drift (deriva genética)
3. The *fittest* is the one that survives (not the strongest!)

Intro. *Biological Motivation*

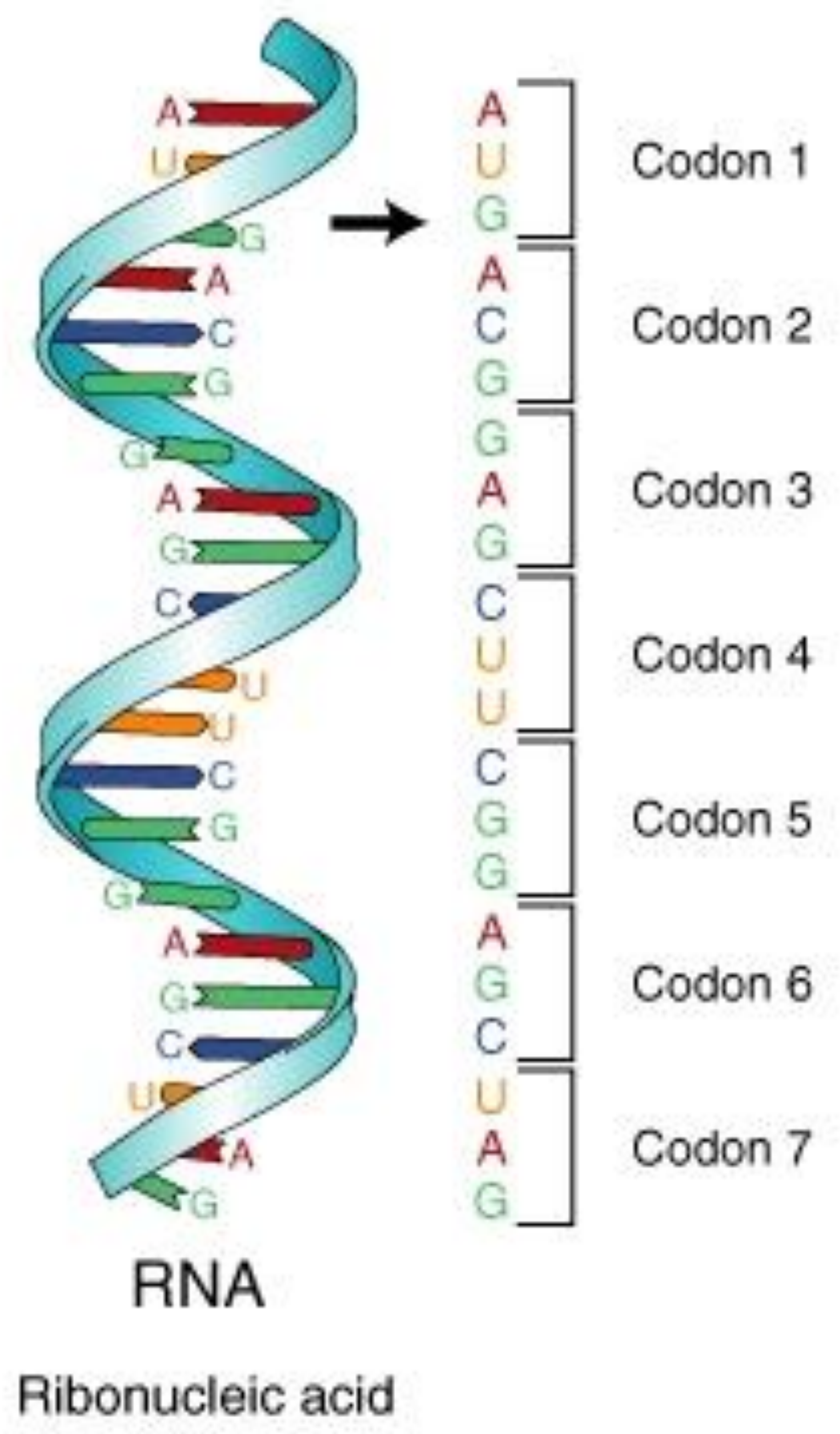
- Organisms are made by *proteins* (sequence of amino-acids)
- DNA codifies all the proteins in an organism
- Useful biological terms for EA:
  - *Chromosome*: structure with DNA & proteins
  - *Gene*: DNA fragment that codifies one protein
  - *Genotype*: sequence of DNA
  - *Phenotype*: characteristics of an individual

- Theory of Evolution, an *algorithmic* perspective:
  - Given a population…

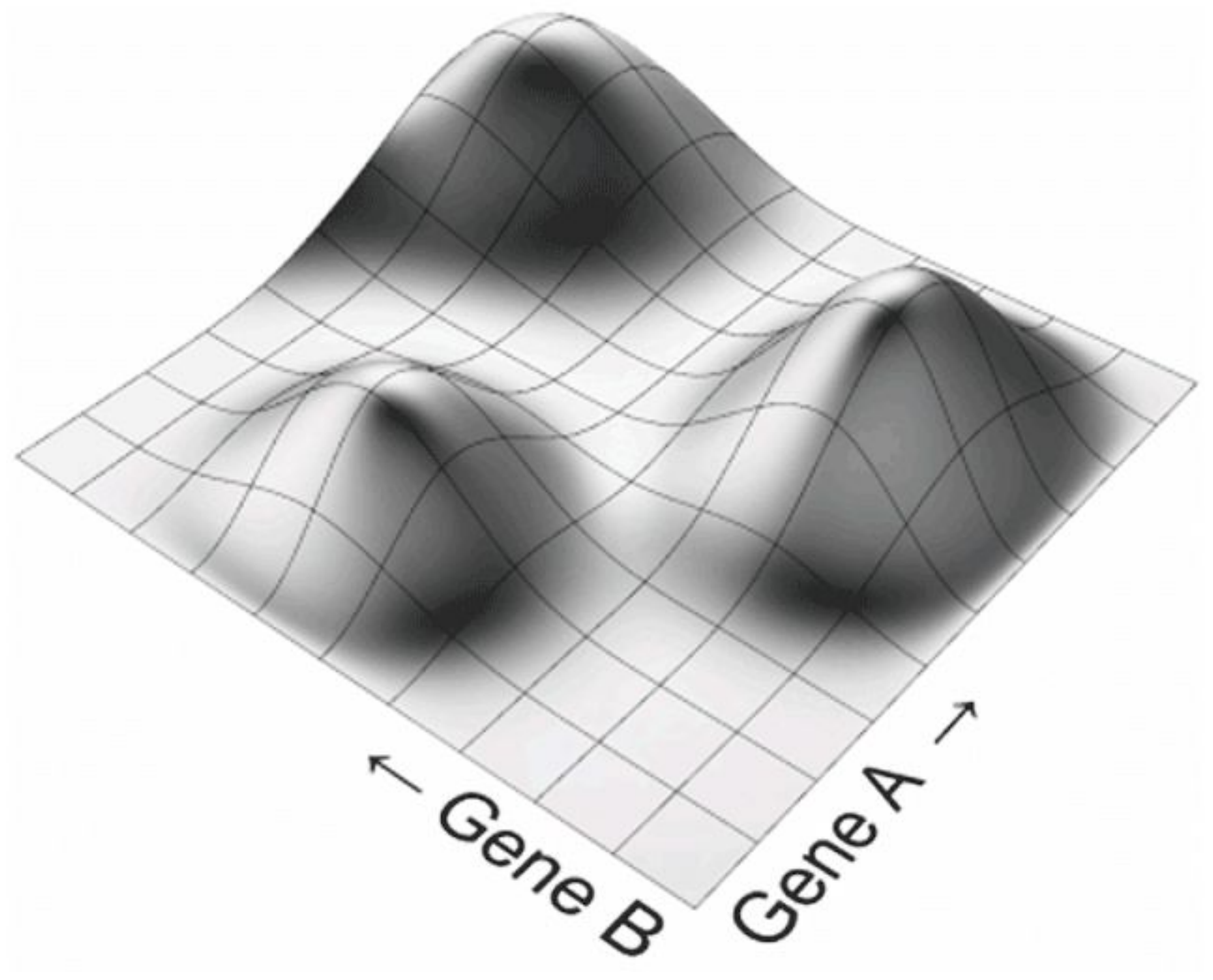
    ✓ There are *differences* among individuals

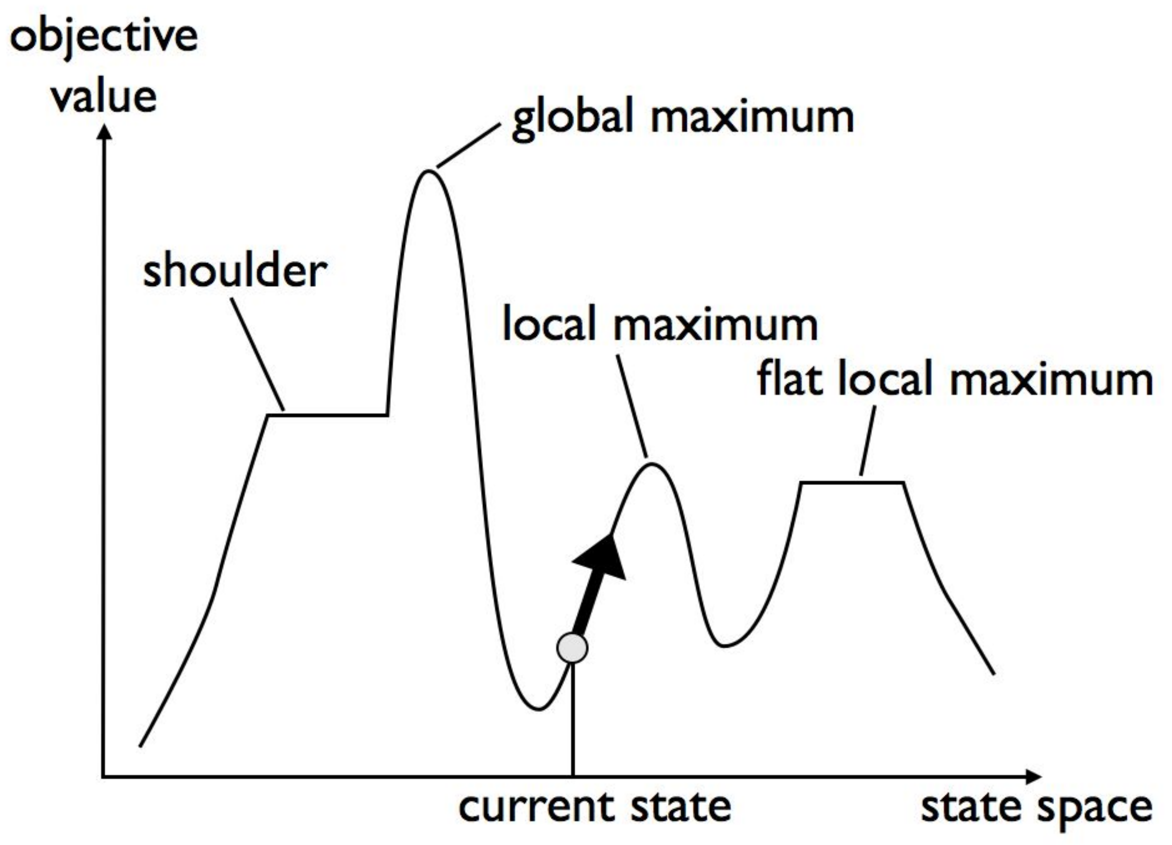
    ✓ Fittest individuals more *likely* to reproduce

    ✓ *Offspring*



Intro. *Evolution as Optimization*





- Biological evolution is, in essence, an *optimization algorithm*…

  ...it optimizes the survival probability

- ***Optimizing → search problem***

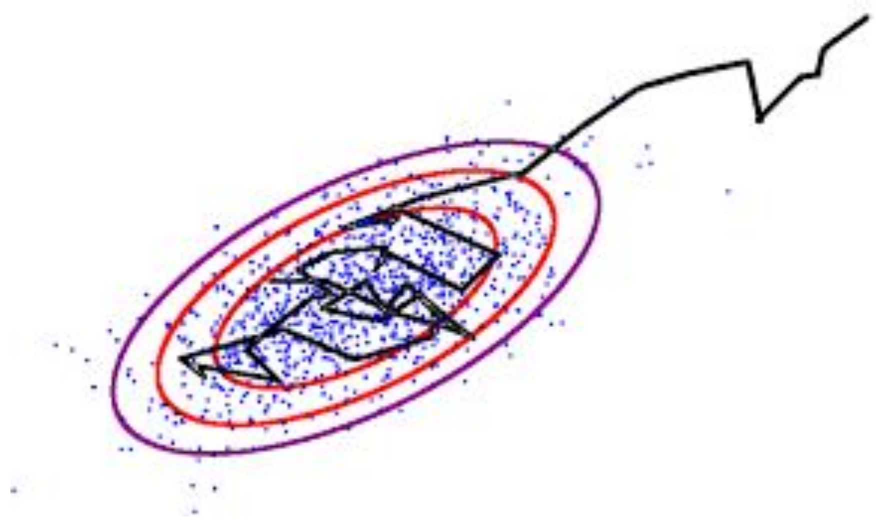
In AI, potential solutions are assessed:

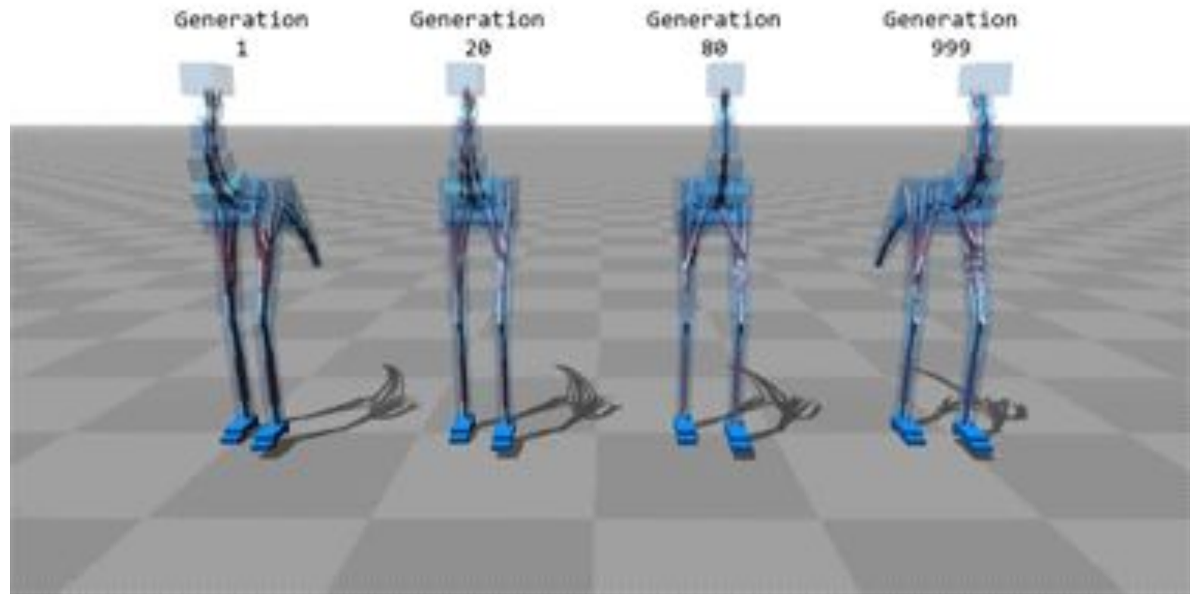
- *Cost function*
- *Objective*: optimize cost function

How to find a solution efficiently?

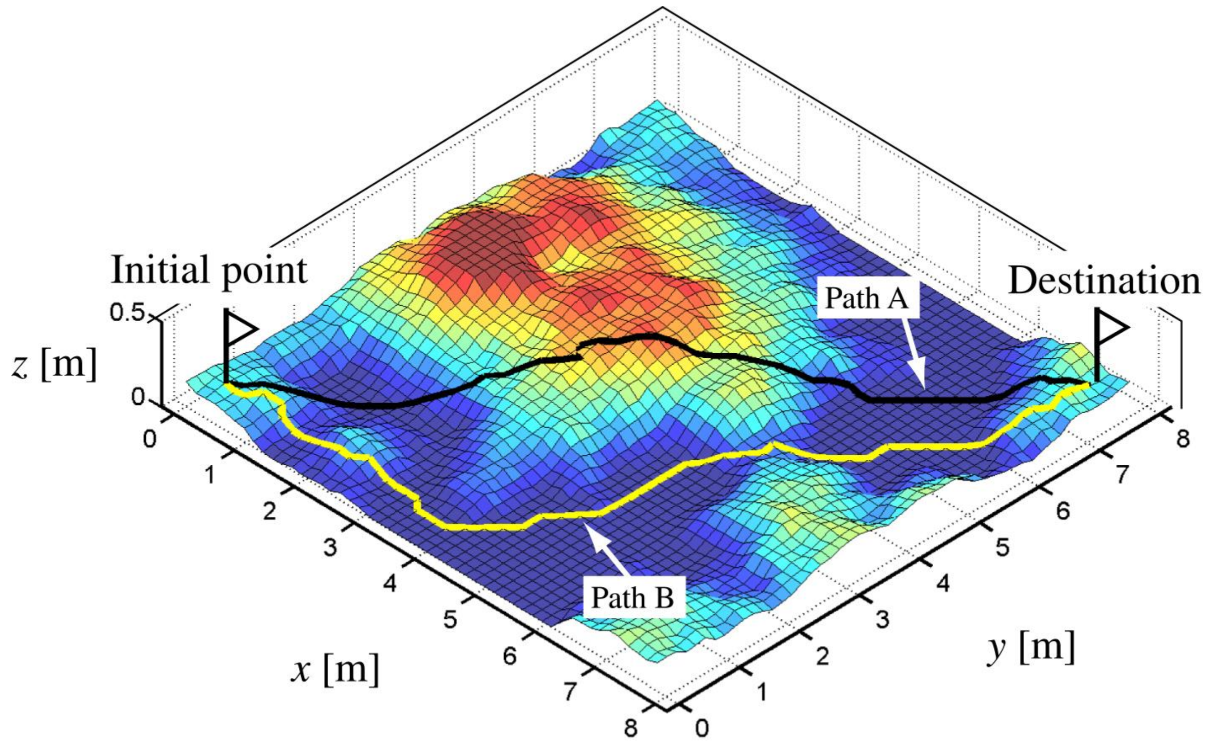
- With *domain knowledge*
- With randomness: *metaheuristics*

Intro. *Metaheuristics*





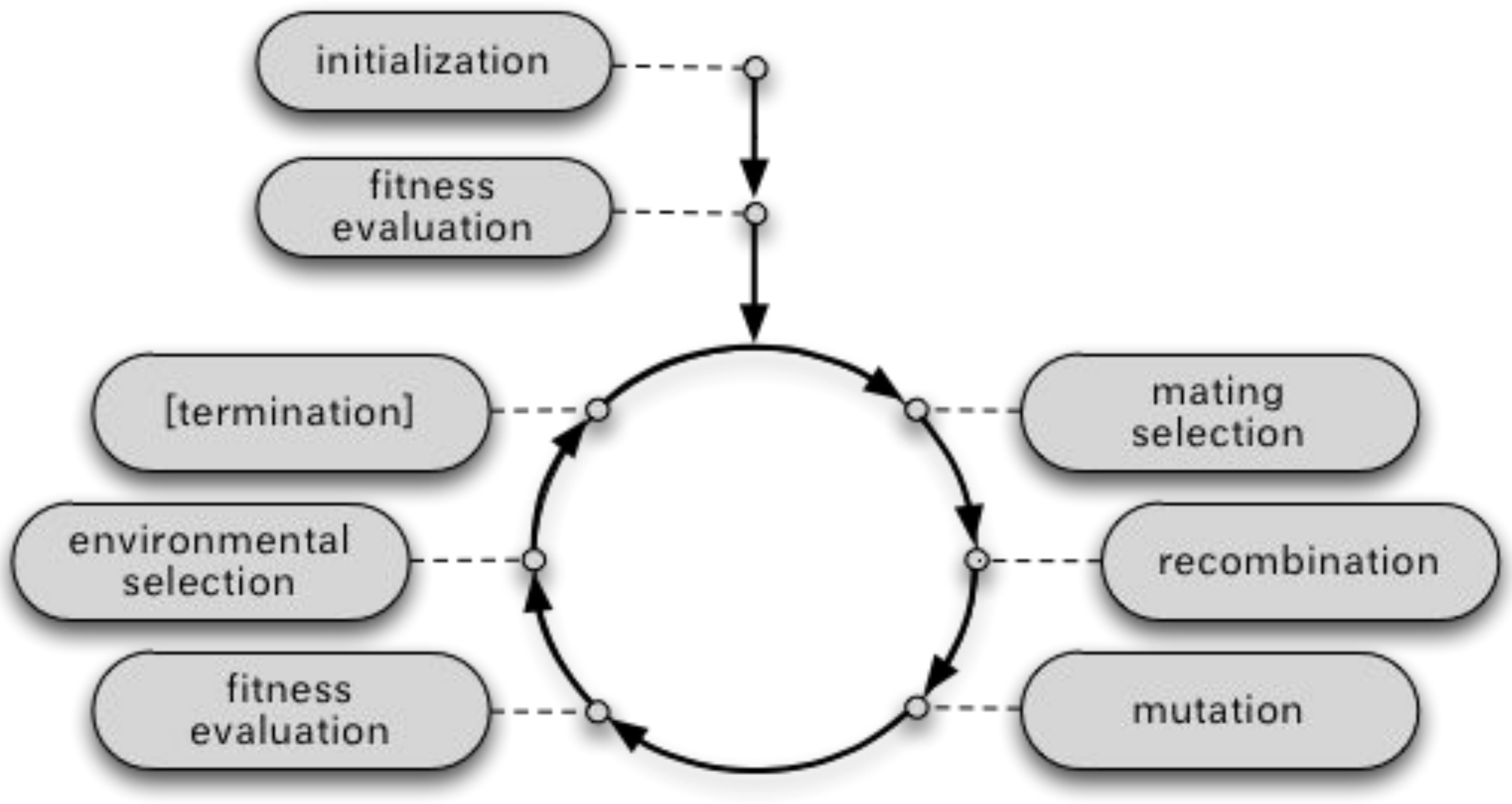
- Stochastic Optimization
  - MCMC
  - Metaheuristics
    - Population-based
      - Evolutionary Algorithms
        - Evolutionary Programming
        - Evolution Strategies
        - Genetic Algorithms
        - Genetic Programming
        - Estimation of Distribution
        - Novelty Search
      - Swarm Intelligence
        - Ant Colony Optimization
        - Particle Swarm Optimization
        - River Formation Dynamics
        - Stochastic Diffusion Search
        - Intelligent Water Drops
        - Charged System Search
        - Bee Colony Optimization
        - Firefly Algorithm
        - Artificial Inmune Systems
        - Bacteria Foraging
    - Trajectory methods
      - Hill-Climbing
      - Simulated Annealing
      - Tabu Search
      - ILS
      - GRASP
      - VNS
      - GLS
  - Gibbs sampling



Intro. *Evolutionary Algorithms - Basics*

1. Large number of *Evolutionary Algorithms*
   - There is no “canonical” algorithm
   - They all imitate biological evolution
2. They use *population*
   - Each individual represents a potential solution
   - Multiple representations
3. Population is *modified*
   - Mutation (1 individual)
   - Crossover (>1 individuals)
   - Multiple genetic operators
4. *Selection* that imitates natural selection
   - Fitness function
5. *Iterative* process

Possible basic algorithm



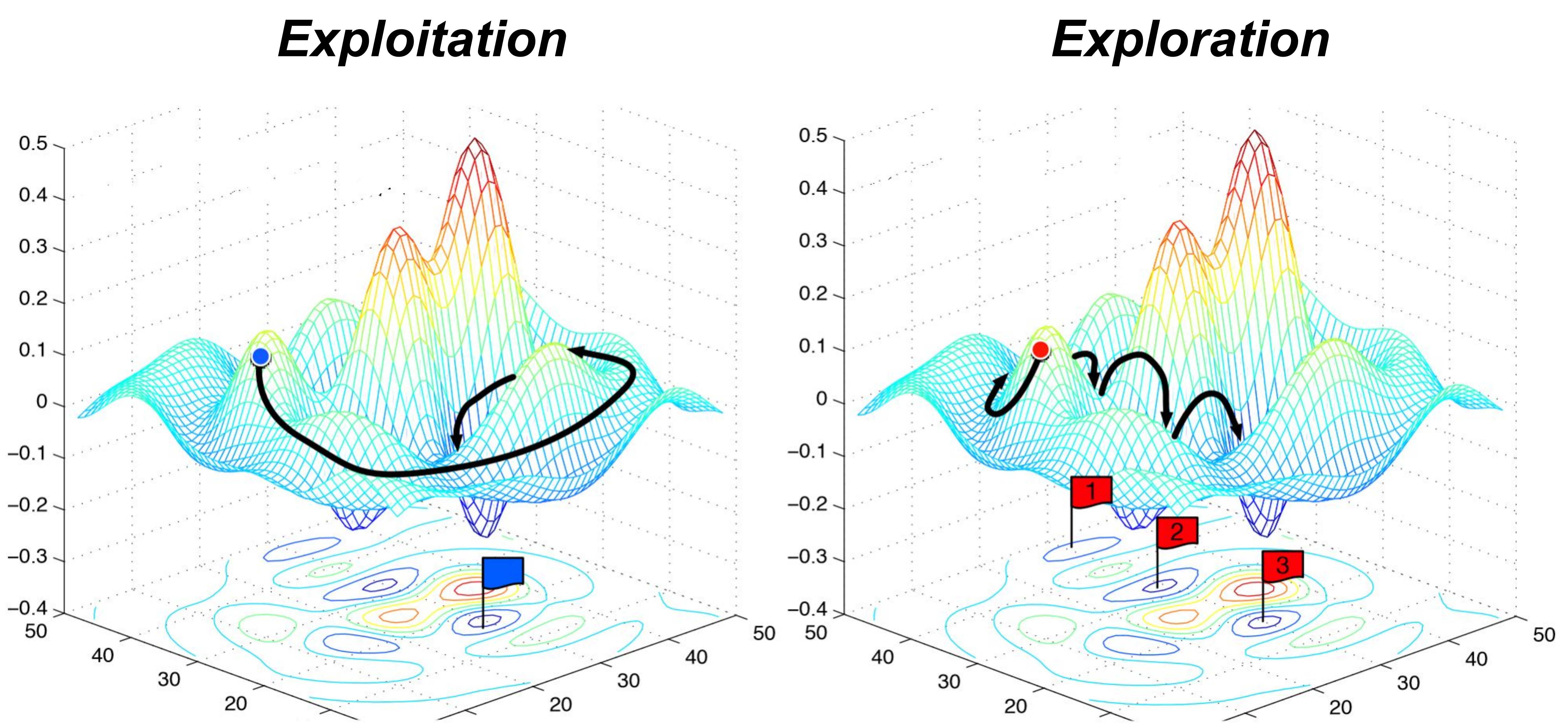
- ***Initialization***:
  - Usually random (rand population, domain knowledge)
- ***Termination criteria***:
  1. Get a desired fitness
  2. Loss of genetic diversity
  3. Number of iterations
  4. Lack of fitness improvement

Intro. *Evolutionary Algorithms - Basics*

Exploration vs Exploitation

- ***Exploration***: search of new regions (global search)
  - Explore the search space
  - Performed by *Mutation*
- ***Exploitation***: search of local solution
  - Exploits acquired knowledge
  - Performed by *Crossover*

Need of Trade-off!!



EA Components

1. Representation
2. Evaluation
3. Selection
4. Genetic Operators

Intro. *EA Components. Representation and Evaluation*

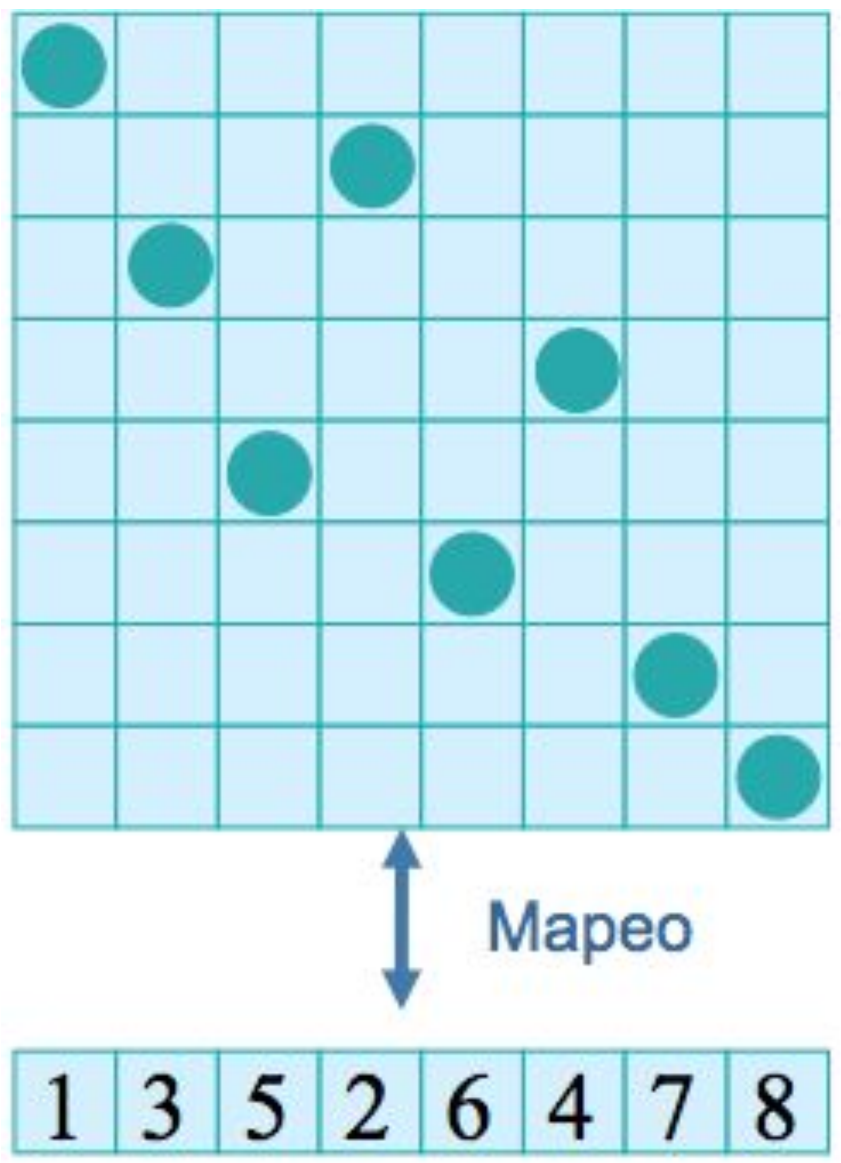
1. **Representation.** Main difference among EAs
   - Strings: *Genetic Algorithms*
   - Real Vectors: *Evolution Strategies*
   - State Machines: *Evolutive Programming*
   - Trees: *Genetic Programming*

   > *Irrelevant, use the most natural genetic operator according to representation* <

2. **Evaluation.** *Fitness value for optimization*
   - *Each individual → potential solution*

Example: 8 queens with a Genetic Algorithm

- Phenotype: board position
- Genotype: integer vector
- Eval: fitness = #queens can be attacked



Intro. *EA Components. Selection*

3. **Selection:** pick individuals for reproduction
   - Imitates *natural selection*
   - Higher reproduction ***probability*** for *high fitness* individuals
     - Randomness helps avoiding local minima
   - Introduces *selective pressure (lion between sheeps)*

High selective pressure reduces genetic diversity

- Faster evolution, higher probability of local maxima
- Remove low fitness individuals
  - Potentially valuable genetic material can be lost

Selection **operators**:

- Tournament size n, roulette-wheel, rank-based...

Tournament size n

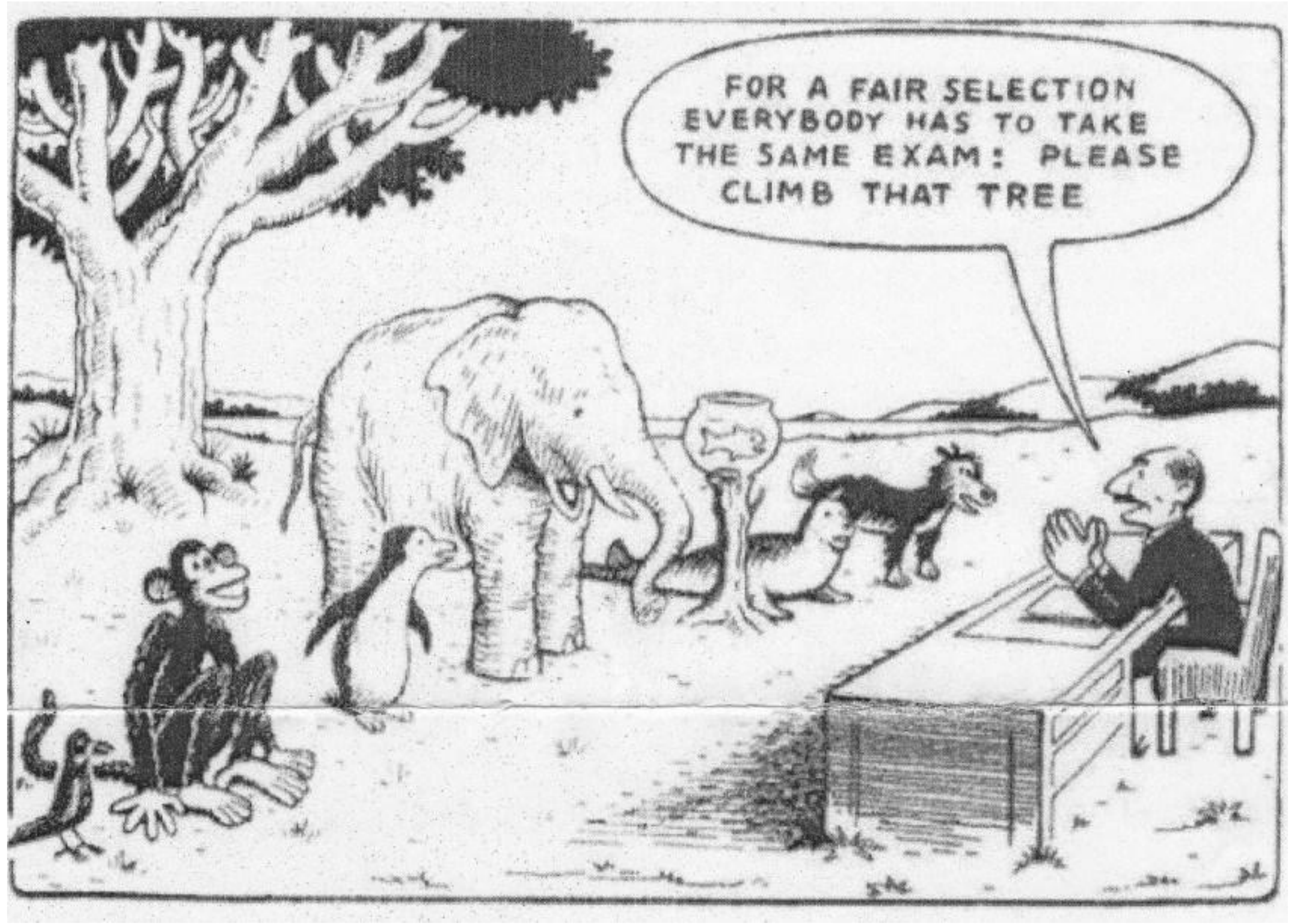
1. Randomly take 'n' individuals
2. Compute their fitness
3. Select one with highest fitness

Variable selection pressure depending on n

Intro. *EA Components. Selection*

Replacement strategy: *select which individual replace (remove)*

- Two basic strategies:
  1. *Generational algorithms*: replace all the offspring
     - Iterations are named *generations*
     - Time is usually measured in generations
  2. *Steady-stade*: replace part of the offspring
     - Criteria: age, fitness, selection, etc
     - Lower memory consumption



- Hybrid strategy: ***Elitism***
  - Replace the population, except the ‘n’ *fittest individuals*
  - n fittest individuals guaranteed to survive (*look alike them*)

Intro. *EA Components. Genetic Operators*

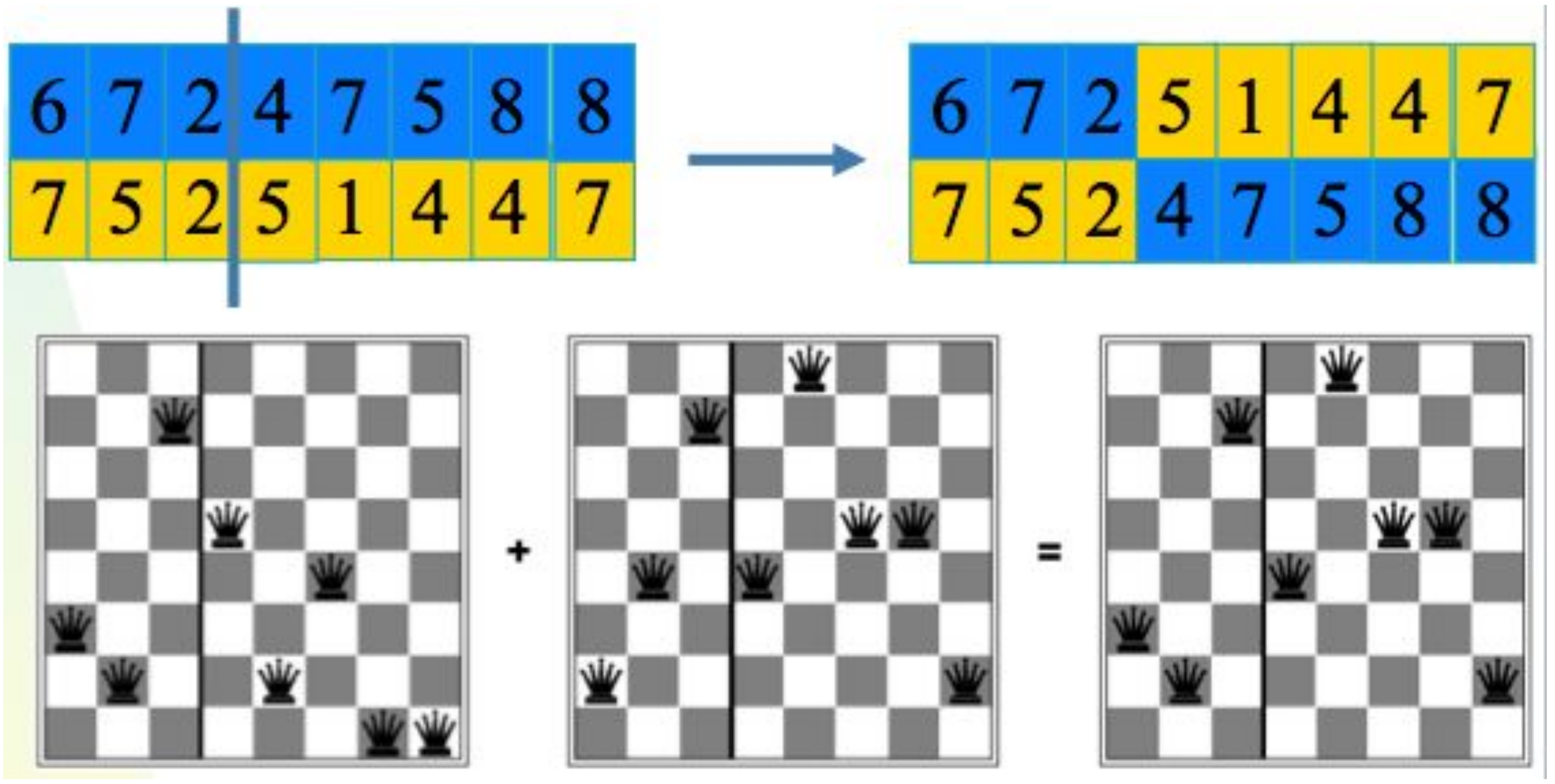
4. **Genetic operators:** build new individuals

   4.1 **Mutation** enhances *exploration*

- Takes a *genotype* and returns another one (keeps genetic diversity)
- Disruptive role: moves population to new regions

   4.2 **Crossover** enhances *exploitation*

- Fuse information from parents (sexual reproduction)
- Offspring uses to be *worse* than its parents
  - ➢ With luck, good components are joined
- Crossover has a constructive role
  - ➢ Join preexistent components
  - ➢ No new genetic material



Evolutionary Algorithms

(+34) 986 120 430 | gradiant@gradiant.org | www.gradiant.org

Genetic Cars 2

http://rednuht.org/genetic_cars_2/

Types of evolutive algorithms

Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem.

WIKIPEDIA
The Free Encyclopedia

Types of evolutive algorithms

Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem

- **Genetic algorithms**:
- **Genetic programming:**
- **Neuroevolution:**

WIKIPEDIA
The Free Encyclopedia

Types of evolutive algorithms

Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem

- **Genetic algorithms**: This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved)

- **Genetic programming:**

- **Neuroevolution:**

WIKIPEDIA
The Free Encyclopedia

Types of evolutive algorithms

Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem

- **Genetic algorithms**: This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved)

- **Genetic programming:** solutions are encoded as trees and the solution usually encodes instructions instead of a string of numbers

- **Neuroevolution:**

WIKIPEDIA
The Free Encyclopedia

Types of evolutive algorithms

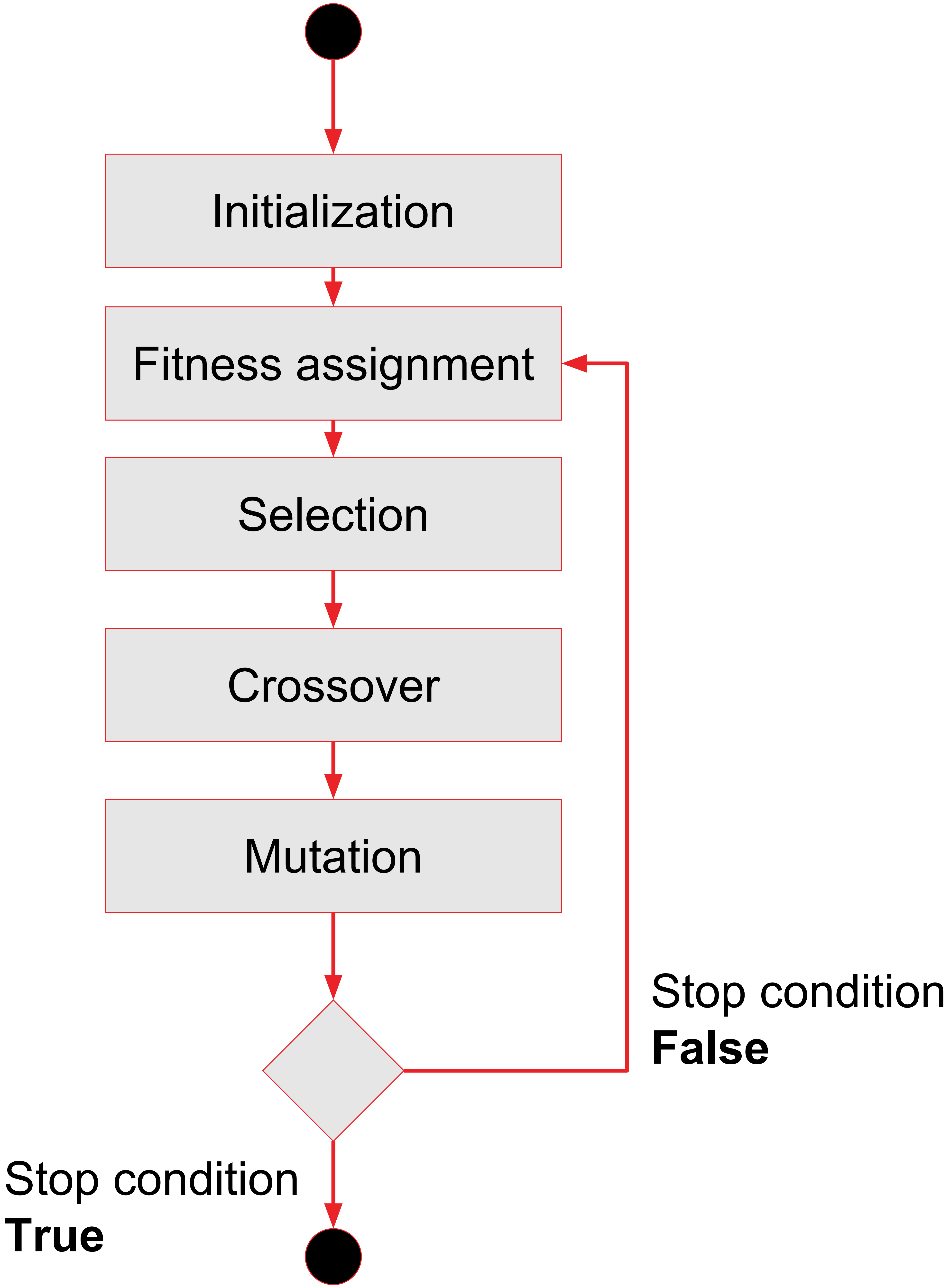
Similar techniques differ in genetic representation and other implementation details, and the nature of the particular applied problem

WIKIPEDIA
The Free Encyclopedia

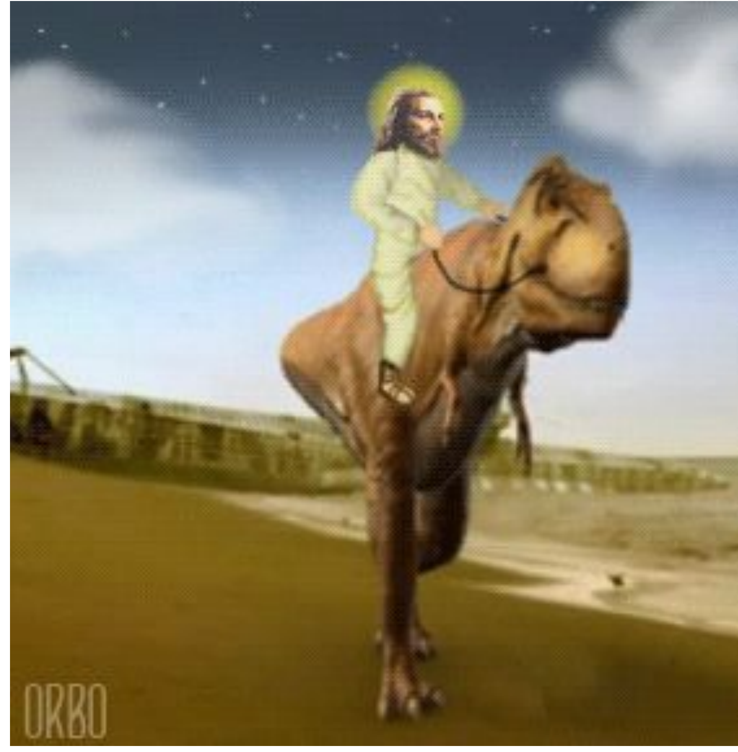
- **Genetic algorithms**: This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved)

- **Genetic programming:** solutions are encoded as trees and the solution usually encodes instructions instead of a string of numbers

- **Neuroevolution:**



www.gradiant.org



Solution representation

- Binary

1	0	0	1

- One of the Oldest and widely used codifications
- Often used to codify non-binary information -> Gray Coding or Binary Codification
- Use it only to represent binary information -> 1001 = 9

- Integer

4	2	3	1

- More Natural Codification for many problems
- Optimization of integer values
- Represent elements as integers ({1,2,3,4} => {Up, Right, Down, Left})

- Floating-Point

1.1	2.5	0.1	6.3

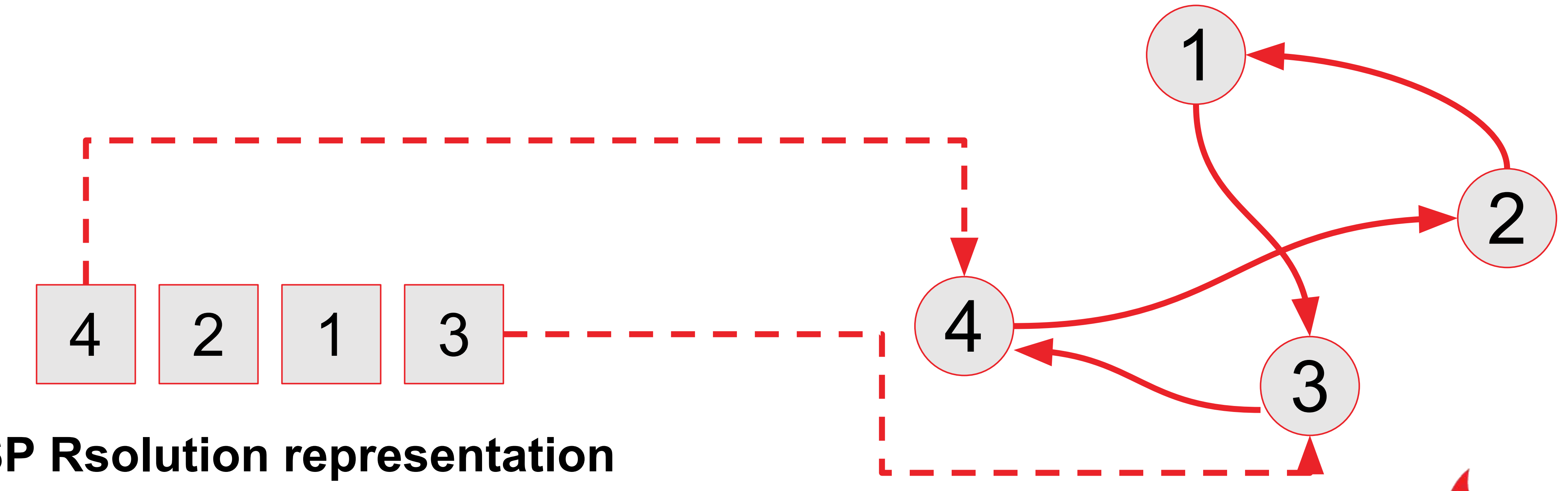
- Common in optimization problems
- Solutions with continous nature

- Permutations

4	2	1	3

Problems that involve order

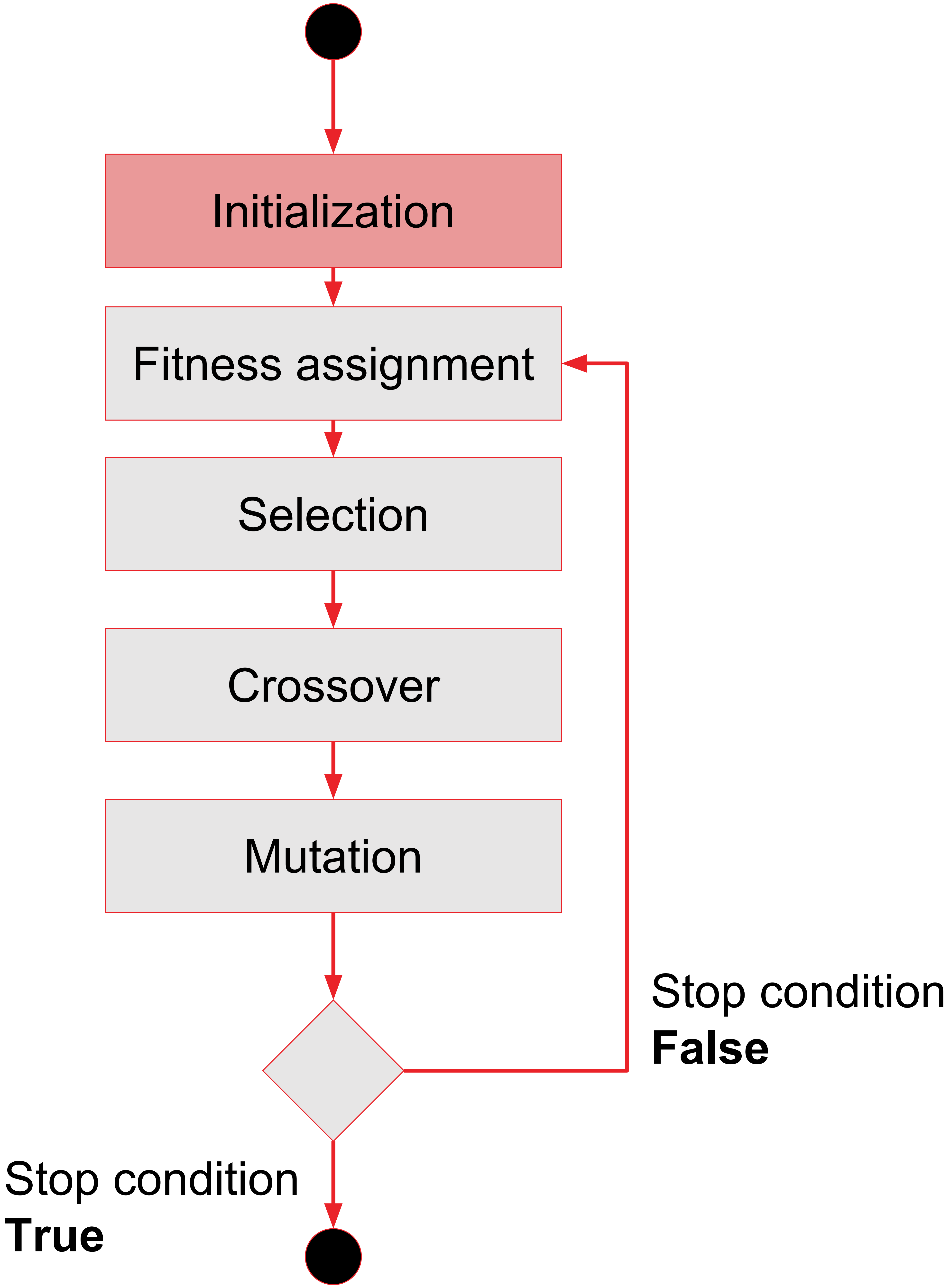
- Sequence of integer
- No repeated numbers
- Range of valid numbers
- Special geneteic operators



TSP Rsolution representation

3	4	2	1

www.gradiant.org


Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True

Initialization

We need to create an initial population of solutions. It could be done:

- Creating random solutions

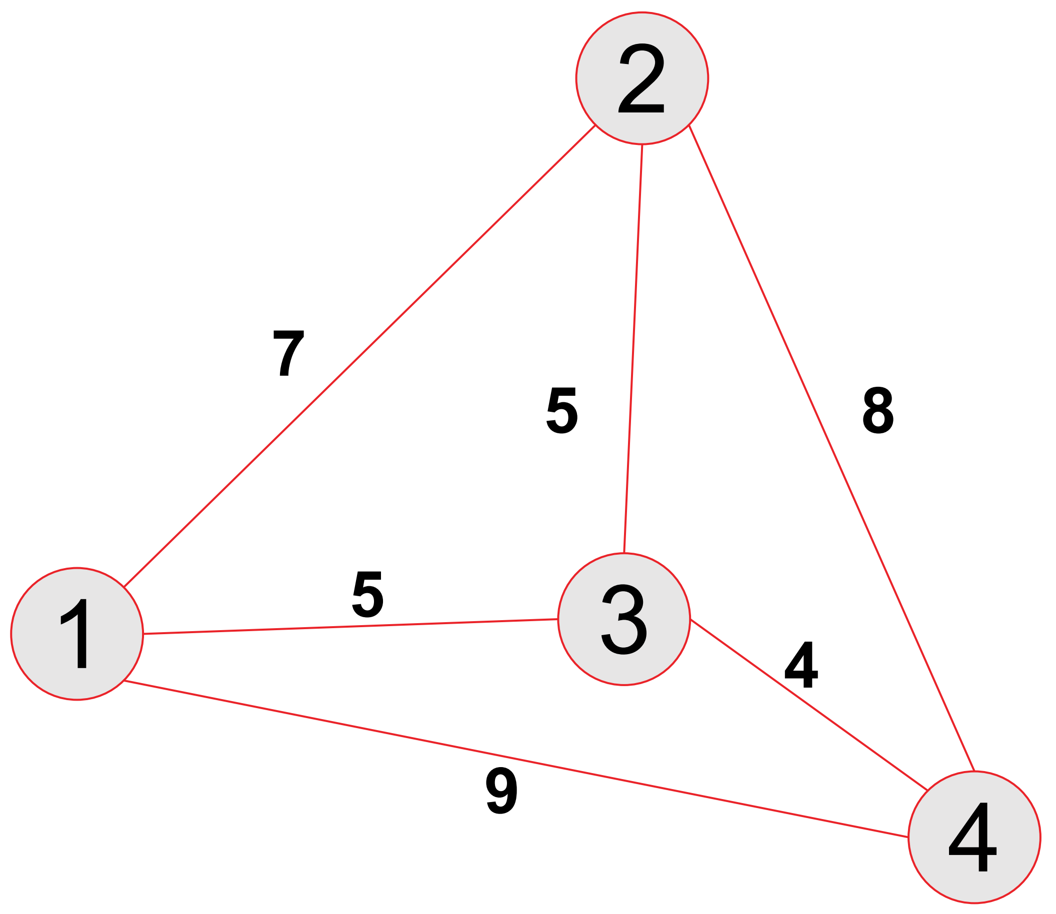
1	2	3	4
3	2	1	4
4	3	2	1

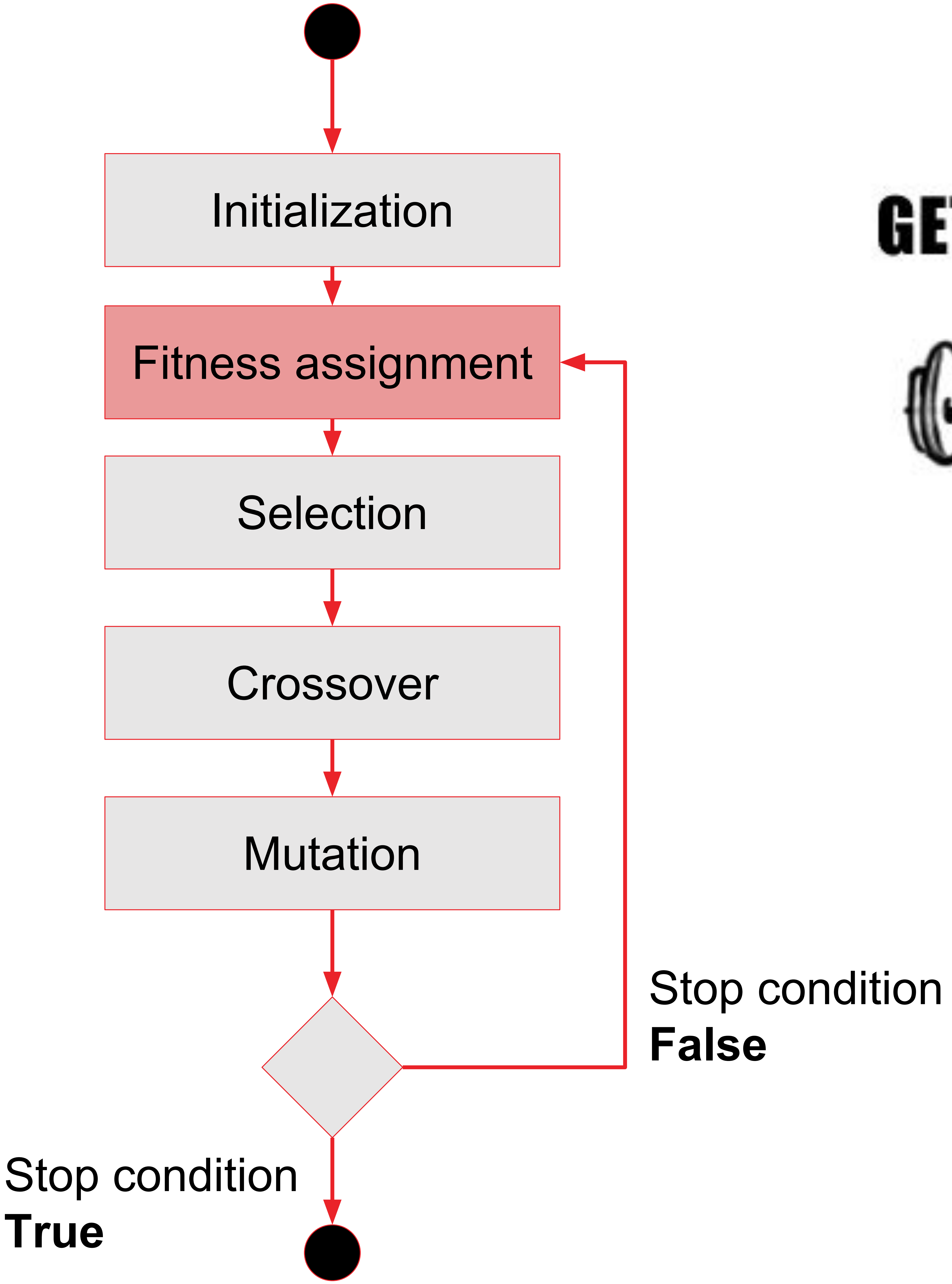
- Heuristics & Metaheuristics
  - Human Biased
  - Ex: Best First

3	4	2	1
2	3	4	1
1	3	4	2

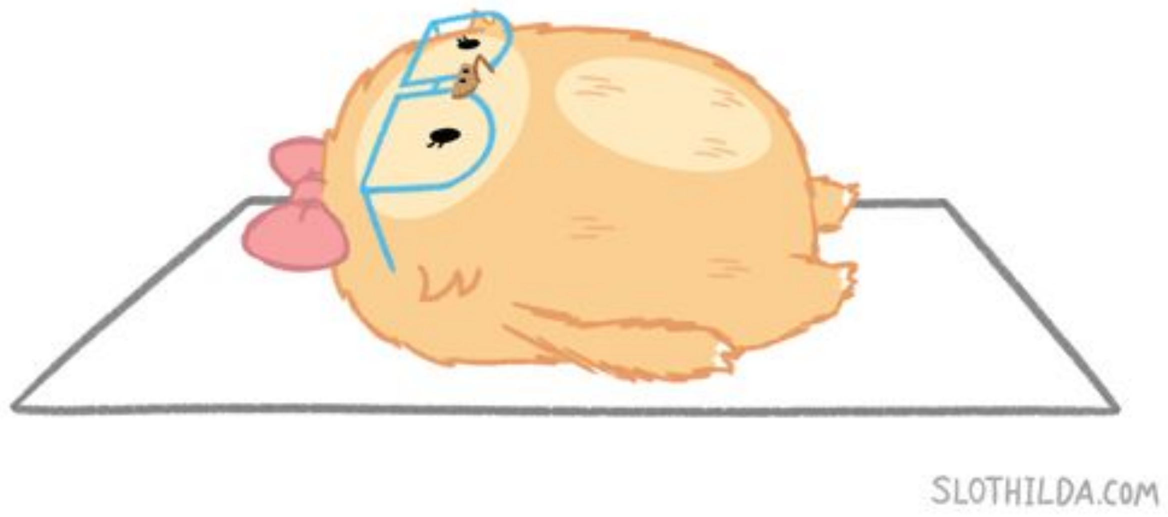
- Mix both of them

1	2	3	4
3	2	1	4
2	3	4	1
1	3	4	2




Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True

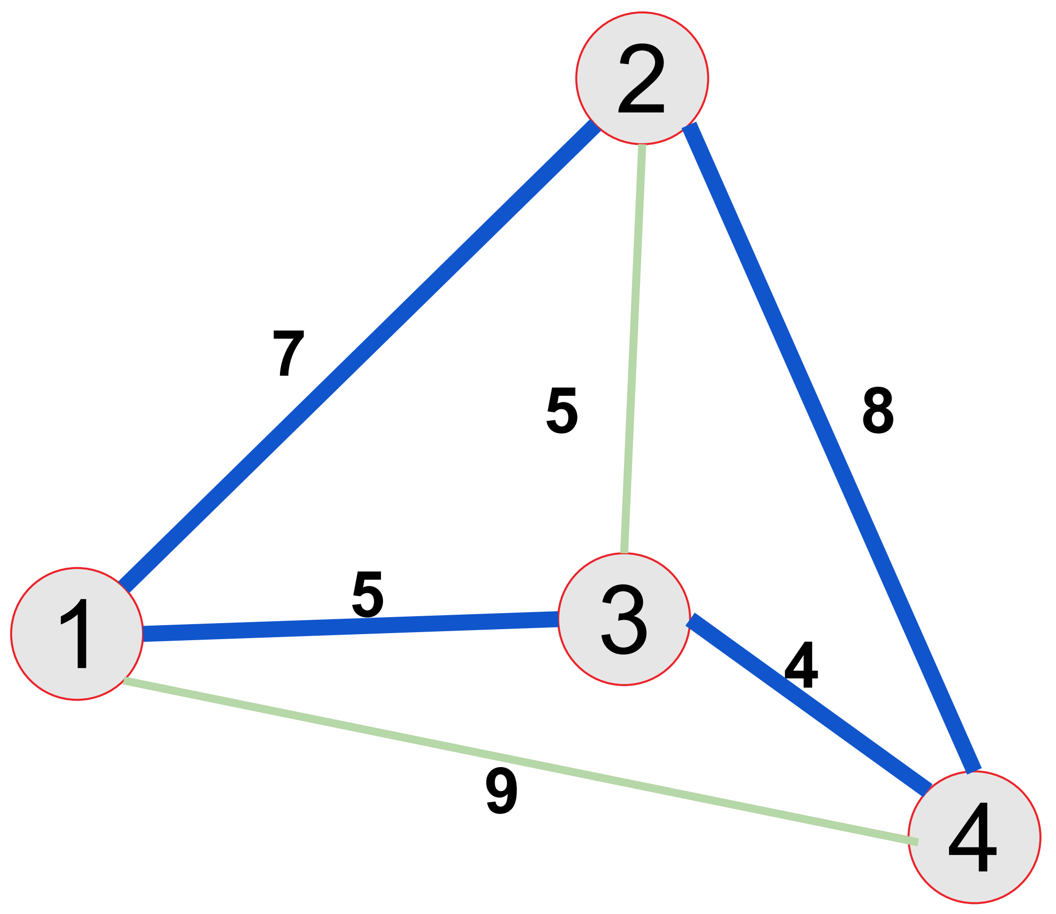

GETTIN' SWOLE

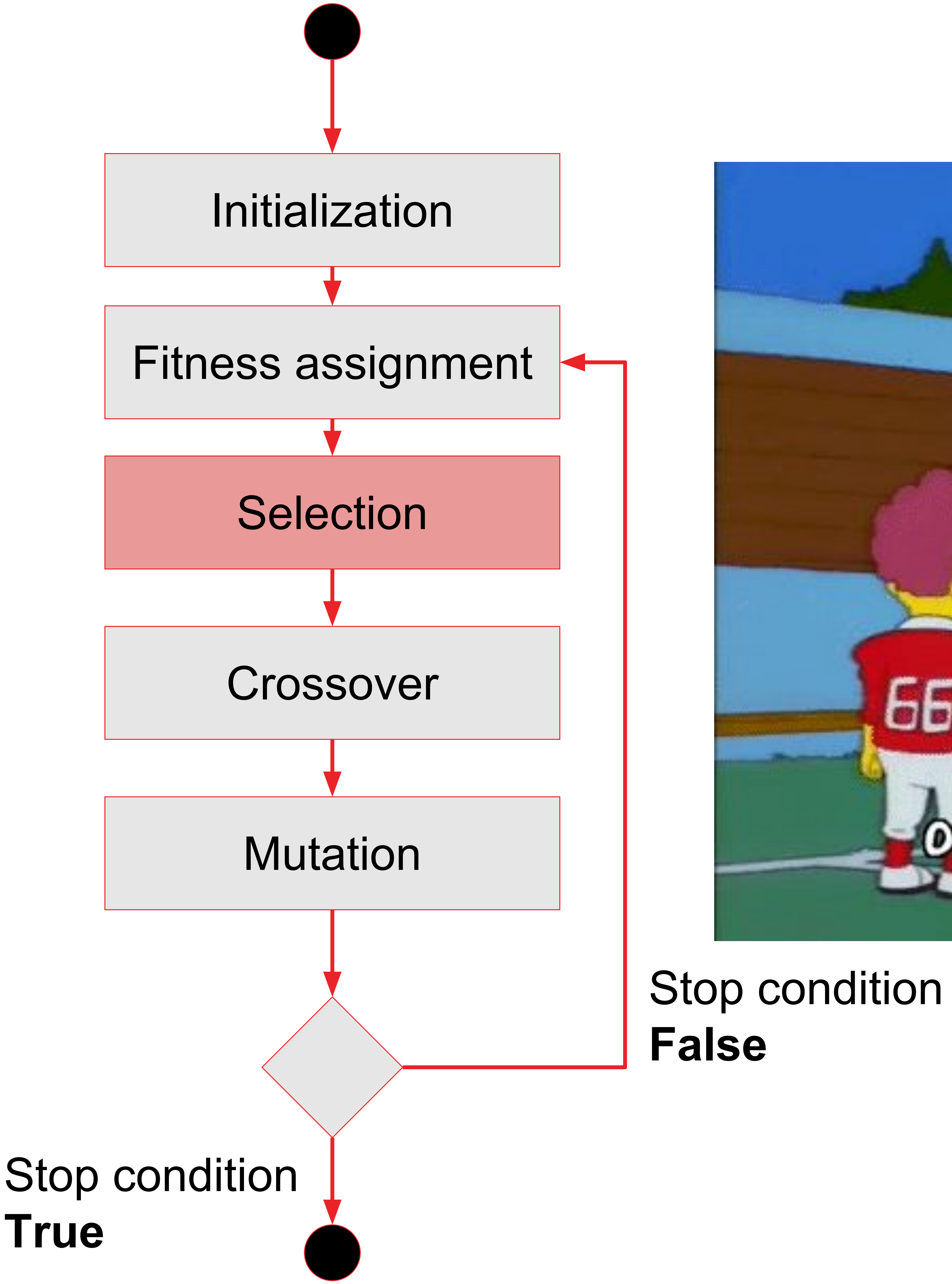

SLOTHILDA.COM

Fitness assignment

Function that evals the fitness of the generated solutions. This function is tightly coupled with the problem domain

1	2	3	4	25
1	3	4	2	24




Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True


OH, ROD, YOU DON'T HAVE TO.
YOU'RE CUT.

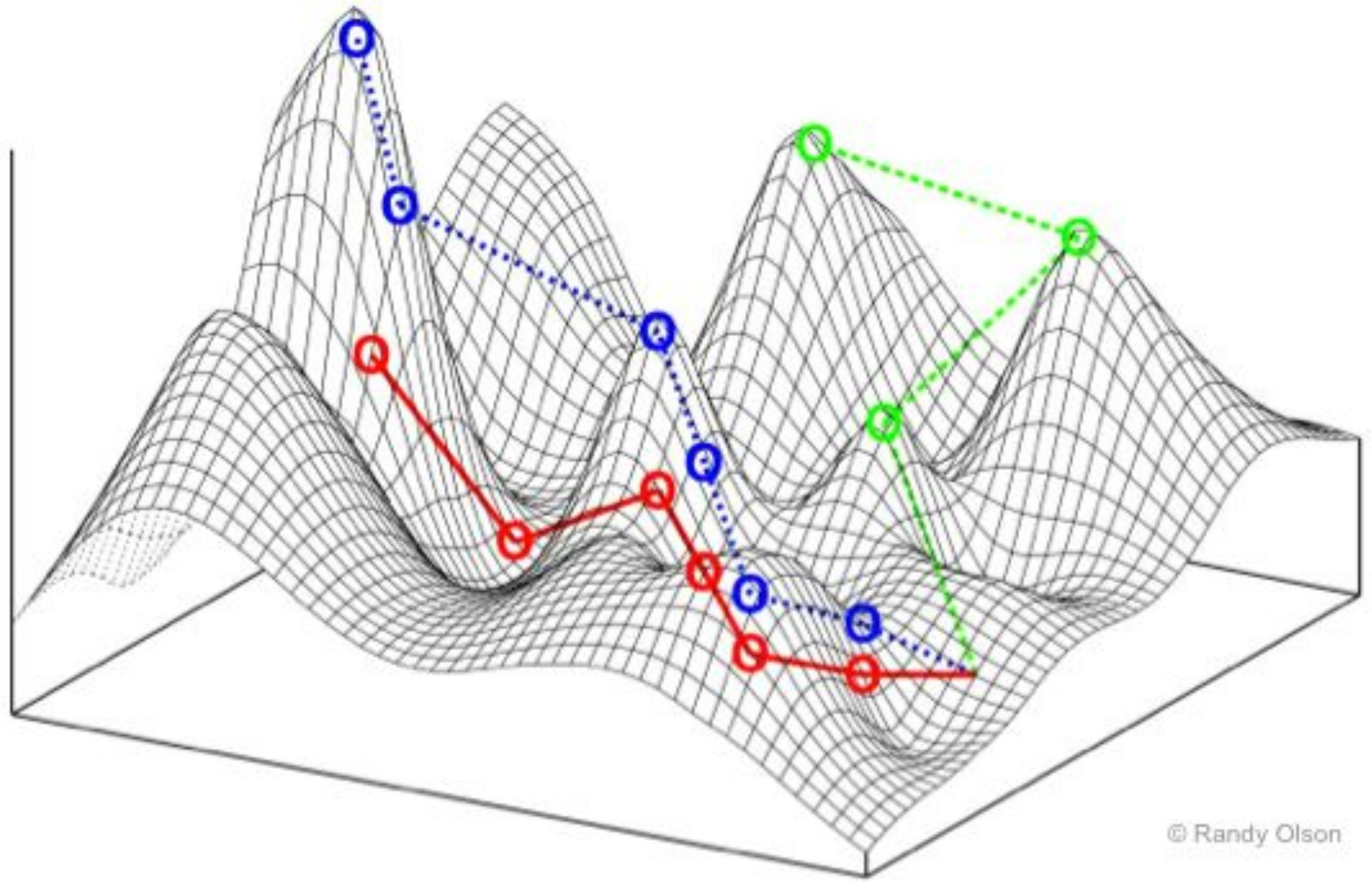
Selection

- Tries to choose individuals from the population by their “*quality*”

- Chosen individuals will be part of the offspring and named as mother/father

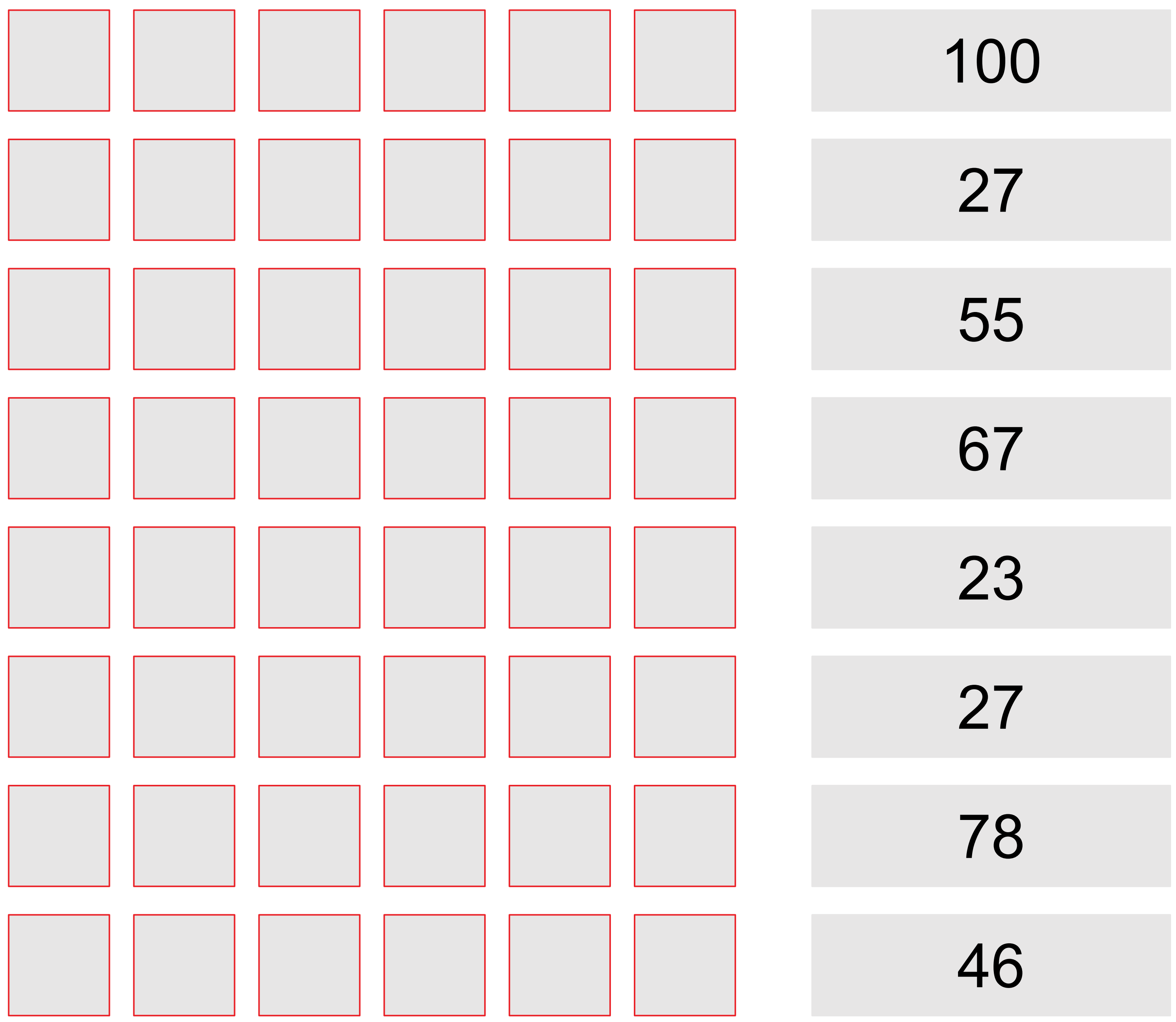
- Selection is probabilistic, hence individuals with a better value for the fitness function will usually survive, but individuals with a poor fitness function value can also survive. ***Selective Pressure***
  - **Elitism**: best N elements of the population are always selected
  - **Tradeoff**: Exploitation vs Exploration

- Some selection techniques:
  - Random N selection
  - Best N Selection
  - Tournament selection of size N
  - Roulette Selection of size N



Selection. *techniques*

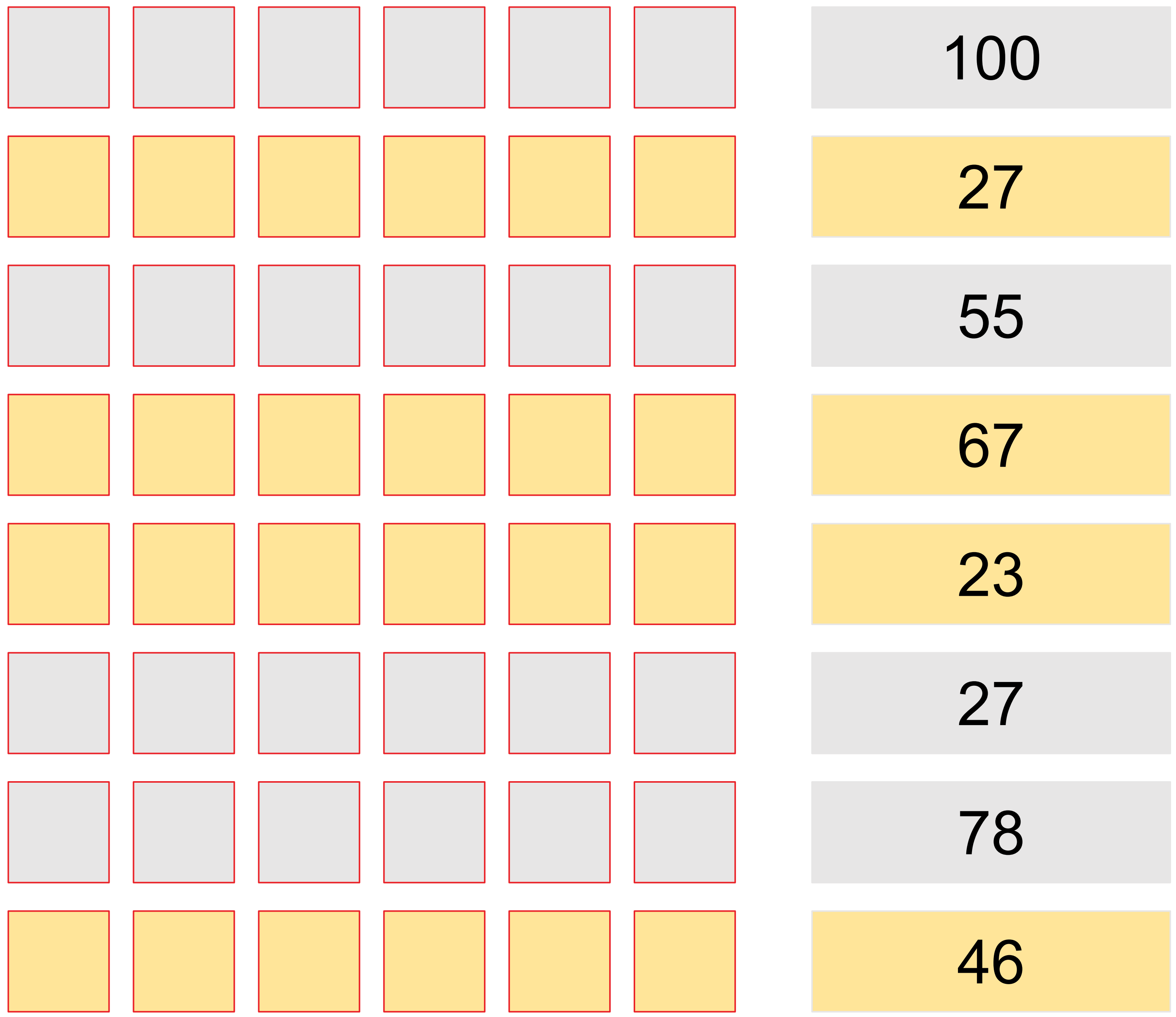
Random N selection



Best N Selection

Selection. *techniques*

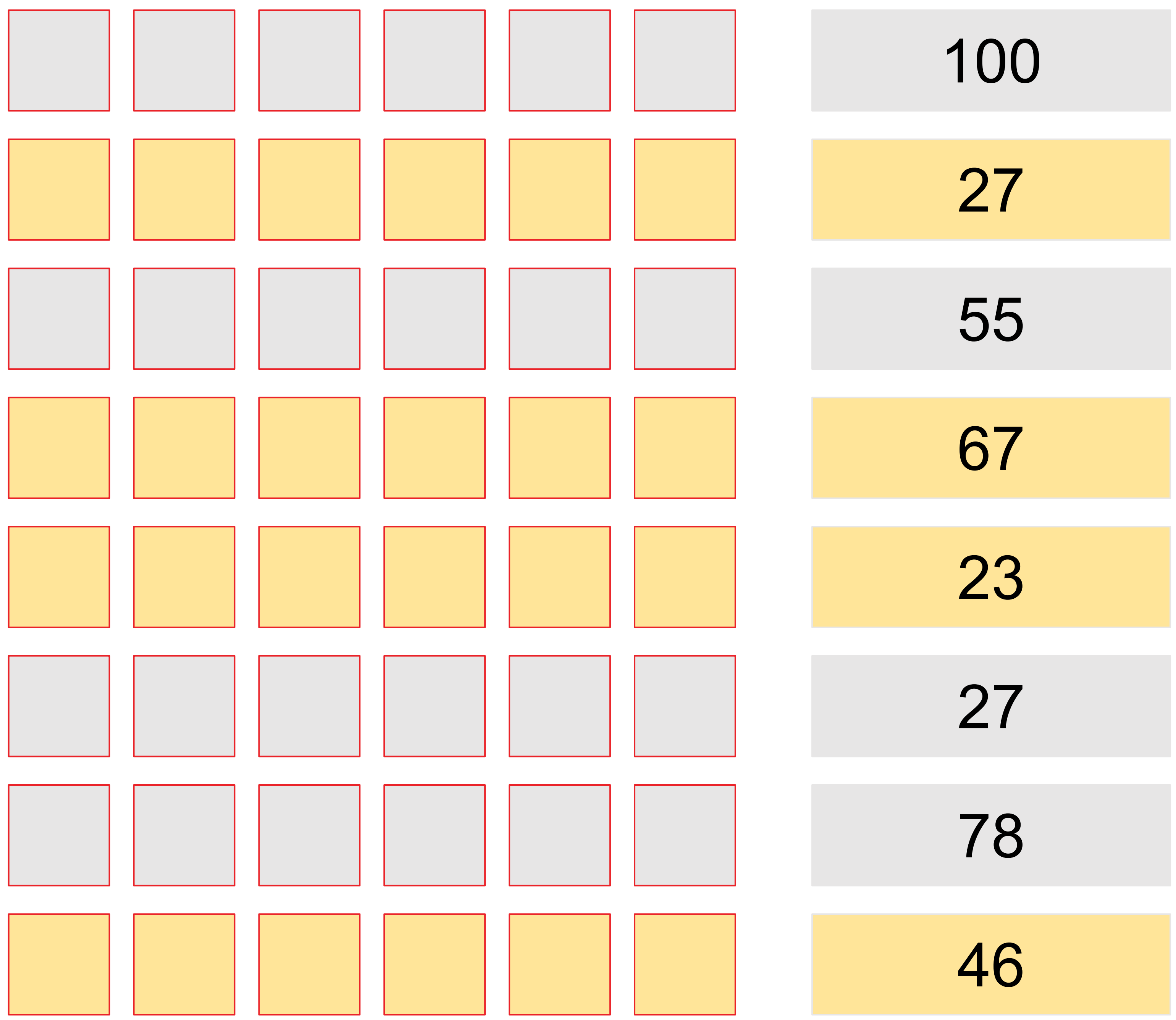
Random N selection (N = 4)



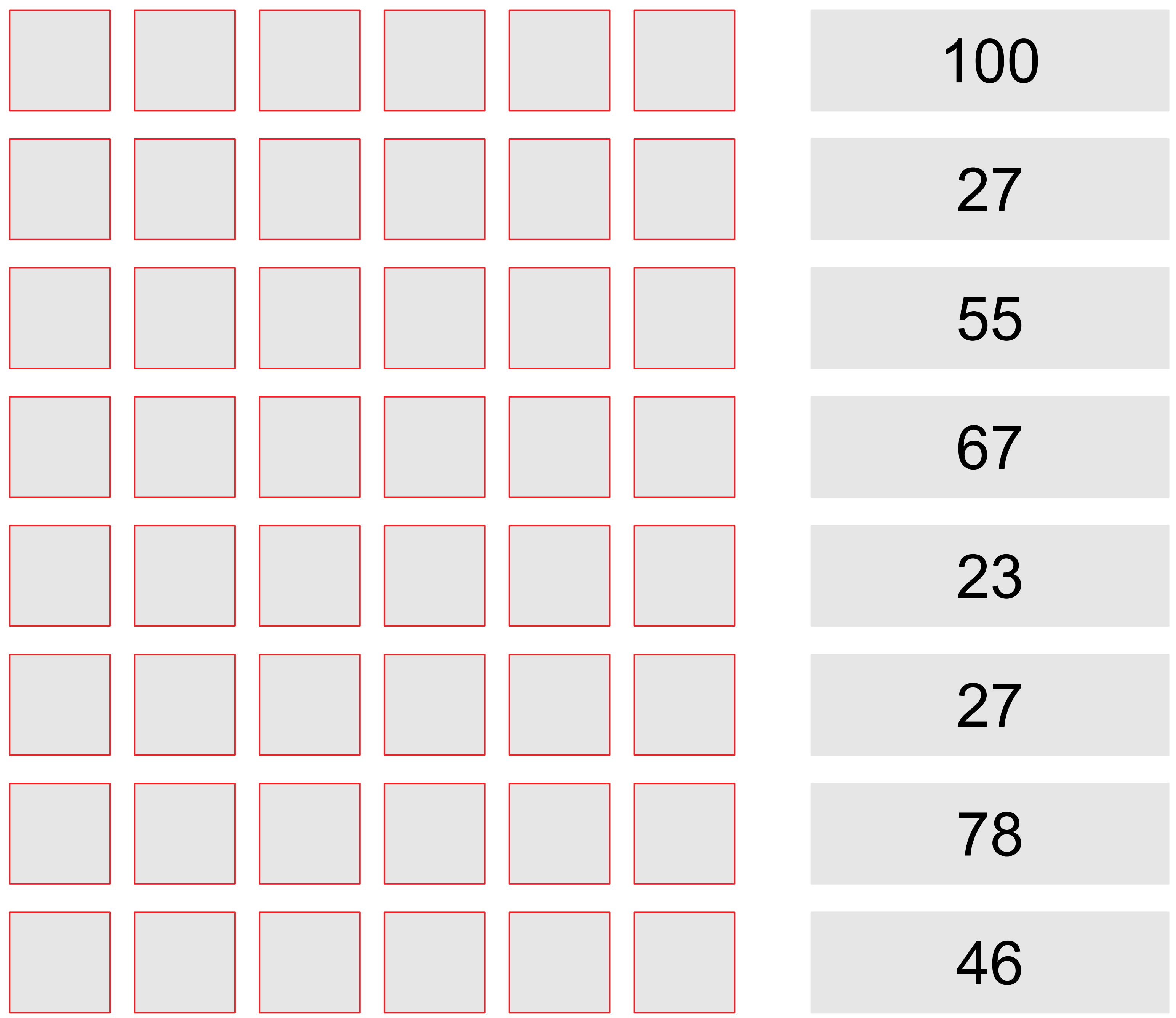
Best N Selection

Selection. *techniques*

Random N selection (N = 4)

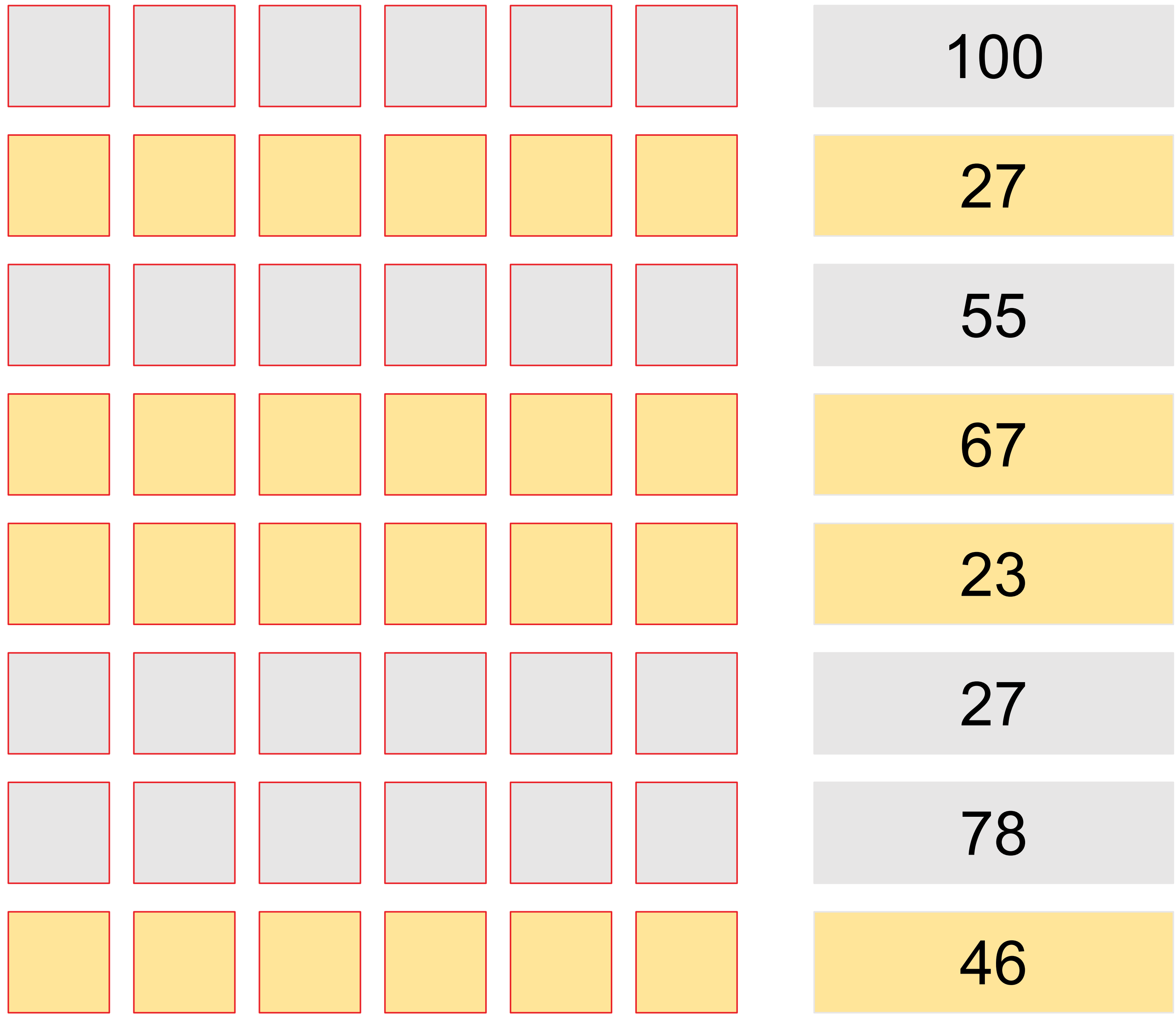


Best N Selection

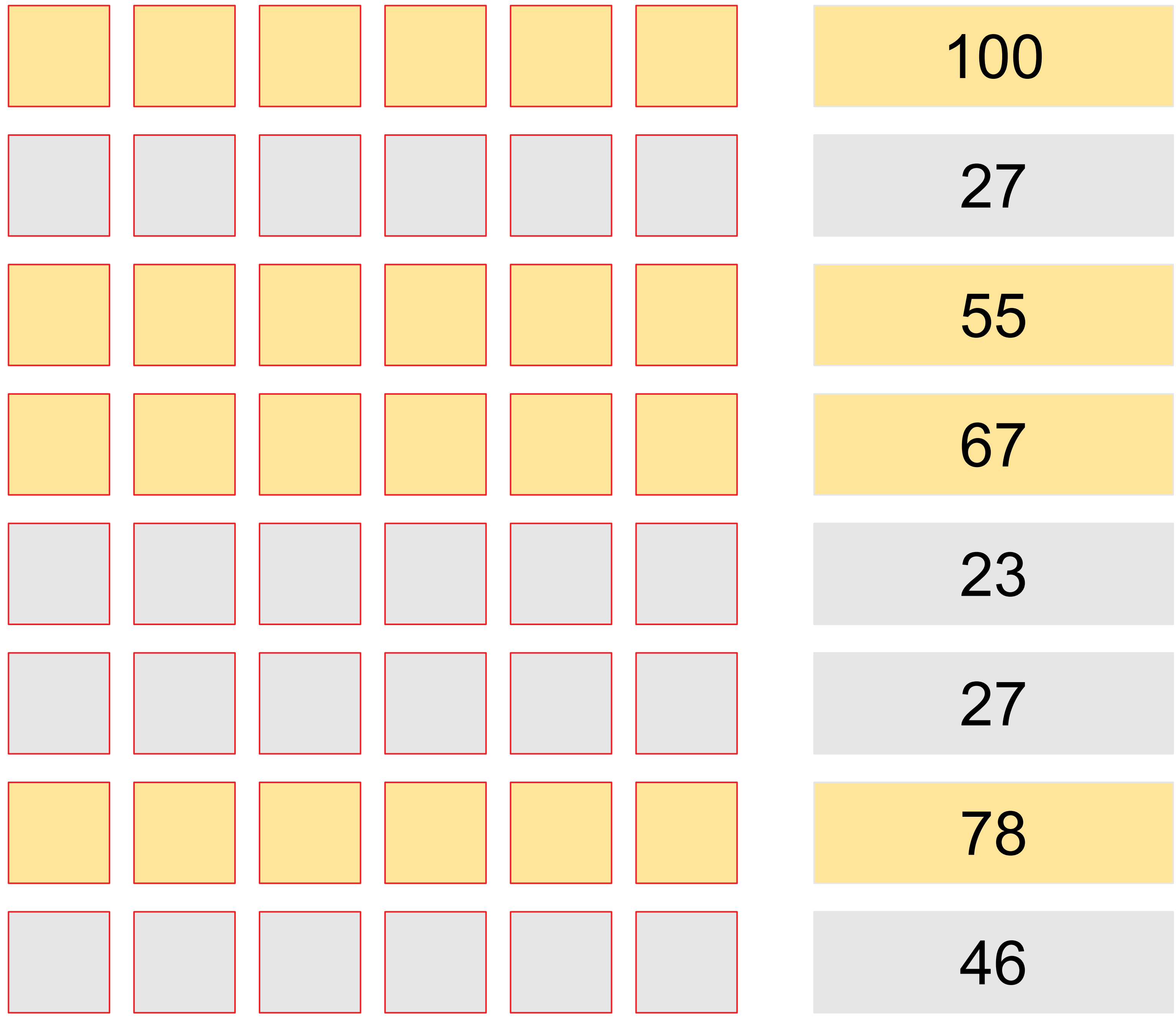


Selection. *techniques*

Random N selection (N = 4)

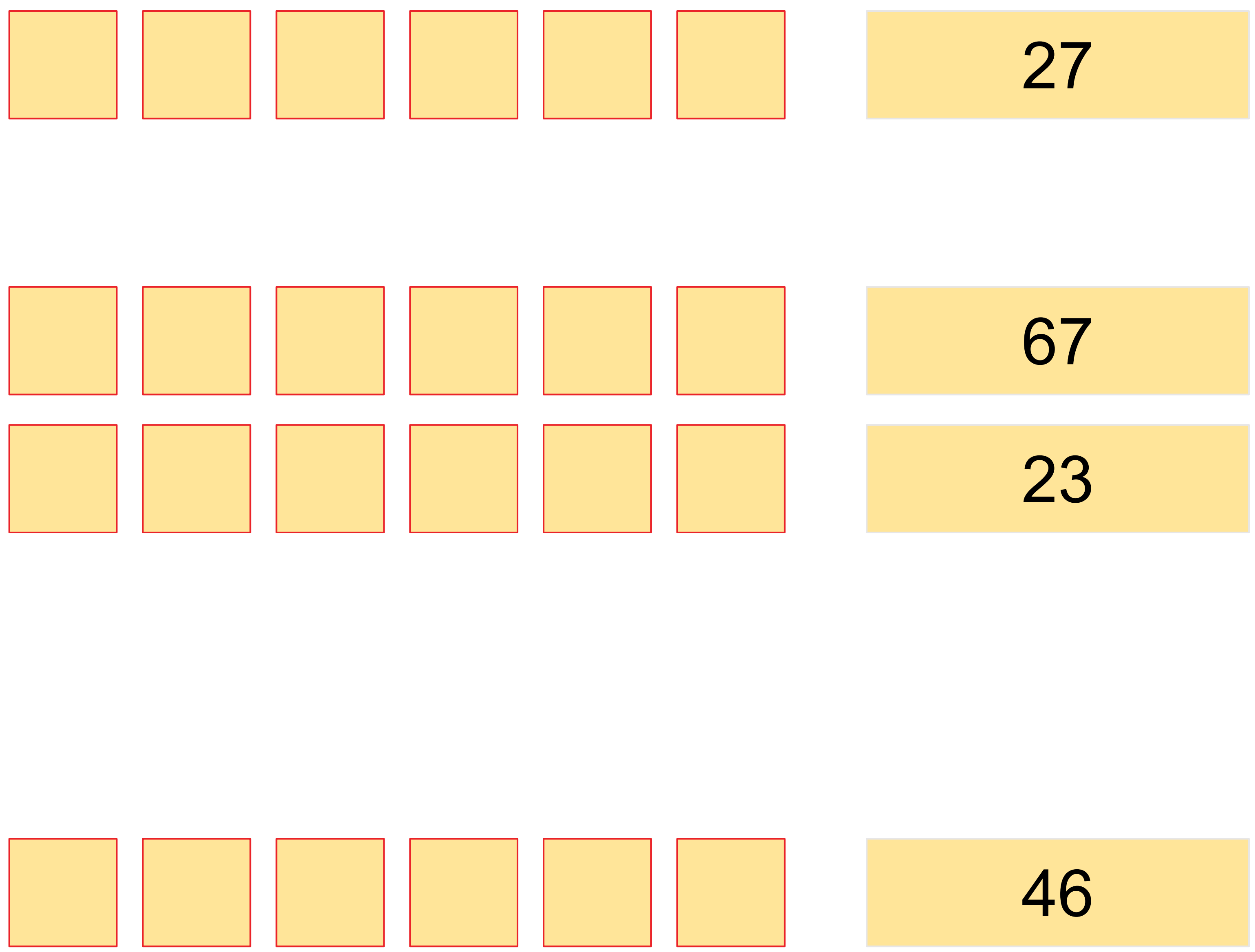


Best N Selection (N = 4)

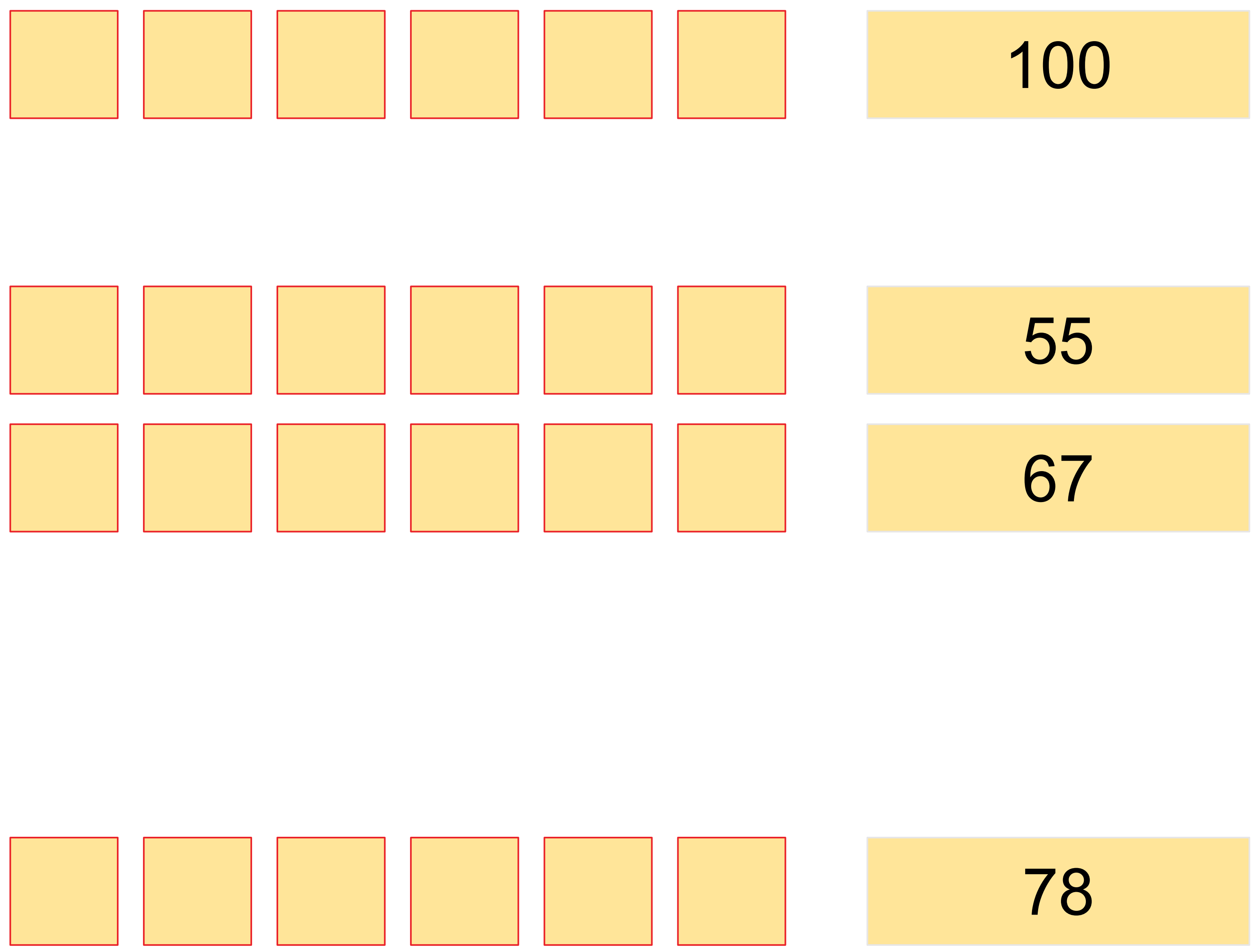


Selection. *techniques*

Random N selection (N = 4)



Best N Selection (N = 4)



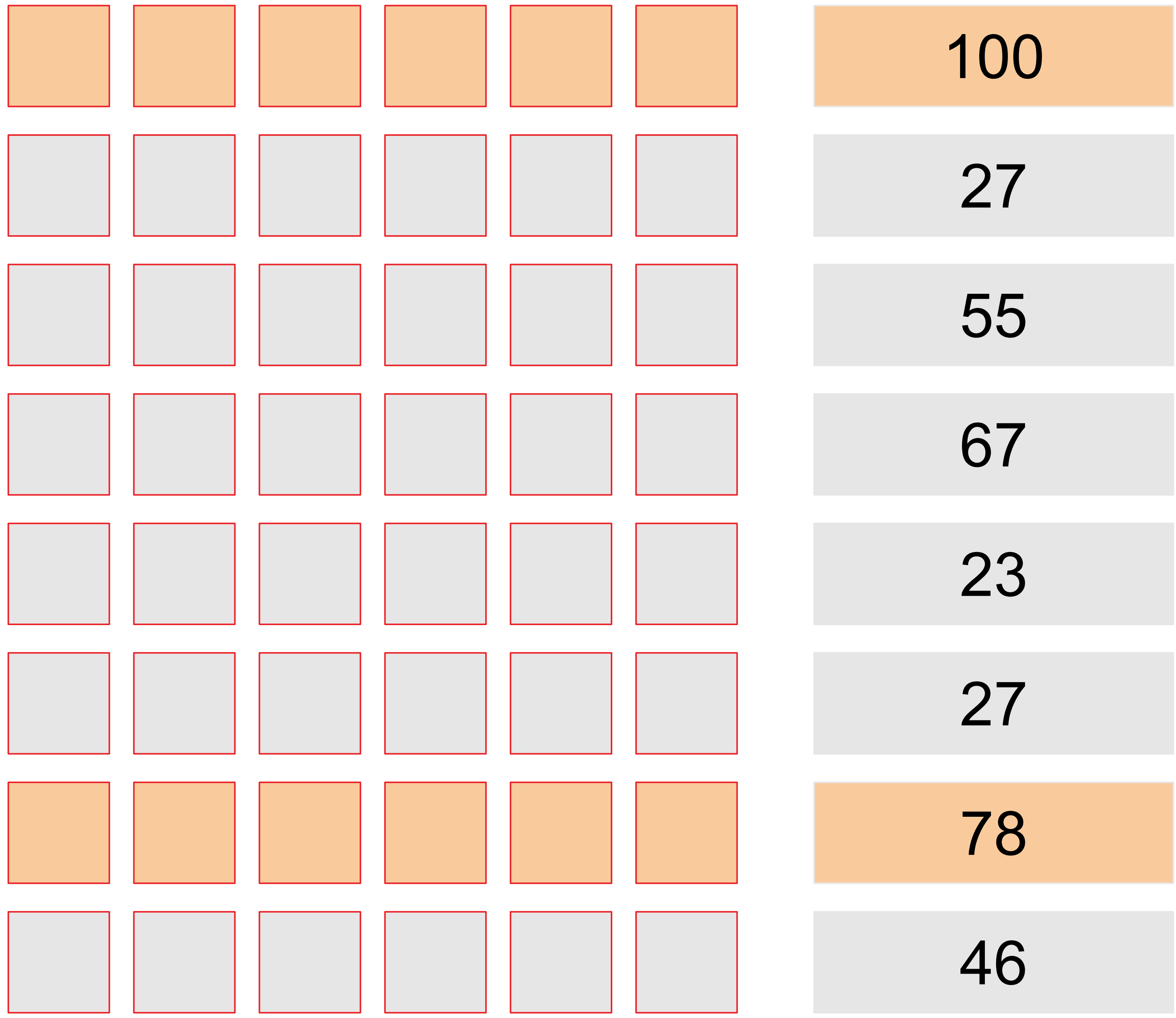
Selection. *techniques*

Tournament selection N (N = 2)

						100
						27
						55
						67
						23
						27
						78
						46

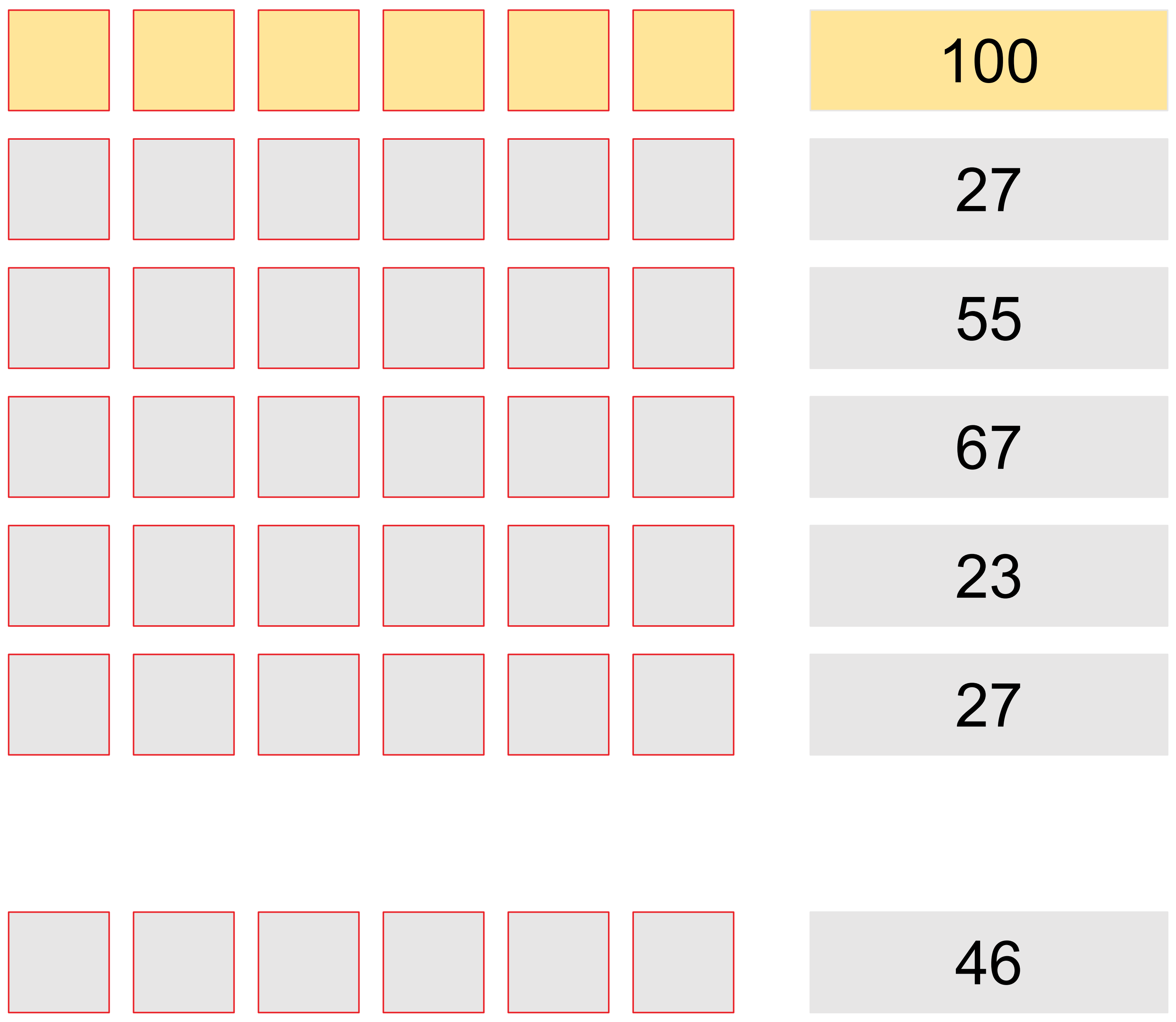
Selection. *techniques*

Tournament selection N (N = 2)



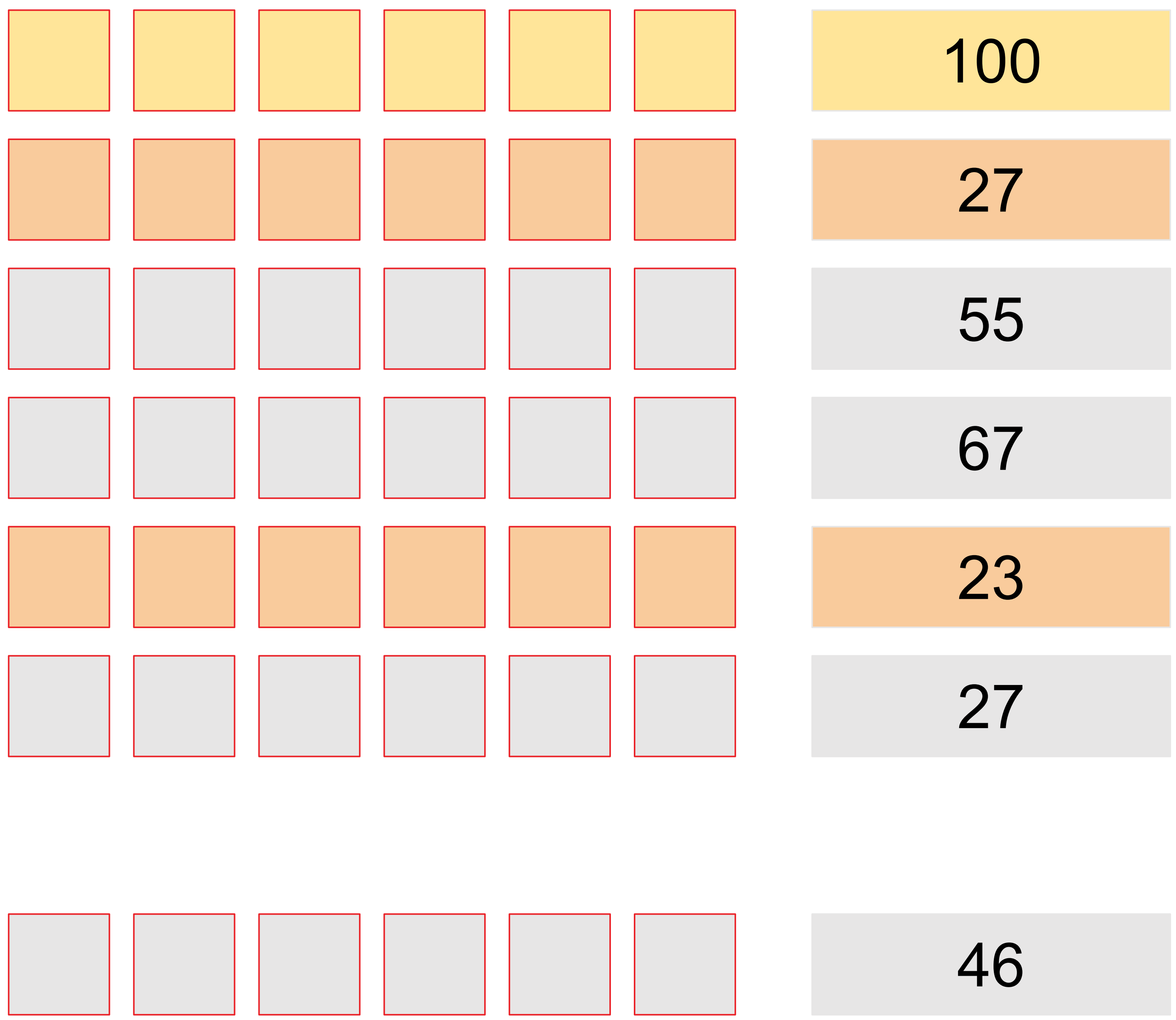
Selection. *techniques*

Tournament selection N (N = 2)



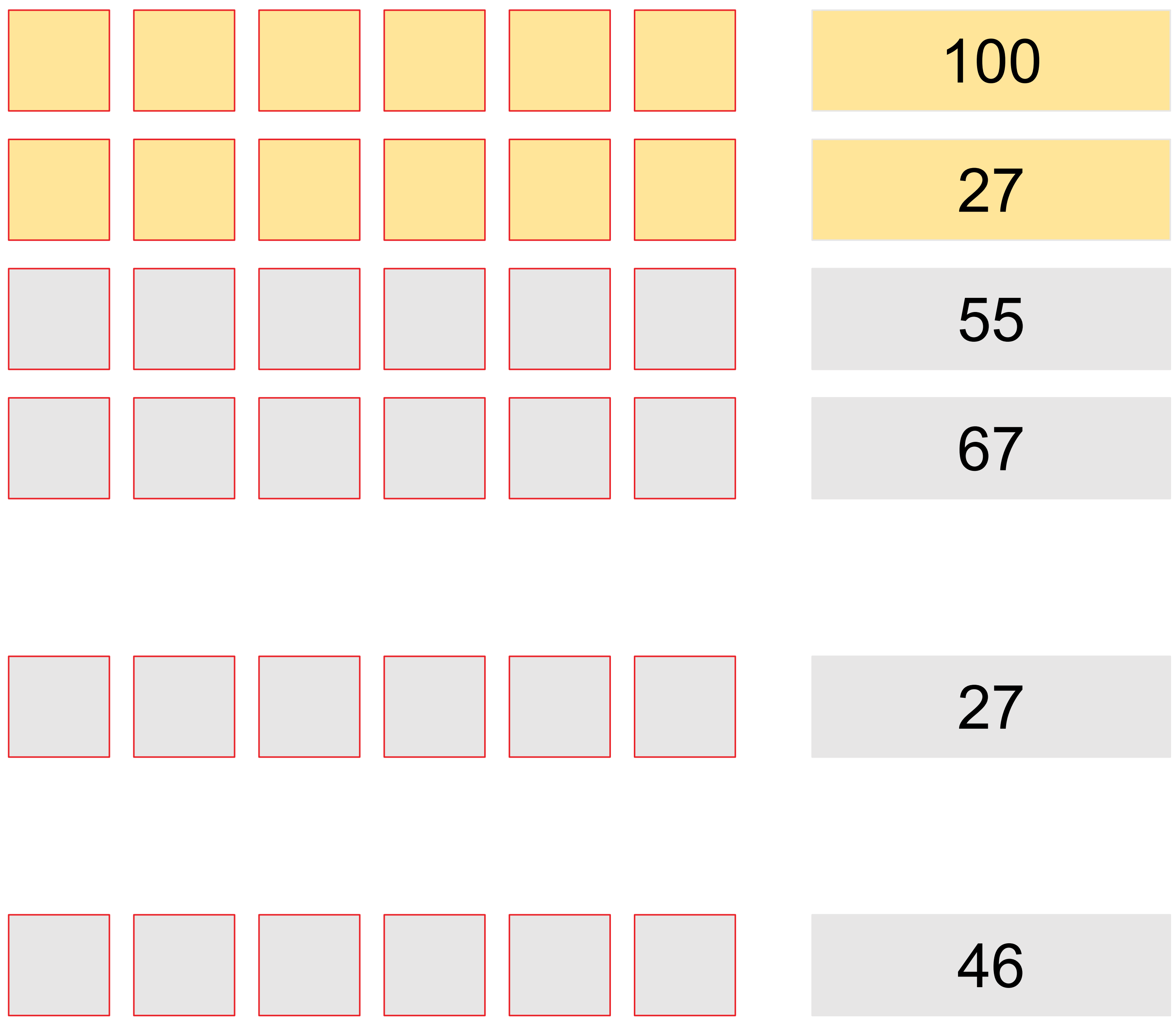
Selection. *techniques*

Tournament selection N (N = 2)



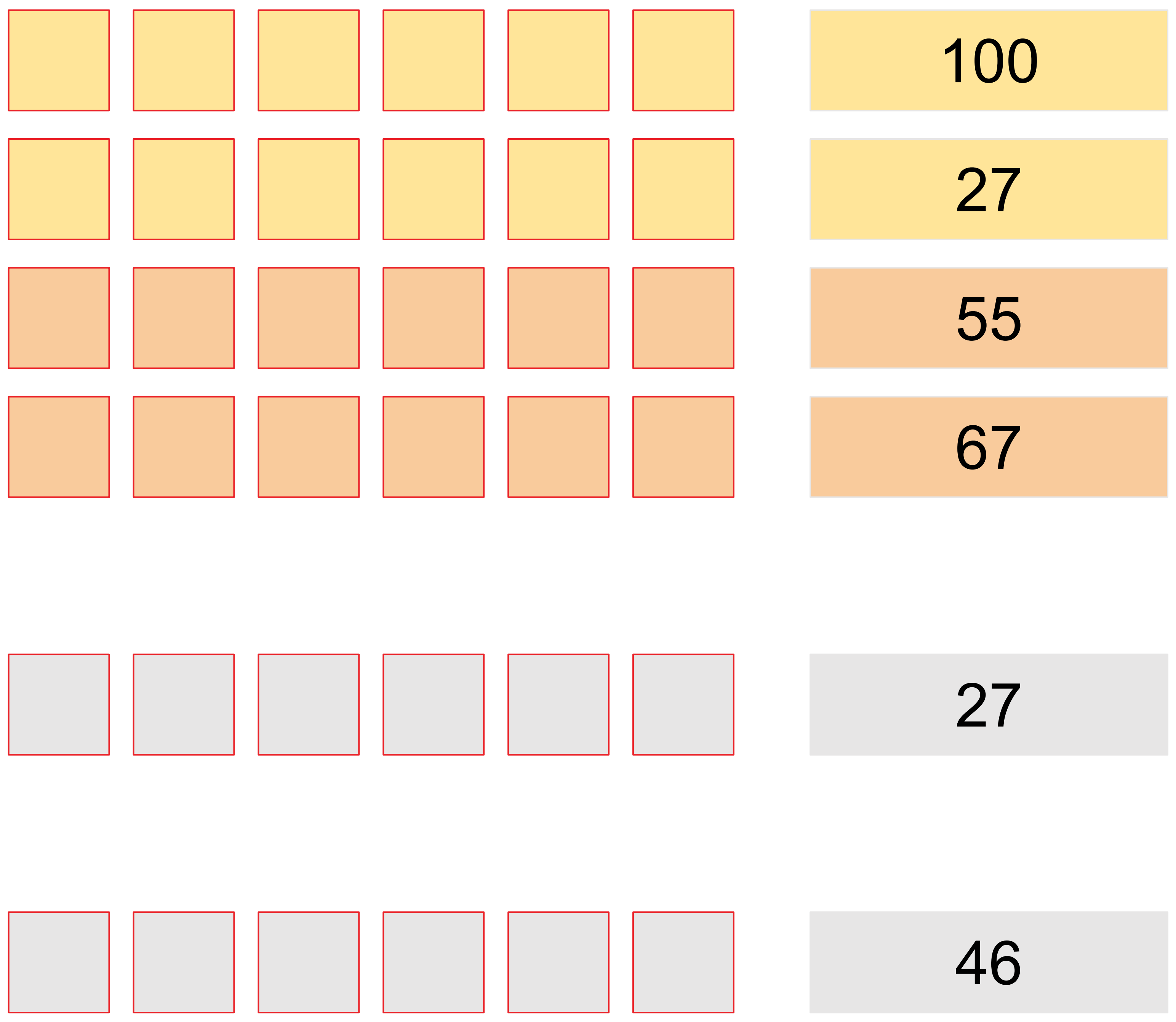
Selection. *techniques*

Tournament selection N (N = 2)



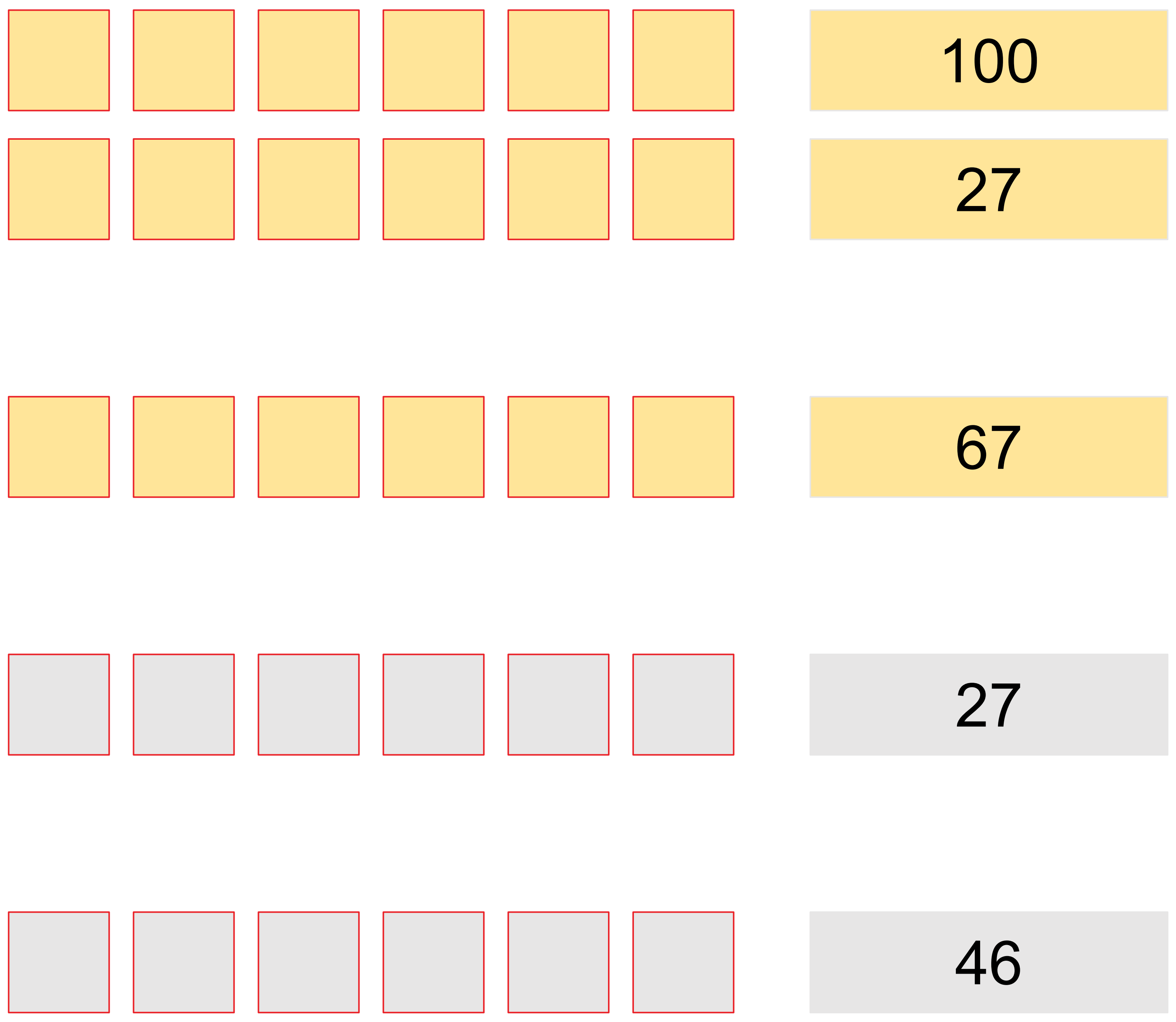
Selection. *techniques*

Tournament selection N (N = 2)



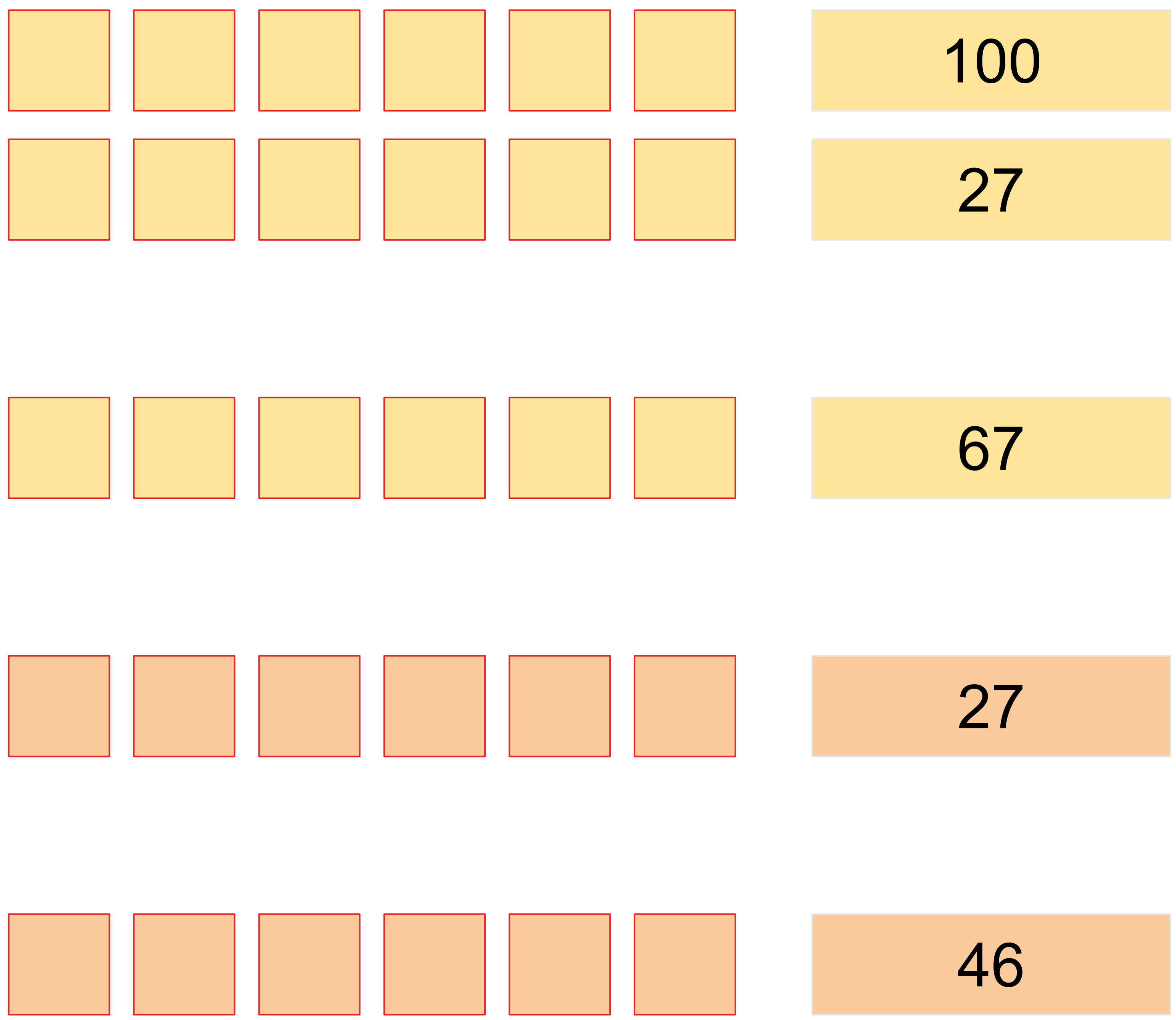
Selection. *techniques*

Tournament selection N (N = 2)



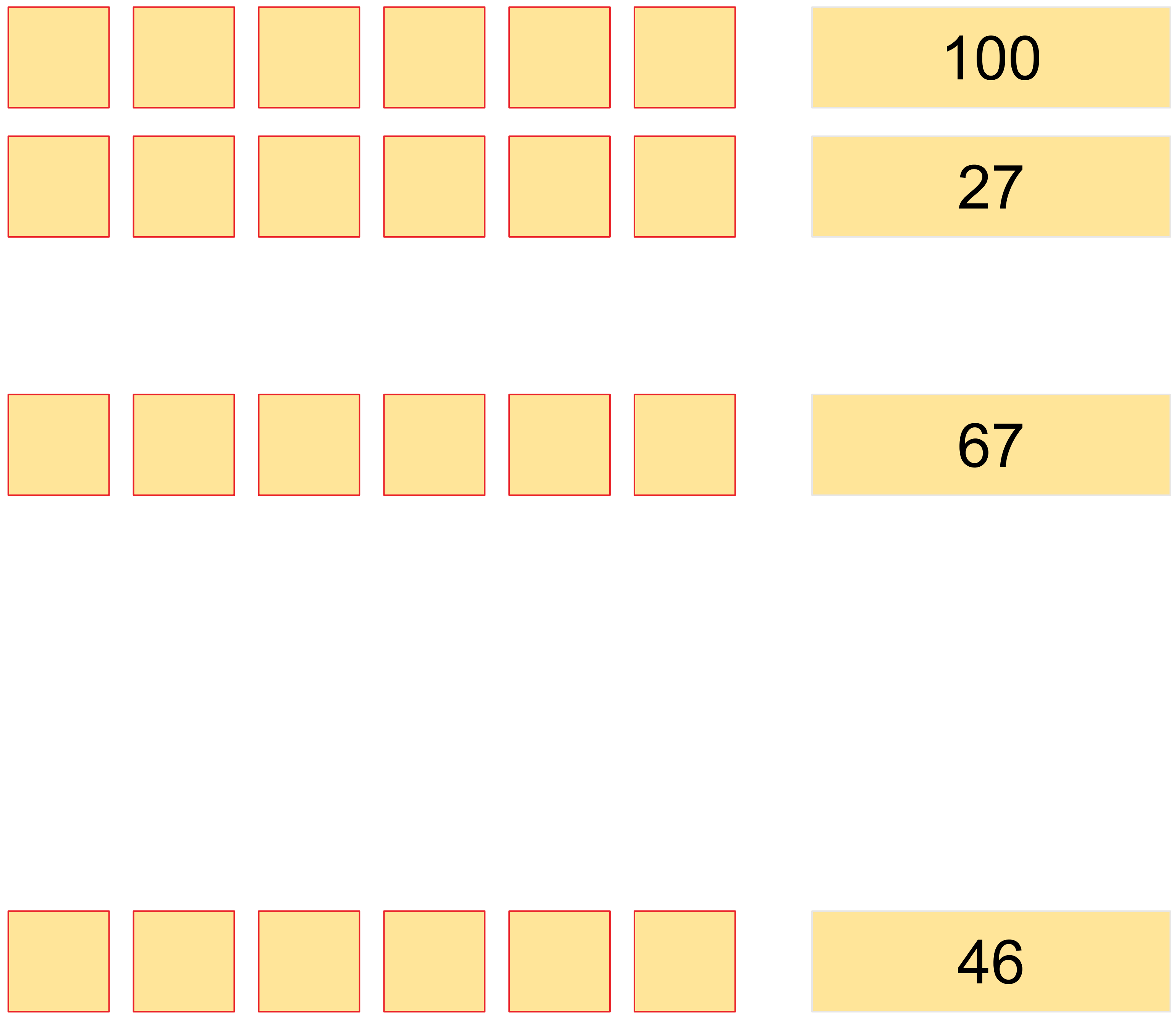
Selection. *techniques*

Tournament selection N (N = 2)



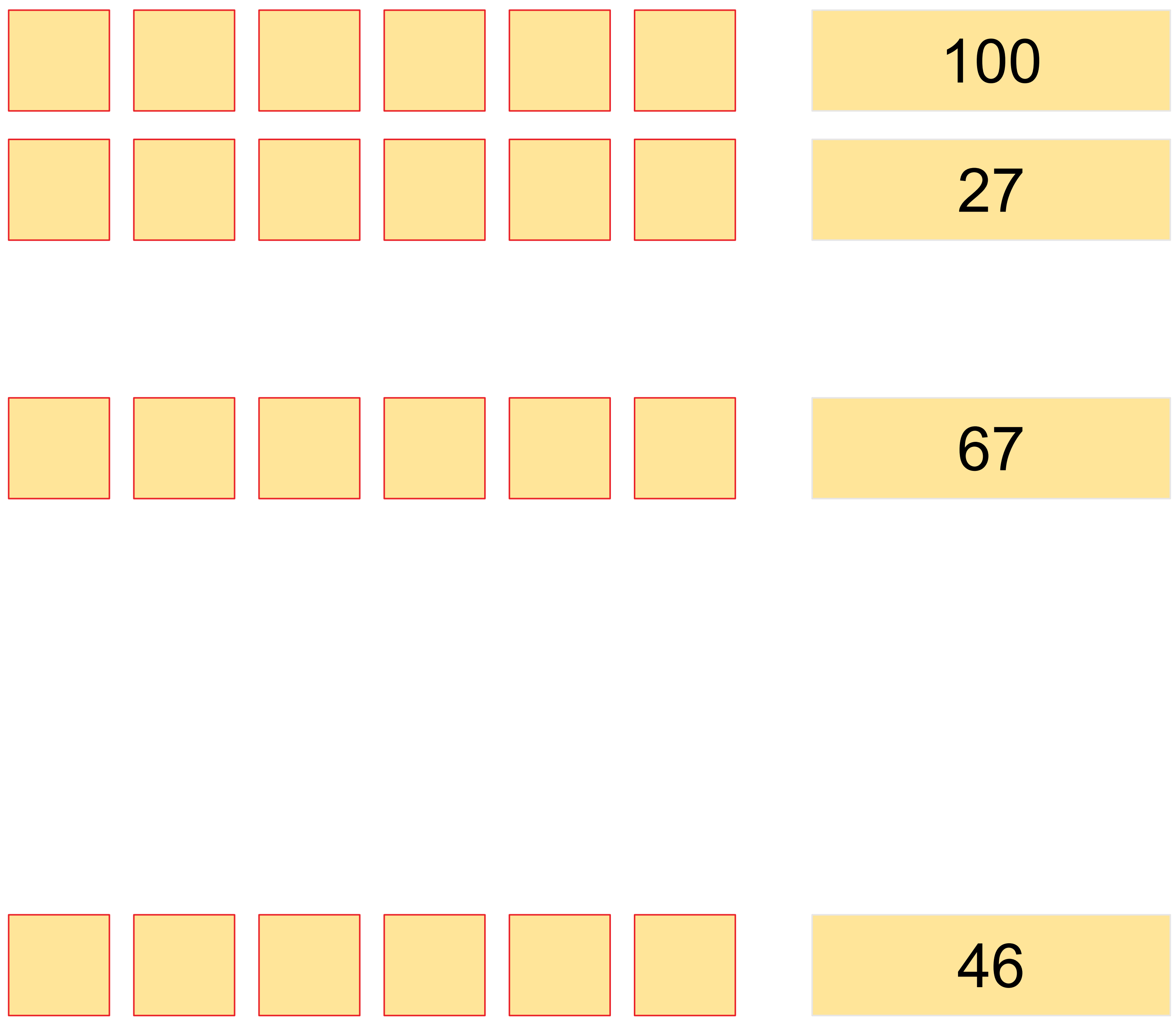
Selection. *techniques*

Tournament selection N (N = 2)

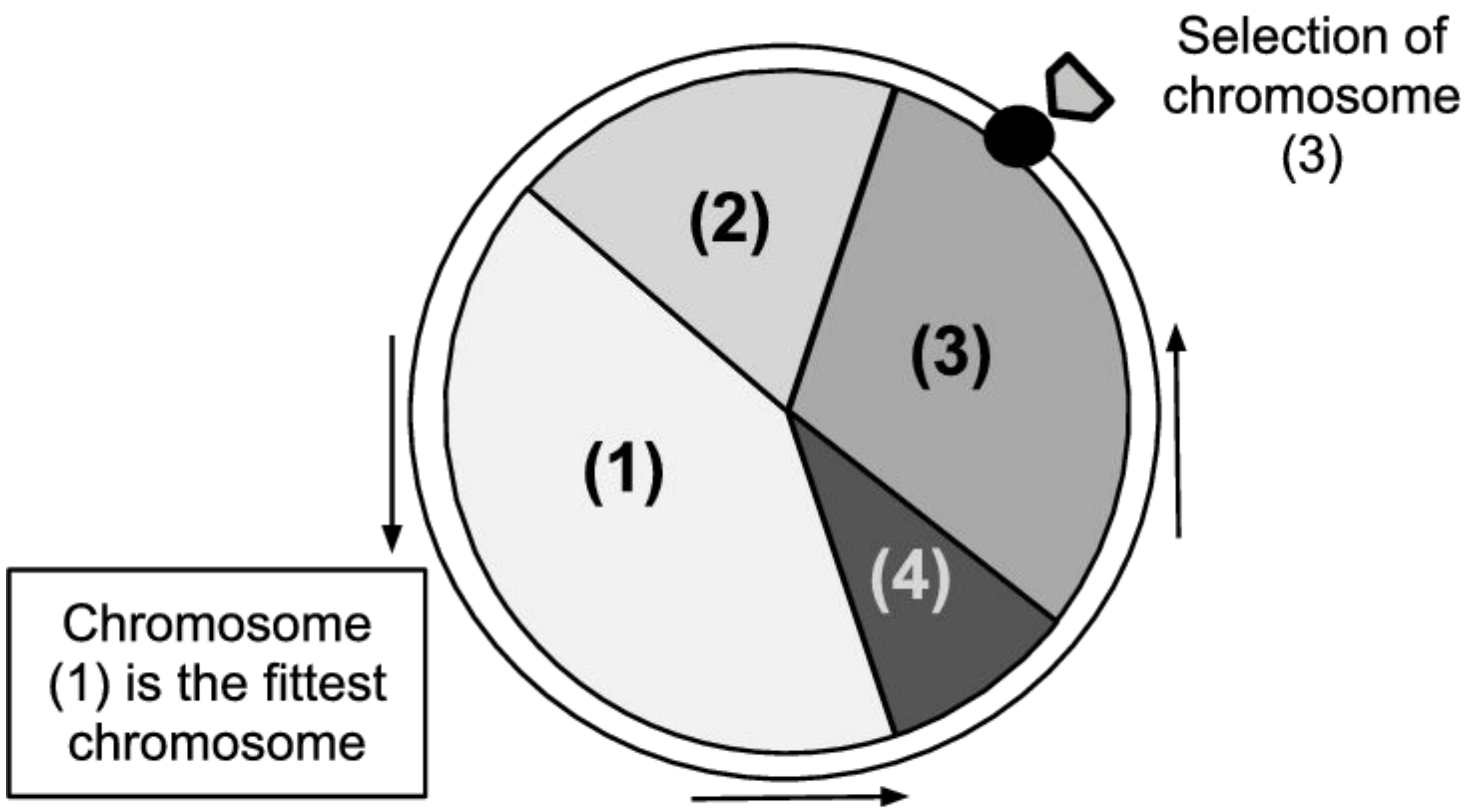


Selection. *techniques*

Tournament selection N (N = 2)

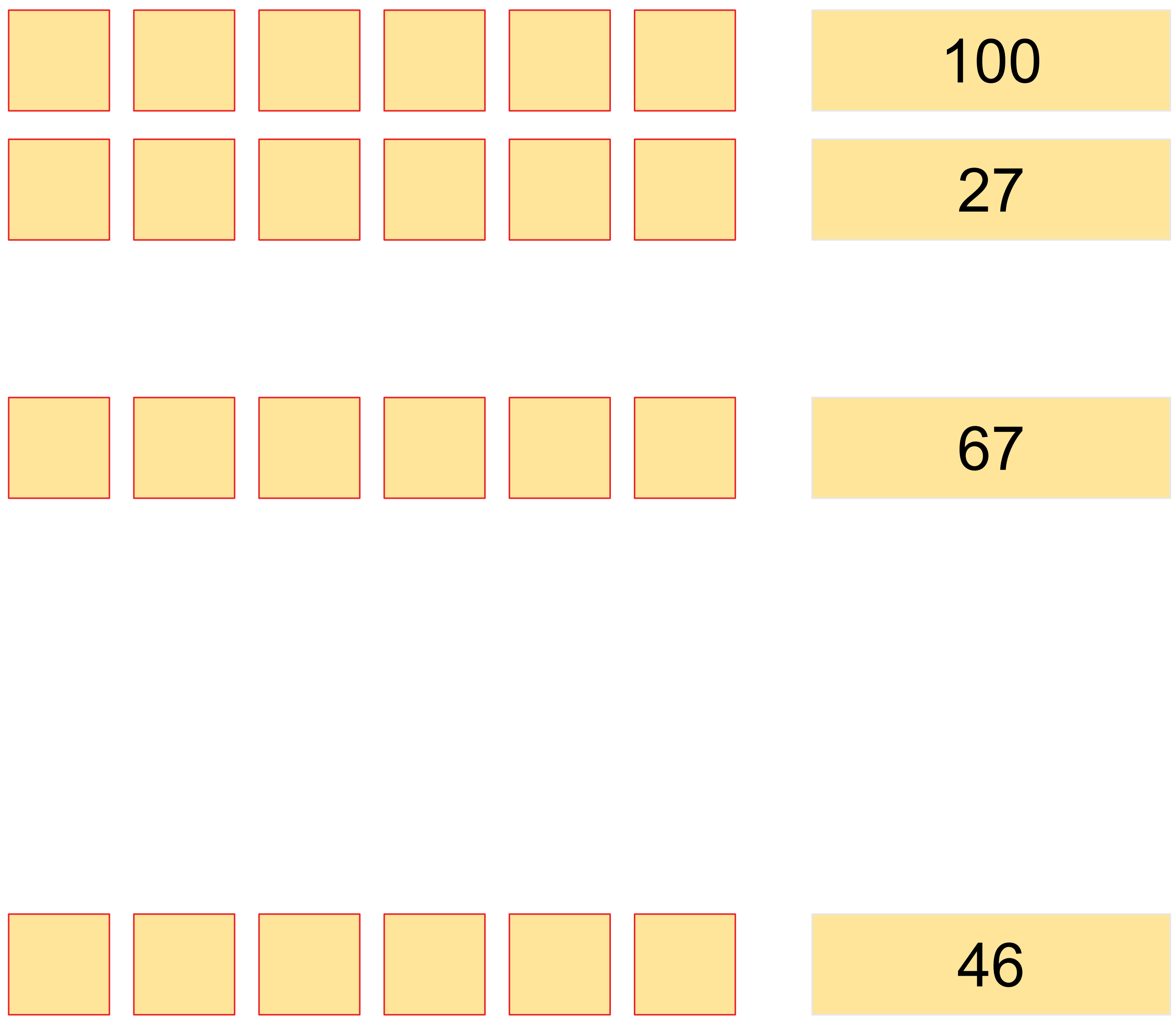


Roulette Selection N (N = 4)

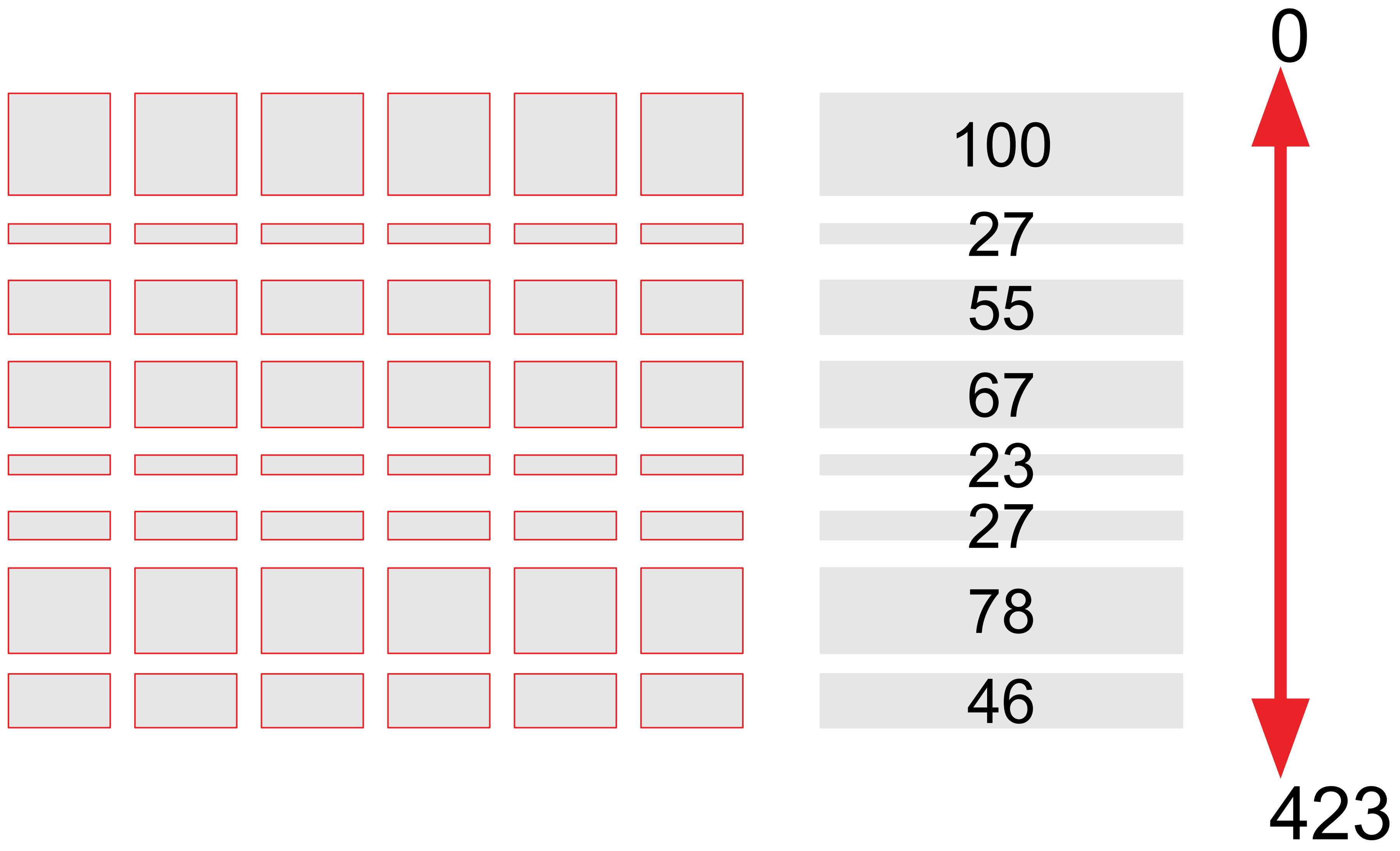


Selection. *techniques*

Tournament selection N (N = 2)



Roulette Selection N (N = 2)



Selection. *techniques*

Tournament selection N (N = 2)

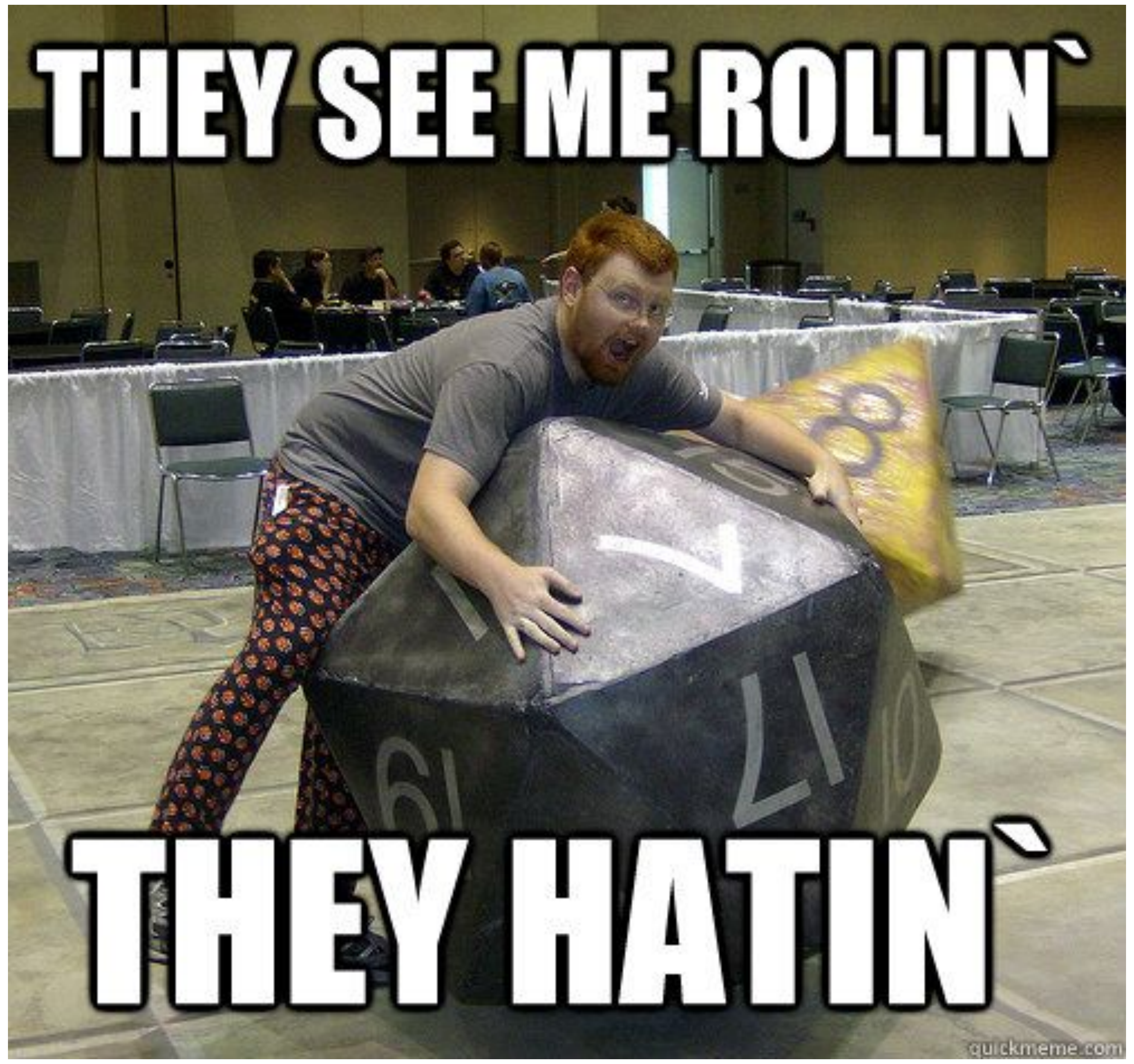
100

27

67

46

Roulette Selection N (N = 2)



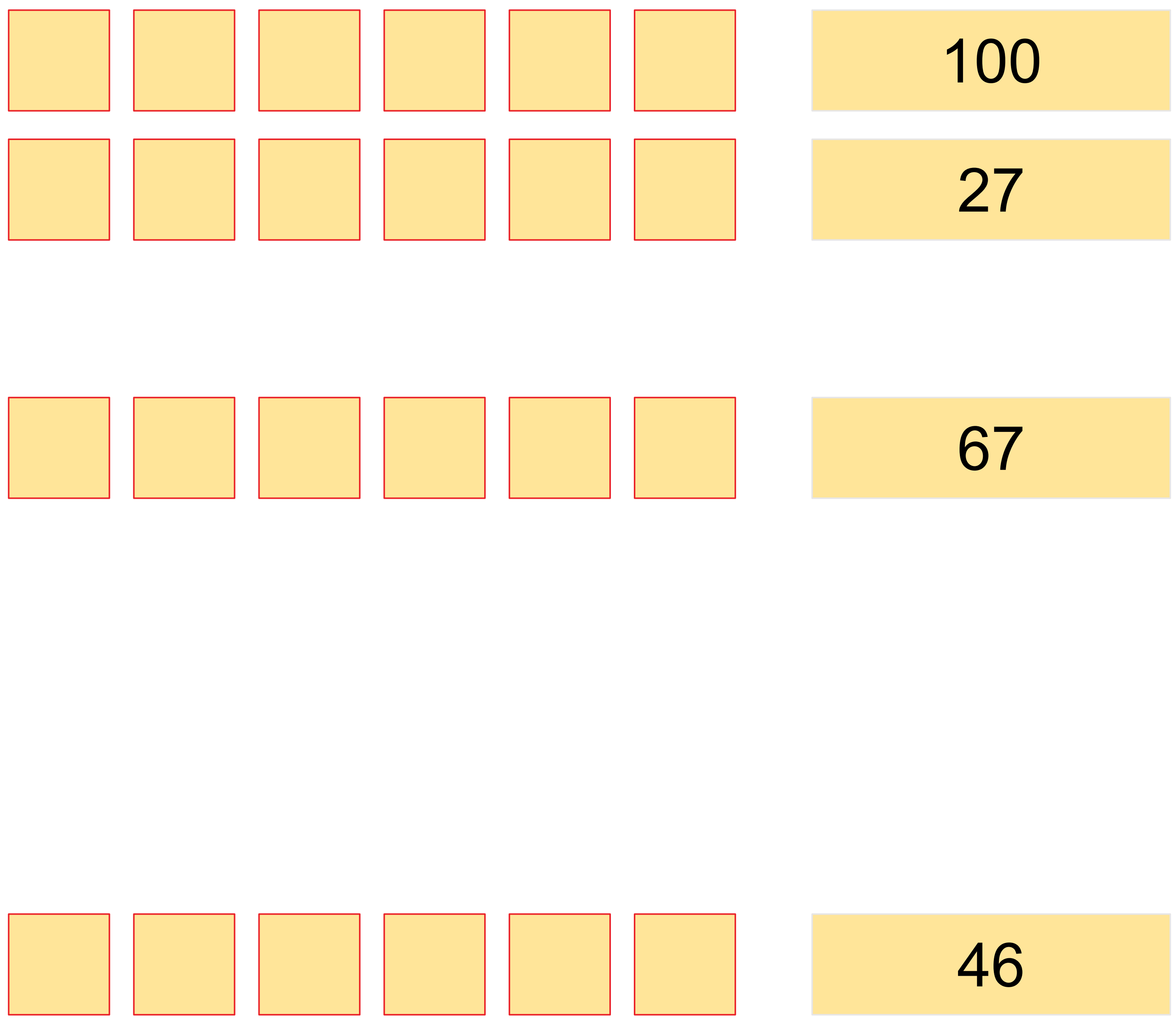
100
27
55
67
23
27
78
46

0

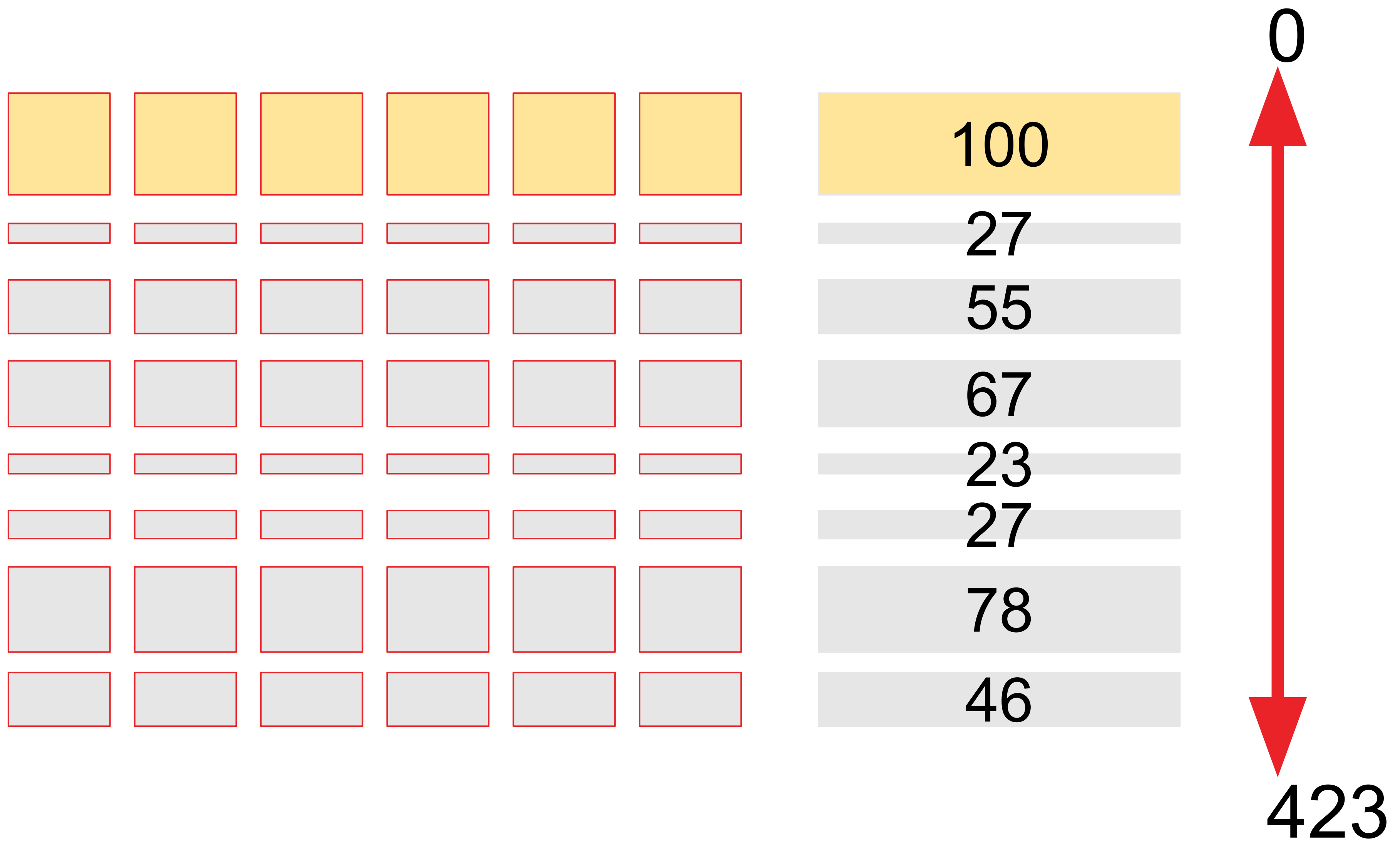
423

Selection. *techniques*

Tournament selection N (N = 2)

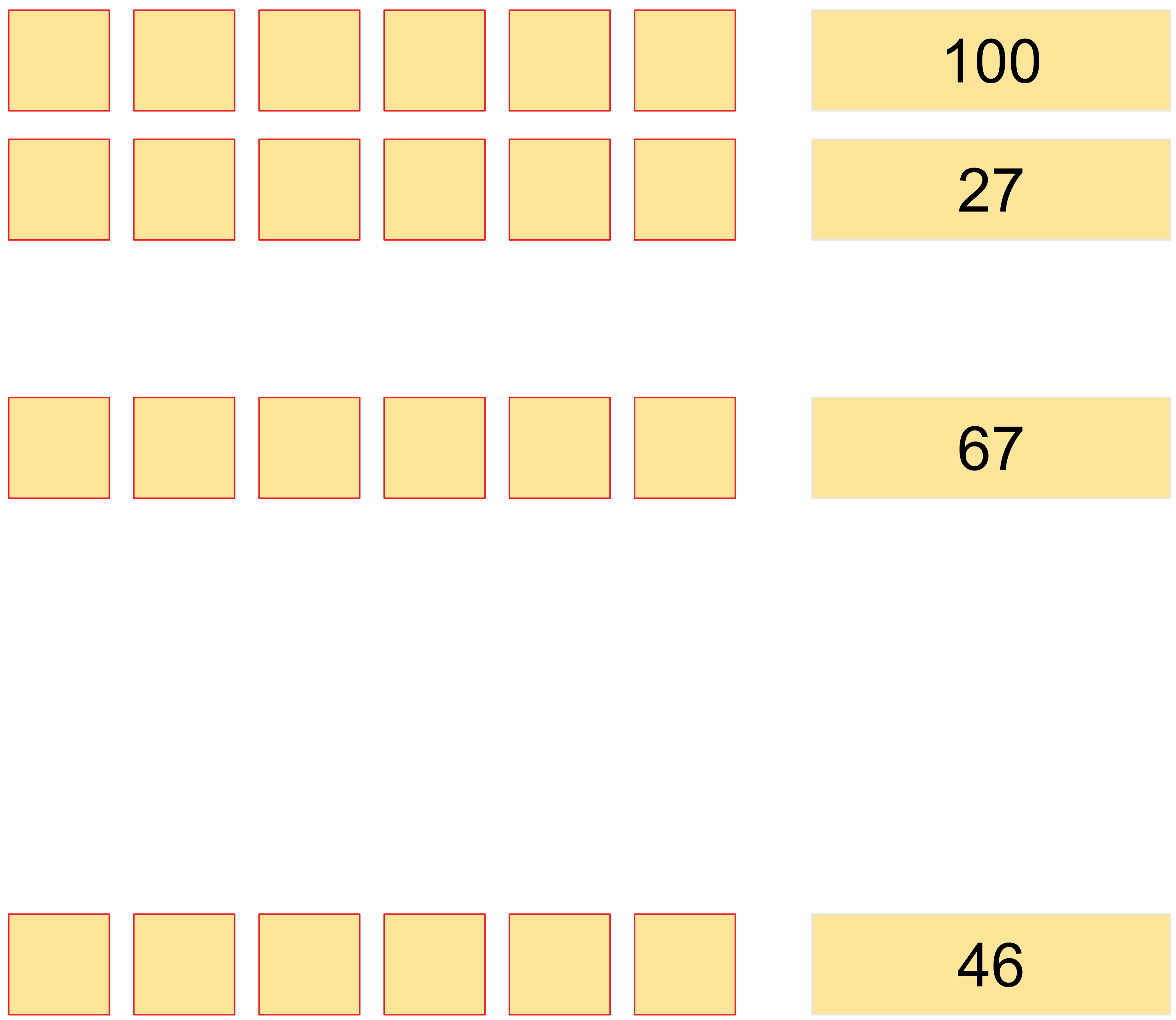


Roulette Selection N (N = 2)

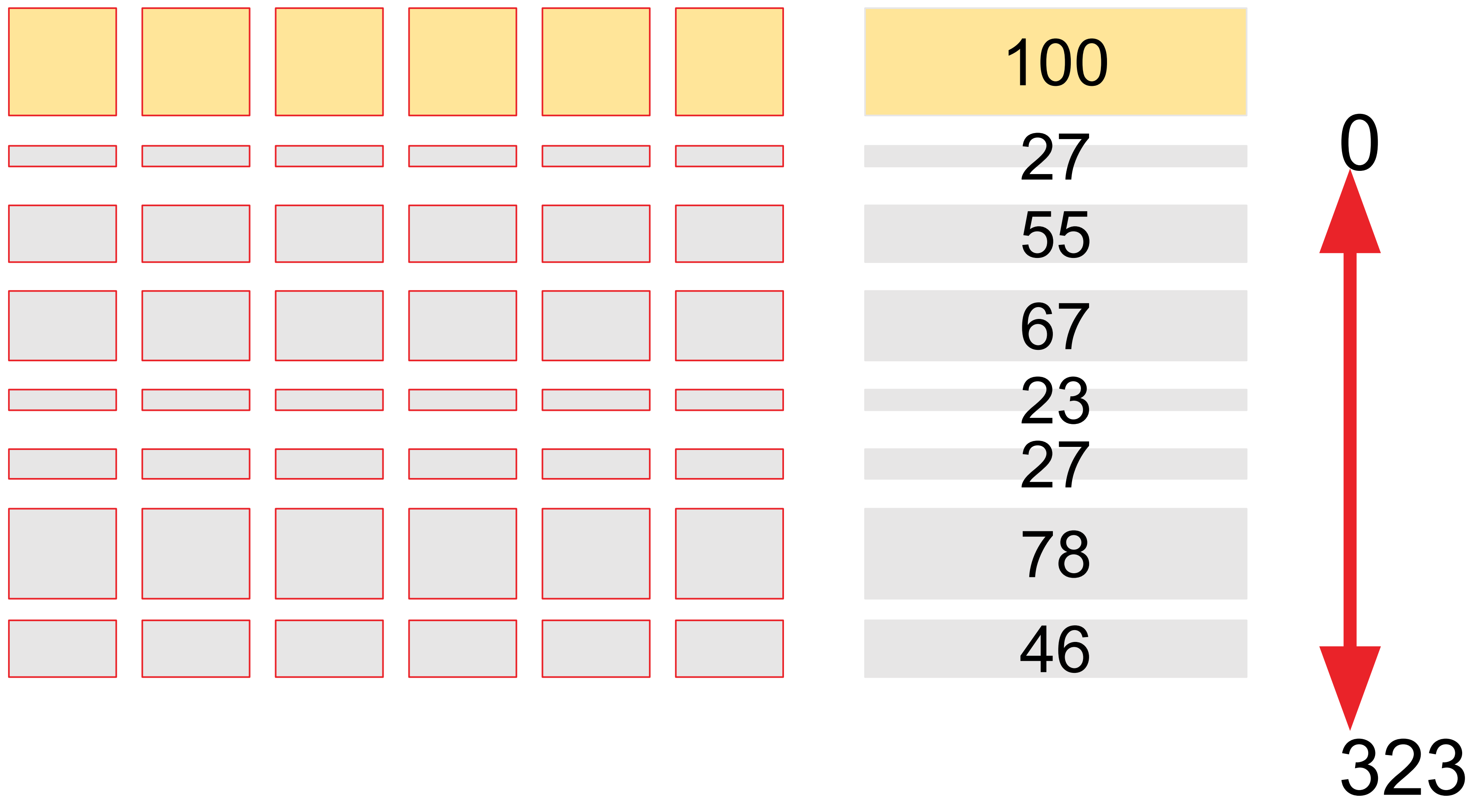


Selection. *techniques*

Tournament selection N (N = 2)

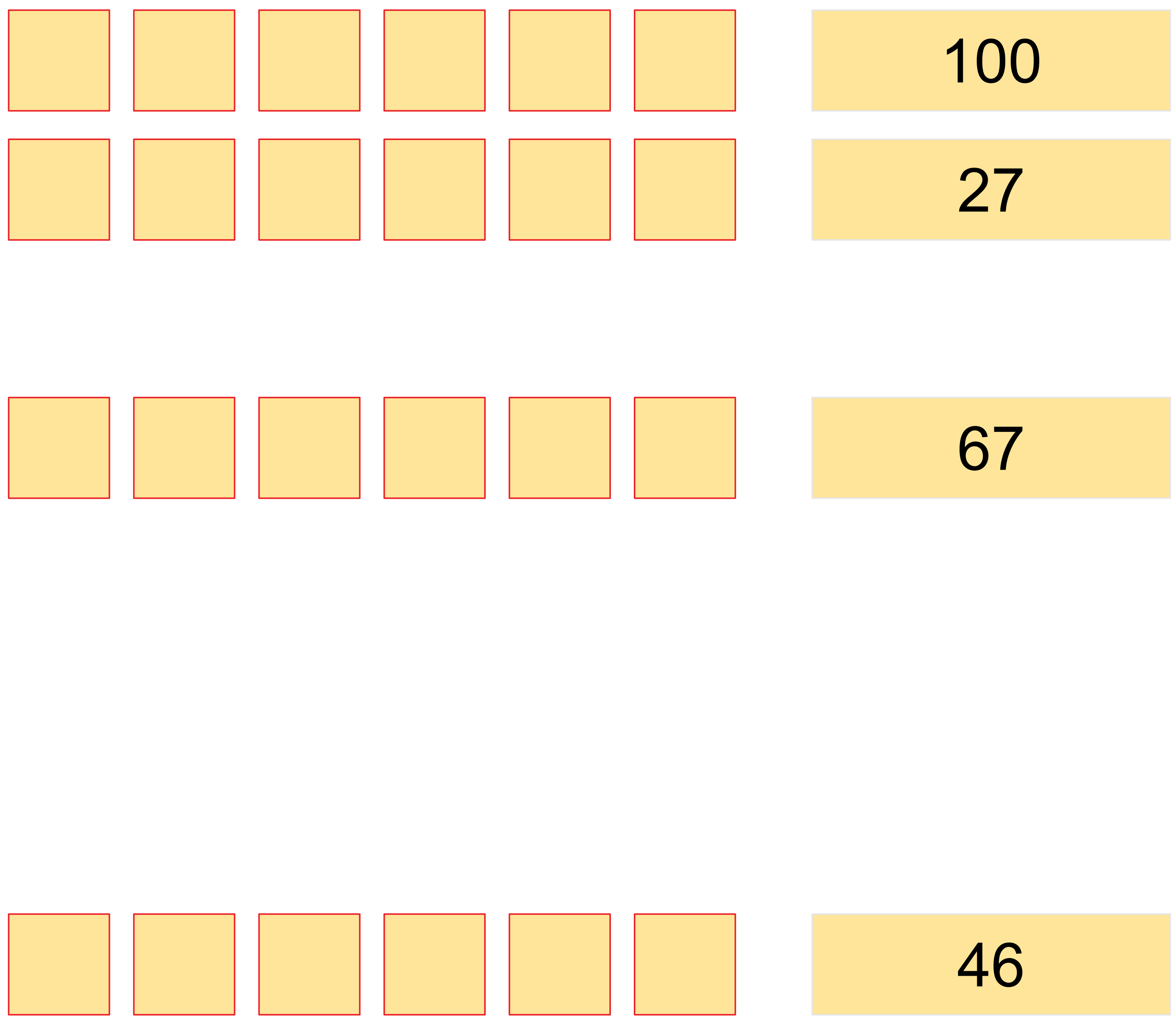


Roulette Selection N (N = 2)

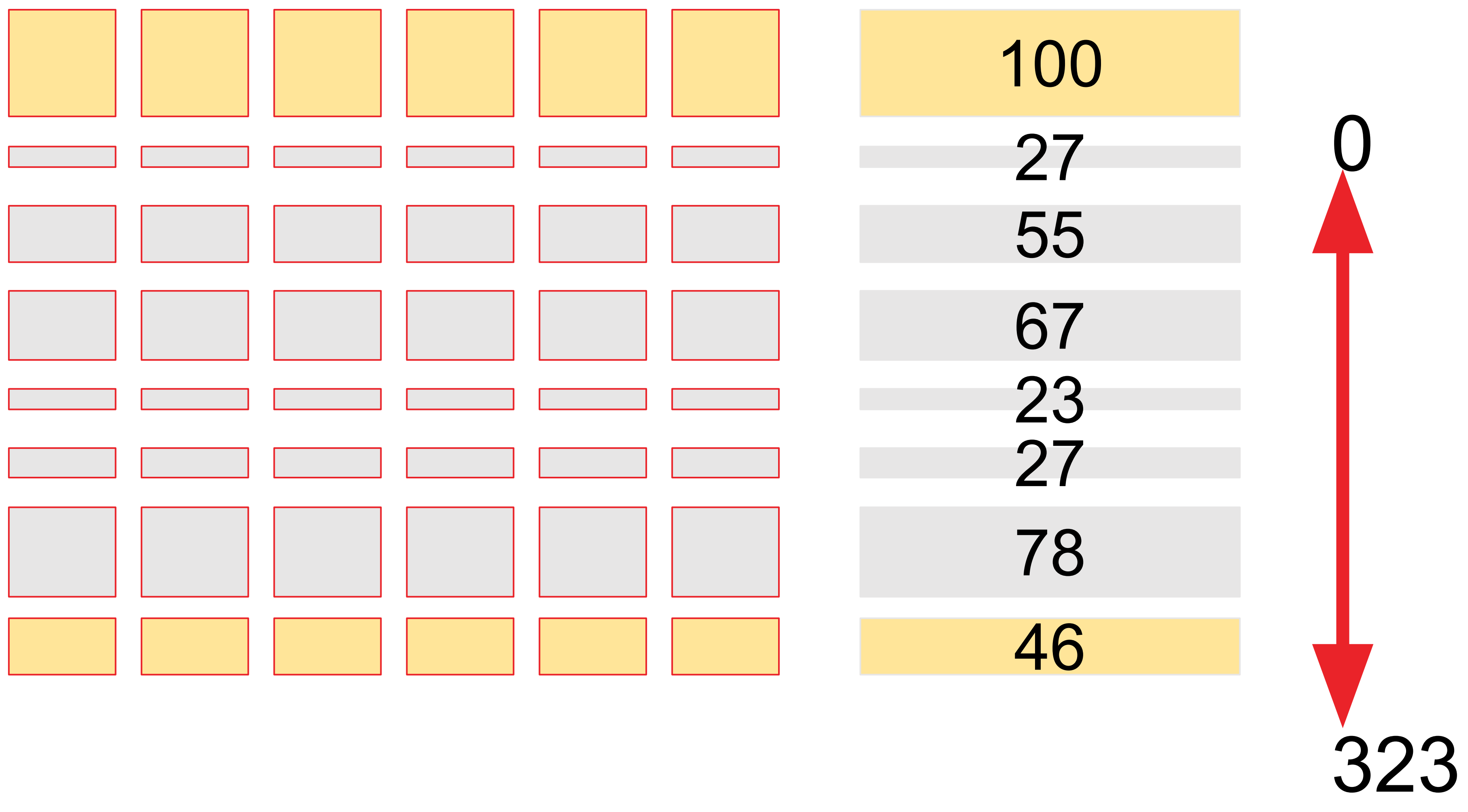


Selection. *techniques*

Tournament selection N (N = 2)

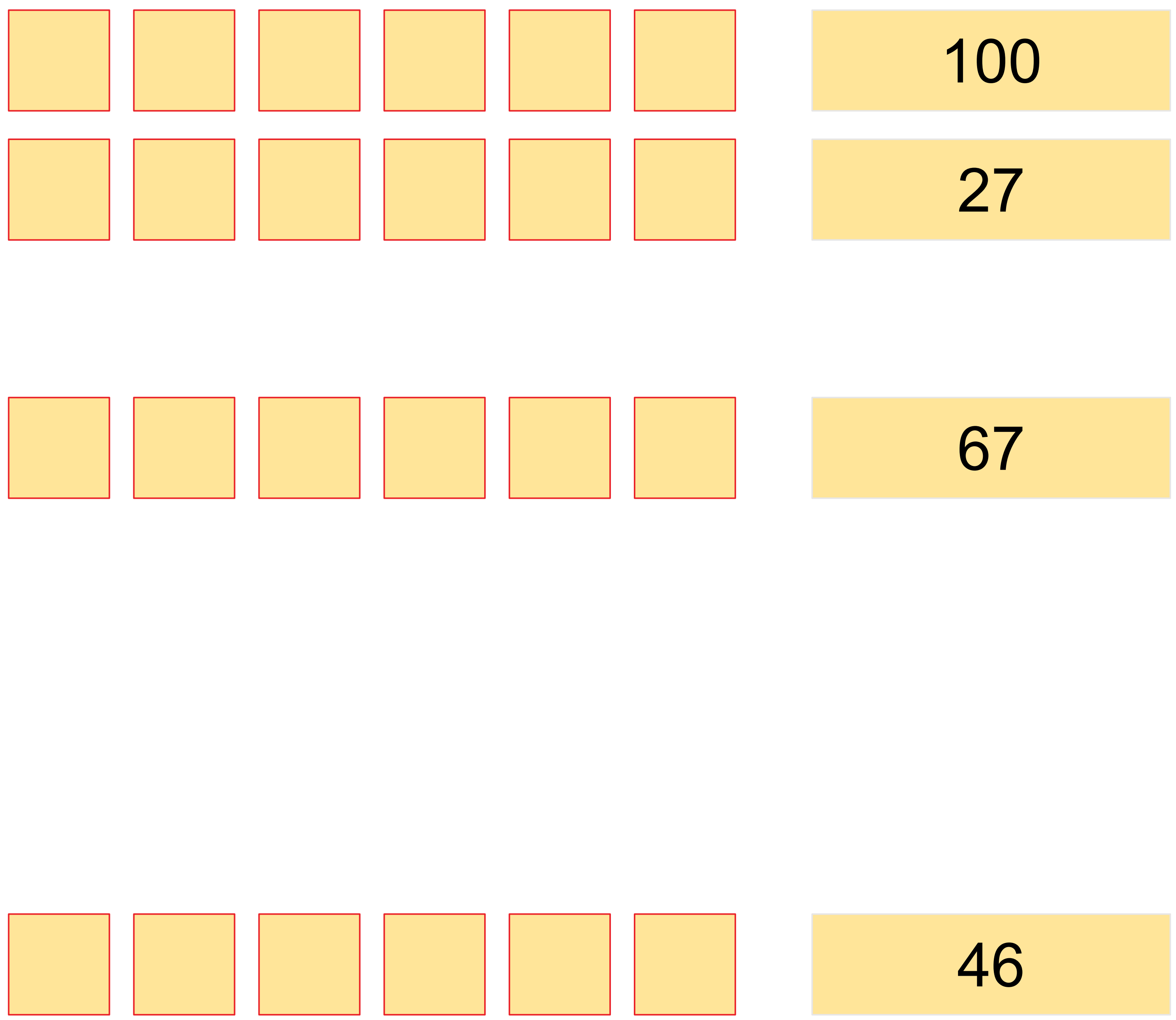


Roulette Selection N (N = 2)

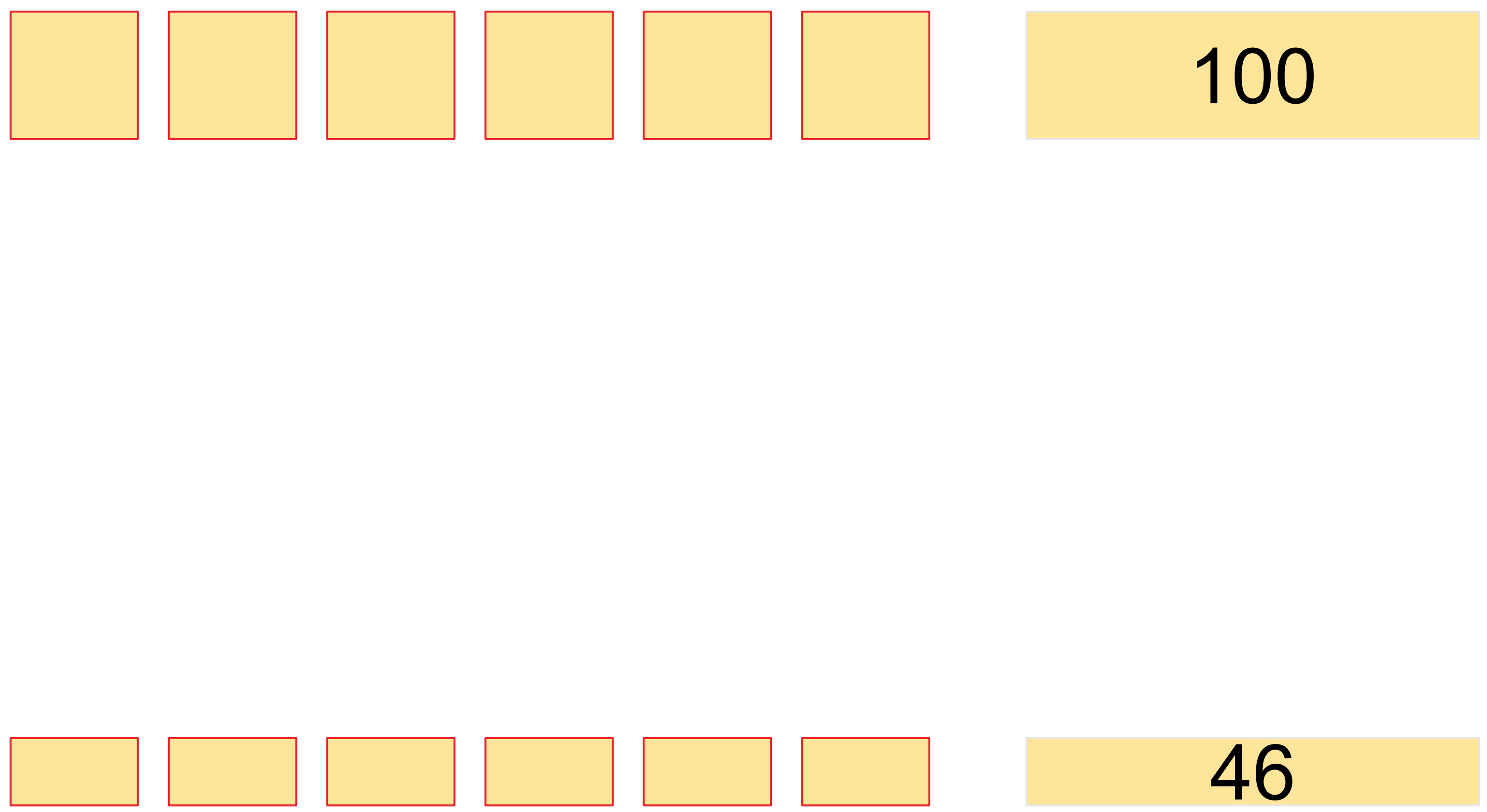


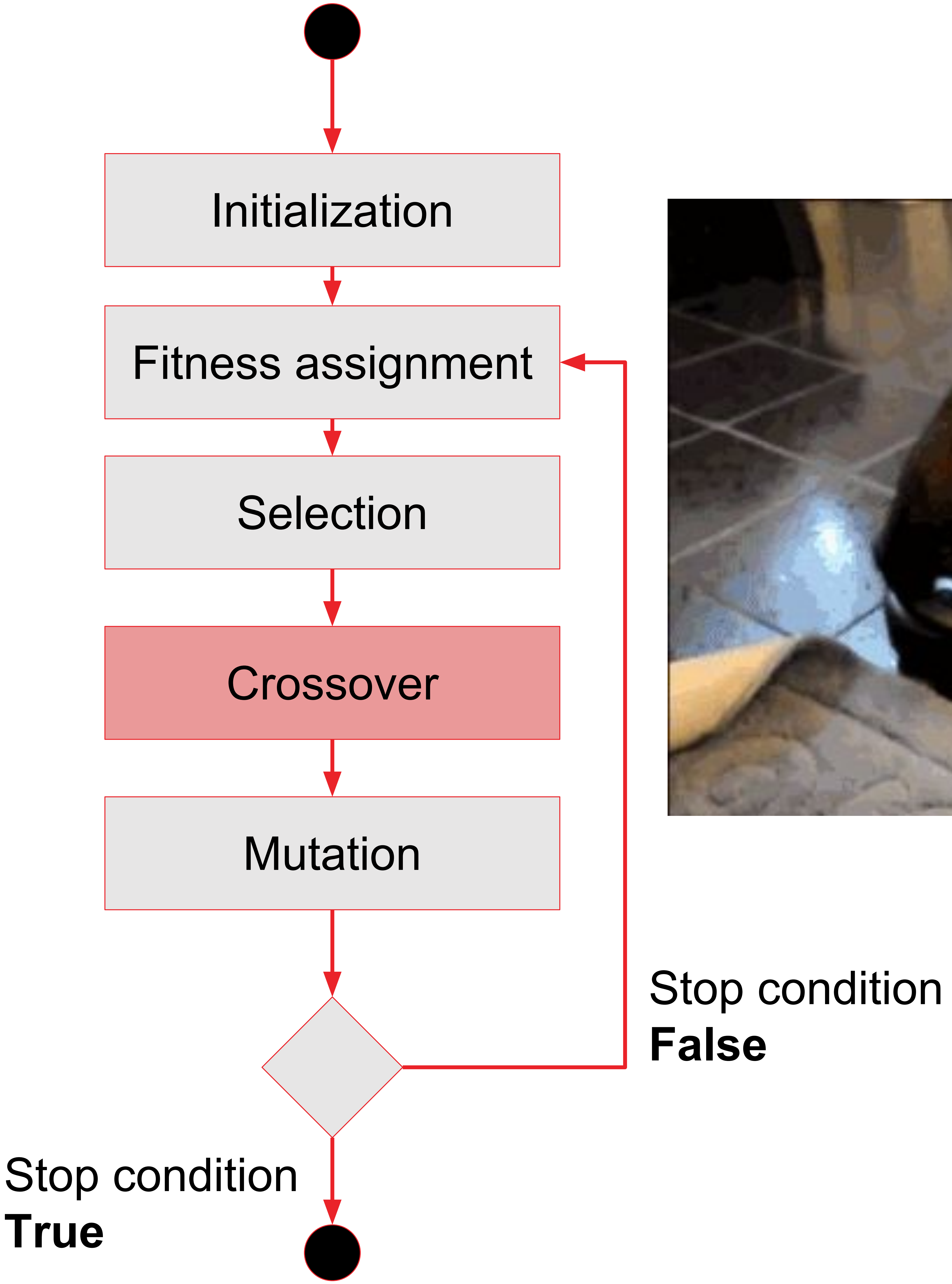
Selection. *techniques*

Tournament selection N (N = 2)



Roulette Selection N (N = 2)




Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True

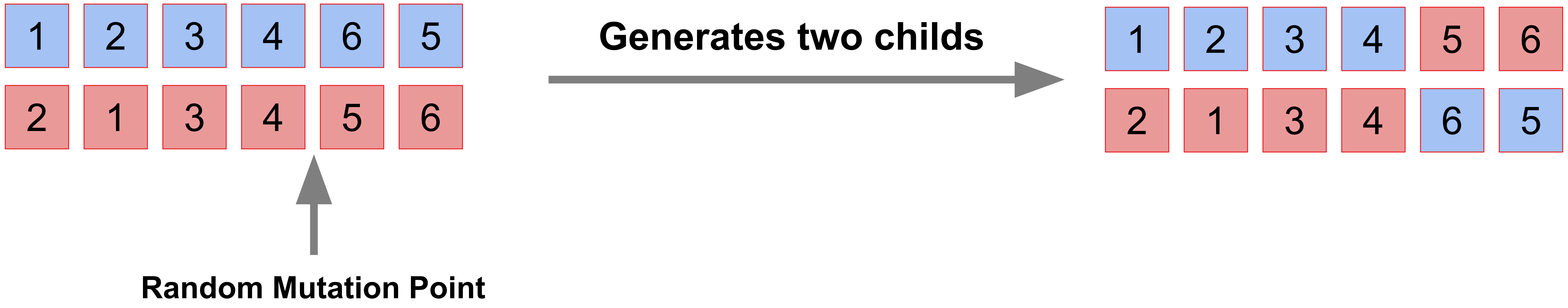
Recombination

They mix genotypes from two parents

- The operators must be designed in a consistent way with the chosen solution representation
- Recombination usually have a high probability to occur, but if they don’t occur, the outputs of the operation are the chosen parents

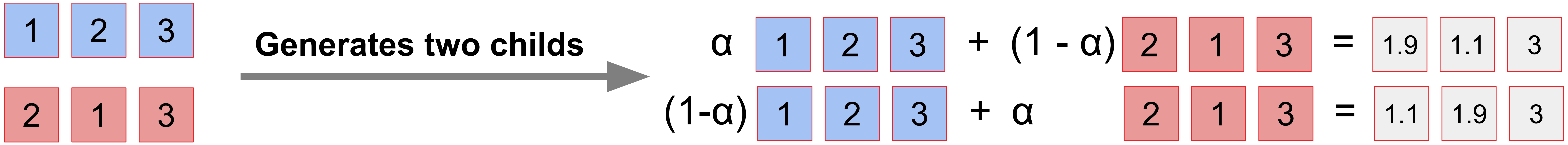
Recombination. *operators*

- One-point random crossover



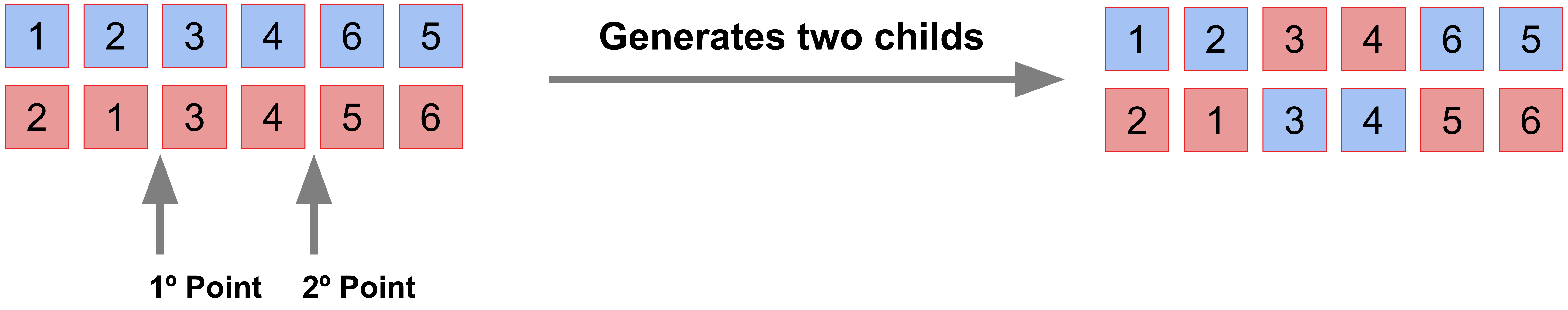
- Interpolation

α(interpolation factor) = 0.1



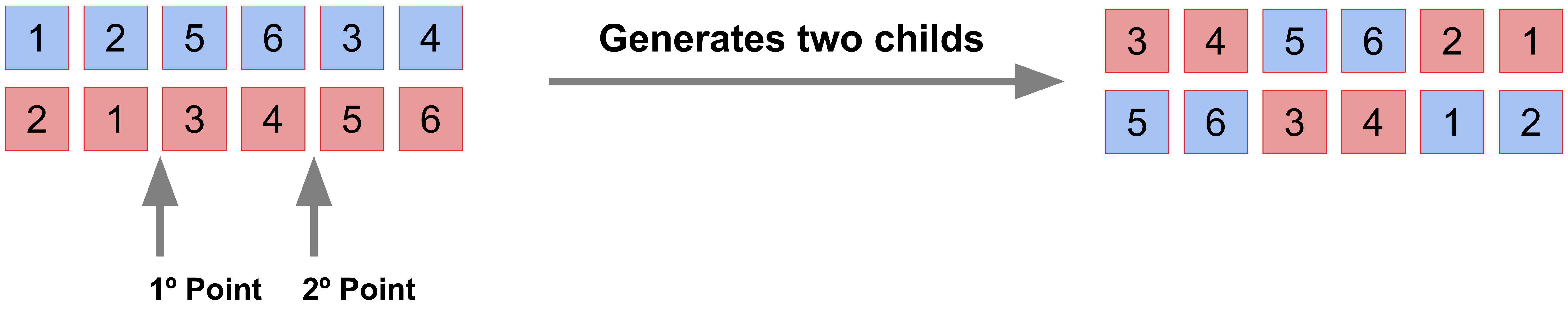
Recombination. *operators*

- Two-point crossover

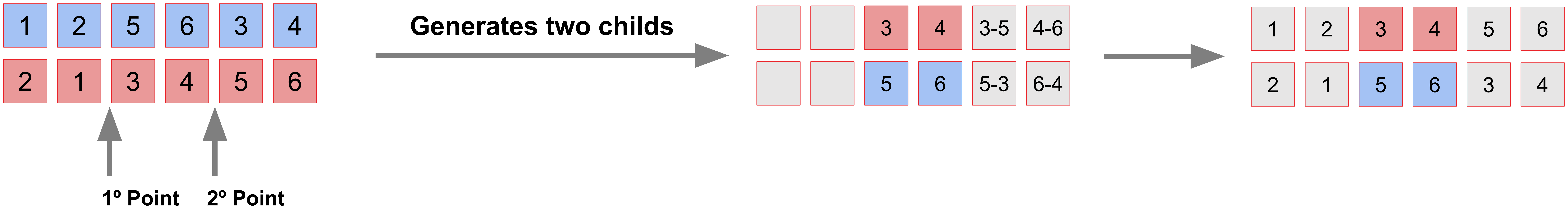


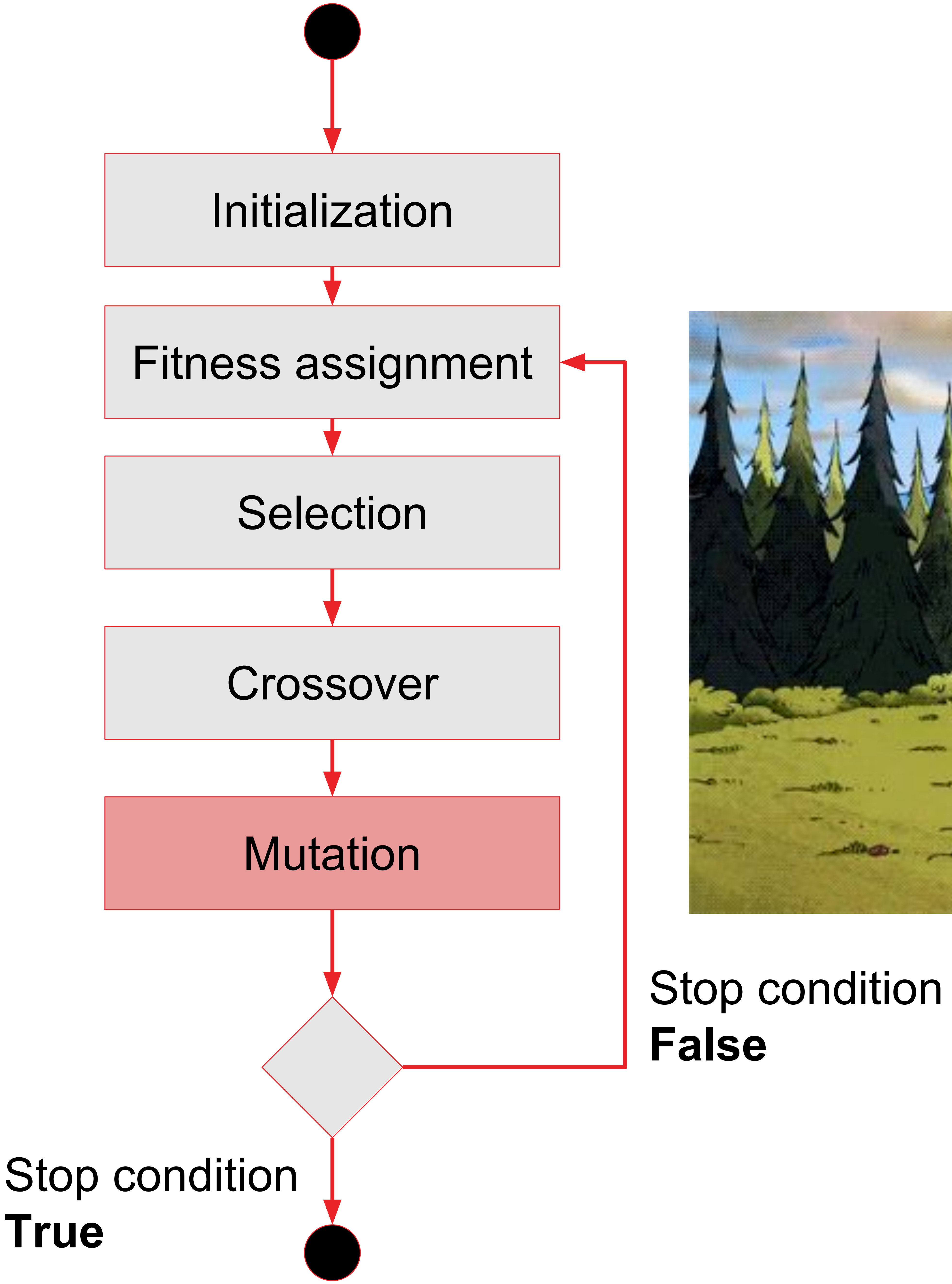
Recombination. *operators for representations with order*

- Order Crossover (OX)



- Partially Mapped Crossover (PMX)




Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True

Mutation

Unitary operator applied to a genotype. It produces an offspring with small changes

- The mutation must allow the solution to reach any point of the search space
- Mutation size should be restricted
- It must generate valid genotypes
- The mutation operation it’s applied over each gen with a really low probability after the the recombination stage

Mutation. *Binary representations*

- **Flip Bit**

Mutation. *Integer representations*

- **Random resetting**

Mutation - Order representations

- **Switch**

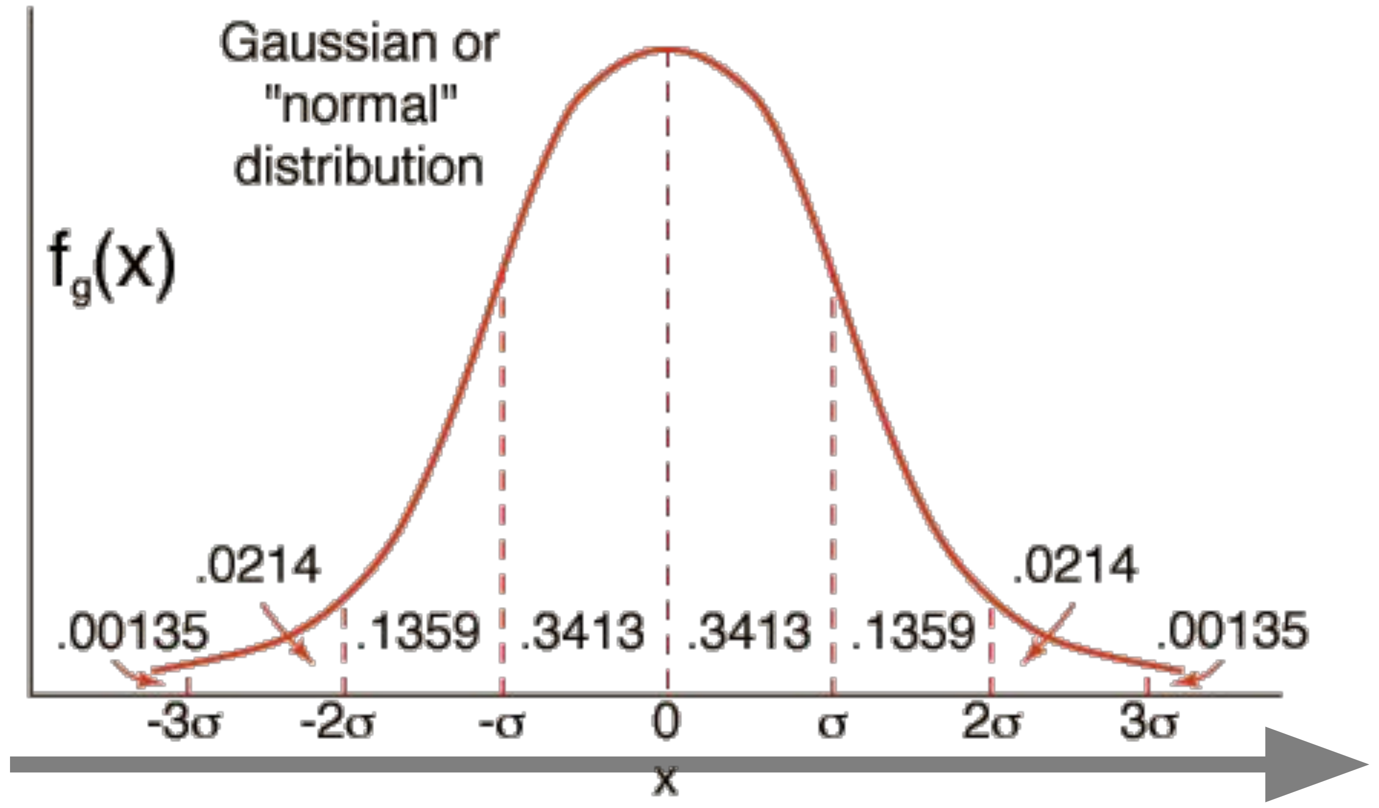
- **Insertion**

- **Mix**

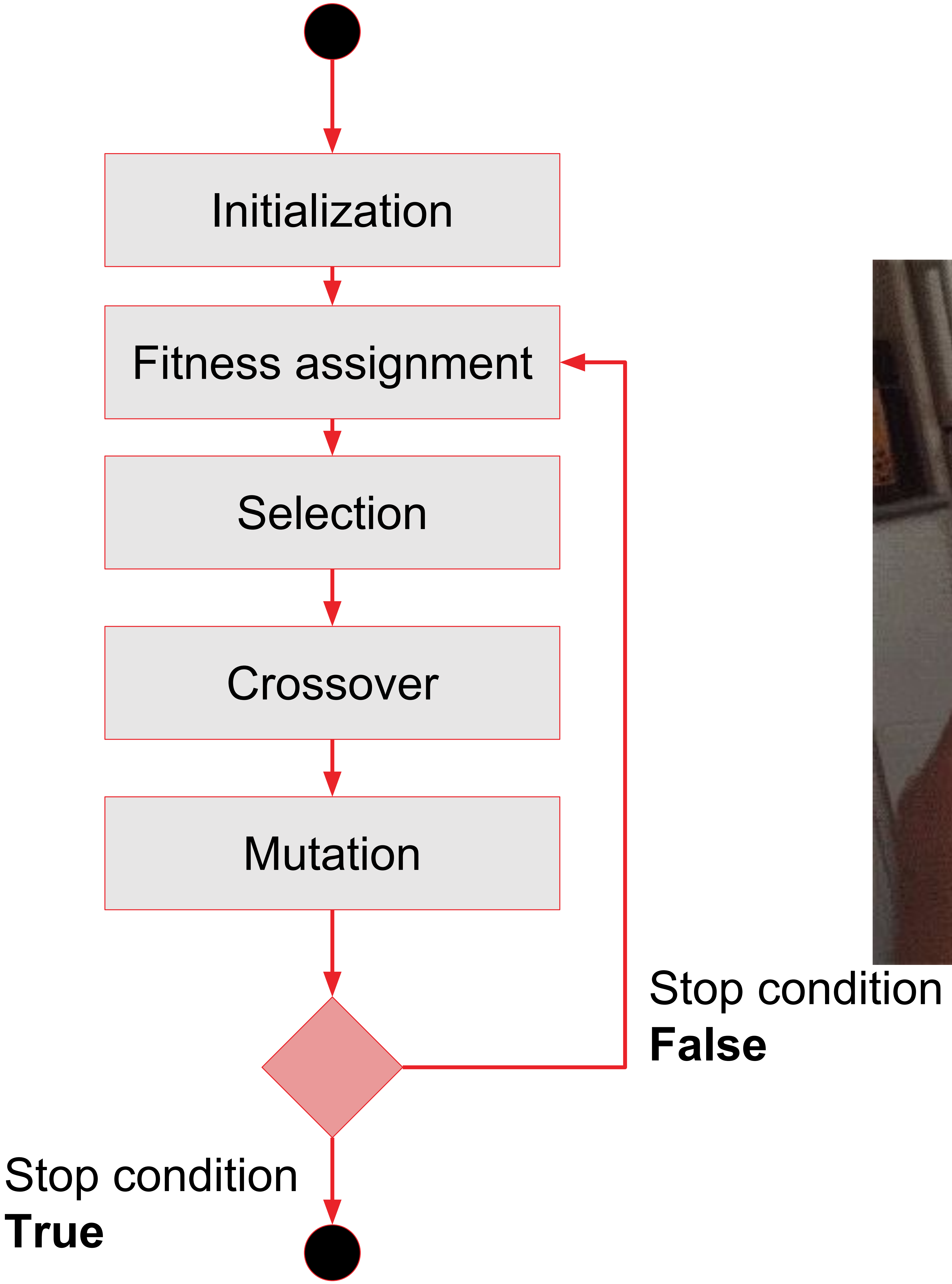
Mutation. *Real representations*

- **Gaussian Noise**
  noise <- N(0,σ)

1.1	0.5	5.7	6.1	1.3	2.9



1.1	0.8	5.7	6.1	1.3	2.9


Initialization
Fitness assignment
Selection
Crossover
Mutation
Stop condition
False
Stop condition
True

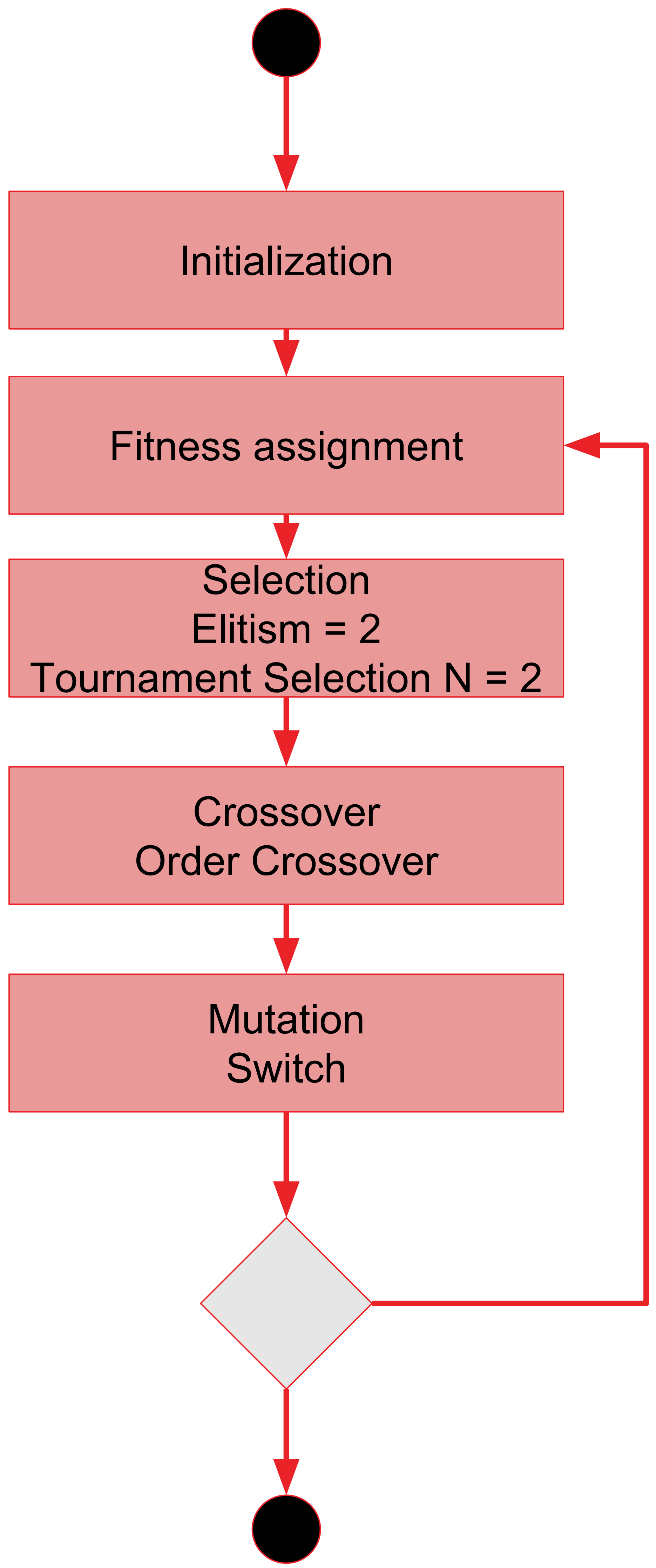

ruinedchildhood

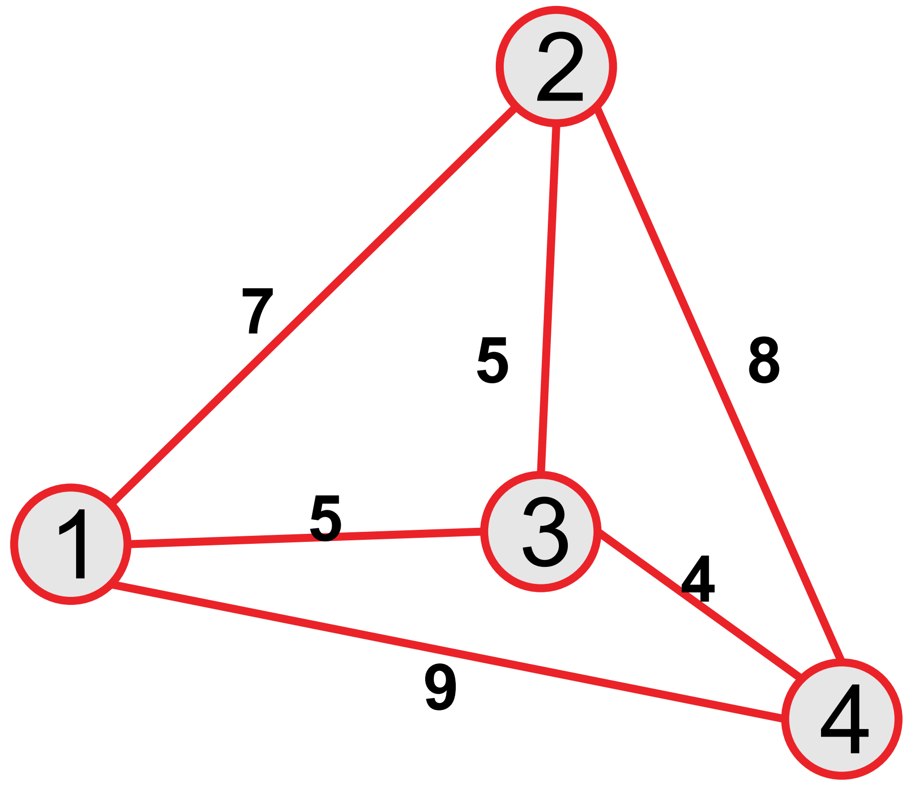
Stop Condition

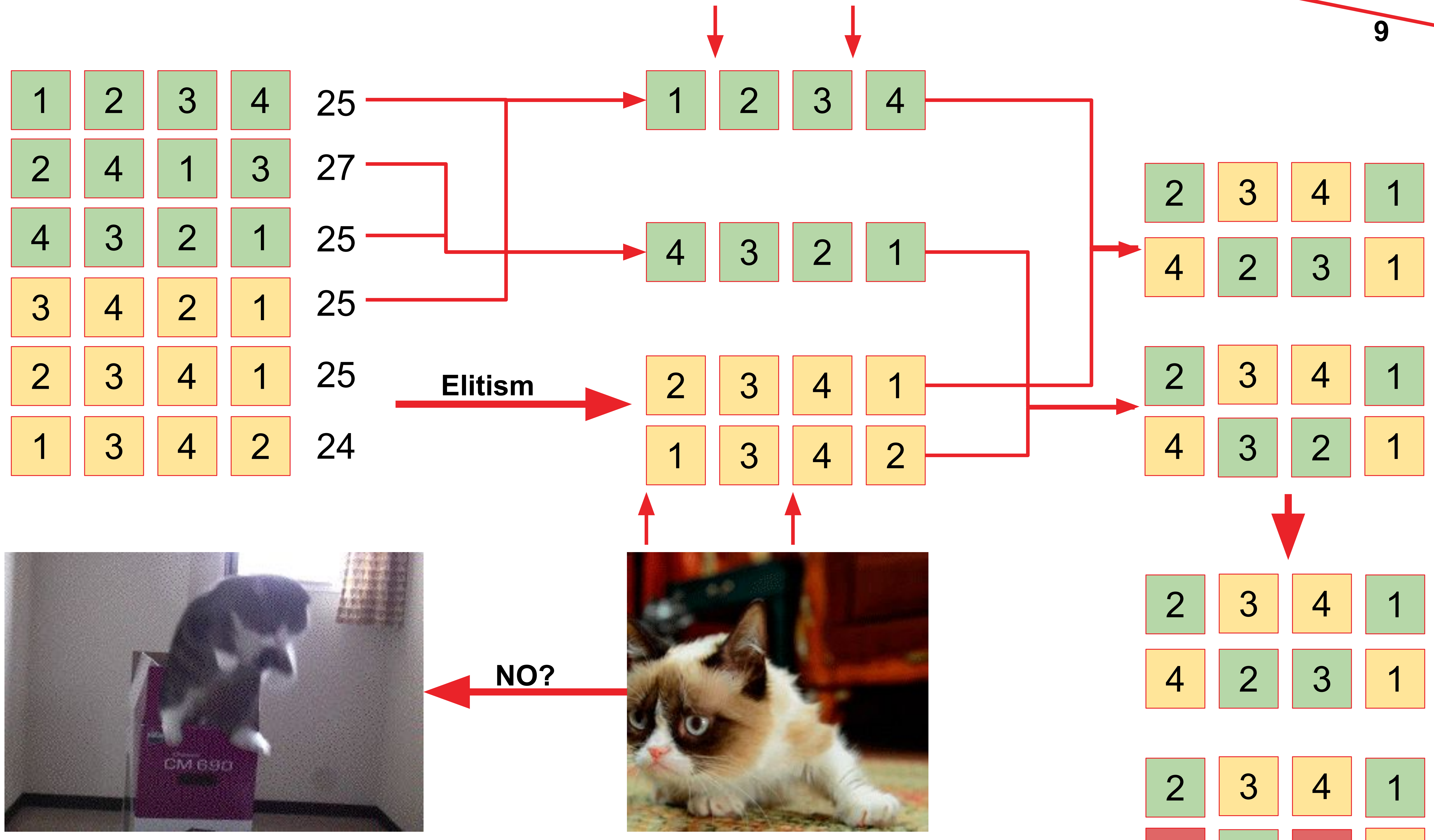
We must provide a condition to stop the evolutionary algorithm

- Number of iterations
- Evolution does not reach a minimum improvement
- Evolution reach a minimum fitness value

Full Example



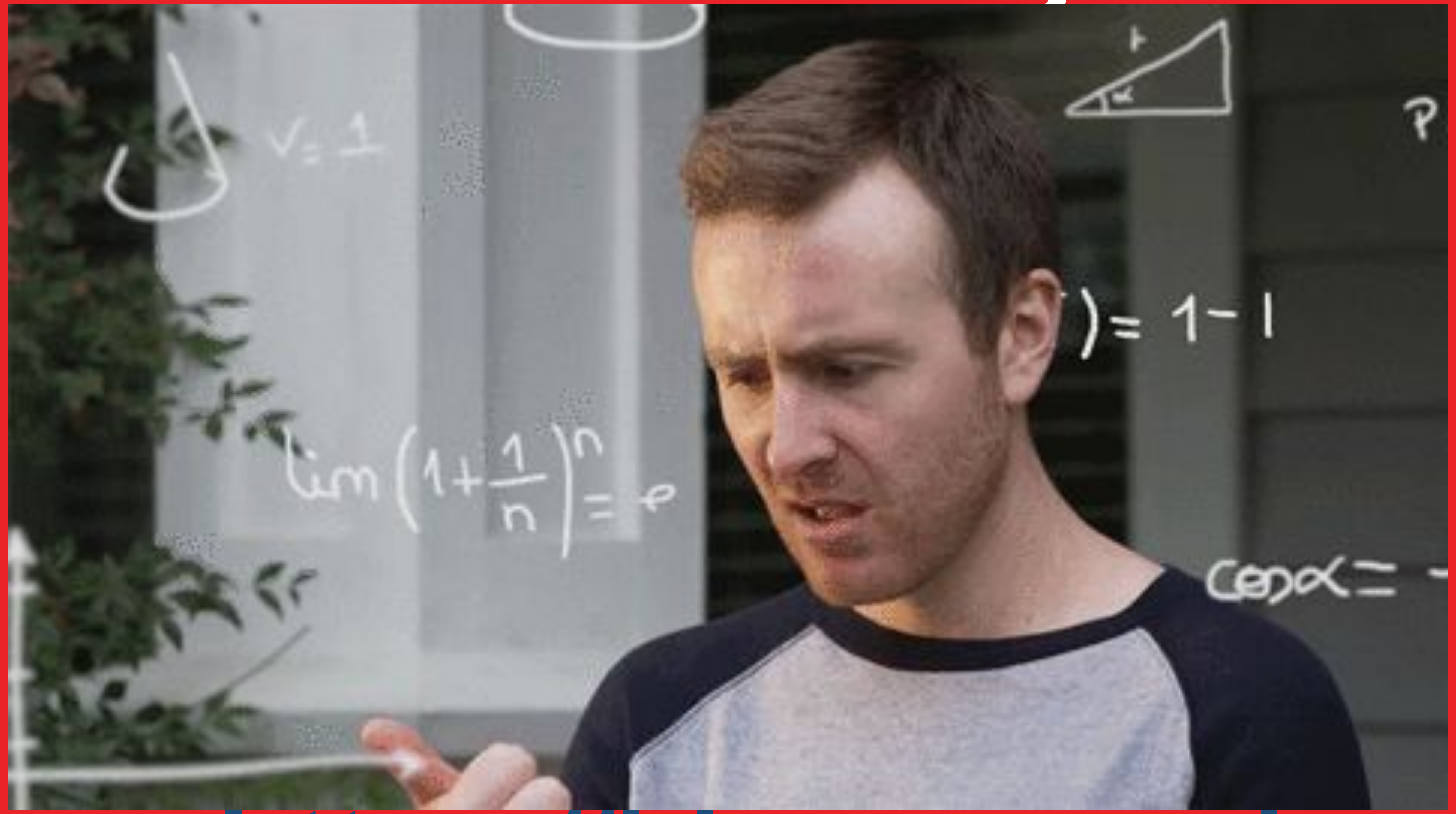




Generate new genotypes to reach original population size and repeat

DONE! SHALL WE STOP?

You may need some motivation



https://blog.openai.com/evolution-strategies/

Let’s check the cars evolutions

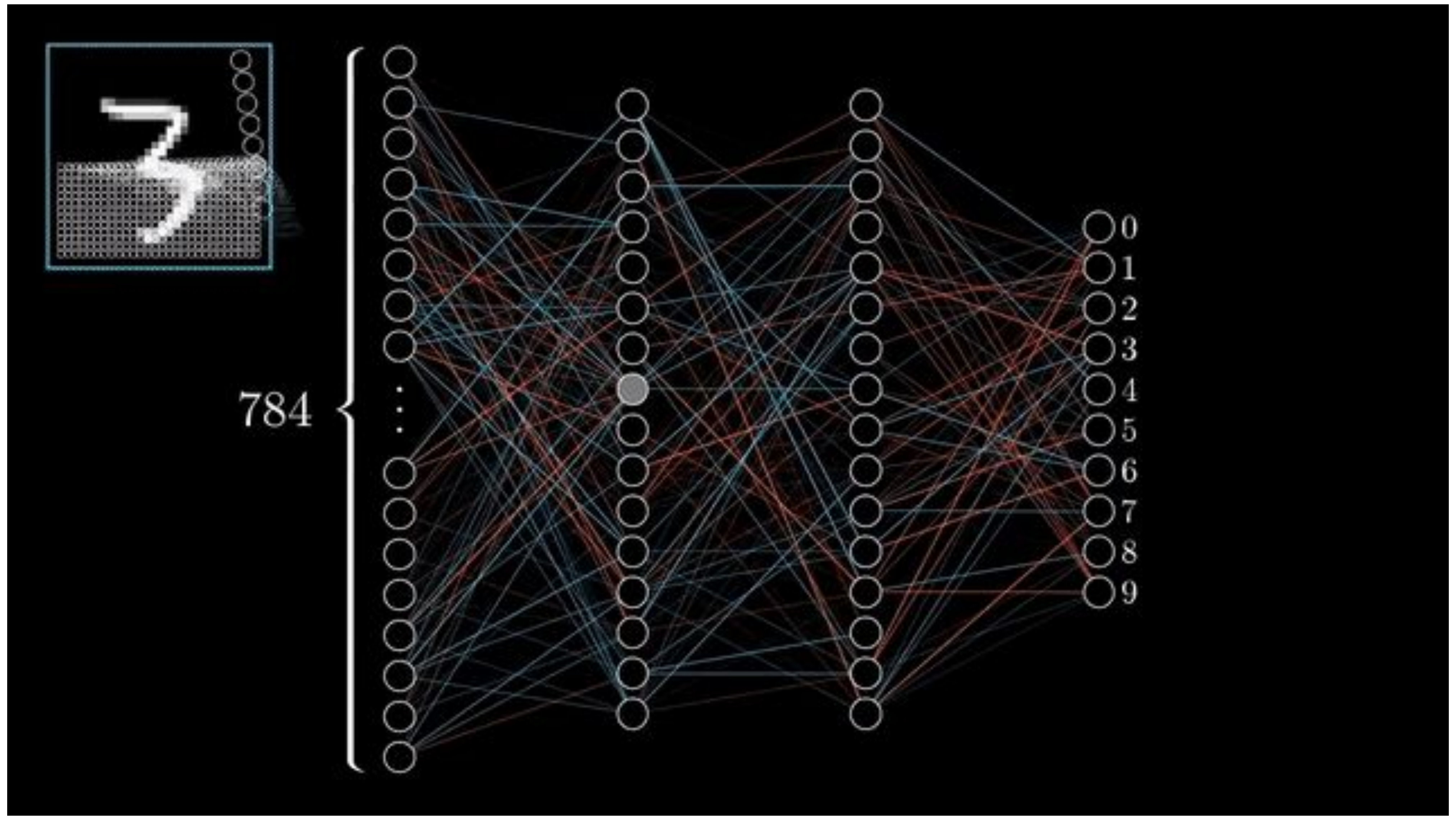
Neuroevolution

Let’s give some context

Raise of Deep Learning

- Many hyperparameters
- Long training times
- Backpropagation
- Many optimization methods

It’s a pain!

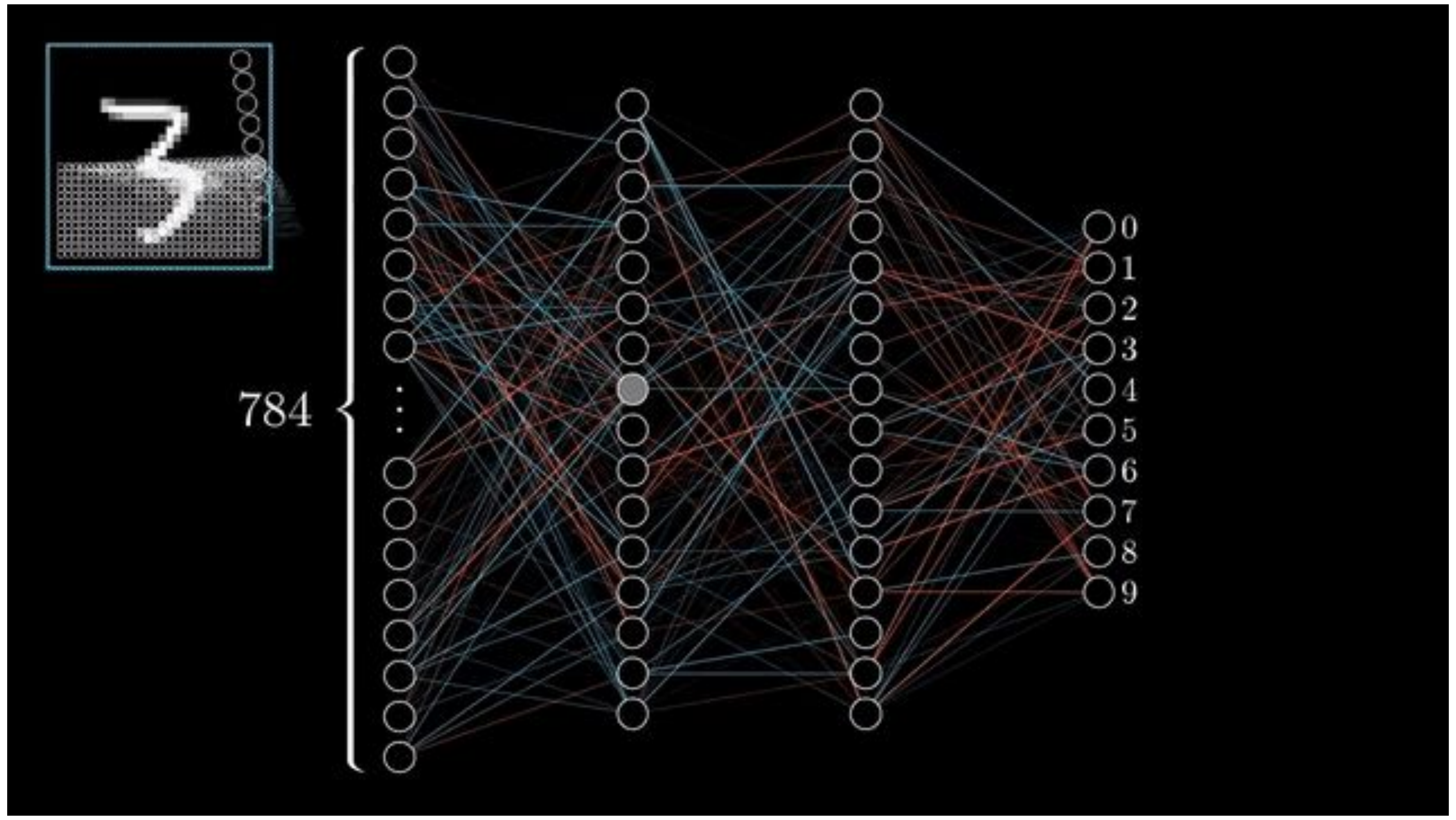


We need an optimization algorithm!

Let's give some context

Raise of Deep Learning

- Many hyperparameters
- Long training times
- Backpropagation
- Many optimization methods
- It's a pain!

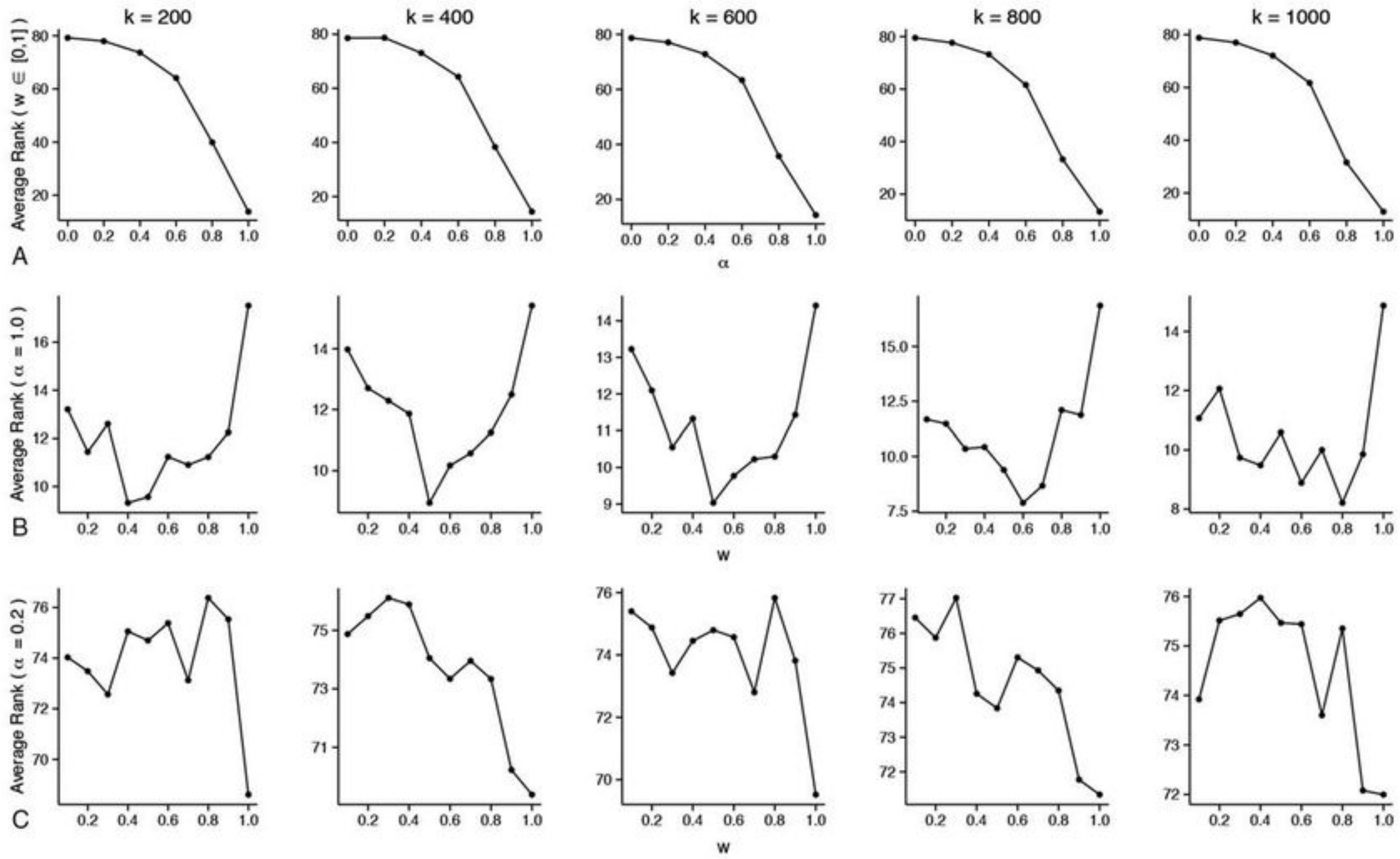


We need an optimization algorithm!

Hyperparameters optimization. *Grid search*

Main keys

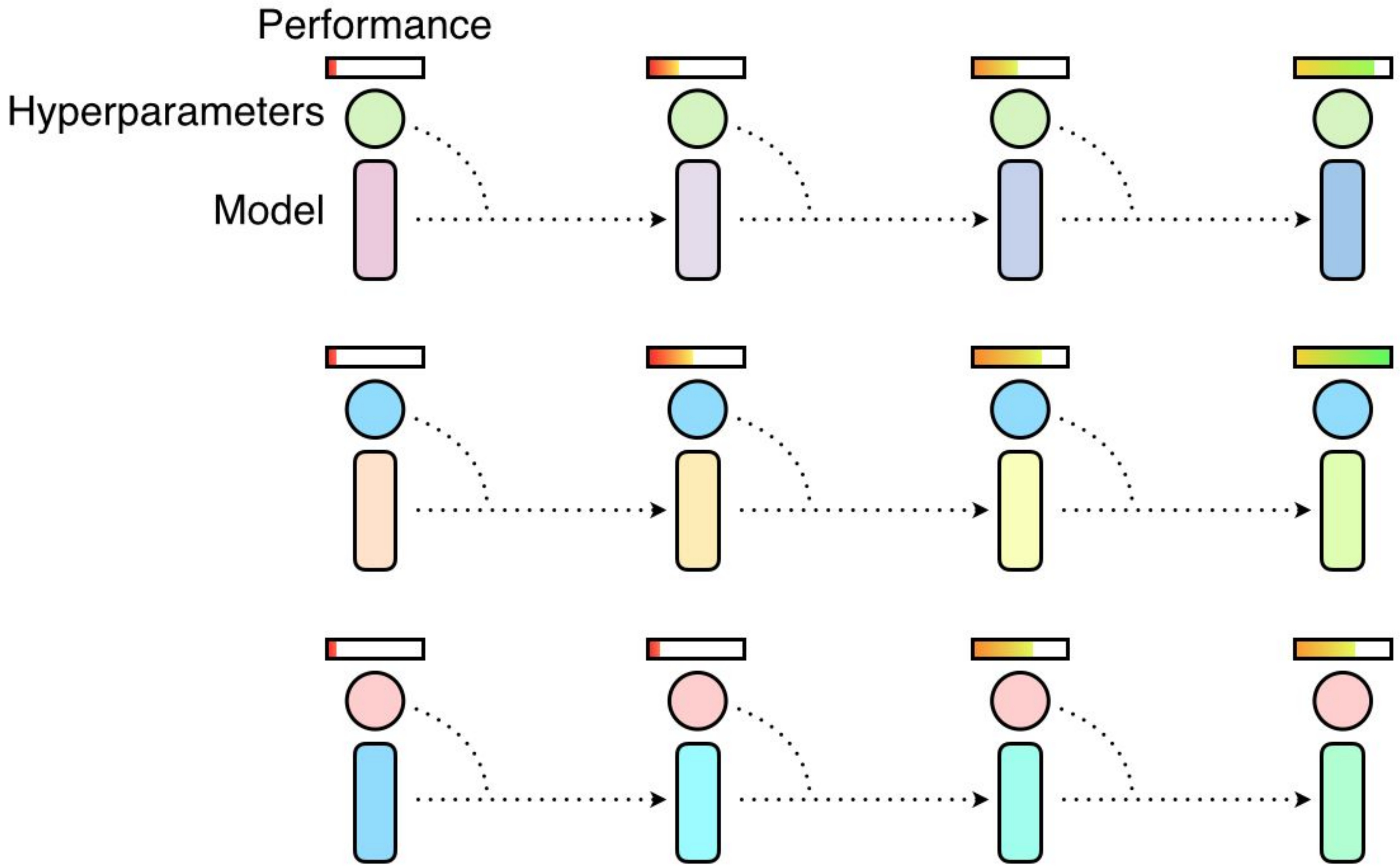
- Exhaustive search through a manually specified subset
- Curse of dimensionality
- Parallelism
- Takes too long



Hyperparameters optimization. *Random search*

Main keys

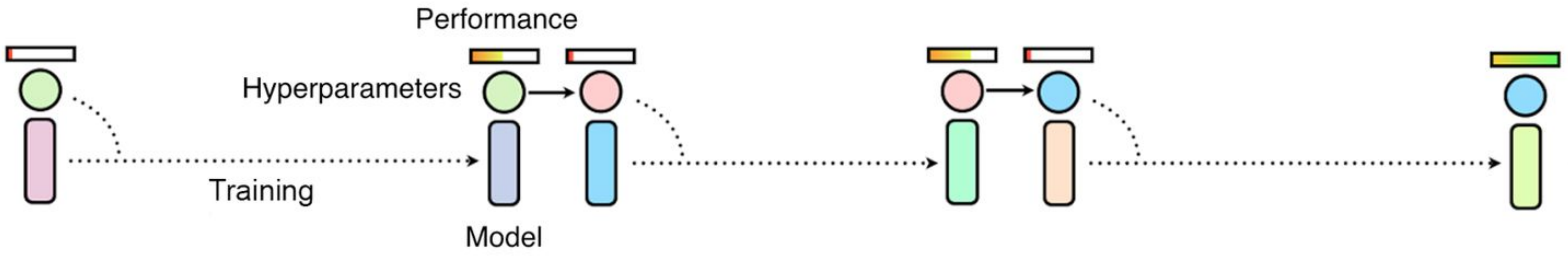
- Random configurations
- Small fraction trained with good hyperparameters
- Low performance
- Waste of resources



Hyperparameters optimization. *Hand tuning*

Main keys

- “Guessed” by the researcher (or engineer, we don’t know it yet)
- By repetition and experience
- Better performance
- This may take too long
- Bayesian optimization (slow)



What is the idea?

"Generate artificial neural networks (their connections weights and/or topology) by using evolutionary algorithms"

S. Risi & J. Togelius

Trending

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans Jonathan Ho Xi Chen Szymon Sidor Ilya Sutskever
OpenAI

Abstract

We explore the use of Evolution Strategies (ES), a class of black box optimization algorithms, as an alternative to popular MDP-based RL techniques such as Q-learning and Policy Gradients. Experiments on MuJoCo and Atari show that ES is a viable solution strategy that scales extremely well with the number of CPUs

Uber Data

Welcoming the Era of Deep Neuroevolution

December 18, 2017

168 | 3K | in | 8 | 461 | G+

By Kenneth O. Stanley, Jeff Clune

On behalf of an Uber AI Labs team that also includes Joel Lehman, Jay Chen, Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, & Xingwen Zhang.

In the field of deep learning, deep neural networks (DNNs) with many layers and millions of connections are now trained routinely through stochastic gradient descent (SGD). Many assume that the ability of SGD to efficiently compute gradients is essential to this capability. However, we are releasing a suite of five papers that support the emerging realization that neuroevolution, where neural networks are optimized through evolutionary algorithms, is also an effective method to train deep neural networks for reinforcement learning (RL) problems. Uber has a multitude of areas where machine learning can improve its operations, and developing a broad range of powerful learning approaches that includes neuroevolution will help us achieve our mission of developing safer and more reliable transportation solutions.

Genetic algorithms as a competitive alternative for training deep neural networks

Intelligent Machines

Evolutionary algorithm outperforms deep-learning machines at video games

Neural networks have garnered all the headlines, but a much more powerful approach is waiting in the wings.

by Emerging Technology from the arXiv July 18, 2018

28 Nov 2017

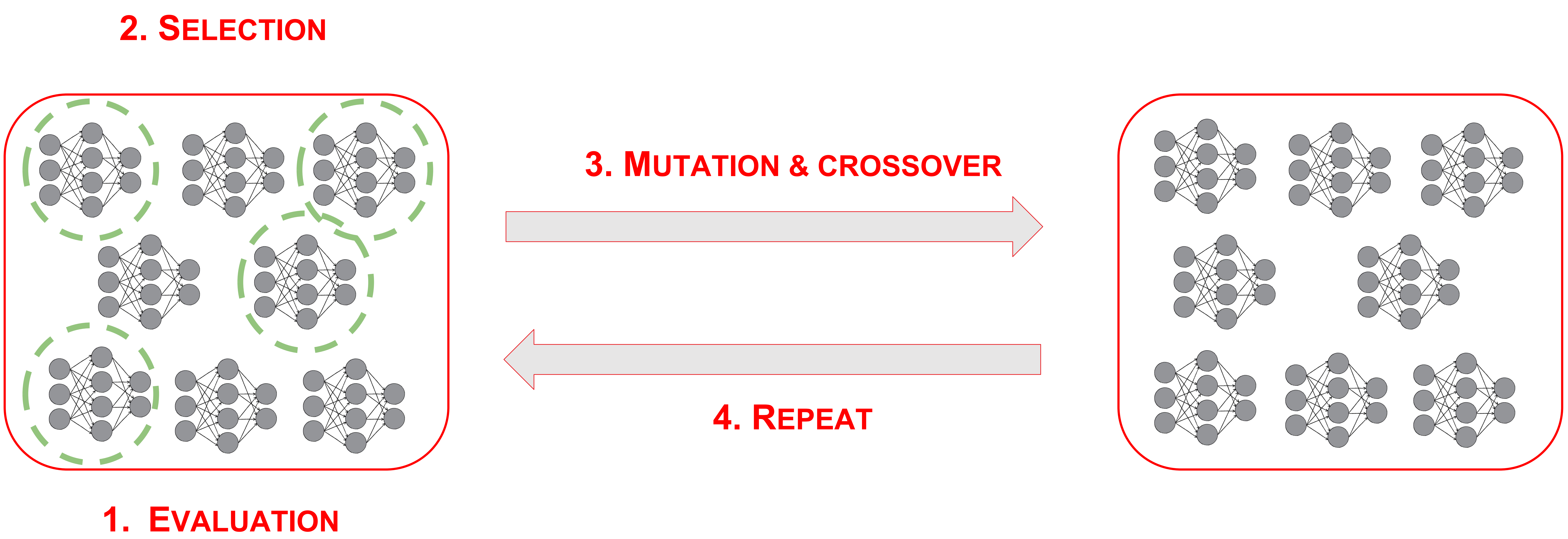
Population Based Training of Neural Networks

Max Jaderberg Valentin Dalibard Simon Osindero Wojciech M. Czarnecki
Jeff Donahue Ali Razavi Oriol Vinyals Tim Green Iain Dunning
Karen Simonyan Chrisantha Fernando Koray Kavukcuoglu
DeepMind, London, UK

Abstract

Neural networks dominate the modern machine learning landscape, but their training and success still suffer from sensitivity to empirical choices of hyperparameters such as model architecture, loss function, and optimisation algorithm. In this work we present *Population Based Training (PBT)*, a simple asynchronous optimisation algorithm which effectively utilises a fixed computational budget to jointly optimise a population of models and their hyperparameters to maximise performance. Importantly, PBT discovers a schedule of hyperparameter settings rather than following the generally sub-optimal strategy of trying to

What is the idea?



Exploration vs Exploitation

Direct representation. *Fixed-topology*

Early algorithms

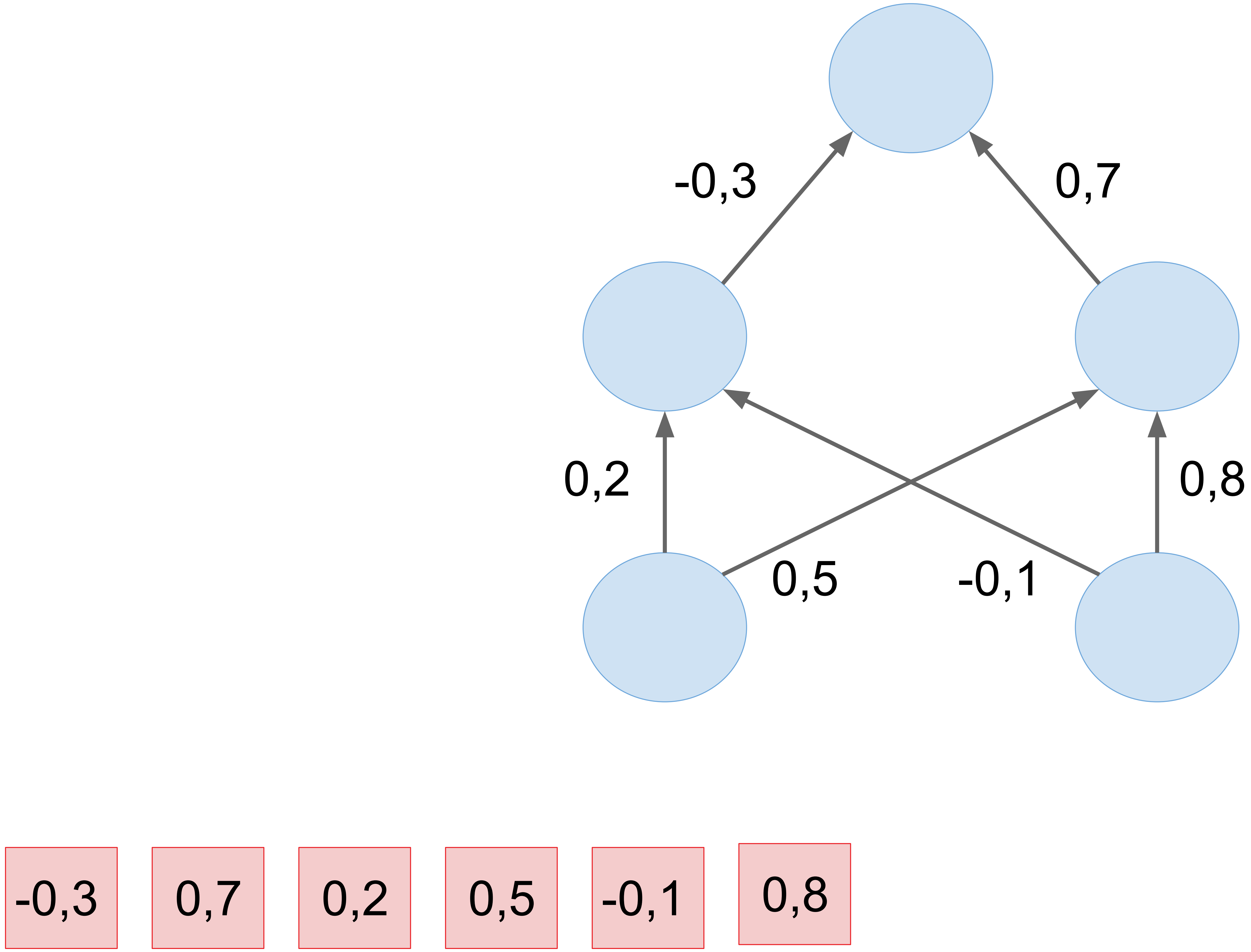
- Basic approach at the ‘80s and ‘90s

Properties

- One gene per hyperparameter
- Fixed topology
- String of characters or real values

Disadvantages

- ANNs never become larger
- Cannot evolve complexity
- Competing conventions



Direct representation. *Cooperative coevolution*

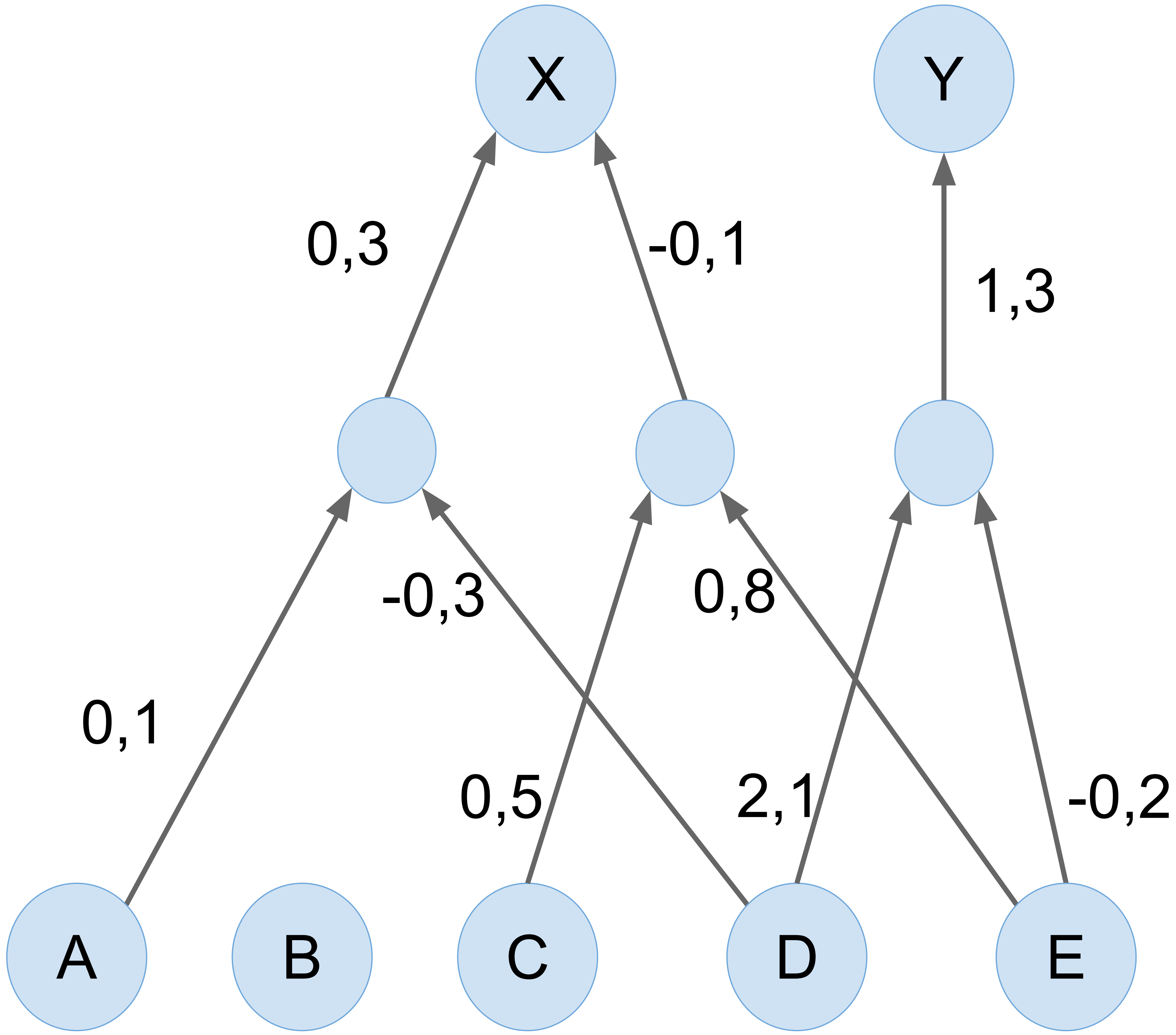
SANE (Symbiotic Adaptive Neuro Evolution)

- Population of neurons with random selection
- Conexion and weights
- Fixed topology: one hidden layer
- Neuron fitness ⇒ average in network

ESP (Enforced SubPopulations)

- Evolves recurrent connections
- Uses information about past experience ⇒ decisions

A	0,1	D	-0,3	X	0,3
C	0,5	E	0,8	X	-0,1
D	2,1	E	-0,2	X	1,3



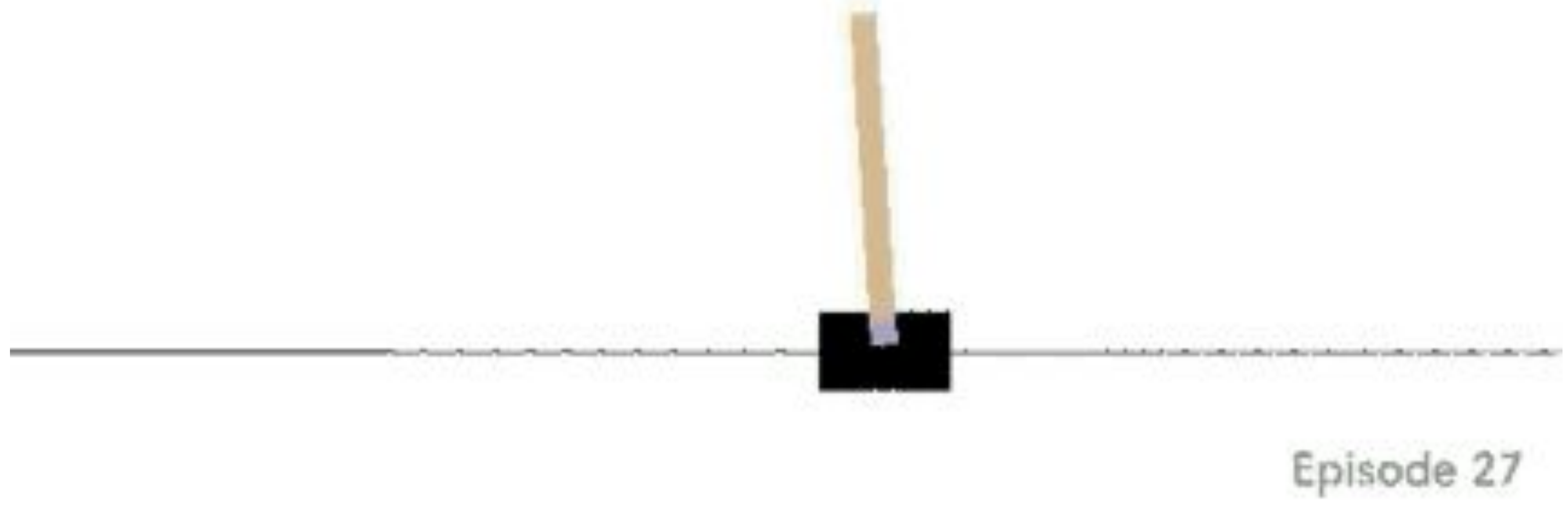
Direct representation. *TWEANN*

Need to evolve complexity

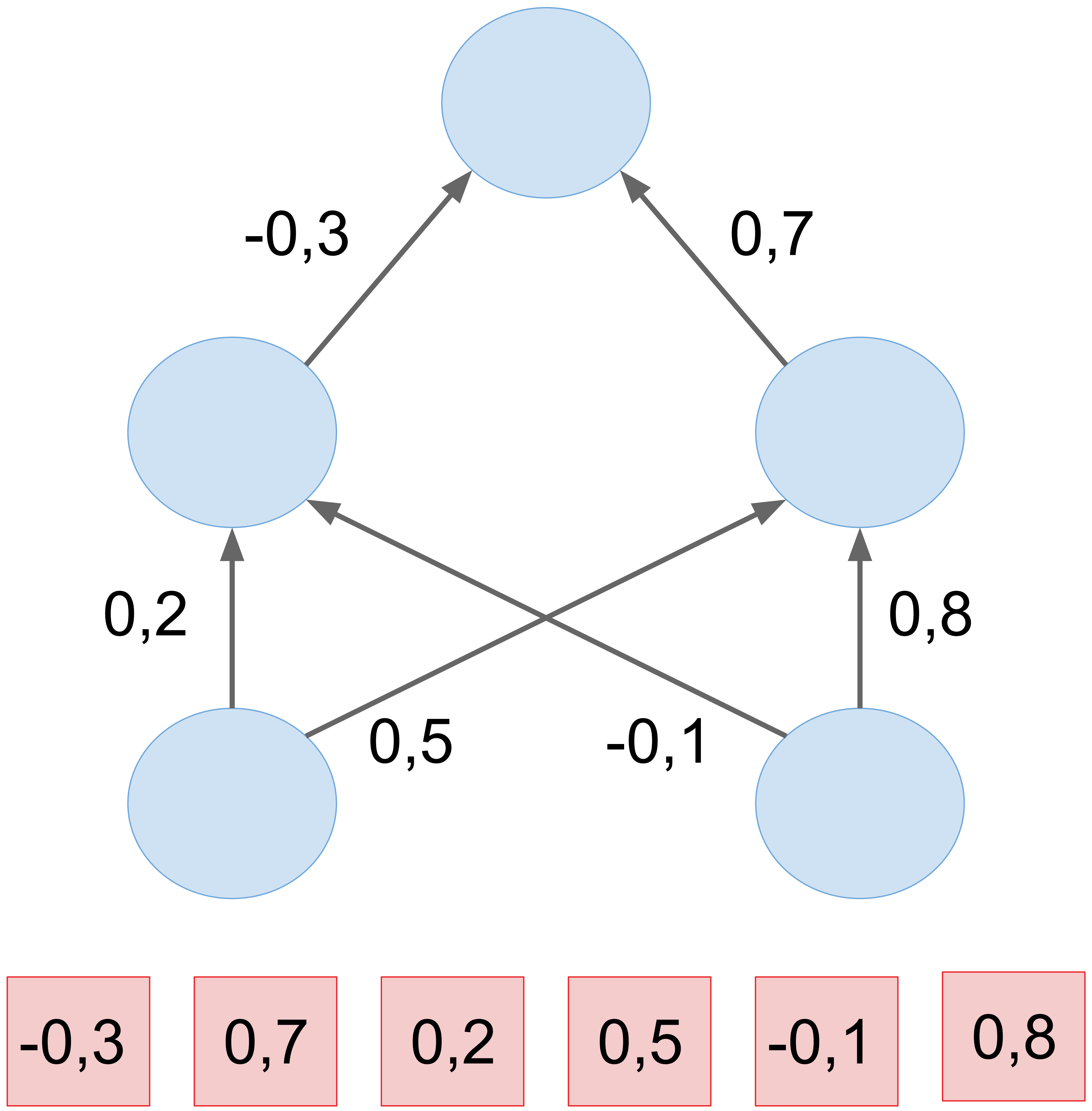
- Topology and Weight Evolving Artificial Neural Networks
- Many approaches at the late ‘90s

Properties

- Topology of a parent ANN may be changed
    - By adding a new connection
    - By adding a new neuron
- Simple problems
    - Pole balancing (benchmark)



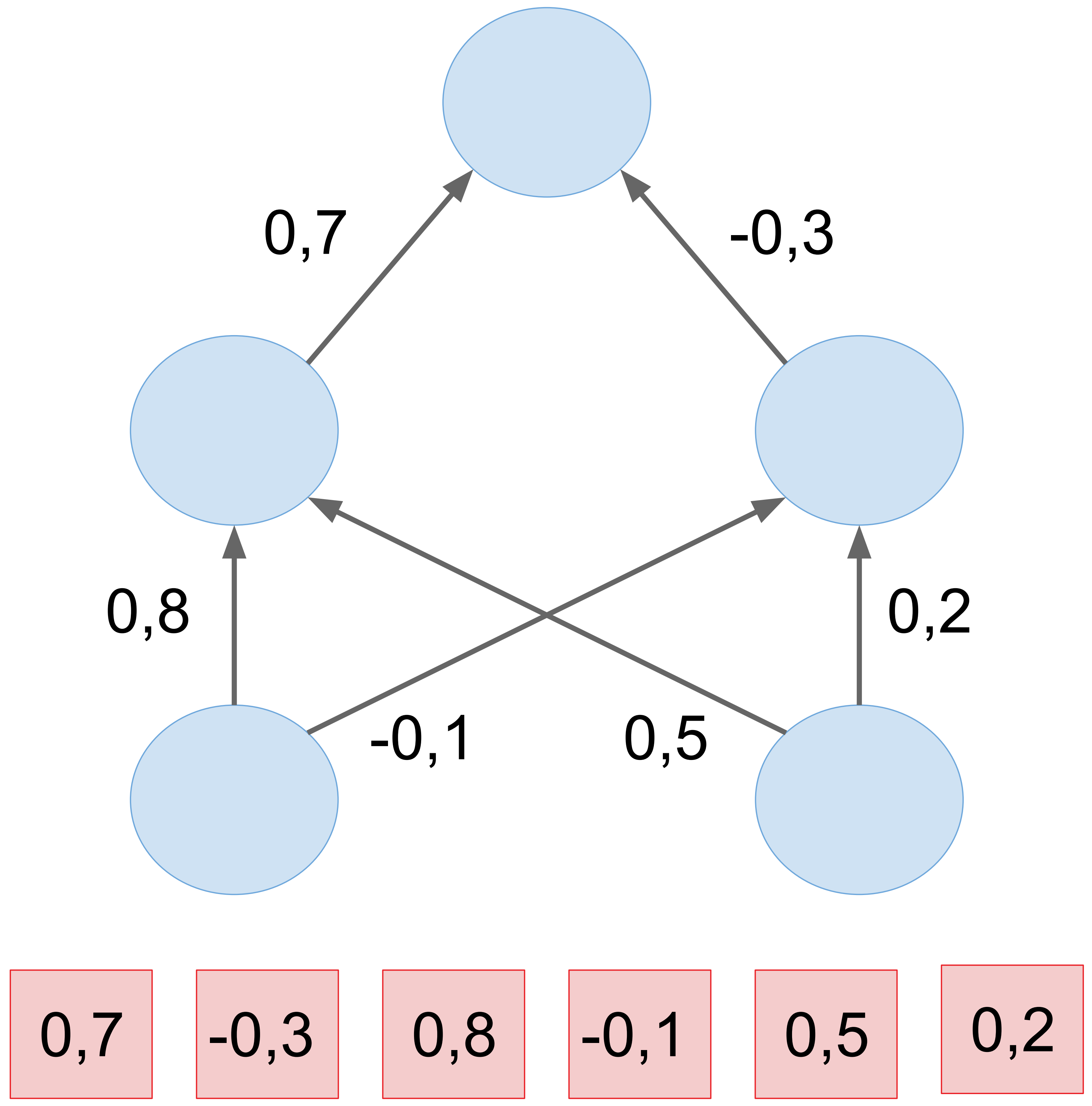
Direct representation. *Competing conventions*



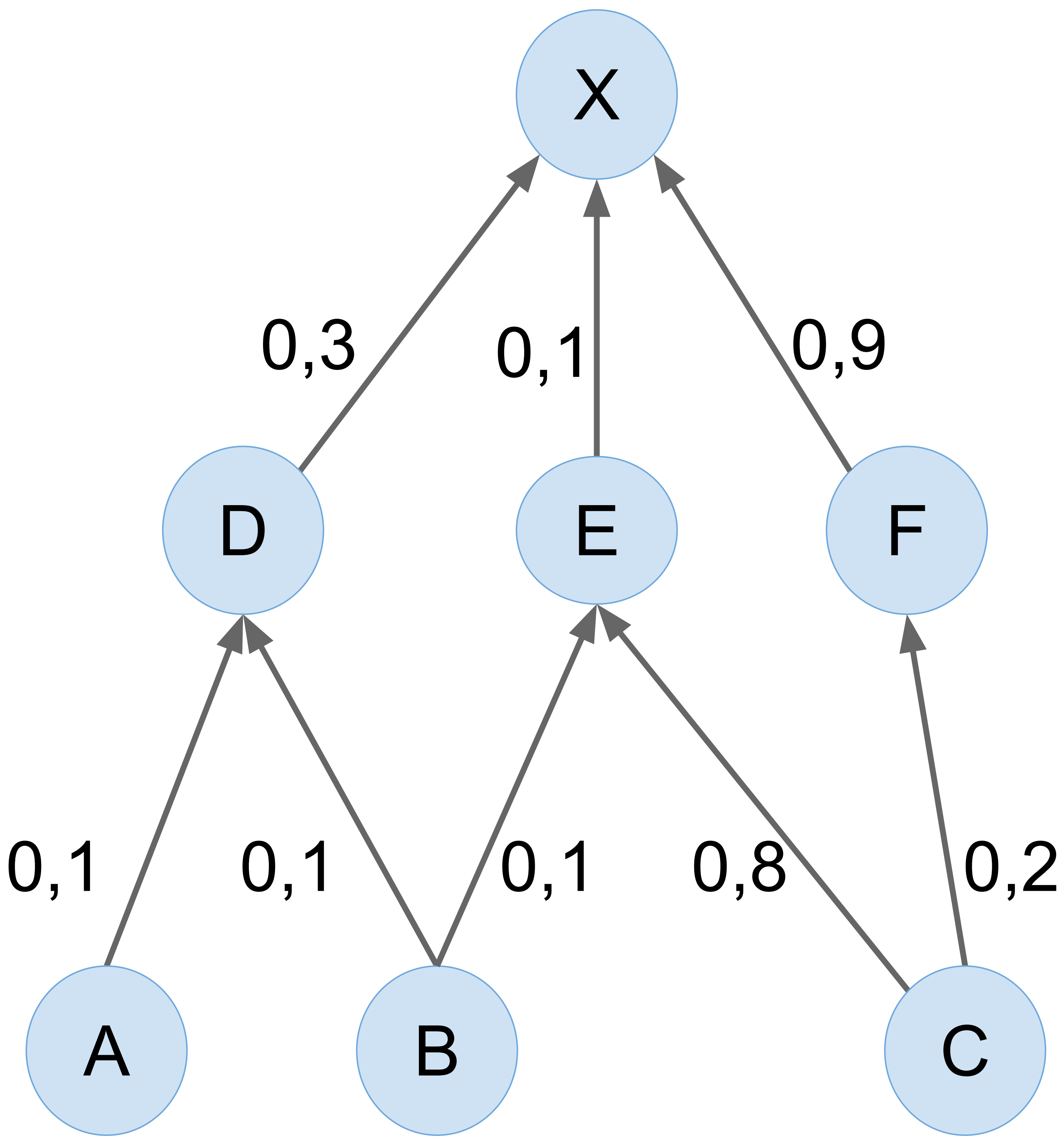
Different genotypes with similar behaviour

Hard to combine parents in “crossover”

Early extinction of some generations



NEAT. *Genetic Encoding*



Keys

- Tacked every connection origin
- Crossover prevention of on competing conventions

Genome

Node Genes

Node A *Sensor*	Node B *Sensor*	Node C *Sensor*	Node D *Hidden*	Node E *Hidden*	Node F *Hidden*	Node X *Output*

Genes connections

In: A Out: D Weight: 0,1 Innov: 1	In: B Out: D Weight: 0,1 Innov: 2	In: B Out: E Weight: 0,1 Innov: 3	In: C Out: E Weight: 0,8 Innov: 4	In: C Out: F Weight: 0,2 Innov: 5	In: D Out: X Weight: 0,3 Innov: 7	In: E Out: X Weight: 0,1 Innov: 8	In: F Out: X Weight: 0,9 Innov: 10

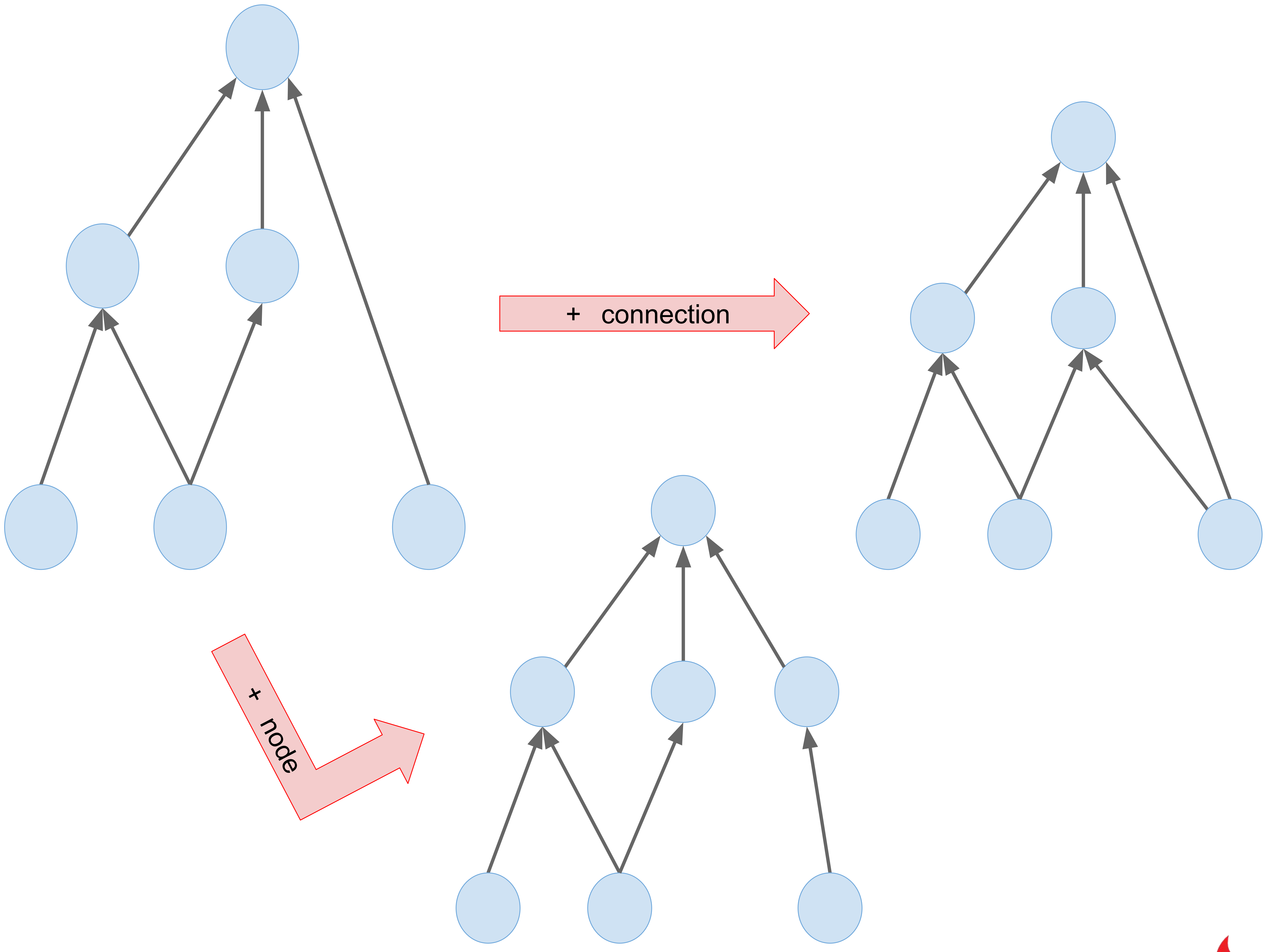
NEAT. *Mutation*

Types

- Weights mutations
- Structures mutations
  - Adding connections
  - Adding nodes

Keys

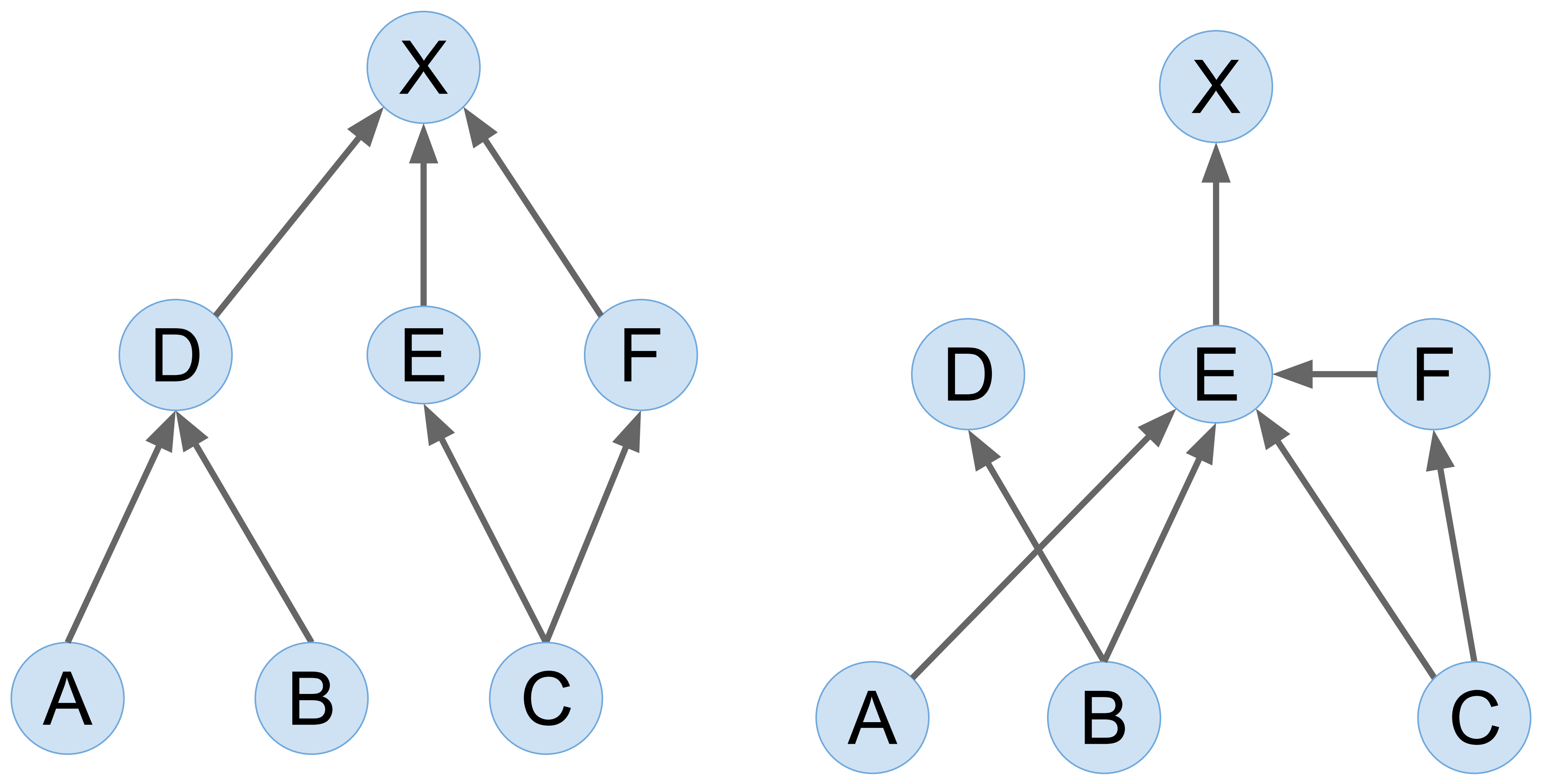
- Genomes get gradually larger
- Fast integration of new nodes



NEAT. *Crossover*

Keys

- Crossover prevention of on competing conventions
- *Matching* vs *Disjoint* vs *Excess*



Innov 1 $A \to D$	Innov 2 $B \to D$	Innov 3 $D \to X$	Innov 4 $C \to E$	Innov 5 $C \to F$		Innov 7 $F \to X$	
Innov 1 $A \to E$	Innov 2 $B \to D$	Innov 3 $B \to E$	Innov 4 $C \to E$	Innov 5 $C \to F$	Innov 6 $F \to E$		Innov 8 $E \to X$

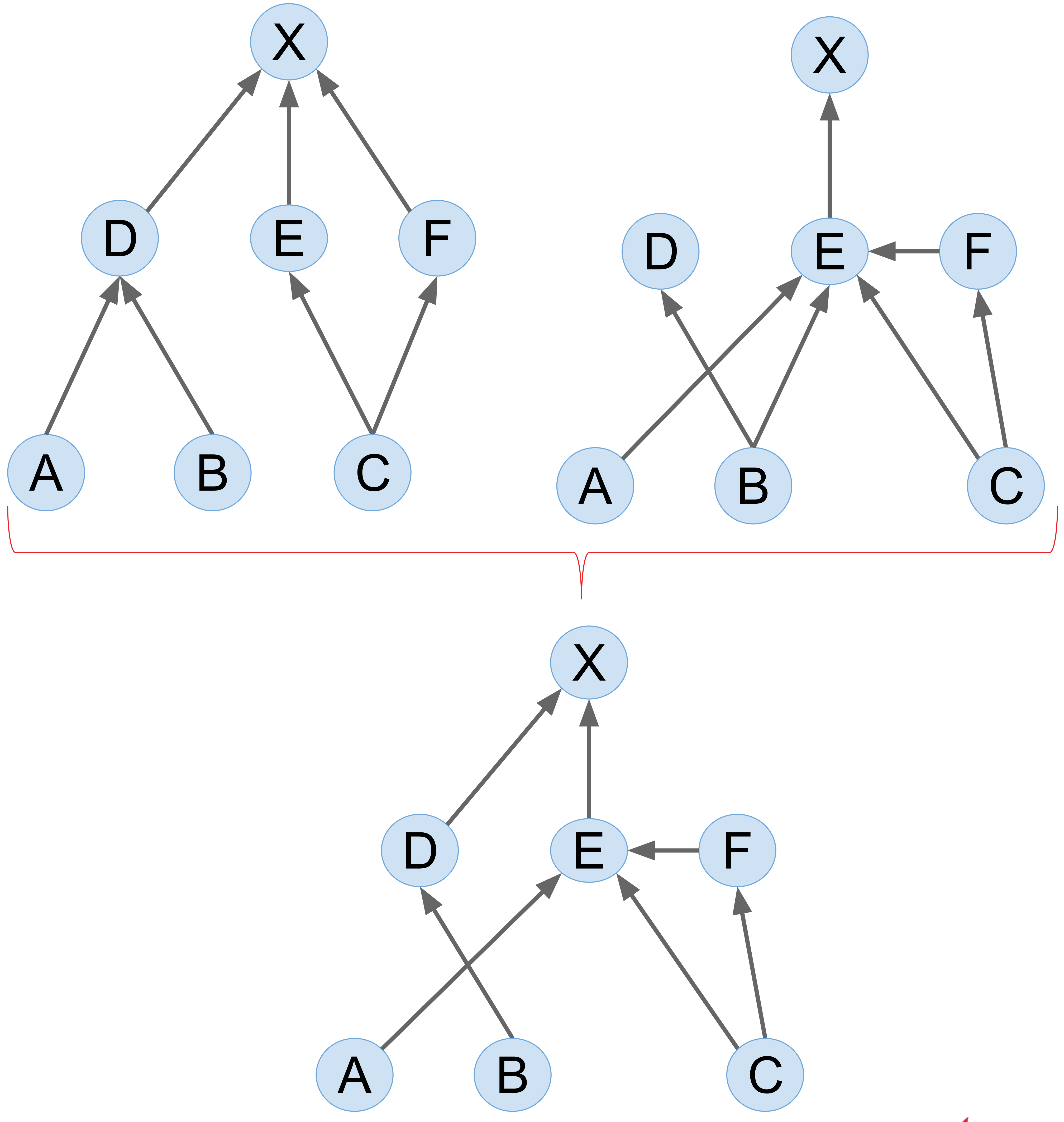
NEAT. *Crossover*

Keys

- Crossover prevention of on competing conventions
- *Matching* vs *Disjoint* vs *Excess*

Innov 1 $A \rightarrow D$	Innov 2 $B \rightarrow D$	Innov 3 $D \rightarrow X$	Innov 4 $C \rightarrow E$	Innov 5 $C \rightarrow F$		Innov 7 $F \rightarrow X$	
Innov 1 $A \rightarrow E$	Innov 2 $B \rightarrow D$	Innov 3 $B \rightarrow E$	Innov 4 $C \rightarrow E$	Innov 5 $C \rightarrow F$	Innov 6 $F \rightarrow E$		Innov 8 $E \rightarrow X$

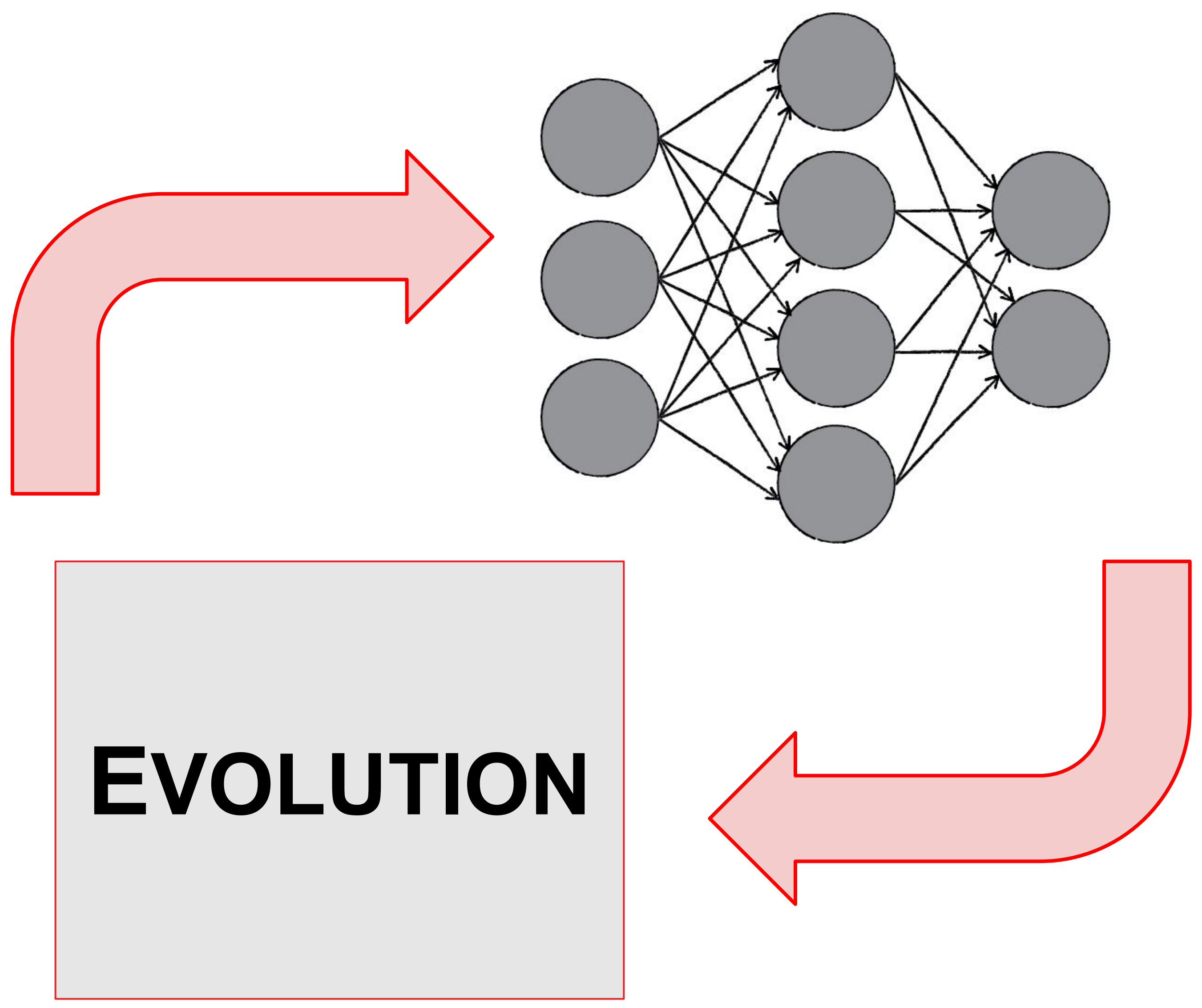
Innov 1 $A \rightarrow E$	Innov 2 $B \rightarrow D$	Innov 3 $D \rightarrow X$	Innov 4 $C \rightarrow E$	Innov 5 $C \rightarrow F$	Innov 6 $F \rightarrow E$	Innov 8 $E \rightarrow X$	



Evaluation approaches

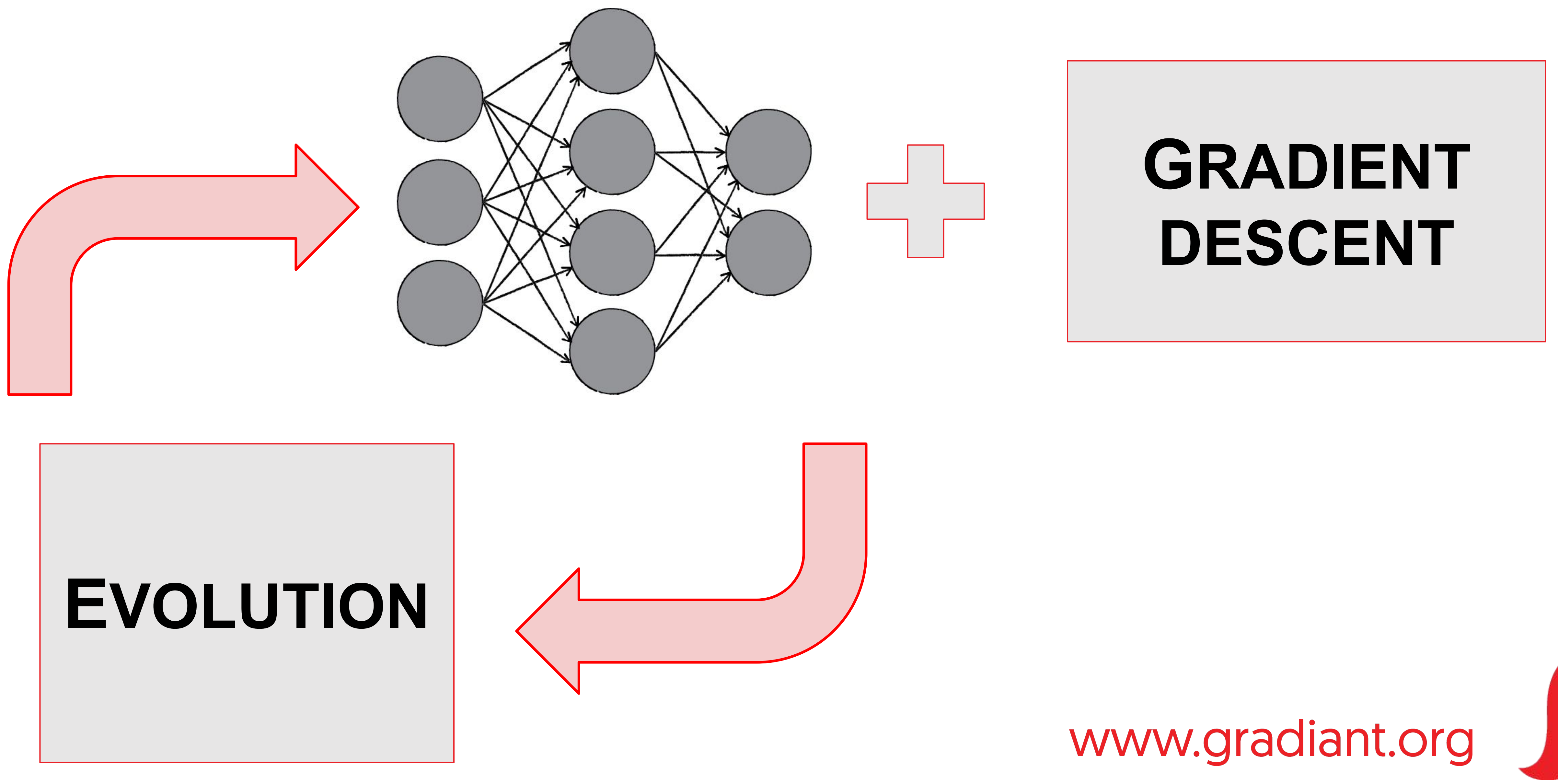
Straight approach

- ANN build and evaluation
- High error on test set
- Pretty slow
- Does not exploit gradient if available

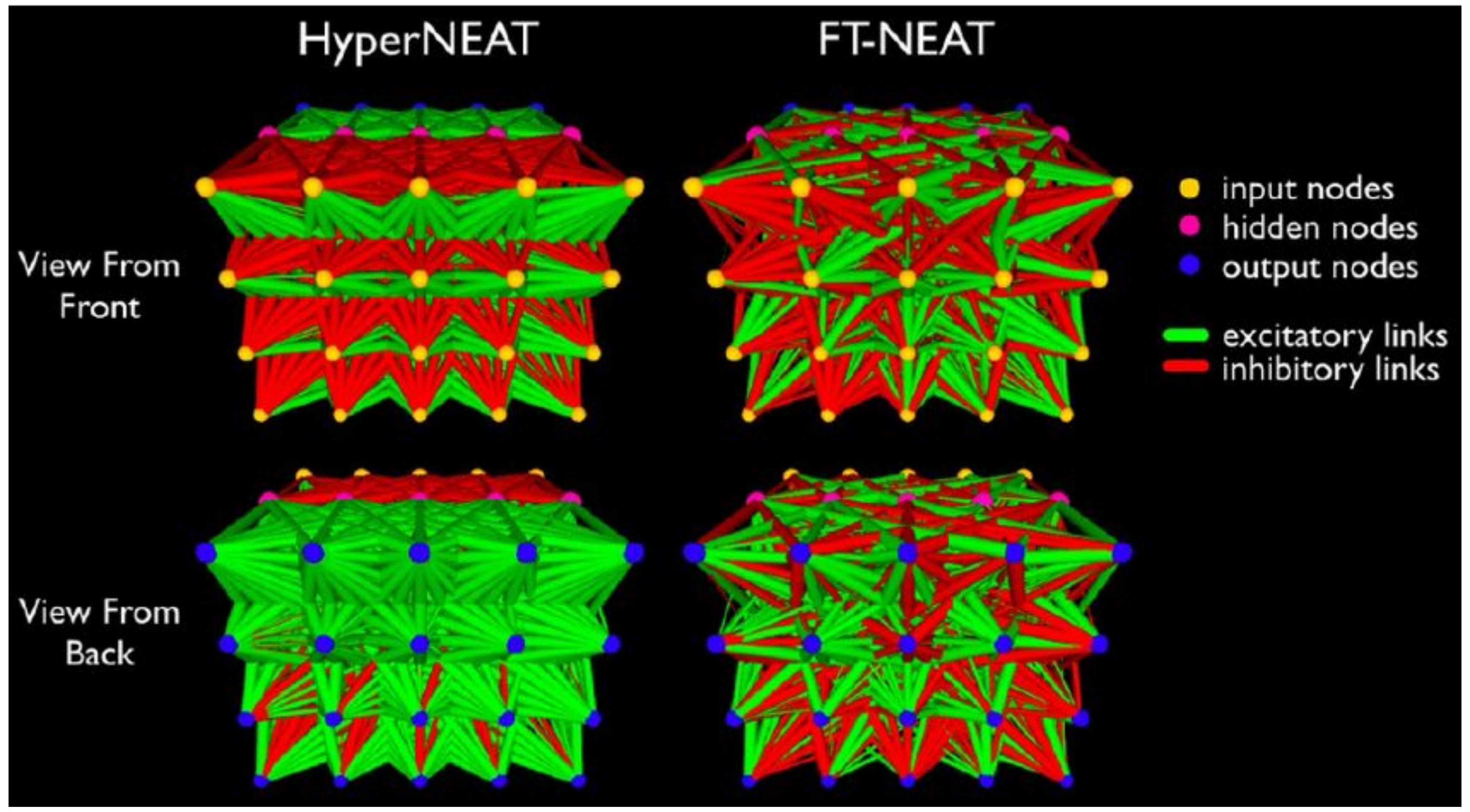


Hybrid approach

- Exploration of EAs
- Exploitation of training (gradient based)
- Training does not change genotype



Scaling up

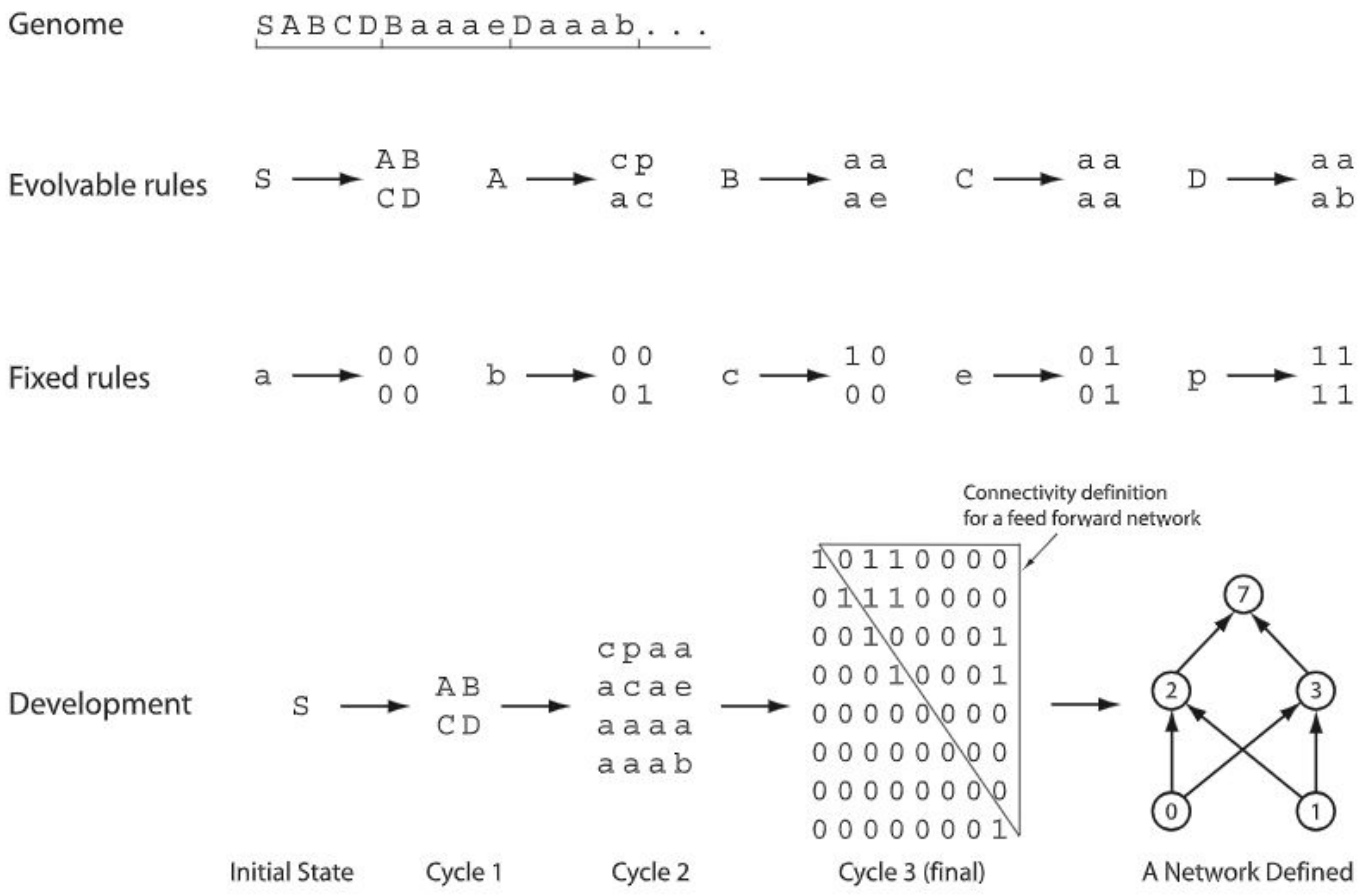


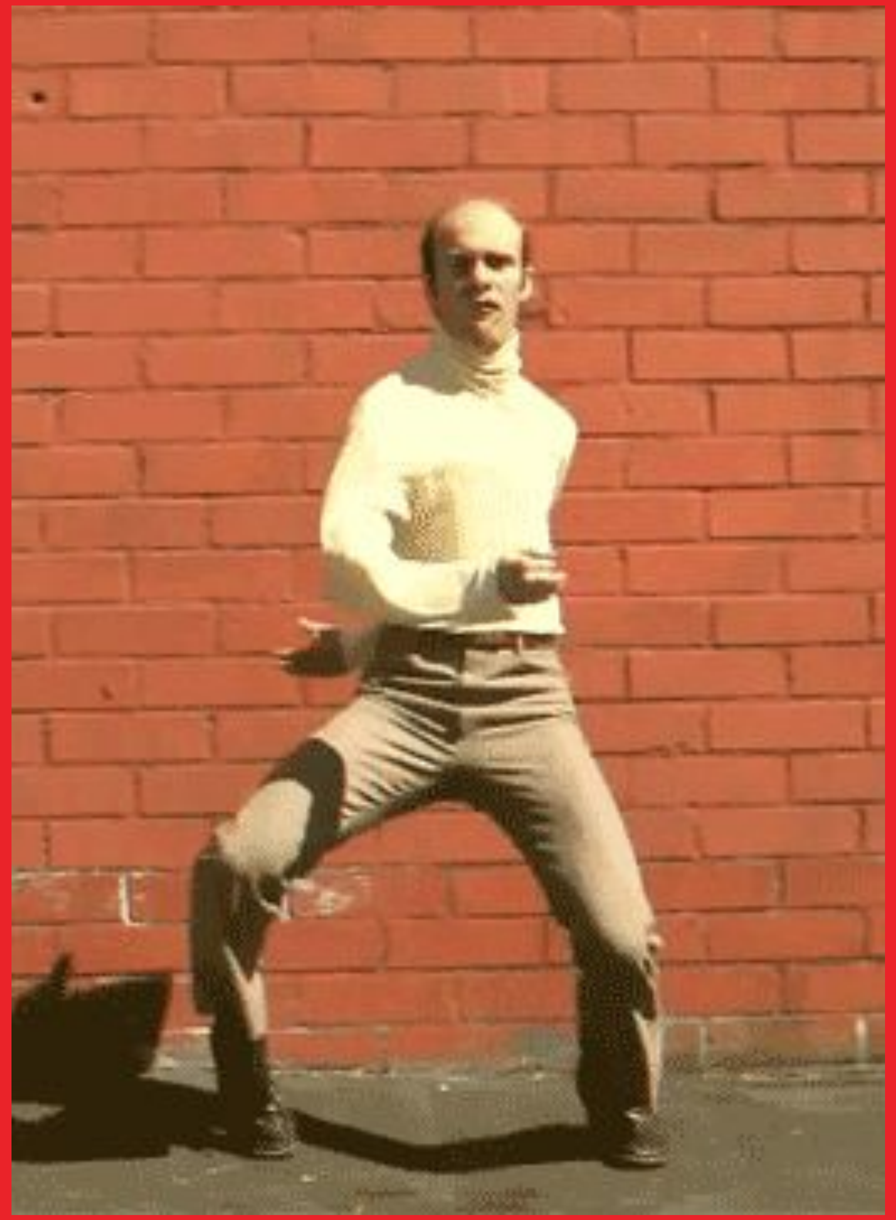
HyperNEAT

FT-NEAT

Kinato’s method

Gruau’s model





Let’s practice!

Frozen Lake

Mario NES