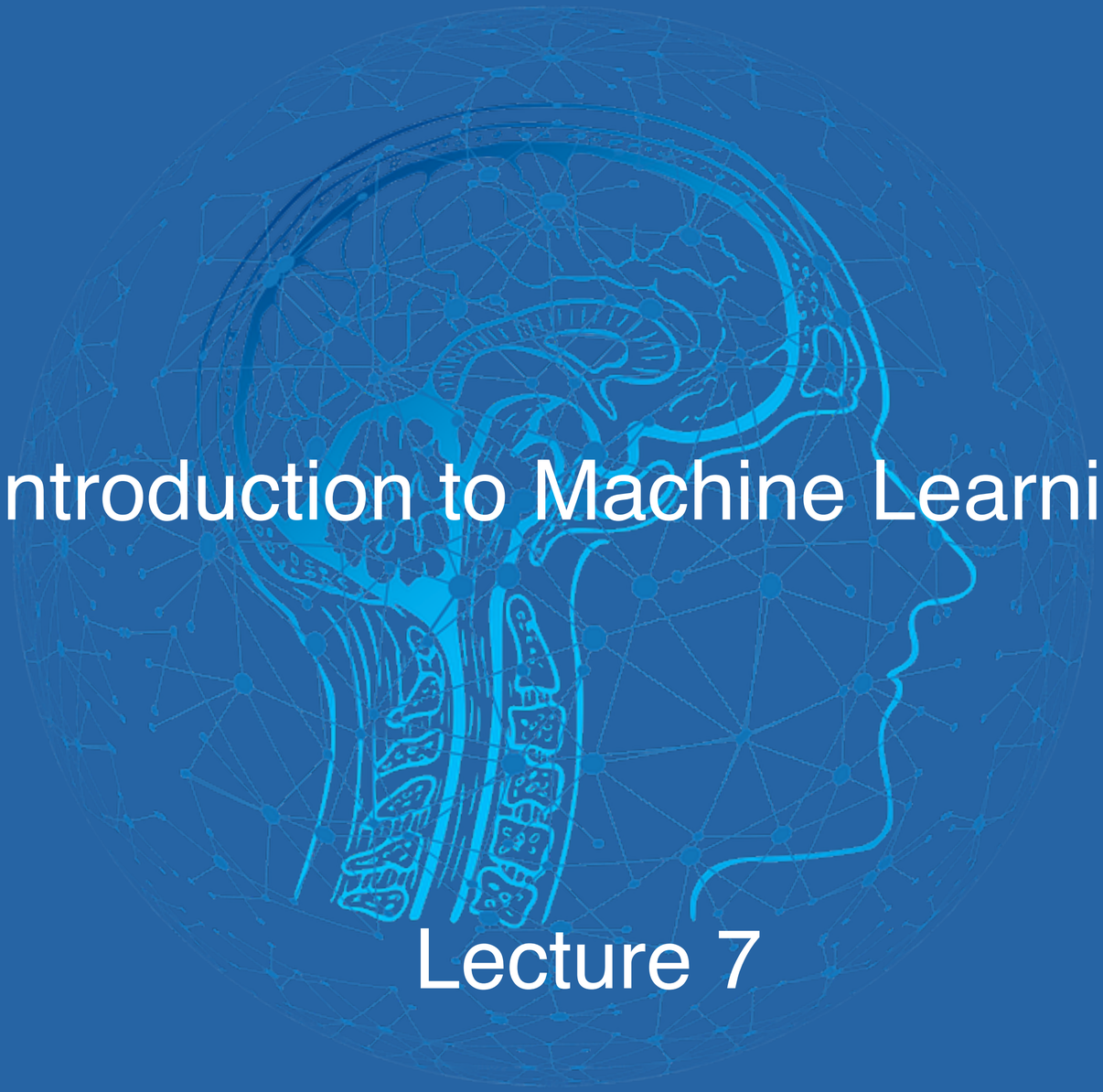
Imperial College London



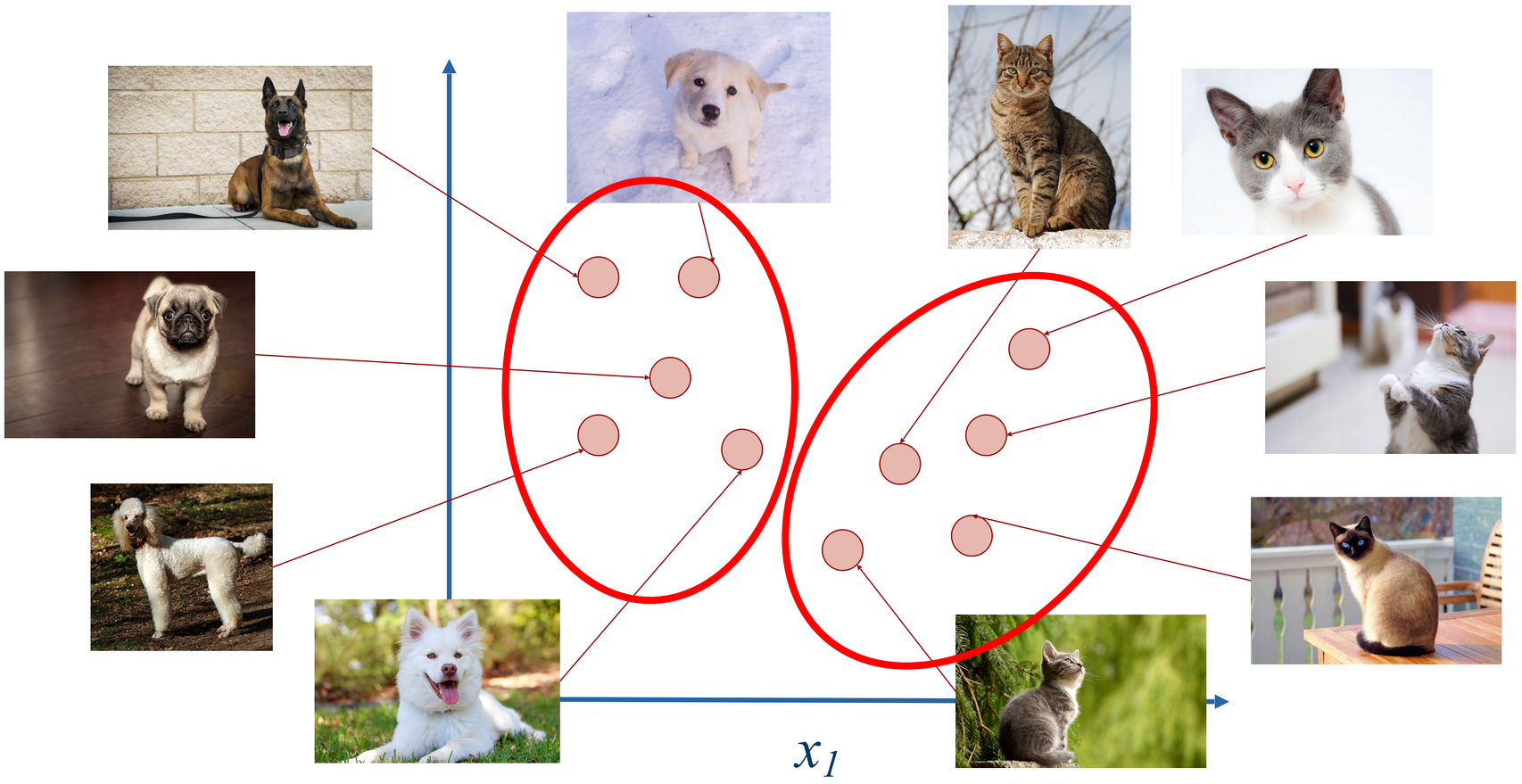
Introduction to Machine Learning

Lecture 7

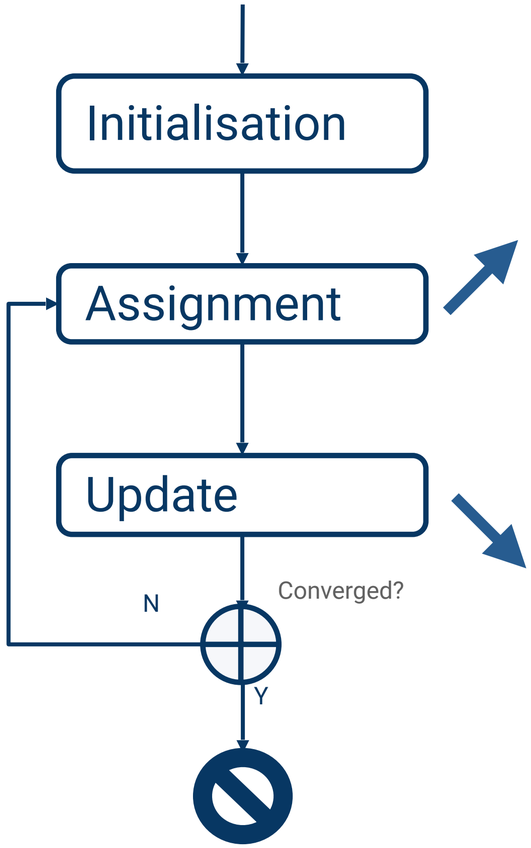
Antoine Cully & Marek Rei & Josiah Wang

In the previous lecture...

Unsupervised Learning

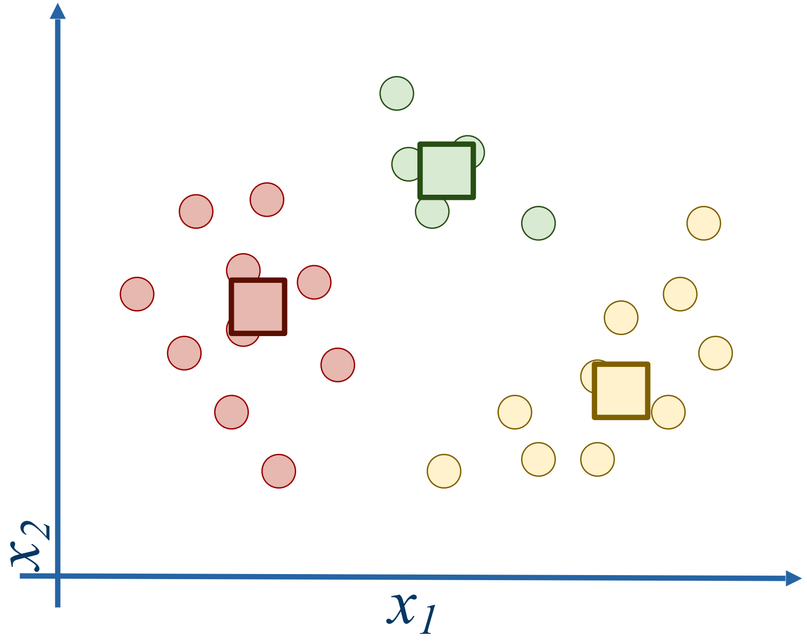


K-means

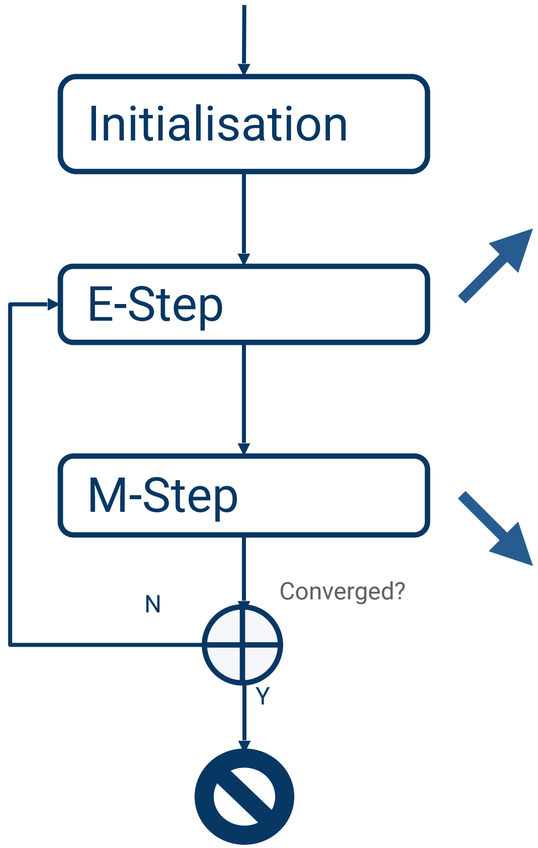


$$\forall i \in \{1, \ldots, N\}$$
$$c^{(i)} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} ||x^{(i)} - \mu_k||^2$$

$$\forall k \in \{1, \ldots, K\}$$
$$\mu_k = \frac{\sum_{i=1}^{N} 1(c^{(i)} = k) \cdot x^{(i)}}{\sum_{i=1}^{N} 1(c^{(i)} = k)}$$



GMM-EM

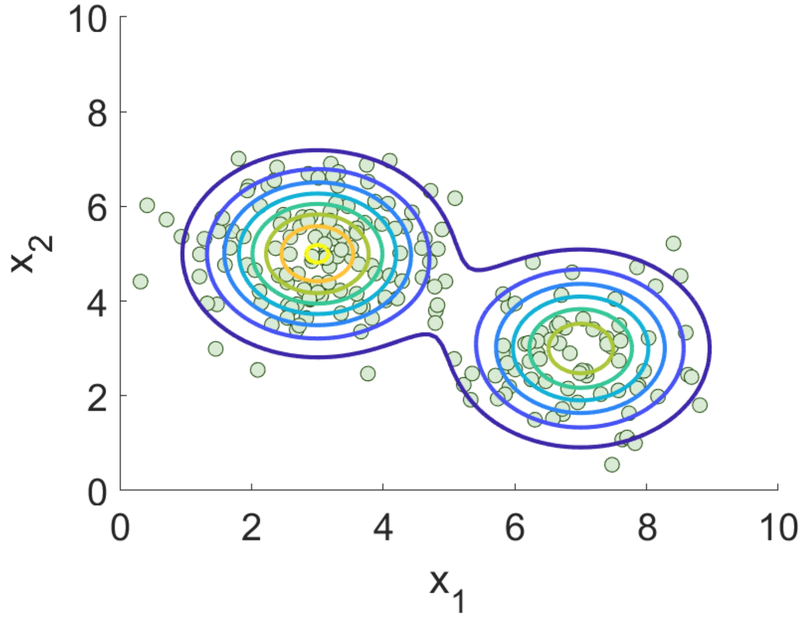


$$r_{ik} = \frac{\pi_k \mathcal{N}\left(\mathbf{x}^{(i)} \middle| \mu_k, \Sigma_k\right)}{\sum_j^K \pi_j \mathcal{N}\left(\mathbf{x}^{(i)} \middle| \mu_j, \Sigma_j\right)}$$

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik} \mathbf{x}^{(i)}$$

$$\mathbf{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik} \left(\mathbf{x}^{(i)} - \hat{\mu}_k\right) \left(\mathbf{x}^{(i)} - \hat{\mu}_k\right)^T$$

$$\pi_k = \frac{N_k}{N} \qquad N_k = \sum_{i=1}^{N} r_{ik}$$



Today...

Course plan

	Lecture	Lecturer
Week 2	Introduction to ML	*Josiah*
Week 3	Instance-based Learning + Decision Trees	*Antoine*
Week 4	Machine Learning Evaluation	*Marek*
Week 5	Artificial Neural Networks I	*Marek*
Week 6	Artificial Neural Networks II	*Marek*
Week 7	Unsupervised Learning	*Antoine*
Week 8	~~Genetic Algorithms~~ Evolutionary Algorithms	*Antoine*

Today's lecture

8 videos:

1. This short intro
2. History and general concepts
3. Genetic algorithms
4. Evolutionary Strategies
5. Novelty Search
6. Quality-Diversity optimisation
7. MAP-Elites
8. Concluding thoughts

To be continued...
(History and general concepts)

History and general concepts

What is a genetic/evolutionary algorithm?

- An Optimisation algorithm for **black-box functions** (no access to the gradient)
- Inspired by natural evolution and concepts from genetics
- This can be seen as a **reinforcement learning problem**

Reinforcement learning problem

"Reinforcement learning is learning what to do so as to maximise a numerical reward signal."

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*.

Traditional RL algorithms

↑

Discrete state and action spaces

Policy search Algorithms

↑

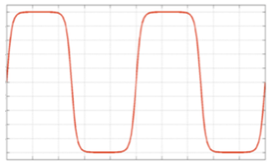
continuous search space

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$$

Kober, J., Bagnell, J.A., and Peters, J. (2013). *The International Journal of Robotics Research*

Black-box optimisation problems

Robot learning:

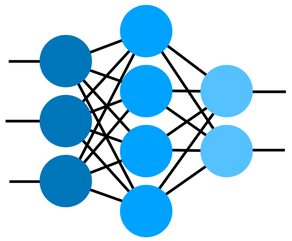
$\mathbf{x}$ →  →  → $f(\mathbf{x})$

Set of parameters

Motor Commands (controller)

Performance from sensors

Hyper-parameter tuning and training:

$\mathbf{x}$ →  →  → F1-score

Set of parameters (nb of layers, neurons, weights values…)

Neural network

Evaluation set

Metric

Others:

$\mathbf{x}$ →  →  → score

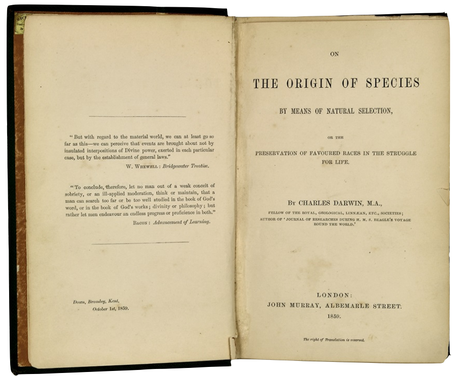
Set of bits

Potential combination

Performance

Introduction and principles

- 1859, Darwin's theory about the origin of species.
- It mainly relies on 4 concepts:
  - **Every individual is different** even within the same specie and these variations are heritable.
  - Ressources are finite: Not all the produced individuals will **survive**
    - **Fight for survival**
  - Individuals with heritable traits **better suited** to the environment will survive.
    - **Natural selection**
  - The survivants **pass their heritable** traits to their offspring.


ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By CHARLES DARWIN, M.A.,

FELLOW OF THE ROYAL, GEOLOGICAL, LINNEAN, ETC., SOCIETIES;

AUTHOR OF "JOURNAL OF RESEARCHES DURING H. M. S. BEAGLE'S VOYAGE ROUND THE WORLD."

LONDON:

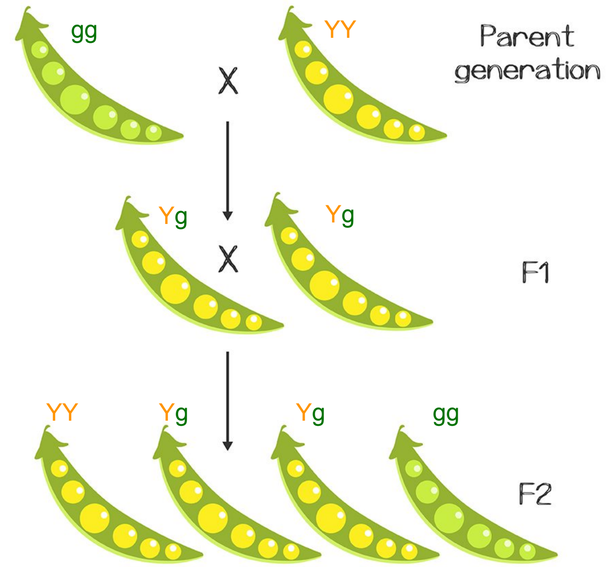
JOHN MURRAY, ALBEMARLE STREET.

1859.

"Variation is a feature of natural populations and every population produces more progeny than its environment can manage. The consequences of this overproduction is that those individuals with the best genetic fitness for the environment will produce offspring that can more successfully compete in that environment. Thus the subsequent generation will have a higher representation of these offspring and the population will have evolved."

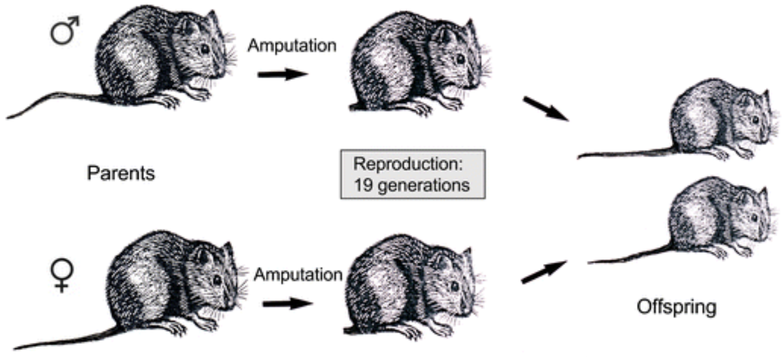
Charles Darwin

Neo-darwinism



- Mendel, 1866 : Principles of statistical inheritance (experiments with peas)
  - Discovery of the concept of genomes (without observing them)
- Weismann, 1883 : Acquired traits are not passed to off-spring (experiments on mice tails).
- Watson, Crick and Franklin, 1953 : DNA structure
- Many discoveries in genetics happen during the 20th century.

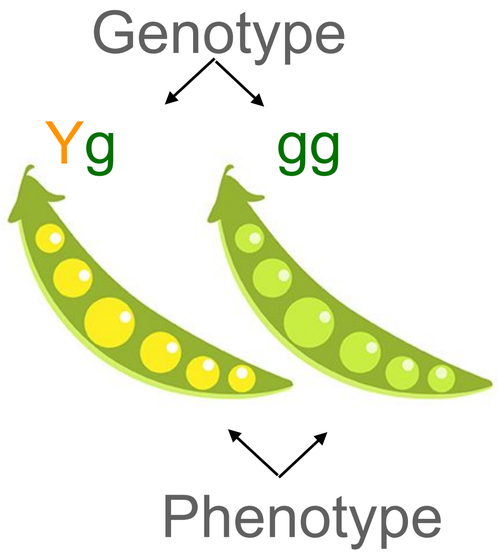
- The Darwinian theory incorporating all these modern concepts in genetics is called:
  Neo-Darwinism



From Karl J. Niklas and Ulrich Kutschera

Modern genetic vocabulary

- **Gene**: sequence of nucleotides in DNA that codes a particular trait.
- **Genotype**: Set of genes
- **DNA** = support of the genotype
- **Phenotype**: physiological expression of the genotype



Algorithms inspired by the neo-darwisism

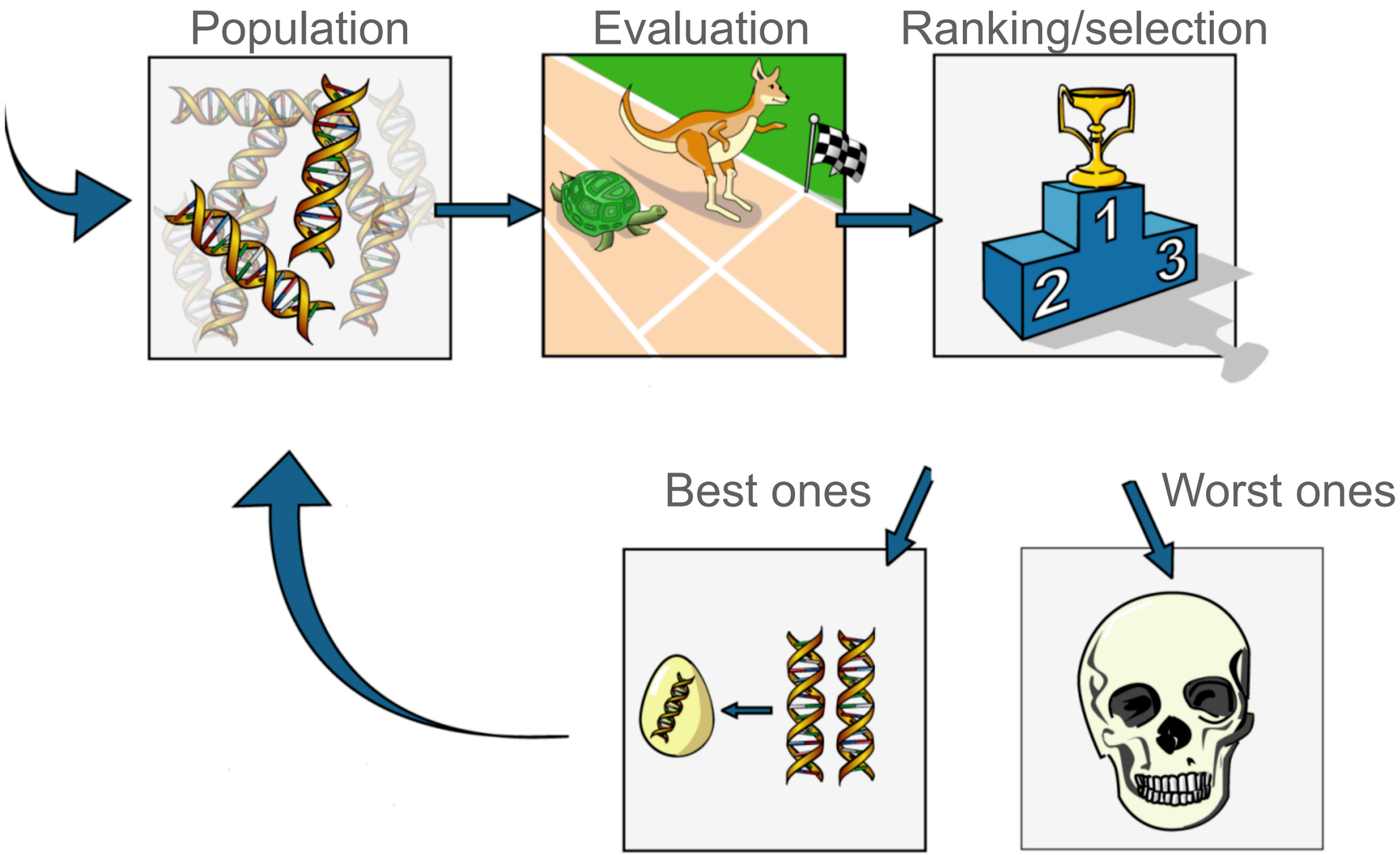
- Three main families of algorithms are independently proposed during the 60s:
  - **Evolutionary Strategies:** In Berlin, Rechenberg et Schwefel.
  - **Evolutionary programming:** Evolution of finite state machines (from UCLA (california), Fogel).
  - **Genetic algorithms:** Design of complex and robust systems (from Michigan university, Holland).

Other approaches

Other, more recent approaches:

- **Genetic programming**: Evolution of programmes
- **Estimated Distribution algorithms**: Statistical model of the population
- **Multi-objective algorithms**: Simultaneously optimise several objectives.

Main concept



Common principle

All these algorithms use a very simplified version of Neo-darwinism as **a common base**:

- They manipulate **a population of solutions**, initially randomly generated.
- Each solution is represented by **a genotype**
- The genotype can be developed into **a phenotype**
- They use a **fitness function** to measure the performance of the phenotypes.
- The **fittest individuals** have a higher probability to pass part of their genotype to the offspring.

Genetic operators

The algorithms are often defined according to three main operators:

- **Selection operator:** Selects the solutions that will be reproduced.
- **Cross-over operator:** mixes the parents' genotype.
- **Mutation operator:** Type and frequency of the variations applied to the genotype after reproduction.

Differences between the algorithms:

- Definition of the Genotype.
- Genetic operators used.

Differences

Genetic algorithm:

- Genotype: binary string of fixed size (ex. 01001010)
- Cross-over: swap portions of the string
- mutation: Random flip of bits

Genetic programming:

- Genotype: Program represented as tree (often in LISP)
- Cross-over: Swap portions of trees
- mutation: Change the symbols in the tree

Evolutionary Strategies:

- Genotype: String of floats/doubles
- Cross-over: usually not used
- Mutation: Draw from a Gaussian distribution

Modern view: unification

Now, we mix all these approaches:

- Genotype are often strings of floats/doubles
- These genotypes can be used to encode graphs
- Heterogenous list of parameters
- Large number of variants of the operators
- …

We now call them **Evolutionary Algorithms (EA)**.

To be continued...
(Genetic algorithms)

Genetic algorithms
Let’s play Mastermind

Example: Solving Mastermind

Concept:

- Each piece can have 6 different colours
- A secret combination of N colour is defined by the game-master
  (N=4 in the original game)
- Duplicates are possible
- **Goal: finding the combination**
- Information:
  - Number of pieces with the right colour and the correct position
  - Number of pieces with the right colour but the wrong position

Defining the Evolutionary algorithm

For this, we need:

- The fitness function
- The genotype and phenotype
- The selection operator
- The cross-over operator
- The mutation operator
- (and some hyper-parameters, like the population size).

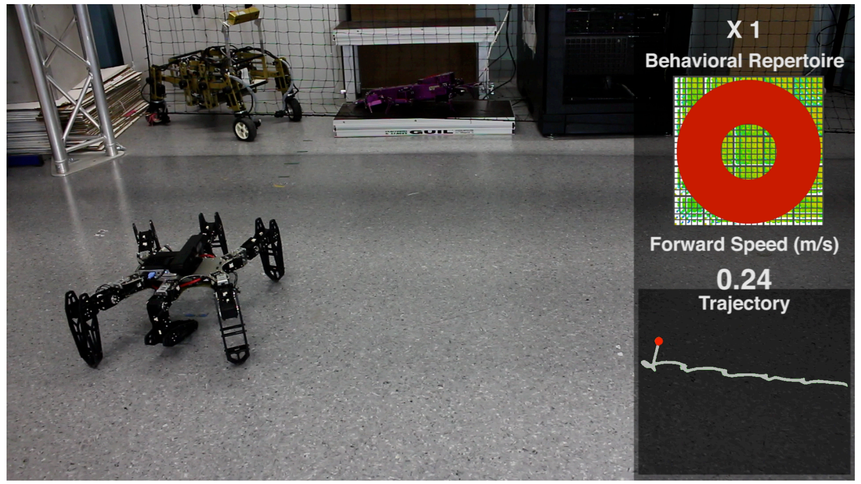
Fitness function

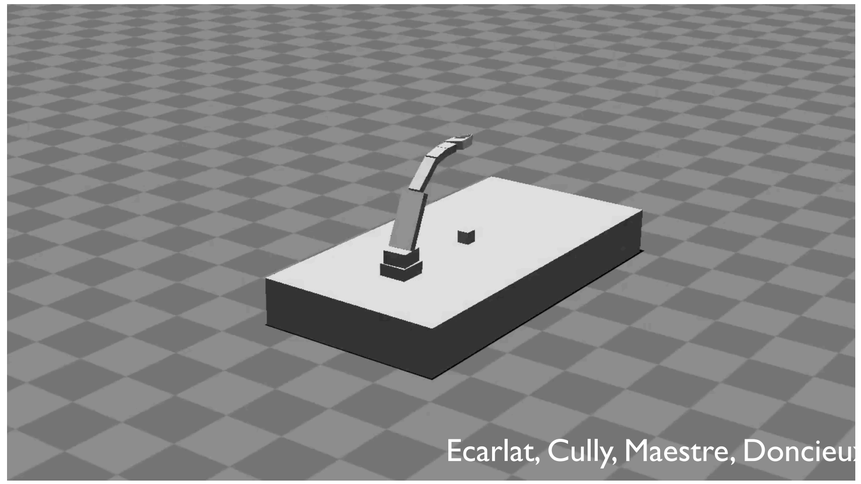
- **Role:** it represents the problem we want to solve
- **Solving the problem = maximising the fitness**
- For the mastermind game:
  - p1 = number of pieces with the right colour and the correct position
  - p2 = number of pieces with the right colour but the wrong position
  - F(x) = p1+0.5 p2
- We need to be sure that the maximum of this function corresponds to solving the problem (F(x) = 4).

Fitness function

- Each problem will have a different fitness function:
  The fitness function is problem specific!
- Teaching a robot to walk?
  F(x) = walking speed = traveled distance after a few seconds.
- Teaching a robot to throw an object to a target?
  F(x) = - distance(object, target).

In EAs we usually maximise, because the are looking for the "fittest individuals".





Genotypes and Phenotypes

- **Role: They represent the potential solutions to the problem**
- For the Mastermind game:
  - Binary string (other options are possible)
  - Given N, the number of hidden pieces
  - Genotype: binary string with N*3 bits
  - Phenotype: We aggregate the bits 3 by 3,
    each trio of bits is turned into an integer
  - Then each integer corresponds to a different colour.
    (0= red, 1=yellow, 2=green, 3=blue…)
- Many other genotypes/phenotypes are possible
  (often problem specific).

Example:

000011101001

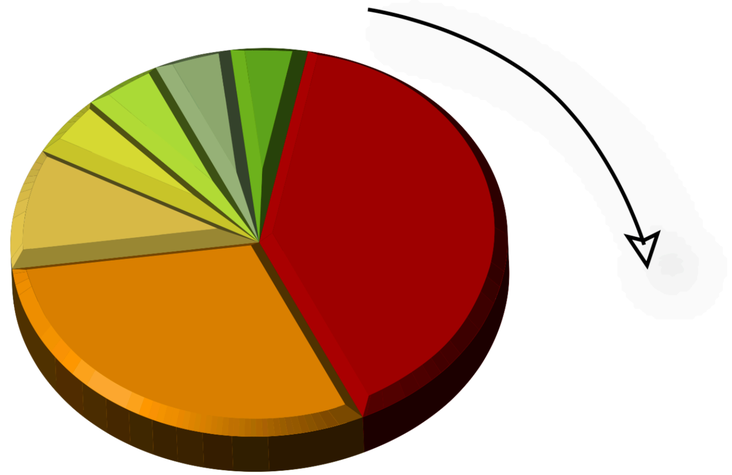
000 011 101 001

0 3 5 1

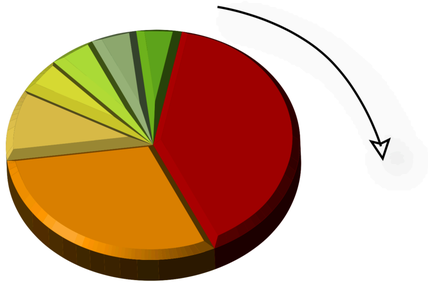
Red, **Blue**, **Orange**, **yellow**.

Selection operators

- **Role: Select the parents for the next generation**
- Many options exist
- A standard approach: the biased roulette wheel.
- Each individual is associated to a portion of the circle
- Individual with a high fitness with have a larger proportion of the circle.
- A random location on the circle is randomly drawn
- The likelihood for an individual to be selected is proportional to its fitness.



Biased roulette wheel

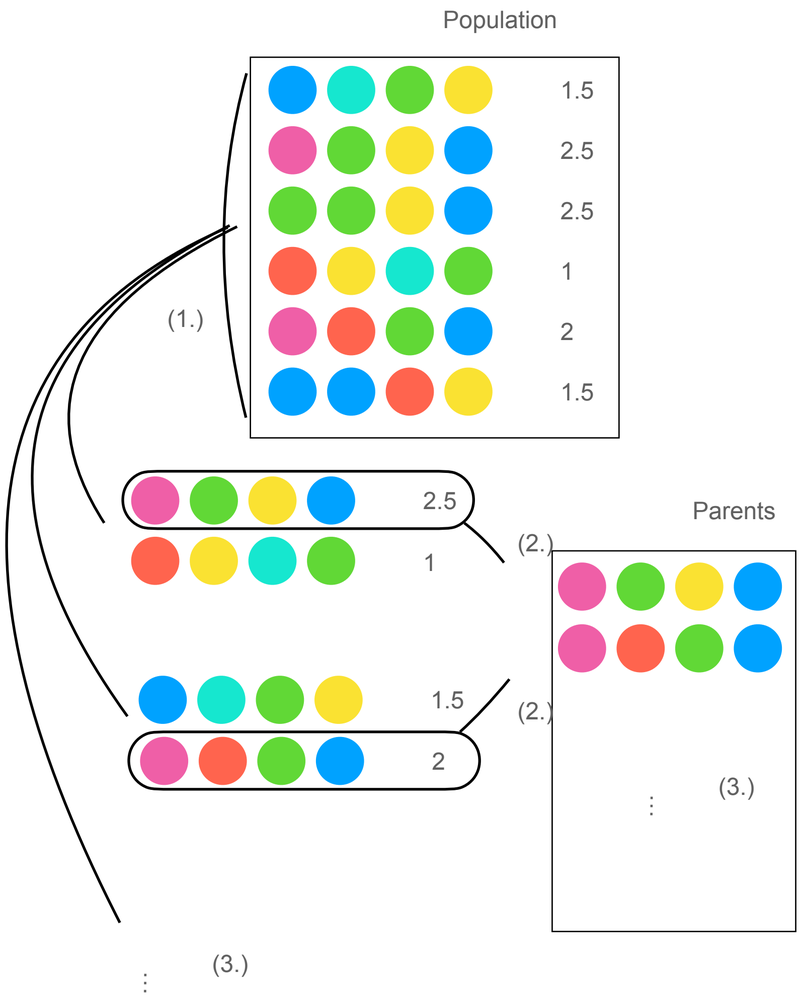


Process:

- Compute the probability p_i to select an individual from the population:
- Compute the cumulative probability
- Randomly generate (uniform distribution) a number r between 0 and 1.
- Select the individual x_i so that:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \qquad q_i = \sum_{j=1}^{i} p_j \qquad q_{i-1} < r \leq q_i$$

Alternative: Tournament



Simple version:

1. Randomly draw two individuals from the population
2. Select the best out of the two
3. Repeat this until you have enough parents.

Numerous variants:

- Drawing multiple individuals
- Not always selecting the best one

Advantages:

- Take into account only relative values of the fitness
- Easy to parallelise

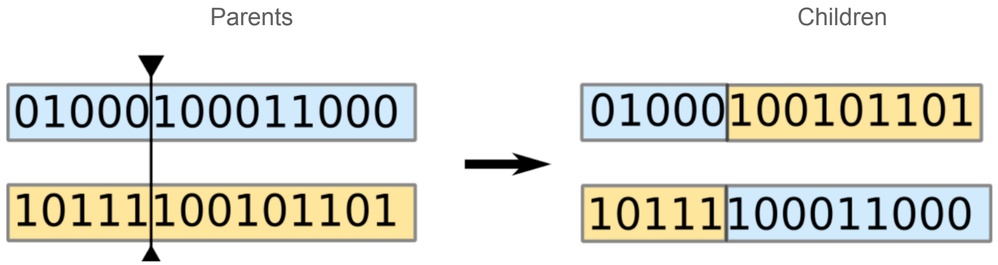
Elitism

- In the general description of an EA, the next generation completely replaces the current generation.
- It is often useful to keep in the new generation a fraction of the best individuals found so far.
- We call this **elitism**
- We usually fix this fraction to 10%
- Advantage: **The fitness of the best individual in the population cannot decrease**.

Cross-over operators

- **Role: Combine the traits of the parents**
- Single-point cross-over: standard operator on strings
- A split point is randomly picked
- The genotype of the offspring (children) is formed by exchanging the portions of genotype

- Variant with multiple split-points exist.



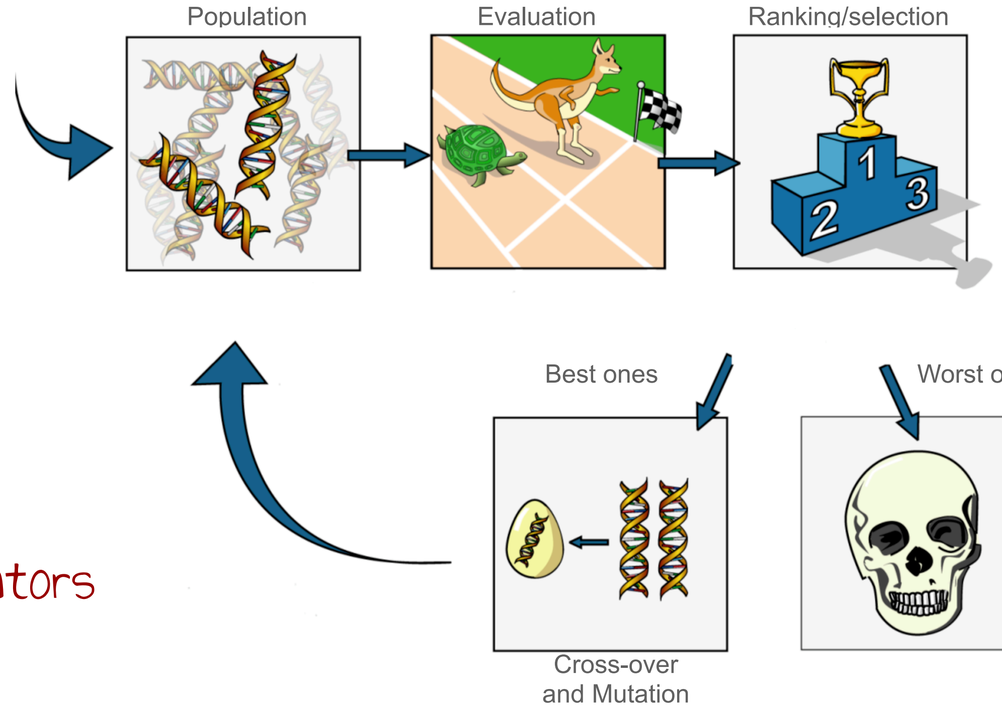
Mutation operators

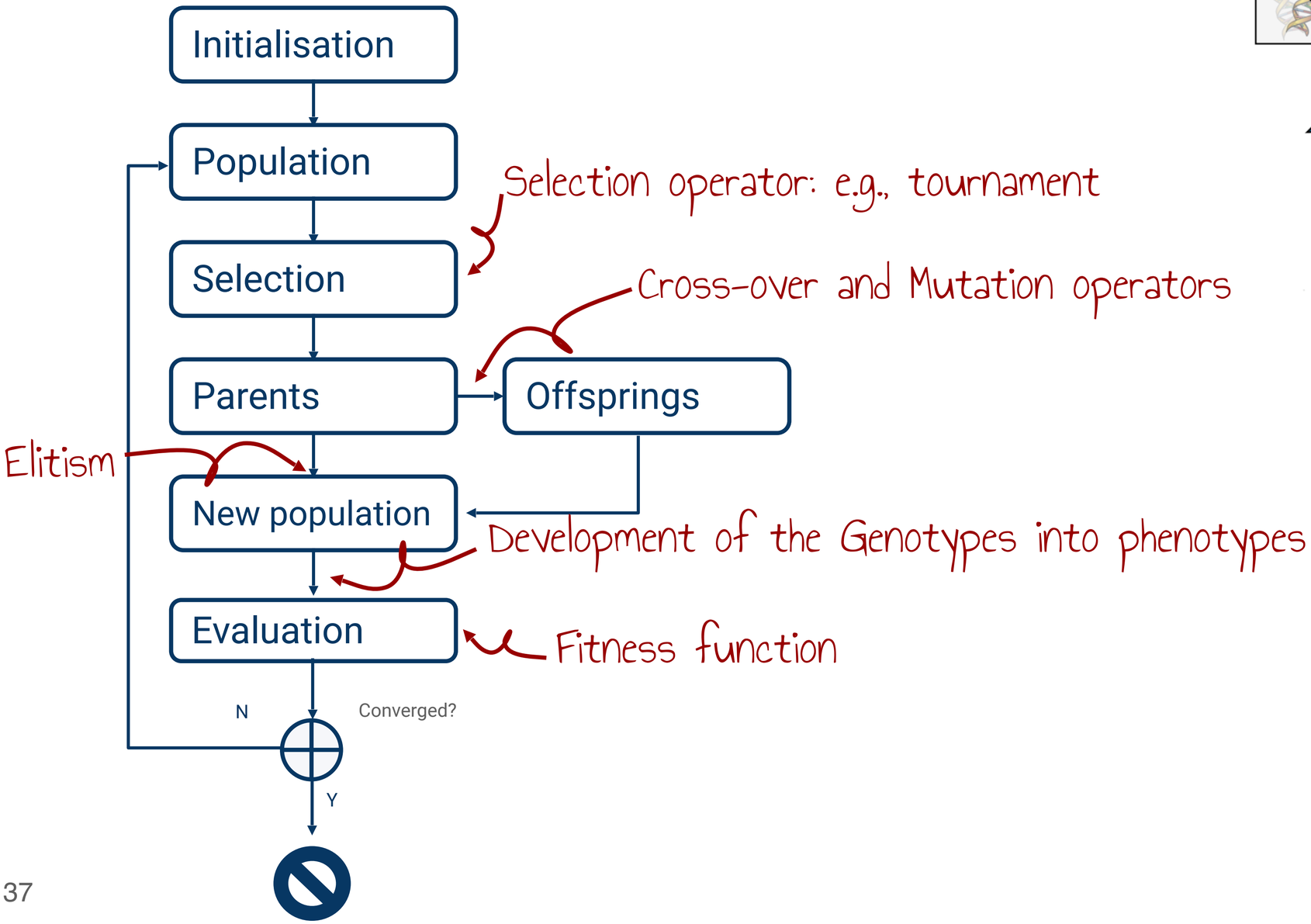
- **Role: explore nearby solutions**
- Standard mutation on binary strings:
  - For each bit of the genotype, a number is randomly generated between 0 and 1 (uniform distribution)
  - If this number is lower than a probability *m* (hyper-parameter), the bit is flipped.
  - *m* is often fixed to 1/(size of the genotype).
- We can also add a specific mutation for the mastermind problem:
  - With standard operators, we cannot exploit the information: good colour, wrong place (we cannot move a part of the genotype).
  - We can swap groups of 3 bits in the genotype with a probability m_2

Stopping criterion

- Commonly used stopping criterion:
  - When A specific fitness values is reached
  - After a pre-defined number of generations/ evaluations
  - When the best fitness in the population stagnate for a predefined number of generations.

Genetic algorithms

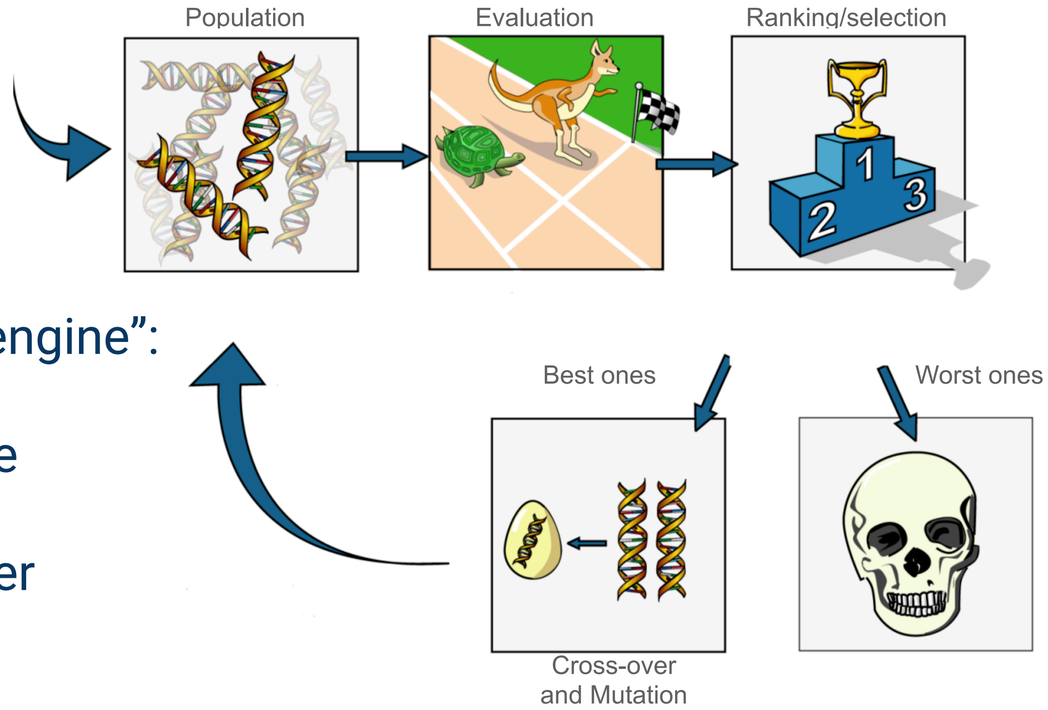


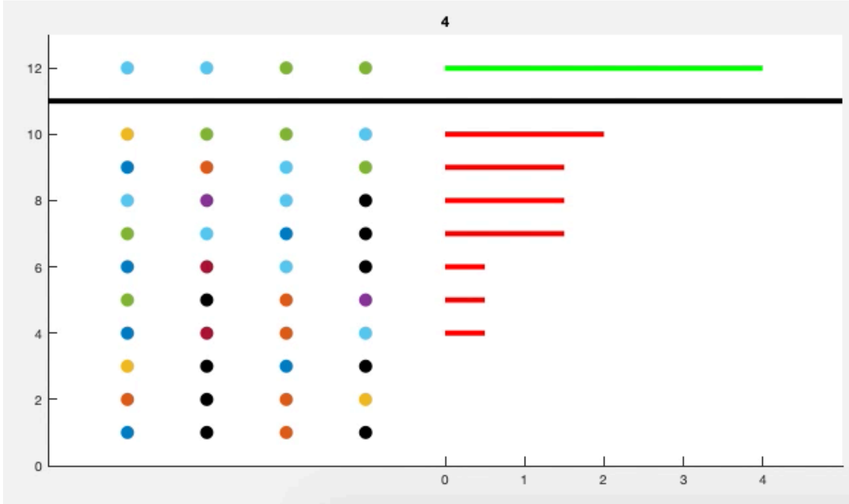


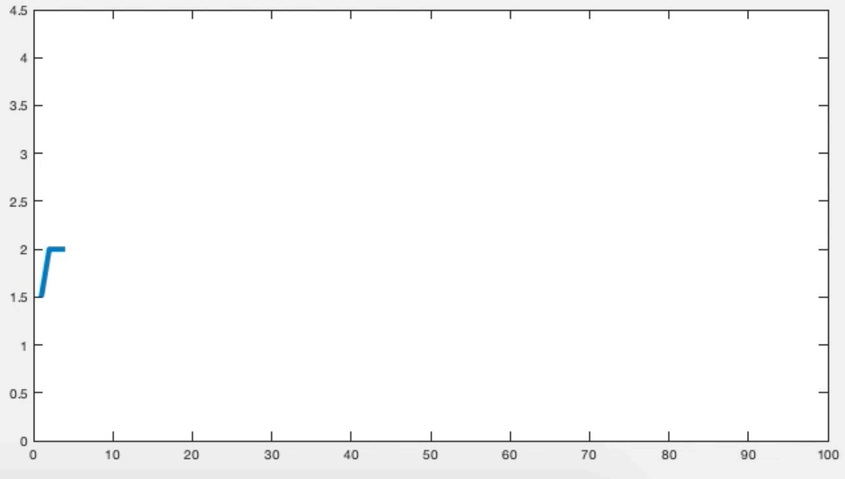
Mastermind: Sum-up

Example of simple implementation:

1. Randomly generate a population of N binary strings
2. Evaluate the fitness of each individual using the “game engine”: F(x) = p1 + 0.5 p2
3. Select N/2 couples of parents (or less if elitism) w.r.t. the fitness.
4. For each couple, create their offspring with the cross-over operator
5. Mutate each child
6. Replace the old population with the new one, return to (2)







To be continued...
(Evolutionary strategies)

Evolutionary Strategies (dealing with real numbers)

Evolutionary Strategies

- Previous part: Genetic algorithm
  - Genotype = binary string
  - Optimises a parameter list of integers

- However, a lot of problems involve real parameters (e.g., neural networks).

- A different approach has been introduced in the 60s:
  Evolutionary Strategy
  - **Genotype** = list of reals
  - **Parent selection**: Uniform
  - **Mutation**: generated from a Gaussian

- Similarities: Notion of population and selection
- Differences: Genotype, no cross-over, selection method.

$(\mu + \lambda) - ES$

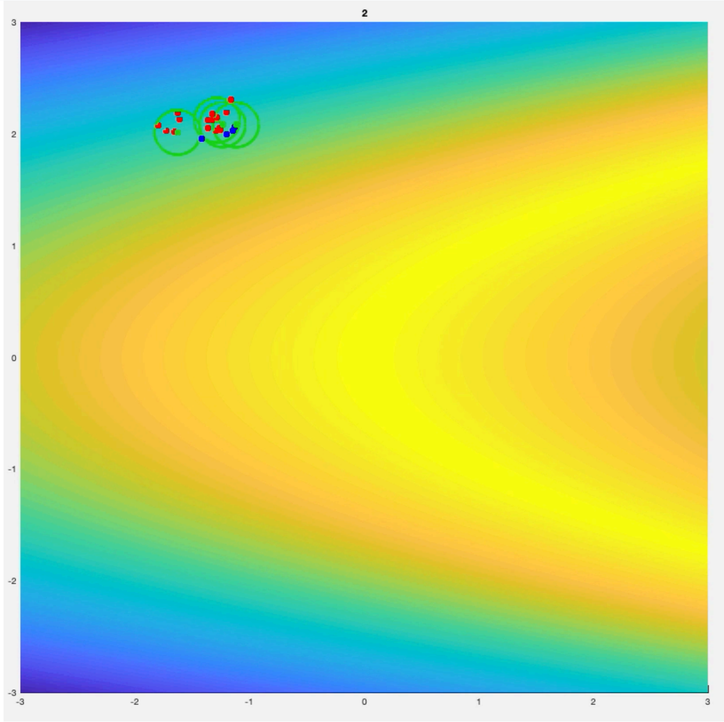
1. Randomly generate a population of $(\mu + \lambda)$individuals.
2. Evaluate the population
3. Select the μ best individuals from the population as parents (called x)
4. Generate λ offsprings (called y) from the parents:

$$y_i = x_j + \mathcal{N}(0,\sigma) \qquad j = randi(\mu)$$

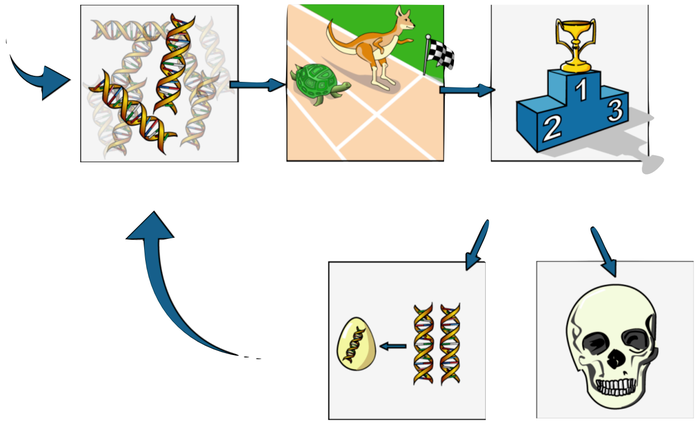
5. Population = union of the parents and offspring:

$$\mathbf{pop} = (\cup_i^{\lambda} y_i) \cup (\cup_j^{\mu} x_j)$$

6. Return to 2



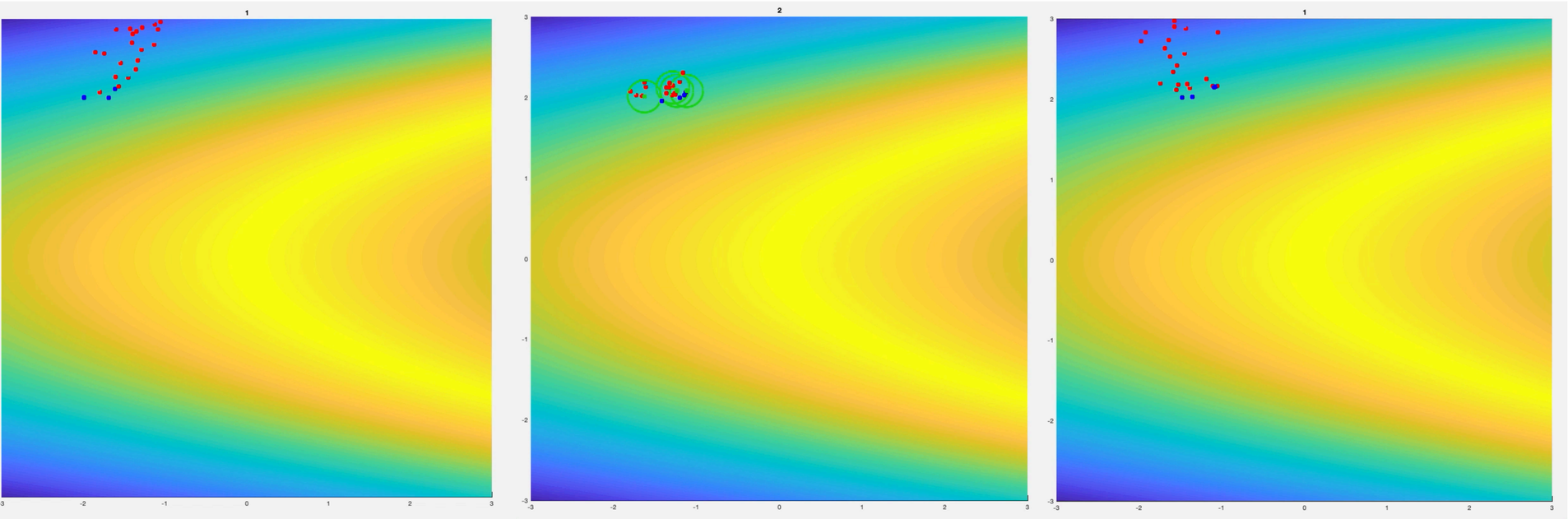
- Very similar to a genetic algorithms
- Interesting, very simple, variant (1+1)-ES
- Usually: $\lambda/\mu \approx 5$
- Main challenge: fixing σ



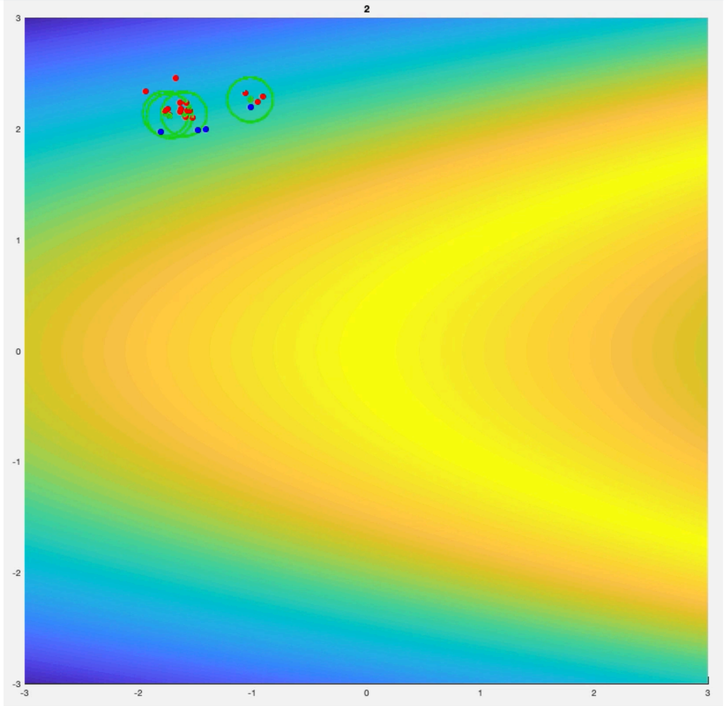
$(\mu + \lambda) - ES$

- The value of sigma is crucial:
  - Large sigma: The population moves quickly to the solution, **but** has a hard time to refine it.
  - Small sigma: The population moves slowly and might be more affected by **local optimums**.

$\sigma = 0.05$ $\sigma = 0.1$ $\sigma = 0.5$



$(\mu + \lambda) - ES$



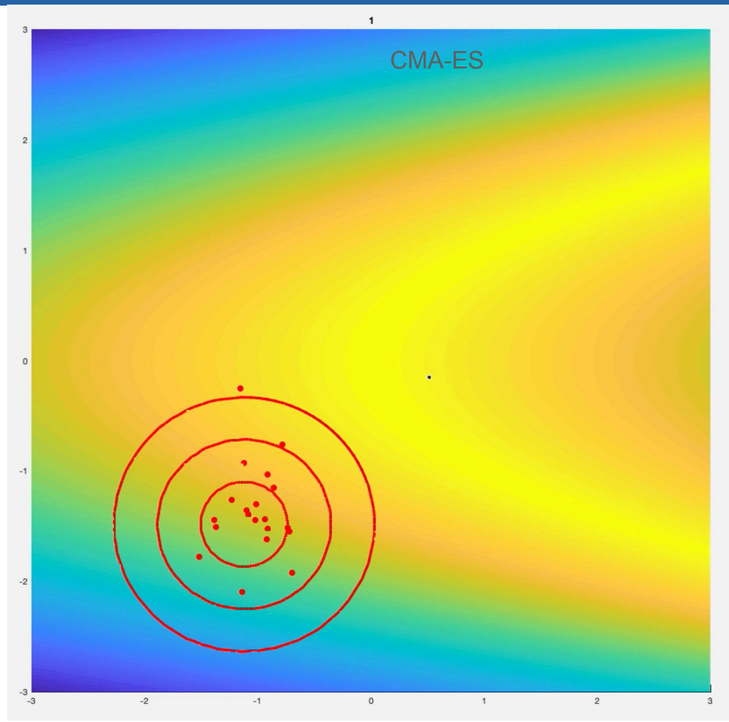
- Adapting sigma over time (or depending on the situation)
- Concept: Add sigma in the genotype:

$$x_j' = \{x_j, \sigma_j\}$$

$$\sigma_i = \sigma_j \exp(\tau_0 \mathcal{N}(0,1))$$
$$y_i = x_j + \sigma_i \mathcal{N}(0,1)$$

- τ_0 Is the learning rate
- heuristic: $\tau_0 \propto 1/\sqrt{(n)}$
  With n = dimension of the genotype.

Other approaches for real numbers



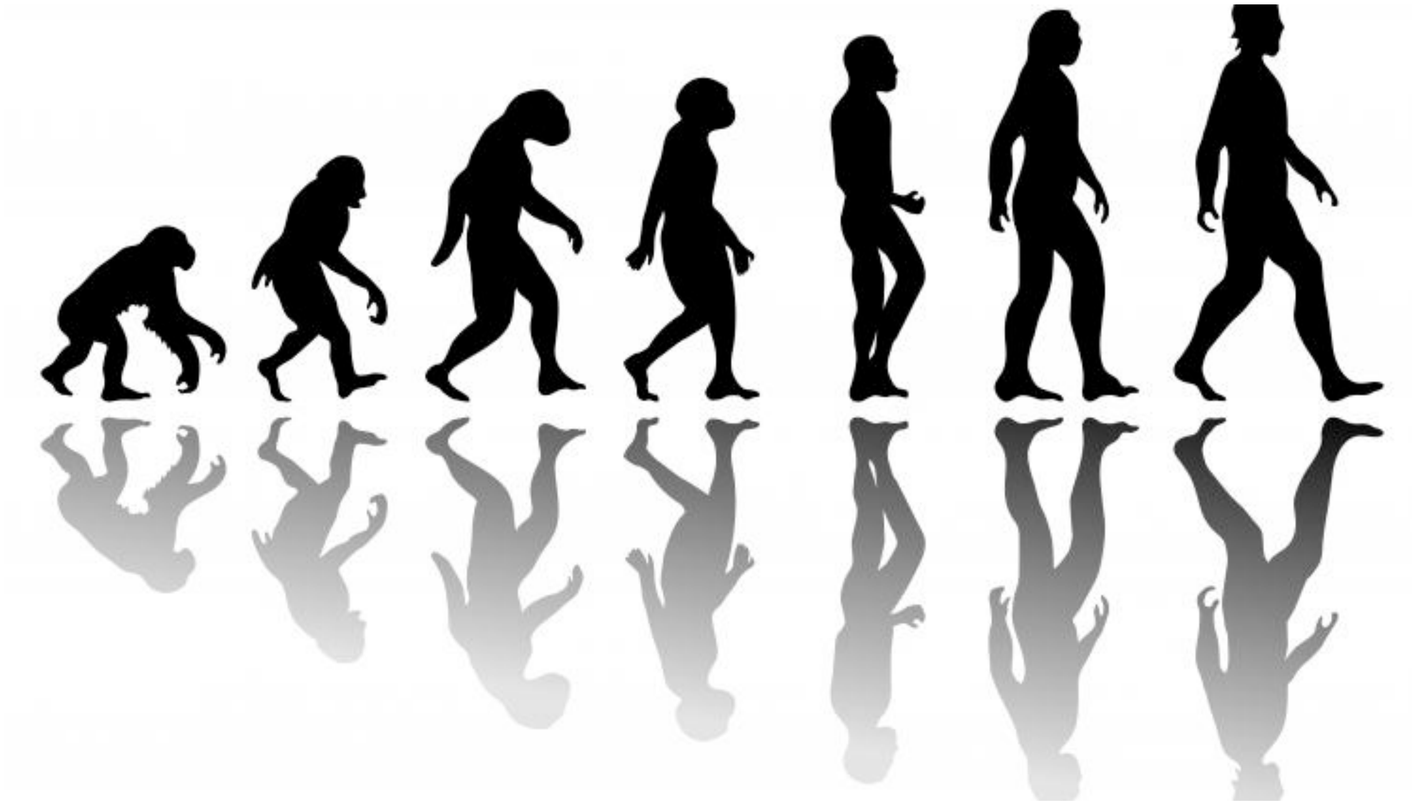
- Other variants of ES:
  - Evolution of a covariance matrix: **CMA-ES algorithm**
    - Very powerful algorithm
    - Wikipedia: https://en.wikipedia.org/wiki/CMA-ES
  - Add Cross-over: BLX-alpha or SBX

- Other variants of Genetic algorithms:
  - Discretise the parameters and then use binary strings
  - We can also use a normal genetic algorithms, **but** with a list of reals instead of bits.
    This requires specific operators.

To be continued...
(Novelty Search)

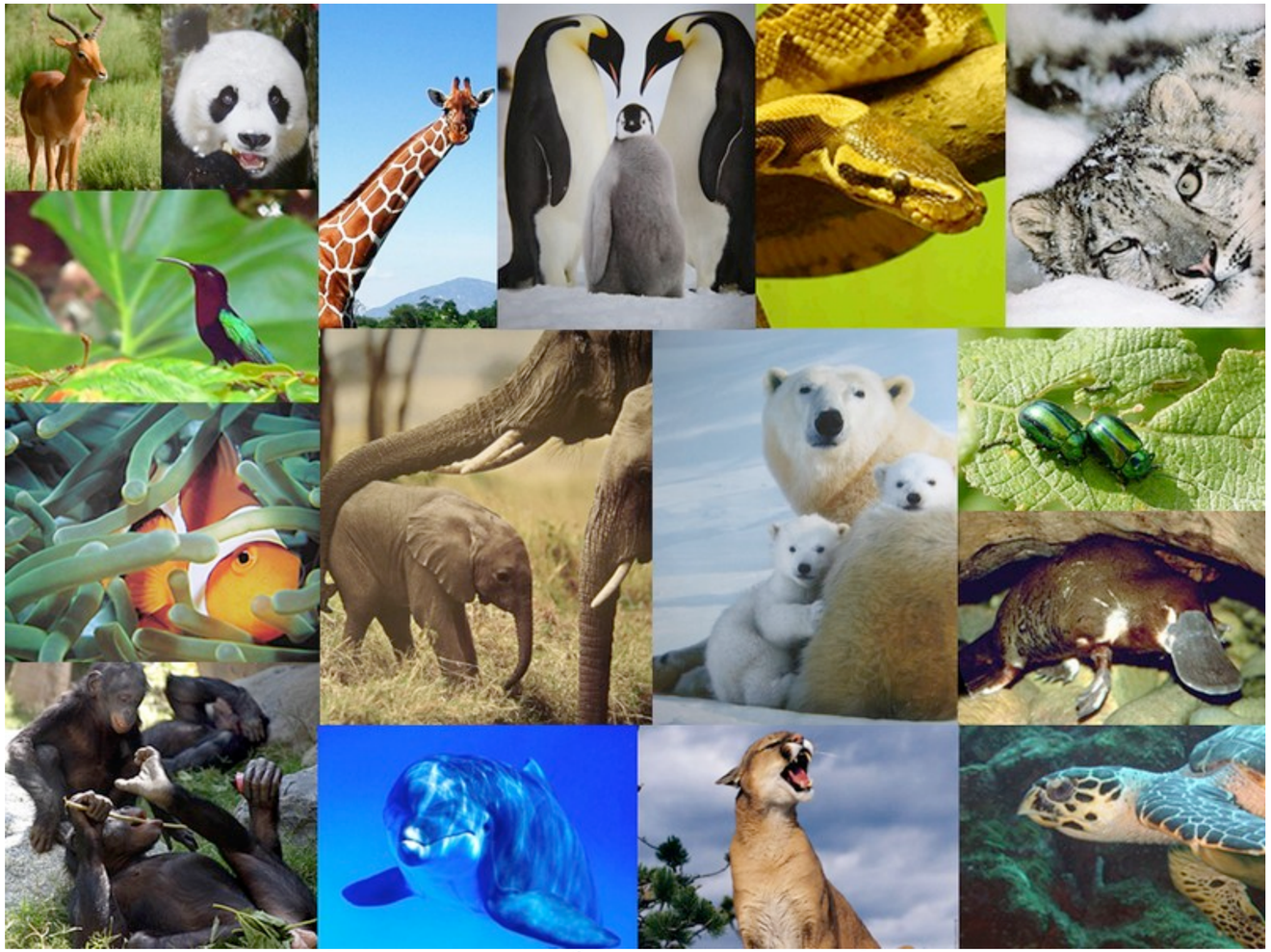
Novelty Search

Natural Evolution as a source of inspiration



Natural evolution finds a lot of brillant solutions

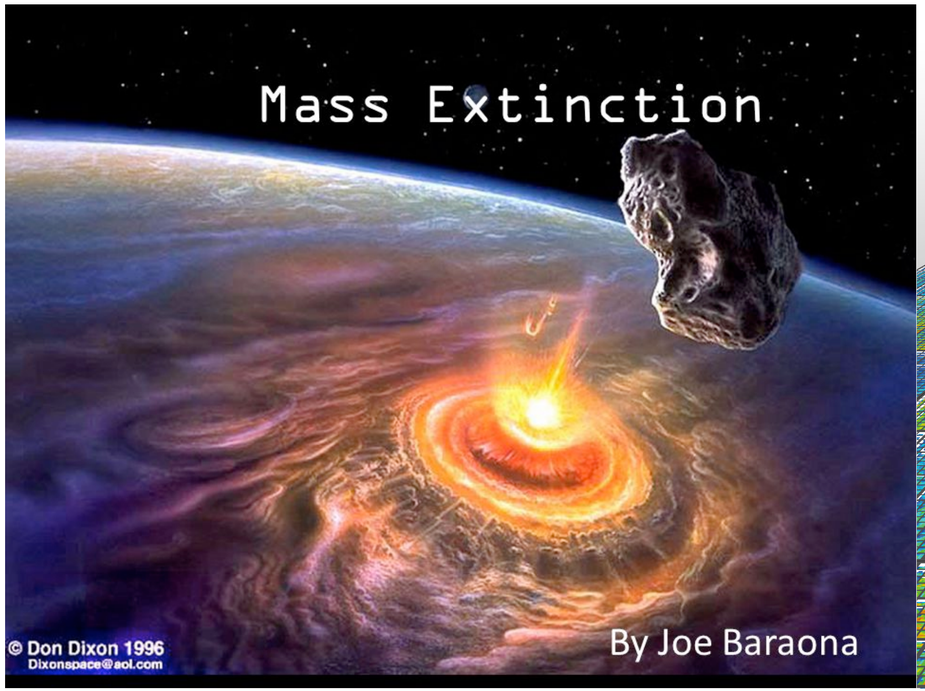
Earth's current species range from 10 millions to 14 millions

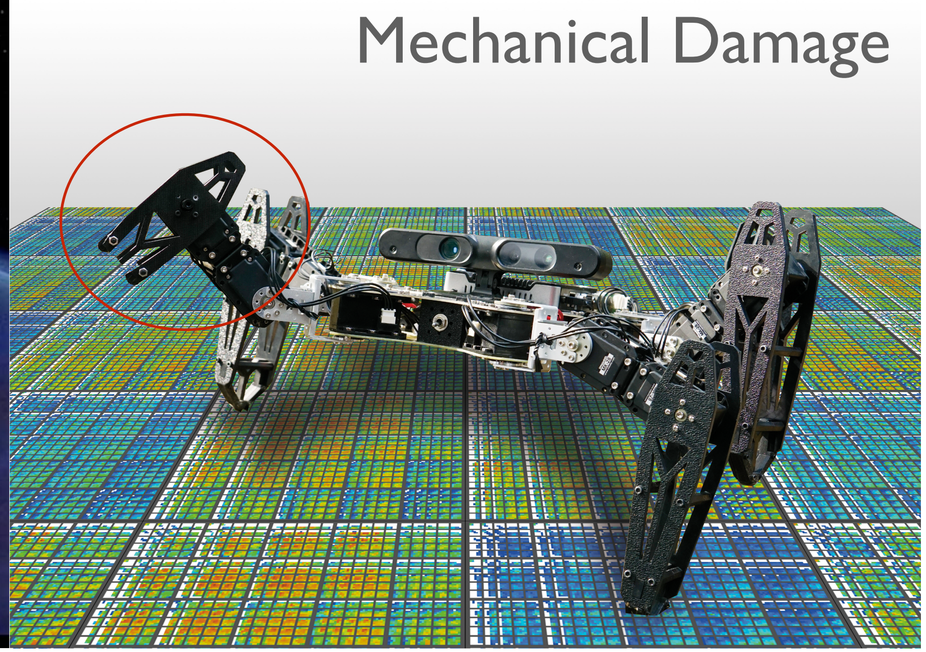


This divergent search aspect of natural evolution is rarely considered in optimisation techniques.

G. Miller; Scott Spoolman (2012). Environmental Science - Biodiversity Is a Crucial Part of the Earth's Natural Capital

Learning large diversity of solutions



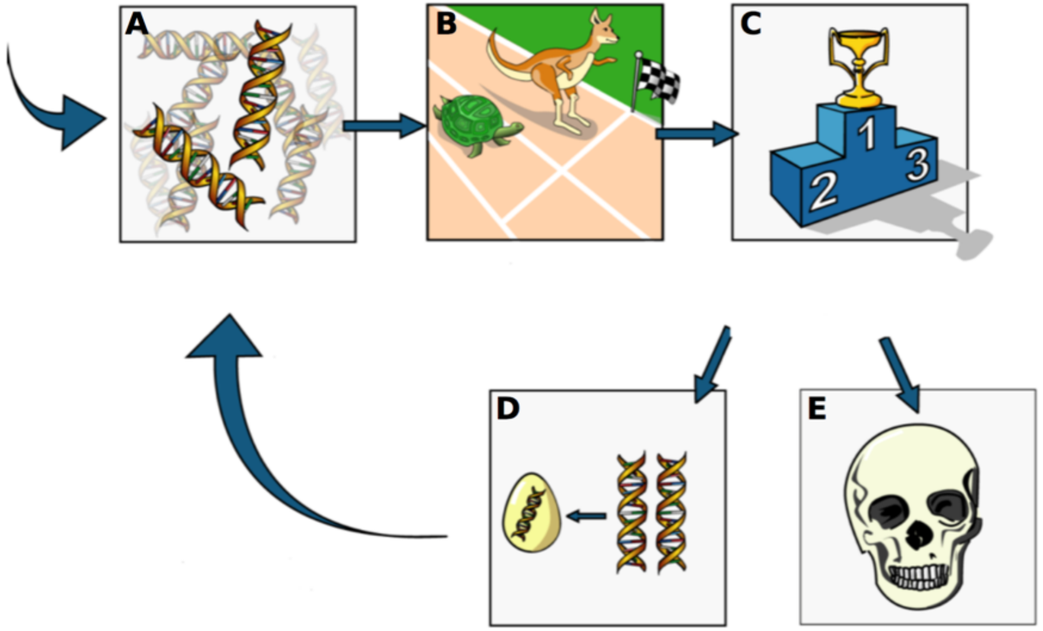


It provides a set of alternatives in case something wrong happen.

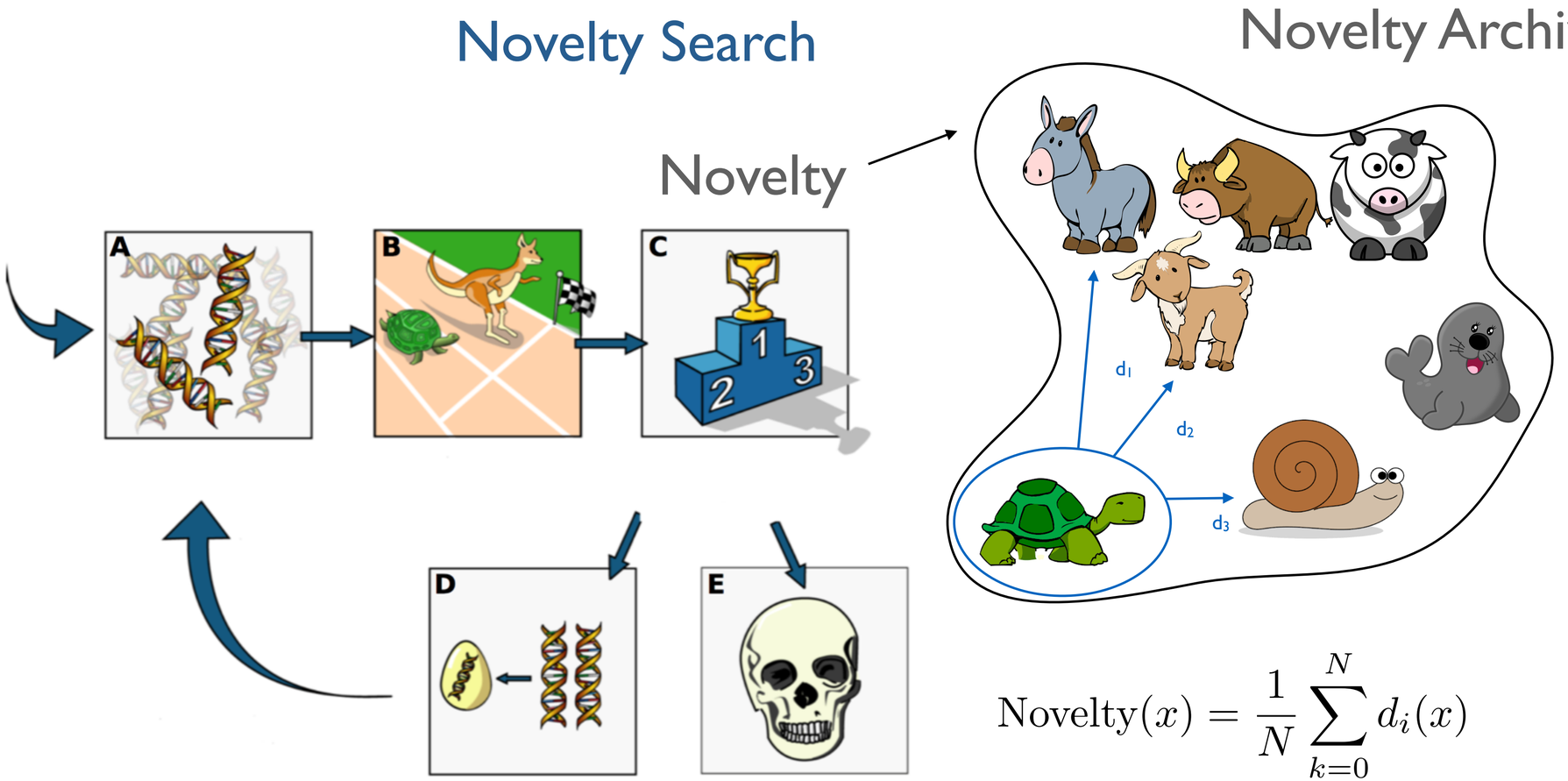
Novelty Search

Traditional Algorithm:

Performance



Novelty Search



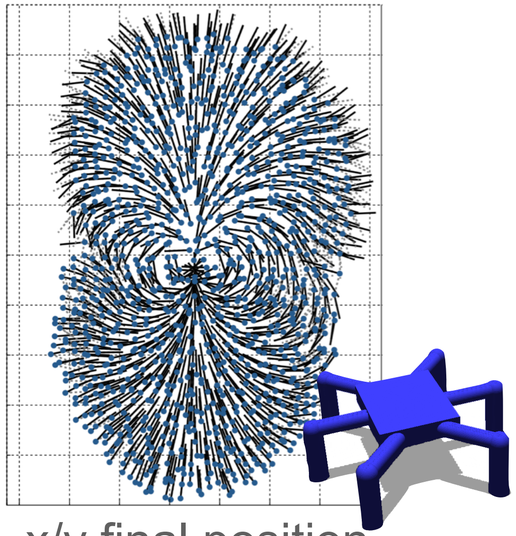
$$\text{Novelty}(x) = \frac{1}{N}\sum_{k=0}^{N} d_i(x)$$

Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: evolution through the search for novelty alone. Evolutionary Computation.

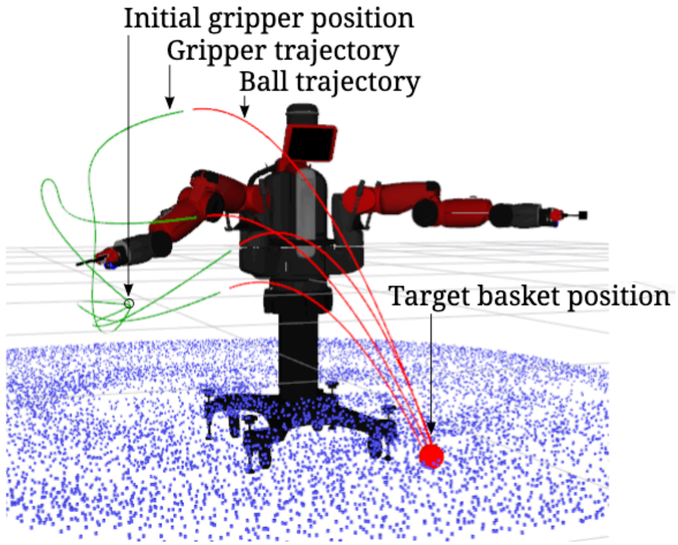
Novelty Search: Behavioural Descriptor

Important concept!

- The **behavioural descriptor** characterises certain aspects of the solutions:
  - It defines the “types of solutions”
- The behavioural descriptor is not necessarily linked to the task (linked to the concept of alignment between the fitness and the descriptor)
- Several solutions are likely to have the same behavioural descriptor, but with different fitness values.



x/y final position



Key-points of the trajectory

Kim, Coninx, Doncieux (2019). From exploration to control: learning object manipulation skills through novelty search and local adaptation. arXiv preprint arXiv:1901.00811.

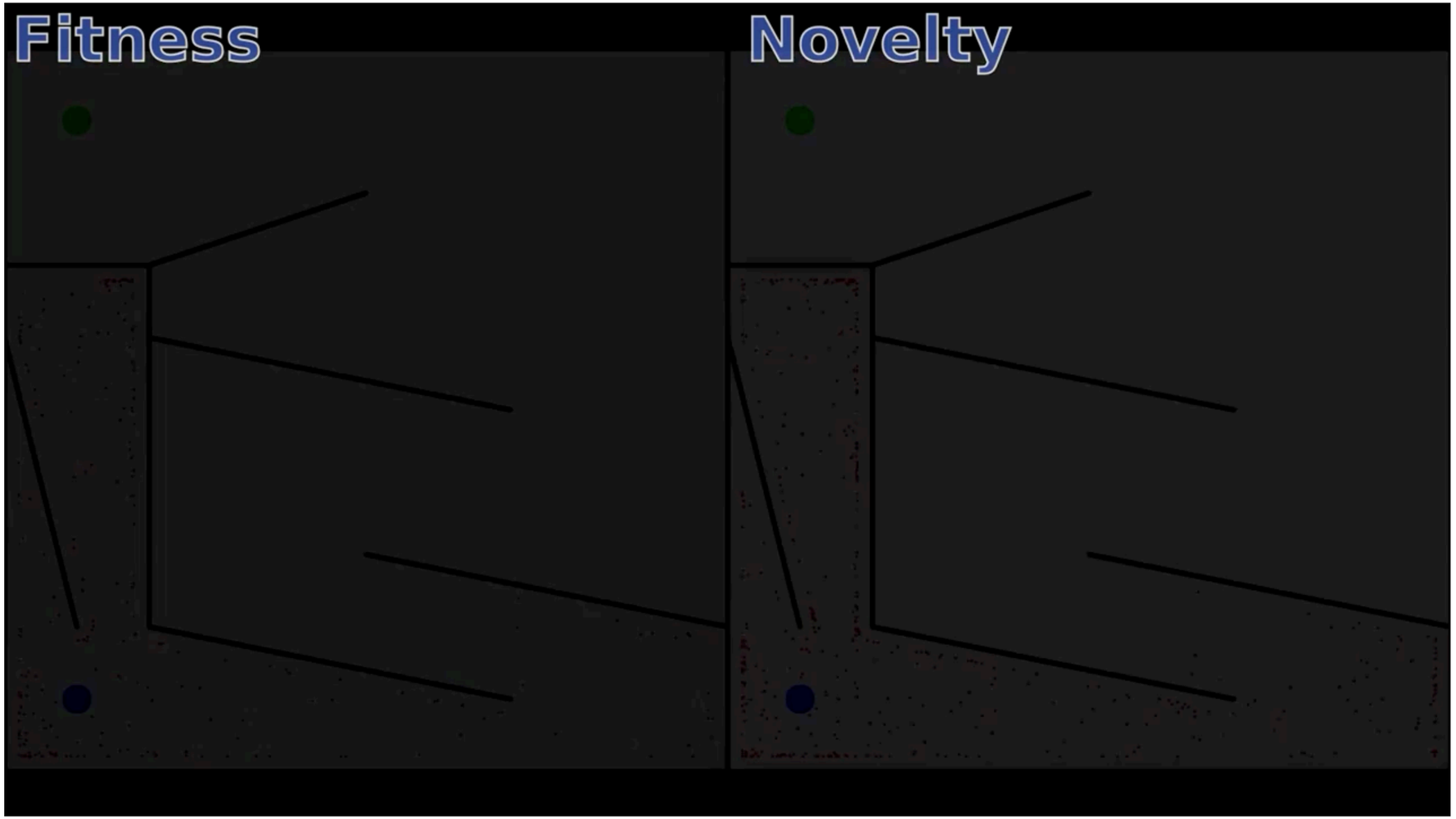
Novelty Search

- Instead of optimising the quality of a solution
- NS optimises the novelty of the solution.

Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: evolution through the search for novelty alone. Evolutionary Computation.

Novelty Search

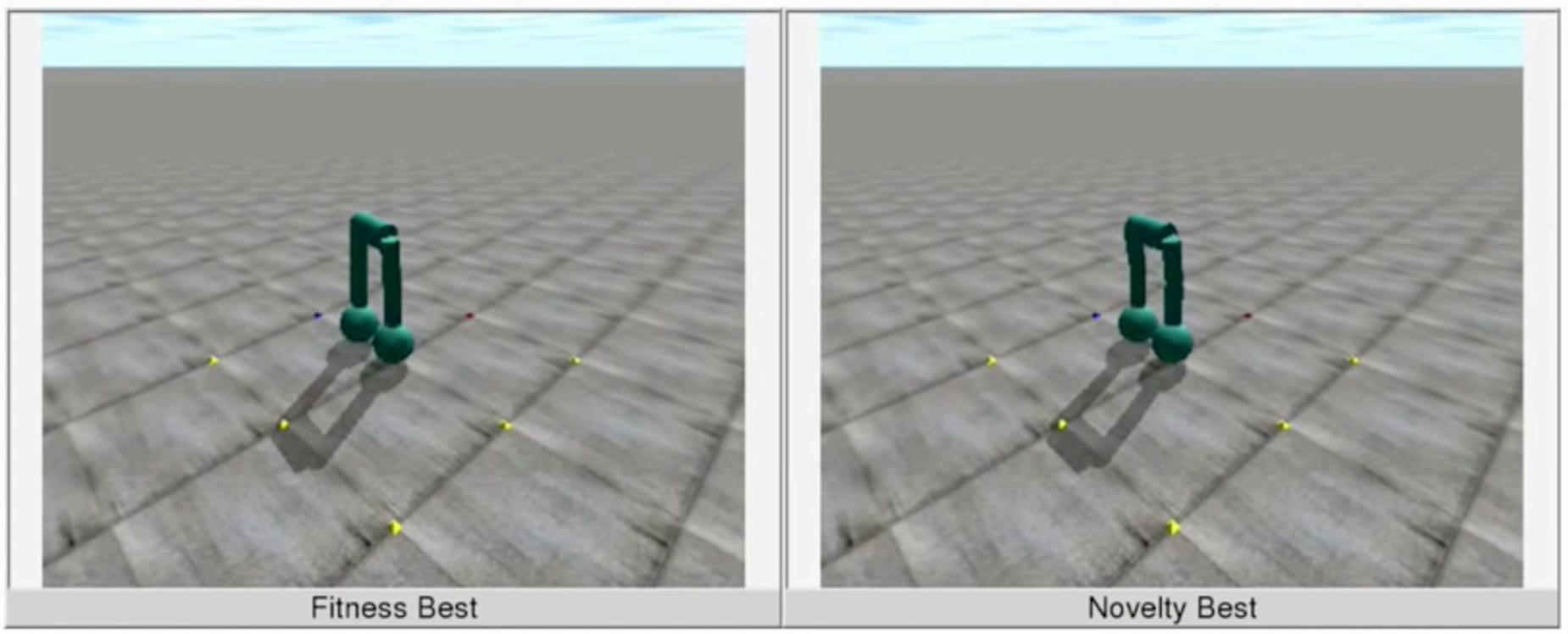
- Instead of optimising the quality of a solution
- NS optimises the novelty of the solution.



Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: evolution through the search for novelty alone. Evolutionary Computation.

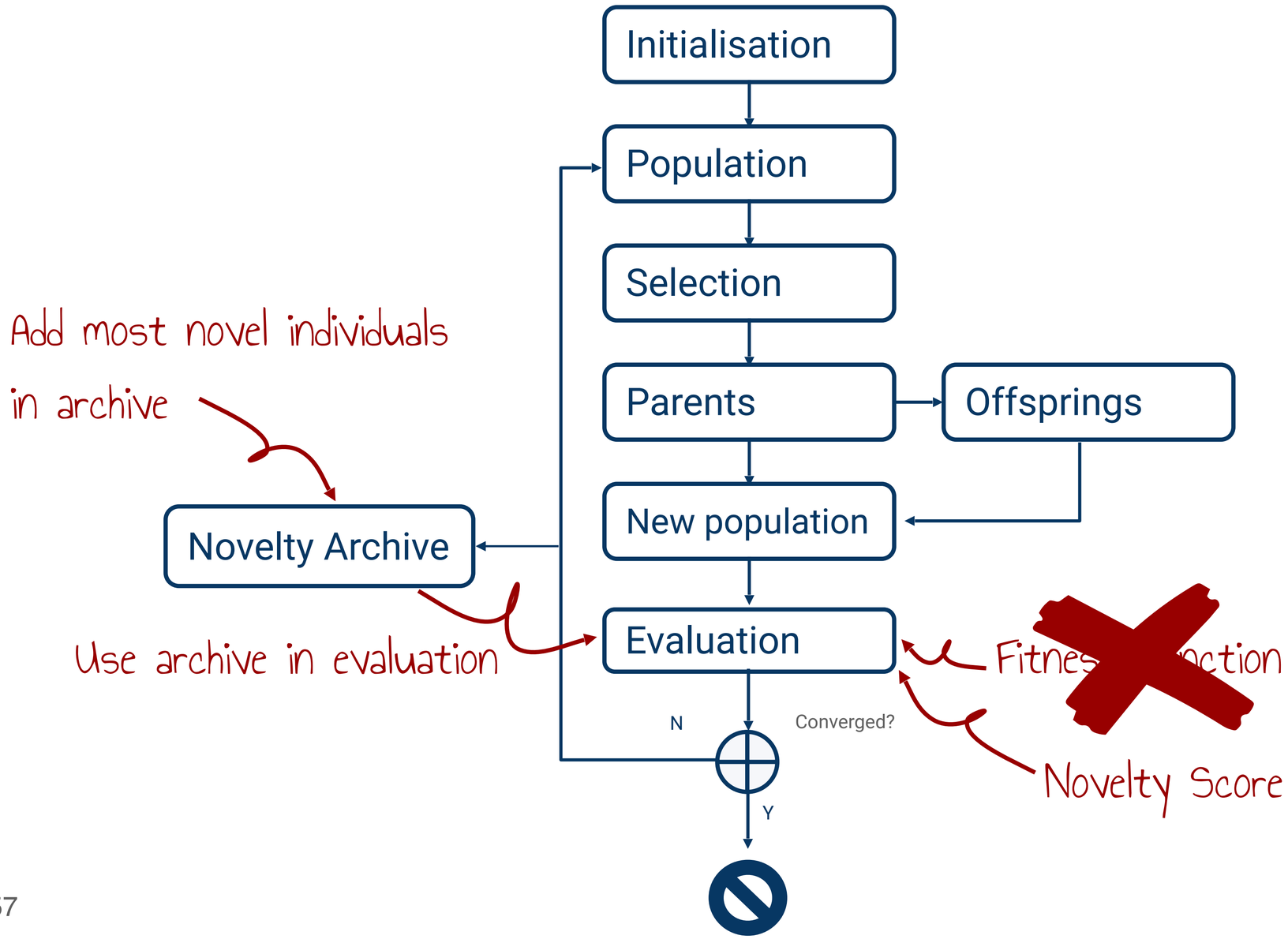
Novelty Search

- Instead of optimising the quality of a solution
- NS optimises the novelty of the solution.



Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: evolution through the search for novelty alone. Evolutionary Computation.

Novelty Search

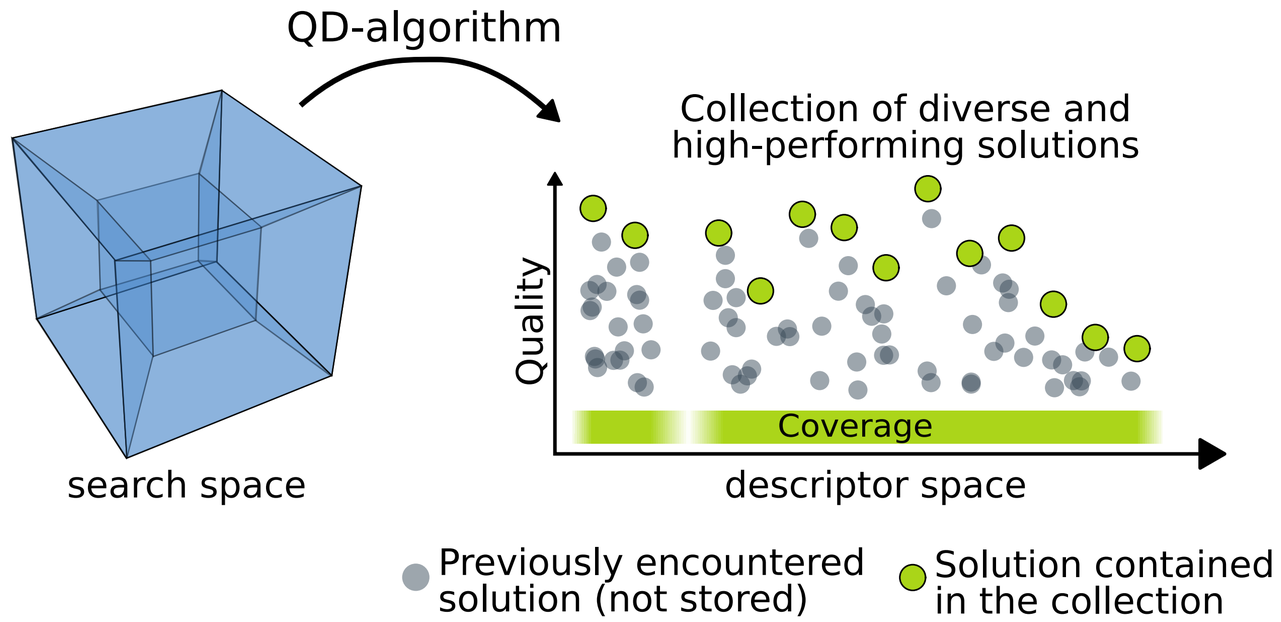


To be continued...
(Quality-Diversity optimisation)

Quality-Diversity optimisation

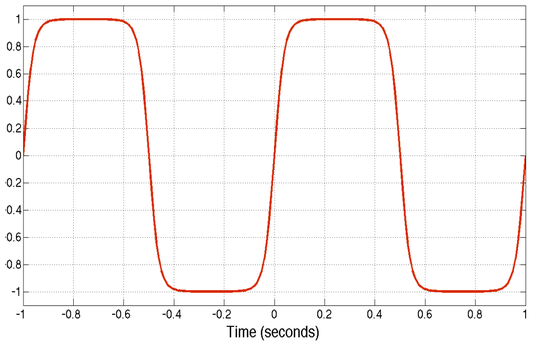
QD algorithms: How does it work?

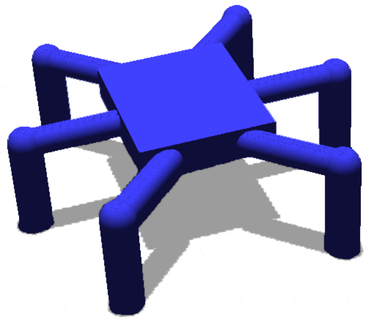
Objective: Learning in a single optimisation process
a large collection of **diverse** and **high-performing** solutions



Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.
Pugh, Soros, Stanley. (2016). Quality diversity: A new frontier for evolutionary computation. Frontiers in Robotics and AI.

Application example: Learning different ways to walk





Genotype:

- 2 degrees of freedom per leg
- Amplitude, Phase, Duty Cycle
- **36 dimensions**

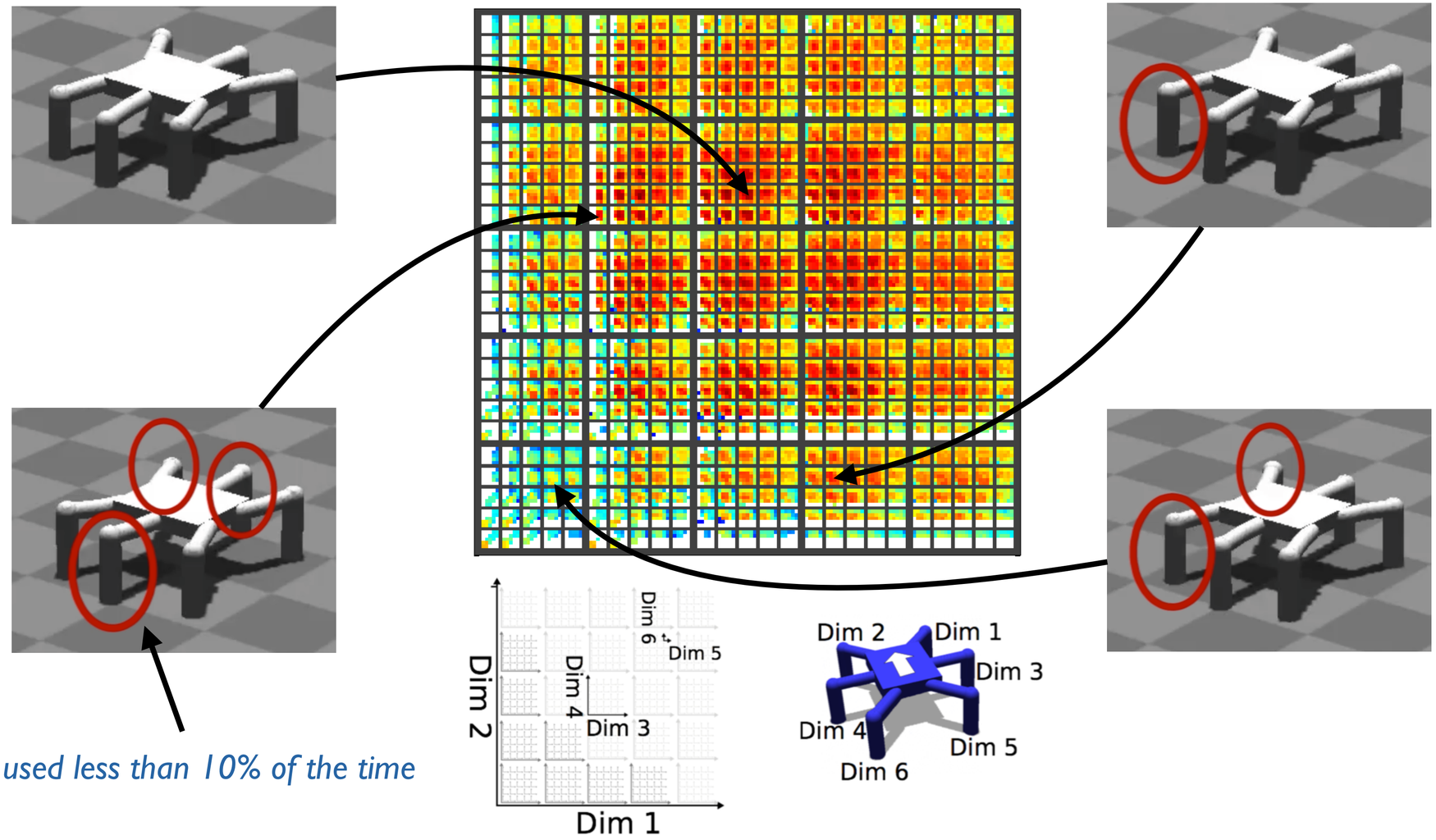
Behavioral Descriptor:

- The proportion of time that each leg touches the ground
- Discrete (0%, 25%, 50%, 75%, 100%)
- **6 dimensions**

Fitness function: $f(x) = \frac{pos_{front}(robot(x), t = 5s)}{5}$

40 million evaluations

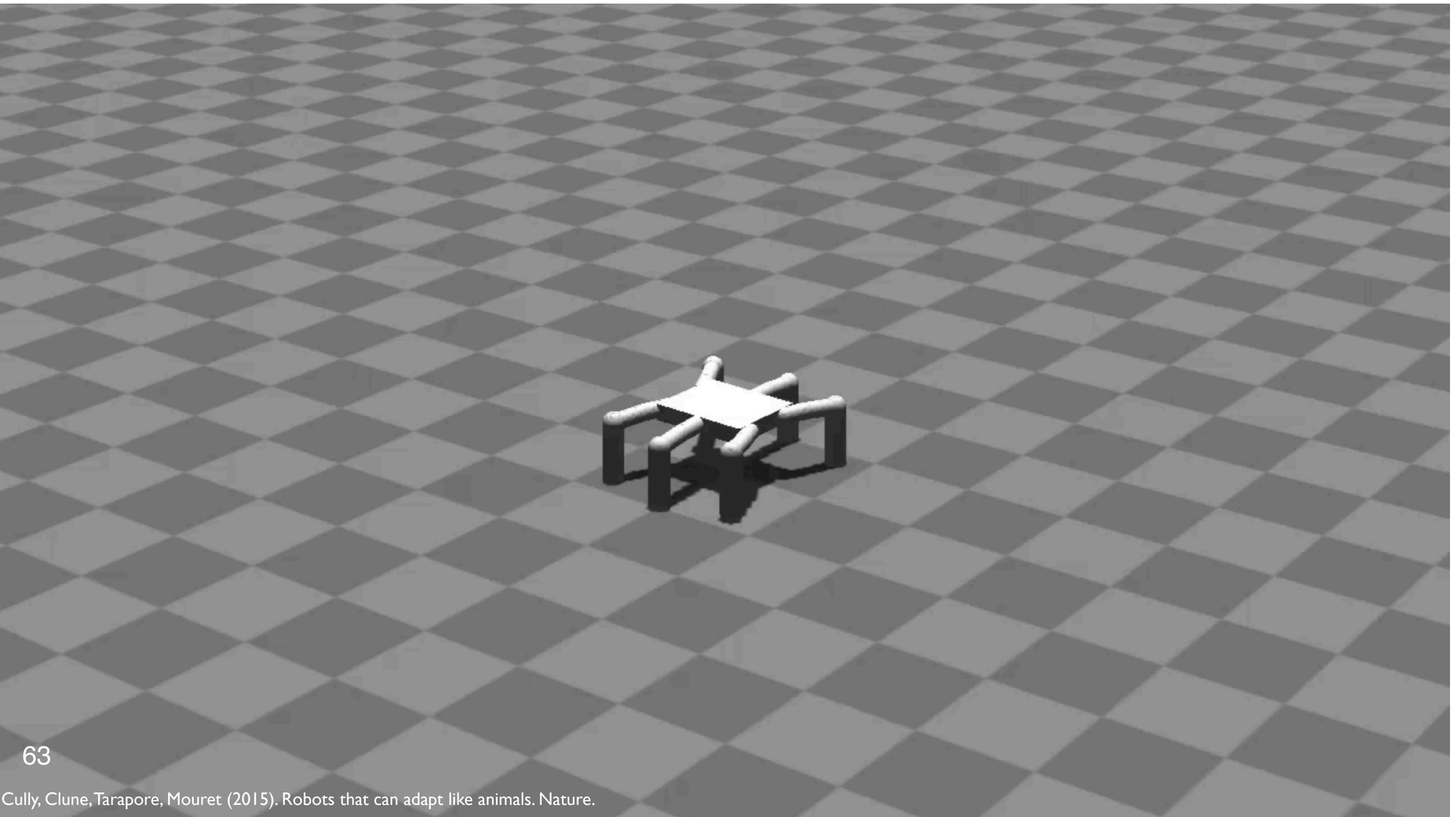
Result: 13 000 different behaviours



Leg used less than 10% of the time

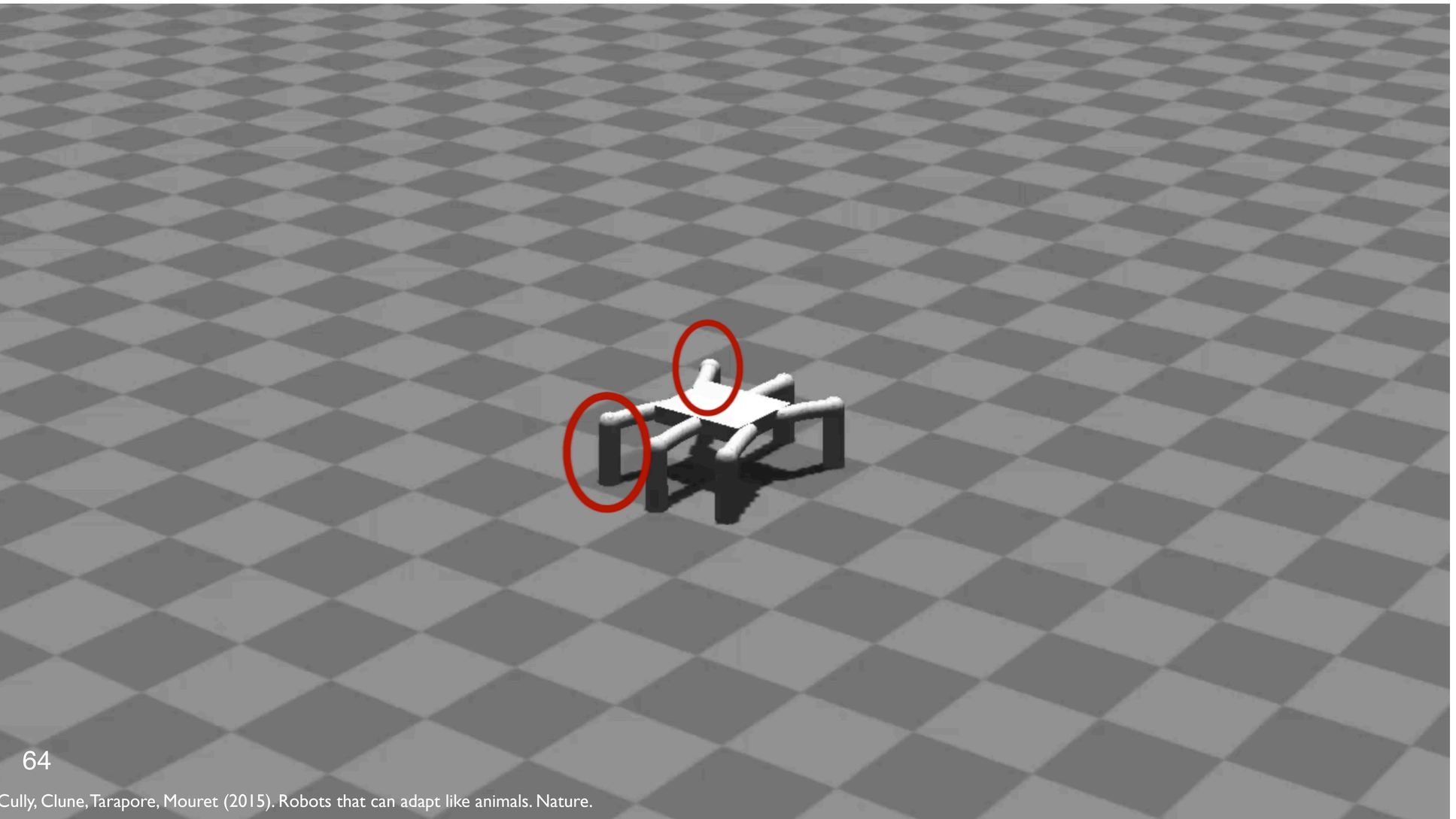
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Hexapod Gait



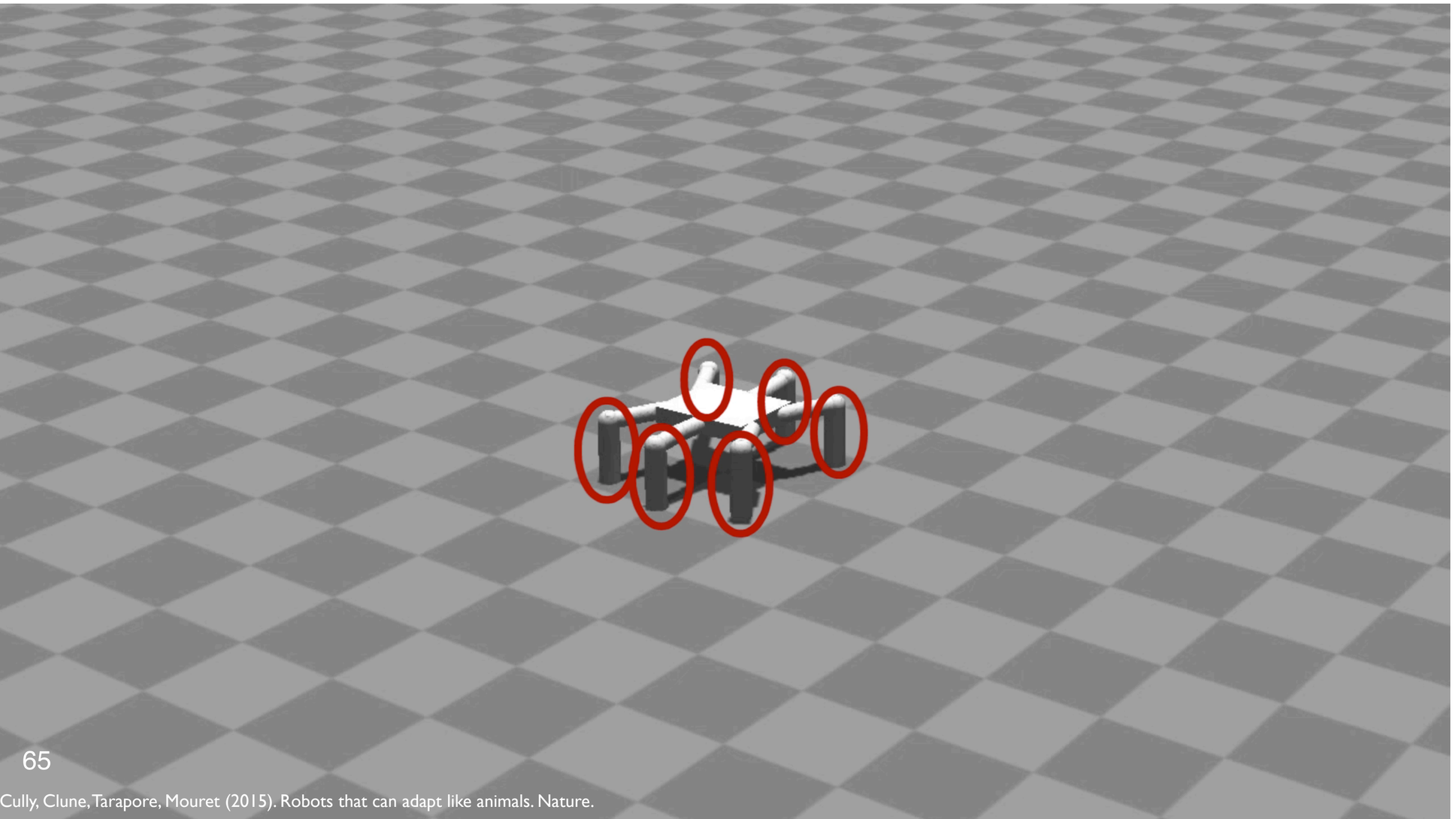
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Quadrupedal Gait



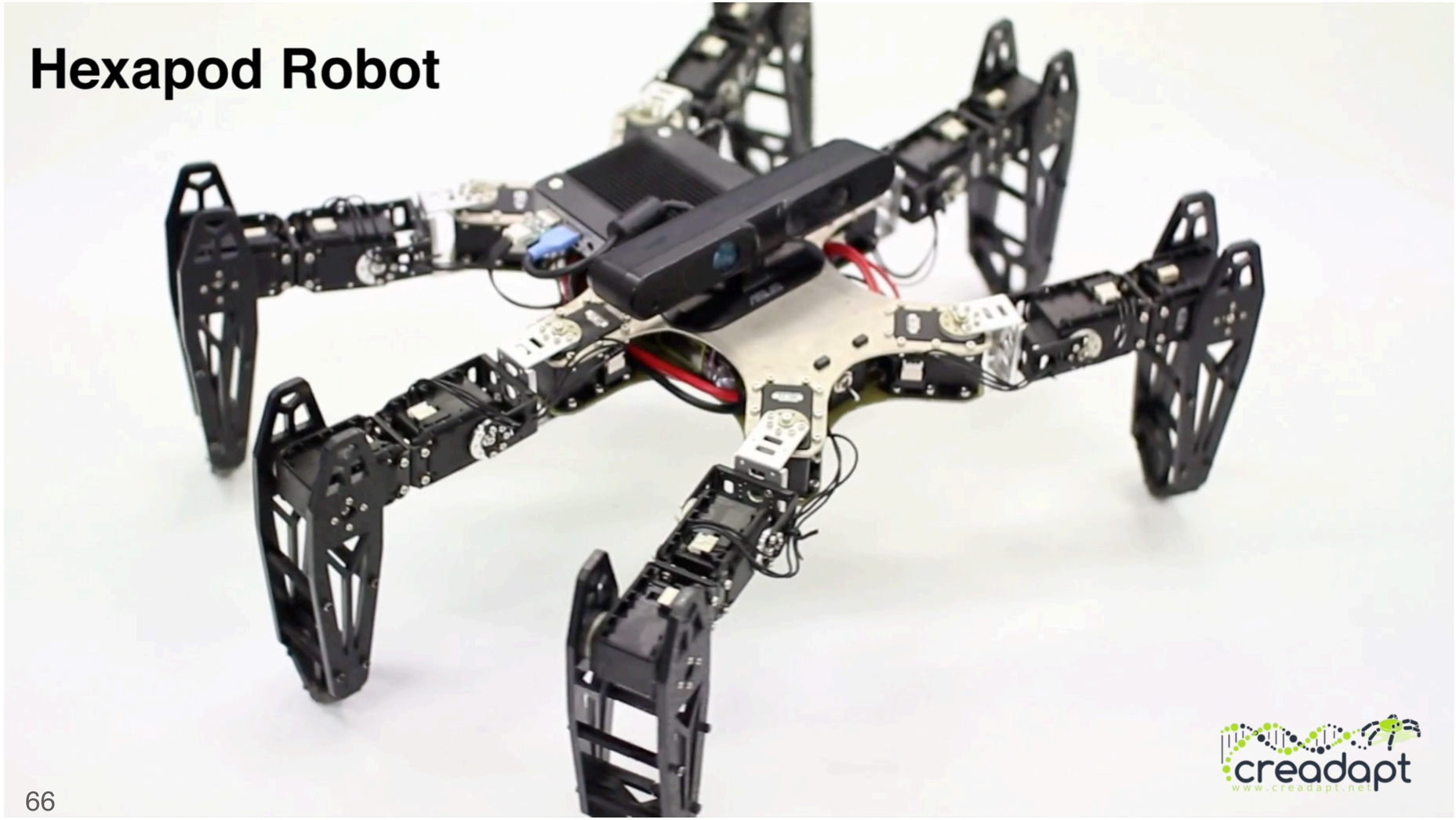
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

« Creative » Gait



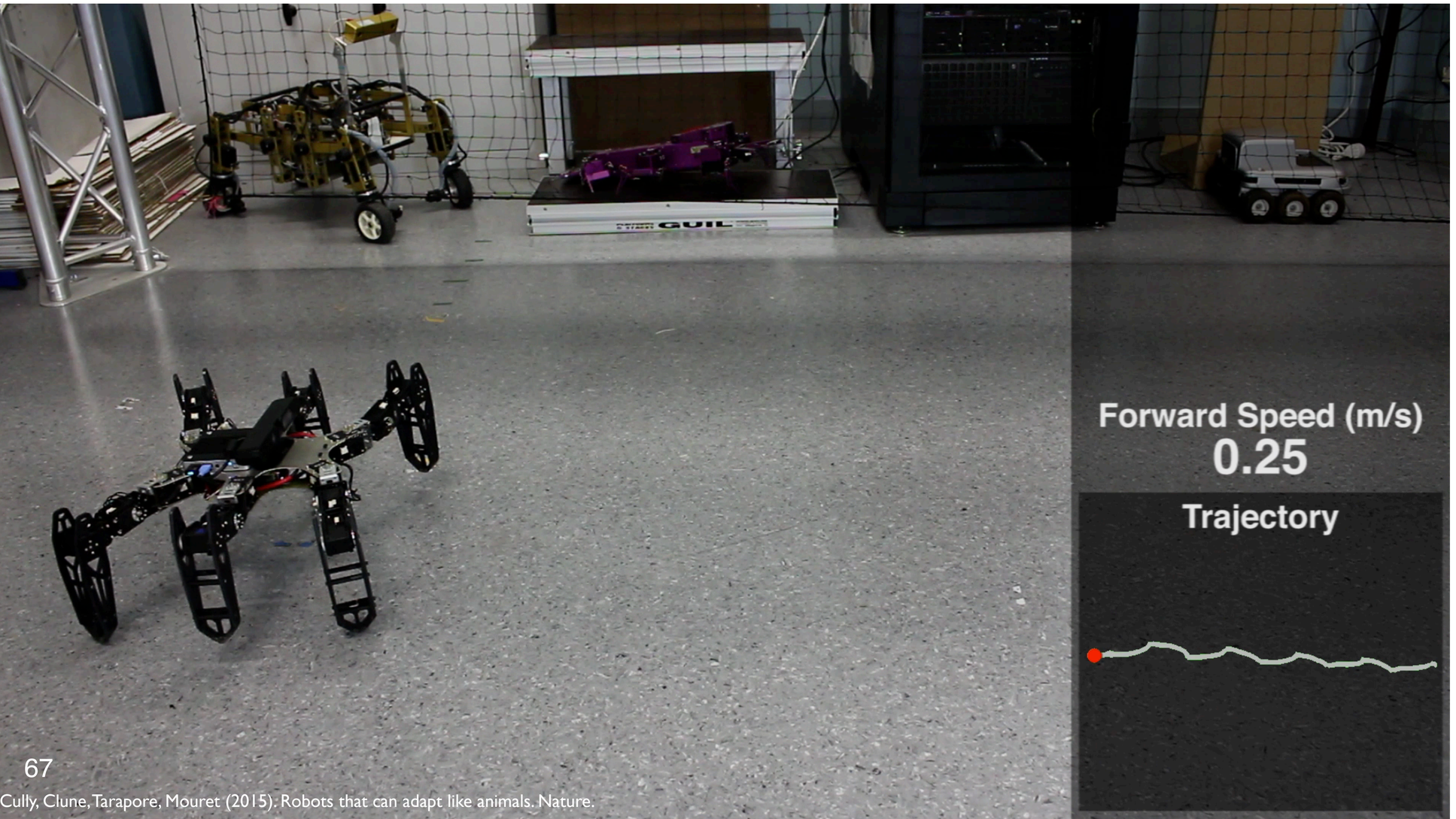
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Application on a robot



Classic Hexapod Gait

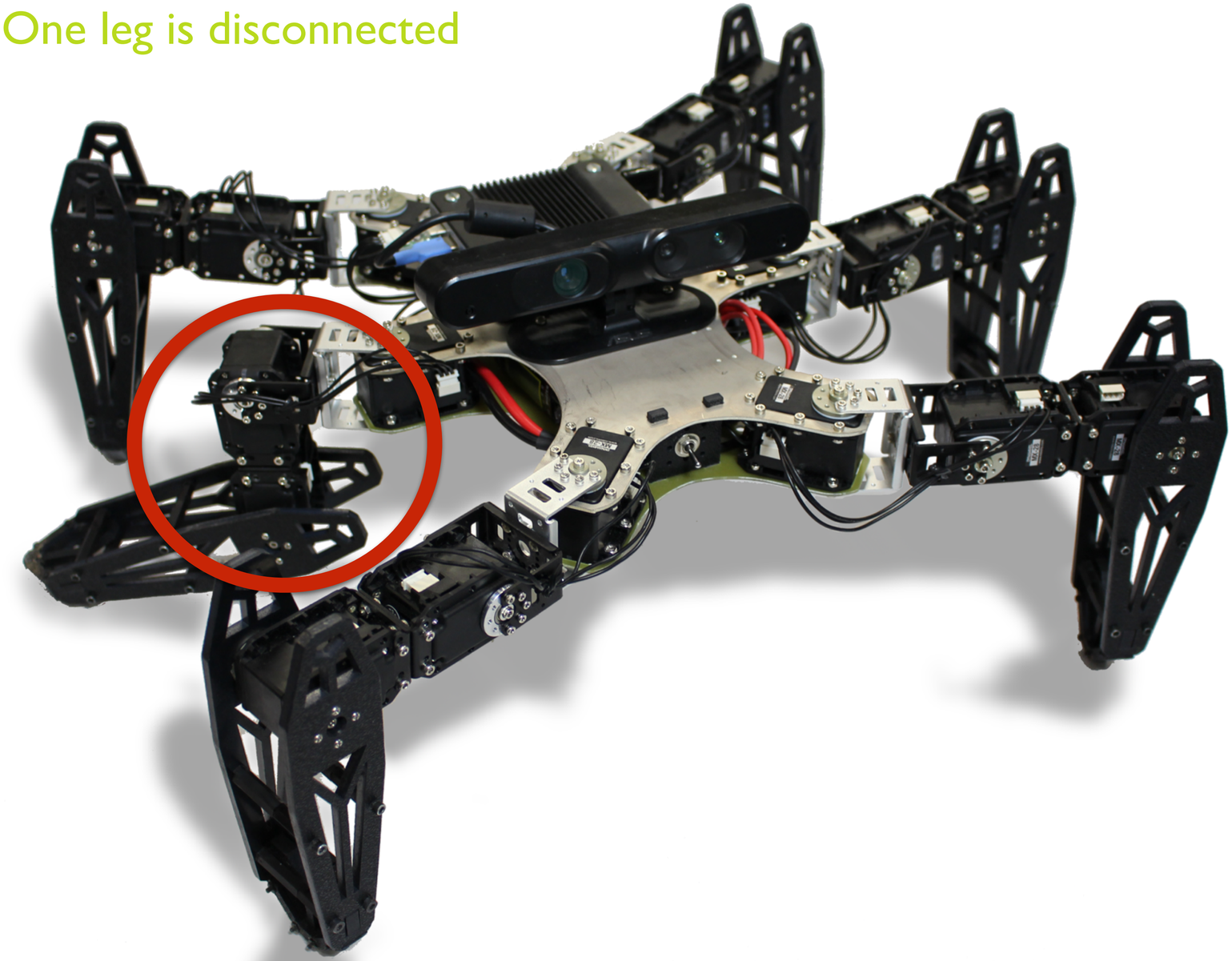
Open-loop controller - Tripod Gait



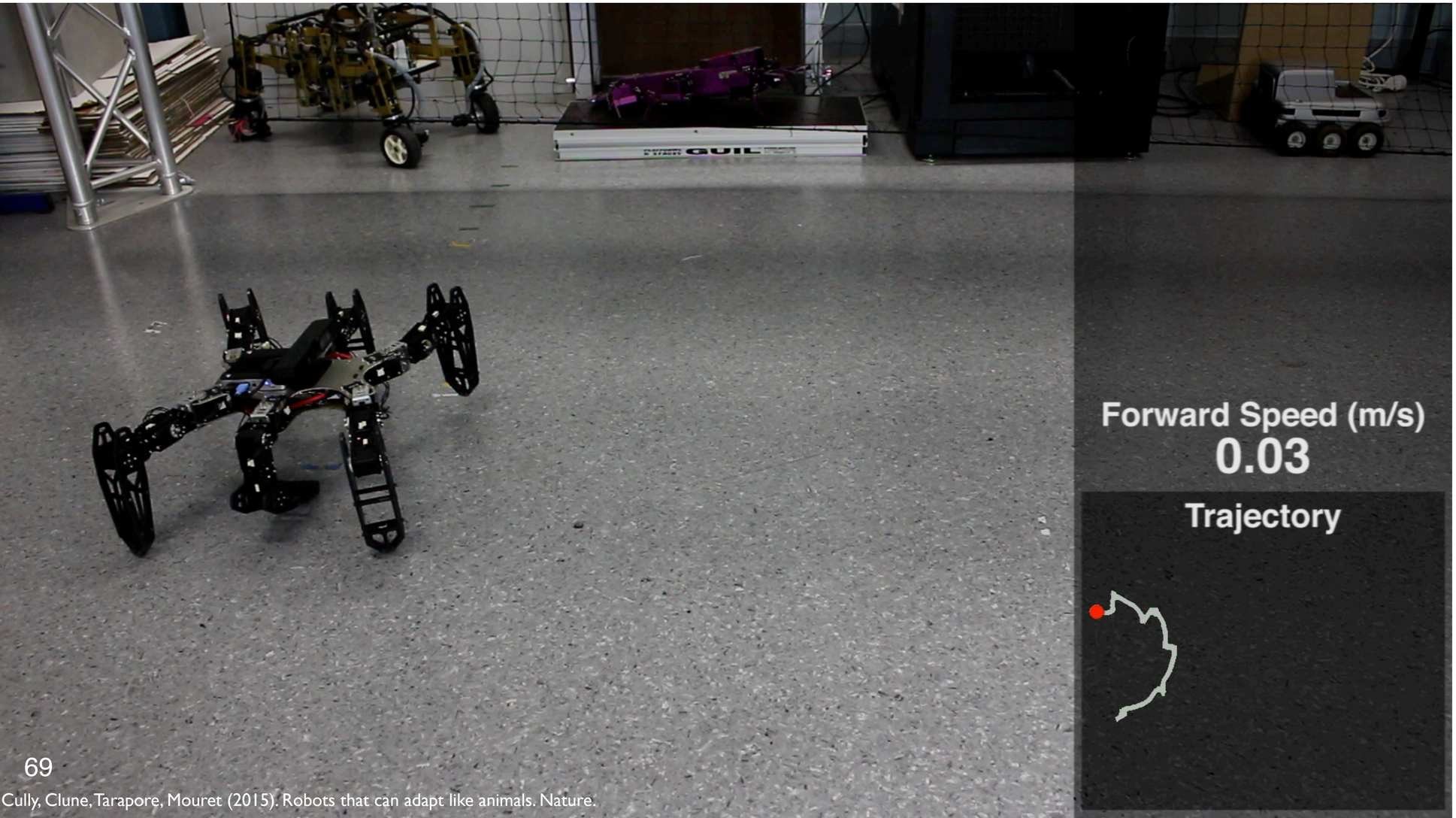
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

A damage occurs ...

One leg is disconnected

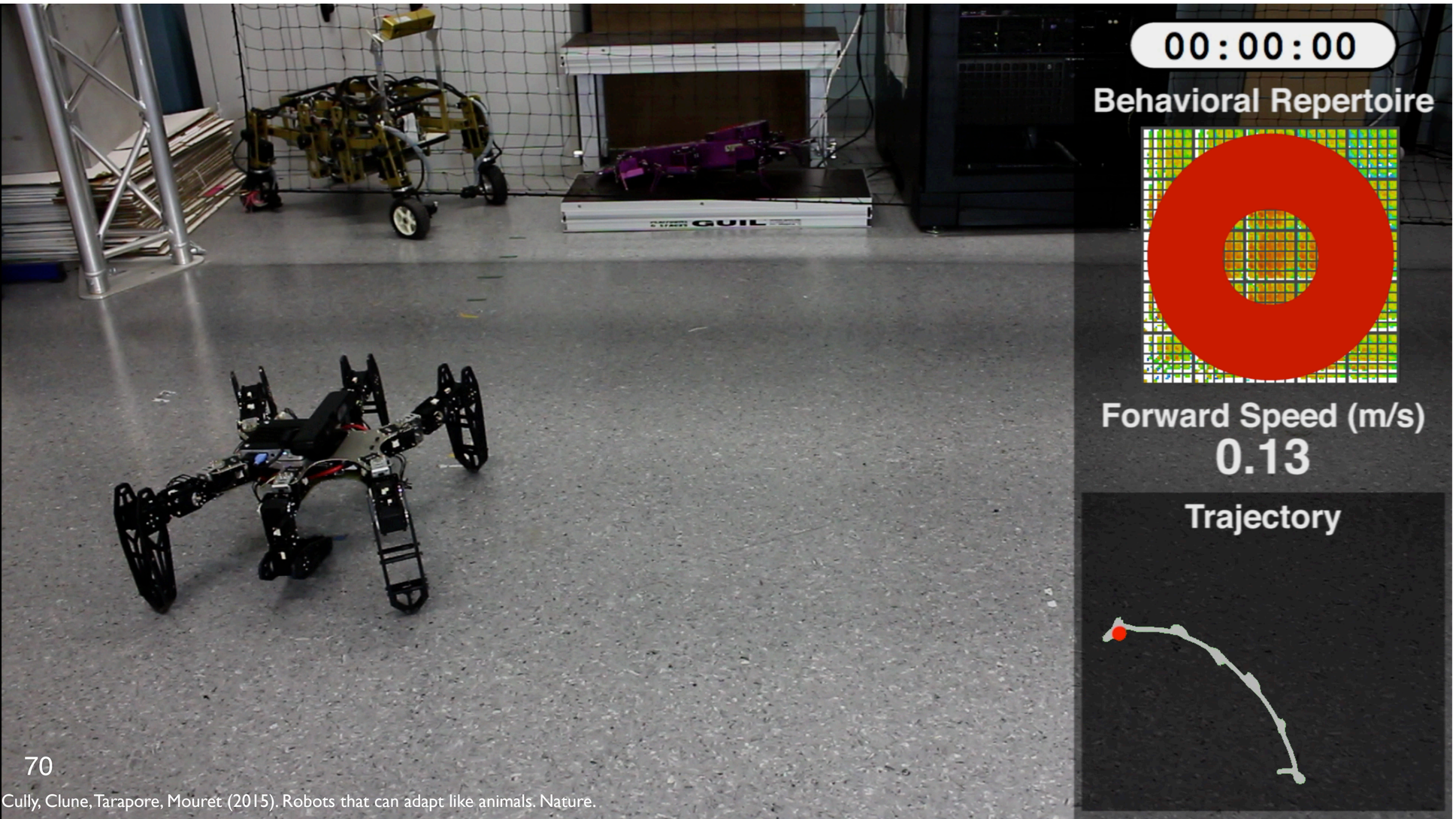


The behavior is seriously affected



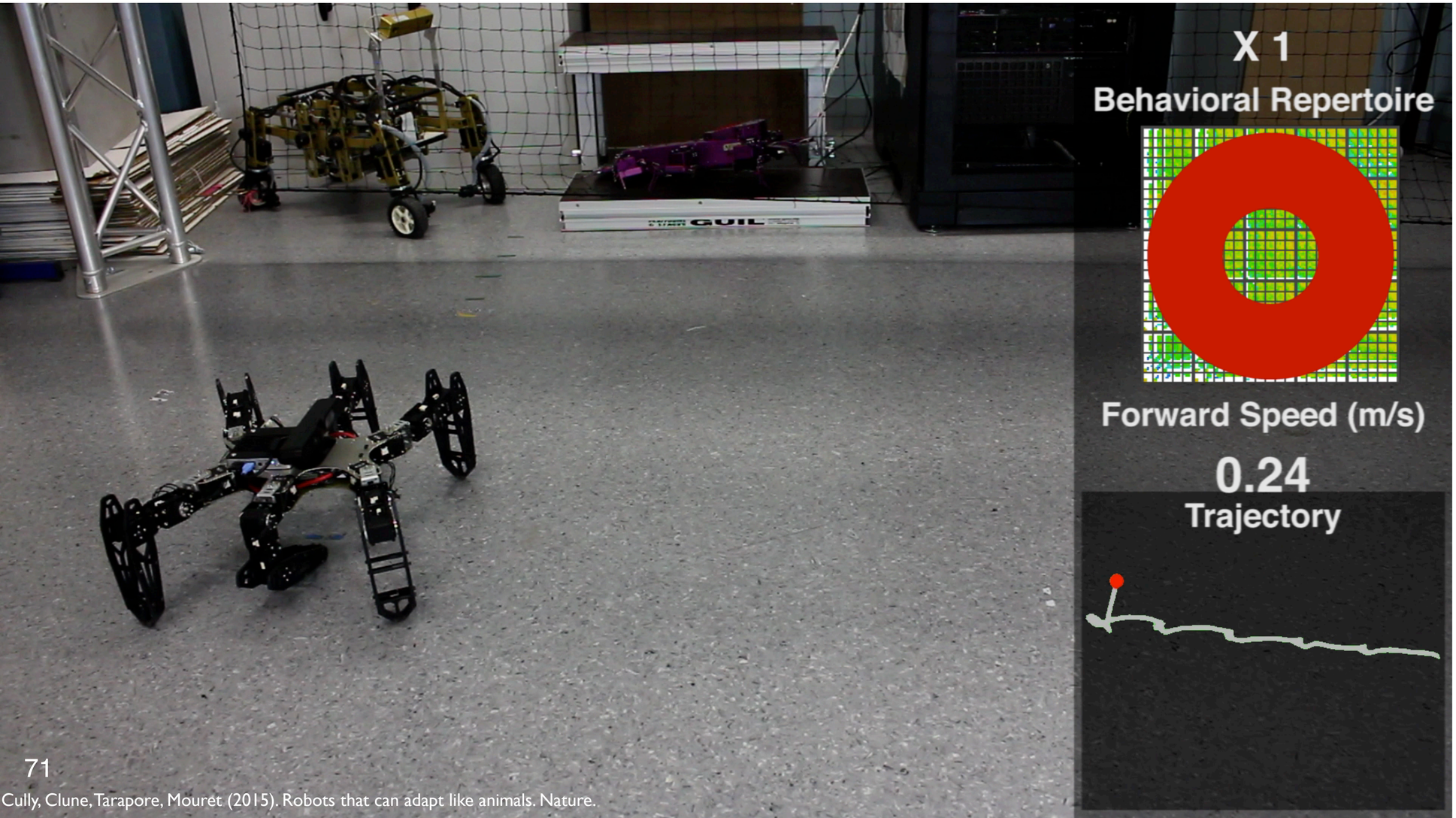
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Learning process



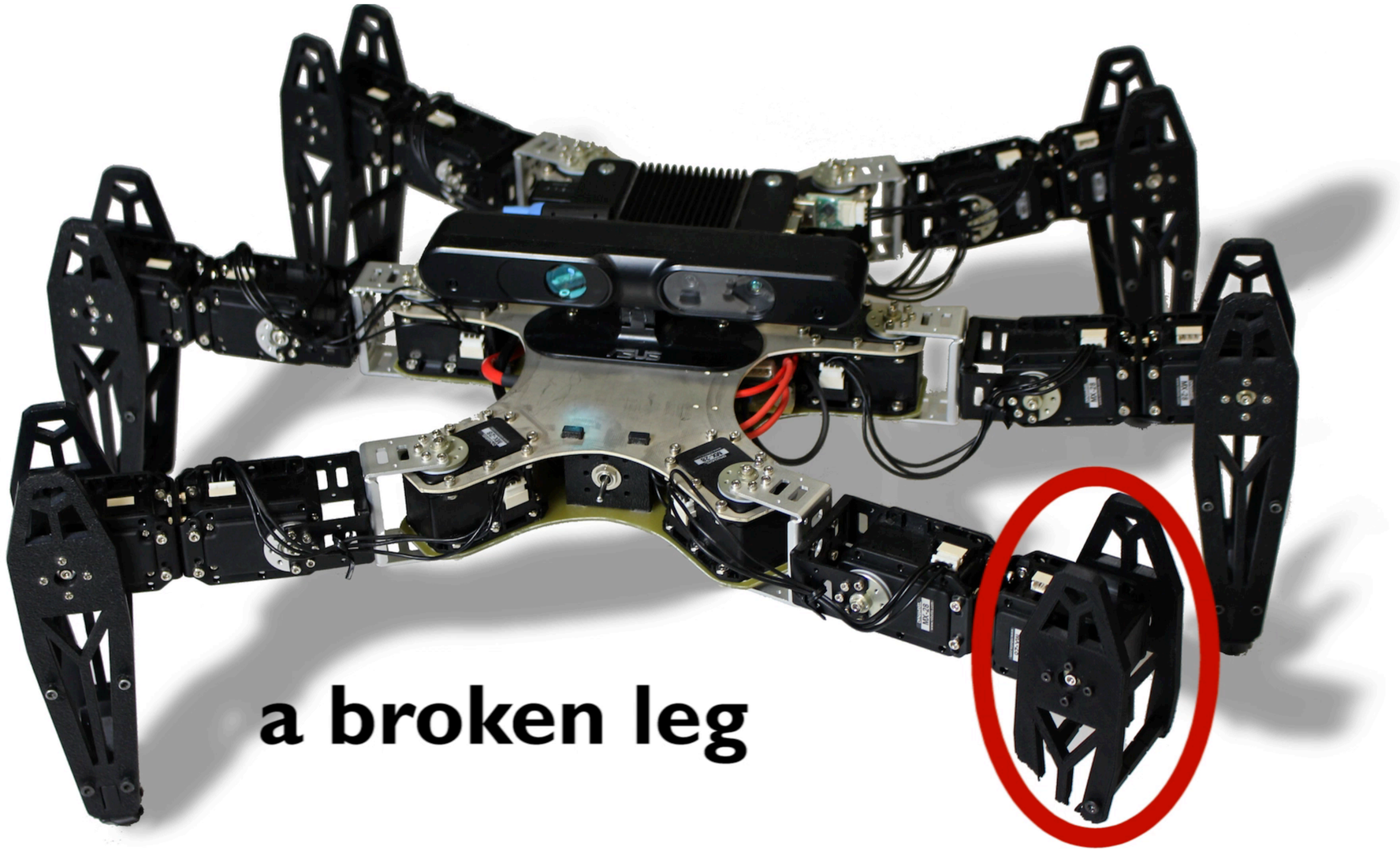
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Result after 40 seconds

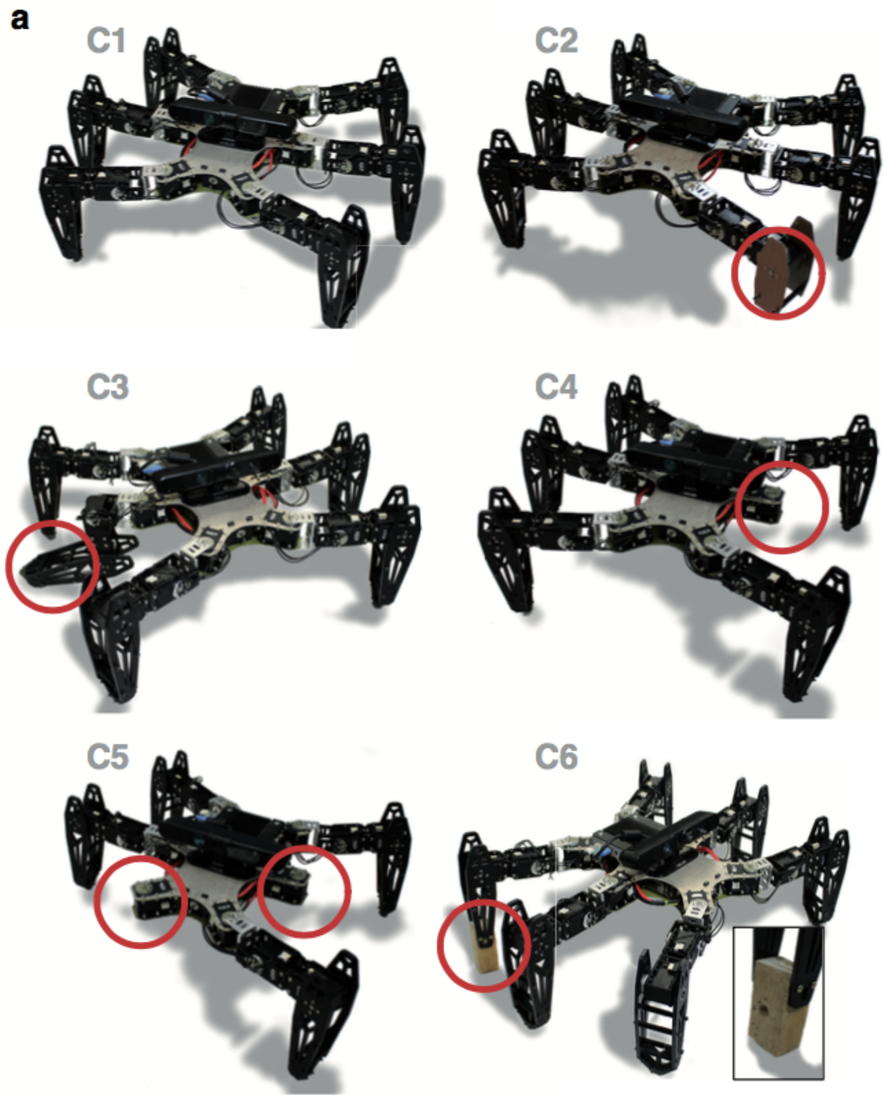


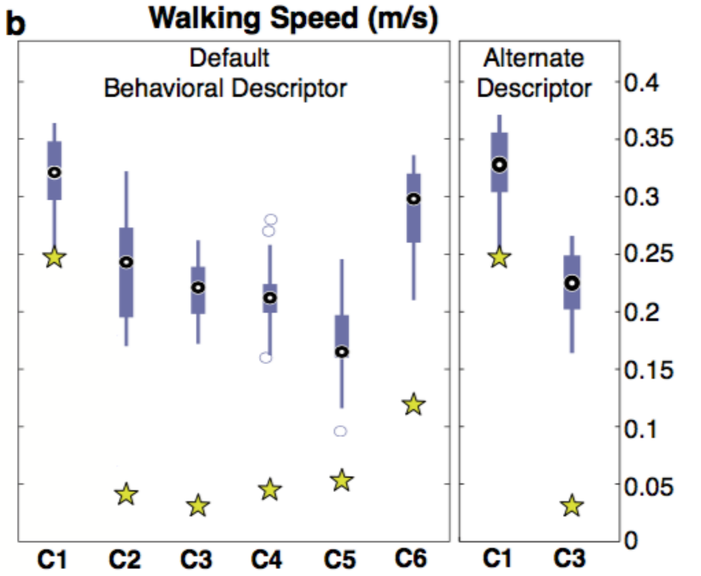
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

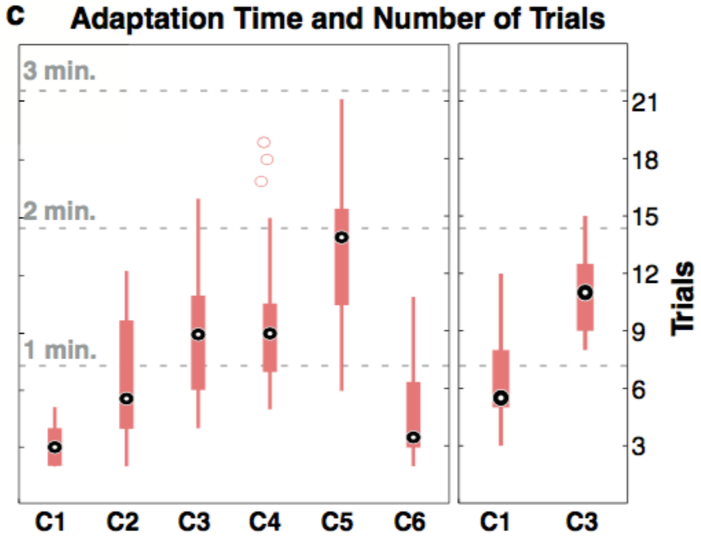
Other examples



All tested scenarios







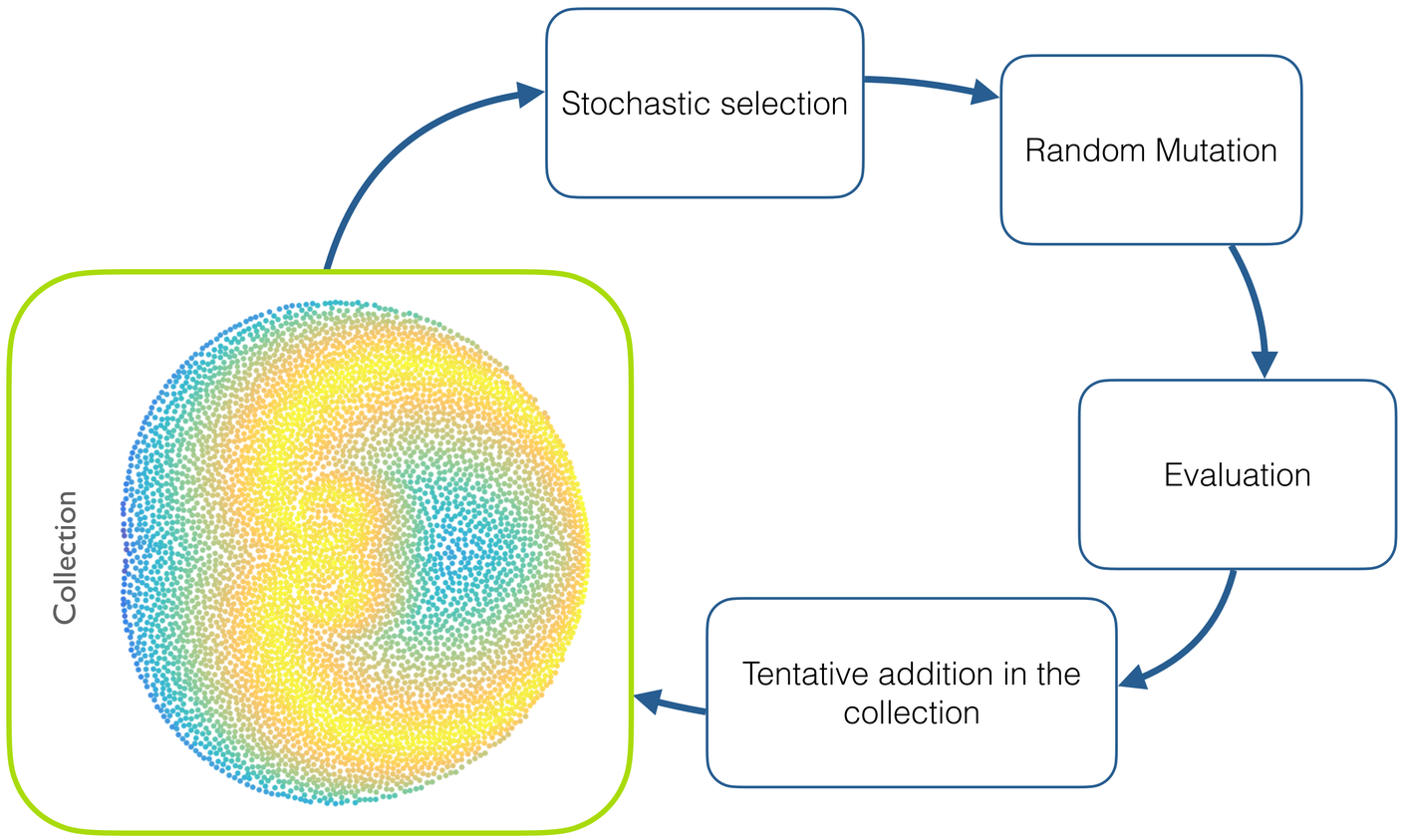
nb replicates:
40 / condition

Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

QD algorithms: How does it work?

Generic pseudo code

- **Most QD algorithms follow the same few steps:**



Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.

QD algorithms: How does it work?

Behavioural Descriptor and Fitness functions

- To build a collection of high-performing and diverse solutions, we need to:
  - Measure the performance of solutions
  - Distinguish different types of solutions
- For that, we use:
  - A **fitness function**, like in most evolutionary algorithms
  - A **behavioural descriptor** (also called behavioural characterisation)

Lehman, Stanley (2010). Abandoning Objectives: Evolution Through the Search for Novelty Alone. Evolutionary Computation.

Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.

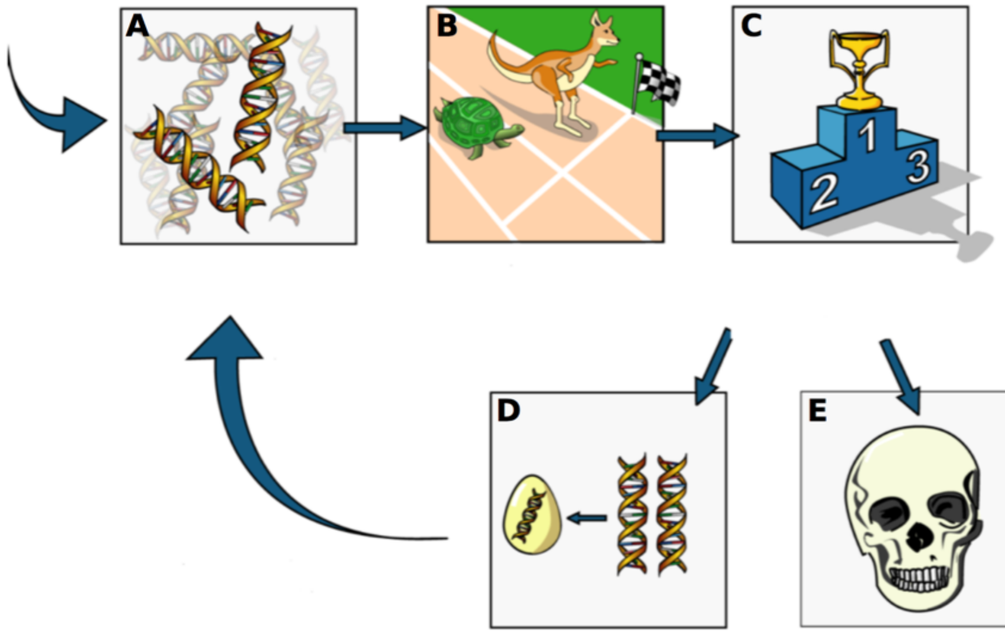
Going further: Novelty Search with Local Competition [1].

Is a QD algorithm when used with an archive management system[2].

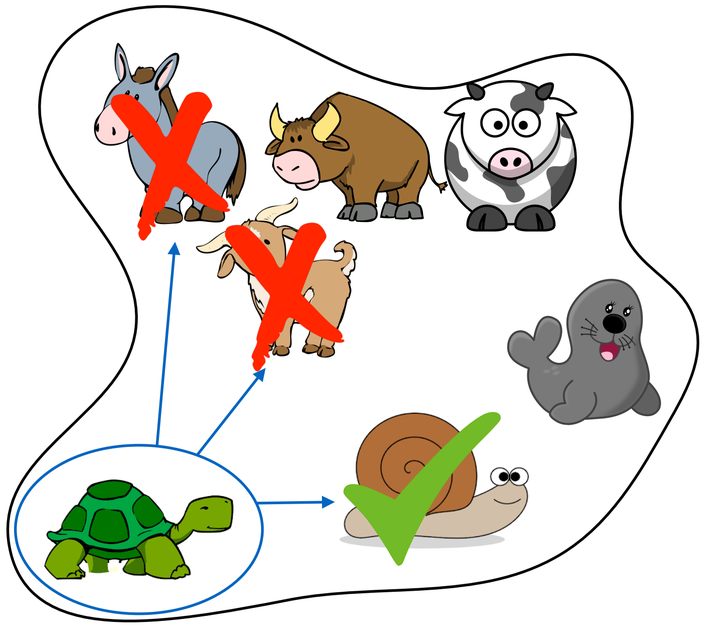
Multi-objectives EA

1)Novelty(x)

2)LC(x)



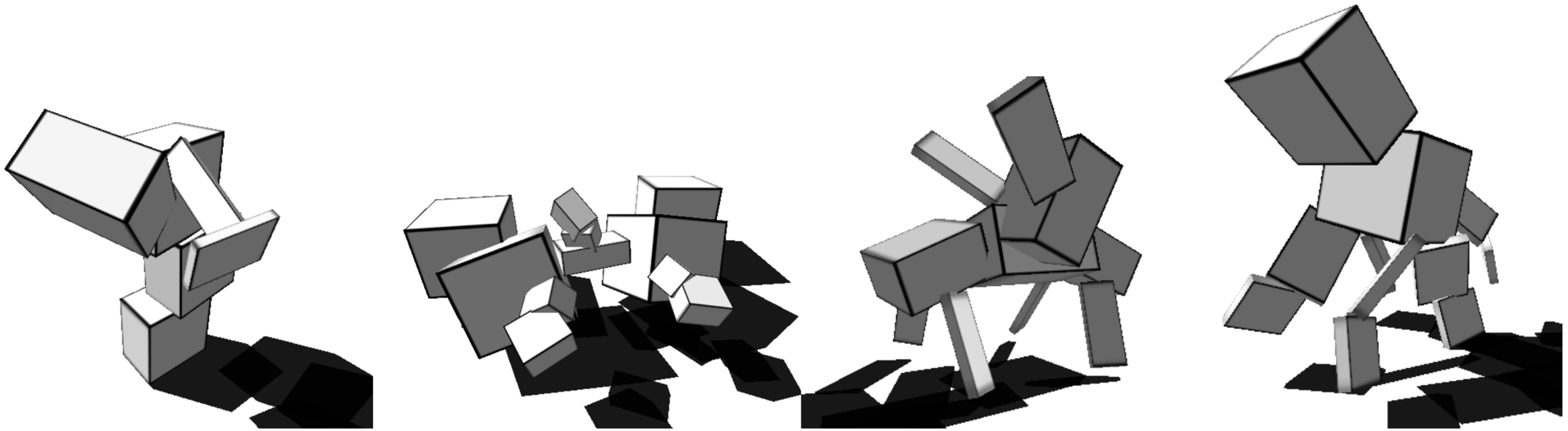
Novelty Archive



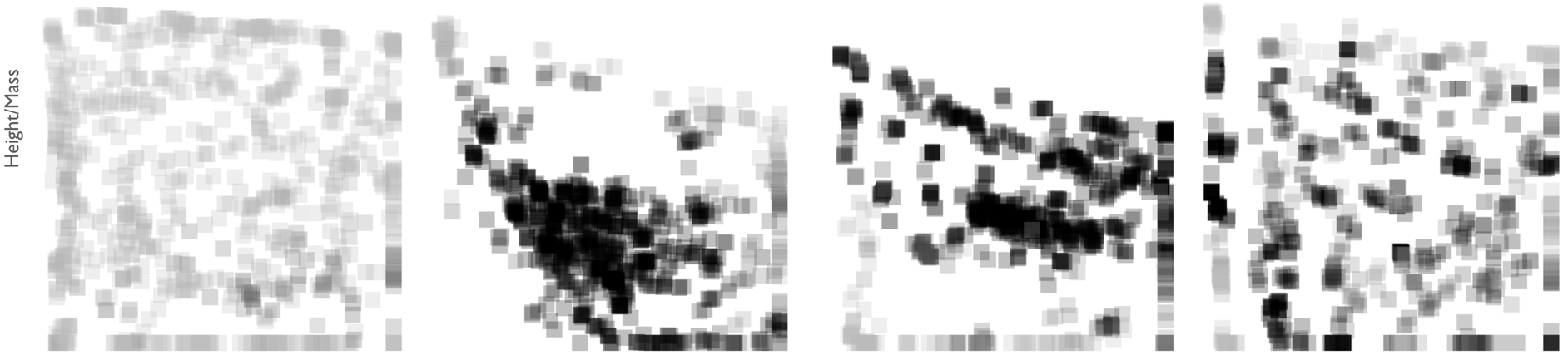
LC(x) = # of outperforming

[1] **Lehman**, J., & **Stanley**, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. *GECCO* 2011.
[2] **Cully**, **Mouret** (2015). Evolving a Behavioral Repertoire for a Walking Robot. Evolutionary Computation. Journal of Evolutionary Computation

Going further: Novelty Search with Local Competition.



(a) Hopper (b) Crab (c) Quadruped (d) Tailed Quadruped



(a) Novelty Only (b) Fitness Only (c) Global Competition (d) Local Competition

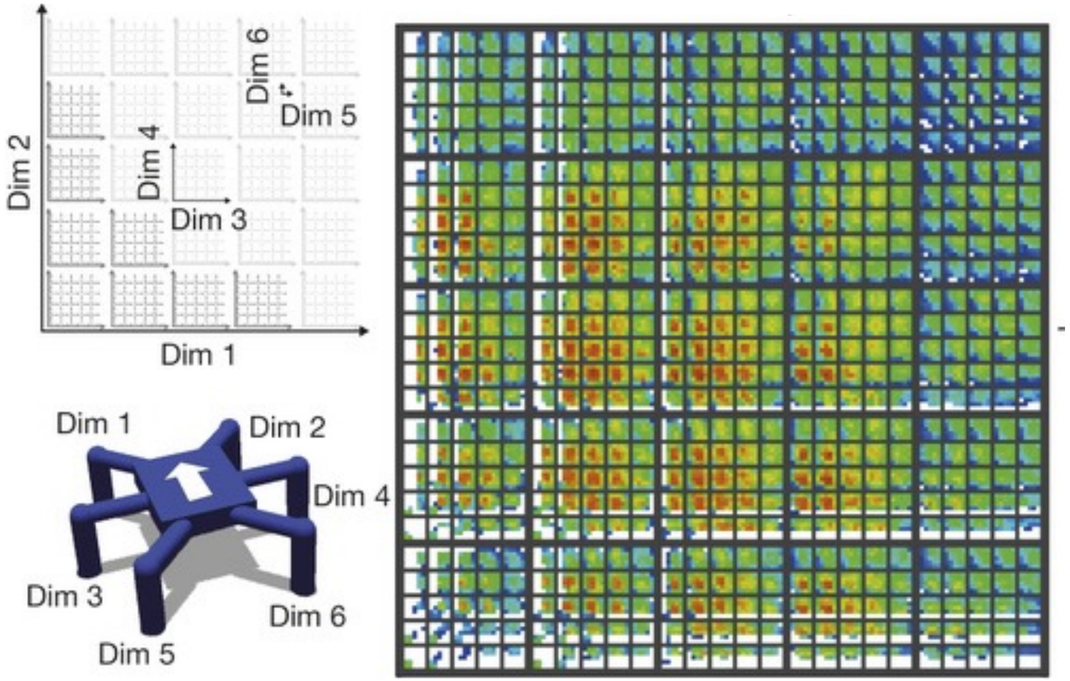
Lehman, J., & Stanley, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. *GECCO* 2011.

To be continued...
(MAP-Elites)

MAP-Elites

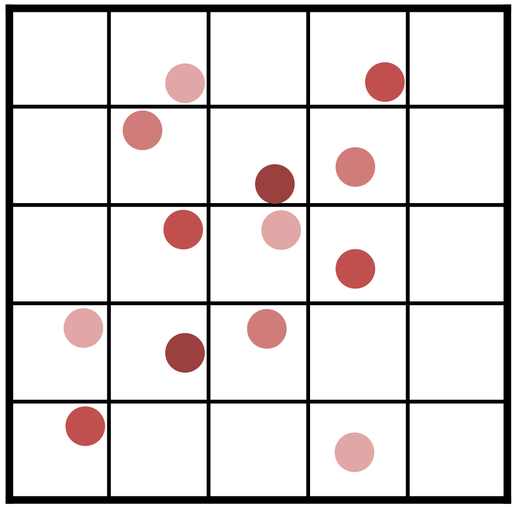
MAP-Elites

- **MAP-Elites** is a very popular QD algorithm.
- Multi-dimensional Archive of Phenotypic Elites.
- Simple, yet very powerful algorithm.
- Goal: Discretise the Behavioural Descriptor space in a grid and try to fill it with the best solutions.



Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

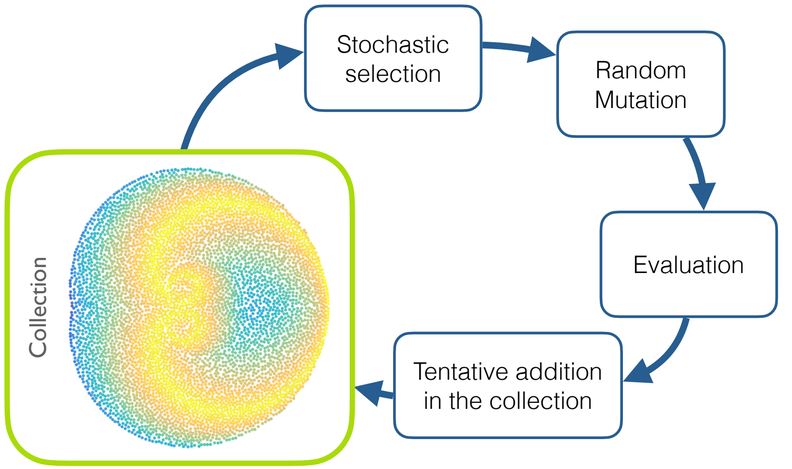
MAP-Elites' Grid

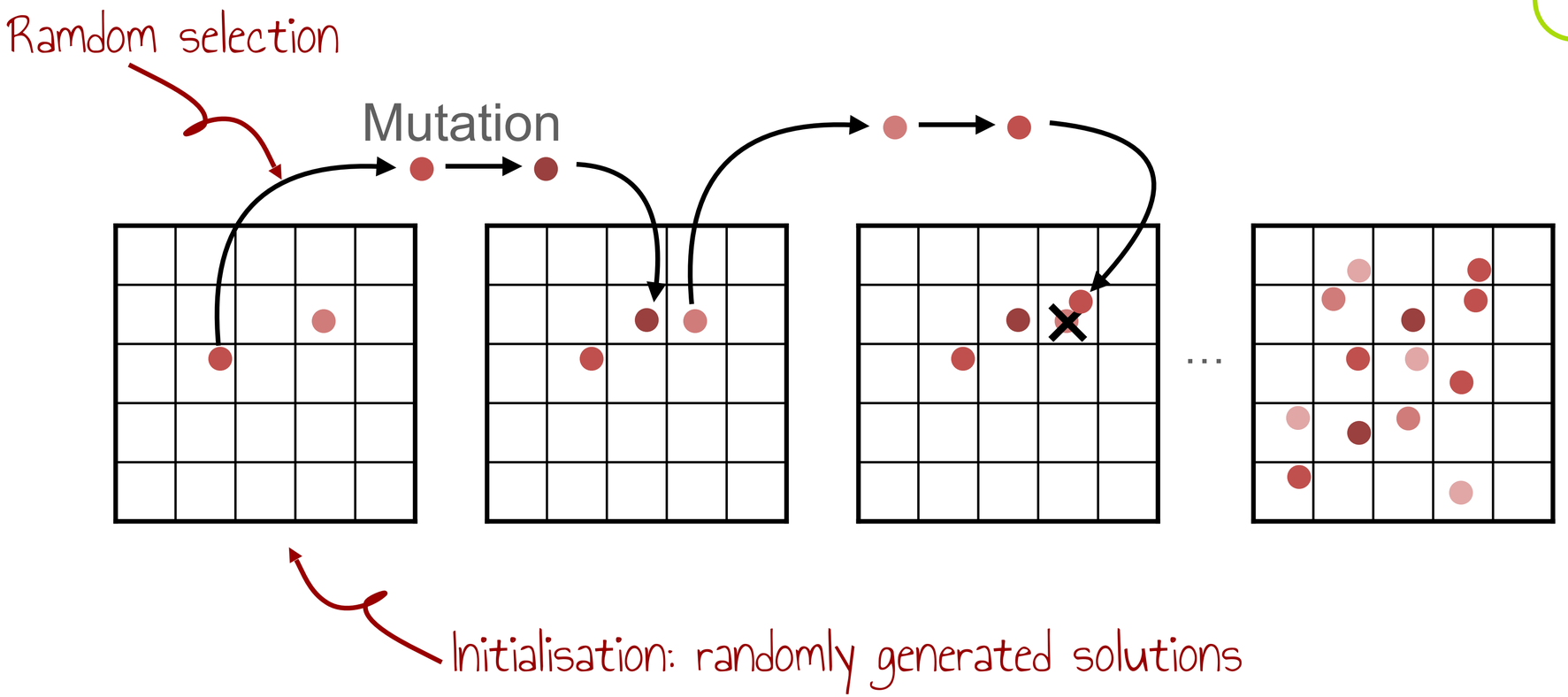


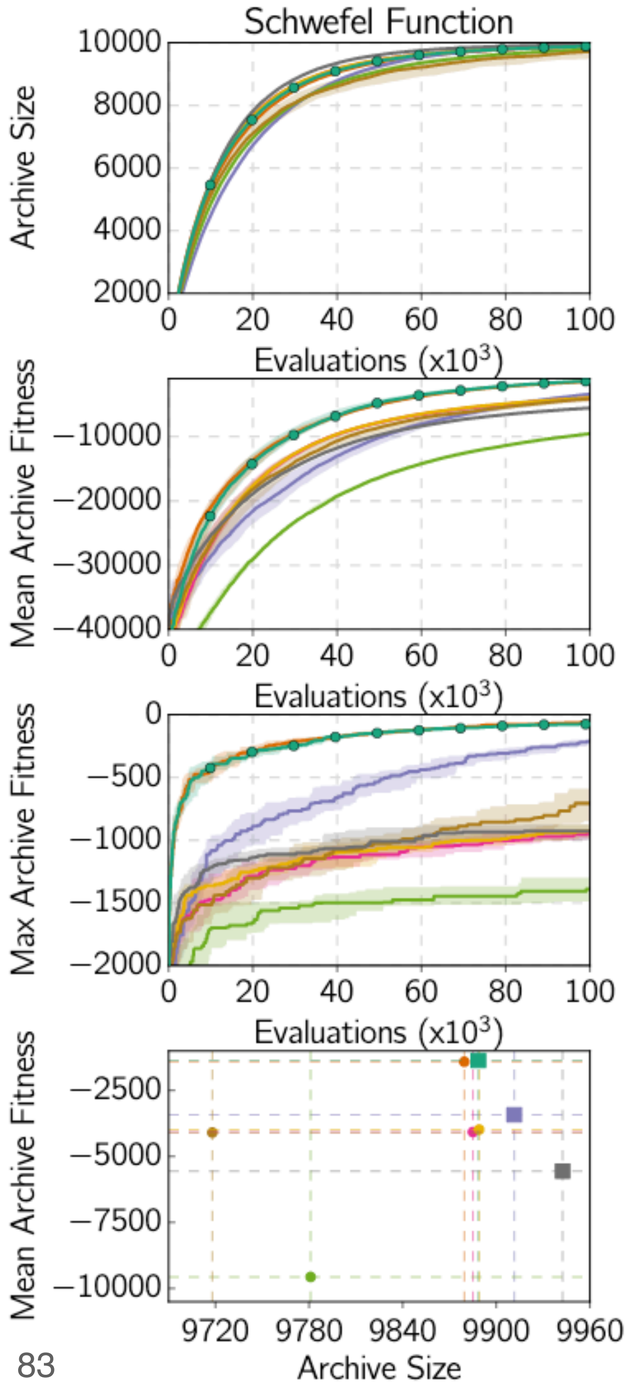
- Discretises the behavioural descriptor space into a set of cells
- **Addition mechanism:**
  - Each new solution goes to the cell corresponding to its BD.
  - If the cell is empty the new solution is added to the grid
  - If the cell is already occupied, the solution with the best fitness is kept
- Hyper-parameter: size of the cells (or resolution of the grid)
- Advantage: Easy to implement
- Drawback: Density not necessarily uniform

MAP-Elites: How does it work?

- **MAP-Elites** = Grid + Uniform random selection
- It is an easy to implement, yet powerful algorithm







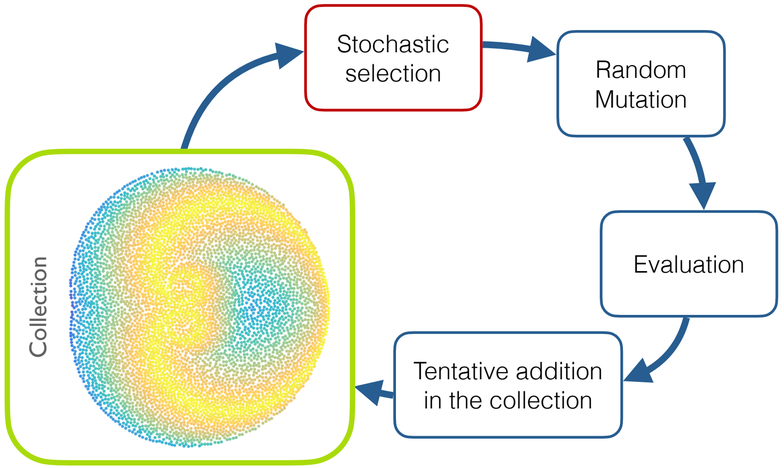
Quantifying performance

- The **diversity** of the solutions in the container
  - Usual metric: archive size
- The **performance** of the solution in the container
  - Usual metric: Max, or mean fitness value
- The **convergence speed** of these two points.
- Often, everything is gathered in the **QD-Score**: the sum of the fitness of all the solutions in the archive (assumed to be strictly positive)
- The trade-off between these different aspects can be represented in a Pareto-front

Pugh, Soros, Stanley. (2016). Quality diversity: A new frontier for evolutionary computation. Frontiers in Robotics and AI.
Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.
Vassiliades, Mouret (2018). Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. GECCO.

A general framework for QD algorithms

- **The selector** is used to select the individual that will be mutated and and evaluated in the next generation
- **The simplest, yet very effective one:**
  - Uniform random selection over the solutions in the container.
- **Alternatively the selection can be proportionally biased according to a score**
  - The fitness
  - The novelty
  - The curiosity score

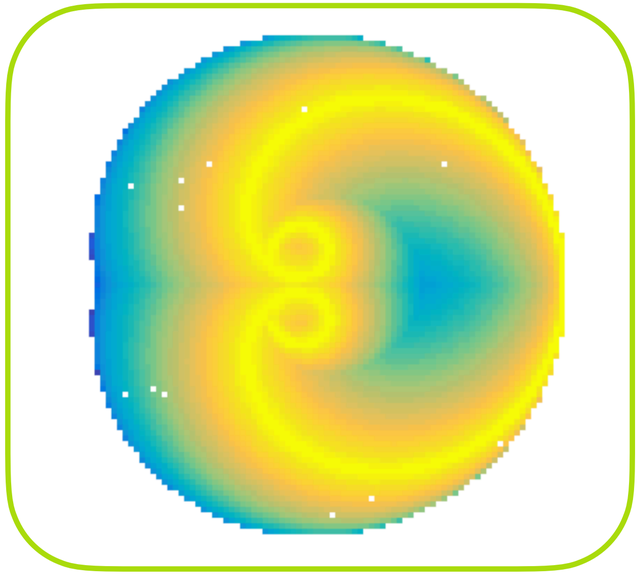


Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.

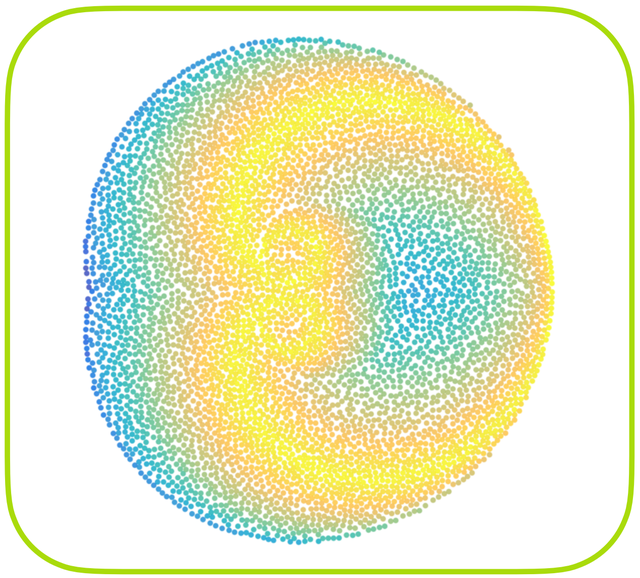
A general framework for QD algorithms

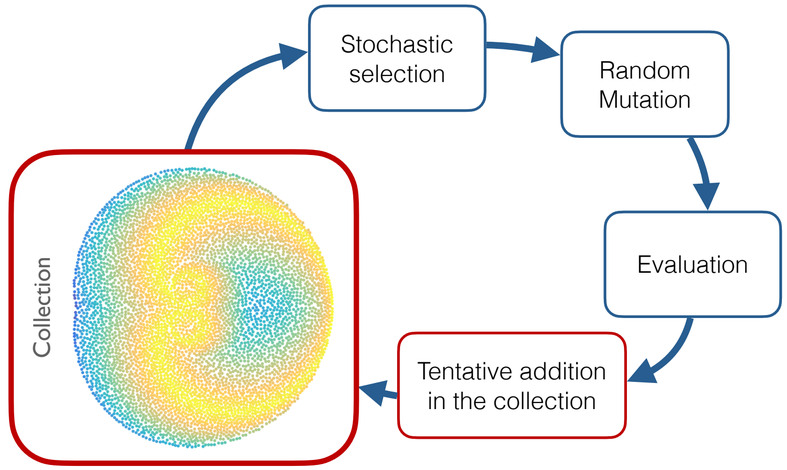
- There are two different ways to store the solutions in QD:

The grid
(from MAP-Elites)



The unstructured archive
(from Novelty Search)





- There are also more advanced mutation and cross-over operators

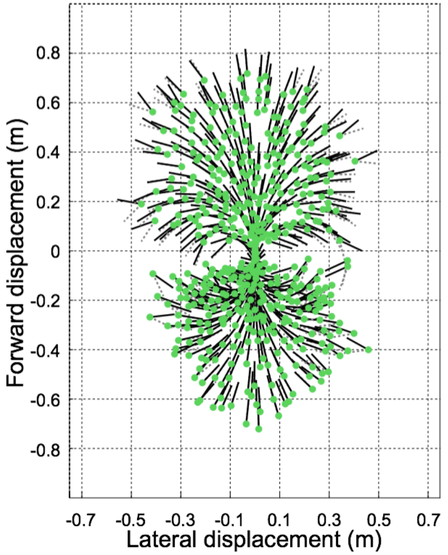
Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. IEEE Trans. in Evolutionary Computation.

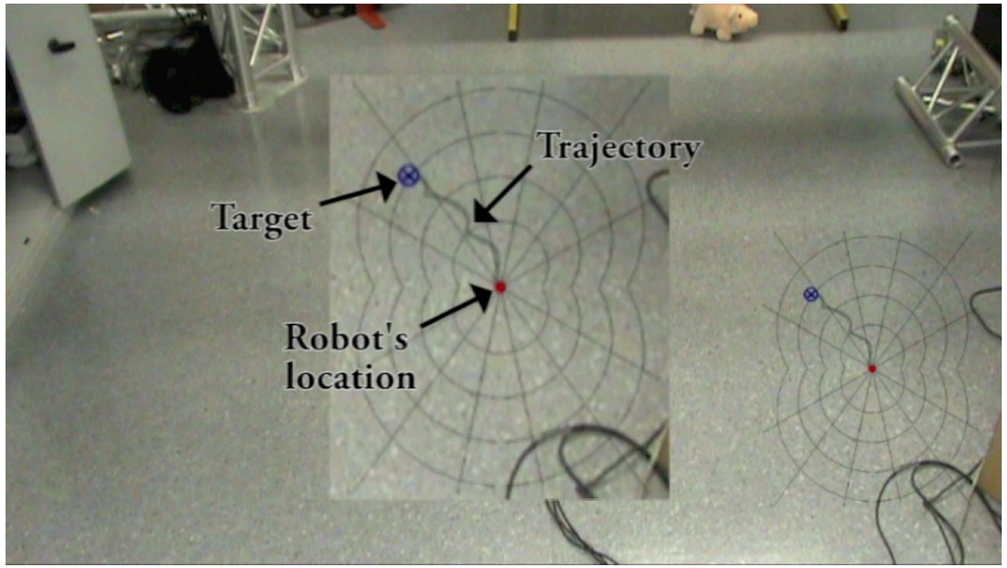
Examples of applications

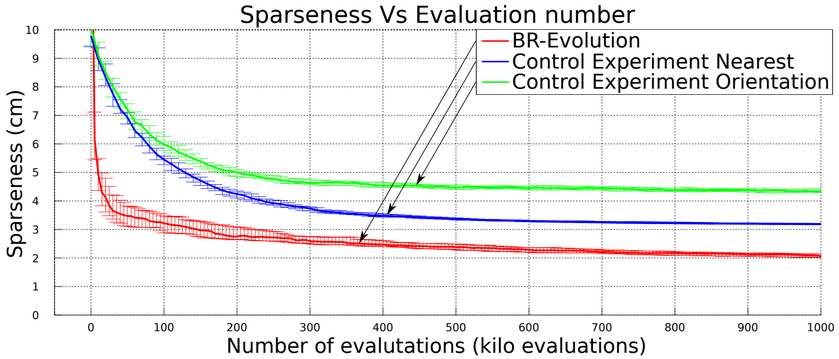
Examples of applications:

Learning to walk in every direction

- **QD algorithm:** unstructured archive + random uniform selector
- **Behavioural descriptor:** X/Y position of the robot after 3 seconds
- **Fitness:** angular error at the end of the trajectory wrt. an ideal circular trajectory







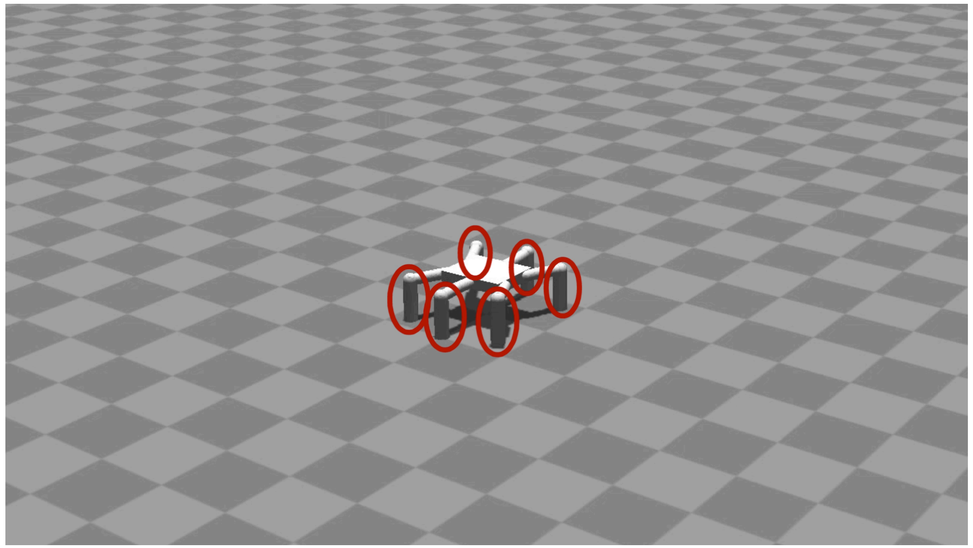
Learning several solutions simultaneously is more effective than learning them one by one.

Cully, Mouret (2015). Evolving a Behavioral Repertoire for a Walking Robot. J. of Evolutionary Computation.
Cully, Mouret (2013). Behavioral repertoire learning in robotics. GECCO.

Examples of applications:

Discovering multiple ways to walk as fast as possible

- Among the 13k different ways to walk the robot has learned some of them are quite creative:



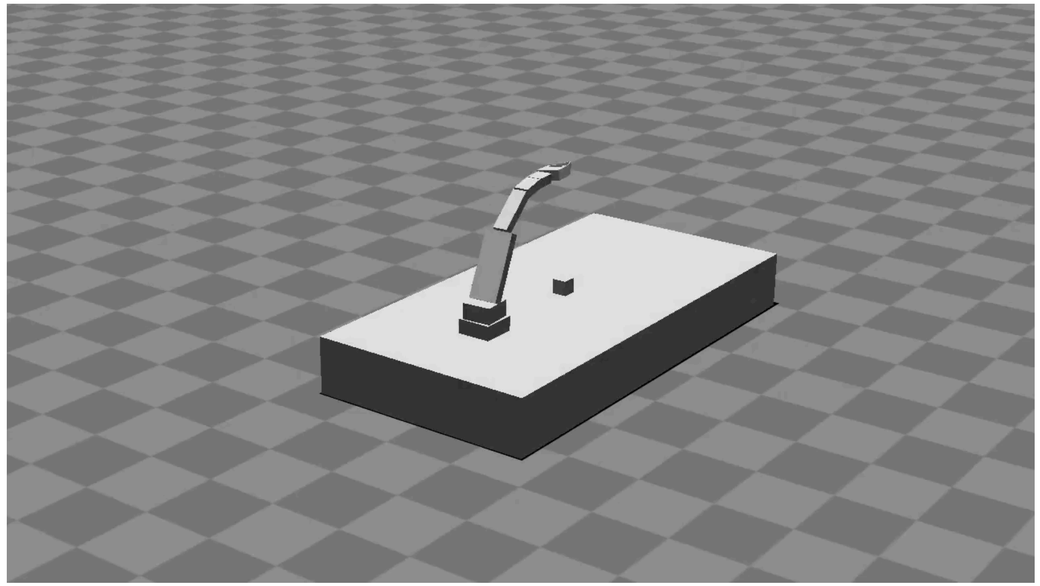
The robot autonomously learned to flip on its back and walk on its knees

 Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. Nature.

Examples of applications:

Another examples of QD's creativity (Learning to push cube)

- **QD algorithm:** MAP-Elites (grid + random uniform selector)
- **Behavioural descriptor:** final position of the cube
- **Fitness:** Energy efficiency of the movement
- Gripper is forced in close position

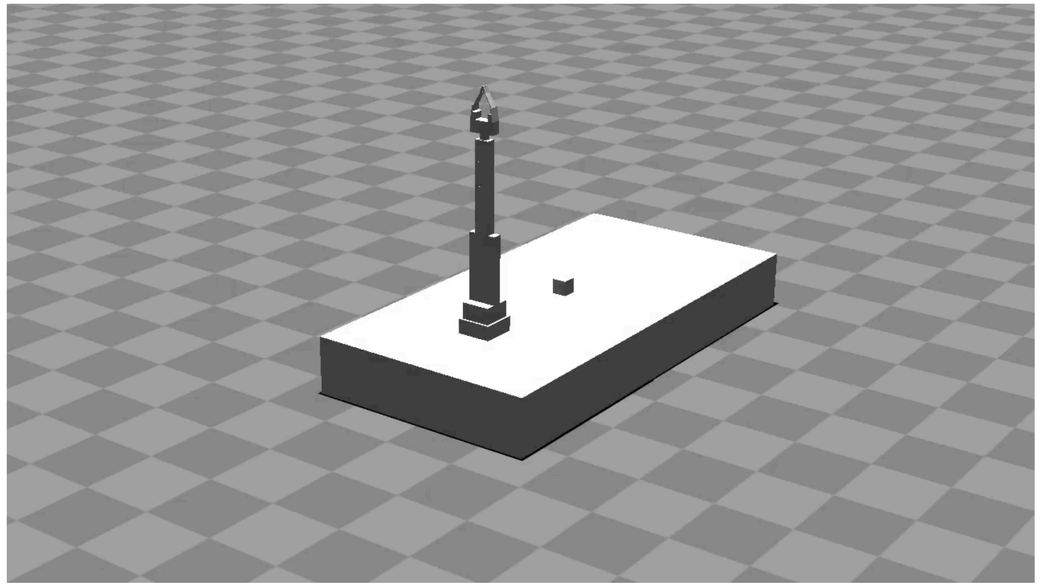


Ecarlat, et al. (2015) Learning a high diversity of object manipulations through an evolutionary-based babbling." IROS. 2015.

Examples of applications:

Another examples of QD's creativity (Learning to push cube)

- **QD algorithm:** MAP-Elites (grid + random uniform selector)
- **Behavioural descriptor:** final position of the cube
- **Fitness:** Energy efficiency of the movement
- Gripper is forced in close position

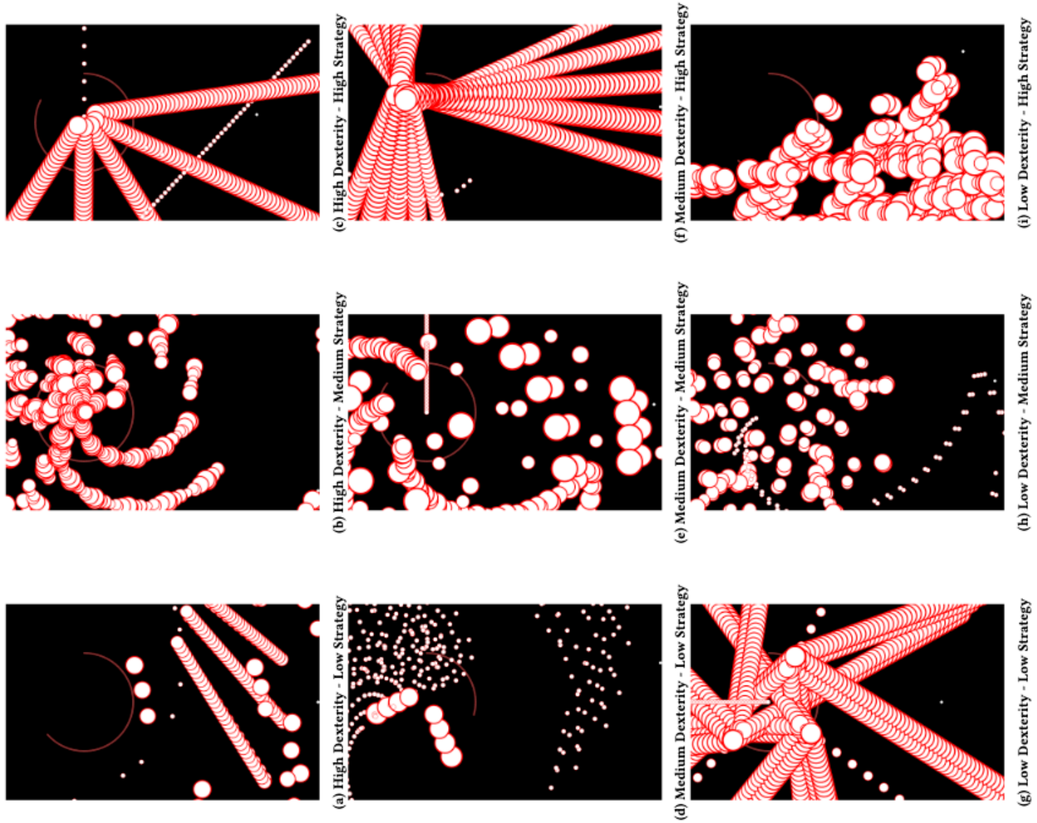


Ecarlat, et al. (2015) Learning a high diversity of object manipulations through an evolutionary-based babbling." IROS. 2015.

Examples of applications:

Content generation in video games

- Use MAP-Elites to generate new levels for bullet hell games.
- Combine MAP-Elites with a feasible-infeasible approach for constraint satisfaction.

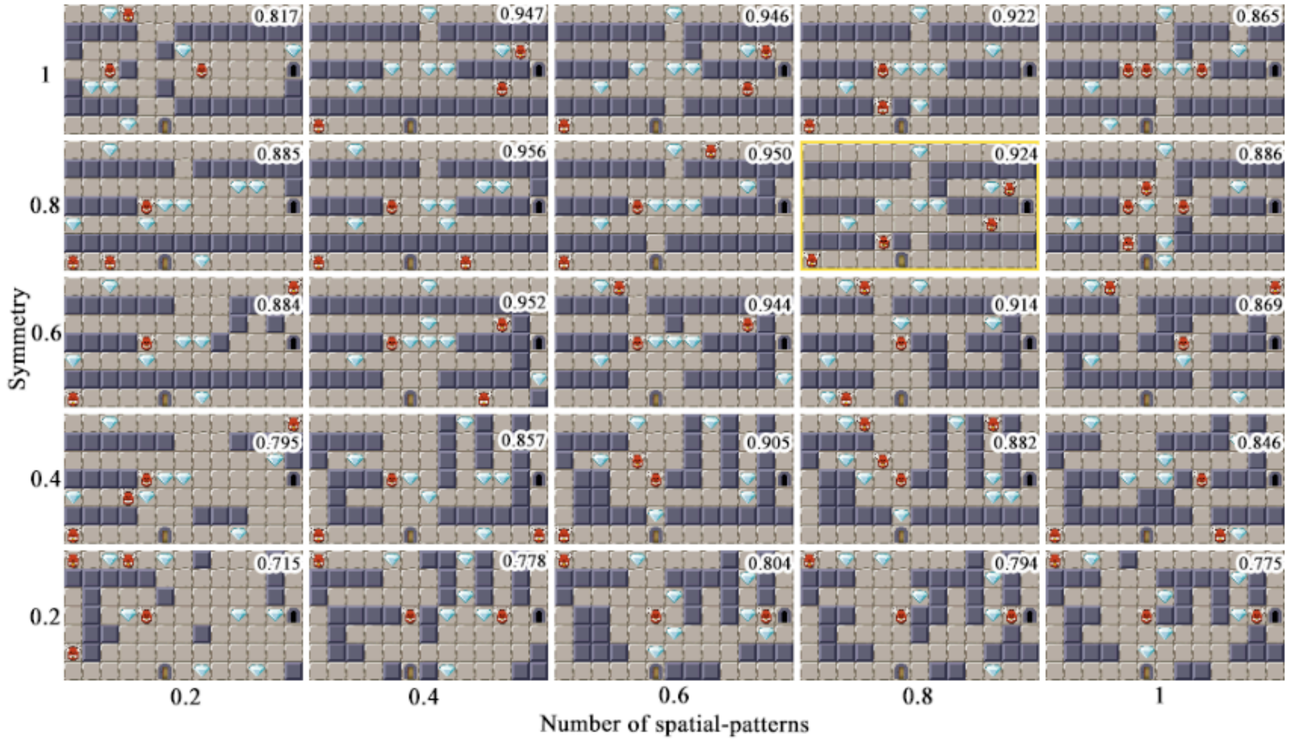


Khalifa, Lee, Nealen, Togelius. (2018). Talakat: Bullet hell generation through constrained map-elites. GECCO.

Examples of applications:

Content generation in video games

- Integration of MAP-Elites in an Evolutionary Dungeon Designer to suggest designs to the users.
- The users can interact with the system to dynamically choose different BD definitions

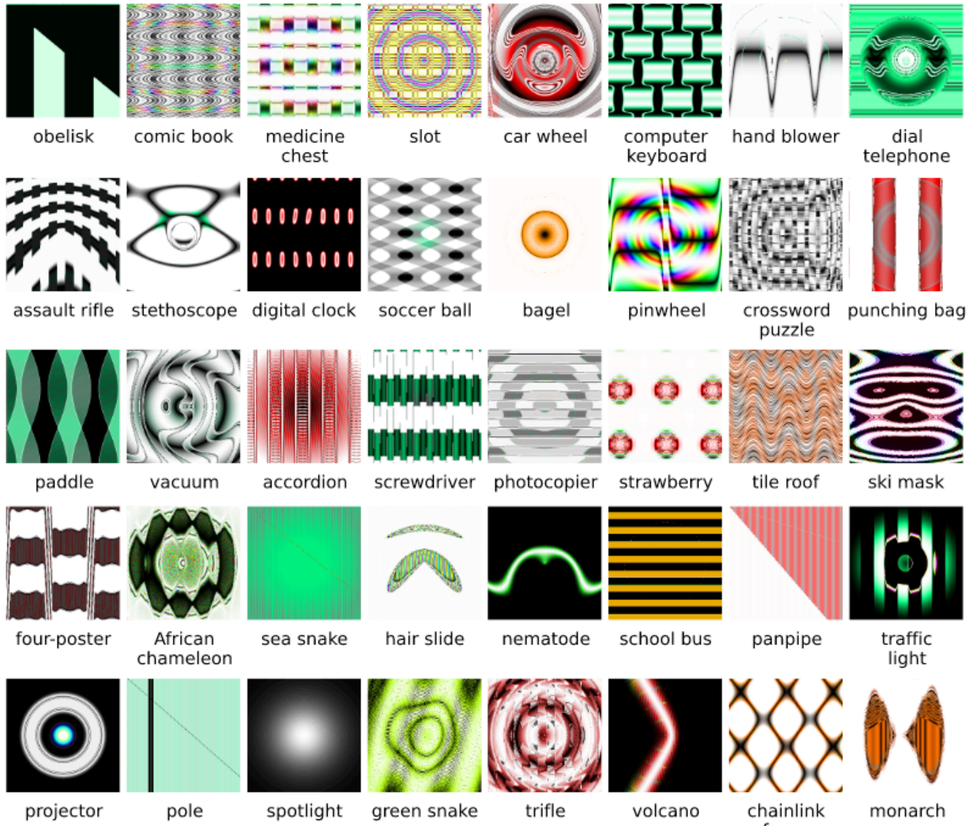


Alvarez, A., Dahlskog, S., Font, J., & Togelius, J. (2019). Empowering quality diversity in dungeon design with interactive constrained MAP-Elites. In 2019 IEEE Conference on Games (CoG) (pp. 1-8). IEEE.

Examples of applications:

Generation of adversarial examples

- MAP-Elites is used to generate a collection of images
- The BD is the class label predicted by a neural network trained on a separate dataset (ImageNet)
- The fitness is the confidence level of the network
- Here, the network has an average confidence of 99.12%



Nguyen, Yosinski, Clune. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. IEEE CVPR

Examples of applications

- **https://quality-diversity.github.io**
  A community website gathering papers, tutorials, and libraries.

2019: 26 Papers

- Adaptive Prior Selection for Repertoire-based Online Adaptation in Robotics Robotics
- Alphastar: An evolutionary computation perspective Games
- An illumination algorithm approach to solving the micro-depot routing problem
- Are quality diversity algorithms better at generating stepping stones than objective-based search?
- Automatic Calibration of Artificial Neural Networks for Zebrafish Collective Behaviours Using a Quality Diversity Algorithm Robotics
- Autonomous Skill Discovery with Quality-diversity and Unsupervised Descriptors Robotics
- Behavioral Repertoire via Generative Adversarial Policy Networks Robotics
- Comparing reliability of grid-based Quality-Diversity algorithms using artificial landscapes
- Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space Games
- Designing neural networks through neuroevolution
- Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites Games
- Evaluating MAP-Elites on Constrained Optimization Problems
- Evolving embodied intelligence from materials to machines Robotics
- Exploration and Exploitation in Symbolic Regression using Quality-Diversity and Evolutionary Strategies Algorithms
- Exploring Self-Assembling Behaviors in a Swarm of Bio-micro-robots using Surrogate-Assisted MAP-Elites Robotics
- Exploring genetic programming systems with MAP-Elites
- Go-Explore: a New Approach for Hard-Exploration Problems Games
- MAP-Elites for Noisy Domains by Adaptive Sampling
- Mapping Hearthstone Deck Spaces with Map-Elites with Sliding Boundaries Games
- Modeling user selection in quality diversity
- Novelty Search for Deep Reinforcement Learning Policy Network Weights by Action Sequence Edit Metric Distance Games
- Novelty search: a theoretical perspective
- POET: open-ended coevolution of environments and their optimized solutions
- Procedural content generation through quality diversity Games
- Quantifying the Effects of Increasing User Choice in MAP-Elites Applied to a Workforce Scheduling and Routing Problem
- Unsupervised Learning and Exploration of Reachable Outcome Space Robotics

2018: 24 Papers

- An approach to evolve and exploit repertoires of general robot behaviours Robotics
- Bayesian optimization with automatic prior selection for data-efficient direct policy search Robotics
- Combining map-elites and incremental evolution to generate gaits for a mammalian quadruped robot Robotics
- Comparing Approaches for Evolving High-Level Robot Control Based on Behaviour Repertoires Robotics
- Data-Efficient Design Exploration through Surrogate-Assisted Illumination
- Discovering the Elite Hypervolume by Leveraging Interspecies Correlation Robotics
- Dynamic mutation in MAP-Elites for robotic repertoire generation Robotics
- Evolution of a Functionally Diverse Swarm via a Novel Decentralised Quality-Diversity Algorithm Robotics
- Evolution of repertoire-based control for robots with complex locomotor systems Robotics
- Evolving Multimodal Robot Behavior via Many Stepping Stones with the Combinatorial Multi-Objective Evolutionary Algorithm Robotics
- Evolving a Repertoire of Controllers for a Multi-function Swarm Robotics
- Exploring genetic programming systems with MAP-Elites
- Hierarchical Behavioral Repertoires with Unsupervised Descriptors Robotics
- Mapping structural diversity in networks sharing a given degree distribution and global clustering: Adaptive resolution grid search evolution with Diophantine equation-based mutations
- Multi-objective Analysis of MAP-Elites Performance Robotics
- Open-ended evolution with multi-containers QD Robotics
- Optimisation and Illumination of a Real-World Workforce Scheduling and Routing Application (WSRP) via Map-Elites
- Prototype discovery using quality-diversity
- Quality Diversity Through Surprise Robotics
- Quality and diversity optimization: A unifying modular framework Review Robotics
- Reset-free trial-and-error learning for robot damage recovery Robotics
- Talakat: Bullet Hell Generation through Constrained Map-Elites Games
- The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities
- Using Centroidal Voronoi Tessellations to Scale Up the Multidimensional Archive of Phenotypic Elites Algorithm

2017: 7 Papers

- A comparison of illumination algorithms in unbounded spaces
- Aerodynamic design exploration through surrogate-assisted illumination
- Comparing multimodal optimization and illumination Robotics
- Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination
- Discovering evolutionary stepping stones through behavior domination
- Feature space modeling through surrogate illumination
- Learning highly diverse robot throwing movements through quality diversity search Robotics

2016: 10 Papers

- An Extended Study of Quality Diversity Algorithms
- EvoRBC: evolutionary repertoire-based control for robots with arbitrary locomotion complexity Robotics
- Gaining insight into quality diversity
- How Do Different Encodings Influence the Performance of the MAP-Elites Algorithm? Robotics
- On the critical role of divergent selection in evolvability Robotics
- Quality Diversity: A New Frontier for Evolutionary Computation Robotics
- Safety-aware robot damage recovery using constrained bayesian optimization and simulated priors Robotics
- Searching for quality diversity when diversity is unaligned with quality
- Towards semi-episodic learning for robot damage recovery Robotics
- Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning

2015: 8 Papers

- Confronting the challenge of quality diversity
- Constrained novelty search: A study on game content generation Games
- Enhancing divergent search through extinction events Robotics
- Evolving a behavioral repertoire for a walking robot Robotics
- Extinction events can accelerate evolution
- Illuminating search spaces by mapping elites Robotics
- Innovation engines: Automated creativity and improved stochastic optimization via deep learning
- Robots that can adapt like animals Robotics

2013: 2 Papers

- Behavioral repertoire learning in robotics Robotics
- Transforming exploratory creativity with DeLeNoX Games

2011: 3 Papers

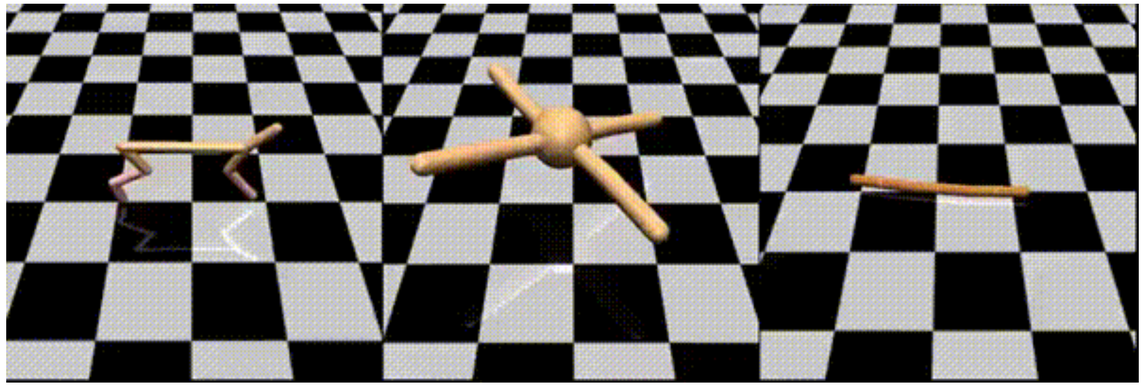
- Abandoning objectives: Evolution through the search for novelty alone Robotics
- Evolving a diversity of virtual creatures through novelty search and local competition Robotics
- Novelty-based multiobjectivization Robotics

To be continued...
(Concluding thoughts)

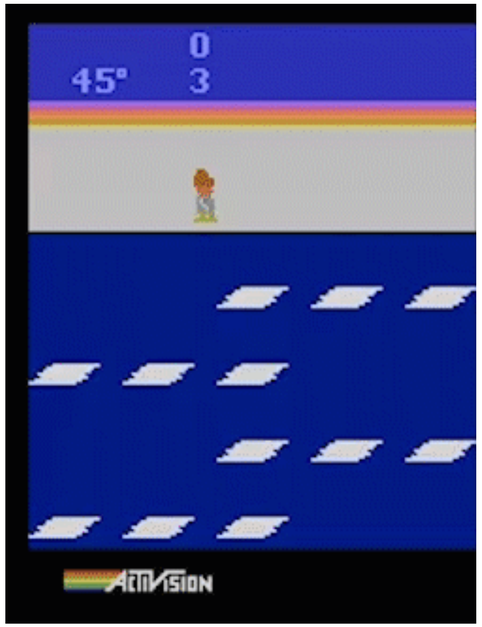
Concluding thoughts

Evolutionary algorithms today?

- Evolutionary algorithms (e.g., CMA-ES) are currently used to train deep neural networks (both the hyper-parameters and the weights). They start to become a real alternative to deep reinforcement learning.



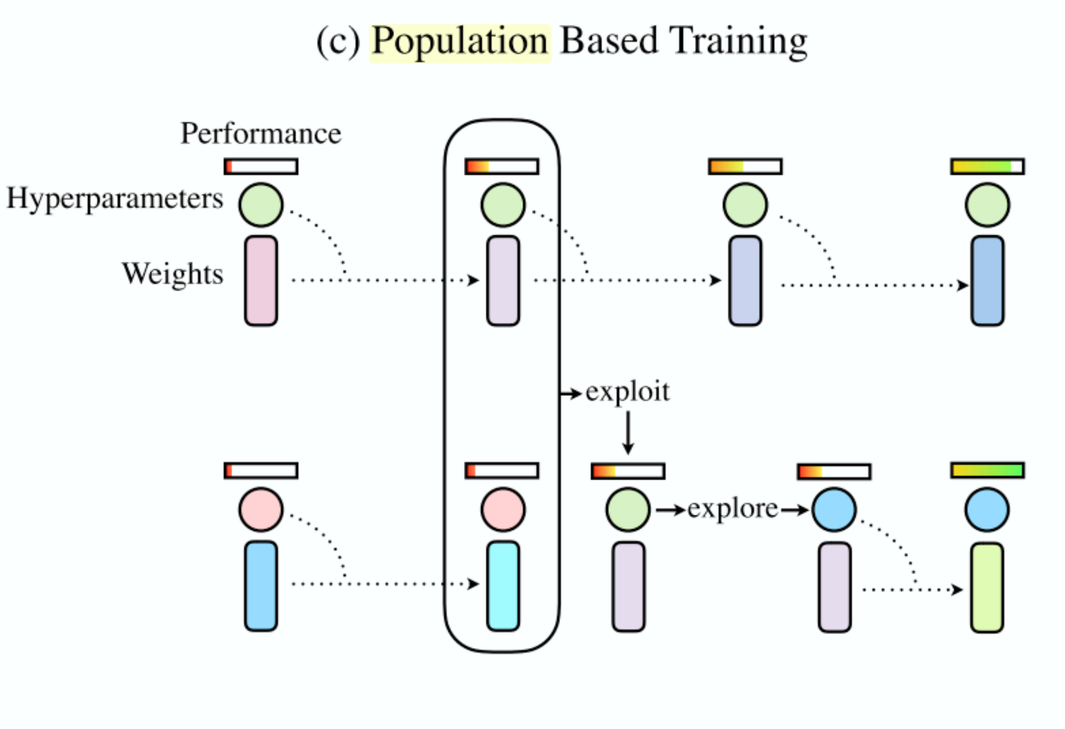
Open-AI labs [1]

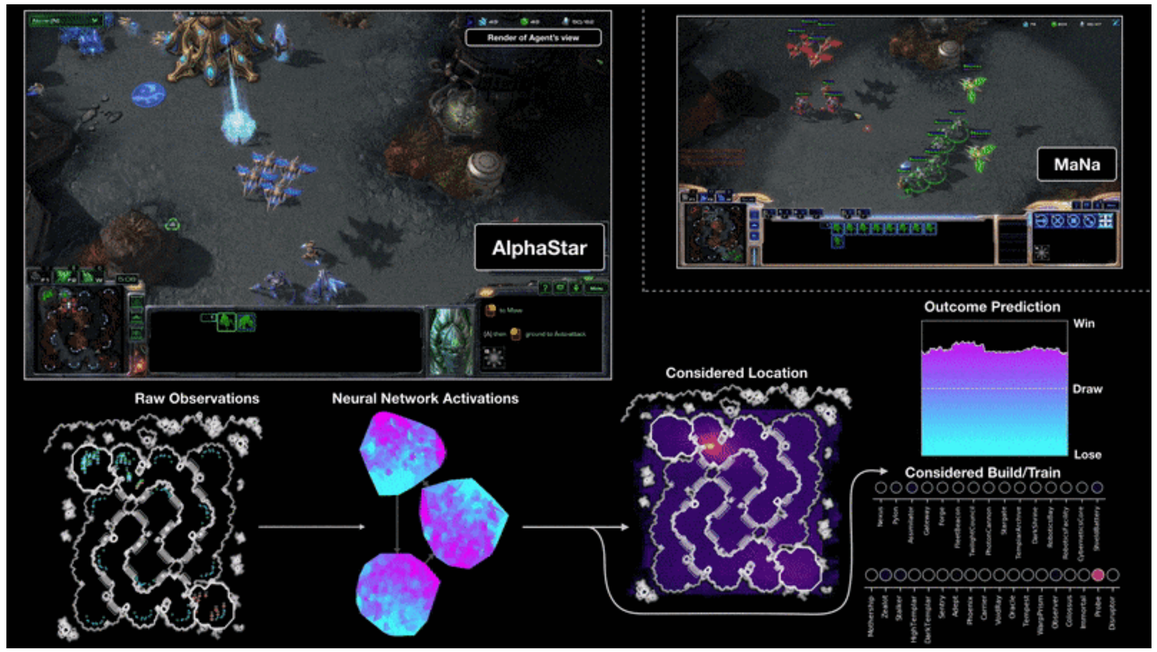


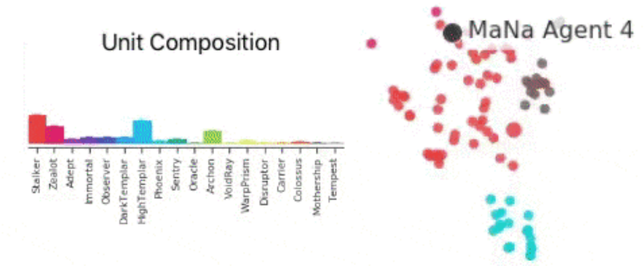
UBER AI-Labs: This GA policy scores 10,500 on Frostbite. DQN, AC3, and ES score less than 1,000 on this game [2].

[1] Salimans, Tim, et al. "Evolution strategies as a scalable alternative to reinforcement learning." *arXiv preprint arXiv:1703.03864* (2017).
[2] Such, Felipe Petroski, et al. "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning." *arXiv preprint arXiv:1712.06567* (2017).

AlphaStar *By DeepMind*

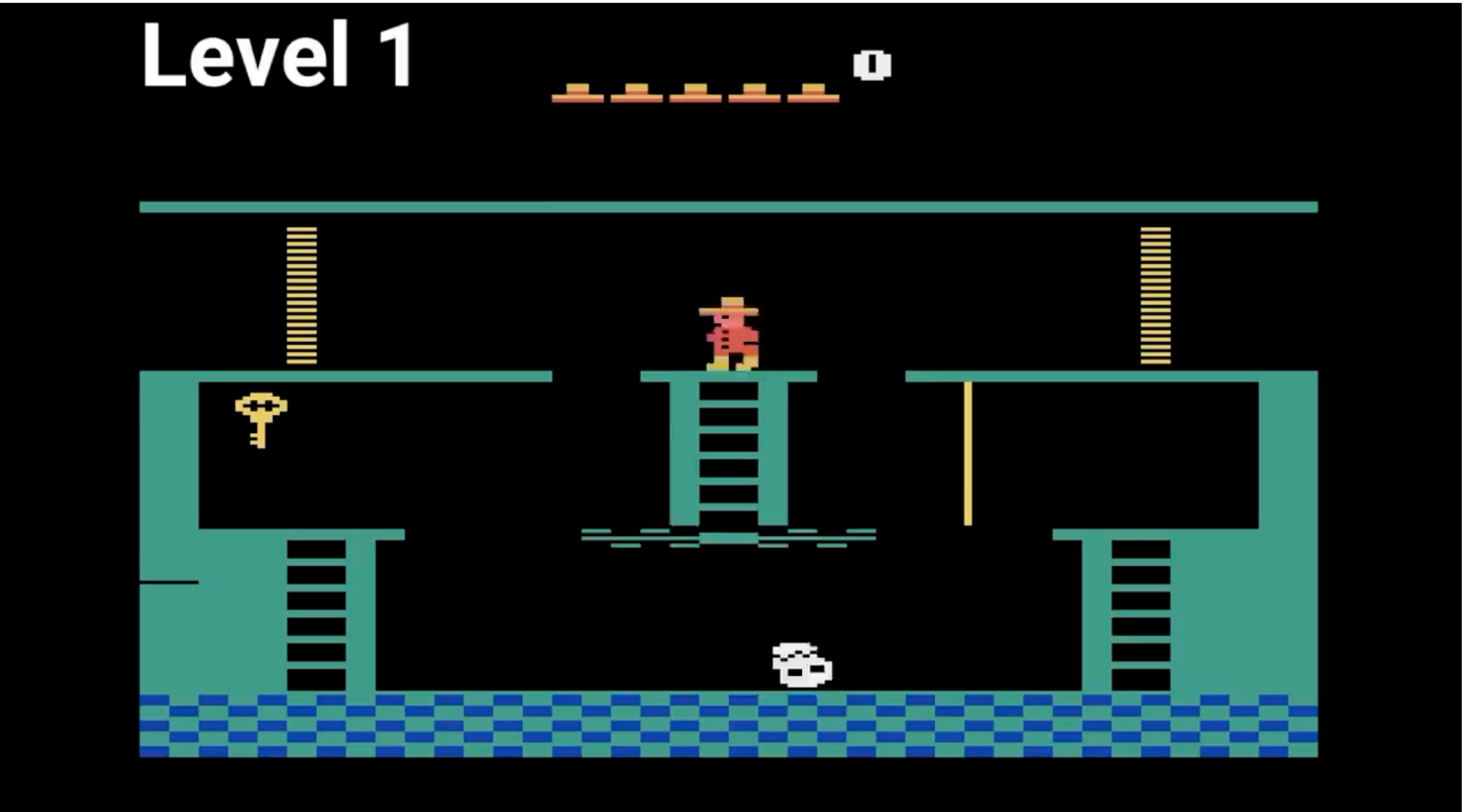






ATARI Games

UBER AI Labs



Recap (1)

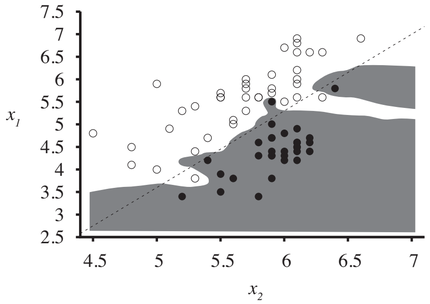
- Genetic algorithms, Evolutionary Strategies, Evolutionary algorithms: **They all rely on the same main concepts**
- The main differences are the **operators** used in each case.
- **A lot of different operators exist** (often problem specific)
- The optimal parameters of an algorithm will often change depending on the problem.
- Today, genetic algorithms on binary strings are rarely used; ES or Evolutionary algorithms in general are more common

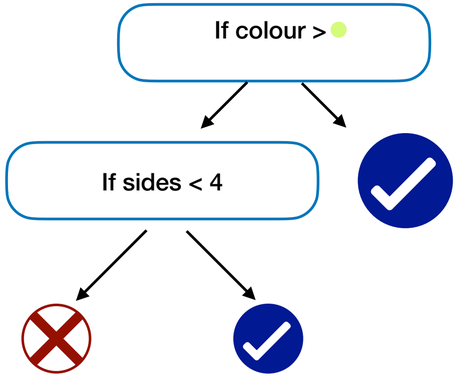
Recap (2)

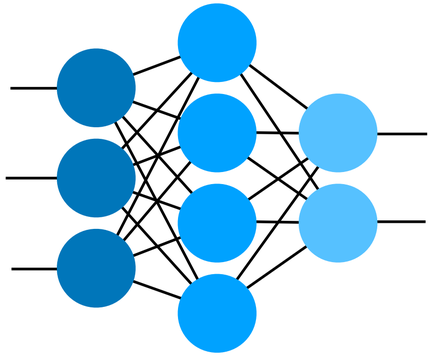
- **Work on black-box problems** (e.g., hyper-parameters of NN)
- Good exploration capabilities:
  If there is enough **diversity** in the population, we can expect that some individuals will explore other regions and leave the local optimums.
- Easy to **parallelise**: Perform the evaluation in parallel.
- Downside: Slower than gradient descend when the gradient is known and when the problem is simple (e.g., convex).

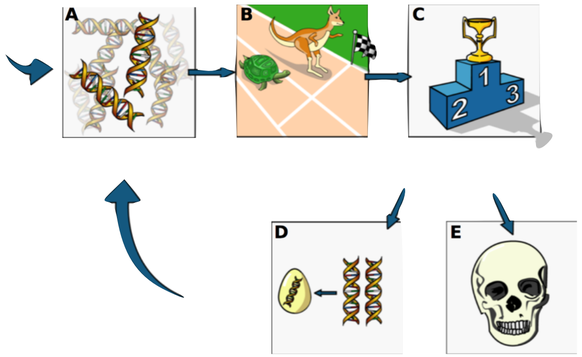
End of our introductory journey

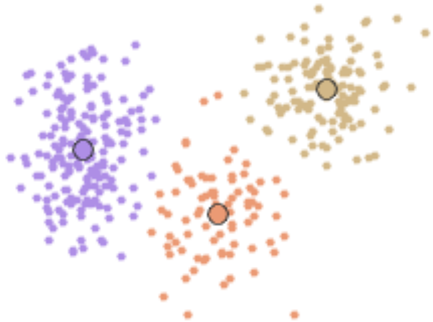
We hope you enjoyed it











See you next week!