Quality-Diversity Optimisation

Antoine Cully
Imperial College London
London, UK
a.cully@imperial.ac.uk

Jean-Baptiste Mouret
Inria Nancy - Grand Est,
CNRS, Université de Lorraine,
F-54600, France
jean-baptiste.mouret@inria.fr

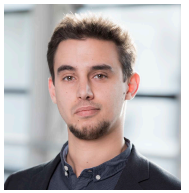
Stéphane Doncieux
ISIR, Sorbonne Université,
CNRS UMR 7222, F-75252, Paris
stephane.doncieux@upmc.fr

http://gecco-2021.sigevo.org/

GECCO '21 Companion, Lille, France

10.1145/3449726.3461403...$15.00

Instructors



- **Antoine Cully** is director of the Adaptive and Intelligent Robotics Lab (AIRL) and Lecturer (assistant Prof) at Imperial College London, UK. His research is at the intersection between artificial intelligence and robotics, and aims at increasing the versatility and adaptation capabilities of robots.

- **Jean-Baptiste Mouret** is a senior research scientist ("directeur de recherche") at Inria, in Nancy, France (https://members.loria.fr/jbmouret/). His main interest is to leverage machine learning and evolutionary computation to make robots more adaptive in the real world. JB Mouret co-introduced MAP-Elites and contributed many ideas about behavioral diversity in evolutionary robotics.

- **Stephane Doncieux** is Professor at Sorbonne University, in Paris, France. His researches are on open-ended learning in robotics with a strong use of evolutionary algorithms. He is deputy director of the Institute of Intelligent Systems and Robotics (ISIR).

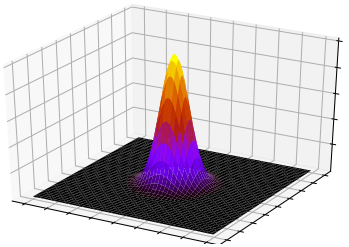
Course Agenda

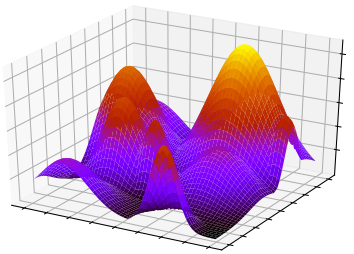
- **Evolution beyond optimisation**
- **Quality *and* Diversity**
- **The two strategies to cover a reachable space**
- **QD algorithms: How does it work?**
- **Examples of applications**
- **More recent concepts**
- **Scaling-up QD**
- **Brief overview of existing implementations**
- **Open questions/challenges**

What is evolution about ?

- Fitness landscape metaphor [Wright 1932]
- Evolution as an optimisation method:

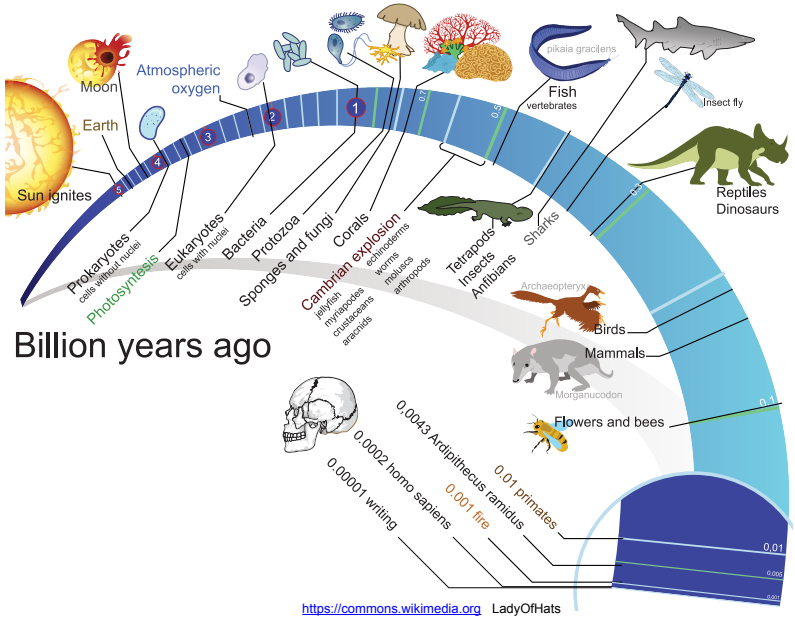
Find x maximising F(x)





Wright (1932), The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in evolution. *Sixth International Congress on Genetics.*

Is that all evolution can do ?



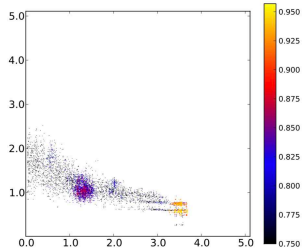
Evolution beyond optimisation

- **From:**
  - Find x maximising F(x) : find ***the*** optimal way of solving a given problem
    - outcome: single value
- **To:**
  - Find all possible x ***of interest***
    - outcome: (large) set of points

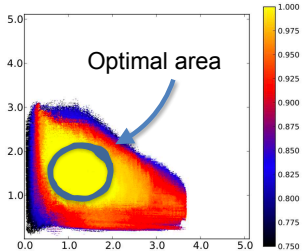
      (definition of interest comes later…)

Why looking for a large set of points instead of a single, optimal, solution ?

- *« **Improved optimization performance**; the algorithm often finds a better solution than the current state-of-the-art search algorithms in complex search spaces because it **explores more of the feature space**, which helps it avoid local optima and thus find different, and often better, fitness peaks . »*



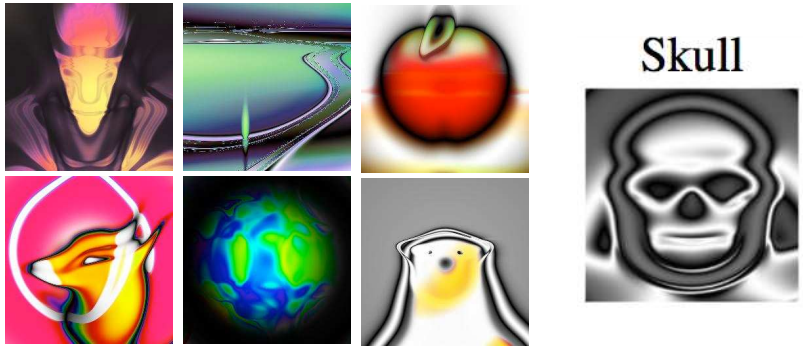
Traditional EA



QD algorithm

Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint*

On the limits of objective-based search



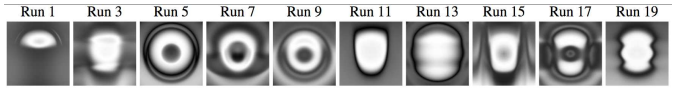
Images generated with CPPN
through an interactive evolution process (PicBreeder)

Secretan, Beato, D'Ambrosio, Rodriguez, Campbell, Folsom-Kovarik, Stanley (2011) Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation journal*

On the limit of objective-based search

Goal: Skull

Results with an objective-based search, after 30 000 generations with a population size of 150:

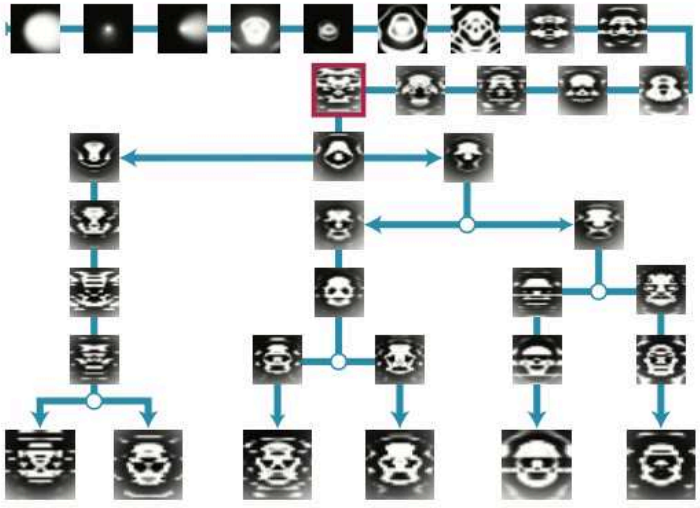


Intermediate steps in Picbreeder:

(population size = 15)

Woolley, Stanley (2011) On the deleterious effects of a priori objectives on evolution and representation. *Proc of GECCO.*

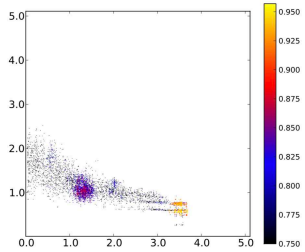
Why looking for a large set of points instead of a single, optimal, solution ?



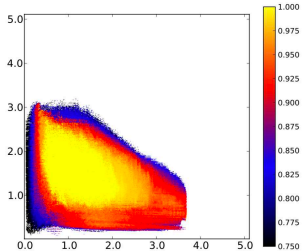
Gaier, Asteroth, Mouret (2019). Are quality diversity algorithms better at generating stepping stones than objective-based search? *GECCO companion (poster)*

Why looking for a large set of points instead of a single, optimal, solution ?

- *« **Illuminating the fitness potential of the entire feature space**,not just the high-performing areas, revealing relationships between dimensions of interest and performance. »*



Traditional EA

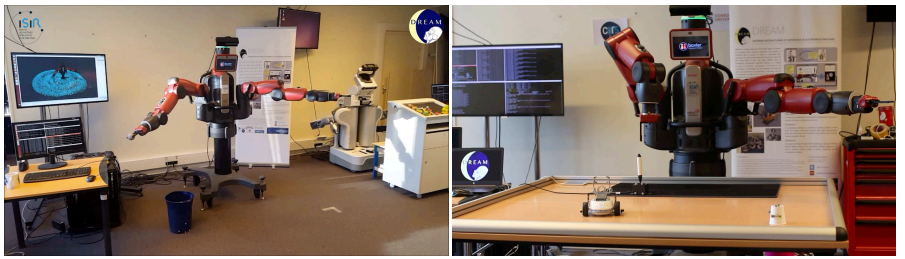


QD algorithm

Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint*

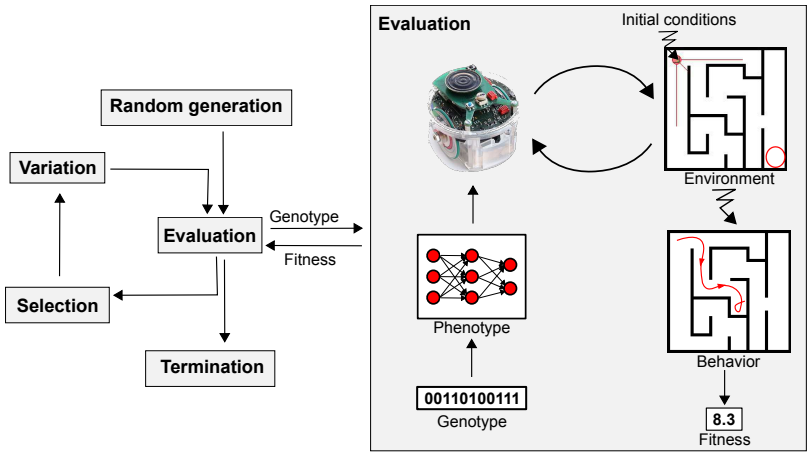
Why looking for a large set of points instead of a single, optimal, solution ?

- Application to robotics: generating (off-line) a repertoire of behaviours to exploit (on-line)



Kim, Coninx, & Doncieux (2021). From exploration to control: learning object manipulation skills through novelty search and local adaptation. *Robotics and Autonomous Systems.*

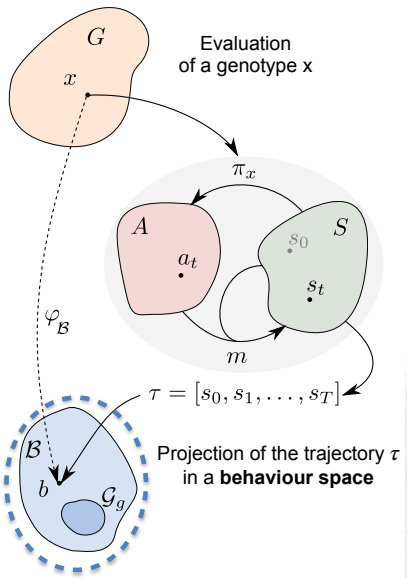
Evolution to design behavioural systems



❖ applications in robotics, video games, engineering, …

Doncieux, Bredeche, Mouret, Eiben (2015) Evolutionary robotics: what, why, and where to. *Front. Robot. AI*

What is a genotype *of interest*?



? ? ?

Find all possible x ***of interest***

Introducing an new characterisation of x: **behaviour descriptor**

Given a set of genotypes $\mathcal{S}$, x is **interesting** if it reaches a behaviour that is original or better than those of $\mathcal{S}$

Definition of **Quality-Diversity algorithms**:

Find a set of solutions {x_i}:

1. covering the whole reachable behaviour space
2. with a good (local) quality

Covering the whole reachable space

❖ A key feature: **evolvability**

« the capability of a system to generate adaptive phenotypic variation and to transmit it via an evolutionary process »

[Hu and Banzhaf, 2010]

❖ 2 strategies: 2 families of QD algorithms

- **Strategy 1**: searching for novelty (Novelty Search)
- **Strategy 2**: searching for empty niches (MAP-Elites)

Hu, Banzhaf (2010) Evolvability and speed of evolutionary algorithms in light of recent developments in biology. *Journal of Artificial Evolution and Applications.*

Covering the whole reachable space

Strategy 1: searching for novelty

❖ Novelty search: replace any goal-oriented fitness by a measure of novelty in the behaviour space

❖ Maximize: $$\rho(x) = \frac{1}{k}\sum_{i=0}^{k} dist(x, \mu_i)$$

❖ $\{\mu_0, \ldots, \mu_{k-1}\}$ are the k-nearest neighbors in pop+archive

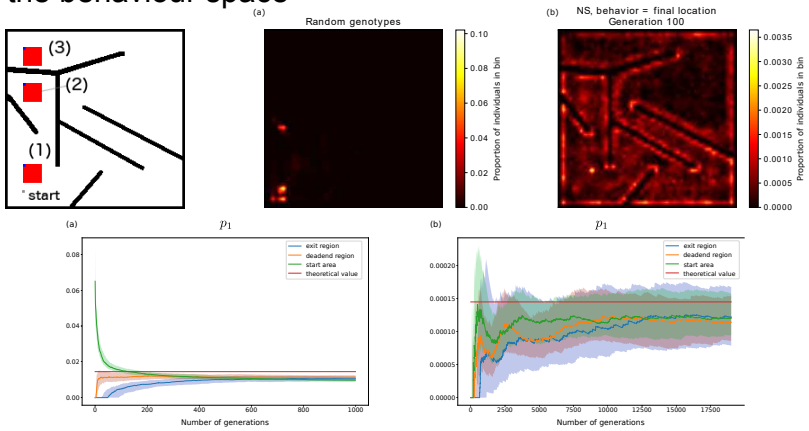
❖ Archive:

- Samples from past generations
- Typically augmented with individuals having a high novelty

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

Covering the whole reachable space
Strategy 1: searching for novelty

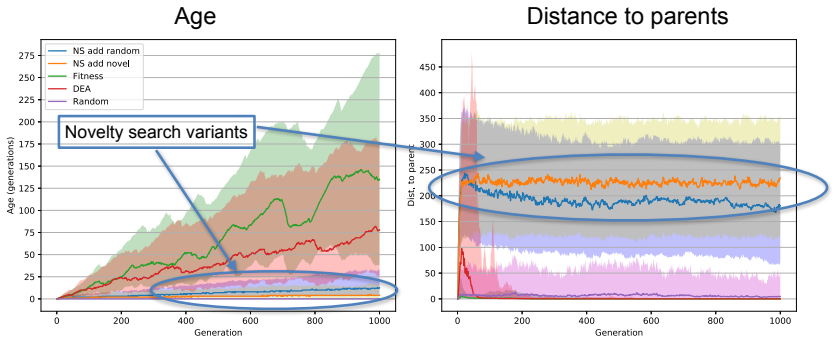
❖Novelty search converges towards a uniform sampling in the behaviour space



Doncieux, Laflaquière, Coninx (2019). Novelty Search: a Theoretical Perspective. *Proc. Of GECCO*

Covering the whole reachable space
Strategy 1: searching for novelty

❖How does it work ? ➡ Evolvability results from a perpetual movement in the behaviour space



Doncieux, Paolo, Laflaquière, Coninx (2020). Novelty Search makes Evolvability Inevitable. *Proc. of GECCO*

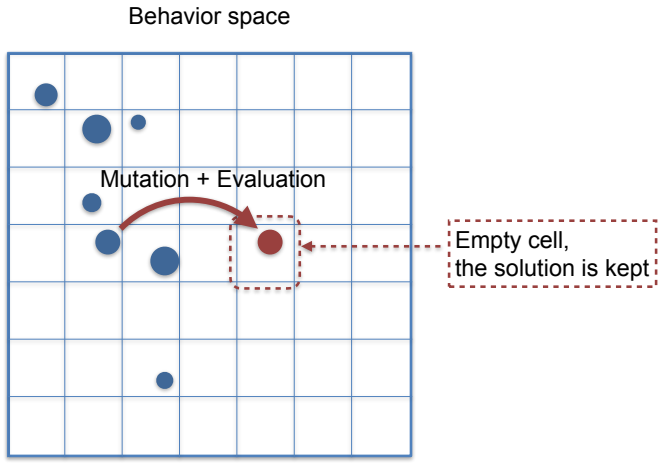
Getting a good (local) quality
Strategy 1: searching for novelty

❖ Multi-objective approach: NSLC (Novelty Search with Local Competition)

- **Novelty objective**: average distance to the k-nearest neighbours
- **Local competition objective**: number of neighbours (among the k nearest) with lower fitness

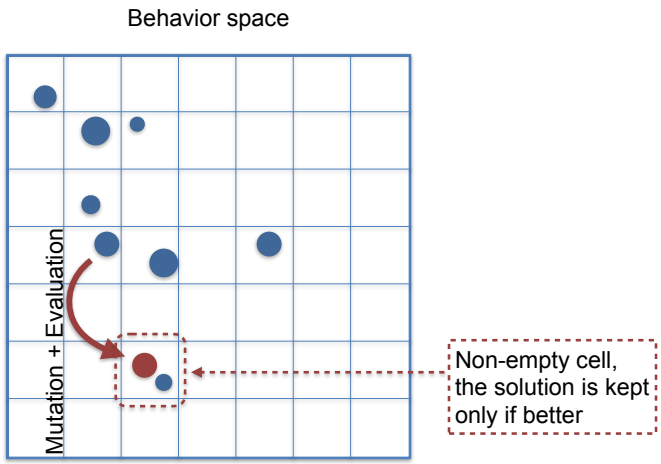
Lehman, Stanley (2011). Evolving a diversity of virtual creatures through novelty search and local competition. *Proc. of GECCO.*

Covering the whole reachable space
Strategy 2: MAP-Elites, searching for empty niches



Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint*

Getting a good (local) quality
Strategy 2: MAP-Elites, searching for empty niches

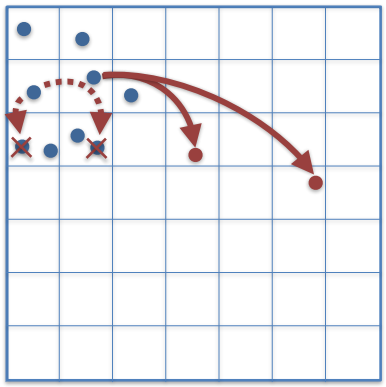


Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint*

Covering the whole reachable space
Strategy 2: searching for empty niches

❖How does it work ?

➡Evolvability results from a « founder » effect



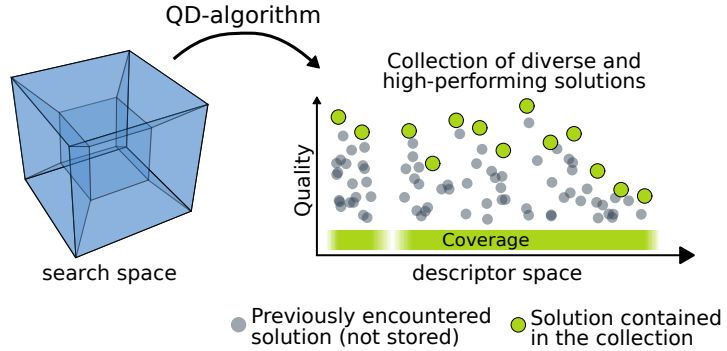
Individuals with a larger evolvability have a better chance to reach empty cells

Lehman, Stanley (2013). Evolvability is inevitable: Increasing evolvability without the pressure to adapt. *PloS one.*

QD algorithms: How does it work?

❖ **QD objective:**

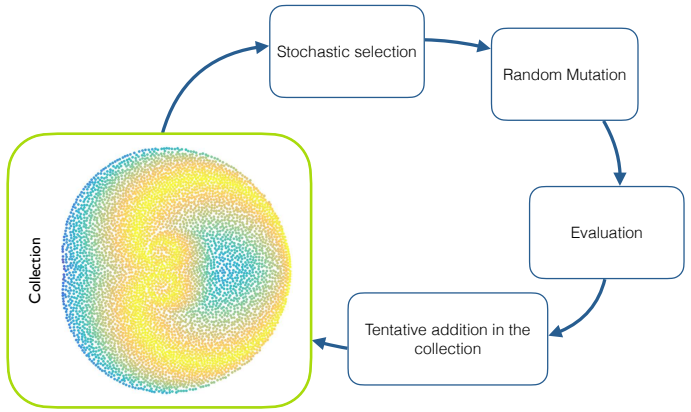
Learning in a single optimisation process
a large collection of diverse and high-performing solutions



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.*

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

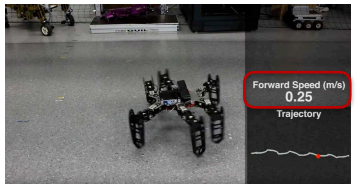
- To build a collection of high-performing and diverse solutions, we need to:
  - Measure the performance of solutions
  - Distinguish different types of solutions
- For that, we use:
  - A **fitness function**, like in most evolutionary algorithms
  - A **behavioural descriptor** (also called behavioural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

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

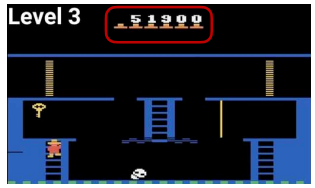
QD algorithms: How does it work?
Behavioural Descriptor and Fitness functions

- The fitness function is directly related to the task
- Examples of fitness functions:

Maximising velocity to learn to walk



Maximising high-score to learn to play Atari

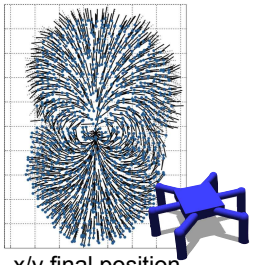


- It defines among two similar solutions, which one we will keep.

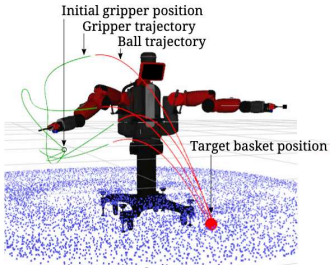
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. *Nature.*
Ecoffet, Huizinga, Lehman, Stanley, Clune (2019). Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995.*

QD algorithms: How does it work?
Behavioural Descriptor and Fitness functions

- 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
- Several solutions are likely to have the same behavioural descriptor, but with different fitness values



x/y final position

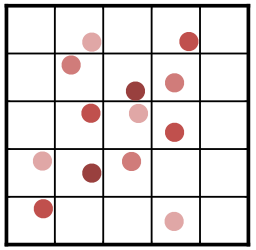


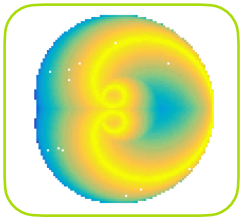
Key-points of the trajectory

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

QD algorithms: How does it work?
The container

- **The grid-based container**
  - 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



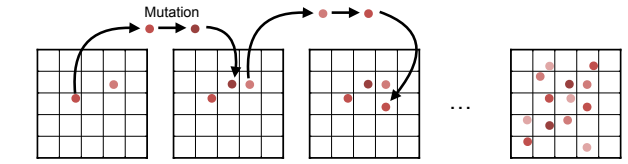


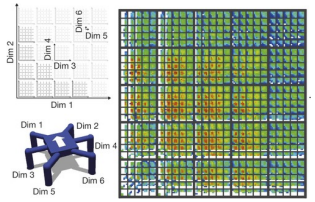
Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909.*

QD algorithms: How does it work?

One specific instance: MAP-Elites

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

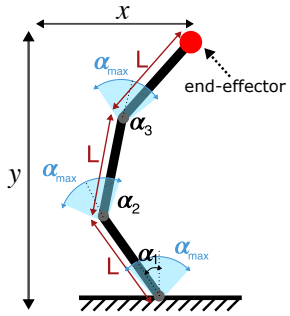


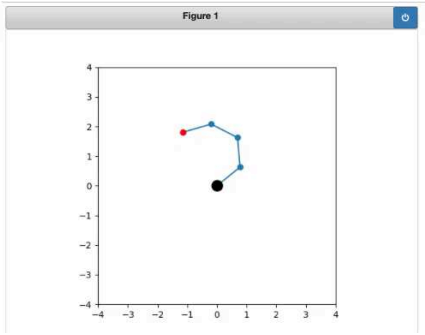


Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.
Cully, Clune, Tarapore, Mouret (2015). Robots that can adapt like animals. *Nature*.

Example: planar arm

see notebook



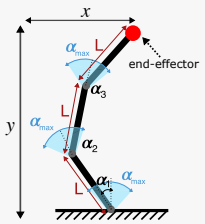


- **Search space:** $[\alpha_1, \cdots, \alpha_n]$ (n-dimensional)
- **Behavior space:** (x,y) (2-dimensional)

[**Notebook**] https://github.com/jbmouret/map_elites_tutorial/blob/main/map_elites.ipynb
[**collab**] https://colab.research.google.com/github/jbmouret/map_elites_tutorial/blob/main/map_elites.ipynb

Example: planar arm

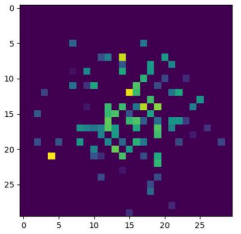
```python
def fitness(genotype):
    # fitness is the standard deviation of joint angles (Smoothness)
    # (we want to minimize it)
    fit = 1 - np.std(genotype)

    # now compute the behavior
    #   scale to [0,2pi]
    g = np.interp(genotype, (0, 1), (0, 2 * math.pi))
    j = forward_kinematics(g, [1]*len(g))
    #  normalize behavior in [0,1]
    b = (j[-1,:]) / (2 * len(g)) + 0.5
    return fit, b# the fitness and the position of the last joint
```



[**Notebook**] https://github.com/jbmouret/map_elites_tutorial/blob/main/map_elites.ipynb
[**collab**] https://colab.research.google.com/github/jbmouret/map_elites_tutorial/blob/main/map_elites.ipynb

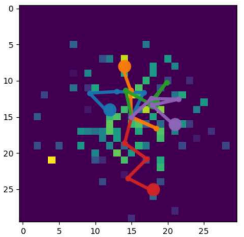
Example: planar arm

```python
# for simplicity, this is 2-Dimensional MAP-Elites
cols = 30
rows = 30
num_random = 100
num_dofs = 4

# we should use a dataclass, but this 3.9+
class Species:
    def __init__(self, genotype, behavior, fitness, niche=[]):
        self.genotype = genotype
        self.behavior = behavior
        self.fitness = fitness
        self.niche = niche

def add_to_archive(archive, species):
    n = species.behavior * np.array([rows, cols])
    x,y = min(round(n[0]), rows-1), min(round(n[1]), cols-1)
    if (not (x,y) in archive) or (archive[(x,y)].fitness < species.fitness):
        archive[(x,y)] = species
        species.niche = (x,y)
```



Archive initialization

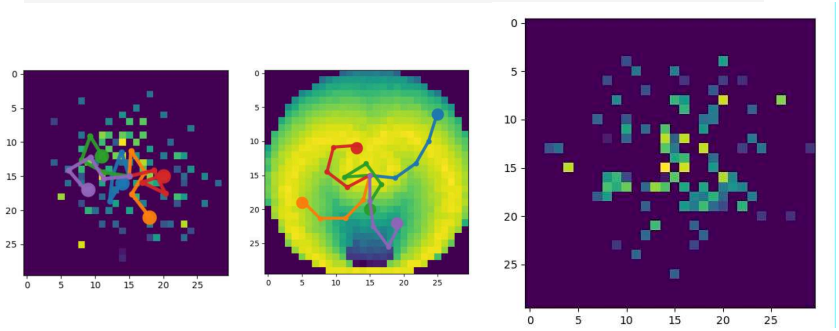
```python
def init_archive(num_random):# fill the archive with some random solutions
    # create an archive: a dictionnary indexed by coordinates
    archive = {}
    for i in range(0, num_random):
        g = np.random.rand(num_dofs)
        f, b = fitness(g)
        add_to_archive(archive, Species(g, b, f))
    return archive
```



[**Notebook**]
[**collab**]

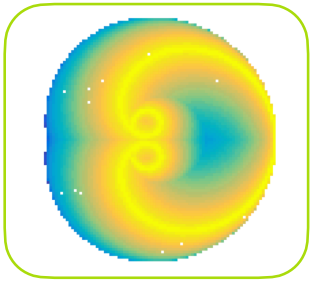
Example: planar arm

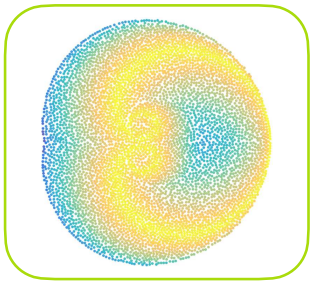
```
num_iterations = 50
batch_size = 500

for j in tqdm(range(0, num_iterations)):
    display_archive(archive)
    for i in range(0, batch_size):
        # pick an existing random point in the archive
        x = random.choice(list(archive.values()))
        # mutate it (we can use more advanced variation techniques: NEAT, etc.)
        g = x.genotype + np.random.normal(0, 0.1, x.genotype.shape[0])
        # compute the fitness
        f, b = fitness(g)
        # add to archive
        add_to_archive(archive, Species(g, b, f))
```



QD algorithms: How does it work?
Different container types

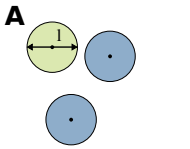
- There are two different ways to store the solutions in QD:
  - The grid (from MAP-Elites)
  - The unstructured archive (from Novelty Search)





Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*
Mouret, Clune (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909.*
Lehman, Stanley (2010). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation Journal.*

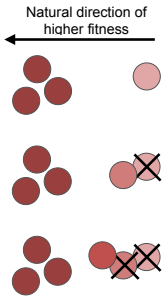
QD algorithms: How does it work?
Different container types



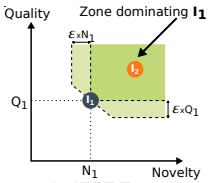
- **The unstructured archive**
  - Based on the distance between solutions in behavioural descriptor space
  - Main principle of the addition mechanism:
    - Each new solution goes to its exact location in the behavioural descriptor space
    - If the nearest solution already in the archive is further than a predefined value “l”, then the solution is added (case **A**)
    - Otherwise, only the best of the two overlapping solutions it kept in the archive (case **B**)

Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*
Lehman, Stanley (2010). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation.*

QD algorithms: How does it work?
Different container types



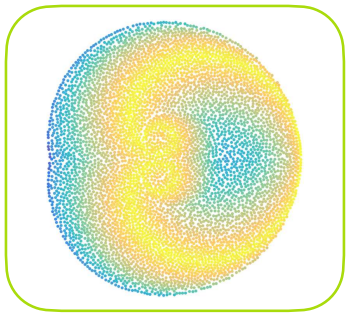
- **The unstructured archive**
  - In practice, the addition mechanism needs to be more complex
  - The erosion problem:
    - In certain cases, for instance when it is easier to have a high fitness in the center of the archive, the structure of the fitness might cause solutions with a high fitness to frequently remove solutions that were novel.
  - To solve this problem we use ε-dominance:
    - A solution replaces an existing one if:
      - it has a better fitness and a higher novelty than the existing one
      - Or, it is making an higher-improvement on the quality or novelty than the decrease it is causing on the other score (up to a certain value)



Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*
Lehman, Stanley (2010). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation Journal.*

QD algorithms: How does it work?
Different container types

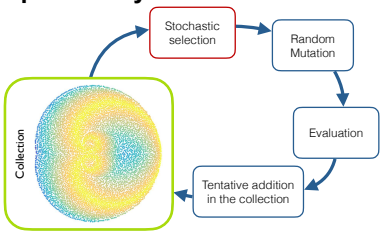
- **The unstructured archive**
  - Hyper-parameter: l value (comparable to cell size) and k (to compute the novelty score.
  - Advantages:
    - The archive can have an arbitrary shape
    - Maximal density set by *l*
  - Drawbacks:
    - Implementation more complex
    - More hyper-parameters



Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*
Lehman, Stanley (2010). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation Journal.*

QD algorithms: How does it work?
Different selector types

- **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

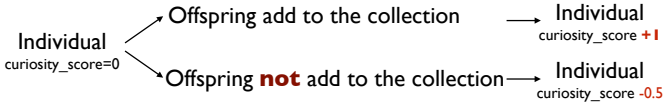


Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*

QD algorithms: How does it work?
Different selector types

Definition III.1: Curiosity Score

The curiosity score represents the propensity of an individual to generate offspring that are added to the collection.

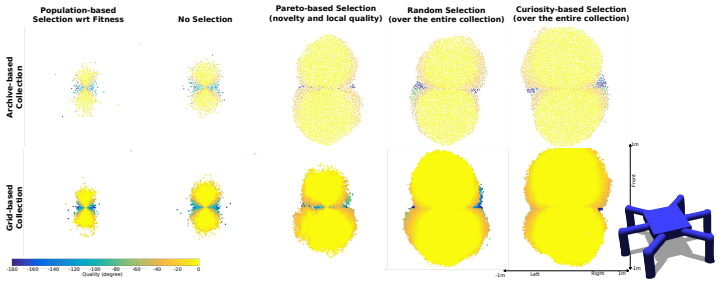


- The score dynamically captures solutions that are likely to generation offspring that will improve the archive.

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

QD algorithms: How does it work?
Different selector types

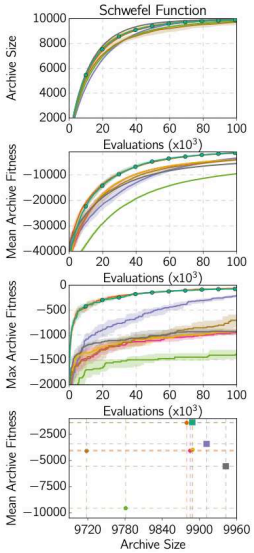
- Other approaches exists, for instance using a population of solutions in parallel to the archive and using this population for the selection.
- However, it has been shown that selecting directly from the archive is better.
- Certain bias, like the novelty or fitness, might cause undesired effects.



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

Quantifying performance

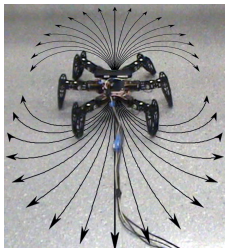
- Different aspects used to compare QD algorithms:
  - The **diversity** of the solutions in the container
    - Usual metric: archive size
  - The **performance** of the solution in the container
    - Usual metric: Max, or mean fitness value
  - The **convergence speed** of these two points.
  - Often, these different information are 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

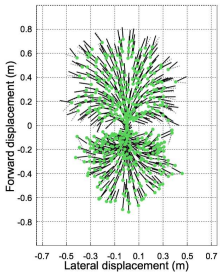


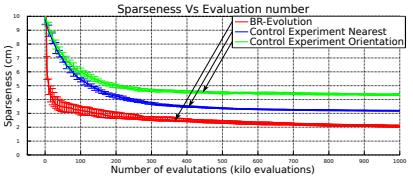
Pugh, Soros, Stanley. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI.*
Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Trans. in Evolutionary Computation.*
Vassiliades, Mouret (2018). Discovering the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

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





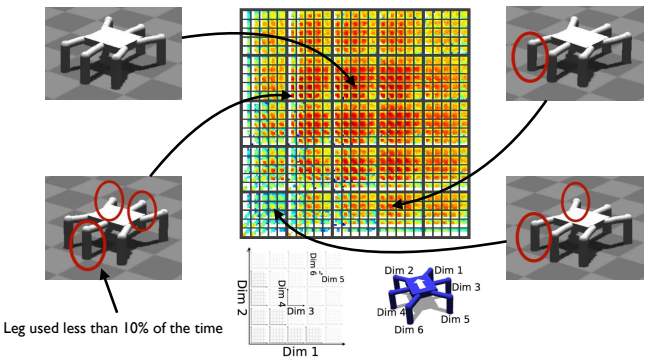


Learning several solutions simultaneously is more effective than learning them one by one.

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

Examples of applications:
Discovering multiple ways to walk as fast as possible

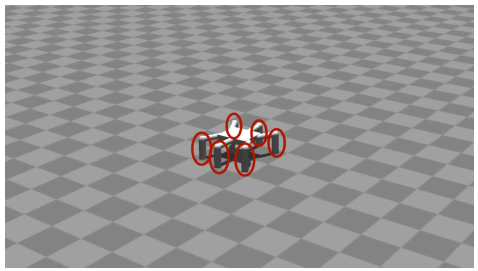
- **QD algorithm:** MAP-Elites (grid + random uniform selector)
- **Behavioural descriptor:** proportion of time that each leg is in contact with the ground (6D)
- **Fitness:** Traveled distance in 5 seconds



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

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:

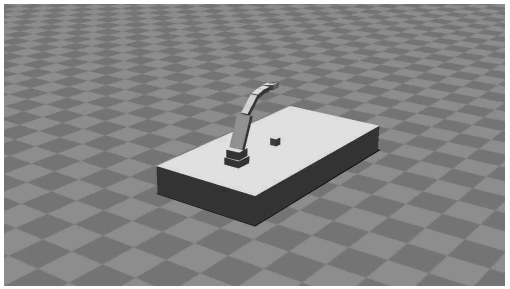


The robot autonomously learned to flip on its back and walk on its knees

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)

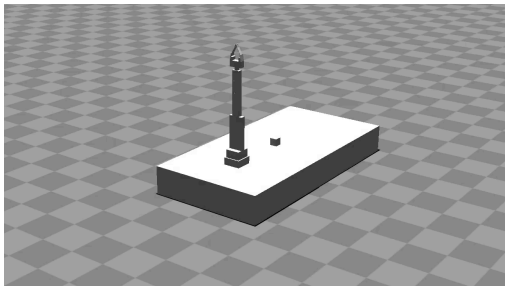
- **QD algorithm:** MAP-Elites (grid + random uniform selector)
- **Behavioural descriptor:** final position of the cube
- **Fitness:** Energy efficiency of the movement
- Gripper is forced in close position



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)

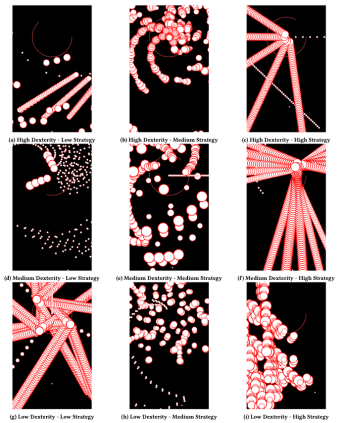
- **QD algorithm:** MAP-Elites (grid + random uniform selector)
- **Behavioural descriptor:** final position of the cube
- **Fitness:** Energy efficiency of the movement
- Gripper is forced in close position

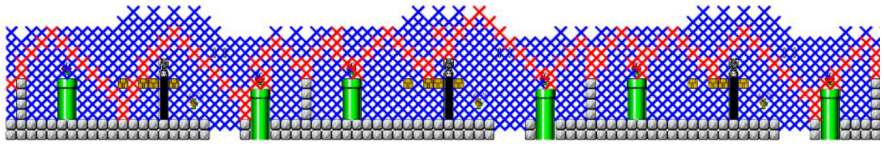


Ecarlat, et al. (2015). Learning a high diversity of object manipulations through an evolutionary-based babbling." *IROS. 2015.*

Examples of applications:
Content generation in video games

- Use MAP-Elites to generate a variety of levels for video games like bullet hell games or Super Mario.

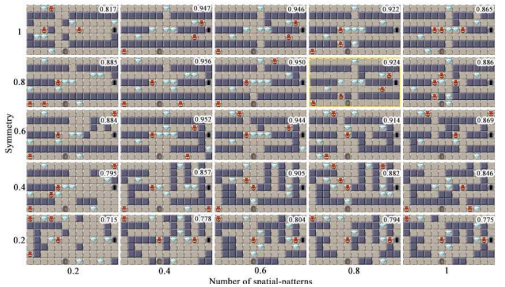


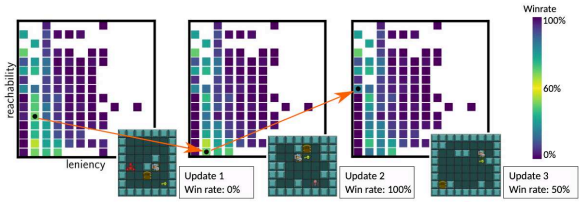


Khalifa, Lee, Nealen, Togelius (2018). Talakat: Bullet hell generation through constrained map-elites. *Proc. of GECCO.*
Schrum, Volz, Risi *(2020). CPPN2GAN: Combining compositional pattern producing networks and gans for large-scale pattern generation. Proc. of GECCO.*

Examples of applications:
Content generation in video games

- This diversity of levels can be used by game designers (interactive MAP-Elites).
- Or directly by games to automatically adjust the difficulty to the players

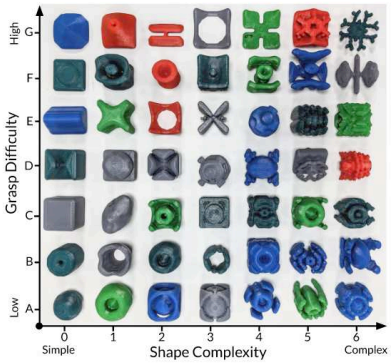


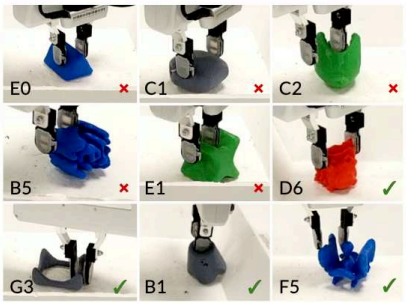


Alvarez, Dahlskog, Font, Togelius (2019). Empowering quality diversity in dungeon design with interactive constrained MAP-Elites. *In 2019 IEEE Conference on Games (CoG).*
González-Duque, Palm, Ha, Risi *(2020). Finding Game Levels with the Right Difficulty in a Few Trials through Intelligent Trial-and-Error. Proc. of IEEE Conference on Games (CoG)*

Examples of applications:
Generation of examples for robot training

- Generating a collection of diverse shapes for grasping tasks with different level of difficulty.

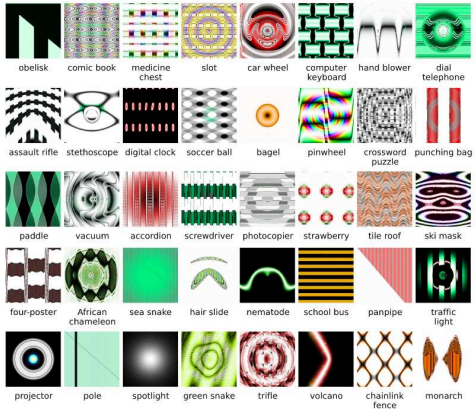




Morrison, Corke, Leitner (2020). Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*.

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:

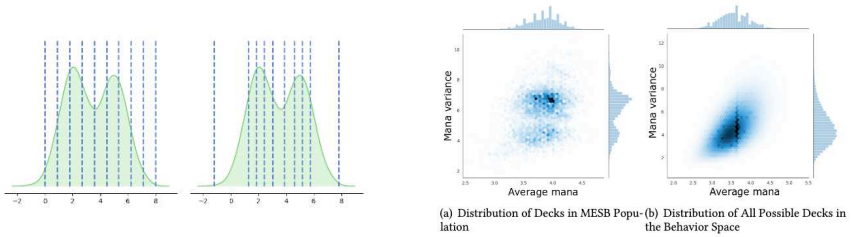
- https://quality-diversity.github.io
  A community website to gather a list of paper
- Add your papers to the list!

More recent concepts

- The recent popularity of QD algorithms led to several research directions to push them further.
  - Advanced containers
    - Containers with sliding boundaries
    - Hierarchical containers
  - Automatic BD definition from high-dimensional data
    - Pre-learning of BD
    - Online learning of BD
    - Meta-learning of BD

More recent concepts
Advanced containers

- MAP-Elites with sliding boundaries
- Instead of using cells of fixed size, the cells can be adjusted based on the distribution of evolved individuals.
- A buffer of the last N evaluated individuals is maintained in a queue data structure.
- Periodically, the boundary lines for the map are recalculated.

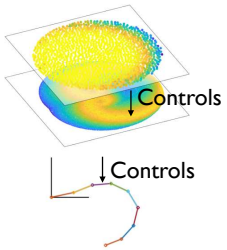


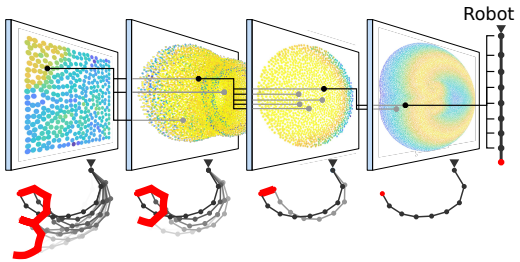
(a) Distribution of Decks in MESB Population (b) Distribution of All Possible Decks in the Behavior Space

Fontaine, Lee, Soros, De Mesentier Silva, Togelius, Hoover (2019). Mapping hearthstone deck spaces through MAP-elites with sliding boundaries. *Proc. of GECCO.*

More recent concepts
Hierarchical BR

- The diversity of behaviours in an archive can form a space of primitive solutions
- Combining multiple primitives can lead to more advanced solutions
- We can use QD to learn a repertoire of such advanced solutions too.
- Advantage: It reduces the dimensionality of the optimisation problems.

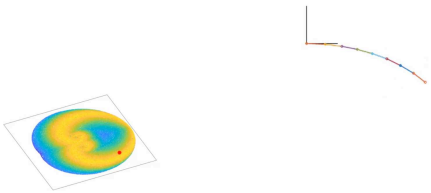




Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. Proc. of GECCO.

More recent concepts
Hierarchical BR

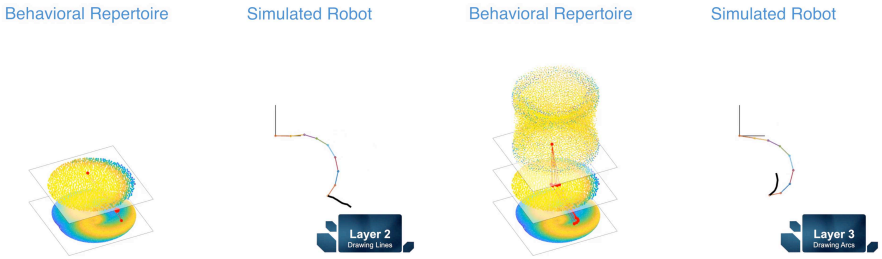
- **Layer 1:**
  - Robotic arm with 8 degrees of freedom:
  - **Controller**: the final angular position of each motor (8 parameters).
  - **Behavioural descriptor**: Final position of the robot's gripper (2 dimensions).
  - **Fitness**: minimising variance of the angular positions.



Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. *Proc. of GECCO.*

More recent concepts
Hierarchical BR

- **Layer 2:** Drawing Lines
  - **Controller:** deltaX deltaY to be done in BD
  - **Behavioural descriptor:** Length and direction of the line
- **Layer 3:** Drawing Arcs and Circles
  - **Controller:** 5 line behaviours (10 parameters).
  - **Behavioural descriptor:** Expected center, radius and length of the arc (3 dimensions)

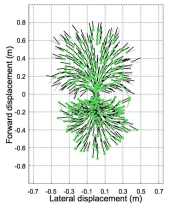


Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. *Proc. of GECCO.*

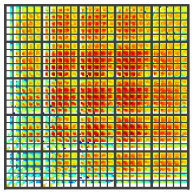
More recent concepts
Automatic BD definition

❖ The BD definition requires a certain expertise. Several works attempts to remove this requirement

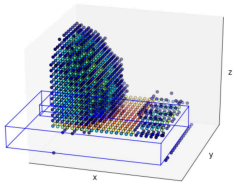
Final position of the Robot (X, Y)



Percentage of time that each leg is in contact with the ground



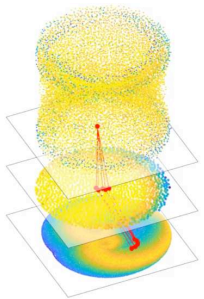
Final position of the cube (X, Y, Z)



Radius, center and length of the arc

Length and direction of the line

Final position of the gripper (X, Y)

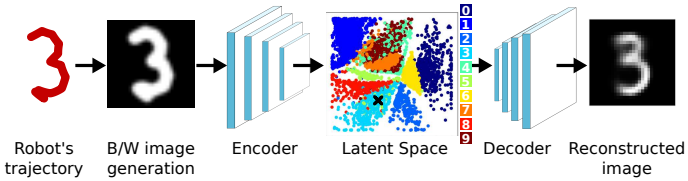


More recent concepts
Automatic BD definition

❖ If a dataset of features from the expected solution exists,

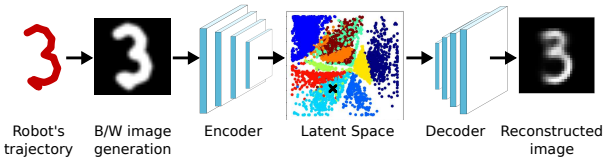
- A dimensionality reduction algorithm (PCA or Auto-Encoder) can automatically learn a low-dimensional representation of these features that can serve as a BD-space.

❖ Example: if we want trajectories that look like digits, we can use a dataset of hand-written digits and learn latent space that will be use as a BD-space



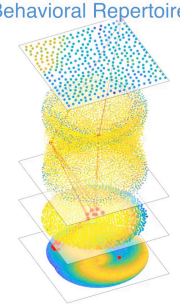
Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. *Proc. of GECCO.*

More recent concepts
Automatic BD definition



❖ This works with the Hierarchical BD context too:

Behavioral Repertoire



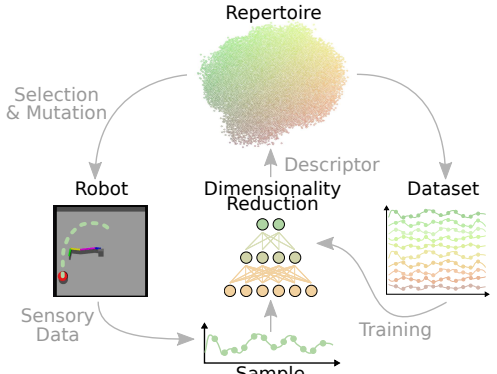
Simulated Robot

Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. *Proc. of GECCO.*

More recent concepts
Automatic BD definition

❖ Alternatively, the **AURORA** algorithm proposes to learn the BD space during the QD evolutionary process:

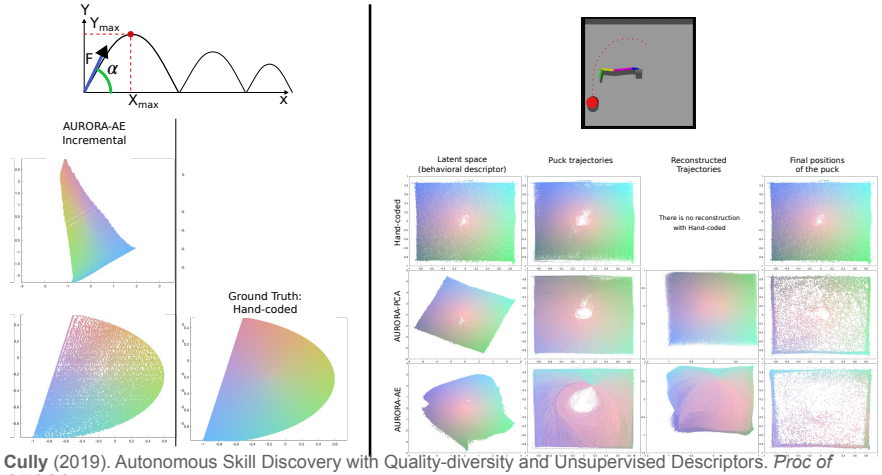
- The information contained in the archive is used as dataset to train the dimensionality reduction algorithm
- AURORA executes a few QD steps and a few training steps of the Auto-Encoder.



Cully (2019). Autonomous Skill Discovery with Quality-diversity and Unsupervised Descriptors. *Proc of GECCO.*

More recent concepts
Automatic BD definition

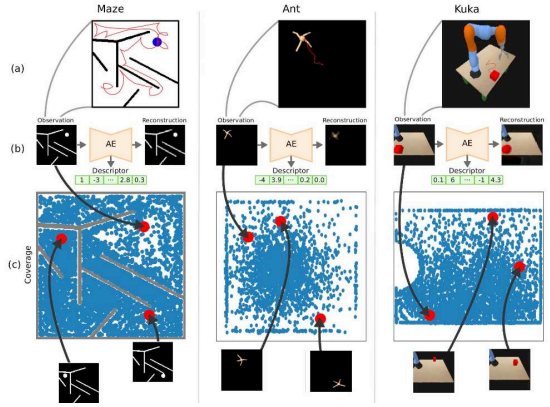
- **AURORA** makes the generation of the descriptor automatic
- It can be used to automatically discover the capabilities of robots.



Cully (2019). Autonomous Skill Discovery with Quality-diversity and Unsupervised Descriptors. *Proc of GECCO.*

More recent concepts
Automatic BD definition

- The **TAXONS** algorithm extends this concept to high-dimensional data (images) and introduces a new selection mechanism based on the reconstruction error.



Paolo, Laflaquiere, Coninx, Doncieux. (2019). Unsupervised Learning and Exploration of Reachable Outcome Space. *algorithms.*

More recent concepts
Automatic BD definition

- **Learning behaviour-performance maps with meta-evolution**
- The optimal BD definition can be learned by meta-learning on meta objective (for instance, adaptation capability).
- CMA-ES is used to generate potential linear combinations of BD definitions that are then used to evolve repertoires. These repertoires are evaluated on a meta objective which guides the CMA-ES population towards better combinations.
- To mitigate the expensive evaluation cost, a database stores all the solutions that have been evaluated from the beginning (genotype, fit, and all BDs). Thanks to that, maps with new combinations can easily be re-created, which enables them to some meta evolution for a reasonable cost.
- (Dedicated talk at GECCO'20)

Bossens, Tarapore, Mouret (2020). Learning behaviour-performance maps with meta-evolution. Proc. of GECCO.

Scaling-up quality diversity algorithms

- Scaling-up to more complex tasks or accelerating QD
  - ➔ *non-uniform selection of the parents:* better parents
    - ➔ *cf* selectors for curiosity, etc.
  - ➔ *better variation operators:* fewer mutations
  - ➔ *surrogate models:* fewer calls to the fitness
- Scaling up to high-dimensional diversity spaces *(behavioral space)*
  - ➔ distance-based archive
  - ➔ Centroidal Voronoi grid (CVT)

variation operators

Scaling-up: better operators

- Mutation and cross-over operators have a long history in evolutionary computation
- One of the most successful approach for optimization is CMA-ES (and derivatives)
  - resample a population at each step (no cross-over/mut.)
  - adapts the mutation strengths for each dimension by adapting the covariance matrix
- Can we do the same for QD algorithms (e.g., MAP-Elites)?

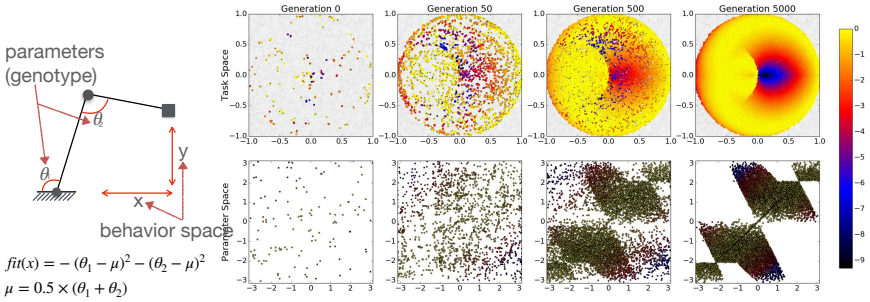
  ➔ It is not straightforward: each niche/cell follows a slightly different gradient

  ➔ We want to derive some knowledge from the whole map

Hansen, Müller, Koumoutsakos (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation.*

variation operators

Looking at the genome: is there a pattern?



$fit(x) = -(\theta_1 - \mu)^2 - (\theta_2 - \mu)^2$

$\mu = 0.5 \times (\theta_1 + \theta_2)$

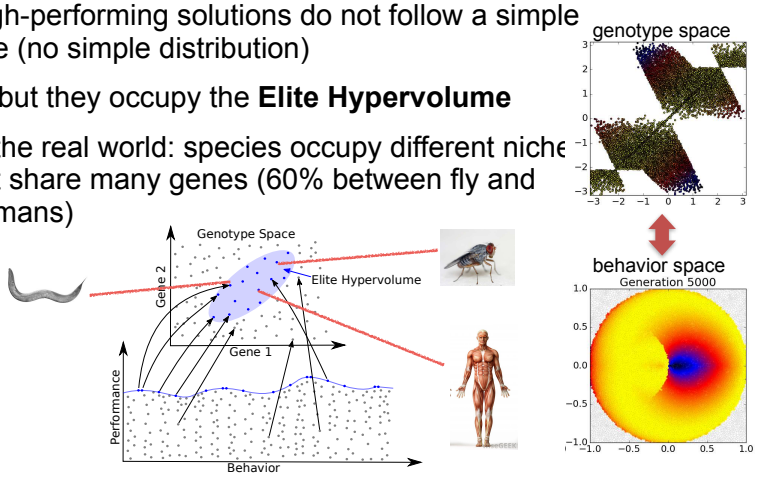
- Well spread in the task space ≠ well spread in genome!
- There is a structure in the genome space: how to exploit it?

Vassiliades, Mouret (2018). Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. *Proc. of GECCO.*

variation operators

The Elite Hypervolume

- High-performing solutions do not follow a simple rule (no simple distribution)
- … but they occupy the **Elite Hypervolume**
- In the real world: species occupy different niche but share many genes (60% between fly and humans)



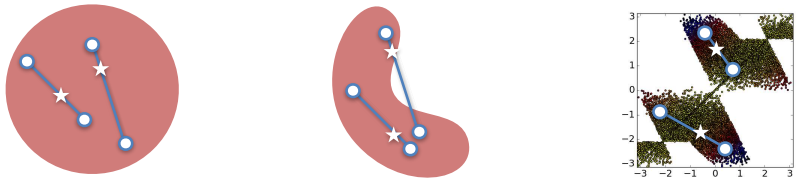
Adams, et al. (2000). The genome sequence of Drosophila melanogaster. *Science*.
Vassiliades, Mouret (2018). Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. *Proc. of GECCO.*

variation operators

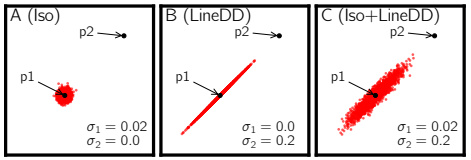
Leveraging the hypervolume

- What is a good variation operator?

  ➔ highly likely to generate an individual in the elite hypervolume



- if we take two points from a convex volume, any point on the segment is in the volume too



$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \sigma_1 \mathcal{N}(\mathbf{0}, \mathbf{I}) + \sigma_2 (\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}) \mathcal{N}(0, 1)$$

random perturbation for each dimension

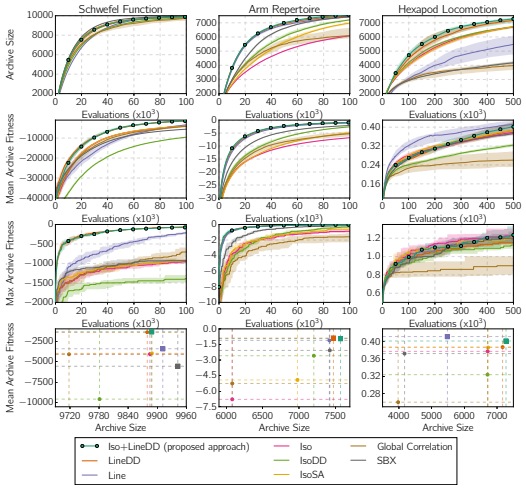
weight a single perturbation by the difference between the parents

➔ directional variation (~ cross-over)

Vassiliades, Mouret (2018). Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. *Proc. of GECCO.*

variation operators

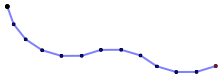
Directional variation



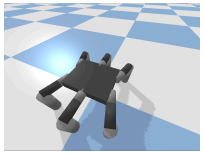
Schwefel function: 12-D ➔ 2D

$$f(\mathbf{y}) = -\sum_{i=1}^{n}\left(\sum_{j=1}^{i} y_j\right)$$

Arm: 12-D ➔ 2D



Hexapod: 36-D ➔ 6-D



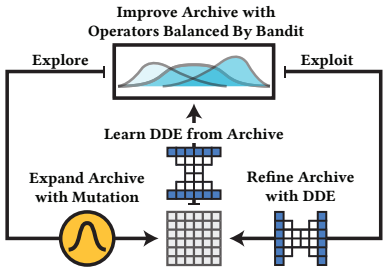
➔ Cross-over very useful in archive-based QD algorithms!

Vassiliades, Mouret (2018). Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. *Proc. of GECCO.*

variation operators

Learning the hypervolume: the DDE

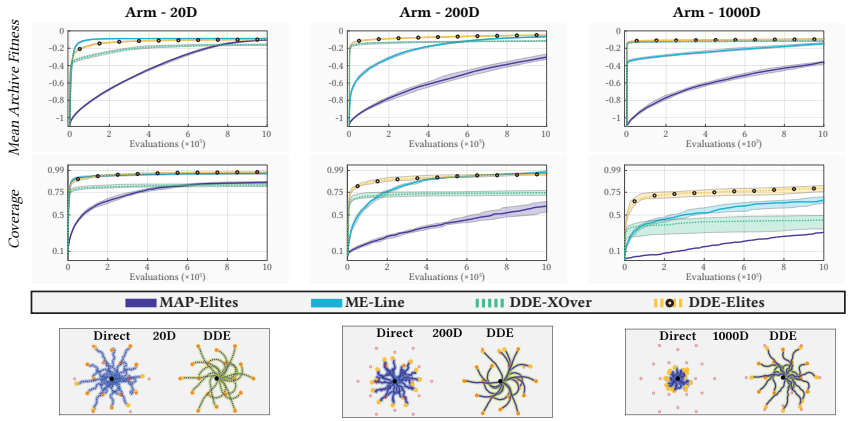
- Concept: learn the shape of the hypervolmme with a variational auto-encoder
- generate new individuals that follows “the recipe” of successful individiduals
- Problem: the auto-encoder (esp. at the beginning) cannot find solutions that do not follow the rules (yet)
- exploration/exploitation trade-off for the representation



Gaier, Asteroth, Mouret (2020). Discovering Representations for Black-box Optimization. *Proc. of GECCO.*

variation operators

Learning the hypervolume: the DDE



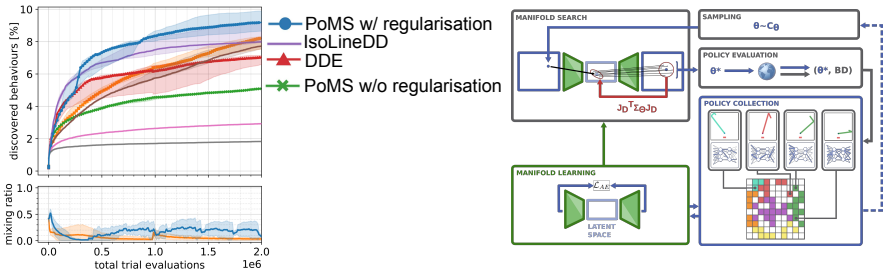
- Unknown if it works well on other tasks (with less regularity)
- The resulting auto-encoder can be used as a encoding for future optimizations

Gaier, Asteroth, Mouret (2020). Discovering Representations for Black-box Optimization. *Proc. of GECCO.*

variation operators

Learning the hypervolume: PoMS

- Policy Manifold Search (PoMS): Explicitly encoding the hypervolume with AE
- The latent space can be directly explored as a learned generative encoding.
- However, mutations have to be regularised based on the decoder’s jacobian to avoid deleterious mutations.



Rakicevic, Cully, Kormushev (2021). Policy Manifold Search for Improving Diversity-based Neuroevolution. *Proc. of GECCO.*

variation operators

Evolutionary strategies

- Evolutionary strategies (fixed σ):
  - sample a new population *n* by adding Gaussian noise ϵ_i
  - update the mean of the population by weighting the perturbation by the fitness: $\theta_{t+1} = \theta_t + \frac{1}{n\sigma}\sum_{i=1}^{n} f(\theta_t^i)\epsilon_i$

fitness of i

perturbation applied

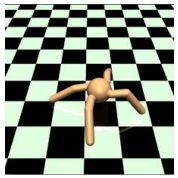
- Surprisingly effective for deep reinforcement learning
- ES for MAP-Elites: (alternate between)
  1. sample a cell with best novelty, optimize for novelty with ES
  2. sample a cell with best fitness, optimize for fitness with ES

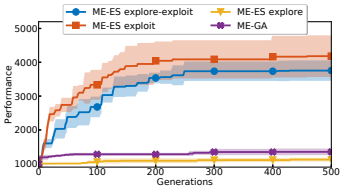
Colas, Huizinga, Madhavan, Clune (2020). Scaling MAP-Elites to Deep Neuroevolution. *Proc. of GECCO.*
Salimans, Ho, Chen, Sidor, Sutskever (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864.*

variation operators

Evolutionary strategies

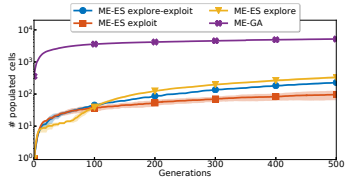
Controller: 2-layer neural network (256 nodes) ➔ 10^5 parameters
Behavior space: 4D (gaits)





mean fitness

- No comparison with directional mutation or cross-over (for now)
- More emphasis on exploitation



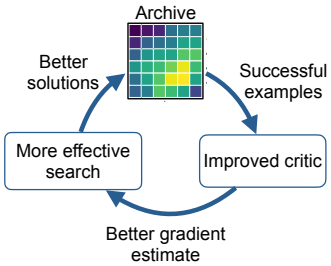
coverage

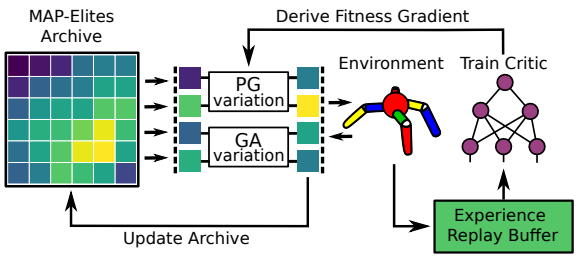
Colas, Huizinga, Madhavan, Clune (2020). Scaling MAP-Elites to Deep Neuroevolution. *Proc. of GECCO.*

variation operators

Policy Gradient Assisted MAP-Elites

- Combining Policy Gradient (TD3) and MAP-Elites
- GA variation operator: Undirected exploration
- PG variation operator: Directed exploration

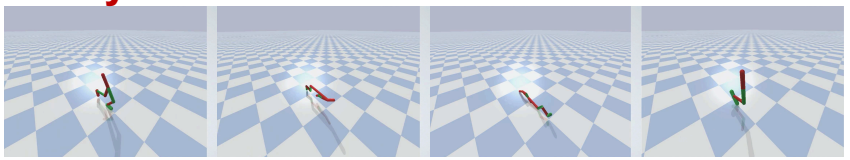




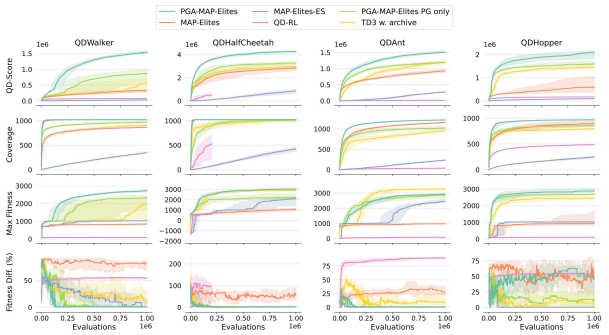
Nilsson, Cully (2021). Policy Gradient Assisted MAP-Elites. *Proc. of GECCO.*

variation operators

Policy Gradient Assisted MAP-Elites



- **QDGym**:
  Fitness: same as in the original Gym environments
  Behavioural Descriptor: proportion of time that each leg is in contact with the ground
- **Policy:**
  Feedforward Network
  [state dim., 128, 128, action dim.] ~ 20k parameters

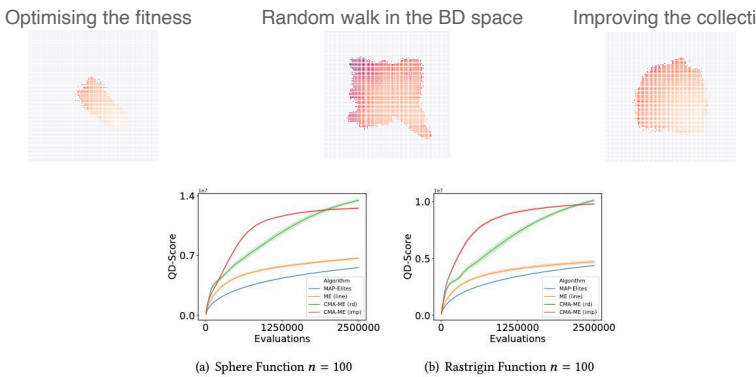


Nilsson, Cully (2021). Policy Gradient Assisted MAP-Elites. *Proc. of GECCO.*

variation operators

Emitter-based QD

- Emitters, introduced by Fontaine et al. are advanced mechanism to generate potential solutions in MAP-Elites
- Covariance Matrix Adaptation MAP-Elites (CMA-ME) proposes emitters based on CMA-ES to follow specific objectives:

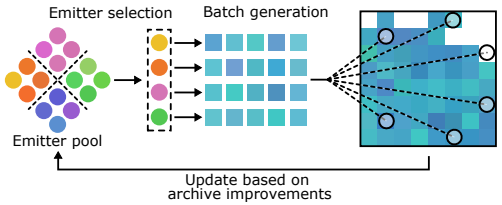


Fontaine, Togelius, Nikolaidis, Hoover (2020). Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space. *Proc. of GECCO.*

variation operators

Emitter-based QD

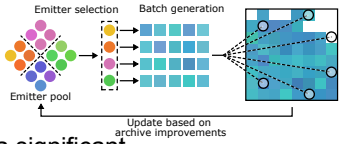
- Multi-Emitters MAP-Elites (ME-MAP-Elites) introduces heterogenous pools of emitters
- A bandit algorithm (UCB-1) picks a different set of emitters at each generation depending on their predicted effectiveness.
- ME-MAP-Elites combines the CMA-ME emitters to look for the local highest fitness, and randomly explore the BD space, but also adds an unbiased emitters that reproduce MAP-Elites's exploration mechanism.



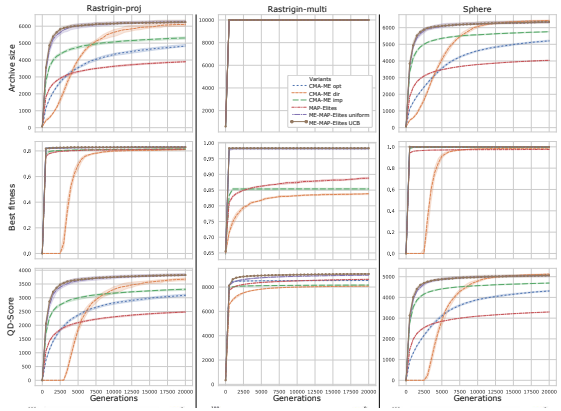
Cully (2021). Multi-Emitter MAP-Elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. *Proc. of GECCO.*

variation operators

Emitter-based QD



- Results show that ME-MAP-Elites offers a significant improvement of the coverage, quality and convergence speed



Cully (2021). Multi-Emitter MAP-Elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. *Proc. of GECCO.*

surrogate models

Surrogate models for quality diversity

- QD algorithms often need 100k to 1M evaluations to be effective
- This is not possible in many applications: robotics, aerodynamics, complex simulations
- … but QD is very interesting for engineering

> "We interviewed 18 architects and manufacturing design professionals. […] **Contrary to our expectations, we found that the computed optimum was often used as the starting point for design exploration, not the end product.** (Bradner, 2014 — Autodesk)"

- Classic solution in optimization: use a **surrogate model**
  ➔ use a few *expensive* "precise evaluations" to learn a *cheaper* predictor (e.g. a neural network) of the fitness
  ➔ use the predicted fitness in the algorithm

Bradner, Iorio, Davis (2014). Parameters tell the design story: Ideation and abstraction in design optimization. *Simulation Series.*

surrogate models

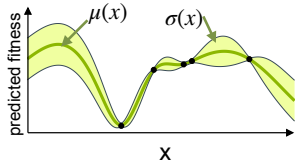
SAIL: Surrogate Assisted Elimination

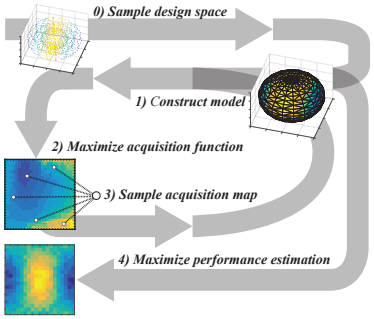
- Use Gaussian processes to model the fitness (mean and uncertainty)
- Create an **acquisition map** with the Upper Confidence Bound (UCB)

$$UCB(x) = \mu(x) + \beta\sigma(x)$$

predicted mean uncertainty

- Select solutions to be evaluated in the acquisition map (uniformly)
- At the end: create the final map with the mean of the model

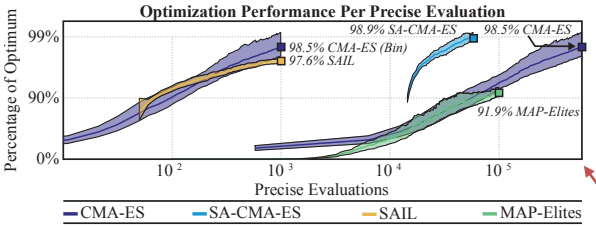


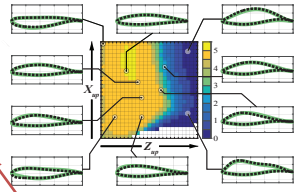


Gaier, Asteroth, Mouret (2018). Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation.*

surrogate models

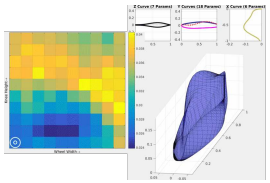
SAIL: Surrogate Assisted Elimination





Log scale!

- SAIL for the map = number of evaluations of CMA-ES for **one** cell
- Nice property: quickly generate a prediction map at any resolution!



- Challenges:
  - do not scale well with the number of parameters (hard to model if > 10)
  - Gaussian processes query is in $O(N^2)$ in the number of samples

Gaier, Asteroth, Mouret (2018). Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation.*

surrogate models

BOP-Elites

- Use a GP to predict the behavioral descriptor (the niche)

$$\mathbb{P}(x \in c|D) = \Phi\left(\frac{\bar{g}(x) - b_u}{s'(x)}\right) - \Phi\left(\frac{\bar{g}(x) - b_l}{s'(x)}\right)$$

- Use Expected Improvement for the second GP (fitness)

$$EI_c(x) = \mathbb{E}\left[max(f(x) - f(\hat{e}), 0)\right] \quad (4)$$

$$= (\bar{f}(x) - f(\hat{e}))\Phi\left(\frac{\bar{f}(x) - f(\hat{e})}{s(x)}\right) + s(x)\phi\left(\frac{\bar{f}(x) - f(\hat{e})}{s(x)}\right) \quad (5)$$

- Combine in an acquisition function:

$$EJIE(x) = \sum_{i=1}^{C} \mathbb{P}(x \in c_i) EI_{c_i}(x)$$

- Optimize EJIE to select the next point

Kent, Branke (2021). Bop-elites, a bayesian optimisation algorithm for quality-diversity search. *arXiv preprint arXiv:2005.04320. 2020 May 8. / GECCO 2021 poster*

Scaling up to high-dimensional behavioral spaces

- We mostly use QD in low-dimensional spaces (2D)
- … but there are many high-dimensional (< 2) behavioral descriptions
  - naturally high-dimensional problems
  - trajectories of robots
  - sensory-motor flux
- A grid in 6D, 5 bins per dimension: 5^6 cells = 15625
  … 12D, 5 bins per dimension: 5^{12} = 244 million
  … 36 D, 2 bins per dimensions: 2^{36} = 68 billions

➔ We need to make the number of bins/cells independent from the dimensionality!

Doncieux, Mouret (2010). Behavioral diversity measures for evolutionary robotics. In *IEEE congress on evolutionary computation.*

High-dimensional behavioral space

Option 1: Distance-based containers

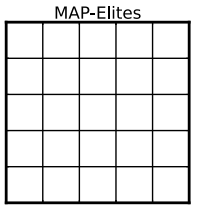
- Option 1: use a container based on distance (not a grid)

  (archive-based container vs grid-based container)
- A solution is added to the archive
  - if the distance to its nearest neighbor is > l
  - or if is better than its *k* nearest neighbors
- In practice use ε-dominance (see the modular framework section)
- **Drawbacks:** need to decide l (not always intuitive) and k
- **Benefits:** the behavior space can have any shape

Cully, Demiris (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation.*

High-dimensional behavioral space

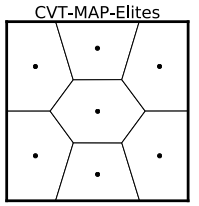
Option 2: Centroidal Voronoi Tesselation



- Classic way to create a grid in computer graphics: **Centroidal Voronoi Tesselation**

  ➔ specify the number of cells

  ➔ get the grid with cells of equal volumes
- Works in any dimension
- Already implemented in many libraries: only need k-means
- Needs to be generated once, then we can use any grid-based QD algorithm



Vassiliades, Chatzilygeroudis, Mouret (2017). Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation.*

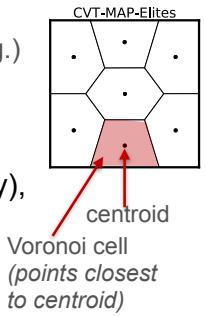
High-dimensional behavioral space

Implementing CVT-MAP-Elites

- Computing the CVT using the LLoyd relaxation:
  1. choose the number of cells (**k**)
  2. sample N random points (N >> number cells)
  3. run k-means on the points (classic clustering alg.) (*this uses the distance between the points*)

  ➔ get ***k*** centroids (sites)

  ➔ the drawing is a Delaunay diagram from the centroids (from any computer graphics library), but it is not needed for CVT-MAP-Elites



- Using the CVT in MAP-Elites:
  - *we need to know the closest centroid to know the cell*

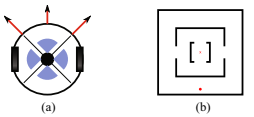
  ➔ in low-dimensional space: use a KD-tree (computer graphics)

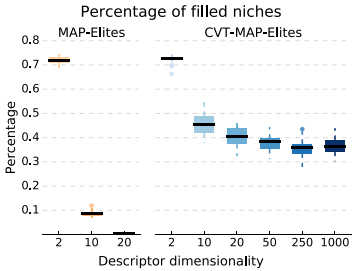
  ➔ in high-dimensional space (> 10): sort by distance

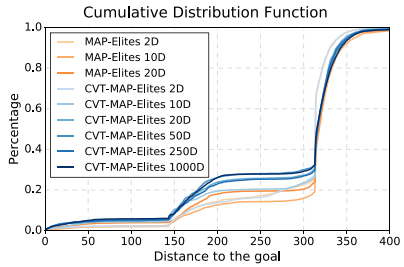
Vassiliades, Chatzilygeroudis, Mouret (2017). Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation.*

High-dimensional behavioral space

CVT-MAP-Elites







- Descriptor: sub-sampling of the trajectory
- Conclusion: almost no change in performance when we increase the dimensionality

Vassiliades, Chatzilygeroudis, Mouret (2017). Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation.*

Overview of QD implementations

- QD algorithms are easy to implement (this is a strength!)
  - easiest: grid-based MAP-Elites + directional mutation
  - more flexible: CVT-MAP-Elites
- Most research group implemented their own version
- That said, implementations can:
  - help you start
  - give you good baselines
  - give you ‘complex' options
  - be faster (better implementation, parallelization)

implementations

Python: notebook

- https://github.com/jbmouret/map_elites_tutorial
- Objective: learning the principles of QD
- Use this:
  - as a teaching tool
- Support:
  - 2-D grid-based MAP-Elites
  - nothing else!

implementations

Python: reference implementations

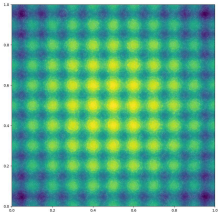
- https://github.com/resibots/pymap_elites
- Objective: **straightforward** implementations that are easy to transform (1 page of code for the algorithm)
- Use this:
  - as a library for MAP-Elites
  - as a **starting point** to ‘hack’ your ideas
- Support:
  - Parallel fitness evaluations (*multiprocessing* module)
  - CVT-MAP-Elites with directional mutation and KD-Tree
  - Multi-task MAP-Elites
  - “not bad” default settings
  - basic plotting (matplotlib)

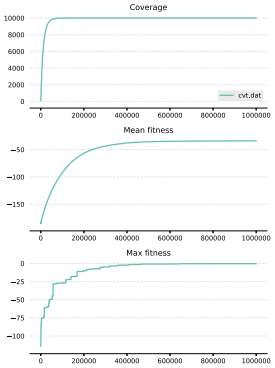
implementations

Example

```
import numpy as np
import math

import map_elites.cvt as cvt_map_elites
import map_elites.common as cm_map_elites

# the function to optimize
def rastrigin(xx):
    x = xx * 10 - 5 # scaling to [-5, 5]
    f = 10 * x.shape[0] + (x * x - 10 * np.cos(2 * math.pi * x)).sum()
    return -f, np.array([xx[0], xx[1]])

px = cm_map_elites.default_params.copy()
px["dump_period"] = 100000

# we use 1M evaluations (this is a hard function)
# map in 2D, genotype in 10D
archive = cvt_map_elites.compute(2, 10, rastrigin,
n_niches=10000, max_evals=1e6, log_file=open('cvt.dat', 'w'), params=px)
```

```
> python3 examples/cvt_rastrigin.py
> python3 plot/plot_progress.py cvt.dat
> python3 plot/plot_2d_map.py centroids_10000_2.dat archive_1000100.dat
```





implementations

Reference for CVT-MAP-Elites

```
# directional variation
def iso_dd(x, y, params):
    assert(x.shape == y.shape)
    p_max = np.array(params["max"])
    p_min = np.array(params["min"])
    a = np.random.normal(0, params['iso_sigma'], size=len(x))
    b = np.random.normal(0, params['line_sigma'])
    z = x.copy() + a + b * (x - y)
    return np.clip(z, p_min, p_max)

def __add_to_archive(s, centroid, archive, kdt):
    niche_index = kdt.query([centroid], k=1)[1][0][0]
    niche = kdt.data[niche_index]
    n = cm.make_hashable(niche)
    s.centroid = n
    if n in archive:
        if s.fitness > archive[n].fitness:
            archive[n] = s
            return 1
        return 0
    else:
        archive[n] = s
        return 1

# map-elites algorithm (CVT variant)
def compute(dim_map, dim_x, f,
            n_niches=1000,
            max_evals=1e5,
            params=cm.default_params,
            log_file=None,
            variation_operator=iso_dd):
    """
    # setup the parallel processing pool
    num_cores = multiprocessing.cpu_count()
    pool = multiprocessing.Pool(num_cores)

    # create the CVT
    c = cm.cvt(n_niches, dim_map,
               params['cvt_samples'], params['cvt_use_cache'])
    kdt = KDTree(c, leaf_size=30, metric='euclidean')
    cm.__write_centroids(c)

    archive = {} # init archive (empty)
    n_evals = 0 # number of evaluations since the beginning
    b_evals = 0 # number evaluation since the last dump
    # main loop
    while (n_evals < max_evals):
        to_evaluate = []
        # random initialization
        if len(archive) <= params['random_init'] * n_niches:
            for i in range(0, params['random_init_batch']):
                x = np.random.uniform(low=params['min'], \
                        high=params['max'], size=dim_x)
                to_evaluate += [(x, f)]
        else:  # variation/selection loop
            keys = list(archive.keys())
            # we select all the parents at the same
            # time because randint is slow
            rand1 = np.random.randint(len(keys), \
                    size=params['batch_size'])
            rand2 = np.random.randint(len(keys), \
                    size=params['batch_size'])
            for n in range(0, params['batch_size']):
                # parent selection
                x = archive[keys[rand1[n]]]
                y = archive[keys[rand2[n]]]
                # copy & add variation
                z = variation_operator(x.x, y.x, params)
                to_evaluate += [(z, f)]
        # evaluation of the fitness for to_evaluate
        s_list = cm.parallel_eval(__evaluate, to_evaluate, pool,
params)
        # natural selection
        for s in s_list:
            __add_to_archive(s, s.desc, archive, kdt)
        # count evals
        n_evals += len(to_evaluate)
        b_evals += len(to_evaluate)


    return archive
```

implementations

Python: qdpy

- pip install qdpy or https://gitlab.com/leo.cazenille/qdpy
- “framework”, implements: MAP-Elites, CVT-MAP-Elites, NSLC, SAIL, …
- a good choice if you do not plan to “hack" the implementations

```
from qdpy import algorithms, containers, benchmarks, plots

# Create container and algorithm. Here we use MAP-Elites, by illuminating a Grid container by evolution.
grid = containers.Grid(shape=(64,64), max_items_per_bin=1, fitness_domain=((0., 1.),),
   features_domain=((0., 1.), (0., 1.)))
algo = algorithms.RandomSearchMutPolyBounded(grid, budget=60000, batch_size=500,
        dimension=3, optimisation_task="maximisation")

# Create a logger to pretty-print everything and generate output data files
logger = algorithms.AlgorithmLogger(algo)

# Define evaluation function
eval_fn = algorithms.partial(benchmarks.illumination_rastrigin_normalised,
        nb_features = len(grid.shape))

# Run illumination process !
best = algo.optimise(eval_fn)

# Print results info
print(algo.summary())

# Plot the results
plots.default_plots_grid(logger)

print("All results are available in the '%s' pickle file." % logger.final_filename)
```

Python: pyRIBS

- https://pyribs.org
- pip install ribs
- official implementation of Covariance Matrix Adaptation MAP-Elites (CMA-ME)

```
import numpy as np

for itr in range(1000):
    solutions = optimizer.ask()

    # Optimize the 10D negative Sphere function.
    objectives = -np.sum(np.square(solutions), axis=1)

    # BCs: first 2 coordinates of each 10D solution.
    bcs = solutions[:, :2]

    optimizer.tell(objectives, bcs)
```

implementations

Sferes$_{v2}$: C++

- https://github.com/sferes2/sferes2
- Implements the generic framework for QD
- Template-based C++11, parallelization with TBB or MPI
- Fast: 1 minute for 1M Rastrigin (CVT-ME) vs 5 minutes in Python
- Complex genotypes (neuro-evolution)
- Experimental framework (variants, cluster submission, etc.)
- Highly customizable:

```
// defines MAP-Elites (grid container, uniform selection)
template <typename Phen, typename Eval, typename Stat, typename Modifier, typename Params>
using MapElites
   = qd::QualityDiversity<Phen, Eval, Stat, Modifier,
           selector::Uniform<Phen, Params>, container::Grid<Phen, Params>, Params>;

// defines CVT-MAP-Elites (CVT container, uniform selection)
template <typename Phen, typename Eval, typename Stat, typename Modifier, typename Params>
using CvtMapElites
    = qd::QualityDiversity<Phen, Eval, Stat, Modifier, selector::Uniform<Phen, Params>,
              container::CVT<Phen, container::SortBasedStorage<int>, Params>, Params>;
```

Mouret, Doncieux (2010). Sferes$_{v2}$: Evolvin’ in the multi-core world. In *IEEE Congress on Evolutionary Computation.*

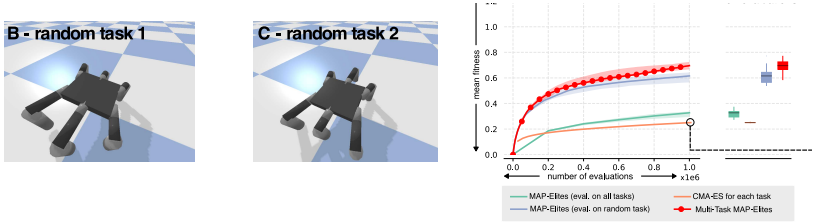
Sferes$_{v2}$: C++

```
FIT_QD(Rastrigin){
    public :
    template <typename Indiv>
    void eval(Indiv & ind){
        // TODO compute Rastrigin function
        this->_value = -f;
        std::vector<double> data = {ind.gen().data(0), ind.gen().data(1)};
        this->set_desc(data);
    }
};
int main(int argc, char **argv)
{
    using namespace sferes;

    typedef Rastrigin<Params> fit_t;
    typedef gen::EvoFloat<10, Params> gen_t;
    typedef phen::Parameters<gen_t, fit_t, Params> phen_t;

    typedef eval::Parallel<Params> eval_t;

    typedef boost::fusion::vector<
        stat::BestFit<phen_t, Params>,
        stat::QdContainer<phen_t, Params>,
        stat::QdProgress<phen_t, Params>
        >
        stat_t;
    typedef modif::Dummy<> modifier_t;
    typedef qd::MapElites<phen_t, eval_t, stat_t, modifier_t, Params>
        qd_t;

    qd_t qd;
    run_ea(argc, argv, qd);
   return 0;
}
```

Open questions and challenges

❖ **Relationship between QD and multitask optimization**

- Example of MT: learning to grasp *k* different objects
- QD: f(x) ➔ f, b Multi-task: f(x, t) ➔ f
- QD algorithms can be modified for multi-task
- Multi-task more general, but QD exploits a "trick" that makes it effective

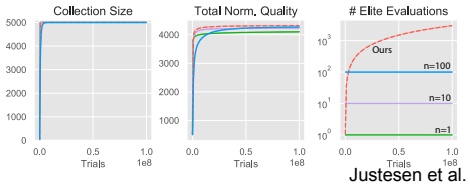


Mouret, Maguire (2020). Quality Diversity for Multi-task Optimization. *Proc of GECCO.*

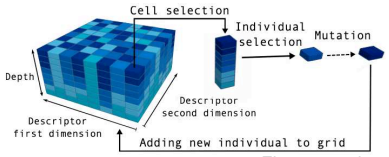
Open questions and challenges

❖ **Noise in the fitness…and in the behavior**

- Many fitness functions are noisy (robots, games)
- Elitist algorithms (like MAP-Elites) do not tolerate noise well
- Classic EC: noisy fitness, QD: noise in fitness AND behavior
- Justesen et al.: works well for noisy fitness, not yet for noisy behaviors
- Flageat et al.: works well for noisy behaviors and fitness, but might miss the best solutions (lack of elitism)



Justesen et al.



Flageat et al.

Justesen, Risi, Mouret (2019). MAP-Elites for noisy domains by adaptive sampling. *GECCO 2019 companion (poster)*
Flageat, Cully (2020). Fast and stable MAP-Elites in noisy domains using deep grids. *Proc. of Alife.*

Open questions and challenge

❖ How to perform QD optimization with a gradient? (Nilsson et al. 2021)

❖ How to exploit high-dimensional maps?

❖ Hierarchical QD (Howard et al. 2019, Cully 2018)

❖ Novel uses of QD?

- e.g., learning generative encodings for future optimizations (Gaier et al, 2020)
- for now, mostly robotics and games: what else?

Howard, Eiben, Kennedy, Mouret, Valencia, Winkler (2019). Evolving embodied intelligence from materials to machines. *Nature Machine Intelligence.*
Gaier, Asteroth, Mouret (2020). Discovering Representations for Black-box Optimization. *Proc. of GECCO.*
Cully, Demiris (2018). Hierarchical Behavioral Repertoires with Unsupervised Descriptors. *Proc. of GECCO.*