Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer, Communication ad Electronic Engineering

FINAL DISSERTATION

Promoting Quality Diversity in Soft Robot co-design evolution

Supervisor

Prof. Giovanni Iacca

Co-supervisor

Andrea Ferigo

Student

Elisa Composta

Academic year 2021/2022

Acknowledgements

I would like to thank Prof. Iacca and Andrea for guiding me with patience and dedication in these months, from the beginning of the project till the realization of this work.
I am also extremely grateful to my family, for always supporting me and believing in all of my choices.
Special thanks to Simone, by my side since the very first day of this journey, with whom I shared the hardest times, the best moments, and all of my dreams.
I would like to extend my sincere thanks to the people I met in these last years, especially my project teammates, for encouraging my personal growth by providing me differing perspectives.

Contents

Abstract

An important and promising robotic subject that is gaining ground is Soft Robotics, a sub-field that examines robots with a soft and deformable body.
What makes these individuals so powerful, is that their malleable morphology allows them to face complex and dynamic environments and gives them the chance to perform well even on the hardest tasks.
This is a promising area, since it will enhance human-machine interaction, having many future bio-related applications, like prostheses, artificial organs, and support for gait rehabilitation. However, it sets many challenges due to the complexity of finding individuals that are both soft and robust, able to adapt to unpredictable environments, therefore it requires deeper studies.

Robots, just like living creatures, should be a good match of body and brain. It's easy to imagine that some bodies are intrinsically better than others: an individual with two flexible legs walks better than a rigid squared box. Another simple and intuitive idea is that the brain certainly plays an equally important role: what is a body without a brain? Even the best bodies can have poor results, having no control of their actions.
Given this premises, it is evident that body and brain must be both optimized, in order make individuals perform their best. However, while many researches focus on the controller optimization, less attention is placed on finding the best bodies, mainly because the co-optimization of body and brain is still a challenging problem.

This work addresses the co-optimization problem of soft robots using Reinforcement Learning to optimize the controller and applying Evolutionary Algorithms to design morphologies.
It presents the importance of co-designing both the body and the controller and it makes a comparison between two existing evolutionary algorithms, MAP-Elites and the Genetic Algorithm.
A focus on the controller optimization highlights its role in improving the performances, since it allows also disadvantaged body to perform their best.

An analysis on the adaptability of individuals was made by evaluating them in environments they have not been trained for, optimizing the controller in the new task.

Even though this work only provides an overview of soft robot evolution, what emerges is that this is a promising field, and further studies might lead to interesting results, with many possible future applications.

1 Introduction

Soft robotics is a research field that evolves robots with a soft and deformable body, which guarantees them flexibility and adaptability, allowing them to face complex and unpredictable environments: for instance, they can perform smooth locomotion on rough terrain [11], or squeeze through tight spaces [4]

To find optimal individuals it's important to optimize both the bodies and the controllers.
It's clear that some individuals are intrinsically better than others at some tasks, only because of their morphology; for example, a robot with two legs surely walks better than a rigid squared box.
This intuitive concept is the basis of Embodied Intelligence, theory that states that the body of an agent plays a fundamental role in simplifying tasks and in making adaptive behaviours emerge through the interaction with the environment [5].
At the same time the controller, which is the brain telling the robot which action to take, has a key role, since it determines the behaviour of the individual in the environment.
For these reasons, it's important to optimize both the body and the controller of the robots, in order to find fitting morphologies with proper behaviours.

However, many existing studies focus on the controller optimization, but little attention is placed on finding the optimal bodies. This is mainly because co-optimizing the body and the controller in robotics is characterized as a challenging problem [1].

What makes soft robots interesting is their flexibility and softness, which makes them ideal for many future bio-related applications, since it promises an enhanced human-machine interaction.
They have been proposed as a support for gait rehabilitation [16] and colonoscopies [17], and some other possible future applications are artificial organs and tissue-mimicking active simulators for training and biomechanical studies [6].

This work proposes soft robot co-evolution by combining two main libraries: Evolution Gym [2], that provides the simulator, the co-optimization structure, the tasks to work on and an implementation of GA, and QDpy [3], for the MAP-Elites algorithm implementation, adjusted to work on soft robot evolution.

This chapter is intended to provide the basic notions of the topics related to this work, including how soft robots differ from conventional rigid robots, an overview on the algorithms used and the benchmark adopted. The second chapter focuses on the importance of optimizing the controller, while a comparison of body optimization algorithms, MAP-Elites and GA, is proposed in the following chapter. An analysis on how individuals behave in different environments is introduced in the fourth chapter. Finally, the conclusions provide an overview of what emerged from the experiments presented and introduce some possible future works.

1.1 Soft Robots

Conventionally, engineers have employed rigid materials to fabricate precise, predictable robotic systems, which are easily modelled as rigid members connected at discrete joints.
Rigid-bodied robots can be efficient on a single task, however they don't easily adapt to the environment. What is more, they are often unsafe for interaction with humans.
A solution might be provided by soft robotics, research field that aims at evolving robots with a deformable structure and muscle-like actuation that emulates biological systems, yielding to adaptability and a smooth interaction with the environment [14].

The body of each robot in Evolution Gym is composed of various types of primitive building blocks, and the control of the robot includes action signals applied on the actuator voxels.

1.1.1 Body

The soft robot evolved in this work are voxel-based, which means they are composed of primitive blocks of various type. This multi-material voxel-based structure provides a general and universal

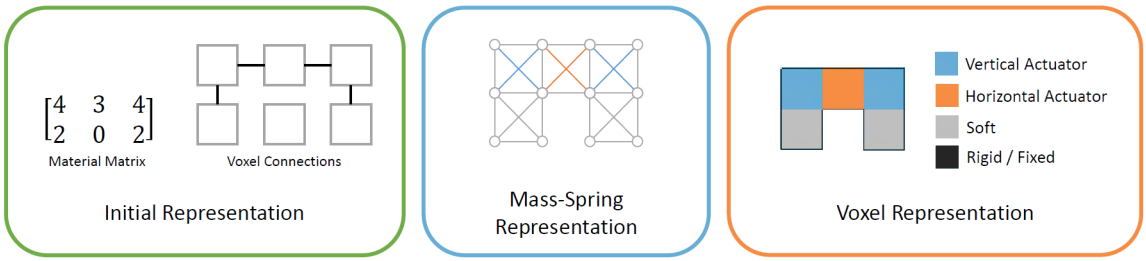


Figure 1.1: Representation of the simulation object[1]

representation for various categories of robot designs, and at the same time results in a modular and expressive structure design space [1].

The robot body is defined as a matrix, where each entry is an integer corresponding to a voxel type, defined in Table 1.1, and a list of connections, describing the adjacent voxels. The bodies of the robots considered in this work are defined by a 5x5 matrix. Note that a generated body is considered valid if it is connected and it has at least one actuator.

Voxel type	**Value**	**Color**	**Behaviour**
Empty	0	white	-
Rigid	1	black	not deformable
Soft	2	grey	deformable
Horizontal actuator	3	orange	can expand/contract horizontally
Vertical actuator	4	blue	can expand/contract vertically
Fixed	5	black	not deformable, fixed position

Table 1.1: Voxel types representation

Soft voxels are deformable under the action of external forces; actuators are the ones that allow the body motion thanks to their gradual expansion/contraction in one direction, according to the action provided by the controller. No robot can be composed of fixed voxels, since they are only used in the definition of the environments to build, for example, the terrain to walk on.

The simulation converts all objects into a set of point masses and springs by turning each voxel into a cross-braced square as shown in the blue panel of Figure 1.1. The figure also displays the voxel graphic representations, together with a legend of the colors chosen to differentiate the voxels types.

1.1.2 Controller

The controller is implemented as a neural network. It takes in input the state of the task, including observations about the robot, the environment and other task-specific information. Details about the observation space of the tasks studied in this work are provided later in this chapter.
The output provided by the controller is the action, a vector with length a^r, where a^r is the number of actuators in the design of robot r. The action vector defines, for each actuator, its contraction/expansion value in relation to its rest length, and can vary in the range $[0.6, 1.6]$.

1.2 Optimization

Co-optimization plays a fundamental role in soft robot evolution: it's important to focus on both the body and the controller, since they both equally contribute to the performance of an individual.
In this work, the co-optimization applied is the one proposed by Evolution Gym, and is formulated as a two-level optimization problem [1].
As shown in Figure 1.2, the controller is optimized using reinforcement learning during the task simulations, and the final reward can be used by the design algorithm to determine the next morphology to work on.

As for the implementation, the design optimization is defined in the outer loop, while the inner loop focuses on control optimization, as shown in Algorithm 1.

In this work, the controller is optimized using a reinforcement learning algorithm, PPO [10], while the design optimization algorithms applied are the genetic algorithm [2] and MAP-Elites [3].

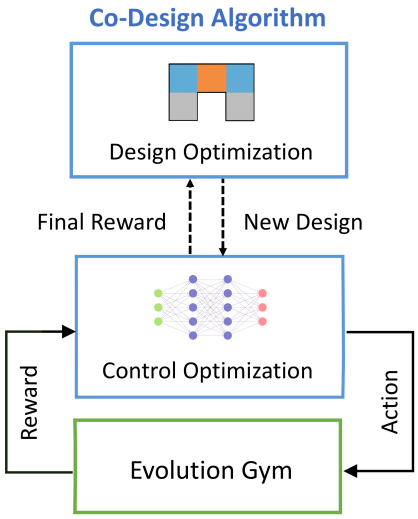


Figure 1.2: General structure of the experiments [1]. The co-design process generates the individuals to evaluate. Evolution Gym runs the simulations according to the given bodies and controllers, and computes the reward.

Algorithm 1 Robot evolution co-optimization algorithm [1]

Require: Task specification T, number of generations n, population size p.

```
S ← ∅                                          // Dataset of robot designs, controllers and rewards
D_1, ..., D_n ← SampleDesigns(p)               // Sample an initial population of robot designs
for i ← 1 to n do
    for j ← 1 to p do
        C_j ← OptimizeControl(T, D_j)          // Optimize the controller of given robot design
        r_j ← EvaluateReward(T, D_j, C_j)      // Evaluate the reward of given design and controller
        S ← S ∪ {(D_j, C_j, r_j)}              // Update the evaluation result to the dataset
    end for
    D_1, ..., D_n ← OptimizeDesigns(S, p)      // Optimize a population of robot designs to evaluate
end for
```

1.2.1 Evolutionary Algorithms

Mainstream artificial intelligence has been very successful at designing algorithms and devices that solved stable problems. But, in doing so, in ended up neglecting fundamental aspects of biological intelligence, such as physical embodiment, behavioral autonomy, evolution and learning, that make biological organisms prone to errors and sometimes difficult to predict, but also so successful to survive in unknown and changing environments [9].

These premises lay out the grounding assumptions on which are founded Evolutionary Algorithms (EA): solving dynamic problems, like learning in unpredictably changing environments, requires equally dynamic solutions, that might be already provided in biology.

There are many types of EA, but the common underlying idea behind all these techniques is the same: given a population of individuals within some environment that has limited resources, competition for those resources causes natural selection (survival of the fittest) [7].

Applying EA to soft robot evolution has proven to be advantageous for the potential to uncover unconventional designs, difficult to anticipate for a human expert, guaranteeing efficiency and robustness. Furthermore, evolution is able to exploit synergistic effects between body and brain that, are often too hard to model analytically. [8]

In this work two EA, introduced in the next paragraphs, are applied to soft robot design op-

timization: the Genetic Algorithm (GA) and the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites).

Genetic Algorithm

The Genetic Algorithm (GA) is an iterative search algorithm that draws inspiration from Charles Darwin's Theory of natural selection. According to this algorithm, only the fittest individuals survive through generations. Mutation and crossover can be applied to the survivors, so as to generate offspring with more chance to fit.

The structure upon which GA are built is the following. In each generation, the population is composed by *pop_size* individuals. In the first generation the designs are sampled. After the controller optimization and the evaluation, the top $x\%$ of individuals survive, and take part to the next generation. The survivors are iteratively sampled, and their variation, applying mutation and crossover, generates new individuals to evaluate.
This iterative process goes on until it meets the termination condition; in this work, the process ends when *max_evaluations* individuals have been evaluated.

The mutation strategy here applied states that given a morphology shape, each cell has a certain probability of mutating, defined by the *mutation_rate* parameter. Each voxel can mutate its type into any of the others with the same probability, except for the empty type: it's three times more likely to become empty, since mutating a voxel from/to empty allows a change in the morphology. In the implementation used, no crossover has been applied.

The obtained body is considered valid if it satisfies the body constraint (being connected and having one actuator at least) and has not been already evaluated. When the proposed body doesn't satisfy these constraints, a new mutation on the same starting individual is proposed, until the maximum number of allowed attempts, *num_attempts*, is reached.

If no valid mutated individual is found after the maximum number of attempts, it is discarded and a new body among the survivors is selected to mutate.

MAP-Elites

In Evolutionary Robotics a population of solutions is evolved to optimize robots that solve a given task. However, in traditional Evolutionary Algorithms, the population of solutions tends to converge to local optima when the problem is complex or the search space is large, a problem known as premature convergence. Quality Diversity algorithms try to overcome premature convergence [13] by encouraging diversity and domain illumination.

An example of QD algorithm is the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites). This algorithm builds a grid, where each dimension consists of a discretization of the chosen features in their domain space, according to the defined *map_shape*. Given the discretization, MAP-Elites searches for the highest performing solution for each cell in the N-dimensional feature space[12], where N is the number of features. Each individual is mapped to one bin only, according to its features value. After an individual is evaluated and its features are computed, it can be inserted to the map if the corresponding bin is empty or its fitness value is greater than the one already stored, which is the occupant elite.

In the first generation, *num_init* individuals with randomly generated bodies are sampled, to firstly explore the map. The later generation is composed of *num_mutated* individuals, obtained by a mutation of bodies already evaluated and stored in the current map.
Individuals are evaluated in batches of size *batch_size*. At the end of each batch, the individuals evaluated can be inserted in the map or disregarded, as previously described.
The new individuals to evaluate in the next batch are generated by mutating the ones currently stored in the map. The mutation strategy applied to these experiments is the same one used on GA, described in the previous section.
Note that all evaluated individuals have different morphologies since, during the experiments, evaluated bodies are hashed and stored, so as to avoid duplicated evaluations.

When all individuals are evaluated, the map represents the best individuals for each combination of features, and diversity is promoted, since different cells map to individuals with different feature values.

The features implemented and used in this work are the actuation and the emptiness of the body, described in detail in the following paragraphs.

Actuation Since Evolution Gym allows the distinction of voxel types, it's possible to define a feature that considers the number of actuators, which allow the robot motion.
The actuation A used in these experiments is defined as follows:

$$A^r = \frac{a_v^r + a_h^r}{n^r}$$

where r is the individual on which the feature is computed, a_v and a_h are the number of vertical and horizontal actuators, and n is the number of (non-empty) voxels composing the morphology. Its domain is $[0, 1]$: when morphologies are composed by actuators only, the feature value is 1, and reducing the number of actuators makes this value tend to 0. However, actuation cannot be precisely 0, since having at least one actuator is one of the constraints to create a valid body.

Emptiness The emptiness E of a robot r considers the number empty cells in the robot morphology, that is the number of potential, but unused, voxels.
Let e be the number of cells of zero value in the 2D robot body matrix (i.e. the empty voxels) and s the maximum number of voxels to fill the matrix, according to the shape definition of individual r:

$$E^r = \frac{e^r}{s^r}$$

The domain for this feature is $[0, 1]$: individuals with an emptiness value close to 1 are composed by few voxels; when emptiness is 0, it means that the body is composed by the maximum number of voxels possible.
Since the robots evolved in this work are defined by a 5x5 matrix, their maximum number of voxels is 25.

1.2.2 Reinforcement Learning

Reinforcement Learning (RL) is a Machine Learning paradigm that aims at mapping situations into actions.
The learner tries actions and gets a numerical reward, positive or negative, according to its performance on the task considered. Thanks to the feedback derived from its interactions with the environment, good behaviours are encouraged and bad performing actions are penalized, taking to long-term results. This trial and error method allows the agent to learn from its own experience, directly interacting with the environment.

The robot controller optimization proposed by Evolution Gym is based on the Proximal Policy Optimization (PPO), a RL algorithm which alternates between sampling data through interaction with the environment, and optimizing a "surrogate" objective function [15]. PPO is an on-policy algorithm, which means that it considers a batch of data using the current policy to evaluate its reward. Its implementation, provided by [10], together with the hyperparameters, has been applied to the experiments introduced in this work.

1.2.3 Tasks

Evolution Gym proposes many tasks, classified into different difficulty levels according to the performance of the baseline algorithms that were tested on them. Each environment has it's own definition of reward. This work focused on three tasks, categorized as *easy* ones: walker, pusher, carrier.

The observation space of the tasks consider parameters regarding the robot or the eventual object it interacts with. Let o be the object considered, the following are defined:

- Velocity of the center of mass: v^o
  2-D vector computed averaging the velocities of all point masses composing o; v_x^o and v_y^o are the components on the x and y directions.

- Position of the center of mass: p^o
  2-D vector computed averaging the positions of all point masses composing o; p^o_x and p^o_y are the components on the x and y directions.
- Number of point masses: n
  number of point masses in the object considered.
- Position of all point masses: c^o
  vector of length $2n$ representing the position of all n point masses of object o, relatively to p^o, the center of mass of the object.

Walker-v0

This task is the most commonly studied in soft robot evolution. The environment is composed by a simple rigid flat terrain, and the goal of the individual is to walk as far as possible.
The robot r is given a reward R for moving in the positive x-direction:

$$R = \Delta p^r_x$$

Once it reaches the end of the terrain, it gets an additional one-time reward of 1.
The observation space has dimension $S \in R^{2n+2}$, and is formed by concatenating vectors v^r, c^r, with lengths 2 and $2n$ respectively.

Figure 1.3: Walker-v0

Pusher-v0

In this task the goal of the robot is to push a box initialized in front of it as far as possible.
The reward is $R = R_1 + R_2$, where R_1 is the score obtained according to the positive x direction motion of both the robot and the box, giving more importance to the latter one:

$$R_1 = 0.5 \cdot \Delta p^r_x + 0.75 \cdot \Delta p^b_x$$

R_2 is the penalty applied according to the separation on the x direction of the two objects:

$$R_2 = -\Delta |p^b_x - p^r_x|$$

The only components that contribute to the reward are p^r_x and p^b_x, which are the x-direction components of the position vector p of the center of mass of the robot and the box, respectively.
The first time the robot successfully reaches the end of the terrain, it gets a reward of 1.
The observation space has dimension $S \in R^{2n+6}$, and is formed by concatenating vectors v^r, $p^b - p^r$, v^b, c^r, with lengths 2, 2, 2, $2n$ respectively.

Figure 1.4: Pusher-v0

Carrier-v0

The goal in this task is to carry a box, initialized above the individual, as far as possible.
The reward is $R = R_1 + R_2$, where R_1 is the score obtained according to the positive x direction motion of both the robot r and the box b, and R_2 is a penalty applied when when the box falls below a certain height:

$$R_1 = 0.5 \cdot \Delta p_x^r + 0.5 \cdot \Delta p_x^b$$

$$R_2 = \begin{cases} 0 & \text{if } p_y^b \geq t_y \\ 10 \cdot \Delta p_y^b & \text{otherwise} \end{cases}$$

where t_y is the height threshold to determine whether to apply the penalty.
Just as in the other tasks described, the robot gets a one-time reward of 1 for successfully reaching the end of the terrain.
The observation space has dimension $S \in R^{2n+6}$, and is formed by concatenating vectors v^r, $p^b - p^r$, v^b, c^r, with lengths 2, 2, 2, $2n$ respectively.

Figure 1.5: Carrier-v0

2 Controller optimization

Co-optimization has a key role in evolving individuals. Some morphologies are clearly more suitable than others for some tasks, but a poorly developed controller can compromise the performance even of the fittest bodies.
This chapter analyzes how performances of individuals vary according to the level of optimization given to the controller.
Note that the results here proposed, as in the rest of the work, are obtained by averaging over the results of six experiments with different seeds. The MAP-Elites parameters applied in this experiments, and in the following ones, are shown in Table 2.1

parameter name	**value**
map_shape	(10, 10)
num_init	500
num_mutated	1500
batch_size	28
mutation_rate	0.1
num_attempts	50

Table 2.1: MAP-Elites main parameters

2.1 Controller episodes

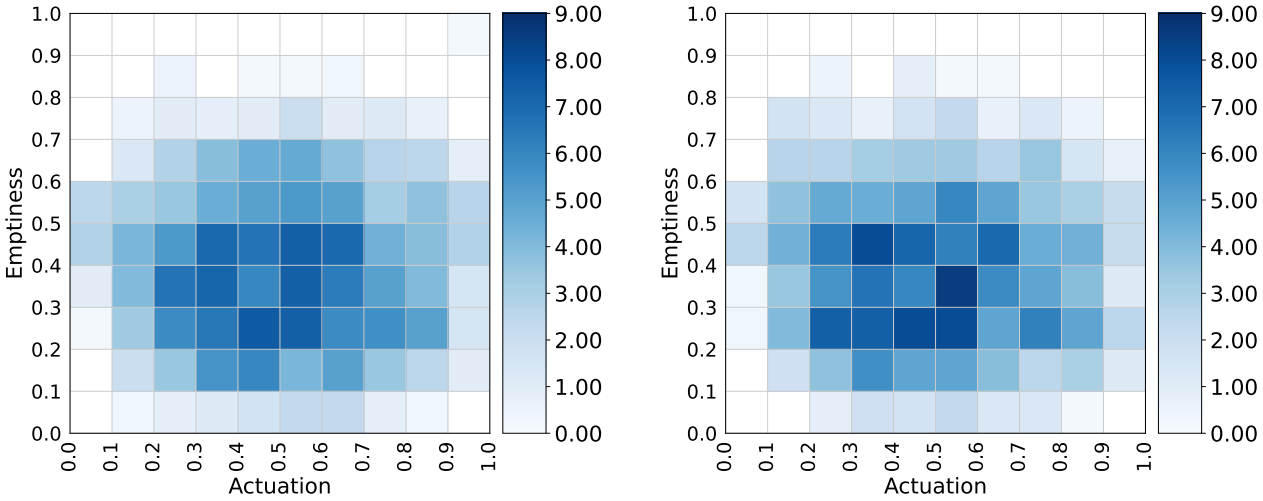
The controller is optimized using PPO algorithm, as introduced in Chapter 1. Being an on-policy algorithm, it considers batch of data using the current policy to evaluate the reward, therefore the fitness value can vary in relation to the size of evaluation interval considered.
This section proposes a comparison of experiments in which the controller of the individuals is optimized using 40 or 60 episodes, to show how this choice can affect soft robot evolution. Simulations are run in the walking environment with MAP-Elites algorithm, introduced in Chapter 1.

2.1.1 Activity

Being a quality diversity algorithm, MAP-Elites aims at promoting diversity.
After the first generation of randomly sampled morphologies, individuals are picked from the map and mutated, with no concern for their fitness.
Given these premises, we expect that working on the controller should not affect the choice of the individuals to evaluate and, consequently, the bins visited.

Figure 2.1 shows a comparison of the activity maps, grids showing how many times a cell has been overwritten; the brighter is the color of a cell, the more its value has been overwritten. As shown in figure, the expectations match the obtained results: both the activity maps are widely filled, and there is not a single specific bin on which they both focus.

Figure 2.2 compares the activity trends, which describe the number of explored bins after a certain number of evaluations; since the maximum number of cells that can be visited is 100, it can be read as a percentage. Note that the trends show both the mean trend (main line) and the variance (shadowed area). The comparison makes it even more evident: the number of bins visited, after a certain number of evaluations, is almost the same, therefore changing the number of episodes for the controller optimization does not affect the choice of individuals.



(a) 40 episodes

(b) 60 episodes

Figure 2.1: Comparison of the activity maps applying controller optimization using 40 (a) or 60 (b) episodes. The overall distribution of the visited bins is not affected.

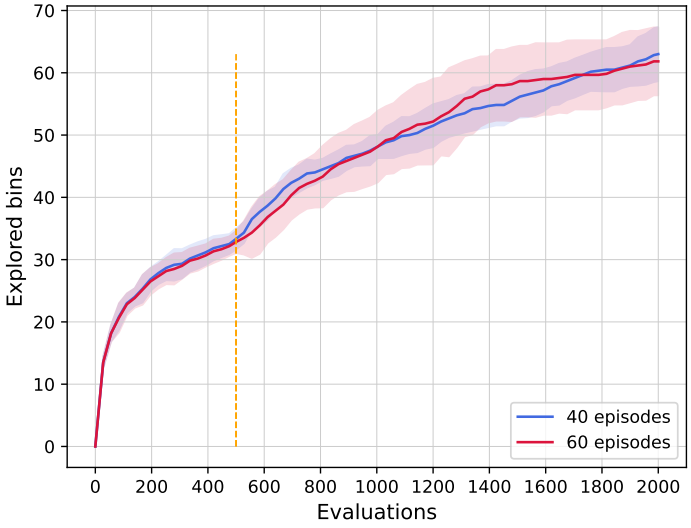


Figure 2.2: Comparison of the activity trends. Changing the number of episodes for the controller optimization does not affect the activity trend, since the choice of individuals to mutate and evaluate doesn't depend on the fitness obtained

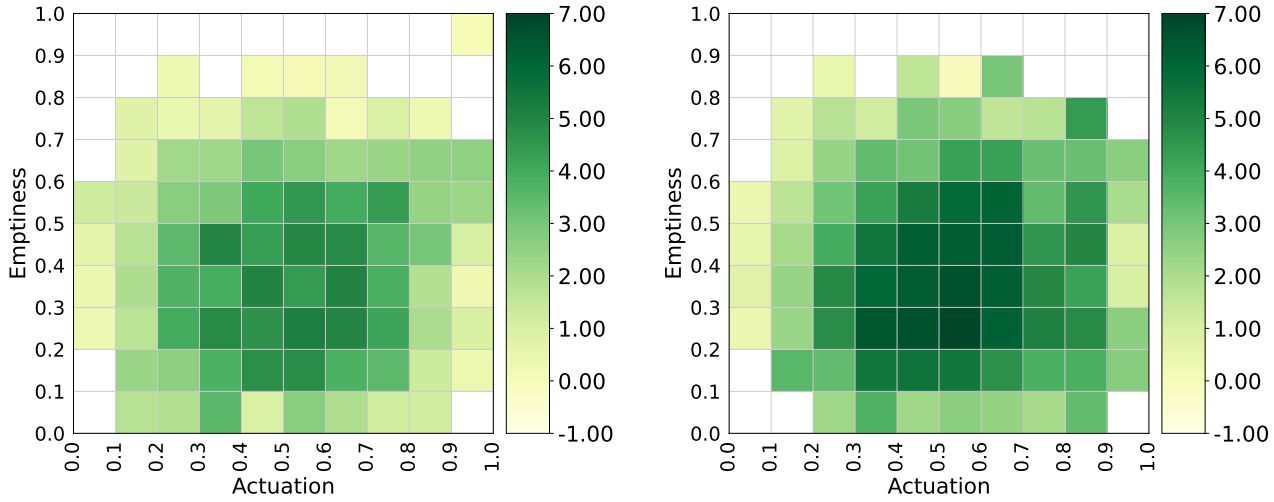
2.1.2 Performance

An analysis on how changing the number of episodes for the controller optimization affects the overall performance is here proposed.

A comparison of the fitness trends, that describe the behaviour of the best fitness value found after a certain number of evaluations, is illustrated in Figure 2.4, and it shows that individuals with a 60-episode-optimized controller obtain a greater reward. What is more, the 40-episodes trend has constantly greater variation, probably because of the intrinsic fitness of some morphologies: the naturally fittest bodies can get a great score even though the controller is not highly optimized, while others need to put more effort on the brain optimization to perform their best.

The effects on the fitness can be also inferred comparing the performance maps, that show for each cell the best fitness value obtained by an individual mapped to that cell; brighter cells are the ones that reached a greater fitness value. The comparison in Figure 2.3, show that optimizing the controller

with more episodes leads to greater scores all over the map, resulting in a widely spread brighter color. This means that the more intensive controller optimization didn't affect a type of individual only, but allowed a general improvement, for all the visited cells, including the border ones.



(a) 40 episodes

(b) 60 episodes

Figure 2.3: Comparison of the performance maps applying controller optimization with 40 (a) or 60 (b) episodes. Controller optimization highly affects the general performance of all individuals.

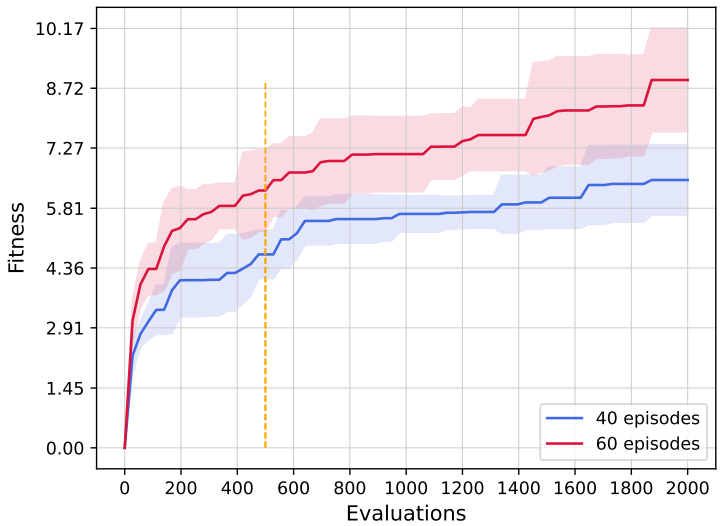


Figure 2.4: Comparison of the fitness trends. A focus on the controller optimization leads to higher fitness.

2.2 Conclusion

This experiments confirmed that the controller optimization has a key role in improving the general performance of the individuals and it consequently leads to a greater reward. However, changing the number of episodes for the controller didn't affect the overall exploration of the map, since MAP-Elites chooses the individuals to mutate and evaluate from the ones stored in the map with no regard for their fitness value, as it was expected of a QD algorithm.

3 Body optimization

This chapter focuses on the morphology optimization. Selecting individuals to survive and mutate is a critical point in body optimization, therefore there are many different techniques with different goals. This chapter provides a comparison between MAP-Elites and GA, introduced in Chapter 1, applied to soft robot evolution. The main difference between these algorithms is that the first one promotes diversity, while the second one allows the survival only of the best individuals per generation. Therefore it is expected that this difference affects the results.

For each experiment all the maps and trends are compared. A vertical dashed in the trend plots indicates the end of the first generation of random individuals of MAP-Elites, therefore it does not have any specific meaning in the GA application.

Since MAP-Elites is a QD algorithm, it's supposed to fill the performance map with a quite uniform color brightness. On the contrary, GA aims at optimizing the best fitness, with no regards for the body diversity.
For these reasons, GA is expected to achieve a better performance, but it's maps will focus only on few cells. On the contrary MAP-Elites, since it promotes diversity, might fill the maps better and more uniformly, at the expense of the fitness value.

The tasks studied in this work are the Walker, the Pusher, and the Carrier, introduced in Chapter 1.

The parameters applied to GA are shown in Table 3.1, while MAP-Elites parameters are the same introduced in Chapter 2; the controller optimization uses 60 episodes in both cases.

parameter name	**value**
pop_size	100
max_evaluations	2000
x (%)	60-0
mutation_rate	0.1
num_attempts	50

Table 3.1: GA main parameters

3.1 Walker

The walker task is the simplest one, since the only interactions the robot has with the environment are the ones with the terrain.
Given these premises, the maximum fitness value is expected to be high using both the algorithms.

3.1.1 Performance analysis

A comparison of the fitness trends, shown in Figure 3.1, points out that the two curves have similar shapes. GA performs slightly better since the first generations, but MAP-Elites gets very good results too.

The performance grids in Figure 3.2 confirm that the two algorithms reach good fitness values, since they both have dark green coloured cells.
The main difference is that MAP-Elites widely fills the map, and the color is uniformly distributed, with a slightly brighter color in the center, whereas the GA map highlights that its best fitness values converge to a specific bin, and a noticeable color difference exists between that cell and the border ones.

3.1.2 Activity analysis

An analysis on the activity maps, shown in Figure 3.3, underlines the different approaches of the two design optimization algorithms.

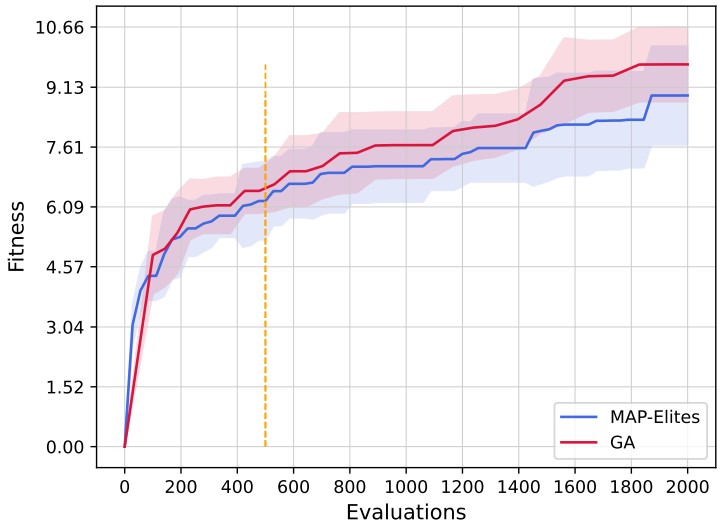
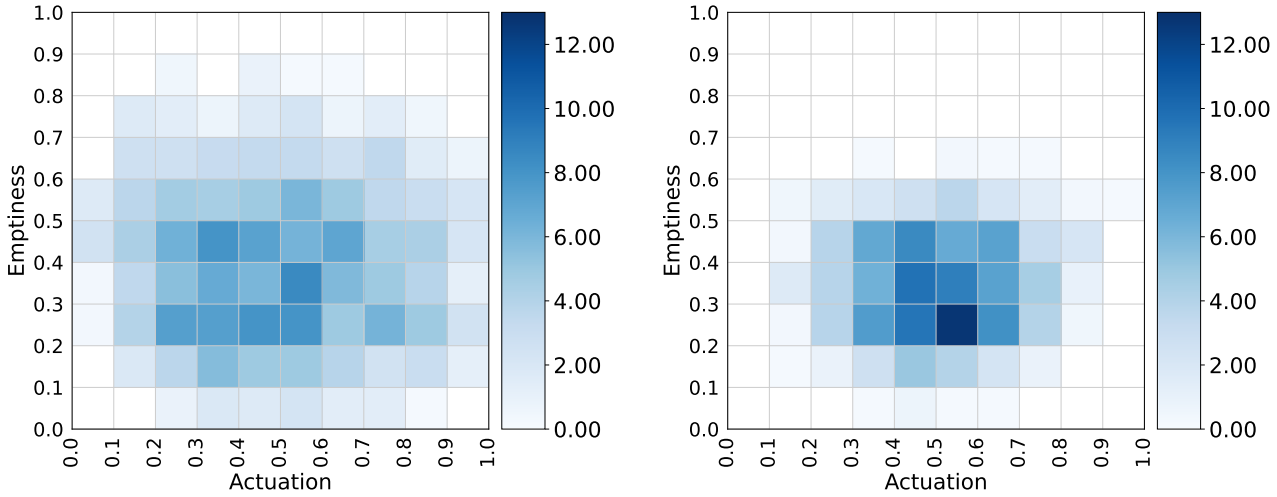


Figure 3.1: Walker task fitness trend comparison. The two algorithms reach high fitness values, and the two curves have similar shapes.

The GA map shows that the value of a specific cell has been overwritten multiple times. It is not by chance that the most active cell is also the one with the best performance (in relation to Figure 3.2): this is due to the fact the only goal of GA is to always get better performing individuals, selecting, and then mutating, the best ones at the end of each generation.



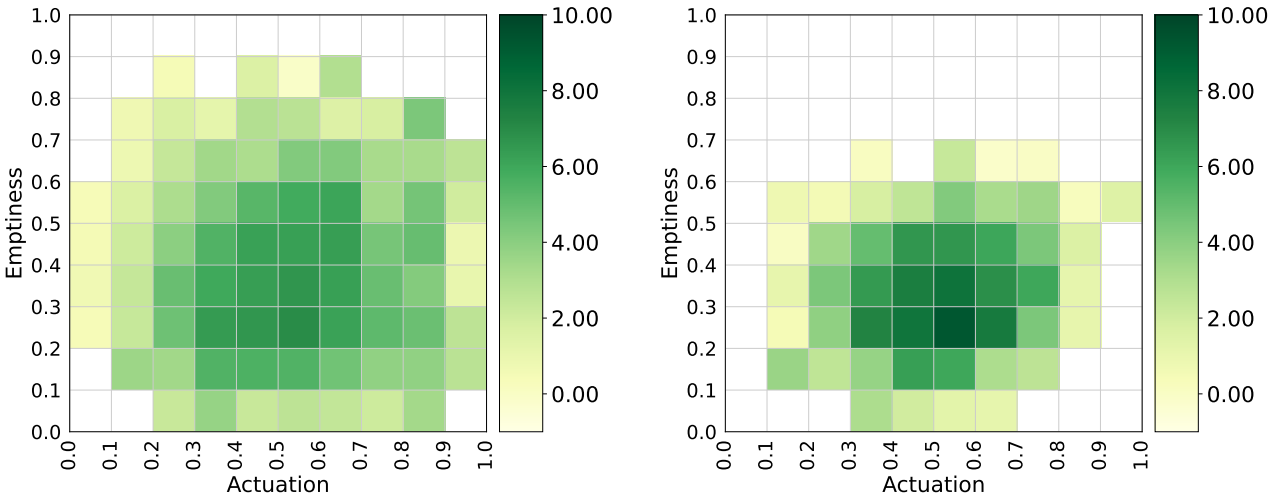
(a) MAP-Elites

(b) GA

Figure 3.3: Walker task activity grid comparison. MAP-Elites (a) fills the map widely and uniformly; GA (b) focuses on few cells and converges to a specific bin, that coincides with the one with the best fitness value.

On the contrary, MAP-Elites activity map is uniform and it's not correlated to the performance map, confirming that the algorithm is not focused on the mere best fitness achievement. However, the cells in the center have a darker colour in comparison with the ones in the border. This is mainly because there's less chance of having individuals with feature values at the end of the domain.

Thanks to the activity trends in Figure 3.4, it is possible to analyze the number of explored bins in the map in relation to the number of evaluated individuals.
The curves of the two algorithms have similar shapes for the first 500 evaluations, which is the number of individuals that MAP-Elites generates randomly. After this threshold, the number of bins explored



(a) MAP-Elites (b) GA

Figure 3.2: Walker task performance grid comparison. MAP-Elites (a) fills the map quite uniformly; GA (b) converges to a specific bin.

by MAP-Elites increases highly and quite constantly, whereas GA continues a much slower rise. This confirms that promoting diversity also means promoting exploration of the feature domains.

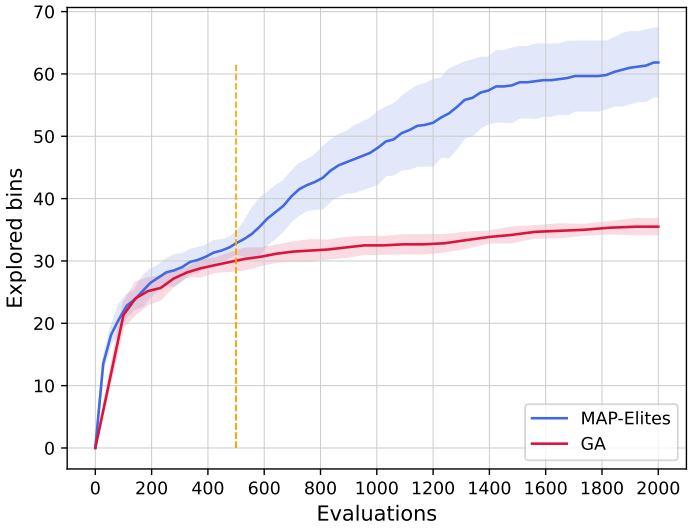


Figure 3.4: Walker task activity trend comparison. After a generation of randomly generated bodies, MAP-Elites continuously explores different cells. GA, after the first few generations, has a much slower rise, since it focuses on few cells.

3.1.3 Top designs

The morphologies of the best performing individuals emphasize the different approaches of the two algorithms.

Using GA, the best individuals per generation survive and new individuals can be generated by a mutation of those who survived. Therefore, the resulting top designs are very similar, as shown in Figure 3.5.

MAP-Elites is a QD algorithm, therefore it promotes diversity: the best performing individuals, shown in Figure 3.6, have different morphologies.

However, most of the top designs obtained by the two algorithms have a great number of horizontal

actuators and they have grown two legs, which is what allows them to walk; this interesting result highlights once again the strong connection with biology, since they resemble natural creatures, even though they had no prior knowledge.

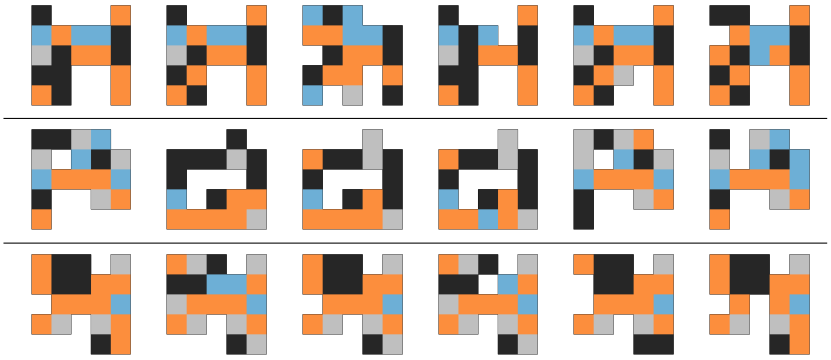


Figure 3.5: GA best performing morphologies on the walker task in three different experiments. For each experiment, top designs look very similar.

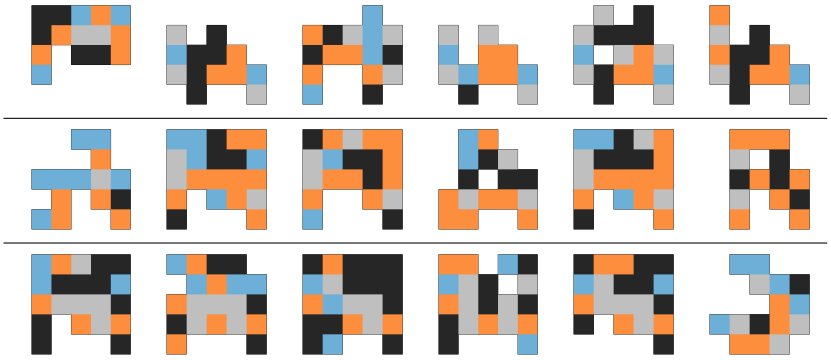


Figure 3.6: MAP-Elites best performing morphologies on the walker task in three different experiments. The algorithm produced well-performing individuals with various designs.

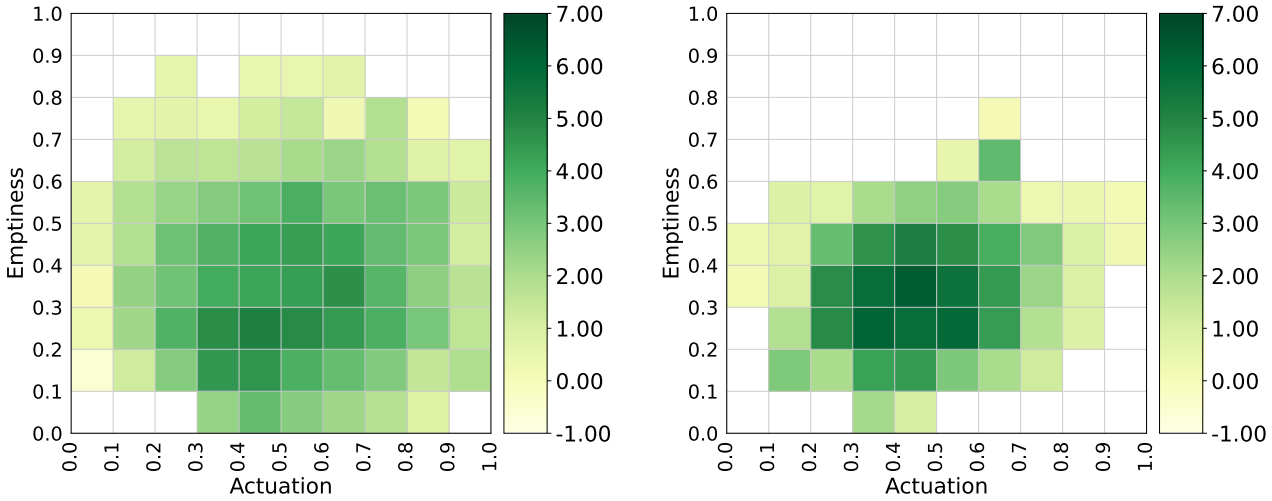
3.2 Pusher

In this environment, the goal is to push an object as far as possible. To do so, individuals must also learn how to walk. The object can't be simply thrown, because a penalty is applied according to its distance from the body.

3.2.1 Performance analysis

An analysis on the performance map, in Figure 3.7, shows that GA finds well-performing individuals that are mapped on few close cells. On the contrary, MAP-Elites widely fills the map, and shows that even boundary cells, poorly touched by GA, can lead to interesting results.

The fitness trend comparison in Figure 3.8 shows that the behaviour of the two algorithms in finding the best fitness value is very similar, especially for the first evaluations. This means that the two of them, after the same amount of evaluations, found vary similar, fitness values.
After further evaluations, they both continued their growth; the reduction of standard deviation in GA confirms that its behaviour is quite regular.



(a) MAP-Elites (b) GA

Figure 3.7: Pusher-v0 performance map comparison. GA (b) best performing individuals converge to few cells, while MAP-Elites map (a) is more uniform.

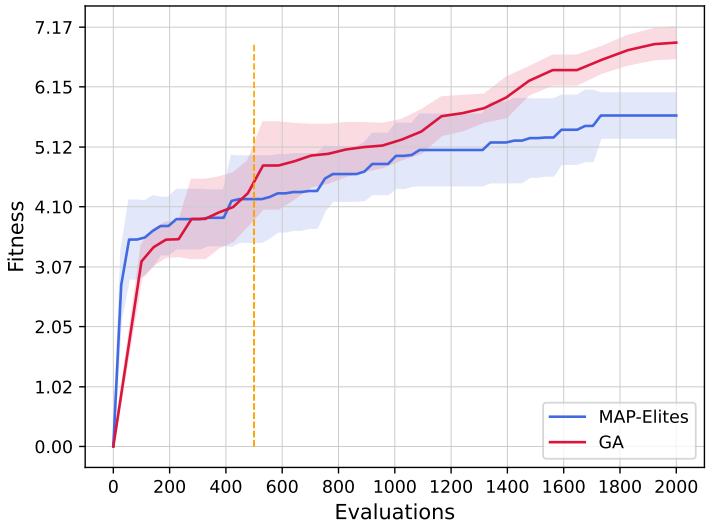


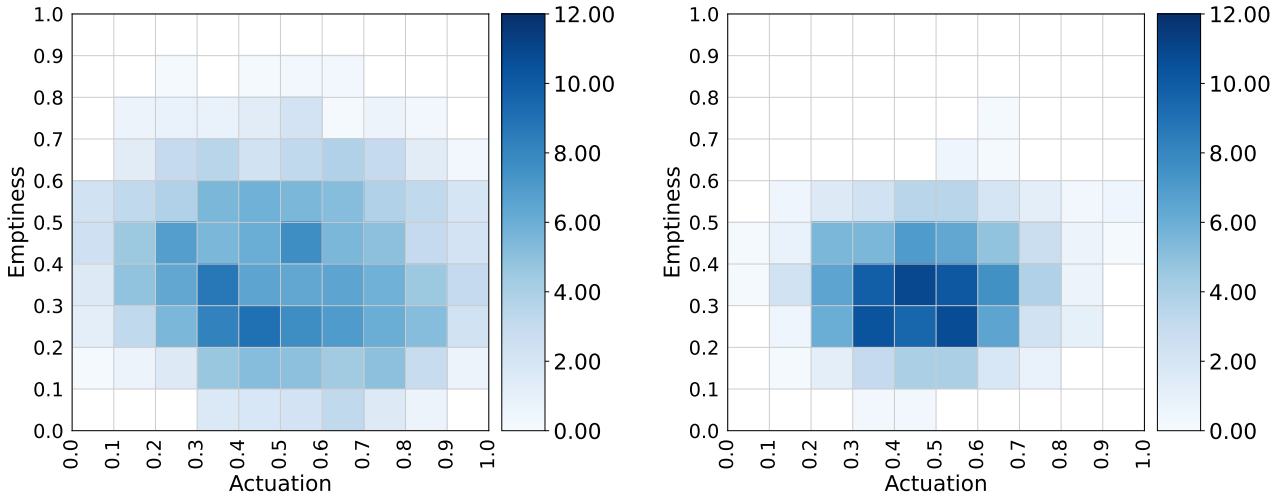
Figure 3.8: Pusher-v0. Fitness trend comparison. The two algorithms have a similar behaviour, but GA gets higher fitness values.

3.2.2 Activity analysis

As it was expected, the activity map of GA, shown in Figure 3.9 focuses only on few cells. More specifically, border cells are the less visited, while it greatly focuses on the cells with the most common feature values. It's not by choice that the most overwritten cells are the one with the best fitness values. On the contrary, MAP-Elites doesn't focus on few cells only. Its central area is more coloured, but there is not a small region that stands out. Also, its most overwritten cells do not coincide with the best performing ones, as it can be seen comparing the activity and the performance maps.

Comparing the number of explored bins behaviour, in Figure 3.10, it is possible to distinguish two different moments. For the first 500 evaluations, the two lines almost coincide. The morphologies evaluated by MAP-Elites in this stage are randomly sampled, so it's an exploration stage. After further evaluations, GA continues a slow increase, that becomes steady at around 35% of explored bins. On the contrary, MAP-Elites continues its rise to about 60% of explored bins, almost twice as GA.

Also, the low standard deviation in the activity trend of GA confirms that this is a common behaviour for the algorithm, which focuses on the performance with no regards for the map exploration.



(a) MAP-Elites

(b) GA

Figure 3.9: Pusher-v0 activity map comparison. GA (b) focuses on few cells only, while MAP-Elites map (a) is widely and more uniformely explored.

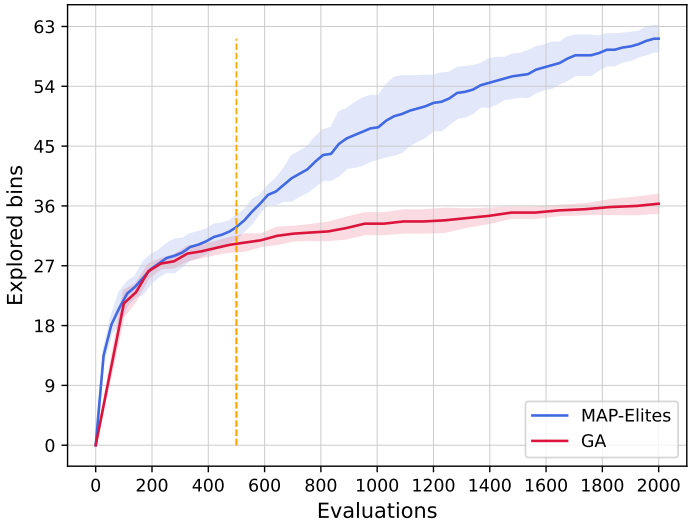


Figure 3.10: Pusher-v0. Activity trend comparison. After a generation of randomly generated bodies, MAP-Elites continuously explores different cells. GA, after the first few generations, has a much slower rise, since it focuses on few cells

3.2.3 Top designs

The designs of the best performing individuals behave differently according to the algorithm applied. The best individuals per generation found by GA are the ones to survive, and they can mutate to generate offspring. This is why the top designs look very similar, as shown in Figure 3.11. On the contrary, MAP-Elites promotes diversity, so the top designs obtained, in Figure 3.12, have different morphologies. Since the pusher task also had the implicit goal of leaning to walk, most of the the individuals obtained grew legs.

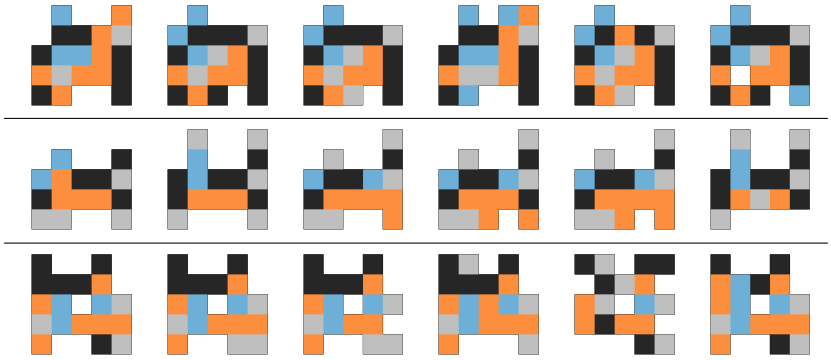


Figure 3.11: GA best performing morphologies on the pusher task in three different experiments. For each experiment, top designs look very similar.

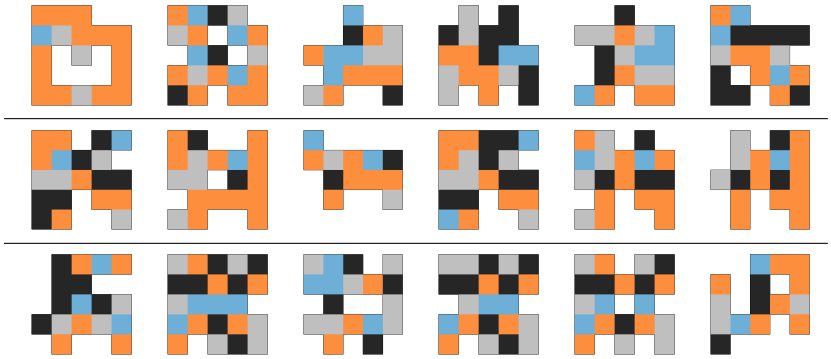


Figure 3.12: MAP-Elites best performing morphologies on the pusher task in three different experiments. The algorithm produced well-performing individuals with various designs.

3.3 Carrier

In this task, the goal is to carry an object as far as possible.
The individuals trained in this task are supposed to learn how to walk and how to carry a box initialized in front of them.

3.3.1 Performance analysis

A comparison of the performance grids in Figure 3.14 show that MAP-Elites fills the map more widely than GA.
GA gets a higher best fitness value, and the distribution of color in its cells shows that the individuals with a good performance converge to few cells only. However, focusing on few cells only precludes it to find well-performing individuals in other map areas. On the contrary, MAP-Elites, exploring the map, found well performing individuals even on the boundary cells, never explored by GA.
Since a penalty is applied when the object falls under a certain height, some individuals stored in the map have a negative fitness value.

The fitness trend comparison in Figure 3.13, shows that GA has higher best-fitness values after all evaluation intervals. Even though they sometimes get very close, MAP-Elites never obtains a higher best-fitness value. This is mainly because GA focuses on obtaining the best individual, therefore it selects its best performing robots to survive and, potentially, mutate. On the contrary, the selection of individuals to mutate in MAP-Elites is not driven by their fitness value.

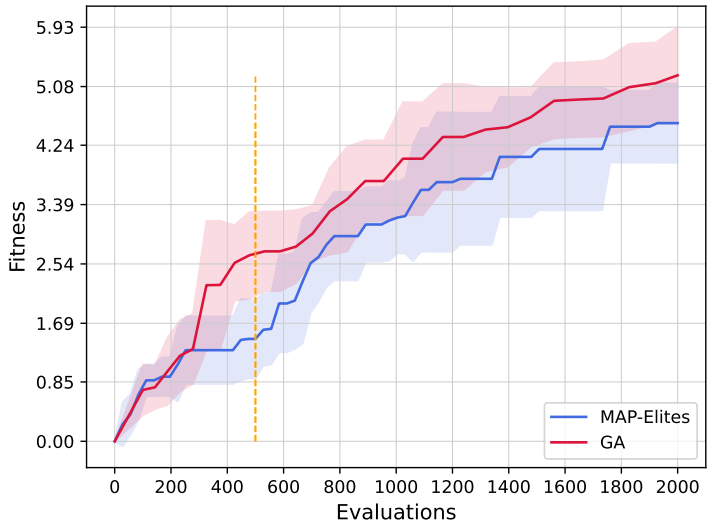
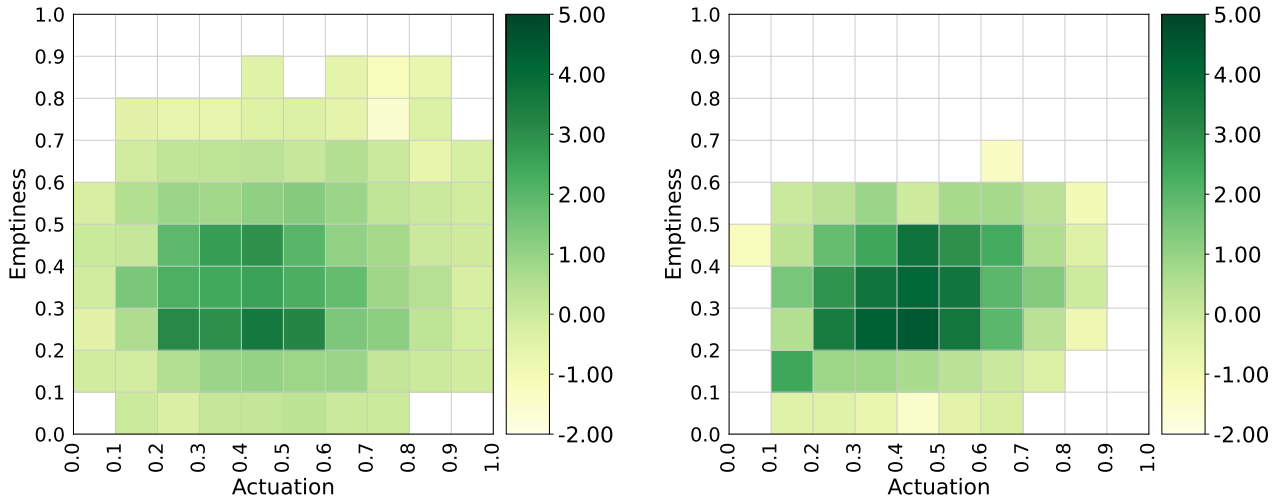


Figure 3.13: Carrier-v0. Fitness trend comparison. The two algorithms have a similar behaviour, but GA gets higher fitness values.



(a) MAP-Elites

(b) GA

Figure 3.14: Carrier-v0 performance map comparison. GA (b) best performing individuals converge to few cells, while MAP-Elites map (a) is more uniform.

3.3.2 Activity analysis

The activity trend comparison in Figure 3.15 highlights the different approaches the two algorithms have in exploring the map.
GA has a rapid growth in the first evaluations, but then it remains stable at around 35% of explored bins. On the contrary, MAP-Elites has a gradual rise for all the experiment, leading to an almost doubled number of explored cells, compared to GA.

This is mainly because GA tends to focus on few cells, as illustrated in Figure 3.16, and trying to get better fitness values from individuals with the same features becomes always harder.
The activity map of MAP-Elites is widely filled, and picking from the many cells with lower fitness values, as previously analyzed from Figure 3.14, increases the chance to overwrite them.

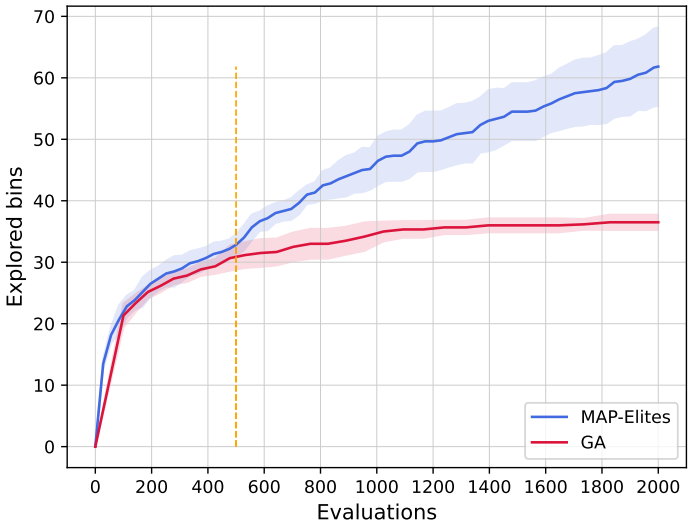
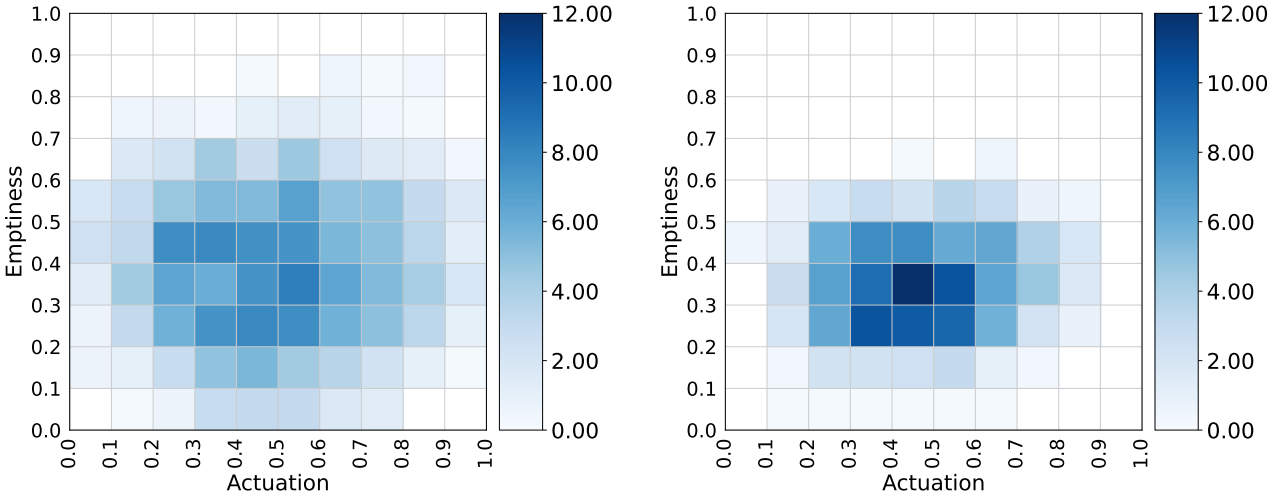


Figure 3.15: Carrier-v0. Activity trend comparison. After a generation of randomly generated bodies, MAP-Elites continuously explores different cells. GA, after the first few generations, has a much slower rise, since it focuses on few cells



(a) MAP-Elites

(b) GA

Figure 3.16: Carrier-v0 activity map comparison. GA (b) focuses on few cells only, while MAP-Elites map (a) is widely and more uniformely explored.

3.3.3 Top designs

GA and MAP-Elites are designed with different goals: the first focuses on obtaining the best performance, the last one on the diversity. This difference is reflected on the resulting top designs. While individuals found by GA, Figure 3.17 look evidently very similar, the ones obtained by MAP-Elites, shown in Figure 3.18, don't. However, the two algorithms agree on the need of extra space above the individual to carry the box. Even though it was not a constraint of the task, all the best performing carriers have empty voxels above them, surrounded by few non-empty voxels to keep them from falling forward or backwards.

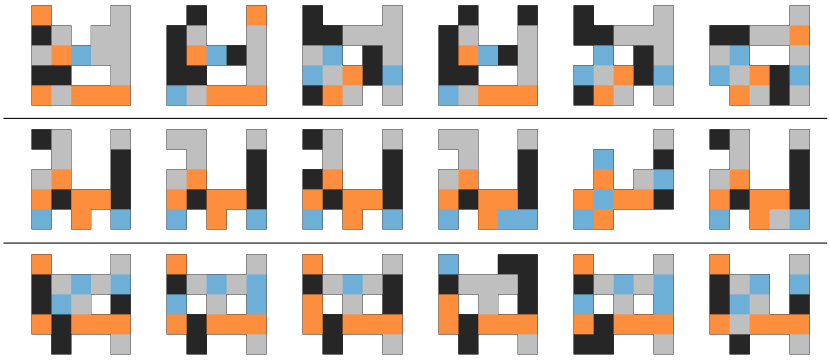


Figure 3.17: GA best performing morphologies on the carrier task in three different experiments. For each experiment, top designs look very similar.

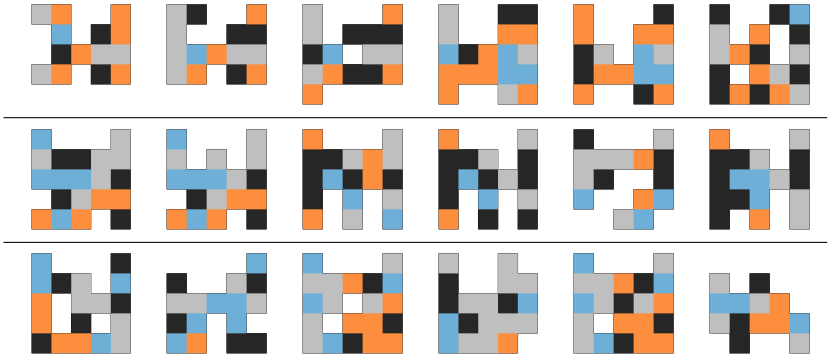


Figure 3.18: MAP-Elites best performing morphologies on the carrier task in three different experiments. The algorithm produced well-performing individuals with various designs.

3.4 Conclusion

The Genetic Algorithm finds greater fitness values but, after a short initial exploration of the map, it focuses only on few cells. The most explored bins correspond to the ones with the greatest fitness, and this is because the fittest individuals survive through generations, therefore they are more likely to be selected and mutated, to generate new individuals. As a consequence, the top individuals have very similar designs.

On the contrary, MAP-Elites widely explores the map and evaluates individuals with different feature values. Even though the central area in the map is usually the most visited, due to the higher chance of having bodies with medium feature values, boundary cells are not disregarded, and they can lead to interesting results. Since this algorithm promotes diversity, the individuals with the best fitness have various morphologies.

Even though the best fitness found by MAP-Elites algorithm is always lower than the one reached by GA, it does not disappoint. This results confirm that QD algorithms can find well performing individuals, that might be rejected by other reward-oriented algorithms like GA.

However, the two algorithms evolved robot bodies with some similarities: they grew legs or evolved space to carry an object, according to their task goal, resembling natural creatures even with no prior knowledge.

4 Multitasking

The goal of soft robotics is to generate a new category of robots, inspired by nature and designed to perform well on dynamic and unpredictable environments.
This chapter analyzes the behaviour of map-elites best performing individuals in tasks they where not designed for. For each experiment, the fittest individuals in a certain task, which are the ones stored in the final map, have been trained in a new task.

Since the walker task observation space is different from the ones of the other two environments it is compared with, validating individuals using both the structures and the controllers already optimized and stored is impossible. For this reason, the morphologies of the best individuals have been used with no optimization, and the controller has been re-optimized in the new environment.

Nevertheless, in the carrier and pusher tasks individuals have also been evaluated using both the body and the controller already optimized in the original task. This has been possible since the observation space of the two tasks is the same.

4.1 Preliminary considerations

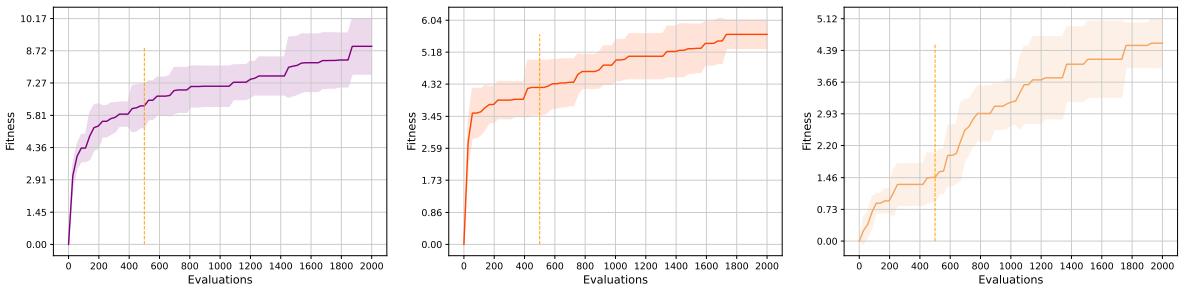
Figure 4.1 displays the fitness trends of experiments in the three tasks, using MAP-Elites to optimize the design, and PPO to optimize the controller.
The three trends are not comparable because of the different reward definition of the three tasks.

The pusher and the carrier are expected to be good walkers, since they both have the implicit goal of learning how to walk, other than interacting with an object. The opposite is not true: even though an individual is a good walker, he might be incapable of interacting with an object, since it only learned how to walk. It might not disappoint as a pusher, as a good morphology with a strong front leg might allow it to oppose to the box resistance. The same cannot be said for carrying a box, as it requires a good conformation at the top of the robot, to prevent the box from falling.

4.2 Task performance comparison

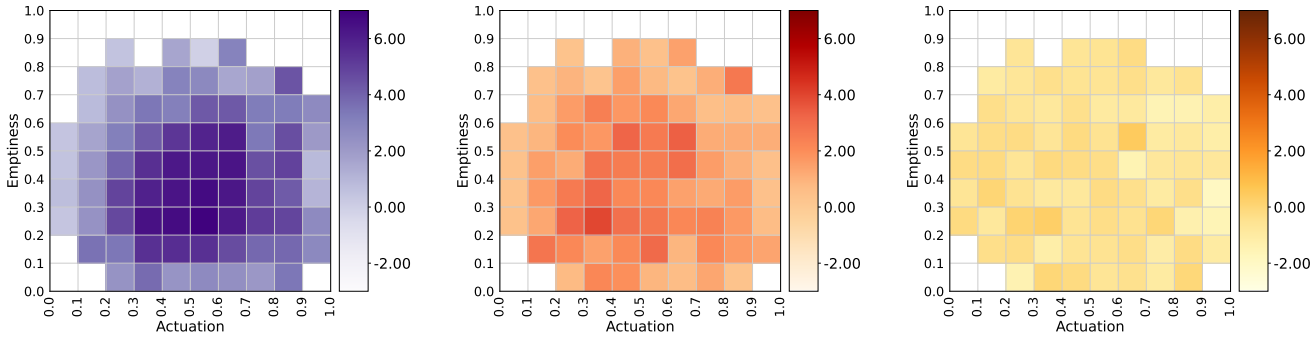
Is an individual who has only learnt how to walk able to push or carry an object? Can a pusher (carrier) carry (push) an object as far as possible? How well can pushers and carriers walk? This section tries to answer to these questions by providing the performance map generated in a task and the ones obtained by training the same individuals in a different environment.

Note that the reward definition vary according to the task, therefore the results obtained by evaluating in different environments are not comparable. For more details about the reward definition of the three tasks, pleas refer to Chapter 1.



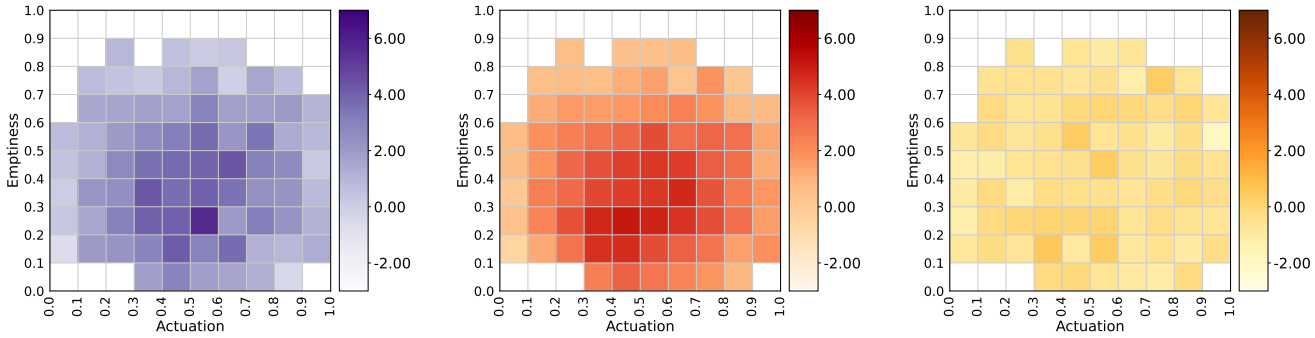
(a) Walker (b) Pusher (c) Carrier

Figure 4.1: Fitness trend in the three tasks.



(a) Walker (b) Pusher (c) carrier

Figure 4.2: Walker designs trained in the walker (a), pusher (b), carrier (c) task. The maps can be compared for the fitness distribution, however the obtained fitness cannot be compared, since its definition is different in the three tasks.



(a) Walker (b) Pusher (c) Carrier

Figure 4.3: Pusher designs trained in the walker (a), pusher (b), carrier (c) task. The maps can be compared for the fitness distribution, however the obtained fitness cannot be compared, since its definition is different in the three tasks.

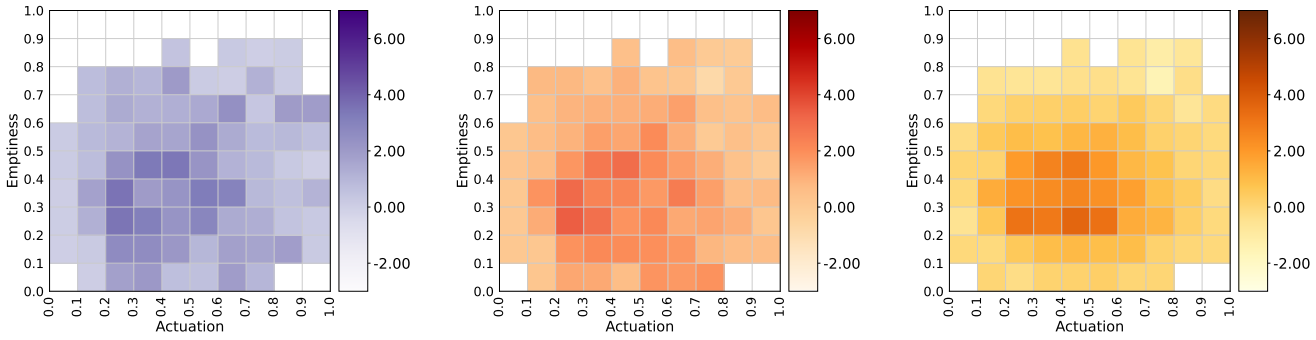
Figure 4.2 shows the performance maps generated by optimizing the controller of the best walker designs in the walking task itself (Map 4.2a), in the pusher and carrier task (Maps 4.2b and 4.2c respectively).
The obtained maps highlights that walkers might be good pushers, however the overall fitness in the carrier task is poor.
The results meet the expectations, since the ability of walking might help to push an object forward, but it doesn't provide any knowledge on how to carry a box.

The results of best pusher designs trained in the three tasks are shown in Figure 4.3. These individuals are well able of pushing an object, for which they had been designed for, and walking forward; however, once again, they get poor results in the carrier task. The maps obtained in the first two tasks are very similar, and this is because walking is an implicit goal while learning to push a box forward.

The best carrier designs obtain interesting, but not excellent, results in the walking and pushing environment, as shown in Figure 4.4. Map 4.4c highlights that the best performing individuals in the carrier task are the ones for which the body and the controller have both been trained in that same environment.

Table 4.1 provides the numerical results of these experiments. Each row defines where the designs of the individuals have been optimized, the columns refer to the environment on which the individuals have been evaluated.

Each cell indicates the mean and the standard deviation of the designs best performing in the environment in row, trained in the task in column. The value of each cell has been computed by considering all the individuals in the final map of six experiments, launched with different seeds. The



(a) Walker (b) Pusher (c) Carrier

Figure 4.4: Carrier designs trained in the walker (a), pusher (b), carrier (c) task. The maps can be compared for the fitness distribution, however the obtained fitness cannot be compared, since its definition is different in the three tasks.

respective mediated maps are the ones shown in Figures 4.2, 4.3, 4.4.

The table confirms that for each task the greatest rewards are obtained by the individuals for which the body and the controller have been optimized in that same environment.

Note the the results in row are not comparable, since each task has its own definition of reward, defined in Chapter 1.

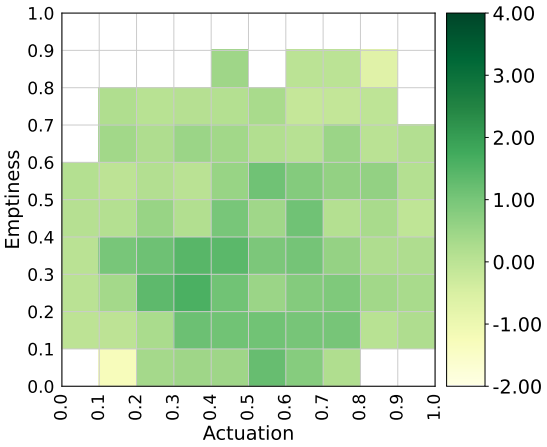
	Walker task	Pusher task	Carrier task
Walker	3.913 ± 2.21	1.7 ± 1.391	-0.602 ± 0.938
Pusher	2.297 ± 1.989	2.713 ± 1.533	-0.424 ± 0.975
Carrier	1.481 ± 1.634	1.275 ± 1.237	0.801 ± 1.286

Table 4.1: Comparison of task performance. Each row defines where the bodies have been optimized; each column refers to the environment on which the controllers have been trained, and where the individuals have been evaluated. Each cell indicates the mean and standard deviation. Please note the the results in row are not comparable since each task defines its own reward.

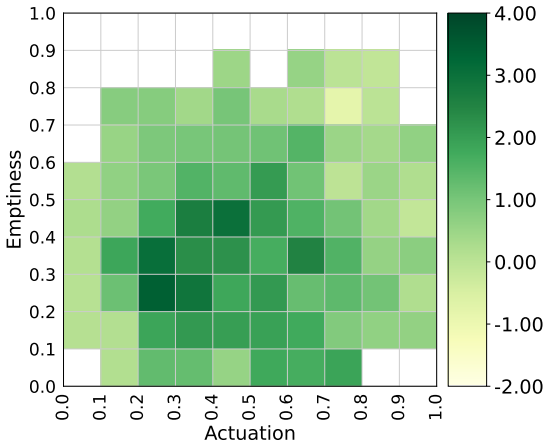
4.3 Validating or training in a new task

The carrier and pusher task have the same observation space, therefore it's possible to use both the body and the controller optimized in an environment on the other task.

The best performing carriers get a higher fitness value in the pusher task if their controller is trained in the new environment, as shown in Figure 4.5.
The first row in Table 4.2 confirms that the mean fitness is greater when the controller is optimized in the new environment, but it also shows that it has a greater standard deviation.

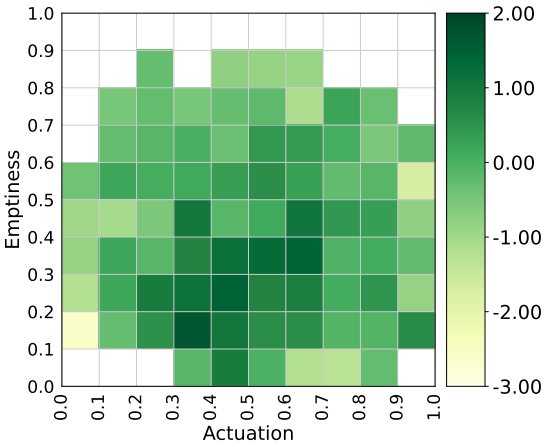


(a) Validation

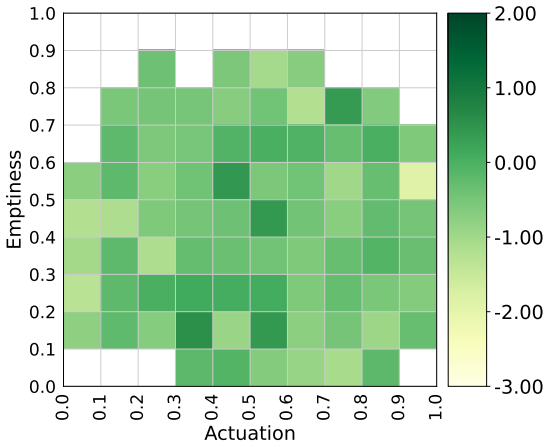


(b) Controller training

Figure 4.5: Performances of the best carrier designs in the pusher task. The the performances of individuals are improved when controllers are trained in the new task (b), compared with the validation only (a).



(a) Validation



(b) Controller training

Figure 4.6: Performances of the best pusher designs in the carrier task. The comparison shows that simply validating individuals (a) gets better results than training controllers (b) in the new task.

Figure 4.6 provides a comparison of the best pusher designs validating and optimizing the controller in the carrier task.
Contrary to the considerations for the carrier designs in the pusher task, the two maps highlight that the validated individuals perform better than the ones whose controller has been trained in the new task.
The results are confirmed by computing the mean of the fitness, as shown in the second row in Table 4.2. However, the table also points out that not optimizing the controller leads to a greater standard deviation.
The greater result with no further optimization is due to the pushers walking ability: since a component of the carrier task reward consists of the position of the robot, this ability allows it to get a good fitness value, despite the behaviour of the box.

	validation	optimization
carrier designs	0.54 ± 0.84	1.275 ± 1.237
pusher designs	0.124 ± 1.369	-0.424 ± 0.975

Table 4.2: Carrier and pusher multitask experiments. Mean and standard deviation for the experiments with validation only or with the controller optimization in the new task. The first row shows the results of the best carrier designs in the pusher task, the second row refers to the best pusher designs in the carrier task.

4.4 Conclusion

This chapter introduced some considerations about using the same designs in various tasks.

The best results in each environment are obtained when the body and the controller are both optimized in the same task, confirming that the robot co-optimization is strongly connected to the environment definition.

The pusher and carrier tasks showed that it's also possible to perform well on a new task even with no controller re-optimization; however, since it doesn't provide excellent results, further analysis might provide interesting and surely more precise information.

The reward definition of the task on which individuals are evaluated greatly affects the results. About this, it was shown that the best pusher designs can get a great reward in the carrier task even with no controller optimization in the new environment, due to their ability to walk, which increases the final reward value in spite of the penalty applied for dropping the box.

5 Conclusions

This work provided an overview of soft robot evolution using Evolution Gym benchmark.
Soft robotics is a promising brand new robotic research field. The main feature of soft robots, which is their deformability and adaptability to complex and unpredictable environments, makes them an interesting subject. Not only are they bio-inspired, but also they will have many future bio-related applications, and their malleability and adaptability will lead to an enhanced human-machine interaction.

A focus on the controller optimization showed that it has a key role in soft robot evolution, since it allows all the individuals to perform their best, even the ones with a disadvantaged morphology.

A comparison of two design algorithms, GA and MAP-Elites, showed that focusing on the mere fitness goal does lead to greater fitness values, but it generates similar bodies, precluding from exploring new potentially well-fitting morphologies. On the contrary, focusing on diversity allows the evaluation of very different robot designs, and it leads to only slightly lower performances.

One of the main reasons why soft robots are considered a promising field is their flexibility and adaptability, and therefore their potential application in different environments. This work revealed that even though the best results are obtained when both the controller and the body are optimized in the same task, it's also possible to get good results in a new environment simply by re-training the controller.
It was also shown that in some environments it's also possible to use the already optimized controller; this avoids the optimization costs and, in some cases, turned out to be even more effective. However, these results depend on the reward definition, and further experiments might take to more accurate results.

The robot bodies evolved autonomously often resembled existing natural creatures, even with no prior knowledge: for instance, they grew legs and set space to carry an object, according to the goal of the task they have been trained on. This demonstrates once again the strong connection that persists between artificial intelligence and biology.

Nevertheless, this work only focused on three easy tasks. The benchmark used proposes a great variety of tasks of different difficulty, therefore it would be interesting to extend the study of soft robot evolution to more complex environments.
The design optimization was limited to two evolutionary algorithms, but many other exist and might lead to interesting results.
What is more, only two body-related features were considered, the actuation and the emptiness, therefore a deeper study might analyze other features, like the energy consumption, relevant for future applications.

This work only provided an overview of soft robot evolution, but many encouraging aspects emerged yet. This is a promising field that deserves to be explored, and further studies will surely take steps towards an enhanced human-machine interaction.

Bibliography

[1] Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. `https://github.com/EvolutionGym/evogym`, 2021.

[3] L. Cazenille. Qdpy: A python framework for quality-diversity. `https://gitlab.com/leo.cazenille/qdpy`, 2018.

[4] Nick Cheney, Josh Bongard, and Hod Lipson. Evolving soft robots in tight spaces. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15. Association for Computing Machinery, 2015.

[5] Matteo Cianchetti. Embodied intelligence in soft robotics through hardware multifunctionality. *Frontiers in robotics and AI*, 8:724056–724056, Nov 2021.

[6] Matteo Cianchetti, Cecilia Laschi, Arianna Menciassi, and Paolo Dario. Biomedical applications of soft robotics. *Nature Reviews Materials*, 3(6):143–153, Jun 2018.

[7] A. E. Eiben and J. E. Smith. *What Is an Evolutionary Algorithm?*, pages 25–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[8] Andrea Ferigo, Giovanni Iacca, and Eric Medvet. Beyond body shape and brain: Evolving the sensory apparatus of voxel-based soft robots. In Pedro A. Castillo and Juan Luis Jiménez Laredo, editors, *Applications of Evolutionary Computation*, pages 210–226, Cham, 2021. Springer International Publishing.

[9] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[10] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

[11] S. Mintchev, D. Zappetti, J. Willemin, and D. Floreano. A soft robot for random exploration of terrestrial environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7492–7497, 2018.

[12] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015.

[13] Jørgen Nordmoen, Frank Veenstra, Kai Olav Ellefsen, and Kyrre Glette. Quality and diversity in evolutionary modular robotics. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2109–2116. IEEE, 2020.

[14] Daniela Rus and Michael T. Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, May 2015.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[16] Yun Seong Song, Yi Sun, Rubia van den Brand, Joachim von Zitzewitz, Silvestro Micera, Grégoire Courtine, and Jamie Paik. Soft robot for gait rehabilitation of spinalized rodents. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[17] Boyu Zhang, Yingwei Fan, Penghui Yang, Tianle Cao, and Hongen Liao. Worm-like soft robot for complicated tubular environments. *Soft Robot.*, 6(3):399–413, June 2019.