Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm

Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret

***Abstract*—The recently introduced Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) is an evolutionary algorithm capable of producing a large archive of diverse, high-performing solutions in a single run. It works by discretizing a continuous feature space into unique regions according to the desired discretization per dimension. While simple, this algorithm has a main drawback: it cannot scale to high-dimensional feature spaces since the number of regions increase exponentially with the number of dimensions. In this paper, we address this limitation by introducing a simple extension of MAP-Elites that has a constant, pre-defined number of regions irrespective of the dimensionality of the feature space. Our main insight is that methods from computational geometry could partition a high-dimensional space into well-spread geometric regions. In particular, our algorithm uses a centroidal Voronoi tessellation (CVT) to divide the feature space into a desired number of regions; it then places every generated individual in its closest region, replacing a less fit one if the region is already occupied. We demonstrate the effectiveness of the new "CVT-MAP-Elites" algorithm in high-dimensional feature spaces through comparisons against MAP-Elites in maze navigation and hexapod locomotion tasks.**

***Index Terms*—MAP-Elites; illumination algorithms; quality diversity; behavioral diversity; centroidal Voronoi tessellation**

I. Introduction

Evolution started from a common ancestor [1] and gave rise to the biodiversity we see today, which is estimated to amount to 1 trillion species [2]. Inspired by this observation, the field of evolutionary robotics has recently seen a shift from evolutionary algorithms (EAs) that aim to return a single, globally optimal solution, to algorithms that explicitly search for a very large number of diverse, high-performing individuals [3], [4], [5], [6], [7], [8], [9], [10].

EAs have traditionally been used for *optimization* purposes [12] with the aim of returning a single solution that corresponds to the global optimum of the underlying search space (Fig. 2A). Various forms of diversity maintenance (niching) techniques have been designed to supply EAs both with the ability to avoid premature convergence to local optima and to perform *multimodal optimization* [13], [14], [15], [16], [17]. In the latter case, the aim is to return multiple solutions that correspond to the peaks of the search space (Fig. 2B).

All authors have the following affiliations: (1) Inria, Villers-lès-Nancy, F-54600, France; (2) CNRS, Loria, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France; (3) Université de Lorraine, Loria, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France; (e-mails: {vassilis.vassiliades, konstantinos.chatzilygeroudis, jean-baptiste.mouret}@inria.fr).

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Project: ResiBots, grant agreement No 637972).

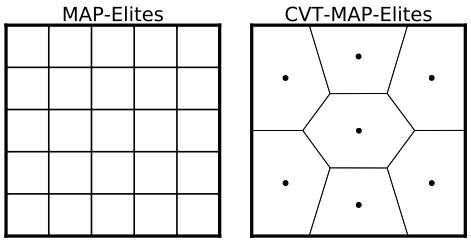


Fig. 1: MAP-Elites discretizes the feature space according to some pre-specified number of discretizations per dimension. This means that the number of niches grow exponentially with the number of additional dimensions or discretizations, thus, it cannot be used in high-dimensional feature spaces. In contrast, Centroidal Voronoi Tessellation (CVT) - MAP-Elites, uses a CVT [11] to partition the feature space into k homogeneous geometric regions, where k is the pre-specified number of niches. Here, MAP-Elites uses 5 discretizations per dimension, resulting in 25 niches, whereas, CVT-MAP-Elites uses 7 niches.

A recent, alternative perspective in the field of evolutionary computation views EAs more as *diversifiers*, rather than as optimizers [8]. Research on such EAs was initiated by the introduction of the Novelty Search (NS) algorithm [19], [20], which looks for individuals that are different from those previously encountered in some *behavior space* or *feature space* (we use both terms interchangeably). The behavior space is defined by features of interest that describe the possible behaviors of individuals over their lifetime [21], [8]. For example, points in this space could be the final positions of a simulated robot in a 2D maze environment. By rewarding novelty instead of fitness, NS promotes behavioral diversity and accumulates potential stepping stones for building more complex solutions [22]. This algorithm relies on the intuition that it can be more beneficial to encourage exploration in the behavior space rather than the genotype space, which is confirmed experimentally for several domains [23], [24].

Yet, purely exploring the behavior space without considering the task performance is not practical in many cases. For example, we might not only be interested in finding controllers that make a robot reach different points in the environment, but also the fastest controller for each point. To address this issue, algorithms like NS with Local Competition [3] and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [6] explicitly maintain a large number of niches while optimizing

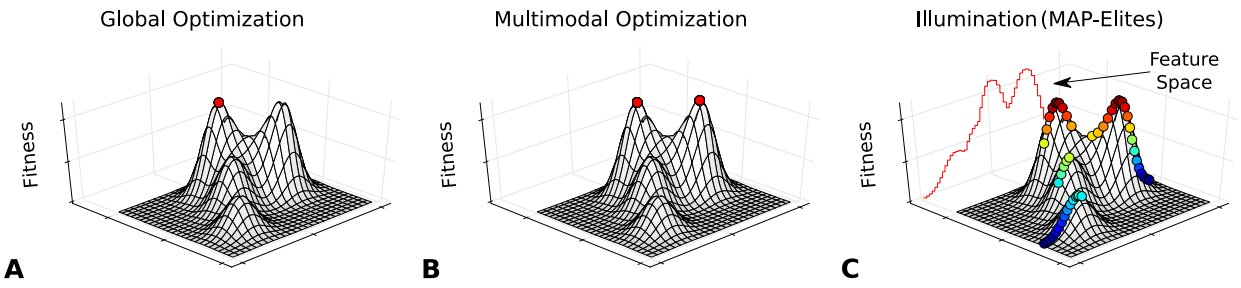


Fig. 2: Algorithms for global optimization aim to find a single global optimum of the underlying parameter space in which they operate (**A**). Multimodal optimization algorithms, on the other hand, aim at finding multiple optima (**B**). Illumination algorithms, such as MAP-Elites (**C**) go one step further by aiming to discover significantly more solutions, each one being the *elite* of some local neighborhood defined in some *feature space* of interest. For finding the solutions of the function illustrated, we used (**A**) Covariance Matrix Adaptation Evolution Strategies [18], (**B**) Restricted Tournament Selection [14], and (**C**) MAP-Elites [6].

them locally. For this reason they are called *illumination* algorithms [6] or *quality diversity* (QD) algorithms [7], [8]. It is important to note that in multimodal optimization, the primary interest is *optimization*, i.e., we are interested in maintaining diversity in order to find the peaks of the underlying genotype or phenotype space. On the contrary, the primary goal of illumination algorithms is *not* optimization, but diversity [8].

Whereas NS and its variants are governed by dynamics that continually push the population to unexplored regions in behavior space, MAP-Elites (Alg. 2) uses a conceptually simpler approach: it discretizes the d-dimensional behavior space into unique bins (Fig. 1 left) by dividing each dimension into a specified number of ranges $(n_1, ..., n_d)$; it then attempts to fill each of the $\prod_{i=1}^{d} n_i$ bins through a variation-selection loop with the corresponding genotype and its fitness, replacing a less fit one if it exists. The final result is an *archive* that contains the *elite* solution of every niche in behavior space. Being an illumination algorithm, MAP-Elites aims to return more information than multimodal optimization algorithms, i.e., solutions that are the best in their local region, but not necessarily ones where the fitness gradient is (near) zero (Fig. 2C). The maximum number of solutions that can be returned (i.e., the number of niches) is controlled by the number of discretization intervals (i.e., $k = \prod_{i=1}^{d} n_i$) and is a *user-defined input* to the algorithm. Put differently, the problem solved by MAP-Elites is "find k (e.g., 10,000) solutions that are as different and as high-performing as possible".

MAP-Elites has been employed in many domains. For instance, it has been used to produce: behavioral repertoires that enable robots to adapt to damage in a matter of minutes [25], [26], perform complex tasks [27], or even adapt to damage while completing their tasks [28], [29]; morphological designs for walking "soft robots", as well as behaviors for a robotic arm [6]; neural networks that drive simulated robots through mazes [7]; images that "fool" deep neural networks [30]; "innovation engines" able to generate images that resemble natural objects [31]; and 3D-printable objects by leveraging feedback from neural networks trained on 2D images [32].

The *grid-based* approach of MAP-Elites requires the user to only specify the number of discretization intervals for each dimension, making the algorithm conceptually simple and straightforward to implement. However, this approach suffers from the curse of dimensionality, since the number of bins increase exponentially with the number of feature dimensions. The increase in the number of niches results in reduced selective pressure, making the algorithm unable to cope with high-dimensional feature spaces even when memory is not a problem. For this reason, MAP-Elites has only been employed in settings with low-dimensional feature spaces (2 to 6 dimensions). However, scaling to high dimensions is a desirable property, as this would potentially allow MAP-Elites to be used with more expressive descriptors and create archives of better quality and diversity.

A way to address this limitation is by employing a method that maximally spreads a desired number of niches in feature spaces of arbitrary dimensionality. In this paper, we achieve this using a technique from computational geometry known as centroidal Voronoi tessellations (CVTs) [11]. In particular, the contribution of this paper is two-fold: (1) we introduce a new algorithm that we call "CVT-MAP-Elites" (Fig. 1 right) and demonstrate its advantage over MAP-Elites in a maze navigation task and the simulated hexapod locomotion task of [25]; (2) we propose a new methodology for assessing the quality of the archives produced by illumination algorithms.

II. Centroidal Voronoi Tessellation MAP-Elites

A Voronoi tessellation [33] is a partitioning of a space into geometric regions based on distance to k pre-specified points which are often called sites. Each region contains all the points that are closer to the corresponding site than to any other. If the sites are also the centroids of each region (and the space is bounded), then the Voronoi tessellation is the CVT [11] of the space (Fig. 1 right). CVTs have found application in problems ranging from data compression to modeling the territorial behavior of animals [11].

There exist various algorithms for constructing CVTs [35]. Lloyd's algorithm [36] is the most widely used in 2D spaces and consists of repeatedly constructing the Voronoi tessellation of the k sites, computing the centroids of the resulting Voronoi regions and moving the sites to their corresponding centroids. However, explicitly constructing Voronoi tessellations in high-dimensional spaces involves complex algorithms [33]. An

Algorithm 1 CVT approximation (adapted from [34])

1: **procedure** CVT(k)
2: $\mathcal{C} \longleftarrow$ sample_points(k) ▷ k random centroids
3: $S \longleftarrow$ sample_points(K) ▷ K random samples
4: **for** i $= 0 \longrightarrow max\_iter$ **do**
5: $\mathcal{I} \longleftarrow$ get_closest_centroid_indices($S, \mathcal{C}$)
6: $\mathcal{C} \longleftarrow$ update_centroids($\mathcal{I}$)
7: **return** centroids $\mathcal{C}$

alternative, simpler approach is to use Monte Carlo methods to obtain a close approximation to a CVT of the feature space [34]. In this work, we follow this approach and construct such an approximation using Alg. 1 (adapted from [34]). The algorithm first randomly initializes k centroids (line 2) and generates K random points ($K >> k$) (line 3) uniformly in the feature space according to the bounds of each dimension (e.g., the feature space could be defined in $[0, 1]^d$). The algorithm then alternates between assigning each point to the closest centroid and updating each centroid to be the mean of its corresponding points (lines 4-6). This procedure is analogous to using a clustering algorithm (such as k-means [37]) to find k clusters in a dataset that contains many well-spread points. Therefore, constructing a CVT can intuitively be seen as forcing the k sites to be well-spread in the space of interest.

Algorithm 2 MAP-Elites algorithm

1: **procedure** MAP-ELITES($[n_1, ..., n_d]$)
2: $(\mathcal{X}, \mathcal{P}) \longleftarrow$ create_empty_archive($[n_1, ..., n_d]$)
3: **for** $i = 1 \rightarrow G$ **do** ▷ *Initialization: G random* $\mathbf{x}$
4: $\mathbf{x} =$ random_solution()
5: ADD_TO_ARCHIVE($\mathbf{x}, \mathcal{X}, \mathcal{P}$)
6: **for** $i = 1 \rightarrow I$ **do** ▷ *Main loop, I iterations*
7: $\mathbf{x} =$ selection($\mathcal{X}$)
8: $\mathbf{x}' =$ variation($\mathbf{x}$)
9: ADD_TO_ARCHIVE($\mathbf{x}', \mathcal{X}, \mathcal{P}$)
10: **return** archive $(\mathcal{X}, \mathcal{P})$
11: **procedure** ADD_TO_ARCHIVE($\mathbf{x}, \mathcal{X}, \mathcal{P}$)
12: $(p, \mathbf{b}) \longleftarrow$ evaluate($\mathbf{x}$)
13: $c \longleftarrow$ get_cell_index($\mathbf{b}$)
14: **if** $\mathcal{P}(c) = null$ or $\mathcal{P}(c) < p$ **then**
15: $\mathcal{P}(c) \longleftarrow p$, $\mathcal{X}(c) \longleftarrow \mathbf{x}$

CVT-MAP-Elites partitions the d-dimensional feature space into k Voronoi regions and then attempts to fill each of the regions through a selection-variation loop. Algorithmically, it first obtains the coordinates of the k centroids ($\mathcal{C}$; Alg. 3, line 2) by constructing the CVT as described above (Alg. 1). It then creates an empty archive with capacity k ($\mathcal{X}$ and $\mathcal{P}$ store the genotypes and performances, respectively). The algorithm then evaluates G random parameter vectors ($\mathbf{x}$), simultaneously recording their performance (p) and feature *descriptor* ($\mathbf{b}$), i.e., their location in feature space (Alg. 3, 4-6). Next, it finds the index (c) of the centroid in $\mathcal{C}$ that is closest to $\mathbf{b}$ (Alg. 3, line 14), which implicitly gives information about its Voronoi region. If the region is free, then the algorithm stores the parameter vector $\mathbf{x}$ in that region; if it is already occupied,

Algorithm 3 CVT-MAP-Elites algorithm

1: **procedure** CVT-MAP-ELITES(k)
2: $\mathcal{C} \longleftarrow$ CVT(k) ▷ Run CVT and get the centroids
3: $(\mathcal{X}, \mathcal{P}) \longleftarrow$ create_empty_archive(k)
4: **for** $i = 1 \rightarrow G$ **do** ▷ *Initialization: G random* $\mathbf{x}$
5: $\mathbf{x} =$ random_solution()
6: ADD_TO_ARCHIVE($\mathbf{x}, \mathcal{X}, \mathcal{P}$)
7: **for** $i = 1 \rightarrow I$ **do** ▷ *Main loop, I iterations*
8: $\mathbf{x} =$ selection($\mathcal{X}$)
9: $\mathbf{x}' =$ variation($\mathbf{x}$)
10: ADD_TO_ARCHIVE($\mathbf{x}', \mathcal{X}, \mathcal{P}$)
11: **return** archive $(\mathcal{X}, \mathcal{P})$
12: **procedure** ADD_TO_ARCHIVE($\mathbf{x}, \mathcal{X}, \mathcal{P}$)
13: $(p, \mathbf{b}) \longleftarrow$ evaluate($\mathbf{x}$)
14: $c \longleftarrow$ get_index_of_closest_centroid($\mathbf{b}, \mathcal{C}$)
15: **if** $\mathcal{P}(c) = null$ or $\mathcal{P}(c) < p$ **then**
16: $\mathcal{P}(c) \longleftarrow p$, $\mathcal{X}(c) \longleftarrow \mathbf{x}$

then the algorithm compares the performance values and keeps only the best parameter vector (Alg. 3, 15-16). Once this is done, CVT-MAP-Elites iterates a simple loop (Alg. 3, 7-10): (1) randomly select one of the occupied regions to obtain the stored $\mathbf{x}$, (2) add some random variation on $\mathbf{x}$ to obtain $\mathbf{x}'$, (3) record the performance and feature descriptor, and (4) attempt to insert the new parameter in the corresponding region.

The differences between MAP-Elites and CVT-MAP-Elites are highlighted in Alg. 2 and Alg. 3: MAP-Elites creates an empty archive based on the desired discretizations per dimension $[n_1, ..., n_d]$ (Alg. 2, line 2), whereas CVT-MAP-Elites first performs the CVT construction and then creates the empty archive of k niches (Alg. 3, line 2,3). MAP-Elites calculates the index c of a descriptor $\mathbf{b}$ in $O(1)$ time (Alg. 2, line 13), whereas CVT-MAP-Elites needs to be equipped with a distance function for doing so (Alg. 2, line 14); the time complexity for the nearest neighbor query could be $O(\log k)$ on average [38], [33] or $O(k)$ in the worst case.

III. EVALUATION

To assess the scalability of CVT-MAP-Elites, we experiment with maze navigation and hexapod locomotion tasks. Both classes of tasks have been successful in testing the ability of illumination algorithms to explicitly generate archives that contain many diverse and high-performing solutions [25], [8].

A few metrics have been used in the literature to evaluate illumination algorithms [6], [7], [8], many of which utilize the MAP-Elites grid. For example, "coverage" [6] measures the expected number of cells an algorithm can fill using a specific descriptor, while "quality diversity score" [7], [8] projects the descriptors to a certain feature space and calculates the sum of the fitness values stored in each cell.

These metrics, however, have two disadvantages that prevent us from using them. First, they are dependent on the behavior space and a particular discretization of MAP-Elites, whereas we would like to compare not only different EAs, but also spaces of different dimensionality (e.g., 2D vs 1000D).

Second, they do not explicitly assess if the archives contain the "right" diversity, where "right" here is task-specific. For example, in our maze navigation experiments, we would like the resulting archives to contain controllers that take the robot from the starting position to the goal using different trajectories. In our hexapod locomotion experiments, we are interested in collecting diverse and high-quality solutions that would be useful even if the dynamics of the robot change due to some damage.

Since the aforementioned metrics cannot work in our case, we propose the following methodology for assessing the quality of the archives produced by each EA-descriptor pair P_i: in an analogy with supervised learning scenarios, training is done during the standard evolutionary phase, while testing for generalization is done during an *evaluation* phase in settings *not* experienced during evolution. Each evaluation setting slightly modifies the simulator in order to test whether different classes of behaviors are found by P_i, without however changing the fitness function (i.e., the way individuals are rewarded). For example, in the maze navigation experiments, an evaluation scenario e would correspond to changing the environment by blocking certain paths of the maze in order to test whether controllers that achieve a particular trajectory are found by P_i. In the hexapod locomotion experiments, e would correspond to changing the dynamics of the robot by removing a leg, in order to test whether P_i has found controllers that perform well in spite of this damage. For each P_i, we generate multiple archives from independent evolutionary trials, in order to make statistical assessments, and use the following metrics:

A. Expected best performance of an EA-descriptor pair in an evaluation scenario e

In order to calculate the "best performance" of an archive returned by P_i, we exhaustively[1] evaluate all solutions contained in the archive by simulating the given evaluation scenario e, and return the fitness value of the fittest solution. To calculate the "expected best performance" of P_i on e, we perform this procedure on the archives returned from all evolutionary runs and take the median value. For example, in the maze experiment where e corresponds to allowing only a single open path towards the goal, this measure would capture whether P_i was able to find at least one solution (controller) that makes the robot follow this path, even if this solution is the only one in the archive that achieves this. Intuitively, this metric asks: can P_i find a solution for a test problem?

B. Expected quality of an EA-descriptor pair

The "expected best performance" metric is the extreme case of a more general metric that asks: how many solutions can P_i find for a test problem? Since in our case "solving a problem" is not a discrete event, i.e., the fitness values are continuous variables, we can define this metric to be the probability that the fitness value X drawn from the archives produced by P_i is less than or equal (if we are minimizing; greater if we are maximizing) to a certain value x, where x is task-specific. If we consider not just one value for x, but a range of values, we can generalize this metric by generating the cumulative distribution function (CDF), for a minimization problem, or the complementary CDF (CCDF), for a maximization problem. That is:

- CDF of P_i: $F_{X\sim P_i}(x) = P(X \leq x)$
- CCDF of P_i: $\widetilde{F}_{X\sim P_i}(x) = P(X > x)$

We calculate these functions by first creating a set of possible fitness values for the given task (e.g., $x \in \{0, 1, 2, ..., 400\}$), and then for each of these values (x) we calculate the ratio of solutions from the archive that have a fitness value less than or equal to x in the case of CDF, or greater than x in the case of CCDF. We perform this procedure on the archives returned from all independent evolutionary runs of P_i (in order not to depend on a particular archive), as well as all evaluation scenarios, and record the median ratio for each possible fitness value. For example, if there are 20 evolutionary runs and 10 evaluation scenarios, this means that the median ratio is calculated over $20 \times 10 = 200$ numbers, for each possible fitness value.

By querying the CDF or CCDF for a certain fitness value that is considered "good" in a given problem is akin to asking: what is the expected percentage of good solutions returned by P_i? For example, suppose that we are comparing P_1 and P_2 in our hexapod locomotion task (i.e., maximize walking speed). If we consider a walking speed of at least 0.3 m/s to be well-performing in our task, we can query the CCDF tables of P_1 and P_2 at the index that corresponds to the value of 0.3 m/s, and get their output values $F_{P_1}(0.3)$ and $F_{P_2}(0.3)$. If $F_{P_1}(0.3) = 0.4$, this means that *randomly picking a solution from an archive returned by* P_1 *has a* 0.4 *chance of being one with a walking speed of at least 0.3 m/s*. If $F_{P_2}(0.3) < F_{P_1}(0.3)$, this means that P_2 is not as effective as P_1 for the same performance level, thus, we can say that the quality of P_2 is lower than the quality of P_1 for a walking speed of at least 0.3 m/s in the considered scenario.

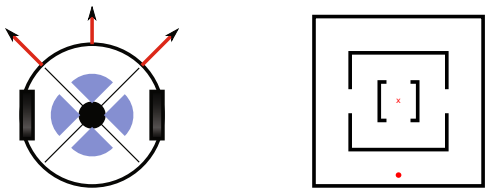
IV. Maze navigation experiments

A. Experimental Setup

1) Simulation and Fitness Function: We use a maze navigation task where a simulated mobile robot (Fig. 3a) is controlled by an artificial neural network whose architecture and parameters are evolved (for parameter settings, see suppl. material, Sec. SIII). The robot starts from the bottom of the arena and needs to reach the goal point at the center (Fig. 3b). At every simulation step, the Euclidean distance between the current position of the robot and the goal point is measured. The fitness function is the smallest distance achieved over the robot's lifetime [39], which is set to 3000 simulation steps.

2) Evaluation phase: We evaluate the archives of the EA-descriptor pairs in 16 different environments (shown in Fig. 4). These environments are created by selectively blocking the openings of the "open" maze environment to effectively allow only one realizable trajectory to the goal per environment.

[1]In our experiments, in case an archive contains more than 10k solutions, we randomly select and evaluate 10k of them to reduce evaluation time.



(a) (b)

Fig. 3: (a) Overview of the simulated mobile robot (diameter: 20 units). The robot is equiped with 3 laser range finder sensors (arrows) that return the normalized distance to the closest obstacle in that direction, and 4 pie-slice sensors that act as a compass towards the goal. The neural network controls the two motors that move the robot by setting their speed (a value between $[-2, 2]$ units per simulation step). (b) The "open" maze environment (size: 1000×1000 square units) employed in our experiments. The circle denotes the starting position and the cross denotes the goal.

3) Behavioral Descriptors: We use 6 behavioral descriptors of increasing dimensionality, that correspond to sampling (x,y) points along the trajectory of the robot. These are: 2D (end-point of the trajectory)[2], 10D (trajectory length = 5), 20D (trajectory length = 10), 50D (trajectory length = 25), 250D (trajectory length = 125), and 1000D (trajectory length = 500). MAP-Elites can only be used with the 2D, 10D and 20D descriptors, as more dimensions require more RAM than what is available. For example, in the 50D case, MAP-Elites requires 4096 TB of RAM just to store the matrix of pointers (not even the contents of the cells). For the 2D, 10D and 20D descriptors, we use a discretization of 71, 3 and 2 per dimension, resulting in 5041, 59049 and 1048576 cells, respectively. For CVT-MAP-Elites, we set the number of niches $k = 5000$ (see suppl. material Sec. SI and Fig. S2, S4A for a comparison between different values of k).

4) Generating centroids: The CVT algorithm relies on randomly sampled points that are clustered to find well-spread centroids. For the maze task, we cannot generate random trajectories by straightforwardly connecting random points in the arena because such trajectories would not match the physical constraints of the robot, which cannot teleport from any point to another in a single time step; and we cannot use a basic random walk because it would need too many samples to cover the space well. Instead, we generate a random point of the trajectory (e.g., the 42nd time step) within the bounds that are physically possible with the robot (here, 42 times ± 2 units from the starting point, while staying within the bounds of the arena, that is, 1000×1000), then we generate another random point of the trajectory (e.g., the 23rd) with updated constraints (23 times ± 2 units from the starting point, and 19 times ± 2 units from the 42nd point), etc. We continue this process until we have chosen all the points of the trajectory.

B. Results

For all experiments, we use 30 independent evolutionary trials and 200k function evaluations. The results show that the expected best performance of both MAP-Elites with the 2D, 10D and 20D descriptors, and CVT-MAP-Elites with the additional 50D, 250D and 1000D descriptors remains constant over all the evaluation scenarios (Fig. 4). The median distance is less than 2 units in most scenarios, while for the 8^{th} and 14^{th} environments, the median distance is still less than 10 units which is equal to the radius of the robot.

The median percentage of filled niches (Fig. 5A) of MAP-Elites decreases sharply from 0.72 (2D) to 0.09 (10D) and 0.005 (20D). These values for CVT-MAP-Elites are 0.73 (2D), 0.46 (10D), 0.40 (20D), 0.38 (50D), 0.36 (250D) and 0.36 (1000D). We also measure the spread of solutions (s) for each archive $\mathcal{A}$ using the following formula:

$$s = \frac{\frac{1}{|\mathcal{A}|} \sum_{\mathbf{b} \in \mathcal{A}} d_{nn}(\mathbf{b}, \mathcal{A})}{\max_{\mathbf{b} \in \mathcal{A}} d_{nn}(\mathbf{b}, \mathcal{A})} \quad (1)$$

where $\mathbf{b}$ is the behavioral descriptor of a solution in $\mathcal{A}$ and $d_{nn}(\mathbf{b}, \mathcal{A})$ is the Euclidean distance of $\mathbf{b}$ to its nearest neighbor in $\mathcal{A}$. The median spread of solutions (Fig. 5B) for MAP-Elites increases from 0.34 (2D) to 0.37 (10D) and then decreases to 0.29 (20D). Interestingly, in CVT-MAP-Elites the solutions become more well-spread in higher dimensions: 0.34 (2D), 0.41 (10D), 0.40 (20D), 0.43 (50D), 0.44 (250D) and 0.45 (1000D).

The expected quality of the archives (Fig. 5C) for a distance of at most 20 units (diameter of the robot) is slightly higher with CVT-MAP-Elites (MAP-Elites vs CVT-MAP-Elites): 2D: 0.02 vs 0.02; 10D: 0.01 vs 0.02; 20D: 0.02 vs 0.03; for the additional descriptors of CVT-MAP-Elites these values are: 0.04 (50D) and 0.03 (250D, 1000D). Overall, these results indicate that CVT-MAP-Elites performs as well as MAP-Elites, but can scale to high dimensions.

V. Hexapod Locomotion Experiments

A. Experimental Setup

1) Simulation and Fitness Function: We use the hexapod locomotion task of [25] using the Dynamic Animation and Robotics Toolkit[3]. The controller is designed to be a simple, open-loop oscillator that actuates each servo by a periodic signal of frequency 1Hz, parameterized by 3 values: the amplitude of oscillation, its phase shift and its duty cycle (i.e., the fraction of each period that the joint angle is positive). Each leg has 3 joints, however, only the movement of the first 2 is defined in the parameters[4] [25]. Therefore, there are 6 parameters per leg, thus, 36 parameters for controlling the whole robot. The fitness function is the forward distance covered in 5 seconds (thus, we maximize walking speed).

2) Evaluation phase: During the evolutionary phase, we use a model of an intact robot to generate the archives, whereas, during the evaluation phase, we evaluate 6 damage cases that correspond to removing a different leg from the hexapod robot (see Fig. 7).

[2]See suppl. material, Fig. S1 for how the niches look like in 2D.

[3]DART can be found in https://github.com/dartsim/dart

[4]The control signal of the third servo of each leg is the opposite of the second one.

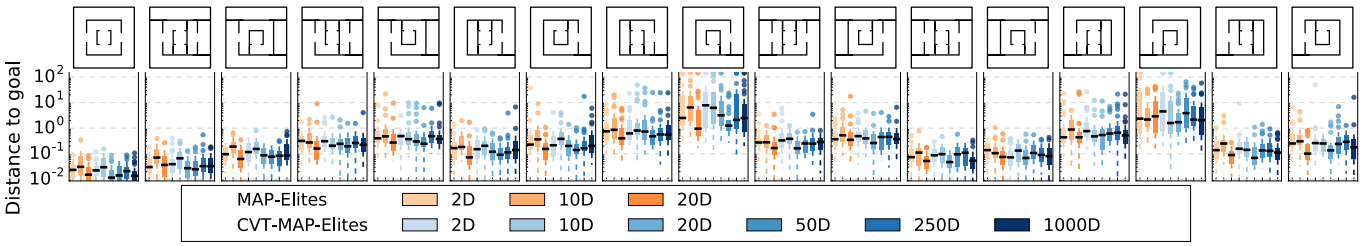


Fig. 4: Best performance (distance to the goal, thus, lower is better) for each algorithm-descriptor pair in the "open" maze environment (leftmost column) and all 16 evaluation environments which are created by selectively blocking certain paths that lead to the center of the maze (the goal). The box plots show the median (black line) and the interquartile range (25^{th} and 75^{th} percentiles) over 30 solutions; the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. In all cases, the median is below 10 units, indicating that the algorithms have good overall performance with all behavioral descriptors in these 1000×1000 square unit environments. The behavior of CVT-MAP-Elites does not deteriorate in high-dimensional spaces (e.g., 1000D), illustrating that the algorithm can scale well. The 8^{th} and 14^{th} evaluation environments, where there are big outliers, are symmetrical and seem to be the most difficult ones, as they require an almost full clockwise turn followed by an almost full counterclockwise turn and vice versa.

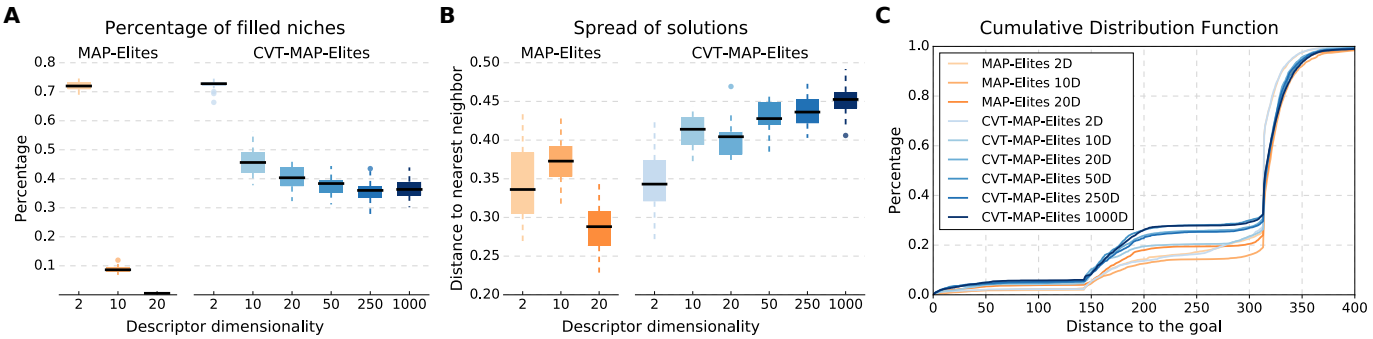


Fig. 5: **(A)** Percentage of filled niches. **(B)** Spread of solutions measured as normalized distance to the nearest neighbor. The box plots show the median (black line) and the interquartile range (25^{th} and 75^{th} percentiles) over 30 solutions; the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. **(C)** The cumulative distribution function calculated as the median from all evaluation scenarios.

3) Behavioral Descriptors: We use 4 behavioral descriptors of increasing dimensionality. For all experiments, CVT-MAP-Elites uses $k = 10$k niches (see suppl. material Sec. SI and Fig. S5, S7A for a comparison between different values of k).

- Duty Factor (6D): It is defined as the proportion of time that each leg is in contact with the ground:

$$\mathbf{b} = \left[\frac{\sum_{t=1}^{T} C_1(t)}{T}, \ldots, \frac{\sum_{t=1}^{T} C_6(t)}{T} \right] \in \mathbb{R}^6 \quad (2)$$

  where $\mathbf{b}$ is the descriptor, $C_i(t)$ denotes the Boolean value of whether leg i is in contact with the ground at time t (i.e., 1: contact, 0: no contact), recorded at each time step (every 15 ms) and averaged over the number of time steps T of the simulation. For MAP-Elites, we use 5 discretizations per dimension, resulting in 15625 bins.
- 12D subset of controller parameters that contains the phase shift for each of the 12 joints. For MAP-Elites, we use 3 discretizations per dimension (thus, 3^{12} bins).
- 24D subset of controller parameters that contains the amplitude of oscillation and the phase shift for each of the 12 joints. For MAP-Elites, we use 2 discretizations per dimension (resulting in nearly 17 million bins).
- Controller Parameters (36D): In this case, the behavior space is the same as the parameter space. MAP-Elites cannot be employed since even using 2 discretizations per dimension requires more than 256 GB of RAM.

B. Results

For all experiments, we use 20 independent evolutionary runs[5] and 75k generations. Overall, our results indicate that during the evolutionary phase (Fig. 6A), the median performance of the best individuals of CVT-MAP-Elites is approximately the same when using the 12D, 24D and 36D descriptors, despite the increase in dimensionality. When using the 6D descriptor, the performance is slightly better, however, this is likely due to the fact that this descriptor is calculated in behavior space, whereas the others in parameter space [23], [24]. With MAP-Elites, the performance of the best individuals when using the 6D descriptor is approximately the same as in CVT-MAP-Elites. However, when the dimensionality of the descriptor increases, the performance of MAP-Elites deteriorates significantly.

[5]We perform only 20 runs as each required more than 24 hours of computation time on modern (2015) 12-core Intel Xeon CPUs.

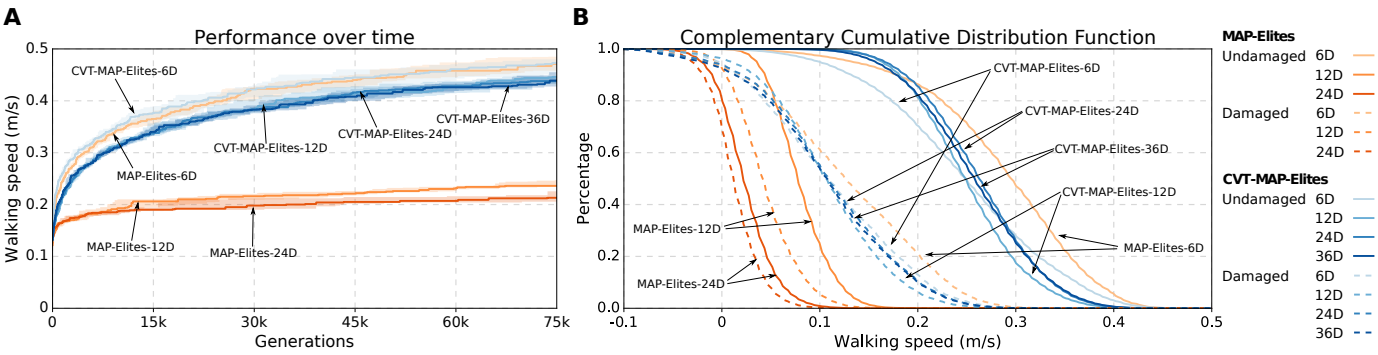


Fig. 6: **(A)** Median performance of the best solution found over the generations. The light zones represent the 25^{th} and 75^{th} percentiles. **(B)** The complementary cumulative distribution function for both the undamaged case (shown with *solid* lines) and all damage conditions (aggregated and shown with *dotted* lines). As the dimensionality of the descriptor increases, the performance of the solutions returned by MAP-Elites deteriorates. In contrast, CVT-MAP-Elites maintains the same level of performance, thus, scaling significantly better than MAP-Elites. Solutions with a negative walking speed signify that the hexapod robot moves backward.

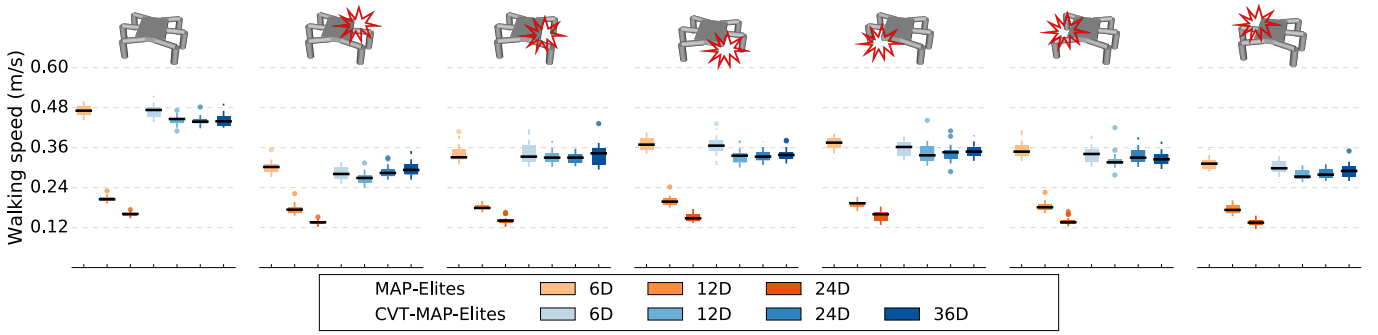


Fig. 7: Best performance (walking speed, thus, higher is better) for each algorithm-descriptor pair in the undamaged case and the 6 damage conditions which correspond to removing a different leg of the hexapod robot. The box plots show the median (black line) and the interquartile range (25^{th} and 75^{th} percentiles) over 20 solutions; the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. The performance of the best solutions found by MAP-Elites diminishes as the dimensionality of the descriptor increases. In contrast, CVT-MAP-Elites retains its performance with the increased dimensionality, thus, it scales significantly better than MAP-Elites.

In both the undamaged and damaged conditions, the expected best performance of MAP-Elites decreases with each increase in descriptor dimensionality, whereas the one of CVT-MAP-Elites remains relatively constant (MAP-Elites vs CVT-MAP-Elites; median gaits given in m/s): undamaged: [6D: 0.47 vs 0.47; 12D: 0.21 vs 0.45; 24D: 0.16 vs 0.44; 36D (CVT-MAP-Elites): 0.44]; damaged (average over 6 settings): [6D: 0.34 vs 0.33; 12D: 0.18 vs 0.31; 24D: 0.15 vs 0.32; 36D (CVT-MAP-Elites): 0.32].

The results of both the undamaged and damaged conditions indicate that MAP-Elites-6D has higher probability of finding better solutions than all other EA-descriptor pairs (Fig. 6B). CVT-MAP-Elites-6D does not perform as well, most likely due to the fact that it uses 10k niches during evolution, whereas MAP-Elites-6D uses 1.5 times more (5^6); this indicates that the number of niches for CVT-MAP-Elites could be better tuned, though this is beyond the scope of this study.

The expected quality of the archives produced by MAP-Elites significantly decreases with higher-dimensional descriptors, whereas with CVT-MAP-Elites it is not (Fig. 6B). For all the damaged cases and in particular for a walking speed of 0.2 m/s, the expected percentage of solutions returned by MAP-Elites that have at least this value is 21.3% for 6D, 0% for 12D and 0% for 24D; for CVT-MAP-Elites these are 13.3% for 6D, 6.5% for 12D, 10.8% for 24D and 10.5% for 36D. Comparing the two algorithms using the 12D and 24D descriptors reveals that the differences are highly significant ($p < 10^{-44}$, Mann-Whitney U test). Thus, randomly choosing among the solutions returned by both algorithms, we obtain higher quality ones with CVT-MAP-Elites, as the descriptor dimensionality increases.

VI. Discussion and Conclusion

We have shown that CVT-MAP-Elites can be applied in problems where the dimensionality is prohibitive for MAP-Elites (e.g., 1000 dimensions in the maze experiments). We have additionally demonstrated that in the hexapod locomotion tasks, CVT-MAP-Elites found gaits that were on average 1.7 to 2.1 times faster (during the evaluation settings) with the corresponding increase in feature space dimensionality. This is

because CVT-MAP-Elites has a more precise control over the balance between diversity and performance. For example, there is more selective pressure for performance when randomly selecting a parent from an archive of 1000 elites than from an archive of 1 million because the niches are bigger in the former case than in the latter (an elite from a big niche "reign" on more solutions). In the extreme case of a single niche, MAP-Elites and CVT-MAP-Elites would act like a stochastic hill climber, that is, with a very strong pressure for performance. In MAP-Elites the increase in dimensionality exponentially increases the number of niches, and as these niches get filled with solutions, selective pressure for performance decreases. In CVT-MAP-Elites, by having fewer niches (thus, solutions) and keeping the same selection method (e.g., 100 parents uniformly at random at every generation), we effectively increase selective pressure for performance.

In all our experiments, we chose to perform distance calculations using the Euclidean norm because it is the distance function used in other quality diversity studies [40], [7], [3]; however, "fractional" norms (Minkowski distance of order $p < 1$) might be more appropriate for high-dimensional spaces, because they provide a better contrast between the farthest and the nearest neighbor [41], [42]. Nevertheless, additional experiments revealed that results are not qualitatively affected by a change to fractional norms, and, in particular, that there is no significant difference between a Euclidean and a fractional norm when considering the generalization performance (see suppl. material, Fig. S3, S4B, S6, S7B).

CVT-MAP-Elites is a natural extension of MAP-Elites since it behaves like the latter in low-dimensional spaces, if given the same amount of well-spread niches. In addition, it does not require any major modifications over MAP-Elites. This is because the CVT routine (not part of the main EA) is responsible for generating the centroids and only needs to run once, before the EA starts; thus, the EA only needs to load these centroids. One can easily substitute the CVT routine with their own implementation, without any change in the EA. Interestingly, such a routine could be designed in a way that places more centers (thus, more variation) along certain dimensions. To ease deployment, we provide a python script for generating the centroids (see suppl. material, Sec. SII).

The computational complexity of the method we provide for constructing the CVT (in Alg. 1) is $O(ndki)$, where n is the number of d-dimensional samples to be clustered, k is the number of clusters and i is the number of iterations needed until convergence. From our experiments, we noticed that the clustering does not need to be very precise when using such a large number of clusters (i.e., thousands), thus, the number of iterations i can be fixed to limit the overall complexity of CVT-MAP-Elites. Nevertheless, one could resort to other CVT construction methods, which could offer significant speed-ups [35]. Furthermore, finding the centroid closest to a given behavior descriptor can be accelerated using data structures such as k-d trees [38] or others that exploit the characteristics of the Voronoi tessellation (e.g., see [43], [33]).

A key factor that enabled CVT-MAP-Elites to scale to 1000 dimensions in the maze experiments is the proper sampling of trajectories to generate the CVT and, as a consequence, the generation of centroids that more closely approximate trajectories that could be followed by the robot. A naive approach for sampling trajectories would sample each element along the trajectory from $[0, 1000]^2$ in the 1000×1000 arena. This, however, would result in unrealistic centroid trajectories, because the robot cannot move in a single time step to any point of the arena (it is constrained by its maximum speed). The behavior descriptors of the hexapod experiments do not have this problem, as they are properly defined in $[0, 1]^d$ (i.e., the behavior space is *dense*). Nevertheless, care needs to be taken when sampling the behavior space of interest during CVT construction: the centroids extracted from the samples have to be representatives of potential behaviors that make sense in the task.

While it is easy to sample behavior descriptors in many evolutionary robotics tasks, there exist many others for which sampling realistic behavior descriptors might prove challenging, thus, complicating the use of CVT-MAP-Elites. This is typically the case for behavior descriptors that involve trajectories of dynamical systems in high-dimensional state spaces. To put it clearly, CVT-MAP-Elites allows MAP-Elites to scale up to high-dimensional spaces, but it is still a *grid-based* algorithm [40], which is a category of quality diversity algorithms that is especially effective when the behavior space can be divided beforehand [40]. By contrast, if the bounds of the space are unknown before evolution, or if the set of possible behavior descriptors is heavily constrained[6], then vanilla CVT-MAP-Elites is probably not the most appropriate algorithm.

This issue can be potentially mitigated by letting the solutions generated by the evolutionary algorithm define the bounds of the space of interest. Thus, instead of pre-calculating the CVT, we can periodically recalculate it throughout the evolutionary run based on whether the bounds have changed [44]. In addition, it is possible to change the bounding volume to a different shape than a hyper-rectangle, as well as to vary the density of the niches so that to have a greater number on the outer part of the volume: such a change could potentially put more pressure towards expanding the volume and explore novel regions in behavior space. At any rate, other quality diversity algorithms could be more effective when nothing is known about the behavior space beforehand [40]. In these cases, it is worth investigating algorithms that rely on distance computations between different generated points [45] (such as Novelty Search with Local Competition [3] or Restricted Tournament Selection [14]), rather than between points and fixed centroids.

REFERENCES

[1] D. L. Theobald, "A formal test of the theory of universal common ancestry," *Nature*, vol. 465, no. 7295, pp. 219–222, 2010.

[2] K. J. Locey and J. T. Lennon, "Scaling laws predict global microbial diversity," *Proc. Natl. Acad. Sci. U.S.A.*, p. 201521291, 2016.

[3] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *GECCO'11*. New York, NY: ACM, 2011, pp. 211–218.

[6]Typically, when the behavior space is *sparse*, i.e., many behavior descriptor combinations are impossible to obtain in the considered system.

[4] A. Cully and J.-B. Mouret, "Behavioral repertoire learning in robotics," in *GECCO'13*. New York, NY: ACM, 2013, pp. 175–182.

[5] J. Clune, J.-B. Mouret, and H. Lipson, "The evolutionary origins of modularity," in *Proc. R. Soc. B*, vol. 280. The Royal Society, 2013, p. 20122863.

[6] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[7] J. K. Pugh, L. B. Soros, P. A. Szerlip, and K. O. Stanley, "Confronting the challenge of quality diversity," in *GECCO'15*. New York, NY: ACM, 2015, pp. 967–974.

[8] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016, doi: 10.3389/frobt.2016.00040.

[9] A. Cully and J.-B. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evol. Comput.*, vol. 24, pp. 59–88, 2016.

[10] D. Smith, L. Tokarchuk, and G. Wiggins, "Rapid Phenotypic Landscape Exploration Through Hierarchical Spatial Partitioning," in *PPSN XIV*. Springer, 2016, pp. 911–920.

[11] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms," *SIAM review*, vol. 41, pp. 637–676, 1999.

[12] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Berlin, Heidelberg: Springer, 2015.

[13] S. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[14] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. of the 6th International Conf. on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann, 1995, pp. 24–31.

[15] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 97–106, 1998.

[16] S. Das, S. Maity, B.-Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization — a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, pp. 71–88, 2011.

[17] M. Preuss, *Multimodal optimization by means of evolutionary algorithms*. Springer, 2015.

[18] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, pp. 159–195, 2001.

[19] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Artificial Life XI*. Cambridge, MA: MIT Press, 2008, pp. 329–336.

[20] ——, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, pp. 189–223, 2011.

[21] S. Doncieux and J.-B. Mouret, "Behavioral diversity measures for evolutionary robotics," in *CEC*. IEEE, 2010.

[22] R. E. Lenski, C. Ofria, R. T. Pennock, and C. Adami, "The evolutionary origin of complex features," *Nature*, vol. 423, pp. 139–144, 2003.

[23] J.-B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evol. Comput.*, vol. 20, pp. 91–133, 2012.

[24] S. Doncieux and J.-B. Mouret, "Beyond black-box optimization: a review of selective pressures for evolutionary robotics," *Evolutionary Intelligence*, vol. 7, no. 2, pp. 71–93, 2014.

[25] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[26] D. Tarapore, J. Clune, A. Cully, and J.-B. Mouret, "How do different encodings influence the performance of the MAP-Elites algorithm?" in *GECCO'16*. New York, NY: ACM, 2016, pp. 173–180.

[27] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, "EvoRBC: Evolutionary Repertoire-based Control for Robots with Arbitrary Locomotion Complexity," in *GECCO'16*, 2016, pp. 93–100.

[28] K. Chatzilygeroudis, A. Cully, and J.-B. Mouret, "Towards semi-episodic learning for robot damage recovery," in *IEEE ICRA Workshop on AI for Long-Term Autonomy*, Stockholm, Sweden, 2016.

[29] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, "Reset-free Trial-and-Error Learning for Data-Efficient Robot Damage Recovery," *arXiv preprint arXiv:1610.04213*, 2016.

[30] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*. IEEE, 2015, pp. 427–436.

[31] ——, "Understanding Innovation Engines: Automated Creativity and Improved Stochastic Optimization via Deep Learning," *Evol. Comput.*, vol. 24, pp. 545–572, 2016.

[32] J. Lehman, S. Risi, and J. Clune, "Creative generation of 3D objects with deep learning and innovation engines," in *Proc. of the 7th Intern. Conf. on Comput. Creativity*. Paris, France: Sony CSL, 2016, pp. 180–187.

[33] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Handbook of Computational Geometry*. Elsevier, 2000, pp. 201–290.

[34] L. Ju, Q. Du, and M. Gunzburger, "Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations," *Parallel Computing*, vol. 28, no. 10, pp. 1477–1500, 2002.

[35] J. C. Hateley, H. Wei, and L. Chen, "Fast methods for computing centroidal Voronoi tessellations," *Journal of Scientific Computing*, vol. 63, no. 1, pp. 185–212, 2015.

[36] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[37] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. on Math. Statist. and Prob.*, vol. 1. Berkeley, CA: Univ. of Calif. Press, 1967, pp. 281–297.

[38] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, pp. 509–517, 1975.

[39] J.-B. Mouret, "Novelty-based multiobjectivization," in *New Horizons in Evolutionary Robotics*. Springer, 2011, pp. 139–154.

[40] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Trans. Evol. Comput.*, 2017.

[41] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International Conference on Database Theory*. Springer, 2001, pp. 420–434.

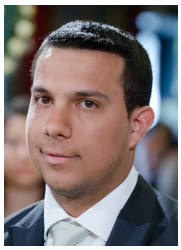
[42] S. Xia, Z. Xiong, Y. Luo, W. Xu, and G. Zhang, "Effectiveness of the Euclidean distance in high dimensional spaces," *Optik*, vol. 126, pp. 5614–5619, 2015.

[43] M. Sharifzadeh and C. Shahabi, "Vor-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries," *Proceedings of the VLDB Endowment*, vol. 3, pp. 1231–1242, 2010.

[44] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, "A comparison of illumination algorithms in unbounded spaces," in *GECCO'17 Companion*. New York, NY: ACM, 2017, pp. 1578–1581.

[45] ——, "Comparing multimodal optimization and illumination," in *GECCO'17 Companion*. New York, NY: ACM, 2017, pp. 97–98.

Vassilis Vassiliades obtained a BSc in Computer Science from the University of Cyprus in 2007, a MSc in Intelligent Systems Engineering from the University of Birmingham, UK in 2008, and a PhD in Computer Science from the University of Cyprus in 2015. Currently, he is a post-doctoral researcher at Inria (Nancy, France). His research interests lie in the field of bio-inspired artificial intelligence, focusing on reinforcement learning, neural networks and evolutionary computation.



Konstantinos Chatzilygeroudis received his BSc and MSc in Computer Science and Engineering from the Department of Computer Engineering and Informatics of University of Patras in Greece (in 2014). He is currently a PhD student at Inria University of Lorraine (Nancy, France). His work focuses on studying and providing abilities to autonomous robots to be able to compensate for unknown damages using reinforcement learning techniques and evolutionary computation.

Jean-Baptiste Mouret is a "research director" (senior scientist) at Inria (Nancy, France) and the principal investigator of an ERC grant (ResiBots – Robots with animal-like resilience, 2015-2020). He was previously an assistant professor at the Pierre and Marie Curie University (Paris, France), from where he also got his PhD. Overall, J.-B. Mouret conducts researches that intertwine machine learning and evolutionary computation to make robots more adaptive.