HAL
archives-ouvertes.fr

Learning behaviour-performance maps with meta-evolution

David Bossens, Jean-Baptiste Mouret, Danesh Tarapore

▶ **To cite this version:**

David Bossens, Jean-Baptiste Mouret, Danesh Tarapore. Learning behaviour-performance maps with meta-evolution. GECCO'20 - Genetic and Evolutionary Computation Conference, Jul 2020, Cancun, Mexico. hal-02555231

HAL Id: hal-02555231
https://hal.inria.fr/hal-02555231

Submitted on 27 Apr 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning behaviour-performance maps with meta-evolution

David M. Bossens
University of Southampton
d.m.bossens@soton.ac.uk

Jean-Baptiste Mouret
Inria, CNRS, Université de Lorraine
jean-baptiste.mouret@inria.fr

Danesh Tarapore
University of Southampton
d.s.tarapore@soton.ac.uk

ABSTRACT

The MAP-Elites quality-diversity algorithm has been successful in robotics because it can create a behaviorally diverse set of solutions that later can be used for adaptation, for instance to unanticipated damages. In MAP-Elites, the choice of the behaviour space is essential for adaptation, the recovery of performance in unseen environments, since it defines the diversity of the solutions. Current practice is to hand-code a set of behavioural features, however, given the large space of possible behaviour-performance maps, the designer does not know a priori which behavioural features maximise a map's adaptation potential. We introduce a new meta-evolution algorithm that discovers those behavioural features that maximise future adaptations. The proposed method applies Covariance Matrix Adaptation Evolution Strategy to evolve a population of behaviour-performance maps to maximise a meta-fitness function that rewards adaptation. The method stores solutions found by MAP-Elites in a database which allows to rapidly construct new behaviour-performance maps on-the-fly. To evaluate this system, we study the gait of the RHex robot as it adapts to a range of damages sustained on its legs. When compared to MAP-Elites with user-defined behaviour spaces, we demonstrate that the meta-evolution system learns high-performing gaits with or without damages injected to the robot.

CCS CONCEPTS

• **Computer systems organization** → *Evolutionary robotics*; • **Computing methodologies** → *Artificial intelligence*; *Evolutionary robotics*; *Learning paradigms*;

KEYWORDS

quality-diversity algorithms, behavioural diversity, meta-learning, evolutionary robotics, damage recovery

ACM Reference Format:
David M. Bossens, Jean-Baptiste Mouret, and Danesh Tarapore. 2020. Learning behaviour-performance maps with meta-evolution. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390181

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '20, July 8–12, 2020, Cancún, Mexico
© 2020 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390181

1 INTRODUCTION

Quality-diversity algorithms [23] evolve an archive of solutions which is, according to a user-defined behaviour space, as diverse as possible while obtaining for each solution a high performance. Two prototypical quality-diversity algorithms are: i) Novelty Search with Local Competition (NS-LC) [17], which optimises behavioural diversity itself on a behavioural distance function (Novelty Search [16]) and which ensures high-quality solutions based on a local competition among behaviourally similar solutions; and ii) Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [19], which records the highest-performing solutions (elites) for each local region in a discretised behaviour space called a behaviour-performance map. Quality-diversity algorithms have been applied widely, with applications in artificial life [17], video game design [7, 14], automated image generation [20], design of robot morphologies and controllers [11, 19, 21], and behaviour adaptation [3, 13].

In quality-diversity algorithms, the designer's choice of a suitable behaviour space or behavioural distance function is essential, as it defines behavioural diversity and therefore the types of solutions obtained. In some cases, the end-user is interested in a particular set of behavioural features and in this case, the behaviour space can be hand-coded based on these features of interest (see, for example [6, 11, 19, 21, 25]).

There has been a shift towards automated behaviour spaces. Recent methods have used auto-encoders to reduce the dimensionality of the behaviour space to avoid the exponential increase in the number of cells in the behaviour-performance maps as its dimensionality is increased [2, 4]. This line of research is part of a wider agenda to exploit deep neural networks for unsupervised learning of behavioural descriptors, although not fully realised [20].

Another line of research in automated behaviour spaces explores multiple behaviour spaces, thereby avoiding the need to select one single behavioural descriptor. Pugh et al. (2016) perform novelty search based on the Pareto-front arising from two different behavioural diversity metrics, illustrating that a behaviour space of interest can be combined efficiently with a behaviour space that drives evolution towards solutions on a deceptive maze [24]. Meyerson et al. (2016) propose an approach which *learns* behaviour spaces for Novelty Search [18]. Using a weight-vector that reflects the importance of each of its constituent features, a policy-based behaviour space is transformed into a new behavioural distance metric. In this approach, the weights are adapted with a fixed, heuristic rule based on a comparison of behaviours in the initial population to behaviours in a population that is found successful on the domain of interest. A more dynamic approach [5], predating quality-diversity algorithms and instead using multi-objective optimisation of behavioural distance and fitness, switches behavioural distance metrics randomly once every few generations, demonstrating that switching between behavioural distance metrics guides

evolution in a favourable manner compared to taking the mean across behavioural distance metrics.

Despite the progress in automated behavioural descriptors, there is a need for automated methods that optimise a particular objective *defined on the level of archives, rather than on the level of individual solutions*. For example, the user may be interested in evolving a behaviour-performance map that optimises a robot's adaptation to unexpected future events, such as damages to its actuators or sudden changes to its environment, or a map that optimises performance on a complex task by defining a discretised space of subtasks. In these cases, it is not known which behaviour space will optimise the meta-level objective. This paper studies a system that *evolves* behavioural descriptors based on an objective defined at the level of a behaviour-performance map – a meta-fitness function. The proposed method automates the choice of behavioural features and thereby learns how to learn behaviour-performance maps.

2 LEARNING BEHAVIOUR-PERFORMANCE MAPS WITH META-EVOLUTION

We develop a meta-evolution system, where each of the solutions evolved (meta-individuals) is a quality-diversity algorithm with its own behaviour space. As a quality-diversity algorithm, we use the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [19] which generates behaviour-performance maps by collecting the best individuals for each cell in a grid over the behaviour space. Unlike traditional MAP-Elites, the system is not limited to pre-defined behavioural features but instead learns the best weighted combination of a larger number of base-features, with the aim of optimising the adaptation potential of the resulting behaviour-performance map. Due to controlling the selection and replacement of individuals, the meta-individuals not only represent the behaviour-performance maps as an end-result but also actively guide evolution. The pseudo-code for our meta-evolution system is provided in Algorithm 1.

2.1 MAP-Elites algorithm

The MAP-Elites algorithm starts by creating an initial population of individuals, each with random genotypes. Each of the individuals in the initial population is then evaluated to obtain its fitness score f and behavioural descriptor β. After this evaluation, the individuals are added to the behaviour-performance map according to the following replacement rule: if the bin corresponding to the individual's behavioural descriptor is empty (i.e., $\mathcal{M}[\beta] = \emptyset$) or if the new individual has a higher fitness than the solution in that bin (i.e., $f > f(\mathcal{M}[\beta])$), place the individual's genotype $\mathbf{g}$ in that bin of the behaviour-performance map (i.e., $\mathcal{M}[\beta] \leftarrow \mathbf{g}$).

After initialisation, the algorithm applies repeated cycles of random selection, genetic variation, evaluation and replacement. Random selection is implemented by selecting individuals at random from the pool of currently filled bins in the behaviour-performance maps. Genetic variation is applied using a mutation operator. Evaluation of the solutions is based on the fitness function. Replacement is based on the above-mentioned replacement rule. After many repetitions of this cycle, the behaviour-performance map is gradually filled with a behaviourally diverse map with high-performing solutions.

Algorithm 1 Meta-evolution with CMA-ES.

1: $\mathcal{D} \leftarrow \emptyset$. ▷ Create empty database.
2: **for** $i = 1$ to p **do** ▷ Create initial database.
3: $\mathbf{g} \leftarrow$ `random-genotype()`.
4: $\mathbf{b}, f \leftarrow$ `eval`$(\mathbf{g})$. ▷ Base-features and fitness.
5: Insert $\langle \mathbf{g}, \mathbf{b}, f \rangle$ into $\mathcal{D}$. ▷ Fill the database.
6: **end for**
7: **for** $j = 1$ to G **do** ▷ Loop over meta-generations.
8: **for** $i = 1$ to λ **do**
9: Set $\mathcal{M}^i \leftarrow \emptyset$. ▷ Empty the map.
10: $\mathbf{w} \sim \mathcal{N}(\mathbf{m}, \sigma\mathbf{C})$. ▷ Sample map-genotype.
11: **for** $\langle \mathbf{g}, \mathbf{b}, f \rangle \in \mathcal{D}$ **do** ▷ Construct map from database.
12: `add-to-map`$(\mathcal{M}^i, \mathbf{w}, \mathbf{g}, \mathbf{b}, f)$.
13: **end for**
14: **end for**
15: **for** $i = 1$ to λ **do**
16: Perform `MAP-Elites-iterations`$(\mathcal{M}^i, \mathbf{w}^i)$.
17: $\mathcal{F}_i \leftarrow$ `Meta-fitness`$(\mathcal{M}^i)$.
18: **end for**
19: $\mathbf{m} \leftarrow$ `Update-mean()`. ▷ See Equation 3.
20: $\mathbf{C} \leftarrow$ `Update-covariance()`. ▷ See Equation 4.
21: $\sigma \leftarrow$ `Update-step()`. ▷ See Equation 5.
22: **end for**
23: **procedure** ADD-TO-MAP$(\mathcal{M}, \mathbf{w}, \mathbf{g}, \mathbf{b}, f)$
24: $\mathbf{W} \leftarrow$ `vec2mat`$(\mathbf{w})$. ▷ Convert to matrix.
25: $\beta \leftarrow \mathbf{W} \cdot \mathbf{b}$. ▷ Apply meta-genotype to get descriptor.
26: **if** $\mathcal{M}[\beta] = \emptyset$ **or** $f > f(\mathcal{M}[\beta])$ **then**
27: $\mathcal{M}[\beta] \leftarrow \mathbf{g}$. ▷ Add individual $\mathbf{g}$ to the map $\mathcal{M}$.
28: **end if**
29: **end procedure**
30: **procedure** MAP-ELITES-ITERATIONS$(\mathcal{M}, \mathbf{w})$
31: **for** $i = 1$ to I **do** ▷ I is the number of iterations
32: $\mathbf{g} \sim \mathcal{M}$. ▷ Sample genotype randomly from map.
33: $\mathbf{g}' \leftarrow$ `mutate`$(\mathbf{g})$. ▷ Mutation.
34: $\mathbf{b}, f \leftarrow$ `eval`$(\mathbf{g})$. ▷ Base-features and fitness.
35: `add-to-map`$(\mathcal{M}, \mathbf{w}, \mathbf{g}, \mathbf{b}, f)$.
36: Insert $\langle \mathbf{g}, \mathbf{b}, f \rangle$ into $\mathcal{D}$. ▷ Fill the database.
37: **end for**
38: **end procedure**

2.2 Meta-evolution with CMA-ES

To evolve meta-individuals, consisting of behaviour-performance maps, we make use of the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [8, 9]. The selection of this traditional evolutionary strategies method is due to its ability to learn functions characterised by noise, allowing to use a stochastic meta-fitness function, and due to the ability to learn with a limited population size, thereby allowing the meta-learning system to converge within a limited number of meta-generations.

Initialisation: The algorithm first applies an initialisation phase to populate the behaviour-performance maps at the first meta-generation. A total number of p random genotypes are created and then evaluated (see l. 2-6 in Algorithm 1; in our experiments, $p = 2000$, to ensure a large diversity of initial solutions). At this point, all the individual solutions are stored in a database $\mathcal{D}$. Each

entry in $\mathcal{D}$ is a tuple $\langle \mathbf{g}, \mathbf{b}, f \rangle$, where $\mathbf{g}$ is the low-level genotype (e.g., the parameters of a robot's controller), $\mathbf{b}$ is an extended behavioural description of the individual according to a large number of N_b user-defined behavioural 'base-features', and f is the fitness of the individual. The database is implemented as a large circular buffer which stores the last $|\mathcal{D}| = 500,000$ entries; this choice is based on a trade-off between the number of solutions retained and the consumption of memory and time to store and process, respectively, the solutions.

Meta-generations: After initialisation, the meta-evolution algorithm starts repeating meta-generations, applying CMA-ES to obtain behaviour-performance maps that are increasingly adaptive, i.e., high-performing on the map-level meta-fitness score $\mathcal{F}$ (see Section 3.2 for its implementation).

The meta-generation first constructs new maps $\mathcal{M}_i$, for each $i \in \{1, \ldots, \lambda\}$ in the meta-population (see l. 8-14 in Algorithm 1). A new meta-genotype $\mathbf{w} \in \mathbb{R}^n$ is generated based on a multivariate normal distribution,

$$\mathbf{w} \sim \mathcal{N}(\mathbf{m}, \sigma \mathbf{C}) \,, \tag{1}$$

where $\mathbf{m} \in \mathbb{R}^n$ is the mean meta-genotype, $\mathbf{C}$ is the covariance matrix, and $\sigma > 0$ is a scalar representing the step-size. Each meta-genotype $\mathbf{w} \in \mathbb{R}^n$ is a rowwise vectorisation of a matrix $\mathbf{W} \in \mathbb{R}^{N_f \times N_b}$, where N_b is the number of user-defined base-features and where N_f is a smaller number of features representing the features on which the behaviour-performance map will be based. To ensure each feature of the resulting behaviour-performance maps is in $[0, 1]$, a rowwise normalisation is performed such that $\sum_j^{N_b} W_{ij} = 1$ for all $i \in \{1, \ldots, N_f\}$.

The algorithm then applies for all database-entries $\langle \mathbf{g}, \mathbf{b}, f \rangle \in \mathcal{D}$ MAP-Elites' replacement rule (see l. 26-28 in Algorithm 1), but the behavioural descriptor β of the low-level genotype $\mathbf{g}$ is first computed by applying the meta-genotype $\mathbf{W}$ to obtain a weighted sum of the base-features (see l. 24-25 in Algorithm 1):

$$\beta \leftarrow \mathbf{W} \cdot \mathbf{b} \,. \tag{2}$$

Once the maps are constructed[1], the meta-individuals $i \in \{1, \ldots, \lambda\}$ independently apply MAP-Elites, each continuing to evolve their own behaviour-performance map $\mathcal{M}_i$ for a predetermined number of I iterations. During these iterations of MAP-Elites, the behavioural description of a low-level genotype is again computed according to Equation 2 and any new individuals arising from MAP-Elites' reproduction phase are added to the database (see l. 36 in Algorithm 1) and are subject to MAP-Elites' replacement rule for the evolution of the map (see l. 35 in Algorithm 1). After terminating MAP-Elites, each meta-individual is then evaluated based on a meta-fitness score $\mathcal{F}$, which represents the adaptation potential of its behaviour-peformance map to unseen contexts.

CMA-ES then updates the mean, covariance and step size parameters (see l. 19-21 in Algorithm 1) as is usually done in the $(\mu/\mu_W, \lambda)$-CMA Evolution Strategy [9], except that the objective of CMA-ES is based on the meta-fitness scores. The first step is to sort meta-individuals according to their meta-fitness, and then select a number of $\mu \leq \lambda$ individuals for reproduction. Reproduction involves mutating the mean towards the best of the selected meta-individuals:

$$\mathbf{m} \leftarrow \mathbf{m} + c_m \sigma \sum_{i=1}^{\mu} v_i (\mathbf{w}^i - \mathbf{m}) \,, \tag{3}$$

where $v_i > 0$, $\sum_{i=1}^{\mu} v_i = 1$; $\mathbf{w}^i$ is the i'th best meta-genotype; σ is the step size; and $c_m \in [0, 1]$ is a learning rate. Second, the covariance matrix is adapted based on a combination of the active rank-μ update [12], which exploits information from the entire population by assigning positive weights to highest-ranking individuals and negative weights to lowest-ranking individuals, and the rank-one update [10], which exploits the correlations between generations based on the evolution path:

$$\mathbf{C} \leftarrow \left(1 - c_1 - c_\mu \sum_j v_j\right) \mathbf{C} + c_\mu \sum_{i=1}^{\lambda} v_i \mathbf{s}_i \mathbf{s}_i^\intercal + c_1 \mathbf{p}_c \mathbf{p}_c^\intercal \,, \tag{4}$$

where c_μ and c_1 are positive weights reflecting the importance of the rank-μ and rank-one term, respectively; $\mathbf{s}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ is the difference of the sampled meta-genotype from the old mean, divided by the step size σ; v_i is a positive scalar in case $i \leq \mu$ and a negative scalar otherwise ; and $\mathbf{p}_c \in \mathbb{R}^n$ is the evolution path, a weighted sum of the past mutation steps. Finally, step-size is controlled:

$$\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma||_2}{\mathbb{E}\left[||\mathcal{N}(\mathbf{0}, \mathbf{I})||_2\right]} - 1\right)\right) \,, \tag{5}$$

where c_σ and d_σ are parameters that affect the damping and $\mathbf{p}_\sigma \in \mathbb{R}^n$ is the conjugate evolution path. The interpretation is the following: i) successive steps that are positively correlated increase the step size to reduce the number of steps to reach a promising region in search space; and ii) successive steps that are negatively correlated decrease the step size to avoid successive steps cancelling out each other.

3 EXPERIMENTAL SETUP

The aim of the experiments is to assess whether meta-evolution with CMA-ES can evolve an archive of robot controllers which adapts rapidly to a wide variety of damages to the robot's legs. We compare the proposed meta-evolution system to MAP-Elites with user-defined behavioural descriptors corresponding to the base-features and MAP-Elites with a randomly chosen combination of base-features. Source code corresponding to the experiments is available at https://github.com/resilient-swarms/meta-cmaes.

3.1 Simulation environment

As the robotic platform, we use the RHex robot (see Figure 1) [26], a hexapod robot which has been shown to be able to move robustly across many different terrains and at high speeds compared to its body frame. Its body and movements are modelled using the DART (Dynamic Animation and Robotics Toolkit) simulator [15]. The task of the Rhex robot is to walk in a straight line and the fitness function f is the total distance the robot has moved forward within a time span of 5 s, on a flat plain surface where the robot faces no damages or obstacles. The control cycle of the robot is set to 5 ms.

Each of the robots legs is controlled by a Buehler clock mechanism, which alternates between a stance phase where the robot leg touches the ground and a swing phase where the leg rotates above ground [26, 27]. The bottom-level genotype $\mathbf{g}$ is comprised

[1] Empirical tests show reconstructing $\lambda = 5$ behaviour-performance maps from the large database of 500,000 solutions consumes on average 1.4 s.

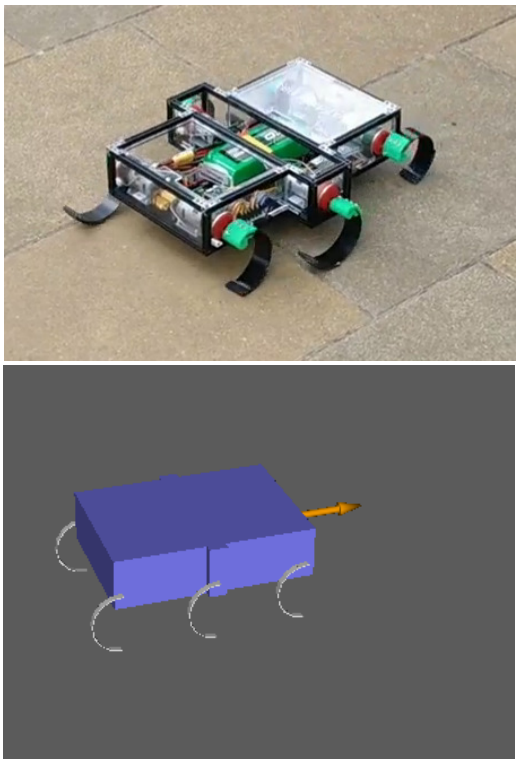


Figure 1: The RHex robot platform and simulation model. The robot is tasked with moving forward in a straight line.

of 24 parameters of the clocks. Parameter 1 defines the period of the clocks, T, between 0.33-1 s (clock speed within 1-3 Hz). For each leg, the following parameters are defined: the duty-factor, the proportion of a single period during which the leg touches the ground (parameters 2-7); the stance angle in $[0, \pi]$, the angle within which the robot leg touches the ground (parameters 8-13); the stance offset in $[-\pi/4, \pi/4]$, the angular offset to the leg's stance phase (parameters 14-19); the phase offset in $[0, T/2]$, the time lag with which the leg touches the ground when compared to leg 1 (parameters 20-24; reference leg 1 is excluded as parameter).

3.2 Experimental conditions

In the experiments, a total of five conditions are investigated.

In the **meta** condition, the meta-fitness of a map $\mathcal{M}$ is defined as

$$\mathcal{F}_{\text{damage}} = \mathbb{E}_{\mathbf{g}\sim\mathcal{M}}\left[\frac{1}{6|D|}\sum_{d\in D}\sum_{l=1}^{6} f(\mathbf{g}; d(l))\right], \tag{6}$$

where D defines a set of two damage-types, selected randomly before evolution and $f(\mathbf{g}; d(l))$ computes the fitness of the genotype $\mathbf{g}$ when a damage of type d is applied to leg l of the RHex robot. To limit computational expense, the expected value in Equation 6 is approximated by sampling, without replacement, 10% of genotypes in the behaviour-performance map. Although fault recovery is often achieved by taking the maximal performance on an individual damage across the archive, we use the performance of a random sample of individuals on diverse damages due to the following: i) evaluating all solutions in the archive would be too expensive; ii) repeatedly sampling the maximum does not represent the maximum across the map whilst repeatedly sampling the average represents the map's average reasonably well; and iii) optimising the expected value of performance on multiple damages may have additional benefits beyond adaptation, such as steering evolution to evolve robust, high-quality controllers. The behaviour-performance maps in the meta condition are based on 3-dimensional features that are weighted combinations of 15 base-features.

The base-features are based on three user-defined behavioural descriptors, which define three control conditions of the experiments. The **duty-factor** is a 6-dimensional descriptor which represents for each leg of the RHex robot the proportion of time it is in contact with the ground. The **body-orientation** is a 6-dimensional descriptor which represents for each orientation (row, pitch, and heave) the proportion of time the angle is higher than $0.005\,\pi$ rad and the proportion of time the angle is lower than $-0.005\,\pi$ rad. The **linear-velocity** is a 3-dimensional descriptor which represents the instantaneous linear velocities, v_x, v_y, and v_z, of the centre-of-mass of the robot for each dimension in the 3-D coordinate system. Some values of the base-features included may not appear to lead to high-performing solutions. For example, moving sideways, as indicated by a high score on the second dimension on the linear-velocity descriptor, would never lead to the highest performance in the normal environment. However, a high- or low-performing behaviour in the normal environment does not necessarily imply a high- or low-performing behaviour when the robot is damaged (for example, a behaviour that goes sideways might go straight when one of the legs is removed); also, the v_x feature of the linear-velocity is normalised to include only positive values[2].

To control for the effect of the weighted sum over base-features, an additional control condition, called **random-weights**, evolves a behaviour-performance map with a 3-dimensional combination of the same 15 base-features, as in the meta condition, but in this case, the weights are fixed during evolution. Each of the 45 weights are randomly initialised from a uniform distribution $U(0, 1)$, after which a rowwise normalisation is performed such that $\sum_j^{15} W_{ij} = 1$ for all $i \in \{1, 2, 3\}$.

3.3 Experimental parameters

Each of the experimental conditions is repeated for 5 independent replicates[3]. For the meta condition, each replicate is also defined by a different damage set on which the meta-fitness is computed. Two unique damage types are chosen randomly from the following: leg-removal, leg-shortening, blocked-joint, or passive-joint. This results in 12 unique damages, defined by a combination of damage type and the affected leg of the RHex robot.

To evolve the individual solutions, all conditions apply the same operators and use the same population parameters (see Table 1), except for parameters only defined for the meta condition. To keep the behaviour-performance maps comparable, the number of allowed solutions is set to 4096 for all conditions; for 3-dimensional behavioural descriptors, this amounts to 16 bins per dimension

[2]This implies that solutions with negative velocity cannot be elite solutions because they have to compete with solutions with a low, positive velocity.

[3]Each independent run is executed on a 16-CPU Intel Xeon 2.20 GHz and takes approximately 120-140 hours.

Table 1: Parameter settings for evolution.

Parameter	Setting
Maximal map coverage	4096 solution
Genotype (**g**)	discretised in $[0, 1]^{24}$
Mutation rate	0.05
Mutation type	random increment/decrement to a gene
Function evaluations	2,800,000
Batch size per generation	400 bottom-level individuals
Initial population (p)	2000 bottom-level individuals
Meta-population size (λ)	5
Meta-genotype (**w**)	$[0, 1]^{45}$

whereas for 6-dimensional behavioural descriptors, this amounts to 4 bins per dimension[4]. Bottom-level genotypes are evolved using a discretised genotype, with increments of 0.025, within $[0, 1]^{24}$; with a rate of 0.05, a relatively low-impact mutation is applied that randomly increments or decrements a gene, thereby allowing an exhaustive local search around the solutions in the behaviour-performance maps. Due to time constraints and the observation of weak convergence, all conditions are given a computational budget of 2.8 million function evaluations – this amounts to 7,000 generations for control conditions and based on our empirical results this corresponds to 260 meta-generations in the meta conditon. For the meta condition, a population size of $\lambda = 5$ is chosen to ensure a fairly rapid convergence whilst maintaining a global search. A number of 5 bottom-level generations per meta-generation is chosen to make a rapid assessment of how well a behaviour space is able to generate new solutions to obtain behaviour-performance maps with high meta-fitness. To ensure the evolution of diverse, high-quality solutions in the meta condition, the batch size, the number of iterations per bottom-level generation, is selected at 400 such that the number of bottom-level evaluations is comparable to the number of meta-fitness evaluations. The meta-level CMA-ES algorithm uses the default settings for its parameters. For control conditions, the iterations are all treated independent of the batch, and therefore we use the same batch size of 400. The initial population is set to 2000 to ensure sufficient diversity in the initial behaviour-performance map of the control conditions and the database of the meta condition.

4 RESULTS

We first analyse how the behaviour-performance maps evolve over time. Then, to evaluate the quality of the final behaviour-performance maps, each condition is subjected to a test phase, where its solutions are tested on damages different from training.

4.1 Evolution

The evolutionary process can give us information on the development of map quality and whether or not our meta-learning system is optimising its meta-fitness. This section first compares all the conditions to evaluate the quality of their resulting maps over time and then analyses the development of the meta condition in terms of the meta-fitness.

Map quality comparison. During evolution, all conditions improve their map quality statistics over time, progressing rapidly initially and then largely converging in the final generations (see Figure 2).

The global performance, the maximal performance across an entire behaviour-performance map, is similar for most conditions, with a distance walked of around 9.5 m at the end of evolution. The meta condition converges to this point more quickly, with a score of 9.5 m after 2,000 generations, and at the end of evolution, has a marginally higher score of 9.7 m. In comparison, the duty-factor has a lower maximal performance in its archives, with a maximal distance walked of 8.8 m.

The average performance across the map is another metric included in our analysis. The duty-factor and linear-velocity conditions have a lower average performance of 4.4 m and 4.2 m, respectively, while the body-orientation has the highest average performance of nearly 5.4 m. The random-weights and meta conditions score in between, with a performance of 4.8-4.9 m.

Finally, we observe the number of unique solutions in the behaviour-performance maps, the coverage. The meta and random-weights conditions have a notably lower coverage of the behaviour space, with 100-200 solutions, when compared to the other conditions, which have 1,000-3,000 solutions. We observe that, in the meta and random-weights conditions, the 3 behavioural features all score in $[0.2, 0.6]$. The explanation for these results is that extreme feature values, such as 0 or 1, for a particular feature is, with few exceptions, only achievable when all base-features have an extreme score (all 0 or all 1).

Meta-fitness evolution. The score on the meta-fitness function progresses in a volatile manner, with consecutive improvements being unpredictable over time, implying the meta-fitness is noisy, however, there is a stable increasing trend (see Fig. 3). These findings indicate that the meta-fitness is highly noisy but nevertheless is being optimised by the meta-level CMA-ES. The relatively large standard-deviation across replicates indicates the varying difficulty levels of the different damage sets.

4.2 Adaptation test of final maps

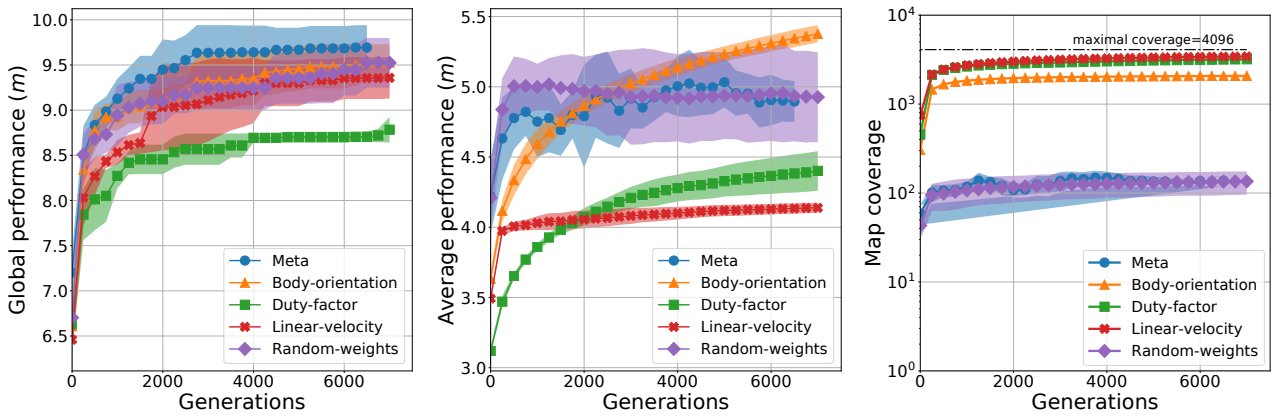
We analyse now the final behaviour-performance maps with regard to their adaptation performance. Performance data, comparing average performance across the map on all damages in the damage set, are collected for all individuals in the behaviour-performance maps, for two distinct damage sets:

- train-set: the damages that the meta condition used for its meta-fitness evaluations;
- test-set: the damages that the meta condition *did not* use for its meta-fitness evaluations.

Each replicate defines a different train-set of 12 unique damages (see Section 3.3), and therefore also a different test-set, consisting of the remaining 12 unique damages.

We perform two analyses: first, to evaluate the generalisation capability of controllers in the evolved maps, we assess the average performance of each controller on all damages in the test-set and

[4]Note that an alternative to defining for each dimension the number of bins would be to use Centroidal Voronoi Tesselations [28], which similarly allows a pre-defined number of solutions albeit with different geometry.



(a) Global performance (b) Average performance (c) Coverage

Figure 2: Evolution of map quality metrics (Mean ± SD, aggregated across replicates of the same condition), when bottom-level individual solutions are evaluated on a flat surface without any damages. The following metrics are shown as a function of the number of MAP-Elites generations[5]: (a) global performance computes the highest bottom-level fitness in the map; (b) average performance computes the mean bottom-level fitness in the map; (c) coverage computes the number of solutions in the archive. For the meta condition, Mean and SD are aggregated over replicates and over $\lambda = 5$ meta-individuals (i.e., its maps).

Table 2: Performance average of the behaviour-performance map on different damages, with Mean ± SD aggregated over replicates. For the meta condition, the behaviour-performance map with the best performance average on the train-set is selected for both the train- and test-set.

	Meta	Body orientation	Duty factor	Linear velocity	Random weight
Train-set	4.57 ± 0.09	4.88 ± 0.06	4.02 ± 0.13	3.91 ± 0.04	4.48 ± 0.24
Test-set	3.05 ± 0.04	3.18 ± 0.03	2.80 ± 0.10	2.80 ± 0.01	3.00 ± 0.14

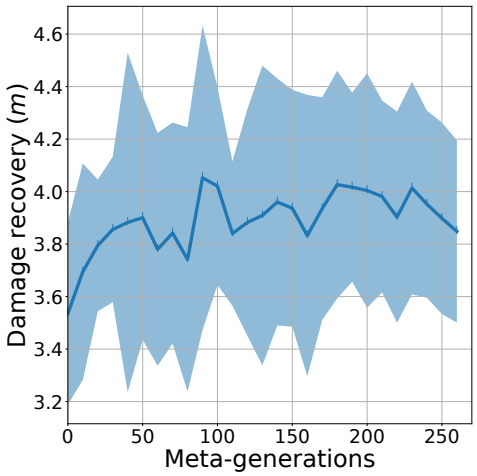


Figure 3: Evolution of the population average of the meta-fitness, which measures the distance travelled when damages are incurred on 10% randomly selected solutions, with *Mean* ± *SD* over 5 replicates.

average this across the map; second, to evaluate fault recovery, we assess the best performance obtained on each damage in the test-set individually by a random search across the map.

Generalisation. We hypothesised that the train- and test-set are best solved by controllers in the meta condition. To assess this hypothesis, we compute summary statistics of the performance in different scenarios (see Table 2).

The results indicate that the meta condition is among the top performers but that the body-orientation typically has the highest performance. The random-weights condition scores lower than the meta condition, while scoring higher than the duty-factor and linear-velocity. Overall, these results indicate that the use of a meta-fitness can help to obtain a performance similar to that of the highest-performing behavioural descriptor.

To compare the meta condition to the other conditions on the test-damages in pairwise manner, we make use of the Wilcoxon rank-sum test, a non-parametric analysis of significance, to ensure statistical power regardless of parametric assumptions, and Cliff's delta [1] as a metric for effect size, to assess the size of the effect

[5]The number of MAP-Elites generations does not take into account additional function evaluations due to the meta-fitness. Therefore, the meta condition's runs are halted with fewer total MAP-Elites generations to ensure the number of fitness evaluations is the same for all conditions.

regardless of sample size. This analysis demonstrates the following: i) the meta condition outperforms the duty-factor and linear velocity conditions, with large effect size ($p < 0.01$, $\Delta = 1.0$ in both cases); ii) the meta condition is outperformed by the body-orientation condition ($p < 0.01$, $\Delta = -1.0$); and iii) the effect of the meta condition with respect to the random-weights is not significant ($p = 0.602$, $\Delta = 0.20$).

To explain the third finding, an additional analysis demonstrates that the random-weights condition does not improve significantly on the lower-performing conditions (duty-factor and linear-velocity; $p = 0.076$, $\Delta = 0.68$ in both cases). Even the highest-performing condition, the body-orientation, has only a marginally significant p-value ($p = 0.047$, $\Delta = -0.76$). These data suggest that the high variance of the random-weights condition leads to high p-values, making the detection of significance difficult with the limited sample size of 5 replicates.

To analyse how a small difference in the average performance across different damages may translate into real-world consequences, we visualise the behaviours of the evolved controllers for different test damages[6]. Taking for a randomly chosen test set controllers with a performance similar to the average scores, 3.03 for the meta condition and 2.80 for the duty-factor condition, we observe that for some damages no differences are observed whereas for other damages large differences can be observed: when leg 6 is removed, controllers obtained from both conditions have a similar performance of 3.5 m; however, when leg 5 and leg 6 are blocked, the meta condition outperforms the duty factor by a large margin (3.9 m vs 1.7 m, and 3.8 m vs 3.1 m, respectively).

Recovering from individual damages. We now evaluate the best solution in the map on individual damages in the test-set. For simplicity of interpretation, we perform a random search across the behaviour-performance maps, where a random controller in the map is selected without replacement and the best performance so far is recorded – this implies that the results presented here may be improved, for example, by using Bayesian optimisation as in previous work [3]. Because robots usually have limited time to adapt and not all maps have 100 or more controllers, we allow 80 function evaluations for this search. To assess the benefit of the MAP-Elites algorithms compared to a traditional evolutionary algorithm, we compare the solutions obtained from the maps to the solutions obtained by evolving controllers from scratch with CMA-ES – with parameters set to default values and the population size set to $\lambda = 5$. All algorithms, including CMA-ES and the random search across the maps for the different MAP-Elites conditions, are run independently for a total of 60 independent runs, where each run represents a combination of the replicate and one of the unique damages in the corresponding test-set. Results, shown in Figure 4, demonstrate that the meta and body-orientation conditions perform equally high, with an initial performance of around 5 m and a final performance of around 6 m. While the other MAP-Elites conditions (duty-factor, linear-velocity, and random) are close in performance, with a performance drop of around 0.5 m, the traditional EA is not successful with a drop of around 1.5 m.

[6]See http://tiny.cc/metacmaes for the video materials.

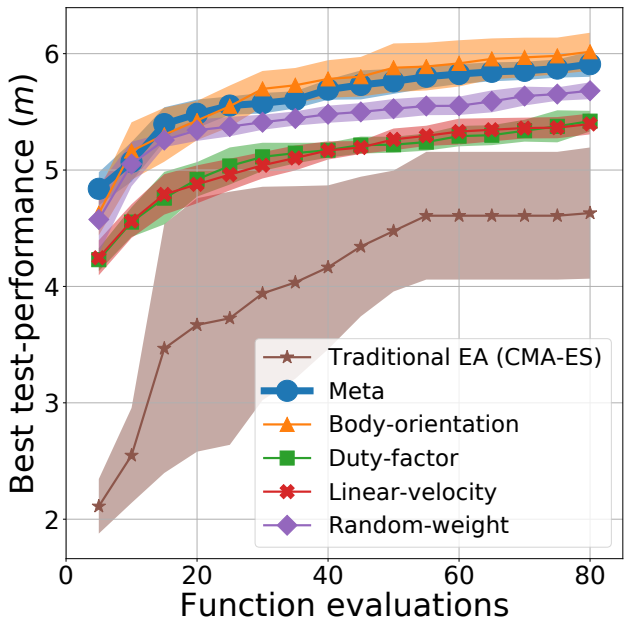


Figure 4: Fault recovery by searching for the best controller for each damage in the test-set individually, based on 5 independent replicates with 12 unique damages each. The x-axis represents the number of function evaluations; for MAP-Elites algorithms, this is a random search across the behaviour-performance map, while for the traditional EA this is a search from scratch. The y-axis represents the best performance (Mean ± SD) so far in the search; the line represents the Mean across the different damages and replicates whilst the shaded area is based on the standard-deviation across replicates. For a given replicate of the meta condition, the behaviour-performance map is selected by performing an analogous search procedure on the train-set and then taking the map with maximal final performance average.

5 DISCUSSION

This paper proposes a novel meta-algorithm for learning a population of behaviour-performance maps. The algorithm makes use of a large database of solutions, allowing an efficient reuse of previously evaluated solutions to construct and evaluate new behaviour-performance maps on-the-fly. Optimising a meta-fitness function, our system learns how to combine features to obtain a performance similar to the highest-performing behavioural descriptor. Results not only indicate a benefit related directly to the meta-fitness, namely that adaptation to diverse scenarios is improved, but also an improved global performance on the bottom-level fitness.

Although there are certain commonalities, such as the reduction of dimensionality [2, 4] and adaptivity over behavioural features [5, 18], our system departs in other ways from previous approaches to automated behavioural diversity. The use of a top-level evolution system allows the interpretation as a meta-evolution system, in which evolutionary mechanisms are used to guide evolution.

A comparatively generic meta-fitness criterion allows a less prescriptive system, which does not use pre-existing rules to adapt the behaviour space (compare Meyerson et al. (2016) [18], for example, where weights for behavioural features are determined by a heuristic equation). With few modifications, a wider set of characteristics of the bottom-level quality-diversity algorithm, other than its behavioural features, could potentially be evolved as well. However, the high cost of quality-diversity algorithms limits the number of parameters that can possibly be optimised in this manner. Therefore, given the limited budget for computation, the use of CMA-ES with a limited number of meta-individuals is justified as a meta-evolution strategy for evolving quality-diversity algorithms. A further benefit of CMA-ES is that, due to exploring the search space with multiple solutions at a time, it has the potential to escape local minima more easily compared to gradient-descent based methods; moreover, noisy cost functions such as our meta-fitness functions can be optimised more readily due to its inherently statistical nature.

While our approach selects the behaviour-performance map during evolution, another approach to automated behavioural diversity is to evolve multiple independent maps in simulation and then automatically select the map that is most suitable for online adaptation in the context of the application [13, 22]. Although these approaches are not mutually exclusive, and may be complementary, a comparison of both would be an interesting avenue for further research.

Our system is amenable for further improvements, and our study has its limitations. The database for restoring individuals may be improved by preventing the loss of high-performing individuals and the addition of low-performing or behaviourally similar individuals. If coverage is a desired end-goal, a different normalisation can be used for the behavioural features. Further, there is a trade-off between frequent meta-fitness evaluations and the MAP-Elites iterations and our current system may not have chosen the optimal setting; since the right balance would be highly problem-dependent, we envision a role for meta-optimisation in this case. Finally, the benefit in performance when compared to a random choice of feature weights is marginal (though not negligible). A possible interpretation is that, due to combining the high-performing body-orientation with lower-performing linear-velocity and duty-factor, the performance is an interpolation between that of the maps that would be constructed from these constituent features; if this is so, the meta-learning system would also be bounded by the maximal performance (i.e., that of the body-orientation). Another possible interpretation is that there is an inherent benefit to using multiple behavioural descriptors, as some other findings suggest [5, 24]. Due to the large variance of the random weights, more samples are required to assess significance. Despite these reservations, our findings suggest that our system is able to learn a higher-performing behaviour space than hand-coded and randomly chosen features.

6 CONCLUSION

Archives evolved by quality-diversity algorithms such as MAP-Elites are often exploited for adaptation to unknown future events. Because the behaviour space that maximises the performance over these unknown events is unknown and because the computational resources are limited – implying that the end-user cannot empirically assess which is the best behavioural descriptor – there is a need for automated behavioural descriptors that learn a suitable behaviour space. This paper introduces a new meta-evolution system which evolves a population of MAP-Elites algorithms, optimising the behaviour space by applying CMA-ES on an adaptation-based meta-fitness criterion. The proposed system represents behaviour-performance maps by weighting a larger set of behavioural base-features and – using a database which stores genotypes, behavioural base-features, and fitness value - the system efficiently reuses solutions found by iterations of the bottom-level MAP-Elites algorithm. We experiment with the evolution of controllers for improved damage recovery with the RHex robot. Our system gives high-quality solutions (i.e., controllers of the RHex robot) on par with the highest-quality hand-coded behaviour space included in the study, with or without damages injected to the robot.

ACKNOWLEDGEMENTS

This work has been supported by the Engineering and Physical Sciences Research Council (EPSRC) under the New Investigator Award grant (EP/R030073/1), the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (GA no. 637972, project "ResiBots") and the Lifelong Learning Machines program (L2M) from DARPA/MTO under Contract No. FA8750-18-C-0103.

REFERENCES

[1] Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin* 114, 3 (1993), 494–509. https://doi.org/10.1037/0033-2909.114.3.494

[2] Antoine Cully. 2019. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the 2019 Genetic and Evolutionary Computation Conference (GECCO 2019)*. 81–89. https://doi.org/10.1145/3321707.3321804

[3] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507. https://doi.org/10.1038/nature14422 arXiv:1407.3501

[4] Antoine Cully and Yiannis Demiris. 2018. Hierarchical Behavioral Repertoires with Unsupervised Descriptors. In *Proceedings of the 2018 Genetic and Evolutionary Computation Conference (GECCO 2018)*. 69–76. https://doi.org/10.1145/3205455.3205571 arXiv:1804.07127

[5] Stephane Doncieux and Jean-Baptiste Mouret. 2013. Behavioral diversity with multiple behavioral distances. In *2013 IEEE Congress on Evolutionary Computation, CEC 2013*. 1427–1434. https://doi.org/10.1109/CEC.2013.6557731

[6] Sondre Engebråten, Oleg Yakimenko, Jonas Moen, and Kyrre Glette. 2018. Towards a Multi-Function Swarm That Adapts to User Preferences. In *International Conference on Robotics and Automation (ICRA 2018)*.

[7] Matthew C. Fontaine, Fernando De Mesentier Silva, Scott Lee, Julian Togelius, L. B. Soros, and Amy K. Hoover. 2019. Mapping hearthstone deck spaces through map-elites with sliding boundaries. In *Proceedings of the 2019 Genetic and Evolutionary Computation Conference (GECCO 2019)*. 161–169. https://doi.org/10.1145/3321707.3321794 arXiv:1904.10656

[8] Nikolaus Hansen. 2007. The CMA Evolution Strategy: A Comparing Review. *Towards a New Evolutionary Computation* 102, 2006 (2007), 75–102. https://doi.org/10.1007/3-540-32494-1_4

[9] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A Tutorial. *arXiv preprint arXiv:1604.00772* (2016), 1–39. arXiv:1604.00772 http://arxiv.org/abs/1604.00772

[10] Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. *Proceedings of the IEEE Conference on Evolutionary Computation* (1996), 312–317. https://doi.org/10.1109/icec.1996.542381

[11] Emma Hart, Andreas S.W. Steyven, and Ben Paechter. 2018. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. In *Proceedings of the 2018 Genetic and Evolutionary Computation Conference (GECCO 2018)*. 101–108. https://doi.org/10.1145/3205455.3205481 arXiv:1804.07655

[12] Grahame A. Jastrebski and Dirk V. Arnold. 2006. Improving evolution strategies through active covariance matrix adaptation. *2006 IEEE Congress on Evolutionary Computation, CEC 2006* (2006), 2814–2821. https://doi.org/10.1109/cec.2006.1688662

[13] Rituraj Kaushik, Pierre Desreumaux, and Jean-Baptiste Mouret. 2020. Adaptive prior selection for repertoire-based online adaptation in robotics. *Frontiers in Robotics and AI* 6 (2020), 151.

[14] Ahmed Khalifa, Andy Nealen, Scott Lee, and Julian Togelius. 2018. Talakat: Bullet hell generation through constrained map-elites. In *Proceedings of the 2018 Genetic and Evolutionary Computation Conference (GECCO 2018)*. 1047–1054. https://doi.org/10.1145/3205455.3205470 arXiv:1806.04718

[15] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. 2018. DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software* 3, 22 (2018), 500. https://doi.org/10.21105/joss.00500

[16] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation* 19, 2 (2011), 189–222. https://doi.org/10.1162/EVCO_a_00025

[17] Joel Lehman and Kenneth O. Stanley. 2011. Evolving a Diversity of Creatures through Novelty Search and Local Competition. In *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM, New York, 1408. https://doi.org/10.1001/jama.1990.03450110053016

[18] Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. 2016. Learning Behavior Characterizations for Novelty Search. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO 2016)*. 149–156. https://doi.org/10.1145/2908812.2908929

[19] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909v1* (2015), 1–15. arXiv:arXiv:1504.04909v1

[20] A. Nguyen, J. Yosinski, and J. Clune. 2015. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*. 545–572. https://doi.org/10.1162/EVCO_a_00189

[21] Jørgen Nordmoen, Kai Olav Ellefsen, and Kyrre Glette. 2018. Combining MAP-Elites and Incremental Evolution to Generate Gaits for a Mammalian Quadruped Robot. In *21st International Conference, EvoApplications 2018*, Kevin Sim and Paul Kaufmann (Eds.). Springer International Publishing, Parma, Italy, 159–170. https://doi.org/10.1007/978-3-319-77538-8

[22] Rémi Pautrat, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2018. Bayesian optimization with automatic prior selection for data-efficient direct policy search. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7571–7578.

[23] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI* 3, July (2016), 1–17. https://doi.org/10.3389/frobt.2016.00040

[24] Justin K Pugh, L B Soros, and Kenneth O Stanley. 2016. Searching for Quality Diversity When Diversity is Unaligned with Quality. In *Proceedings of the 14th Conference on Parallel Problem Solving from Nature, PPSN XIV*. Edinburgh, UK, 880–889. https://doi.org/10.1007/978-3-642-15844-5

[25] Justin K. Pugh, L. B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. 2015. Confronting the challenge of quality diversity. In *Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*. ACM, New York, NY, USA, 967–974. https://doi.org/10.1145/2739480.2754664

[26] U Saranli, M Buehler, and D E Koditschek. 2001. RHex: A Simple and Highly Mobile Robot. *International Journal of Robotics Research* 20, 7 (2001), 616–631.

[27] J. Seipel and P. Holmes. 2007. A simple model for clock-actuated legged locomotion. *Regular and Chaotic Dynamics* 12, 5 (2007), 502–520. https://doi.org/10.1134/S1560354707050048

[28] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 623–630.