DISCOVERING UNSUPERVISED BEHAVIOURS FROM FULL STATE TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving open-ended learning capabilities is a promising approach to enable robots to face the unbounded complexity of the real-world. Among existing methods, the ability of Quality-Diversity algorithms to generate large collections of diverse and high-performing skills is instrumental in this context. However, most of those algorithms rely on a hand-coded behavioural descriptor to characterise the diversity, hence requiring prior knowledge about the considered tasks. In this work, we propose an additional analysis of Autonomous Robots Realising their Abilities; a Quality-Diversity algorithm that autonomously finds behavioural characterisations. We evaluate our approach on a simulated robotic environment, where the robot has to autonomously discover its abilities from its full state trajectories. All algorithms were applied to three tasks: navigation, moving forward with a high velocity, and performing half-rolls. The experimental results show that the algorithm under study discovers autonomously collections of solutions that are diverse with respect to all tasks. More specifically, our approach autonomously finds policies that make the robot move to diverse positions, but also utilise its legs in diverse ways, and even perform half-rolls.

1 INTRODUCTION

One of the motivations of using learning algorithms in robotics is to enable robots to discover on their own their abilities. Ideally, a robot should be able to discover on its own how to manipulate new objects (Johns et al., 2016) or how to adapt its behaviours when facing an unseen situation like a mechanical damage (Cully et al., 2015). However, despite many impressive breakthroughs in learning algorithms for robotics applications during the last decade (Akkaya et al., 2019; Hwangbo et al., 2019), these methods still require a significant amount of engineering to become effective, for instance, to define reward functions, state definitions, or characterise the expected behaviours.

Finding all possible behaviours for a robot is an open-ended process, as the set of all achievable robot behaviours is unboundedly complex. There are plenty of different ways to execute each type of behaviour. For instance, for moving forward, rotating or jumping, the robot can move at different speeds, and use different locomotion gaits. Supplementary behaviours can emerge from interactions with objects and with other agents.

An attempt to discover autonomously the behaviours of a robot has been proposed with the AURORA algorithm (Cully, 2019; Grillotti & Cully, 2021). This algorithm leverages the creativity of Quality-Diversity (QD) optimisation algorithms to generate a collection of diverse and high-performing behaviours, called a behavioural repertoire. QD algorithms usually require the definition of a manually-defined Behavioural Descriptor (BD) to characterise the different types of behaviours contained in the repertoire. Instead, AURORA uses dimensionality-reduction techniques to perform the behavioural characterisation in an unsupervised manner. The resulting algorithms enable robots to autonomously discover a large diversity of behaviours without requiring a user to define a performance function or a behavioural descriptor. Nevertheless, in prior work, AURORA has only been used to learn behavioural characterisations from *hand-defined* sensory data (e.g. an image of the robot at the end of the episode).

In this work, we propose an extended analysis of AURORA, where behavioural characterisations are learned based on the complete trajectory of the robot raw states. In that sense, no prior information is injected in the sensory data. We evaluated AURORA on a simulated hexapod robot with

a neural network controller on three tasks: navigation, moving forward with a high velocity, and performing half-rolls. Furthermore, we compare AURORA to several baselines, all of which use hand-coded BDs. In most cases, the collections obtained via AURORA present more diversity than those obtained from hand-coded QD algorithms. In particular, AURORA automatically discovers behaviours that make the robot navigate to diverse positions, while using its legs in diverse ways, and perform half-rolls.

2 BACKGROUND

2.1 QUALITY-DIVERSITY ALGORITHMS

Quality-Diversity (QD) algorithms are a subclass of evolutionary algorithms that aim at finding a container of both diverse and high-performing policies. In addition to standard evolutionary algorithms, QD algorithms consider a Behavioural Descriptor (BD), which is a low-dimensional vector that characterises the behaviour of a policy. QD algorithms use the BDs to quantify the novelty of a policy with respect to the solutions already in the container. The container of policies produced by the QD algorithm in an iterative manner: (1) policies are selected from the container and undergo random variations (e.g. mutations, cross-overs); (2) their performance score and BD are evaluated; and (3) we try to add them back to the container. If they are novel enough compared to solutions that are already in the container, they are added to the container. If they are better than similar policies in the container, they replace these policies.

2.2 AUTONOMOUS ROBOTS REALISING THEIR ABILITIES (AURORA)

Quality diversity algorithms are a promising tool to generate a large diversity of behaviours in robotics. However, the definition of the BD space might not always be straightforward when the robot or its capabilities are unknown. AURORA is a QD algorithm designed to discover the abilities of robots, by maximising the diversity of behaviours contained in the repertoire in an unsupervised manner. AURORA automatically defines the BD by encoding the high-dimensional sensory data generated by the robot during a controller execution into low-dimensional representations. The encoding can be achieved using any dimensionality reduction technique, such as, Auto-Encoder (AE) (Grillotti & Cully, 2021) or Principal Component Analysis (Cully, 2019).

To generate a behavioural repertoire, AURORA alternates between QD phases and Encoder phases. During the QD phase, policies undergo the same process as in standard QD algorithms: selection from the container, evaluation, and attempt to add to the container. However, the evaluation is performed in a slightly different way: instead of a low-dimensional BD the QD task returns high-dimensional sensory data, that is then encoded using the dimensionality reduction technique. The latent encoding of the sensory data is used as the BD for the rest of the QD phrase.

During the Encoder phase, the dimensionality reduction structure (e.g. the AE) is trained using the high-dimensional data from all the policies present in the container. Once the encoder is trained, the unsupervised BDs are recomputed with the new encoder for all policies present in the container.

Alternating these two phases enables AURORA to progressively build its behavioural repertoires by discovering new solutions, while refining its encoding of the high-dimensional sensory data generated by the robot every time new solutions are added to the archive. In this work, the sensory data collected by the robot corresponds to the full-state trajectory of the robot: $\boldsymbol{s}_{1:T}$.

3 LEARNING DIVERSE BEHAVIOURS FROM FULL STATE TRAJECTORIES

In this section, we provide a theoretical analysis of the AURORA algorithm, to support its ability to generate a large collection of diverse behaviours in an unsupervised manner. In essence, we aim to find a container of policies leading to diverse behaviours. We consider that a *behaviour* is defined as the succession of full states the agents goes through: $\boldsymbol{s}_{1:T}$. Then the overall problem of finding diverse behaviours $\left(\boldsymbol{s}^{(i)}_{1:T}\right)_i$ can be expressed as an entropy maximisation problem. Considering the behaviour $\boldsymbol{s}_{1:T}$ as a random variable, we aim to maximise the entropy of $\boldsymbol{s}_{1:T}$, written $H\left(\boldsymbol{s}_{1:T}\right)$. In

the following sections we will explain (1) why it can be ineffective to maximise directly the diversity of full states $\boldsymbol{s}_{1:T}$; and (2) how AURORA can be used to maximise indirectly this diversity.

3.1 Problem with High-dimensional Behavioural Descriptors

Maximising the entropy of full states $H\left(\boldsymbol{s}_{1:T}\right)$ is theoretically possible by using a QD algorithm with $\boldsymbol{s}_{1:T}$ as a Behavioural Descriptor (BD). However, it is computationally inefficient to use QD algorithms with high-dimensional BDs.

Indeed, as the dimensionality of that problem is high, a consequent number of samples of $\boldsymbol{s}_{1:T}$ is required to optimise $H\left(\boldsymbol{s}_{1:T}\right)$. So a significant number of policies is needed in the container. However, QD algorithms become less efficient when the number of policies increases. When the number of policies in the container increases, the selection pressure of the container decreases, which means that the policies from the container are mostly less likely to be selected (Cully & Demiris, 2018).

Also, in QD algorithms, unstructured containers always rely on the k-nearest neighbours algorithm to compute the novelty of a policy. And the complexity of that algorithm can be $O\left(\dim \mathcal{B} \log|\mathcal{C}|\right)$ in average (Bentley, 1975), and $O\left(\dim \mathcal{B}|\mathcal{C}|\right)$ in the worst case ($\mathcal{B}$ refers to the BD space, and $|\mathcal{C}|$ is the number of policies in the container $\mathcal{C}$). Hence, the computational complexity of one QD generation increases linearly with dimensionality of the BD $\dim \mathcal{B}$. That is why the overall procedure of adding policies to the container at each QD generation may become too computationally expensive if the dimension of $\mathcal{B}$ increases significantly.

3.2 Learning Low-dimensional Descriptors from High-dimensional State Trajectories

The encoder update phase of AURORA optimises the parameters $\boldsymbol{\eta}_{\mathcal{E}}$ of the encoder $\mathcal{E}$; this encoder is then used to define a low-dimensional BD $\boldsymbol{b}_{\mathcal{E}}$ from state trajectories: $\boldsymbol{b}_{\mathcal{E}} = \mathcal{E}(\boldsymbol{s}_{1:T})$. Figure 1 summarises the dependencies between the different variables considered in standard QD algorithms and AURORA. The mutual information $I\left(\cdot,\cdot\right)$ between $\boldsymbol{s}_{1:T}$ and $\boldsymbol{b}_{\mathcal{E}}$ can be expressed in two symmetrical ways:

$$I\left(\boldsymbol{s}_{1:T}, \boldsymbol{b}_{\mathcal{E}}\right) = H\left(\boldsymbol{s}_{1:T}\right) - H\left(\boldsymbol{s}_{1:T} \mid \boldsymbol{b}_{\mathcal{E}}\right) \tag{1}$$

$$I\left(\boldsymbol{s}_{1:T}, \boldsymbol{b}_{\mathcal{E}}\right) = H\left(\boldsymbol{b}_{\mathcal{E}}\right) - H\left(\boldsymbol{b}_{\mathcal{E}} \mid \boldsymbol{s}_{1:T}\right) \tag{2}$$

As $\boldsymbol{b}_{\mathcal{E}}$ is a function of $\boldsymbol{s}_{1:T}$, we have: $H\left(\boldsymbol{b}_{\mathcal{E}} \mid \boldsymbol{s}_{1:T}\right) = 0$. Then, from equations 1 and 2 we get:

$$H\left(\boldsymbol{s}_{1:T}\right) = H\left(\boldsymbol{b}_{\mathcal{E}}\right) + H\left(\boldsymbol{s}_{1:T} \mid \boldsymbol{b}_{\mathcal{E}}\right) \tag{3}$$

$$= H\left(\boldsymbol{b}_{\mathcal{E}}\right) + H\left(\boldsymbol{s}_{1:T} \mid \mathcal{E}\left(\boldsymbol{s}_{1:T}\right)\right) \tag{4}$$

According to equation 4, $H\left(\boldsymbol{s}_{1:T}\right)$ can be decomposed into two components:

- The entropy on the BD space $H\left(\boldsymbol{b}_{\mathcal{E}}\right)$, which can be maximised via a QD algorithm.
- The conditional entropy of the state trajectories with respect to their encoding $H\left(\boldsymbol{s}_{1:T} \mid \mathcal{E}\left(\boldsymbol{s}_{1:T}\right)\right)$. This term quantifies the information that is lost between $\boldsymbol{s}_{1:T}$ and its encoding $\boldsymbol{b}_{\mathcal{E}} = \mathcal{E}\left(\boldsymbol{s}_{1:T}\right)$.

Inequality 4 gives a lower bound on the entropy of state trajectories:

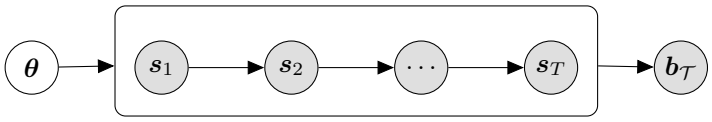
$$H\left(\boldsymbol{s}_{1:T}\right) \geq H\left(\boldsymbol{b}_{\mathcal{E}}\right) \tag{5}$$

And Fano's inequality (Cover & Thomas, 2012, p. 38) gives a bounding on the differences between those entropies:

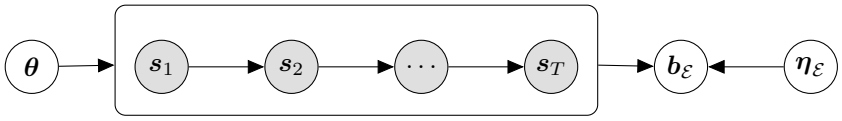
$$H\left(\boldsymbol{s}_{1:T}\right) - H\left(\boldsymbol{b}_{\mathcal{E}}\right) = H\left(\boldsymbol{s}_{1:T} \mid \mathcal{E}\left(\boldsymbol{s}_{1:T}\right)\right) \tag{6}$$

$$H\left(\boldsymbol{s}_{1:T}\right) - H\left(\boldsymbol{b}_{\mathcal{E}}\right) \leq H\left(e\right) + P(e)\log|\mathcal{X}_{\mathcal{O}^T}| \tag{7}$$

where $P(e) = P(\boldsymbol{s}_{1:T} \neq \widehat{\boldsymbol{s}_{1:T}})$, $\widehat{\boldsymbol{s}_{1:T}}$ represents a reconstruction of the state trajectories $\boldsymbol{s}_{1:T}$ from the encoding $\mathcal{E}(\boldsymbol{s}_{1:T}) = \boldsymbol{b}_{\mathcal{E}}$, and $\mathcal{X}_{\mathcal{O}^T}$ represents the discretised support of the state trajectories $\boldsymbol{s}_{1:T}$.



(a) Graphical Model of standard Quality-Diversity optimisation tasks



(b) Graphical Model of an optimisation task using AURORA

Figure 1: Graphical Models of the Quality-Diversity optimisation algorithms in the standard case (1a) and using AURORA (1b). $\boldsymbol{\theta}$ corresponds to the parameters of the agent's policy. At each time-step t, $\boldsymbol{s}_t$ represent the state of the agent. $\boldsymbol{b}_\mathcal{T}$ represents the task behavioural descriptor based on the states of the robot; $\boldsymbol{b}_\mathcal{E}$ is a behavioural descriptor based on the successive states of the robot $\boldsymbol{s}_{1:T}$. Finally, $\boldsymbol{\eta}_\mathcal{E}$ corresponds to the parameters of the encoder $\mathcal{E}$ used to reduce the dimensionality of the state trajectories $\boldsymbol{s}_{1:T}$.

As explained in section 2.2, AURORA alternates between two phases: an *encoder update* phase, and a *QD* phase. During the encoder update phase, we minimise the difference $H\left(\boldsymbol{s}_{1:T}\right) - H\left(\boldsymbol{b}_\mathcal{E}\right) = H\left(\boldsymbol{s}_{1:T}\right) - H\left(\mathcal{E}(\boldsymbol{s}_{1:T})\right)$ by training the encoder $\mathcal{E}$. Indeed, training $\mathcal{E}$ will reduce the probability of reconstruction error $P(e)$, with the intention of lessening the difference $H\left(\boldsymbol{s}_{1:T}\right) - H\left(\boldsymbol{b}_\mathcal{E}\right)$ (see inequality 7). During the QD phase, now that the difference between both sides of inequality 5 is minimised, we increase the lower bound of $H\left(\boldsymbol{s}_{1:T}\right)$, namely $H\left(\boldsymbol{b}_\mathcal{E}\right)$. As $\boldsymbol{b}_\mathcal{E}$ is a low-dimensional BD, maximising $H\left(\boldsymbol{b}_\mathcal{E}\right)$ can be intuitively achieved by applying a standard Quality-Diversity algorithm, with $\boldsymbol{b}_\mathcal{E}$ as BD.

In the end, the two phases of AURORA successively (1) train the encoder to minimise the difference between the entropy of successive states $H\left(\boldsymbol{s}_{1:T}\right)$ and its lower bound $H\left(\mathcal{E}(\boldsymbol{s}_{1:T})\right)$, and (2) use QD iterations, to maximise the lower bound $H\left(\boldsymbol{b}_\mathcal{E}\right) = H\left(\mathcal{E}(\boldsymbol{s}_{1:T})\right)$. This intuitively explains why AURORA can be used to maximise the diversity of full state trajectories $\boldsymbol{s}_{1:T}$.

4 Related Works

The problem of autonomously finding diverse behaviours has been addressed with Mutual-Information (MI) maximisation approaches (Gregor et al., 2016; Eysenbach et al., 2018; Sharma et al., 2019). Such methods aim to maximise the mutual information between the states explored by the robot, and a latent variable representing the behaviour. Those MI-based approaches then return a single policy, which is conditioned on the behaviour latent variable. That differs from QD algorithms, which return a container of different policies. Also, MI-maximisation algorithms do not necessarily require the definition of a BD. For example, in the work of Eysenbach et al. (2018), diverse robotic behaviours are learned from full robot states. Similarly to AURORA, when the robot states are described with high-dimensional images, an encoder can be used to compute tractable low-dimensional representations (Liu & Abbeel, 2021b;a). While sharing the same purpose, QD and MI-maximisation algorithms are fundamentally different in their core design. Those differences make it challenging to compare the two approaches in a fair manner.

In QD algorithms, selecting the most appropriate definitions for the BD and the performance requires a certain level of expertise. For instance, MAP-Elites is often used with the same hexapod robot, but with two different sets of definitions: 1) the leg duty-factor for the BD, defined as the time proportion each leg is in contact with the ground, with the average speed for the performance (Cully et al., 2015; Cully & Demiris, 2018) or 2) the final location of the robot after walking during three seconds for the BD, with an orientation error for the performance (Cully & Mouret, 2013; Duarte et al., 2016; Chatzilygeroudis et al., 2018). The resulting behavioural repertoires will find different types of applications. The first set of definitions is designed for fast damage recovery as it provides a

diversity of ways to achieve high-speed gaits, while the second one is designed for navigation tasks as the repertoire enables the robot to move in various directions.

To avoid the expertise requirements in the behaviour descriptor definition, several methods have been proposed to enable the automatic definition of the behavioural descriptors. As described previously, AURORA aims to tackle this problem by using a dimensionality reduction algorithm (a Deep Auto-Encoder (Masci et al., 2011)) to project the features of the solutions into a latent space and use this latent space as a behavioural descriptor space. TAXONS (Paolo et al., 2020) is another QD algorithm adopting the same principle; it demonstrated that this approach can scale to camera images to produce large behavioural repertoires for different types of robots. The same concept was also used in the context of Novelty Search prior to AURORA and TAXONS to generate assets for video games with the DeLeNoX algorithm (Liapis et al., 2013). All these methods aim to maximise the diversity of the produced solutions, without specifically considering a downstream task.

A different direction to automatically define a behavioural descriptor is to use Meta-Learning. The idea is to find the BD definition that maximises the performance of the resulting collection of solutions when used in a downstream task. For instance, Bossen et al. (Bossens et al., 2020) considers the damage recovery capabilities provided by a behavioural repertoire as a Meta-Learning objective and search for the linear combination of a pre-defined set of behaviour descriptors that will maximise this objective. A related approach proposed by Meyerson et al. (Meyerson et al., 2016) uses a set of training tasks to learn what would be the most appropriate BD definition to solve a "test task".

5 Experimental Setup

5.1 Agent: Neural-network controlled hexapod

In all our experiments, our agent consists of a simulated hexapod robot controlled via a neural network controller. The hexapod robot presents 18 controllable joints (3 per leg). Each leg l has two Degrees of Freedom: its hip angle α_l and its knee angle β_l. The ankle angle is always set to the opposite of the knee angle.

The neural network controller is a Multi-Layer Perceptron with a single hidden layer of size 8. It takes as input the current joint angles and velocities $(\alpha_l, \beta_l, \dot{\alpha}_l, \dot{\beta}_l)_{1 \leq l \leq 6}$ (of dimension 24), and outputs the target values to reach for the joint angles $(\alpha_l, \beta_l)_{1 \leq l \leq 6}$ (of dimension 12). The controller has in total $(24 + 1) * 8 + (8 + 1) * 12 = 308$ independent parameters, operates at a frequency of 50Hz, while each episode lasts for 3 seconds. The environment and the controller are deterministic.

5.2 Dimensionality Reduction Algorithm of AURORA

For each evaluated policy, the state trajectory corresponds to the joint positions, and torso positions and orientations collected at a frequency of 10Hz. In the end, this represents 18 streams ($(\alpha_i, \beta_i)_{i=1...6}$, and the positional and rotational coordinates of the torso), with each of those streams containing 30 measurements. The dimensionality reduction algorithm used in AURORA is a reconstruction-based auto-encoder. The input data is using two 1D convolutional layers, with 128 filters and a kernel of size 3. Those convolutional layers are followed by one fully-connected linear layer of size 256. The decoder is made of a fully-connected linear layer of size 256, followed by three 1D deconvolutions with respectively 128, 128 and 18 filters.

The loss function used to train the Encoder corresponds to the mean squared error between the observation passed as input and the reconstruction obtained as output. The training is performed using the Adam gradient-descent optimiser (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. As the encoder needs to be updated less and less frequently as the number of iterations increase, the number of iterations between two encoder updates is linearly increased over time: if the first encoder update happens at iteration 10, then the following encoder updates occur at iterations 30, 60, 100...

5.3 QD Tasks

For the hexapod environment, we consider different types of QD Tasks. A QD task evaluates the diversity and the performance of policies present in the container returned by a QD algorithm. Each

QD task is characterised by a performance score function $\boldsymbol{\theta} \mapsto f(\boldsymbol{\theta})$, and a BD function $\boldsymbol{\theta} \mapsto \boldsymbol{b}_{\mathcal{T}}(\boldsymbol{\theta})$ (where $\boldsymbol{\theta}$ represents the policy parameters).

We consider three QD tasks with the hexapod agent: Navigation (Nav), Moving Forward (Forw), and Half-roll (Roll).

5.3.1 NAVIGATION (NAV)

The Navigation task is inspired from Chatzilygeroudis et al. (2018) and Kaushik et al. (2020). This task aims at finding a container of controllers reaching diverse final (x_T, y_T) positions in $T = 3$ seconds following circular trajectories. Hence the hand-coded BD in this case is $\boldsymbol{b}_{\mathcal{T}} = (x_T, y_T)$, while the performance f is the final orientation error as defined in the literature (Cully & Mouret, 2013; Chatzilygeroudis et al., 2018).

5.3.2 MOVING FORWARD (FORW)

The Moving Forward task is inspired from the work of Cully et al. (2015), which aims to obtain a container with a variety of ways to move forward at a high velocity. The hand-coded BD associated to that task $\boldsymbol{b}_{\mathcal{T}}$ is the Duty Factor, which evaluates the proportion of time the each leg is in contact with the ground. The performance score promotes solutions achieving the highest x position at the end of the episode: $f = x_T$.

5.3.3 HALF-ROLL (ROLL)

Half-roll QD task aims at finding a diversity of ways to perform half-rolls, i.e. behaviours such that the hexapod ends with its back on the floor (where the pitch angle of the torso is approximately equal to $-\frac{\pi}{2}$). In particular, the behaviours are characterised via the final yaw and roll angle of the torso. And the performance is measured as the distance between the final pitch angle α_T^{pitch} of the hexapod and $-\frac{\pi}{2}$.

$$\boldsymbol{b}_{\mathcal{T}} = \left(\alpha_T^{yaw}, \alpha_T^{roll}\right) \quad f = -\left|\alpha_T^{pitch} - \left(-\frac{\pi}{2}\right)\right| \tag{8}$$

5.4 METRIC: COVERAGE PER MINIMUM PERFORMANCE

The containers returned by any QD algorithm always result in a trade-off between the diversity and the total performance of the solutions from the container. To evaluate the coverage depending on the quality of the solutions, we study the evolution of the coverage given several minimum performance scores. With a performance f_{min} and a BD space discretised into a grid, we define the *"coverage given minimum performance f_{min}"* as: the number of cells filled with policies whose performance is higher than f_{min}.

The coverage is calculated by discretising the task Behavioural Descriptor (BD) space $\mathcal{B}_{\mathcal{T}}$ into a 50×50 grid. The coverage then corresponds the number of grid cells containing at least one policy from the unstructured container.

5.5 COMPARED ALGORITHMS AND VARIANTS

We compare AURORA (see Section 2.2), to two different variants having hand-defined BDs: Hand-Coded-x (HC-x) and Mean Streams (MeS).

Hand-Coded-x (HC-x): considers a low-dimensional BD that is defined by hand as the BD of the QD task x, where x can correspond to any QD task (Navigation, Moving Forward or Half-roll). The HC variant corresponds to a standard QD algorithm with an unstructured archive (Cully & Demiris, 2018) and the mechanism of container size control introduced in previous work (Grillotti & Cully, 2021).

Mean Streams (MeS): uses a BD that encompasses more information from the full state of the agent than the Hand-Coded variant detailed above. The BD used by this variant considers the 18 data streams collected by the hexapod, and average each one of those streams to obtain a BD of dimension 18. The comparison between this variant and AURORA will be appropriate to evaluate the utility of the automatic dimensionality reduction algorithm.

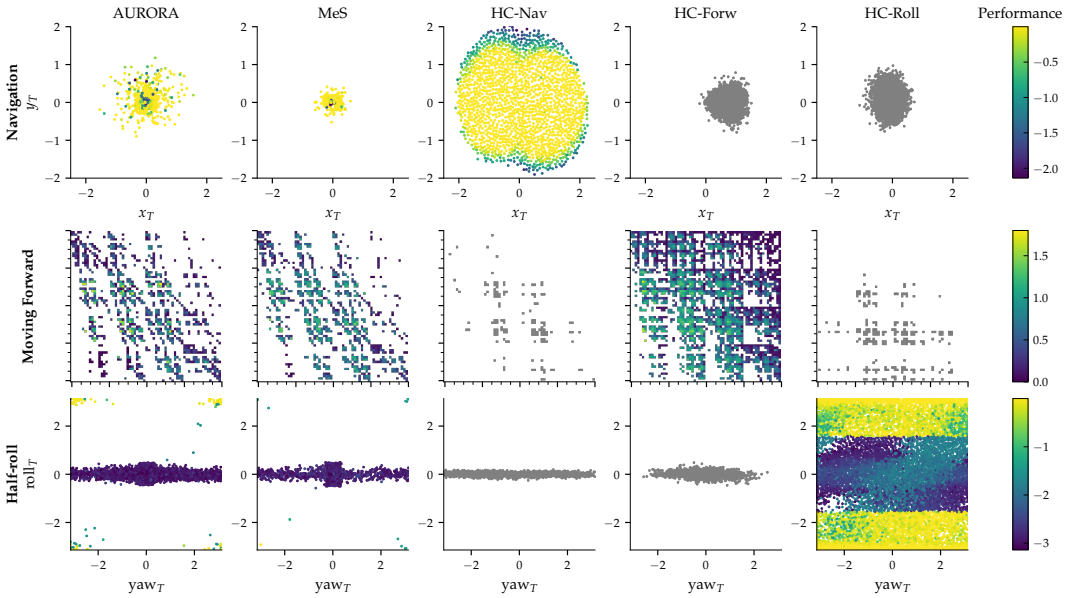


Figure 2: Containers returned by the different variants (columns) for each Quality-Diversity task (rows). Each dot corresponds to one policy present in the container, after projection in the Behavioural Descriptor space $\mathcal{B}_\mathcal{T}$ of the QD task. The colour of each dot is representative of the performance score of the policy. If the container result from a Hand-Coded variant that is not associated to the task (e.g. the container generated by HC-Nav for the Half-roll task), then the solutions are shown in grey. Also, note that the BD space of Duty Factors (second row) has six dimensions, the containers are presented using the same type of grids as in the work of Cully et al. (2015).

5.6 Implementation Details

All our experiments were run for 15,000 iterations with a uniform QD selector. We only use polynomial mutations Deb & Agrawal (1999) as variation operators with $\eta_m = 10$, and a mutation rate of 0.3. The target container size set for all algorithms is set to $N_\mathcal{C}^{target} = 5{,}000$ for the Moving Forward and the Half-roll tasks. In the case of the Navigation Task, we set this number to $N_\mathcal{C}^{target} = 1{,}500$ to obtain appropriate comparisons between the different approaches. To keep the container size around $N_\mathcal{C}^{target}$, we perform a container update every $T_\mathcal{C} = 10$ iterations (as explained in Section 2.1).

All our implementation is based on $\text{Sferes}_{\text{v2}}$ (Mouret & Doncieux, 2010) and uses on the DART simulator (Lee et al., 2018), while the auto-encoders are coded and trained using the C++ API of PyTorch (Paszke et al., 2019). For each QD task, all variants were run for 10 replications.

6 Results

Our objective is to show that AURORA discovers diversity in all directions of the BD space. To that end, we take the containers returned by the different algorithms, and we project them in the BD spaces associated to each Quality-Diversity (QD) task. Then we evaluate the achieved coverage in the BD space by considering the coverage for several minimum performance scores.

From Figure 3, it can be noticed that AURORA systematically obtains a lower total coverage than the variant using the hand-coded BD of the task (e.g. HC-Forw for the Forward task). Such variants use a BD that is well-adapted to the QD problem to solve; but this kind of BD requires human expertise to be defined.

AURORA can also be compared to Hand-coded containers after a projection in the BD spaces of other tasks, such as the container from HC-Nav projected in the BD space of the Moving Forward task. We notice that those kinds of containers (shown in grey in Figs 2 and 3) achieve a coverage that is at most equivalent, if not worse, compared to the containers of AURORA. For example, in the

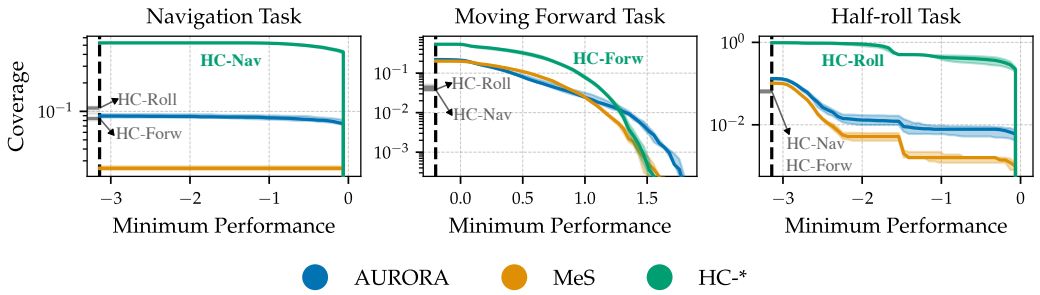


Figure 3: Coverage per minimum performance (presented in section 5.4), considered with all downstream tasks under study. If the container results from a Hand-Coded variant that is not associated to the task (e.g. the container generated by HC-Forw for the Navigation task), then the performance of solutions are not considered, and only the total coverage is presented. The vertical dashed line shows where to read the total coverage (without any minimal performance) achieved by each variant. For each variant, the bold line represents the median coverage; and the shaded area indicates the interquartile range.

Half-roll task, the Hand-coded baselines HC-Nav and HC-Forw do not find any solution making the hexapod fall on its back; whereas AURORA discovers such behaviours (see in corners of Half-roll containers in Fig. 2). In the end, AURORA manages to find containers of behaviours exhibiting diversity in all BD spaces.

The MeS variant, whose BD corresponds to the mean of each stream of sensory data, does not provide the same level of efficiency as AURORA. In the three tasks, AURORA obtains a coverage that is higher or equal to the coverage obtained for MeS (Fig. 3). When considering high minimum performance, the gap between the coverages from AURORA and MeS is even more apparent. This suggests that the automatic definition of an unsupervised BD may improve the performance upon hand-defined BDs, even if those are calculated considering all full states of the robot. In the end, by considering the entire full-state trajectory of the agent, AURORA can learn a large diversity of skills in an unsupervised manner, which is more general than any of the compared methods.

7 Conclusion and Future Work

In this paper, we provided an additional analysis of AURORA, a QD algorithm that automatically defines its BDs. In that analysis, we studied what kind of behaviours emerge when AURORA encodes the full state-trajectory of an hexapod robot. Our experimental evaluation demonstrated that AURORA autonomously finds diverse types of behaviours such as: navigating to different positions, using its legs in different manners, and performing half-rolls.

In order to limit the computational complexity of AURORA, the container is forced to have a limited size (usually of the order of magnitude of 10,000). That property may prevent the algorithm from exploring in detail all the different types of behaviours in open-ended environments. For example, it would be interesting to obtain simultaneously a complete behavioural sub-container specialised in half-rolls, and another sub-container specialised in navigation. This problem has been efficiently addressed by Etcheverry et al. (2020) with IMGEP-HOLMES, an algorithm that manages to discover several specialised niches of behaviours. It would be interesting to study how AURORA could integrate the same mechanisms as IMGEP-HOLMES to handle open-endedness. It would also be relevant to try our approach on more open-ended problems, where the robot can interact with various objects and other robots.

References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

David M Bossens, Jean-Baptiste Mouret, and Danesh Tarapore. Learning behaviour-performance maps with meta-evolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 49–57, 2020.

Konstantinos Chatzilygeroudis, Vassilis Vassiliades, and Jean-Baptiste Mouret. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Antoine Cully. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, pp. 81–89. Association for Computing Machinery, Inc, jul 2019. ISBN 9781450361118.

Antoine Cully and Yiannis Demiris. Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, apr 2018. ISSN 1089778X.

Antoine Cully and Jean Baptiste Mouret. Behavioral repertoire learning in robotics. In *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, pp. 175–182, New York, New York, USA, 2013. ACM Press. ISBN 9781450319638.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

Kalyanmoy Deb and Samir Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pp. 235–243. Springer, 1999.

Miguel Duarte, Jorge Gomes, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evorbc: evolutionary repertoire-based control for robots with arbitrary locomotion complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 93–100. ACM, 2016.

Mayalen Etcheverry, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Hierarchically organized latent modules for exploratory search in morphogenetic systems. *Advances in Neural Information Processing Systems*, 33:4846–4859, 2020.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Luca Grillotti and Antoine Cully. Unsupervised behaviour discovery with quality-diversity optimisation. *arXiv preprint arXiv:2106.05648*, 2021.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Edward Johns, Stefan Leutenegger, and Andrew J Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4461–4468. IEEE, 2016.

Rituraj Kaushik, Pierre Desreumaux, and Jean-Baptiste Mouret. Adaptive prior selection for repertoire-based online adaptation in robotics. *Frontiers in Robotics and AI*, pp. 151, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jeongseok Lee, Michael X Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S Srinivasa, Mike Stilman, and C Karen Liu. Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500, 2018.

Antonios Liapis, Héctor P Martínez, Julian Togelius, and Georgios N Yannakakis. Transforming Exploratory Creativity with DeLeNoX. Technical report, 2013.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021a.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34, 2021b.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pp. 52–59. Springer, 2011.

Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 149–156, 2016.

J.-B. Mouret and S. Doncieux. SFERESv2: Evolvin' in the multi-core world. In *Proc. of Congress on Evolutionary Computation (CEC)*, pp. 4079–4086, 2010.

Giuseppe Paolo, Alban Laflaquiere, Alexandre Coninx, and Stephane Doncieux. Unsupervised learning and exploration of reachable outcome space. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2379–2385. IEEE, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.