2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Model-Based Quality-Diversity Search for Efficient Robot Learning

Leon Keller[1], Daniel Tanneberg[1], Svenja Stark[1], Jan Peters[1,2]

Abstract— **Despite recent progress in robot learning, it still remains a challenge to program a robot to deal with open-ended object manipulation tasks. One approach that was recently used to autonomously generate a repertoire of diverse skills is a novelty based Quality-Diversity (QD) algorithm. However, as most evolutionary algorithms, QD suffers from sample-inefficiency and, thus, it is challenging to apply it in real-world scenarios. This paper tackles this problem by integrating a neural network that predicts the behavior of the perturbed parameters into a novelty based QD algorithm. In the proposed Model-based Quality-Diversity search (M-QD), the network is trained concurrently to the repertoire and is used to avoid executing unpromising actions in the novelty search process. Furthermore, it is used to adapt the skills of the final repertoire in order to generalize the skills to different scenarios. Our experiments show that enhancing a QD algorithm with such a forward model improves the sample-efficiency and performance of the evolutionary process and the skill adaptation.**

I. Introduction

Open-ended learning, or life-long learning, refers to a process where a robot has to autonomously acquire skills and knowledge by interacting with an environment *forever* [1]. In real-world scenarios robots are often exposed to changing environments or even need to solve new tasks without being trained on them beforehand, and not all future tasks can be foreseen to specify them for learning. As manually modifying the programming of the robot is time consuming and costly, it is a necessity that robots are able to deal with these challenges autonomously with little or no human intervention to develop throughout their lifespan [2].

A common task in life-long learning is to explore an unknown environment. Quality-diversity algorithms [3] have been used to make robots autonomously explore a user-defined behavior space [4]. These algorithms are typically evolutionary algorithms that aim at generating a repertoire of skills which is as diverse as possible by choosing the parents of each generation based on a novelty score. While these algorithms often are able to build a repertoire which covers the behavior space reasonably well, they suffer from sample-inefficiency – a crucial limitation for robotic applications. In this work, we address the sample-inefficiency problem by incorporating a neural network into the evolutionary process. This model is trained from the samples in the repertoire and used to select the most promising children at each generation, thus avoiding to execute children that are unlikely to improve the repertoire. Quality-diversity search algorithms (with and without model) produce a discrete skill repertoire. Thus, in order to generalize to arbitrary desired behaviors in a continuous behavior space, skills in the repertoire need to be adapted. Our approach achieves this adaptation by using the gradient descent approach from [4], extended by the learned neural network. The nearest neighbour of the desired behavior in the repertoire is selected and adapted to the desired behavior over multiple optimization steps.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No #713010 (GOAL-Robots) and No #640554 (SKILLS4ROBOTS). Calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

[1] Intelligent Autonomous Systems, Technische Universität Darmstadt
[2] Robot Learning Group, Max-Planck Institute for Intelligent Systems

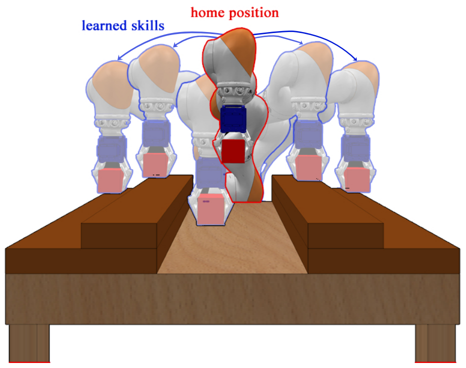


Fig. 1: Overlay of four learned skills from the skill repertoire obtained by M-QD on the `Place` task, where the goal is to place the grasped objects safely onto the table and shelves.

The contribution of this work is twofold: First, we propose a Model-based Quality-Diversity search (M-QD) algorithm that extendes QD search with a neural network that learns to score actions. Second, we propose an action adaptation strategy using this learned model. We show for both cases that adding the model helps to increase sample efficiency, and therefore reduces the number of expensive robot rollouts. We evaluate the proposed M-QD and the adaptation with multiple tasks in a 2D environment and a robot environment.

A. Related Work

A popular family of algorithms that has arisen from novelty search [5] is called Quality-Diversity search (QD) [3] which has been applied to robotic tasks [4]. These QD algorithms rate solutions based on a quality function and aim at finding qualitative solutions, in addition to building a diverse repertoire covering a defined behavior space as good as possible. The Novelty Search with Local Competition algorithm (NSLC) [6], for example, tries to achieve these objectives by using a multi-objective genetic algorithm to optimize a nearest neighbour-based local quality function in addition to the novelty of found solutions. Behavioral repertoire evolution (BR-Evolution) [7], in contrast, tracks all found solutions in an archive and progressively improves the quality of this archive by replacing solutions with better solutions that have a similar behavior. The multi-dimensional archive of phenotypic elites (MAP-elites) [8], [9] discretizes the behavior space in multiple bins and iteratively fills these bins with high quality solutions.

When using such novelty search algorithms to build a repertoire for a continuous behavior space, the skills of the discrete repertoire have to be generalized in order to estimate actions for arbitrary desired behaviors. One recent approach [4] tries to achieve that by applying gradient descent, where the gradient of the mapping between actions and behaviors is estimated using a local linearization and samples from the repertoire. However, this approach often needs many adaptation steps and fails if the task is highly non-linear. Another approach uses a behavioral repertoire as a training set for a conditioned generative adversial network [10] and then generates solutions for a given goal using this network. Similarly, the map-based Bayesian optimization algorithm (M-BOA) [11] adapts a behavior-performance map, generated by MAP-elites, using Bayesian optimization.

Models have been integrated into evolutionary search processes in a number of ways [12]. Estimation of distribution algorithms [13] iteratively train a model to estimate the probability distribution of promising solutions and then, instead of mutation and crossover operators, use this model to generate new populations. Other algorithms directly try to build an inverse model of the mapping between objective and decision space [14], [15], [16]. Lastly, in a process called surrogate modelling [17], [18], [19], [20], [21], many algorithms try to build a model of the objective function in order to reduce the number of computationally or otherwise expensive fitness evaluations.

Similar to our method, the Surrogate-Assisted Illumination (SAIL) algorithm [22] aims to minimize the number of evaluations of a novelty search algorithm by integrating an approximated model of the objective function.

Our method is based upon a quality-diversity algorithm which iteratively builds an unstructured repertoire of skills [4]. However, our approach is not purely evolutionary: similar as in model-based evolutionary learning, we incorporate a model into the QD algorithms to accelerate learning. Compared to popular model-based evolutionary learning, our method is most similar to surrogate modelling, however, our model predicts the behavior and quality of a given candidate instead of the fitness with respect to an objective function. By using the predicted behavior and quality, we can eliminate samples that are unlikely to find a novel or to improve a known behavior, resulting in less samples that have to be evaluated on the robot.

II. Learning A Repertoire of Diverse Skills

As described in the introduction, our approach consists of two distinct parts. The first part focuses on learning a diverse repertoire of skills covering a user-defined behavior space, whereas the second part focuses on adapting these skills in order to reach behaviors which are not present in the repertoire. We first introduce the quality-diversity search algorithm which builds the foundation of our approach, and introduce our model-based quality-diversity search variation. Lastly, we describe the gradient descent approach which is used to adapt the skills of the learned repertoires.

A. Formalization

We define an action $a \in A \subseteq \mathbb{R}^n$ as a vector of real valued parameters. When an action gets executed, the corresponding environment transitions from a start state $s_{\text{start}} \in S \subseteq \mathbb{R}^n$ to a destination state $s_{\text{dest}} \in S \subseteq \mathbb{R}^n$. We refer to this destination state as the behavior of an action. This behavior $s_{\text{dest}} = b$ is task-specific and can include any observable state of the robot (e.g. joint angles) and/or its environment (e.g. position of objects). Additionally, each action gets assigned a task-specific quality score q. Actions are evaluated with respect to their behavior b and their quality q using the evaluation functions

$$f_b(a) = b \quad \text{and} \quad f_q(a) = q \ .$$

We refer to the tuple $s = (a, b, q)$ as a skill, including its action a, and the corresponding behavior b and quality q.

B. Quality-Diversity Search

Quality-Diversity (QD) search is a novelty-based evolutionary search algorithm. In contrast to classical evolutionary algorithms, the evolution is mainly driven by novelty, which measures how unique the behavior of an skill is, instead of a fitness function. Additionally, QD uses a task-specific quality score to prefer better performing actions whenever multiple actions produce a similar behavior. The goal of a QD algorithm is to learn a repertoire of diverse and good performing skills, covering a given or learned behavior space.

The novelty of skills in the repertoire is measured according to a novelty score function. In this work, we use the average Euclidean distance to the K nearest neighbours as novelty score function, similar as in [4], [5] and given by

$$\text{nov}(a) = \frac{1}{K} \sum_{i=1}^{K} d(f_b(a_{nn_i}), f_b(a)) \ ,$$

where $d(.)$ denotes the Euclidean distance and $\{a_{nn_1}, \ldots, a_{nn_K}\}$ are the K nearest neighbours of a, measured in behavior space. More precisely, the nearest neighbours are given by the K actions in the repertoire whose behavior is closest to the behavior of a, measured using the Euclidean distance.

The algorithm is initialized with an empty repertoire. The first generation of skills is created randomly by sampling uniformly in action space. All following generations are created by applying crossover and mutation operators to parent skills, which are sampled from the repertoire proportionally to their novelty score. Every time a new generation is created, all its actions get evaluated in order to get their behavior and quality. After that, every candidate skill is compared to its nearest neighbour in the current repertoire. If the distance between the behavior and its closest neighbour is greater than a predefined threshold value t_{dist}, the new skill is added to the repertoire. If the distance is smaller than the threshold t_{dist}, the new skill replaces the closest neighbour, if it performs

Algorithm 1 Model-based quality-diversity search (M-QD)

```
Initialize empty repertoire
Initialize model $\phi$ with random parameters
while current_generation $\leq$ max_generation do
    parents $\leftarrow$ repertoire.sample(population size)
    population $\leftarrow$ populate(parents)
    survivor $\leftarrow \emptyset$
    for all action a $\in$ population do
        $\tilde{b}$, $\tilde{q} \leftarrow \phi$(a)
        if $\widetilde{\text{nov}}(a, \tilde{b}) > t_{\text{nov}}$ or $\widetilde{\text{qua}}(a, \tilde{q}) > t_{\text{qua}}$ then
            b, q $\leftarrow$ evaluate(a)
            $b_{nn}, q_{nn} \leftarrow$ repertoire.closest(b)
            if $d(b, b_{nn}) > t_{\text{dist}}$ then
                repertoire.add((a, b, q))
            else if $q > q_{nn}$ then
                repertoire.replace(nn, (a, b, q))
            end if
        end if
    end for
    update novelty score of all skills in the repertoire
    update model $\phi$ using mini-batches from the repertoire
end while
```

better according to the task-specific quality score. Otherwise, the skill is not added to the repertoire. All skills that are removed or not added to the repertoire are saved in a separate archive. After all skills of a generation are processed, the novelty score of all skills in the repertoire is updated and a new generation is formed. This process is repeated for a number of generations in order to form a large, diverse and highly qualitative repertoire.

C. Model-Based Quality-Diversity Search (M-QD)

In order to increase the sample-efficiency of the QD-search, we incorporate a model into the evolutionary process described in the previous section. The model $\phi(a) = (b, q)$ is trained to predict the behavior b and quality q for a given action a. As a model, fully-connected feed-forward neural networks with one hidden layer are used with `ReLU` activation for the neurons in the hidden layer and `tanh` for the neurons of the output layer, in order to bound the output of the network. The model is trained in an online fashion by using normalized mini-batches of skills sampled from the repertoire to perform gradient descent steps with the *Adam* [23] optimizer after every generation to minimize the mean-squared error. In order to avoid overfitting to early samples, training only starts after a warm up phase, in which the repertoire is filled with initial skills using the previously described quality-diversity search without the model.

In the evolutionary process, the model is used to predict the behavior $\tilde{b}$ and the quality $\tilde{q}$ of all new candidate actions a in the current generation before executing them. Based on these predictions, a novelty and a quality improvement score are estimated for each action

$$\widetilde{\text{nov}}(a, \tilde{b}) = \frac{1}{K} \sum_{i=1}^{K} d(f_b(a_{nn_i}), \tilde{b}) \ ,$$

$$\widetilde{\text{qua}}(a, \tilde{q}) = \frac{1}{K} \sum_{i=1}^{K} (\tilde{q} - f_q(a_{nn_i})) \ ,$$

where $d(.)$ denotes the Euclidean distance and $\{a_{nn_1}, \dots, a_{nn_K}\}$ are the K nearest neighbours of a, measured in behavior space. All actions with a novelty score greater than threshold t_{nov} are evaluated and eventually added to the repertoire, following the same procedure as in the quality-diversity search without model. Actions with a predicted novelty below the threshold t_{nov} are only evaluated if their predicted quality improvement score is greater than the threshold t_{qua}. Otherwise, they are discarded without executing them, assuming that they are unpromising and would not improve the repertoire. Due to this procedure, the sample efficiency is increased as samples that are unlikely to produce a novel behavior or improve a known behavior do not need to be executed in an expensive rollout.

In practice, the hyper parameters t_{nov} and t_{qua} need to be chosen carefully. When chosen too low, no actions would be discarded, nullifying the benefit of the model. In contrast, when chosen too high, too many actions would be discarded, putting an excessive confidence on the model's predictions and, thus, hinder exploration of the behavior space.

D. Adaptation of Skills

While the QD-search typically produces a repertoire which covers the behavior space reasonably well, it is still discrete and thus we have to adapt skills in order to reach arbitrary given behaviors. For that purpose, we build upon the gradient descent approach proposed in [4] and extend it with the learned model. Similar as in [4], for a given desired behavior b^*, the skill which is closest in behavior space $s_{nn} = (a_{nn}, b_{nn}, q_{nn})$ is selected from the repertoire and used to initialize the gradient descent $(a_j, b_j)|_{j=0} = (a_{nn}, b_{nn})$. Afterwards, a new action is computed by performing one step of the gradient descent given by

$$a_{j+1} = a_j + \lambda \widetilde{J}(a_j)^+ (b^* - b_j) \ ,$$

where λ is the adaptation step size. Evaluating this new action a_{j+1} obtains its behavior $b_{j+1} = f_b(a_{j+1})$. This process is repeated until the difference between b_j and b^* is smaller than a threshold t_{dist} or the number of total performed steps is greater than a predefined value.

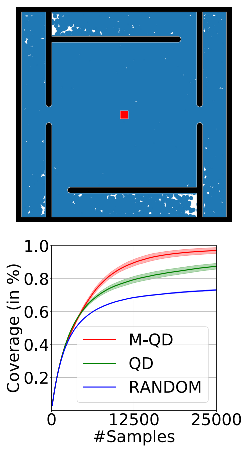
In [4] the gradient $\widetilde{J}(a_j)$ is estimated from samples in the repertoire using a least squares approach and a local linearization of the mapping between action and behaviors

$$\widetilde{J}(a_j) = BG^T(GG^T)^{-1} \ ,$$

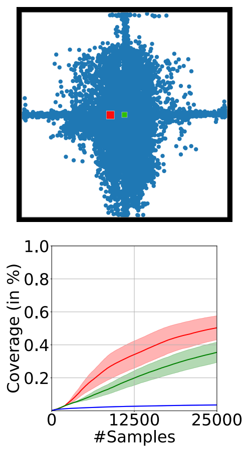
where $G = [a_{nn_1}, \dots, a_{nn_K}] - a_j$ and $B = [b_{nn_1}, \dots, b_{nn_K}] - b_j$ and $\{(a_{nn_1}, b_{nn_1}), \dots, (a_{nn_K}, b_{nn_K})\}$ are the K nearest neighbours of a_j in action space.

In order to obtain a better estimation our adaptation strategy, in contrast, estimates the gradient $\widetilde{J}(a_j)$ using the learned neural network model.

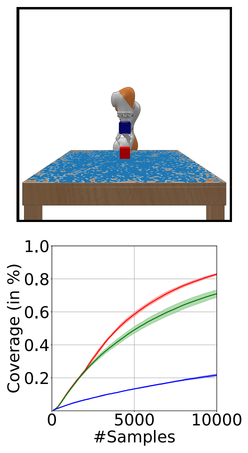
As our neural network model is differentiable, we can directly use its analytical derivative to obtain the gradient $\widetilde{J}(a_j)$. However, given a non differentiable model, the gradient can be estimated, for example, by using a finite difference approach given by $\widetilde{J}(a_j) = \frac{\phi(a_j+h) - \phi(a_j)}{h}$, where $\phi(.)$ denotes the learned model.



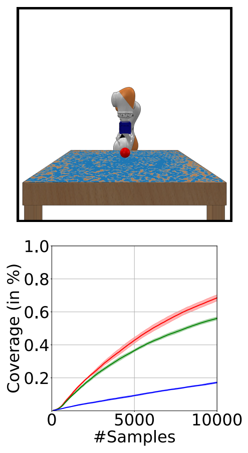
(a) `2D Obstacle`



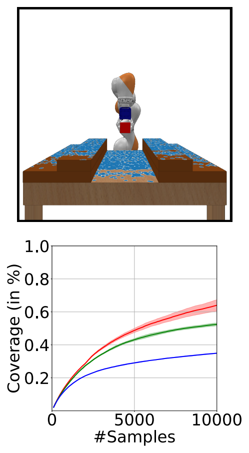
(b) `2D Object`



(c) `PushCube`



(d) `PushSphere`



(e) `Place`

Fig. 2: (Top) Overlays of the setup of the tasks and repertoires learned using M-QD. Each blue point corresponds to a behavior archived by a skill in the repertoire. (Bottom) Evolution of the behavior-space coverage on the different tasks. Results are averaged over 25 random seeds (10 for the robotic tasks), the plots show the mean and one standard deviation.

III. Experiments

We evaluate and compare the performance of M-QD in two environments and, in total, five different tasks. First, we introduce these tasks and describe their basic setup and goal. Consecutively, we report the empirical results of the conducted experiments, demonstrating the benefits of M-QD.

A. 2D Environment

We first implemented a 2D environment which consists of a bounded two-dimensional space and an agent which is placed on a predefined position in this space. An action $a = (a_0, a_1, \ldots, a_n)$ is composed of multiple sub-actions $a_i = (\dot{x}, \dot{y}, T)$, which are executed consecutively. Each sub-action consists of a velocity in $\dot{x}$ and $\dot{y}$ direction and a duration T. When executing a sub-action, $\dot{x}$ and $\dot{y}$ are directly applied to the agent for T time steps.

1) 2D Obstacle Task: For the first task, a number of obstacles are added to the task space. An action consists of three sub-actions and the behavior of an action is defined as the final position of the agent. The goal is to learn a repertoire of skills which enables the agent to move to every point in the space, thus navigating around the obstacles. Figure 2a shows the whole setup of this `2D Obstacle` task, including the behavior space coverage after learning.

2) 2D Object Task: For the second task, a move-able object is placed on a predefined position in the task space. An action consists of five sub-actions and the behavior of an action is defined as the final position of this object. The goal is to learn a repertoire which enables the agent to push this object to an arbitrary position in the space. Figure 2b shows the setup of this `2D Object` task, including the behavior space coverage after learning.

Formally, for both tasks, the behavior and the quality of an action are defined as

$$f_{\text{b}}(a) = (x_{\text{fin}}, y_{\text{fin}}) \text{ and } f_{\text{q}}(a) = \frac{\delta_{\text{min}}}{\delta} \ ,$$

respectively, where $(x_{\text{fin}}, y_{\text{fin}})$ refers to the final position of the agent or object. The parameter δ_{min} refers to the minimum distance the agent/object has to move to reach the behavior corresponding to action a, and δ refers to the actually moved distance of the agent/object. Intuitively, if two actions lead to the same behavior, the action which moves a smaller distance is considered to be better.

B. Robotic Environment

For the second environment, we implemented a more complex robotic environment: It consists of an LBR iiwa 14 R820 robot by KUKA which is placed on a small platform beside a table (Figure 1). The joint angles of the robot are preset such that the endeffector points downwards and is exactly placed above the center of the table. An action is compounded of multiple via-points $a = (a_1, a_2, \ldots, a_n)$, where each via-point $a_i = (x, y, z, \gamma, T)$ consists of an x, y and z position as well as an orientation γ around the z-axis, and a duration of T time steps. The x- and y-orientation are kept fixed, such that the endeffector always points downwards. When executing an action, the robot moves along these via-points for the given durations.

The Virtual Robotics Experimentation Platform (V-REP) is used for simulation. We used the PyRep library [24], as it provides faster communication and parallel simulation. It is a toolkit for robot learning research, built on top of V-REP.

1) Object Pushing Task: For the first robotic task, a sphere or cube is placed on the center of the table. An action consists of two via-points, where the z-coordinates are kept fixed to the z-coordinate of the sphere or cube to increase the likelihood of interacting with the object. The gripper of the robot is kept closed during the whole movement. The behavior of an action is defined as the final object position. The goal is to learn a repertoire which enables the robot to push the object to an arbitrary position on the table. Figure 2c and 2d show the `PushCube` and `PushSphere` task setup, including the coverage of the learned skill repertoires. The quality of an action is defined s.t. actions which move the end-effector least are considered best, given by

$$f_{\text{q}}(a) = -\delta_{ee} \ ,$$

TABLE I: Final behavior-space coverage ratio of the repertoires. Results are averaged over 10 random seeds (25 for 2D environments), denoted are the mean and one standard deviation.

	M-QD	QD	RANDOM
`2D Obstacle`	**0.971 ± 0.018**	0.875 ± 0.017	0.731 ± 0.004
`2D Object`	**0.504 ± 0.073**	0.366 ± 0.064	0.035 ± 0.002
`PushCube`	**0.828 ± 0.059**	0.709 ± 0.047	0.216 ± 0.073
`PushSphere`	**0.685 ± 0.0185**	0.560 ± 0.009	0.171 ± 0.004
`Place`	**0.639 ± 0.034**	0.524 ± 0.009	0.349 ± 0.001

with δ_{ee} denoting the total distance the end-effector moved, measured by the Euclidean distance.

2) Object Placing Task: For the second robotic task, the robot has to learn to place an object safely onto a table and shelves. Therefore, additionally shelves with different heights are added to the table. The robot starts with a cube grasped with its end-effector. An action consists of five via-points, which are executed after each other, as described above. After the last via-point is reached, the gripper is opened and the cube is released. The behavior of an action is defined as the final position of the cube on the table or shelves. The quality of an action is defined such that skills in which the cube drops as least as possible are considered best, i.e., when the cube is placed safely onto the surface

$$f_{\mathrm{q}}(a) = d(\mathrm{obj}_s, \mathrm{obj}_e) \ ,$$

where obj_s refers to the position of the object after the last via-point is reached but before the gripper is opened, and obj_e refers to the final position of the cube on the surface. The goal is to learn a repertoire which enables the robot to place the cube on every position on the table as safe as possible, whereas safe here refers to a minimal release height. Figure 2d shows the whole setup and the covered behavior space, we refer to this task as `Place`. Figure 1 illustrates four skills learned on this task.

Formally, for all three robotic tasks the behavior of an action is defined as

$$f_{\mathrm{b}}(a) = (x_{\mathrm{fin}}, y_{\mathrm{fin}}) \ ,$$

where $(x_{\mathrm{fin}}, y_{\mathrm{fin}})$ refers to the final position of the object.

C. Results

In order to evaluate our approach we performed experiments on all five tasks and compared it against two baselines.

TABLE II: Hyperparameters used in the Experiments

	2D	Robotic
Max. generations	250	100
Population size	100	100
K	5	5
t_{dist}	0.02	0.02
t_{nov}	0.04	0.04
t_{qua}	0.0	0.0
Learning rate	1e-3	1e-3
Batch size	16	16
No. of batches	500	500
No. of warmup samples	1000	1000
No. of hidden neurons	32	64
Adaptation step size	0.1	0.1
No. of adaptation steps	10	10

1) Building A Repertoire: We first evaluated the learning of the skill repertoire, and compared the quality-diversity search with and without our proposed model extension. Additionally, we compared to another baseline, which we refer to as RANDOM: This baseline accumulates a repertoire by sampling actions uniformly from the action space. We ran all three algorithms for 250 generations in the 2D Environments and averaged the results over 25 random seeds. For the robotic environments, we ran the algorithms for 100 generations and 10 different random seeds. We compared the algorithms in terms of average quality and coverage of the behavior space of the final skill repertoire. In all our tasks, the behavior space is bounded. In order to compute the coverage of this bounded space, we define that each skill covers a circular space around its behavior, where the radius of that circle is given by the distance threshold of the quality-diversity search. The coverage of a repertoire in percent of the total behavior space is then given by the area of the union of all these circles, divided by the total area of the behavior space. The average quality of a repertoire is defined as the average quality of its skills.

Figure 2 shows the evolution of the coverage as well as the final repertoires overlayed into the setup of the different tasks. Table I reports the coverage of the final repertoires. In terms of coverage, the random search performed poorly on all the tasks compared to the quality-diversity approaches. Only on the 2D-Obstacle and the Place task, random search was able to build a repertoire which covers large parts of the behavior space, highlighting the simplicity of the search space in these tasks. The repertoires of QD and M-QD both cover the behavior space reasonably well, however, M-QD outperforms QD by a margin of more than 10 percent on all of the tasks, and reaches this coverage with less samples. In terms of average quality, M-QD and QD outperformed RANDOM, while the average quality of repertoires generated with M-QD was slightly below the repertoires generated with QD. For example, on the `2D Obstacle` task, RANDOM achieved an average quality of 0.63 ± 0.02, while the repertoires generated with QD and M-QD achieved an average quality of 0.83 ± 0.01 and 0.80 ± 0.01 respectively. We observed this small drop in average quality between M-QD and QD on every task, however, we consider it to be negligible, as in our application scenario a good coverage of the behavior space with less samples is more important.

2) Adaptation Of Skills: Next, the proposed model-based adaptation approach is evaluated. Therefore, we evaluated the gradient descent approach using the model to estimate the gradient as well as the approach which uses a local linearization. Additionally we compared the results to just using the nearest neighbour in the repertoire instead of applying an adaptation strategy. As repertoire we used the final repertoires produced by the model-based quality-diversity search algorithm for all the strategies. We randomly generated 1000 goals uniformly in behavior space and averaged the results of the different approaches over 25 different repertoires (10 for the robotic tasks).

We report the behavioral error as well as the number of

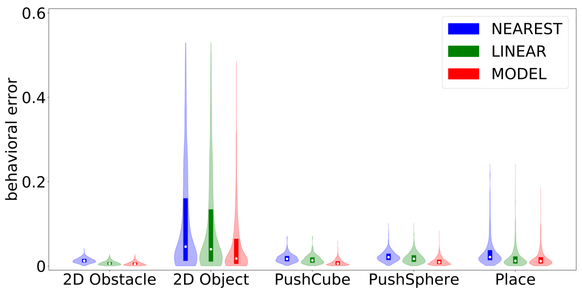


Fig. 3: Behavioral error after applying the different adaptation strategies. Results are averaged over 25 repertoires (10 for the robotic tasks) and 1000 randomly generated desired behaviors for each repertoire.

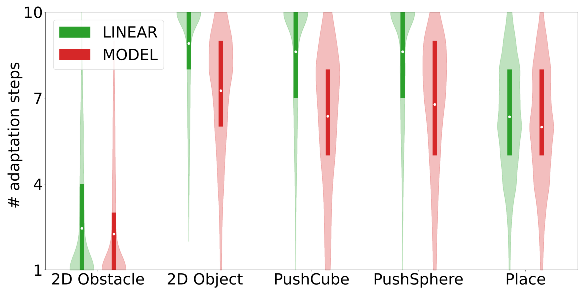


Fig. 4: Number of adaptation steps performed for a behavioral error below the distance threshold, with a maximum of 10 steps. Results are averaged over 25 repertoires (10 for the robotic tasks) and 1000 randomly generated desired behaviors for each.

required adaptation steps to achieve these errors in Figure 3 & 4. Both adaptation strategies outperform nearest neighbour on all the tasks. Adaptation with the learned model outperforms the local linearization adaptation. The difference between the model-based approach and the local linearization is bigger on more complicated environments and negligible on the simpler ones, as for example `2D Obstacle` and `Place`. We suspect that this is the case as in these simple environments the local linearization already yields a reasonable estimate of the true gradient. Furthermore, the model-based approach requires less adaptation steps than the linear, improving again the sample-efficiency.

IV. Conclusion

We proposed a Model-based Quality-Diversity search algorithm (M-QD) for learning a diverse repertoire of skills in an open-ended setting. The approach is evaluated empirically on a 2D environment as well as on a more complex robotic environment. The results of our experiments show that M-QD outperforms the model-free quality-diversity search in terms of sample-efficiency and behavior space coverage when learning a skill repertoire. Moreover, the results show that in comparison to a simple local linearization approach, using a model-based adaptation strategy reduces the number of needed adaptation steps and the average error in behavior space – improving the repertoire generalization and skill adaptation efficiency.

However, despite this twofold improved efficiency, the amount of rollouts required by the model-based quality-diversity search algorithm makes learning from scratch on a real system still challenging. Thus, in future research, we aim at integrating Sim2Real techniques, as for example presented in [4], into our approach in order to adapt repertoires and models learned in simulation to real robots, or to concurrently learn in simulation and on the real system.

References

[1] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and autonomous systems*, 1995.
[2] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: A survey," *IEEE transactions on autonomous mental development*, 2009.
[3] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, 2017.
[4] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: learning object manipulation skills through novelty search and local adaptation," *arXiv preprint arXiv:1901.00811*, 2019.
[5] J. Lehman and K. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, 2011.
[6] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011.
[7] A. Cully and J.-B. Mouret, "Behavioral repertoire learning in robotics," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 2013.
[8] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.
[9] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, "Scaling up map-elites using centroidal voronoi tessellations," *arXiv preprint arXiv:1610.05729*, 2016.
[10] M. Jegorova, S. Doncieux, and T. Hospedales, "Generative adversarial policy networks for behavioural repertoire," *arXiv preprint arXiv:1811.02945*, 2018.
[11] A. Cully, J. Clune, and J.-B. Mouret, "Robots that can adapt like natural animals," *arXiv preprint arXiv:1407.3501*, 2014.
[12] R. Cheng, C. He, Y. Jin, and X. Yao, "Model-based evolutionary algorithms: a short survey," *Complex & Intelligent Systems*, 2018.
[13] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2001.
[14] I. Giagkiozis and P. Fleming, "Pareto front estimation for decision making," *Evolutionary computation*, 2014.
[15] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, 2015.
[16] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Bayesian network as an adaptive parameter setting approach for genetic algorithms," *Complex & Intelligent Systems*, 2016.
[17] B. Wilson, D. Cappelleri, T. Simpson, and M. Frecker, "Efficient pareto frontier exploration using surrogate approximations," *Optimization and Engineering*, 2001.
[18] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, 2005.
[19] ——, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, 2011.
[20] M. Tabatabaei, J. Hakanen, M. Hartikainen, K. Miettinen, and K. Sindhya, "A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods," *Structural and Multidisciplinary Optimization*, 2015.
[21] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, 2017.
[22] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-efficient design exploration through surrogate-assisted illumination," *Evol. Comput.*, 2018.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.
[24] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.