Few-shot Quality-Diversity Optimization

Achkan Salehi, Alexandre Coninx, Stephane Doncieux

Few-shot Quality-Diversity Optimization

Achkan Salehi†, Alexandre Coninx†, Stephane Doncieux†

***Abstract*—In the past few years, a considerable amount of research has been dedicated to the exploitation of previous learning experiences and the design of Few-shot and Meta Learning approaches, in problem domains ranging from Computer Vision to Reinforcement Learning based control. A notable exception, where to the best of our knowledge, little to no effort has been made in this direction is Quality-Diversity (QD) optimization. QD methods have been shown to be effective tools in dealing with deceptive minima and sparse rewards in Reinforcement Learning. However, they remain costly due to their reliance on inherently sample inefficient evolutionary processes. We show that, given examples from a task distribution, information about the paths taken by optimization in parameter space can be leveraged to build a prior population, which when used to initialize QD methods in unseen environments, allows for few-shot adaptation. Our proposed method does not require backpropagation. It is simple to implement and scale, and furthermore, it is agnostic to the underlying models that are being trained. Experiments carried in both sparse and dense reward settings using robotic manipulation and navigation benchmarks show that it considerably reduces the number of generations that are required for QD optimization in these environments. Our code is available at https://github.com/salehiac/FAERY-original.**

***Index Terms*—Evolutionary Robotics; Reinforcement Learning; Transfer Learning**

I. Introduction

MANY engineering problems can be formulated as optimization problems, where the goal is to find a set of optimal or near-optimal parameters for a model. Obtaining such solutions is often made difficult by the presence of deceptive optima in the loss/fitness[1] landscape, and in the case of Reinforcement Learning by sparse rewards. In such situations, methods that solely rely on gradient or higher-order local signals tend to fail as a result of poor exploration.

In part because of their natural ability to deal with those issues, Quality-Diversity [1] (QD) methods have garnered considerable interest in the past few years, and have been applied to many problems, ranging from robotics [2], [3], [4] to video games [5], [6] and open-ended learning [7], [8]. The aim of these methods is to obtain a diverse set of well-performing solutions to a given problem. To do so, they not only aim to optimize the fitnesses of the agents in the population, but also maximize their *behavioral diversity* according to some (learned or hand-designed) behavior descriptor space. This results in divergent exploration of parameter space, which prevents the search from getting trapped in deceptive minima. Unfortunately, the underlying evolutionary processes on which QD approaches almost always rely make them highly data-inefficient, and they are thus expensive to train from scratch. Some works such as QD-RL [9] address the sample-inefficiency problem in dense reward settings by replacing evolutionary algorithms with policy-gradient methods and optimizing the population jointly for behavioral novelty and expected return. However, this does not work with sparse rewards and the resulting policies do not readily transfer to novel environments and need to be trained from scratch for each new task.

In this work, we consider a complementary but orthogonal direction that can handle both sparse and dense rewards. Our aim is, given a training set sampled from a task distribution $\mathcal{T}$, to learn a prior population that when used to initialize a QD algorithm to solve an unseen task $t_{new} \sim \mathcal{T}$ will require only few generations of training (instead of the typical hundreds or thousands). Our proposed approach, which we call FAERY, for **F**ew-shot Qu**A**lity-Div**ER**sit**Y** Optimization, maintains a prior population $\mathcal{P}$ that is used to initialize independent QD optimizations, and that is continuously refined by maximizing two meta-objectives. The latter are computed based on statistics gathered from the paths taken by the evolutionary processes in each of the QD instances. As the resulting prior $\mathcal{P}$ is used to improve the efficiency of learning, FAERY falls under the broad definition of meta-learning as "learning to learn", although it would be excluded by stricter, more recent definitions [10]. Note that the proposed method does not require gradients, which makes it suitable for sparse reward environments. Furthermore, it has the appealing characteristic of being easy to implement and scale.

We perform experiments on navigation tasks with sparse rewards in randomly generated mazes, and on robotic manipulation tasks from the Meta-World [11] benchmarks where we use the dense reward to accelerate the policy search. We distinguish intra-task generalization from cross-task generalization, both of which will be formally defined in section V. While the primary focus of the conducted experiments is on few-shot adaptation in the latter case, we also present results indicating that using the priors learned by the proposed approach on a source task t_{source} allows effective knowledge transfer to target tasks, especially in comparison to direct transfer of expert policies from t_{source} to the target environments.

We set the context and formalize our objectives in the following section (§II). Section III is dedicated to the positioning of our work with respect to the literature. We describe FAERY in section IV and present experimental evaluations on navigation tasks and meta-world in V. Closing discussions and remarks are the subject of VI.

II. Problem Formulation and Notations

We place ourselves in a Reinforcement Learning context and assume that each environment t_i sampled from a task distribution $\mathcal{T}$ can be modeled as a Partially Observable Markov Decision Process (POMDP) noted $< \mathcal{S}, \mathcal{A}, r, \rho, \gamma, \Omega, \rho_{obs}, \rho_{init} >$ where r is a reward function, $\rho : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the transition probabilities $\rho(s_{t+1}, a_t, s_t) = p(s_{t+1}|s_t, a_t)$ and $\rho_{obs}(o, s_t, a_t) = p(o|s_t, a_t)$ is the probability of and element of the observation space Ω given a state-action pair. The scalar $\gamma \in [0, 1]$ is a discount factor and ρ_{init}

Manuscript received September 9, 2021; accepted January 28, 2022. This letter was recommended for publication by Associate Editor T. Horii and Editor T. Ogata upon evaluation of the reviewers' comments.

This work was supported by the European Union's H2020-EU.1.2.2 Research and Innovation Program through FET Project VeriDream under Grant Agreement Number 951992.

† Sorbonne Université, CNRS, ISIR, F-75005 Paris, France. Email: {achkan.salehi, alexandre.coninx, stephane.doncieux}@sorbonne-universite.fr

Final version reference and link: A. Salehi, A. Coninx and S. Doncieux, "Few-shot Quality-Diversity Optimization," in IEEE Robotics and Automation Letters, doi: 10.1109/LRA.2022.3148438. https://ieeexplore.ieee.org/document/9705622

[1]The term fitness, borrowed from the evolutionary computation literature, is synonymous with return. In this paper, it specifically refers to discounted cumulative reward.

specifies the distribution over initial states. In this work, we consider that the initial state is fixed up to an infinitesimal perturbation, *i.e.* ρ_{init} is a Gaussian $\mathcal{N}(s_0, \epsilon)$.

As is usual in the literature [1], [12], [13], we broadly define the term Quality-Diversity (QD) as referring to any algorithm that combines fitness-based optimization with divergent exploration and produces a *behaviorally diverse* set of solutions that are locally optimal according to some algorithm-dependent notion of neighborhood. For example, in hexapod robot locomotion, one could consider walking speed as the fitness to optimize and the gait as the behavior [2]. A QD algorithm would ideally construct a set of controllers that, given a desired gait (*e.g.* due to damaged limbs) will be able to walk as fast as possible, resulting in increased robustness to environmental changes. It should be noted that in some cases as in navigation problems with sparse rewards [14], this diverse set of solutions —often referred to as an archive in the literature —is only useful to avoid convergence to deceptive local minima, and as a result is a byproduct of the algorithm, not the end goal.

To compute behavioral diversity, we assume the existence of a (hand-designed or learned) behavior function $\mathcal{B}(\tau) \mapsto \mathfrak{B}$ which maps each trajectory $\tau = \{(s_i, a_i)\}_i$ to a metric behavior space $\mathfrak{B}$. Based on that space, different criteria of behavioral diversity such as Novelty [14], Surprise [15] or Curiosity [1] can be considered. Note that Quality-Diversity methods usually enable dynamic exploration by storing a number of previously encountered behaviors and agents in an archive, which can be structured [16] or structureless [13]. However, which of these two categories an algorithm falls into and which diversity metric it uses is irrelevant to our work. This is because we address a drawback that is common to all the aforementioned QD methods, namely, their reliance on sample-inefficient evolutionary processes.

Given a problem $t \sim \mathcal{T}$, a typical QD instance will run for $g \in \{1, ..., \mathcal{G}_{QD}\}$ generations until the evaluation budget $\mathcal{G}_{QD}$ is exceeded, or a satisfaction condition (*e.g.* coverage and average fitness) is reached at some generation $\mathcal{G}^*$. Note that $\mathcal{G}^*$ is a random variable. In what follows, we will note

$$\hat{\mathcal{G}}(t) = \mathbb{E}[\mathcal{G}^*] \tag{1}$$

the expected number of generations in which a given QD method reaches the satisfaction conditions on an environment $t \sim \mathcal{T}$.

For an unseen problem $t_n \sim \mathcal{T}$, the value $\hat{\mathcal{G}}(t_n)$ is in practice often very high. Assuming a training set $\mathcal{T}_{train}$ with examples from $\mathcal{T}$, we formulate our aim as learning a prior ω that if taken into account by the QD algorithm will significantly decrease the number of generations that are required to solve t_n:

$$\mathbb{E}[\mathcal{G}^*|\omega] \ll \hat{\mathcal{G}}(t_n). \tag{2}$$

In this paper, we assume that a binary signal ψ_i can be associated to each task t_i such that

$$\psi_i(\theta_l) = \begin{cases} 1 & \text{if } \theta_l \text{ solves } t_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

holds.

This is a small restriction, as most problems can be formulated in this way. The objective of tasks that do not have clear goal states/behaviors usually is either return maximization or pure exploration. In those cases, one can set $\psi_i(\theta_i) = 1$ if the returns or the quality of exploration (according to some well-defined criterion) exceed some threshold.

Throughout all sections, $\Delta(\Theta, \lambda)$ will denote a mutation operator that will return λ policies resulting from perturbations made to the parameters of the population Θ, according to some mutation scheme. Furthermore, μ and λ will respectively denote population and offspring size.

III. Related Work

The proposed approach is naturally related to previous works on data-efficient QD as well as to the literature on few-shot and meta-learning.

Data-efficient QD. In general, most of those works fall into one of two categories. The first one is composed of methods that attempt to improve the data-efficiency of the evolutionary processes used in QD algorithms by using information from the reward signal or from diversity criteria to bias the sampling of new individuals from a parent population. While traditional QD methods do select the population based on fitness and diversity, they apply random mutations to their weights in a manner that is independent from the problem at hand, which is nearly always suboptimal. Recently, [17], [18] have proposed to combine populations of CMA-ES [19] instances (often called emitters in the literature) with QD algorithms, and both of these works show increased efficiency in sparse reward settings. Focusing on problem domains with dense rewards, QD-RL [9], replaces sample-inefficient evolutionary algorithms with policy-gradient methods and optimizes the population jointly for behavioral novelty and expected return. In a similar manner, ME-ES [20] leverages the more efficient ES strategies to replace the random exploration of vanilla MAP-elites. In other recent works [21], in addition to selecting and mutating most novel individuals and in order to accelerate behavior space coverage, the authors propose to push less novel individuals toward the most novel ones by creating targets in behavior space, in a manner reminiscent of Goal Exploration Processes [22].

Methods from the second family do not directly address the low sample-efficiency of the evolutionary methods. Instead, they attempt to model and learn different components from the decision process of interest in a data-efficient manner. For example, learning a model of the transition probabilities can improve data-efficiency on a real-word system: first, such a model allows sample-inefficient exploration to be carried out in simulation, thus reducing the need for evaluations on the real system. Second, it can be used to guide the search to areas in which the model expects higher returns. An example is the SAIL algorithm [23], where costly fitness evaluations are replaced by calls to a surrogate function that is estimated via Bayesian Optimization, which is both guided (to high reward regions) and exploited (for final predictions) by MAP-elites instances. In M-QD [24], the authors propose to learn a model that predicts the behavior and quality given an individual, thus avoiding parameter space areas that are associated with low quality or low behavioral novelty. Note that many of the problems that are addressed by this second family of approaches have also been studied in the context of model-based Reinforcement Learning [25], from which many ideas can be directly applied to QD optimization.

We emphasize that our proposed method is orthogonal to the approaches discussed above, as we address the construction of rapidly adaptable priors based on previous experience. That is, FAERY can be combined with the aforementioned approaches.

Few-shot learning and meta-learning. In supervised learning, many approaches to few-shot learning have been developed, ranging from using simple hand-crafted label signatures as priors [26] to more complex metric-learning [27] or meta-learning based [28] methods. In Reinforcement Learning, few-shot learning is almost always approached via meta-learning [29], [30], [31], [32], [33], [34], [35]. Earlier definitions of meta-learning include any algorithm that changes the meta-parameters of another learning algorithm, such that the performance of the latter is improved [36]. Under these definitions as well as more recent but similarly broad ones [37], approaches such as hyperparameter tuning via cross-validation [38] would fall into the meta-learning category. Thus, some recent attempts at providing a definition add some restrictions, such as the requirement for training conditions to replicate testing conditions and/or the requirement for end to end optimization of the inner algorithm with respect to the outer objective [10]. The method we propose aims at improving the efficacy and efficiency of learning by exploiting meta-data associated with distinct training processes, but decouples the optimization of the meta-objectives from the inner optimization. As such, we feel that it

falls in between the aforementioned definitions. Whether we consider FAERY as meta-learning or not, it presents several advantages. First, in contrast with most works (*e.g.* [29], [34], [30], [39], [32], [31]), it does not require gradients. This not only makes the method more suitable for deceptive or sparse reward problems, but also leaves more freedom in the design and addition of additional meta-objectives. Second, like [29], [33], it is agnostic to the underlying models that are being trained. Finally, it is simple to implement and scale.

IV. FAERY

Given a task distribution $\mathcal{T}$, our aim is to leverage learning experiences on tasks that are sampled from it to learn a population of prior policies $\mathcal{P} = \{p_1, ..., p_\mu\}$. Those priors should be distributed in parameter space in a manner that facilitates future QD optimizations on new tasks $t_n \sim \mathcal{T}$. In other words, setting $\omega \triangleq \mathcal{P}$ should satisfy equation 2. We learn these priors by optimizing two meta-objectives, which intuitively quantify an agent's polyvalence and adaptation speed.

Our proposed method is summarized in algorithm 1. Each iteration of the outer loop optimizes the prior population based on statistics gathered from M distinct QD instances, that we will note $\{Q_i\}_{i=0}^{M-1}$. Note that given $i \neq j$, two instances Q_i and Q_j will in general run on two different environments t_i, t_j sampled from the same task distribution $\mathcal{T}$. Each Q_i will be initialized with a population $\mathcal{P} \cup \Delta(\mathcal{P}, \lambda)$, and will optimize that population during G_{QD} generations, using an evolutionary algorithm. The paths taken by the evolutionary process during the execution of Q_i can be represented as a forest, composed of μ trees, such that the root of the $i-th$ tree corresponds to the policy p_i from the prior population. This allows each solution found by the QD algorithm to be uniquely mapped to a single parent in the prior population[2] . An example of such a forest is given in figure 1.

Noting ξ_{ij} the set of descendants of p_j that result from the evolution process of Q_i, we define the two following fitnesses for each policy $p_j \in \mathcal{P}$:

$$\begin{cases} f_0(p_j) = \sum_{i=0}^{M-1} \sum_{s \in \xi_{ij}} \psi_i(s) \\ f_1(p_j) = \frac{-1}{f_0(p_j)} \sum_{i=0}^{M-1} \sum_{s \in \xi_{ij}} \psi_i(s) d_m(s, p_j). \end{cases} \tag{4}$$

where $d_m(s, p_j)$ returns the expected number of mutations necessary to reach s from p_j. We approximate that value with the depth of s in the forest associated to Q_i. The first objective in that equation, f_0, is the cumulated number of solutions that have evolved from p_j over all optimizations $\{Q_i\}_{i=0}^{M}$. Regarding the second objective, the value $-1 \times f_1$ corresponds to the number of mutations that are necessary to reach a solution from p_j, averaged over all environments $\{Q_i\}_{i=0}^{M}$. Both f_0 and f_1 are maximized.

Once those objectives are computed, any Pareto Optimization scheme (*e.g.* NSGA2 [40]) can be used to select the best performing individuals from $\mathcal{P} \cup \Delta(\mathcal{P}, \lambda)$. Those individuals will replace the ones in $\mathcal{P}$, and thus constitute the new prior that will be used in the next iteration of the algorithm.

The motivation behind the definitions given for the two fitness functions is perhaps better explained via a comparison with the way in which optimization-based meta-learning algorithms such as MAML [29] operate. In the latter, adaptability to many tasks is enabled by computing the meta-update based on the sum of the losses from different sampled tasks, and quick adaptation of the prior is encouraged by limiting the number of gradient updates and sample-splitting. In FAERY, those two roles are respectively carried out by the maximization of f_0 and f_1: maximizing f_0 drives the p_i to areas of parameter space where they can serve as priors for an increased number of tasks, and maximizing f_1, which minimizes the number of mutations, acts in a manner that is similar to reducing the number of gradient updates. Contrary to MAML-like approaches, however, the proposed optimization scheme can be used in both sparse and dense reward settings, as it does not rely on gradients.

[2]In this formulation, we have only considered QD algorithms that solely use mutations for evolution, and do not rely on crossover operations. The reasons for this choice are twofold: first, it simplifies notations. Second, crossover is less frequently used in the current literature. Nonetheless, extending our approach to handle crossovers is straightforward: each solution in the tree will result in updates to multiple parents instead of updates to a single one.

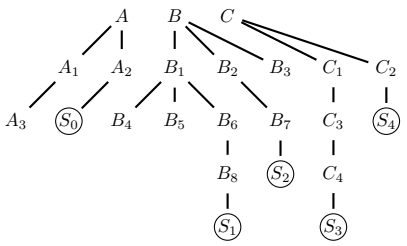

Fig. 1: Example of an evolution forest associated to a single toy QD instance Q_i, initialized with the prior population $\{A, B, C\}$. Each edge indicates a parent-offspring relationship between two nodes. The S_i are the solutions found by the QD algorithm. In this example, $f_0^i(A) = 1, f_0^i(B) = 2, f_0^i(C) = 2$ and $f_1^i(A) = 2, f_1^i(B) = 3.5, f_1^i(C) = 3$.

Algorithm 1: The FAERY algorithm

Input: Train task distribution $\mathcal{T}_{train}$, sample size M at each meta iteration, maximum number of meta iterations G_{outer}, prior population size μ, number of offsprings λ, maximum number of optimization steps per problem G_{QD}, maximum number s_{max} of successful policies per environment, constant factor c_λ specifying population size in each QD optimization, mutation operator Δ
Output: Prior population $\mathcal{P}$

```
P = InitializeRandomPopulation()
for g_o ∈ [0, G_outer) do
    tasks ← SampleTasks(T_train, M)
    offsprings ← Δ(P, λ)
    pop=offsprings ∪P
    for r ∈pop do
        // counter of number of solutions that evolve from r
        r.η ← 0
        // list of depths of solutions in the evolution tree of r
        r.D ← EmptyList()
        // fitnesses to maximize (see equation 4)
        r.f_0 ← 0
        r.f_1 ← −∞
    end
    for t_i ∈ tasks do
        solutions, evolution_paths ←
          QualityDiversityAlgorithm(t_i, ψ_i, pop, c_λ,
          s_max)
        for s ∈ solutions do
            // get root of corresponding evolution tree
            r ← GetRoot(s, evolution_paths)
            r.η ← r.η + 1
            r.D ← concatenate(r.D, {s.depth})
        end
    end
    for r ∈pop do
        if r.η ≠ 0 then
            r.f_0 ← r.η
            r.f_1 ← −1× mean(r.D)
        end
    end
    // e.g. with NSGA2
    P ← ParetoBasedSelection(pop, f_0, f_1)
end
```

V. EXPERIMENTS

We first provide formal definitions and outline the objectives of the experiments.

Single-task vs cross-task generalization. In this section, we will consider that each task t_i has a set of goal areas $\mathcal{H}(t_i) = \{h_1^i, ..., h_s^i\}$ in some Euclidean *goal space*. The objects of interest in the environment (*e.g.* objects to assemble in a robot manipulation task, or walls in a maze) will be noted $\Upsilon(t_i) = \{o_1^i, ..., o_v^i\}$.

Let $\mathcal{T}$ denote a task distribution. We will consider that $\mathcal{T}$ is a *single-task distribution* iif the goals and initial object poses of any two environments $t_i, t_j \sim \mathcal{T}$ (with $i \neq j$) are related by Euclidean transformations: $\forall h^i \in \mathcal{H}(t_i), \forall o^i \in \Upsilon(t_i)$, there exists a unique $h^j \in \mathcal{H}(t_j)$ and a unique $o^j \in \Upsilon(t_j)$ such that $h^j = R_h^{ij}(h^i)$ and $o^j = R_o^{ij}(o^i)$, where R_h^{ij}, R_o^{ij} are some Euclidean transforms. An example of such task distributions is a robot manipulation task where the aim is to grasp a ball and throw it in a basket, and where the poses of the ball and the basket are randomly initialized. We will say about experiments evaluating the generalization capacities of an algorithm with such a $\mathcal{T}$ that they assess the *single-task* or *intra-task* generalization capabilities of that algorithm. In such cases, it seems reasonable to expect some degree of generalization, *i.e.* that knowledge acquired in some t_i is likely to help in solving some t_j from that distribution in the future.

Now, consider a task distribution $\mathcal{T}_{manip}$ which is used to randomly assign dramatically different tasks to a robotic arm. For example, $t_i, t_j \sim \mathcal{T}_{manip}$ could respectively correspond to assembling lego pieces and hitting a ball so that it goes through a hole. In that case, it is not immediately obvious whether knowledge gained in one environment will facilitate learning in the other. The corresponding POMDPs could greatly differ. We will refer to generalization in such a setting as *cross-task* generalization. In this work, we restrict ourselves to cross-task transfer between POMDP distributions which share the same action set, and whose state-spaces have the same dimensionality (without necessarily overlapping).

Note that in contrast with many works on transfer learning, we are not interested in transfer between two different fixed POMDPs, but between two POMDP distributions.

Experiments outline. While FAERY is formulated with both intra-task and cross-task generalization in mind, we will primarily focus on evaluating its use in few-shot adaptation for single-task generalization (§V-A). Regarding the cross-task setting, we only provide cross-task knowledge transfer results based on priors learned on a source distribution (§V-B). The reason for this is that large-scale experiments in the cross-task setting are costly, and so we leave exhaustive investigations into that matter for future works.

In all experiments, we used NSGA2 [40] with Novelty [3] [14] and fitness objectives as the underlying QD method, in place of the `QualityDiversityAlgorithm` placeholder from algorithm 1. As the data-efficiency issues that are addressed by FAERY stem from evolutionary mechanisms that are common to all QD families, it is reasonable to expect that results obtained with NSGA2 will also be a strong indicator of what can be expected of the application of FAERY to other QD algorithms.

The policies are implemented as fully connected feed-forward neural networks with $tanh$ activations and the bounded polynomial operator [40] is used for mutations. As recommended in [41], the Novelty objective is computed based on the $k = 15$ nearest neighbors, using a maximum archive size of 5000 individuals. We will note M_{train}, M_{test} the number of training and testing environments. Note that train/test datasets are disjoint in all the experiments, *i.e.* an environment sampled from the test distribution $\mathcal{T}_{test}$ has zero probability of being sampled from $\mathcal{T}_{train}$ and vice versa.

[3]For the reader's convenience, the definition of Novely is stated in this footnote. Consider a policy θ and its behavior descriptor b_θ. Let $R_{ref} \triangleq \{\theta_j\}_j$ be a reference set (usually composed of the current population and a subset of previously visited individuals). Then the Novelty score of θ is computed as $\frac{1}{K}\sum_{i=0}^{K} d(b_\theta, b_{\theta_i})$ where the $\{\theta_i\}_{i=0}^{K}$ are such that the corresponding b_{θ_i} are the K nearest neighbors of b_θ in R_{ref}.

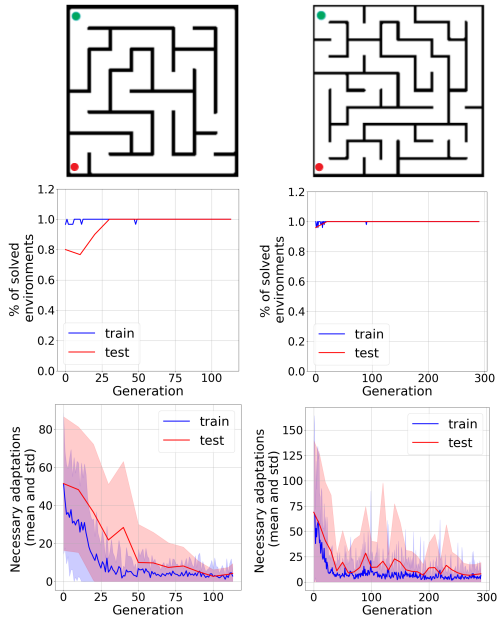

Fig. 2: Examples of random mazes sampled from the maze distributions, and the results of the proposed method (second and third rows). The first and second columns respectively correspond to 8×8 and 10×10 mazes. The horizontal axis of the plots in the second and last rows indicates the number of generations for which the prior population has been optimized by FAERY (*i.e.* the number of meta-updates), which should not be confused with the number of generations needed for a single QD algorithm to converge. Notice that on this axis, generation 0 corresponds to a QD optimization that is performed from scratch as no priors have been learned up to this point. Note that all train/test environments that are sampled at generation i on this axis will have their corresponding QD instances initialized using the priors that are obtained from generation $i-1$. **(Top row)** Example mazes that are sampled from the 8×8 and 10×10 maze distributions. **(Middle row)** The ratio of environments among the M sampled ones that the QD instances are able to solve. **(Bottom row)** the average number of generations necessary to solve the tasks (that are solvable at a given generation).

A. Few-shot Learning For Single-Task Generalization

Our aim in this section is to demonstrate for the single-task setting, in both sparse and dense reward scenarios, that the proposed algorithm reduces the number of generations necessary for QD algorithms to obtain a set of diverse but well-performing agents. We performed experiments in those two environments:

- Navigation tasks in randomly generated mazes composed of 8×8 and 10×10 cells.
- Robotic manipulation environments from the meta-world [11] benchmark, which is based on the mujoco physics engine[4].

The mazes differ from each other in the placement of walls and thus in layout, and the manipulation tasks differ in the poses of the goals and initial placements of objects of interest.

1) Few-Shot Maze Navigation: For this experiment, two datasets of randomly generated mazes[5] were used. The first one, that we will designate by `mazes8x8` is comprised of 1200 train and 200 test mazes, where the wall positions have been sampled from an 8×8 grid. The second dataset, `mazes10x10` is comprised of 2400 train and 400 test mazes, and was similarly sampled based on a 10×10 grid. Examples from those environments are given in the top row of

[4]http://www.mujoco.org/

[5]Maze generation is a classical topic in procedural content generation [42]. We used a simple depth-first search approach with back-tracking.

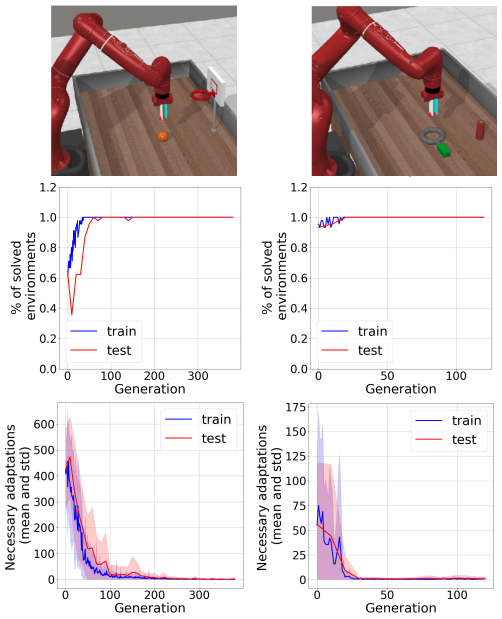

Fig. 3: Each of the two columns corresponds to an example task from metaworld. **(Top row)** Example frames from the `basketball-v2` and `assembly-v2` tasks from the benchmark. **(Middle row)** The ratio of environments among the M sampled ones that the QD instances are able to solve at each generation of FAERY. For example, on the `basketball-v2` task, we see that without optimized priors (generation 0 on the horizontal axis), only 60% of the M QD methods are successful. After about 100 meta-updates to the prior population however, we see that that all QD methods successfully solve their environments. **(Bottom row)** The average number of generations necessary to solve tasks (that are solvable at a given generation). As in the maze experiments, the horizontal axis of the plots is the number of generations for which the prior population is optimized by FAERY (*i.e.* the number of meta-updates), which should not be confused with the number of generations needed for a single QD algorithm to converge. Note that all train/test environments that are sampled at generation i on this axis will have their corresponding QD instances initialized using the priors that are obtained from generation $i-1$.

figure 2. In all mazes, the agent starts in the area in the bottom left marked with a red circle, and the goal of the environment is to reach the top left area marked with a green circle. What changes from an environment to another is the layout of the maze. In other terms, the set Υ is composed of cell borders, which constitute building blocks for walls.

The perception and actions of the agent are similar to those that were introduced in the deceptive maze example from [14]. The agent is equipped with proximity sensors which allow it to sense the presence of walls[6]. The observation space is 5-dimensional, and the action space is the $2-d$ force vector. The rewards in this environment are sparse: the agent only receives a positive reward when the goal area is reached. As a result, QD optimization in such an environment often behaves in a manner that is equivalent to pure Novelty search.

The shallow networks that we used to encode the policies had 3 hidden layers, each of dimension 10. For `mazes8x8`, we set $\lambda = \mu = 24$, and the number of sampled train mazes in each iteration of FAERY was set to $M_{train} = M_{test} = 30$. In the `mazes10x10` experiments, both population and offspring sizes where set to 40, and we had $M_{train} = M_{test} = 50$. As in the original paper that introduced the Novely metric [14], we used the final $2d$ position of the robot in a trajectory as its behavior descriptor for computing Novelty.

The results for both cases are reported in figure 2. In both experiments, we can see that initially, the M instance of QD optimization that are initialized with a random population i) Are not able to solve all randomly sampled tasks (middle row of figure 2) and ii) take many generations (in average, ~ 60 and ~ 75 for `mazes8x8` and `mazes10x10` respectively) to solve the sampled environments that they are able to solve. However, once FAERY starts learning the prior population $\mathcal{P}$, we see that the ratio of solved environments at each generation increases to 100%, and the number of required generations dramatically decreases during both training and testing, until it stabilizes at a much smaller average (< 5 for the `mazes8x8` and ~ 8 for `mazes10x10`). Note that the bottom row of figure 2 excludes the environments that have not been solved, which means that it understates the number of optimization steps that are wasted by QD optimizations that do not use the priors learned by FAERY. For example, in the very first iteration of the `mazes8x8` experiments, only about 80% of the environments are solved, which means that the remaining 20% of the QD instances run for the maximum allowed number of generations G_{QD}, which was set to 200 in this experiment.

2) Few-shot Single-Task Robotic Manipulation: The networks that we used to encode the policies for these experiments have a single hidden layer of dimension 50. The values of μ, λ and M_{test}, M_{train} were set on a per-task distribution basis, and were mainly dictated by our available computational resources at the time of the experiments. To accelerate overall QD optimization, we used the dense rewards that have been made available in recent releases of the benchmark (v2). The behavior descriptors that we used to compute the novelty objective were the final $3d$ positions of the objects of interest being manipulated: for example, in the `basketball-v2` task, this was the final position of the ball, and in the `assembly-v2` task, this was the $6d$ concatenation of the final positions of the two objects being assembled. Note that each task name such as for example `hammer-v2` designates a distribution of tasks, from which M environments can be sampled, that will differ in terms of initial conditions of objects and/or goals. For all experiments in this section, G_{QD} was set to 800.

We performed experiments on 25 task distributions chosen from the ones available in the subset of metaworld [11] dedicated to single-task generalization (the ML1 benchmark). The 25 tasks were chosen according to the following criteria and constraints: first, meta-learning experiments in general become very demanding in terms of computational resources when the problems being solved are based on physical simulations. It was thus impractical for us to consider all environments. Second, many environments in the metaworld benchmark, especially those where the task is correlated with simple (*e.g.* $1d$) behavior descriptors can readily be solved by QD methods in very few generations. While we did include some of these tasks in our experiments, we tried to focus on environments that were more challenging for QD methods. Finally, we observed that the solutions found by QD optimization in some environments such as `bin-picking-v2` and `shelf-place-v2` are very often brittle as they exploit particularities/bugs in the physics engine to achieve the goal. While reasonable, generalizable behaviors eventually emerge, they are rare enough that they make the number of generations required for QD optimization from scratch prohibitive. Therefore, we do not include them in this subsection, but will use them in the next one V-B to show the potential of FAERY in providing "stepping stones" [7] for generalization across tasks.

Figure 3 illustrates the performance of our method on two of the selected environments. As in the navigation experiments, the middle rows of this figure indicate that QD methods for which no priors are available (generation 0) are not able to solve all of the sampled environments. This effect seems to be much more

[6]More precisely, two Boolean "bumpers" that detect collisions with walls and three rangefinders which are each associated to a single direction. Taking as reference the axis pointing in the forward direction of the robot, the latter are positioned at angles $-45°$, $0°$, $45°$ and return the distance to the closest intersection point along those rays. Their range is limited to about one tenth of the environment.

task name	QD $\mathcal{R}$[1]	QD $\mathcal{G}$[2]	FAERY $\mathcal{R}$[3]	FAERY $\mathcal{G}$[4]	FAERY it_c[5]	μ[6]	λ[7]	M_{train}[8]	M_{test}[9]
assembly-v2	0.88	72.6	**1.0**	**0.68**	35	35	35	45	45
door-close-v2	1.0	1.3	1.0	**0.1**	15	35	35	45	45
window-open-v2	1.0	0.26	1.0	**0.0**	15	130	130	50	40
drawer-open-v2	1.0	9.75	1.0	**1.93**	250	35	35	45	45
sweep-into-v2	1.0	15.28	1.0	**3.6**	100	80	80	25	25
basketball-v2	0.63	449.0	**1.0**	**1.22**	275	40	40	45	45
button-press-wall-v2	1.0	3.2	1.0	**0.1**	15	35	35	45	45
door-lock-v2	1.0	30.63	1.0	**0.4**	290	40	40	45	45
handle-pull-v2	1.0	6.6	1.0	**0.62**	100	35	35	25	25
disassemble-v2	1.0	34.13	1.0	**0.35**	150	35	35	45	45
sweep-v2	1.0	32.18	1.0	**1.38**	35	130	130	50	40
box-close-v2	1.0	3.5	1.0	**0.44**	100	130	130	30	25
dial-turn-v2	1.0	1.82	1.0	**0.04**	10	35	35	45	45
hand-insert-v2	0.91	29.87	**1.0**	**2.0**	150	35	35	45	45
soccer-v2	1.0	1.2	1.0	**0.0**	10	40	40	20	20
button-press-topdown-v2	0.62	203.17	**1.0**	**0.02**	210	35	35	45	45
push-v2	1.0	9.7	1.0	**0.4**	100	100	100	10	10
stick-push-v2	1.0	24.42	1.0	**0.02**	80	40	40	45	45
window-close-v2	1.0	2.6	1.0	**0.0**	10	40	40	45	45
reach-wall-v2	1.0	1.75	1.0	**0.37**	15	40	40	45	45
plate-slide-v2	1.0	2.7	1.0	**0.0**	10	100	100	10	10
plate-slide-back-v2	1.0	123.73	1.0	**0.4**	90	40	40	45	45
plate-slide-back-side-v2	1.0	132.4	1.0	**0.02**	90	40	40	45	45
hammer-v2	0.94	81.58	**1.0**	**5.2**	120	40	40	45	45
lever-pull-v2	1.0	3.9	1.0	**0.0**	80	100	100	10	10

[1] **QD** $\mathcal{R}$: percentage of solved environments (normalized in $[0, 1]$) when using QD optimization from scratch.
[2] **QD** $\mathcal{G}$: Average number of necessary generations required to solve an environment, when using QD optimization from scratch.
[3] **FAERY** $\mathcal{R}$: percentage of solved environments (normalized in $[0, 1]$) when using QD optimization with priors learned by FAERY.
[4] **FAERY** $\mathcal{G}$: Average number of necessary generations required to solve an environment, when using QD with priors learned by FAERY.
[5] **FAERY** it_c: Number of generation at which the prior population has converged in terms of FAERY $\mathcal{R}$ and FAERY $\mathcal{G}$.
[6,7] μ, λ: respectively population size and number of offsprings.
[8,9] M_{train}, M_{test}: number of environments used for training and testing.

TABLE I: Comparison of ratio of solved environments and adaptation speed between QD optimization from scratch and QD optimization based on FAERY priors. For each task, the results are averaged on 3 different realizations of samples of size M_{test}. The entries that are highlighted using the color red are those where QD optimization from scratch either fails to solve all environments or takes > 20 generations on average to solve an environment. Values corresponding to tasks where the proposed method results in significant improvements are highlighted with bold fonts. Note that the values of μ, λ and M_{test}, M_{train} were mainly dictated by our available computational resources at the time of the experiments, and are only reported for reproducibility.

pronounced for `basketball-v2`, where a QD method initialized from scratch seems incapable of solving more than 60% of the sampled environments. However, as FAERY learns a suitable prior $\mathcal{P}$, the number of solved environments sampled from those tasks rises to a 100%. Similarly, the number of generations required for QD optimizations significantly decreases: for both of those tasks, it falls from hundreds of generations to < 3 on average.

More detailed results on all 25 environments are presented in table I. For simplicity in the tabulation of the results, we define the following symbols: $\mathcal{R}$ will denote the ratio of solved environments among the M environments that are considered at anyone time. The average number of necessary generations is noted $\mathcal{G}$. Note that in that table QD $\mathcal{R}$, QD $\mathcal{G}$ correspond to QD optimization on M test problems from scratch, and FAERY $\mathcal{R}$, FAERY $\mathcal{G}$ correspond to QD optimization using the learned priors after convergence. In the table, the generation number at which the priors have converged is given in the column titled FAERY it_c. For reproducibility, parameters of the experiments $(\lambda, \mu, M_{train}, M_{test})$ are also reported. In the table, the QD $\mathcal{R}$ and QD $\mathcal{G}$ entries of the tasks where QD optimization from scratch fails to solve all environments or takes > 20 generations on average are highlighted with the color red. The FAERY $\mathcal{R}$, FAERY $\mathcal{G}$ values of Tasks where the priors learned by FAERY result in significant improvements are written using bold fonts. As in the previously discussed examples, it can be seen that QD optimization from scratch is often unable to solve all sampled environments and requires many generations before finding a solution. Both of these aspects significantly improve once our method starts learning the prior population: at convergence, all newly sampled environments are solved in few (generally < 3) generations.

Time savings. The computational speedup when using the learned priors on a new environment results from the reduction in the number of necessary adaptations. For a maximum computational budget of G_{QD} generations and noting $time(\Delta)$ the time taken by each QD population update, an approximate lower bound to the time savings is given by $(\text{QD}\mathcal{R} \times \text{QD}\mathcal{G} + (1 - \text{QD}\mathcal{R}) \times \text{G}_{\mathcal{QD}} - \text{FAERY}_G) \times time(\Delta)$, where the notations are those that were defined in table I. In our experiments on metaworld, a single update step of the population takes on average ~ 1.8 seconds for population sizes of 35 and 40, and takes ~ 2 seconds for population sizes of 100 and above when running on 48 Intel Xeon Silver 4214 CPU cores. As a result, in tasks such as `basketball-v2`, using the priors enables adaptation in a matter of seconds where learning from scratch would take ~ 15 to 20 minutes while leaving some environments unsolved. Note however that in a few tasks such as `box-close-v2`, only a few seconds will be saved by using the priors in a new environment as QD$\mathcal{G}$ is already low. Nonetheless, even in such cases, learning and using the priors can be advantageous due to the cumulated time that will be saved on (potentially numerous) future problems, especially in applications such as continual learning or those that require online/near real-time performance on new problems.

task name	prior type	% envs solved	Mean required QD generations
shelf-place-v2	None	0.0	N.A
shelf-place-v2	basketball-v2 solvers	20%	582
shelf-place-v2	basketball-v2 FAERY	**24%**	650.25
bin-picking-v2	None	95%	130.68
bin-picking-v2	basketball-v2 solvers	93%	183
bin-picking-v2	basketball-v2 FAERY	**100%**	151
pick-place-v2	None	51%	117
pick-place-v2	basketball-v2 solvers	55%	90
pick-place-v2	basketball-v2 FAERY	**77%**	119

TABLE II: **Cross-task generalization.** Results of QD optimization (NSGA2 with Novelty and fitness) averaged over 135 environments on three different tasks with different prior types. This results seem to hint that using the priors learned by the proposed method can result in significant boosts in terms of percentage of solved environments. Note that the last column indicates the average number of generations per *solved* environment., which understates the gains in terms of time savings as QD instances that are unable to solve an environment will run for G_{QD} generations.

B. Cross-Task Generalization

The objective of this experiment is to evaluate the potential use of the priors learned by FAERY in knowledge transfer from a source task $\mathcal{T}_{source}$ to target distributions $\{\mathcal{T}^i_{target}\}_i$ in the cross-task settings discussed in the beginning of section V.

Assuming that we are given a target distribution $\mathcal{T}^i_{target}$ and an appropriate source task $\mathcal{T}_{source}$, the question of how to transfer knowledge from the latter to an agent learning on the former remains. Many approaches, such as MDP homomorphism based ones [43], [44] rely on learning explicit mappings between fixed MDPs. As our aim is transfer learning from a distribution $\mathcal{T}_{source}$ rather than from a particular (PO)MDP, learning such a mapping is impractical. Furthermore, as previously stated in section V, we are interested in cross-task transfer learning between POMDP distributions that share their action spaces, and whose state spaces have the same dimensionality. Therefore, a suitable transfer mechanism in this setting is to simply fine-tune a population of policies learned on environments from $\mathcal{T}_{source}$ on environments from $\mathcal{T}_{target}$. A way to do this would be to take a set of policies $Z = \{\theta_1, ..., \theta_\mu\}$ that solve environments $\{t_1, ..., t_\mu\}$ randomly sampled from $\mathcal{T}_{source}$. In this section, we show that using the priors $\mathcal{P}$ learned by FAERY is more advantageous than using Z.

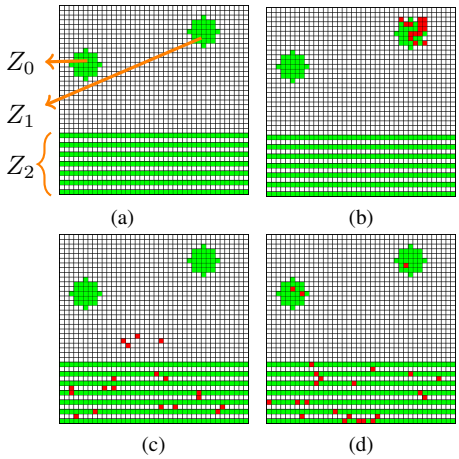

Fig. 4: **(a)** The toy environment distribution used to demonstrate the complementarity between the two fitnesses. **(b)** Results of a run in which only adaptivity (f_1) is optimized. **(c)** results of a single run where only polyvalence (f_0) is maximized. **(d)** Results when both f_0 and f_1 are jointly maximized. See section V-C for details.

The choice of source and task distributions can be guided by metric functions defined between (PO)MDPs. However, those metrics often have shortcomings. For example, distance functions based on generative models [45] are dependent on exploration, and graph-based approaches [46] are limited to finite state spaces. As a result, as in most of the current Reinforcement Learning literature, we let intuition guide our choice of source and task targets. In this experiment, we chose `basketball-v2` as the source task for transfer, and consider `shelf-place-v2`, `pick-place-v2` and `bin-picking-v2` as the target tasks[7]. The latter three environments had proven to be difficult to solve in a reasonable number of generations using QD optimization, mainly because of the high reward returned for brittle policies that exploit the particularities and bugs in the physics simulator, making the occurrence of proper grasping behaviors rare. The aims of `pick-place-v2` and `shelf-place-v2` are, as the names indicate, to pick an object and place it respectively at a given $3d$ location in space and on a shelf. The goal of `bin-picking-v2` is to pick up items from a bin and place them in another one.

The goal of the source task is to throw a ball into a hoop, and successful policies learn to grasp the ball before moving the end-effector and releasing it towards the hoop. As the three difficult discussed environments require similar behaviors, it seems likely that some of the knowledge gained from the basketball environment will be transferable to them.

The results of our experiments are reported in table II. For each of the target tasks, as indicated in the *prior type*, three experiments are considered: 1) QD optimization without priors 2) QD optimization with priors that consist of μ random solutions to μ environments from $\mathcal{T}_{source}$ and 3) QD optimization using priors learned by FAERY. In each case, the results are computed based on 135 samples from the corresponding task distribution.

Those results clearly indicate that the use of priors significantly facilitates optimization. In particular, no solutions were found for `shelf-place-v2` in a reasonable number of generations ($G_{QD} = 2000$) without using priors. While this does indeed show that `basketball-v2` is a suitable source task, we see that there is a difference in experiments that were initialized with solvers from $\mathcal{T}$ and those that were initialized with the FAERY priors. The latter seem to constantly outperform the former. However, the margin between the two range from small on `shelf-place-v2` to significant on `pick-place-v2` and `bin-picking-v2`. Note that as in previous tables and figures, the mean number of generations does not take into account environments that a method is not able to solve and for which G_{QD} generations are nonetheless wasted. It therefore understates the reduction in learning time that results from utilizing the learned priors. While more exhaustive experiments should be carried in the future, those results seem to hint at the potential of our method for generalization across different tasks.

C. Ablation Study.

We demonstrate the complementarity of the two fitnesses f_0, f_1 on a toy environment distribution whose simplicity allows us to provide further insights via visualization. Each environment from this distribution is a 40×40 grid, where a single cell (g_1, g_2) is designated as the goal, which an agent is expected to correctly guess in a single action [8]. The distribution of those environments is illustrated in figure 4(a). The green areas, noted Z_0, Z_1, Z_2 in that figure, indicate the cells that can be chosen as goal. In other words, an environment is instantiated by taking an empty 40×40 grid together with a single goal cell (g_1, g_2) chosen from the green areas. For simplicity, each agent is parameterized by the single action (i, j) it can take. This action is also used as the behavior descriptor. The experiment then consists in running FAERY with $\lambda = \mu = 25$ for 70 generations, initializing all agents randomly on the first row of the grid, and using half the cells from of each Z_i area as the training set. Mutation is realized by random increments to the agents in the left, right, up and down directions. We ran the experiment 15 times.

As shown in figure 4(b), when optimizing only for adaptivity (f_1 from eq. 4), the meta-population gets trapped in the Z_1 area without exploring the other two. This means that while the resulting population is able to quickly adapt to environments sampled from Z_1, none of its individuals are able to do so for Z_0 and Z_2. Optimizing only for polyvalence (f_0 from eq. 4), however, moves the meta-population towards the parts where the distribution of tasks has more mass, abandoning other areas. This is illustrated in figure 4(c), where the population is clearly drawn to Z_2 while ignoring the two other areas. Finally, as shown in figure 4, optimizing both f_0, f_1 jointly allows the population to be well-distributed among all three task distribution modes.

Note that figures 4 (b,c,d) illustrate the most likely outcome that we observed during the different runs, but the stochasticity of

[7]We did not use the ML10 benchmark from metaworld, since except for `place-shelf-v2`, the tasks that compose that benchmark present very little challenge for QD methods, and we preferred to select more difficult tasks.

[8]Each environment from this distribution can be seen as a multi-armed bandit problem with a horizon of 1 and binary rewards that are null everywhere except for a single action.

the mutation and selection operators (in breaking ties) can lead to slightly different results. While in all experiments, optimizing only f_1 results in a situation where the meta-population remains in Z_1 and optimizing only f_0 leaves at least one of Z_1 or Z_2 uncovered, in 13.3% of experiments, optimizing the two fitnesses jointly results in a meta-population that has individuals close to but not inside Z_0.

VI. Conclusion

We presented an algorithm that leverages previous QD experience to enable Few-Shot Quality Diversity optimization on previously unseen tasks. It does not require dense rewards nor gradients and is agnostic to the underlying QD optimization method, as well as to the models used to represent policies. Furthermore, it is easy to implement and scale. The empirical validations that were presented in single-task generalization settings indicate that our approach not only considerably reduces the number of generations required for QD optimization on a new problem, but also hint that it could be a promising direction for multi-task generalization.

References

[1] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2017.

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

[3] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: learning object manipulation skills through novelty search and local adaptation," *Robotics and Autonomous Systems*, vol. 136, p. 103710, 2021.

[4] D. M. Bossens and D. Tarapore, "Qed: using quality-environment-diversity to evolve resilient robot swarms," *IEEE Transactions on Evolutionary Computation*, 2020.

[5] M. Charity, A. Khalifa, and J. Togelius, "Baba is y'all: Collaborative mixed-initiative level design," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 542–549.

[6] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[7] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[8] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. Stanley, "Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9940–9951.

[9] G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud, "Qd-rl: Efficient mixing of quality and diversity in reinforcement learning," *arXiv preprint arXiv:2006.08505*, 2020.

[10] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[11] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1094–1100.

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

[13] 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, pp. 211–218.

[14] S. K. O. Lehman, Joel, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[15] D. Gravina, A. Liapis, and G. Yannakakis, "Surprise search: Beyond objectives and novelty," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 677–684.

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

[17] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance matrix adaptation for the rapid illumination of behavior space," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 94–102.

[18] G. Paolo, A. Coninx, S. Doncieux, and A. Laflaquière, "Sparse reward exploration via novelty search and emitters," in *The Genetic and Evolutionary Computation Conference 2021 (GECCO 2021)*, 2021.

[19] M. W. Iruthayarajan and S. Baskar, "Covariance matrix adaptation evolution strategy based design of centralized pid controller," *Expert systems with Applications*, vol. 37, no. 8, pp. 5775–5781, 2010.

[20] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, "Scaling map-elites to deep neuroevolution," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 67–75.

[21] L. Shi, S. Li, Q. Zheng, M. Yao, and G. Pan, "Efficient novelty search through deep reinforcement learning," *IEEE Access*, vol. 8, pp. 128 809–128 818, 2020.

[22] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *arXiv preprint arXiv:1708.02190*, 2017.

[23] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-efficient design exploration through surrogate-assisted illumination," *Evolutionary computation*, vol. 26, no. 3, pp. 381–410, 2018.

[24] L. Keller, D. Tanneberg, S. Stark, and J. Peters, "Model-based quality-diversity search for efficient robot learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9675–9680.

[25] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[26] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International conference on machine learning*. PMLR, 2015, pp. 2152–2161.

[27] N. Dong and E. P. Xing, "Few-shot semantic segmentation with prototype learning." in *BMVC*, vol. 3, no. 4, 2018.

[28] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, pp. 3630–3638, 2016.

[29] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[30] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[31] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1842–1850.

[32] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl ^{2}: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[33] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "Es-maml: Simple hessian-free meta learning," *arXiv preprint arXiv:1910.01215*, 2019.

[34] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," *arXiv preprint arXiv:1805.09801*, 2018.

[35] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[36] T. Schaul and J. Schmidhuber, "Metalearning," *Scholarpedia*, vol. 5, no. 6, p. 4650, 2010, revision #91489.

[37] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[38] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[39] J. Oh, M. Hessel, W. M. Czarnecki, Z. Xu, H. P. van Hasselt, S. Singh, and D. Silver, "Discovering reinforcement learning algorithms," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[41] J. Gomes, P. Mariano, and A. L. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 943–950.

[42] B. De Kegel and M. Haahr, "Procedural puzzle generation: a survey," *IEEE Transactions on Games*, vol. 12, no. 1, pp. 21–40, 2019.

[43] P. Castro and D. Precup, "Using bisimulation for policy transfer in mdps," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010.

[44] J. Sorg and S. Singh, "Transfer via soft homomorphisms," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2009, pp. 741–748.

[45] H. B. Ammar, E. Eaton, M. E. Taylor, D. C. Mocanu, K. Driessens, G. Weiss, and K. Tuyls, "An automated measure of mdp similarity for transfer in reinforcement learning," in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[46] J. Song, Y. Gao, H. Wang, and B. An, "Measuring the distance between finite markov decision processes," in *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, 2016, pp. 468–476.