Sparse Reward Exploration via Novelty Search and Emitters

Giuseppe Paolo, Alexandre Coninx, Stéphane Doncieux, Alban Laflaquière

Sparse Reward Exploration via Novelty Search and Emitters

Giuseppe Paolo, Alexandre Coninx, Stéphane Doncieux, Alban Laflaquière

Sparse Reward Exploration via Novelty Search and Emitters

Giuseppe Paolo
AI Lab, SoftBank Robotics Europe
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR
Paris, France
giuseppe.paolo@softbankrobotics.com

Alexandre Coninx
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR
Paris, France
alexandre.coninx@sorbonne-universite.fr

Stephane Doncieux
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR
Paris, France
stephane.doncieux@sorbonne-universite.fr

Alban Laflaquière
AI Lab, SoftBank Robotics Europe
Paris, France
alaflaquiere@softbankrobotics.com

ABSTRACT

Reward-based optimization algorithms require both exploration, to find rewards, and exploitation, to maximize performance. The need for efficient exploration is even more significant in sparse reward settings, in which performance feedback is given sparingly, thus rendering it unsuitable for guiding the search process. In this work, we introduce the SparsE Reward Exploration via Novelty and Emitters (SERENE) algorithm, capable of efficiently exploring a search space, as well as optimizing rewards found in potentially disparate areas. Contrary to existing emitters-based approaches, SERENE separates the search space exploration and reward exploitation into two alternating processes. The first process performs exploration through Novelty Search, a divergent search algorithm. The second one exploits discovered reward areas through emitters, i.e. local instances of population-based optimization algorithms. A meta-scheduler allocates a global computational budget by alternating between the two processes, ensuring the discovery and efficient exploitation of disjoint reward areas. SERENE returns both a collection of diverse solutions covering the search space and a collection of high-performing solutions for each distinct reward area. We evaluate SERENE on various sparse reward environments and show it compares favorably to existing baselines.

KEYWORDS

Novelty search, sparse rewards, emitters, evolutionary algorithm, quality diversity

ACM Reference Format:
Giuseppe Paolo, Alexandre Coninx, Stephane Doncieux, and Alban Laflaquière. 2021. Sparse Reward Exploration via Novelty Search and Emitters. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3449639.3459314

P

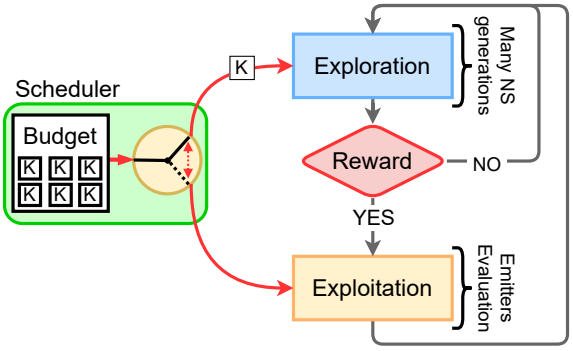

Figure 1: SERENE consists of two exploration and exploitation processes, controlled by a scheduler. The exploration process searches for novel solutions through Novelty Search. The exploitation process uses emitters to optimize the rewards discovered during exploration. The scheduler alternates between the two processes by splitting the total evaluation budget into chunks of size K to assign to either of them.

1 INTRODUCTION

Embodied agents solve tasks by learning a *policy* dictating how to act in different situations. This is done by evaluating the agent's performance on the task through a *reward function*. Learning strategies for such agents can be divided in two groups: step-based and episode-based [33]. The former expects a reward after each step. On the contrary, episode-based ones need rewards only at the end of each training episode. So much reliance on the reward forces some constraints: the reward function must be well designed and provide feedback as frequently as possible. In many complex scenarios where the reward is given only if specific conditions are met, such constraints are impossible to respect. These are known as *sparse reward* situations and can prove very difficult to tackle. In this work, we consider sparse reward settings in which the reward is obtained only in small disjoint areas of the whole search space. One example would be a robotic arm trying to push an object to one of a few given positions. The search space consists of all the positions the object can achieve, while the reward is given only if the object reaches one

of the goals. In such situations, a standard Reinforcement Learning (RL) [35] agent typically explores by trying random actions. The probability of finding a reward this way tends to zero, rendering learning impractical. Therefore, the way *exploration* is performed is fundamental when dealing with sparse rewards settings.

In recent years, many algorithms have been proposed to solve this problem [5, 7, 15, 24, 36]. Among them, Novelty Search (NS) is an evolutionary algorithm that focuses only on exploration, while ignoring any possible reward [24]. By doing so, NS tends towards a uniform exploration of the search space [14], avoiding the need for a well-defined reward function. At the same time, its strength is also its limitation: considering all the non-rewarding areas as valuable as the rewarding ones prevents the algorithm from finding the best possible solutions. Augmenting NS with the ability to shift its focus from pure exploration to reward exploitation could help address this issue. One possible way of doing so is by using multi-objective optimization methods like NSGA-II [12]. However, merging exploration and exploitation through a Pareto front can degrade the exploring power of the algorithm. A different approach is taken by Quality-Diversity (QD) algorithms, a family of methods that build a set of both diverse and high-quality solutions [32].

In this work, we introduce SparsE Reward Exploration via Novelty search and Emitters (SERENE), a QD algorithm addressing sparse reward problems. SERENE augments NS with *emitters* [17] to perform rewards maximization while keeping its exploration ability, thanks to a clear separation between the exploration and exploitation. Introduced as a way to improve the efficiency of MAP-Elites (ME) [27] in the CMA-ME method [17], *emitters* are instances of reward-based evolutionary algorithms scheduled to perform a local search in the search space. In the original formulation, ME acts as a scheduler by initializing emitters in different areas of the search space. The emitters then perform both local exploration and exploitation of the reward, leading to degraded performances in settings with very sparse rewards, where not all policies can obtain a reward. Conversely, SERENE decouples exploration from exploitation to better deal with such situations. The former is performed through NS, completely ignoring the reward. Once a reward area is found, SERENE spawns emitters focusing solely on its maximization, with no dependency on the exploration process. This allows our algorithm to shift its focus between exploration and exploitation at any moment. Persisting in exploring even after some reward areas have been found is essential, since other reward areas could be present in the search space.

In the following, we will discuss other works tackling the sparse rewards problem in Section 2. In Section 3 we will analyze the methods SERENE draws from and explain in detail the concept of *emitter*. The method itself will be introduced in Section 4, tested in Section 5, and the results discussed in Section 6. We will conclude with Section 7 by pointing at possible extensions and improvements.

2 RELATED WORK

2.1 Sparse rewards

Step-based algorithms expect a reward at every step, making dealing with sparse reward particularly difficult; this is the case for many RL algorithms. Following the recently increased interest in the problem, many new approaches have been proposed to deal with this sparsity. Some methods work on improving the data efficiency of the search [1, 28]. Others introduce some artificial curiosity by counting the number of times a state is visited, and push exploration by making less-visited states more rewarding [2, 36]. Another strategy uses additional shaped rewards to aid in approaching the task [37]. A population of RL agents can also be used to increase exploration while learning a policy [13, 22, 30]. However, none of these methods explicitly separates exploration and exploitation.

Episode-based methods, and more specifically evolutionary algorithms [38], are better suited for dealing with sparse reward settings, given the more relaxed dependency on the reward. For this reason, many works combined evolutionary algorithms with RL. Some works use Evolutionary Strategy (ES) to collect the data over which a RL agent is then trained [23, 31]. These approaches take advantage of the exploration of evolution-based methods and the higher data efficiency of RL.

Separating exploration from exploitation has proven useful for overcoming deceptive gradients in sparse reward settings [6, 7, 15]. In the work from Colas et al. [7], a reward-agnostic exploration phase is first performed through Goal Exploration Processes [18]; then a RL based policy is learned on the collected data. A similar two-step process is used in GO-Explore [15] to solve ATARI games. Conversely, QD-RL [6] separates exploration and exploitation by taking advantage of a QD population trained through an actor-critic approach. Half of the population is optimized for quality, while the other half is optimized for diversity.

2.2 Divergent search algorithms

Divergent search methods, as the one used by Cideron et al. [6], generate solutions by looking for a set of diverse policies. This prevents getting stuck in local optima that could limit the performance of the solutions. One of the first algorithms developed in this direction is NS [24]. Since, many divergent search algorithms have been developed, using different mechanisms to drive the search: curiosity [34], empowerment [5], surprise [20], diversity [10, 11, 16, 32], and novelty [25].

QD [11, 32] is a family of divergent search algorithms that searches for a set of diverse solutions while also improving on their quality. A well-known QD algorithm is ME [27], a method that drives the search for novel policies by discretizing the search space into a grid and filling its cells with high-performing solutions.

QD algorithms have been extended by combining them with ES [3] to increase their efficiency and speed of convergence [8, 9, 17]. Conti et al. [8] augment an ES with NS's novelty objective to look for novel solutions while improving their performances. At the same time, the approach followed by Fontaine et al. [17], and then extended by Cully [9], uses ME as a scheduler for modified instances of CMA-ES [21], named *emitters*. Exploration of the search space and reward exploitation are both performed through emitters. However, fusing the two aspects can limit performances in sparse reward settings where reward-based algorithms struggle to explore.

In this work, we take inspiration from CMA-ME [17] by combining emitters with NS to keep the two aspects, i.e. exploration and exploitation, separated. This allows our method to avoid the shortcomings of exploring through emitters. In the next section, we

describe in detail how both NS and emitters work before detailing the functioning of SERENE.

3 BACKGROUND

The notation used in this work is based on the one introduced by Doncieux et al. [14] and is directly inspired by the RL literature.

3.1 Novelty Search

NS is an evolutionary algorithm that replaces the usual fitness metrics used by evolutionary algorithms with a *novelty* metric. This metric pushes the search towards novel areas of the search space. The novelty is calculated in a hand-defined *behavior space* $\mathcal{B}$ in which the behavior of each policy $\theta_i \in \Theta$ is represented. When a policy is evaluated, it traverses a sequence of states $\tau = [s_0, \cdots, s_T]$, where the initial state s_0 is constant for every policy. Traversed states are observed through some sensors generating a sequence of observations $\tau_{\mathcal{O}} = [o_0, \cdots, o_T]$, with $o_t \in \mathcal{O}$. From the sequence of observations it is possible to extract a representation $b_i \in \mathcal{B}$ of the policy's behavior by using an observer function $O_{\mathcal{B}} : \mathcal{O} \to \mathcal{B}$. This whole process can be summarized by introducing a *behavior function* directly mapping a policy θ_i to its behavior descriptor b_i:

$$\phi(\theta_i) = b_i. \tag{1}$$

Once computed, the behavior descriptors are used to calculate the policies' novelty as:

$$\eta(\theta_i) = \frac{1}{|J|} \sum_{j \in J} \text{dist}(b_i, b_j) = \frac{1}{|J|} \sum_{j \in J} \text{dist}\big(\phi(\theta_i), \phi(\theta_j)\big), \tag{2}$$

where J is the set of indexes of the k policies closest to θ_i in the behavior space.

The novelty of the policies is calculated at each generation and used to choose the policies for the next generation. Moreover, N_Q policies are sampled to be stored into an *archive*, returned as outcome of the algorithm. This archive is also used to keep track of the already explored areas of the space $\mathcal{B}$. This is done by choosing the $|J|$ closest neighbors used in equation (2) not only from the current population and offspring but also from the archive. By choosing the most novel policies from the previous generation to compose the population, the search is always pushed towards less explored areas of $\mathcal{B}$. Notwithstanding its capacity for exploration, NS cannot exploit the rewards potentially found during the search. This can lead to low rewarding solutions.

3.2 Emitters

An emitter [9, 17] is an instance of a reward-based Evolutionary algorithm (EA), such as CMA-ES [21]. Its objective is to rapidly examine a small area of the search space while optimizing on the reward. The CMA-ME algorithm [9, 17] combines emitters with ME [27], by using the latter as a scheduler for the emitters evaluation. It works by initializing a population of policies θ by sampling their parameters from a distribution $\mathcal{N}(\mu, \Sigma)$ and adding them to the ME archive. The algorithm then samples one of these policies and uses it to initialize the population of the emitter $\mathcal{E}_i$. At this point, $\mathcal{E}_i$ is evaluated until a termination criterion is met; e.g. a lack of increase of the reward found. Moreover, the policies found during the evaluation of the emitter are added to the ME archive according to ME addition strategy. After the termination of $\mathcal{E}_i$, a new emitter is initialized by sampling another policy from the archive. This is repeated until the whole evaluation budget is depleted.

Different types of algorithms can be used as emitters, changing how the search is performed and how the policies are selected. This shows the flexibility of the approach. At the same time, previous works [9, 17] perform exploration through reward-following emitters. This reduces performances in situations where the reward is very sparse and many of the policies do not get any reward.Decoupling the exploitation of the reward from the exploration allows to more efficiently deal with sparse rewards settings [7].

4 METHOD

SERENE disentangles the exploration of the behavior space $\mathcal{B}$ from the exploitation of the reward through a two-steps process. In the first phase, called *exploration phase*, $\mathcal{B}$ is explored by performing NS. As per equation (1), the policies θ_i found during exploration are assigned a behavior descriptor $\phi(\theta_i)$. A policy obtaining a reward means that its $\phi(\theta_i)$ belongs to the subspace of rewarding behaviors $\mathcal{B}_{\text{Rew}} \subseteq \mathcal{B}$. It is in this subspace that the exploitation of the reward happens. This is done in the second phase, called *exploitation phase*, in which emitters are initialized using the rewarding policies found in $\mathcal{B}_{\text{Rew}}$ during exploration. During the exploitation phase the most rewarding policies are stored to be returned as result of the algorithm. Moreover, particularly novel policies found by the emitters are also stored. By launching emitters only in the neighborhoods of the reward areas, SERENE keeps the exploitation of the reward separated from the exploration of the search space. This results in taking the best of both worlds: the exploration power of NS and the focused exploitation of reward-based algorithms.

The exploitation and exploration phases are alternated repeatedly through a meta-scheduler. This scheduler divides a total evaluation budget *Bud* in smaller chunks of size K_{Bud} and assigns them to either one of the two phases. The whole process is illustrated in Figure 1 and described in Algorithm 1.

Algorithm 1: SERENE

INPUT: evaluation budget *Bud*, budget chunk size K_{Bud}, population size M, emitter population size $M_{\mathcal{E}}$, offspring per policy m, mutation parameter σ, number of policies added to novelty archive N_Q;
RESULT: Novelty archive $\mathcal{A}_{\text{Nov}}$, rewarding archive $\mathcal{A}_{\text{Rew}}$;
$\mathcal{A}_{\text{Nov}} = \emptyset$; $\mathcal{A}_{\text{Rew}} = \emptyset$;
$\mathcal{Q}_{\text{Em}} = \emptyset$; $\mathcal{Q}_{\text{Cand\_Nov}} = \emptyset$; $\mathcal{Q}_{\text{Cand\_Em}} = \emptyset$;
Sample population Γ_0;
Split *Bud* in chunks of size K_{Bud};
while *Bud not depleted* **do**
  if Γ_0 **then**
    Evaluate $\theta_i, \forall \theta_i \in \Gamma_0$;
    Calculate $b_i = \phi(\theta_i) \in \mathcal{B}, \forall \theta_i \in \Gamma_0$;
  ExplorationPhase (K_{Bud}, m, σ, $\mathcal{A}_{\text{Nov}}$, $\mathcal{Q}_{\text{Cand\_Em}}$, Γ_g, N_Q);
  if ***not*** $\mathcal{Q}_{Cand\_Em} == \emptyset$ ***or not*** $\mathcal{Q}_{Em} == \emptyset$ **then**
    ExploitationPhase (K_{Bud}, $\mathcal{Q}_{\text{Cand\_Em}}$, λ, m, $\mathcal{Q}_{\text{Em}}$, $\mathcal{A}_{\text{Nov}}$, $\mathcal{A}_{\text{Rew}}$, $M_{\mathcal{E}}$);

To keep track of policies generated during the different phases, SERENE uses the following buffers and containers:

- *novelty archive* $\mathcal{A}_{\text{Nov}}$: a repertoire of the novel policies found during the *exploration phase*, and returned as first output of SERENE;
- *reward archive* $\mathcal{A}_{\text{Rew}}$: a repertoire of rewarding policies found during the *exploitation phase*, returned as second output of SERENE;
- *candidates emitter buffer* $\mathcal{Q}_{\text{Cand\_Em}}$: a buffer containing the rewarding policies $\phi(\theta_i) \in \mathcal{B}_{\text{Rew}}$ found during the *exploration phase* and used in the *exploitation phase* to initialize emitters;
- *emitter buffer* $\mathcal{Q}_{\text{Em}}$: a buffer containing all the initialized emitters to be evaluated during the *exploitation phase*;
- *novelty candidates buffer* $\mathcal{Q}_{\text{Cand\_Nov}}$: a buffer containing the most novel policies found by the emitter. Each emitter has its own instance of this buffer and the policies in it are sampled for addition to the novelty archive $\mathcal{A}_{\text{Nov}}$ once the emitter is terminated.

A high-level overview of how these sets interact during the two phases is given in Figure 2, and a more detailed description is proposed in the two following subsections.

Exploration phase

SERENE starts by generating an initial population Γ_0 of size M. This is done by sampling the parameters of the population's policies θ_j from a normal distribution $\mathcal{N}(0, I)$. The population is used to explore the behavior space $\mathcal{B}$ through NS. At each generation g, a mutation operator generates m new policies θ_j^i (offspring) from each of the policies $\theta_j \in \Gamma_g$:

$$\forall j, i \in \{1, \dots, M\} \times \{1, \dots, m\}, \theta_j^i = \theta_j + \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, \sigma I). \quad (3)$$

The resulting offspring population Γ_g^m, of size $m \times M$, is then evaluated to obtain the behavior descriptors $\phi(\theta_j^i) = b_j^i \in \mathcal{B}$. The novelty of Γ_g and Γ_g^m is then calculated using equation (2) and is used to generate the next generation population Γ_{g+1} by taking the most novel policies from the current population and the offsprings. At the same time, N_Q policies among the offsprings are uniformly sampled to be added to the *novelty archive* $\mathcal{A}_{\text{Nov}}$. Finally, all the rewarding policies found are stored in the *candidates emitters buffer* $\mathcal{Q}_{\text{Cand\_Em}}$. The process just described is detailed in Algorithm 2.

The exploration phase is executed for the K_{Bud} evaluation steps in the given budget chunk, where each evaluation step corresponds to one policy evaluation. Once the chunk is depleted, the scheduler assigns the next chunk to the *exploitation phase* only if $\mathcal{Q}_{\text{Cand\_Em}} \neq \emptyset$. On the contrary, another *exploration phase* is performed. This means that in the worst case scenario where no reward can be discovered, i.e. $\mathcal{B}_{\text{Rew}} = \emptyset$, SERENE performs exactly like NS.

Exploitation phase

The *exploitation phase* consists of two sub-steps: the *bootstrapping step*, in which the policies in the candidates emitter buffer $\mathcal{Q}_{\text{Cand\_Em}}$ are used to initialize and bootstrap emitters, and the *emitter step*, in which the initialized emitters are evaluated.

Algorithm 2: Exploration Phase

INPUT: budget chunk K_{Bud}, number of offspring per parent m, mutation parameter σ, novelty archive $\mathcal{A}_{\text{Nov}}$, candidate emitters buffer $\mathcal{Q}_{\text{Cand\_Em}}$, population Γ_g, number of policies N_Q;

while K_{Bud} *not depleted* **do**

- Generate offspring Γ_g^m from population Γ_g;
- Evaluate $\theta_i,\ \forall \theta_i \in \Gamma_g^m$;
- Calculate $b_i = \phi(\theta_i) \in \mathcal{B},\ \forall \theta_i \in \Gamma_g^m$;
- Calculate $\eta(\theta_i) = \frac{1}{|J|} \sum_{j \in J} \text{dist}(b_i, b_j),\ \forall \theta_i \in \Gamma_g^m \bigcup \Gamma_g$;
- $\mathcal{A}_{\text{Nov}} \leftarrow N_Q$ samples from Γ_g^m;
- **if** $\phi(\theta_i) \in \mathcal{B}_{Rew}$ **then**
  - $\mathcal{Q}_{\text{Cand\_Em}} \leftarrow \theta_i$
- Generate Γ_{g+1} from most novel $\theta_i \in \Gamma_g^m \bigcup \Gamma_g$;

Bootstrap step. During this step, emitters are initialized from the rewarding policies θ_i in the candidates emitter buffer, and their potential for reward improvement evaluated. This insures that only emitters capable of improving the rewards are considered for full evaluation, reducing wasted evaluation budget. The policies used to initialize the emitters are selected according to their novelty with respect to the reward archive $\mathcal{A}_{\text{Rew}}$. This enables SERENE to focus on less explored areas of the rewarding behavior space $\mathcal{B}_{\text{Rew}}$. The whole bootstrapping phase lasts $K_{Bud}/3$ evaluations.

As discussed in Section 3.2, an emitter is an instance of a *reward-based EA*. Contrary to previous work [9, 17], in this work we do not use estimation-of-distribution algorithms like CMA-ES [21] because the estimation of the covariance matrix Σ is unreliable when the population size is smaller than the dimension of the parameter space Θ. CMA-ES circumvents the issue by using information from previous generations to calculate Σ. While stabilizing Σ, this also leads to a less efficient use of the evaluation budget. Hence, in this work we use as emitter an *elitist evolutionary algorithm* that does not require any estimation of distribution. Conversely, it composes its population with the most rewarding policies from the previous generation's population and offspring, while the offspring are generated according to equation 3.

An emitter $\mathcal{E}_i$ based on this algorithm consists of: a population P containing $M_{\mathcal{E}}$ policies $\tilde{\theta} \in \Theta$; a population of offspring P^m of size $m \times M_{\mathcal{E}}$; a generation counter γ; a tracker for the maximum reward found so far R_γ; an improvement measure $I(\cdot)$; a novelty measure η_i equal to the novelty of the policy used to initialize the emitter; and a *novelty candidate buffer* $\mathcal{Q}_{\text{Cand\_Nov}}$. The emitter $\mathcal{E}_i$ is initialized from a policy θ_i in the candidates emitter buffer by sampling its initial population P_0 from the distribution $\mathcal{N}(\theta_i, \sigma_i I)$. To keep the emitter's exploration local and prevent overlapping with the search space of possible nearby emitters, we initialize σ_i as:

$$\sigma_i = \frac{\min_j \left(\text{dist}(\theta_i, \theta_j)\right)}{3},\ \forall \theta_j \in \Gamma_g^m \cup \Gamma_g. \quad (4)$$

This shapes $\mathcal{N}(\theta_i, \sigma_i I)$ such that all other θ_j are at least 3 standard deviation away from its center. Once $\mathcal{E}_i$ has been initialized, its potential is evaluated by running it for λ generations and calculating its *emitter improvement* $I(\mathcal{E}_i)$. This improvement is defined as the difference between the average rewards obtained during the most

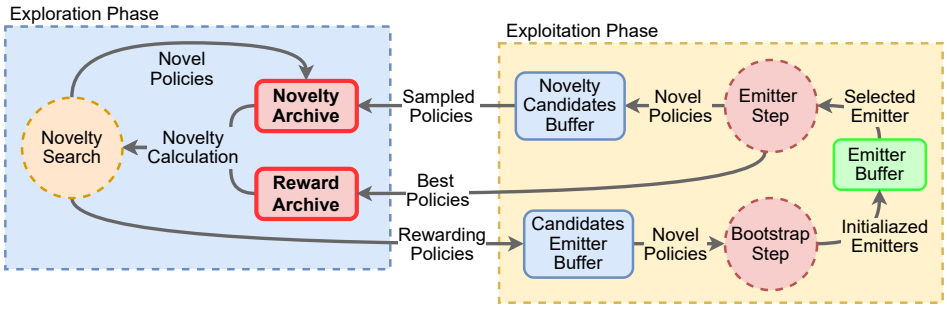

Figure 2: Overview of the sets used by SERENE to keep track of the explored areas and the initialized emitters. Highlighted in red are the two archives returned as final result of the algorithm execution.

recent and the initial generations of the emitter:

$$I(\mathcal{E}_i) = \frac{1}{\lambda M_{\mathcal{E}}} \left(\sum_{\gamma=T-\lambda/2}^{T} \sum_{j=0}^{M_{\mathcal{E}}} r_{(\gamma,j)} - \sum_{\gamma=\gamma_0}^{\lambda/2} \sum_{j=0}^{M_{\mathcal{E}}} r_{(\gamma,j)} \right). \quad (5)$$

Here T is the last evaluated generation, $r_{(\gamma,j)}$ is the reward of policy $\tilde{\theta}_j \in P_\gamma$, and γ_0 is the generation at which the emitter is at the beginning of the exploitation phase; it is always $\gamma_0 = 0$ for an emitter in the bootstrap step. If $I(\mathcal{E}_i) \leq 0$, the chances for the emitter to find better solutions than the initial ones are low, so it is not worth allotting more budget to its evaluation. On the contrary, $I(\mathcal{E}_i) > 0$ means that the emitter has high potential for improvement. Thus all the initialized emitters for which $I(\mathcal{E}_i) > 0$ are added to the *emitter buffer* $\mathcal{Q}_{\text{Em}}$ for further evaluation.

Emitter step. The initialized emitters in the *emitter buffer* $\mathcal{Q}_{\text{Em}}$ are run during this step. It starts by calculating the pareto front between the improvement $I(\mathcal{E}_i)$ and the novelty $\eta(\mathcal{E}_i)$ of each of the emitters $\mathcal{E}_i$ in the emitter buffer. The emitter to run is then sampled from the front of the *non-dominated* emitters. Using both the novelty and the fitness to select which emitter to run allows SERENE to focus both on the less explored and most promising areas of $\mathcal{B}_{\text{Rew}}$.

The policies $\tilde{\theta}_j$ generated by an emitter can be stored either for the reward they achieve or for their novelty. At every generation γ all the policies $\tilde{\theta}_j$ in the current population with a reward $r(\tilde{\theta}_j) > R_{\gamma-1}$ are added to the reward archive $\mathcal{A}_{\text{Rew}}$. Additionally, the policies $\tilde{\theta}_j$ with a novelty higher than the emitter novelty η_i are stored into the emitter's *novelty candidates buffer* $\mathcal{Q}_{\text{Cand\_Nov}}$.

The emitter $\mathcal{E}_i$ is run until either the given budget chunk is depleted or a termination condition is met. In the first case, SERENE recalculates $I(\mathcal{E}_i)$ from the beginning of the *emitter phase* and assigns the next budget chunk to the *exploration phase*. On the contrary, if a termination condition is met, $\mathcal{E}_i$ is discarded and another emitter to evaluate is sampled from the Pareto front. There can be multiple termination conditions. The one used in this work is inspired from the *stagnation criterion* [21], stopping the emitter when there is no more improvement on the reward. A detailed definition of the termination condition is presented in Appendix C. Before starting the new emitter evaluation, N_Q policies from the terminated emitter's *novelty candidates buffer* are uniformly sampled to be added to $\mathcal{A}_{\text{Nov}}$. In addition to saving particularly novel solutions as part of the final result, this prevents the exploration phase from re-exploring areas covered by emitters during the exploitation phase.

The whole *exploitation phase* is detailed in Algorithm 3.

The code repository is available at: github.com/GPaolo/SERENE.

5 EXPERIMENTS

In this section we want to verify if SERENE can efficiently deal with sparse reward settings, find all disjoint reward areas, and optimize the reward in each of them. For the evaluation, we consider the four sparse rewards environment illustrated in Figure 3:

Curling: A two Degrees of Freedom (DoF) robotic arm controlled by a 3 layers Neural Network (NN) with each layer of size 5. The arm has to push the blue ball into one of the two goal areas shown in orange and green. A reward is provided only if the ball stops in one of the two areas. The controller takes as input a 6-dimensional vector containing the ball pose (x, y), and the two joints angles and velocities. The output of the controller is the speed of each joint at the next timestep. The size of the parameter space Θ is 94, and each policy is run in the environment for 500 timesteps.

Hardmaze: Introduced in the original NS paper [24], it consists of a two-wheeled robot, in blue, whose task is to navigate the maze and reach either one of the green and orange areas. Contrary to the original formulation, in which only a single binary-reward area was present, here the reward areas are two and provide continuous rewards. At the same time, the reward is only given if the robot stops in one of the two areas. The robot is controlled by a 2-layers NN with each layer of size 5. The controller takes as input the reading of the 5 distance sensors mounted on the robot; shown in red in Figure 3. Its output is the 2-dimensional vector containing the speed of the 2 wheels at the next timestep. The size of the parameter space Θ is 63, and each policy is run in the environment for 2000 timesteps.

Redundant arm: A 20-DoF robotic arm [26] in which the arm's end-effector has to reach one of the 3 colored goal areas. The arm is controller by a NN with 2 layers of size 5. The controller takes as input the 20-dimensional vector of each joint's position, and outputs the 20-dimensional joint's torque vector. The size of the parameter space Θ is 228, and each policy is run in the environment for 100 timesteps.

Robotic ant maze: Introduced by Cideron et al. [6], it consists in a 4-legged robotic ant in a maze. There are two goal areas and the task is for the ant to navigate the maze and reach the center of one of

Algorithm 3: Exploitation Phase

INPUT: budget chunk K_{Bud}, candidate emitters buffer $\mathcal{Q}_{\text{Cand\_Em}}$, number of bootstrap generations λ, emitter population size $M_{\mathcal{E}}$, number of offspring per policy m, emitters buffer $\mathcal{Q}_{\text{Em}}$, rewarding archive $\mathcal{A}_{\text{Rew}}$, novelty archive $\mathcal{A}_{\text{Nov}}$;

`*/Bootstrap step/*`

while $K_{Bud}/3$ *not depleted* **do**

- Select most novel policy θ_i from $\mathcal{Q}_{\text{Cand\_Em}}$;
- Calculate σ_i;
- Initialize: $\mathcal{E}_i$, $\mathcal{Q}^i_{\text{Cand\_Nov}} = \emptyset$, and P_0;
- **for** $\gamma \in \{0, \dots, \lambda\}$ **do**
  - **if** P_0 **then**
    - Evaluate $\tilde{\theta}_j, \forall \tilde{\theta}_j \in P_0$;
  - Generate offspring population P^m_γ from P_γ;
  - Evaluate $\tilde{\theta}_j, \forall \tilde{\theta}_j \in P^m_\gamma$;
  - Generate $P_{\gamma+1}$ from best $\tilde{\theta}_j \in P^m_\gamma \bigcup P_\gamma$;
- Calculate $I(\mathcal{E}_i)$;
- **if** $I(\mathcal{E}_i) > 0$ **then**
  - $\mathcal{Q}_{\text{Em}} \leftarrow \mathcal{E}_i$;

`*/Emitters step/*`

Calculate pareto fronts in $\mathcal{Q}_{\text{Em}}$;

while $2/3 K_{Bud}$ *not depleted* **do**

- Sample $\mathcal{E}_i$ from *non-dominated emitters* in $\mathcal{Q}_{\text{Em}}$;
- **while** ***not*** $terminate(\mathcal{E}_i)$ **do**
  - Generate offspring population P^m_γ from P_γ;
  - Evaluate $\tilde{\theta}_j, \forall \tilde{\theta}_j \in P^m_\gamma$;
  - $\mathcal{A}_{\text{Rew}} \leftarrow \tilde{\theta}_j,\ \forall \tilde{\theta}_j \in P^m_\gamma \mid r(\tilde{\theta}_j) > R_\gamma$;
  - $\mathcal{Q}^i_{\text{Cand\_Nov}} \leftarrow \tilde{\theta}_j,\ \forall \tilde{\theta}_j \in P^m_g \mid \eta(\tilde{\theta}_j) > \eta_i$;
  - Generate $P_{\gamma+1}$ from best $\tilde{\theta}_j \in P^m_\gamma \bigcup P_\gamma$;
  - Update $I(\mathcal{E}_i)$ and R_γ;
  - **if** $terminate(\mathcal{E}_i)$ **then**
    - $\mathcal{A}_{\text{Nov}} \leftarrow N_Q$ samples from $\mathcal{Q}^i_{\text{Cand\_Nov}}$;
    - Discard emitter $\mathcal{E}_i$;

them. The robot is controlled by a 3-layers NN, with each layer of size 10. The input of the controller is the 29-dimensional observation returned by the environment at each step, while its output is the 8-dimensional joint's torque control. The size of the parameter space Θ is 574, and each policy is run in the environment for 3000 timesteps.

For all environments, the reward is given only if inside the reward area, and as a continual value in the $[0, 1]$ range. The reward varies with the distance to the center of the area and is highest directly at the center.

It can be expressed as:

$$r(\theta) = \begin{cases} 0, & \text{if } d_r > radius \\ \frac{radius - d_r}{radius}, & \text{if } d_r \leq radius \end{cases}$$

where d_r is the distance from the center and $radius$ is the radius of the reward area.

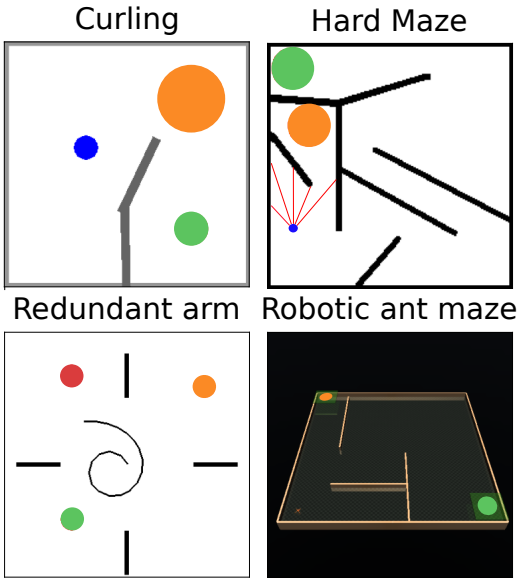

Figure 3: Testing environments: Curling, HardMaze, Redundant arm, Robotic ant maze.

Baselines

We compare SERENE against 5 different baselines:

- **NS**[24]: vanilla NS, that performs pure exploration and does not attempt to improve on the reward;
- **NSGA-II**[12]: a multi-objective evolutionary algorithm optimizing both the novelty and the reward;
- **CMA-ME**[17]: the original algorithm introducing emitters that combines ME with emitters over a 50×50 grid covering the behavior space of all environments. Among the various emitters proposed by the authors we selected the "optimizing" emitter;
- **ME**[27]: vanilla MAP-Elites that uses a 50×50 grid to cover the behavior space of every environment;
- **RND**: pure random search in which no selection happens, and every policy is sampled from a normal distribution $\mathcal{N}(0, I)$.

The parameters used during the experiments are listed in Appendix B. The statistical results are computed over 15 runs for each experiment.

6 RESULTS

This section discusses the results obtained during the experiments.

6.1 Budgeting

Balancing the exploration of the search space and the exploitation of the reward is an aspect of paramount importance for reward-based algorithms. Even more so in sparse reward environments. This balance can be studied by analyzing the amount of evaluation budget dedicated to either one of the two aspects. The exploration budget consists of all the evaluated policies that did not get any reward. On the contrary, the exploitation budget is obtained by counting all the evaluated policies that collected some reward from one of the reward areas.

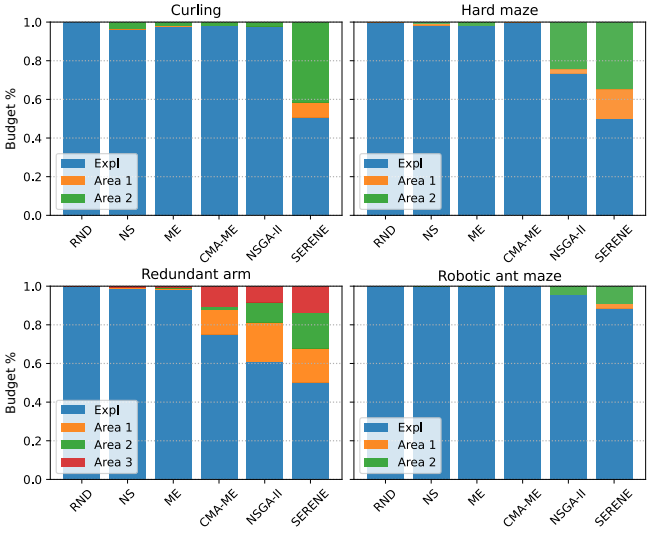

Figure 4: Average budget percentage between the exploration of the search space (in blue) and the exploitation of each reward areas (other colors).

As Figure 4 shows, SERENE has a more balanced budget split between exploration (in blue) and exploitation (other colors) compared to the other baselines. In situations in which exploration is harder, a bigger part of the budget is assigned to exploration rather than exploitation of the reward. This is the case for the robotic ant maze environment. Additionally, due to the way emitters are selected, the algorithm can shift its exploitation focus among the different reward areas. Figure 4 shows that most of SERENE's exploitation budget is assigned to the green reward area in the Curling, Hard maze and Robotic ant maze environments. As it can be seen in Figure 3, this area is more difficult to discover and to reach with respect to the orange area. This makes the exploitation of the orange reward area faster, having both the novelty and the improvement go to zero rapidly. On the contrary, being the green area harder to reach, its novelty will remain higher for longer, making SERENE select more emitters focused on it. The effect can also be seen in Figure 6, where the reward for area 1 quickly reaches higher values compared to the one of reward area 2. At the same time, in the Redundant arm environment where the 3 reward areas are equally easy to discover and to reach, this effect is less present and the exploitation budget is more evenly split between them. The ability to switch its focus is similar to intrinsic motivation based methods [4, 19] and allows SERENE to reach high rewards in all reward areas. Other baselines exhibit a less balanced distribution of the evaluation budget, as they do not explicitly separate exploration from exploitation.

6.2 Exploration

Performing good exploration in situations of sparse rewards is fundamental in order to discover all the possible rewarding areas of the search space. In our experiments, we measured the exploration capacity of each of the tested algorithms through the *coverage metric* [27, 29]. It is evaluated by discretizing the search space in a 50×50 grid and calculating the percentage of cells occupied by the policies found during the search. This metric does not include any measure of the performance of the solutions in the cells.

The plots in Figure 5 show that SERENE can perform exploration with an efficiency comparable to NS, notwithstanding the lower budget assigned to exploring the search space. At the same time, Figure 5 shows that the final coverage obtained by ME is similar to the one of NS and SERENE.

On the contrary, although based on ME, CMA-ME results are more variable across all environments, and exhibit lower exploration compared to ME. This effect is likely due to the reliance on emitters for exploration, leading to more local exploration in the parameter space Θ. It can prove useful in environments like Curling or Redundant arm, where a small change in parameters leads to big behavioral changes, increasing the probability of finding a reward. On the contrary, environments like Hard Maze or Robotic ant maze in which this does not happen can prove more challenging to explore.

At the same time, the exploration performance of NSGA-II is poor. In the Redundant arm environment, exploration is even lower than the random search baseline. This result is likely due to the multi-objective approach of optimizing both novelty and reward through Pareto fronts. Therefore, as soon as a reward area is discovered, the best strategy to improve the front is to focus on the reward because this scales better than the novelty.

6.3 Exploitation

Figure 6 shows the average maximum reward achieved by the algorithms in the reward areas of all environments. Emitters solely focusing on exploiting the reward allow SERENE to reach almost the maximum reward on the easiest to reach reward areas in less than 10^5 evaluations. High rewards are also achieved on the harder to reach areas, even if the required time is higher. On the contrary, ME improves on the reward at a much slower pace. This is likely due to the random selection of policies from the archive to generate new policies. In a sparse reward environment in fact, the probability of selecting a rewarding policy is proportional to the ratio between the rewarding and non-rewarding areas. The sparser the reward is, i.e. the smaller the reward area is, the lower the probability of selecting a rewarding policy from the archive is, and the slower the exploitation gets. A similar trend is exhibited by CMA-ME: even if able to reach high rewards on the discovered reward areas, it is slow in its optimization. At the same time, even NS reached high rewards on almost all environments, but without any explicit reward optimization it did not exploit the reward areas to the maximum. The multi-objective approach NSGA-II can always find at least one of the multiple reward areas, but then tends to extensively focus on it, instead of also exploring other areas. For this reason only the easiest reward area is exploited to high values in all environments, while the harder reward area is seldom exploited.

7 CONCLUSION AND FUTURE WORK

In this work we introduced SERENE, a method that efficiently deals with sparse reward environments by augmenting NS with emitters. Contrary to similar methods using emitters, SERENE keeps exploration and exploitation of the reward as two distinct processes. Exploration is carried out by taking advantage of NS to discover all the reachable reward areas. These areas are then exploited by using

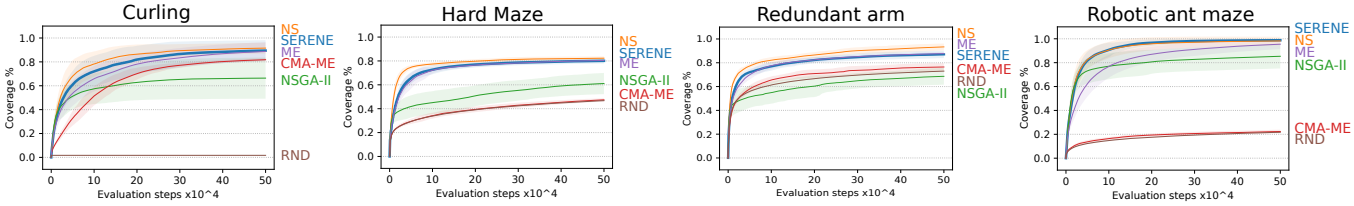

Figure 5: Average coverage with respect to the given evaluation budget. The shaded areas represent one standard deviation.

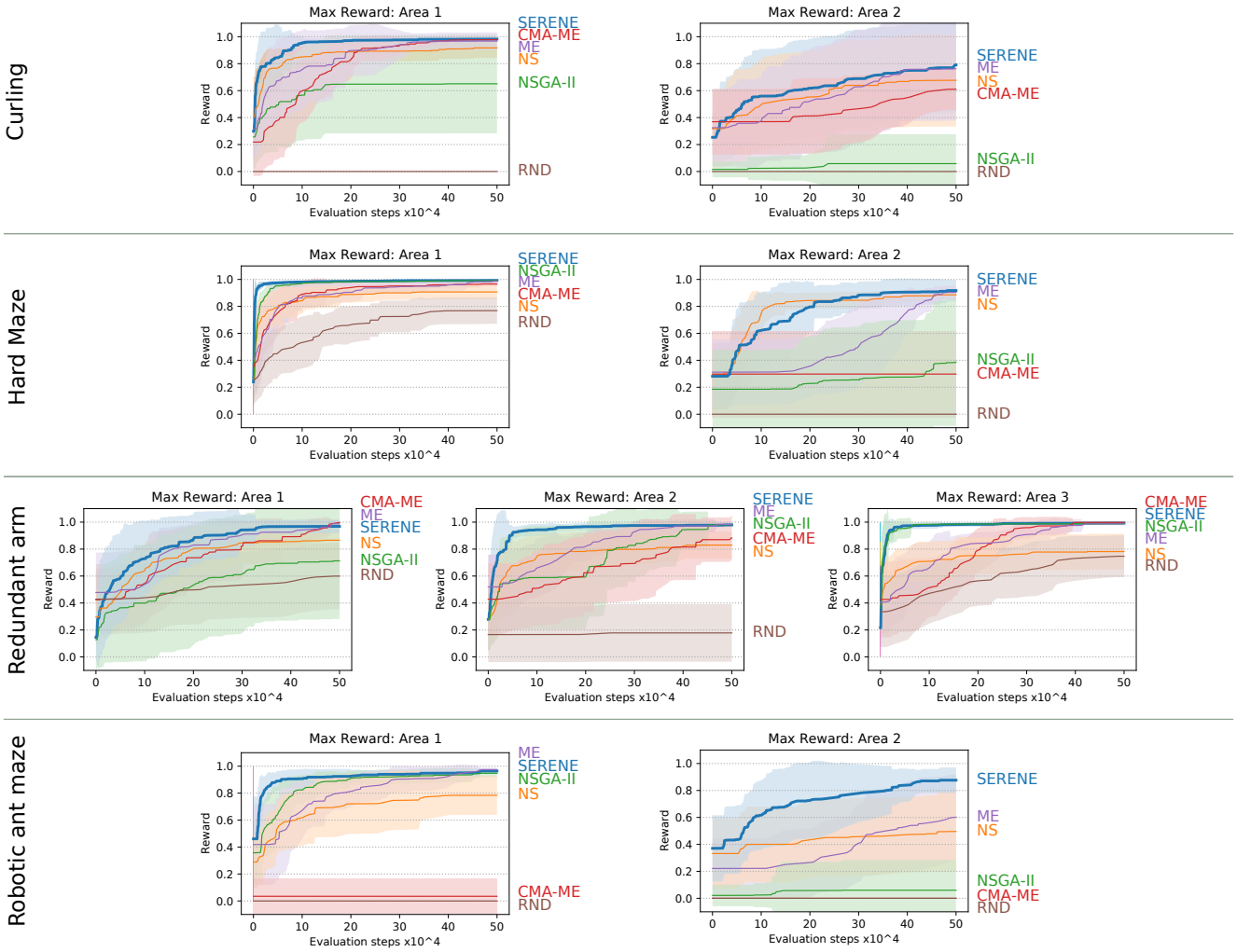

Figure 6: Average maximum reward reached in all the reward areas. The shaded areas represent one standard deviation.

local instances of population-based optimization algorithms called emitters. By using a meta-scheduler, SERENE can automatically assign the evaluation budget to either exploration or exploitation. This is advantageous also in situations in which no reward is present: in the absence of reward to exploit, SERENE performs exactly like NS.

SERENE has been tested on four different sparse reward environments, reaching high performances on all of them. Notwithstanding these encouraging results, the method still suffers from the same limitations as other QD methods, and first and foremost from the prior hand-design of the behavior space $\mathcal{B}$. In the future we will work on addressing this limitation by learning a behavior descriptor that could foster exploration towards rewarding solutions.

At the same time, it has been highlighted by Cully [9] that many kind of emitters can be used to address different kind of problems. Evaluating and combining different types of emitters is also an exciting line of work to extend the current method.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems*. 5048–5058.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016), 1471–1479.

[3] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies–A comprehensive introduction. *Natural computing* 1, 1 (2002), 3–52.

[4] Sebastian Blaes, Marin Vlastelica, Jia-Jie Zhu, and Georg Martius. 2019. Control What You Can: Intrinsically Motivated Task-Planning Agent. In *Advances in Neural Information Processing (NeurIPS'19)*. Curran Associates, Inc., 12520–12531.

[5] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. 2020. Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills. *arXiv preprint arXiv:2002.03647* (2020).

[6] Geoffrey Cideron, Thomas Pierrot, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. 2020. QD-RL: Efficient Mixing of Quality and Diversity in Reinforcement Learning. *arXiv preprint arXiv:2006.08505* (2020).

[7] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. 2018. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *International Conference on Machine Learning*. PMLR, 1039–1048.

[8] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in neural information processing systems*. 5027–5038.

[9] Antoine Cully. 2020. Multi-Emitter MAP-Elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. *arXiv preprint arXiv:2007.05352* (2020).

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

[11] Antoine Cully and Yiannis Demiris. 2017. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (2017), 245–259.

[12] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[13] Thang Doan, Bogdan Mazoure, Moloud Abdar, Audrey Durand, Joelle Pineau, and R Devon Hjelm. 2019. Attraction-repulsion actor-critic for continuous control reinforcement learning. *arXiv preprint arXiv:1909.07543* (2019).

[14] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. 2019. Novelty search: a theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 99–106.

[15] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (2021), 580–586.

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

[17] Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. 2020. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 genetic and evolutionary computation conference*. 94–102.

[18] Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. 2017. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190* (2017).

[19] Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. 2013. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences* 17, 11 (2013), 585–593.

[20] Daniele Gravina, Antonios Liapis, and Georgios Yannakakis. 2016. Surprise search: Beyond objectives and novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 677–684.

[21] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[22] Whiyoung Jung, Giseung Park, and Youngchul Sung. 2020. Population-guided parallel policy search for reinforcement learning. *arXiv preprint arXiv:2001.02907* (2020).

[23] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*. 1188–1200.

[24] Joel Lehman and Kenneth O Stanley. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. 329–336.

[25] Joel Lehman and Kenneth O Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 211–218.

[26] Pontus Loviken and Nikolas Hemion. 2017. Online-learning and planning in high dimensions with finite element goal babbling. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, 247–254.

[27] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[28] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. 2018. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*. 9191–9200.

[29] Giuseppe Paolo, Alban Laflaquiere, Alexandre Coninx, and Stephane Doncieux. 2019. Unsupervised Learning and Exploration of Reachable Outcome Space. *algorithms* 24 (2019), 25.

[30] Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. 2020. Effective diversity in population-based reinforcement learning. *arXiv preprint arXiv:2002.00632* (2020).

[31] Aloïs Pourchot and Olivier Sigaud. 2018. CEM-RL: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222* (2018).

[32] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.

[33] Olivier Sigaud and Freek Stulp. 2019. Policy search in continuous action domains: an overview. *Neural Networks* 113 (2019), 28–40.

[34] Christopher Stanton and Jeff Clune. 2016. Curiosity search: producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime. *PloS one* 11, 9 (2016), e0162235.

[35] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[36] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*. 2753–2762.

[37] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. 2019. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *Advances in Neural Information Processing Systems*. 10376–10386.

[38] Pradnya A Vikhar. 2016. Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*. IEEE, 261–265.

A FINAL ARCHIVE DISTRIBUTION

In figure 7 we show the distribution of the behaviors of the policies in the final archive. Each point is represents different policy. In blue are the policies that do not get any reward, thus considered *exploratory*, while in orange are rewarding policies, considered *exploitative*. For SERENE the *exploratory policies* are the ones in the *novelty archive* $\mathcal{A}_N$, while the *exploitative policies* are the ones in the *rewarding archive* $\mathcal{A}_R$.

We can see that even if the coverage metric values for SERENE are lower with respect to ME, the search space is well covered. Moreover, the reward areas are densely explored.

B HYPERPARAMETERS

The values of the hyperparameters used during the experiments are listed here. For each experiment we used a budget of $Bud = 500000$ evaluations, with the chunk size set to $K_{Bud} = 1000$. The population size is $M = 100$, and for each policy we generate $m = 2$ offspring. As mutation parameter we used $\sigma = 0.5$, while the number of policies uniformly sampled to be added to the novelty archive is $N_Q = 5$. SERENE uses an emitter population size of $M_{\mathcal{E}} = 6$, with a bootstrap phase for each emitter of $\lambda = 6$ generations. For CMA-ME we used the same parameters used by Fontaine et al. [17]: 15 emitters, each one with a population size of 37. In every experiment, the policies parameters are bounded in the $[-5, 5]$ range.

C TERMINATION CRITERION

The termination condition used for our emitters is inspired by the *stagnation criteria* introduced in [21]. We track the history of the rewards obtained over the last $120 + 20 * n/\lambda$ emitter's generations. Where n is the size of the parameter space Θ and λ is the emitter's population size. The emitter is terminated if either the maximum or the median of the last 20 rewards is not better than the maximum or the median of the first 20 rewards.

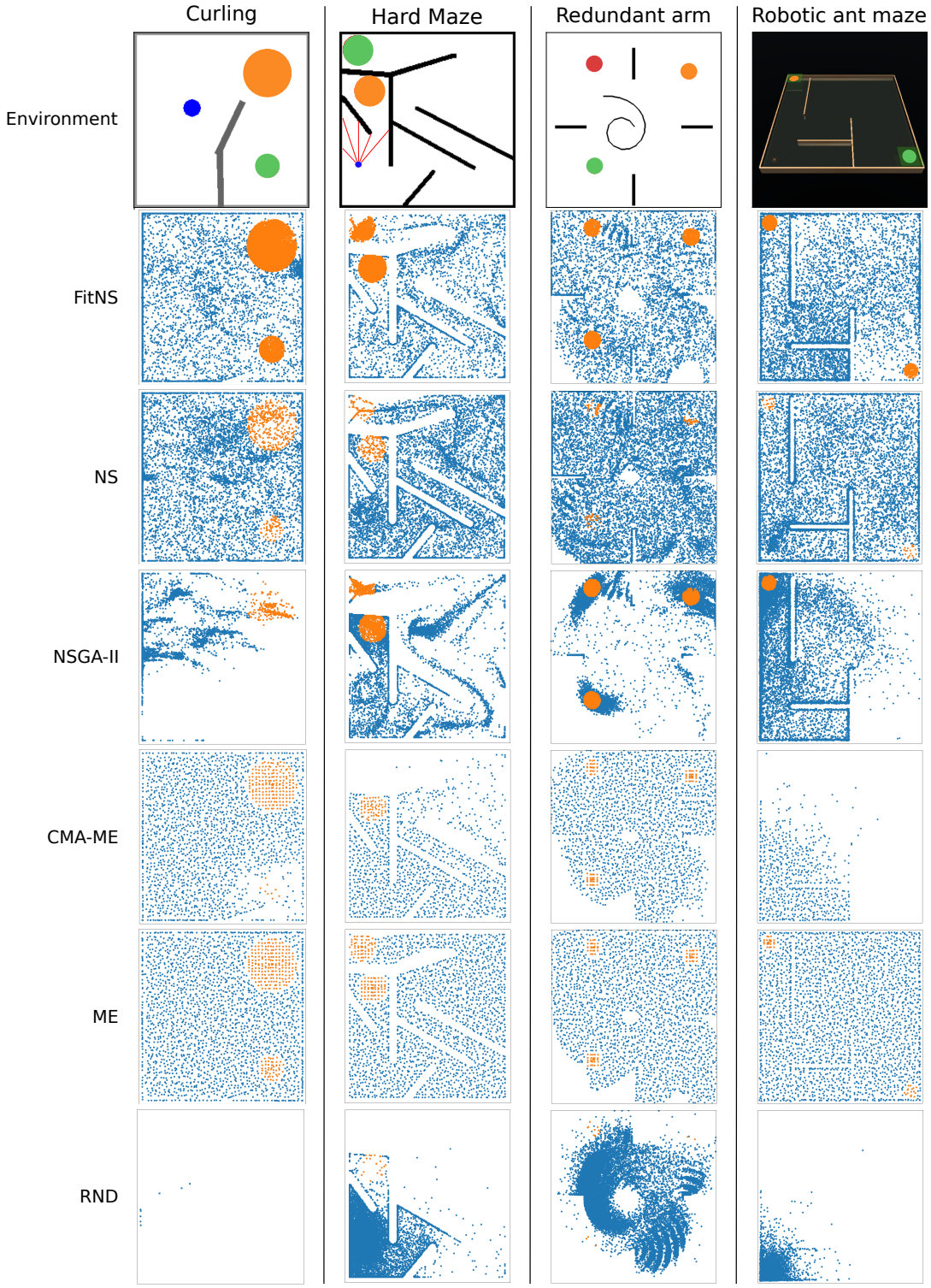

Figure 7: Distribution of the behavior descriptors of the archived policies. On each column are shown the results for an environment, while on each row is shown the distribution for each experiment. The archive plotted are from the runs achieving highest coverage. In blue are the policies with no reward, in orange the policies with a reward. For SERENE in blue are the policies in the *novelty archive* and in orange the policies in the *reward archive*.