A Implementation Details

A.1 Algorithm Hyperparameters

We give in Table 2 all the hyperparameters used for GDA-QD.

HYPERPARAMETER	MAP-ELITES	PGA-ME	DA-QD-EXT	GDA-QD
Policy hidden layer sizes	[64, 64]	[64, 64]	[64, 64]	[64, 64]
Batch size, B	512	512	512	512
Iso coefficient, σ_1	0.01	0.01	0.01	0.01
Line coefficient, σ_2	0.1	0.1	0.1	0.1
Max. imagined iterations, N	-	-	100	100
Size of add buffer $\mathcal{B}_{add}$	-	-	512	512
Surrogate hidden layer sizes	-	-	[512, 512]	[512, 512]
Surrogate ensemble size	-	-	4	4
Surrogate learning rate	-	-	0.001	0.001
Surrogate batch size	-	-	512	512
Max. model training steps, n_{steps}	-	-	2000	2000
Surrogate replay buffer size	-	-	4000000	4000000
Model update period, J_{model}	-	-	25	25
Max epochs since improvement	-	-	10	10
Proportion Mutation ga	-	0.5	-	-
Num. critic training steps	-	300	-	300
Num. PG training steps	-	100	-	100
PGA replay buffer size	-	1000000	-	1000000
Critic hidden layer size	-	[256, 256]	-	[256, 256]
Critic learning rate	-	0.0003	-	0.0003
Greedy learning rate	-	0.0003	-	0.0003
PG update learning rate	-	0.001	-	0.001
Noise clip	-	0.5	-	0.5
Policy noise	-	0.2	-	0.2
Discount γ	-	0.99	-	0.99
Reward scaling	-	1.0	-	1.0
Transitions batch size	-	256	-	256
Soft tau update	-	0.005	-	0.005
Proportion model, p_{model}	-	-	-	0.9

Table 2: Hyperparameters of GDA-QD and baselines.

Perturbation Operators. We use a directional variation by Vassiliades and Mouret [41] which is defined by the equation:

$$\tilde{\theta} = \theta_1 + \sigma_1 \mathcal{N}(0, I) + \sigma_2 \mathcal{N}(0, 1)(\theta_2 - \theta_1) \tag{5}$$

where θ_1 and θ_2, are the parameters of two policies from the population. Perturbed parameters, $\tilde{\theta}$, are obtained by adding Gaussian noise with a covariance matrix $\sigma_1 N(0, I)$ to θ_1 controlled by isolated noise coefficient σ_1. The resulting vector is then moved along the line from θ_1 to θ_2 by line coefficient σ_2. These hyperparameters can be found in Table 2 and have been found empirically.

DA-QD details. QD is performed in imagination until the add buffer $\mathcal{B}_{add}$ is filled or until a maximum number of imagined iterations N. This is to prevent the algorithm being stuck in imagination especially when the algorithm is close to convergence.

The dynamics model is trained every J_{model} iterations of the full QD loop.

Dynamics model details. The hyperparamters of the model architecture can be found in Table 1. We train the model by minimizing the negative log-likelihood as done in Chua et al. [4]. Everytime the model is trained, the replay buffer is randomnly split into a train and validation sets. The model is trained until the the improvement of the validation loss is below 1% or until a maximum number of gradient steps is reached n_{steps}. The dynamics model rollout length in imagination is always equivalent to the corresponding episode length of the environment.

Policy Gradient operator details We use the hyper-parameters from the PGA-ME [30] for the training of the TD3 agents (both actor and critic components). Greedy here refers to the base actor policy/agent in TD3. PG learning rate here refers to the learning rate used when applying the policy gradient perturbations to policies sampled from the population.

A.2 Environment Hyperparameters

We provide the different hyper-parameters used for the different environments considered.

HYPERPARAMETER	POINTMAZE	ANTTRAP	ANTMAZE
Episode Length, T	250	200	500
Evaluation Budget, J	1,000,000	1,000,000	4,000,000
Population size	2500	2500	2500

Table 3: Environment hyperparameters used.

B Ablation of the proportion of policy-gradient

In this section, we aim to demonstrate the importance of both types of perturbations used in GDA-QD: policy-gradient and random model-filtered perturbations. We run an ablation of the different proportions p_{model} and $p_{gradient} = 1 - p_{model}$ of perturbations at each generation. In Figure 6, we can see that the configuration of $p_{gradient} = 0.1$ performs the best. This indicates that while the policy gradient operators are important, they are also generally wasteful evaluations. This is further backed up by the fact that the performance deteriorates as the proportion $p_{gradient}$ increases over p_{model}. However, it is also still key to maintain some proportion of policy gradient updates $p_{gradient}$, as not using them is detrimental to the max total reward which results in an overall lower performance of the population Θ.

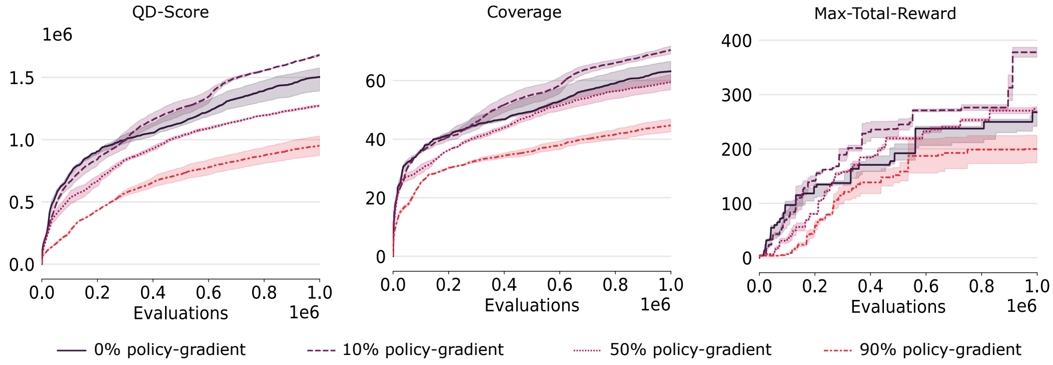

Figure 6: QD-Score (left), Coverage (middle) and Max-Total-Reward (right) on the AntTrap task of GDA-QD with different values of $p_{gradient}$. $p_{gradient}$ gives the proportion of policies improved using policy-gradients. Each experiment is replicated 3 times, the solid line corresponds to the median over replications and the shaded area to the first and third quartiles.

C Supplemetary Results

C.1 Visualization of population

This section provides the plots of the population of policies for the PointMaze (Figure 8) and AntMaze (Figure 9) environments respectively.

Fig. 4 plotted PointMaze for 500,000 evaluations for greater presentation clarity. We show the full performance curve after 1 million evaluations here to ensure convergence and the visualization of the final population of policies.

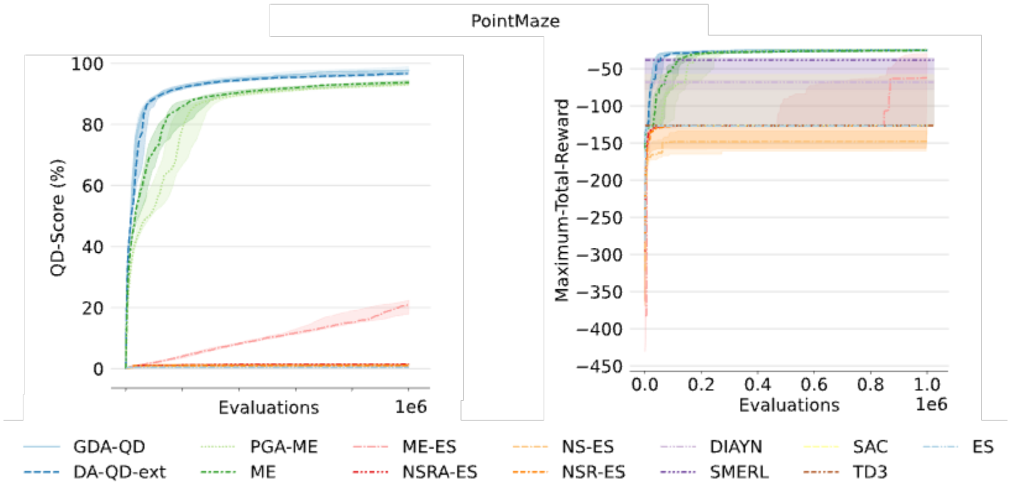

Figure 7: QD-Score (left) and Max-Total-Reward (right) on the PointMaze task. The solid line corresponds to the median over 15 replications and the shaded area to the first and third quartiles. This plot shows performance over 1 million evaluations.

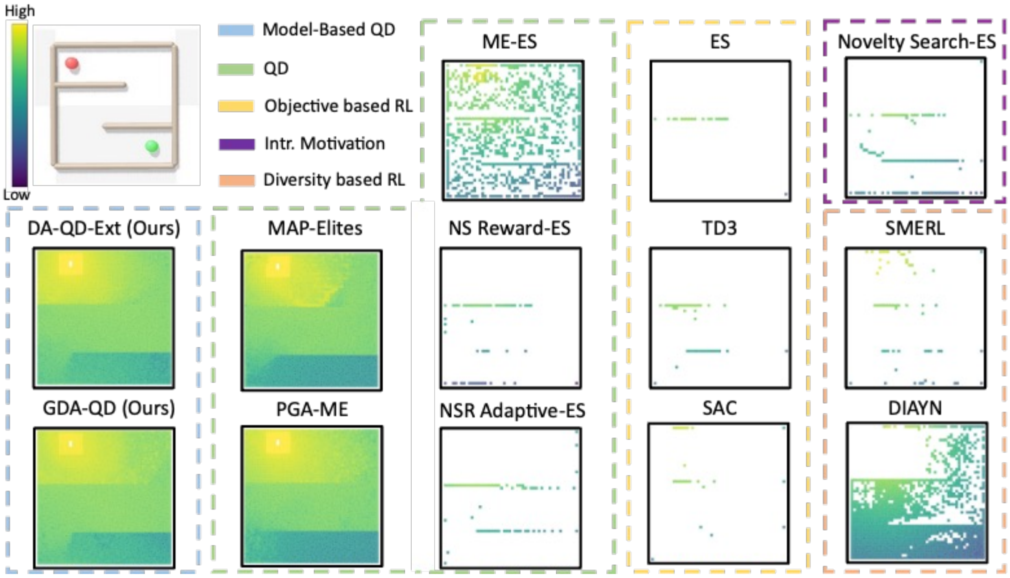

Figure 8: Final population of methods in the PointMaze environment (top left). Each policy in the population is represented as a filled square in the final position (descriptor) it reaches within the episode. The color of the square of policy indicates the total reward, where the lighter the better.

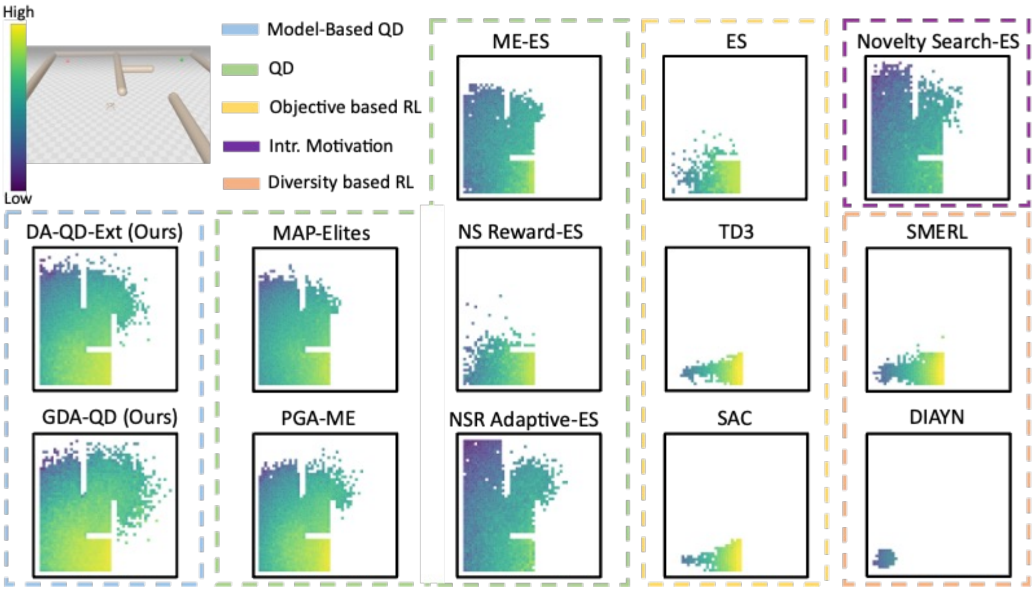

Figure 9: Final population of methods in the AntMaze environment (top left). Each policy in the population is represented as a filled square in the final position (descriptor) it reaches within the episode. The color of the square of policy indicates the total reward, where the lighter the better. The boundaries of the maze clearly appears on this plot showing the deceptive nature of the task. Most objective-based baselines do not manage escape this.

C.2 Coverage curves

Figure 10 shows the coverage curve for each of the baseline algorithms. This metric measures the diversity and supplements the Max. Total Reward (which measures the quality) and the QD-score (which measures both quality and diversity).

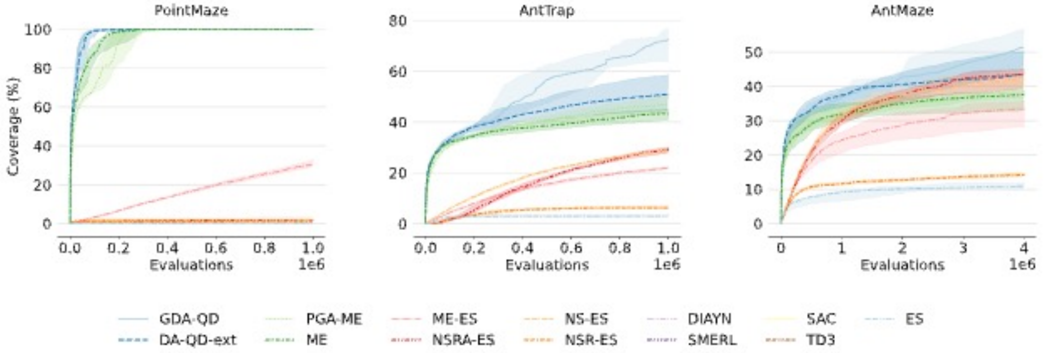

Figure 10: Coverage on the PointMaze, AntTrap and AntMaze tasks. Each experiment is replicated 15 times, the solid line corresponds to the median over replications and the shaded area to the first and third quartiles.

C.3 Metrics Definitions and Discussion

As explained in Section 4, we evaluate the performance of the algorithms on QD-Score, Max Total Reward and Coverage. For fair comparison with single-policy and latent-conditioned RL baselines such as TD3, SAC, DIAYN and SMERL, we run the algorihtms for the same number of steps as evaluation budget × episode length. This way, it has access to the same number of steps as the population based QD methods. We take the best performance of the agents throughout the learning process and plot them as a horizontal line to represent best performance.

For the exceptional case of the AntMaze, the reward at every timestep corresponds to the distance from the goal at every timestep. This corresponds to a deceptive reward for this environment setup as discussed in [24]. To be fair to RL baselines, a time-step reward is given instead of a just a final reward at the end of episode which corresponds to the distance to the goal. However, this poses some problems during evaluation as the max total reward if not representative of solving task. Solving the maze requires moving further away from the goal, resulting in decreasing rewards at every timestep and a final episode return that could possibly be lower that not solving the maze. For example, moving quickly and getting stuck hovering around the region close to the wall near the goal would be able to give a high episodic return while definitely not solving the Maze. This is evident and is done by the ES, TD3, SAC, SMERL baselines as seen in Figure 9. One way to solve this could be to give a very large reward for reaching the goal. To avoid reward tuning, we opt to keep the simple reward function of distance to the goal. Hence, using this metric naively would be evaluating the task wrongly. Instead, we evaluate by normalising the max total reward obtained by the final distance travelled by the policy at the end of the episode.

D NSRA-ES dynamics in hard exploration deceptive reward tasks

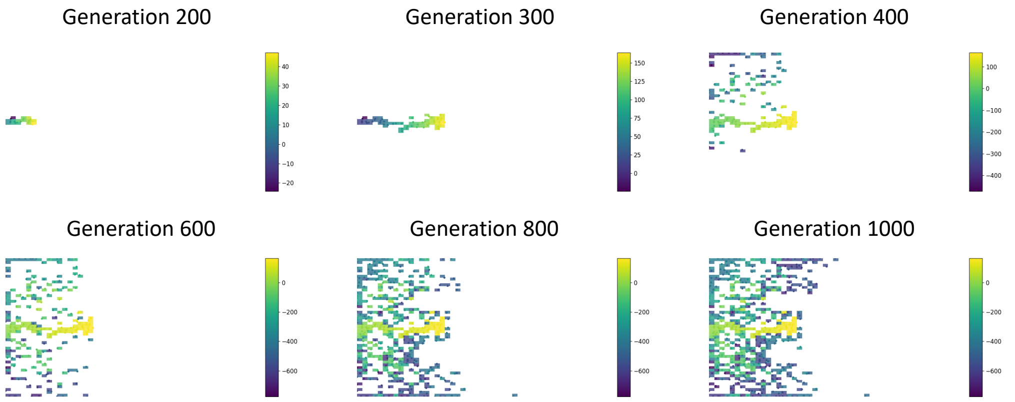

Figure 11: Population of NSRA-ES [8] at multiple generation number on the AntTrap environment. Each policy in the population is represented with a dot in the final position it manages to reach within the episode. The color of the square around each policy indicates its total reward, the lighter the better. The obstacle clearly appears on this plot, as the empty area in the middle.

Figure 11 shows a visualisation of the population generated by NSRA-ES at multiple generations number in AntTrap. NSRA-ES has an adaptive mechanism to weigh the exploitation and exploration in the objective given to its ES process. It starts with full exploitation, as can be seen in the first two visualisations in Figure 11. However, after a few hundred generations (~ 300), NSRA-ES reaches the end of the trap, and thus the maximum total-reward it can expect to get in this direction. Thus, the adaptive mechanism kick in and the proportion of exploration raises, giving more weight to the novelty reward, as can clearly be seen in the following visualisations. However, NSRA-ES adaptive mechanism has one limitation: the proportion of exploitation in the ES-objective can only go up again when the algorithm finds solutions at least as performing as the best solutions found so far. However, in this complex control task, finding such high-performing solutions requires fine-tuning newly discovered solutions. Thus, after approximately 400 generations, NSRA-ES remains in full exploration (novelty reward only) and does not manage to find high-performing solutions.