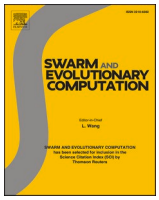
Swarm and Evolutionary Computation 99 (2025) 102140



Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



Using genetic programming to improve data collection for offline reinforcement learning

David Halder *, Georgios Douzas, Fernando Bacao

NOVA Information Management School Instituto Superior de Estatística e Gestão de Informação, Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal

ARTICLE INFO

Keywords:
Offline reinforcement learning
Genetic programming
Evolutionary reinforcement learning
Evolutionary algorithms
Data efficiency

ABSTRACT

Offline Reinforcement Learning (RL) learns policies solely from fixed pre-collected datasets, making it applicable to use-cases where data collection is expensive or risky. Consequently, the performance of these offline learners is highly dependent on the dataset used. Still the questions of how this data is collected and what dataset characteristics are needed are not thoroughly investigated. Simultaneously, evolutionary methods have reemerged as a promising alternative to classic RL, leading to the field of evolutionary RL (EvoRL), combining the two learning paradigms to exploit their supplementary attributes. This study aims to join these research directions and examine the effects of Genetic Programming (GP) on dataset characteristics in RL and its potential to enhance the performance of offline RL algorithms. A comparative approach was employed, comparing Deep Q-Networks (DQN) and GP for data collection across multiple environments and collection modes. The exploration and exploitation capabilities of these methods were quantified and a comparative analysis was conducted to determine whether data collected through GP led to superior performance in multiple offline learners. The findings indicate that GP demonstrates strong and stable performance in generating high-quality experiences with competitive exploration. GP exhibited lower uncertainty in experience generation compared to DQN and produced high trajectory quality datasets across all environments. More offline algorithms showed statistically significant performance gains with GP-collected data than trained on DQN-collected trajectories. Furthermore, their performance was less dependent on the environment, as the GP consistently generated high-quality datasets. This study showcases the effective combination of GP's properties with offline learners, suggesting a promising avenue for future research in optimizing data collection for RL.

1. Introduction

In 2019, OpenAIFive defeated the Dota 2 world champions, showcasing a major success in Reinforcement Learning (RL) with Deep Learning [1]. This accomplishment emphasized the potential of RL beyond video games, extending to domains such as healthcare [2], autonomous driving [3] and education [4].

In the context of RL, "data collection" refers to the process by which an agent gathers experiences from interacting with an environment. These experiences, often represented as transition tuples (state, action, reward, next state), are accumulated and stored for offline learning. Classic RL's trial-and-error learning makes it unsuitable for expensive or dangerous data collection. To address this challenge, a distinct form of RL evolved, emphasizing a data-centric methodology and eliminating the necessity for environmental interactions: offline RL. Experiences are collected by an unknown policy to train the agent, avoiding further environmental interactions. However, in practice offline RL poses major algorithmic challenges, distributional shift between the training data and the states witnessed during deployment being a central one [5]. Consequently, many algorithmic advances to learn more effectively from pre-collected data have been proposed.

Recent studies have shifted the focus from enhancing learning algorithms to optimizing the data generation process and exploring the synergies between dataset characteristics and learning algorithms [6,7]. To achieve this and to improve data efficiency in RL, this line of research disentangles the RL process into acting (data collection) and learning (inference). One algorithm interacts with the environment, e.g., playing a videogame, and stores experiences (transition tuples). Another

* Corresponding author at: NOVA Information Management School Instituto Superior de Estatística e Gestão de Informação, Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal.
E-mail address: davidhalder@outlook.de (D. Halder).

https://doi.org/10.1016/j.swevo.2025.102140
Received 6 October 2024; Received in revised form 4 July 2025; Accepted 20 August 2025
2210-6502/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

algorithm then uses these tuples to learn a policy offline. Thus, these subprocesses are decoupled and solely connected through a transition memory, where the experiences generated are stored and drawn from for learning [8].

Simultaneously, Evolutionary Computation (EC) methods resurfaced as an alternative approach to classic RL for sequential control problems, as they are simpler to implement, handle sparse rewards effectively and are easier to scale in a distributed setting [9]. Their high exploration capabilities motivated combining EC and RL to leverage complementary strengths. This area of study is referred to as Evolutionary RL (EvoRL).

Despite the large interest in these research areas, no studies have yet provided a comprehensive analysis benchmarking EC for data generation in RL, with the goal of analyzing its effects on the collected datasets and subsequent offline learners. This study investigates how data collection via Evolutionary Algorithms (EA), particularly Genetic Programming (GP), affects experience distribution in RL. Furthermore this work aims to analyze the dataset traits given by experiences collected via and compares GP's data collection effectiveness to Deep-Q-Networks (DQN) and its impact on offline RL performance. The research questions are as follows:

- **Research Question 1**: How does the use of GP in data collection influence the explorational and exploitational power of the dataset in comparison to a DQN approach?
- **Research Question 2**: To what extent do off-policy and offline RL algorithms benefit from datasets generated by GP in comparison to collection via DQN?

Given the established properties and advantages of EAs, it is hypothesized that data collection using GP leads to superior performance in subsequently offline trained RL models compared to data collection using DQN. This work builds upon the methodologies and foundational principles established by Schweighofer et al. [6] and Riedmiller et al. [8] respectively. Their studies provide the techniques and the essential framework that underpin this research.

Experiments are conducted to address the objectives and research questions. The experiments compare collectors across environments and modes, benchmarking offline RL methods to test for significant differences. For the analysis of the dataset characteristics, previously established indicators are applied, taking datasets of policies with different expert levels as the basis for the relative exploration and exploitation capabilities. The experiments are set up in Python. The full code can be found here: https://github.com/dropthedave/offlineRL_thesis.

This study reviews related works, outlines the methodology, presents and discusses results, and concludes with limitations and future research directions.

2. Related works

2.1. Dataset characterization and evaluation

Several studies have emphasized the importance of dataset characterization and evaluation, addressing gaps in understanding how dataset properties influence model performance.

Schweighofer et al. [6] highlighted the limited research on dataset characteristics in offline RL and defined two policy traits: exploratory (high information yield) and exploitative (high expected return). They introduced two measures, State-Action Coverage (SACo) and Trajectory Quality (TQ), used in this study (detailed in Section 4.2). They showed that off-policy algorithms need high SACo, offline RL works best with high SACo and TQ, and Behavior Cloning (BC) [10] performs well with high TQ.

Concurrently, Swazinna et al. [11] proposed three indicators for measuring dataset characteristics: Estimated relative return improvement, estimated action stochasticity and a combined offline indicator, uniting the first two. These indicators measure the return expectation, the exploratory behavior, and the combination respectively.

Unlike prior works, Singh et al. [12] adapted an algorithm to address challenges in collected data characteristics. They examined action variability across states, suggesting offline RL should align with or deviate from behavior policy based on state-action distribution. They proposed Conservative Q-Learning (CQL) [13] with Reweighting Distribution Constraints, demonstrating its performance improvement and emphasizing the role of dataset characteristics in offline RL.

A similar approach was adopted by Jiang et al. [14] with a focus on the imbalance over the state space given in RL datasets. They specified properties of imbalanced datasets in offline RL and showed that learning methods based on distributional constraints are ineffective in extracting policies from them. They proposed an offline RL algorithm incorporating trajectories from larger datasets, demonstrating its empirical superiority over baselines.

Like this work, the studies presented had a focus on the characteristics of the collected data and their effects on offline RL. This study aims to build upon this work to find out how datasets collected by GP are distributed and what characteristics they show in terms of exploitation and exploration in comparison to DQN.

2.2. Data collection optimization for offline RL

2.2.1. Self-supervised and unsupervised RL

Another research direction aims to improve data collection for offline RL. Levine [15] proposed a framework optimizing environment exploration via self-supervised or unsupervised RL.

Levine argued this approach enables collection policies to develop a causal understanding of their environment. Unlike classic RL, which receives rewards with each interaction, unsupervised RL uses reward-free pre-training. For example, Curiosity-Driven Exploration incentivizes agents to explore novel states, increasing diversity in states and actions [16]. Offline RL complements this data collection by enabling the use of large pre-collected datasets. Levine [15] suggested that combining unsupervised RL with offline RL could create scalable, automated algorithms.

Yarats et al. [7] followed this framework, proposing ExORL, a data-centric methodology for offline RL. Their framework uses unsupervised RL for collection before the collected trajectories are relabeled and then used to train an off-policy RL algorithm. They tested nine unsupervised data collection algorithms: knowledge-, data-, or competence-based. After relabeling the collected data, it was applied to five offline/off-policy RL algorithms for performance comparison. The results show that exploratory data enables an off-policy RL algorithm to outperform state-of-the-art offline RL algorithms, suggesting that data generation is as important as algorithmic advances in the field of offline RL.

Lambert et al. [17] followed a similar approach. While their pre-training/data collection phase is like other unsupervised RL approaches, they used the relabeled offline transitions collected before to support the learning phase. They concluded that with this method, performances comparable to classic online training can be achieved, but more research towards improved exploration methods is needed.

Sun et al. [18] recently proposed another approach for unsupervised data collection in RL. Their method also leverages intrinsic model predictive control. However, in contrast to Lambert et al. [17] their framework, Curiosity-Driven Unsupervised Data Collection, additionally exploits importance weights to give additional attention to under-learned tuples and expands the feature space gradually by using an adaptive temporal distance, instead of a fixed one.

Endrawis et al. [19] proposed a data collection method for offline RL utilizing IM as well. However, contrary to the previously presented studies, their framework follows a self-supervised approach instead of an unsupervised one. Their idea was to drive the agent towards unexplored states by using goal conditioned RL. The goals are set towards states which are considered novel at that point in time according to the

IM. Thus, exploration is maximized as the agent will directly navigate towards it due to the goal conditioned policy.

The presented works focus on offline RL data collection, enhancing it by increasing exploration. Furthermore, some of them also work in a decoupled manner, exploiting the enhanced flexibility of this framework by combining different collection and learning algorithms. Nevertheless, none of the papers exploits the advantages of EAs to improve the generation of experiences.

2.2.2. EvoRL for performance

EvoRL combines RL and EC to leverage their strengths and offset weaknesses, as they can be complementary in certain environments. RL has high sample efficiency and can learn from previously generated experiences, but its exploration is limited. EAs offer global exploration but have low learning efficiency [20]. The next section overviews studies merging RL and EAs. This review focuses on approaches using EAs to generate diverse experiences while improving population performance. Given that this represents just one of numerous research paths within EvoRL, the interested reader is referred to Li et al. [20], Sigaud [21], Majid et al. [22] and Bai et al. [23].

One researched method to leverage the advantages of both learning paradigms is policy/actor injection. EvoRL methods with policy injection do not fully decouple data collection and learning. They mostly utilize EAs to provide diverse samples and subsequently use RL methods for maximal information extraction from individual experiences and thereby increasing sample efficiency. These methods periodically inject a trained RL agent into the population to guide learning [20]. This exploits RL and EC advantages but limits algorithm flexibility, as learner structures must align.

A pivotal EvoRL study implementing actor injection is Evolutionary RL (ERL), introduced by Khadka & Tumer [24]. In ERL a population of actors is episodically evaluated, selected, and altered by a GA. The generated experiences are saved in the replay buffer, and then used by the actor and the critic to learn from them recurrently, utilizing the learning procedure of the Deep Deterministic Policy Gradient (DDPG) algorithm [25]. Periodically, the actor is injected into the population, to leverage the information learned via gradient descent.

ERL has been the foundation of many subsequent studies. For instance, Collaborative ERL was issued by Khadka et al. [26] in the following year, implementing a resource manager in the ERL framework to enable parallelization. Espositi & Bonarini [27] proposed the algorithm X-DDPG, a version of ERL, which aims to solve the ineffectiveness of GAs in the presence of high variance of the fitness function. On top of policy injection, it employs two separate replay buffers, which respectively contain the experiences of the genetic and the RL side of the algorithm. Recently, B. Zheng & Cheng [28] introduced T-ERL, a hybrid algorithm combining ES and Twin Delayed DDPG (TD3) [29]. It employs a very similar approach to X-DDPG, constructing two replay buffers for RL and evolutionary experiences and then sampling from both, ensuring a highly diverse sample of trajectories from two different learning methods.

H. Zheng et al. [30] proposed Cooperative Heterogeneous Deep RL, a collaborative EvoRL framework extending ERL by maintaining a population of global and local agents. Global agents are off-policy, using experiences from others, while local agents are on-policy or population-based EA agents, focusing on effective local exploration to enhance data diversity. A year later Suri et al. [31] presented Evolution-based Soft Actor Critic (SAC) [32], which implements the ERL framework but replaces DDPG with SAC and the GA with an adjusted ES algorithm. Furthermore, they implemented an automatic mutation tuning mechanism, which enables individuals to perform highly at the given task while acting as randomly as possible, achieving high levels of exploration and exploitation, without using backpropagation updates.

Early EvoRL adaptations faced catastrophic forgetting, where ANNs lose prior knowledge when learning new information. This occurs with GOs using direct encoding [21]. Bodnar et al. [33] addressed this with Proximal Distilled ERL. Building on ERL, they introduced a genetic memory, which can be seen as a small replay buffer for every individual. Consequently, the data collected is used more effectively, since the individual experiences collected by the actors can be directly exploited. Using this genetic memory, the authors proposed two new GOs which enable the offspring to inherit ancestral experiences from its parents while not causing catastrophic forgetting.

Marchesini et al. [34] enhanced data collection with novel GOs. They enhanced the data collection process by proposing the concept of safe mutations which explore safer actions, minimizing costs. They implemented a replay buffer storing only unsafe states for the RL agent. Periodically this buffer is used to create a population consisting of mutations of the RL agent. This population then explores safer behaviors and adapts the parameters of the policy accordingly.

An additional example of more complex variation methods in EvoRL was presented by Lü et al. [35], building upon ERL and adapting it using Imitation Learning. They proposed an algorithm called Recruitment-Imitation Mechanism. This framework consists of a dual-actor and single critic RL agent which can recruit high performing actors from an evolutionary population, which then act in a dual policy decision making system. Simultaneously, low-fitness actors from the population imitate behaviors from the RL agent to improve their adaptability. These mechanisms enhance both the evolutionary process and RL experience generation. On top of that, this structure bypasses the issue that the RL and EA agents must have the same structure for actor injection.

Beyond catastrophic forgetting, EAs suffer high sample complexity, requiring multiple environment interactions for selection. To tackle this challenge and make data generation more efficient, some evolutionary frameworks utilize a fitness surrogate, estimating the fitness of individuals, to speed up convergence.

Kim et al. [36] addressed this by coupling a TD3 agent with population-based search. However, in contrast to other methods like ERL, their algorithm Policy Gradient with Population Based Search (PGPS) uses the Q-critic as a surrogate model to filter for promising actors for the coming generation. These actors account for half of the population, while the rest is filled with actors from the evolutionary loop and the current TD3 actor, which also gets a soft update towards the elite individual. They showed that their approach increases sample efficiency while exploiting the characteristics of the EA applied. Wang et al. [37] followed a similar approach, using a critic network for evaluating the members of the evolutionary population for their framework SC-ERL. Their framework exhibits a broader applicability compared to PGPS, extending its utility across various other ERL architectures.

A recent surrogate fitness approach is $ERL - RE^2$ [38]. For a given probability, the fitness of the actors is evaluated using a policy-extended value function approximator, increasing sample efficiency. However, their main contribution is a two-scale representation where all policies are decomposed into a joint state representation and sovereign linear policy representations, enabling an efficient transfer of information across individuals. This ensures that the data collected is used effectively.

In recent years, research on combining evolutionary methods with RL has expanded significantly. These studies have not only increased the performance of the respective collection and learning algorithms but have also shown the promising potential of this domain. However, gaps remain. While most of the methods presented utilize the experience replay mechanism, none of the approaches are fully decoupled. Consequently, the structure of the function approximators used by each component must be similar, with the work of Lü et al. [35] being the sole exception, making a combination of GP and DQN unfeasible.

This might explain that while various EAs have been combined with RL methods and even though there has been some research in the direction of RL assisted GP [39], no research efforts have been made towards hybrid algorithms, in the discussed setting of capitalizing on GP for higher diversity and exploration and consequently higher

performance. Dataset properties and algorithmic effects remain underexplored. To our knowledge, no fully offline RL methods have been combined with evolutionary approaches, marking a key research gap.

2.2.3. Diversity encouraging EvoRL

Previous EvoRL frameworks promote exploration and agent diversity, but EAs still focus on optimizing population performance. Nevertheless, the EAs used are still aiming to optimize the performance of the population. Despite advances like speciation and fitness sharing [40], parameter diversity doesn't guarantee behavioral diversity. This challenge exists because neural networks can produce identical behaviors through different weight configurations [23]. To address this, extensive research by various researchers has been directed towards methods which encourage diversity in the state or action space by actively rewarding diverse states and actions.

This work focuses on EvoRL methods using Novelty Search (NS) [41] and Quality Diversity (QD) [42] to stay within scope. The interested reader is referred to the following papers for a detailed description of Surprise Search [43], Evolvability Search [44] and Curiosity Search [45].

NS optimizes population diversity while ignoring environmental rewards. Thus, NS is useful in reward-sparse or reward-free environments. Q. Liu et al. [46] presented an approach utilizing NS enhancing data collection by maintaining high behavioral diversity. Their population-guided NS algorithm consists of n sub-populations, each with a fixed number of exploring agents and one chief agent. The chief agent evaluates the exploring agents' policies based on novelty, distributes the optimal policy to all sub-populations, and updates it if an exploring agent outperforms the current optimal policy. That way it is ensured that the policy of the exploring agents is highly explorative at all points in time.

A comparable approach is applied by NS-RL [47], which utilizes RL to improve the novelty of individuals created by a GA. This is achieved by conducting an off-policy gradient using the collected experiences to the least novel actors. After the parameters have been changed, the novelty is evaluated by the top agent, in terms of novelty, which acts as the critic. If the novelty improves, the adapted policies replace the original ones increasing the exploration potential of the population used for filling the replay buffer.

Unlike NS, which focuses solely on discovering novel solutions, QD aims to generate solutions that are not only novel but also maximize performance or reward, ensuring both diversity and high quality in the final set of solutions. Pierrot et al. [48] proposed a QD-RL algorithm, where a population is initiated from a Map-Elites [49] grid and then mutated by an off-policy agent. Half the population is mutated towards performance, and the other half towards diversity. The solutions then get tested on the environment and the experiences collected get saved in the replay buffer for further mutations via gradient descent. The final scores and information about the behavior are saved in the Map-Elites Grid, to sample a new population which is explorative as well as exploitative.

With their algorithm Policy Gradient Assisted Map Elites, Nilsson & Cully [50] enabled Map-Elites to efficiently evolve large neural networks. Solutions are altered using the gradient of a TD3 algorithm as well as the GOs of a GA. The operators are independent and used for direct performance improvement and divergent search respectively. In contrast to QD-RL, the solutions are altered towards diversity and performance at the same time. If the evolved ANN is either novel enough or more efficient, it is added to the MAP-Elites Archive. The CM-Mega algorithm [51] follows a similar methodology, also utilizing Map-Elites as the QD approach. However, for the improvement of quality they use a combination of TD3 and ES, and for the improvement of diversity they implemented an ES.

Like performance-focused EvoRL algorithms, these works lack data analysis, impact studies on learning algorithms, and integration with offline RL—highlighting a research gap.

2.2.4. Decoupled approaches

With their study, Riedmiller et al. [8] offered a new perspective on RL. They argued while data efficiency in RL has made major developments in learning, reaching from pure online to fully offline, the question of how the experiences are generated is vastly understudied. They hypothesized that data-efficient RL requires considering both collecting and learning, proposing a new structure: Collect and Infer. This framework models RL as two separate processes which are only interconnected through transition memory. The main advantage of this framework is that the data collection process can be optimized separately, since the policy which is optimal for collecting data may be very different from the optimal behavior for the task of interest. Treating RL as two processes offers high flexibility in algorithm combinations. As an illustration, ERL [24] utilizes two distinct algorithms for RL. Nonetheless, they remain intertwined due to policy injection, necessitating the utilization of learners which are structurally equal. A decoupled approach isolates each component's effects, enhancing understanding of performance drivers.

The aforementioned studies either utilized self- and unsupervised RL in combination with offline RL in a decoupled manner or combined EAs with online or off-policy RL algorithms (Section 3.1 to 3.3.3). However, there is a lack of research in combining EAs with offline RL in a framework like the one described by Riedmiller et al. [8] or Levine [15].

The work probably most closely related to this research was presented by Colas et al. [52]. In their study they combined a Goal Exploration Process, a Curiosity Search algorithm, with a DDPG. Similar to this study, the methods connect only through the replay buffer. Across two benchmarks, filling the replay buffer with a GEP before training DDPG proved more sample-efficient, with less variance and better final performance than training DDPG from scratch. Zimmer & Doncieux [53] presented a similar approach. They collected trajectories via a Neuroevolution algorithm, extracted a discrete state action representation and then inserted them into the replay buffer of a DQN algorithm. They showed that combining EA-derived knowledge with RL improves generalization to new domains and enables faster learning than direct policy search.

While these studies share very close similarities with this research, no comparison was made between different combinations of collection and learning algorithms. Although off-policy algorithms were used, no fully offline RL approaches were applied. Furthermore, no detailed analysis of dataset distributions, their characteristics in terms of exploration and exploitation and the effects of the collection algorithms applied on it was conducted. This study fills this gap by providing a comprehensive analysis of RL in a decoupled manner by combining two collection algorithms, which utilize four different collection modes, with four different learning algorithms across two RL environments. This study evaluates both learner performance and the explorational and exploitational efficiency of collection algorithms.

3. Methodology

3.1. Framework architecture

In this study, RL tasks involve interactions between an algorithm and an environment through sequences of states S, actions A and rewards R. Data collection is conducted using a GP algorithm and a DQN. Hyperparameters for training and evolution are detailed in Appendix C, with the framework illustrated in Fig. 3.1.

For the GP algorithm the process starts with random initialization of solutions composed of predefined functions and terminals. Each individual interacts with the environment for one episode, with the sum of rewards assigned as its fitness score. Based on fitness, parents are selected via tournament selection (four participants, best individual chosen), undergo one-point crossover (producing two offspring by swapping random subtrees), and are varied through subtree mutation (replacing a random subtree with one generated by a grow function

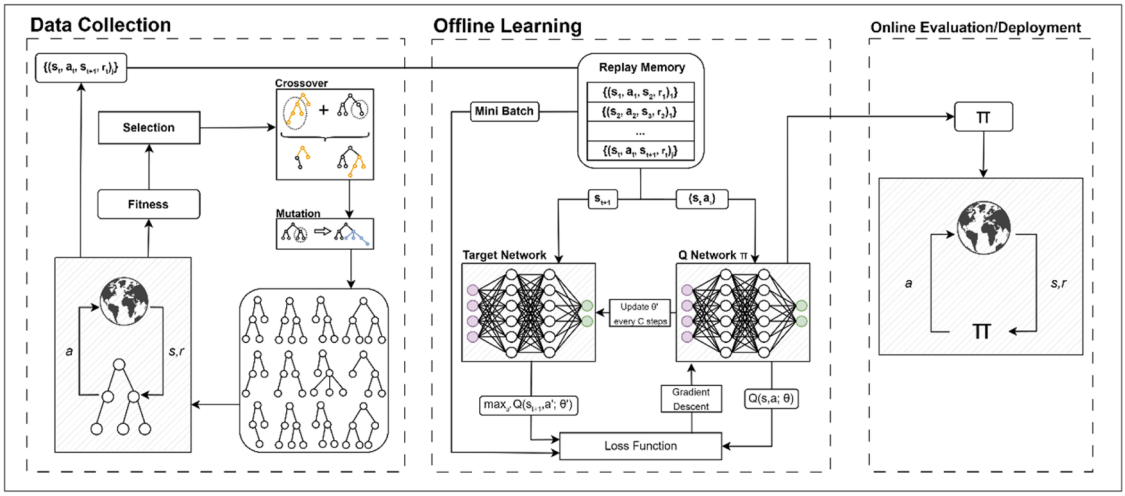


Fig. 3.1. Decoupled evolutionary offline RL exemplified by GP for data collection and DQN for offline learning. A fully sized version can be found in Appendix F.

within set depth bounds). This cycle repeats until 100,000 trajectories are collected, saved as transitions $D = \left\{(s_t, a_t, s_{t+1}, r_t)_j\right\}$ in the replay buffer.

GP uses standard syntax tree representation with a seven-part function set. Full initialization ensures all solutions are of equal size within set bounds [54], to ensure sufficient complexity despite the short evolutionary period. Evolutionary operators and hyperparameters are specified in Appendix C.1.

For data collection using DQN the replay buffer is initially pre-populated using a random policy. The agent then interacts with the environment for 100,000 steps, updating the policy at each training step. Experience tuples, consisting of state, action, reward, and next state, are stored in the replay buffer, which has a size of 50,000. During training, an epsilon-greedy policy is employed to balance exploration and exploitation. The epsilon value starts at ϵ_init (1) and decays linearly over a set period (1,000 steps) towards the minimum epsilon value (ϵ_min = 0.01), after which it remains fixed. This decay ensures that the agent explores initially but gradually shifts to exploitation as it learns. The Adam optimizer is used with a learning rate of 0.0001, and the Huber loss with $\lambda = 1$ is applied for robust performance. Target updates occur every 100 steps, and the network is updated after each training step. The entire training process aims to generate a diverse and informative dataset for offline reinforcement learning.

The data collected can then be used by any off-policy or offline RL algorithm, given that it uses a replay buffer. For clarity, the learning process is explained using a DQN [55]. The DQN consists of two identical ANNs: a main and a target network. A minibatch of transitions is sampled from the replay buffer (dataset D). Next, using the inherent target-network, a target is set following,

$$y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}, \tag{3.1}$$

where a' denotes the action which maximizes the Q-value and θ^- the weights of the target network. Using this target, the parameters of the main network are updated, performing a gradient descent step following the equation

$$\delta = \left(y_j - Q(s_j, a_j; \theta)\right)^2, \tag{3.2}$$

where θ are the network parameters and δ symbolizes the Temporal Difference Error. Every C-steps the main network's parameters are copied to the target network. This process continues until the stopping criterion of 500,000 training steps is reached, as commonly done in offline RL [6].

After training, the algorithm is tested online for 100 episodes to evaluate its performance. The exact experimental design for the evaluation will be explained in detail in Section 4.4.

3.2. Assessing GPs influence on dataset characteristics

To answer the first research question, the following procedure will be followed. Two experiments will be conducted in OpenAI's Classic Control RL environments: Cart Pole (CP) and Mountain Car (MC). The goal of CP is to balance a pole on a cart for 500 timesteps without it falling or the cart leaving the frame. The agent receives state information about the cart position, cart velocity, pole angle, and pole angular velocity. The action space consists of two discrete actions of either pushing the cart to the left or to the right. For every timestep that the pole is standing, the agent receives a reward of +1. The objective of MC is to get the car out of a valley to the end of a hill, using the car's position and velocity along the x-axis. "Actions are limited to accelerating left, right, or not at all. It is episodic, with episodes ending after 200 timesteps or when the goal is reached. For every timestep without reaching the goal, the agent receives a reward of −1 [56]. Further details about the environment can be found in Appendix B.

Following Schweighofer et al. [6], five different levels of datasets are created with the two collection algorithms respectively:

- **Random Dataset**: Collected by a random behavior policy and serves as a naïve baseline for data collection.
- **Expert (Fully) Dataset**: Collected with the fully trained/elite policy after training/evolution.
- **Mixed Dataset**: Generated using a mixture of the expert (20 %) and the random dataset (80 %).
- **Noisy Dataset**: Collected applying the fully trained/elite policy, that selects actions $\epsilon-$ greedy with $\epsilon = 0.2$.
- **Replay Dataset**: Data collected during the online training/evolutionary process. Consequently, this dataset consists of trajectories sampled by multiple policies.

The data collection process is run five independent times. After data collection is concluded, the dataset will be analyzed quantitatively and visually if the dimensions of the state-action space allow it. For the quantitative analysis the empirical measures TQ and SACo introduced by Schweighofer et al. [6] are applied.

TQ is a measure of the normalized average trajectory return and aims to quantify how exploitative the behavioral policy is. To do so, the expected trajectory return is estimated by the average return of the tra-

jectories in the collected data,

$$\overline{g}(D) = \frac{1}{B}\sum_{b=0}^{B}\sum_{t=0}^{T_b}\gamma^t r_{b,t} \tag{3.3}$$

for B trajectories with their respective lengths T_b. To allow comparisons between MDPs, the average trajectory return is normalized with the best and the worst behavior observed in the same MDP. Consequently, the quality of the analyzed trajectories is relative to the comparative datasets used. Accordingly, the equation for TQ is:

$$TQ(D) := \frac{\overline{g}(D) - \overline{g}(D_{\min})}{\overline{g}(D_{\text{expert}}) - \overline{g}(D_{\min})}. \tag{3.4}$$

In practice D_{min} is a dataset collected by a random policy and D_{expert} a dataset collected by the expert policy. For the conducted experiments the D_{expert} will be the return generated via the DQN online training. The expert and random datasets will remain fixed throughout the experiments for comparability.

SACo aims to measure the level of exploration of a behavior policy. It is based on the number of unique state action pairs $u_{s,a}(D)$ given in a Dataset D, since it directly corresponds to the exploration capabilities of the policies analyzed. For SACo, the unique state-action pairs are normalized with the unique state-action pairs of a reference dataset D_{ref}. This reference dataset was collected of the same MDP and is of the same size as D. The equation of SACo is composed as follows:

$$\text{SACo}(D) := \frac{u_{s,a}(D)}{u_{s,a}(D_{\text{ref}})}\ . \tag{3.5}$$

For the reference dataset D_{ref} it is sensible to take the replay dataset, since it is assumed to have the highest diversity in terms of state-action pairs, as it was collected by multiple policies. For these experiments, the replay dataset of the online training of the DQN will be used and fixed throughout the experiments.

Due to time and memory constraints, the probabilistic counting method HyperLogLog [57] is used. For a detailed explanation and justification for this method, it is referred to Schweighofer et al. [6].

Both the individual outcomes from the five independent runs and the aggregated results, for instance, across different environments, will be analyzed. Following the evaluation methodology for RL proposed by Agarwal et al. [58], the Interquartile Mean (IQM) is calculated as the aggregate metric, since it is more statistically efficient than the median but still robust to outliers. To give an indication of variability between runs, either the interquartile range (IQR) or the standard deviation are analyzed additionally.

3.3. Comparative analysis of offline RL algorithm performance

The initial experiments will address the second research question by training four offline algorithms using datasets from the first experiment. The goal is to evaluate the effectiveness of different data collection methods and determine which algorithms work best with specific datasets. The following offline learning algorithms are implemented in this study.

Behavior Cloning (BC) [10]: This algorithm is trained on the collected dataset, directly mapping states to actions as seen in the data. The ANN predicts actions based on input states and adjusts its parameters using Cross Entropy Loss and backpropagation, serving as a baseline by mimicking the collection algorithm.

Deep Q-Network (DQN) [55]: Trained offline using the dataset as a replay buffer, DQN is evaluated for its performance when trained on data generated by evolutionary techniques, despite being designed as an online RL method.

Random Ensemble Mixture (REM) [59]: This Q-learning algorithm uses multiple action-value approximations, weighted through a randomly drawn categorical distribution. Action selection is based on the average of these approximations. REM has shown strong performance when trained on DQN replay datasets, making it an interesting candidate for datasets collected by GP.

Conservative Q-Learning (CQL) [13]: CQL addresses value overestimation from distributional shifts by adding a regularization term. It has shown substantial improvements over other offline RL methods. For this work, a DQN algorithm is used in the CQL framework.

During training, learning curves for each algorithm will be collected by interacting with the online environment every 200 iterations. Further details on offline training and evaluation can be found in Appendix C.2.

After training, all algorithms will be evaluated, and performance differences will be tested for statistical significance. Following Colas et al. [60], 100 evaluation episodes will be conducted independently of training, without noise. Statistical tests will assess whether performance differences exist between algorithms trained with data collected by GP or DQN.

To choose an appropriate test, assumptions are assessed based on the datasets collected. Thirty-two individual tests will be conducted, holding factors (environment, collection mode, and offline algorithm) fixed (ceteris paribus) to compare only the collection algorithms (GP and DQN). Two-tailed tests will first be used to check for significant differences in mean performance, followed by one-tailed tests to confirm the set hypothesis.

However, this test design has an inherent limitation. When comparing multiple pairs of algorithms in the same setting, the likelihood of encountering at least one false positive increase linearly with the number of comparisons made [60]. In order to adjust for this issue the p-values are corrected with the Bonferroni-Holm Method [61], prior to rejecting or accepting the Null-Hypotheses.

4. Results and discussion

4.1. Effect of GP on the exploration-exploitation balance

The analysis of TQ-value distribution across environments reveals several noteworthy observations. The replay datasets demonstrate similarity, although with a higher IQR observed for the GP-collected datasets. This indicates an increased variability between the runs. In contrast to the DQN, the noisy GP approach does not show any outliers, thus a higher median but a lower mean value. Despite the presence of one outlier, the expert dataset generated by GP exhibits considerably less variability, as well as a higher mean and median value compared to DQN. This indicates that the expert policies derived from GP are more consistent and achieve higher exploitation on average (Fig. 4.1).

Inspecting the values for SACo, there is a noteworthy difference between the replay datasets. As the DQN's number of unique state-action pairs acts as the reference value (taken from run one), its values are close to one. In comparison, the GP exploitation performance is notably worse, with a mean value of 0.44, indicating it finds less than half the unique state-action pairs compared to DQN. For the noisy datasets, a different pattern emerges: the boxplots show similar distributions, but the GP datasets have higher variability and slightly lower mean and median values compared to DQN. The expert dataset reveals a low IQR for the DQN at the lower end of the SACo range but includes several outliers, indicating uncertainty. The GP algorithm's distribution is similar, without any values outside the IQR (Fig. 4.1).

Mapping the IQM values of the metrics applied across the environments unveils multiple insights. In the CP environment, the mixed approach produces TQ and SACo values close to the random dataset, indicating it is unsuitable for this problem. GP achieves a TQ of almost one for the expert dataset, surpassing DQN by approximately 0.3 points with a comparable SACo level. The GP's noisy dataset shows weaker exploitation but similar exploration to the RL algorithm. In contrast, for the replay dataset, the TQ of both approaches is similar, but GP's exploration is almost 0.6 points lower than the RL approach (Fig. 4.2).

The results for the MC environment are mostly similar. The main

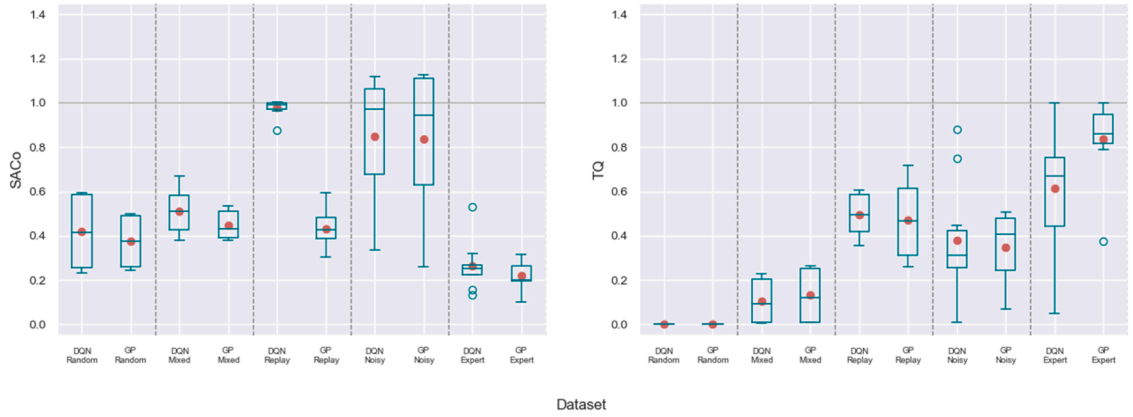


Fig. 4.1. Distribution of TQ and SACo over all environments and runs, split by the collection paradigm and mode. The red dot represents the mean value. The boxplots are based on the study provided by Schweighofer et al. [6].

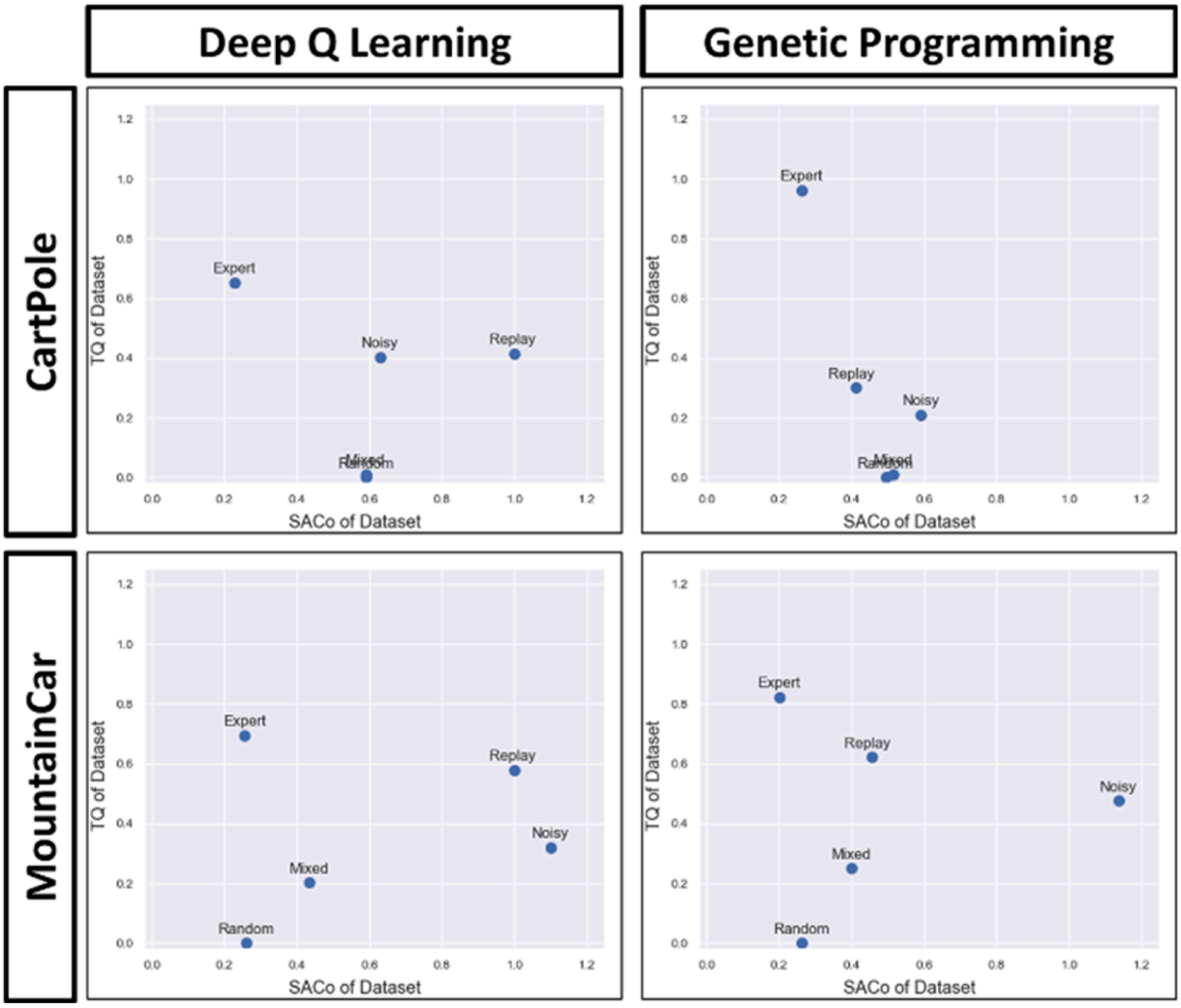


Fig. 4.2. TQ and SACo for CP and MC divided by DQN and GP. The Metrics are calculated across 5 independent runs taking the IQM as an aggregate. The graphs are based on the study provided by Schweighofer et al. [6].

differences are shown by the mixed datasets, which show a higher value of TQ independent of the collection algorithm and the noisy datasets, which both show an exceptionally high level of SACo and for the GP approach also a medium high level of TQ (Fig. 4.2).

The visualization of the state-action space of the MC environment confirms these results. The high level of exploration achieved by the DQN replay dataset can clearly be seen by the various actions taken across the state-space, while the GP approach shows a closer resemblance to the expert dataset. This resemblance can also be seen in the noisy GP dataset, while a very high variety of actions taken is visible.

The high level of TQ and low level of SACo in the expert sets, especially in the generated by GP, can be seen clearly since the actions taken for a given state are quite homogenous and many states are not reached at all (Fig. 4.3).

While some of the reported results aligned with our expectations, others were unanticipated. As established before, EAs are very stable and have very good convergence properties [35], which explains the low variability and high TQ of the data collected by the fully trained policy. The low performance in terms of exploration of the GP collection, on the other hand, is surprising, since the high exploration capabilities of EAs are one of their main advantages. A possible explanation of this outcome is the applied experimental design. The two factors that foster exploration are the diversity within the population and the gradient free random search [20], conducted through the evolutionary process respectively the GOs. Since this process was interrupted after 100,000 steps these two factors may not have been able to fully unfold. To enable the variation over generations, the population had to be kept comparatively small (40 individuals), since otherwise the trajectories would have been generated within the first generations. Nevertheless, also with this small population the evolutionary process only lasted for approximately 20 generations, depending on the problem and the performance of the solutions. For example, the MC episodes get shorter with increasing performance, while the CP episode length is positively correlated to performance.

Nevertheless, the results show the effectiveness of EAs in the context of data collection. Even though it was limited to some extent due to the experimental design, the GP algorithm produces trajectories of very high quality, with a very low variability. This emphasizes the stableness and reliability of the method, which is especially important in the context of RL since often results can be very hard to reproduce, due to non-determinism in the benchmarks and variance intrinsic to the methods [62]. However, it must be taken into account that this stableness could also be amplified by the environments used, since they are comparatively simple.

4.2. Effect of GP vs. DQN generated data on offline learners

For the third research question, first the learning curves will be analyzed, connecting their progressions to the algorithms and dataset characteristics. Next the performance of the best policy during offline training will be set into relation to the DQN online performance, the collection algorithm and mode as well as its TQ and SACo scores. Finally, the performance data collected after the training will be analyzed and tested for statistically significant differences.

Examining the replay learning curves for the CP environment, there are several noteworthy observations. For algorithms trained with the DQN-collected data, one can see that there is progress over time, whereas the algorithms trained with the GP data are either stable or not showing improvement over time. The picture for the GP trained algorithms is the same for the MC environment. It shows mediocre but stable performance. However, here the algorithms trained on the DQN collected data show the same picture. In both environments REM is an exception, as it shows worse performance over time (Fig. 4.4).

The curves connected to the expert dataset draw a very different picture. Especially the BC and CQL algorithms trained with GP collected data show top performance in the CP environment over the whole time of training. Moreover, the offline-DQN shows a steep learning curve over time when trained with GP data. The performance of the algorithms with the DQN data are comparatively low and show no increase over time. The curves for the MC environment are like the curves seen for the replay dataset. Mediocre and stable performance, with the performance of the GP-algorithms higher on average. Furthermore, the REM algorithm seemed to be not able to learn from the DQN expert data, showing minimal performance over time (Fig. 4.4).

For the CP environment, CQL was the only learner which showed a clear learning trend using the GP dataset generated via the mixed approach. The other algorithms show very little and unstable learning over time. For the MC environment there is no clear learning trend visible for all algorithms and datasets. Finally, the noisy datasets were used to train the offline algorithms. For the CP environment the GP trained offline DQN and the DQN trained REM show a very steep learning curve over time. Furthermore, CQL was not able to learn using noisy data regardless of the collector. On the MC environment the offline algorithms could not increase their performance overtime using the noisy data. While they are all showing relatively high performance using the DQN data, the REM algorithm seems to have unlearned the task over time dropping down to the minimal reward after 200,000 training steps (Fig. 4.5).

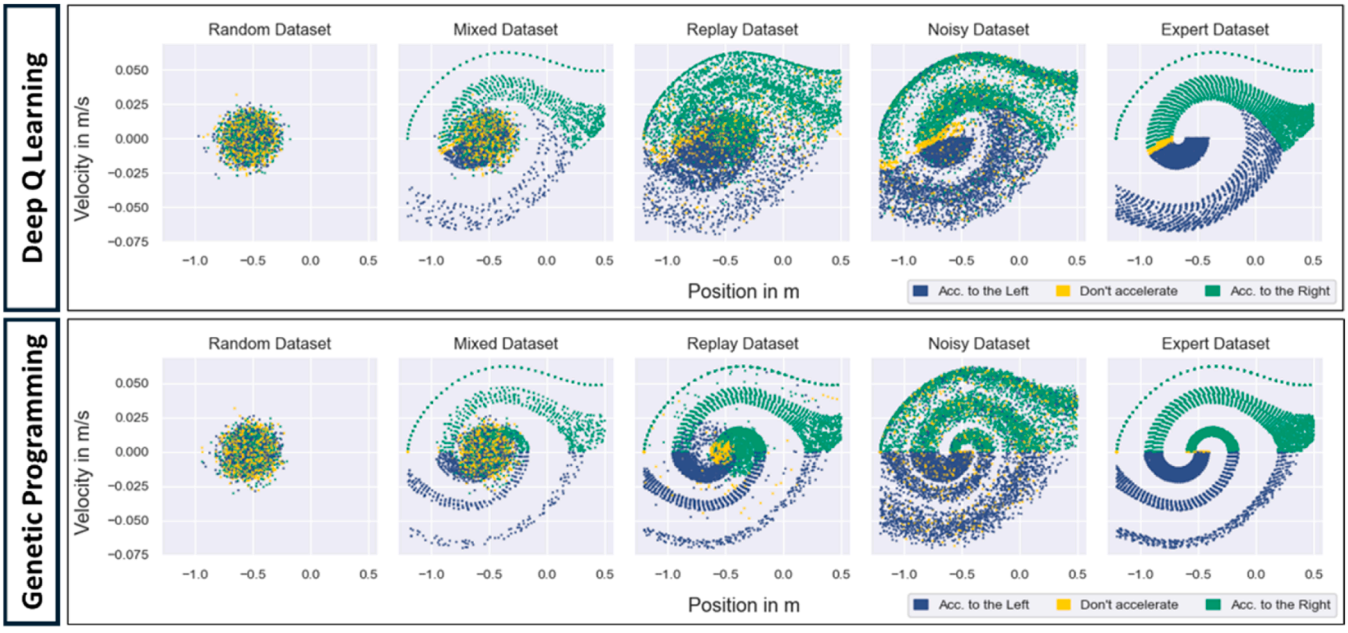


Fig. 4.3. Effect of the different behavioral policies and collection modes on the MC environment. The visualizations of all the independent runs can be found in Appendix E.1. The graphs are inspired by the visualization provided by Schweighofer et al. [6].

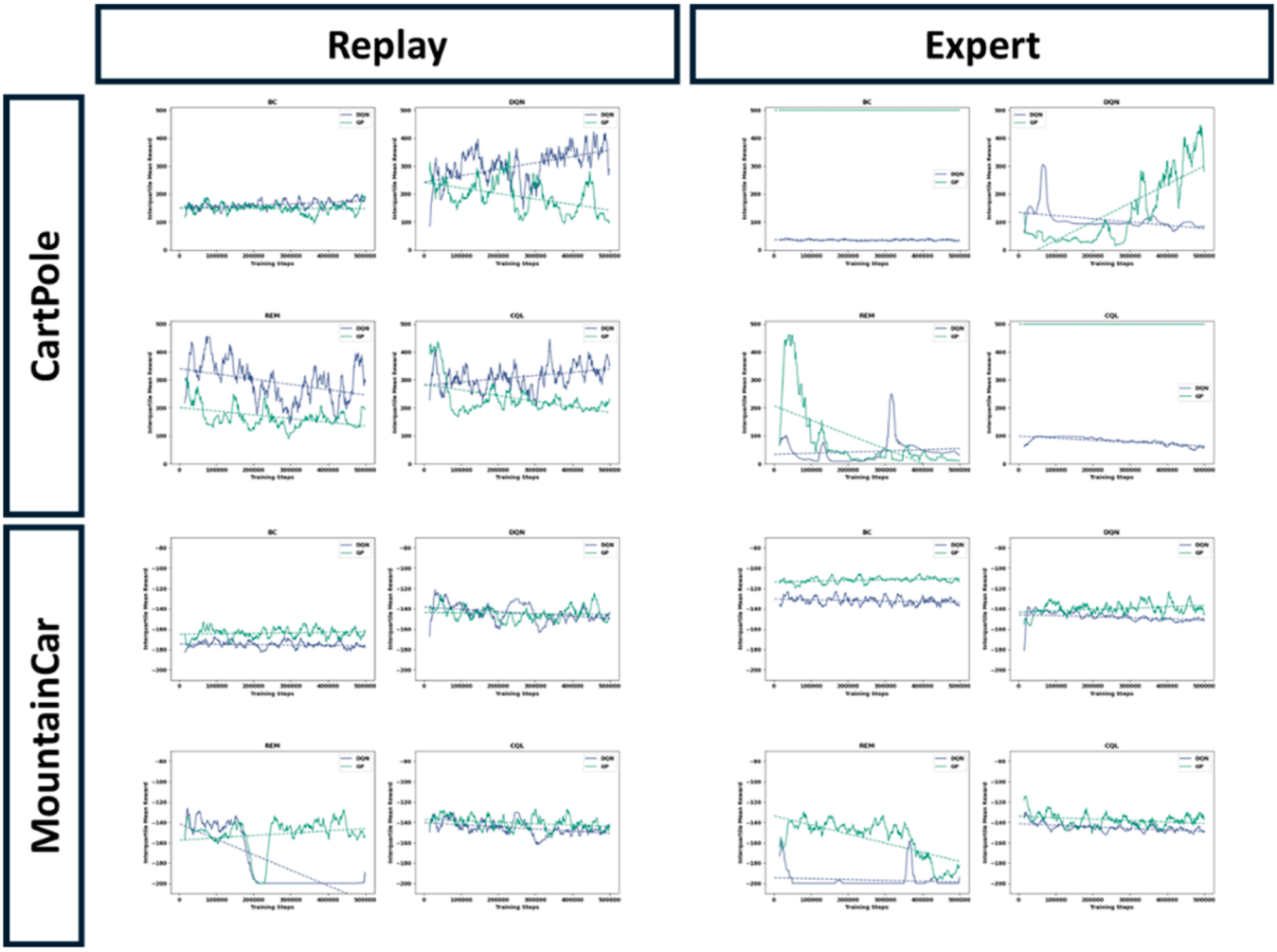


Fig. 4.4. Learning curves for the offline algorithms BC, DQN, REM, and CQL, displayed for the environments Cart Pole (CP) and Mountain Car (MC), and for the data collection modes Replay and Expert. The curves are color-coded by the collection algorithm (blue for DQN and green for GP). The curves are smoothed using a moving average over 15 points.

Looking at the performance of the offline algorithms in the context of the TQ and SACo scores corroborates the findings presented before. BC shows a positive correlation with TQ, since it is very close to a supervised learning approach. The high performances of CQL and BC in the CP environment are explained by the extremely high TQ of the expert GP dataset. Other than that, the performances are comparable, and no clear advantage is visible via this perspective (Fig. 4.6).

Fig. 4.7 compares the final performances of the offline RL Algorithms trained on CP datasets. Apart from CQL, the algorithms trained with the replay data generated by the DQN clearly outperformed the ones trained with the GP datasets. For the expert datasets this picture is inversed. Here the algorithms trained with the GP datasets show peak performances, while the algorithms trained with the DQN datasets show very low values. As seen in the learning curves, REM was not able to learn from the data generated in this case (Fig. 4.5). In the mixed section only CQL trained on DQN data showed good results on average. CQL is also the odd one out for the noisy CP data. However, here it is the only algorithm which underperformed when trained on GP data. The other learners showed superior performance compared to the algorithms trained with the DQN generated experiences.

The learners trained with the MC replay data show an opposite picture. The algorithms trained on the GP generated trajectories show a higher performance after offline training. Merely the offline DQN performed slightly better with data generated by its online version. The expert cohort is also dominated by learners trained on GP data, with REM being the only exception. It is especially notable that BC showed the highest performance globally trained on the mentioned data. The mixed data only showed a difference for the REM learner trained with GP data, outperforming the other methods. Finally, the noisy generated datasets for the MC environment enabled the offline algorithms trained on DQN experiences to outperform its counterparts. Once again REM is the sole exception, performing higher trained on noisy GP data (Fig. 4.8).

In the final analysis, the central performances of algorithms trained with GP and DQN-generated experiences are compared to identify statistical differences. To choose a suitable statistical test, different assumptions are addressed. First, the samples collected are all the same size ($N = 100$). Second, Colas et al. [60] point out that it is reasonable to assume that RL performances are measured independently and at random. Next, the performance data was tested for normality using the Shapiro-Wilk test [63] and for equal variances using Levene's test [64]. Following these results and the findings of Colas et al. [60], the Welch's t-test [65] is applied with a low confidence level ($\alpha = 0.025$), as it is shown to be robust even under non-normal conditions.

Effect sizes were calculated using Cohen's d, which measures the standardized difference between two means [66]. It is important to note that this study focuses on detecting large effect sizes due to computational constraints, meaning smaller effects might not be detected with the given sample size. This is particularly relevant since a wide range of effect sizes were identified. Detailed results and test statistics are

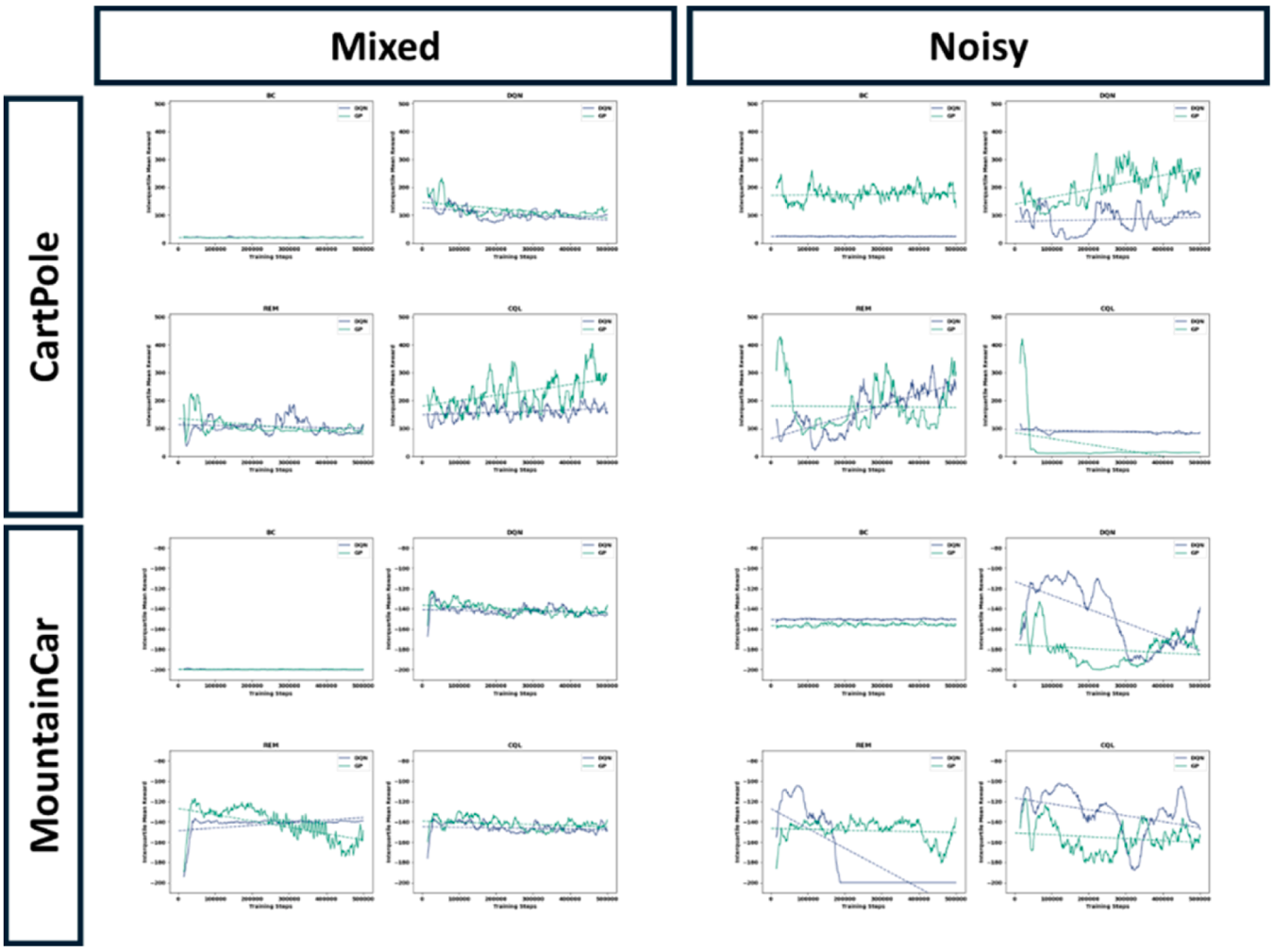


Fig. 4.5. Learning curves for the offline algorithms BC, DQN, REM, and CQL, displayed for the environments Cart Pole (CP) and Mountain Car (MC), and for the data collection modes Mixed and Noisy. The curves are color-coded by the collection algorithm (blue for DQN and green for GP). The curves are smoothed using a moving average over 15 points.

provided in Appendix E.2.

After selecting Welch's *t*-test, hypotheses are formulated. The two-tailed test checks whether the mean performance differs between GP- and DQN-trained learners. The one-tailed test assesses if GP-trained learners outperform DQN-trained ones.

Examining the outcomes, several key observations can be made: Out of the 32 two-tailed tests conducted, 21 showed a significant difference in the mean performance after correction. Specifically, 13 (62 %) of these tests displayed a significantly higher central performance for learners trained with GP collected data, according to the one-tailed Welchś *t*-test (Table 4.1). The algorithms mostly profited from the modes expert and noisy when trained on GP datasets. With four and five significant one-tailed tests BC and REM showed the comparatively highest performance with GP collected data. Non-significant performance disparities were predominantly observed in the MC environment. Despite this, the distribution of significant tests favoring the GP collection was relatively balanced across different environments. Conversely, 75 % of the tests that indicated higher performance for learners trained with DQN-collected data were associated with the CP environment.

Considering all findings from the descriptive analysis of learning curves, the relative performance during training, the performance data collected after training, and the results of the statistical tests, the outcomes are consistent and aligned. Also taking the results of the earlier experiments concerning the characteristics of the collected dataset into account a clear picture is drawn. The high TQ witnessed in data generated via GP enabled algorithms that were not designed for offline training (BC and DQN) to effectively learn, regardless of the environment. Even though the explorative power of GP was below expectations, learners for which Schweighofer et al. [6] concluded that they need a high value of SACo, like REM and DQN showed effective offline learning. Especially, if the exploration was enhanced by the collection mode (noisy, mixed).

However, these findings are not all exclusive for data collection via GP. Especially the replay datasets of the DQN showed superior results than the same ones generated by GP. Nevertheless, one clear difference can be witnessed. While the DQN data was mostly successfully used for training in the CP environment, the algorithms using the GP generated trajectories and surpassing its counterparts were spread across the environments used. This might be related to the indifference of the distribution of rewards seen in EAs [9]. That could also be one of the reasons that there are more instances where the learner trained on GP generated trajectories outperforms its counterpart with statistical significance (67 %). While these 67 % represent only a simple majority and does not provide conclusive evidence for the superiority of EAs in the realm of data collection for RL, it does indicate that GP is, at the very least, a competitive method in this domain.

These results need to be assessed critically. First and foremost, because the offline algorithms could only be run once on one collected dataset. Consequently, there is a chance that the offline learner shows a below average performance for this run. Moreover, as discussed before,

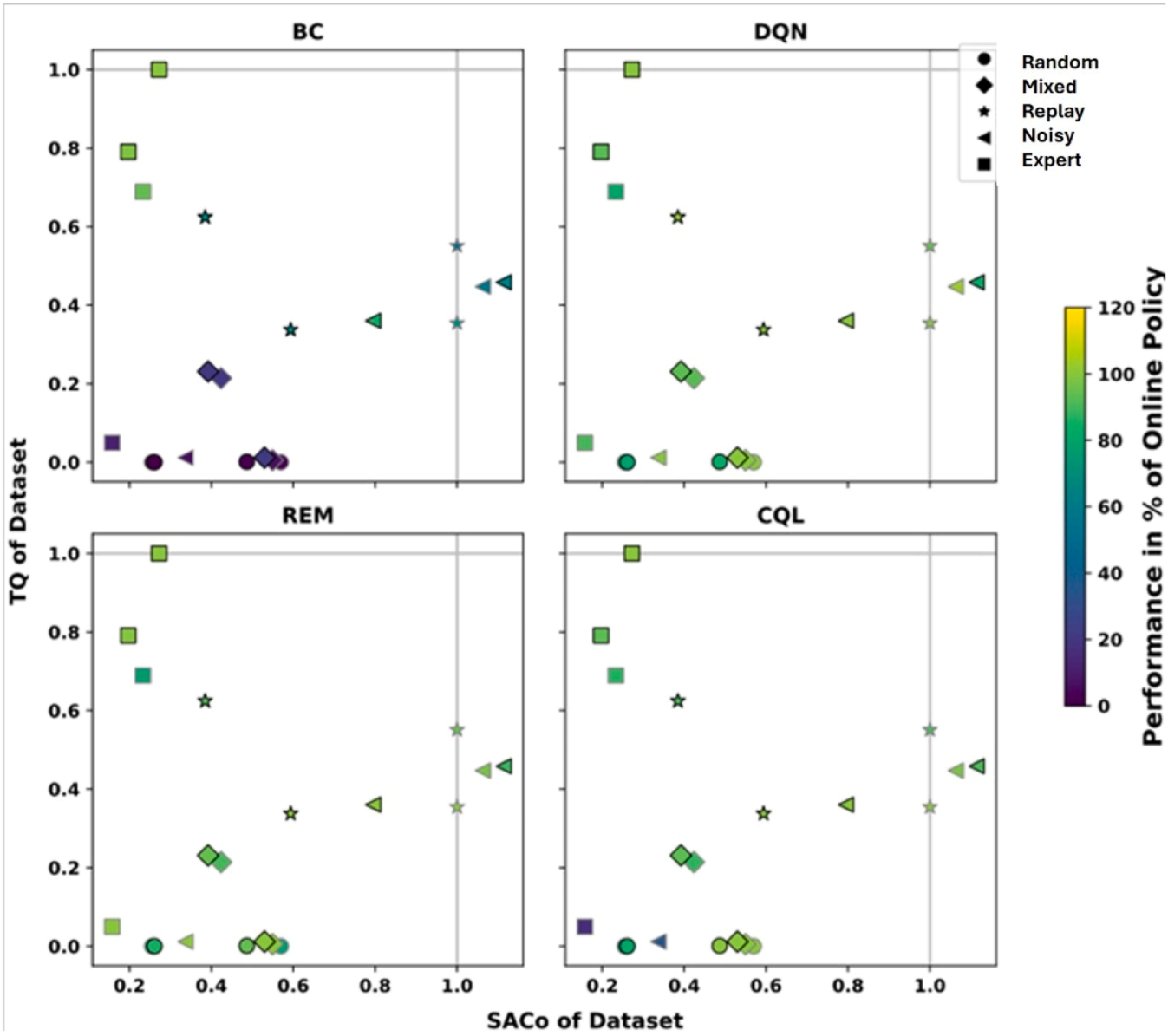


Fig. 4.6. Each point denotes one of the ten datasets used for training. The performance of the highest performing policy found in offline training is shown by the color of each point. It is related to the highest performance seen in DQN online training per environment. This is held fixed for both collection methods, to make performances comparable. The GP collected data is marked by a black border and the DQN one by a grey border. The graph is inspired by the analysis of Schweighofer et al. [6].

the sample size used for the statistical tests is relatively small, especially for samples with very small effect sizes. Finally, the number and the level of complexity of the environments used is too small to make final claims about the superiority of any method. Nevertheless, despite these limitations, the results establish the need for more extensive research in future studies and warrant the application of EAs in offline RL.

5. Conclusions

This study aimed to investigate the effect of GP on dataset characteristics in RL and if it improves the performance of offline and off-policy RL learners when used for data collection.

To answer these questions, a comparative approach was followed. By comparing a GP with a DQN across multiple collection modes and environments, insightful data could be collected for both learners to answer how the use of GP in data collection influences the explorational and exploitational power of the dataset in comparison to a DQN approach.

To determine to what extent off-policy and offline RL algorithms benefit from datasets generated by GP compared to those collected via DQN, various collected experiences were evaluated across multiple offline learners, providing a comprehensive view of the effects while accounting for potential harmonic interactions between the collector and the learner. By analyzing the results of the different experiments taking various dimensions, variables and previous research into account, we were enabled to make clear conclusions about the effects of GP in the given domain. Finally, statistical difference testing was applied to achieve an uncluttered and unbiased view regarding the dominance of a method.

Based on the findings of previous studies, it was expected to see very high exploratory capabilities by the GP, coupled with stable results over different trial runs. Consequently, We hypothesized that offline learners would achieve superior performance when trained on experiences generated by GP in comparison to those generated by DQN. While the

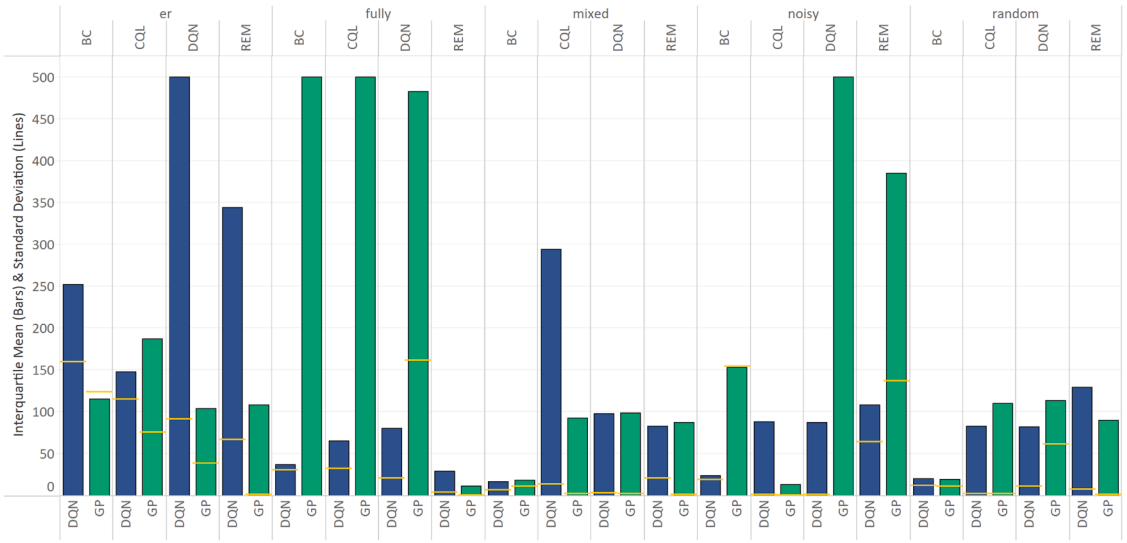


Fig. 4.7. Performance of the best-found policy during offline training using CP data. The policies have collected rewards for 100 episodes greedily after being trained for 500,000 steps. The bar plots (green and blue) show the IQM of the rewards and the lines (yellow) the standard deviation. A fully sized version can be found in Appendix F.

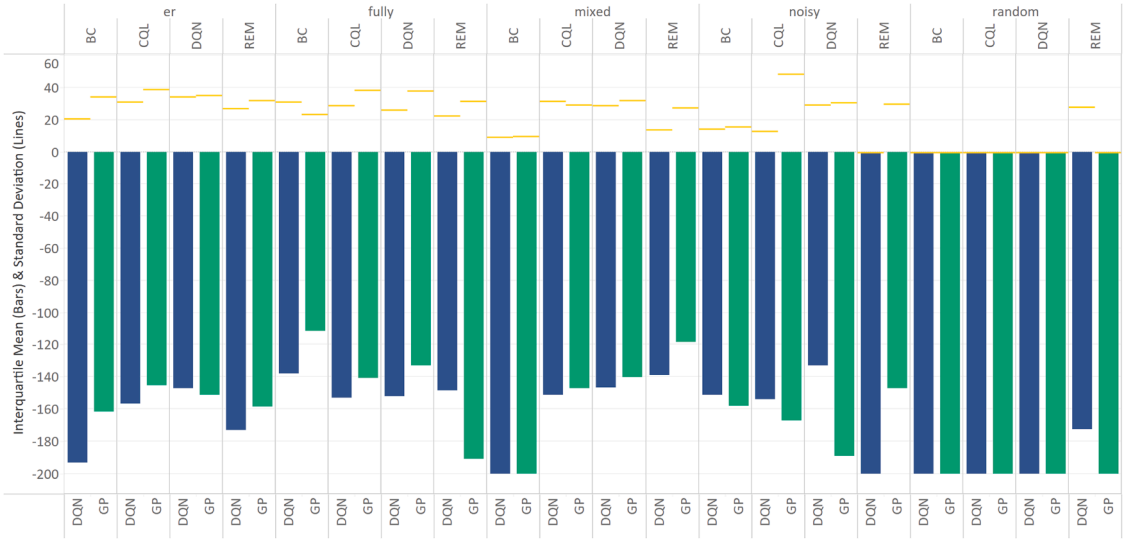


Fig. 4.8. Performance of the best-found policy during offline training using MC data. The policies have collected rewards for 100 episodes greedily after being trained for 500,000 steps. The bar plots (green and blue) show the IQM of the rewards and the lines (yellow) the standard deviation. A fully sized version can be found in Appendix F.

exploratory scores did fall behind expectations, others were confirmed. In comparison to the DQN approach, GP showed less uncertainty in generating high quality trajectories. Furthermore, out of all statistical tests conducted across multiple modes, environments and offline learners, which showed a significant difference, the final performance was higher for the learners trained on GP generated data in 62 % of the cases.

These findings do not only address the identified research gap, combining a decoupled approach with EAs and offline RL, but also open new avenues for research. Building on the frameworks of Schweighofer et al. [6] and Riedmiller et al. [8], it could be shown that GP can surpass other methods in generating data for RL in an effective matter, showing robust exploitation and exploration capabilities.

This study provides empirical evidence, showing the efficacy of GP in the context of data-centric RL. Furthermore, this study successfully adapted the Collect and Infer framework employing an EA as a collector. Overall, this work has demonstrated the potential of GP in the domain of RL data collection and provided a solid foundation for future research to build upon.

6. Limitations and future works

The clearest limitation of this work was computational resources. Without access to additional GPUs and memory the experiments ran for multiple days. Especially the offline training took over 200 h to complete, which limited the offline training to a single run per collection mode. This also limited the choice of environments, since more complex RL problems are also more computationally heavy. That was shown by Schweighofer et al. [6], since their executed experiments on more sophisticated problems took 20 days to run, even though 27 GPUs were used.

Another limitation, that was also influenced by the high demand for computational resources, was the experimental design chosen for the collection of the data. To make the collection between the DQN and the

Table 4.1
Results for the one tailed Welch's *t*-test and the Bonferroni-Holm correction. Here only the tests which showed statistical significance for the one-tailed test, after the correction are displayed. For the results of all the *t*- and preliminary tests please refer to Appendix E.2.

Nr	Test (One tailed Welch`s t)	t-scores	p-value	Corrected p-value
1	('CP', 'noisy', 'DQN')	1885.333	1.324E-227	4.237E-226
2	('CP', 'fully', 'BC')	141.086	3.030E-116	9.392E-115
3	('CP', 'fully', 'CQL')	128.359	3.343E-112	1.003E-110
4	('MC', 'noisy', 'REM')	20.700	4.788E-38	1.389E-36
5	('CP', 'fully', 'DQN')	19.776	4.171E-37	1.168E-35
6	('CP', 'noisy', 'REM')	13.505	1.804E-27	4.871E-26
7	('CP', 'noisy', 'BC')	9.770	1.265E-16	3.290E-15
8	('MC', 'er', 'BC')	6.726	1.426E-10	3.566E-09
9	('CP', 'mixed', 'REM')	6.331	3.313E-09	7.951E-08
10	('MC', 'fully', 'BC')	4.689	2.664E-06	6.127E-05
11	('MC', 'fully', 'DQN')	4.153	2.551E-05	5.613E-04
12	('MC', 'er', 'REM')	3.647	1.708E-04	3.588E-03
13	('MC', 'mixed', 'REM')	3.213	8.082E-04	1.616E-02

GP algorithm comparable, each algorithm only got to interact with the environment exactly 100,000 times, which results in the same number of collected tuples. The consequences of this have been discussed at the end of Section 5.1. A possible solution for this would be to increase the number of interactions for both collection algorithms. However, this was not feasible given the overall methodology and the hardware at hand since the datasets were collected using five different modes (including a random dataset), and each being run five times independently.

For future work it would be useful to test this approach not only for a higher number of generations but also in more complex environments. MDPs with sparse rewards could be particularly interesting, since EAs have shown to handle them particularly [9].

Another study, which would be insightful, would be to compare different fitness functions for data collection. In this study to set the fitness, each individual only gets to interact with the environment once and the fitness is the sum of rewards. Other versions could include mechanisms to foster diversity, like seen in NS or QD algorithms, where the individual gets rewarded for exploring new states. The implementation of these algorithms in the methodology used would be appealing as well.

Finally, the implementation of more advanced GP algorithms like Cartesian GP [67] or Tangled Program Graphs [68] in the methodology used would be very interesting. Especially Tangled Program Graphs emerge as promising candidate for collection, since they have shown comparable performance to Deep Neural Networks, for a fraction of their computational costs [69]. Exploiting this characteristic could make data collection in RL not only more effective but also significantly more efficient.

Funding

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS.

Author contributions

David Halder: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Georgios Douzas:** Supervision. **Fernando Bacao:** Supervision.

All authors have read and agreed to the published version of the manuscript.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

David Halder reports administrative support and article publishing charges were provided by NOVA University Lisbon NOVA Information Management School. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Contents of the Appendix

A.1. List of Figures

Fig. B.1: Graphical interfaces of the environments used for the experiments conducted. Enumerating from left to right: MC, CP. xiv.
Fig. E.2: Effect of the different behavioral policies on the MC environment. Collected via DQN. xxi.
Fig. E.3: Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 1, Baseline). xxi.
Fig. E.4: Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 1–4). Top left: Experiment 1 (Baseline). Top right: Experiment 2. Bottom left: Experiment 3. Bottom right: Experiment 4. xxii.
Fig. E.5: Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 5–8). Top left: Experiment 5. Top right: Experiment 6. Bottom left: Experiment 7. Bottom right: Experiment 8. xxiii.

A.2. List of Tables

Table B.1: Characteristics of the MDPs applied. *Minimum or maximum expected return depends on starting state s_0. xiv.
Table C.2: Hyperparameters for the GP Algorithms applied. xvi.
Table C.3: Hyperparameters of the online RL learning/Data Collection xvii.
Table C.4: Hyperparameters for offline training. xviii.
Table D.5: Maximum return of the policy trained online. These values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed in bold. xix.
Table D.6: Average return of dataset trajectories per environment, collection algorithm and run. The values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed here in bold *gD*min. xix.
Table D.7: Unique state-action pairs per environment, collection algorithm and run. The values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed here in bold. xx.
Table E.8: The results for the preliminary statistical tests. The alpha level for the Levene test for equal variance and the Shapiro Wilk tests for

normality is $\alpha = 0.05$. xxiii.

Table E.9: Results for the two tailed Welchś *t*-test and the Bonferroni-Holm correction. Bold highlights statistical significance ($\alpha = 0.025$). The table is ordered by corrected p-value ascending. xxv.

Table E.10: Results for the one tailed Welchś *t*-test and the Bonferroni-Holm correction. Bold highlights statistical significance ($\alpha = 0.025$). The table is ordered by corrected p-value ascending. xxvi.

Appendix B. Environments

Both environments used are episodic, deterministic MDPs, with different dynamics in terms of states, actions, and rewards. Table B.1 shows their characteristics and traits like the dimensions of the state *dim(s)*, the number of legal actions $|A|$, the maximum length of an episode, after which it is either done or truncated, T_{max}, as well as the minimum and maximum expected return g_{min}, g_{max} of an episode [6,70]. An overview over the graphical user interfaces is given in Fig. B.1.

Table B.1
Characteristics of the MDPs applied. *Minimum or maximum expected return depends on starting state s_0.

Environment	*dim(s)*	$\|A\|$	T_{max}	g_{min}	g_{max}
CartPole-v1	4	2	500	9*	500
MountainCar-v0	2	3	200	−200	−90*

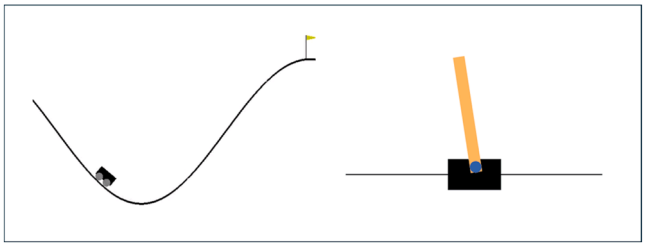


Fig. B.1. Graphical interfaces of the environments used for the experiments conducted. Enumerating from left to right: MC, CP.

Appendix C. Implementation details

C.1. Genetic programming

The GP algorithm is implemented using the Distributed EAs in Python (DEAP) library [71]. An overview of the hyperparameters can be found in Table C.2. The representation of the solutions are classic syntax trees, built out of a seven-fold function set, where 'lf' stands for the logistic function and 'sign' for a representation of the sign respective character. The division is protected against the division by zero like described in [72]. For the terminal nodes, ephemeral constants are used, that produce randomly generated numbers in the given range which remains constant for the tree but may vary between trees. The variation probabilities are set like described in [73], and the mutation step (ms) like in [74].

The GP algorithm is not stopped after reaching a certain fitness or number of generations but after 100,000 experience tuples are collected, to make the collection phase comparable to the DQN benchmark. A fixed size limit of 1,000 is implemented to make the experiments computationally feasible.

Table C.2
Hyperparameters for the GP algorithms applied.

Hyperparameter	Value
Representation	Syntax - Tree
Function set	{+, -, x, /, lf, sign, sine}
Terminals (ephemeral constants)	[−1, 1]
Population Size	40
Generations	-
Number of interactions	100,000
Evaluation Episodes	1
Tournament contestants (Fitness)	4
Tournament contestants (Size)	2
Mutation Probability	10 %
Maximum Mutation	5
Minimum Mutation	2
Size Limit	1,000
Crossover Probability	90 %
Mutation Step *(ms)*	0.001

After the evolutionary process, the highest performing policy is extracted and used to generate the datasets under the settings described.

C.2. Reinforcement learning

The RL algorithms in this work were applied using the implementations provided by [6]. The architecture of the ANNs is as follows: The input nodes of the ANN are dependent on the previously described state-space of the respective environment. Subsequently, three hidden linear layers are implemented with a respective size of 256 nodes. Lastly, the size of the output layer is defined by the action space of the environment. For REM, the output size is determined by multiplying the number of actions by the number of estimators. While the activation function for the last layer is linear, the other layers use the SELU activation function with proper initialization of weights. The BC algorithm applied is identical to the action-value networks but uses a softmax-activation in the last layer.

For data collection a DQN algorithm is used. For this, the replay buffer is pre-populated by a random policy and then trained for 100,000 interactions with the environment. In every training step, the policy is updated, and the experience tuple is saved into the replay buffer. Furthermore, an epsilon greedy policy is applied which starts at the value of the ϵ_{init}, linearly decays over a period of steps towards the ϵ_{min}, where it remains fixed after it was reached. Lastly, for all algorithms the Adam optimizer was used. The hyperparameters for the online training can be found in Table C.3.

Table C.3
Hyperparameters of the online RL learning/data collection.

Hyperparameter	Value
Algorithms	DQN
Learning rate	0.0001
Batch Size	32
Optimizer	Adam
Loss	Huber, $\lambda = 1$
Number of interactions	100,000
ϵ_{init}	1
ϵ_{min}	0.01
ϵ decay period	1.000
Target Update Frequency	100 steps
Network Update Frequency	1 step
Experience Replay Buffer size	50,000
Evaluation Frequency	200

Every 200 episodes the policy gets evaluated. It acts greedy based on the current value estimates for 10 episodes. The data is then saved for further analysis.

For the offline training, the hyperparameters remain identical to the online training, shown in Table C.3. The parameters which were altered for offline training or are algorithm specific can be found in Table C.4. The algorithm specific hyperparameters are relying on the parameters, provided by the original authors.

Out of the five independent dataset collection runs, the dataset from the first run is always chosen to train the offline RL algorithms, to make the results comparable, reproducible and not biased. By adhering to a predefined rule of using the first run, it is ensured that the data selection process is entirely objective and transparent. This approach precludes any accusations of selectively choosing data that might unduly favor the hypothesis.

Table C.4
Hyperparameters for offline training.

Algorithm	Hyperparameter	Value
All	Dataset number	1
All	Evaluation Frequency (steps)	1,000
All	Training Steps	500,000
All	Batch Size	128
REM	Number of estimators J	200
CQL	Temperature parameter α	0.1

The algorithms are evaluated for ten steps after 1,000 steps of offline training. These evaluations are then used for the analysis and to create the learning curves. The learning curves are once created using the mean performance and the standard deviation and once using the IQM. The experiences gained during evaluation are not used to train the algorithm. The maximum of the returns is compared to the performance of the benchmark DQN online algorithm, to put the results into perspective. After the offline training is concluded, the algorithms get evaluated for 100 more episodes, to assess the effect of data collection via GP on fully offline trained policies.

Appendix D. State Action Coverage (SACo) & Trajectory Quality (TQ) Calculations

Table D.5
Maximum return of the policy trained online. These values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed in bold.

Environment	Collection Alg.	$\overline{g}(D_{\text{expert}})$				
		Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
CartPole-v1	DQN	**500**	500	500	500	500
MountainCar-V0	DQN	**−97.2**	−105.8	−102.3	−94.1	−94.2

Table D.6
Average return of dataset trajectories per environment, collection algorithm and run. The values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed here in bold $\overline{\boldsymbol{g}}(\boldsymbol{D}_{\min})$.

Environment	Coll. Alg.	Dataset	$\overline{g}(D)$				
			Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
CartPole-v1	DQN	Random	**21.8**	22.2	22.0	22.4	22.3
		Mixed	24.3	27.1	27.2	27.0	27.5
		Replay	191.2	229.8	225.6	212.5	222.1
		Noisy	27.3	145.7	443.7	116.4	381.0
		Expert	45.6	202.6	500.0	132.6	500.0
	GP	Random	22.3	22.0	22.3	22.1	22.2
		Mixed	27.3	27.0	27.3	27.1	27.3
		Replay	183.1	223.0	165.7	145.6	148.7
		Noisy	194.0	154.8	80.3	55.7	133.4
		Expert	500.0	500.0	200.8	440.4	486.3
MountainCar-v0	DQN	Random	**−200.0**	−200.0	−200.0	−200.0	−200.0
		Mixed	−177.2	−176.1	−182.0	−183.1	−178.8
		Replay	−126.7	−124.8	−133.6	−136.2	−126.8
		Noisy	−152.4	−167.7	−164.4	−174.8	170.1
		Expert	−126.7	−124.8	−133.6	−136.2	−126.8
	GP	Random	−200.0	−200.0	−200.0	−200.0	−200.0
		Mixed	−175.4	−173.6	−173.2	−175.1	−174.8
		Replay	−133.6	−146.3	−140.7	−129.5	−135.5
		Noisy	−151.2	−152.5	−150.8	−151.7	−151.3
		Expert	−115.9	−115.9	−114.3	−116.5	−116.3

Table D.7
Unique state-action pairs per environment, collection algorithm and run. The values serving as the reference for the calculation of the indicators are held fixed throughout the analyses and are displayed here in bold.

Environment	Coll. Alg.	Dataset	$u_{s,a}\ (D)$				
			Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
CartPole-v1	DQN	Random	54,495	55,515	56,964	56,488	56,699
		Mixed	52,572	56,642	52,541	59,253	64,078
		Replay	**95,667**	96,031	94,251	95,541	94,432
		Noisy	32,224	64,466	50,141	65,348	85,479
		Expert	15,076	24,368	12,776	25,511	50,688
	GP	Random	46,534	46,721	47,149	47,499	47,616
		Mixed	50,686	51,127	49,772	46,407	44,298
		Replay	56,799	48,220	28,931	31,469	37,822
		Noisy	76,058	72,053	40,475	25,073	56,571
		Expert	26,049	30,067	26,871	22,260	9,607
MountainCar-v0	DQN	Random	3,217	3,283	3,265	2,936	3,136
		Mixed	5,350	5,482	5,987	4,773	5,272
		Replay	**12,628**	12,609	12,191	11,091	12,205
		Noisy	13,424	13,450	14,121	13,272	13,847
		Expert	2,941	3,359	4,028	2,828	3,179
	GP	Random	3,296	3,327	3,274	3,169	3,106
		Mixed	4,950	4,996	4,997	4,779	4,831
		Replay	4,857	6,114	6,046	5,028	5,809
		Noisy	14,066	13,863	13,797	14,211	14,142
		Expert	2,488	2,570	2,496	2,455	2,490

Appendix E. Results

E.1. Visualizations of the Sampled MC Trajectories

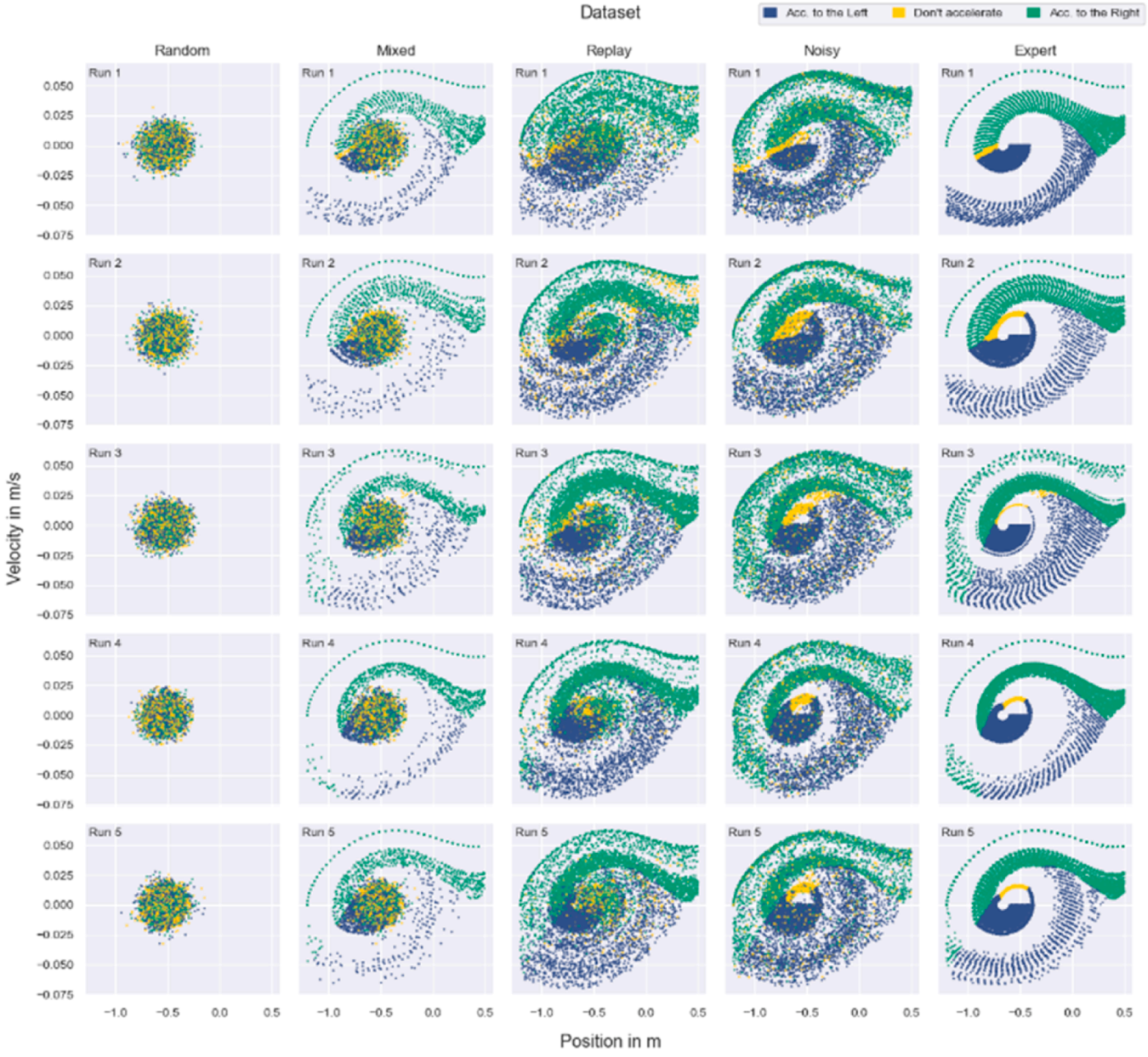


Fig. E.2. Effect of the different behavioral policies on the MC environment. Collected via DQN.

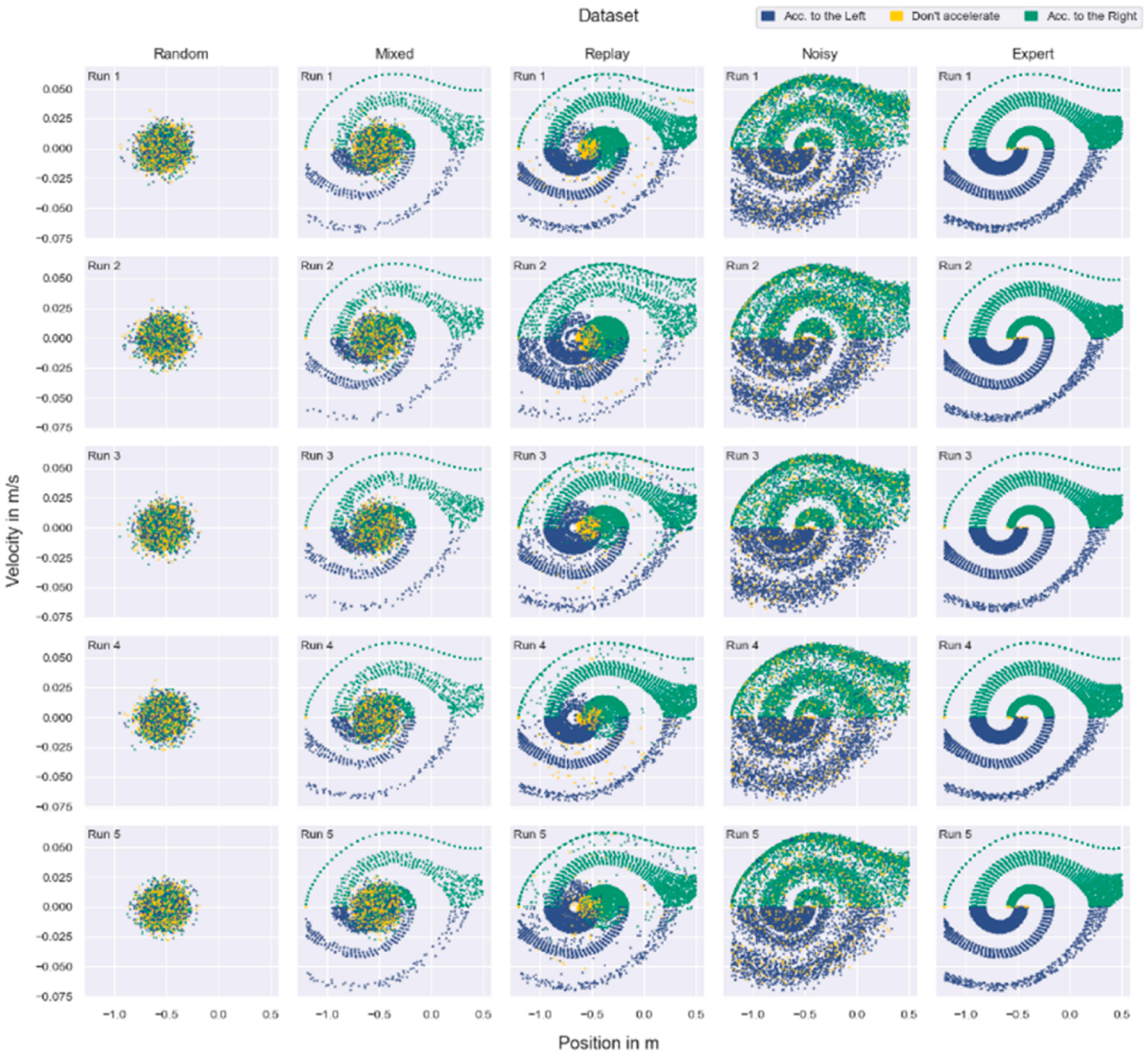


Fig. E.3. Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 1, Baseline).

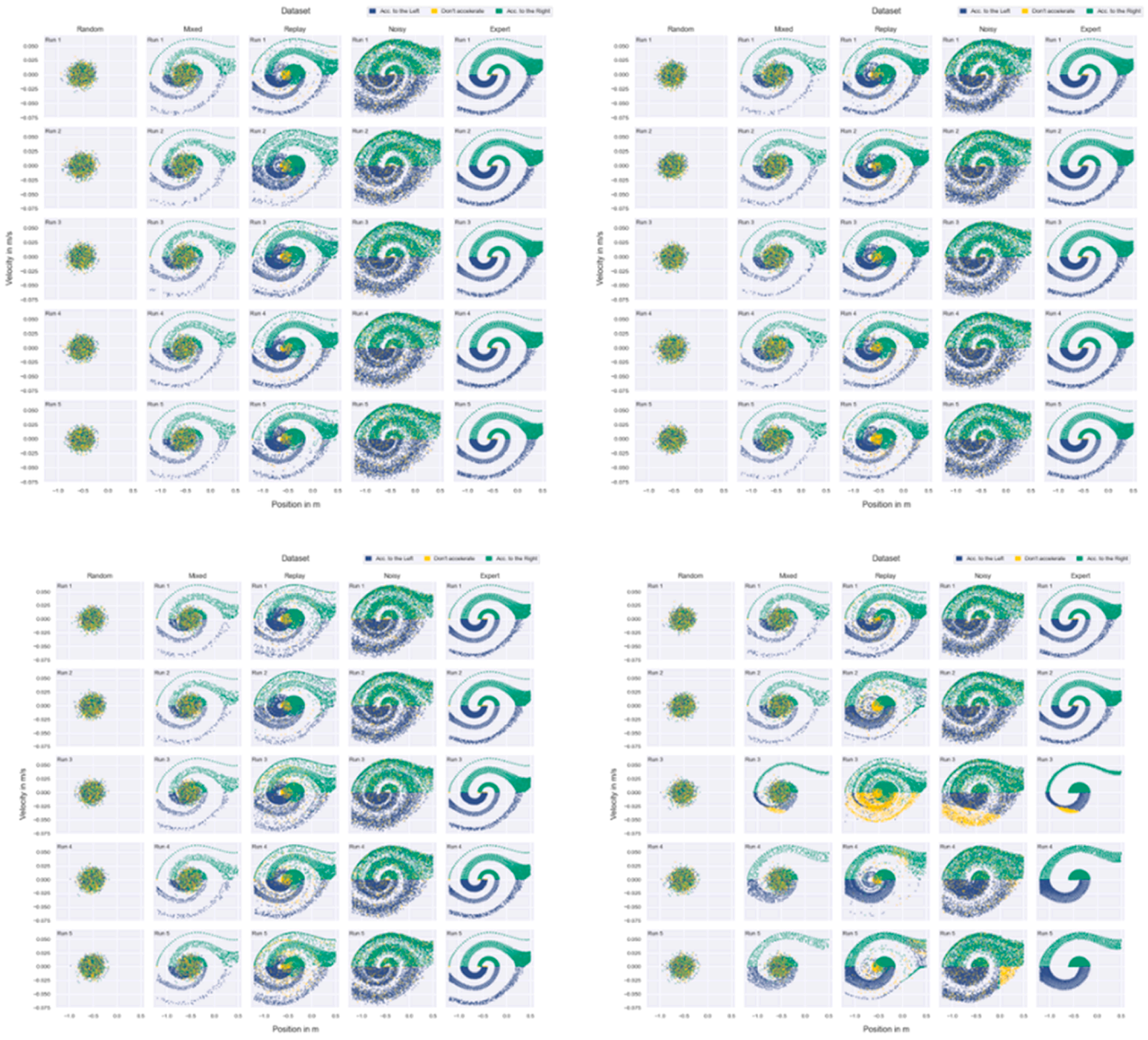


Fig. E.4. Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 1–4). Top left: Experiment 1 (Baseline). Top right: Experiment 2. Bottom left: Experiment 3. Bottom right: Experiment 4.

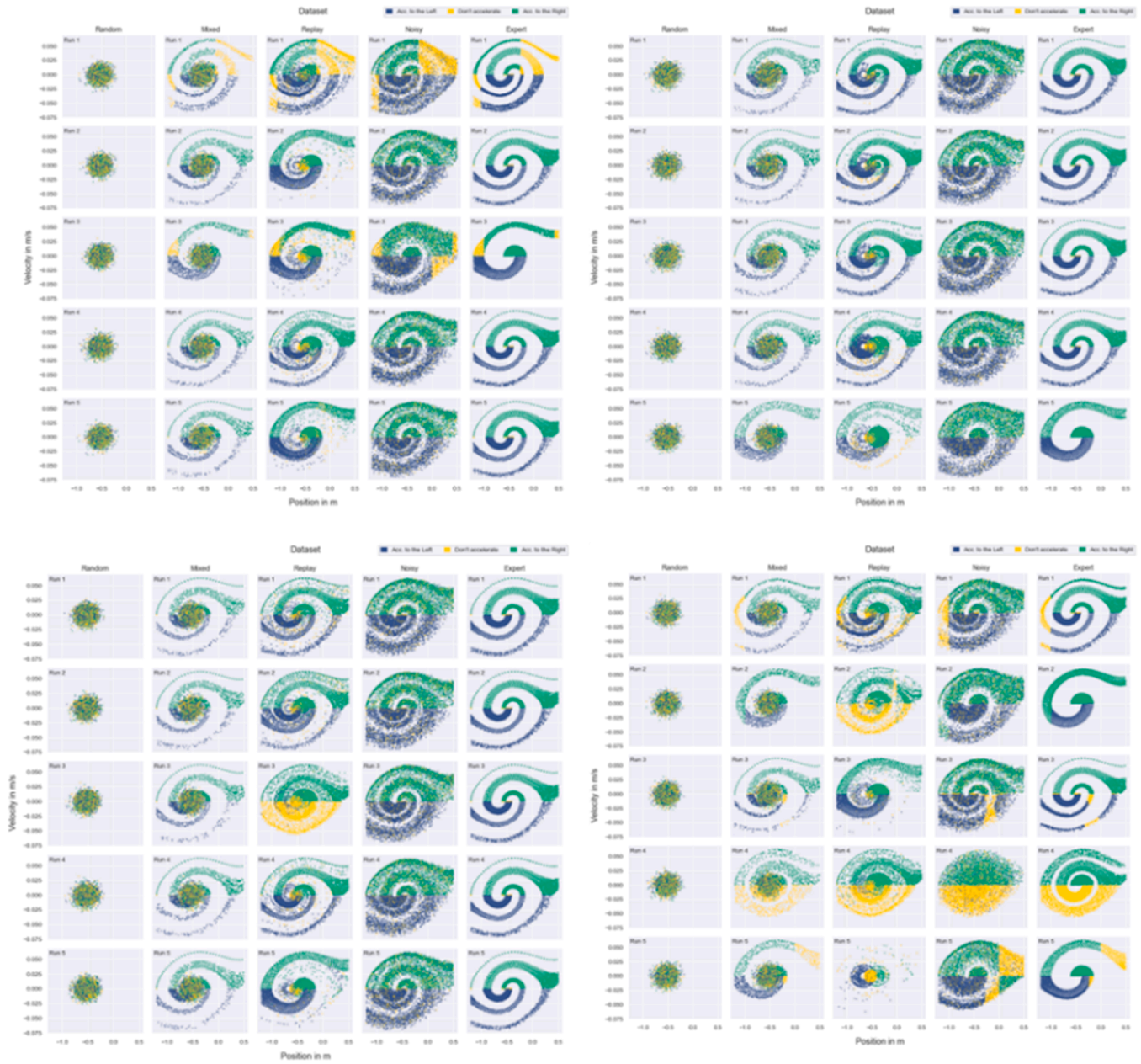


Fig. E.5. Effect of the different behavioral policies on the MC environment. Collected via GP (Experiment 5–8). Top left: Experiment 5. Top right: Experiment 6. Bottom left: Experiment 7. Bottom right: Experiment 8.

E.2. Statistical Tests and Results

Table E.8
The results for the preliminary statistical tests. The alpha level for the Levene test for equal variance and the Shapiro Wilk tests for normality is $\alpha = 0.05$.

Test	Levene Test p-values	Shapiro-Wilk Test p-values (DQN)	Shapiro-Wilk Test p-values (GP)	Effect size: Cohen's d
('CP', 'noisy', 'DQN')	7.086e-30	0.00793	1.000	266.626
('CP', 'noisy', 'CQL')	2.192e-07	0.02290	7.078e-05	36.675
('CP', 'mixed', 'CQL')	2.612e-05	9.217e-15	0.04244	18.161
('CP', 'fully', 'BC')	6.991e-12	5.821e-14	1.000	19.953
('CP', 'fully', 'CQL')	5.773e-22	1.105e-12	1.000	18.153
('CP', 'er', 'DQN')	0.375	1.516e-20	2.150e-19	5.211
('CP', 'fully', 'REM')	1.183e-09	2.776e-08	4.016e-06	5.128
('CP', 'er', 'REM')	7.065e-12	3.671e-09	0.01620	4.883
('MC', 'noisy', 'REM')	1.096e-38	1.000	2.677e-08	2.927

(*continued on next page*)

Table E.8 (*continued*)

Test	Levene Test p-values	Shapiro-Wilk Test p-values (DQN)	Shapiro-Wilk Test p-values (GP)	Effect size: Cohen's d
('CP', 'fully', 'DQN')	1.530e-06	2.274e-08	1.675e-14	2.797
('CP', 'noisy', 'REM')	2.540e-18	7.387e-12	2.000e-10	1.910
('CP', 'noisy', 'BC')	1.631e-25	5.723e-15	3.377e-07	1.382
('MC', 'er', 'BC')	5.734e-06	1.726e-10	1.112e-05	0.951
('MC', 'noisy', 'DQN')	0.124	6.544e-15	3.919e-14	0.909
('CP', 'mixed', 'REM')	1.766e-10	3.026e-13	0.00161	0.895
('CP', 'er', 'BC')	0.000138	2.205e-05	5.005e-09	0.841
('MC', 'fully', 'BC')	0.00361	4.581e-08	6.521e-10	0.663
('MC', 'fully', 'REM')	0.166	1.805e-12	1.859e-14	0.645
('MC', 'fully', 'DQN')	1.577e-06	3.348e-07	1.758e-09	0.587
('MC', 'er', 'REM')	0.783	5.305e-12	4.788e-08	0.516
('MC', 'mixed', 'REM')	0.175	1.833e-09	2.050e-17	0.454
('MC', 'noisy', 'CQL')	2.485e-41	4.047e-05	1.639e-10	0.276
('MC', 'mixed', 'BC')	0.032	7.586e-09	4.679e-12	0.269
('MC', 'noisy', 'BC')	0.600	1.243e-09	1.175e-08	0.259
('CP', 'mixed', 'DQN')	0.126	4.978e-06	0.02406	0.228
('MC', 'fully', 'CQL')	0.000124	5.041e-10	1.374e-09	0.223
('MC', 'er', 'CQL')	0.024	2.812e-10	3.659e-11	0.205
('MC', 'mixed', 'BC')	0.218	5.284e-20	1.768e-17	0.175
('MC', 'mixed', 'DQN')	0.047	4.709e-08	7.719e-06	0.077
('CP', 'er', 'CQL')	0.813	9.585e-18	6.365e-12	0.076
('MC', 'er', 'DQN')	0.990	7.679e-10	1.918e-10	0.055
('MC', 'mixed', 'CQL')	0.658	0.928	7.548e-08	0.013

Table E.9
Results for the two tailed Welch's *t*-test and the Bonferroni-Holm correction. Bold highlights statistical significance ($\alpha = 0.025$). The table is ordered by corrected p-value ascending.

Nr.	Test (Two-sided Welch's *t*-test)	t-values	p-values	Corrected p-values
1	('CartPole-v1', 'noisy', 'DQN')	1885.333	2.600e-227	8.475e-226
2	('CartPole-v1', 'noisy', 'CQL')	−259.331	1.900e-201	5.881e-200
3	('CartPole-v1', 'mixed', 'CQL')	−128.416	1.600e-119	4.765e-118
4	('CartPole-v1', 'fully', 'BC')	141.086	6.100e-116	1.757e-114
5	('CartPole-v1', 'fully', 'CQL')	128.359	6.700e-112	1.872e-110
6	('CartPole-v1', 'er', 'DQN')	−36.847	3.760e-72	1.015e-70
7	('CartPole-v1', 'fully', 'REM')	−36.264	3.670e-64	9.547e-63
8	('CartPole-v1', 'er', 'REM')	−34.529	4.270e-57	1.067e-55
9	('MountainCar-v0', 'noisy', 'REM')	20.700	9.580e-38	2.298e-36
10	('CartPole-v1', 'fully', 'DQN')	19.776	8.340e-37	1.919e-35
11	('CartPole-v1', 'noisy', 'REM')	13.505	3.610e-27	7.938e-26
12	('CartPole-v1', 'noisy', 'BC')	9.770	2.530e-16	5.315e-15
13	('MountainCar-v0', 'er', 'BC')	6.726	2.850e-10	5.706e-09
14	('MountainCar-v0', 'noisy', 'DQN')	−6.425	9.680e-10	1.838e-08
15	('CartPole-v1', 'mixed', 'REM')	6.331	6.630e-09	1.193e-07
16	('CartPole-v1', 'er', 'BC')	−5.948	1.310e-08	2.234e-07
17	('MountainCar-v0', 'fully', 'BC')	4.689	5.330e-06	8.525e-05
18	('MountainCar-v0', 'fully', 'REM')	−4.558	9.580e-06	1.437e-04
19	('MountainCar-v0', 'fully', 'DQN')	4.153	5.100e-05	7.143e-04
20	('MountainCar-v0', 'er', 'REM')	3.647	3.420e-04	4.442e-03
21	('MountainCar-v0', 'mixed', 'REM')	3.213	1.616e-03	1.940e-02
22	('MountainCar-v0', 'noisy', 'CQL')	1.954	5.313e-02	5.844e-01
23	('CartPole-v1', 'mixed', 'BC')	1.901	5.910e-02	5.910e-01
24	('MountainCar-v0', 'noisy', 'BC')	−1.831	6.854e-02	6.169e-01
25	('CartPole-v1', 'mixed', 'DQN')	1.612	1.085e-01	8.684e-01
26	('MountainCar-v0', 'fully', 'CQL')	1.580	1.158e-01	8.684e-01
27	('MountainCar-v0', 'er', 'CQL')	1.451	1.484e-01	8.906e-01
28	('MountainCar-v0', 'mixed', 'BC')	1.235	2.183e-01	1.000e+00
29	('MountainCar-v0', 'mixed', 'DQN')	0.547	5.848e-01	1.000e+00
30	('CartPole-v1', 'er', 'CQL')	0.538	5.915e-01	1.000e+00
31	('MountainCar-v0', 'er', 'DQN')	−0.386	7.002e-01	1.000e+00
32	('MountainCar-v0', 'mixed', 'CQL')	0.090	9.283e-01	1.000e+00

Table E.10
Results for the one tailed Welch's *t*-test and the Bonferroni-Holm correction. Bold highlights statistical significance ($\alpha = 0.025$). The table is ordered by corrected p-value ascending.

Nr	Test (One -sided Welch's *t*-test)	t-values	p-values	Corrected p-values
1	('CartPole-v1', 'noisy', 'DQN')	1885.333	1.324e-227	4.200e-226
2	('CartPole-v1', 'fully', 'BC')	141.086	3.030e-116	9.400e-115
3	('CartPole-v1', 'fully', 'CQL')	128.359	3.343e-112	1.000e-110
4	('MountainCar-v0', 'noisy', 'REM')	20.700	4.788e-38	1.390e-36
5	('CartPole-v1', 'fully', 'DQN')	19.776	4.172e-37	1.170e-35
6	('CartPole-v1', 'noisy', 'REM')	13.505	1.804e-27	4.870e-26
7	('CartPole-v1', 'noisy', 'BC')	9.770	1.266e-16	3.290e-15
8	('MountainCar-v0', 'er', 'BC')	6.726	1.426e-10	3.570e-09
9	('CartPole-v1', 'mixed', 'REM')	6.331	3.313e-09	7.950e-08
10	('MountainCar-v0', 'fully', 'BC')	4.689	2.664e-06	6.130e-05
11	('MountainCar-v0', 'fully', 'DQN')	4.153	2.551e-05	0.000561
12	('MountainCar-v0', 'er', 'REM')	3.647	0.000171	0.003588
13	('MountainCar-v0', 'mixed', 'REM')	3.213	0.000808	0.016163
14	('MountainCar-v0', 'noisy', 'CQL')	1.954	0.026564	0.504714
15	('CartPole-v1', 'mixed', 'BC')	1.901	0.029549	0.531882
16	('CartPole-v1', 'mixed', 'DQN')	1.612	0.054274	0.922656
17	('MountainCar-v0', 'fully', 'CQL')	1.580	0.057882	0.926116
18	('MountainCar-v0', 'er', 'CQL')	1.451	0.074214	1.000
19	('MountainCar-v0', 'mixed', 'BC')	1.235	0.109153	1.000
20	('MountainCar-v0', 'mixed', 'DQN')	0.547	0.292414	1.000
21	('CartPole-v1', 'er', 'CQL')	0.538	0.295756	1.000
22	('MountainCar-v0', 'mixed', 'CQL')	0.090	0.464175	1.000
23	('MountainCar-v0', 'er', 'DQN')	−0.386	0.649899	1.000
24	('MountainCar-v0', 'noisy', 'BC')	−1.831	0.965730	1.000
25	('MountainCar-v0', 'fully', 'REM')	−4.558	0.999995	1.000
26	('CartPole-v1', 'er', 'BC')	−5.948	0.999999	1.000
27	('MountainCar-v0', 'noisy', 'DQN')	−6.424	1.000000	1.000
28	('CartPole-v1', 'noisy', 'CQL')	−259.331	1.000000	1.000
29	('CartPole-v1', 'er', 'REM')	−34.529	1.000000	1.000
30	('CartPole-v1', 'fully', 'REM')	−36.263	1.000000	1.000
31	('CartPole-v1', 'er', 'DQN')	−36.847	1.000000	1.000
32	('CartPole-v1', 'mixed', 'CQL')	−128.416	1.000000	1.000

Appendix F. Further Figures

Fig. F.6, Fig. F.7, Fig. F.8, Fig. F.9, Fig. F.10, Fig. F.11.

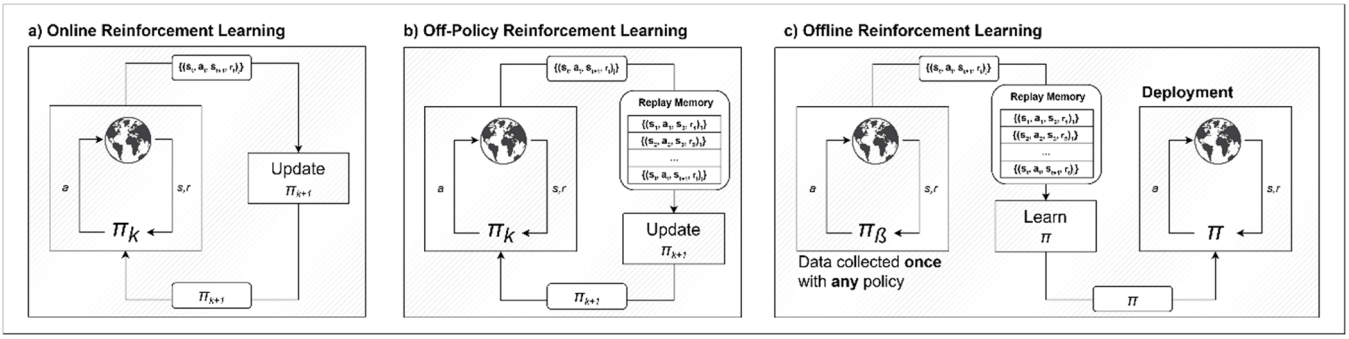


Figure F.6. Illustration of classic online RL (a), off-policy RL (b) and offline RL (c). This image is based on the pictural illustration of Levine et al. [5].

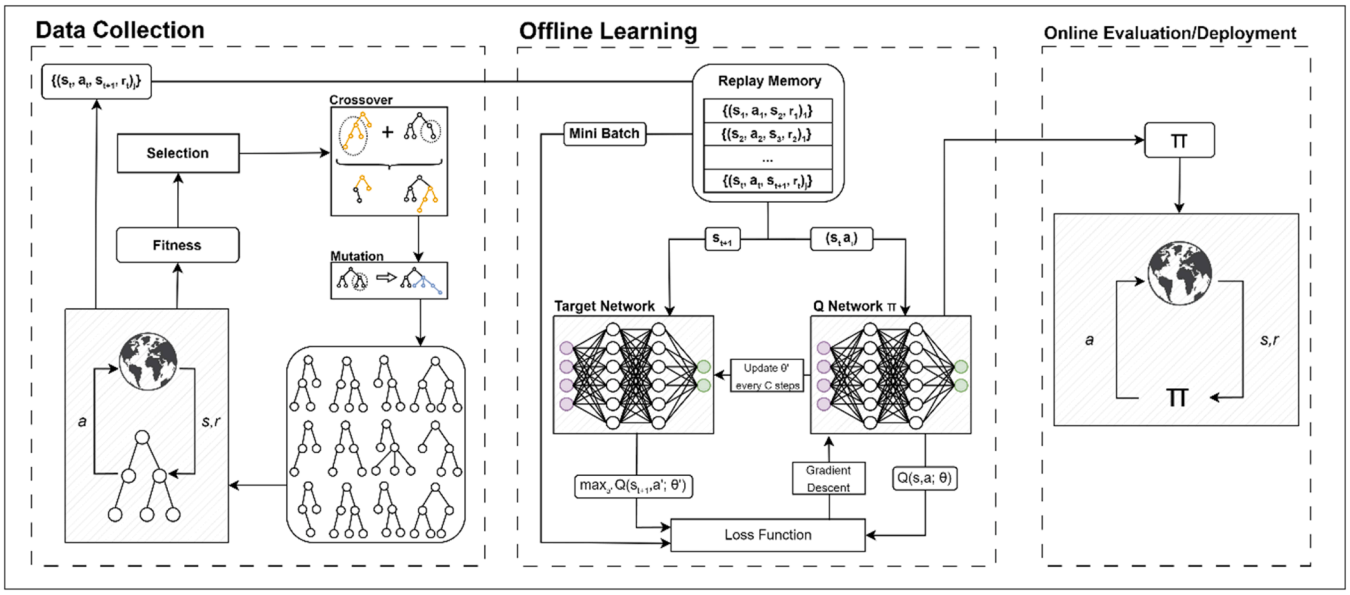


Figure F.7. Decoupled Evolutionary offline RL exemplified by GP for data collection and DQN for offline learning.

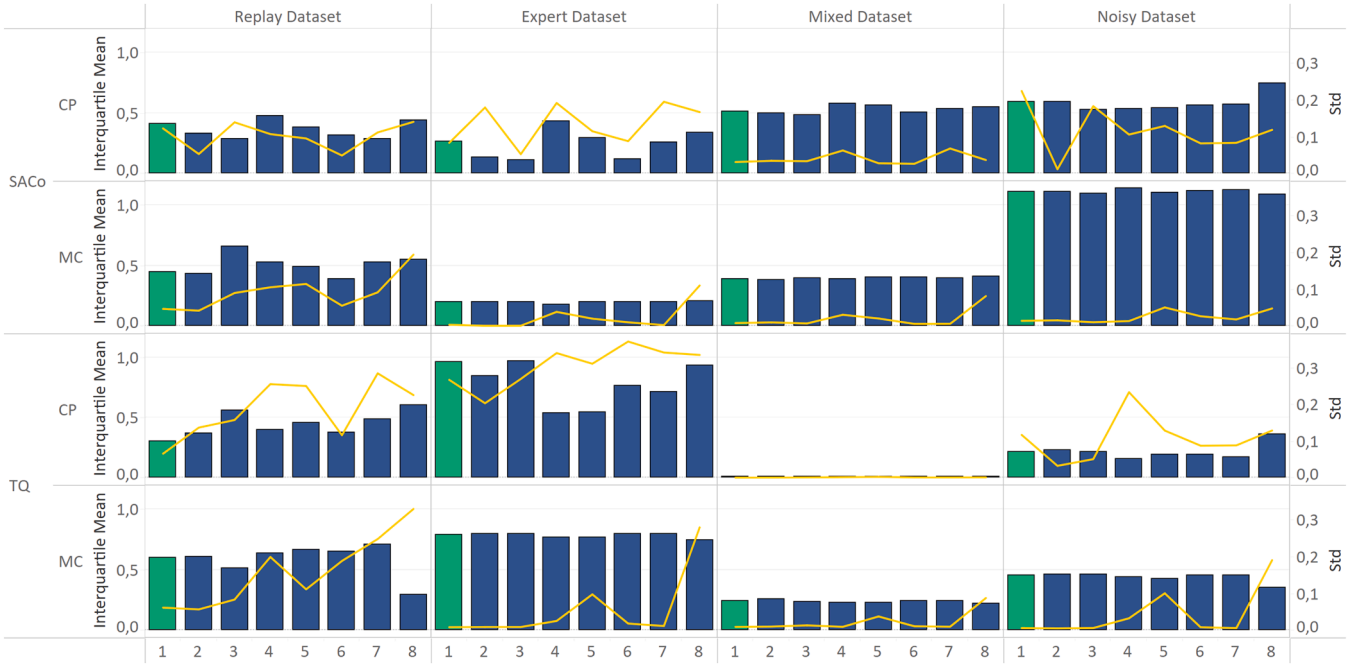


Figure F.8. Results of the trials measured across five independent runs. The numbers on the x-axis represent the number of the experiment. The standard deviation (yellow) is calculated using the mean value. The values of the bar plots (green for baseline, blue for experiments) show the IQM of TQ and SACo for the respective environment.

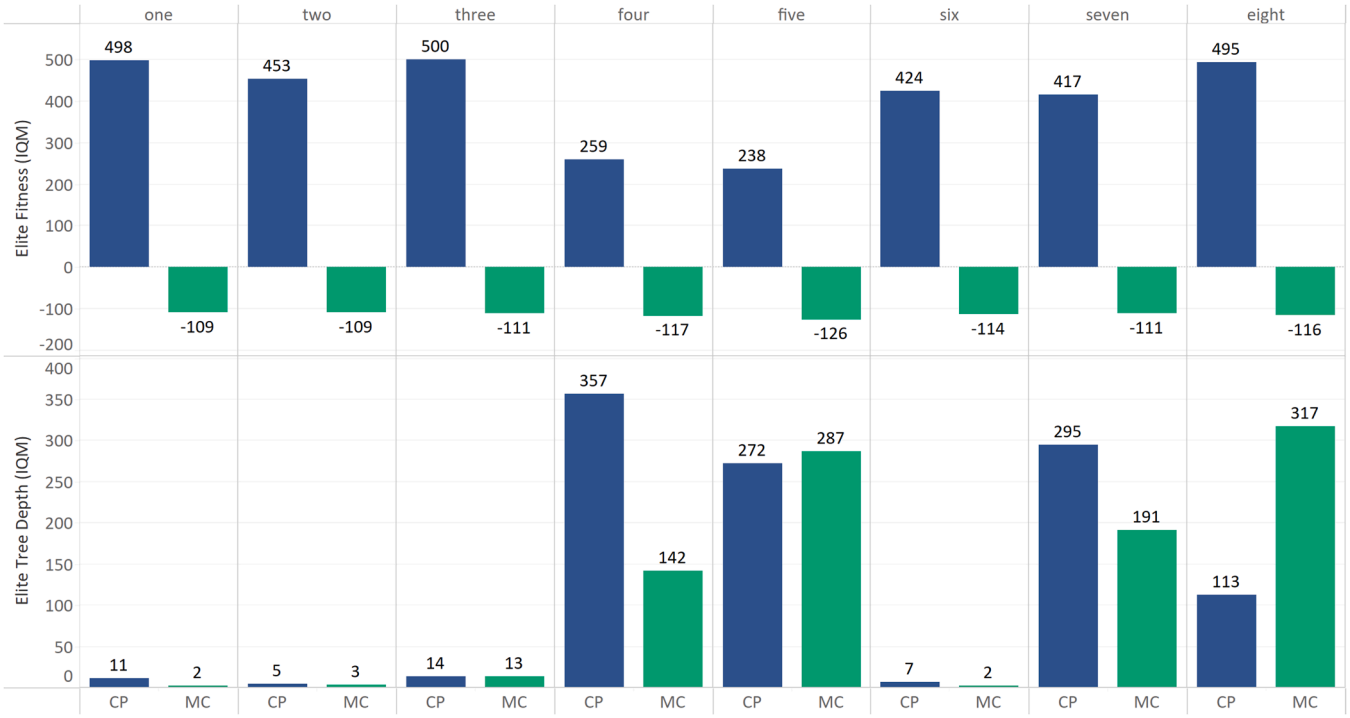


Figure F.9. Results of the trials measured across five independent runs. The numbers on the x-axis represent the number of the experiment. The standard deviation (yellow) is calculated using the mean value. The values of the bar plots (green for baseline, blue for experiments) show the IQM of TQ and SACo for the respective environment.

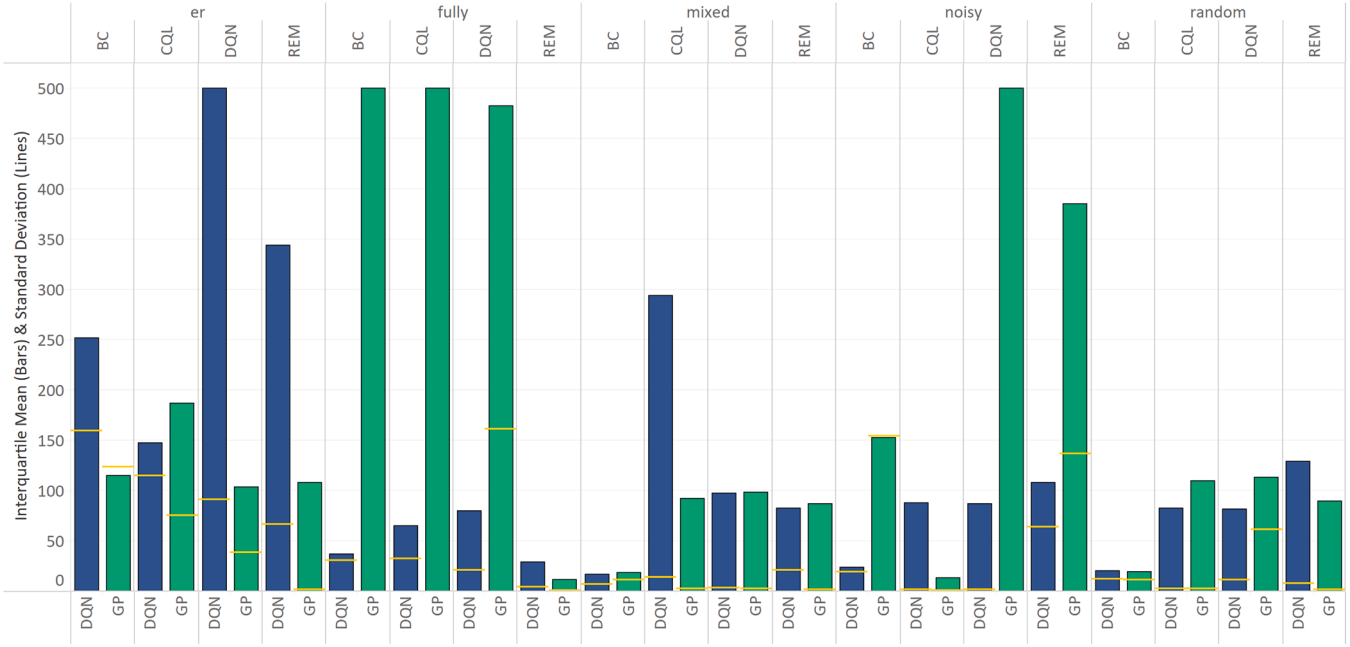


Figure F.10. Performance of the best-found policy in during offline training using CP data. The policies have collected rewards for 100 episodes greedily after being trained for 500.000 steps. The bar plots (green and blue) show the IQM of the rewards and the lines (yellow) the standard deviation.

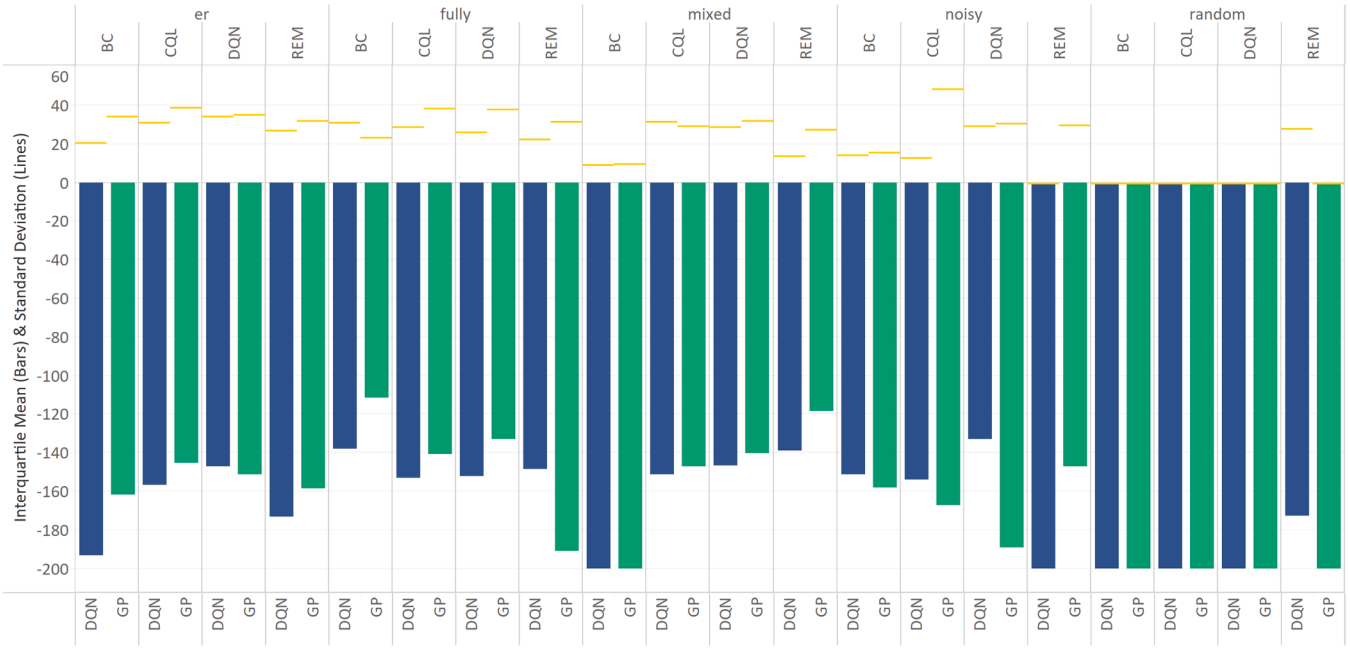


Figure F.11. Performance of the best-found policy in during offline training using MC data. The policies have collected rewards for 100 episodes greedily after being trained for 500.000 steps. The bar plots (green and blue) show the IQM of the rewards and the lines (yellow) the standard deviation.

Data availability

Data will be made available on request.

References

[1] OpenAI, OpenAI five defeats Dota 2 world champions. https://openai.com/research/openai-five-defeats-dota-2-world-champions, 2019 accessed April 5, 2024.

[2] S. Liu, K.C. See, K.Y. Ngiam, L.A. Celi, X. Sun, M. Feng, Reinforcement learning for clinical decision support in Critical care: comprehensive review, J. Med. Internet. Res. 22 (2020) e18477, https://doi.org/10.2196/18477.

[3] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A.A. Sallab, S. Yogamani, et al., Deep Reinforcement Learning for autonomous driving: a survey, IEEe Trans. Intell. Transp. Syst. 23 (2022) 4909–4926, https://doi.org/10.1109/TITS.2021.3054625.

[4] Singla A., Rafferty A.N., Radanovic G., Heffernan N.T. Reinforcement learning for education: opportunities and challenges 2021. https://doi.org/10.48550/arXiv.2107.08828.

[5] Levine S., Kumar A., Tucker G., Fu J. Offline Reinforcement Learning: tutorial, review, and perspectives on open problems 2020. https://doi.org/10.48550/arXiv.2005.01643.

[6] K. Schweighofer, M. Dinu, A. Radler, M. Hofmarcher, V.P. Patil, A. Bitto-nemling, et al., A dataset perspective on offline reinforcement learning, Proc. 1st Conf. Lifelong Learn. Agents 199 (2022) 470–517. PMLR.

[7] Yarats D., Brandfonbrener D., Liu H., Laskin M., Abbeel P., Lazaric A., et al. Don't change the algorithm, change the data: exploratory data for offline reinforcement learning 2022. https://doi.org/10.48550/arXiv.2201.13425.

[8] Riedmiller M., Springenberg J.T., Hafner R., Heess N. Collect & Infer – a fresh look at data-efficient reinforcement learning 2021.

[9] Salimans T., Ho J., Chen X., Sidor S., Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning 2017. https://doi.org/10.48550/arXiv.1703.03864.

[10] D.A. Pomerleau, Efficient training of artificial neural networks for autonomous navigation, Neural Comput. 3 (1991) 88–97, https://doi.org/10.1162/neco.1991.3.1.88.

[11] P. Swazinna, S. Udluft, T. Runkler, Measuring data quality for dataset selection in offline reinforcement learning, in: 2021 IEEE Symp. Ser. Comput. Intell. SSCI, 2021, pp. 1–8, https://doi.org/10.1109/SSCI50451.2021.9660006.

[12] Singh A., Kumar A., Vuong Q., Chebotar Y., Levine S. Offline RL with realistic datasets: heteroskedasticity and support constraints 2022. https://doi.org/10.48550/arXiv.2211.01052.

[13] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative Q-learning for offline reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 1179–1191. Curran Associates, Inc.

[14] Jiang L., Chen S., Qiu J., Xu H., Chan W.K., Ding Z. Offline reinforcement learning with imbalanced datasets 2023. https://doi.org/10.48550/arXiv.2307.02752.

[15] S. Levine, Understanding the world through action, in: Proc. 5th Conf. Robot Learn 164, 2022, pp. 1752–1757. PMLR.

[16] M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, et al., URLB: unsupervised reinforcement learning benchmark, in: Thirty-fifth conf. Neural Inf. Process. Syst. Datasets Benchmarks Track Round 2, 2021.

[17] Lambert N., Wulfmeier M., Whitney W., Byravan A., Bloesch M., Dasagi V., et al. The challenges of exploration for offline reinforcement learning 2022. https://doi.org/10.48550/arXiv.2201.11861.

[18] C. Sun, H. Qian, C. Miao, CUDC: a curiosity-driven unsupervised data collection method with adaptive temporal distances for offline reinforcement learning, Proc. AAAI. Conf. Artif. Intell. 38 (2024) 15145–15153, https://doi.org/10.1609/aaai.v38i13.29437.

[19] S. Endrawis, G. Leibovich, G. Jacob, G. Novik, A. Tamar, Efficient self-supervised data collection for offline robot learning, in: 2021 IEEE Int. Conf. Robot. Autom. ICRA, IEEE Press, Xi'an, China, 2021, pp. 4650–4656.

[20] P. Li, J. Hao, H. Tang, X. Fu, Y. Zheng, K. Tang, Bridging evolutionary algorithms and reinforcement learning: a comprehensive survey, CoRR (2024) abs/2401.11963.

[21] Sigaud O. Combining evolution and deep reinforcement learning for policy search: a survey 2022. https://doi.org/10.48550/arXiv.2203.14009.

[22] A.Y. Majid, S. Saaybi, V. Francois-Lavet, R.V. Prasad, C. Verhoeven, Deep reinforcement Learning versus Evolution strategies: a comparative survey, IEEe Trans. Neural Netw. Learn. Syst. (2024) 1–19, https://doi.org/10.1109/TNNLS.2023.3264540.

[23] H. Bai, R. Cheng, Y. Jin, Evolutionary Reinforcement Learning: a survey, Intell. Comput. 2 (2023) 0025, https://doi.org/10.34133/icomputing.0025.

[24] S. Khadka, K. Tumer, Evolution-guided policy gradient in reinforcement learning, in: Proc. 32nd Int. Conf. Neural Inf. Process. Syst., Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 1196–1208.

[25] Lillicrap T.P., Hunt J.J., Pritzel A., Heess N., Erez T., Tassa Y., et al. Continuous Control with Deep Reinforcement Learning 2019.

[26] Khadka S., Majumdar S., Nassar T., Dwiel Z., Tumer E., Miret S., et al. Collaborative evolutionary reinforcement learning. 2019.

[27] F. Esposito, A. Bonarini, Gradient Bias to Solve the Generalization Limit of Genetic Algorithms Through Hybridization With Reinforcement Learning, 12565, Springer International Publishing, Cham, 2020, pp. 273–284, https://doi.org/10.1007/978-3-030-64583-0_26.

[28] B. Zheng, R. Cheng, Rethinking population-assisted off-policy reinforcement learning, in: Proc. Genet. Evol. Comput. Conf., 2023, pp. 624–632, https://doi.org/10.1145/3583131.3590512.

[29] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: Proc. 35th Int. Conf. Mach. Learn. 80, 2018, pp. 1587–1596. PMLR.

[30] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, C. Zhang, Cooperative heterogeneous deep reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 17455–17465. Curran Associates, Inc.

[31] Suri K., Shi X.Q., Plataniotis K.N., Lawryshyn Y.A. Maximum mutation reinforcement learning for scalable control 2021. https://doi.org/10.48550/arXiv.2007.13690.

[32] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proc. 35th Int. Conf. Mach. Learn 80, 2018, pp. 1861–1870. PMLR.
[33] C. Bodnar, B. Day, P. Lió, Proximal distilled evolutionary reinforcement learning, Proc. AAAI. Conf. Artif. Intell. 34 (2020) 3283–3290, https://doi.org/10.1609/aaai.v34i04.5728.
[34] E. Marchesini, D. Corsi, A. Farinelli, Exploring safer behaviors for deep reinforcement learning, Proc. AAAI. Conf. Artif. Intell. 36 (2022) 7701–7709, https://doi.org/10.1609/aaai.v36i7.20737.
[35] S. Lü, S. Han, W. Zhou, J. Zhang, Recruitment-imitation mechanism for evolutionary reinforcement learning, Inf. Sci. 553 (2021) 172–188, https://doi.org/10.1016/j.ins.2020.12.017.
[36] N. Kim, H. Baek, H. Shin, PGPS : Coupling policy gradient with population-based search, Openreview (2020).
[37] Y. Wang, T. Zhang, Y. Chang, B. Liang, X. Wang, B. Yuan, A surrogate-assisted controller for expensive evolutionary reinforcement learning, Inf. Sci. 616 (2022) 539–557, https://doi.org/10.1016/j.ins.2022.10.134.
[38] J. Hao, P. Li, H. Tang, Y. Zheng, X. Fu, Z. Meng, ERL-re^2: efficient evolutionary reinforcement learning with shared State representation and individual policy representation, Elev. Int. Conf. Learn. Represent (2023) arXiv.
[39] Guo Y., Wang H., He L., Pedrycz W., Suganthan P.N., Song Y. A reinforcement learning-assisted genetic programming algorithm for team formation problem considering person-job matching 2023. https://doi.org/10.48550/arXiv.2304.04022.
[40] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2002) 99–127, https://doi.org/10.1162/106365602320169811.
[41] J. Lehman, K.O. Stanley, Abandoning objectives: evolution through the search for novelty alone, Evol. Comput. 19 (2011) 189–223, https://doi.org/10.1162/EVCO_a_00025.
[42] J.K. Pugh, L.B. Soros, K.O. Stanley, Quality diversity: a new frontier for evolutionary computation, Front. Robot. AI. 3 (2016).
[43] Gravina D., Liapis A., Yannakakis G.N. Surprise search for evolutionary divergence 2017.
[44] H. Mengistu, J.A. Lehman, J. Clune, Evolvability search: directly selecting for Evolvability in order to study and produce it, in: Proc. Genet. Evol. Comput. Conf. 2016, 2016, pp. 141–148, https://doi.org/10.1145/2908812.2908838.
[45] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: Proc. 34th Int. Conf. Mach. Learn. 70, 2017, pp. 2778–2787. PMLR.
[46] Liu Q., Wang Y., Liu X. PNS: population-guided novelty search for reinforcement learning in hard exploration environments 2021. https://doi.org/10.48550/arXiv.1811.10264.
[47] L. Shi, S. Li, Q. Zheng, M. Yao, G. Pan, Efficient novelty search through deep reinforcement learning, IEEe Access. 8 (2020) 128809–128818, https://doi.org/10.1109/ACCESS.2020.3008735.
[48] T. Pierrot, V. Macé, G. Cideron, N. Perrin, K. Beguir, O. Sigaud, Sample efficient quality diversity for neural continuous control, Openreview (2020).
[49] Mouret J.-B., Clune J. Illuminating search spaces by mapping elites 2015. https://doi.org/10.48550/arXiv.1504.04909.
[50] O. Nilsson, A. Cully, Policy gradient assisted MAP-elites, in: Proc. Genet. Evol. Comput. Conf., Association for Computing Machinery, New York, NY, USA, 2021, pp. 866–875, https://doi.org/10.1145/3449639.3459304.
[51] B. Tjanaka, M.C. Fontaine, J. Togelius, S. Nikolaidis, Approximating gradients for differentiable quality diversity in reinforcement learning, CoRR (2022) abs/2202.03666.
[52] C. Colas, O. Sigaud, P.-Y. Oudeyer, GEP-PG: decoupling exploration and exploitation in deep reinforcement learning algorithms, in: Proc. 35th Int. Conf. Mach. Learn. 80, 2018, pp. 1039–1048. PMLR.
[53] M. Zimmer, S. Doncieux, Bootstrapping Q -learning for robotics from Neuro-evolution results, IEEe Trans. Cogn. Dev. Syst. 10 (2018) 102–119, https://doi.org/10.1109/TCDS.2016.2628817.
[54] J.R. Koza, R. Poli, Genetic Programming. Search Methodol. Introd. Tutor. Optim. Decis. Support Tech, Springer US, Boston, MA, 2005, pp. 127–164, https://doi.org/10.1007/0-387-28356-0_5.
[55] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, https://doi.org/10.1038/nature14236.
[56] Brockman G., Cheung V., Pettersson L., Schneider J., Schulman J., Tang J., et al. OpenAI Gym 2016. https://doi.org/10.48550/arXiv.1606.01540.
[57] P. Flajolet, É. Fusy, O. Gandouet, F. Meunier, HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm, in: Discrete Math Theor Comput Sci 2007;DMTCS Proceedings vol. AH, 2007, https://doi.org/10.46298/dmtcs.3545. Conference on Analysis of Algorithms (AofA 07).
[58] R. Agarwal, M. Schwarzer, P.S. Castro, A.C. Courville, M. Bellemare, Deep reinforcement learning at the edge of the statistical precipice, Adv. Neural. Inf. Process. Syst. 34 (2021) 29304–29320. Curran Associates, Inc.
[59] R. Agarwal, D. Schuurmans, M. Norouzi, An optimistic perspective on offline reinforcement learning, in: Proc. 37th Int. Conf. Mach. Learn. 119, 2020, pp. 104–114. PMLR.
[60] C. Colas, O. Sigaud, P.-Y. Oudeyer, A Hitchhiker's guide to statistical comparisons of reinforcement learning algorithms, 2022, https://doi.org/10.48550/arXiv.1904.06979.
[61] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[62] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Proc. Thirty-Second AAAI Conf. Artif. Intell. Thirtieth Innov. Appl. Artif. Intell. Conf. Eighth AAAI Symp. Educ. Adv. Artif. Intell., AAAI Press, New Orleans, Louisiana, USA, 2018.
[63] S.S. Shapiro, M.B. Wilk, An analysis of variance test for normality (Complete Samples), Biometrika 52 (1965) 591–611, https://doi.org/10.2307/2333709.
[64] H. Levene, Robust tests for equality of variances. Contrib. Probab. Stat. Essays Honor Harold Hotell, Stanford University Press, Palo Alto, California, 1960, pp. 278–292.
[65] B.L. Welch, The generalization of `student's' Problem when several different population variances are involved, Biometrika 34 (1947) 28–35, https://doi.org/10.2307/2332510.
[66] J. Cohen, Statistical Power Analysis For the Behavioral Sciences, 2nd ed., L. Erlbaum Associates, New York, 1988 https://doi.org/10.4324/9780203771587.
[67] Miller J. Cartesian genetic programming. vol. 43. 2003. https://doi.org/10.1007/978-3-642-17310-3.
[68] S. Kelly, R.J. Smith, M.I. Heywood, Emergent Policy Discovery For Visual Reinforcement Learning Through Tangled Program Graphs: A Tutorial, Springer International Publishing, Cham, 2019, pp. 37–57, https://doi.org/10.1007/978-3-030-04735-1_3.
[69] K. Desnos, N. Sourbier, P.-Y. Raumer, O. Gesny, P.M. Gegelati, Lightweight artificial intelligence through generic and evolvable tangled program graphs, in: Association for Computing Machinery, New York, NY, USA, 2021, pp. 35–43, https://doi.org/10.1145/3441110.3441575, vol. 14.
[70] Farama Foundation. Gymnasium Documentation, Gymn Class Control. https://gymnasium.farama.org/environments/classic_control.html, 2023 (accessed April 23, 2024).
[71] F.-A. Fortin, F.-M.D. Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (2012) 2171–2175.
[72] J.R. Koza, Genetic programming as a means for programming computers by natural selection, Stat. Comput. 4 (1994) 87–112, https://doi.org/10.1007/BF00175355.
[73] L. Vanneschi, M. Castelli, L. Manzoni, S. Silva, A New Implementation of Geometric Semantic GP and its Application to Problems in Pharmacokinetics, 2013, pp. 205–216, https://doi.org/10.1007/978-3-642-37207-0_18.
[74] A. Moraglio, K. Krawiec, C.G. Johnson, Geometric Semantic Genetic Programming. Parallel Probl. Solving Nat. - PPSN XII, Springer, Berlin, Heidelberg, 2012, pp. 21–31, https://doi.org/10.1007/978-3-642-32937-1_3.