Universiteit Leiden
The Netherlands

Data Science & Artificial Intelligence

Decoupling Exploration and Exploitation in Intrinsic Motivation Methods

Valeria Rezan

Supervisors:
Thomas Moerland & Lindsay Spoor

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/07/2025

Abstract

Efficient exploration remains one of the core challenges in reinforcement learning. Intrinsic motivation (IM) methods tackle this by augmenting the reward signal with exploration bonuses, enabling more directed behavior. However, these bonuses often introduce non-stationary biases into the value function, which destabilizes learning and increases sensitivity to hyperparameters. To address these limitations, this thesis explores a decoupling strategy that separates the learning objectives of exploration and exploitation. The method maintains two distinct Q-functions: an exploratory Q-function, updated using either intrinsic rewards or a combination of intrinsic and extrinsic rewards, and a separate exploitation Q-function, updated exclusively with extrinsic rewards. Although such a decoupled approach has been used in prior work, it was rarely the focus and was mostly applied in deep RL settings, where it is harder to isolate the benefits due to function approximation noise. To provide a more controlled analysis, we implement and evaluate two decoupled variants of the Dyna-Q+ algorithm in purely tabular settings across five environments, including non-stationary and deceptive tasks. Our results show that decoupled agents consistently outperform the integrated IM baseline in terms of policy quality, learning stability, and robustness to hyperparameter tuning. These findings suggest that explicitly separating exploration and exploitation can mitigate key limitations of current IM methods and offer a promising direction for future reinforcement learning research.

Contents

1 Introduction

The primary objective of an agent in Reinforcement Learning (RL) is to interact with an unknown environment to maximize cumulative rewards over time. This typically involves favoring actions that have previously yielded high returns. However, relying solely on exploitation is insufficient, as it prevents the agent from discovering potentially better actions that have not yet been tried. To discover truly optimal behavior, the agent must also explore the environment effectively. As a result, the agent's performance critically depends on its ability to explore and balance exploration and exploitation [3].

1.1 Random Exploration Methods

Despite the importance of effective exploration, many modern reinforcement learning approaches still rely on basic random exploration strategies, such as ϵ-greedy or Gaussian noise, to guide the agent through the environment. These methods are simple, well-studied, and, under certain assumptions, can theoretically allow the agent to explore the entire state-action space and converge to an optimal policy, given sufficient time [43]. Despite their simplicity, these methods have been successfully combined with deep learning techniques and applied to challenging domains, including Atari games, the MuJoCo physics simulator, controller tuning, autonomous drone landing, self-driving vehicles, and healthcare applications [23, 14, 21]. Nevertheless, random exploration methods remain highly inefficient, as they fail to prioritize informative or promising areas of the environment. This often leads to redundant visits to already explored states and a slow discovery of rewarding states, particularly in high-dimensional or continuous environments. This major limitation has motivated researchers to develop more effective and informative exploration strategies. One prominent example is intrinsic motivation, which is the focus of this work.

1.2 Intrinsic Motivation and its Limitations

Intrinsic motivation methods address the inefficiency of Random Exploration by providing internal rewards when the agent discovers novel or uncertain states, encouraging a more directed and adaptive exploration of the environment. The idea of intrinsically motivated exploration draws inspiration from developmental psychology, specifically the way infants spontaneously explore their surroundings and acquire new skills through curiosity-driven behavior [31]. In reinforcement learning, intrinsic motivation methods provide the agent with additional intrinsic rewards that are independent of the environment's extrinsic reward signal [9]. In other words, the agent is encouraged to act not because of any external payoff but because of an internal drive such as curiosity or the desire to reduce uncertainty.

Intrinsic motivation can take many forms depending on its underlying drive, for example, cooperative incentives in multi-agent RL [17] or rewards for acquiring unsupervised skills that transfer across tasks [13]. In this work, we focus on exploration-based intrinsic motivation. In this approach, the agent receives an intrinsic reward r_i for exploring the environment in meaningful ways, for example by visiting novel states or states that have not been encountered recently. This is typically achieved using self-supervised predictions [30] or by maintaining counts of state visits [29]. The total reward r received by the agent is then computed as a weighted sum of the extrinsic reward r_e and the intrinsic reward, expressed as $r = r_e + \kappa r_i$, where κ is a weighting factor. This

way, the agent's intrinsic reward augments its objective with a signal that dynamically adapts based on its experiences. This results in a more informed exploration process, as the agent uses its prior experiences to guide its exploration, rather than relying on random strategies. Extrinsic rewards, on the other hand, are provided by the environment and remain fixed, irrespective of the agent's interaction history.

Intrinsic motivation has enabled agents to solve challenging tasks such as maze navigation, robotic manipulation, multi-agent coordination, and learning in procedurally generated environments [6]. This intrinsic drive helps agents explore more efficiently, especially in environments where extrinsic rewards are sparse or delayed.

Nonetheless, an important problem associated with this approach is often overlooked. Intrinsic motivation methods lead to non-stationary rewards, which violate the Markov Decision Property [35]. This results in two main issues. Firstly, intrinsic bonuses introduce a slowly decreasing bias in the target policy [49]. When a combined reward is used to update the policy, the intrinsic component, often dependent on state-space coverage, can skew learning by promoting suboptimal exploration and exploitation. In other words, the agent may struggle to learn the optimal action sequence as the policy is continuously influenced by the exploration bonus. Secondly, intrinsic motivation methods are highly sensitive to hyperparameters, particularly the scale of the exploration weight (κ) and the decay rate of the intrinsic reward [35]. For the agent to effectively balance exploration and exploitation, the intrinsic bonus must be sufficiently large, often comparable to the extrinsic reward, especially in the early stages of training. As a result, careful tuning is critical to the success of these methods.

1.3 Decoupling Exploration and Exploitation

The limitations of intrinsic motivation methods introduced in the previous subsection motivate the idea of decoupling exploration and exploitation. Instead of relying on a single policy to handle both tasks, we can separate them into two distinct components: one dedicated to exploration, which gathers information about the environment, and another focused purely on exploitation, which uses only extrinsic rewards to learn from the collected data. This separation isolates the intrinsic reward signal from the main value function, thereby stabilizing the learning process and mitigating negative effects caused by non-stationarity.

This approach has been explored in several studies, particularly in the field of robotics, where sparse rewards are common [38, 12, 10]. However, the decoupling of exploration and exploitation was often not the primary focus of these works or was investigated only in the context of sophisticated deep reinforcement learning methods [35, 49]. Nevertheless, it is crucial to investigate this topic in isolation as a primary research question. Without such isolation, it becomes difficult to determine whether performance improvements are due to the decoupling itself or to other enhancements in the exploration strategy. Moreover, it is also beneficial to examine this question in the context of a simple tabular agent, independent of deep function approximation, which can introduce substantial noise and variance, thereby complicating the interpretation of results related to decoupling.

For these reasons, I propose to investigate the benefits of decoupled exploration by applying this approach to a simple tabular agent, enabling a more controlled and interpretable analysis. [1]

[1] The code for this project is available at this GitHub repository.

1.4 Research Questions and Work Outline

To understand the implications of decoupled exploration, we investigate its performance in a set of controlled environments characterized by sparse rewards, where efficient exploration is particularly important.

For each environment, the goal is to address the following research questions:

- How does a decoupled approach compare to traditional integrated IM methods in terms of overall performance, as measured by evaluation returns?
- How does the use of decoupled exploration influence state space coverage during training?
- How sensitive is the decoupled approach to the intrinsic reward scaling factor compared to the integrated approach?

To address the research questions outlined above, the remainder of this thesis is organized as follows. The Preliminaries section introduces the foundational concepts necessary for understanding the rest of this work. It covers Markov Decision Processes (MDPs), Temporal-Difference (TD) methods, random exploration strategies, count-based intrinsic motivation techniques, and the model-based Dyna-Q algorithm, which serves as the primary framework on which the proposed decoupling approach was implemented. In the Related Work section, I review prior research on exploration strategies in reinforcement learning, with particular emphasis on approaches involving random exploration and intrinsic motivation methods.The Methodology section discusses the design of the experiments, including the environments used for testing, evaluation metrics, and hyperparameter search. In the Results section, the experimental outcomes are presented, supported by relevant graphs and visualizations. Finally, the Discussion section provides an interpretation of the results, addresses the limitations of the current study, and outlines directions for future research.

2 Preliminaries

In this section, we present the theoretical preliminaries necessary for a thorough understanding of this work. We begin by defining the Markov Decision Process, a fundamental framework in reinforcement learning. Next, we outline the key components required to solve a task within an MDP. We then introduce the exploration methods employed in this study. Finally, we describe the Dyna-Q agent, which forms the basis for our subsequent investigation into decoupled exploration.

2.1 Markov Decision Process

A **Markov Decision Process (MDP)** is a formal framework used to describe sequential decision-making in dynamic systems [32]. The system, controlled by a decision-maker or agent, evolves over a set of discrete time steps T, which we assume to be finite for the purposes of this work.

An MDP can be formally defined by a 5-tuple (S, A, P, R, γ), where:

- S is the set of all possible states the agent can encounter,
- $A(s_t)$ is the set of all possible actions the agent can take in state s_t,
- $P(s_{t+1} \mid s_t, a_t)$ is the transition probability function, which defines the probability of reaching state s_{t+1} from state s_t after taking action a_t,
- $R(s_t, a_t)$ is the reward function, which provides the expected reward received after performing action a_t in state s_t,
- $\gamma \in [0, 1]$ is the discount factor that determines the importance of future rewards.

At each time step $t \in T$, the agent observes a specific state of the system $s_t \in S$. Based on this observation, the agent selects one of the available actions $a_t \in A(s_t)$. As a result of this action, the agent experiences a new state s_{t+1} and a reward r sampled from the transition probability function $P(s_{t+1} \mid s_t, a_t)$ and the reward function $R(s_t, a_t)$ accordingly. In classical MDPs, transitions can be reversible, meaning that the agent may be able to probe the environment without committing to a state change. However, in most reinforcement learning settings, the interaction is *irreversible*: once an action is taken, the system transitions to a new state and cannot return to the previous state unless explicitly modeled.

The behavior of a decision-making agent is modeled by a **policy function** $\pi(s_t, a)$, which defines the probability of taking action a when in state s_t. The objective of the agent is to learn an **optimal policy**, denoted $\pi^*(s)$, that maximizes the expected cumulative discounted reward over time:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{1}$$

A fundamental property of an MDP is the assumption that the transition and reward functions depend only on the current state and action [32]. In other words, the environment is assumed to be *memoryless*. This assumption is important because it significantly simplifies the problem: the agent does not need to consider the full history of previous states to take select the next action.

With the MDP framework established, we now turn to the question of how to compute optimal agent behavior.

2.2 Solving a Markov Decision Process

Solving an MDP involves identifying an optimal policy π^* that maximizes the expected return from any initial state [43]. This goal can be achieved by continuously evaluating the agent's behavior and refining it based on these evaluations. To understand this process, we first introduce the concept of value functions.

Value Functions: In reinforcement learning, value functions quantify how good it is to be in a certain state or to take a specific action [43]. There are two commonly used value functions:

- The *state-value function* $V(s)$ gives the expected cumulative discounted reward when starting from state s and following policy π thereafter:

$$V(s) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right] \tag{2}$$

- The *state-action value function*, or Q-function, $Q(s, a)$ represents the expected return from taking action a in state s and then following policy π:

$$Q(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right] \tag{3}$$

These functions provide a foundation for learning and improving policies.

Policy Evaluation and Improvement: To compute an optimal policy, we begin by evaluating the current policy. This process uses the Bellman expectation equations to estimate the expected returns [43]:

$$V(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s' \mid s, a)\,[R(s, a) + \gamma V(s')] \tag{4}$$

Equivalently, we can evaluate the Q-function as:

$$Q(s, a) = \sum_{s' \in S} P(s' \mid s, a) \left[R(s, a) + \gamma \sum_{a' \in A} \pi(a' \mid s') Q(s', a')\right] \tag{5}$$

Rather than relying on full episodes to estimate the expected returns, these formulations exploit the recursive nature of the value functions. Note that the evaluation step should be performed iteratively over all state–action pairs until the value functions converge.

After evaluating the policy, we can improve it by choosing actions that maximize expected returns [43]. The improved policy π' is given by:

$$\pi'(s) = \arg\max_a \sum_{s'} P(s' \mid s, a)\,[R(s, a) + \gamma V(s')] \quad \text{or} \tag{6}$$

$$\pi'(s) = \arg\max_a Q(s, a) \tag{7}$$

Just like policy evaluation, policy improvement should be performed for all states and repeated iteratively until convergence. Thereby, by alternating evaluation and improvement, the agent converges to the optimal policy. This procedure is known as **policy iteration**.

Although either $V(s)$ or $Q(s,a)$ can be used to guide learning, using Q-values is often more direct and practical. This is because the policy can be derived simply by selecting the action with the highest Q-value in each state. In contrast, deriving a policy from $V(s)$ requires an additional Bellman update over all actions to determine the best action at each state, adding computational overhead.

Value Iteration: An alternative approach, called **value iteration**, integrates policy evaluation and improvement into a single step. It leverages the fact that there is no need to store the policy function explicitly, as it can be implicitly represented by the value function itself. Instead of evaluating a fixed policy, we iteratively update the value function using a greedy policy at each step [43]. The update rule is:

$$V(s) = \max_a \sum_{s' \in S} P(s' \mid s,a) \left[R(s,a) + \gamma V(s') \right] \tag{8}$$

Alternatively, we can perform value iteration over Q-values:

$$Q(s,a) = \sum_{s'} P(s' \mid s,a) \left[R(s,a) + \gamma \max_{a'} Q(s',a') \right] \tag{9}$$

The value or Q-value iteration update is applied repeatedly for all states until convergence. Once the value function has stabilized, the optimal policy can be recovered using Equation 6 or Equation 7, depending on the chosen value function.

One remaining challenge with this solution is that it assumes access to the environment's dynamics, specifically the transition and reward functions, which are often unknown to the agent in practice. In the next subsection, we introduce the Temporal Difference (TD) approach, which addresses this limitation by learning directly from experience without requiring a model of the environment.

2.3 Model-free Learning with Temporal Difference

Instead of relying on full knowledge of the environment, Temporal Difference learning uses experiences gathered through interaction to update value estimates. A typical experience consists of a transition (s, a, r, s'), where the agent starts in state s, takes action a, receives reward r, and arrives in the next state s'. The value function is then updated using the TD(0) update rule [43]:

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right] \tag{10}$$

Here, $\alpha \in (0, 1]$ is the learning rate, which determines the step size of the update and balances the influence of new information against previously accumulated estimates. This update rule reflects the principle of bootstrapping: improving value estimates using other learned estimates rather than waiting for the full return.

Now, if we want to apply the update rule to the Q-value function, we encounter the issue that we only observe the next state, but the action taken in that state remains unknown. This leads to two possible approaches.

The first, known as an **off-policy method**, assumes that we always consider the greedy action in the next state. Note that the actual action taken by the agent may not be greedy, since exploration is still necessary to discover the environment. In this setting, the behavioral policy (used to interact with the environment) differs from the target policy (used for learning updates). An example of this approach is **Q-learning**, with the following update rule [48]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

In contrast, the second approach, known as an **on-policy method**, assumes that the agent uses the same policy both for action selection and for the update rule [34]. An example of such methods is the SARSA algorithm, which uses the following update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \text{ where } a_{t+1} \sim \pi(\cdot \mid s_{t+1})$$

While both approaches are widely used in reinforcement learning, this study specifically focuses on the off-policy method DynaQ, described in detail in Section 2.5.

2.4 Introducing Exploration

To identify an optimal sequence of actions, an agent must first discover promising trajectories through its environment. This process requires active exploration to gather information about the environment's dynamics. In this work, we implement two widely-used exploration strategies: ϵ-greedy and count-based intrinsic motivation.

ϵ-greedy

The ϵ-greedy strategy is one of the most common exploration techniques in reinforcement learning, valued for its simplicity and theoretical convergence guarantees [42]. It belongs to the class of *uniform exploration* methods, where exploratory actions are selected randomly, independent of the current state.

Under an ϵ-greedy policy, the agent selects a random action with probability ϵ, and with probability $1 - \epsilon$, it chooses the action that maximizes its current estimate of the action-value function $Q(s, a)$. Formally:

$$a_t \sim \begin{cases} \text{uniform}(\mathcal{A}(s_t)) & \text{with probability } \epsilon, \\ \text{argmax}_{a \in \mathcal{A}(s_t)} Q(s_t, a) & \text{with probability } 1 - \epsilon, \end{cases}$$

To encourage more directed exploration at the beginning of the learning process and gradually shift toward exploitation as the agent gains experience, a decaying ϵ-greedy strategy was proposed [8]. In this approach, the exploration rate ϵ is reduced over time by applying a slowly decreasing decay factor after each time step.

Count-based intrinsic motivation

Despite the simplicity of random-based exploration, it remains inefficient, particularly in environments with large or continuous state spaces. To guide exploration toward more informative states, the agent can incorporate an additional intrinsic reward, which reflects either the novelty of a visited state or the agent's uncertainty about it. This approach is referred to as intrinsic motivation or bonus-based exploration.

Various methods for computing intrinsic rewards have been proposed, as discussed in the Related Work section. In this thesis, however, I focus on tabular environments for which count-based exploration methods are particularly well-suited. Unlike other intrinsic motivation approaches that rely on learned predictive models to estimate novelty or uncertainty, count-based methods simply track state–action visitation counts, making them both straightforward to implement and computationally efficient. Specifically, I implement two count-based methods:

1. **Novelty-based exploration**, measured as $\frac{1}{\sqrt{N(s,a)}}$, where $N(s,a)$ is the number of times action a was selected at state s [39].
2. **Recency-based exploration**, measured as $\sqrt{\tau(s,a)}$, where $\tau(s,a)$ is the number of timesteps that have passed since action a was selected in state s [4].

Although numerous variants of novelty and recency exist, these two formulations are chosen for their simplicity and widespread adoption. Both approaches encourage exploration by rewarding the agent for visiting less-explored regions of the environment. However, novelty-based rewards consider only the overall visitation count, which may cause the agent to revisit states that were explored recently but still have low counts. In contrast, recency-based methods emphasize the time elapsed since the last visit, making them potentially more suitable for dynamic environments where adapting to recent changes is critical. Novelty-based rewards, on the other hand, may be more informative in stationary environments, where a recency-driven agent might repeatedly revisit states that have proven unproductive simply because enough time has passed since their last visit. Nevertheless, while a detailed comparison of these two methods may be of interest, it is not the focus of this work. Both novelty and recency formulations are employed to present results, but they are not directly compared.

2.5 Model-Based Enhancement: The Dyna-Q Framework

The TD approach described in the subsection 2.3 forms the foundation for many modern reinforcement learning algorithms. It is fully model-free, meaning it does not require knowledge of the environment's transition or reward functions in order to learn an optimal policy. However, the method can be significantly enhanced when such dynamics can be estimated or learned. This idea underlines model-based approaches such as Dyna-Q, which augments the basic framework by incorporating a learned model of the environment to perform additional planning updates [41].

Specifically, in contrast to purely model-free methods, Dyna-Q stores observed experiences in the form of transition counts $N(s,a,s')$ and cumulative rewards $R(s,a)$. Using these, the transition probability function and reward function can be approximated as:

$$\hat{P}(s' \mid s,a) = \frac{N(s,a,s')}{\sum_{s''} N(s,a,s'')}, \quad \hat{R}(s,a) = \frac{R(s,a)}{\sum_{s'} N(s,a,s')}$$

This learned model enables the agent to simulate experience and perform updates for any state-action pair, a process referred to as planning (see Algorithm 1). Such planning updates significantly improve the efficiency and performance of learning, and form the basis of the framework used in this work.

Algorithm 1: Dyna-Q

Input: Learning rate α, discount factor γ, exploration rate ϵ, number of planning steps n
$Q(s,a) \leftarrow 0$ for all (s,a)
Transition counts $N(s,a,s') \leftarrow 0$
Cumulative rewards $R_{\text{sum}}(s,a) \leftarrow 0$
Set of observed state-action pairs $\mathcal{M} \leftarrow \emptyset$
while *not converged* **do**
  Observe current state s
  Select action a using ϵ-greedy policy and $Q(s,\cdot)$
  Take action a, observe reward r and next state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
  Update model statistics:
    $N(s,a,s') \leftarrow N(s,a,s') + 1$
    $R_{\text{sum}}(s,a) \leftarrow R_{\text{sum}}(s,a) + r$
    Add (s,a) to $\mathcal{M}$ if not already present
  for $i = 1$ **to** n **do**
    Randomly sample $(\tilde{s},\tilde{a})$ from $\mathcal{M}$
    Estimate transition probabilities $\hat{P}(\cdot \mid \tilde{s},\tilde{a})$ and expected reward $\hat{R}(\tilde{s},\tilde{a})$ from counts
    Sample next state $\tilde{s}'$ according to $\hat{P}(\cdot \mid \tilde{s},\tilde{a})$
      $Q(\tilde{s},\tilde{a}) \leftarrow Q(\tilde{s},\tilde{a}) + \alpha\big[\hat{R}(\tilde{s},\tilde{a}) + \gamma \max_{a'} Q(\tilde{s}',a') - Q(\tilde{s},\tilde{a})\big]$

3 Related Work

This section provides a brief overview of prior work relevant to the current research. It begins with a discussion of commonly used random exploration strategies, followed by an outline of popular intrinsic motivation methods. The final part focuses on studies that investigate the decoupling of exploration and exploitation, highlighting their limitations and how they relate to the present work.

3.1 Uninformed Exploration

Uninformed or undirected exploration methods are approaches that do not depend on any exploration-specific knowledge. These methods generally rely solely on randomness and are commonly used in many RL tasks due to their simplicity. The most basic method in this category is the random walk, which was utilized in action selection mechanisms by Anderson [2].

Another basic random exploration strategy is the ϵ-greedy approach [42], described in Section 2.4. This approach gained its popularity as it required only a single parameter and did not necessitate any additional memory for tracking exploration. Despite this, ϵ-greedy exploration remains inefficient, as it often forces the agent to choose sub-optimal actions, even after sufficient learning [45].

Another method that is considered to engage in more strategic exploration is Softmax action selection [5]. In this approach, the probability of selecting an action is proportional to its estimated value and calculated using a Boltzmann distribution over the Q-values:

$$P(a \mid s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}$$

where τ is the temperature parameter that controls the randomness of the action selection. However, a limitation of softmax exploration is that it cannot be directly applied to continuous state-action spaces [15]. Additionally, it does not account for the uncertainty in the state-action value estimates.

Noise perturbation, on the other hand, is another random method commonly used in continuous action spaces such as robot control [46]. This approach involves adding noise perturbations to the learned policy: $\pi'(s) = \pi(s) + \mathcal{N}$, where $\mathcal{N}$ is a noise term sampled from a stochastic process, such as a Gaussian distribution for continuous action spaces or a Dirichlet process for discrete ones [22].

Despite their simplicity and theoretical guarantees, uninformed exploration methods often require hundreds of millions of environment interactions to learn effectively, highlighting the need for more informative and targeted exploration strategies.

3.2 Integrated Intrinsic Motivation Methods

Several methods have been developed to obtain more informative explorational trajectories. One of the most prevalent methods is intrinsic motivation or bonus-based methods, which are the central focus of this paper.

Count-based IM

One of the earliest forms of IM methods is count-based exploration, which utilizes the state, action counts to calculate intrinsic bonuses [33, 25, 27, 18, 11, 44, 20].

Although not explicitly framed as intrinsic motivation, one of the earliest ideas of incentivizing exploration through count-based reward bonuses was introduced in Sutton's Dyna-Q+ framework [40]. This approach extends the original Dyna architecture by incorporating a recency-based bonus, encouraging the agent to revisit less recently explored state-action pairs. The Dyna-Q+ agent demonstrated superior performance compared to the standard Dyna-Q agent in two non-stationary environments. This influential experiment will be revisited in a later section.

Despite the near-optimal performance guarantees achieved in tabular settings, these methods are generally not suitable for environments with continuous state-action spaces [1]. To address this issue, several algorithms have extended count-based methods by introducing function approximation of the state-action counts [4, 29, 28]. These approaches estimate pseudo-counts via density models and compute intrinsic rewards based on these approximations.

Prediction-based IM

In 1991, Schmidhuber proposed a theoretical model in which intrinsic rewards are derived from changes in an agent's knowledge of the environment, measured through discrepancies in the prediction error of the next state given the current state and action [36]. Interestingly, although this bonus is implemented quite differently from count-based novelty methods, both share a similar underlying principle: rarely visited state-action pairs typically lead to greater uncertainty and thus higher prediction errors, which in turn result in larger intrinsic rewards. However, prediction-based intrinsic motivation approaches are highly sensitive to the quality of the predictive model and may perform poorly in stochastic or noisy environments, where high prediction errors do not necessarily indicate meaningful or learnable structure. Nevertheless, the principle of prediction-based intrinsic rewards laid the groundwork for several widely used IM methods in deep reinforcement learning, such as the Intrinsic Curiosity Module (ICM) [30], and Random Network Distillation (RND) [7].

3.3 Decoupled Exploration and Exploitation

Most modern intrinsic motivation methods optimize exploration and exploitation jointly, typically by improving exploration through returns obtained during exploitation [47, 26, 24]. Although such approaches can lead to improved exploration and consequently better exploitation [19], the parameters may still converge to a local optimum due to the chicken-and-egg problem between the incentive to explore and the motivation to exploit. This issue becomes especially pronounced when the two objectives interfere, such as in environments with deceptive rewards [49]. To address this limitation, decouled IM methods have recently been proposed [38, 12, 10]. In this setting, the exploration and exploitation objectives are typically formulated as separate optimization problems. This is often done in an offline manner, where a behavior policy collects informative trajectories aimed at learning the environment dynamics . The learned dynamics are then used by the exploitative policy to optimize performance and learn the optimal solution.

Nonetheless, the decoupling of the two processes is not the central aim in most existing studies. Since these methods do not operate in the same way as the previously combined approaches, it is difficult to conclude whether the performance improvement is due to the decoupling itself or simply a result of a more effective exploration strategy. However, two papers explicitly formulate this question as the primary focus of their research [49, 35]. In these works, existing intrinsic motivation mechanisms (e.g., RND and ICM) are employed within a decoupled framework, where one behavior

policy is trained using intrinsic or intrinsic-extrinsic rewards, and a separate exploitative policy is trained solely on extrinsic rewards. The decoupled approach is then compared to the combined approach in the environments with deceptive rewards, demonstrating improved performance over the integrated methods.

However, the decoupling of explorational and exploitation polices has so far been tested only in the realm of deep reinforcement learning methods such as DQN or PPO [35]. This complicates comparisons, as additional factors like variance introduced by function approximation can obscure the effects of decoupling. To mitigate this, I propose returning to the basics by investigating classical tabular agents with decoupled approaches, thereby eliminating potential interference from other model components.

4 Methodology

This section describes the baseline model used for comparison and outlines the implementation details of the proposed decoupled approach. A small reference table at the end summarizes all models tested and compared in this paper.

4.1 Baselines

For the baseline model architecture, the Dyna-Q framework was selected due to its well-established status as a tabular method based on the standard Q-learning update rule. More specifically, we employ two variations of this algorithm:

- Vanilla Dyna-Q: Introduced in Section 2.5, this is the standard Dyna-Q agent using tabular Q-learning with planning updates based solely on extrinsic rewards.
- Dyna-Q+: This variant adds an intrinsic bonus to the extrinsic reward during planning updates, computed as: $r' = r'_e + \kappa r_i$, where r'_e is the simulated extrinsic reward, r_i is the intrinsic reward, and k is the scaling factor.

Dyna-Q+ Design

Although DynaQ+ is a well-known algorithm frequently cited in reinforcement learning textbooks, there is no publicly available code for its exact implementation. Therefore, in the first stage of my experimentation, I analyzed two possible variations of DynaQ+.

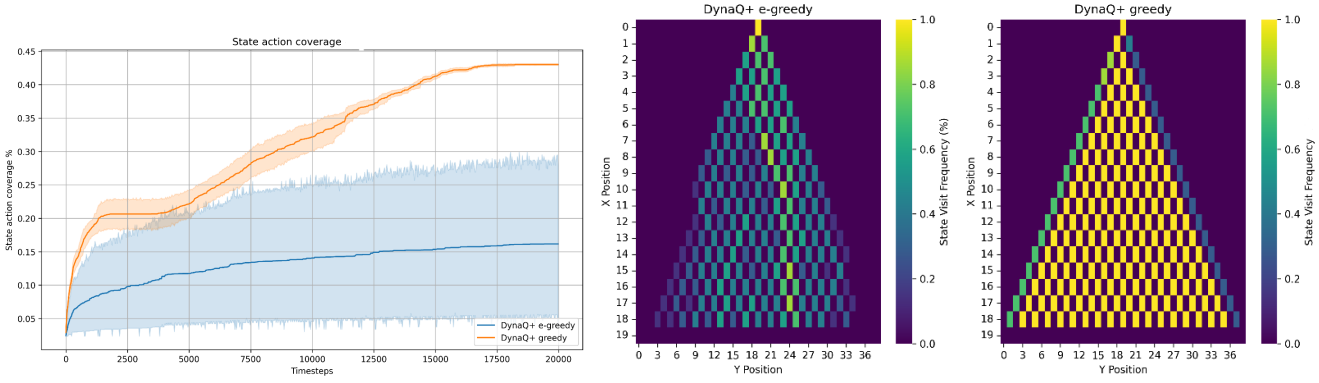
The first variation follows the standard approach, where sample updates are performed only on state-action pairs that have been previously visited. Although the intrinsic bonus is added to the extrinsic reward during these updates, these rewards alone cannot guide the agent through the environment. This is because the agent must first visit all state-action pairs before they can be sampled. Consequently, the agent's exploration must be guided both by intrinsic motivation and by some random exploration method, such as ϵ-greedy. A sensible strategy would be to start with $epsilon = 1$ and slowly decay it over time, thereby gradually shifting from random to targeted exploration. Nevertheless, if the state-action space is large, this method might prove ineffective because, toward the end, ϵ becomes small, making it unlikely for the agent to discover unvisited state-action pairs. Another approach would be to optimistically initialize the Q-values, however, this can be challenging since finding an appropriate starting point is difficult, particularly in environments with sparse or deceptive rewards.

The second variation of the Dyna-Q+ agent employs a greedy action selection strategy but samples random state-action pairs during the planning updates, which were not necessarily previously visited [16]. If the sampled state-action pair had never been visited before, the reward is assumed to be zero, and the next state is set to be the same as the current state (i.e., a self-loop). Otherwise, the next state and extrinsic reward are sampled from the model using transition and reward counts, as described by the following equation:

$$(s', r_e) = \begin{cases} (s, 0), & \text{if } (s, a) \text{ is unvisited} \\ \text{model}(s, a), & \text{otherwise} \end{cases}$$

In this variation of Dyna-Q+, the agent continues to repeat the same actions as in the previous episode until the accumulated intrinsic bonus from trying alternative actions becomes greater than that of the previously selected ones. Once the agent begins to explore other actions, the model is updated, and intrinsic bonuses start to propagate through the state-action space.

To evaluate this hypothesis, I conducted a preliminary experiment aimed at assessing the quality of state-action space exploration. For this purpose, a binary tree environment was designed. At each state, the agent can choose to move either downward-left or downward-right. When the agent reaches a leaf node, the episode ends and resets. All state-action pairs yield a reward of zero, thereby removing any extrinsic learning signals. The results of this experiment are shown in Figure 1.



(a) (b)

Figure 1: **State-action** coverage (%) of the binary tree environment (a), and **state** visitation frequency of each state (b) for Visited-Only (left) and Random Sampling (right) DynaQ+ variants. Both models were tested using the same intrinsic reward weight ($\kappa = 0.0008$), learning rate ($\alpha = 0.2$), discount factor ($\gamma = 0.95$), and 5 planning steps. The Visited-Only DynaQ+ model was tested with an ϵ-greedy policy starting at $\epsilon = 1$, decaying by a factor of 0.999 per step to a minimum of 0.05. Results were averaged over 30 runs. As illustrated in both figures, the Random Sampling Dyna-Q+ agent demonstrates more efficient exploration of the environment compared to the Visited-Only DynaQ+ variation.

As shown in Figure 1, the Random-Sampling DynaQ+ variant with a greedy action selection policy outperforms Visited-Only DynaQ+ with an ϵ-greedy behavioral policy. To leverage the strengths of both approaches, one could consider a hybrid method that combines Random-Sampling DynaQ+ with a decaying-ϵ policy. Such a variant could benefit from more effective early-stage exploration while retaining the ability to generalize across unvisited state-action pairs through random-based state-action sampling.

Nevertheless, for the scope of this study, I utilize the Random-Sampling DynaQ+ with greedy action selection. This choice ensures that the agent's exploration is driven solely by intrinsic rewards, allowing a more controlled comparison with traditional random exploration methods that do not incorporate intrinsic motivation.

4.2 Decoupled DynaQ+

The proposed approach aims to decouple exploration and exploitation by introducing two separate Q-functions:

- Q_{explore}: This function guides the agent's navigation through the environment and is updated only during learning updates, using either intrinsic reward κr_i alone or a combination of intrinsic and extrinsic rewards $r'_e + \kappa r_i$.
- Q_{exploit}: Although not actively used by the agent during environment interaction, this function is used to find the optimal sequence of actions. It is updated purely with extrinsic rewards during both active and simulated updates.

Similar to the baseline Dyna-Q+ model, the Decoupled Dyna-Q+ agent employs a greedy policy over Q_{explore} for action selection and performs planning updates on randomly sampled state-action pairs from the environment. Additionally, depending on whether Q_{explore} utilizes only intrinsic rewards or a combination of both intrinsic and extrinsic rewards, two variants are explored: **Decoupled Dyna-Q+ (I)** and **Decoupled Dyna-Q+ (I+E)**, respectively. An overview of the proposed approach is presented in the following table.

Algorithm 2: Decoupled Dyna-Q with novelty bonus

Input: Learning rate α, discount factor γ, exploration rate ϵ, planning steps n, intrinsic weight κ, extrinsic weight $\lambda \in \{0, 1\}$

$Q_{\text{explore}}(s, a) \leftarrow 0$, $Q_{\text{exploit}}(s, a) \leftarrow 0$ for all (s, a)

Transition counts $N(s, a, s') \leftarrow 0$

Cumulative rewards $R_{\text{sum}}(s, a) \leftarrow 0$

Observed state-action pairs $\mathcal{M} \leftarrow \emptyset$

while *not converged* **do**

- Observe current state s
- Select action a greedily from $Q_{\text{explore}}(s, \cdot)$
- Execute a, observe reward r and next state s'
- $Q_{\text{exploit}}(s, a) \leftarrow Q_{\text{exploit}}(s, a) + \alpha\big[r + \gamma \max_{a'} Q_{\text{exploit}}(s', a') - Q_{\text{exploit}}(s, a)\big]$
- Update model:
  - $N(s, a, s') \leftarrow N(s, a, s') + 1$
  - $R_{\text{sum}}(s, a) \leftarrow R_{\text{sum}}(s, a) + r$
  - Add (s, a) to $\mathcal{M}$ if not present
- **for** $i = 1$ **to** n **do**
  - Sample $(\tilde{s}, \tilde{a})$
  - **if** $N(\tilde{s}, \tilde{a}, \cdot) = 0$ **then**
    - $\hat{r} \leftarrow 0, \quad \tilde{s}' \leftarrow \tilde{s}$
  - **else**
    - Estimate $\hat{P}(\cdot|\tilde{s}, \tilde{a})$, $\hat{R}(\tilde{s}, \tilde{a})$ from counts
    - Sample $\tilde{s}' \sim \hat{P}(\cdot|\tilde{s}, \tilde{a})$
    - $\hat{r} \leftarrow \hat{R}(\tilde{s}, \tilde{a})$
  - $Q_{\text{exploit}}(\tilde{s}, \tilde{a}) \leftarrow Q_{\text{exploit}}(\tilde{s}, \tilde{a}) + \alpha\big[\hat{r} + \gamma \max_{a'} Q_{\text{exploit}}(\tilde{s}', a') - Q_{\text{exploit}}(\tilde{s}, \tilde{a})\big]$
  - $\tilde{r} \leftarrow \lambda \cdot \hat{r} + \kappa \cdot r_{\text{int}}(\tilde{s}, \tilde{a})$
  - $Q_{\text{explore}}(\tilde{s}, \tilde{a}) \leftarrow Q_{\text{explore}}(\tilde{s}, \tilde{a}) + \alpha\big[\tilde{r} + \gamma \max_{a'} Q_{\text{explore}}(\tilde{s}', a') - Q_{\text{explore}}(\tilde{s}, \tilde{a})\big]$

4.3 Intrinsic Reward Methods

For both the baseline Dyna-Q+ and the decoupled variants, two types of count-based intrinsic bonuses were employed: recency and novelty, as introduced in Section 2.4. These variations were not used to compare which intrinsic motivation strategy is superior, but rather to verify that the observed effects are consistent across different IM mechanisms. In the main experiments, recency-based bonuses were used throughout, while results with novelty-based bonuses are provided in the appendix for completeness. This choice was arbitrary and does not reflect any methodological preference.

4.4 Overview of Model Configurations

The table below provides an overview of the four models used in the experiments (two baselines, Dyna-Q and Dyna-Q+, and two decoupled variants, Decoupled DynaQ+ (I) and Decoupled DynaQ+ (I+E)), including the rewards used to update their respective Q-functions, as well as the behavioral policy applied in each model.

	Dyna-Q	Dyna-Q+	Decoupled Dyna-Q+ (I)	Decoupled Dyna-Q+ (I+E)
Q_{explore}	–	–	κr_i	$r_e + \kappa r_i$
Q_{exploit}	r_e	$r_e + \kappa r_i$	r_i	r_i
Behavioral Policy	ϵ-greedy	Greedy	Greedy	Greedy

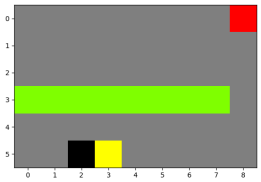
Table 1: Model Overview

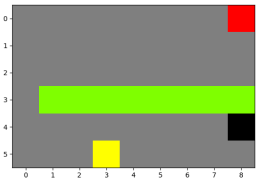
5 Experimental Set-Up

In this section, we introduce the experimental setup used to address the research questions. Specifically, we present the environment in which the models were tested, the evaluation metrics used to assess model performance, and the hyperparameter settings employed during training.

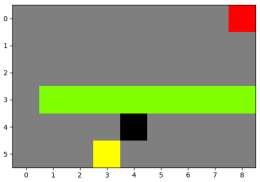
5.1 Environments

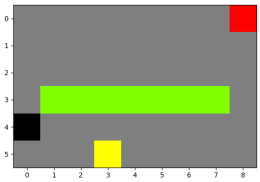
The decoupled approaches were evaluated against baseline models across five environments, depicted in Figure 2. All environments are characterized by sparse and uninformative rewards, making them particularly challenging for standard methods. The difficulty of the environments increases progressively, and the specific challenges posed by each environment are discussed below.



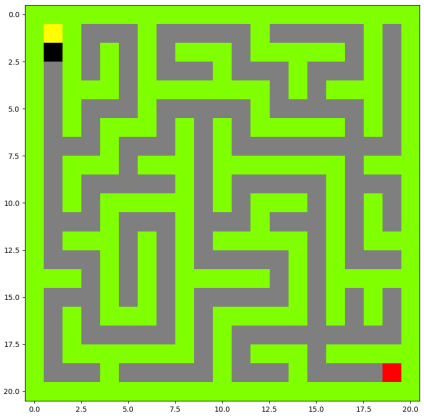


(a) Blocking Maze before (left) and after (right) the change.

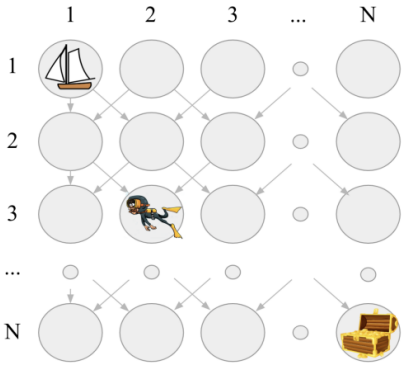




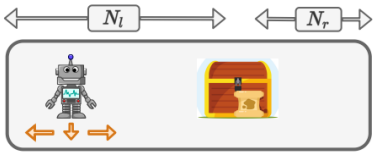
(b) Shortcut Maze before (left) and after (right) the change.



(c) Maze environment



(d) DeepSea environment, from [50]



(e) Hallway environment, from [35]

Figure 2: Five environments used for model evaluation. The first level of difficulty is the Maze environment (c), characterized by sparse rewards: all state-action pairs yield a reward of 0, except for the goal location, which yields 1. The Maze and Shortcut Maze environments (a, b) introduce an additional challenge of non-stationarity, as the environment changes throughout training. The DeepSea environment (d) introduces deceptive rewards that make exploitation particularly difficult, while the Hallway environment (e) features deceptive rewards in terms of both exploration and exploitation.

Maze: An environment with four possible actions: up, down, left, and right. The agent starts in the upper-right corner and must reach the goal location in the lower-left corner, depicted in Figure 2c by yellow and red, respectively. The primary challenge of this environment is the *sparsity of rewards*: the agent receives a reward of 1 only upon reaching the goal state and 0 otherwise.

Blocking Maze: The Blocking Maze environment was first introduced by Sutton to compare Dyna-Q to Dyna-Q+ [41]. This environment is a simple maze with the starting position in the lower center and the goal in the upper-right corner, separated by a wall in the middle (Figure 2a). At each state, the agent can move left, right, up, or down. As in the previous environment, the agent receives a reward of 1 only upon reaching the goal and 0 otherwise. The main challenge in this environment, in addition to sparse rewards, is *non-stationarity*. During the first half of the learning process, the passage through the wall is located on the left side. In the second half, the wall shifts one cell to the right—blocking the agent's current path and opening a new one on the opposite side. This forces the agent to re-explore the maze to discover the new path. However, as the change in the wall directly interferes with the optimal solution previously found, the agent has no choice but to explore in order to discover a new solution. This may allow even the standard Dyna-Q, without intrinsic bonuses, to eventually find the new path. The key question in this environment is: Which agent will find the new path faster?

Shortcut Maze: Similar setup as in the Blocking Maze, but with a small differences. Initially, the passage through the wall is open on the left side (Figure 2b). After some time, a new shortcut opens on the right side without blocking the original path. Unlike in the Blocking Maze, where the agent is forced to re-explore due to the blocked path, here the agent must continue exploration even after discovering a working solution in order to find the improved path, making the environment more challenging.

DeepSea: The DeepSea environment [50], visualized in Figure 2d, proposes a new challenge to the models. It adds a layer of complexity through *unforgiving and deceptive rewards*. The environment consists of an $N \times N$ grid, with the starting location at the top left and the goal location at the bottom right. At each timestep, the agent moves one row down, and the episode ends once the bottom row is reached. At each step, the agent can choose to go either left or right, receiving a reward of 0 or $-\frac{0.01}{N}$, respectively. In addition, the agent receives a reward of 1 if it reaches the goal location. This setup makes the rewards not only sparse, but also deceptive from an exploitation point of view. The agent must learn to "endure" small negative rewards in order to reach the goal, and one mistake is enough to make the goal *unreachable*. The difficulty of the environment can be controlled by the size N: higher values of N make it more difficult for the agent to obtain the optimal return of 0.99. In our experiments, we test the models in DeepSea with $N \in \{8, 10, 14, 20, 24, 30\}$.

Hallway: The Hallway environment was originally proposed by Shafer et. al. [35] to increase the challenge introduced by DeepSea. The additional difficulty in this environment lies in the fact that the rewards are not only *deceptive in terms of exploitation, but also in terms of exploration.* The environment represents a 1D hallway, with the starting location on the left side (Figure 2e). At each step, the agent can choose from three possible actions: moving left (reward of 0), staying in

place (reward of -0.01), or moving right (reward of -0.01). The goal location can be reached by moving N_l cells to the right. Unlike in DeepSea, where the goal is located at the far end of the grid, in Hallway, the goal is not necessarily at the right end. There may be an additional N_r empty cells beyond the goal location. A positive reward of 1 is granted each time the agent chooses to remain at the goal location. An episode terminates after $2N_l$ timesteps. In our experiments, N_l was set to 10. The value of N_r was set to either 0 or 10. The main challenge of this environment is the conflict between the objective of continued exploration and the objective of staying at the discovered goal location to collect the reward. [2]

5.2 Evaluation Metrics

To assess the performance of the models, we run periodic evaluations in which each agent follows a greedy policy based on Q_{exploit}. Since the agent's policy and all environments are deterministic, a single run of the evaluation is performed. Each evaluation consists of a single run of an episode with a maximum length of 100 timesteps, and evaluations are performed every 25 training timesteps. To answer the three research questions defined in Section 1.4, we use the following three metrics:

1. **Agent Performance:** Measured by the Area Under the Learning Curve (AULC) of the evaluation returns, defined by the following equation:

$$\text{AULC} = \sum_{i=1}^{N} R_i$$

   where R_i is the discounted cumulative reward collected during the i-th evaluation and N is the total number of evaluation episodes.

2. **State-Action Coverage:** Measured as the percentage of distinct state-action pairs visited throughout training:

$$\text{Coverage} = \frac{|\{(s,a) \in \mathcal{D}\}|}{|\mathcal{S} \times \mathcal{A}|} \times 100\%$$

   where $\mathcal{D}$ is the set of all encountered state-action pairs, and $\mathcal{S} \times \mathcal{A}$ is the full state-action space.

3. **Hyperparameter Variance:** To measure the robustness of the models to exploration hyperparameter, we report the mean and variance of the final evaluation returns for different values of the exploration weight $\kappa \in \{0.00001, 0.0001, 0.0005, 0.001, 0.005\}$:

$$\bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i, \quad \text{Var}_\kappa = \frac{1}{N}\sum_{i=1}^{N}(R_i - \bar{R})^2$$

   A high mean return with low variance indicates that the model performs well and is robust to the hyperparameter choice.

[2] In the original environment, a reward of 1 was given only after staying at the goal for 10 steps, which required memory. Since tabular agents lack memory, the requirement was reduced to 1 step. To introduce minimal memory, the state representation was extended to a tuple `(previous_state, current_state)`, allowing the agent to retain one-step history.

5.3 Hyperparameter Configuration

To ensure a fair comparison between the agents, all common hyperparameters were fixed to the same values across models and environments, as shown in Table 2. An exception was the intrinsic bonus weight κ, which varies across environments but was kept consistent across all models within each environment: Table 3. These values were selected based on a grid search procedure and represent the best-performing configuration averaged over 10 repetitions. For the DeepSea and Hallway environments, the grid search was performed with fixed sizes of $N = 10$ and $N_l = 10$, $N_r = 10$, respectively. Additionally, the discount factor γ was set to 0.95 for all experiments. To speed up the subsequent experiments, the number of planning updates was set to 5 for all models and environment configurations. This design choice was found to affect all models similarly, and thus does not bias the comparison.

All experiments were run for the number of timesteps specified in Table 4 for each environment. For the Shortcut and Blocking Maze environments, the timestep schedules were initialized according to the original experimental setup proposed by Sutton [41]. In the Maze, DeepSea, and Hallway environments, the total number of timesteps was set to 15,000, which roughly corresponds to the time required for the baseline Dyna-Q+ algorithm to learn an optimal or near-optimal policy.

Table 2: Fixed hyperparameters used across all environments for each model.

Hyperparameter	**DynaQ**	**DynaQ+**	**Dec. DynaQ+ (I)**	**Dec. DynaQ+ (I&E)**
Learning rate (α)	0.2	0.2	0.2	0.2
Discount factor (γ)	0.95	0.95	0.95	0.95
Exploration rate (ϵ)	1	–	–	–
Discount factor for ϵ	0.999	–	–	–
Planning steps	5	5	5	5
Bonus type	Recency	Recency	Recency	Recency

Table 3: Intrinsic bonus weight κ used across environments for all models.

	IM type	**Maze**	**BlockingMaze**	**Shortcut**	**DeepSea**	**Hallway**
κ	recency	0.00001	0.0008	0.0008	0.008	0.01
κ	novelty	0.001	0.05	0.01	0.01	0.1

Table 4: Total training timesteps and change point per environment.

Environment	**Maze**	**BlockingMaze**	**Shortcut**	**DeepSea**	**Hallway**
Total timesteps	15,000	6,000	8,000	15,000	15,000
Env. change timestep	–	1,000	3,000	–	–

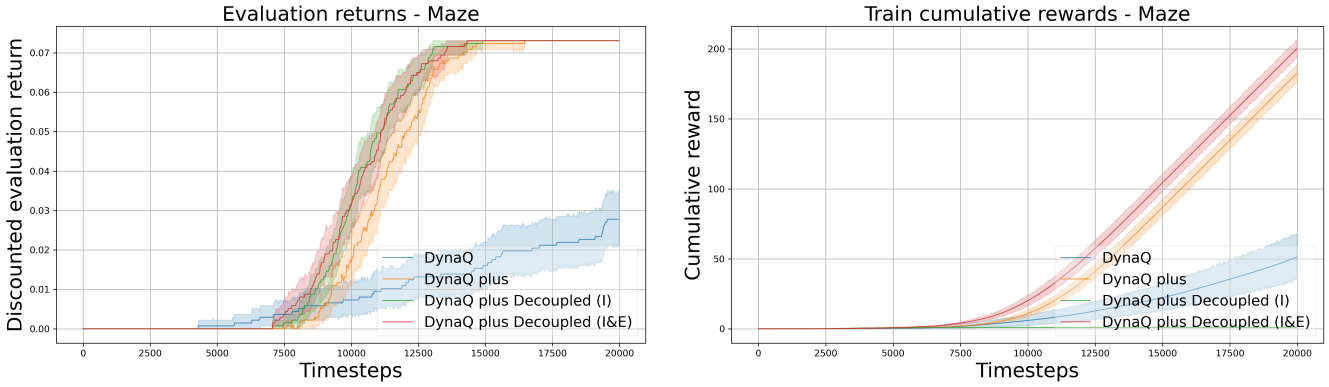
6 Results

In this section, we present and briefly discuss the experimental results for each environment. To address the research questions individually, we provide supporting metrics and visualizations for each of them in a separate subsection. Details of the model's hyperparameters are listed in Table 2. Additional results of the models using novelty-based intrinsic motivation are presented in Appendix C. All results were averaged over 100 repetitions.

6.1 Agent Performance

In this subsection, we aim to answer the first research question by comparing the performance of the agents across all tested environments. The results are visualized using evaluation return and cumulative training return curves. Exact AULC scores for each agent–environment pair are provided in the Appendix A, along with additional visualizations of Q-value heatmaps that help illustrate the agents' final behavior presented in the Appendix B.

Maze



(a) (b)

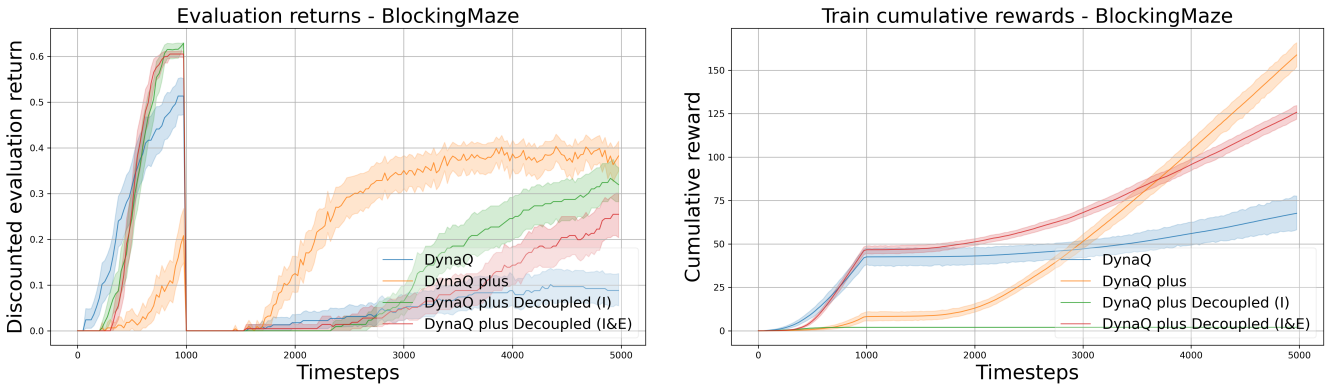
Figure 3: Agent performance in the sparse reward Maze environment: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

Several conclusions can be drawn about the agent's performance in the sparse reward environment, as visualized in Figure 3. First, the DynaQ+ agent outperforms the standard DynaQ, as expected in this type of environment, due to its more targeted exploratory behavior driven by intrinsic bonuses. Secondly, the decoupled approaches demonstrate slightly better performance compared to DynaQ+, likely because they maintain a separate Q-value table that is updated exclusively with extrinsic rewards. This separation allows the agents to more effectively disentangle exploration from exploitation, leading to faster and more stable learning of accurate Q-values.

Regarding the training results, the observed outcomes align with expectations. Both DynaQ+ and Decoupled DynaQ+ (I+E) attain the highest cumulative training rewards. This is primarily due to their reliance on extrinsic rewards, as the influence of intrinsic motivation is minimal with a

low intrinsic reward coefficient ($\kappa = 0.00001$). Higher values of κ have would cause unstable and inefficient learning behavior. Next in performance is the DynaQ agent, which performs moderately during training, followed by the Decoupled DynaQ (I) agent. This outcome is expected, as the latter relies solely on intrinsic rewards. Consequently, its training performance is limited to occasionally visiting states that have not been explored for some time.

Blocking Maze



(a) (b)

Figure 4: Agent performance in the non-stationary Blocking Maze environment with sparse rewards: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

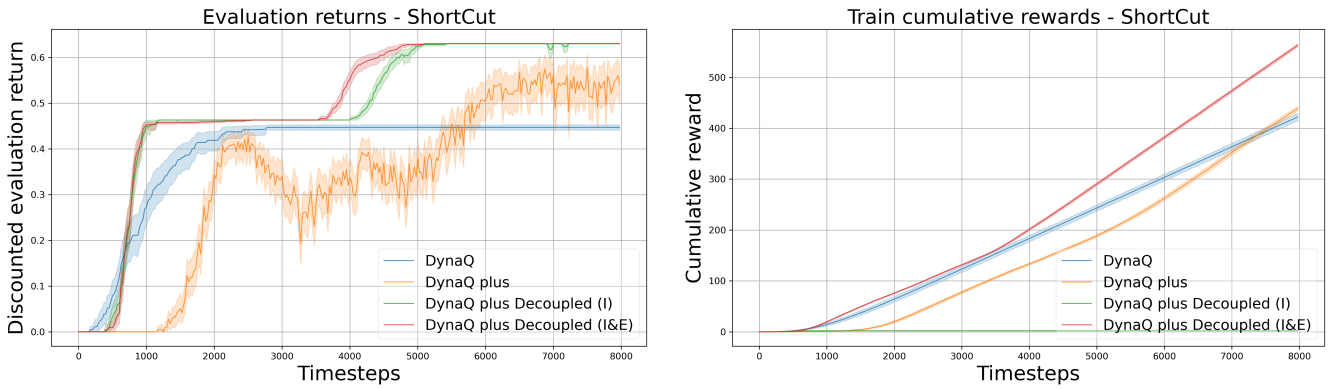
In the Blocking Maze environment, we observe several interesting results. Firstly, as shown in Figure 4a, *before* the change in the environment, the Decoupled approaches outperform both DynaQ+ and standard DynaQ. This is consistent with the experimental results from the previous environment, where the Decoupled methods also achieved slightly better performance. The reason for DynaQ+'s slower performance during this phase is that 1000 timesteps were not sufficient to reduce the exploratory bias introduced by its intrinsic rewards.

However, *after* the pathway is blocked at timestep 1000, we observe that DynaQ+ outperforms the Decoupled approaches. This is because it takes time for the Decoupled methods (particularly in their Q_{exploit} components) to unlearn the previously valid model dynamics, specifically that the now-blocked pathway is no longer viable, and to stop propagating inflated Q-values along that path. Even after a new, longer path becomes available, the transition model in the decoupled methods still contains outdated information about nearby states, causing the agent to mistakenly favor the blocked path. Importantly, this behavior also occurs because the blocked path was shorter than the newly opened path. If the reverse were true, the agent would likely have switched to the new, shorter path immediately. Another additional observation is that Decoupled DynaQ+ (I+E) takes slightly longer than Decoupled DynaQ+ (I) to adapt. This is likely because the (I+E) variant visited the blocked path more frequently, and thus needs more time to relearn accurate transition counts. This effect is also visible in the corresponding performance curves and suggests the hypothesis that, in non-stationary environments, fully decoupled exploration based solely on intrinsic rewards may

be more efficient. This is because it avoids bias by promoting more even visitation of state-action pairs, which in turn facilitates easier and more accurate updating of transition counts.

By contrast, although DynaQ+'s Q_{exploit} values are somewhat noisier due to the influence of intrinsic rewards, it adapts more quickly to the environment change and relearns more suitable values after the pathway becomes blocked. Nevertheless, all intrinsic motivation methods still performed better than the baseline DynaQ with random exploration, although even the baseline eventually began to discover the new pathway, as was expected. This is because, in this environment, the agent has no choice but to explore other parts of the state space in order to find an alternative route.

Shortcut Maze



(a) (b)

Figure 5: Agent performance in the non-stationary Shortcut Maze environment with sparse rewards: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

In the slightly more challenging Shortcut Environment, several key observations can be made, as shown in Figure 5. First of all, before the change in the environment, the Decoupled approaches perform best, followed by the DynaQ agent, and finally the DynaQ+ agent, which converges to the available optimal path at around 3000 timesteps.

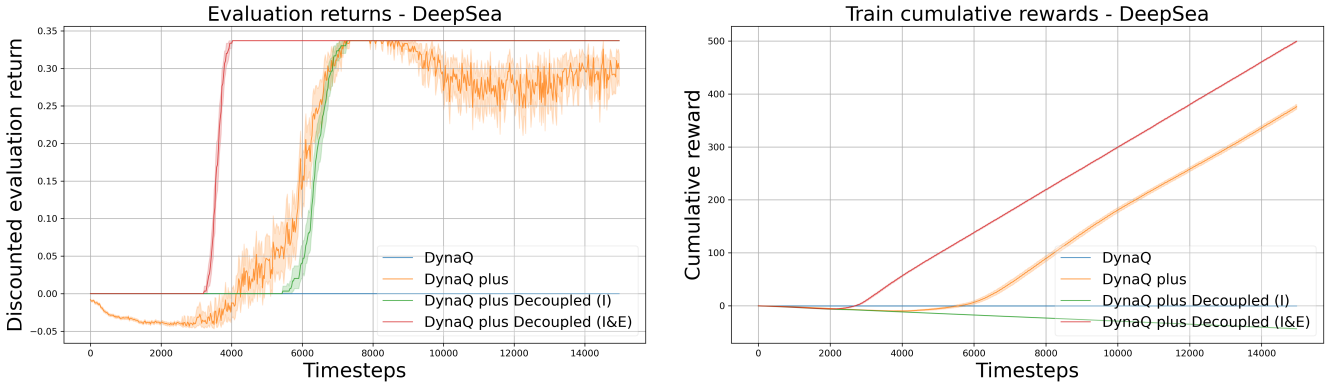
After the shortcut is opened, the standard DynaQ agent, as expected, fails to discover it due to the lack of exploration after converging to the previous optimal policy. In contrast, DynaQ+ begins to explore the environment again as a result of growing intrinsic bonuses. These bonuses introduce a bias in the Q_{exploit} values, as reflected in the temporary drop in performance observed around 3000 timesteps. Although this exploration appears unstable, the agent eventually succeeds in discovering the shortcut.

An intuitive way to reduce this instability would be to decrease the intrinsic bonus weight or introduce a decay rate. However, while this might work in the Maze environment, it is undesirable in the Shortcut Environment, as the agent may then fail to discover the shortcut altogether. This highlights the difficulty of achieving a balance between stable learning and exploratory behavior when relying on a single Q-function for state-action value approximation.

On the other hand, both decoupled approaches successfully find the shortcut without exhibiting temporary bias due to the separation of the exploration and exploitation objectives. The reason

why in this environment, the agents immediately manage to switch to the newly discovered path, unlike their struggle in the Blocking Maze, is that the new path is shorter than the previous one. As a result, even if the old values continue to propagate during planning updates, the Q-value of moving along the new path remains higher than that of the blocked or longer path, guiding the agent to switch promptly.

DeepSea



(a) (b)

Figure 6: Agent performance in the DeepSea environment of size N=20: (a) Discounted evaluation return vs. timesteps. Note that the green line is located below the red curve starting from the 7000th timestep. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

The results of agent performance in the DeepSea environment are illustrated in Figure 6. Firstly, we observe that in the presence of deceptive rewards, the standard DynaQ model fails to find the optimal path. This highlights the limitation of standard Random exploration strategies in such environments. In contrast, intrinsic motivation methods prove to be effective, as they can discover the optimal path. However, for this to happen, the intrinsic reward weight must be carefully tuned: it should slightly exceed the small negative reward but remain lower than the final positive reward.

As seen in the results, the DynaQ+ agent exhibits more unstable behavior due to the bias introduced by intrinsic bonuses, as previously discussed in earlier environment results. Regarding the Decoupled approaches, both models successfully find the optimal path. The Decoupled method that learns from both intrinsic and extrinsic rewards manages to do so more quickly. This can be attributed to the high dimensionality of the state-action space in the DeepSea environment, where reaching the goal state is particularly difficult. A single incorrect action often leads to a different state with no further opportunity to visit the goal. In this setting, the Decoupled DynaQ+(I) agent may visit the goal location only once, and its Q_{exploit} table would then require time to propagate this value through planning updates alone. In contrast, the Decoupled DynaQ(I+E) agent continues to re-explore the goal location due to the presence of extrinsic rewards, resulting in a faster update of the Q_{exploit} values.

The performance of the models across additional maze sizes in the DeepSea environment is shown in Figure 7. This graph confirms the conclusions drawn from the previous results: DynaQ fails

to find the optimal path for all environment sizes. DynaQ+ exhibits relatively high variance, and its performance starts to decline for maze sizes larger than $N > 15$. The Decoupled (I) approach fails beyond $N > 24$, while the Decoupled (E and I) method consistently achieves optimal performance for all tested values of N.

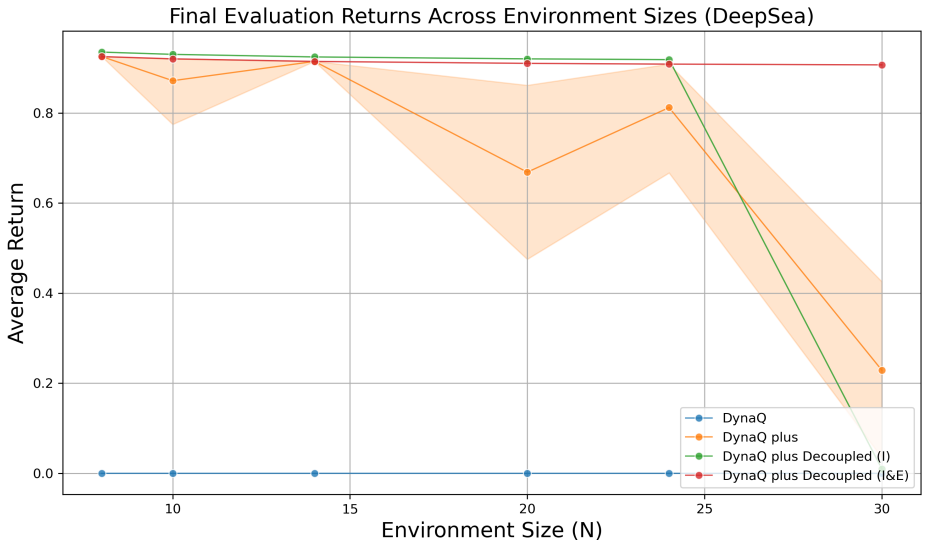
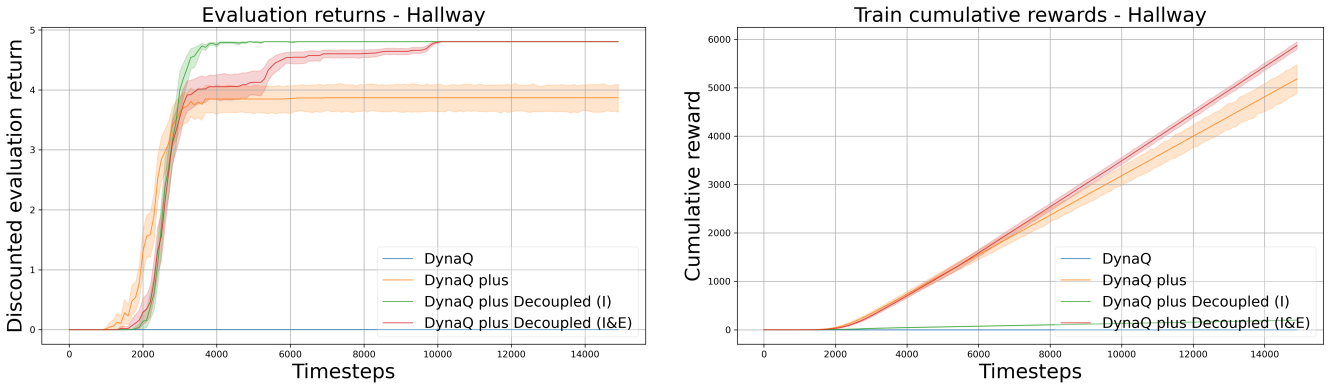


Figure 7: Final not-discounted evaluation returns of the models in the DeepSea environment for different maze sizes $N \in \{8, 10, 14, 20, 24, 30\}$. All models were trained for 20.000 timesteps. Shaded regions indicate 95% confidence intervals.

Hallway



(a) (b)

Figure 8: Agent performance in the Hallway environment with sparse rewards and $N_l = 10$, $N_r = 0$: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

Figure 8 shows the agent performance in the Hallway environment, where the goal is positioned at the far end of a 1D corridor. As in the DeepSea environment, the standard DynaQ agent fails to

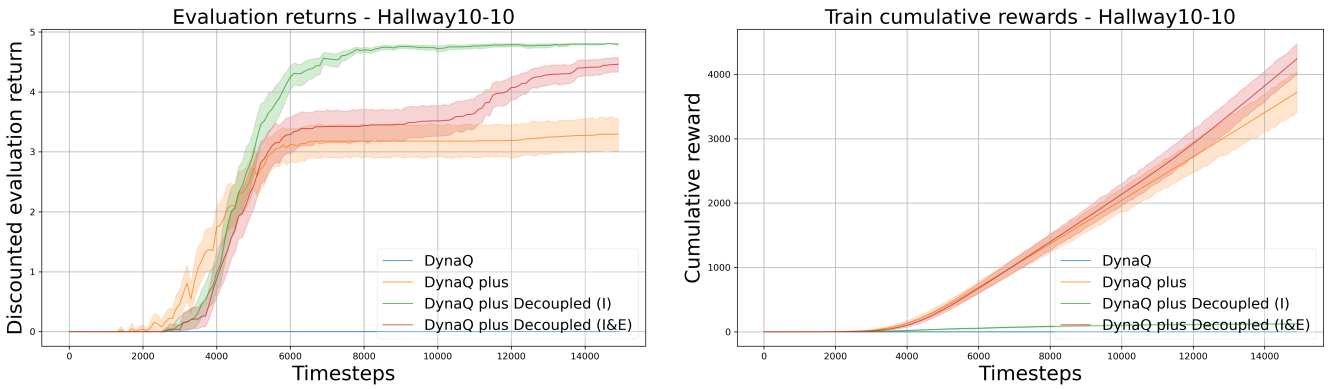
reach the optimal solution, due to the small negative rewards discouraging rightward exploration. The baseline DynaQ+ agent is able to find the goal but converges to a suboptimal policy. This is likely due to increasing intrinsic bonuses that continue to incentivize exploration, even after the goal has been found.

Interestingly, the purely intrinsically motivated agent (Decoupled DynaQ+ (I)) outperforms the intrinsic-extrinsic (I+E) variant. This is somewhat surprising, given that staying at the goal state could be viewed as an exploitative behavior. However, further inspection reveals the reason: the (I+E) agent converges to a suboptimal solution due to its exploitative incentive. Recall that in the Hallway environment, we augment the state space with one-step memory, encoding each state as a tuple of the form $(s_{\text{prev}}, s_{\text{curr}})$. This allows the agent to distinguish between merely visiting state 10 (e.g., $(9, 10)$) and remaining there (e.g., $(10, 10)$). Now, when the I+E agent starts to explore its environment, it accidentally tends to loop through the sequence $(9, 10) \rightarrow (10, 10) \rightarrow (10, 9) \rightarrow (9, 10)$. This loop arises because the agent receives a reward of 1 when first entering the goal state from $(9, 10)$, causing that state to be assigned a high Q-value. Instead of learning to remain in $(10, 10)$, the agent exploits this by repeatedly returning to $(9, 10)$ to relive the high reward transition instead of exploring other actions in state (10, 10). As a result, it fails to converge to the optimal policy of staying at the goal for multiple steps. Over time, the (I+E) agent manages to escape this loop. As the intrinsic bonus of staying in $(10, 10)$ increases, the agent eventually explores this action and updates its value appropriately, making it more attractive than the looping behavior. This also explains why the intrinsic-only agent ultimately reaches better performance. The Decoupled DynaQ+ (I) agent performs better in this setting because it explores independently of the extrinsic reward structure and is less prone to prematurely exploiting misleading transitions.

Originally, this suboptimal solution was not intended in the environment design and can be avoided by assigning the reward only after the agent stays at the goal location for *more* than one timestep. This modification prevents the agent from receiving a reward by taking the 'stay' action in state $(9, 10)$, ensuring that the reward can only be obtained by remaining in state $(10, 10)$. However, I decided to keep the unintended design as it reveals a valuable insight: agents whose exploration is influenced by extrinsic rewards are more likely to fall into suboptimal solutions. Thereby, although pure intrinsic motivation methods may be slower to converge (as seen in the DeepSea environment), they are safer in the sense that they prevent the agent from getting trapped in local minima caused by misleading reward signals.

Figure 9 shows the results for the Hallway environment where ten additional cells were added after the goal location. As we can observe, all intrinsic motivation methods take more time to converge to either an optimal or suboptimal solution. In this setting, the Decoupled DynaQ+ (I) agent converges to the optimal solution at around 6000 timesteps, whereas it required only 4000 timesteps in the previous version of the environment, where the goal was placed at the very end. This difference can be attributed to the increased number of state-action pairs, which naturally requires more time for the agent to sufficiently explore and identify the optimal path.

Another interesting observation is that in this environment, the Decoupled DynaQ+ (I+E) agent converges even more slowly to the optimal solution. A possible explanation is that with a larger state-action space, this model is susceptible to falling into *two* possible local optima. In this version of the Hallway, a second loop emerges on the right side: $(11, 10) \rightarrow (10, 10) \rightarrow (10, 11) \rightarrow (11, 10)$, which forms a second local minimum. The agent might first fall into this loop, then explore the previously discussed suboptimal loop (e.g., involving $(9, 10)$), before eventually discovering the truly optimal strategy. As for DynaQ+, this model again fails to converge to the optimal solution,



(a) (b)

Figure 9: Agent performance in the Hallway environment with sparse rewards and $N_l = 10$, $N_r = 10$: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

further confirming its limitations in environments that require persistent and unbiased exploration.

6.2 State-Action Space Coverage

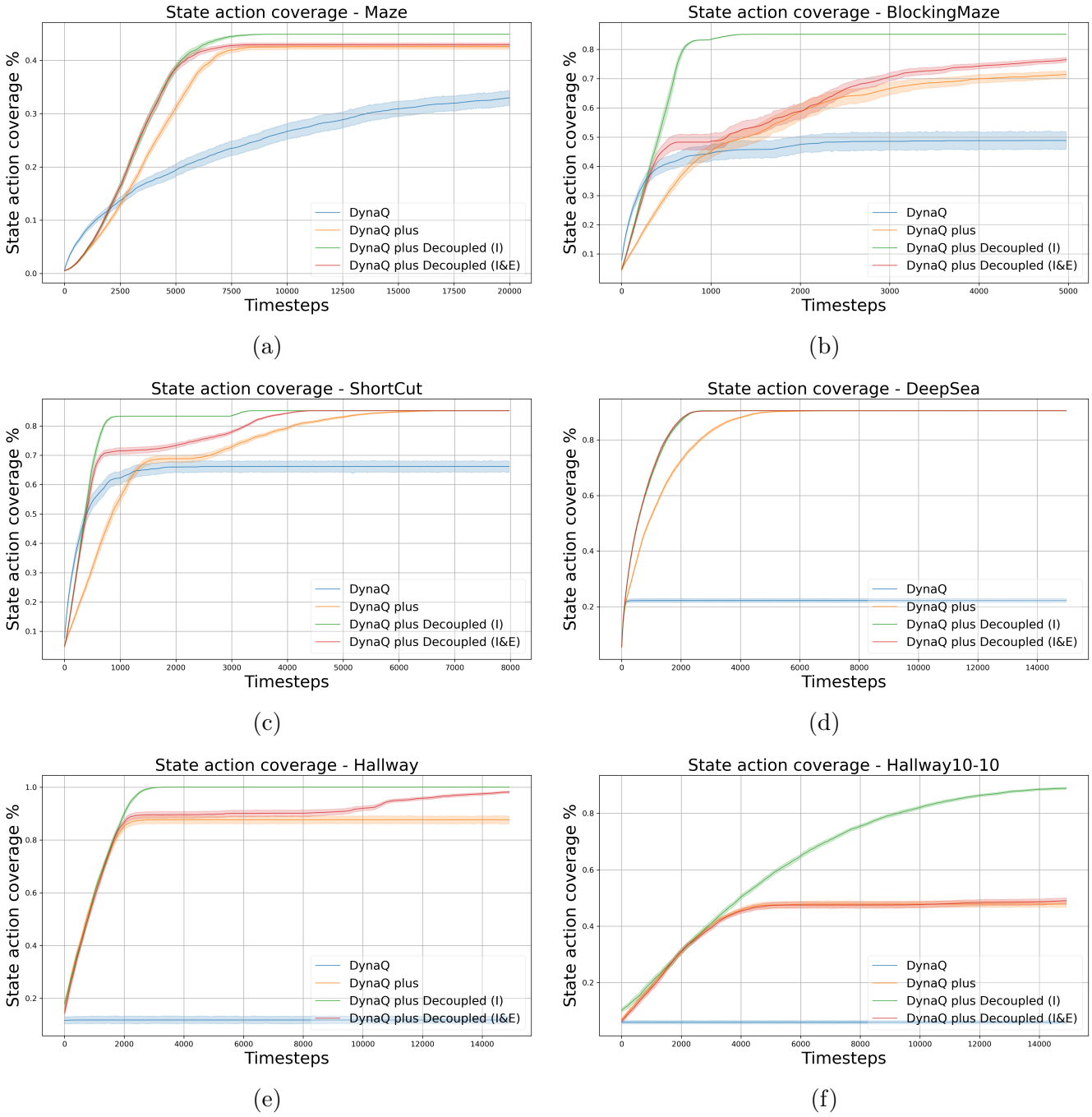
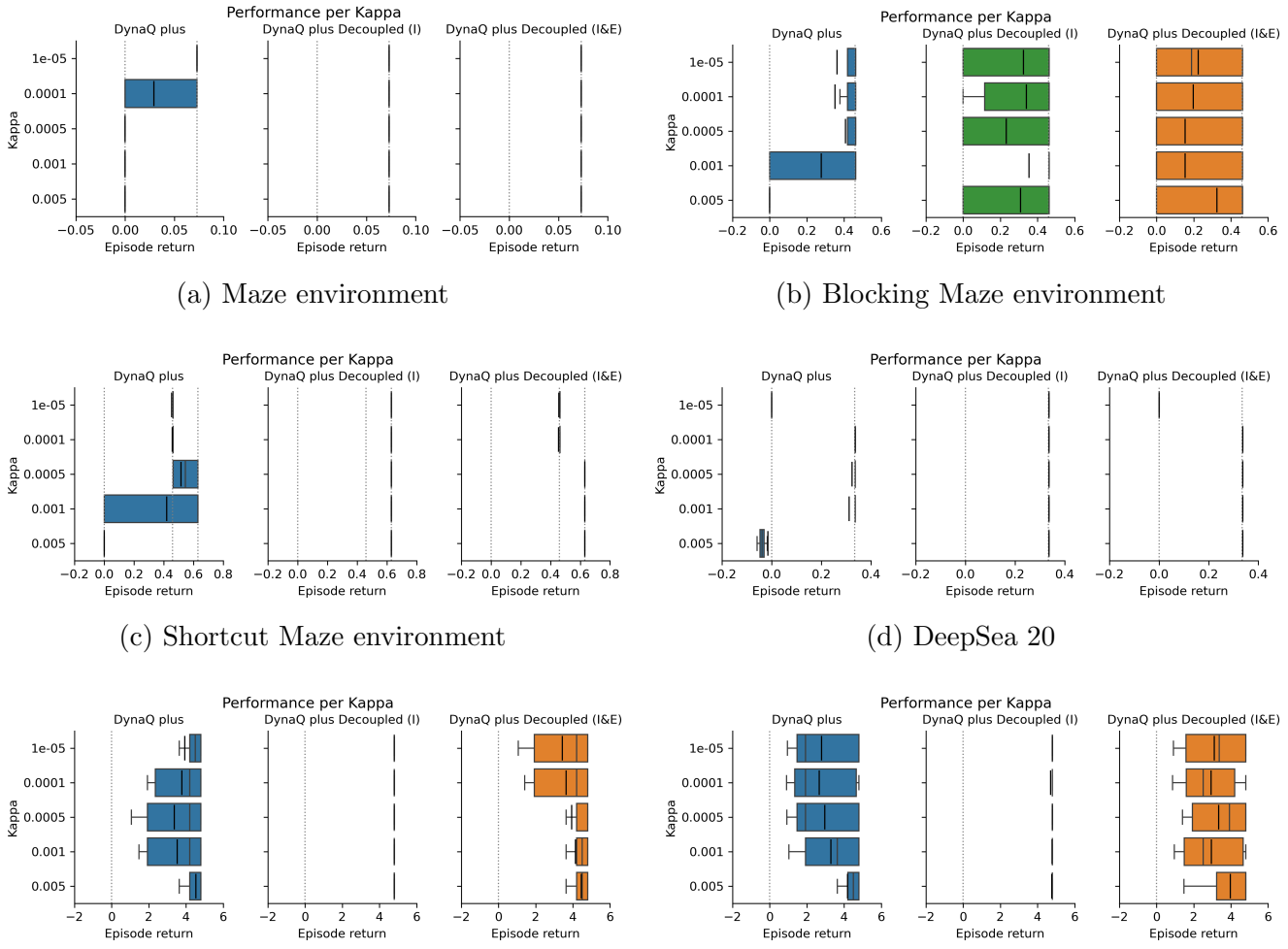


Figure 10: State-action coverage (%) of the models across the tested environments. The x-axis represents the training timesteps, while the y-axis shows the percentage of state-action pairs visited during training.

To address the second research question of whether the decoupled approaches explore the environment more efficiently than the integrated DynaQ+ approach, we refer to Figure 10. First, we observe that intrinsic motivation methods explore the environments more effectively than the

random exploration method (blue line), particularly in sparse reward, non-stationary, and deceptive environments. When comparing decoupled (green and red lines) and integrated approaches (yellow line), we find that the decoupled methods achieve a better exploration in all tested environments. Lastly, comparing the Decoupled (I) and Decoupled (I+E) methods, we see that the approach relying solely on intrinsic rewards explores the environment more thoroughly. This is expected, as the Decoupled DynaQ+ (I) agent is driven purely by exploration and is not influenced by external rewards, leading to more even exploration.

6.3 Hyperparameter Sensibility



(a) Maze environment

(b) Blocking Maze environment

(c) Shortcut Maze environment

(d) DeepSea 20

(e) Hallway 10-0

(f) Hallway 10-10

Figure 11: Average evaluation returns across environments, measured after the training phase with $\kappa \in \{0.00001, 0.0001, 0.0005, 0.001, 0.005\}$. Shaded regions indicate 95% confidence intervals. A method that is insensitive to hyperparameters will have the final average episodic returns concentrated to the right for all hyperparameter values.

The hyperparameter sensitivity of the models is visualized in Figure 11. The mean final returns with the corresponding variance scores can be found in Appendix A. The mean values are represented

by black ticks, and the boxes denote the 95% confidence intervals. Note that the DynaQ model is not included in this visualization since it does not depend on the κ parameter.

From the figure, we observe that the DynaQ+ model is highly sensitive to the value of κ, with performance varying significantly depending on the weight. In contrast, the Decoupled DynaQ+ (I) model is completely insensitive to the scale of κ. Although there is performance variation observed in the Blocking Maze environment, this variation is not attributed to different hyperparameter values but rather to the general performance of the model in this environment, as illustrated in Figure 4a.

The Decoupled DynaQ+ (I+E) model, unlike DynaQ+, which is sensitive to both very small and very large values of κ, is mainly sensitive only to very small values of κ, as shown in Figures 11c 11d, and 11e. This is expected: when κ is too small, the agent becomes overly exploitative, hindering exploration and preventing it from discovering shortcuts (e.g., in the Shortcut environment) or the optimal path (e.g., in DeepSea). On the other hand, increasing κ shifts the agent toward exploratory behavior, making it behave more similarly to Decoupled DynaQ+ (I), which is not detrimental in this context. Nevertheless, we can conclude that Decoupled DynaQ+ (I) is more robust than Decoupled DynaQ+ (I+E), as it is completely independent of the κ value and can therefore be utilized without tuning this parameter.

7 Discussion

In this section, we discuss the main insights gained from the conducted experiments, focusing on the three core research questions addressed in this work. We interpret the results in light of prior research and theoretical expectations. Following this, we outline the key limitations of our approach and conclude with potential directions for future work.

7.1 Key Insights

Model performance:

In terms of model performance, it can be concluded that decoupled approaches outperform the standard DynaQ+ in stationary, non-stationary, and deceptive environments. This is primarily because, in decoupled approaches, the Q-table used for evaluation remains unaffected by intrinsic rewards, which helps avoid biased value propagation. As a result, these methods are able to find the optimal path more efficiently, as demonstrated in the Maze environment. Another factor that contributes to the weaker performance of Dyna-Q+ is its learning instability. As discussed earlier, this can be partially mitigated by introducing a decaying κ parameter or selecting a lower κ value in general. However, such adjustments are not effective in more challenging environments such as the Shortcut Maze, Blocking Maze, and DeepSea, where continuous exploration is essential throughout training. Since intrinsic bonuses are directly accumulated in the Q-values of Dyna-Q+, this leads to performance drops when the agent's behavior is driven too strongly by outdated or misleading exploration signals, before eventually reverting to the optimal policy.

Comparing the two decoupled approaches, the Decoupled Dyna-Q+ (I+E) method generally performs better in environments without sub-optimal solutions, such as the Shortcut and DeepSea environments. This outcome is expected: incorporating extrinsic rewards into the exploratory Q-function encourages the agent to reach goal states more frequently and propagate meaningful reward signals more rapidly, thereby accelerating learning.

However, in environments that contain misleading local optima, such as the Blocking Maze and the Hallway, a more unbiased exploration strategy proves to be more effective. In these settings, the Decoupled Dyna-Q+ (I) agent, which relies solely on intrinsic rewards for exploration, is less likely to become trapped in deceptive loops driven by external rewards. By decoupling exploration entirely from the extrinsic reward signal, this method promotes broader state space coverage and greater resilience against reward-induced bias. As a result, the intrinsic-only agent is also better suited for non-stationary environments. Its uniform exploration facilitates faster re-learning of the transition dynamics after changes in the environment, enabling quicker adaptation to the new optimal policy. Finally, an important observation is that maximizing rewards during training is not strictly necessary for learning an optimal policy, as evidenced by the performance of the Decoupled Dyna-Q+ (I) model. Nonetheless, if the goal is to maximize cumulative rewards throughout the entire learning process then the (I+E) variant clearly provides a more favorable trade-off by balancing reward acquisition and exploration.

State-action coverage:

In terms of state-action coverage, it can be concluded that decoupled approaches also explore the environment better. Decoupled DynaQ+ (I) also shows better exploration compared to the (I+E) method due to the fact that its exploration is not biased by the extrinsic rewards. However, as can be seen in the Shortcut and Blocking Maze environments, as well as in the Maze, better

exploration does not necessarily lead to better performance. Although the Decoupled DynaQ+ (I) method explores more, the (I+E) variant might perform better overall, but is more likely to converge to a sub-optimal solution if one exists.

Finally, an additional advantage of decoupled approaches is that you can maintain separate learning rates for the exploration and exploitation Q-tables. The exploration learning rate can be set to a value close to 1 to rapidly explore the environment, while keeping the learning rate for exploitation lower for more stable value learning.

Hyperparameter sensitivity:

With respect to hyperparameter sensitivity, the results indicate that the decoupled approaches are generally more robust to the choice of intrinsic reward weight compared to the integrated Dyna-Q+ method. The integrated variant is highly sensitive to the value of the balancing parameter κ, which must be carefully tuned for each environment. In some cases, such as the Shortcut Maze, no clearly optimal κ value could be identified, highlighting the difficulty of finding a stable balance between exploration and exploitation in this setting. In contrast, the Decoupled Dyna-Q+ (I) model is independent of κ, and thus does not require any tuning of this parameter. The Decoupled Dyna-Q+ (I & E) model, as previously discussed, is sensitive to lower values of κ, but becomes less sensitive at higher values, where its behavior increasingly resembles that of the intrinsic-only variant. Nevertheless, the need to fine-tune κ in the Decoupled (I&E) variant, so that intrinsic and extrinsic rewards are balanced on a comparable scale, remains a limitation relative to the simpler and more stable Decoupled (I) approach.

Overview:

In the context of prior research on decoupled exploration and exploitation [35, 49], discussed in the Related Work section, our findings largely align with existing conclusions, specifically, that decoupled approaches tend to be more robust to hyperparameter scaling and often outperform integrated methods. However, this study also offers new insights, demonstrating that intrinsic-only exploration is not merely a viable alternative but can, in fact, outperform the I+E method in certain settings and provide a safer exploration strategy. The key insights gained from our extended analysis are summarized in the comparison table below.

Table 5: Comparison of Dyna-Q+, Decoupled Dyna-Q+ (I), and Decoupled Dyna-Q+ (I+E)

Criterion	**Dyna-Q+**	**Decoupled (I)**	**Decoupled (I&E)**
Stationary Performance	Weak; unstable	Strong; stable	Strong; fast convergence; stable
Non-Stationary Performance	Poor adaptation	Fast re-learning	Slower but still effective
Suboptimal Trap Robustness	Fails to escape	Avoids traps	Sometimes misled
Exploration Quality	Low	High	Moderate
Sensitivity to κ	High; per-env tuning	None	Moderate; stable when κ is high
Learning Rate Control	Single rate	Separate rates	Separate rates
Training Reward	Moderate	Low	High

7.2 Limitations

This work faces several limitations that should be considered. First of all, some of the environments used are not ideally suited for fair comparisons, particularly the Hallway and Blocking Maze environments. For example, it would be more appropriate to compare the Blocking Maze with the Shortcut Maze, where the newly opened path is shorter than the original one, allowing for a more balanced and meaningful evaluation. Alternatively, the change in the environment could be introduced after all models have converged to the optimal path in order to ensure a fairer test of adaptability. In the case of the Hallway environment, it may be beneficial to increase the model's capacity by using a more expressive architecture, such as a recurrent neural network, or by incorporating more historical information into the state representation.

In addition, the models were not tested in non-deterministic environments. Although such environments may not drastically alter the agent's theoretical behavior, they are still worth exploring to validate the robustness of the approaches. Moreover, only two types of intrinsic motivation methods were included in this study. Expanding the scope to include prediction-based or episodic novelty-based approaches could offer a more comprehensive understanding of how different forms of intrinsic rewards interact with decoupled architectures.

Finally, we did not include any analysis of the additional computational and memory overhead introduced by the decoupled approach, which stores two Q-tables and performs twice as many updates. While this may not be a significant issue in tabular environments, it could become a major drawback in more complex settings.

7.3 Future Work

To address the introduced limitations, this project can be expanded in several directions. Firstly, it would be valuable to test the methods in other sparse tabular environments, including non-deterministic ones, to ensure the robustness of the results and to potentially uncover additional insights. Secondly, it would be worthwhile to implement other intrinsic motivation methods, such as prediction-based approaches, or to explore combinations of different IM signals, such as novelty and recency, to support more robust reward assignment.

Additionally, as noted in the methodology, both Dyna-Q+ and the decoupled approaches might benefit from introducing a small amount of random exploration at the beginning of training. Moreover, at present, the agent is guided solely by Q_{explore}, but alternating between episodes of exploration and exploitation could help narrow the performance gap between the Decoupled (I+E) and Decoupled (I) variants.

Finally, although our exploitation policy adheres to a well-defined MDP, the exploration component violates the MDP assumption, since the intrinsic rewards depend on the agent's history. To mitigate this, we could consider reformulating the exploration problem into an MDP. This approach has been proposed in prior work [37], and it would be interesting to investigate whether applying it improves both performance and exploration behavior in our setting.

8 Conclusion

This thesis explored the decoupling of exploration and exploitation in intrinsic motivation methods. The decoupling was studied in tabular settings by introducing and evaluating two decoupled variants of the Dyna-Q+ algorithm. One model performed exploration using only intrinsic rewards, Decoupled Dyna-Q+ (I), while the other used a combination of intrinsic and extrinsic rewards, Decoupled Dyna-Q+ (I+E). Both were compared to the standard integrated IM baseline.

Through systematic evaluation across a diverse set of tabular environments, we demonstrated that decoupled methods consistently outperform integrated IM approaches in terms of model performance. This improvement is attributed to more stable learning, as the clean exploitation Q-values are free from the non-stationary bias introduced by intrinsic rewards. Furthermore, the decoupled approaches were shown to cover the state-action space more effectively and rapidly, resulting in improved exploratory behavior.

An additional advantage of the decoupled approach, specifically the intrinsic-only variant, is that it is independent of the intrinsic reward weight κ, which otherwise requires careful tuning in integrated IM methods. Comparing the two decoupled models, we find that Decoupled Dyna-Q+ (I+E) generally performs better due to its use of extrinsic signals to guide exploration. However, it is also more sensitive to hyperparameter settings and tends to explore slightly less than the intrinsic-only variant. Consequently, both decoupled models have their strengths and limitations, and their effectiveness depends on the characteristics of the environment. The intrinsic-only variant may perform better in non-stationary and deceptive environments due to its more uniform exploration and faster re-learning of environment dynamics, while the I+E variant can be more effective in stationary environments thanks to faster Q-value propagation.

These findings validate the benefits of separating exploration from exploitation and suggest that decoupling is a promising direction for improving intrinsic motivation methods. Future work could extend this analysis by incorporating exploration schedules based on either Q_{explore} or Q_{exploit}, evaluating additional IM methods, and testing across a wider range of environment types.

References

[1] S. Amin, M. Gomrokchi, H. Satija, H. Van Hoof, and D. Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

[2] C. W. Anderson. *Learning and problem-solving with multilayer connectionist systems (adaptive, strategy learning, neural networks, reinforcement learning)*. University of Massachusetts Amherst, 1986.

[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[4] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[5] J. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

[6] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[7] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[8] O. Caelen and G. Bontempi. Improving the exploration strategy in bandit algorithms. In *International Conference on Learning and Intelligent Optimization*, pages 56–68. Springer, 2007.

[9] N. Chentanez, A. Barto, and S. Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

[10] C. Colas, O. Sigaud, and P.-Y. Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *International conference on machine learning*, pages 1039–1048. PMLR, 2018.

[11] W. Dabney, G. Ostrovski, and A. Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

[12] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[13] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[15] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):8762–8782, 2024.

[16] T. Z. Huan, Y. Sun, and R. Chan. Chapter 8: Intrinsic motivation. https://introrl.readthedocs.io/en/latest/chapter_8.html, 2023. Accessed: 2025-06-10.

[17] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pages 3040–3049. PMLR, 2019.

[18] L. P. Kaelbling. *Learning in embedded systems*. MIT press, 1993.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[20] S. M. Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[21] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.

[22] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, pages 6925–6935. PMLR, 2021.

[25] M. C. Machado, M. G. Bellemare, and M. Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

[26] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[27] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13:103–130, 1993.

[28] S. Narayanan Sasikumar. Exploration in feature space for reinforcement learning. *arXiv e-prints*, pages arXiv–1710, 2017.

[29] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[30] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[31] J. Piaget, M. Cook, et al. *The origins of intelligence in children*, volume 8. International universities press New York, 1952.

[32] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[33] T. Rashid, B. Peng, W. Boehmer, and S. Whiteson. Optimistic exploration even with a pessimistic initialisation. *arXiv preprint arXiv:2002.12174*, 2020.

[34] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[35] L. Schäfer, F. Christianos, J. P. Hanna, and S. V. Albrecht. Decoupled reinforcement learning to stabilise intrinsically-motivated exploration. *arXiv preprint arXiv:2107.08966*, 2021.

[36] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[37] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pages 8583–8592. PMLR, 2020.

[38] M. Sobol Mark, A. Sharma, F. Tajwar, R. Rafailov, S. Levine, and C. Finn. Offline retraining for online rl: Decoupled policy learning to mitigate exploration bias. *arXiv e-prints*, pages arXiv–2310, 2023.

[39] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[40] R. S. Sutton. Integrated modeling and control based on reinforcement learning and dynamic programming. *Advances in neural information processing systems*, 3, 1990.

[41] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[42] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995.

[43] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[44] P. Tadepalli and D. Ok. Model-based average reward reinforcement learning. *Artificial intelligence*, 100(1-2):177–224, 1998.

[45] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

[46] J. Wang, Y. Liu, and B. Li. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6202–6209, 2020.

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

[48] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[49] W. F. Whitney, M. Bloesch, J. T. Springenberg, A. Abdolmaleki, K. Cho, and M. Riedmiller. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

[50] A. Zouitine, D. Bertoin, P. Clavier, M. Geist, and E. Rachelson. Rrls: Robust reinforcement learning suite. *arXiv preprint arXiv:2406.08406*, 2024.

A Numerical results

Tables 6 and 7 present the exact AULC values and hyperparameter sensitivity scores corresponding to the evaluation return curves and final accuracy results discussed in Section 6.1 and 6.3, respectively.

A.1 Agent Performance: AULC scores

Table 6: AULC scores (mean ± 95% CI) across environments. Best results per column are in bold.

	Maze	Blocking Maze	ShortCut	DeepSea	Hallway10-0	Hallway10-10
DynaQ	$7.46^{\pm}2.5$	$18.89^{\pm}3.0$	$125.34^{\pm}2.17$	$0.0^{\pm}0.0$	$0.0^{\pm}0.0$	$0.0^{\pm}0.0$
DynaQ+	$25.55^{\pm}0.95$	$\mathbf{43.21^{\pm}2.11}$	$104.04^{\pm}1.36$	$104.13^{\pm}1.52$	$488.23^{\pm}29.07$	$347.91^{\pm}30.96$
Dec. (I&E)	$\mathbf{28.19^{\pm}0.99}$	$23.4^{\pm}2.71$	$\mathbf{159.98^{\pm}0.68}$	$\mathbf{153.42^{\pm}0.43}$	$562.07^{\pm}7.65$	$387.35^{\pm}22.82$
Dec. (I)	$28.03^{\pm}0.89$	$30.36^{\pm}2.88$	$157.3^{\pm}0.66$	$115.58^{\pm}0.94$	$\mathbf{587.01^{\pm}3.14}$	$\mathbf{480.72^{\pm}7.51}$

A.2 Hyperparameter Sensitivity

Table 7: Final Accuracy and Variance depending on the intrinsic weight κ

Maze Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	0.07	0.00	0.07	0.00	0.07	0.00
0.00010	0.03	0.00	0.07	0.00	0.07	0.00
0.00050	0	0.00	0.07	0.00	0.07	0.00
0.00100	0	0.00	0.07	0.00	0.07	0.00
0.00500	0	0.00	0.07	0.00	0.07	0.00

Blocking Maze Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	0.03	0.32	0.04	0.11	0.04	0.03
0.00010	0.05	0.32	0.04	0.21	0.05	0.05
0.00050	0.01	0.39	0.03	0.23	0.05	0.05
0.00100	0.35	0.04	0.25	0.05	0.16	0.04
0.00500	0	0	0.30	0.04	0.37	0.03

ShortCut Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	0.07	0.01	0.01	0.00	0.06	0.02
0.00010	0.02	0.01	0.01	0.00	0.05	0.02
0.00050	0.01	0.01	0.01	0.00	0.04	0.02
0.00100	0.00	0.00	0.01	0.00	0.03	0.02
0.00500	0.00	0.00	0.01	0.00	0.01	0.00

DeepSea Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	0.0	0.00	0.07	0.00	0.07	0.00
0.00010	0.0	0.00	0.07	0.00	0.07	0.00
0.00050	0.0	0.00	0.07	0.00	0.07	0.00
0.00100	0.0	0.00	0.07	0.00	0.07	0.00
0.00500	0.0	0.00	0.07	0.00	0.07	0.00

Hallway 10-0 Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	3.94	1.62	4.80	0.00	3.43	1.82
0.00010	3.77	1.35	4.80	0.00	3.65	1.79
0.00050	3.37	2.14	4.80	0.00	3.94	1.28
0.00100	3.53	1.87	4.80	0.00	3.15	1.00
0.00500	4.52	0.11	4.80	0.00	4.15	0.15

Hallway 10-10 Environment

κ	DynaQ+		Dec. (I)		Dec. (I+E)	
	Mean	Var	Mean	Var	Mean	Var
0.00001	2.78	2.31	4.80	0.00	3.10	2.40
0.00010	2.66	2.61	4.70	0.27	2.94	2.17
0.00050	2.96	2.33	4.80	0.00	3.34	1.81
0.00100	3.29	1.90	4.78	0.01	2.94	2.18
0.00500	4.18	0.77	4.76	0.02	3.97	1.36

B Additional visualizations

The following figures show the maximum Q_{exploit} values obtained after training for each model and environment. The arrows indicate the actions corresponding to the highest Q-values. Each heatmap is averaged over 100 repetitions.

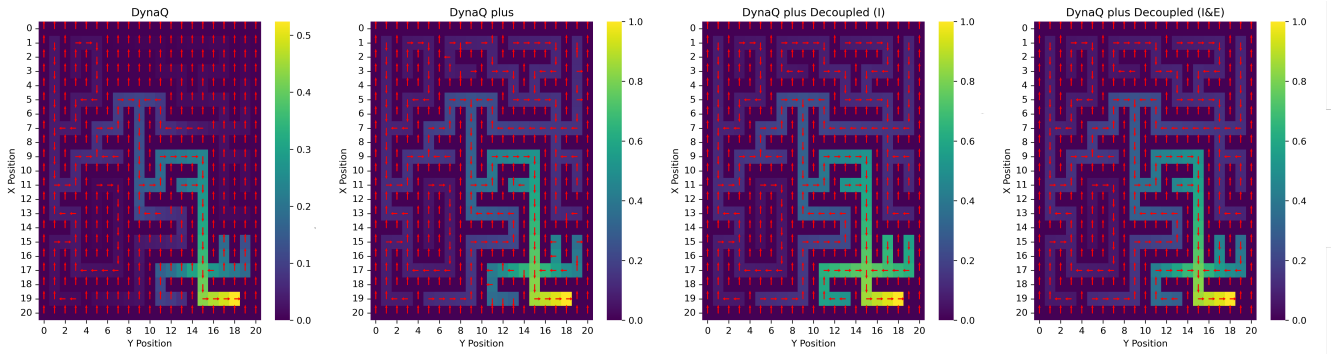


Figure 12: Maze environment

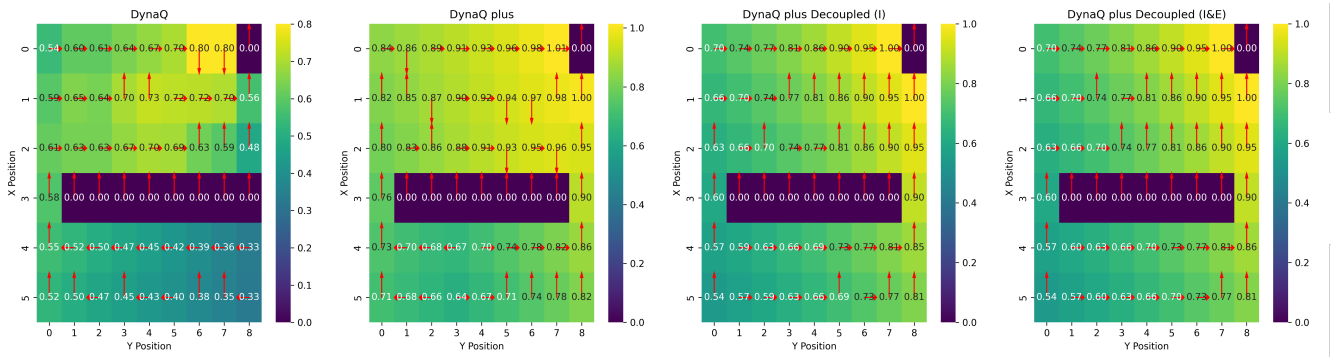


Figure 13: Blocking Maze environment

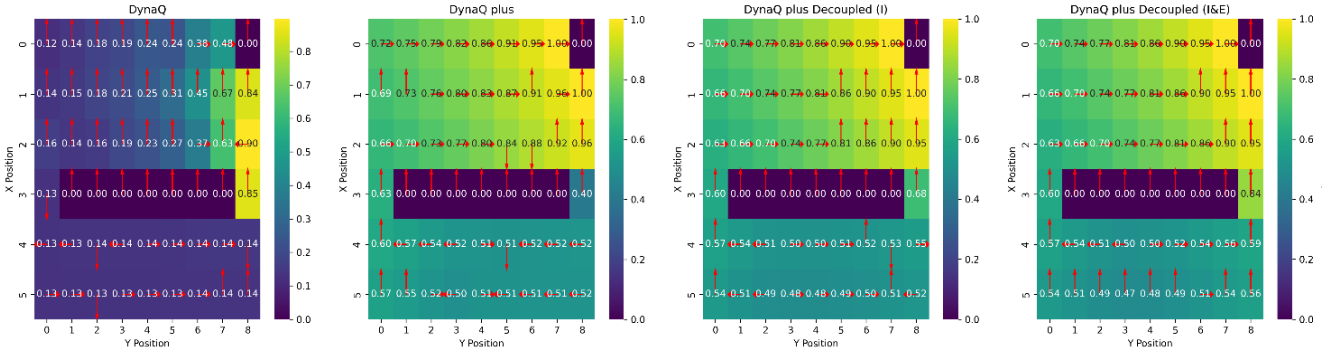


Figure 14: Shortcut Maze environment

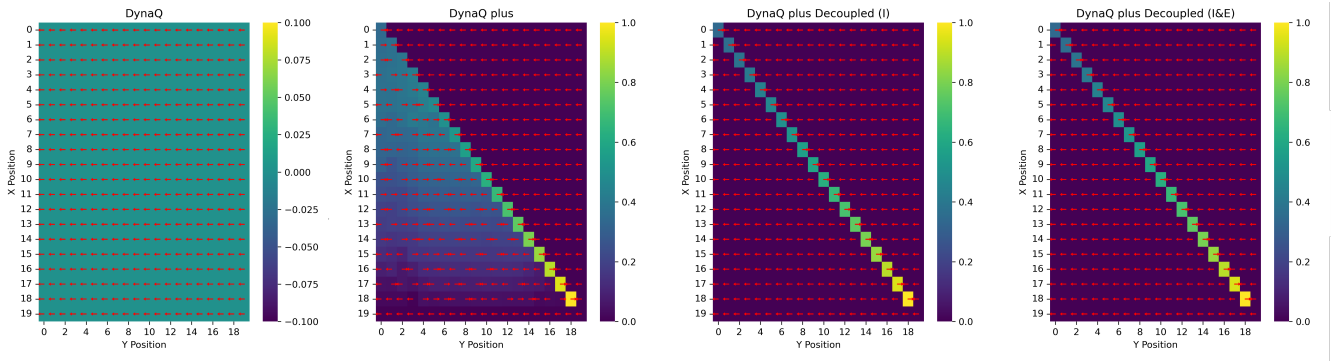


Figure 15: DeepSea 20

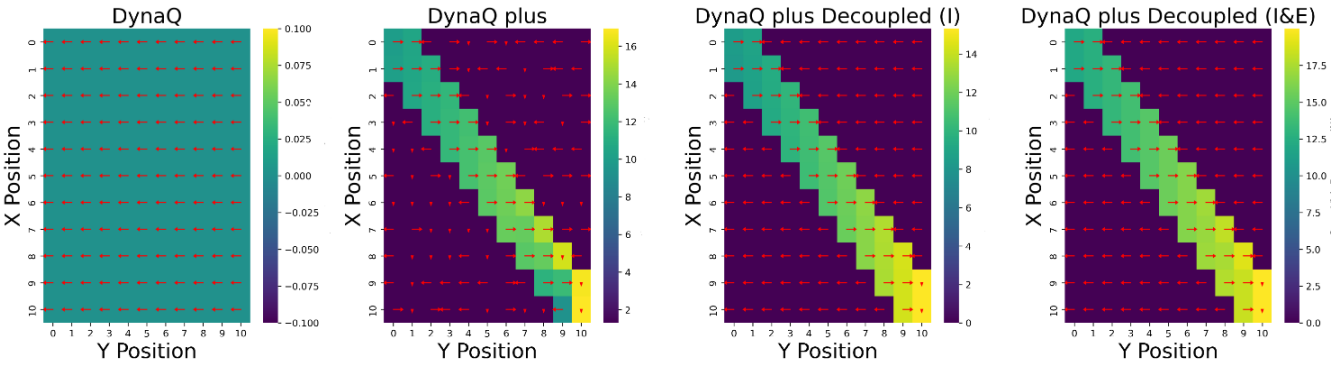


Figure 16: Hallway 10-0

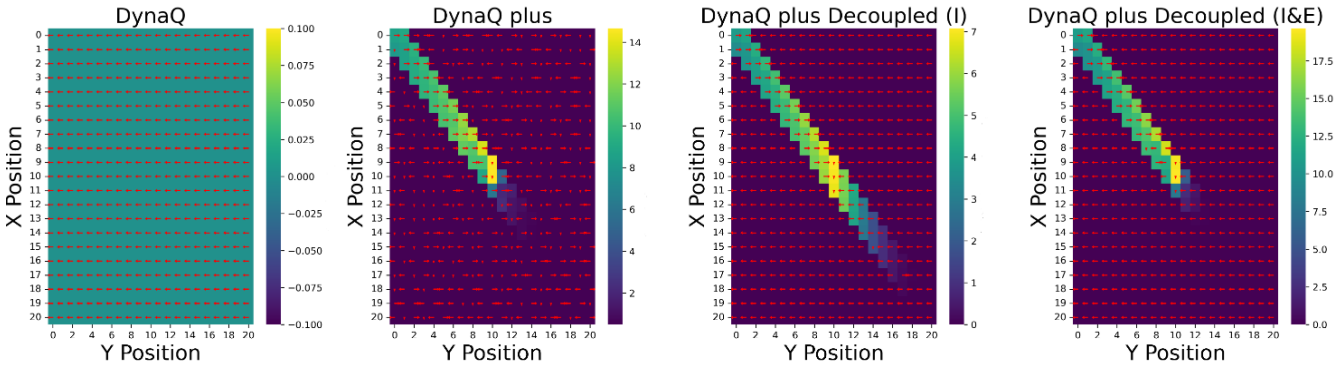
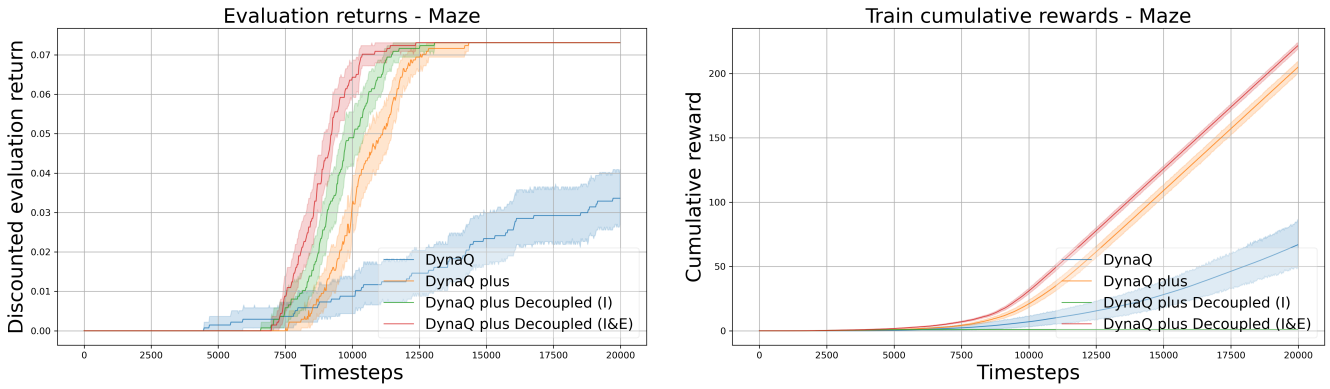


Figure 17: Hallway 10-10

C Additional results: Novelty

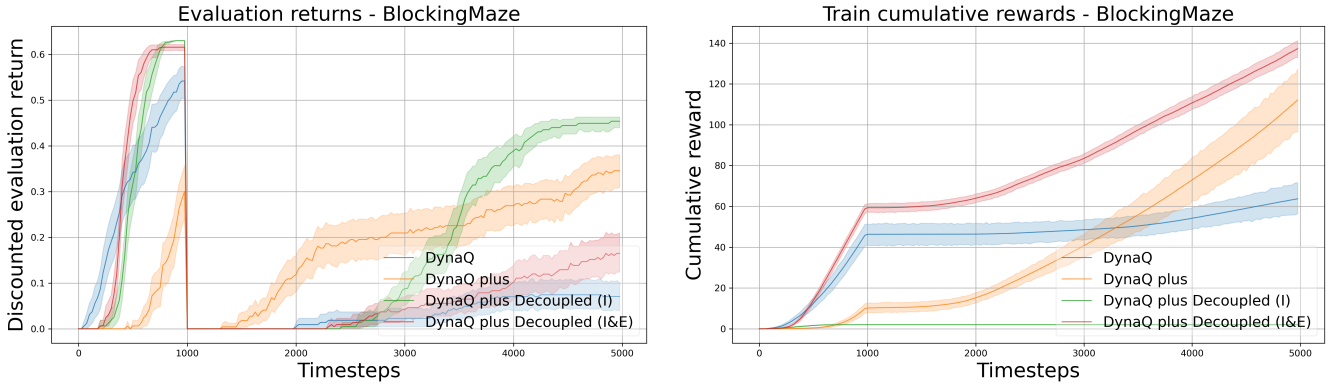
In this appendix, we present additional results obtained using the novelty-based intrinsic motivation method. Although minor differences are observed, the decoupled approaches consistently outperform the integrated methods for this IM type as well. Moreover, the trends identified in the experiments using recency-based bonuses are also evident here, confirming the robustness of our findings across different IM formulations.

C.1 Agent Performance



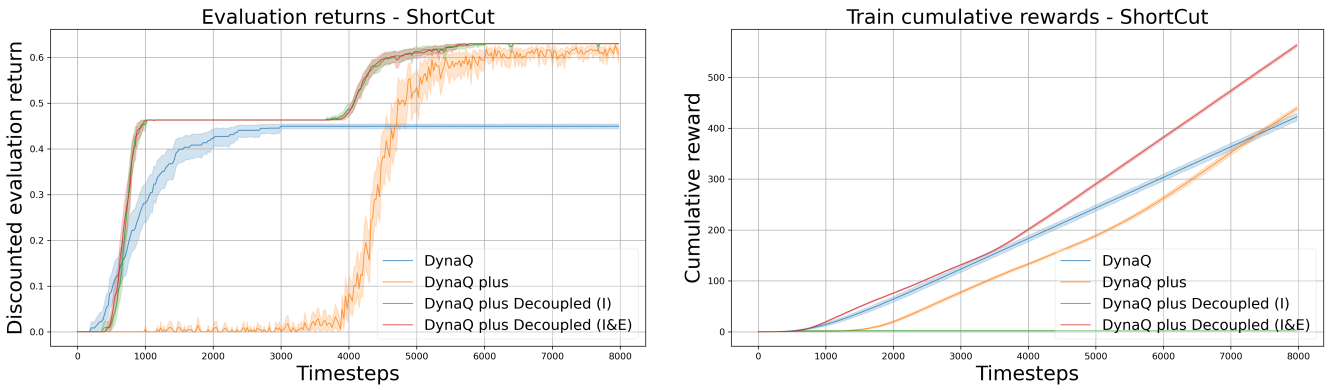
(a) (b)

Figure 18: Agent performance in the Maze environment: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.



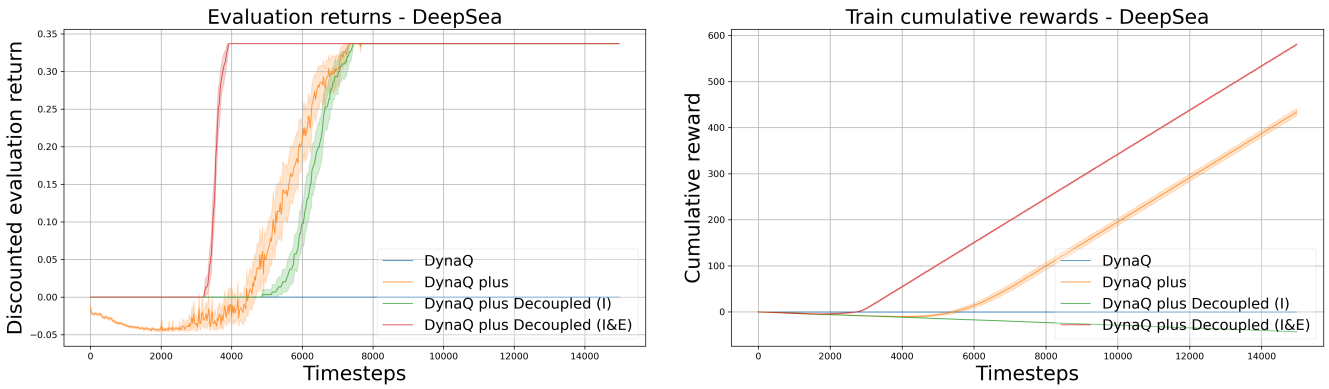
(a) (b)

Figure 19: Agent performance in the Blocking Maze environment: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.



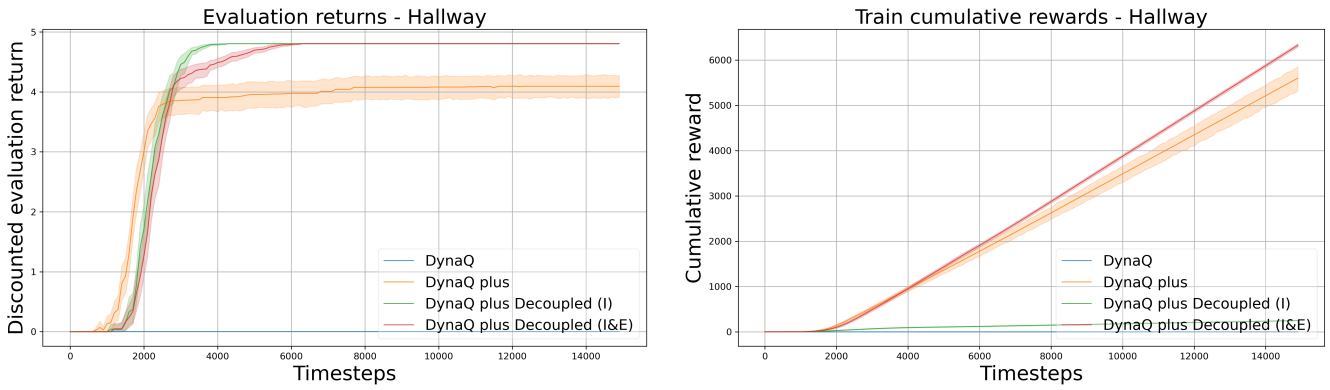
(a) (b)

Figure 20: Agent performance in the Shortcut environment: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.



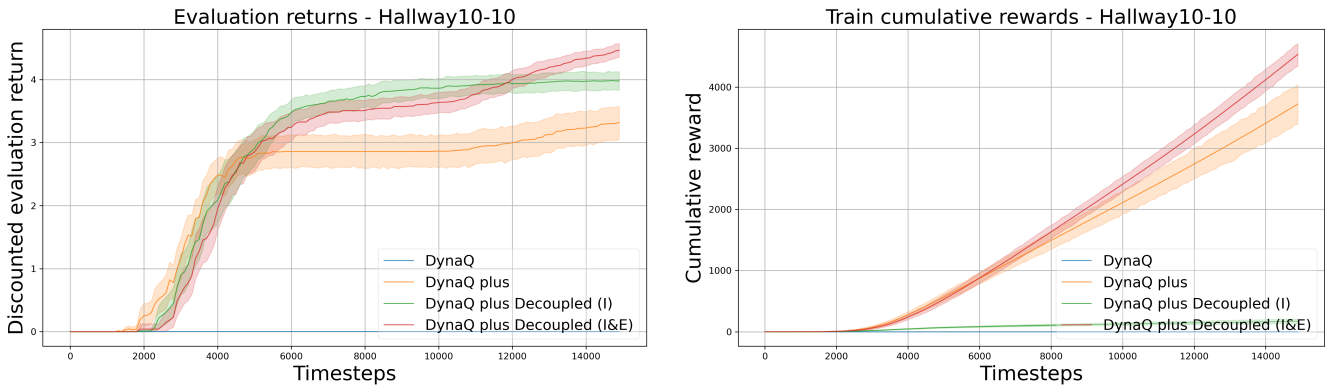
(a) (b)

Figure 21: Agent performance in the DeepSea-20 environment: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.



(a) (b)

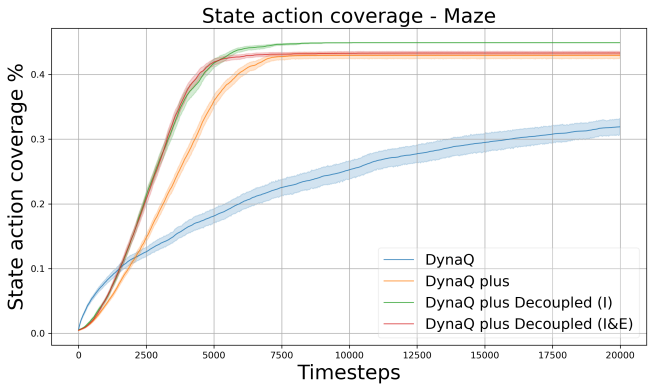
Figure 22: Agent performance in the Hallway 10-0 environment with sparse rewards and $N_l = 10$, $N_r = 0$: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.



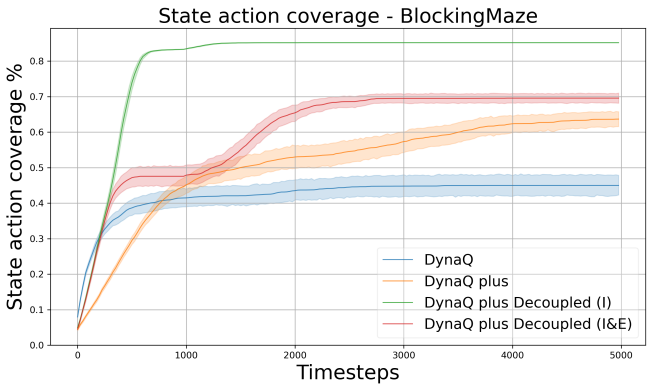
(a) (b)

Figure 23: Agent performance in the Hallway 10-10 environment with sparse rewards and $N_l = 10$, $N_r = 10$: (a) Discounted evaluation return vs. timesteps. (b) Cumulative training reward vs. timesteps. Shaded regions represent the 95% confidence intervals.

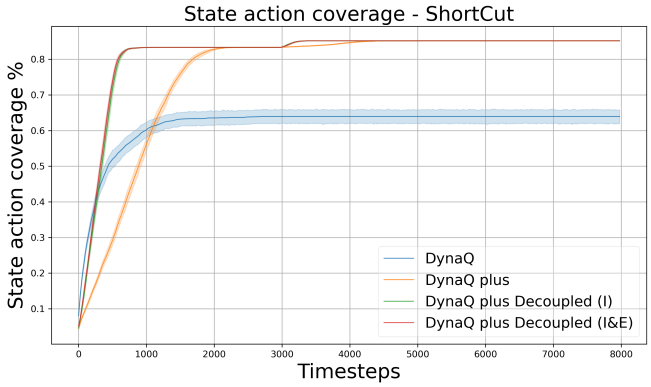
C.2 State-action Space Coverage



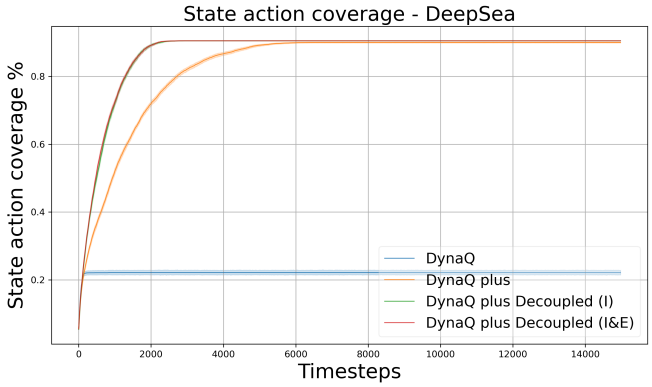
(a) Maze environment



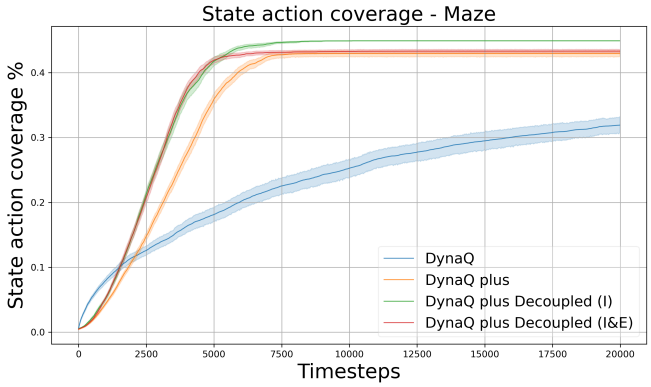
(b) Blocking Maze environment



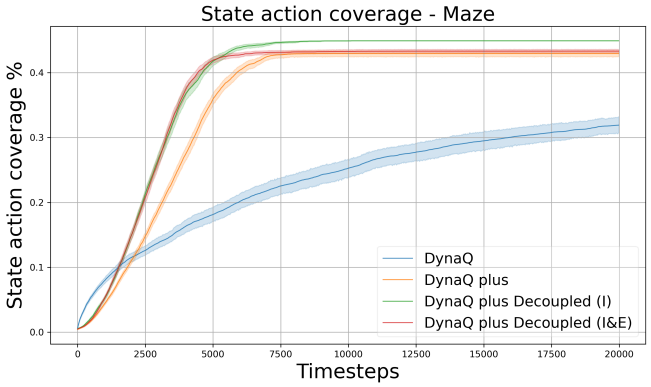
(c) Shortcut Maze environment



(d) DeepSea 20



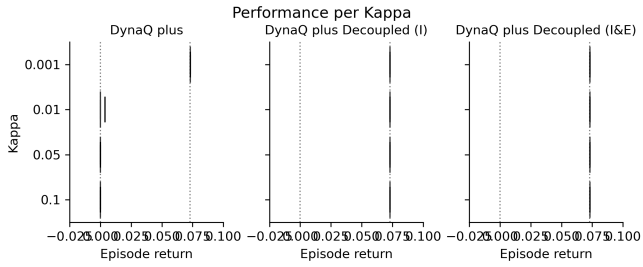
(e) Hallway 10-0



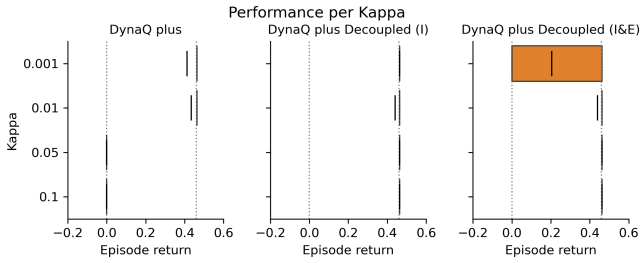
(f) Hallway 10-10

Figure 24: State-action coverage (%) of the models across the tested environments. The x-axis represents the training timesteps, while the y-axis shows the percentage of state-action pairs visited during training.

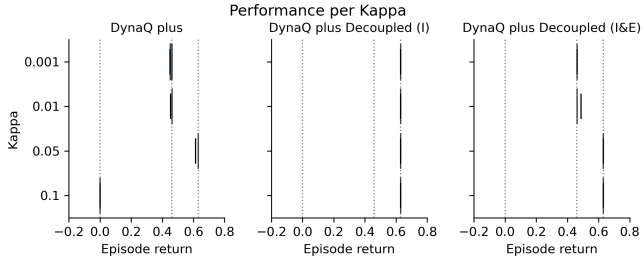
C.3 Hyperparameter Sensitivity



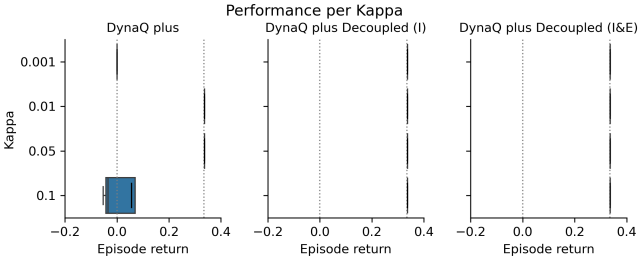
(a) Maze environment



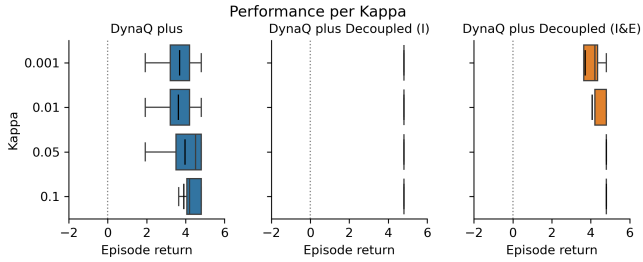
(b) Blocking Maze environment



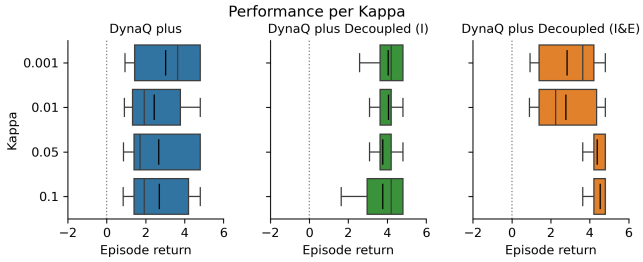
(c) Shortcut Maze environment



(d) DeepSea 20



(e) Hallway 10-0



(f) Hallway 10-10

Figure 25: Average evaluation returns across environments, measured after the training phase with $\kappa \in \{0.001, 0.01, 0.05, 0.1\}$. Shaded regions indicate 95% confidence intervals. A method that is insensitive to hyperparameters will have the final average episodic returns concentrated to the right for all hyperparameter values.